

EVENT: Start with the library "c-prog2".

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;                                EXACT-TIME LOOP CASE
;;
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

THEOREM: loop-translation-2

```

(car (stmt) = 'loop-mg)
→ (translate (cinfo, cond-list, stmt, proc-list)
    = discard-label (add-code (translate (make-cinfo (append (code (cinfo),
                                                                list (list ('dl,
                                                                label-cnt (cinfo),
                                                                nil,
                                                                '(no-op))))),
                                                                cons (cons ('leave,
                                                                1 + label-cnt (cinfo)),
                                                                label-alist (cinfo)),
                                                                1 + (1 + label-cnt (cinfo))),
                                                                cond-list,
                                                                loop-body (stmt),
                                                                proc-list),
    list (list ('jump, label-cnt (cinfo)),
    list ('dl,
    1 + label-cnt (cinfo),
    nil,
    '(push-constant
    (nat 2))),
    '(pop-global c-c))))

```

THEOREM: loop-meaning-r-2

```

(car (stmt) = 'loop-mg)
→ (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)
    = if n ≈ 0 then signal-system-error (mg-state, 'timed-out)
      elseif ¬ normal (mg-state) then mg-state
      elseif resources-inadequatep (stmt, proc-list, sizes)
      then signal-system-error (mg-state, 'resource-error)
      else remove-leave (mg-meaning-r (stmt,
                                         proc-list,
                                         mg-meaning-r (loop-body (stmt),
                                                         proc-list,
                                                         mg-state,

```

$n - 1,$
 $sizes),$
 $n - 1,$
 $sizes))$ **endif**

EVENT: Disable loop-meaning-r-2.

THEOREM: loop-body-doesnt-halt

$((\text{car}(stmt) = \text{'loop-mg})$
 $\wedge \text{normal}(mg\text{-state})$
 $\wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, proc\text{-list}, mg\text{-state}, n, sizes))))$
 $\rightarrow (\text{mg-psw}(\text{mg-meaning-r}(\text{loop-body}(stmt), proc\text{-list}, mg\text{-state}, n - 1, sizes))$
 $= \text{'run})$

THEOREM: loop-sub1-body-doesnt-halt

$((\text{car}(stmt) = \text{'loop-mg})$
 $\wedge \text{normal}(mg\text{-state})$
 $\wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, proc\text{-list}, mg\text{-state}, n, sizes)))$
 $\wedge \text{normal}(\text{mg-meaning-r}(\text{loop-body}(stmt), proc\text{-list}, mg\text{-state}, n - 1, sizes)))$
 $\rightarrow (\text{mg-psw}(\text{mg-meaning-r}(stmt,$
 $\quad proc\text{-list},$
 $\quad \text{mg-meaning-r}(\text{loop-body}(stmt),$
 $\quad \quad proc\text{-list},$
 $\quad \quad mg\text{-state},$
 $\quad \quad n - 1,$
 $\quad \quad sizes),$
 $\quad n - 1,$
 $\quad sizes))$
 $= \text{'run})$

THEOREM: clock-equivalence-loop-case

$((\text{car}(stmt) = \text{'loop-mg})$
 $\wedge \text{normal}(mg\text{-state})$
 $\wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, proc\text{-list}, mg\text{-state}, n, sizes))))$
 $\rightarrow (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{loop-body}(stmt),$
 $\quad proc\text{-list},$
 $\quad mg\text{-state},$
 $\quad n - 1,$
 $\quad sizes))))$

THEOREM: loop-body-exact-time-hyps

$((n \neq 0)$
 $\wedge (\neg \text{resources-inadequatep}(stmt,$
 $\quad proc\text{-list},$

```

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

```

```

cons (list ('jump, label-cnt (cinfo)),
      cons (cons ('dl,
                  cons (1 + label-cnt (cinfo),
                        '(nil
                          (push-constant
                           (nat 2))))),
            cons ('(pop-global c-c),
                  code2))))))
 $\wedge$  ok-mg-statep (mg-state, cons ('leave, r-cond-list))
 $\wedge$  cond-subsetp (cons ('leave, r-cond-list), t-cond-list)
 $\wedge$  (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (make-cinfo (append (code (cinfo),
                                                         list (cons ('dl,
                                                                    cons (label-cnt (cinfo),
                                                                    '(nil
                                                                      (no-op))))))),
                                                         cons (cons ('leave,
                                                                    1 + label-cnt (cinfo),
                                                                    label-alist (cinfo),
                                                                    1 + (1 + label-cnt (cinfo))),
                                                                    t-cond-list,
                                                                    loop-body (stmt),
                                                                    proc-list))),
                                                         cons (list ('jump, label-cnt (cinfo)),
                                                                    cons (cons ('dl,
                                                                    cons (1 + label-cnt (cinfo),
                                                                    '(nil
                                                                      (push-constant
                                                                       (nat 2))))),
                                                                    cons ('(pop-global c-c), code2))))))
      user-defined-procp (subr, proc-list)
 $\wedge$  plistp (temp-stk)
 $\wedge$  listp (ctrl-stk)
 $\wedge$  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
 $\wedge$  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\wedge$  normal (mg-state)
 $\wedge$  all-cars-unique (mg-alist (mg-state))
 $\wedge$  ( $\neg$  resource-errorp (mg-meaning-r (loop-body (stmt),
                                           proc-list,
                                           mg-state,
                                           n - 1,
```

list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))

THEOREM: cond-list-superset-preserves-ok-mg-statement

(subset (*cond-list1*, *cond-list2*)
 $\wedge$ ok-mg-statement (*stmt*, *cond-list1*, *name-alist*, *proc-list*)
 $\rightarrow$ ok-mg-statement (*stmt*, *cond-list2*, *name-alist*, *proc-list*)

THEOREM: adding-leave-preserves-statement-okness-begin-case

ok-mg-statement (begin-body (*stmt*),
cons ('leave, append (when-labels (*stmt*), *cond-list*)),
name-alist,
proc-list)
 $\rightarrow$ ok-mg-statement (begin-body (*stmt*),
append (when-labels (*stmt*), cons ('leave, *cond-list*)),
name-alist,
proc-list)

THEOREM: adding-leave-preserves-statement-okness

ok-mg-statement (*stmt*, *cond-list*, *name-alist*, *proc-list*)
 $\rightarrow$ ok-mg-statement (*stmt*, cons ('leave, *cond-list*), *name-alist*, *proc-list*)

