

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

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

THEOREM: prog2-meaning-r-2

$$\begin{aligned} & (\text{car } (stmt) = \text{'prog2-mg}) \\ \rightarrow & \quad (\text{mg-meaning-r } (stmt, proc-list, mg-state, n, sizes) \\ & \quad = \text{ if } n \simeq 0 \text{ then signal-system-error } (mg-state, \text{'timed-out}) \\ & \quad \text{elseif } \neg \text{normal } (mg-state) \text{ then } mg-state \\ & \quad \text{elseif resources-inadequatep } (stmt, proc-list, sizes) \\ & \quad \text{then signal-system-error } (mg-state, \text{'resource-error}) \\ & \quad \text{else mg-meaning-r } (\text{prog2-right-branch } (stmt), \\ & \quad \quad proc-list, \\ & \quad \quad \text{mg-meaning-r } (\text{prog2-left-branch } (stmt), \\ & \quad \quad \quad proc-list, \\ & \quad \quad \quad mg-state, \\ & \quad \quad \quad n - 1, \\ & \quad \quad \quad sizes), \\ & \quad \quad n - 1, \\ & \quad \quad sizes) \text{ endif}) \end{aligned}$$

THEOREM: mg-meaning-r-prog2-left-non-normal

$$\begin{aligned} & ((\text{car } (stmt) = \text{'prog2-mg}) \\ \wedge & \quad (\text{cc } (mg-state) = \text{'normal}) \\ \wedge & \quad (\neg \text{resource-errorp } (\text{mg-meaning-r } (stmt, proc-list, mg-state, n, sizes))) \\ \wedge & \quad (\neg \text{normal } (\text{mg-meaning-r } (\text{prog2-left-branch } (stmt), \\ & \quad \quad proc-list, \\ & \quad \quad mg-state, \\ & \quad \quad n - 1, \\ & \quad \quad sizes)))) \\ \rightarrow & \quad (\text{mg-meaning-r } (stmt, proc-list, mg-state, n, sizes) \\ & \quad = \text{ mg-meaning-r } (\text{prog2-left-branch } (stmt), \\ & \quad \quad proc-list, \\ & \quad \quad mg-state, \\ & \quad \quad n - 1, \\ & \quad \quad sizes)) \end{aligned}$$

THEOREM: prog2-translation-2

$$(\text{car } (stmt) = \text{'prog2-mg})$$

$$\begin{aligned}
\rightarrow & \text{ (translate} (cinfo, cond-list, stmt, proc-list) \\
= & \text{ translate} (\text{translate} (cinfo, \\
& \quad cond-list, \\
& \quad \text{prog2-left-branch} (stmt), \\
& \quad proc-list), \\
& \quad cond-list, \\
& \quad \text{prog2-right-branch} (stmt), \\
& \quad proc-list))
\end{aligned}$$

THEOREM: prog2-left-branch-doesnt-halt

$$\begin{aligned}
& ((\text{car} (stmt) = 'prog2-mg) \\
& \wedge \text{normal} (mg-state) \\
& \wedge (\neg \text{resource-errorp} (\text{mg-meaning-r} (stmt, proc-list, mg-state, n, sizes)))) \\
\rightarrow & \text{ (mg-psw} (\text{mg-meaning-r} (\text{prog2-left-branch} (stmt), \\
& \quad proc-list, \\
& \quad mg-state, \\
& \quad n - 1, \\
& \quad sizes))) \\
= & 'run)
\end{aligned}$$

THEOREM: prog2-right-branch-doesnt-halt

$$\begin{aligned}
& ((\text{car} (stmt) = 'prog2-mg) \\
& \wedge \text{normal} (mg-state) \\
& \wedge \text{normal} (\text{mg-meaning-r} (\text{prog2-left-branch} (stmt), \\
& \quad proc-list, \\
& \quad mg-state, \\
& \quad n - 1, \\
& \quad sizes))) \\
& \wedge (\neg \text{resource-errorp} (\text{mg-meaning-r} (stmt, proc-list, mg-state, n, sizes)))) \\
\rightarrow & \text{ (mg-psw} (\text{mg-meaning-r} (\text{prog2-right-branch} (stmt), \\
& \quad proc-list, \\
& \quad \text{mg-meaning-r} (\text{prog2-left-branch} (stmt), \\
& \quad proc-list, \\
& \quad mg-state, \\
& \quad n - 1, \\
& \quad sizes), \\
& \quad n - 1, \\
& \quad sizes))) \\
= & 'run)
\end{aligned}$$

