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

THEOREM: prog2-meaning-r-2
\[(\text{car (stmt) } = \text{'prog2-mg}) \rightarrow (\text{mg-meaning-r (stmt, proc-list, mg-state, n, sizes) } = \begin{cases} \text{if } n \approx 0 \text{ then signal-system-error (mg-state, 'timed-out)} \\
\text{elseif } \neg \text{ normal (mg-state) then mg-state} \\
\text{elseif resources-inadequatep (stmt, proc-list, sizes) then signal-system-error (mg-state, 'resource-error)} \\
\text{else mg-meaning-r (prog2-right-branch (stmt), proc-list, mg-meaning-r (prog2-left-branch (stmt), proc-list, mg-state, n - 1, sizes)), n - 1, sizes) endif} \end{cases})\]

THEOREM: mg-meaning-r-prog2-left-non-normal
\[((\text{car (stmt) } = \text{'prog2-mg}) \land (\text{cc (mg-state) } = \text{'normal}) \land (\neg \text{ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)))}) \land (\neg \text{ normal (mg-meaning-r (prog2-left-branch (stmt), proc-list, mg-state, n - 1, sizes)))}) \rightarrow (\text{mg-meaning-r (stmt, proc-list, mg-state, n, sizes) } = \text{mg-meaning-r (prog2-left-branch (stmt), proc-list, mg-state, n - 1, sizes)})\]

THEOREM: prog2-translation-2
\[(\text{car (stmt) } = \text{'prog2-mg})\]
→ (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-doesn’t-halt
((car (stmt) = 'prog2-mg)
 ∧ normal (mg-state)
 ∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes))))
→ (mg-psw (mg-meaning-r (prog2-left-branch (stmt),
  proc-list,
  mg-state,
  n − 1,
  sizes))
  = 'run)

THEOREM: prog2-right-branch-doesn’t-halt
((car (stmt) = 'prog2-mg)
 ∧ normal (mg-state)
 ∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes))))
→ (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)
 ∧ (cc (mg-state) = 'normal)
 ∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes))))
 ∧ (¬ normal (mg-meaning-r (prog2-left-branch (stmt),

\(\text{proc-list,}\)
\(\text{mg-state,}\)
\(n-1,\)
\(\text{sizes})\))

\(\rightarrow\) \(\text{clock (stmt, proc-list, mg-state, n)}\)
\(=\) \(\text{clock (prog2-left-branch (stmt), proc-list, mg-state, n - 1)}\)

**Theorem:** prog2-code-rewrite

\(((\text{car (stmt)} = \text{'prog2-mg})\)
\(\wedge\) \(\text{ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, y)})\)

\(\rightarrow\) \(\text{append (code (translate (cinfo,}\)
\(\text{t-cond-list,}\)
\(\text{prog2-left-branch (stmt),}\)
\(\text{proc-list)}),}\)
\(\text{append (code (translate (nullify (translate (nullify (cinfo),}\)
\(\text{t-cond-list,}\)
\(\text{prog2-left-branch (stmt),}\)
\(\text{proc-list)}),}\)
\(\text{t-cond-list,}\)
\(\text{prog2-right-branch (stmt),}\)
\(\text{proc-list)}),}\)
\(y)\)
\(=\) \(\text{append (code (translate (cinfo, t-cond-list, stmt, proc-list)), y)})\)

**Event:** Disable prog2-code-rewrite.

**Theorem:** prog2-left-branch-translation-parameters-ok

\(((\text{car (stmt)} = \text{'prog2-mg})\)
\(\wedge\) \(\text{ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2)})\)

\(\rightarrow\) \(\text{ok-translation-parameters (cinfo,}\)
\(\text{t-cond-list,}\)
\(\text{prog2-left-branch (stmt),}\)
\(\text{proc-list,}\)
\(\text{append (code (translate (nullify (translate (nullify (cinfo),}\)
\(\text{t-cond-list,}\)
\(\text{prog2-left-branch (stmt),}\)
\(\text{proc-list)}),}\)
\(\text{t-cond-list,}\)
\(\text{prog2-right-branch (stmt),}\)
\(\text{proc-list)}),}\)
\(\text{code2)})\)

