Event: Start with the library "c-loop".

Theorem: if-condition-boolean-identifierp
((car (stmt) = 'if-mg)
∧ ok-mg-statep (mg-state, cond-list)
∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
∧ signatures-match (mg-alist (mg-state), name-alist))
→ boolean-identifierp (if-condition (stmt), mg-alist (mg-state))

Theorem: fetch-boolean-identifier-ins-okp
(boolean-identifierp (b, mg-vars)
∧ all-cars-unique (mg-vars)
∧ (length (temp-stk) < MG-MAX-TEMP-STK-SIZE)
∧ mg-vars-list-ok-in-p-state (mg-vars, bindings, temp-stk))
→ ((type (value (b, bindings)) = ’nat)
∧ listp (value (b, bindings))
∧ (cddr (value (b, bindings)) = nil)
∧ small-naturalp (untag (value (b, bindings)), 32)
∧ ((untag (value (b, bindings)) < (1 + length (temp-stk))) = t))

Theorem: if-translation-2
(car (stmt) = 'if-mg)
→ (translate (cinfo, cond-list, stmt, proc-list)
   = add-code (translate (add-code (translate (make-cinfo (append (code (cinfo),
               list (list ('push-local,
                    if-condition (stmt)),
               ' (fetch-temp-stk),
               list ('test-bool-and-jump,
                    'false,
                    label-alist (cinfo)),
               1 + (1 + label-cnt (cinfo)))),
               cond-list,
               if-true-branch (stmt),
               proc-list),
               list (list ('jump,
               1 + label-cnt (cinfo)),
list ('dl, label-cnt (cinfo), nil, '(no-op))),
cond-list,
if-false-branch (stmt), proc-list),
list (list ('dl, 1 + label-cnt (cinfo), nil, '(no-op))))

**Theorem:** if-meaning-r-2

\[
\text{car } (\text{stmt}) = '\text{if-mg}\]
\[\rightarrow (\text{mg-meaning-r } (\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{sizes}))
\]
\[
= \begin{cases} 
\text{if } n \simeq 0 \text{ then signal-system-error } (\text{mg-state}, '\text{timed-out}) \\
\text{elseif } \neg \text{normal } (\text{mg-state}) \text{ then } \text{mg-state} \\
\text{elseif } \text{resources-inadequatep } (\text{stmt}, \text{proc-list}, \text{sizes}) \\
\text{then signal-system-error } (\text{mg-state}, '\text{resource-error}) \\
\text{elseif } \text{mg-expression-falsep } (\text{if-condition } (\text{stmt}), \text{mg-state}) \\
\text{then mg-meaning-r } (\text{if-false-branch } (\text{stmt}), \\
\text{proc-list}, \text{mg-state}, n - 1, \text{sizes}) \\
\text{else mg-meaning-r } (\text{if-true-branch } (\text{stmt}), \\
\text{proc-list}, \text{mg-state}, n - 1, \text{sizes}) \end{cases} \text{ endif}
\]

**Theorem:** if-false-branch-doesnt-halt

\[
\left( (n \not\equiv 0) \right. \\
\land \left( \text{car } (\text{stmt}) = '\text{if-mg} \right) \\
\land \left( \text{normal } (\text{mg-state}) \right) \\
\land \left( \text{mg-expression-falsep } (\text{if-condition } (\text{stmt}), \text{mg-state}) \right) \\
\land \left( \neg \text{resource-errorp } (\text{mg-meaning-r } (\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{sizes}))) \right) \\
\rightarrow (\text{mg-psw } (\text{mg-meaning-r } (\text{if-false-branch } (\text{stmt}), \\
\text{proc-list}, \text{mg-state}, n - 1, \text{sizes})))
\]
\[
= '\text{run}
\]

**Theorem:** if-true-branch-doesnt-halt

\[
\left( (n \not\equiv 0) \right. \\
\land \left( \text{car } (\text{stmt}) = '\text{if-mg} \right) \\
\land \left( \text{normal } (\text{mg-state}) \right) \\
\rightarrow (\text{mg-meaning-r } (\text{if-true-branch } (\text{stmt}), \\
\text{proc-list}, \text{mg-state}, n - 1, \text{sizes})))
\]

\[
= '\text{run}
\]
∧ (¬ mg-expression-falsep (if-condition (stmt), mg-state))
∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)))
→ (mg-psw (mg-meaning-r (if-true-branch (stmt),
proc-list,
mg-state,
 n − 1,
 sizes))
   = 'run)

