EVENT: Start with the library "c-predefined4".

THEOREM: fetch-label-0-case-2
cdr (assoc (*cc*, cons ('(routineerror . 0), make-label-alist (*lst*, 0))))
= 0

EVENT: Disable fetch-label-0-case-2.

THEOREM: mg-cond-to-p-nat-p-objectp-type-nat
((length (*cond-list*) < (((exp (2, MG-WORD-SIZE) − 1) − 1) − 1))
∧ (p-word-size (*state*) = MG-WORD-SIZE))
→ p-objectp-type ('nat, mg-cond-to-p-nat (*cc*, *cond-list*), *state*)

EVENT: Disable mg-cond-to-p-nat-p-objectp-type-nat.

THEOREM: mg-cond-to-p-nat-index-lessp
(length (*lst*) < *n*)
→ ((untag (mg-cond-to-p-nat (*c*, *lst*)) < (1 + (1 + *n*))) = **t**)

EVENT: Disable mg-cond-to-p-nat-index-lessp.

THEOREM: set-alist-value-map-down-values-length-doesnt-shrink
(mg-vars-list-ok-in-p-state (*x*, *bindings*, *temp-stk*) ∧ mg-alistp (*x*))
→ ((length (map-down-values (set-alist-value (*name*, *value*, *x*),
bindings,
temp-stk))
< length (map-down-values (*x*, *bindings*, *temp-stk*)))
= **f**)

EVENT: Disable set-alist-value-map-down-values-length-doesnt-shrink.

THEOREM: extra-bindings-doesnt-affect-formal-types-preserved
(car (*x*) ∉ listcars (*y*))
→ (formal-types-preserved (*y*, cons (*x*, *z*)) = formal-types-preserved (*y*, *z*))

THEOREM: not-simple-identifiers-array-identifiers
(definedp (*x*, *alist*)
∧ mg-alistp (*alist*)
∧ (¬ simple-identifierp (*x*, *alist*)))
→ array-identifierp (*x*, *alist*)

THEOREM: mg-meaning-preserves-signatures-match2
plistp (*alist*)

→ signatures-match (*alist*,
mg-alist (mg-meaning (*stmt*,
proc-list,
mg-state (*cc*, *alist*, *psw*),
n)))

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                              ;;
;;                              PROCEDURE CALLS                                 ;;
;;                                                                              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

EVENT: Disable make-cond-list.

EVENT: Disable make-call-var-alist.

EVENT: Disable mg-var-ok-in-p-state.

EVENT: Enable proc-call-code.

THEOREM: proc-call-meaning-r-2
(car (*stmt*) = `'proc-call-mg`)
→ (mg-meaning-r (*stmt*, *proc-list*, *mg-state*, *n*, *sizes*)
= **if** *n* ≃ `0` **then** signal-system-error (*mg-state*, `'timed-out`)
elseif ¬ normal (*mg-state*) **then** *mg-state*
elseif resources-inadequatep (*stmt*, *proc-list*, *sizes*)
then signal-system-error (*mg-state*, `'resource-error`)
else map-call-effects (mg-meaning-r (def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list,
make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*,
proc-list)),
n − 1,
list (t-size (*sizes*)
+ data-length (def-locals (fetch-called-def (*stmt*,
proc-list))),
c-size (*sizes*)
+ (`2`
+ length (def-locals (fetch-called-def (*stmt*,

proc-list)))
+ length (def-formals (fetch-called-def (*stmt*,
proc-list))))
fetch-called-def (*stmt*, *proc-list*),
stmt,
mg-state) **endif**)

THEOREM: call-translation-2
(car (*stmt*) = `'proc-call-mg`)
→ (translate (*cinfo*, *cond-list*, *stmt*, *proc-list*)
= make-cinfo (append (code (*cinfo*),
proc-call-code (*cinfo*,
stmt,
cond-list,
def-locals (fetch-called-def (*stmt*,
proc-list)),
length (def-cond-locals (fetch-called-def (*stmt*,
proc-list))))),
label-alist (*cinfo*),
label-cnt (*cinfo*)
+ (1 + (1 + length (call-conds (*stmt*))))))

THEOREM: locals-pointers-bigger0
all-cars-unique (*locals*)
→ all-pointers-bigger (collect-pointers (map-call-locals (*locals*, *n*), *locals*),
n)

THEOREM: no-p-aliasing-locals
((*n* ∈ **N**) ∧ all-cars-unique (*locals*))
→ no-p-aliasing (map-call-locals (*locals*, *n*), *locals*)

THEOREM: map-call-formals-all-pointers-smaller3
(ok-actual-params-list (*actuals*, *mg-vars*)
∧ data-param-lists-match (*actuals*, *formals*, *mg-vars*)
∧ all-cars-unique (*formals*)
∧ mg-alistp (*mg-vars*)
∧ mg-vars-list-ok-in-p-state (*mg-vars*, *bindings*, *temp-stk*))
→ all-pointers-smaller (collect-pointers (map-call-formals (*formals*,
actuals,
bindings),
make-call-param-alist (*formals*,
actuals,
mg-vars)),
length (*temp-stk*))

```
;; These got lost somewhere
```

THEOREM: array-alist-element-lengths-match
(mg-alistp (*mg-vars*)
 ∧ definedp (*x*, *mg-vars*)
 ∧ (¬ simple-mg-type-refp (cadr (assoc (*x*, *mg-vars*)))))
→ (length (caddr (assoc (*x*, *mg-vars*)))
  = array-length (cadr (assoc (*x*, *mg-vars*))))

EVENT: Disable array-alist-element-lengths-match.

EVENT: Enable no-p-aliasing.

THEOREM: extra-bindings-dont-affect-no-p-aliasing
no-duplicates (listcars (append (*bindings1*, *lst*)))
→ (no-p-aliasing (append (*bindings1*, *bindings2*), *lst*)
  = no-p-aliasing (*bindings2*, *lst*))

THEOREM: extra-bindings-dont-affect-no-p-aliasing2
no-duplicates (listcars (append (*bindings2*, *lst*)))
→ (no-p-aliasing (append (*bindings1*, *bindings2*), *lst*)
  = no-p-aliasing (*bindings1*, *lst*))

THEOREM: actual-pointers-distinct
(ok-actual-params-list (*actuals*, *mg-vars*)
 ∧ defined-identifierp (*x*, *mg-vars*)
 ∧ data-param-lists-match (*actuals*, *formals*, *mg-vars*)
 ∧ ok-mg-formal-data-params-plistp (*formals*)
 ∧ all-cars-unique (*mg-vars*)
 ∧ no-duplicates (cons (*x*, *actuals*))
 ∧ no-p-aliasing (*bindings*, *mg-vars*)
 ∧ all-cars-unique (*formals*)
 ∧ mg-alistp (*mg-vars*))
→ (untag (cdr (assoc (*x*, *bindings*)))
  ∉ collect-pointers (map-call-formals (*formals*, *actuals*, *bindings*),
  make-call-param-alist (*formals*,
  actuals,
  mg-vars)))

THEOREM: actual-pointers-distinct2
(ok-actual-params-list (*actuals*, *mg-vars*)
 ∧ defined-identifierp (*x*, *mg-vars*)
 ∧ data-param-lists-match (*actuals*, *formals*, *mg-vars*)

$\wedge$ ok-mg-formal-data-params-plistp (*formals*)
$\wedge$ mg-vars-list-ok-in-p-state (*mg-vars*, *bindings*, *temp-stk*)
$\wedge$ all-cars-unique (*mg-vars*)
$\wedge$ no-duplicates (cons (*x*, *actuals*))
$\wedge$ no-p-aliasing (*bindings*, *mg-vars*)
$\wedge$ all-cars-unique (*formals*)
$\wedge$ mg-alistp (*mg-vars*)
$\wedge$ ($\neg$ simple-mg-type-refp (cadr (assoc (*x*, *mg-vars*)))))
$\rightarrow$ disjoint (n-successive-pointers (cdr (assoc (*x*, *bindings*)),
array-length (cadr (assoc (*x*, *mg-vars*)))),
collect-pointers (map-call-formals (*formals*, *actuals*, *bindings*),
make-call-param-alist (*formals*,
actuals,
mg-vars)))

THEOREM: no-p-aliasing-formals
(ok-actual-params-list (*actuals*, *mg-vars*)
$\wedge$ data-param-lists-match (*actuals*, *formals*, *mg-vars*)
$\wedge$ ok-mg-formal-data-params-plistp (*formals*)
$\wedge$ mg-vars-list-ok-in-p-state (*mg-vars*, *bindings*, *temp-stk*)
$\wedge$ all-cars-unique (*mg-vars*)
$\wedge$ no-duplicates (*actuals*)
$\wedge$ no-p-aliasing (*bindings*, *mg-vars*)
$\wedge$ all-cars-unique (*formals*)
$\wedge$ mg-alistp (*mg-vars*))
$\rightarrow$ no-p-aliasing (map-call-formals (*formals*, *actuals*, *bindings*),
make-call-param-alist (*formals*, *actuals*, *mg-vars*))

THEOREM: actual-params-list-ok-in-mg-alist
((car (*stmt*) = `'proc-call-mg`)
$\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*))
$\rightarrow$ ok-actual-params-list (call-actuals (*stmt*), mg-alist (*mg-state*))

THEOREM: data-param-lists-match-in-mg-alist
((car (*stmt*) = `'proc-call-mg`)
$\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*))
$\rightarrow$ data-param-lists-match (call-actuals (*stmt*),
def-formals (fetch-called-def (*stmt*, *proc-list*)),
mg-alist (*mg-state*))

THEOREM: call-local-names-unique

((car (*stmt*) = '`proc-call-mg`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*))
→ no-duplicates (append (listcars (def-locals (fetch-called-def (*stmt*, *proc-list*))),
listcars (def-formals (fetch-called-def (*stmt*, *proc-list*)))))

THEOREM: no-p-aliasing-in-call-environment
((car (*stmt*) = '`proc-call-mg`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ user-defined-procp (*subr*, *proc-list*)
∧ plistp (*temp-stk*)
∧ listp (*ctrl-stk*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ all-cars-unique (mg-alist (*mg-state*)))
→ no-p-aliasing (make-frame-alist (fetch-called-def (*stmt*, *proc-list*),
stmt,
ctrl-stk,
temp-stk),
make-call-var-alist (mg-alist (*mg-state*),
stmt,
fetch-called-def (*stmt*, *proc-list*)))

THEOREM: call-exact-time-translation-parameters-ok
((car (*stmt*) = '`proc-call-mg`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*))
→ ok-translation-parameters (make-cinfo (**nil**,
cons ('`(routineerror . 0)`,
make-label-alist (make-cond-list (fetch-called-def (*stmt*, *proc-list*)),
`0`)),
`1`),
make-cond-list (fetch-called-def (*stmt*, *proc-list*)),

def-body (fetch-called-def (*stmt*, *proc-list*)),
proc-list,
cons (`'(dl 0 nil (no-op))`,
cons (list (`'pop*`,
data-length (def-locals (fetch-called-def (*stmt*,
proc-list)))),
`'((ret))`)))

THEOREM: call-body-rewrite
code (translate-def-body (assoc (call-name (*stmt*), *proc-list*), *proc-list*))
= append (code (translate (make-cinfo (**nil**,
cons (`'(routineerror . 0)`,
make-label-alist (make-cond-list (fetch-called-def (*stmt*,
proc-list)),
`0`)),
`1`),
make-cond-list (fetch-called-def (*stmt*, *proc-list*)),
def-body (fetch-called-def (*stmt*, *proc-list*)),
proc-list)),
cons (`'(dl 0 nil (no-op))`,
cons (list (`'pop*`,
data-length (def-locals (fetch-called-def (*stmt*,
proc-list)))),
`'((ret))`))))

DEFINITION:
mg-locals-list-ok-induction-hint (*locals*, *temp-stk*)
= **if** *locals* ≃ **nil** **then** **t**
elseif simple-mg-type-refp (cadar (*locals*))
then mg-locals-list-ok-induction-hint (cdr (*locals*),
cons (mg-to-p-simple-literal (caddar (*locals*)),
temp-stk))
else mg-locals-list-ok-induction-hint (cdr (*locals*),
append (reverse (mg-to-p-simple-literal-list (caddar (*locals*))),
temp-stk)) **endif**