THEOREM: clock-prog2-left-non-normal

$$\begin{aligned}
& ((\text{car} (stmt) = 'prog2-mg) \\
& \wedge (\text{cc} (mg-state) = 'normal) \\
& \wedge (\neg \text{resource-errorp} (\text{mg-meaning-r} (stmt, proc-list, mg-state, n, sizes)))) \\
& \wedge (\neg \text{normal} (\text{mg-meaning-r} (\text{prog2-left-branch} (stmt),
\end{aligned}$$

```

    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-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

$$\begin{aligned}
 & ((\text{car}(\text{stmt}) = \text{'prog2-mg}) \\
 & \quad \wedge \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2})) \\
 & \quad \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) \\
 & \quad \quad = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \\
 & \quad \quad \quad \text{code2}))) \\
 \rightarrow & (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) \\
 & \quad = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \\
 & \quad \quad \quad \text{t-cond-list}, \\
 & \quad \quad \quad \text{prog2-left-branch}(\text{stmt}), \\
 & \quad \quad \quad \text{proc-list})), \\
 & \quad \quad \quad \text{append}(\text{code}(\text{translate}(\text{nullify}(\text{translate}(\text{nullify}(\text{cinfo}), \\
 & \quad \quad \quad \quad \quad \text{t-cond-list}, \\
 & \quad \quad \quad \quad \quad \text{prog2-left-branch}(\text{stmt}), \\
 & \quad \quad \quad \quad \quad \text{proc-list})), \\
 & \quad \quad \quad \quad \quad \text{t-cond-list}, \\
 & \quad \quad \quad \quad \quad \text{prog2-right-branch}(\text{stmt}), \\
 & \quad \quad \quad \quad \quad \text{proc-list})), \\
 & \quad \quad \quad \quad \quad \text{code2})))
 \end{aligned}$$

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

THEOREM: prog2-nonnnormal-left-state2-equals-final

$$\begin{aligned}
 & ((n \not\leq 0) \\
 & \quad \wedge (\neg \text{resources-inadequatep}(\text{stmt}, \\
 & \quad \quad \quad \text{proc-list}, \\
 & \quad \quad \quad \text{list}(\text{length}(\text{temp-stk}), \\
 & \quad \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \\
 & \quad \wedge (\text{car}(\text{stmt}) = \text{'prog2-mg}) \\
 & \quad \wedge \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \\
 & \quad \wedge \text{ok-mg-def-plistp}(\text{proc-list}) \\
 & \quad \wedge \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2})) \\
 & \quad \wedge \text{ok-mg-statep}(\text{mg-state}, \text{r-cond-list}) \\
 & \quad \wedge \text{cond-subsetp}(\text{r-cond-list}, \text{t-cond-list}) \\
 & \quad \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) \\
 & \quad \quad = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \\
 & \quad \quad \quad \text{code2})) \\
 & \quad \wedge \text{user-defined-procp}(\text{subr}, \text{proc-list}) \\
 & \quad \wedge \text{plistp}(\text{temp-stk}) \\
 & \quad \wedge \text{listp}(\text{ctrl-stk}) \\
 & \quad \wedge \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \\
 & \quad \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
 & \quad \quad \quad \text{temp-stk}) \\
 & \quad \wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state}))
 \end{aligned}$$