THEOREM: if-code-rewrite1
((car (stmt) = 'if-mg)
∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list))
→ (append (code (translate (cinfo, t-cond-list, stmt, proc-list)), code2)
   = append (code (translate (add-code (translate (make-cinfo (append (code (cinfo),
list (list ('push-local, if-condition (stmt)),
  '(fetch-temp-stk),
list ('test-bool-and-jump, 'false,
  label-cnt (cinfo))),(cinfo)),
  label-alist (cinfo),
1 + (1 + label-cnt (cinfo))),
t-cond-list,
if-true-branch (stmt),
proc-list),
list (list ('jump, 1 + label-cnt (cinfo)),
  cons ('dl,
    cons (label-cnt (cinfo),
      '(nil (no-op))))),
  t-cond-list,
if-false-branch (stmt),
proc-list),
  cons (cons ('dl,
    cons (1 + label-cnt (cinfo),
      'l (no-op)))),
  code2))

THEOREM: if-code-rewrite2
((car (stmt) = 'if-mg)
∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list))
→ (append (code (translate (cinfo, t-cond-list, stmt, proc-list)), code2)
   = append (code (translate (make-cinfo (append (code (cinfo),

list (list ('push-local, 
    if-condition (stmt), 
    '(fetch-temp-stk), 
    list ('test-bool-and-jump, 
        'false, 
        label-cnt (cinfo))))), 

    label-alist (cinfo), 
    1 + (1 + label-cnt (cinfo))), 

    t-cond-list, 
    if-true-branch (stmt), 
    proc-list)), 

    cons (list ('jump, 1 + label-cnt (cinfo)), 
    cons (cons ('dl, 
        cons (label-cnt (cinfo), 
            'nil (no-op))), 
        append (code (translate (nullify (translate (make-cinfo (nil, 
            label-alist (cinfo), 
            1 + (1 + label-cnt (cinfo)) 
            t-cond-list, 
            if-true-branch (stmt), 
            proc-list))), 
    t-cond-list, 
    if-true-branch (stmt), 
    proc-list)), 

    cons (cons ('dl, 
    cons (1 + label-cnt (cinfo), 
        'nil (no-op))), 

    code2))))))

THEOREM: if-false-branch-hyps
((n ≠ 0) 
  ∧ (car (stmt) = 'if-mg) 
  ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list) 
  ∧ ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2) 
  ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list)) 
    = append (code (translate (cinfo, t-cond-list, stmt, proc-list)), 
    code2)) 
  ∧ user-defined-procp (subr, proc-list) 
  ∧ normal (mg-state) 
  ∧ (¬ resource-errorp (mg-meaning-r (stmt, 
    proc-list, 
    mg-state, 
    n), 
    list (length (temp-stk)), 
    4)
\[ p-ctrl-stk-size (\text{ctrl-stk})) \]

\[ \land \quad \text{mg-expression-falsep (if-condition (stmt), mg-state))} \]

\[ \rightarrow \quad \text{ok-mg-statement (if-false-branch (stmt),} \]

\[ r-\text{cond-list}, \]

\[ \text{name-alist,} \]

\[ \text{proc-list}) \]

\[ \land \quad \text{ok-translation-parameters (add-code (translate (make-cinfo append (code (cinfo),} \]

\[ \text{list (list ('push-local,} \]

\[ \text{if-condition (stmt)),} \]

\[ '(\text{fetch-temp-stk),} \]

\[ \text{list ('test-bool-and-jump} \]

\[ '\text{false,} \]

\[ \text{label-cnt (cinfo)))),} \]

\[ \text{label-alist (cinfo),} \]

\[ 1 + (1 + \text{label-cnt (cinfo)))),} \]

\[ t-\text{cond-list}, \]

\[ \text{if-true-branch (stmt),} \]

\[ \text{proc-list),} \]

\[ \text{list (list ('jump,} \]

\[ 1 + \text{label-cnt (cinfo),} \]