THEOREM: definedp-caar
definedp (*x*, cons (cons (*x*, *y*), *z*))

THEOREM: tag-length-plistp-2
length-plistp (tag (*x*, *y*), `2`)

THEOREM: mg-locals-list-ok-in-call-environment
(all-cars-unique (*locals*) ∧ ok-mg-local-data-plistp (*locals*))
→ mg-vars-list-ok-in-p-state (*locals*,

append (map-call-locals (*locals*,
length (*temp-stk*)),
lst),
append (reverse (mg-to-p-local-values (*locals*)),
temp-stk))

THEOREM: simple-formal-ok-for-actual2
(definedp (*actual*, *mg-alist*)
∧ data-params-match (*actual*, *formal*, *mg-alist*)
∧ mg-vars-list-ok-in-p-state (*mg-alist*, *bindings*, *temp-stk*)
∧ simple-mg-type-refp (cadr (*formal*)))
→ ok-temp-stk-index (cdr (assoc (*actual*, *bindings*)), *temp-stk*)

DEFINITION:
mg-formals-list-ok-induction-hint (*formals*, *actuals*)
= **if** *formals* ≃ **nil then t**
else mg-formals-list-ok-induction-hint (cdr (*formals*),
cdr (*actuals*)) **endif**

THEOREM: array-formal-ok-for-actual
(defined-identifierp (*actual*, *mg-alist*)
∧ data-params-match (*actual*, *formal*, *mg-alist*)
∧ mg-vars-list-ok-in-p-state (*mg-alist*, *bindings*, *temp-stk*)
∧ (¬ simple-mg-type-refp (cadr (*formal*))))
→ ok-temp-stk-array-index (cdr (assoc (*actual*, *bindings*)),
append (*local-values*, *temp-stk*),
array-length (cadr (*formal*)))

THEOREM: mg-formals-list-ok-in-call-environment0
(all-cars-unique (*formals*)
∧ ok-actual-params-list (*actuals*, *mg-alist*)
∧ data-param-lists-match (*actuals*, *formals*, *mg-alist*)
∧ mg-vars-list-ok-in-p-state (*mg-alist*, *bindings*, *temp-stk*))
→ mg-vars-list-ok-in-p-state (make-call-param-alist (*formals*,
actuals,
mg-alist),
map-call-formals (*formals*, *actuals*, *bindings*),
append (*local-values*, *temp-stk*))

THEOREM: mg-formals-list-ok-in-call-environment1
(all-cars-unique (*formals*)
∧ mg-alistp (*mg-alist*)
∧ ok-actual-params-list (*actuals*, *mg-alist*)
∧ ok-mg-formal-data-params-plistp (*formals*)
∧ data-param-lists-match (*actuals*, *formals*, *mg-alist*)

$\wedge$ mg-vars-list-ok-in-p-state (*mg-alist*, *bindings*, *temp-stk*))
$\rightarrow$ mg-vars-list-ok-in-p-state (make-call-param-alist (*formals*,
actuals,
mg-alist),
map-call-formals (*formals*, *actuals*, *bindings*),
append (*local-values*,
map-down-values (*mg-alist*,
bindings,
temp-stk)))

THEOREM: mg-vars-list-ok-in-call-environment
((*n* $\not\simeq$ `0`)
$\wedge$ ($\neg$ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
$\wedge$ (car (*stmt*) = `'proc-call-mg`)
$\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ ok-mg-def-plistp (*proc-list*)
$\wedge$ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
$\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
$\wedge$ cond-subsetp (*r-cond-list*, *t-cond-list*)
$\wedge$ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
$\wedge$ user-defined-procp (*subr*, *proc-list*)
$\wedge$ plistp (*temp-stk*)
$\wedge$ listp (*ctrl-stk*)
$\wedge$ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
$\wedge$ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*)
$\wedge$ normal (*mg-state*)
$\wedge$ all-cars-unique (mg-alist (*mg-state*)))
$\rightarrow$ mg-vars-list-ok-in-p-state (make-call-var-alist (mg-alist (*mg-state*),
stmt,
fetch-called-def (*stmt*,
proc-list)),
append (map-call-locals (def-locals (fetch-called-def (*stmt*,
proc-list)),
length (*temp-stk*)),
map-call-formals (def-formals (fetch-called-def (*stmt*,
proc-list)),

call-actuals (*stmt*),
bindings (top (*ctrl-stk*)))),
append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (*stmt*,
proc-list)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)))

THEOREM: proc-call-doesnt-halt
((car (*stmt*) = `'proc-call-mg`)
∧ normal (*mg-state*)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (¬ resource-errorp (mg-meaning-r (def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list,
make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*,
proc-list)),
n − 1,
list (length (append (reverse (mg-to-p-local-values (def-locals (fetch-call

map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
p-ctrl-stk-size (cons (p-frame (make-frame-alist (fetch-called-def (*s*
p
stmt,
ctrl-stk,
temp-stk),
tag (`'pc`,
cons (*subr*,
1 + (length (code (*cinfo*))

+ data-length (def-

+ length (def-locals

+ length (call-actu

ctrl-stk))))))

THEOREM: proc-call-doesnt-halt2
((car (*stmt*) = 'proc-call-mg)
∧ normal (*mg-state*)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (¬ resource-errorp (mg-meaning-r (def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list,
make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*,
proc-list)),
n − 1,
list (length (*temp-stk*)
+ data-length (def-locals (fetch-called-def (*stmt*,
proc-list))),
p-ctrl-stk-size (*ctrl-stk*)
+ 2
+ length (def-locals (fetch-called-def (*stmt*,
proc-list)))
+ length (def-formals (fetch-called-def (*stmt*,
proc-list)))))))

THEOREM: proc-call-exact-time-hyps
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,

list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'proc-call-mg`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ cond-subsetp (*r-cond-list*, *t-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ plistp (*temp-stk*)
∧ listp (*ctrl-stk*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ normal (*mg-state*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (ok-mg-statement (def-body (fetch-called-def (*stmt*, *proc-list*)),
make-cond-list (fetch-called-def (*stmt*, *proc-list*)),
make-name-alist (fetch-called-def (*stmt*, *proc-list*)),
proc-list)
∧ ok-translation-parameters (make-cinfo (**nil**,
cons (`'(routineerror . 0)`,
make-label-alist (make-cond-list (fetch-called-def (*stmt*,
proc-
`0`)),
`1`),
make-cond-list (fetch-called-def (*stmt*,
proc-list)),
def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list,
cons (`'(dl 0 nil (no-op))`,

cons (list (’pop*,
data-length (def-locals (fetch-called-def (*stmt*,
proc-list)))),
’((ret)))))
∧ ok-mg-statep (make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*,
proc-list)),
make-cond-list (fetch-called-def (*stmt*, *proc-list*)))
∧ cond-subsetp (make-cond-list (fetch-called-def (*stmt*, *proc-list*)),
make-cond-list (fetch-called-def (*stmt*, *proc-list*)))
∧ (code (translate-def-body (assoc (call-name (*stmt*), *proc-list*),
proc-list))
= append (code (translate (make-cinfo (**nil**,
cons (’(routineerror
. 0),
make-label-alist (make-cond-list (fetch-called-def (*stm*
pro
0)),
1),
make-cond-list (fetch-called-def (*stmt*,
proc-list)),
def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list)),
cons (’(dl 0 nil (no-op)),
cons (list (’pop*,
data-length (def-locals (fetch-called-def (*stmt*,
proc-list)))),
’((ret)))))))
∧ user-defined-procp (call-name (*stmt*), *proc-list*)
∧ plistp (append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (*stmt*,
proc-list)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)))
∧ listp (cons (p-frame (make-frame-alist (fetch-called-def (*stmt*,
proc-list),
stmt,
ctrl-stk,
temp-stk),
tag (’pc,
cons (*subr*,
1 + (length (code (*cinfo*))

```
                                                    +  data-length (def-locals (fetch-called-def (stmt,
                                                                                                  proc-list)))
                                                    +  length (def-locals (fetch-called-def (stmt,
                                                                                             proc-list)))
                                                    +  length (call-actuals (stmt))))))),
                                        ctrl-stk))
∧  mg-vars-list-ok-in-p-state (mg-alist (make-call-environment (mg-state,
                                                                stmt,
                                                                fetch-called-def (stmt,
                                                                                  proc-list))),
                                        bindings (top (cons (p-frame (make-frame-alist (fetch-called-def (stmt,
                                                                                                          proc-list),
                                                                                        stmt,
                                                                                        ctrl-stk,
                                                                                        temp-stk),
                                                                      tag ('pc,
                                                                           cons (subr,
                                                                                 1 + (length (code (cinfo))
                                                                                      +  data-length (def-locals (fe

                                                                                      +  length (def-locals (fetch-c

                                                                                      +  length (call-actuals (stmt
                                                                                ctrl-stk))),
                                                              append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (stmt,
                                                                                                                                   proc-list
                                                                                 map-down-values (mg-alist (mg-state),
                                                                                                  bindings (top (ctrl-stk)),
                                                                                                  temp-stk)))
∧  no-p-aliasing (bindings (top (cons (p-frame (make-frame-alist (fetch-called-def (stmt,
                                                                                    proc-list),
                                                                  stmt,
                                                                  ctrl-stk,
                                                                  temp-stk),
                                                tag ('pc,
                                                     cons (subr,
                                                           1 + (length (code (cinfo))
                                                                +  data-length (def-locals (fetch-called-d

                                                                +  length (def-locals (fetch-called-def (st
                                                                                                         pr
                                                                +  length (call-actuals (stmt))))))),
                                                ctrl-stk))),
                                    mg-alist (make-call-environment (mg-state,
```

stmt,
fetch-called-def (*stmt*,
proc-list))))
∧ signatures-match (mg-alist (make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*,
proc-list))),
make-name-alist (fetch-called-def (*stmt*,
proc-list)))
∧ normal (make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*, *proc-list*)))
∧ all-cars-unique (mg-alist (make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*,
proc-list))))
∧ (¬ resource-errorp (mg-meaning-r (def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list,
make-call-environment (*mg-state*,
stmt,
fetch-called-def (*stmt*,
proc-list)),
$n - 1$,
list (length (append (reverse (mg-to-p-local-values (def-locals (fetch

map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
p-ctrl-stk-size (cons (p-frame (make-frame-alist (fetch-called-d

stmt,
ctrl-stk,
temp-stk),
tag ('pc,
cons (*subr*,
1 + (length (code (*ci
+ data-length

+ length (def-

+ length (call
ctrl-stk)))))))

THEOREM: simple-typed-literal-listp
simple-typed-literalp (*exp*, *type*) → listp (mg-to-p-simple-literal (*exp*))

DEFINITION:
push-array-value-induction-hint (*array-value*, *code*, *temp-stk*)
= **if** *array-value* ≃ **nil then t**
else push-array-value-induction-hint (cdr (*array-value*),
append (*code*,
list (list (`'push-constant`,
mg-to-p-simple-literal (car (*array-value*))))),
push (mg-to-p-simple-literal (car (*array-value*)),
temp-stk)) **endif**

THEOREM: push-local-array-values-code-effect
(((length (*temp-stk*) + length (*array-value*)) < MG-MAX-TEMP-STK-SIZE)
∧ simple-mg-type-refp (*array-elemtype*)
∧ simple-typed-literal-plistp (*array-value*, *array-elemtype*)
∧ (cdddr (assoc (*subr*, *proc-list-code*))
= append (*code*,
append (push-local-array-values-code (*array-value*),
code2))))
→ (p (p-state (tag (`'pc`, cons (*subr*, length (*code*))),
ctrl-stk,
temp-stk,
proc-list-code,
data-segment,
max-ctrl,
MG-MAX-TEMP-STK-SIZE,
word-size,
`'run`),
length (*array-value*))
= p-state (tag (`'pc`,
cons (*subr*, length (*code*) + length (*array-value*))),
ctrl-stk,
append (reverse (mg-to-p-simple-literal-list (*array-value*)),
temp-stk),
proc-list-code,
data-segment,
max-ctrl,
MG-MAX-TEMP-STK-SIZE,
word-size,
`'run`))

EVENT: Enable length-cons.