```


$$\begin{array}{l}
\wedge \text{ signatures-match}(\text{mg-alist } (\text{mg-state}), \text{ name-alist}) \\
\wedge \text{ normal } (\text{mg-state}) \\
\wedge \text{ all-cars-unique } (\text{mg-alist } (\text{mg-state})) \\
\wedge (\neg \text{ resource-errorp } (\text{mg-meaning-r } (\text{stmt}, \\
\quad \quad \quad \text{ proc-list}, \\
\quad \quad \quad \text{ mg-state}, \\
\quad \quad \quad n, \\
\quad \quad \quad \text{ list } (\text{length } (\text{temp-stk}), \\
\quad \quad \quad \quad \quad \quad \text{ p-ctrl-stk-size } (\text{ctrl-stk})))))) \\
\wedge (\neg \text{ normal } (\text{mg-meaning-r } (\text{prog2-left-branch } (\text{stmt}), \\
\quad \quad \quad \text{ proc-list}, \\
\quad \quad \quad \text{ mg-state}, \\
\quad \quad \quad n - 1, \\
\quad \quad \quad \text{ list } (\text{length } (\text{temp-stk}), \\
\quad \quad \quad \quad \quad \quad \text{ p-ctrl-stk-size } (\text{ctrl-stk})))))) \\
\rightarrow (\text{p-state } (\text{tag } ('pc, \\
\quad \quad \quad \text{ cons } (\text{subr}, \\
\quad \quad \quad \quad \quad \text{ if } \text{ normal } (\text{mg-meaning-r } (\text{prog2-left-branch } (\text{stmt}), \\
\quad \quad \quad \quad \quad \quad \text{ proc-list}, \\
\quad \quad \quad \quad \quad \text{ mg-state}, \\
\quad \quad \quad \quad \quad n - 1, \\
\quad \quad \quad \quad \quad \text{ list } (\text{length } (\text{temp-stk}), \\
\quad \quad \quad \quad \quad \quad \text{ p-ctrl-stk-size } (\text{ctrl-stk})))) \\
\quad \quad \quad \text{ then } \text{ length } (\text{code } (\text{translate } (\text{cinfo}, \\
\quad \quad \quad \quad \quad \text{ t-cond-list}, \\
\quad \quad \quad \quad \quad \text{ prog2-left-branch } (\text{stmt}), \\
\quad \quad \quad \quad \quad \text{ proc-list}))) \\
\quad \quad \quad \text{ else } \text{ find-label } (\text{fetch-label } (\text{cc } (\text{mg-meaning-r } (\text{prog2-left-branch } (\text{stmt}), \\
\quad \quad \quad \quad \quad \text{ proc-list}, \\
\quad \quad \quad \quad \quad \text{ mg-state}, \\
\quad \quad \quad \quad \quad n - 1, \\
\quad \quad \quad \quad \quad \text{ list } (\text{length } (\text{temp-stk}), \\
\quad \quad \quad \quad \quad \quad \text{ p-ctrl-stk-size } (\text{ctrl-stk})))), \\
\quad \quad \quad \quad \quad \text{ label-alist } (\text{translate } (\text{cinfo}, \\
\quad \quad \quad \quad \quad \text{ t-cond-list}, \\
\quad \quad \quad \quad \quad \text{ prog2-left-branch } (\text{stmt}), \\
\quad \quad \quad \quad \quad \text{ proc-list}))), \\
\quad \quad \quad \quad \quad \text{ append } (\text{code } (\text{translate } (\text{cinfo}, \\
\quad \quad \quad \quad \quad \text{ t-cond-list}, \\
\quad \quad \quad \quad \quad \text{ prog2-left-branch } (\text{stmt}), \\
\quad \quad \quad \quad \quad \text{ proc-list}))), \\
\quad \quad \quad \quad \quad \text{ append } (\text{code } (\text{translate } (\text{nullify } (\text{translate } (\text{nullify } (\text{cinfo}), \\
\quad \quad \quad \quad \quad \quad \text{ t-cond-list}, \\
\quad \quad \quad \quad \quad \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-nonnornal-left-state2-equals-final.

THEOREM: prog2-right-branch-translation-parameters-ok  
 $((\text{car } (\text{stmt})) = \text{'prog2-mg})$   
 $\wedge \text{ ok-translation-parameters } ((\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}))$   
 $\rightarrow \text{ ok-translation-parameters } (\text{translate } ((\text{cinfo},$   
 $\text{t-cond-list},$   
 $\text{prog2-left-branch } (\text{stmt}),$   
 $\text{proc-list}),$   
 $\text{t-cond-list},$   
 $\text{prog2-right-branch } (\text{stmt}),$

*proc-list,*  
*code2)*

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

THEOREM: prog2-right-branch-hyps  
 $((n \not\leq 0) \wedge (\neg \text{resources-inadequatep}(\text{stmt}, \text{proc-list}, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \wedge (\text{car}(\text{stmt}) = \text{'prog2-mg}) \wedge \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \wedge \text{ok-mg-def-plistp}(\text{proc-list}) \wedge \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \wedge \text{ok-mg-statep}(\text{mg-state}, \text{r-cond-list}) \wedge \text{cond-subsetp}(\text{r-cond-list}, \text{t-cond-list}) \wedge (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \text{code2})) \wedge \text{user-defined-procp}(\text{subr}, \text{proc-list}) \wedge \text{plistp}(\text{temp-stk}) \wedge \text{listp}(\text{ctrl-stk}) \wedge \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \text{bindings}(\text{top}(\text{ctrl-stk})), \text{temp-stk}) \wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state})) \wedge \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \wedge \text{normal}(\text{mg-state}) \wedge \text{all-cars-unique}(\text{mg-alist}(\text{mg-state})) \wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, \text{n}, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk})))))) \wedge \text{normal}(\text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}, \text{proc-list}, \text{mg-state}, \text{n} - 1, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk})))))) \rightarrow (\text{ok-mg-statement}(\text{prog2-right-branch}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk}))))))$

