EVENT: Start with the library "c-proc-call1".

```
;; We now devote some effort to showing that the temp-stk at this
;; point (following the return) is the final temp-stk.  This is
;; perhaps the most difficult section of the proof.
```

DEFINITION:
popn-deposit-induction-hint (*temp-stk*, *n*)
= **if** length (*temp-stk*) ≃ 0 **then t**
elseif *n* ≃ 0 **then t**
else popn-deposit-induction-hint (cdr (*temp-stk*), *n* − 1) **endif**

THEOREM: popn-rput
((untag (*x*) ≮ (length (*temp-stk*) − *n*)) ∧ ok-temp-stk-index (*x*, *temp-stk*))
→ (popn (*n*, rput (*value*, untag (*x*), *temp-stk*)) = popn (*n*, *temp-stk*))

EVENT: Enable deposit-temp.

THEOREM: popn-deposit-array-value
((untag (*x*) ≮ (length (*temp-stk*) − *n*))
∧ ok-temp-stk-array-index (*x*, *temp-stk*, length (*array-value*)))
→ (popn (*n*, deposit-array-value (*array-value*, *x*, *temp-stk*))
= popn (*n*, *temp-stk*))

THEOREM: popn-locals1
(all-pointers-bigger (collect-pointers (*bindings*, *alist*),
length (*temp-stk1*) − *n*)
∧ mg-alistp (*alist*)
∧ mg-vars-list-ok-in-p-state (*alist*, *bindings*, *temp-stk1*))
→ (popn (*n*, map-down-values (*alist*, *bindings*, *temp-stk1*))
= popn (*n*, *temp-stk1*))

THEOREM: popn-locals
(all-pointers-bigger (collect-pointers (*bindings*, *alist*), length (*temp-stk*))
∧ mg-alistp (*alist*)
∧ (*n* = length (*lst*))
∧ mg-vars-list-ok-in-p-state (*alist*, *bindings*, append (*lst*, *temp-stk*)))
→ (popn (*n*, map-down-values (*alist*, *bindings*, append (*lst*, *temp-stk*)))
= *temp-stk*)

DEFINITION:
drop-formals-induction-hint (*alist*, *locals*, *temp-stk*, *n*)
= **if** *alist* ≃ **nil then t**

elseif caar (*alist*) ∈ listcars (*locals*)
then drop-formals-induction-hint (cdr (*alist*),
locals,
deposit-alist-value (car (*alist*),
map-call-locals (*locals*,
n),
temp-stk),
n)
else drop-formals-induction-hint (cdr (*alist*), *locals*, *temp-stk*, *n*) **endif**

THEOREM: map-down-values-drop-formals-restriction
(all-cars-unique (*alist*)
∧ mg-vars-list-ok-in-p-state (restrict (*alist*, listcars (*locals*)),
map-call-locals (*locals*, *n*),
temp-stk))
→ (map-down-values (restrict (*alist*, listcars (*locals*)),
append (map-call-locals (*locals*, *n*), *lst*),
temp-stk)
= map-down-values (restrict (*alist*, listcars (*locals*)),
map-call-locals (*locals*, *n*),
temp-stk))

DEFINITION:
drop-locals-induction-hint (*alist*, *formals*, *temp-stk*, *actuals*, *bindings*)
= **if** *alist* ≃ **nil then t**
elseif caar (*alist*) ∈ listcars (*formals*)
then drop-locals-induction-hint (cdr (*alist*),
formals,
deposit-alist-value (car (*alist*),
map-call-formals (*formals*,
actuals,
bindings),
temp-stk),
actuals,
bindings)
else drop-locals-induction-hint (cdr (*alist*),
formals,
temp-stk,
actuals,
bindings) **endif**

THEOREM: map-down-drop-locals-restriction
(all-cars-unique (*alist*)
∧ no-duplicates (append (listcars (*formals*), listcars (*locals*))))
→ (map-down-values (restrict (*alist*, listcars (*formals*)),

append (map-call-locals (*locals*, *n*),
map-call-formals (*formals*, *actuals*, *bindings*)),
temp-stk)
= map-down-values (restrict (*alist*, listcars (*formals*)),
map-call-formals (*formals*, *actuals*, *bindings*),
temp-stk))

THEOREM: restrict-cons
(*x* ≠ *y*)
→ (assoc (*x*, restrict (*lst*, cons (*y*, *z*))) = assoc (*x*, restrict (*lst*, *z*)))

THEOREM: copy-out-params-restriction-cons
(*x* ∉ listcars (*lst1*))
→ (copy-out-params (*lst1*, *lst2*, restrict (*new-alist*, cons (*x*, *z*)), *old-alist*)
= copy-out-params (*lst1*, *lst2*, restrict (*new-alist*, *z*), *old-alist*))

THEOREM: copy-out-params-restriction
all-cars-unique (*formals*)
→ (copy-out-params (*formals*, *actuals*, *new-alist*, *old-alist*)
= copy-out-params (*formals*,
actuals,
restrict (*new-alist*, listcars (*formals*)),
old-alist))

EVENT: Disable deposit-temp.

THEOREM: deposit-temp-deposit-array-value-commute6
(mg-vars-list-ok-in-p-state (*lst*, *bindings*, *temp-stk*)
∧ no-p-aliasing (*bindings*, *lst*)
∧ mg-alistp (*lst*)
∧ all-cars-unique (*lst*)
∧ (*x* ∈ *lst*)
∧ (*y* ∈ *lst*)
∧ (car (*x*) ≠ car (*y*))
∧ (¬ simple-mg-type-refp (cadr (*y*)))
∧ (length (*value*) = array-length (cadr (*y*))))
→ (deposit-temp (*z*,
cdr (assoc (car (*x*), *bindings*)),
deposit-array-value (*value*,
cdr (assoc (car (*y*), *bindings*)),
temp-stk))
= deposit-array-value (*value*,
cdr (assoc (car (*y*), *bindings*)),
deposit-temp (*z*,

cdr (assoc (car (*x*), *bindings*)),
temp-stk)))

THEOREM: deposit-array-value-deposit-alist-value-commute2
(mg-vars-list-ok-in-p-state (*lst*, *bindings*, *temp-stk*)
∧ no-p-aliasing (*bindings*, *lst*)
∧ mg-alistp (*lst*)
∧ all-cars-unique (*lst*)
∧ (*x* ∈ *lst*)
∧ (*y* ∈ *lst*)
∧ (¬ simple-mg-type-refp (cadr (*x*)))
∧ (length (*value*) = array-length (cadr (*x*)))
∧ (car (*x*) ≠ car (*y*)))
→ (deposit-array-value (*value*,
cdr (assoc (car (*x*), *bindings*)),
deposit-alist-value (*y*, *bindings*, *temp-stk*))
= deposit-alist-value (*y*,
bindings,
deposit-array-value (*value*,
cdr (assoc (car (*x*),
bindings)),
temp-stk)))

THEOREM: deposit-array-value-doesnt-affect-map-down-values
(mg-alistp (cons (*x*, *mg-vars*))
∧ all-cars-unique (cons (*x*, *mg-vars*))
∧ no-p-aliasing (*bindings*, cons (*x*, *mg-vars*))
∧ mg-vars-list-ok-in-p-state (cons (*x*, *mg-vars*), *bindings*, *temp-stk*)
∧ (¬ simple-mg-type-refp (cadr (*x*)))
∧ (length (*value*) = array-length (cadr (*x*))))
→ (map-down-values (*mg-vars*,
bindings,
deposit-array-value (*value*,
cdr (assoc (car (*x*), *bindings*)),
temp-stk))
= deposit-array-value (*value*,
cdr (assoc (car (*x*), *bindings*)),
map-down-values (*mg-vars*, *bindings*, *temp-stk*)))

THEOREM: extra-binding-doesnt-affect-copy-out-params
(car (*x*) ∉ listcars (*formals*))
→ (copy-out-params (*formals*, *actuals*, cons (*x*, *new-alist*), *old-alist*)
= copy-out-params (*formals*, *actuals*, *new-alist*, *old-alist*))

DEFINITION:

map-down-copy-out-params-induction-hint (*formals*, *old-alist*, *actuals*, *new-alist*)
= **if** *formals* ≃ **nil then t**
else map-down-copy-out-params-induction-hint (cdr (*formals*),
set-alist-value (car (*actuals*),
caddr (assoc (caar (*formals*),
restrict (*new-alist*,
listcars (*formals*))))
old-alist),
cdr (*actuals*),
cdr (*new-alist*)) **endif**

THEOREM: map-down-copy-facts
(listp (*formals*) ∧ (listcars (*alist*) = append (listcars (*formals*), *locals*)))
→ (listp (*alist*)
∧ (caar (*alist*) = caar (*formals*))
∧ (caar (*alist*) ∈ listcars (*formals*)))

EVENT: Disable map-down-copy-facts.

THEOREM: map-down-copy-facts2
(listp (*formals*)
∧ (listcars (*alist*) = append (listcars (*formals*), *locals*))
∧ formal-types-preserved (*formals*, restrict (*alist*, listcars (*formals*))))
→ (cadar (*formals*) = cadar (*alist*))

EVENT: Disable map-down-copy-facts2.

THEOREM: map-down-copy-facts3
(listp (*formals*)
∧ (listcars (*alist*) = append (listcars (*formals*), *locals*))
∧ all-cars-unique (*alist*))
→ (restrict (*alist*, listcars (*formals*))
= cons (car (*alist*), restrict (cdr (*alist*), listcars (cdr (*formals*)))))

EVENT: Disable map-down-copy-facts3.

```
;; This one gives the looping problem when interrupted during the base case.
;; This occurs only when I have the (maintain-rewrite-path)
```

THEOREM: map-down-copy-out-params-relation-new
(ok-actual-params-list (*actuals*, *old-alist*)
∧ data-param-lists-match (*actuals*, *formals*, *old-alist*)

$\wedge$ ok-mg-formal-data-params-plistp (*formals*)
$\wedge$ (listcars (*new-alist*) = append (listcars (*formals*), *locals*))
$\wedge$ all-cars-unique (*old-alist*)
$\wedge$ no-p-aliasing (*bindings*, *old-alist*)
$\wedge$ mg-vars-list-ok-in-p-state (*old-alist*, *bindings*, *temp-stk*)
$\wedge$ mg-vars-list-ok-in-p-state (restrict (*new-alist*, listcars (*formals*)),
map-call-formals (*formals*, *actuals*, *bindings*),
map-down-values (*old-alist*,
bindings,
temp-stk))
$\wedge$ no-duplicates (*actuals*)
$\wedge$ all-cars-unique (*formals*)
$\wedge$ all-cars-unique (*new-alist*)
$\wedge$ mg-alistp (*old-alist*)
$\wedge$ mg-alistp (*new-alist*)
$\wedge$ formal-types-preserved (*formals*, restrict (*new-alist*, listcars (*formals*))))
$\rightarrow$ (map-down-values (restrict (*new-alist*, listcars (*formals*)),
map-call-formals (*formals*, *actuals*, *bindings*),
map-down-values (*old-alist*, *bindings*, *temp-stk*))
= map-down-values (copy-out-params (*formals*,
actuals,
restrict (*new-alist*,
listcars (*formals*)),
old-alist),
bindings,
temp-stk))

THEOREM: formals-meaning-signature
((car (*stmt*) = `'proc-call-mg`)
$\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ ok-mg-def-plistp (*proc-list*))
$\rightarrow$ signatures-match (make-call-param-alist (def-formals (fetch-called-def (*stmt*,
proc-list)),
call-actuals (*stmt*),
mg-alist (*mg-state*)),
restrict (mg-alist (mg-meaning (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*)),

$n - 1)),$
listcars (def-formals (fetch-called-def (*stmt*,
proc-list)))))

EVENT: Disable formals-meaning-signature.