**Event:** Disable prog2-left-branch-translation-parameters-ok.
THEOREM: prog2-left-branch-code-body-rewrite
\((\text{car} (\text{stmt}) = \text{prog2-mg})\)
\(\land\) ok-translation-parameters \((\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2})\)
\(\land\) (code (translate-def-body (assoc \((\text{subr}, \text{proc-list})\), \text{proc-list}))
  \(=\) append (code (translate (\(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}\))),
  \text{code2}))
\(\rightarrow\) (code (translate-def-body (assoc \((\text{subr}, \text{proc-list})\), \text{proc-list}))
  \(=\) append (code (translate \(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}\)),
  \text{pro2-left-branch (stmt), proc-list}),
  \text{code2}))

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

THEOREM: prog2-nonnormal-left-state2-equals-final
\((n \not\equiv 0)\)
\(\land\) (\(\neg\) resources-inadequately \((\text{stmt}, \text{proc-list})\))
\(\land\) ok-mg-statement \((\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list})\)
\(\land\) ok-mg-def-plistp \((\text{proc-list})\)
\(\land\) ok-translation-parameters \((\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2})\)
\(\land\) cond-subsetp \((\text{r-cond-list}, \text{t-cond-list})\)
\(\land\) (code (translate-def-body (assoc \((\text{subr}, \text{proc-list})\), \text{proc-list}))
  \(=\) append (code (translate \(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}\))),
  \text{code2}))
\(\land\) user-defined-procp \((\text{subr}, \text{proc-list})\)
\(\land\) plistp \((\text{temp-stk})\)
\(\land\) listp \((\text{ctrl-stk})\)
\(\land\) mg-vars-list-ok-in-p-state \((\text{mg-alist} (\text{mg-state}))\)
\(\land\) bindings (top \((\text{ctrl-stk}))\)
\(\land\) no-p-aliasing (bindings (top (\(\text{ctrl-stk})\)), \text{mg-alist (mg-state)}\)

4
∧ 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-left-branch (stmt), proc-list, mg-state, n − 1, list (length (temp-stk), p-ctrl-stk-size (ctrl-stk)))))
then length (code (translate (cinfo, t-cond-list, prog2-left-branch (stmt), proc-list))))
else find-label (fetch-label (cc (mg-meaning-r (prog2-left-branch (stmt), proc-list, mg-state, n − 1, 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)))),
\[\begin{align*}
&\text{ctrl-stk}, \\
&\text{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))))}, \\
&\text{bindings (top (ctrl-stk)), temp-stk),} \\
&\text{translate-proc-list (proc-list),} \\
&\text{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))}, \\
&\text{MG-MAX-CTRL-STK-SIZE,} \\
&\text{MG-MAX-TEMP-STK-SIZE,} \\
&\text{MG-WORD-SIZE,} \\
&\text{'run)} \\
&= \text{p-state (tag ('pc, cons (subr,} \\
&\text{if normal (mg-meaning-r (stmt, proc-list, mg-state, n, list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))} \\
&\text{then length (code (translate (cinfo, t-cond-list, stmt, proc-list))))} \\
&\text{else find-label (fetch-label (cc (mg-meaning-r (stmt, proc-list, mg-state, n, list (length (temp-stk),}
\begin{verbatim}
label-alist (translate (cinfo, t-cond-list, stmt, proc-list)),
append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
(code2) endif),

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


THEOREM: 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),

\end{verbatim}
EVENT: Disable prog2-right-branch-translation-parameters-ok.