THEOREM: loop-sub1-body-exact-time-hyps

((*n* $\neq$ 0)
 $\wedge$ ($\neg$ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
 $\wedge$ (car (*stmt*) = 'loop-mg)
 $\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 $\wedge$ ok-mg-def-plistp (*proc-list*)
 $\wedge$ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
 $\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
 $\wedge$ cond-subsetp (*r-cond-list*, *t-cond-list*)
 $\wedge$ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
 $\wedge$ user-defined-procp (*subr*, *proc-list*)
 $\wedge$ plistp (*temp-stk*)
 $\wedge$ listp (*ctrl-stk*)
 $\wedge$ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
 $\wedge$ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
 $\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*)

$$\begin{aligned}
& \wedge \text{normal}(mg\text{-state}) \\
& \wedge \text{all-cars-unique}(mg\text{-alist}(mg\text{-state})) \\
& \wedge (\neg \text{resource-errorp}(mg\text{-meaning-r}(stmt, \\
& \qquad \qquad \qquad proc\text{-list}, \\
& \qquad \qquad \qquad mg\text{-state}, \\
& \qquad \qquad \qquad n, \\
& \qquad \qquad \qquad \text{list}(\text{length}(temp\text{-stk}), \\
& \qquad \qquad \qquad \text{p-ctrl-stk-size}(ctrl\text{-stk})))))) \\
& \wedge \text{normal}(mg\text{-meaning-r}(\text{loop-body}(stmt), \\
& \qquad \qquad \qquad proc\text{-list}, \\
& \qquad \qquad \qquad mg\text{-state}, \\
& \qquad \qquad \qquad n - 1, \\
& \qquad \qquad \qquad \text{list}(\text{length}(temp\text{-stk}), \text{p-ctrl-stk-size}(ctrl\text{-stk})))))) \\
\rightarrow & (\text{ok-mg-statement}(stmt, \text{cons}('leave, r\text{-cond-list}), name\text{-alist}, proc\text{-list}) \\
& \wedge \text{ok-mg-def-plistp}(proc\text{-list}) \\
& \wedge \text{ok-translation-parameters}(cinfo, \\
& \qquad \qquad \qquad t\text{-cond-list}, \\
& \qquad \qquad \qquad stmt, \\
& \qquad \qquad \qquad proc\text{-list}, \\
& \qquad \qquad \qquad code2) \\
& \wedge \text{ok-mg-statep}(mg\text{-meaning-r}(\text{loop-body}(stmt), \\
& \qquad \qquad \qquad proc\text{-list}, \\
& \qquad \qquad \qquad mg\text{-state}, \\
& \qquad \qquad \qquad n - 1, \\
& \qquad \qquad \qquad \text{list}(\text{length}(temp\text{-stk}), \\
& \qquad \qquad \qquad \text{p-ctrl-stk-size}(ctrl\text{-stk}))), \\
& \qquad \qquad \qquad \text{cons}('leave, r\text{-cond-list})) \\
& \wedge \text{cond-subsetp}(\text{cons}('leave, r\text{-cond-list}), t\text{-cond-list}) \\
& \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(subr, proc\text{-list}), proc\text{-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(cinfo, \\
& \qquad \qquad \qquad t\text{-cond-list}, \\
& \qquad \qquad \qquad stmt, \\
& \qquad \qquad \qquad proc\text{-list})), \\
& \qquad \qquad \qquad code2)) \\
& \wedge \text{user-defined-procp}(subr, proc\text{-list}) \\
& \wedge \text{plistp}(temp\text{-stk}) \\
& \wedge \text{listp}(ctrl\text{-stk}) \\
& \wedge \text{mg-vars-list-ok-in-p-state}(mg\text{-alist}(mg\text{-meaning-r}(\text{loop-body}(stmt), \\
& \qquad \qquad \qquad proc\text{-list}, \\
& \qquad \qquad \qquad mg\text{-state}, \\
& \qquad \qquad \qquad n - 1, \\
& \qquad \qquad \qquad \text{list}(\text{length}(temp\text{-stk}), \\
& \qquad \qquad \qquad \text{p-ctrl-stk-size}(ctrl\text{-stk}))))), \\
& \qquad \qquad \qquad \text{bindings}(\text{top}(ctrl\text{-stk}))),
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{ no-p-aliasing } (\text{bindings } (\text{top } (ctrl\text{-}stk)), \\
& \quad \text{mg-alist } (\text{mg-meaning-r } (\text{loop-body } (stmt), \\
& \quad \quad \text{proc-list}, \\
& \quad \quad \text{mg-state}, \\
& \quad \quad n - 1, \\
& \quad \quad \text{list } (\text{length } (temp\text{-}stk), \\
& \quad \quad \quad \text{p-ctrl-stk-size } (ctrl\text{-}stk)))))) \\
& \wedge \text{ signatures-match } (\text{mg-alist } (\text{mg-meaning-r } (\text{loop-body } (stmt), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{list } (\text{length } (temp\text{-}stk), \\
& \quad \quad \text{p-ctrl-stk-size } (ctrl\text{-}stk))))), \\
& \quad \quad \quad \text{name-alist}) \\
& \wedge \text{ normal } (\text{mg-meaning-r } (\text{loop-body } (stmt), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{list } (\text{length } (temp\text{-}stk), \\
& \quad \quad \text{p-ctrl-stk-size } (ctrl\text{-}stk)))))) \\
& \wedge \text{ all-cars-unique } (\text{mg-alist } (\text{mg-meaning-r } (\text{loop-body } (stmt), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{list } (\text{length } (temp\text{-}stk), \\
& \quad \quad \text{p-ctrl-stk-size } (ctrl\text{-}stk)))))) \\
& \wedge (\neg \text{ resource-errorp } (\text{mg-meaning-r } (stmt, \\
& \quad \text{proc-list}, \\
& \quad \text{mg-meaning-r } (\text{loop-body } (stmt), \\
& \quad \quad \text{proc-list}, \\
& \quad \quad \text{mg-state}, \\
& \quad \quad n - 1, \\
& \quad \quad \text{list } (\text{length } (temp\text{-}stk), \\
& \quad \quad \quad \text{p-ctrl-stk-size } (ctrl\text{-}stk))))), \\
& \quad \quad \quad n - 1, \\
& \quad \quad \text{list } (\text{length } (temp\text{-}stk), \\
& \quad \quad \quad \text{p-ctrl-stk-size } (ctrl\text{-}stk))))))
\end{aligned}$$

THEOREM: loop-clock-nonnormal-nonleave

$$\begin{aligned}
& ((\text{car } (stmt) = \text{'loop-mg}) \\
& \wedge (n \neq 0) \\
& \wedge \text{ normal } (mg\text{-state}) \\
& \wedge (\neg \text{ resource-errorp } (\text{mg-meaning-r } (stmt, \text{proc-list}, \text{mg-state}, n, \text{sizes})))
\end{aligned}$$

$$\begin{aligned}
& \wedge (\neg \text{normal}(\text{mg-meaning-r}(\text{loop-body}(stmt), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{sizes}))) \\
& \wedge (\text{cc}(\text{mg-meaning-r}(\text{loop-body}(stmt), \text{proc-list}, \text{mg-state}, n - 1, \text{sizes})) \\
& \quad \neq \text{'leave})) \\
& \rightarrow (\text{clock}(stmt, \text{proc-list}, \text{mg-state}, n) \\
& \quad = (1 + \text{clock}(\text{loop-body}(stmt), \text{proc-list}, \text{mg-state}, n - 1)))
\end{aligned}$$