THEOREM: locals-meaning-signature
((car (*stmt*) = '`proc-call-mg`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*))
→ signatures-match (def-locals (fetch-called-def (*stmt*, *proc-list*)),
restrict (mg-alist (mg-meaning (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*)),
$n - 1)),$
listcars (def-locals (fetch-called-def (*stmt*,
proc-list)))))

THEOREM: param-alist-mg-vars-ok0
(ok-actual-params-list (*actuals*, *mg-vars*)
∧ data-param-lists-match (*actuals*, *formals*, *mg-vars*)
∧ ok-mg-formal-data-params-plistp (*formals*)
∧ mg-alistp (*mg-vars*)
∧ all-cars-unique (*formals*)
∧ mg-vars-list-ok-in-p-state (*mg-vars*, *bindings*, *temp-stk*))
→ mg-vars-list-ok-in-p-state (make-call-param-alist (*formals*,
actuals,
mg-vars),
map-call-formals (*formals*, *actuals*, *bindings*),
temp-stk)

THEOREM: param-alist-mg-vars-ok
((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*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),

temp-stk)
∧ signatures-match (mg-alist (*mg-state*), *name-alist*))
→ mg-vars-list-ok-in-p-state (make-call-param-alist (def-formals (fetch-called-def (*stmt*, *proc-list*)),
call-actuals (*stmt*),
mg-alist (*mg-state*)),
map-call-formals (def-formals (fetch-called-def (*stmt*, *proc-list*)),
call-actuals (*stmt*),
bindings (top (*ctrl-stk*))),
temp-stk)

EVENT: Disable param-alist-mg-vars-ok.

THEOREM: locals-alist-mg-vars-ok0
(all-cars-unique (*locals*) ∧ ok-mg-local-data-plistp (*locals*))
→ mg-vars-list-ok-in-p-state (*locals*,
map-call-locals (*locals*, length (*temp-stk*)),
append (reverse (mg-to-p-local-values (*locals*)),
temp-stk))

THEOREM: locals-alist-mg-vars-ok
((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*))
→ mg-vars-list-ok-in-p-state (def-locals (fetch-called-def (*stmt*, *proc-list*)),
map-call-locals (def-locals (fetch-called-def (*stmt*, *proc-list*)),
length (*temp-stk*)),
append (reverse (mg-to-p-local-values (def-locals (fetch-called-def (*stmt*, *proc-list*))))),
temp-stk))

THEOREM: no-p-aliasing-in-call-alists-new
((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*)
∧ plistp (*temp-stk*)
∧ listp (*ctrl-stk*)
∧ 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*)
$\wedge$ all-cars-unique (mg-alist (*mg-state*)))
$\rightarrow$ no-p-aliasing (append (map-call-formals (def-formals (fetch-called-def (*stmt*, *proc-list*)),
call-actuals (*stmt*),
bindings (top (*ctrl-stk*))),
map-call-locals (def-locals (fetch-called-def (*stmt*, *proc-list*)),
length (*temp-stk*))),
append (restrict (mg-alist (mg-meaning (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
mg-psw (*mg-state*)),
$n - 1$)),
listcars (def-formals (fetch-called-def (*stmt*, *proc-list*)))),
restrict (mg-alist (mg-meaning (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
mg-psw (*mg-state*)),
$n - 1$)),
listcars (def-locals (fetch-called-def (*stmt*, *proc-list*))))))

```
;; The proof of this is incredible.  This is a prime candidate for
;; cleaning up the proof.

;; At this point, we have exitted from the call.
```

THEOREM: ret-temp-stk-equals-final-temp-stk
(($n \not\simeq 0$)
$\wedge$ ($\neg$ 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))))))
→   (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)),
                                                   n − 1,
                                                   list (data-length (def-locals (fetch-called-def (stmt,
                                                                                                    proc-list))
                                                         +   length (temp-stk),
                                                         2
                                                         +   length (def-locals (fetch-called-def (stmt,
```

$$
\begin{aligned}
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}))) \\
&\qquad\qquad\qquad\qquad + \quad \text{length}\,(\text{def-formals}\,(\text{fetch-called-def}\,(\textit{stmt}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}) \\
&\qquad\qquad\qquad\qquad + \quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))), \\
&\quad \text{append}\,(\text{map-call-locals}\,(\text{def-locals}\,(\text{fetch-called-def}\,(\textit{stmt}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list})), \\
&\qquad\qquad\qquad\qquad \text{length}\,(\textit{temp-stk})), \\
&\qquad\quad \text{map-call-formals}\,(\text{def-formals}\,(\text{fetch-called-def}\,(\textit{stmt}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list})), \\
&\qquad\qquad\qquad\qquad \text{call-actuals}\,(\textit{stmt}), \\
&\qquad\qquad\qquad\qquad \text{bindings}\,(\text{top}\,(\textit{ctrl-stk})))), \\
&\quad \text{append}\,(\text{reverse}\,(\text{mg-to-p-local-values}\,(\text{def-locals}\,(\text{fetch-called-def}\,(\textit{stmt}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list})))), \\
&\qquad\quad \text{map-down-values}\,(\text{mg-alist}\,(\textit{mg-state}), \\
&\qquad\qquad\qquad\qquad \text{bindings}\,(\text{top}\,(\textit{ctrl-stk})), \\
&\qquad\qquad\qquad\qquad \textit{temp-stk})))) \\
= \quad & \text{map-down-values}\,(\text{mg-alist}\,(\text{mg-meaning}\,(\textit{stmt},\ \textit{proc-list},\ \textit{mg-state},\ n)), \\
&\qquad\qquad \text{bindings}\,(\text{top}\,(\textit{ctrl-stk})), \\
&\qquad\qquad \textit{temp-stk}))
\end{aligned}
$$

EVENT: Disable proc-call-meaning-2.

```
;; (push-global c-c)
```

```
;; (push-global c-c)

(prove-lemma call-state2-step4-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 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
  (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)) 2)))
       CTRL-STK
       (PUSH
(MG-COND-TO-P-NAT
 (CC
  (MG-MEANING
   (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)))
 (MAKE-COND-LIST (FETCH-CALLED-DEF STMT PROC-LIST)))
(MAP-DOWN-VALUES (MG-ALIST (MG-MEANING STMT PROC-LIST MG-STATE N))
 (BINDINGS (TOP CTRL-STK))
 TEMP-STK))
       (TRANSLATE-PROC-LIST PROC-LIST)
       (LIST
(LIST 'C-C
      (MG-COND-TO-P-NAT
       (CC
(MG-MEANING
 (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)))
       (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 1 3)
   (REWRITE RET-TEMP-STK-EQUALS-FINAL-TEMP-STK)
   UP
   (DIVE 5 1 2 1 1 1)
   (REWRITE MG-MEANING-EQUIVALENCE2
    (($T-SIZE1
      (LENGTH
       (APPEND
        (REVERSE
         (MG-TO-P-LOCAL-VALUES (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
        (MAP-DOWN-VALUES (MG-ALIST MG-STATE)
                         (BINDINGS (TOP CTRL-STK))
                         TEMP-STK))))
     ($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)))))
   TOP
   (DIVE 1)
   X
   (S LEMMAS)
   (DIVE 1 1 2)
   (REWRITE TRANSLATE-DEF-BODY-REWRITE)
   (DIVE 1 1)
   (REWRITE CALL-TRANSLATION-2)
   UP UP UP S
   (DIVE 1)
```

```
(= (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))
         1))
UP
(S LEMMAS)
(REWRITE GET-LENGTH-PLUS)
(REWRITE GET-LENGTH-PLUS)
(REWRITE GET-LENGTH-PLUS)
(REWRITE GET-LENGTH-PLUS)
(S LEMMAS)
UP X UP X
(S LEMMAS)
(DIVE 1)
X
(DIVE 1)
(REWRITE MAP-DOWN-VALUES-PRESERVES-LENGTH)
UP
(REWRITE RESOURCES-ADEQUATE-TEMP-STK-NOT-MAX)
UP
(S LEMMAS)
X
(S LEMMAS)
TOP S DROP
(PROVE (ENABLE TAG))
(REWRITE SIGNATURES-MATCH-PRESERVES-MG-VARS-LIST-OK
         (($X (MG-ALIST MG-STATE))))
(REWRITE MG-MEANING-PRESERVES-SIGNATURES-MATCH)
(REWRITE OK-MG-STATEP-ALIST-PLISTP)
(REWRITE MG-MEANING-PRESERVES-MG-ALISTP)
(S LEMMAS)
(S LEMMAS)
(DIVE 2)
(REWRITE LENGTH-PUSH-LOCALS-VALUES-CODE)
TOP S
(REWRITE CALLED-DEF-FORMALS-OK)
(USE-LEMMA PROC-CALL-DOESNT-HALT)
(DEMOTE 19)
(DIVE 1 1)
S TOP
(S-PROP MAKE-CALL-ENVIRONMENT)
S
(S LEMMAS)
```

```
   SPLIT PROVE
   (S LEMMAS)
   PROVE PROVE)))

;; (jump-case ....)
```

THEOREM: call-state2-step5-effect

$$
\begin{aligned}
&((n \not\simeq \texttt{0})\\
&\wedge\ (\neg\ \text{resources-inadequatep}\,(\mathit{stmt},\\
&\qquad\qquad \mathit{proc\text{-}list},\\
&\qquad\qquad \text{list}\,(\text{length}\,(\mathit{temp\text{-}stk}),\\
&\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\mathit{ctrl\text{-}stk}))))\\
&\wedge\ (\text{car}\,(\mathit{stmt}) = \texttt{'proc-call-mg})\\
&\wedge\ \text{ok-mg-statement}\,(\mathit{stmt},\ \mathit{r\text{-}cond\text{-}list},\ \mathit{name\text{-}alist},\ \mathit{proc\text{-}list})\\
&\wedge\ \text{ok-mg-def-plistp}\,(\mathit{proc\text{-}list})\\
&\wedge\ \text{ok-translation-parameters}\,(\mathit{cinfo},\ \mathit{t\text{-}cond\text{-}list},\ \mathit{stmt},\ \mathit{proc\text{-}list},\ \mathit{code2})\\
&\wedge\ \text{ok-mg-statep}\,(\mathit{mg\text{-}state},\ \mathit{r\text{-}cond\text{-}list})\\
&\wedge\ \text{cond-subsetp}\,(\mathit{r\text{-}cond\text{-}list},\ \mathit{t\text{-}cond\text{-}list})\\
&\wedge\ (\text{code}\,(\text{translate-def-body}\,(\text{assoc}\,(\mathit{subr},\ \mathit{proc\text{-}list}),\ \mathit{proc\text{-}list}))\\
&\qquad =\ \ \text{append}\,(\text{code}\,(\text{translate}\,(\mathit{cinfo},\ \mathit{t\text{-}cond\text{-}list},\ \mathit{stmt},\ \mathit{proc\text{-}list})),\\
&\qquad\qquad\quad \mathit{code2}))\\
&\wedge\ \text{user-defined-procp}\,(\mathit{subr},\ \mathit{proc\text{-}list})\\
&\wedge\ \text{plistp}\,(\mathit{temp\text{-}stk})\\
&\wedge\ \text{listp}\,(\mathit{ctrl\text{-}stk})\\
&\wedge\ \text{mg-vars-list-ok-in-p-state}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}),\\
&\qquad\qquad \text{bindings}\,(\text{top}\,(\mathit{ctrl\text{-}stk})),\\
&\qquad\qquad \mathit{temp\text{-}stk})\\
&\wedge\ \text{no-p-aliasing}\,(\text{bindings}\,(\text{top}\,(\mathit{ctrl\text{-}stk})),\ \text{mg-alist}\,(\mathit{mg\text{-}state}))\\
&\wedge\ \text{signatures-match}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}),\ \mathit{name\text{-}alist})\\
&\wedge\ \text{normal}\,(\mathit{mg\text{-}state})\\
&\wedge\ \text{all-cars-unique}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}))\\
&\wedge\ (\neg\ \text{resource-errorp}\,(\text{mg-meaning-r}\,(\mathit{stmt},\\
&\qquad\qquad\qquad\qquad \mathit{proc\text{-}list},\\
&\qquad\qquad\qquad\qquad \mathit{mg\text{-}state},\\
&\qquad\qquad\qquad\qquad n,\\
&\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\mathit{temp\text{-}stk}),\\
&\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\mathit{ctrl\text{-}stk})))))\\
&\rightarrow\ (\text{p-step}\,(\text{p-state}\,(\text{tag}\,(\texttt{'pc},\\
&\qquad\qquad\qquad\qquad \text{cons}\,(\mathit{subr},\\
&\qquad\qquad\qquad\qquad\quad \text{length}\,(\text{code}\,(\mathit{cinfo}))\\
&\qquad\qquad\qquad\qquad\quad +\ \ \text{data-length}\,(\text{def-locals}\,(\text{fetch-called-def}\,(\mathit{stmt},\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{proc\text{-}list})))\\
&\qquad\qquad\qquad\qquad\quad +\ \ \text{length}\,(\text{def-locals}\,(\text{fetch-called-def}\,(\mathit{stmt},
\end{aligned}
$$

