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

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                              ;;
;;                          EXACT-TIME PROG2 CASE                               ;;
;;                                                                              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: prog2-meaning-r-2
(car (*stmt*) = 'prog2-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 mg-meaning-r (prog2-right-branch (*stmt*),
proc-list,
mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
n − 1,
sizes),
n − 1,
sizes) **endif**)

THEOREM: mg-meaning-r-prog2-left-non-normal
((car (*stmt*) = 'prog2-mg)
∧ (cc (*mg-state*) = 'normal)
∧ (¬ resource-errorp (mg-meaning-r (*stmt*, *proc-list*, *mg-state*, *n*, *sizes*)))
∧ (¬ normal (mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
n − 1,
sizes))))
→ (mg-meaning-r (*stmt*, *proc-list*, *mg-state*, *n*, *sizes*)
= mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
n − 1,
sizes))

THEOREM: prog2-translation-2
(car (*stmt*) = 'prog2-mg)

$\rightarrow$ (translate (*cinfo*, *cond-list*, *stmt*, *proc-list*)
= translate (translate (*cinfo*,
cond-list,
prog2-left-branch (*stmt*),
proc-list),
cond-list,
prog2-right-branch (*stmt*),
proc-list))

THEOREM: prog2-left-branch-doesnt-halt
((car (*stmt*) = `'prog2-mg`)
$\wedge$ normal (*mg-state*)
$\wedge$ ($\neg$ resource-errorp (mg-meaning-r (*stmt*, *proc-list*, *mg-state*, *n*, *sizes*))))
$\rightarrow$ (mg-psw (mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,
sizes))
= `'run`)

THEOREM: prog2-right-branch-doesnt-halt
((car (*stmt*) = `'prog2-mg`)
$\wedge$ normal (*mg-state*)
$\wedge$ normal (mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,
sizes))
$\wedge$ ($\neg$ resource-errorp (mg-meaning-r (*stmt*, *proc-list*, *mg-state*, *n*, *sizes*))))
$\rightarrow$ (mg-psw (mg-meaning-r (prog2-right-branch (*stmt*),
proc-list,
mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,
sizes),
$n - 1$,
sizes))
= `'run`)

THEOREM: clock-prog2-left-non-normal
((car (*stmt*) = `'prog2-mg`)
$\wedge$ (cc (*mg-state*) = `'normal`)
$\wedge$ ($\neg$ resource-errorp (mg-meaning-r (*stmt*, *proc-list*, *mg-state*, *n*, *sizes*)))
$\wedge$ ($\neg$ normal (mg-meaning-r (prog2-left-branch (*stmt*),

proc-list,
mg-state,
$n - 1$,
sizes))))
→ (clock (*stmt*, *proc-list*, *mg-state*, *n*)
= clock (prog2-left-branch (*stmt*), *proc-list*, *mg-state*, $n - 1$))

THEOREM: prog2-code-rewrite
((car (*stmt*) = 'prog2-mg)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *y*))
→ (append (code (translate (*cinfo*,
t-cond-list,
prog2-left-branch (*stmt*),
proc-list)),
append (code (translate (nullify (translate (nullify (*cinfo*),
t-cond-list,
prog2-left-branch (*stmt*),
proc-list)),
t-cond-list,
prog2-right-branch (*stmt*),
proc-list)),
y))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)), *y*))

EVENT: Disable prog2-code-rewrite.

THEOREM: prog2-left-branch-translation-parameters-ok
((car (*stmt*) = 'prog2-mg)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*))
→ ok-translation-parameters (*cinfo*,
t-cond-list,
prog2-left-branch (*stmt*),
proc-list,
append (code (translate (nullify (translate (nullify (*cinfo*),
t-cond-list,
prog2-left-branch (*stmt*),
proc-list)),
t-cond-list,
prog2-right-branch (*stmt*),
proc-list)),
code2))

EVENT: Disable prog2-left-branch-translation-parameters-ok.