\[ \text{cons ('dl,} \]

\[ \text{cons (label-cnt (cinfo),} \]

\[ '(\text{nil (no-op))))))),} \]

\[ t-\text{cond-list}, \]

\[ \text{if-false-branch (stmt),} \]

\[ \text{proc-list),} \]

\[ \text{cons (cons ('dl,} \]

\[ \text{cons (1 + \text{label-cnt (cinfo),} \]

\[ (\text{nil (no-op))))))),} \]

\[ \text{code2))} \]

\[ \land \quad \text{(code (translate-def-body (assoc (subr, proc-list), proc-list))} \]

\[ = \quad \text{append (code (translate (add-code (translate (make-cinfo append (code (cinfo),} \]

\[ \text{list (list ('push-local,} \]

\[ \text{if-condition (stmt)),} \]

\[ '(\text{fetch-temp-stk),} \]

\[ \text{list ('test-bool-and-jump} \]

\[ '\text{false,} \]

\[ \text{label-cnt (cinfo)))),} \]

\[ \text{label-alist (cinfo),} \]

\[ 1 + (1 + \text{label-cnt (cinfo)))),} \]

\[ t-\text{cond-list}, \]

\[ \text{if-true-branch (stmt),} \]

\[ \text{proc-list),} \]

\[ \text{code2}}) \]

\[ 5 \]
Theorem: if-true-branch-hyps

\[(n \neq 0) \land (\text{car}(\text{stmt}) = 'if-mg) \land \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \land \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \land (\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})) \land \text{user-defined-procp}(\text{subr}, \text{proc-list}) \land \text{normal}(\text{mg-state}) \land (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, n - 1, \text{list}(\text{length}(\text{temp-stk}), p-\text{ctrl-stk-size}(\text{ctrl-stk})))))) \land (\neg \text{mg-expression-falsep}(\text{if-condition}(\text{stmt}, \text{mg-state})))) \rightarrow (\text{ok-mg-statement}(\text{if-true-branch}(\text{stmt}), \text{r-cond-list}, \text{name-alist}, \text{proc-list})) \land \text{ok-translation-parameters}(\text{make-cinfo}(\text{append}(\text{code}(\text{cinfo})), \text{list}(\text{'push-local}, \text{if-condition}(\text{stmt})))),
'(fetch-temp-stk),
  list ('test-bool-and-jump,
    'false,
    label-cnt (cinfo))),
label-alist (cinfo),
1 + (1 + label-cnt (cinfo))",

\[ t \text{-cond-list}, \]
if-true-branch (stmt),
proc-list,
cons (list ('jump,
  1 + label-cnt (cinfo)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
      cons (label-cnt (cinfo),
  ')(nil
  (no-op)))),
append (code (translate (nullify (translate (make-cinfo (nil,
        label-alist (cinfo),
        1 + (1 + label-cnt (cinfo)))))),
  t \text{-cond-list},
if-true-branch (stmt),
proc-list)),
  cons (cons ('dl,
\[
\text{cons}(\text{label-cnt}(cinfo),
\quad \text{'(nil \ (no-op)))},
\]
\[
\text{append} \ (\text{code} \ \text{(translate} \ \text{(nullify} \ \text{(translate} \ \text{(make-cinfo} \ (nil, \label-alist \ (cinfo), \quad 1 + (1 + \label-cnt \ (cinfo), \quad t-cond-list, \quad \text{if-true-branch} \ (stmt), \quad proc-list)), \quad t-cond-list, \quad \text{if-false-branch} \ (stmt), \quad proc-list)), \quad \text{cons}(\text{cons}(\text{'d1,} \quad \text{cons}(1 + \label-cnt \ (cinfo), \quad \text{'(nil \ (no-op))))}, \quad \text{code2))}))\]
\& \quad (\neg \text{resource-errorp} \ (\text{mg-meaning-r} \ (\text{if-true-branch} \ (stmt), \quad 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}))))))
\]