```
                                                                  proc-list)))
                  +   length (call-actuals (stmt))
                  +   2)),
            ctrl-stk,
            push (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                proc-list)),
                                                             proc-list,
                                                             mg-state (cc (mg-state),
                                                                          make-call-var-alist (mg-alist (mg-stat
                                                                                                         stmt,
                                                                                                         fetch-called-def (
                                                                                                                           
                                                                          mg-psw (mg-state)),
                                                             n - 1)),
                                                   make-cond-list (fetch-called-def (stmt,
                                                                                     proc-list))),
                                         map-down-values (mg-alist (mg-meaning (stmt,
                                                                                proc-list,
                                                                                mg-state,
                                                                                n)),
                                                           bindings (top (ctrl-stk)),
                                                           temp-stk)),
            translate-proc-list (proc-list),
            list (list ('c-c,
                        mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                      proc-list)),
                                                                   proc-list,
                                                                   mg-state (cc (mg-state),
                                                                                make-call-var-alist (mg-alist (mg-s
                                                                                                               stmt,
                                                                                                               fetch-called-de
                                                                                                                               
                                                                                mg-psw (mg-state)),
                                                                   n - 1)),
                                                         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,
                        find-label (get (untag (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-d
```

proc-list,
mg-state (cc (*mg-state*),
make-call-var
mg-psw (*mg-s*
$n - 1$)),
make-cond-list (fetch-called-def (*stmt*,
proc-list
cons (label-cnt (*cinfo*),
cons (label-cnt (*cinfo*),
append (cond-case-jump-label-list (1 + label-cnt (*cinfo*)
1 + length (call-cond
label-cnt-list (label-cnt (*cinfo*),
length (def-cond-locals (fetch-call
append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)))),
ctrl-stk,
map-down-values (mg-alist (mg-meaning (*stmt*,
proc-list,
mg-state,
n)),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list,
mg-state (cc (*mg-state*),
make-call-var-alist (mg-alist (*mg-stat*
stmt,
fetch-called-def (*s*
p
mg-psw (*mg-state*)),
$n - 1$)),
make-cond-list (fetch-called-def (*stmt*,
proc-list))))),
MG-MAX-CTRL-STK-SIZE,

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

;;;;;;;;;;;;;;;;;;;;;; The Normal Return Case ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; In the schema for normal return, n = 1;

THEOREM: call-add1-lc-not-in-code
((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*))
 → (¬ find-labelp (1 + label-cnt (*cinfo*), code (*cinfo*))))

THEOREM: mg-cond-to-p-nat-normal
mg-cond-to-p-nat ('normal, *state*) = '(nat 2)

THEOREM: call-state2-step6-effect-normal-body-equals-final
((*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)))))
∧   normal (mg-meaning (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)),
                                n − 1)))
→   (p-step (p-state (tag ('pc,
                               cons (subr,
                                     find-label (get (untag (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-calle

                                                                                        proc-list,
                                                                                        mg-state (cc (mg-sta
                                                                                                     make-call-



                                                                                                     mg-psw (m
                                                                                        n − 1)),
                                                                                  make-cond-list (fetch-called-def (stmt,
                                                                                                                    proc-
                                                                  cons (label-cnt (cinfo),
                                                                        cons (label-cnt (cinfo),
                                                                              append (cond-case-jump-label-list (1 + label-cnt (ci
                                                                                                                 1 + length (call-c
                                                                                                    label-cnt-list (label-cnt (cinfo),
                                                                                                                    length (def-cond-locals (fetch-

                                                                  append (code (translate (cinfo,
                                                                                           t-cond-list,
                                                                                           stmt,
                                                                                           proc-list)),
                                                                                code2)))),
                                                ctrl-stk,
                                                map-down-values (mg-alist (mg-meaning (stmt,
                                                                                       proc-list,
                                                                                       mg-state,
```

```
                                                                  n)),
                                                bindings(top(ctrl-stk)),
                                                temp-stk),
                                    translate-proc-list(proc-list),
                                    list(list('c-c,
                                              mg-cond-to-p-nat(cc(mg-meaning(def-body(fetch-called-def(stmt,
                                                                                                       proc-list)),
                                                                                     proc-list,
                                                                                     mg-state(cc(mg-state),
                                                                                              make-call-var-alist(mg-alist(mg-s
                                                                                                                           stmt,
                                                                                                                           fetch-called-de


                                                                                              mg-psw(mg-state)),
                                                                                     n - 1)),
                                                               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,
                    if normal(mg-meaning-r(stmt,
                                           proc-list,
                                           mg-state,
                                           n,
                                           list(length(temp-stk),
                                                p-ctrl-stk-size(ctrl-stk))))
                    then length(code(translate(cinfo,
                                               t-cond-list,
                                               stmt,
                                               proc-list)))
                    else find-label(fetch-label(cc(mg-meaning-r(stmt,
                                                                proc-list,
                                                                mg-state,
                                                                n,
                                                                list(length(temp-stk),
                                                                     p-ctrl-stk-size(ctrl-stk)))),
                                                label-alist(translate(cinfo,
                                                                      t-cond-list,
                                                                      stmt,
                                                                      proc-list))),
                                    append(code(translate(cinfo,
```

t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

```
;;;;;;;;;;;;;;;;;;;;;;  The Non-Normal Return Cases ;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: call-lc-not-in-code
((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*))
→ (¬ find-labelp (label-cnt (*cinfo*), code (*cinfo*)))

EVENT: Disable call-lc-not-in-code.

```
;; The 'routineerror case
```

THEOREM: mg-cond-to-p-nat-routineerror
mg-cond-to-p-nat (`'routineerror`, *state*) = `'(nat 1)`

```
;; (push-constant (nat 1))
```

THEOREM: call-state2-step6-effect-routineerror-body

$$
\begin{array}{ll}
 & ((n \not\simeq \texttt{0}) \\
\wedge & (\neg\ \text{resources-inadequatep}\,(\mathit{stmt}, \\
 & \qquad \mathit{proc\text{-}list}, \\
 & \qquad \text{list}\,(\text{length}\,(\mathit{temp\text{-}stk}), \\
 & \qquad\quad \text{p-ctrl-stk-size}\,(\mathit{ctrl\text{-}stk})))) \\
\wedge & (\text{car}\,(\mathit{stmt}) = \texttt{'proc-call-mg}) \\
\wedge & \text{ok-mg-statement}\,(\mathit{stmt}, \mathit{r\text{-}cond\text{-}list}, \mathit{name\text{-}alist}, \mathit{proc\text{-}list}) \\
\wedge & \text{ok-mg-def-plistp}\,(\mathit{proc\text{-}list}) \\
\wedge & \text{ok-translation-parameters}\,(\mathit{cinfo}, \mathit{t\text{-}cond\text{-}list}, \mathit{stmt}, \mathit{proc\text{-}list}, \mathit{code2}) \\
\wedge & \text{ok-mg-statep}\,(\mathit{mg\text{-}state}, \mathit{r\text{-}cond\text{-}list}) \\
\wedge & \text{cond-subsetp}\,(\mathit{r\text{-}cond\text{-}list}, \mathit{t\text{-}cond\text{-}list}) \\
\wedge & (\text{code}\,(\text{translate-def-body}\,(\text{assoc}\,(\mathit{subr}, \mathit{proc\text{-}list}), \mathit{proc\text{-}list})) \\
 & \quad = \quad \text{append}\,(\text{code}\,(\text{translate}\,(\mathit{cinfo}, \mathit{t\text{-}cond\text{-}list}, \mathit{stmt}, \mathit{proc\text{-}list})), \\
 & \qquad\quad \mathit{code2})) \\
\wedge & \text{user-defined-procp}\,(\mathit{subr}, \mathit{proc\text{-}list}) \\
\wedge & \text{plistp}\,(\mathit{temp\text{-}stk}) \\
\wedge & \text{listp}\,(\mathit{ctrl\text{-}stk}) \\
\wedge & \text{mg-vars-list-ok-in-p-state}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}), \\
 & \qquad \text{bindings}\,(\text{top}\,(\mathit{ctrl\text{-}stk})), \\
 & \qquad \mathit{temp\text{-}stk}) \\
\wedge & \text{no-p-aliasing}\,(\text{bindings}\,(\text{top}\,(\mathit{ctrl\text{-}stk})), \text{mg-alist}\,(\mathit{mg\text{-}state})) \\
\wedge & \text{signatures-match}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}), \mathit{name\text{-}alist}) \\
\wedge & \text{normal}\,(\mathit{mg\text{-}state}) \\
\wedge & \text{all-cars-unique}\,(\text{mg-alist}\,(\mathit{mg\text{-}state})) \\
\wedge & (\neg\ \text{resource-errorp}\,(\text{mg-meaning-r}\,(\mathit{stmt}, \\
 & \qquad \mathit{proc\text{-}list}, \\
 & \qquad \mathit{mg\text{-}state}, \\
 & \qquad n, \\
 & \qquad \text{list}\,(\text{length}\,(\mathit{temp\text{-}stk}), \\
 & \qquad\quad \text{p-ctrl-stk-size}\,(\mathit{ctrl\text{-}stk}))))) \\
\wedge & (\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\mathit{stmt}, \mathit{proc\text{-}list})), \\
 & \qquad \mathit{proc\text{-}list}, \\
 & \qquad \text{mg-state}\,(\text{cc}\,(\mathit{mg\text{-}state}), \\
 & \qquad\quad \text{make-call-var-alist}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}), \\
 & \qquad\qquad \mathit{stmt}, \\
 & \qquad\qquad \text{fetch-called-def}\,(\mathit{stmt}, \\
 & \qquad\qquad\quad \mathit{proc\text{-}list})), \\
 & \qquad\quad \text{mg-psw}\,(\mathit{mg\text{-}state})), \\
 & \qquad n - 1)) \\
 & \quad = \quad \texttt{'routineerror}))
\end{array}
$$

```
→  (p-step (p-state (tag ('pc,
                        cons (subr,
                              find-label (get (untag (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-calle

                                                                                       proc-list,
                                                                                       mg-state (cc (mg-sta
                                                                                                     make-call-



                                                                                                     mg-psw (m
                                                                                       n − 1)),
                                                                             make-cond-list (fetch-called-def (stmt,
                                                                                                               proc-
                                                                  cons (label-cnt (cinfo),
                                                                        cons (label-cnt (cinfo),
                                                                              append (cond-case-jump-label-list (1 + label-cnt (ci
                                                                                                                 1 + length (call-c
                                                                                    label-cnt-list (label-cnt (cinfo),
                                                                                                    length (def-cond-locals (fetch-

                                                        append (code (translate (cinfo,
                                                                                 t-cond-list,
                                                                                 stmt,
                                                                                 proc-list)),
                                                                code2)))),
                                    ctrl-stk,
                                    map-down-values (mg-alist (mg-meaning (stmt,
                                                                           proc-list,
                                                                           mg-state,
                                                                           n)),
                                                     bindings (top (ctrl-stk)),
                                                     temp-stk),
                                    translate-proc-list (proc-list),
                                    list (list ('c-c,
                                                mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                              proc-list)),
                                                                                  proc-list,
                                                                                  mg-state (cc (mg-state),
                                                                                               make-call-var-alist (mg-alist (mg-s
                                                                                                                              stmt,
                                                                                                                              fetch-called-de