THEOREM: prog2-right-branch-hyps
\[ ((n \neq 0) \land (\not\ resources-inadequatep(stmt, proc-list, list\ (length\ (temp-stk), p-ctrl-stk-size\ (ctrl-stk)))) \land (car\ (stmt) = \ 'prog2-mg) \land ok-mg-statement(stmt, r-cond-list, name-alist, proc-list) \land ok-mg-def-plistp(proc-list) \land ok-translation-parameters(cinfo, t-cond-list, stmt, proc-list, code2) \land ok-mg-statemap(mg-state, r-cond-list) \land cond-subsetp(r-cond-list, t-cond-list) \land \ (code\ (translate-def-body\ assoc\ (subr, proc-list), proc-list)) \not=\ \ append\ (code\ (translate\ (cinfo, t-cond-list, stmt, proc-list)), code2)) \land user-defined-procp(subr, proc-list) \land plistp(temp-stk) \land listp(ctrl-stk) \land mg-vars-list-ok-in-p-state(mg-alist\ (mg-state), bindings\ (top\ (ctrl-stk)), temp-stk) \land no-p-aliasing(bindings\ (top\ (ctrl-stk)), mg-alist\ (mg-state)) \land signatures-match(mg-alist\ (mg-state), name-alist) \land normal(mg-state) \land all-cars-unique(mg-alist\ (mg-state)) \land (\not\ resource-errorp(mg-meaning-r(stmt, proc-list, mg-state, n, list\ (length\ (temp-stk), p-ctrl-stk-size\ (ctrl-stk)))) \land normal(mg-meaning-r(prog2-left-branch(stmt), proc-list, mg-state, n - 1, list\ (length\ (temp-stk), p-ctrl-stk-size\ (ctrl-stk)))) \rightarrow (ok-mg-statement(prog2-right-branch(stmt), r-cond-list, name-alist),
\( \text{proc-list} \)
\( \land \) ok-translation-parameters (translate (\( \text{cinfo} \),
\( t-cond-list \),
\( \text{prog2-left-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( t-cond-list \),
\( \text{prog2-right-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( \text{code2} \))
\( \land \) ok-mg-statep (mg-meaning-r (\( \text{prog2-left-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( \text{mg-state} \),
\( n - 1 \),
\( \text{list} (\text{length} (\text{temp-stk}),
\( \text{p-ctrl-stk-size} (\text{ctrl-stk})) \)),
\( r-cond-list \))
\( \land \) (code (translate-def-body (assoc (\( \text{subr} \),
\( \text{proc-list} \)),
\( \text{proc-list} \))
= append (code (translate (translate (\( \text{cinfo} \),
\( t-cond-list \),
\( \text{prog2-left-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( t-cond-list \),
\( \text{prog2-right-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( \text{code2} \)),
\( \text{code2} \))
\( \land \) mg-vars-list-ok-in-p-state (mg-alist (mg-meaning-r (\( \text{prog2-left-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( \text{mg-state} \),
\( n - 1 \),
\( \text{list} (\text{length} (\text{temp-stk}),
\( \text{p-ctrl-stk-size} (\text{ctrl-stk})) \)),
\( \text{bindings} \) (top (\( \text{ctrl-stk} \)),
\( \text{temp-stk} \))
\( \land \) no-p-aliasing (bindings (top (\( \text{ctrl-stk} \))),
\( \text{mg-alist} \) (mg-meaning-r (\( \text{prog2-left-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( \text{mg-state} \),
\( n - 1 \),
\( \text{list} (\text{length} (\text{temp-stk}),
\( \text{p-ctrl-stk-size} (\text{ctrl-stk})) \))),
\( \text{bindings} \) (top (\( \text{ctrl-stk} \)),
\( \text{temp-stk} \))
\( \land \) signatures-match (mg-alist (mg-meaning-r (\( \text{prog2-left-branch} (\text{stmt}) \),
\( \text{proc-list} \),
\( \text{mg-state} \),
\( n - 1 \),
\( \text{list} (\text{length} (\text{temp-stk}),
\( \text{p-ctrl-stk-size} (\text{ctrl-stk})) \))),
\( \text{bindings} \) (top (\( \text{ctrl-stk} \)),
\( \text{temp-stk} \))
Theorem: prog2-state2-equals-state3
\((n \not\equiv 0)\) \\
\(\land \neg \text{resources-inadequatep}(stmt,\) \\
\(\quad \text{proc-list},\) \\
\(\quad \text{list (length (temp-stk),}\) \\
\(\quad \quad \text{p-ctrl-stk-size (ctrl-stk))))\) \\
\(\land (\text{car (stmt)} = \text{'prog2-mg})\) \\
\(\land \text{ok-mg-statement}(stmt, \text{r-cond-list, name-alist, proc-list})\) \\
\(\land \text{ok-mg-def-plistp}(\text{proc-list})\) \\
\(\land \text{ok-translation-parameters}(\text{cinfo, t-cond-list, stmt, proc-list, code2})\) \\
\(\land \text{ok-mg-statep}(\text{mg-state, r-cond-list})\) \\
\(\land \text{cond-subsetp}(\text{r-cond-list, t-cond-list})\) \\
\(\land (\text{code (translate-def-body (assoc (subr, proc-list), proc-list))} = \text{append (code (translate (cinfo, t-cond-list, stmt, proc-list)), code2)})\) \\
\(\land \text{user-defined-procp}(\text{subr, proc-list})\) \\
\(\land \text{plistp (temp-stk)}\) \\
\(\land \text{listp (ctrl-stk)}\) \\
\(\land \text{mg-vars-list-ok-in-p-state}(\text{mg-alist (mg-state),}\) \\
\(\quad \text{bindings (top (ctrl-stk)),}\) \\
\(\quad \text{temp-stk})\) \\
\(\land \text{no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))}\) \\
\(\land \text{signatures-match (mg-alist (mg-state), name-alist)}\)
∧ 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-left-branch (stmt), proc-list, mg-state, n - 1, list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
then length (code (translate (cinfo, t-cond-list, prog2-left-branch (stmt), proc-list)))
else find-label (fetch-label (cc (mg-meaning-r (prog2-left-branch (stmt), proc-list, mg-state, n - 1, 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))