THEOREM: loop-clock-normal

$$\begin{aligned}
& ((\text{car}(stmt) = \text{'loop-mg}) \\
& \wedge (n \neq 0) \\
& \wedge \text{normal}(\text{mg-state}) \\
& \wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, \text{proc-list}, \text{mg-state}, n, \text{sizes}))) \\
& \wedge \text{normal}(\text{mg-meaning-r}(\text{loop-body}(stmt), \text{proc-list}, \text{mg-state}, n - 1, \text{sizes}))) \\
& \rightarrow (\text{clock}(stmt, \text{proc-list}, \text{mg-state}, n) \\
& \quad = (1 + ((1 + \text{clock}(\text{loop-body}(stmt), \text{proc-list}, \text{mg-state}, n - 1)) \\
& \quad \quad + \text{clock}(stmt, \\
& \quad \quad \text{proc-list}, \\
& \quad \quad \text{mg-meaning-r}(\text{loop-body}(stmt), \\
& \quad \quad \text{proc-list}, \\
& \quad \quad \text{mg-state}, \\
& \quad \quad n - 1, \\
& \quad \quad \text{sizes}), \\
& \quad \quad n - 1))))
\end{aligned}$$

THEOREM: loop-clock-nonnormal-leave

$$\begin{aligned}
& ((\text{car}(stmt) = \text{'loop-mg}) \\
& \wedge (n \neq 0) \\
& \wedge \text{normal}(\text{mg-state}) \\
& \wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, \text{proc-list}, \text{mg-state}, n, \text{sizes}))) \\
& \wedge (\neg \text{normal}(\text{mg-meaning-r}(\text{loop-body}(stmt), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{sizes}))) \\
& \wedge (\text{cc}(\text{mg-meaning-r}(\text{loop-body}(stmt), \text{proc-list}, \text{mg-state}, n - 1, \text{sizes})) \\
& \quad = \text{'leave})) \\
& \rightarrow (\text{clock}(stmt, \text{proc-list}, \text{mg-state}, n) \\
& \quad = (3 + \text{clock}(\text{loop-body}(stmt), \text{proc-list}, \text{mg-state}, n - 1)))
\end{aligned}$$

THEOREM: loop-step-initial-equals-state1

$$\begin{aligned}
& ((\text{car}(stmt) = \text{'loop-mg}) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, \text{name-alist}, \text{proc-list})
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{ok-mg-def-plistp} (proc\text{-}list) \\
& \wedge \text{ok-translation-parameters} (cinfo, t\text{-}cond\text{-}list, stmt, proc\text{-}list, code2) \\
& \wedge (\text{code} (\text{translate-def-body} (\text{assoc} (subr, proc\text{-}list), proc\text{-}list)) \\
& \quad = \text{append} (\text{code} (\text{translate} (cinfo, t\text{-}cond\text{-}list, stmt, proc\text{-}list)), \\
& \quad \quad \quad code2)) \\
& \wedge \text{user-defined-procp} (subr, proc\text{-}list) \\
\rightarrow & (\text{p-step} (\text{map-down} (mg\text{-}state, \\
& \quad \quad \quad proc\text{-}list, \\
& \quad \quad \quad ctrl\text{-}stk, \\
& \quad \quad \quad temp\text{-}stk, \\
& \quad \quad \quad \text{tag} ('pc, \text{cons} (subr, \text{length} (\text{code} (cinfo))))), \\
& \quad \quad \quad t\text{-}cond\text{-}list)) \\
& = \text{map-down} (mg\text{-}state, \\
& \quad \quad \quad proc\text{-}list, \\
& \quad \quad \quad ctrl\text{-}stk, \\
& \quad \quad \quad temp\text{-}stk, \\
& \quad \quad \quad \text{tag} ('pc, \\
& \quad \quad \quad \text{cons} (subr, \\
& \quad \quad \quad \quad \text{length} (\text{code} (\text{make-cinfo} (\text{append} (\text{code} (cinfo), \\
& \quad \quad \quad \quad \quad \text{list} (\text{cons} ('dl, \\
& \quad \quad \quad \quad \quad \quad \text{cons} (\text{label-cnt} (cinfo), \\
& \quad \quad \quad \quad \quad \quad \quad '(\text{nil} \\
& \quad \quad \quad \quad \quad \quad \quad \quad (\text{no-op})))))), \\
& \quad \quad \quad \text{cons} (\text{cons} ('leave, \\
& \quad \quad \quad \quad 1 + \text{label-cnt} (cinfo)), \\
& \quad \quad \quad \quad \text{label-alist} (cinfo)), \\
& \quad \quad \quad 1 + (1 + \text{label-cnt} (cinfo)))))), \\
& \quad \quad \quad t\text{-}cond\text{-}list))
\end{aligned}$$

THEOREM: nonleave-nonnormal-body-meaning-preserved

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge \text{normal} (mg\text{-}state) \\
& \wedge (\text{car} (stmt) = 'loop\text{-}mg) \\
& \wedge (\neg \text{resource-errorp} (\text{mg-meaning-r} (stmt, proc\text{-}list, mg\text{-}state, n, sizes))) \\
& \wedge (\neg \text{normal} (\text{mg-meaning-r} (\text{loop-body} (stmt), \\
& \quad \quad \quad proc\text{-}list, \\
& \quad \quad \quad mg\text{-}state, \\
& \quad \quad \quad n - 1, \\
& \quad \quad \quad sizes))) \\
& \wedge (\text{cc} (\text{mg-meaning-r} (\text{loop-body} (stmt), proc\text{-}list, mg\text{-}state, n - 1, sizes)) \\
& \quad \neq 'leave)) \\
\rightarrow & (\text{mg-meaning-r} (stmt, proc\text{-}list, mg\text{-}state, n, sizes) \\
& \quad = \text{mg-meaning-r} (\text{loop-body} (stmt), proc\text{-}list, mg\text{-}state, n - 1, sizes))
\end{aligned}$$

THEOREM: loop-code-rewrite

```

(car (stmt) = 'loop-mg)
→ (append (code (translate (cinfo, t-cond-list, stmt, proc-list)), code2)
    = append (code (translate (make-cinfo (append (code (cinfo),
                                                    list (cons ('dl,
                                                                cons (label-cnt (cinfo),
                                                                    '(nil
                                                                      (no-op))))))),
                                                    cons (cons ('leave,
                                                                1 + label-cnt (cinfo)),
                                                                label-alist (cinfo)),
                                                                1 + (1 + label-cnt (cinfo))),
                                                    t-cond-list,
                                                    loop-body (stmt),
                                                    proc-list)),
              cons (list ('jump, label-cnt (cinfo)),
                    cons (cons ('dl,
                                cons (1 + label-cnt (cinfo),
                                    '(nil
                                      (push-constant (nat 2))))),
                          cons ('(pop-global c-c), code2))))))

```

```

(prove-lemma loop-nonnormal-nonleave-state2-equals-final (rewrite)
  (IMPLIES
    (AND (NOT (ZEROP N))
      (equal (car STMT) 'LOOP-MG)
      (NORMAL MG-STATE)
      (not (resource-errorp (mg-meaning-r stmt proc-list mg-state n
        (list (length temp-stk)
              (p-ctrl-stk-size ctrl-stk))))))
      (not (NORMAL (MG-MEANING-R (LOOP-BODY STMT) proc-list MG-STATE (sub1 n)
        (list (length temp-stk) (p-ctrl-stk-size ctrl-stk))))))
      (not (equal (cc (MG-MEANING-R (LOOP-BODY STMT) proc-list MG-STATE (sub1 N)
        (list (length temp-stk) (p-ctrl-stk-size ctrl-stk))))
        'leave)))
    (equal
      (P-STATE
        (TAG 'PC
          (CONS SUBR
            (IF
              (NORMAL (MG-MEANING-R (LOOP-BODY STMT)
                PROC-LIST MG-STATE
                (SUB1 N)