**Theorem:** if-initial-step1

\[
((n \not\equiv 0) \quad \& \quad (\text{car} \ (stmt) = \text{'if-mg}) \quad \& \quad (\neg \text{resources-inadequatep} \ (stmt, \quad proc-list, \quad \text{list} \ (\text{length} \ \text{(temp-stk)}, \quad \text{p-ctrl-stk-size} \ (\text{ctrl-stk}))))
\]
\[
\& \quad \text{ok-mg-statement} \ (stmt, \ r-cond-list, \ name-alist, \ proc-list) \quad \& \quad \text{mg-vars-list-ok-in-p-state} \ (mg-alist \ (\text{mg-state}), \quad \text{bindings} \ \text{(top} \ \text{(ctrl-stk)})), \quad \text{temp-stk})
\]
\[
\& \quad \text{ok-mg-statep} \ (\text{mg-state}, \ r-cond-list) \quad \& \quad \text{ok-mg-def-plistp} \ (proc-list)
\]
\[
\& \quad (\text{code} \ \text{(translate-def-body} \ \text{(assoc} \ (\text{subr}, \ proc-list), \ proc-list))) \quad = \quad \text{append} \ (\text{code} \ \text{(translate} \ \text{(cinfo}, \ t-cond-list, \ stmt, \ proc-list)), \quad \text{code2}))
\]
\[
\& \quad \text{user-defined-procp} \ (\text{subr}, \ proc-list) \quad \& \quad \text{normal} \ (\text{mg-state}) \quad \& \quad (\neg \text{resource-errorp} \ (\text{mg-meaning-r} \ (stmt, \quad proc-list, \quad \text{mg-state},
\]

8
\[
\begin{align*}
&\rightarrow (p\text{-}step (map\text{-}down (mg\text{-}state, proc\text{-}list, ctrl\text{-}stk, temp\text{-}stk, \\
&\quad \text{tag ('pc, cons (subr, length (code (cinfo))), t-cond-list]))) \\
&= p\text{-}state (tag ('pc, cons (subr, length (code (cinfo)) + 1)), ctrl\text{-}stk, \\
&\quad \text{push (value (if-condition (stmt), bindings (top (ctrl\text{-}stk)))),} \\
&\quad \text{map\text{-}down\text{-}values (mg\text{-}alist (mg\text{-}state),} \\
&\quad \quad \text{bindings (top (ctrl\text{-}stk)),} \\
&\quad \quad \text{temp\text{-}stk}),} \\
&\quad \text{translate\text{-}proc\text{-}list (proc\text{-}list),} \\
&\quad \text{list (list ('c-c,} \\
&\quad \quad \text{mg\text{-}cond\text{-}to\text{-}p\text{-}nat (cc (mg\text{-}state), t-cond-list))},} \\
&\quad \text{MG\text{-}MAX\text{-}CTRL\text{-}STK\text{-}SIZE,} \\
&\quad \text{MG\text{-}MAX\text{-}TEMP\text{-}STK\text{-}SIZE,} \\
&\quad \text{MG\text{-}WORD\text{-}SIZE,} \\
&\quad \text{'run}))
\end{align*}
\]

**Theorem**: if-initial-step2

\[
((n \not\equiv 0) \\
\land (\text{car (stmt) = 'if-mg}) \\
\land (\neg \text{resources\text{-}inadequatep (stmt,} \\
\quad \text{proc\text{-}list,} \\
\quad \text{list (length (temp\text{-}stk),} \\
\quad \quad \text{p\text{-}ctrl\text{-}stk\text{-}size (ctrl\text{-}stk))))} \\
\land \text{ok\text{-}mg\text{-}statement (stmt, r-cond-list, name\text{-}alist, proc\text{-}list)} \\
\land \text{mg\text{-}vars\text{-}list\text{-}ok\text{-}in\text{-}p\text{-}state (mg\text{-}alist (mg\text{-}state),} \\
\quad \text{bindings (top (ctrl\text{-}stk)),} \\
\quad \text{temp\text{-}stk)} \\
\land \text{ok\text{-}mg\text{-}statep (mg\text{-}state, r-cond-list)} \\
\land \text{signatures\text{-}match (mg\text{-}alist (mg\text{-}state), name\text{-}alist)} \\
\land \text{ok\text{-}mg\text{-}def\text{-}listp (proc\text{-}list)} \\
\land \text{all\text{-}cars\text{-}unique (mg\text{-}alist (mg\text{-}state))} \\
\land (\text{code (translate\text{-}def\text{-}body (assoc (subr, proc\text{-}list), proc\text{-}list)})} \\
\quad = \text{append (code (translate (cinfo, t-cond-list, stmt, proc\text{-}list)),} \\
\quad \quad \text{code2))} \\
\land \text{user\text{-}defined\text{-}procp (subr, proc\text{-}list)} \\
\land \text{normal (mg\text{-}state)} \\
\land (\neg \text{resource\text{-}errorp (mg\text{-}meaning\text{-}r (stmt,} \\
\quad \text{...}) ...)
\]