DEFINITION:
locals-values-induction-hint (*locals*, *code*, *temp-stk*)
= **if** *locals* ≃ **nil then t**
elseif simple-mg-type-refp (cadr (car (*locals*)))
then locals-values-induction-hint (cdr (*locals*),
append (*code*,
list (list ('`push-constant`,
mg-to-p-simple-literal (caddar (*locals*)))),
push (mg-to-p-simple-literal (caddr (car (*locals*))),
temp-stk))
else locals-values-induction-hint (cdr (*locals*),
append (*code*,
push-local-array-values-code (caddr (car (*locals*)))),
append (reverse (mg-to-p-simple-literal-list (caddr (car (*locals*)))),
temp-stk)) **endif**

THEOREM: call-push-locals-values-effect
(((length (*temp-stk*) + data-length (*locals*)) < MG-MAX-TEMP-STK-SIZE)
∧ ok-mg-local-data-plistp (*locals*)
∧ (cdddr (assoc (*subr*, *proc-list-code*))
= append (*code*, append (push-locals-values-code (*locals*), *code2*))))
→ (p (p-state (tag ('`pc`, cons (*subr*, length (*code*))),
ctrl-stk,
temp-stk,
proc-list-code,
data-segment,
max-ctrl,
MG-MAX-TEMP-STK-SIZE,
word-size,
'`run`),
data-length (*locals*))
= p-state (tag ('`pc`,
cons (*subr*, data-length (*locals*) + length (*code*))),
ctrl-stk,
append (reverse (mg-to-p-local-values (*locals*)), *temp-stk*),
proc-list-code,
data-segment,
max-ctrl,
MG-MAX-TEMP-STK-SIZE,
word-size,
'`run`))

DEFINITION:
locals-addresses-induction-hint (*locals*, *code*, *temp-stk*, *n*)

```
=  if locals ≃ nil then t
   elseif simple-mg-type-refp (cadr (car (locals)))
   then locals-addresses-induction-hint (cdr (locals),
                                         append (code,
                                                 list (list ('push-temp-stk-index,
                                                             n))),
                                         push (tag ('nat,
                                                    (length (temp-stk) − n) − 1),
                                               temp-stk),
                                         n)
   else locals-addresses-induction-hint (cdr (locals),
                                         append (code,
                                                 list (list ('push-temp-stk-index,
                                                             n))),
                                         push (tag ('nat,
                                                    (length (temp-stk)
                                                     − n)− 1),
                                               temp-stk),
                                         1 + (n − array-length (cadr (car (locals))))) endif
```

```
THEOREM: call-push-locals-addresses-effect
(((length (temp-stk) + length (locals)) < MG-MAX-TEMP-STK-SIZE)
 ∧  ok-mg-local-data-plistp (locals)
 ∧  (cdddr (assoc (subr, proc-list-code))
      =  append (code,
                 append (push-locals-addresses-code (locals, n), code2)))
 ∧  (n < length (temp-stk))
 ∧  (n ≮ (data-length (locals) − 1)))
→  (p (p-state (tag ('pc, cons (subr, length (code))),
                ctrl-stk,
                temp-stk,
                proc-list-code,
                data-segment,
                max-ctrl,
                MG-MAX-TEMP-STK-SIZE,
                word-size,
                'run),
       length (locals))
    =  p-state (tag ('pc, cons (subr, length (code) + length (locals))),
                ctrl-stk,
                append (reverse (ascending-local-address-sequence (locals,
                                                                   (length (temp-stk)
                                                                    −  n)− 1)),
                        temp-stk),
```

proc-list-code,
data-segment,
max-ctrl,
MG-MAX-TEMP-STK-SIZE,
word-size,
`'run`))

DEFINITION:
call-push-actuals-induction-hint (*actuals*, *code*, *temp-stk*, *ctrl-stk*)
= **if** *actuals* ≃ **nil then t**
else call-push-actuals-induction-hint (cdr (*actuals*),
append (*code*,
list (list (`'push-local`,
car (*actuals*)))),
cons (cdr (assoc (car (*actuals*),
bindings (top (*ctrl-stk*)))),
temp-stk),
ctrl-stk) **endif**

THEOREM: call-push-actuals-effect
(((length (*temp-stk*) + length (*actuals*)) < MG-MAX-TEMP-STK-SIZE)
∧ (cdddr (assoc (*subr*, *proc-list-code*))
= append (*code*, append (push-actuals-code (*actuals*), *code2*))))
→ (p (p-state (tag (`'pc`, cons (*subr*, length (*code*))),
ctrl-stk,
temp-stk,
proc-list-code,
data-segment,
max-ctrl,
MG-MAX-TEMP-STK-SIZE,
word-size,
`'run`),
length (*actuals*))
= p-state (tag (`'pc`, cons (*subr*, length (*code*) + length (*actuals*))),
ctrl-stk,
append (reverse (mg-actuals-to-p-actuals (*actuals*,
bindings (top (*ctrl-stk*)))),
temp-stk),
proc-list-code,
data-segment,
max-ctrl,
MG-MAX-TEMP-STK-SIZE,
word-size,
`'run`))

THEOREM: call-push-parameters-effect1
(((length (*temp-stk*)
  + data-length (*locals*)
  + length (*locals*)
  + length (*actuals*))
 < MG-MAX-TEMP-STK-SIZE)
 ∧ ok-mg-local-data-plistp (*locals*)
 ∧ (cdddr (assoc (*subr*, *proc-list-code*))
    = append (*code*,
              append (push-parameters-code (*locals*, *actuals*), *code2*))))
→ (p (p-state (tag (`'pc`, cons (*subr*, length (*code*))),
               ctrl-stk,
               temp-stk,
               proc-list-code,
               data-segment,
               max-ctrl,
               MG-MAX-TEMP-STK-SIZE,
               word-size,
               `'run`),
      data-length (*locals*) + length (*locals*) + length (*actuals*))
   = p-state (tag (`'pc`,
                   cons (*subr*,
                         length (*code*)
                         + data-length (*locals*)
                         + length (*locals*)
                         + length (*actuals*))),
              ctrl-stk,
              append (reverse (mg-actuals-to-p-actuals (*actuals*,
                                                        bindings (top (*ctrl-stk*)))),
                      append (reverse (ascending-local-address-sequence (*locals*,
                                                                         length (*temp-stk*))),
                              append (reverse (mg-to-p-local-values (*locals*)),
                                      temp-stk))),
              proc-list-code,
              data-segment,
              max-ctrl,
              MG-MAX-TEMP-STK-SIZE,
              word-size,
              `'run`))

THEOREM: call-push-parameters-effect
((car (*stmt*) = `'proc-call-mg`)
 ∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 ∧ ok-mg-def-plistp (*proc-list*)

```
∧  ok-mg-statep(mg-state, r-cond-list)
∧  (code(translate-def-body(assoc(subr, proc-list), proc-list))
    =  append(code(translate(cinfo, t-cond-list, stmt, proc-list)),
              code2))
∧  user-defined-procp(subr, proc-list)
∧  mg-vars-list-ok-in-p-state(mg-alist(mg-state),
                              bindings(top(ctrl-stk)),
                              temp-stk)
∧  (¬ resources-inadequatep(stmt,
                            proc-list,
                            list(length(temp-stk),
                                 p-ctrl-stk-size(ctrl-stk))))
→  (p(map-down(mg-state,
                proc-list,
                ctrl-stk,
                temp-stk,
                tag('pc, cons(subr, length(code(cinfo)))),
                t-cond-list),
       data-length(def-locals(fetch-called-def(stmt, proc-list)))
       +  length(def-locals(fetch-called-def(stmt, proc-list)))
       +  length(call-actuals(stmt)))
    =  p-state(tag('pc,
                   cons(subr,
                        length(code(cinfo))
                        +  data-length(def-locals(fetch-called-def(stmt,
                                                                   proc-list)))
                        +  length(def-locals(fetch-called-def(stmt,
                                                              proc-list)))
                        +  length(call-actuals(stmt)))),
               ctrl-stk,
               append(reverse(mg-actuals-to-p-actuals(call-actuals(stmt),
                                                      bindings(top(ctrl-stk)))),
                      append(reverse(ascending-local-address-sequence(def-locals(fetch-called-def(stmt
                                                                                                  proc
                                                                     length(temp-stk))),
                             append(reverse(mg-to-p-local-values(def-locals(fetch-called-def(stmt,
                                                                                            proc-list)
                                                                map-down-values(mg-alist(mg-state),
                                                                                bindings(top(ctrl-stk)),
                                                                                temp-stk)))),
                             translate-proc-list(proc-list),
                             list(list('c-c,
                                       mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
                             MG-MAX-CTRL-STK-SIZE,
```

MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: call-call-step
((*n* ≠ 0)
 ∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
 ∧ (car (*stmt*) = 'proc-call-mg)
 ∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 ∧ ok-mg-def-plistp (*proc-list*)
 ∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
 ∧ ok-mg-statep (*mg-state*, *r-cond-list*)
 ∧ cond-subsetp (*r-cond-list*, *t-cond-list*)
 ∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
 = append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
 ∧ user-defined-procp (*subr*, *proc-list*)
 ∧ plistp (*temp-stk*)
 ∧ listp (*ctrl-stk*)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
 ∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
 ∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
 ∧ normal (*mg-state*)
 ∧ all-cars-unique (mg-alist (*mg-state*))
 ∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (p-step (p-state (tag ('pc,
cons (*subr*,
length (code (*cinfo*))
+ data-length (def-locals (fetch-called-def (*stmt*,
proc-list)))
+ length (def-locals (fetch-called-def (*stmt*,
proc-list)))
+ length (call-actuals (*stmt*)))),
ctrl-stk,

```
                append (reverse (mg-actuals-to-p-actuals (call-actuals (stmt),
                                                          bindings (top (ctrl-stk)))),
                        append (reverse (ascending-local-address-sequence (def-locals (fetch-called-def (s
                                                                                                          p
                                                                           length (temp-stk))),
                                append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (stmt,
                                                                                                proc-l
                                                map-down-values (mg-alist (mg-state),
                                                                 bindings (top (ctrl-stk)),
                                                                 temp-stk)))),
                translate-proc-list (proc-list),
                list (list ('c-c,
                        mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))
=   p-state (tag ('pc, cons (call-name (stmt), 0)),
             push (p-frame (append (pairlist (cadr (assoc (call-name (stmt),
                                                           translate-proc-list (proc-list))),
                                              append (ascending-local-address-sequence (def-locals (fetch-c

                                                                                                length (temp-stk
                                                                    mg-actuals-to-p-actuals (call-actuals (stmt),
                                                                                             bindings (top (ctrl-stk)))))
                                                   pair-temps-with-initial-values (caddr (assoc (call-name (stmt),
                                                                                                 translate-proc-list (proc
                                   tag ('pc,
                                        cons (subr,
                                              length (code (cinfo))
                                              +   data-length (def-locals (fetch-called-def (stmt,
                                                                                             proc-list)))
                                              +   length (def-locals (fetch-called-def (stmt,
                                                                                        proc-list)))
                                              +   length (call-actuals (stmt))
                                              +   1))),
                          ctrl-stk),
             append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (stmt,
                                                                                  proc-list)))),
                     map-down-values (mg-alist (mg-state),
                                      bindings (top (ctrl-stk)),
                                      temp-stk)),
             translate-proc-list (proc-list),
             list (list ('c-c,
```

mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: call-steps-to-body
((*n* ≠ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'proc-call-mg`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ cond-subsetp (*r-cond-list*, *t-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ plistp (*temp-stk*)
∧ listp (*ctrl-stk*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ normal (*mg-state*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (p (map-down (*mg-state*,
proc-list,
ctrl-stk,
temp-stk,
tag (`'pc`, cons (*subr*, length (code (*cinfo*)))),
t-cond-list),
data-length (def-locals (fetch-called-def (*stmt*, *proc-list*)))

```
    +   length (def-locals (fetch-called-def (stmt, proc-list)))
    +   length (call-actuals (stmt))
    +   1)
=   p-state (tag ('pc, cons (call-name (stmt), 0)),
             push (p-frame (append (pairlist (cadr (assoc (call-name (stmt),
                                                           translate-proc-list (proc-list))),
                                                  append (ascending-local-address-sequence (def-locals (fetch-
                                                                                                        length (temp-stk
                                                                                           mg-actuals-to-p-actuals (call-actuals (stmt),
                                                                                                                    bindings (top (ctrl-stk)))))
                                                                    pair-temps-with-initial-values (caddr (assoc (call-name (stmt),
                                                                                                                  translate-proc-list (proc
                                                      tag ('pc,
                                                           cons (subr,
                                                                 length (code (cinfo))
                                                                 +   data-length (def-locals (fetch-called-def (stmt,
                                                                                                                proc-list)))
                                                                 +   length (def-locals (fetch-called-def (stmt,
                                                                                                           proc-list)))
                                                                 +   length (call-actuals (stmt))
                                                                 +   1))),
                                   ctrl-stk),
             append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (stmt,
                                                                                  proc-list)))),
                             map-down-values (mg-alist (mg-state),
                                              bindings (top (ctrl-stk)),
                                              temp-stk)),
             translate-proc-list (proc-list),
             list (list ('c-c,
                         mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
             MG-MAX-CTRL-STK-SIZE,
             MG-MAX-TEMP-STK-SIZE,
             MG-WORD-SIZE,
             'run))
```

EVENT: Disable deposit-alist-value.