$\wedge \quad \text{ok-translation-parameters}(\text{translate}(\text{cinfo},$   
 $\quad \quad \quad t\text{-cond-list},$   
 $\quad \quad \quad \text{prog2-left-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list}),$   
 $\quad \quad \quad t\text{-cond-list},$   
 $\quad \quad \quad \text{prog2-right-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list},$   
 $\quad \quad \quad \text{code2}))$   
 $\wedge \quad \text{ok-mg-statep}(\text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list},$   
 $\quad \quad \quad \text{mg-state},$   
 $\quad \quad \quad n - 1,$   
 $\quad \quad \quad \text{list}(\text{length}(\text{temp-stk}),$   
 $\quad \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))),$   
 $\quad \quad \quad r\text{-cond-list})$   
 $\wedge \quad (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})))$   
 $\quad = \quad \text{append}(\text{code}(\text{translate}(\text{translate}(\text{cinfo},$   
 $\quad \quad \quad t\text{-cond-list},$   
 $\quad \quad \quad \text{prog2-left-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list}),$   
 $\quad \quad \quad t\text{-cond-list},$   
 $\quad \quad \quad \text{prog2-right-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list})),$   
 $\quad \quad \quad \text{code2}))$   
 $\wedge \quad \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list},$   
 $\quad \quad \quad \text{mg-state},$   
 $\quad \quad \quad n - 1,$   
 $\quad \quad \quad \text{list}(\text{length}(\text{temp-stk}),$   
 $\quad \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))),$   
 $\quad \quad \quad \text{bindings}(\text{top}(\text{ctrl-stk})),$   
 $\quad \quad \quad \text{temp-stk})$   
 $\wedge \quad \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})),$   
 $\quad \quad \quad \text{mg-alist}(\text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list},$   
 $\quad \quad \quad \text{mg-state},$   
 $\quad \quad \quad n - 1,$   
 $\quad \quad \quad \text{list}(\text{length}(\text{temp-stk}),$   
 $\quad \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))))$   
 $\wedge \quad \text{signatures-match}(\text{mg-alist}(\text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}),$   
 $\quad \quad \quad \text{proc-list},$   
 $\quad \quad \quad \text{mg-state},$   
 $\quad \quad \quad n - 1,$

$$\begin{aligned}
& \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))), \\
& \quad \text{name-alist}) \\
\wedge & \text{ all-cars-unique}(\text{mg-alist}(\text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))), \\
\wedge & (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{prog2-right-branch}(\text{stmt}), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))), \\
& \quad n - 1, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \\
\end{aligned}$$

THEOREM: prog2-state2-equals-state3

$$\begin{aligned}
& ((n \not\simeq 0) \\
\wedge & (\neg \text{resources-inadequatep}(\text{stmt}, \\
& \quad \text{proc-list}, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \\
\wedge & (\text{car}(\text{stmt}) = \text{'prog2-mg}) \\
\wedge & \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \\
\wedge & \text{ok-mg-def-plistp}(\text{proc-list}) \\
\wedge & \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \\
\wedge & \text{ok-mg-statep}(\text{mg-state}, \text{r-cond-list}) \\
\wedge & \text{cond-subsetp}(\text{r-cond-list}, \text{t-cond-list}) \\
\wedge & (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \\
& \quad \text{code2})) \\
\wedge & \text{user-defined-procp}(\text{subr}, \text{proc-list}) \\
\wedge & \text{plistp}(\text{temp-stk}) \\
\wedge & \text{listp}(\text{ctrl-stk}) \\
\wedge & \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \\
& \quad \text{bindings}(\text{top}(\text{ctrl-stk})), \\
& \quad \text{temp-stk})) \\
\wedge & \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state})) \\
\wedge & \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist})
\end{aligned}$$

```


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