9
\[
\text{proc-list},
\text{mg-state},
n,
\text{list (length (temp-stk),}
\text{p-ctrl-stk-size (ctrl-stk))})
\rightarrow (p\text{-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 1)),}
\text{ctrl-stk,}
\text{push (value (if-condition (stmt), bindings (top (ctrl-stk))),}
\text{map-down-values (mg-alist (mg-state),}
\text{bindings (top (ctrl-stk)),
\text{temp-stk))},
\text{translate-proc-list (proc-list),}
\text{list (list ('c-c,}
\text{mg-cond-to-p-nat (cc (mg-state), t-cond-list))},
\text{MG-MAX-CTRL-STK-SIZE,}
\text{MG-MAX-TEMP-STK-SIZE,}
\text{MG-WORD-SIZE,}
\text{'run}))}
= \text{p-state (tag ('pc, cons (subr, length (code (cinfo)) + 2)),}
\text{ctrl-stk,}
\text{push (rget (untag (value (if-condition (stmt),}
\text{bindings (top (ctrl-stk))))),}
\text{map-down-values (mg-alist (mg-state),}
\text{bindings (top (ctrl-stk)),
\text{temp-stk))},
\text{map-down-values (mg-alist (mg-state),}
\text{bindings (top (ctrl-stk)),
\text{temp-stk))},
\text{translate-proc-list (proc-list),}
\text{list (list ('c-c,}
\text{mg-cond-to-p-nat (cc (mg-state), t-cond-list))},
\text{MG-MAX-CTRL-STK-SIZE,}
\text{MG-MAX-TEMP-STK-SIZE,}
\text{MG-WORD-SIZE,}
\text{'run}))}
\]

**Theorem:** boolean-value-maps-down
(\text{boolean-identifierp (b, mg-vars) \land mg-vars-list-ok-in-p-state (mg-vars, bindings, temp-stk) \land mg-alistp (mg-vars) \land no-p-aliasing (bindings, mg-vars) \land all-cars-unique (mg-vars))
\rightarrow (rget (untag (value (b, bindings)),
\text{map-down-values (mg-vars, bindings, temp-stk))}

10
Theorem: if-find-labelp-lemma1
\[ ((\text{car} (\text{stmt}) = \text{'if-mg}) \land \text{ok-mg-statement} (\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \land \text{ok-translation-parameters} (\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2})) \rightarrow \neg \text{find-labelp} (\text{label-cnt} (\text{cinfo}), \text{code} (\text{translate} (\text{make-cinfo} (\text{append} (\text{code} (\text{cinfo}), \text{list (list ('push-local, 'if-condition (\text{stmt})), 'fetch-temp-stk)), list (list ('test-bool-and-jump, 'false, \text{label-cnt} (\text{cinfo})))), \text{t-cond-list}, 1 + (1 + \text{label-cnt} (\text{cinfo}))), \text{if-true-branch} (\text{stmt}), \text{proc-list}))))) \]