THEOREM: call-local-var-lists-match1

```
(k ∈ N)
→   (pairlist (listcars (locals), ascending-local-address-sequence (locals, k))
    =   map-call-locals (locals, k))
```

THEOREM: call-formal-var-lists-match

data-param-lists-match (*actuals*, *formals*, *alist*)
→ (pairlist (listcars (*formals*), mg-actuals-to-p-actuals (*actuals*, *bindings*))
= map-call-formals (*formals*, *actuals*, *bindings*))

DEFINITION:
map-down-locals-doesnt-affect-formals-induction-hint (*formals*,
actuals,
temp-stk,
mg-alist,
bindings)
= **if** *formals* ≃ **nil then t**
else map-down-locals-doesnt-affect-formals-induction-hint (cdr (*formals*),
cdr (*actuals*),
deposit-alist-value (list (caar (*formals*),
cadar (*formals*),
caddr (assoc (car (*actu*
mg-alist)
cons (cons (car (car (*formal*
cdr (assoc (car (
bind
map-call-formals (cdr
cdr
bin
temp-stk),
mg-alist,
bindings) **endif**

THEOREM: map-down-locals-doesnt-affect-formals
all-cars-unique (append (*formals*, *locals*))
→ (map-down-values (make-call-param-alist (*formals*, *actuals*, *mg-alist*),
append (map-call-locals (*locals*, *n*),
map-call-formals (*formals*, *actuals*, *bindings*)),
temp-stk)
= map-down-values (make-call-param-alist (*formals*,
actuals,
mg-alist),
map-call-formals (*formals*, *actuals*, *bindings*),
temp-stk))

DEFINITION:
map-down-formals-doesnt-affect-locals-induction-hint (*locals*, *temp-stk*, *n*, *lst*)
= **if** *locals* ≃ **nil then t**
elseif simple-mg-type-refp (cadr (car (*locals*)))
then map-down-formals-doesnt-affect-locals-induction-hint (cdr (*locals*),
deposit-alist-value (car (*locals*),

append (map-call-locals (*l*
r
lst),
temp-stk),
$1 + n$,
lst)
else map-down-formals-doesnt-affect-locals-induction-hint (cdr (*locals*),
deposit-alist-value (car (*locals*),
append (map-call-locals (*lo*
n)
lst),
temp-stk),
array-length (cadr (car (*locals*)))
$+ \quad n$,
lst) **endif**

THEOREM: map-down-formals-doesnt-affect-locals
all-cars-unique (*locals*)
$\rightarrow$ (map-down-values (*locals*,
append (map-call-locals (*locals*, *n*), *lst*),
temp-stk)
$=$ map-down-values (*locals*, map-call-locals (*locals*, *n*), *temp-stk*))

THEOREM: map-down-skips-non-referenced-segment
(mg-alistp (*vars*) $\wedge$ mg-vars-list-ok-in-p-state (*vars*, *bindings*, *temp-stk*))
$\rightarrow$ (map-down-values (*vars*, *bindings*, append (*lst*, *temp-stk*))
$=$ append (*lst*, map-down-values (*vars*, *bindings*, *temp-stk*)))

DEFINITION:
map-down-locals-induction-hint (*locals*, *n*, *temp-stk*)
$=$ **if** *locals* $\simeq$ **nil then t**
elseif simple-mg-type-refp (cadr (car (*locals*)))
then map-down-locals-induction-hint (cdr (*locals*),
$1 + n$,
cons (mg-to-p-simple-literal (caddar (*locals*)),
temp-stk))
else map-down-locals-induction-hint (cdr (*locals*),
array-length (cadr (car (*locals*)))
$+ \quad n$,
append (reverse (mg-to-p-simple-literal-list (caddr (car (*locals*)))),
temp-stk)) **endif**

THEOREM: map-down-locals-equals-reverse-values
((n = length (*temp-stk*))
$\wedge$ all-cars-unique (*locals*)

$\wedge$ ok-mg-local-data-plistp (*locals*)
$\wedge$ plistp (*temp-stk*))
$\rightarrow$ (map-down-values (*locals*,
map-call-locals (*locals*, *n*),
append (reverse (mg-to-p-local-values (*locals*)),
temp-stk))
$=$ append (reverse (mg-to-p-local-values (*locals*)), *temp-stk*))

THEOREM: map-down-again-preserves-values
(mg-vars-list-ok-in-p-state (*mg-vars*, *bindings*, *temp-stk*)
$\wedge$ ok-actual-params-list (*actuals*, *mg-vars*)
$\wedge$ data-param-lists-match (*actuals*, *formals*, *mg-vars*)
$\wedge$ all-cars-unique (*mg-vars*)
$\wedge$ plistp (*temp-stk*)
$\wedge$ no-p-aliasing (*bindings*, *mg-vars*)
$\wedge$ all-cars-unique (*formals*)
$\wedge$ mg-alistp (*mg-vars*))
$\rightarrow$ (map-down-values (make-call-param-alist (*formals*, *actuals*, *mg-vars*),
map-call-formals (*formals*, *actuals*, *bindings*),
map-down-values (*mg-vars*, *bindings*, *temp-stk*))
$=$ map-down-values (*mg-vars*, *bindings*, *temp-stk*))

EVENT: Enable deposit-temp.

THEOREM: array-formal-ok-for-actual2
(definedp (*actual*, *mg-alist*)
$\wedge$ data-params-match (*actual*, *formal*, *mg-alist*)
$\wedge$ mg-vars-list-ok-in-p-state (*mg-alist*, *bindings*, *temp-stk*)
$\wedge$ ($\neg$ simple-mg-type-refp (cadr (*formal*))))
$\rightarrow$ ok-temp-stk-array-index (cdr (assoc (*actual*, *bindings*)),
map-down-values (*mg-alist*, *bindings*, *temp-stk*),
array-length (cadr (*formal*)))

EVENT: Disable array-formal-ok-for-actual2.

THEOREM: call-environment-mg-vars-list-ok1
(no-p-aliasing (*bindings*, *mg-vars*)
$\wedge$ mg-alistp (*mg-vars*)
$\wedge$ all-cars-unique (*formals*)
$\wedge$ ok-actual-params-list (*actuals*, *mg-vars*)
$\wedge$ ok-mg-formal-data-params-plistp (*formals*)
$\wedge$ data-param-lists-match (*actuals*, *formals*, *mg-vars*)
$\wedge$ mg-vars-list-ok-in-p-state (*mg-vars*, *bindings*, *temp-stk*))
$\rightarrow$ mg-vars-list-ok-in-p-state (make-call-param-alist (*formals*,

actuals,
mg-vars),
map-call-formals (*formals*, *actuals*, *bindings*),
map-down-values (*mg-vars*, *bindings*, *temp-stk*))

THEOREM: map-down-preserves-references
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'proc-call-mg)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ cond-subsetp (*r-cond-list*, *t-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ plistp (*temp-stk*)
∧ listp (*ctrl-stk*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ normal (*mg-state*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (*stmt*,
proc-list)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))
= map-down-values (make-call-var-alist (mg-alist (*mg-state*),
stmt,
fetch-called-def (*stmt*,

proc-list)),
make-frame-alist (fetch-called-def (*stmt*,
proc-list),
stmt,
ctrl-stk,
temp-stk),
append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (*stmt*,
proc-list)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))))

THEOREM: call-step-initial-equals-state1
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'proc-call-mg)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ cond-subsetp (*r-cond-list*, *t-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ plistp (*temp-stk*)
∧ listp (*ctrl-stk*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ normal (*mg-state*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (p-state (tag ('pc, cons (call-name (*stmt*), 0)),

```
        push (p-frame (append (pairlist (cadr (assoc (call-name (stmt),
                                                     translate-proc-list (proc-list))),
                                              append (ascending-local-address-sequence (def-locals (fetch-calle

                                                                                       length (temp-stk)),
                                                     mg-actuals-to-p-actuals (call-actuals (stmt),
                                                                              bindings (top (ctrl-stk))))),
                                              pair-temps-with-initial-values (caddr (assoc (call-name (stmt),
                                                                                            translate-proc-list (proc-list)
                                  tag ('pc,
                                       cons (subr,
                                             length (code (cinfo))
                                             +  data-length (def-locals (fetch-called-def (stmt,
                                                                                           proc-list)))
                                             +  length (def-locals (fetch-called-def (stmt,
                                                                                      proc-list)))
                                             +  length (call-actuals (stmt))
                                             +  1))),
                             ctrl-stk),
        append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (stmt,
                                                                             proc-list)))),
                        map-down-values (mg-alist (mg-state),
                                         bindings (top (ctrl-stk)),
                                         temp-stk)),
        translate-proc-list (proc-list),
        list (list ('c-c, mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
        MG-MAX-CTRL-STK-SIZE,
        MG-MAX-TEMP-STK-SIZE,
        MG-WORD-SIZE,
        'run)
=   map-down (make-call-environment (mg-state,
                                     stmt,
                                     fetch-called-def (stmt,
                                                       proc-list)),
              proc-list,
              cons (p-frame (make-frame-alist (fetch-called-def (stmt,
                                                                 proc-list),
                                               stmt,
                                               ctrl-stk,
                                               temp-stk),
                             tag ('pc,
                                  cons (subr,
                                        1 + (length (code (cinfo))
                                             +  data-length (def-locals (fetch-called-def (stmt,
```