\textbf{code2}})) \textbf{endif}),

\textit{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))

\textbf{Event:} Disable \texttt{prog2-state2-equals-state3}. 

12
Theorem: prog2-state4-equals-final

\( (n \neq 0) \land \neg \text{resources-inadequate}(\text{stmt}, \text{proc-list}, \text{list(length(temp-stk), p-ctrl-stk-size(ctrl-stk)))) \land (\text{car(stmt)} = \text{prog2-mg}) \land \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \land \text{ok-mg-def-plistp}(\text{proc-list}) \land \text{ok-translation-parameters}(\text{cinfo, t-cond-list, stmt, proc-list, code2}) \land \text{ok-mg-statep}(\text{mg-state, r-cond-list}) \land \text{cond-subsetp}(\text{r-cond-list, t-cond-list}) \land (\text{code}(\text{translate-def-body}(\text{assoc(subr, proc-list), proc-list})) = \text{append}(\text{code}(\text{translate}(\text{cinfo, t-cond-list, stmt, proc-list}))), \text{code2})) \land \text{user-defined-procp}(\text{subr, proc-list}) \land \text{plistp(temp-stk)} \land \text{listp(ctrl-stk)} \land \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \text{bindings(top(ctrl-stk), temp-stk)}) \land \text{no-p-aliasing}(\text{bindings(top(ctrl-stk), mg-alist(\text{mg-state}))}) \land \text{signatures-match}(\text{mg-alist(\text{mg-state}, name-alist}) \land \text{normal}(\text{mg-state}) \land \text{all-cars-unique}(\text{mg-alist(\text{mg-state}))} \land (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{list(length(temp-stk), p-ctrl-stk-size(ctrl-stk))}))) \land \text{normal}(\text{mg-meaning-r}(\text{prog2-left-branch(stmt)}, \text{proc-list}, \text{mg-state}, n - 1, \text{list(length(temp-stk), p-ctrl-stk-size(ctrl-stk))}))) \rightarrow (\text{p-state}(\text{tag(’pc, cons(subr, if normal(\text{mg-meaning-r}(\text{prog2-right-branch(stmt)}, \text{proc-list}, \text{mg-meaning-r}(\text{prog2-left-branch(stmt)}, \text{proc-list}, \text{mg-state}, n - 1, \text{list(length(temp-stk), p-ctrl-stk-size(ctrl-stk))}))))}))
\textbf{then} length (code (translate (translate ($cinfo$, $t\text{-}cond\text{-}list$), 
pro2\text{-}left\text{-}branch ($stmt$), 
proc\text{-}list),
$t\text{-}cond\text{-}list$,
pro2\text{-}right\text{-}branch ($stmt$), 
proc\text{-}list)))

\textbf{else} find-label (fetch-label (cc (mg\text{-}meaning\text{-}r (pro2\text{-}right\text{-}branch ($stmt$), 
proc\text{-}list),
mg\text{-}meaning\text{-}r (pro2\text{-}left\text{-}branch ($stmt$), 
proc\text{-}list),
mg\text{-}state, 
$n - 1$, 
list (length ($temp\text{-}stk$), 
p-ctrl-stk\text{-}size ($ctrl\text{-}stk$))))