Theorem: if-initial-step3-false-test
\[ ((n \not\equiv 0) \land (\text{car} (\text{stmt}) = \text{'if-mg}) \land (\neg \text{resources-inadequatep} (\text{stmt}, \text{proc-list}, \text{list (length (\text{temp-stk}), \text{p-ctrl-stk-size (\text{ctrl-stk})}))}) \land \text{ok-mg-statement} (\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \land \text{mg-vars-list-ok-in-p-state} (\text{mg-alist} (\text{mg-state}), \text{bindings (top (\text{ctrl-stk})), \text{temp-stk}}) \land \text{ok-mg-statep} (\text{mg-state}, \text{r-cond-list}) \land \text{ok-translation-parameters} (\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \land \text{signatures-match} (\text{mg-alist} (\text{mg-state}), \text{name-alist}) \land \text{ok-mg-def-plistp} (\text{proc-list}) \land \text{all-cars-unique} (\text{mg-alist} (\text{mg-state})) \land (\text{code} (\text{translate-def-body} (\text{assoc (subr, proc-list)}), \text{proc-list}))) = \text{append} (\text{code} (\text{translate} (\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}))), \text{code2})) \land \text{user-defined-procp} (\text{subr}, \text{proc-list}) \land \text{normal (mg-state)} \land \text{no-p-aliasing} (\text{bindings (top (\text{ctrl-stk})), mg-alist (mg-state)}) \land (\neg \text{resource-errorp} (\text{mg-meaning-r} (\text{stmt}, \text{proc-list}))) \]
\[
\begin{align*}
\text{mg-state}, \\
n, \\
\text{list (length (temp-stk),} \\
p-ctrl-stk-size (ctrl-stk))))
\end{align*}
\]
\[
\land \quad \text{mg-expression-falsep (if-condition (stmt, mg-state))}
\]
\[
\rightarrow (p\text{-step (p-state (tag (pc, cons (subr, length (code (cinfo)) + 2)),} \\
\text{ctrl-stk),} \\
push (rget (untag (value (if-condition (stmt),} \\
\text{bindings (top (ctrl-stk))))),} \\
\text{map-down-values (mg-alist (mg-state),} \\
\text{bindings (top (ctrl-stk), temp-stk))},} \\
\text{map-down-values (mg-alist (mg-state),} \\
\text{bindings (top (ctrl-stk), temp-stk))},} \\
\text{translate-proc-list (proc-list),} \\
\text{list (list (c-c,} \\
\text{mg-cond-to-p-nat (cc (mg-state), t-cond-list))},} \\
\text{MG-MAX-CTRL-STK-SIZE,} \\
\text{MG-MAX-TEMP-STK-SIZE,} \\
\text{MG-WORD-SIZE,} \\
\text{'run)})
\]
\[
= \quad \text{p-state (tag (pc,} \\
\text{cons (subr,} \\
\text{length (code (translate (make-cinfo (append (code (cinfo),} \\
\text{list (list (push-local,} \\
\text{if-condition (stmt),} \\
\text{'(fetch-temp-stk),} \\
\text{list ('test-bool-and-jump,} \\
\text{'false,} \\
\text{label-cnt (cinfo))},} \\
\text{label-alist (cinfo),} \\
\text{1 + (1 + label-cnt (cinfo))},} \\
\text{t-cond-list,} \\
\text{if-true-branch (stmt),} \\
\text{proc-list}))})} \\
\text{+ 1)),} \\
\text{ctrl-stk,} \\
\text{map-down-values (mg-alist (mg-state),} \\
\text{bindings (top (ctrl-stk), temp-stk))},} \\
\text{translate-proc-list (proc-list),} \\
\text{list (list (c-c,} \\
\text{mg-cond-to-p-nat (cc (mg-state), t-cond-list))},}
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
Theorem: if-initial-step4-false-test