proc-list)))
+ length (def-locals (fetch-called-def (*stmt*,
proc-list)))
+ length (call-actuals (*stmt*))))))),
ctrl-stk),
append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (*stmt*,
proc-list)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
tag (`'pc`,
cons (call-name (*stmt*),
length (code (make-cinfo (**nil**,
cons (`'(routineerror`
`. 0)`,
make-label-alist (make-cond-list (fetch-called-d
`0)),
`1`))))),
make-cond-list (fetch-called-def (*stmt*, *proc-list*))))

```
;; This lemma characterizes the bevavior all of the way through the call-body
```

```
;; This lemma characterizes the bevavior all of the way through the call-body

(prove-lemma call-step-initial-to-state2 (rewrite)
      (IMPLIES
       (AND (NOT (ZEROP N))
    (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
    (EQUAL (CAR STMT) 'PROC-CALL-MG)
    (OK-MG-STATEMENT STMT R-COND-LIST NAME-ALIST PROC-LIST)
    (OK-MG-DEF-PLISTP PROC-LIST)
    (OK-TRANSLATION-PARAMETERS CINFO T-COND-LIST STMT PROC-LIST CODE2)
    (OK-MG-STATEP MG-STATE R-COND-LIST)
    (COND-SUBSETP R-COND-LIST T-COND-LIST)
    (EQUAL (CODE (TRANSLATE-DEF-BODY (ASSOC SUBR PROC-LIST)
     PROC-LIST))
   (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
   CODE2))
```

```
    (USER-DEFINED-PROCP SUBR PROC-LIST)
    (PLISTP TEMP-STK)
    (LISTP CTRL-STK)
    (MG-VARS-LIST-OK-IN-P-STATE (MG-ALIST MG-STATE)
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
    (NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
   (MG-ALIST MG-STATE))
    (SIGNATURES-MATCH (MG-ALIST MG-STATE)
      NAME-ALIST)
    (NORMAL MG-STATE)
    (ALL-CARS-UNIQUE (MG-ALIST MG-STATE))
    (NOT (RESOURCE-ERRORP (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))
           (EQUAL
      (P
       (MAP-DOWN                                              ;; state1
(MAKE-CALL-ENVIRONMENT MG-STATE STMT
       (FETCH-CALLED-DEF STMT PROC-LIST))
PROC-LIST
(CONS
 (P-FRAME
  (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
    STMT CTRL-STK TEMP-STK)
  (TAG 'PC
       (CONS SUBR
     (ADD1
      (PLUS (LENGTH (CODE CINFO))
    (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
    (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
    (LENGTH (CALL-ACTUALS STMT)))))))
 CTRL-STK)
(APPEND
 (REVERSE
  (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
 (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
  (BINDINGS (TOP CTRL-STK))
  TEMP-STK))
(TAG 'PC
     (CONS
      (CALL-NAME STMT)
      (LENGTH
       (CODE
```

```
(MAKE-CINFO NIL
    (CONS
     '(ROUTINEERROR . 0)
     (MAKE-LABEL-ALIST
      (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
      0))
    1)))))
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))
       (CLOCK (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
      PROC-LIST
      (MAKE-CALL-ENVIRONMENT MG-STATE STMT
     (FETCH-CALLED-DEF STMT PROC-LIST))
      (SUB1 N)))
      (P-STATE                                        ;; state2
       (TAG 'PC
    (CONS
     (CALL-NAME STMT)
     (IF
      (NORMAL
       (MG-MEANING-R
(DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
PROC-LIST
(MAKE-CALL-ENVIRONMENT MG-STATE STMT
       (FETCH-CALLED-DEF STMT PROC-LIST))
(SUB1 N)
(LIST
 (LENGTH
  (APPEND
   (REVERSE
    (MG-TO-P-LOCAL-VALUES
     (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
   (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
    (BINDINGS (TOP CTRL-STK))
    TEMP-STK)))
 (P-CTRL-STK-SIZE
  (CONS
   (P-FRAME
    (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
      STMT CTRL-STK TEMP-STK)
    (TAG 'PC
 (CONS SUBR
       (ADD1
(PLUS
 (LENGTH (CODE CINFO))
```

```
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))))
      (LENGTH
       (CODE
(TRANSLATE
 (MAKE-CINFO NIL
     (CONS
      '(ROUTINEERROR . 0)
      (MAKE-LABEL-ALIST
       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
       0))
     1)
 (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
 (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
 PROC-LIST)))
      (FIND-LABEL
       (FETCH-LABEL
(CC
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MAKE-CALL-ENVIRONMENT MG-STATE STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
  (SUB1 N)
  (LIST
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES
                                (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
STMT CTRL-STK TEMP-STK)
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
```

```
   (LENGTH (CODE CINFO))
   (DATA-LENGTH
    (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH
    (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
(LABEL-ALIST
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST)))
       (APPEND
(CODE
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST))
(CONS
 '(DL 0 NIL (NO-OP))
 (CONS
  (LIST 'POP* (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
  '((RET)))))))))
       (CONS
(P-FRAME
 (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
   STMT CTRL-STK TEMP-STK)
 (TAG 'PC
      (CONS SUBR
    (ADD1
```

```
     (PLUS (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
CTRL-STK)
       (MAP-DOWN-VALUES
(MG-ALIST
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MAKE-CALL-ENVIRONMENT MG-STATE STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
  (SUB1 N)
  (LIST
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES
       (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
STMT CTRL-STK TEMP-STK)
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
   (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
(BINDINGS
 (TOP
  (CONS
   (P-FRAME
    (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
      STMT CTRL-STK TEMP-STK)
    (TAG 'PC
 (CONS SUBR
```

```
       (ADD1
(PLUS (LENGTH (CODE CINFO))
      (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))
(APPEND
 (REVERSE
  (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
 (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
  (BINDINGS (TOP CTRL-STK))
  TEMP-STK)))
       (TRANSLATE-PROC-LIST PROC-LIST)
       (LIST
(LIST 'C-C
      (MG-COND-TO-P-NAT
       (CC
(MG-MEANING-R
 (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
 PROC-LIST
 (MAKE-CALL-ENVIRONMENT MG-STATE STMT
(FETCH-CALLED-DEF STMT PROC-LIST))
 (SUB1 N)
 (LIST
  (LENGTH
   (APPEND
    (REVERSE
     (MG-TO-P-LOCAL-VALUES
      (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
     (BINDINGS (TOP CTRL-STK))
     TEMP-STK)))
  (P-CTRL-STK-SIZE
   (CONS
    (P-FRAME
     (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
       STMT CTRL-STK TEMP-STK)
     (TAG 'PC
  (CONS SUBR
(ADD1
 (PLUS
  (LENGTH (CODE CINFO))
  (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
  (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
```

```
  (LENGTH (CALL-ACTUALS STMT)))))))
    CTRL-STK)))))
       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
       (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
       'RUN)))
       (EQUAL
(P (MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK
     (TAG 'PC
  (CONS SUBR (LENGTH (CODE CINFO))))
     T-COND-LIST)
           (plus (data-length (def-locals (fetch-called-def stmt proc-list)))
 (length (def-locals (fetch-called-def stmt proc-list)))
 (length (call-actuals stmt))
 1
 (CLOCK (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
PROC-LIST
(MAKE-CALL-ENVIRONMENT MG-STATE STMT
       (FETCH-CALLED-DEF STMT PROC-LIST))
(SUB1 N))))
      (P-STATE                                    ;; state2
       (TAG 'PC
    (CONS
     (CALL-NAME STMT)
     (IF
      (NORMAL
       (MG-MEANING-R
(DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
PROC-LIST
(MAKE-CALL-ENVIRONMENT MG-STATE STMT
       (FETCH-CALLED-DEF STMT PROC-LIST))
(SUB1 N)
(LIST
 (LENGTH
  (APPEND
   (REVERSE
    (MG-TO-P-LOCAL-VALUES
     (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
   (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
    (BINDINGS (TOP CTRL-STK))
    TEMP-STK)))
 (P-CTRL-STK-SIZE
  (CONS
   (P-FRAME
    (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
```

```
      STMT CTRL-STK TEMP-STK)
    (TAG 'PC
 (CONS SUBR
       (ADD1
(PLUS
 (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))))
      (LENGTH
       (CODE
(TRANSLATE
 (MAKE-CINFO NIL
     (CONS
      '(ROUTINEERROR . 0)
      (MAKE-LABEL-ALIST
       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
       0))
     1)
 (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
 (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
 PROC-LIST)))
      (FIND-LABEL
       (FETCH-LABEL
(CC
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MAKE-CALL-ENVIRONMENT MG-STATE STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
  (SUB1 N)
  (LIST
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES
                                (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
```

```
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
STMT CTRL-STK TEMP-STK)
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
   (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
(LABEL-ALIST
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST)))
       (APPEND
(CODE
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST))
(CONS
 '(DL 0 NIL (NO-OP))
 (CONS
  (LIST 'POP* (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
  '((RET)))))))))
       (CONS
(P-FRAME
 (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
```

```
   STMT CTRL-STK TEMP-STK)
 (TAG 'PC
      (CONS SUBR
    (ADD1
     (PLUS (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
CTRL-STK)
       (MAP-DOWN-VALUES
(MG-ALIST
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MAKE-CALL-ENVIRONMENT MG-STATE STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
  (SUB1 N)
  (LIST
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES
       (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
STMT CTRL-STK TEMP-STK)
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
   (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
(BINDINGS
 (TOP
  (CONS
   (P-FRAME
```

```
    (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
      STMT CTRL-STK TEMP-STK)
    (TAG 'PC
 (CONS SUBR
       (ADD1
(PLUS (LENGTH (CODE CINFO))
      (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))
(APPEND
 (REVERSE
  (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
 (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
  (BINDINGS (TOP CTRL-STK))
  TEMP-STK)))
       (TRANSLATE-PROC-LIST PROC-LIST)
       (LIST
(LIST 'C-C
      (MG-COND-TO-P-NAT
       (CC
(MG-MEANING-R
 (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
 PROC-LIST
 (MAKE-CALL-ENVIRONMENT MG-STATE STMT
(FETCH-CALLED-DEF STMT PROC-LIST))
 (SUB1 N)
 (LIST
  (LENGTH
   (APPEND
    (REVERSE
     (MG-TO-P-LOCAL-VALUES
      (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
     (BINDINGS (TOP CTRL-STK))
     TEMP-STK)))
  (P-CTRL-STK-SIZE
   (CONS
    (P-FRAME
     (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
       STMT CTRL-STK TEMP-STK)
     (TAG 'PC
  (CONS SUBR
(ADD1
```

```
(PLUS
 (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))))
      (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
      (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
      'RUN)))
((INSTRUCTIONS
  (ADD-ABBREVIATION @INITIAL
                   (MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK
                             (TAG 'PC
                                  (CONS SUBR (LENGTH (CODE CINFO))))
                             T-COND-LIST))
  (ADD-ABBREVIATION @STATE1
   (MAP-DOWN
    (MAKE-CALL-ENVIRONMENT MG-STATE STMT
                           (FETCH-CALLED-DEF STMT PROC-LIST))
    PROC-LIST
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                        STMT CTRL-STK TEMP-STK)
      (TAG 'PC
       (CONS SUBR
        (ADD1
            (PLUS (LENGTH (CODE CINFO))
                  (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                  (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                  (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)
    (APPEND
     (REVERSE
        (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                      (BINDINGS (TOP CTRL-STK))
                      TEMP-STK))
    (TAG 'PC
     (CONS
      (CALL-NAME STMT)
      (LENGTH
       (CODE
        (MAKE-CINFO NIL
```

```
       (CONS
        '(ROUTINEERROR . 0)
        (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                          0))
       1)))))
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))))
(ADD-ABBREVIATION @STATE2
 (P-STATE
  (TAG 'PC
   (CONS
    (CALL-NAME STMT)
    (IF
     (NORMAL
      (MG-MEANING-R
       (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
       PROC-LIST
       (MAKE-CALL-ENVIRONMENT MG-STATE STMT
                              (FETCH-CALLED-DEF STMT PROC-LIST))
       (SUB1 N)
       (LIST
        (LENGTH
         (APPEND
          (REVERSE
           (MG-TO-P-LOCAL-VALUES
                            (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
          (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                           (BINDINGS (TOP CTRL-STK))
                           TEMP-STK)))
        (P-CTRL-STK-SIZE
         (CONS
          (P-FRAME
           (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                             STMT CTRL-STK TEMP-STK)
           (TAG 'PC
            (CONS SUBR
             (ADD1
              (PLUS
                (LENGTH (CODE CINFO))
                (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                (LENGTH (CALL-ACTUALS STMT)))))))
          CTRL-STK)))))
     (LENGTH
      (CODE
```

```
  (TRANSLATE
   (MAKE-CINFO NIL
    (CONS
     '(ROUTINEERROR . 0)
     (MAKE-LABEL-ALIST
                    (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                    0))
    1)
   (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
   (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
   PROC-LIST)))
(FIND-LABEL
 (FETCH-LABEL
  (CC
   (MG-MEANING-R
    (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
    PROC-LIST
    (MAKE-CALL-ENVIRONMENT MG-STATE STMT
                           (FETCH-CALLED-DEF STMT PROC-LIST))
    (SUB1 N)
    (LIST
     (LENGTH
      (APPEND
       (REVERSE
        (MG-TO-P-LOCAL-VALUES
                       (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
       (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                        (BINDINGS (TOP CTRL-STK))
                        TEMP-STK)))
     (P-CTRL-STK-SIZE
      (CONS
       (P-FRAME
        (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                          STMT CTRL-STK TEMP-STK)
        (TAG 'PC
         (CONS SUBR
          (ADD1
           (PLUS
            (LENGTH (CODE CINFO))
            (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
            (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
            (LENGTH (CALL-ACTUALS STMT)))))))
       CTRL-STK)))))
  (LABEL-ALIST
```

```
      (TRANSLATE
       (MAKE-CINFO NIL
        (CONS
         '(ROUTINEERROR . 0)
         (MAKE-LABEL-ALIST
                       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                       0))
        1)
       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
       (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
       PROC-LIST)))
    (APPEND
     (CODE
      (TRANSLATE
       (MAKE-CINFO NIL
        (CONS
         '(ROUTINEERROR . 0)
         (MAKE-LABEL-ALIST
                       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                       0))
        1)
       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
       (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
       PROC-LIST))
     (CONS
      '(DL 0 NIL (NO-OP))
      (CONS
       (LIST 'POP* (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
       '((RET)))))))))
(CONS
 (P-FRAME
  (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                    STMT CTRL-STK TEMP-STK)
  (TAG 'PC
   (CONS SUBR
    (ADD1
        (PLUS (LENGTH (CODE CINFO))
              (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
              (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
              (LENGTH (CALL-ACTUALS STMT)))))))
 CTRL-STK)
(MAP-DOWN-VALUES
 (MG-ALIST
  (MG-MEANING-R
```

```
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MAKE-CALL-ENVIRONMENT MG-STATE STMT
                         (FETCH-CALLED-DEF STMT PROC-LIST))
  (SUB1 N)
  (LIST
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES
                          (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                      (BINDINGS (TOP CTRL-STK))
                      TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                        STMT CTRL-STK TEMP-STK)
      (TAG 'PC
       (CONS SUBR
        (ADD1
         (PLUS
             (LENGTH (CODE CINFO))
             (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
             (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
             (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
(BINDINGS
 (TOP
  (CONS
   (P-FRAME
    (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                      STMT CTRL-STK TEMP-STK)
    (TAG 'PC
     (CONS SUBR
      (ADD1
       (PLUS (LENGTH (CODE CINFO))
             (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
             (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
             (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))
(APPEND
 (REVERSE
```

```
      (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                     (BINDINGS (TOP CTRL-STK))
                     TEMP-STK)))
  (TRANSLATE-PROC-LIST PROC-LIST)
  (LIST
   (LIST 'C-C
    (MG-COND-TO-P-NAT
     (CC
      (MG-MEANING-R
       (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
       PROC-LIST
       (MAKE-CALL-ENVIRONMENT MG-STATE STMT
                              (FETCH-CALLED-DEF STMT PROC-LIST))
       (SUB1 N)
       (LIST
        (LENGTH
         (APPEND
          (REVERSE
           (MG-TO-P-LOCAL-VALUES
                            (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
          (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                           (BINDINGS (TOP CTRL-STK))
                           TEMP-STK)))
        (P-CTRL-STK-SIZE
         (CONS
          (P-FRAME
           (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                             STMT CTRL-STK TEMP-STK)
           (TAG 'PC
            (CONS SUBR
             (ADD1
              (PLUS
                (LENGTH (CODE CINFO))
                (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                (LENGTH (CALL-ACTUALS STMT)))))))
          CTRL-STK)))))
     (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
  (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
  'RUN))
(ADD-ABBREVIATION @BODY-TIME
            (CLOCK (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
                   PROC-LIST
```

```
                    (MAKE-CALL-ENVIRONMENT MG-STATE STMT
                                           (FETCH-CALLED-DEF STMT PROC-LIST))
                    (SUB1 N)))
  (ADD-ABBREVIATION @TIME-TO-STATE1
            (PLUS (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                  (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                  (LENGTH (CALL-ACTUALS STMT))
                  1))
  PROMOTE
  (DIVE 1 2)
  (= (PLUS (PLUS (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                 (LENGTH (CALL-ACTUALS STMT))
                 1)
           @BODY-TIME))
  UP
  (REWRITE P-PLUS-LEMMA)
  (DIVE 1)
  (REWRITE CALL-STEPS-TO-BODY)
  (REWRITE CALL-STEP-INITIAL-EQUALS-STATE1)
  UP UP
  (DEMOTE 19)
  S-PROP)))
```