THEOREM: prog2-left-branch-code-body-rewrite
((car (*stmt*) = 'prog2-mg)
 ∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
 ∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
     = append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
                    code2)))
→ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
     = append (code (translate (*cinfo*,
                                t-cond-list,
                                prog2-left-branch (*stmt*),
                                proc-list)),
                    append (code (translate (nullify (translate (nullify (*cinfo*),
                                                                 t-cond-list,
                                                                 prog2-left-branch (*stmt*),
                                                                 proc-list)),
                                                    t-cond-list,
                                                    prog2-right-branch (*stmt*),
                                                    proc-list)),
                                   code2)))

EVENT: Disable prog2-left-branch-code-body-rewrite.

THEOREM: prog2-nonnormal-left-state2-equals-final
((*n* ≠ 0)
 ∧ (¬ resources-inadequatep (*stmt*,
                             proc-list,
                             list (length (*temp-stk*),
                                  p-ctrl-stk-size (*ctrl-stk*))))
 ∧ (car (*stmt*) = 'prog2-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*))

$$
\begin{array}{ll}
\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-r}\,(\text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad \textit{proc-list}, \\
 & \qquad\qquad\qquad\qquad\qquad \textit{mg-state}, \\
 & \qquad\qquad\qquad\qquad\qquad n-1, \\
 & \qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\textit{temp-stk}), \\
 & \qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))))) \\
\rightarrow & (\text{p-state}\,(\text{tag}\,(\texttt{'pc}, \\
 & \qquad\qquad \text{cons}\,(\textit{subr}, \\
 & \qquad\qquad\quad \textbf{if}\ \text{normal}\,(\text{mg-meaning-r}\,(\text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{mg-state}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad n-1, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\textit{temp-stk}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))) \\
 & \qquad\qquad\quad \textbf{then}\ \text{length}\,(\text{code}\,(\text{translate}\,(\textit{cinfo}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list}))) \\
 & \qquad\qquad\quad \textbf{else}\ \text{find-label}\,(\text{fetch-label}\,(\text{cc}\,(\text{mg-meaning-r}\,(\text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{mg-state}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad n-1, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{list}\,(\text{length}\,(\textit{temp-stk}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk}))), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{label-alist}\,(\text{translate}\,(\textit{cinfo}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list}))), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad \text{append}\,(\text{code}\,(\text{translate}\,(\textit{cinfo}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list})), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{append}\,(\text{code}\,(\text{translate}\,(\text{nullify}\,(\text{translate}\,(\text{nullify}\,(\textit{cinfo}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{prog2-left-branch}\,(
\end{array}
$$

```
                                                                    proc-list)),
                                                          t-cond-list,
                                                          prog2-right-branch (stmt),
                                                          proc-list)),
                                                code2))) endif)),
        ctrl-stk,
        map-down-values (mg-alist (mg-meaning-r (prog2-left-branch (stmt),
                                                 proc-list,
                                                 mg-state,
                                                 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 (prog2-left-branch (stmt),
                                                        proc-list,
                                                        mg-state,
                                                        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))),
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))

EVENT: Disable prog2-nonnormal-left-state2-equals-final.

THEOREM: prog2-right-branch-translation-parameters-ok
((car (*stmt*) = 'prog2-mg)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*))
→ ok-translation-parameters (translate (*cinfo*,
t-cond-list,
prog2-left-branch (*stmt*),
proc-list),
t-cond-list,
prog2-right-branch (*stmt*),

proc-list,
code2)

EVENT: Disable prog2-right-branch-translation-parameters-ok.

THEOREM: prog2-right-branch-hyps
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'prog2-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 (prog2-left-branch (*stmt*),
proc-list,
mg-state,
n − 1,
list (length (*temp-stk*), p-ctrl-stk-size (*ctrl-stk*)))))
→ (ok-mg-statement (prog2-right-branch (*stmt*),
r-cond-list,
name-alist,

proc-list)
∧ ok-translation-parameters (translate (*cinfo*,
t-cond-list,
prog2-left-branch (*stmt*),
proc-list),
t-cond-list,
prog2-right-branch (*stmt*),
proc-list,
code2)
∧ ok-mg-statep (mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))),
r-cond-list)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (translate (*cinfo*,
t-cond-list,
prog2-left-branch (*stmt*),
proc-list),
t-cond-list,
prog2-right-branch (*stmt*),
proc-list)),
code2))
∧ mg-vars-list-ok-in-p-state (mg-alist (mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,
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 (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))))
∧ signatures-match (mg-alist (mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,

list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
name-alist)
∧ all-cars-unique (mg-alist (mg-meaning-r (prog2-left-branch (*stmt*),
proc-list,
mg-state,
$n - 1$,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))))
∧ (¬ resource-errorp (mg-meaning-r (prog2-right-branch (*stmt*),
proc-list,
mg-meaning-r (prog2-left-branch (*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*))))))

THEOREM: prog2-state2-equals-state3
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'prog2-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*)

$$
\begin{array}{ll}
\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 & \text{normal}\,(\text{mg-meaning-r}\,(\text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad \textit{proc-list}, \\
 & \qquad\qquad\qquad\qquad\qquad \textit{mg-state}, \\
 & \qquad\qquad\qquad\qquad\qquad n-1, \\
 & \qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\textit{temp-stk}),\ \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))) \\
\rightarrow & (\text{p-state}\,(\text{tag}\,(\texttt{'pc}, \\
 & \qquad\qquad\qquad \text{cons}\,(\textit{subr}, \\
 & \qquad\qquad\qquad\quad \textbf{if}\ \text{normal}\,(\text{mg-meaning-r}\,(\text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{mg-state}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad n-1, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\textit{temp-stk}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))) \\
 & \qquad\qquad\qquad\quad \textbf{then}\ \text{length}\,(\text{code}\,(\text{translate}\,(\textit{cinfo}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}))) \\
 & \qquad\qquad\qquad\quad \textbf{else}\ \text{find-label}\,(\text{fetch-label}\,(\text{cc}\,(\text{mg-meaning-r}\,(\text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{proc-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{mg-state}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad n-1, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{length}\,(\textit{temp-stk}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{p-ctrl-stk-size}\,(\textit{ctrl-stk})))), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{label-alist}\,(\text{translate}\,(\textit{cinfo}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list}))), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{append}\,(\text{code}\,(\text{translate}\,(\textit{cinfo}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{prog2-left-branch}\,(\textit{stmt}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{proc-list})), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{append}\,(\text{code}\,(\text{translate}\,(\text{nullify}\,(\text{translate}\,(\text{nullify}\,(\textit{cinfo}), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{t-cond-list}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{prog2-left-branch}\,(\\
 & \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\qquad\qquad\qquad\quad \textit{t-cond-list},
\end{array}
$$