```

```

      (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
      (LENGTH
      (CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
' (NIL (NO-OP)))))
      (CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
      (LABEL-ALIST CINFO))
      (ADD1 (ADD1 (LABEL-CNT CINFO)))))
T-COND-LIST
      (LOOP-BODY STMT)
PROC-LIST)))
      (FIND-LABEL
      (FETCH-LABEL
(CC (MG-MEANING-R (LOOP-BODY STMT)
      PROC-LIST MG-STATE
      (SUB1 N)
      (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LABEL-ALIST
      (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
' (NIL (NO-OP)))))
      (CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
      (LABEL-ALIST CINFO))
      (ADD1 (ADD1 (LABEL-CNT CINFO)))))
T-COND-LIST
      (LOOP-BODY STMT)
PROC-LIST)))
      (APPEND
(CODE
      (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
' (NIL (NO-OP)))))
      (CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
      (LABEL-ALIST CINFO))
      (ADD1 (ADD1 (LABEL-CNT CINFO)))))

```

```

T-COND-LIST
  (LOOP-BODY STMT)
  PROC-LIST))
(CONS (LIST 'JUMP (LABEL-CNT CINFO))
      (CONS (CONS 'DL
                  (CONS (ADD1 (LABEL-CNT CINFO))
                        '(NIL (PUSH-CONSTANT (NAT 2))))))
      (CONS '(POP-GLOBAL C-C) CODE2))))))
CTRL-STK
(MAP-DOWN-VALUES
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
                        PROC-LIST MG-STATE
                        (SUB1 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 (LOOP-BODY STMT)
                                           PROC-LIST MG-STATE
                                           (SUB1 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)
(P-STATE;; final
 (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 PROMOTE
  (= (MG-MEANING-R STMT PROC-LIST MG-STATE N
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))
    (MG-MEANING-R (LOOP-BODY STMT)
      PROC-LIST MG-STATE
      (SUB1 N)
      (LIST (LENGTH TEMP-STK)
        (P-CTRL-STK-SIZE CTRL-STK)))
    0)
  S
  (S LEMMAS)
  (ENABLE DISCARD-LABEL ADD-CODE)
  S
  (DIVE 2 2 2 1 1 2 1)
  TOP
  (S LEMMAS)
  (DIVE 1 2 2 1 1)
  X TOP S
  (DIVE 1)
  (REWRITE NONLEAVE-NONNORMAL-BODY-MEANING-PRESERVED)
  TOP S)))

```

THEOREM: loop-leave-body-meaning-preserved
 $((n \neq 0)$

```

 $\wedge$  normal(mg-state)
 $\wedge$  (car(stmt) = 'loop-mg)
 $\wedge$  ( $\neg$  resource-errorp(mg-meaning-r(stmt, proc-list, mg-state, n, sizes)))
 $\wedge$  (cc(mg-meaning-r(loop-body(stmt), proc-list, mg-state, n - 1, sizes))
    = 'leave))
 $\rightarrow$  (mg-meaning-r(stmt, proc-list, mg-state, n, sizes)
    = set-condition(mg-meaning-r(loop-body(stmt),
                                     proc-list,
                                     mg-state,
                                     n - 1,
                                     sizes),
                    'normal))

```

THEOREM: loop-find-labelp-lemma1

```

((car(stmt) = 'loop-mg)
  $\wedge$  ok-mg-statement(stmt, r-cond-list, name-alist, proc-list)
  $\wedge$  ok-translation-parameters(cinfo, t-cond-list, stmt, proc-list, code2))
 $\rightarrow$  ( $\neg$  find-labelp(1 + label-cnt(cinfo),
    code(translate(make-cinfo(append(code(cinfo),
                                     list(cons('dl,
                                               cons(label-cnt(cinfo),
                                               '(nil
                                               (no-op)))))),
    cons(cons('leave,
              1 + label-cnt(cinfo)),
          label-alist(cinfo)),
          1 + (1 + label-cnt(cinfo))),
    t-cond-list,
    loop-body(stmt),
    proc-list))))

```

THEOREM: loop-find-labelp-lemma2

```

((car(stmt) = 'loop-mg)
  $\wedge$  ok-mg-statement(stmt, r-cond-list, name-alist, proc-list)
  $\wedge$  ok-translation-parameters(cinfo, t-cond-list, stmt, proc-list, code2))
 $\rightarrow$  ( $\neg$  find-labelp(label-cnt(cinfo), code(cinfo)))

```

```

(prove-lemma loop-leave-state2-step1-effect (rewrite)
  (IMPLIES
    (AND (NOT (ZEROP N))
      (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
        (LIST (LENGTH TEMP-STK)
          (P-CTRL-STK-SIZE CTRL-STK)))))

```

```

(EQUAL (CAR STMT) 'LOOP-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 (cc (mg-meaning-r (loop-body stmt) proc-list mg-state (sub1 n)
(list (length temp-stk)
(p-ctrl-stk-size ctrl-stk)))))
'leave))
(equal
(p-step
(P-STATE
(TAG 'PC
(CONS SUBR
(IF (NORMAL (MG-MEANING-R (LOOP-BODY STMT) PROC-LIST MG-STATE (SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))))
(LENGTH
(CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
(CONS (LABEL-CNT CINFO)
'(NIL (NO-OP)))))
(CONS (CONS 'LEAVE

```

```

      (ADD1 (LABEL-CNT CINFO)))
      (LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO))))
  T-COND-LIST
  (LOOP-BODY STMT)
  PROC-LIST)))
(FIND-LABEL
 (FETCH-LABEL
  (CC (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))
  (LABEL-ALIST
   (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
    (LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
        '(NIL (NO-OP)))))
    (CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
      (LABEL-ALIST CINFO))
    (ADD1 (ADD1 (LABEL-CNT CINFO))))
    T-COND-LIST
    (LOOP-BODY STMT)
    PROC-LIST)))
  (APPEND
   (CODE
    (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
    (LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
        '(NIL (NO-OP)))))
    (CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
      (LABEL-ALIST CINFO))
    (ADD1 (ADD1 (LABEL-CNT CINFO))))
    T-COND-LIST
    (LOOP-BODY STMT)
    PROC-LIST)))
    (CONS (LIST 'JUMP (LABEL-CNT CINFO))
    (CONS (CONS 'DL
      (CONS (ADD1 (LABEL-CNT CINFO))
        '(NIL (PUSH-CONSTANT (NAT 2)))))
      (CONS '(POP-GLOBAL C-C) CODE2))))))
  CTRL-STK