Event: Enable make-call-environment.

```
(prove-lemma call-state2-step1-effect (rewrite)
      (IMPLIES
       (AND (NOT (ZEROP N))
    (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
    (EQUAL (CAR STMT) 'PROC-CALL-MG)
    (OK-MG-STATEMENT STMT R-COND-LIST NAME-ALIST PROC-LIST)
    (OK-MG-DEF-PLISTP PROC-LIST)
    (OK-TRANSLATION-PARAMETERS CINFO T-COND-LIST STMT PROC-LIST CODE2)
    (OK-MG-STATEP MG-STATE R-COND-LIST)
    (COND-SUBSETP R-COND-LIST T-COND-LIST)
    (EQUAL (CODE (TRANSLATE-DEF-BODY (ASSOC SUBR PROC-LIST)
     PROC-LIST))
   (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
   CODE2))
```

```
    (USER-DEFINED-PROCP SUBR PROC-LIST)
    (PLISTP TEMP-STK)
    (LISTP CTRL-STK)
    (MG-VARS-LIST-OK-IN-P-STATE (MG-ALIST MG-STATE)
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
    (NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
   (MG-ALIST MG-STATE))
    (SIGNATURES-MATCH (MG-ALIST MG-STATE)
      NAME-ALIST)
    (NORMAL MG-STATE)
    (ALL-CARS-UNIQUE (MG-ALIST MG-STATE))
    (NOT (RESOURCE-ERRORP (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))))
     (equal
      (p-step
              (P-STATE                                        ;; state2
       (TAG 'PC
    (CONS
     (CALL-NAME STMT)
     (IF
      (NORMAL
       (MG-MEANING-R
(DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
PROC-LIST
(MAKE-CALL-ENVIRONMENT MG-STATE STMT
       (FETCH-CALLED-DEF STMT PROC-LIST))
(SUB1 N)
(LIST
 (LENGTH
  (APPEND
   (REVERSE
    (MG-TO-P-LOCAL-VALUES
     (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
   (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
    (BINDINGS (TOP CTRL-STK))
    TEMP-STK)))
 (P-CTRL-STK-SIZE
  (CONS
   (P-FRAME
    (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
      STMT CTRL-STK TEMP-STK)
    (TAG 'PC
```

```
 (CONS SUBR
       (ADD1
(PLUS
 (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))))
      (LENGTH
       (CODE
(TRANSLATE
 (MAKE-CINFO NIL
     (CONS
      '(ROUTINEERROR . 0)
      (MAKE-LABEL-ALIST
       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
       0))
     1)
 (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
 (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
 PROC-LIST)))
      (FIND-LABEL
       (FETCH-LABEL
(CC
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MAKE-CALL-ENVIRONMENT MG-STATE STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
  (SUB1 N)
  (LIST
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES
                                (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
STMT CTRL-STK TEMP-STK)
```

```
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
   (LENGTH (CODE CINFO))
   (DATA-LENGTH
    (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH
    (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
(LABEL-ALIST
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST)))
       (APPEND
(CODE
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST))
(CONS
 '(DL 0 NIL (NO-OP))
 (CONS
  (LIST 'POP* (Data-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
  '((RET)))))))))
       (CONS
(P-FRAME
 (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
```

```
   STMT CTRL-STK TEMP-STK)
 (TAG 'PC
      (CONS SUBR
    (ADD1
     (PLUS (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
CTRL-STK)
       (MAP-DOWN-VALUES
(MG-ALIST
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MAKE-CALL-ENVIRONMENT MG-STATE STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
  (SUB1 N)
  (LIST
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES
       (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
STMT CTRL-STK TEMP-STK)
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
   (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
(BINDINGS
 (TOP
  (CONS
   (P-FRAME
```

```
    (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
      STMT CTRL-STK TEMP-STK)
    (TAG 'PC
 (CONS SUBR
       (ADD1
(PLUS (LENGTH (CODE CINFO))
      (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH (CALL-ACTUALS STMT)))))))
   CTRL-STK)))
(APPEND
 (REVERSE
  (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
 (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
  (BINDINGS (TOP CTRL-STK))
  TEMP-STK)))
       (TRANSLATE-PROC-LIST PROC-LIST)
       (LIST
(LIST 'C-C
      (MG-COND-TO-P-NAT
       (CC
(MG-MEANING-R
 (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
 PROC-LIST
 (MAKE-CALL-ENVIRONMENT MG-STATE STMT
(FETCH-CALLED-DEF STMT PROC-LIST))
 (SUB1 N)
 (LIST
  (LENGTH
   (APPEND
    (REVERSE
     (MG-TO-P-LOCAL-VALUES
      (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
     (BINDINGS (TOP CTRL-STK))
     TEMP-STK)))
  (P-CTRL-STK-SIZE
   (CONS
    (P-FRAME
     (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
       STMT CTRL-STK TEMP-STK)
     (TAG 'PC
  (CONS SUBR
(ADD1
```

```
 (PLUS
  (LENGTH (CODE CINFO))
  (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
  (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
  (LENGTH (CALL-ACTUALS STMT)))))))
    CTRL-STK)))))
        (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
        (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
        'RUN))
        (P-STATE (TAG 'PC
      (CONS
       (CALL-NAME STMT)
       (PLUS
        (LENGTH
(CODE
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
 0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST)))
       1)))
(CONS
 (P-FRAME
  (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
   (LENGTH TEMP-STK))
  (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
    (CALL-ACTUALS STMT)
    (BINDINGS (TOP CTRL-STK))))
  (TAG 'PC
       (CONS SUBR
     (ADD1 (PLUS (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
 CTRL-STK)
(MAP-DOWN-VALUES
 (MG-ALIST
  (MG-MEANING-R
   (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
```

```
   PROC-LIST
   (MG-STATE (CC MG-STATE)
     (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
  STMT
  (FETCH-CALLED-DEF STMT PROC-LIST))
     (MG-PSW MG-STATE))
   (SUB1 N)
   (LIST (PLUS (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
       (LENGTH TEMP-STK))
 (PLUS 2
       (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
       (LENGTH (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST)))
       (P-CTRL-STK-SIZE CTRL-STK)))))
 (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
  (LENGTH TEMP-STK))
 (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
   (CALL-ACTUALS STMT)
   (BINDINGS (TOP CTRL-STK))))
 (APPEND
  (REVERSE
   (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
  (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
   (BINDINGS (TOP CTRL-STK))
   TEMP-STK)))
(TRANSLATE-PROC-LIST PROC-LIST)
(LIST
 (LIST 'C-C
       (MG-COND-TO-P-NAT
(CC
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MG-STATE (CC MG-STATE)
    (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
 STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
    (MG-PSW MG-STATE))
  (SUB1 N)
  (LIST
   (PLUS
    (LENGTH
     (MG-TO-P-LOCAL-VALUES
      (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (LENGTH (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
```

```
     (BINDINGS (TOP CTRL-STK))
     TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (APPEND
       (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
(LENGTH TEMP-STK))
       (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
 (CALL-ACTUALS STMT)
 (BINDINGS (TOP CTRL-STK))))
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
   (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
    CTRL-STK)))))
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
               (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
'RUN)))
 ((INSTRUCTIONS
   (ENABLE LENGTH-CONS)
   PROMOTE
   (DIVE 1)
   X
   (S LEMMAS)
   (DIVE 1 1 1)
   (= *
    (LENGTH
     (CODE
      (TRANSLATE
       (MAKE-CINFO NIL
        (CONS
           '(ROUTINEERROR . 0)
           (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                             0))
        1)
       (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
       (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
       PROC-LIST)))
    0)
```

```
UP
(DIVE 2)
(REWRITE TRANSLATE-DEF-BODY-REWRITE
 (($CINFO
   (MAKE-CINFO NIL
    (CONS
        (CONS 'ROUTINEERROR 0)
        (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                          0))
    1))
  ($T-COND-LIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))
  ($CODE2
   (LIST '(DL 0 NIL (NO-OP))
         (LIST 'POP*
               (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
         '(RET)))
  ($STMT (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST)))))
UP
(REWRITE GET-LENGTH-CAR)
(S LEMMAS)
UP X UP X
(S LEMMAS)
X
(S LEMMAS)
(DIVE 1 2 2 1)
(= *
 (LENGTH
  (CODE
   (TRANSLATE
    (MAKE-CINFO NIL
     (CONS
        '(ROUTINEERROR . 0)
        (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                          0))
     1)
    (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
    (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
    PROC-LIST)))
 0)
UP UP UP UP
(DIVE 3 1 1 5 1 1)
(REWRITE LENGTH-MG-TO-P-LOCAL-VALUES)
NX
(REWRITE MAP-DOWN-VALUES-PRESERVES-LENGTH)
```

```
UP NX
(DIVE 1)
X
(S LEMMAS)
(S-PROP P-FRAME-SIZE)
(S LEMMAS)
TOP S SPLIT PROVE PROVE
(REWRITE OK-MG-STATEP-MG-ALIST-MG-ALISTP)
(REWRITE CALLED-DEF-FORMALS-OK)
S-PROP SPLIT
(DIVE 1 1)
(REWRITE FETCH-LABEL-0-CASE-2)
UP
(REWRITE FIND-LABEL-APPEND)
UP DROP
(PROVE (ENABLE UNLABEL))
(DIVE 1)
(REWRITE FIND-LABELP-MONOTONIC-LESSP)
TOP S
(S LEMMAS)
(S LEMMAS)
(PROVE (ENABLE OK-MG-STATEMENT OK-PROC-CALL))
(DIVE 1 1)
X UP
(S LEMMAS)
UP
(PROVE (ENABLE FETCH-CALLED-DEF FETCH-DEF))
S-PROP SPLIT
(DIVE 1 1)
(REWRITE FETCH-LABEL-0-CASE-2)
UP
(REWRITE FIND-LABEL-APPEND)
TOP DROP
(PROVE (ENABLE UNLABEL))
(DIVE 1)
(REWRITE FIND-LABELP-MONOTONIC-LESSP)
UP S
(S LEMMAS)
(S LEMMAS))))
```