```
                                                                  prog2-right-branch (stmt),
                                                                  proc-list)),
                                                        code2))) endif)),
                ctrl-stk,
                map-down-values (mg-alist (mg-meaning-r (prog2-left-branch (stmt),
                                                        proc-list,
                                                        mg-state,
                                                        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 (prog2-left-branch (stmt),
                                                               proc-list,
                                                               mg-state,
                                                               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)
    =   map-down (mg-meaning-r (prog2-left-branch (stmt),
                                proc-list,
                                mg-state,
                                n − 1,
                                list (length (temp-stk),
                                      p-ctrl-stk-size (ctrl-stk))),
                  proc-list,
                  ctrl-stk,
                  temp-stk,
                  tag ('pc,
                       cons (subr,
                             length (code (translate (cinfo,
                                                      t-cond-list,
                                                      prog2-left-branch (stmt),
                                                      proc-list))))),
                  t-cond-list))
```

EVENT: Disable prog2-state2-equals-state3.

THEOREM: prog2-state4-equals-final

```
((n ≠ 0)
 ∧  (¬ resources-inadequatep (stmt,
                              proc-list,
                              list (length (temp-stk),
                                    p-ctrl-stk-size (ctrl-stk))))
 ∧  (car (stmt) = 'prog2-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 (prog2-left-branch (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 (prog2-right-branch (stmt),
                                                 proc-list,
                                                 mg-meaning-r (prog2-left-branch (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 (translate (cinfo,
                                                             t-cond-list,
                                                             prog2-left-branch (stmt),
                                                             proc-list),
                                                        t-cond-list,
                                                        prog2-right-branch (stmt),
                                                        proc-list)))
                    else find-label (fetch-label (cc (mg-meaning-r (prog2-right-branch (stmt),
                                                                    proc-list,
                                                                    mg-meaning-r (prog2-left-branch (stmt)
                                                                                  proc-list,
                                                                                  mg-state,
                                                                                  n − 1,
                                                                                  list (length (temp-stk),
                                                                                        p-ctrl-stk-size (ctrl-s
                                                                    n − 1,
                                                                    list (length (temp-stk),
                                                                          p-ctrl-stk-size (ctrl-stk)))),
                                                     label-alist (translate (translate (cinfo,
                                                                                        t-cond-list,
                                                                                        prog2-left-branch (stmt),
                                                                                        proc-list),
                                                                                   t-cond-list,
                                                                                   prog2-right-branch (stmt),
                                                                                   proc-list))),
                                                     append (code (translate (translate (cinfo,
                                                                                         t-cond-list,
                                                                                         prog2-left-branch (stmt),
                                                                                         proc-list),
                                                                                    t-cond-list,
                                                                                    prog2-right-branch (stmt),
                                                                                    proc-list)),
                                                                  code2)) endif)),
                    ctrl-stk,
                    map-down-values (mg-alist (mg-meaning-r (prog2-right-branch (stmt),
                                                             proc-list,
                                                             mg-meaning-r (prog2-left-branch (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 (prog2-right-branch (stmt),
                                                            proc-list,
                                                            mg-meaning-r (prog2-left-branch (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))),
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))

EVENT: Disable prog2-state4-equals-final.

THEOREM: prog2-normal-left-exact-time-schema
((*stmt-time* = (*left-time* + *right-time*))
∧ (p (*initial*, *left-time*) = *state2*)
∧ (*state2* = *state3*)
∧ (p (*state3*, *right-time*) = *state4*)
∧ (*state4* = *final*))
→ (p (*initial*, *stmt-time*) = *final*)