                                                                                               mg-psw (mg-state)),
```

$n - 1$)),
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*,
length (code (*cinfo*))
+ length (push-parameters-code (def-locals (fetch-called-def (*stmt*,
proc-list)),
call-actuals (*stmt*)))
+ 4)),
ctrl-stk,
push ('(nat 1),
map-down-values (mg-alist (mg-meaning (*stmt*,
proc-list,
mg-state,
n)),
bindings (top (*ctrl-stk*)),
temp-stk)),
translate-proc-list (*proc-list*),
'((c-c (nat 1))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```
;; (pop-global c-c)
```

THEOREM: call-state2-step7-effect-routineerror-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)))))
∧  (cc (mg-meaning (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)),
                   n − 1))
     =  'routineerror))
→  (p-step (p-state (tag ('pc,
                          cons (subr,
                                length (code (cinfo))
                                +  length (push-parameters-code (def-locals (fetch-called-def (stmt,
                                                                                             proc-list)),
                                                                 call-actuals (stmt)))
                                +  4)),
                    ctrl-stk,
                    push ('(nat 1),
                          map-down-values (mg-alist (mg-meaning (stmt,
                                                                 proc-list,
                                                                 mg-state,
                                                                 n)),
                                           bindings (top (ctrl-stk)),
                                           temp-stk)),
```

translate-proc-list (*proc-list*),
'((c-c (nat 1))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

= p-state (tag ('pc,
cons (*subr*,
length (code (*cinfo*))
+ length (push-locals-values-code (def-locals (fetch-called-def (*stmt*, *proc-list*))))
+ length (def-locals (fetch-called-def (*stmt*, *proc-list*)))
+ length (call-actuals (*stmt*))
+ 5)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning (*stmt*, *proc-list*, *mg-state*, *n*)),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
'((c-c (nat 1))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```
;; (jump "routineerror")
```

THEOREM: call-state2-step8-effect-routineerror-body-equals-final
((*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)))))
∧   (cc (mg-meaning (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)),
                     n − 1))
      =   'routineerror))
→   (p-step (p-state (tag ('pc,
                           cons (subr,
                                 length (code (cinfo))
                                 +   length (push-locals-values-code (def-locals (fetch-called-def (stmt,
                                                                                                   proc-list))))
                                 +   length (def-locals (fetch-called-def (stmt,
                                                                           proc-list)))
                                 +   length (call-actuals (stmt))
                                 +   5)),
                      ctrl-stk,
                      map-down-values (mg-alist (mg-meaning (stmt,
                                                             proc-list,
                                                             mg-state,
                                                             n)),
                                       bindings (top (ctrl-stk)),
```

```
                                        temp-stk),
                    translate-proc-list (proc-list),
                    '((c-c (nat 1))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
=   p-state (tag ('pc,
                  cons (subr,
                        if normal (mg-meaning-r (stmt,
                                                 proc-list,
                                                 mg-state,
                                                 n,
                                                 list (length (temp-stk),
                                                       p-ctrl-stk-size (ctrl-stk))))
                        then length (code (translate (cinfo,
                                                      t-cond-list,
                                                      stmt,
                                                      proc-list)))
                        else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                        proc-list,
                                                                        mg-state,
                                                                        n,
                                                                        list (length (temp-stk),
                                                                              p-ctrl-stk-size (ctrl-stk)))),
                                                          label-alist (translate (cinfo,
                                                                                  t-cond-list,
                                                                                  stmt,
                                                                                  proc-list))),
                                                    append (code (translate (cinfo,
                                                                             t-cond-list,
                                                                             stmt,
                                                                             proc-list)),
                                                            code2)) endif)),
                  ctrl-stk,
                  map-down-values (mg-alist (mg-meaning-r (stmt,
                                                           proc-list,
                                                           mg-state,
                                                           n,
                                                           list (length (temp-stk),
                                                                 p-ctrl-stk-size (ctrl-stk)))),
                                   bindings (top (ctrl-stk)),
                                   temp-stk),
                  translate-proc-list (proc-list),
```

list (list ('c-c,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

;;;;;;;;;;;;;; The Non-Normal/Non-Routineerror Exit Case ;;;;;;;;;;;;;;;

THEOREM: body-condition-member-make-cond-list
((*n* ≠ 0)
∧ (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*)
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ normal (*mg-state*)
∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
→ (cc (mg-meaning (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*)),
n − 1))
∈ cons ('normal,
cons ('routineerror,
make-cond-list (fetch-called-def (*stmt*, *proc-list*)))))

THEOREM: leave-not-in-make-cond-list
((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*))
→ (`'leave` ∉ make-cond-list (fetch-called-def (*stmt*, *proc-list*)))

THEOREM: body-condition-not-leave
((*n* ≠ `0`)
 ∧ (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*)
 ∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
 bindings (top (*ctrl-stk*)),
 temp-stk)
 ∧ normal (*mg-state*)
 ∧ (¬ resource-errorp (mg-meaning-r (*stmt*,
 proc-list,
 mg-state,
 n,
 list (length (*temp-stk*),
 p-ctrl-stk-size (*ctrl-stk*))))))
→ (cc (mg-meaning (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*)),
 n − 1))
 ≠ `'leave`)

```
;; checkpoint
```

DEFINITION:
find-def-conds-induction-hint (*index*, *call-conds*, *lc*)
= **if** *call-conds* ≃ **nil** **then** **t**
 else find-def-conds-induction-hint (*index* − 1,
 cdr (*call-conds*),
 1 + *lc*) **endif**

THEOREM: get-indexed-push-constant-instruction
((*k* ≠ 0) ∧ (*k* < (1 + length (*call-conds*))))
→ (get ((*k* − 1) ∗ 3,
append (cond-conversion (*call-conds*, 1 + *lc*, *cond-list*, *label-alist*),
code))
= list ('dl,
k + *lc*,
nil,
list ('push-constant,
mg-cond-to-p-nat (get (*k* − 1, *call-conds*), *cond-list*))))

THEOREM: find-def-conds-label1
((*index* < (1 + length (*call-conds*))) ∧ (*index* ≠ 0) ∧ (*lc* ≠ 0))
→ (find-label (*index* + *lc*,
append (cond-conversion (*call-conds*,
1 + *lc*,
t-cond-list,
label-alist),
code))
= ((*index* − 1) ∗ 3))

THEOREM: find-labelp-def-conds
((*index* < (1 + length (*call-conds*))) ∧ (*index* ≠ 0) ∧ (*lc* ≠ 0))
→ find-labelp (*index* + *lc*,
append (cond-conversion (*call-conds*,
1 + *lc*,
t-cond-list,
label-alist),
code))

THEOREM: call-conds-index-lessp
((car (*stmt*) = 'proc-call-mg)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ (*x* ∈ def-conds (fetch-called-def (*stmt*, *proc-list*))))
→ ((index (*x*, def-conds (fetch-called-def (*stmt*, *proc-list*)))
< (1 + length (call-conds (*stmt*))))
= **t**)

THEOREM: call-def-cond-label-find-labelp
((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*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ (*x* ∈ def-conds (fetch-called-def (*stmt*, *proc-list*))))

$\rightarrow$ (¬ find-labelp (index (x, def-conds (fetch-called-def ($stmt$, $proc\text{-}list$)))
+ (1 + label-cnt ($cinfo$)),
code ($cinfo$)))

```
;; Because of the above lemma, we know that the body-condition is either in the
;; def-conds or in the def-local-conds. We consider each case separately.

;; (push-constant (list 'nat condition-index))
```

THEOREM: call-state2-step6-effect-call-conds-body
(($n \not\simeq$ `0`)
$\wedge$ (¬ resources-inadequatep ($stmt$,
$proc\text{-}list$,
list (length ($temp\text{-}stk$),
p-ctrl-stk-size ($ctrl\text{-}stk$))))
$\wedge$ (car ($stmt$) = `'proc-call-mg`)
$\wedge$ ok-mg-statement ($stmt$, $r\text{-}cond\text{-}list$, $name\text{-}alist$, $proc\text{-}list$)
$\wedge$ ok-mg-def-plistp ($proc\text{-}list$)
$\wedge$ ok-translation-parameters ($cinfo$, $t\text{-}cond\text{-}list$, $stmt$, $proc\text{-}list$, $code2$)
$\wedge$ ok-mg-statep ($mg\text{-}state$, $r\text{-}cond\text{-}list$)
$\wedge$ cond-subsetp ($r\text{-}cond\text{-}list$, $t\text{-}cond\text{-}list$)
$\wedge$ (code (translate-def-body (assoc ($subr$, $proc\text{-}list$), $proc\text{-}list$))
= append (code (translate ($cinfo$, $t\text{-}cond\text{-}list$, $stmt$, $proc\text{-}list$)),
$code2$))
$\wedge$ user-defined-procp ($subr$, $proc\text{-}list$)
$\wedge$ plistp ($temp\text{-}stk$)
$\wedge$ listp ($ctrl\text{-}stk$)
$\wedge$ mg-vars-list-ok-in-p-state (mg-alist ($mg\text{-}state$),
bindings (top ($ctrl\text{-}stk$)),
$temp\text{-}stk$)
$\wedge$ no-p-aliasing (bindings (top ($ctrl\text{-}stk$)), mg-alist ($mg\text{-}state$))
$\wedge$ signatures-match (mg-alist ($mg\text{-}state$), $name\text{-}alist$)
$\wedge$ normal ($mg\text{-}state$)
$\wedge$ all-cars-unique (mg-alist ($mg\text{-}state$))
$\wedge$ (¬ resource-errorp (mg-meaning-r ($stmt$,
$proc\text{-}list$,
$mg\text{-}state$,
n,
list (length ($temp\text{-}stk$),
p-ctrl-stk-size ($ctrl\text{-}stk$)))))
$\wedge$ (¬ normal (mg-meaning (def-body (fetch-called-def ($stmt$, $proc\text{-}list$)),
$proc\text{-}list$,
mg-state (cc ($mg\text{-}state$),

make-call-var-alist (mg-alist (*mg-state*),
stmt,
fetch-called-def (*stmt*,
proc-list)),
mg-psw (*mg-state*)),
$n - 1$)))
$\wedge$ (cc (mg-meaning (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*)),
$n - 1$))
$\neq$ `'routineerror`)
$\wedge$ (cc (mg-meaning (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*)),
$n - 1$))
$\in$ def-conds (fetch-called-def (*stmt*, *proc-list*))))
$\rightarrow$ (p-step (p-state (tag (`'pc`,
cons (*subr*,
find-label (get (untag (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-calle

proc-list,
mg-state (cc (*mg-sta*
make-call-

mg-psw (*m*
$n - 1$)),
make-cond-list (fetch-called-def (*stmt*,
proc-
cons (label-cnt (*cinfo*),
cons (label-cnt (*cinfo*),
append (cond-case-jump-label-list (1 + label-cnt (*ci*
1 + length (call-c

```
                                                                label-cnt-list (label-cnt (cinfo),
                                                                                length (def-cond-locals (fetch-

                                                    append (code (translate (cinfo,
                                                                             t-cond-list,
                                                                             stmt,
                                                                             proc-list)),
                                                                 code2)))),
                                    ctrl-stk,
                                    map-down-values (mg-alist (mg-meaning (stmt,
                                                                           proc-list,
                                                                           mg-state,
                                                                           n)),
                                                               bindings (top (ctrl-stk)),
                                                               temp-stk),
                                    translate-proc-list (proc-list),
                                    list (list ('c-c,
                                                mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                              proc-list)),
                                                                                  proc-list,
                                                                                  mg-state (cc (mg-state),
                                                                                               make-call-var-alist (mg-alist (mg-s
                                                                                                                              stmt,
                                                                                                                              fetch-called-de