THEOREM: call-body-mg-vars-list-ok1
$((n \not\simeq 0)$
$\wedge \quad (\neg$ resources-inadequatep$(stmt,$
proc-list,

$$
\begin{array}{ll}
 & \quad \text{list}\,(\text{length}\,(\textit{temp-stk}),\\
 & \qquad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk}))))\\
\wedge & (\text{car}\,(\textit{stmt}) = \texttt{'proc-call-mg})\\
\wedge & \text{ok-mg-statement}\,(\textit{stmt},\ \textit{r-cond-list},\ \textit{name-alist},\ \textit{proc-list})\\
\wedge & \text{ok-mg-def-plistp}\,(\textit{proc-list})\\
\wedge & \text{ok-translation-parameters}\,(\textit{cinfo},\ \textit{t-cond-list},\ \textit{stmt},\ \textit{proc-list},\ \textit{code2})\\
\wedge & \text{ok-mg-statep}\,(\textit{mg-state},\ \textit{r-cond-list})\\
\wedge & \text{cond-subsetp}\,(\textit{r-cond-list},\ \textit{t-cond-list})\\
\wedge & (\text{code}\,(\text{translate-def-body}\,(\text{assoc}\,(\textit{subr},\ \textit{proc-list}),\ \textit{proc-list}))\\
 & \quad = \quad \text{append}\,(\text{code}\,(\text{translate}\,(\textit{cinfo},\ \textit{t-cond-list},\ \textit{stmt},\ \textit{proc-list})),\\
 & \qquad\quad \textit{code2}))\\
\wedge & \text{user-defined-procp}\,(\textit{subr},\ \textit{proc-list})\\
\wedge & \text{plistp}\,(\textit{temp-stk})\\
\wedge & \text{listp}\,(\textit{ctrl-stk})\\
\wedge & \text{mg-vars-list-ok-in-p-state}\,(\text{mg-alist}\,(\textit{mg-state}),\\
 & \qquad\qquad\qquad\qquad \text{bindings}\,(\text{top}\,(\textit{ctrl-stk})),\\
 & \qquad\qquad\qquad\qquad \textit{temp-stk})\\
\wedge & \text{no-p-aliasing}\,(\text{bindings}\,(\text{top}\,(\textit{ctrl-stk})),\ \text{mg-alist}\,(\textit{mg-state}))\\
\wedge & \text{signatures-match}\,(\text{mg-alist}\,(\textit{mg-state}),\ \textit{name-alist})\\
\wedge & \text{normal}\,(\textit{mg-state})\\
\wedge & \text{all-cars-unique}\,(\text{mg-alist}\,(\textit{mg-state}))\\
\wedge & (\neg\ \text{resource-errorp}\,(\text{mg-meaning-r}\,(\textit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad \textit{mg-state},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad n,\\
 & \qquad\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\textit{temp-stk}),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))))\\
\rightarrow & \text{mg-vars-list-ok-in-p-state}\,(\text{mg-alist}\,(\text{mg-meaning-r}\,(\text{def-body}\,(\text{fetch-called-def}\,(\textit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{mg-state}\,(\text{cc}\,(\textit{mg-state}),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{make-call-var-alist}\,(\text{mg-alist}\,(\textit{mg-state}),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{fetch-called-def}\,(\textit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-lis}\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{mg-psw}\,(\textit{mg-state})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad n-1,\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{data-length}\,(\text{def-locals}\,(\text{fetch-called-def}\,(\textit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}))\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad + \quad \text{length}\,(\textit{temp-stk}),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 2\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad + \quad \text{length}\,(\text{def-locals}\,(\text{fetch-called-def}\,(\textit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}))\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad + \quad \text{length}\,(\text{def-formals}\,(\text{fetch-called-def}\,(\textit{stmt},
\end{array}
$$