\[(n \not= 0) \land (\text{car}(\text{stmt}) = '\text{if-mg}) \land \neg \text{resources-inadequatep}(\text{stmt}, \text{proc-list}, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk})))) \land \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \land \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \text{bindings}(\text{top}(\text{ctrl-stk})), \text{temp-stk}) \land \text{ok-mg-statem}(\text{mg-state}, \text{r-cond-list}) \land \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \land \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \land \text{ok-mg-def-plistp}(\text{proc-list}) \land \text{all-cars-unique}(\text{mg-alist}(\text{mg-state})) \land (\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})) \land \text{user-defined-procp}(\text{subr}, \text{proc-list}) \land \text{normal}(\text{mg-state}) \land \text{no-p-aliasing}(\text{bindings}(\text{top}(\text{ctrl-stk})), \text{mg-alist}(\text{mg-state})) \land (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, \text{list}(\text{length}(\text{temp-stk}), \text{p-ctrl-stk-size}(\text{ctrl-stk})))))) \land \text{mg-expression-falsep}(\text{if-condition}(\text{stmt}, \text{mg-state})) \rightarrow (\text{p-step}(\text{p-state}(\text{tag}(\text{'pc}, \text{cons}(\text{subr}, \text{length}(\text{code}(\text{translate}(\text{make-cinfo}(\text{append}(\text{code}(\text{cinfo}), \text{list}(\text{'push-local}, \text{if-condition}(\text{stmt})), \text{'(fetch-temp-stk)}, \text{list}(\text{'test-bool-and-jump, \text{'false, \text{label-cnt}(\text{cinfo})}))))), \text{label-alist}(\text{cinfo})), 13)
\[ 1 + \left(1 + \text{label-cnt}(\text{cinfo})\right), \]
\[ t\text{-}\text{cond}\text{-}\text{list}, \]
\[ \text{if-true-branch}(\text{stmt}), \]
\[ \text{proc}\text{-}\text{list}) \]
\[ + 1), \]
\[ \text{ctrl}\text{-}\text{stk}, \]
\[ \text{map}\text{-}\text{down}\text{-}\text{values}(\text{mg}\text{-}\text{alist}(\text{mg}\text{-}\text{state}), \]
\[ \text{bindings}(\text{top}(\text{ctrl}\text{-}\text{stk}), \]
\[ \text{temp}\text{-}\text{stk}), \]
\[ \text{translate}\text{-}\text{proc}\text{-}\text{list}(\text{proc}\text{-}\text{list}), \]
\[ \text{list}(\text{list}(\text{c-c}, \]
\[ \text{mg}\text{-}\text{cond}\text{-}\text{to}\text{-}p\text{-}\text{nat}(\text{cc}(\text{mg}\text{-}\text{state}, \text{t}\text{-}\text{cond}\text{-}\text{list}))), \]
\[ \text{MG}\text{-}\text{MAX}\text{-CTRL}\text{-}\text{STK}\text{-}\text{SIZE}, \]
\[ \text{MG}\text{-}\text{MAX}\text{-TEMP}\text{-}\text{STK}\text{-}\text{SIZE}, \]
\[ \text{MG}\text{-}\text{WORD}\text{-}\text{SIZE}, \]
\[ \text{'}\text{run}\text{'})) \]
\[ = \text{map}\text{-}\text{down}(\text{mg}\text{-}\text{state}, \]
\[ \text{proc}\text{-}\text{list}, \]
\[ \text{ctrl}\text{-}\text{stk}, \]
\[ \text{temp}\text{-}\text{stk}, \]
\[ \text{tag}(\text{'}\text{pc}\text{', \]
\[ \text{cons}(\text{subr}, \]
\[ \text{length}(\text{code}(\text{add}\text{-}\text{code}(\text{translate}(\text{make}\text{-}\text{cinfo}(\text{append}(\text{code}(\text{cinfo}), \]
\[ \text{list}(\text{list}(\text{'}\text{push}\text{-}\text{local}, \]
\[ \text{if}\text{-}\text{condition}(\text{stmt}), \]
\[ \text{'}\text{fetch}\text{-}\text{temp}\text{-}\text{stk} \]
\[ \text{list}(\text{'test}\text{-}\text{bool}\text{-}a\text{-}\text{false}, \]
\[ \text{false}, \]
\[ \text{label}\text{-}\text{cnt}(\text{cinfo}), \]
\[ \text{label}\text{-}\text{alist}(\text{cinfo}), \]
\[ 1 + \left(1 + \text{label}\text{-}\text{cnt}(\text{cinfo})\right), \]
\[ t\text{-}\text{cond}\text{-}\text{list}, \]