                                                                                               mg-psw (mg-state)),
                                                                                  n - 1)),
                                                                     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,
                        length (code (cinfo))
                        +  length (push-parameters-code (def-locals (fetch-called-def (stmt,
                                                                                       proc-list)),
                                                                     call-actuals (stmt)))
                        +  6
                        +  ((index (cc (mg-meaning (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-
                                                    mg-psw (mg-state)),
                                                n - 1)),
                                        def-conds (fetch-called-def (stmt,
                                                                     proc-list))) - 1)
                                *  3)
                            +  1)),
                        ctrl-stk,
                        push (mg-cond-to-p-nat (get (index (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                                        proc-list)),
                                                                            proc-list,
                                                                            mg-state (cc (mg-state),
                                                                                         make-call-var-alist (mg-alist (
                                                                                                              stmt,
                                                                                                              fetch-call


                                                                                         mg-psw (mg-state)),
                                                                            n - 1)),
                                                                  def-conds (fetch-called-def (stmt,
                                                                                               proc-list))) - 1,
                                                               call-conds (stmt)),
                                                        t-cond-list),
                                             map-down-values (mg-alist (mg-meaning (stmt,
                                                                                    proc-list,
                                                                                    mg-state,
                                                                                    n)),
                                                              bindings (top (ctrl-stk)),
                                                              temp-stk)),
                        translate-proc-list (proc-list),
                        list (list ('c-c,
                                    mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                                  proc-list)),
                                                                      proc-list,
                                                                      mg-state (cc (mg-state),
                                                                                   make-call-var-alist (mg-alist (mg-stat
                                                                                                        stmt,
                                                                                                        fetch-called-def (s
                                                                                                                          p
                                                                                   mg-psw (mg-state)),
                                                                      n - 1)),
                                                            make-cond-list (fetch-called-def (stmt,
```

proc-list))))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: get-indexed-pop-global-instruction
((*k* ≠ `0`) ∧ (*k* < (1 + length (*call-conds*))))
→ (get (((*k* − 1) ∗ `3`) + `1`,
append (cond-conversion (*call-conds*, 1 + *lc*, *cond-list*, *label-alist*),
code))
= `'(pop-global c-c)`)

THEOREM: call-state2-step7-effect-call-conds-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*)))))

$$
\begin{array}{ll}
\wedge & (\neg\ \text{normal}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\mathit{stmt},\ \mathit{proc\text{-}list})),\\
 & \qquad \mathit{proc\text{-}list},\\
 & \qquad \text{mg-state}\,(\text{cc}\,(\mathit{mg\text{-}state}),\\
 & \qquad\qquad \text{make-call-var-alist}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}),\\
 & \qquad\qquad\qquad \mathit{stmt},\\
 & \qquad\qquad\qquad \text{fetch-called-def}\,(\mathit{stmt},\\
 & \qquad\qquad\qquad\qquad \mathit{proc\text{-}list})),\\
 & \qquad\qquad \text{mg-psw}\,(\mathit{mg\text{-}state})),\\
 & \qquad n-1)))\\
\wedge & (\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\mathit{stmt},\ \mathit{proc\text{-}list})),\\
 & \qquad \mathit{proc\text{-}list},\\
 & \qquad \text{mg-state}\,(\text{cc}\,(\mathit{mg\text{-}state}),\\
 & \qquad\qquad \text{make-call-var-alist}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}),\\
 & \qquad\qquad\qquad \mathit{stmt},\\
 & \qquad\qquad\qquad \text{fetch-called-def}\,(\mathit{stmt},\\
 & \qquad\qquad\qquad\qquad \mathit{proc\text{-}list})),\\
 & \qquad\qquad \text{mg-psw}\,(\mathit{mg\text{-}state})),\\
 & \qquad n-1))\\
 & \neq\ \texttt{'routineerror})\\
\wedge & (\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\mathit{stmt},\ \mathit{proc\text{-}list})),\\
 & \qquad \mathit{proc\text{-}list},\\
 & \qquad \text{mg-state}\,(\text{cc}\,(\mathit{mg\text{-}state}),\\
 & \qquad\qquad \text{make-call-var-alist}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}),\\
 & \qquad\qquad\qquad \mathit{stmt},\\
 & \qquad\qquad\qquad \text{fetch-called-def}\,(\mathit{stmt},\\
 & \qquad\qquad\qquad\qquad \mathit{proc\text{-}list})),\\
 & \qquad\qquad \text{mg-psw}\,(\mathit{mg\text{-}state})),\\
 & \qquad n-1))\\
 & \in\ \text{def-conds}\,(\text{fetch-called-def}\,(\mathit{stmt},\ \mathit{proc\text{-}list}))))\\
\rightarrow & (\text{p-step}\,(\text{p-state}\,(\text{tag}\,(\texttt{'pc},\\
 & \qquad \text{cons}\,(\mathit{subr},\\
 & \qquad\qquad \text{length}\,(\text{code}\,(\mathit{cinfo}))\\
 & \qquad\qquad +\ \ \text{length}\,(\text{push-parameters-code}\,(\text{def-locals}\,(\text{fetch-called-def}\,(\mathit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{proc\text{-}list})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{call-actuals}\,(\mathit{stmt})))\\
 & \qquad\qquad +\ \ \texttt{6}\\
 & \qquad\qquad +\ \ ((\text{index}\,(\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\mathit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{proc\text{-}list})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{proc\text{-}list},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{mg-state}\,(\text{cc}\,(\mathit{mg\text{-}state}),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{make-call-var-alist}\,(\text{mg-alist}\,(\mathit{mg\text{-}state})\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{stmt},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{fetch-called-def}\,(\mathit{st}\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{pr}
\end{array}
$$

```
                                                    mg-psw (mg-state)),
                                                n - 1)),
                                    def-conds (fetch-called-def (stmt,
                                                                 proc-list))) - 1)
                            * 3)
                    + 1)),
ctrl-stk,
push (mg-cond-to-p-nat (get (index (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                proc-list)
                                                           proc-list,
                                                           mg-state (cc (mg-state),
                                                                        make-call-var-alist (mg-al
                                                                                             stmt,
                                                                                             fetch-


                                                                        mg-psw (mg-state)),
                                                           n - 1)),
                                                 def-conds (fetch-called-def (stmt,
                                                                              proc-list))) - 1,
                                               call-conds (stmt)),
                                    t-cond-list),
                map-down-values (mg-alist (mg-meaning (stmt,
                                                       proc-list,
                                                       mg-state,
                                                       n)),
                                    bindings (top (ctrl-stk)),
                                    temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
            mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                          proc-list)),
                                                       proc-list,
                                                       mg-state (cc (mg-state),
                                                                    make-call-var-alist (mg-alist (mg-s
                                                                                                   stmt,
                                                                                                   fetch-called-de


                                                                    mg-psw (mg-state)),
                                                       n - 1)),
                                             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,
                       length (code (cinfo))
                       + length (push-locals-values-code (def-locals (fetch-called-def (stmt,
                                                                                        proc-list))))
                       + length (def-locals (fetch-called-def (stmt,
                                                               proc-list)))
                       + length (call-actuals (stmt))
                       + 6
                       + ((index (cc (mg-meaning (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-
                                                                     mg-psw (mg-state)),
                                                           n - 1)),
                                 def-conds (fetch-called-def (stmt,
                                                              proc-list))) - 1)
                          * 3)
                       + 2)),
                 ctrl-stk,
                 map-down-values (mg-alist (mg-meaning (stmt,
                                                        proc-list,
                                                        mg-state,
                                                        n)),
                                  bindings (top (ctrl-stk)),
                                  temp-stk),
                 translate-proc-list (proc-list),
                 list (cons ('c-c,
                             put (mg-cond-to-p-nat (get (index (cc (mg-meaning (def-body (fetch-called-def (stm
                                                                                                            pro
                                                                                    proc-list,
                                                                                    mg-state (cc (mg-state),
                                                                                              make-call-var-alist (
                                                                                                                   
                                                                                                                   
                                                                                                                   
                                                                                              mg-psw (mg-state)),
                                                                                    n - 1)),
                                                                  def-conds (fetch-called-def (stmt,
```

proc-list))) − 1,
call-conds (*stmt*)),
t-cond-list),
0,
list (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (*stmt*,
proc-list))
proc-list,
mg-state (cc (*mg-state*),
make-call-var-alist (mg-alis
stmt,
fetch-c
mg-psw (*mg-state*)),
n − 1)),
make-cond-list (fetch-called-def (*stmt*,
proc-list))))))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: get-indexed-jump-instruction
((*k* ≉ 0) ∧ (*k* < (1 + length (*call-conds*))))
→ (get (((*k* − 1) ∗ 3) + 2,
append (cond-conversion (*call-conds*, 1 + *lc*, *cond-list*, *label-alist*),
code))
= list ('jump, fetch-label (get (*k* − 1, *call-conds*), *label-alist*)))

THEOREM: convert-condition1-index-equivalence
((length (*def-conds*) = length (*call-conds*)) ∧ (*x* ∈ *def-conds*))
→ (convert-condition1 (*x*, *def-conds*, *call-conds*)
= get (index (*x*, *def-conds*) − 1, *call-conds*))

THEOREM: nonnormal-cond-conversion-not-normal
('normal ∉ *call-conds*)
→ (convert-condition1 (*cc*, *def-conds*, *call-conds*) ≠ 'normal)

THEOREM: call-state2-step8-effect-call-conds-body-equals-final
((*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*)

$$
\begin{array}{ll}
\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\qquad \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}))))) \\
\wedge & (\neg\ \text{normal}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\textit{stmt},\ \textit{proc-list})), \\
 & \qquad\qquad\qquad\qquad \textit{proc-list}, \\
 & \qquad\qquad\qquad\qquad \text{mg-state}\,(\text{cc}\,(\textit{mg-state}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad \text{make-call-var-alist}\,(\text{mg-alist}\,(\textit{mg-state}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{stmt}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{fetch-called-def}\,(\textit{stmt}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list})), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad \text{mg-psw}\,(\textit{mg-state})), \\
 & \qquad\qquad\qquad\qquad n - 1))) \\
\wedge & (\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\textit{stmt},\ \textit{proc-list})), \\
 & \qquad\qquad\qquad\quad \textit{proc-list}, \\
 & \qquad\qquad\qquad\quad \text{mg-state}\,(\text{cc}\,(\textit{mg-state}), \\
 & \qquad\qquad\qquad\qquad\quad \text{make-call-var-alist}\,(\text{mg-alist}\,(\textit{mg-state}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{stmt}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{fetch-called-def}\,(\textit{stmt}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list})), \\
 & \qquad\qquad\qquad\qquad\quad \text{mg-psw}\,(\textit{mg-state})), \\
 & \qquad\qquad\qquad\quad n - 1)) \\
 & \quad \neq \quad \texttt{'routineerror}) \\
\wedge & (\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\textit{stmt},\ \textit{proc-list})), \\
 & \qquad\qquad\qquad\quad \textit{proc-list},
\end{array}
$$

```
                    mg-state (cc (mg-state),
                              make-call-var-alist (mg-alist (mg-state),
                                                   stmt,
                                                   fetch-called-def (stmt,
                                                                     proc-list)),
                              mg-psw (mg-state)),
                    n − 1))
          ∈   def-conds (fetch-called-def (stmt, proc-list))))
 →   (p-step (p-state (tag ('pc,
                       cons (subr,
                             length (code (cinfo))
                             +   length (push-locals-values-code (def-locals (fetch-called-def (stmt,
                                                                                                  proc-list))))
                             +   length (def-locals (fetch-called-def (stmt,
                                                                       proc-list)))
                             +   length (call-actuals (stmt))
                             +   6
                             +   ((index (cc (mg-meaning (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 (st
                                                                                                           pr
                                                                    mg-psw (mg-state)),
                                                          n − 1)),
                                         def-conds (fetch-called-def (stmt,
                                                                      proc-list))) − 1)
                                  ∗   3)
                             +   2)),
                  ctrl-stk,
                  map-down-values (mg-alist (mg-meaning (stmt,
                                                         proc-list,
                                                         mg-state,
                                                         n)),
                                   bindings (top (ctrl-stk)),
                                   temp-stk),
                  translate-proc-list (proc-list),
                  list (cons ('c-c,
                              put (mg-cond-to-p-nat (get (index (cc (mg-meaning (def-body (fetch-called-def (