```

```

(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 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 (LOOP-BODY STMT)
        PROC-LIST MG-STATE
        (SUB1 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))
    (P-STATE
      (TAG 'PC
        (CONS SUBR
          (PLUS
            (LENGTH
              (CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                (LIST (CONS 'DL
                  (CONS (LABEL-CNT CINFO)
                    '(NIL (NO-OP))))))
                (CONS (CONS 'LEAVE
                  (ADD1 (LABEL-CNT CINFO)))
                  (LABEL-ALIST CINFO))
                (ADD1 (ADD1 (LABEL-CNT CINFO))))
                  T-COND-LIST
                  (LOOP-BODY STMT)
                  PROC-LIST)))
            2)))
            CTRL-STK
            (PUSH '(NAT 2)
              (MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
                PROC-LIST MG-STATE
                (SUB1 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 'LEAVE T-COND-LIST)))
(MG-MAX-CTRL-STK-SIZE)
(MG-MAX-TEMP-STK-SIZE)
(MG-WORD-SIZE)
'RUN)))
((INSTRUCTIONS PROMOTE
  (DIVE 1)
  S
  (= (CC (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (CONS (LENGTH TEMP-STK)
      (CONS (P-CTRL-STK-SIZE CTRL-STK)
        'NIL))))
    'LEAVE
    0)
  S
  (S LEMMAS)
  (DIVE 1 1 1 2 2)
  (REWRITE FIND-LABEL-APPEND)
  (DIVE 2)
  X
  (DIVE 1)
  X UP TOP
  (DIVE 1)
  S
  (S LEMMAS)
  (DIVE 1 1 2)
  (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (DIVE 1 1)
  (REWRITE LOOP-TRANSLATION-2)
  UP
  (S LEMMAS)
  UP UP
  (S LEMMAS)
  (REWRITE GET-LENGTH-PLUS)
  X UP
  (S LEMMAS)
  UP X
  (DIVE 1)

```

```

X
(DIVE 1)
(REWRITE MAP-DOWN-VALUES-PRESERVES-LENGTH)
UP
(REWRITE RESOURCES-ADEQUATE-TEMP-STK-NOT-MAX)
UP S X
(S LEMMAS)
(DIVE 1 2 2 1)
(REWRITE FIND-LABEL-APPEND)
(DIVE 2)
X
(DIVE 1)
X TOP
(S LEMMAS)
(DIVE 1)
(REWRITE LOOP-FIND-LABELP-LEMMA1)
TOP S
(DIVE 1 1)
(REWRITE MG-MEANING-EQUIVALENCE)
TOP
(REWRITE SIGNATURES-MATCH-PRESERVES-MG-VARS-LIST-OK
  (($X (MG-ALIST MG-STATE))))
(REWRITE MG-MEANING-PRESERVES-SIGNATURES-MATCH)
(REWRITE OK-MG-STATEP-ALIST-PLISTP)
(PROVE (ENABLE LOOP-BODY-DOESNT-HALT))
(DIVE 1 1)
(REWRITE MG-MEANING-EQUIVALENCE)
TOP
(REWRITE MG-MEANING-PRESERVES-MG-ALISTP
  (($R-COND-LIST (CONS 'LEAVE R-COND-LIST))))
(REWRITE LOOP-BODY-EXACT-TIME-HYPS)
(REWRITE OK-LOOP-STATEMENT)
(PROVE (ENABLE LOOP-BODY-DOESNT-HALT))
(DIVE 1)
(REWRITE LOOP-FIND-LABELP-LEMMA1)
TOP S)))

```

```

(prove-lemma loop-leave-state2-step2-effect (rewrite)
  (IMPLIES
    (AND (NOT (ZEROP N))
      (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
        (LIST (LENGTH TEMP-STK)

```

```

(P-CTRL-STK-SIZE CTRL-STK))))
(EQUAL (CAR STMT) 'LOOP-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 (cc (mg-meaning-r (loop-body stmt) proc-list mg-state (sub1 n)
(list (length temp-stk)
(p-ctrl-stk-size ctrl-stk))))
'leave))
(equal
(p-step
(P-STATE
(TAG 'PC
(CONS SUBR
(PLUS
(LENGTH
(CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
(CONS (LABEL-CNT CINFO)
' (NIL (NO-OP)))))))
(CONS (CONS 'LEAVE
(ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))

```

```

      (ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST)))
  2)))
  CTRL-STK
  (PUSH '(NAT 2)
(MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 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 'LEAVE T-COND-LIST)))
  (MG-MAX-CTRL-STK-SIZE)
  (MG-MAX-TEMP-STK-SIZE)
  (MG-WORD-SIZE)
  'RUN))
  (P-STATE;; final
  (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 PROMOTE
  (DIVE 1)
  X
  (S LEMMAS)
  (DIVE 1 1 2)
  (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (REWRITE LOOP-CODE-REWRITE)
  UP
  (REWRITE GET-LENGTH-PLUS)
  X X UP X UP X
  (DIVE 1)
  X UP S-PROP X
  (S LEMMAS)
  S UP S
  (= (MG-MEANING-R STMT PROC-LIST MG-STATE N
                    (LIST (LENGTH TEMP-STK)
                          (P-CTRL-STK-SIZE CTRL-STK)))
     (SET-CONDITION (MG-MEANING-R (LOOP-BODY STMT)
                                   PROC-LIST MG-STATE
                                   (SUB1 N)
                                   (LIST (LENGTH TEMP-STK)
                                         (P-CTRL-STK-SIZE CTRL-STK))))
     'NORMAL)
  0)
  SPLIT
  (ENABLE MG-COND-TO-P-NAT)
  S S
  (ENABLE LENGTH-CONS)
  (S LEMMAS)
  S
  (DIVE 1)
  (REWRITE LOOP-LEAVE-BODY-MEANING-PRESERVED)
  TOP S)))

```

```

(prove-lemma loop-normal-body-step-state2-equals-state3 (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) 'LOOP-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))))))
      (normal (mg-meaning-r (loop-body stmt) proc-list mg-state (sub1 n)
        (list (length temp-stk)
          (p-ctrl-stk-size ctrl-stk))))))
    (equal
      (p-step
        (P-STATE
          (TAG 'PC
            (CONS SUBR
              (IF
                (NORMAL (MG-MEANING-R (LOOP-BODY STMT) PROC-LIST MG-STATE (SUB1 N)
                  (LIST (LENGTH TEMP-STK)
                    (P-CTRL-STK-SIZE CTRL-STK))))))

```

```

        (LENGTH
        (CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
        (LIST (CONS 'DL
        (CONS (LABEL-CNT CINFO)
' (NIL (NO-OP))))))
        (CONS (CONS 'LEAVE
        (ADD1 (LABEL-CNT CINFO)))
        (LABEL-ALIST CINFO))
        (ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST)))
        (FIND-LABEL
        (FETCH-LABEL
(CC (MG-MEANING-R (LOOP-BODY STMT)
        PROC-LIST MG-STATE
        (SUB1 N)
        (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LABEL-ALIST
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
        (CONS (LABEL-CNT CINFO)
' (NIL (NO-OP))))))
(CONS (CONS 'LEAVE
        (ADD1 (LABEL-CNT CINFO)))
        (LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST)))
        (APPEND
(CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
        (CONS (LABEL-CNT CINFO)
' (NIL (NO-OP))))))
(CONS (CONS 'LEAVE
        (ADD1 (LABEL-CNT CINFO)))
        (LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)