$n - 1$, 
list (length ($temp\text{-}stk$), 
p-ctrl-stk\text{-}size ($ctrl\text{-}stk$)))

\textbf{endif},

$ctrl\text{-}stk$,
map\text{-}down\text{-}values (mg\text{-}alist (mg\text{-}meaning\text{-}r (pro2\text{-}right\text{-}branch ($stmt$), 
proc\text{-}list),
mg\text{-}meaning\text{-}r (pro2\text{-}left\text{-}branch ($stmt$), 
proc\text{-}list),
mg\text{-}state, 
$code2$))

14
\[ n - 1, \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))),} \]
\[ n - 1, \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))),} \]
\[ \text{bindings (top (ctrl-stk)),} \]
\[ \text{temp-stk),} \]
\[ \text{translate-proc-list (proc-list),} \]
\[ \text{list (list (}} \]
\[ \text{mg-cond-to-p-nat (cc (mg-meaning-r (prog2-right-branch (stmt),} \]
\[ \text{proc-list,} \]
\[ \text{mg-meaning-r (prog2-left-branch (stmt),} \]
\[ \text{proc-list,} \]
\[ \text{mg-state,} \]
\[ n - 1, \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))),} \]
\[ n - 1, \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))),} \]
\[ t\text{-cond-list))),} \]
\[ \text{MG-MAX-CTRL-STK-SIZE,} \]
\[ \text{MG-MAX-TEMP-STK-SIZE,} \]
\[ \text{MG-WORD-SIZE,} \]
\[ '\text{run)} \]
\[ = \text{p-state (tag ('pc,} \]
\[ \text{cons (subr,} \]
\[ \text{if normal (mg-meaning-r (stmt,} \]
\[ \text{proc-list,} \]
\[ \text{mg-state,} \]
\[ n,} \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk)))),} \]
\[ \text{then length (code (translate (cinfo,} \]
\[ t\text{-cond-list,} \]
\[ stmt,} \]
\[ \text{proc-list))))} \]
\[ \text{else find-label (fetch-label (cc (mg-meaning-r (stmt,} \]
\[ \text{proc-list,} \]
\[ \text{mg-state,} \]
\[ n,} \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk)))),} \]
\[ 15 \]
Event: Disable prog2-state4-equals-final.

Theorem: prog2-normal-left-exact-time-schema

\[(stmt-time = (left-time + right-time)) \land (p(initial, left-time) = state2) \land (state2 = state3) \land (p(state3, right-time) = state4) \land (state4 = final) \rightarrow (p(initial, stmt-time) = final)\]

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) code2)
        proc-list)
      t-cond-list
    (append (code (translate (nullify (translate (nullify cinfo)
        proc-list)
      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)
  20
(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)))))
 (list (length temp-stk)
 (p-ctrl-stk-size ctrl-stk)))
 (sub1 n)
 (list (length temp-stk)
 (p-ctrl-stk-size ctrl-stk)))
 (sub1 n)
 (list (length temp-stk)
 (p-ctrl-stk-size ctrl-stk))))
 (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)))))
 (list (length temp-stk)
 (p-ctrl-stk-size ctrl-stk)))
 (sub1 n)
 (list (length temp-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)))
      (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

22
(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-ctr1-stk-size ctrl-stk)))
  (sub1 n)
  (list (length temp-stk)
    (p-ctr1-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-ctr1-stk-size ctrl-stk)))
      (sub1 n)
      (list (length temp-stk)
        (p-ctr1-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-ctr1-stk-size ctrl-stk))))))
    )
  )
)
(defun 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 ;; 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

24
(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))))
      (len (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))))

27
(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)))
        28)
(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))))))
(drop 0)
(demote 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)
              (...))
(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)
(dive 1)
(rewrite prog2-right-branch-hyps)
top s
(rewrite prog2-right-branch-hyps)
(rewrite prog2-right-branch-hyps)
(rewrite prog2-right-branch-hyps)
(dive 1)
(rewrite prog2-right-branch-hyps)
top s
(dive 1)
(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

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