                                                                                 proc-list,
                                                                                 mg-state (cc (mg-state),
```

```
                                                          make-call-var-ali

                                                          mg-psw (mg-stat
                                                     n − 1)),
                                             def-conds (fetch-called-def (stmt,
                                                                          proc-list))) − 1,
                                         call-conds (stmt)),
                                       t-cond-list),
                              0,
                              list (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                                  proc-lis
                                                 proc-list,
                                                 mg-state (cc (mg-state),
                                                          make-call-var-alist (mg-
                                                                               stm
                                                                               fetc

                                                          mg-psw (mg-state)),
                                                 n − 1)),
                                         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,
                       if normal (mg-meaning-r (stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list (length (temp-stk),
                                                      p-ctrl-stk-size (ctrl-stk))))
                       then length (code (translate (cinfo,
                                                     t-cond-list,
                                                     stmt,
                                                     proc-list)))
                       else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                       proc-list,
                                                                       mg-state,
                                                                       n,
                                                                       list (length (temp-stk),
```

p-ctrl-stk-size (*ctrl-stk*)))),
label-alist (translate (*cinfo*,
t-cond-list,
stmt,
proc-list))),
append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list ('`c-c`,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'`run`))

THEOREM: get-member-label-cnt-list
(*cc* ∈ *lst*)
→ (get (index (*cc*, *lst*) − 1, label-cnt-list (*lc*, length (*lst*))) = *lc*)

```
;; This is the case where the condition returned by the call is not in the
;; def-conds list.  It must therefore be in the def-cond-locals list and we
;; return routineerror.
;;
;; (push-constant (list 'nat condition-index))
```

THEOREM: call-state2-step6-effect-local-conds-body

$$
\begin{array}{l}
((n \not\simeq \texttt{0}) \\
\wedge \quad (\neg\ \text{resources-inadequatep}\,(\mathit{stmt}, \\
\qquad\qquad\qquad\qquad \mathit{proc\text{-}list}, \\
\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\mathit{temp\text{-}stk}), \\
\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\mathit{ctrl\text{-}stk})))) \\
\wedge \quad (\text{car}\,(\mathit{stmt}) = \texttt{'proc-call-mg}) \\
\wedge \quad \text{ok-mg-statement}\,(\mathit{stmt},\ \mathit{r\text{-}cond\text{-}list},\ \mathit{name\text{-}alist},\ \mathit{proc\text{-}list}) \\
\wedge \quad \text{ok-mg-def-plistp}\,(\mathit{proc\text{-}list}) \\
\wedge \quad \text{ok-translation-parameters}\,(\mathit{cinfo},\ \mathit{t\text{-}cond\text{-}list},\ \mathit{stmt},\ \mathit{proc\text{-}list},\ \mathit{code2}) \\
\wedge \quad \text{ok-mg-statep}\,(\mathit{mg\text{-}state},\ \mathit{r\text{-}cond\text{-}list}) \\
\wedge \quad \text{cond-subsetp}\,(\mathit{r\text{-}cond\text{-}list},\ \mathit{t\text{-}cond\text{-}list}) \\
\wedge \quad (\text{code}\,(\text{translate-def-body}\,(\text{assoc}\,(\mathit{subr},\ \mathit{proc\text{-}list}),\ \mathit{proc\text{-}list})) \\
\qquad = \quad \text{append}\,(\text{code}\,(\text{translate}\,(\mathit{cinfo},\ \mathit{t\text{-}cond\text{-}list},\ \mathit{stmt},\ \mathit{proc\text{-}list})), \\
\qquad\qquad\quad \mathit{code2})) \\
\wedge \quad \text{user-defined-procp}\,(\mathit{subr},\ \mathit{proc\text{-}list}) \\
\wedge \quad \text{plistp}\,(\mathit{temp\text{-}stk}) \\
\wedge \quad \text{listp}\,(\mathit{ctrl\text{-}stk}) \\
\wedge \quad \text{mg-vars-list-ok-in-p-state}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}), \\
\qquad\qquad\qquad\qquad \text{bindings}\,(\text{top}\,(\mathit{ctrl\text{-}stk})), \\
\qquad\qquad\qquad\qquad \mathit{temp\text{-}stk}) \\
\wedge \quad \text{no-p-aliasing}\,(\text{bindings}\,(\text{top}\,(\mathit{ctrl\text{-}stk})),\ \text{mg-alist}\,(\mathit{mg\text{-}state})) \\
\wedge \quad \text{signatures-match}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}),\ \mathit{name\text{-}alist}) \\
\wedge \quad \text{normal}\,(\mathit{mg\text{-}state}) \\
\wedge \quad \text{all-cars-unique}\,(\text{mg-alist}\,(\mathit{mg\text{-}state})) \\
\wedge \quad (\neg\ \text{resource-errorp}\,(\text{mg-meaning-r}\,(\mathit{stmt}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{proc\text{-}list}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{mg\text{-}state}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad n, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{list}\,(\text{length}\,(\mathit{temp\text{-}stk}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\mathit{ctrl\text{-}stk}))))) \\
\wedge \quad (\neg\ \text{normal}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\mathit{stmt},\ \mathit{proc\text{-}list})), \\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{proc\text{-}list}, \\
\qquad\qquad\qquad\qquad\qquad\quad \text{mg-state}\,(\text{cc}\,(\mathit{mg\text{-}state}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{make-call-var-alist}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{stmt}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{fetch-called-def}\,(\mathit{stmt}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{proc\text{-}list})), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{mg-psw}\,(\mathit{mg\text{-}state})), \\
\qquad\qquad\qquad\qquad\qquad\quad n - 1))) \\
\wedge \quad (\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\mathit{stmt},\ \mathit{proc\text{-}list})), \\
\qquad\qquad\qquad\qquad\quad \mathit{proc\text{-}list}, \\
\qquad\qquad\qquad\qquad\quad \text{mg-state}\,(\text{cc}\,(\mathit{mg\text{-}state}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{make-call-var-alist}\,(\text{mg-alist}\,(\mathit{mg\text{-}state}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{stmt},
\end{array}
$$

fetch-called-def (*stmt*,
proc-list)),
mg-psw (*mg-state*)),
$n - 1$))
$\neq$ 'routineerror)
$\wedge$ (cc (mg-meaning (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*)),
$n - 1$))
$\notin$ def-conds (fetch-called-def (*stmt*, *proc-list*))))
$\rightarrow$ (p-step (p-state (tag ('pc,
cons (*subr*,
find-label (get (untag (mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-calle

proc-list,
mg-state (cc (*mg-sta*
make-call-

mg-psw (*m*
$n - 1$)),
make-cond-list (fetch-called-def (*stmt*,
proc-
cons (label-cnt (*cinfo*),
cons (label-cnt (*cinfo*),
append (cond-case-jump-label-list (1 + label-cnt (*ci*
1 + length (call-c
label-cnt-list (label-cnt (*cinfo*),
length (def-cond-locals (fetch-

append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)))),
ctrl-stk,
map-down-values (mg-alist (mg-meaning (*stmt*,
proc-list,

```
                                                    mg-state,
                                                    n)),
                                        bindings (top (ctrl-stk)),
                                        temp-stk),
                            translate-proc-list (proc-list),
                            list (list ('c-c,
                                        mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                                      proc-list)),
                                                                                   proc-list,
                                                                                   mg-state (cc (mg-state),
                                                                                                make-call-var-alist (mg-alist (mg-s
                                                                                                                               stmt,
                                                                                                                               fetch-called-de

                                                                                                                    mg-psw (mg-state)),
                                                                                   n - 1)),
                                                             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,
                        length (code (cinfo))
                        +   length (push-parameters-code (def-locals (fetch-called-def (stmt,
                                                                                        proc-list)),
                                                                      call-actuals (stmt)))
                        +   4)),
             ctrl-stk,
             push ('(nat 1),
                   map-down-values (mg-alist (mg-meaning (stmt,
                                                          proc-list,
                                                          mg-state,
                                                          n)),
                                              bindings (top (ctrl-stk)),
                                              temp-stk)),
             translate-proc-list (proc-list),
             list (list ('c-c,
                         mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (stmt,
                                                                                       proc-list)),
                                                                    proc-list,
                                                                    mg-state (cc (mg-state),
                                                                                 make-call-var-alist (mg-alist (mg-stat
```

stmt,
fetch-called-def (*s*
p
mg-psw (*mg-state*)),
$n - 1$)),
make-cond-list (fetch-called-def (*stmt*,
proc-list))))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: call-state2-step7-effect-local-conds-body
(($n \not\simeq$ `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*)))))
∧ (¬ normal (mg-meaning (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)),
                                    n − 1)))
∧   (cc (mg-meaning (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)),
                    n − 1))
    ≠   'routineerror)
∧   (cc (mg-meaning (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)),
                    n − 1))
    ∉   def-conds (fetch-called-def (stmt, proc-list))))
→   (p-step (p-state (tag ('pc,
                           cons (subr,
                                 length (code (cinfo))
                                 +   length (push-parameters-code (def-locals (fetch-called-def (stmt,
                                                                                                 proc-list)),
                                                                   call-actuals (stmt)))
                                 +   4)),
                    ctrl-stk,
                    push ('(nat 1),
                          map-down-values (mg-alist (mg-meaning (stmt,
                                                                 proc-list,
                                                                 mg-state,
                                                                 n)),
                                           bindings (top (ctrl-stk)),
                                           temp-stk)),
                    translate-proc-list (proc-list),
```

list (list ('c-c,
mg-cond-to-p-nat (cc (mg-meaning (def-body (fetch-called-def (*stmt*,
proc-list)),
proc-list,
mg-state (cc (*mg-state*),
make-call-var-alist (mg-alist (*mg-s*
stmt,
fetch-called-de

mg-psw (*mg-state*)),
$n - 1$)),
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*,
length (code (*cinfo*))
+ length (push-locals-values-code (def-locals (fetch-called-def (*stmt*,
proc-list))))
+ length (def-locals (fetch-called-def (*stmt*,
proc-list)))
+ length (call-actuals (*stmt*))
+ 5)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning (*stmt*,
proc-list,
mg-state,
n)),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
'((c-c (nat 1))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: call-state2-step8-effect-local-conds-body-equals-final
((*n* $\not\simeq$ 0)
$\wedge$ ($\neg$ resources-inadequatep (*stmt*,
proc-list,

$$
\begin{array}{ll}
 & \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}))))\\
\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\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\qquad\qquad \text{bindings}\,(\text{top}\,(\textit{ctrl-stk})),\\
 & \qquad\qquad\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\qquad\qquad \textit{proc-list},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{mg-state},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad n,\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\textit{temp-stk}),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))))\\
\wedge & (\neg\ \text{normal}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\textit{stmt},\ \textit{proc-list})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list},\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{mg-state}\,(\text{cc}\,(\textit{mg-state}),\\
 & \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\quad \textit{stmt},\\
 & \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\quad \textit{proc-list})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{mg-psw}\,(\textit{mg-state})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad n-1)))\\
\wedge & (\text{cc}\,(\text{mg-meaning}\,(\text{def-body}\,(\text{fetch-called-def}\,(\textit{stmt},\ \textit{proc-list})),\\
 & \qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list},\\
 & \qquad\qquad\qquad\qquad\qquad\quad \text{mg-state}\,(\text{cc}\,(\textit{mg-state}),\\
 & \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\quad \textit{stmt},\\
 & \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\quad \textit{proc-list})),\\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{mg-psw}\,(\textit{mg-state})),
\end{array}
$$

```
                    n − 1))
    ≠   'routineerror)
∧   (cc (mg-meaning (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)),
                    n − 1))
    ∉   def-conds (fetch-called-def (stmt, proc-list))))
→   (p-step (p-state (tag ('pc,
                           cons (subr,
                                 length (code (cinfo))
                                 +   length (push-locals-values-code (def-locals (fetch-called-def (stmt,
                                                                                                    proc-list))))
                                 +   length (def-locals (fetch-called-def (stmt,
                                                                           proc-list)))
                                 +   length (call-actuals (stmt))
                                 +   5)),
                    ctrl-stk,
                    map-down-values (mg-alist (mg-meaning (stmt,
                                                           proc-list,
                                                           mg-state,
                                                           n)),
                                     bindings (top (ctrl-stk)),
                                     temp-stk),
                    translate-proc-list (proc-list),
                    '((c-c (nat 1))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
    =   p-state (tag ('pc,
                      cons (subr,
                            if normal (mg-meaning-r (stmt,
                                                     proc-list,
                                                     mg-state,
                                                     n,
                                                     list (length (temp-stk),
                                                           p-ctrl-stk-size (ctrl-stk))))
                            then length (code (translate (cinfo,
                                                          t-cond-list,
```