```

```

PROC-LIST))
(CONS (LIST 'JUMP (LABEL-CNT CINFO))
      (CONS (CONS 'DL
                  (CONS (ADD1 (LABEL-CNT CINFO))
                        '(NIL (PUSH-CONSTANT (NAT 2))))))
      (CONS '(POP-GLOBAL C-C) CODE2))))))
CTRL-STK
(MAP-DOWN-VALUES
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
                        PROC-LIST MG-STATE
                        (SUB1 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 (LOOP-BODY STMT)
                                             PROC-LIST MG-STATE
                                             (SUB1 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))
 (MAP-DOWN (MG-MEANING-R (LOOP-BODY STMT);; state3
                        PROC-LIST MG-STATE
                        (SUB1 N)
                        (LIST (LENGTH TEMP-STK)
                              (P-CTRL-STK-SIZE CTRL-STK))))
 (PROC-LIST CTRL-STK TEMP-STK
  (TAG 'PC
        (CONS SUBR (LENGTH (CODE CINFO))))
  T-COND-LIST)))
((INSTRUCTIONS
  PROMOTE (DIVE 1) S X (S LEMMAS) (DIVE 1 1 2) (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (REWRITE LOOP-CODE-REWRITE) UP (REWRITE GET-LENGTH-CAR) S UP (ENABLE LABELLEDP)
  X UP X (DIVE 1) X UP S-PROP X (S LEMMAS) (DIVE 1 2 1) (REWRITE DEFINEDP-CAR-ASSOC)
  NX (DIVE 2) (REWRITE TRANSLATE-DEF-BODY-REWRITE) (REWRITE LOOP-CODE-REWRITE) (DIVE 1
  (REWRITE NEW-CODE-APPENDED-TO-OLD1) UP UP (S LEMMAS) (REWRITE FIND-LABEL-APPEND)
  (DIVE 2) X UP (REWRITE PLUS-0-REWRITE) S TOP S (DIVE 1)

```

(REWRITE LOOP-FIND-LABELP-LEMMA2) TOP S PROVE (REWRITE CAR-DEFINEDP-DEFINED-PROCP)))

THEOREM: loop-never-leave

((car (stmt) = 'loop-mg) ∧ normal (mg-state))
 → (cc (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)) ≠ 'leave)

THEOREM: loop-normal-body-state4-equals-final

((n ≠ 0)
 ∧ (¬ resources-inadequatep (stmt,
 proc-list,
 list (length (temp-stk),
 p-ctrl-stk-size (ctrl-stk))))
 ∧ (car (stmt) = 'loop-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-r (loop-body (stmt),
 proc-list,
 mg-state,
 n - 1,
 list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))))
 → (p-state (tag ('pc,
 cons (subr,

```

if normal (mg-meaning-r (stmt,
                        proc-list,
                        mg-meaning-r (loop-body (stmt),
                        proc-list,
                        mg-state,
                        n - 1,
                        list (length (temp-stk),
                        p-ctrl-stk-size (ctrl-stk))),
                        n - 1,
                        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-meaning-r (loop-body (stmt),
                        proc-list,
                        mg-state,
                        n - 1,
                        list (length (temp-stk),
                        p-ctrl-stk-size (ctrl-stk))),
                        n - 1,
                        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-meaning-r (loop-body (stmt),
                        proc-list,
                        mg-state,
                        n - 1,
                        list (length (temp-stk),
                        p-ctrl-stk-size (ctrl-stk))),

```

```

                                n - 1,
                                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-meaning-r (loop-body (stmt),
                                                                proc-list,
                                                                mg-state,
                                                                n - 1,
                                                                list (length (temp-stk),
                                                                      p-ctrl-stk-size (ctrl-stk))),
                                                                n - 1,
                                                                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)
=  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))),
                                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: loop-nonnormal-nonleave-exact-time-schema

```

((stmt-time = (1 + body-time))
 ∧ (p-step (initial) = state1)
 ∧ (p (state1, body-time) = state2)
 ∧ (state2 = final))
→ (p (initial, stmt-time) = final)

```

THEOREM: loop-leave-body-exact-time-schema

```

((stmt-time = (3 + body-time))
 ∧ (p-step (initial) = state1)
 ∧ (p (state1, body-time) = state2)
 ∧ (p (state2, 2) = final))
→ (p (initial, stmt-time) = final)

```

THEOREM: loop-normal-body-exact-time-schema

```

((stmt-time = (1 + ((1 + body-time) + loop-sub1-time)))

```

$$\begin{aligned}
& \wedge \text{ (p (state1, body-time) = state2)} \\
& \wedge \text{ (p (state3, loop-sub1-time) = state4)} \\
& \wedge \text{ (p-step (initial) = state1)} \\
& \wedge \text{ (p-step (state2) = state3)} \\
& \wedge \text{ (state4 = final)} \\
& \rightarrow \text{ (p (initial, stmt-time) = final)}
\end{aligned}$$

```

(prove-lemma loop-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) 'LOOP-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 (LOOP-BODY STMT)
          (CONS 'LEAVE R-COND-LIST)

```

```

NAME-ALIST PROC-LIST)
(OK-MG-DEF-PLISTP PROC-LIST)
(OK-TRANSLATION-PARAMETERS
 (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
      '(NIL (NO-OP)))))))
(CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST
(CONS (LIST 'JUMP (LABEL-CNT CINFO))
(CONS (CONS 'DL
(CONS (ADD1 (LABEL-CNT CINFO))
      '(NIL (PUSH-CONSTANT (NAT 2))))))
(CONS '(POP-GLOBAL C-C) CODE2))))
(OK-MG-STATEP MG-STATE
(CONS 'LEAVE R-COND-LIST))
(COND-SUBSETP (CONS 'LEAVE R-COND-LIST)
T-COND-LIST)
(EQUAL
 (CODE (TRANSLATE-DEF-BODY (ASSOC SUBR PROC-LIST)
PROC-LIST))
 (APPEND
(CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
      '(NIL (NO-OP)))))))
(CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST))
(CONS (LIST 'JUMP (LABEL-CNT CINFO))
      (CONS (CONS 'DL
      (CONS (ADD1 (LABEL-CNT CINFO))
      '(NIL (PUSH-CONSTANT (NAT 2))))))
      (CONS '(POP-GLOBAL C-C) 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 (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))))
        (EQUAL
(P
(MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK           ;; state1
(TAG 'PC
(CONS SUBR
(LENGTH (CODE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
(CONS (LABEL-CNT CINFO)
'(NIL (NO-OP))))))
(CONS (CONS 'LEAVE
(ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO)))))))
T-COND-LIST)
        (CLOCK (LOOP-BODY STMT) PROC-LIST MG-STATE (SUB1 N))           ;; body-time
        (P-STATE;; state2
(TAG 'PC
(CONS SUBR
(IF
(NORMAL (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LENGTH
(CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL

```