proc-list)
+ p-ctrl-stk-size (*ctrl-stk*)))),
append (map-call-locals (def-locals (fetch-called-def (*stmt*,
proc-list)),
length (*temp-stk*)),
map-call-formals (def-formals (fetch-called-def (*stmt*,
proc-list)),
call-actuals (*stmt*),
bindings (top (*ctrl-stk*)))),
append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (*stmt*,
proc-list)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)))

```
;; (pop* (data-length locals))
```


```
;; (pop* (data-length locals))
```


```
(prove-lemma call-state2-step2-effect (rewrite)
      (IMPLIES
       (AND (NOT (ZEROP N))
    (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
    (EQUAL (CAR STMT) 'PROC-CALL-MG)
    (OK-MG-STATEMENT STMT R-COND-LIST NAME-ALIST PROC-LIST)
    (OK-MG-DEF-PLISTP PROC-LIST)
    (OK-TRANSLATION-PARAMETERS CINFO T-COND-LIST STMT PROC-LIST CODE2)
    (OK-MG-STATEP MG-STATE R-COND-LIST)
    (COND-SUBSETP R-COND-LIST T-COND-LIST)
    (EQUAL (CODE (TRANSLATE-DEF-BODY (ASSOC SUBR PROC-LIST)
     PROC-LIST))
   (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
   CODE2))
    (USER-DEFINED-PROCP SUBR PROC-LIST)
    (PLISTP TEMP-STK)
    (LISTP CTRL-STK)
    (MG-VARS-LIST-OK-IN-P-STATE (MG-ALIST MG-STATE)
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
```

```
    (NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
   (MG-ALIST MG-STATE))
    (SIGNATURES-MATCH (MG-ALIST MG-STATE)
      NAME-ALIST)
    (NORMAL MG-STATE)
    (ALL-CARS-UNIQUE (MG-ALIST MG-STATE))
    (NOT (RESOURCE-ERRORP (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))))
     (equal
       (p-step
 (P-STATE (TAG 'PC
     (CONS
      (CALL-NAME STMT)
      (PLUS
       (LENGTH
(CODE
 (TRANSLATE
  (MAKE-CINFO NIL
      (CONS
       '(ROUTINEERROR . 0)
       (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
 0))
      1)
  (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST)))
       1)))
(CONS
 (P-FRAME
  (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
   (LENGTH TEMP-STK))
  (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
    (CALL-ACTUALS STMT)
    (BINDINGS (TOP CTRL-STK))))
  (TAG 'PC
       (CONS SUBR
     (ADD1 (PLUS (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
 CTRL-STK)
(MAP-DOWN-VALUES
 (MG-ALIST
```

```
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MG-STATE (CC MG-STATE)
    (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
 STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
    (MG-PSW MG-STATE))
  (SUB1 N)
  (LIST (PLUS (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH TEMP-STK))
(PLUS 2
      (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (LENGTH (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST)))
      (P-CTRL-STK-SIZE CTRL-STK)))))
(APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
 (LENGTH TEMP-STK))
(MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
  (CALL-ACTUALS STMT)
  (BINDINGS (TOP CTRL-STK))))
(APPEND
 (REVERSE
  (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
 (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
  (BINDINGS (TOP CTRL-STK))
  TEMP-STK)))
(TRANSLATE-PROC-LIST PROC-LIST)
(LIST
 (LIST 'C-C
      (MG-COND-TO-P-NAT
(CC
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MG-STATE (CC MG-STATE)
    (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
 STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
    (MG-PSW MG-STATE))
  (SUB1 N)
  (LIST
   (PLUS
    (LENGTH
     (MG-TO-P-LOCAL-VALUES
```

```
      (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (LENGTH (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
     (BINDINGS (TOP CTRL-STK))
     TEMP-STK)))
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (APPEND
       (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
(LENGTH TEMP-STK))
       (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
 (CALL-ACTUALS STMT)
 (BINDINGS (TOP CTRL-STK))))
      (TAG 'PC
   (CONS SUBR
 (ADD1
  (PLUS
   (LENGTH (CODE CINFO))
   (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
   (LENGTH (CALL-ACTUALS STMT)))))))
    CTRL-STK)))))
(MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
(MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
'RUN))
       (P-STATE
 (TAG 'PC
      (CONS
       (CALL-NAME STMT)
       (PLUS
(LENGTH
 (CODE
  (TRANSLATE
   (MAKE-CINFO NIL
       (CONS
'(ROUTINEERROR . 0)
(MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  0))
       1)
   (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
   (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
   PROC-LIST)))
2)))
 (CONS
```

```
  (P-FRAME
   (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
    (LENGTH TEMP-STK))
   (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
     (CALL-ACTUALS STMT)
     (BINDINGS (TOP CTRL-STK))))
   (TAG 'PC
(CONS SUBR
      (ADD1 (PLUS (LENGTH (CODE CINFO))
  (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
  (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
  (LENGTH (CALL-ACTUALS STMT)))))))
  CTRL-STK)
 (POPN (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
  (MAP-DOWN-VALUES
   (MG-ALIST
    (MG-MEANING-R
     (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
     PROC-LIST
     (MG-STATE (CC MG-STATE)
       (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
    STMT
    (FETCH-CALLED-DEF STMT PROC-LIST))
       (MG-PSW MG-STATE))
     (SUB1 N)
     (LIST (PLUS (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH TEMP-STK))
   (PLUS 2
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (P-CTRL-STK-SIZE CTRL-STK)))))
   (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
    (LENGTH TEMP-STK))
   (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
     (CALL-ACTUALS STMT)
     (BINDINGS (TOP CTRL-STK))))
   (APPEND
    (REVERSE
     (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
     (BINDINGS (TOP CTRL-STK))
     TEMP-STK))))
 (TRANSLATE-PROC-LIST PROC-LIST)
 (LIST
```

```
  (LIST 'C-C
(MG-COND-TO-P-NAT
 (CC
  (MG-MEANING-R
   (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
   PROC-LIST
   (MG-STATE (CC MG-STATE)
     (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
  STMT
  (FETCH-CALLED-DEF STMT PROC-LIST))
     (MG-PSW MG-STATE))
   (SUB1 N)
   (LIST
    (PLUS
     (LENGTH
      (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (LENGTH (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
    (P-CTRL-STK-SIZE
     (CONS
      (P-FRAME
       (APPEND
(MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
 (LENGTH TEMP-STK))
(MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
  (CALL-ACTUALS STMT)
  (BINDINGS (TOP CTRL-STK))))
       (TAG 'PC
    (CONS SUBR
  (ADD1
   (PLUS (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
 (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
        (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
 'RUN)))
 ((INSTRUCTIONS PROMOTE
   (DIVE 1)
   X
   (S LEMMAS)
   (DIVE 1 1 2)
```

```
(REWRITE TRANSLATE-DEF-BODY-REWRITE
 (($CINFO
   (MAKE-CINFO NIL
    (CONS
        (CONS 'ROUTINEERROR 0)
        (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                          0))
    1))
  ($T-COND-LIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))
  ($CODE2
   (LIST '(DL 0 NIL (NO-OP))
         (LIST 'POP*
               (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
         '(RET)))
  ($STMT (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST)))))
UP
(REWRITE GET-LENGTH-PLUS)
(S LEMMAS)
UP X UP X
(S LEMMAS)
(DIVE 1)
X UP S
(S LEMMAS)
UP S-PROP
(DIVE 1)
(DIVE 1)
(REWRITE MAP-DOWN-VALUES-PRESERVES-LENGTH)
(S LEMMAS)
(DIVE 1)
(REWRITE LENGTH-MG-TO-P-LOCAL-VALUES)
UP UP
(= * F)
UP S
(REWRITE CALLED-DEF-FORMALS-OK)
(REWRITE CALL-BODY-MG-VARS-LIST-OK1)
(DIVE 1 1)
(REWRITE MG-MEANING-EQUIVALENCE)
UP UP
(REWRITE MG-MEANING-PRESERVES-MG-ALISTP
        (($R-COND-LIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))
         ($NAME-ALIST (MAKE-NAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
(DIVE 1)
(= *
   (MAKE-CALL-ENVIRONMENT MG-STATE STMT
```

```
                          (FETCH-CALLED-DEF STMT PROC-LIST))
   0)
UP
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
S PROVE
(REWRITE CALL-EXACT-TIME-HYPS1)
(DIVE 1)
S UP
(REWRITE CALL-SIGNATURES-MATCH2)
(DIVE 1)
(REWRITE MORE-RESOURCES-PRESERVES-NOT-RESOURCE-ERRORP
 (($T-SIZE1
   (LENGTH
    (APPEND
     (REVERSE
      (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                      (BINDINGS (TOP CTRL-STK))
                      TEMP-STK))))
  ($C-SIZE1
   (P-CTRL-STK-SIZE
    (CONS
     (P-FRAME
      (MAKE-FRAME-ALIST (FETCH-CALLED-DEF STMT PROC-LIST)
                        STMT CTRL-STK TEMP-STK)
      (TAG 'PC
       (CONS SUBR
        (ADD1
          (PLUS (LENGTH (CODE CINFO))
                (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
UP S
(S LEMMAS)
(DIVE 1 1 1)
(REWRITE LENGTH-MG-TO-P-LOCAL-VALUES)
NX
(REWRITE MAP-DOWN-VALUES-PRESERVES-LENGTH)
TOP DROP PROVE
(REWRITE OK-MG-STATEP-MG-ALIST-MG-ALISTP)
(REWRITE CALLED-DEF-FORMALS-OK)
(DIVE 1 1)
X
```

```
   (S LEMMAS)
   (S-PROP P-FRAME-SIZE)
   (S LEMMAS)
   UP TOP DROP PROVE
   (DIVE 1 1 3)
   (= *
      (MAKE-CALL-ENVIRONMENT MG-STATE STMT
                             (FETCH-CALLED-DEF STMT PROC-LIST))
      0)
   TOP
   (DIVE 1)
   (REWRITE PROC-CALL-DOESNT-HALT)
   TOP S
   (S-PROP MAKE-CALL-ENVIRONMENT)
   (PROVE (ENABLE OK-MG-STATEMENT OK-PROC-CALL))
   (REWRITE PROC-CALL-EXACT-TIME-HYPS)
   (DIVE 1 1)
   X TOP
   (PROVE (ENABLE FETCH-CALLED-DEF FETCH-DEF)))))

;; (ret)




;; (ret)

(prove-lemma call-state2-step3-effect (rewrite)
      (IMPLIES
       (AND (NOT (ZEROP N))
    (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
    (EQUAL (CAR STMT) 'PROC-CALL-MG)
    (OK-MG-STATEMENT STMT R-COND-LIST NAME-ALIST PROC-LIST)
    (OK-MG-DEF-PLISTP PROC-LIST)
    (OK-TRANSLATION-PARAMETERS CINFO T-COND-LIST STMT PROC-LIST CODE2)
    (OK-MG-STATEP MG-STATE R-COND-LIST)
    (COND-SUBSETP R-COND-LIST T-COND-LIST)
    (EQUAL (CODE (TRANSLATE-DEF-BODY (ASSOC SUBR PROC-LIST)
     PROC-LIST))
   (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
   CODE2))
    (USER-DEFINED-PROCP SUBR PROC-LIST)
```

```
    (PLISTP TEMP-STK)
    (LISTP CTRL-STK)
    (MG-VARS-LIST-OK-IN-P-STATE (MG-ALIST MG-STATE)
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
    (NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
   (MG-ALIST MG-STATE))
    (SIGNATURES-MATCH (MG-ALIST MG-STATE)
      NAME-ALIST)
    (NORMAL MG-STATE)
    (ALL-CARS-UNIQUE (MG-ALIST MG-STATE))
    (NOT (RESOURCE-ERRORP (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))))
     (equal
       (p-step
        (P-STATE
 (TAG 'PC
      (CONS
       (CALL-NAME STMT)
       (PLUS
(LENGTH
 (CODE
  (TRANSLATE
   (MAKE-CINFO NIL
       (CONS
'(ROUTINEERROR . 0)
(MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
  0))
       1)
   (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
   (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
   PROC-LIST)))
2)))
 (CONS
  (P-FRAME
   (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
    (LENGTH TEMP-STK))
   (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
     (CALL-ACTUALS STMT)
     (BINDINGS (TOP CTRL-STK))))
   (TAG 'PC
(CONS SUBR
      (ADD1 (PLUS (LENGTH (CODE CINFO))
```

```
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
 CTRL-STK)
(POPN (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (MAP-DOWN-VALUES
  (MG-ALIST
   (MG-MEANING-R
    (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
    PROC-LIST
    (MG-STATE (CC MG-STATE)
      (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
   STMT
   (FETCH-CALLED-DEF STMT PROC-LIST))
      (MG-PSW MG-STATE))
    (SUB1 N)
    (LIST (PLUS (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
(LENGTH TEMP-STK))
  (PLUS 2
(LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
(LENGTH (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST)))
(P-CTRL-STK-SIZE CTRL-STK)))))
  (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
   (LENGTH TEMP-STK))
  (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
    (CALL-ACTUALS STMT)
    (BINDINGS (TOP CTRL-STK))))
  (APPEND
   (REVERSE
    (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
   (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
    (BINDINGS (TOP CTRL-STK))
    TEMP-STK))))
(TRANSLATE-PROC-LIST PROC-LIST)
(LIST
 (LIST 'C-C
(MG-COND-TO-P-NAT
 (CC
  (MG-MEANING-R
   (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
   PROC-LIST
   (MG-STATE (CC MG-STATE)
     (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
 STMT
```

```
  (FETCH-CALLED-DEF STMT PROC-LIST))
     (MG-PSW MG-STATE))
   (SUB1 N)
   (LIST
    (PLUS
     (LENGTH
      (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
     (LENGTH (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
      (BINDINGS (TOP CTRL-STK))
      TEMP-STK)))
    (P-CTRL-STK-SIZE
     (CONS
      (P-FRAME
       (APPEND
(MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
 (LENGTH TEMP-STK))
(MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
  (CALL-ACTUALS STMT)
  (BINDINGS (TOP CTRL-STK))))
      (TAG 'PC
    (CONS SUBR
  (ADD1
   (PLUS (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
 (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
 (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
 'RUN))
     (P-STATE
         (TAG 'PC
       (CONS SUBR
    (ADD1 (PLUS (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT))))))
        ctrl-stk
 (POPN (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
  (MAP-DOWN-VALUES
   (MG-ALIST
    (MG-MEANING-R
     (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
     PROC-LIST
```

```
    (MG-STATE (CC MG-STATE)
      (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
   STMT
   (FETCH-CALLED-DEF STMT PROC-LIST))
      (MG-PSW MG-STATE))
    (SUB1 N)
    (LIST (PLUS (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
(LENGTH TEMP-STK))
  (PLUS 2
(LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
(LENGTH (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST)))
(P-CTRL-STK-SIZE CTRL-STK)))))
  (APPEND (MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
   (LENGTH TEMP-STK))
  (MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
    (CALL-ACTUALS STMT)
    (BINDINGS (TOP CTRL-STK))))
  (APPEND
   (REVERSE
    (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
   (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
    (BINDINGS (TOP CTRL-STK))
    TEMP-STK))))
(TRANSLATE-PROC-LIST PROC-LIST)
(LIST
 (LIST 'C-C
(MG-COND-TO-P-NAT
(CC
 (MG-MEANING-R
  (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST))
  PROC-LIST
  (MG-STATE (CC MG-STATE)
    (MAKE-CALL-VAR-ALIST (MG-ALIST MG-STATE)
 STMT
 (FETCH-CALLED-DEF STMT PROC-LIST))
    (MG-PSW MG-STATE))
  (SUB1 N)
  (LIST
   (PLUS
    (LENGTH
     (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
    (LENGTH (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
     (BINDINGS (TOP CTRL-STK))
     TEMP-STK)))
```

```
    (P-CTRL-STK-SIZE
     (CONS
      (P-FRAME
       (APPEND
(MAP-CALL-LOCALS (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))
 (LENGTH TEMP-STK))
(MAP-CALL-FORMALS (DEF-FORMALS (FETCH-CALLED-DEF STMT PROC-LIST))
  (CALL-ACTUALS STMT)
  (BINDINGS (TOP CTRL-STK))))
       (TAG 'PC
    (CONS SUBR
  (ADD1
   (PLUS (LENGTH (CODE CINFO))
 (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
 (LENGTH (CALL-ACTUALS STMT)))))))
     CTRL-STK)))))
 (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))))
 (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
 'RUN)))
 ((INSTRUCTIONS PROMOTE
   (DIVE 1)
   X
   (S LEMMAS)
   (DIVE 1 1 2)
   (REWRITE TRANSLATE-DEF-BODY-REWRITE
    (($CINFO
      (MAKE-CINFO NIL
       (CONS
           (CONS 'ROUTINEERROR 0)
           (MAKE-LABEL-ALIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST))
                             0))
       1))
     ($T-COND-LIST (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))
     ($CODE2
      (LIST
       '(DL 0 NIL (NO-OP))
       (LIST 'POP* (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
       '(RET)))
     ($STMT (DEF-BODY (FETCH-CALLED-DEF STMT PROC-LIST)))))
   UP
   (REWRITE GET-LENGTH-PLUS)
   (S LEMMAS)
   UP X UP X
```

```
(S LEMMAS)
X
(S LEMMAS)
UP prove
(PROVE (ENABLE OK-MG-STATEMENT OK-PROC-CALL))
(DIVE 1 1)
X TOP
(PROVE (ENABLE FETCH-CALLED-DEF FETCH-DEF)))))
```

EVENT: Make the library "c-proc-call1".

Index