stmt,
proc-list)))
else find-label (fetch-label (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
label-alist (translate (*cinfo*,
t-cond-list,
stmt,
proc-list))),
append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```
;; Time for the final proc-call lemma
```

THEOREM: call-exact-time-schema-normal-case

$$
\begin{array}{ll}
((\textit{stmt-time} = (\textit{locals-data-length} \\
\qquad\qquad + \quad \textit{locals-length} \\
\qquad\qquad + \quad \textit{actuals-length} \\
\qquad\qquad + \quad \texttt{1} \\
\qquad\qquad + \quad \textit{body-time} \\
\qquad\qquad + \quad \texttt{5} \\
\qquad\qquad + \quad \texttt{1})) \\
\wedge \quad (\mathrm{p}\,(\textit{initial}, \\
\qquad \textit{locals-data-length} \\
\qquad + \quad \textit{locals-length} \\
\qquad + \quad \textit{actuals-length} \\
\qquad + \quad \texttt{1} \\
\qquad + \quad \textit{body-time}) \\
\qquad = \quad \textit{state2}) \\
\wedge \quad (\mathrm{p}\,(\textit{state2}, \texttt{6}) = \textit{final})) \\
\rightarrow \quad (\mathrm{p}\,(\textit{initial}, \textit{stmt-time}) = \textit{final})
\end{array}
$$

THEOREM: call-exact-time-schema-nonnormal-case

$$
\begin{array}{ll}
((\textit{stmt-time} = (\textit{locals-data-length} \\
\qquad\qquad + \quad \textit{locals-length} \\
\qquad\qquad + \quad \textit{actuals-length} \\
\qquad\qquad + \quad \texttt{1} \\
\qquad\qquad + \quad \textit{body-time} \\
\qquad\qquad + \quad \texttt{5} \\
\qquad\qquad + \quad \texttt{3})) \\
\wedge \quad (\mathrm{p}\,(\textit{initial}, \\
\qquad \textit{locals-data-length} \\
\qquad + \quad \textit{locals-length} \\
\qquad + \quad \textit{actuals-length} \\
\qquad + \quad \texttt{1} \\
\qquad + \quad \textit{body-time}) \\
\qquad = \quad \textit{state2}) \\
\wedge \quad (\mathrm{p}\,(\textit{state2}, \texttt{8}) = \textit{final})) \\
\rightarrow \quad (\mathrm{p}\,(\textit{initial}, \textit{stmt-time}) = \textit{final})
\end{array}
$$

```
(prove-lemma proc-call-exact-time-lemma (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)))))
    (IMPLIES
     (AND
      (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-MG-DEF-PLISTP PROC-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)))))
      (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)))
      (EQUAL
       (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))))))
      (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
    (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))
      (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
      (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)))
      (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
      (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)))
       (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))))
      (NOT
       (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))
 (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)))))))
     (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)
   (CLOCK STMT PROC-LIST MG-STATE N))
(P-STATE
 (TAG 'PC
      (CONS SUBR
    (IF
     (NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N
   (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
```

```
    (LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
    (FIND-LABEL
     (FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
    (LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))
  (LABEL-ALIST (TRANSLATE CINFO T-COND-LIST STMT
  PROC-LIST)))
     (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
     CODE2)))))
 CTRL-STK
 (MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R STMT PROC-LIST MG-STATE N
  (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
  (BINDINGS (TOP CTRL-STK))
  TEMP-STK)
 (TRANSLATE-PROC-LIST PROC-LIST)
 (LIST
  (LIST 'C-C
(MG-COND-TO-P-NAT (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
    (LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))
  T-COND-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 @FINAL
    (P-STATE
     (TAG 'PC
      (CONS SUBR
       (IF
        (NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N
                              (LIST (LENGTH TEMP-STK)
                                    (P-CTRL-STK-SIZE CTRL-STK))))
        (LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
        (FIND-LABEL
             (FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
                                            (LIST (LENGTH TEMP-STK)
                                                  (P-CTRL-STK-SIZE CTRL-STK))))
                          (LABEL-ALIST (TRANSLATE CINFO T-COND-LIST STMT
```

```
                                                PROC-LIST)))
          (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
                  CODE2)))))
  CTRL-STK
  (MAP-DOWN-VALUES
                (MG-ALIST (MG-MEANING-R STMT PROC-LIST MG-STATE N
                                        (LIST (LENGTH TEMP-STK)
                                              (P-CTRL-STK-SIZE CTRL-STK))))
                (BINDINGS (TOP CTRL-STK))
                TEMP-STK)
  (TRANSLATE-PROC-LIST PROC-LIST)
  (LIST
   (LIST 'C-C
     (MG-COND-TO-P-NAT (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
                                         (LIST (LENGTH TEMP-STK)
                                               (P-CTRL-STK-SIZE CTRL-STK))))
                       T-COND-LIST)))
  (MG-MAX-CTRL-STK-SIZE) (MG-MAX-TEMP-STK-SIZE) (MG-WORD-SIZE)
  'RUN))
(ADD-ABBREVIATION @STMT-TIME
                  (CLOCK STMT PROC-LIST MG-STATE N))
(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 @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 @LOCALS-DATA-LENGTH
               (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
(ADD-ABBREVIATION @LOCALS-LENGTH
                  (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
(ADD-ABBREVIATION @ACTUALS-LENGTH
                  (LENGTH (CALL-ACTUALS STMT)))
PROMOTE
(DEMOTE 19)
(DIVE 1 1)
PUSH TOP PROMOTE
(CLAIM (EQUAL (P @INITIAL
                 (PLUS @LOCALS-DATA-LENGTH @LOCALS-LENGTH @ACTUALS-LENGTH
                       1 @BODY-TIME))
              @STATE2)
       0)
(CLAIM
 (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
        (PLUS (LENGTH TEMP-STK)
              (DATA-LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST))))
        (PLUS (P-CTRL-STK-SIZE CTRL-STK)
              (PLUS 2
                    (LENGTH (DEF-LOCALS (FETCH-CALLED-DEF STMT PROC-LIST)))
                    (LENGTH (DEF-FORMALS (FETCH-CALLED-DEF STMT
                                                           PROC-LIST))))))))
 0)
(CLAIM (EQUAL @STMT-TIME
```

```
              (PLUS @LOCALS-DATA-LENGTH @LOCALS-LENGTH @ACTUALS-LENGTH 1
                    @BODY-TIME 5 1))
       0)
(CLAIM (EQUAL (P @STATE2 6) @FINAL) 0)
(DEMOTE 20 22 23)
(GENERALIZE ((@ACTUALS-LENGTH ACTUALS-LENGTH)
             (@LOCALS-LENGTH LOCALS-LENGTH)
             (@LOCALS-DATA-LENGTH LOCALS-DATA-LENGTH)
             (@STATE2 STATE2)
             (@STATE1 STATE1)
             (@BODY-TIME BODY-TIME)
             (@STMT-TIME STMT-TIME)
             (@FINAL FINAL)
             (@INITIAL INITIAL)))
DROP
(USE-LEMMA CALL-EXACT-TIME-SCHEMA-NORMAL-CASE)
DEMOTE
(S-PROP AND OR NOT IMPLIES FIX ZEROP IFF NLISTP)
(CONTRADICT 23)
(DIVE 1)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-0-UNWINDING-LEMMA)
(DIVE 1 1 1 1 1)
(REWRITE CALL-STATE2-STEP1-EFFECT)
UP
(REWRITE CALL-STATE2-STEP2-EFFECT)
UP
(REWRITE CALL-STATE2-STEP3-EFFECT)
UP
(REWRITE CALL-STATE2-STEP4-EFFECT)
UP
(REWRITE CALL-STATE2-STEP5-EFFECT)
UP
(REWRITE CALL-STATE2-STEP6-EFFECT-NORMAL-BODY-EQUALS-FINAL)
TOP S-PROP
(DEMOTE 21)
(DIVE 1 1)
(REWRITE MG-MEANING-EQUIVALENCE)
TOP S
```

```
(DEMOTE 16)
DROP PROVE
(DIVE 1)
(REWRITE PROC-CALL-DOESNT-HALT2)
TOP S S
(CONTRADICT 22)
(DIVE 1)
X
(= (CAR STMT) 'PROC-CALL-MG 0)
S
(DIVE 2 2 2 2 2 2 1)
(= * T 0)
TOP DROP PROVE
(DEMOTE 21)
(DIVE 1 1)
(REWRITE MG-MEANING-EQUIVALENCE)
TOP S
(DIVE 1)
(REWRITE PROC-CALL-DOESNT-HALT2)
TOP S
(CLAIM (EQUAL @STMT-TIME
              (PLUS @LOCALS-DATA-LENGTH @LOCALS-LENGTH @ACTUALS-LENGTH 1
                    @BODY-TIME 5 3))
       0)
(DEMOTE 21)
(DIVE 1 1 1)
(REWRITE MG-MEANING-EQUIVALENCE)
TOP PROMOTE
(CLAIM (EQUAL (P @STATE2 8) @FINAL) 0)
(DEMOTE 20 21 23)
(GENERALIZE ((@ACTUALS-LENGTH ACTUALS-LENGTH)
             (@LOCALS-LENGTH LOCALS-LENGTH)
             (@LOCALS-DATA-LENGTH LOCALS-DATA-LENGTH)
             (@STATE2 STATE2)
             (@STATE1 STATE1)
             (@BODY-TIME BODY-TIME)
             (@STMT-TIME STMT-TIME)
             (@FINAL FINAL)
             (@INITIAL INITIAL)))
DROP
(USE-LEMMA CALL-EXACT-TIME-SCHEMA-NONNORMAL-CASE)
DEMOTE
(S-PROP AND OR NOT IMPLIES FIX ZEROP IFF NLISTP)
(CONTRADICT 23)
```

```
(DROP 19 20 21 23)
(DIVE 1)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-0-UNWINDING-LEMMA)
(DIVE 1 1 1 1 1 1 1)
(REWRITE CALL-STATE2-STEP1-EFFECT)
UP
(REWRITE CALL-STATE2-STEP2-EFFECT)
UP
(REWRITE CALL-STATE2-STEP3-EFFECT)
UP
(REWRITE CALL-STATE2-STEP4-EFFECT)
UP
(REWRITE CALL-STATE2-STEP5-EFFECT)
UP
(CLAIM
 (EQUAL
  (CC
   (MG-MEANING
           (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)))
  'ROUTINEERROR)
 0)
(REWRITE CALL-STATE2-STEP6-EFFECT-ROUTINEERROR-BODY)
UP
(REWRITE CALL-STATE2-STEP7-EFFECT-ROUTINEERROR-BODY)
UP
(REWRITE CALL-STATE2-STEP8-EFFECT-ROUTINEERROR-BODY-EQUALS-FINAL)
UP S-PROP
(CLAIM
 (MEMBER
```