```

(CONS (LABEL-CNT CINFO)
      '(NIL (NO-OP))))))
  (CONS (CONS 'LEAVE
            (ADD1 (LABEL-CNT CINFO)))
        (LABEL-ALIST CINFO))
    (ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST)))
(FIND-LABEL
 (FETCH-LABEL
  (CC (MG-MEANING-R (LOOP-BODY STMT)
                   PROC-LIST MG-STATE
                   (SUB1 N)
                   (LIST (LENGTH TEMP-STK)
                         (P-CTRL-STK-SIZE CTRL-STK))))
   (LABEL-ALIST
    (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                                   (LIST (CONS 'DL
                                             (CONS (LABEL-CNT CINFO)
                                                    '(NIL (NO-OP))))))
              (CONS (CONS 'LEAVE
                        (ADD1 (LABEL-CNT CINFO)))
                    (LABEL-ALIST CINFO))
                  (ADD1 (ADD1 (LABEL-CNT CINFO))))))
   T-COND-LIST
   (LOOP-BODY STMT)
   PROC-LIST)))
  (APPEND
   (CODE
    (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                                   (LIST (CONS 'DL
                                             (CONS (LABEL-CNT CINFO)
                                                    '(NIL (NO-OP))))))
              (CONS (CONS 'LEAVE
                        (ADD1 (LABEL-CNT CINFO)))
                    (LABEL-ALIST CINFO))
                  (ADD1 (ADD1 (LABEL-CNT CINFO))))))
   T-COND-LIST
   (LOOP-BODY STMT)
   PROC-LIST)))
  (CONS (LIST 'JUMP (LABEL-CNT CINFO))
        (CONS (CONS 'DL
                  (CONS (ADD1 (LABEL-CNT CINFO))

```

```

      '(NIL (PUSH-CONSTANT (NAT 2))))))
(CONS '(POP-GLOBAL C-C) CODE2)))))))))
      CTRL-STK
      (MAP-DOWN-VALUES
(MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 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 (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 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)))
(IMPLIES
(AND
  (OK-MG-STATEMENT STMT
    (CONS 'LEAVE 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-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
  (CONS 'LEAVE R-COND-LIST))
(COND-SUBSETP (CONS 'LEAVE 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-MEANING-R (LOOP-BODY STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
 (BINDINGS (TOP CTRL-STK))
 TEMP-STK)
 (NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
 (SIGNATURES-MATCH
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
 NAME-ALIST)
 (NORMAL (MG-MEANING-R (LOOP-BODY STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
 (ALL-CARS-UNIQUE
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
 (NOT
 (RESOURCE-ERRORP
(MG-MEANING-R STMT PROC-LIST
 (MG-MEANING-R (LOOP-BODY STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)

```

```

(P-CTRL-STK-SIZE CTRL-STK))))))
(EQUAL
(P (MAP-DOWN (MG-MEANING-R (LOOP-BODY STMT)                      ;; state3
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
PROC-LIST CTRL-STK TEMP-STK
(TAG 'PC
(CONS SUBR (LENGTH (CODE CINFO))))
T-COND-LIST)
(CLOCK STMT PROC-LIST                      ;; loop-sub1-time
(MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 N)))
(P-STATE;; state4
(TAG 'PC
(CONS SUBR
(IF
(NORMAL (MG-MEANING-R STMT PROC-LIST
(MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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-MEANING-R (LOOP-BODY STMT)
      PROC-LIST MG-STATE
      (SUB1 N)
      (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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-MEANING-R (LOOP-BODY STMT)
      PROC-LIST MG-STATE
      (SUB1 N)
      (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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))))
      (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;; final
      (TAG 'PC
      (CONS SUBR
      (IF
      (NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N

```

```
;; initial
```

```
;; stmt-time
```

```

    (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 @STMT-TIME
        (CLOCK STMT PROC-LIST MG-STATE N))
    (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 @STATE1
    (MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK
      (TAG 'PC
        (CONS SUBR
          (LENGTH (CODE (MAKE-CINFO (APPEND (CODE CINFO)
            (LIST (CONS 'DL
              (CONS (LABEL-CNT CINFO)
                '(NIL (NO-OP)))))))
            (CONS (CONS 'LEAVE
              (ADD1 (LABEL-CNT CINFO)))
                (LABEL-ALIST CINFO))
              (ADD1 (ADD1 (LABEL-CNT CINFO))))))))
          T-COND-LIST))
  (ADD-ABBREVIATION @BODY-TIME
    (CLOCK (LOOP-BODY STMT)
      PROC-LIST MG-STATE
      (SUB1 N)))

```

```

(ADD-ABBREVIATION @STATE2
(P-STATE
(TAG 'PC
(CONS SUBR
(IF
(NORMAL (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LENGTH
(CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
(CONS (LABEL-CNT CINFO)
'(NIL (NO-OP)))))))
(CONS (CONS 'LEAVE
(ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST)))
(FIND-LABEL
(FETCH-LABEL
(CC (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LABEL-ALIST
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
(CONS (LABEL-CNT CINFO)
'(NIL (NO-OP)))))))
(CONS (CONS 'LEAVE
(ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST)))
(APPEND
(CODE

```

```

(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
      (CONS (LABEL-CNT CINFO)
        '(NIL (NO-OP))))))
(CONS (CONS 'LEAVE
      (ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO)))
  T-COND-LIST
  (LOOP-BODY STMT)
  PROC-LIST))
(CONS (LIST 'JUMP (LABEL-CNT CINFO))
(CONS (CONS 'DL
      (CONS (ADD1 (LABEL-CNT CINFO))
        '(NIL (PUSH-CONSTANT (NAT 2))))
      (CONS '(POP-GLOBAL C-C) CODE2))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 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 (LOOP-BODY STMT)
        PROC-LIST MG-STATE
        (SUB1 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 @STATE3
  (MAP-DOWN (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))

```

```

PROC-LIST CTRL-STK TEMP-STK
(TAG 'PC
(CONS SUBR (LENGTH (CODE CINFO))))
T-COND-LIST))
(ADD-ABBREVIATION @LOOP-SUB1-TIME
(CLOCK STMT PROC-LIST
(MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 N)))
(ADD-ABBREVIATION @STATE4
(P-STATE
(TAG 'PC
(CONS SUBR
(IF
(NORMAL (MG-MEANING-R STMT PROC-LIST
(MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 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))
PROMOTE
(DEMOTE 19)
(DIVE 1 1)
PUSH TOP PROMOTE
(CLAIM (EQUAL (P-STEP @INITIAL) @STATE1)
0)
(CLAIM (NOT (NORMAL (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))))
0)
(DROP 19)
(CLAIM (NOT (EQUAL (CC (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)

```