```

```

                                         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  
 $((n \not\simeq 0) \wedge (\neg \text{resources-inadequatep}(\text{stmt}, \text{proc-list}, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk}))))$   
 $\wedge (\text{car}(\text{stmt}) = \text{'prog2-mg})$   
 $\wedge \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list})$   
 $\wedge \text{ok-mg-def-plistp}(\text{proc-list})$   
 $\wedge \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2})$   
 $\wedge \text{ok-mg-statep}(\text{mg-state}, \text{r-cond-list})$   
 $\wedge \text{cond-subsetp}(\text{r-cond-list}, \text{t-cond-list})$   
 $\wedge (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list}))$   
 $= \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \text{code2}))$   
 $\wedge \text{user-defined-procp}(\text{subr}, \text{proc-list})$   
 $\wedge \text{plistp}(\text{temp-stk})$   
 $\wedge \text{listp}(\text{ctrl-stk})$   
 $\wedge \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \text{bindings}(\text{top}(\text{ctrl-stk})), \text{temp-stk})$   
 $\wedge \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state}))$   
 $\wedge \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist})$   
 $\wedge \text{normal}(\text{mg-state})$   
 $\wedge \text{all-cars-unique}(\text{mg-alist}(\text{mg-state}))$   
 $\wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk}))))))$   
 $\wedge \text{normal}(\text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}, \text{proc-list}, \text{mg-state}, n - 1, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk}))))))$   
 $\rightarrow (\text{p-state}(\text{tag}(\text{'pc}), \text{cons}(\text{subr}, \text{if normal}(\text{mg-meaning-r}(\text{prog2-right-branch}(\text{stmt}, \text{proc-list}, \text{mg-meaning-r}(\text{prog2-left-branch}(\text{stmt}, \text{proc-list}, \text{mg-state}, n - 1, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk})))))))))))$



```

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  
 $((stmt\text{-}time = (left\text{-}time + right\text{-}time)) \wedge (p(initial, left\text{-}time) = state2) \wedge (state2 = state3) \wedge (p(state3, right\text{-}time) = state4) \wedge (state4 = final)) \rightarrow (p(initial, stmt\text{-}time) = final)$

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))
      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
        (tag 'pc
        (cons subr (length (code cinfo))))) ;; state1

```

```

        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-proc 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
      (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-nonnornal-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

- all-cars-unique, 5, 8, 10, 11, 13
- bindings, 4, 6–10, 12, 13, 15, 16
- cc, 1, 2, 5–7, 11, 12, 14–16
- clock, 3
- clock-prog2-left-non-normal, 2
- code, 3–16
- cond-subsetp, 4, 8, 10, 13
- fetch-label, 5, 7, 11, 14, 16
- find-label, 6, 7, 12, 14, 16
- label-alist, 5, 7, 11, 14, 16
- length, 4–16
- map-down, 12
- map-down-values, 6, 7, 12, 15, 16
- mg-alist, 4–13, 15, 16
- mg-cond-to-p-nat, 6, 7, 12, 15, 16
- mg-max-ctrl-stk-size, 6, 7, 12, 15, 16
- mg-max-temp-stk-size, 6, 7, 12, 15, 16
- mg-meaning-r, 1–3, 5–16
- mg-meaning-r-prog2-left-non-normal, 1
- mg-psw, 2
- mg-vars-list-ok-in-p-state, 4, 8–10, 13
- mg-word-size, 6, 7, 12, 15, 16
- no-p-aliasing, 4, 8–10, 13
- normal, 1–3, 5, 6, 8, 11, 13–15
- nullify, 3–6, 11
- ok-mg-def-plistp, 4, 8, 10, 13
- ok-mg-statement, 4, 8–10, 13
- ok-mg-statep, 4, 8–10, 13
- ok-translation-parameters, 3, 4, 7–10, 13
- p, 16
- p-ctrl-stk-size, 4–16
- p-state, 6, 7, 12, 15, 16
- plistp, 4, 8, 10, 13
- prog2-code-rewrite, 3
- prog2-left-branch, 1–15
- prog2-left-branch-code-body-rewrite, 4
- prog2-left-branch-doesnt-halt, 2
- prog2-left-branch-translation-parameters-ok, 3
- prog2-meaning-r-2, 1
- prog2-nonnorma-left-state2-equals-final, 4
- prog2-normal-left-exact-time-schema, 16
- prog2-right-branch, 1–4, 6–10, 12–15
- prog2-right-branch-doesnt-halt, 2
- prog2-right-branch-hyps, 8
- prog2-right-branch-translation-parameters-ok, 7
- prog2-state2-equals-state3, 10
- prog2-state4>equals-final, 13
- prog2-translation-2, 1
- resource-errorp, 1, 2, 5, 8, 10, 11, 13
- resources-inadequatep, 1, 4, 8, 10, 13
- signal-system-error, 1
- signatures-match, 5, 8, 10, 13
- tag, 6, 7, 12, 14, 16
- top, 4, 6–10, 12, 13, 15, 16
- translate, 2–16
- translate-def-body, 4, 8–10, 13
- translate-proc-list, 6, 7, 12, 15, 16
- user-defined-procp, 4, 8, 10, 13