EVENT: Disable prog2-normal-left-exact-time-schema.

```
(prove-lemma exact-time-lemma-prog2-case (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) 'prog2-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 (prog2-left-branch stmt)
   r-cond-list name-alist proc-list)
  (ok-mg-def-plistp proc-list)
  (ok-translation-parameters cinfo t-cond-list
     (prog2-left-branch stmt)
     proc-list
     (append (code (translate (nullify (translate (nullify cinfo)
  t-cond-list
```

```
 (prog2-left-branch stmt)
 proc-list))
     t-cond-list
     (prog2-right-branch 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
    (prog2-left-branch stmt)
    proc-list))
   (append (code (translate (nullify (translate (nullify cinfo)
t-cond-list
(prog2-left-branch stmt)
proc-list))
    t-cond-list
    (prog2-right-branch 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 (prog2-left-branch 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 cinfo))))
```

```
       t-cond-list)
     (clock (prog2-left-branch stmt)                                ;; left-time
    proc-list mg-state
    (sub1 n)))
  (p-state;; state2
   (tag 'pc
(cons subr
      (if
  (normal (mg-meaning-r (prog2-left-branch stmt)
proc-list mg-state
(sub1 n)
(list (length temp-stk)
      (p-ctrl-stk-size ctrl-stk))))
  (length (code (translate cinfo t-cond-list
   (prog2-left-branch stmt)
   proc-list)))
(find-label
 (fetch-label (cc (mg-meaning-r (prog2-left-branch stmt)
proc-list mg-state
(sub1 n)
(list (length temp-stk)
      (p-ctrl-stk-size ctrl-stk))))
      (label-alist (translate cinfo t-cond-list
      (prog2-left-branch stmt)
      proc-list)))
 (append
  (code (translate cinfo t-cond-list
   (prog2-left-branch stmt)
   proc-list))
  (append (code (translate (nullify (translate (nullify cinfo)
       t-cond-list
       (prog2-left-branch stmt)
       proc-list))
   t-cond-list
   (prog2-right-branch stmt)
   proc-list))
  code2))))))
   ctrl-stk
   (map-down-values
    (mg-alist (mg-meaning-r (prog2-left-branch 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 (prog2-left-branch 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 (prog2-right-branch stmt)
   r-cond-list name-alist proc-list)
  (ok-mg-def-plistp proc-list)
  (ok-translation-parameters (translate cinfo t-cond-list
(prog2-left-branch stmt)
proc-list)
     t-cond-list
     (prog2-right-branch stmt)
     proc-list code2)
  (ok-mg-statep (mg-meaning-r (prog2-left-branch stmt)
      proc-list mg-state
      (sub1 n)
      (list (length temp-stk)
    (p-ctrl-stk-size ctrl-stk)))
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 (translate cinfo t-cond-list
     (prog2-left-branch stmt)
     proc-list)
  t-cond-list
  (prog2-right-branch 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 (prog2-left-branch 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 (prog2-left-branch stmt)
 proc-list mg-state
 (sub1 n)
 (list (length temp-stk)
       (p-ctrl-stk-size ctrl-stk)))))
  (signatures-match
   (mg-alist (mg-meaning-r (prog2-left-branch stmt)
   proc-list mg-state
   (sub1 n)
   (list (length temp-stk)
 (p-ctrl-stk-size ctrl-stk))))
   name-alist)
  (normal (mg-meaning-r (prog2-left-branch 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 (prog2-left-branch stmt)
   proc-list mg-state
   (sub1 n)
   (list (length temp-stk)
 (p-ctrl-stk-size ctrl-stk)))))
  (not
   (resource-errorp
    (mg-meaning-r (prog2-right-branch stmt)
  proc-list
  (mg-meaning-r (prog2-left-branch 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))))))
 (equal