```

                                (LIST (LENGTH TEMP-STK)
                                    (P-CTRL-STK-SIZE CTRL-STK))))
                                'LEAVE))
                                0)
(CLAIM (EQUAL @STATE2 @FINAL) 0)
(CLAIM (EQUAL @STMT-TIME (ADD1 @BODY-TIME))
0)
(DEMOTE 19 20 23 24)
DROP
(GENERALIZE ((@STATE4 STATE4)
              (@LOOP-SUB1-TIME LOOP-SUB1-TIME)
              (@STATE3 STATE3)
              (@STATE2 STATE2)
              (@BODY-TIME BODY-TIME)
              (@STATE1 STATE1)
              (@FINAL FINAL)
              (@STMT-TIME STMT-TIME)
              (@INITIAL INITIAL)))
(USE-LEMMA LOOP-NONNORMAL-NONLEAVE-EXACT-TIME-SCHEMA)
DEMOTE
(S-PROP AND OR NOT IMPLIES FIX ZEROP IFF NLISTP)
(CONTRADICT 24)
(DROP 19 23 24)
(DIVE 1)
(REWRITE LOOP-CLOCK-NONNORMAL-NONLEAVE)
TOP S
(CONTRADICT 23)
(DROP 19 20 23)
(DIVE 1)
(REWRITE LOOP-NONNORMAL-NONLEAVE-STATE2-EQUALS-FINAL)
TOP S-PROP
(CLAIM (EQUAL (P @STATE2 2) @FINAL) 0)
(CLAIM (EQUAL @STMT-TIME (PLUS 3 @BODY-TIME))
0)
(DEMOTE 19 20 23 24)
DROP
(GENERALIZE ((@STATE4 STATE4)
              (@LOOP-SUB1-TIME LOOP-SUB1-TIME)
              (@STATE3 STATE3)
              (@STATE2 STATE2)
              (@BODY-TIME BODY-TIME)
              (@STATE1 STATE1)
              (@FINAL FINAL)
              (@STMT-TIME STMT-TIME)

```

```

                                (@INITIAL INITIAL)))
DROP
(USE-LEMMA LOOP-LEAVE-BODY-EXACT-TIME-SCHEMA)
PROVE
(CONTRADICT 24)
(DIVE 1)
(REWRITE LOOP-CLOCK-NONNORMAL-LEAVE)
TOP S-PROP
(CONTRADICT 23)
(DIVE 1)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-0-UNWINDING-LEMMA)
(DIVE 1)
(REWRITE LOOP-LEAVE-STATE2-STEP1-EFFECT)
UP
(REWRITE LOOP-LEAVE-STATE2-STEP2-EFFECT)
TOP S-PROP
(DEMOTE 19)
(DIVE 1 1)
PUSH TOP PROMOTE
(CLAIM (EQUAL @STMT-TIME
              (ADD1 (PLUS (ADD1 @BODY-TIME)
                          @LOOP-SUB1-TIME)))
0)
(CLAIM (EQUAL (P-STEP @STATE2) @STATE3)
0)
(CLAIM (EQUAL @STATE4 @FINAL) 0)
(DEMOTE 19 20 22 23 24 25)
DROP
(GENERALIZE ((@STATE4 STATE4)
              (@LOOP-SUB1-TIME LOOP-SUB1-TIME)
              (@STATE3 STATE3)
              (@STATE2 STATE2)
              (@BODY-TIME BODY-TIME)
              (@STATE1 STATE1)
              (@FINAL FINAL)
              (@STMT-TIME STMT-TIME)
              (@INITIAL INITIAL)))
DROP
(USE-LEMMA LOOP-NORMAL-BODY-EXACT-TIME-SCHEMA)
PROVE
(CONTRADICT 25)
(DROP 19 20 22 23 24 25)

```

```

(DIVE 1)
(REWRITE LOOP-NORMAL-BODY-STATE4-EQUALS-FINAL)
TOP S-PROP
(CONTRADICT 24)
(DIVE 1)
(REWRITE LOOP-NORMAL-BODY-STEP-STATE2-EQUALS-STATE3)
TOP S-PROP
(CONTRADICT 23)
(DIVE 1)
(REWRITE LOOP-CLOCK-NORMAL
  (($SIZES (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK)))))
UP S
(USE-LEMMA LOOP-SUB1-BODY-EXACT-TIME-HYPS)
SPLIT
(CONTRADICT 21)
(DIVE 1)
(REWRITE LOOP-STEP-INITIAL-EQUALS-STATE1)
TOP S-PROP
(DROP 19)
(USE-LEMMA LOOP-BODY-EXACT-TIME-HYPS)
DEMOTE
(S-PROP AND OR NOT IMPLIES FIX ZEROP IFF NLISTP))))

```

EVENT: Make the library "c-loop".

Index

- add-code, 1
- adding-leave-preserves-statement
 - okness, 5
 - okness-begin-case, 5
- all-cars-unique, 3, 4, 6, 7, 26
- begin-body, 5
- bindings, 3–7, 26, 28, 29
- cc, 8, 9, 14, 26–29
- clock, 8
- clock-equivalence-loop-case, 2
- code, 1, 3–6, 9, 10, 14, 26–29
- cond-list-superset-preserves-ok
 - mg-statement, 5
- cond-subsetp, 3–6, 26
- discard-label, 1
- fetch-label, 27, 29
- find-label, 27, 29
- find-labelp, 14
- label-alist, 1, 3, 4, 9, 10, 14, 27, 29
- label-cnt, 1, 3, 4, 9, 10, 14
- length, 3, 5–7, 9, 26–29
- loop-body, 1–4, 6–10, 14, 26–28
- loop-body-doesnt-halt, 2
- loop-body-exact-time-hyps, 2
- loop-clock-nonnormal-leave, 8
- loop-clock-nonnormal-nonleave, 7
- loop-clock-normal, 8
- loop-code-rewrite, 10
- loop-find-labelp-lemma1, 14
- loop-find-labelp-lemma2, 14
- loop-leave-body-exact-time-sche
 - ma, 29
- loop-leave-body-meaning-preserve
 - d, 13
- loop-meaning-r-2, 1
- loop-never-leave, 26
- loop-nonnormal-nonleave-exact-ti
 - me-schema, 29
- loop-normal-body-exact-time-sche
 - ma, 29
- loop-normal-body-state4-equals-
 - final, 26
- loop-step-initial-equals-state1, 8
- loop-sub1-body-doesnt-halt, 2
- loop-sub1-body-exact-time-hyps, 5
- loop-translation-2, 1
- make-cinfo, 1, 3, 4, 9, 10, 14
- map-down, 9
- map-down-values, 28, 29
- mg-alist, 3–7, 26, 28, 29
- mg-cond-to-p-nat, 28, 29
- mg-max-ctrl-stk-size, 28, 29
- mg-max-temp-stk-size, 28, 29
- mg-meaning-r, 1–3, 5–9, 14, 26–29
- mg-psw, 2
- mg-vars-list-ok-in-p-state, 3–5, 7, 26
- mg-word-size, 28, 29
- no-p-aliasing, 3–5, 7, 26
- nonleave-nonnormal-body-meaning
 - preserved, 9
- normal, 1–4, 6–9, 14, 26–28
- ok-mg-def-plistp, 3, 5, 6, 9, 26
- ok-mg-statement, 3, 5, 6, 8, 14, 26
- ok-mg-statep, 3–6, 26
- ok-translation-parameters, 3–6, 9, 14, 26
- p, 29, 30
- p-ctrl-stk-size, 3, 5–7, 26–29
- p-state, 28, 29
- p-step, 9, 29, 30
- plistp, 3–6, 26
- remove-leave, 2
- resource-errorp, 2, 3, 5–9, 14, 26
- resources-inadequatep, 1, 3, 5, 26

- set-condition, 14
- signal-system-error, 1
- signatures-match, 3–5, 7, 26
- subset, 5

- tag, 9, 27, 29
- top, 3–7, 26, 28, 29
- translate, 1, 3–6, 9, 10, 14, 26–29
- translate-def-body, 3–6, 9, 26
- translate-proc-list, 28, 29

- user-defined-procp, 3–6, 9, 26

- when-labels, 5