```
  (CC
   (MG-MEANING
           (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)))
  (DEF-CONDS (FETCH-CALLED-DEF STMT PROC-LIST)))
 0)
(REWRITE CALL-STATE2-STEP6-EFFECT-CALL-CONDS-BODY)
UP
(REWRITE CALL-STATE2-STEP7-EFFECT-CALL-CONDS-BODY)
UP
(REWRITE CALL-STATE2-STEP8-EFFECT-CALL-CONDS-BODY-EQUALS-FINAL)
UP S-PROP PROVE PROVE
(DEMOTE 16 19)
DROP PROVE
(REWRITE CALL-STATE2-STEP6-EFFECT-LOCAL-CONDS-BODY)
UP
(REWRITE CALL-STATE2-STEP7-EFFECT-LOCAL-CONDS-BODY)
UP
(REWRITE CALL-STATE2-STEP8-EFFECT-LOCAL-CONDS-BODY-EQUALS-FINAL)
TOP S-PROP
(DEMOTE 16 19)
DROP PROVE
(DEMOTE 16 19)
DROP PROVE
(DEMOTE 16 19)
DROP PROVE
(DIVE 1)
(REWRITE PROC-CALL-DOESNT-HALT2)
TOP S
(CONTRADICT 22)
(DROP 19 20 22)
(DIVE 1)
X
(= (CAR STMT) 'PROC-CALL-MG 0)
S
(DIVE 2 2 2 2 2 2 1)
(= * F 0)
TOP S
```

```
(DEMOTE 19)
(DIVE 1 1 1)
(REWRITE MG-MEANING-EQUIVALENCE)
TOP S
(DIVE 1)
(REWRITE PROC-CALL-DOESNT-HALT2)
TOP S
(CONTRADICT 20)
(DROP 20)
(DIVE 1)
(REWRITE CALL-STEP-INITIAL-TO-STATE2)
TOP S-PROP
(DEMOTE 19)
S-PROP SPLIT
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(DIVE 1)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
TOP S
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
(DIVE 1)
(REWRITE PROC-CALL-EXACT-TIME-HYPS)
TOP S)))
```

EVENT: Make the library `"c-proc-call2"`.

Index

all-cars-unique, 2–10, 17, 20, 24, 27, 29, 34, 38, 43, 47, 50, 53
all-pointers-bigger, 1
array-length, 3, 4

bindings, 7–11, 17–20, 22–32, 34, 36–38, 40, 41, 43, 44, 46, 47, 49–55
body-condition-member-make-cond-list, 31
body-condition-not-leave, 32

call-actuals, 6, 8, 9, 11, 18, 26–29, 36, 39, 41, 44, 49, 51, 52, 54
call-add1-lc-not-in-code, 20
call-conds, 19, 21, 25, 33, 35, 37, 40, 42, 45, 48
call-conds-index-lessp, 33
call-def-cond-label-find-labelp, 33
call-exact-time-schema-nonnormal-case, 56
call-exact-time-schema-normal-case, 56
call-lc-not-in-code, 23
call-state2-step5-effect, 17
call-state2-step6-effect-call-conds-body, 34
call-state2-step6-effect-local-conds-body, 47
call-state2-step6-effect-normal-body-equals-final, 20
call-state2-step6-effect-routinee rror-body, 24
call-state2-step7-effect-call-conds-body, 38
call-state2-step7-effect-local-conds-body, 50
call-state2-step7-effect-routinee rror-body, 26
call-state2-step8-effect-call-conds-body-equals-final, 42
call-state2-step8-effect-local-conds-body-equals-final, 52
call-state2-step8-effect-routinee rror-body-equals-final, 28
cc, 6, 7, 9, 10, 18, 19, 21–27, 29–32, 34–37, 39–55
code, 10, 17, 19–30, 34, 36, 38, 39, 41, 43–55
collect-pointers, 1
cond-case-jump-label-list, 19, 21, 25, 35, 48
cond-conversion, 33, 38, 42
cond-subsetp, 10, 17, 20, 24, 26, 28, 34, 38, 43, 47, 50, 53
convert-condition1, 42
convert-condition1-index-equivalence, 42
copy-out-params, 3, 4, 6
copy-out-params-restriction, 3
copy-out-params-restriction-cons, 3

data-length, 10, 17
data-param-lists-match, 5, 7
def-body, 6, 7, 9, 10, 18, 19, 21, 22, 24, 25, 27, 29, 31, 32, 34–37, 39–45, 47–54
def-cond-locals, 19, 21, 25, 36, 48
def-conds, 33–35, 37, 39–42, 44, 45, 48, 51, 54
def-formals, 6–9, 11
def-locals, 7–11, 17, 18, 26–29, 36, 39, 41, 44, 49, 51, 52, 54
deposit-alist-value, 2, 4
deposit-array-value, 1, 3, 4
deposit-array-value-deposit-alist-value-commute2, 4
deposit-array-value-doesnt-affect-map-down-values, 4
deposit-temp, 3, 4

deposit-temp-deposit-array-value
-commute6, 3
drop-formals-induction-hint, 1, 2
drop-locals-induction-hint, 2

extra-binding-doesnt-affect-cop
y-out-params, 4

fetch-called-def, 6–11, 17–19, 21, 22,
24–29, 31–45, 47–54
fetch-label, 22, 30, 42, 46, 55
find-def-conds-induction-hint, 32
find-def-conds-label1, 33
find-label, 19, 21, 23, 25, 30, 33, 36,
46, 48, 55
find-labelp, 20, 23, 33, 34
find-labelp-def-conds, 33
formal-types-preserved, 5, 6
formals-meaning-signature, 6

get, 19, 21, 25, 33, 36–38, 40, 42, 45,
46, 48
get-indexed-jump-instruction, 42
get-indexed-pop-global-instructi
on, 38
get-indexed-push-constant-instr
uction, 33
get-member-label-cnt-list, 46

index, 33, 34, 37, 40–42, 44–46

label-alist, 22, 30, 46, 55
label-cnt, 19–21, 23, 25, 34–36, 48
label-cnt-list, 19, 21, 25, 36, 46, 48
leave-not-in-make-cond-list, 32
length, 1, 3, 4, 8–11, 17–36, 38, 39,
41–55
listcars, 2–7, 9
locals-alist-mg-vars-ok, 8
locals-alist-mg-vars-ok0, 8
locals-meaning-signature, 7

make-call-param-alist, 6–8
make-call-var-alist, 6, 7, 9, 10, 18,
19, 21, 22, 24, 25, 27, 29,
31, 32, 35–37, 39–45, 47–
54
make-cond-list, 18, 19, 21, 22, 25,
26, 31, 32, 35, 36, 38, 40,
42, 45, 48–50, 52
map-call-formals, 2, 3, 6–9, 11
map-call-locals, 2, 3, 8, 9, 11
map-down-copy-facts, 5
map-down-copy-facts2, 5
map-down-copy-facts3, 5
map-down-copy-out-params-inducti
on-hint, 4, 5
map-down-copy-out-params-relati
on-new, 5
map-down-drop-locals-restrictio
n, 2
map-down-values, 1–4, 6, 11, 18, 19,
22, 23, 25–28, 30, 36, 37,
40, 41, 44, 46, 49, 51, 52,
54, 55
map-down-values-drop-formals-re
striction, 2
mg-alist, 6–11, 17–32, 34–55
mg-alistp, 1, 3, 4, 6, 7
mg-cond-to-p-nat, 18–23, 25, 26, 31,
33, 35–38, 40, 42, 45, 46,
48–50, 52, 55
mg-cond-to-p-nat-normal, 20
mg-cond-to-p-nat-routineerror, 23
mg-max-ctrl-stk-size, 18, 19, 22, 23,
26, 28, 30, 31, 36, 38, 40,
42, 45, 46, 49, 50, 52, 54,
55
mg-max-temp-stk-size, 18, 20, 22,
23, 26, 28, 30, 31, 36, 38,
40, 42, 45, 46, 49, 50, 52,
54, 55
mg-meaning, 7, 9, 11, 18, 19, 21,
22, 24–29, 31, 32, 35–37,
39–45, 47–54
mg-meaning-r, 10, 11, 17, 21–24, 27,
29–32, 34, 38, 43, 45–47,
50, 53–55
mg-psw, 6, 7, 9, 10, 18, 19, 21, 22,

24, 25, 27, 29, 31, 32, 35–37, 39–45, 47–54
mg-state, 6, 7, 9, 10, 18, 19, 21, 22, 24, 25, 27, 29, 31, 32, 35–37, 39–45, 47–54
mg-to-p-local-values, 8, 11
mg-vars-list-ok-in-p-state, 1–4, 6–8, 10, 17, 20, 24, 27, 29, 31, 32, 34, 38, 43, 47, 50, 53
mg-word-size, 18, 20, 22, 23, 26, 28, 30, 31, 36, 38, 40, 42, 45, 46, 49, 50, 52, 54, 55

no-duplicates, 2, 6
no-p-aliasing, 3, 4, 6, 9, 10, 17, 20, 24, 27, 29, 34, 38, 43, 47, 50, 53
no-p-aliasing-in-call-alists-new, 8
nonnormal-cond-conversion-not-normal, 42
normal, 10, 17, 20–22, 24, 27, 29–32, 34, 35, 38, 39, 43, 45, 47, 50, 51, 53, 54

ok-actual-params-list, 5, 7
ok-mg-def-plistp, 6–8, 10, 17, 20, 23, 24, 26, 28, 31–34, 38, 43, 47, 50, 53
ok-mg-formal-data-params-plistp, 6, 7
ok-mg-local-data-plistp, 8
ok-mg-statement, 6–8, 10, 17, 20, 23, 24, 26, 28, 31–34, 38, 42, 47, 50, 53
ok-mg-statep, 7, 8, 10, 17, 20, 24, 26, 28, 31–34, 38, 43, 47, 50, 53
ok-temp-stk-array-index, 1
ok-temp-stk-index, 1
ok-translation-parameters, 10, 17, 20, 23, 24, 26, 28, 33, 34, 38, 43, 47, 50, 53

p, 56
p-ctrl-stk-size, 10, 11, 17, 20–24, 26–32, 34, 38, 42, 43, 45–47, 50, 53–55
p-state, 18, 20, 22, 23, 26, 28, 30, 31, 36, 38, 41, 42, 45, 46, 49, 50, 52, 54, 55
p-step, 18, 22, 26, 28, 30, 36, 41, 45, 49, 52, 54
param-alist-mg-vars-ok, 7
param-alist-mg-vars-ok0, 7
plistp, 8, 10, 17, 20, 24, 27, 29, 34, 38, 43, 47, 50, 53
popn, 1, 11
popn-deposit-array-value, 1
popn-deposit-induction-hint, 1
popn-locals, 1
popn-locals1, 1
popn-rput, 1
push, 18, 26, 27, 37, 40, 49, 51
push-locals-values-code, 28, 29, 41, 44, 52, 54
push-parameters-code, 26, 27, 36, 39, 49, 51
put, 42, 45

resource-errorp, 10, 17, 21, 24, 27, 29, 31, 32, 34, 38, 43, 47, 50, 53
resources-inadequatep, 10, 17, 20, 24, 26, 28, 34, 38, 42, 47, 50, 53
restrict, 2, 3, 5–7, 9
restrict-cons, 3
ret-temp-stk-equals-final-temp-stk, 9
reverse, 8, 11
rput, 1

set-alist-value, 5
signatures-match, 7–10, 17, 20, 24, 27, 29, 31, 32, 34, 38, 43, 47, 50, 53
simple-mg-type-refp, 3, 4

tag, 18, 19, 21, 23, 25–30, 36, 37, 40, 41, 44, 46, 48, 49, 51, 52, 54, 55
top, 7–11, 17–20, 22–32, 34, 36–38, 40, 41, 43, 44, 46, 47, 49–55
translate, 10, 17, 19–25, 27, 29, 30, 34, 36, 38, 43, 45–48, 50, 53, 55
translate-def-body, 10, 17, 20, 24, 27, 29, 34, 38, 43, 47, 50, 53
translate-proc-list, 18, 19, 22, 23, 25, 26, 28, 30, 36, 37, 40, 41, 44, 46, 49, 51, 52, 54, 55

untag, 1, 19, 21, 25, 35, 48
user-defined-procp, 10, 17, 20, 24, 27, 29, 34, 38, 43, 47, 50, 53