  (p (map-down (mg-meaning-r (prog2-left-branch 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 (translate cinfo t-cond-list
   (prog2-left-branch stmt)
   proc-list)))))
       t-cond-list)
     (clock (prog2-right-branch stmt)                               ;; right-time
    proc-list
    (mg-meaning-r (prog2-left-branch 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 (prog2-right-branch stmt)
proc-list
(mg-meaning-r (prog2-left-branch 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 (translate cinfo t-cond-list
      (prog2-left-branch stmt)
      proc-list)
   t-cond-list
   (prog2-right-branch stmt)
   proc-list)))
(find-label
 (fetch-label
```

```
  (cc (mg-meaning-r (prog2-right-branch stmt)
    proc-list
    (mg-meaning-r (prog2-left-branch 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 (translate cinfo t-cond-list
     (prog2-left-branch stmt)
     proc-list)
  t-cond-list
  (prog2-right-branch stmt)
  proc-list)))
 (append (code (translate (translate cinfo t-cond-list
     (prog2-left-branch stmt)
     proc-list)
  t-cond-list
  (prog2-right-branch stmt)
  proc-list))
 code2)))))
   ctrl-stk
   (map-down-values
    (mg-alist (mg-meaning-r (prog2-right-branch stmt)
    proc-list
    (mg-meaning-r (prog2-left-branch 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 (prog2-right-branch stmt)
     proc-list
     (mg-meaning-r (prog2-left-branch 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                    ;; initial
               (tag 'pc
                    (cons subr (length (code cinfo))))
               t-cond-list)
     (clock stmt proc-list mg-state n))                                ;; stmt-time
  (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
   (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 @left-time
                     (clock (prog2-left-branch stmt)
                            proc-list mg-state
                            (sub1 n)))
   (add-abbreviation @state2
    (p-state
     (tag 'pc
  (cons subr
(if
    (normal (mg-meaning-r (prog2-left-branch stmt)
  proc-list mg-state
  (sub1 n)
  (list (length temp-stk)
(p-ctrl-stk-size ctrl-stk))))
    (length (code (translate cinfo t-cond-list
     (prog2-left-branch stmt)
     proc-list)))
  (find-label
   (fetch-label (cc (mg-meaning-r (prog2-left-branch stmt)
  proc-list mg-state
  (sub1 n)
  (list (length temp-stk)
(p-ctrl-stk-size ctrl-stk))))
(label-alist (translate cinfo t-cond-list
(prog2-left-branch stmt)
proc-list)))
   (append
    (code (translate cinfo t-cond-list
     (prog2-left-branch stmt)
     proc-list))
    (append (code (translate (nullify (translate (nullify cinfo)
 t-cond-list
 (prog2-left-branch stmt)
 proc-list))
     t-cond-list
     (prog2-right-branch stmt)
     proc-list))
    code2))))))
```

```
     ctrl-stk
     (map-down-values
      (mg-alist (mg-meaning-r (prog2-left-branch 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 (prog2-left-branch 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 (prog2-left-branch 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 (translate cinfo t-cond-list
                                                       (prog2-left-branch stmt)
                                                       proc-list)))))
                   t-cond-list))
   (add-abbreviation @right-time
                     (clock (prog2-right-branch stmt)
                            proc-list
                            (mg-meaning-r (prog2-left-branch 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 (prog2-right-branch stmt)
  proc-list
  (mg-meaning-r (prog2-left-branch 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 (translate cinfo t-cond-list
(prog2-left-branch stmt)
proc-list)
     t-cond-list
     (prog2-right-branch stmt)
     proc-list)))
  (find-label
   (fetch-label
    (cc (mg-meaning-r (prog2-right-branch stmt)
      proc-list
      (mg-meaning-r (prog2-left-branch 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 (translate cinfo t-cond-list
       (prog2-left-branch stmt)
       proc-list)
    t-cond-list
    (prog2-right-branch stmt)
    proc-list)))
   (append (code (translate (translate cinfo t-cond-list
       (prog2-left-branch stmt)
       proc-list)
    t-cond-list
    (prog2-right-branch stmt)
```

```
 proc-list))
code2)))))
  ctrl-stk
  (map-down-values
   (mg-alist (mg-meaning-r (prog2-right-branch stmt)
   proc-list
   (mg-meaning-r (prog2-left-branch 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 (prog2-right-branch stmt)
                            proc-list
                            (mg-meaning-r (prog2-left-branch 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 (not (normal (mg-meaning-r (prog2-left-branch stmt)
                                  proc-list mg-state
                                  (sub1 n)
                                  (list (length temp-stk)
                                        (p-ctrl-stk-size ctrl-stk)))))
```

```
       0)
(drop 19)
(claim (equal @state2 @final) 0)
(claim (equal @stmt-time @left-time)
       0)
(demote 19 21 22)
drop
(generalize ((@right-time right-time)
             (@left-time left-time)
             (@state4 state4)
             (@state3 state3)
             (@state2 state2)
             (@final final)
             (@stmt-time stmt-time)
             (@initial initial)))
prove
(contradict 22)
(dive 1)
(rewrite clock-prog2-left-non-normal
         (($sizes (list (length temp-stk)
                        (p-ctrl-stk-size ctrl-stk)))))
up s
(demote 16)
s
(contradict 21)
(dive 1)
(rewrite prog2-nonnormal-left-state2-equals-final)
top s-prop
(demote 19)
(dive 1 1)
push top promote
(claim (equal @stmt-time
              (plus @left-time @right-time))
       0)
(claim (equal @state2 @state3) 0)
(claim (equal @state4 @final) 0)
(demote 19 21 22 23 24)
drop
(generalize ((@right-time right-time)
             (@left-time left-time)
             (@state4 state4)
             (@state3 state3)
             (@state2 state2)
             (@final final)
```

```
            (@stmt-time stmt-time)
            (@initial initial)))
(use-lemma prog2-normal-left-exact-time-schema)
prove
(contradict 24)
(dive 1)
(rewrite prog2-state4-equals-final)
drop top prove
(contradict 23)
(dive 1)
(rewrite prog2-state2-equals-state3)
top s
(contradict 22)
(dive 1)
(rewrite clock-prog2)
s up
(rewrite plus-equality-lemma1)
(dive 2 3)
(rewrite mg-meaning-equivalence)
top s s
(dive 1)
(rewrite prog2-left-branch-doesnt-halt)
top s split
(rewrite prog2-right-branch-hyps)
(rewrite prog2-right-branch-hyps)
(rewrite prog2-right-branch-hyps)
(dive 1)
(rewrite prog2-right-branch-hyps)
top s
(rewrite prog2-right-branch-hyps)
(rewrite prog2-right-branch-hyps)
(rewrite prog2-right-branch-hyps)
(rewrite prog2-right-branch-hyps)
(dive 1)
(rewrite prog2-right-branch-hyps)
top s
(drop 19)
split
(rewrite ok-prog2-statement)
(rewrite prog2-left-branch-translation-parameters-ok)
(dive 1)
(rewrite prog2-left-branch-code-body-rewrite)
up s s
(dive 1)
```

```
(rewrite prog2-left-branch-doesnt-halt)
top s)))
```

EVENT: Disable exact-time-lemma-prog2-case.

EVENT: Make the library "c-prog2".

Index