\[ \text{if-true-branch}(\text{stmt}), \]
\[ \text{proc}\text{-}\text{list}), \]
\[ \text{list}(\text{list}(\text{'jump}, \]
\[ 1 + \text{label}\text{-}\text{cnt}(\text{cinfo}), \]
\[ \text{cons}(\text{'}\text{dl}, \]
\[ \text{cons}(\text{label}\text{-}\text{cnt}(\text{cinfo}), \]
\[ \text{'}\text{nil}, \]
\[ \text{(no-op))))))))\), \]
\[ t\text{-}\text{cond}\text{-}\text{list}) \]
(prove-lemma if-step-normal-false-state2-equals-final (rewrite)
  (implies
    (and (not (zerop n))
      (equal (car stmt) 'if-mg)
      (not (resources-inadequatep stmt proc-list)
        (list (length temp-stk)
          (p-ctrl-stk-size ctrl-stk))))
    (ok-mg-statement stmt r-cond-list name-alist proc-list)
    (mg-vars-list-ok-in-p-state (mg-alist mg-state)
      (bindings (top ctrl-stk))
      temp-stk)
    (ok-mg-statep mg-state r-cond-list)
    (ok-translation-parameters cinfo t-cond-list stmt proc-list code2)
    (signatures-match (mg-alist mg-state) name-alist)
    (ok-mg-def-plistp proc-list)
    (all-cars-unique (mg-alist mg-state))
    (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)
    (normal mg-state)
    (no-p-aliasing (bindings (top ctrl-stk)) (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))))
    (MG-EXPRESSION-FALSEP (IF-CONDITION STMT) MG-STATE)
    (normal (mg-meaning-r (if-false-branch stmt) proc-list mg-state (sub1 n)
      (LIST (LENGTH TEMP-STK)
        (P-CTRL-STK-SIZE CTRL-STK))))
    (equal
      (p-step
        (P-STATE
          (TAG 'PC
            (CONS SUBR
              (IF
                (NORMAL (MG-MEANING-R (IF-FALSE-BRANCH STMT)
                  PROC-LIST MG-STATE
                  (SUB1 N)
                  (LIST (LENGTH TEMP-STK)
                    (P-CTRL-STK-SIZE CTRL-STK))))
                (LENGTH
                  (CODE
                    (TRANSLATE
                      15
```
(ADD-CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO))
(List (List 'PUSH-LOCAL
(IF-CONDITION STMT))
'(FETCH-TEMP-STK)
(List 'TEST-BOOL-AND-JUMP
'FALSE
(LABEL-CNT CINFO))))
(LABEL-ALIST CINFO)
(ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST)
(List (List 'JUMP (ADD1 (LABEL-CNT CINFO)))
(CONS 'DL
 (CONS (LABEL-CNT CINFO)
 '(NIL (NO-OP))))))
T-COND-LIST
(IF-FALSE-BRANCH STMT)
PROC-LIST))
(FIND-LABEL
(FETCH-LABEL
(CC (MG-MEANING-R (IF-FALSE-BRANCH STMT)
 PROC-LIST MG-STATE
(SUB1 N)
(List (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
LABEL-ALIST
(TRANSLATE
(ADD-CODE
 (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO))
 (List (List 'PUSH-LOCAL
 (IF-CONDITION STMT))
 '(FETCH-TEMP-STK)
(List 'TEST-BOOL-AND-JUMP
'FALSE
 (LABEL-CNT CINFO))))
LABEL-ALIST CINFO)
(ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST)
(List (List 'JUMP (ADD1 (LABEL-CNT CINFO)))
(CONS 'DL

16
(CONS (LABEL-CNT CINFO)
  '(NIL (NO-OP))))))
T-COND-LIST
(IF-FALSE-BRANCH STMT)
PROC-LIST))
(APPEND
(CODE
 (TRANSLATE
  (ADD-CODE
   (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
   (LIST (LIST 'PUSH-LOCAL
   (IF-CONDITION STMT))
   (FETCH-TEMP-STK)
   (LIST 'TEST-BOOL-AND-JUMP
   'FALSE
   (LABEL-CNT CINFO))))
   (LABEL-ALIST CINFO)
   (ADD1 (ADD1 (LABEL-CNT CINFO))))
   T-COND-LIST
   (IF-TRUE-BRANCH STMT)
   PROC-LIST)
   (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO))))
   (CONS 'DL
   (CONS (LABEL-CNT CINFO)
     '(NIL (NO-OP)))))
   T-COND-LIST
   (IF-FALSE-BRANCH STMT)
   PROC-LIST))
   (CONS (CONS 'DL
   (CONS (ADD1 (LABEL-CNT CINFO))
     '(NIL (NO-OP)))))
   CODE2))))))
CTRL-STK
(MAP-DOWN-VALUES
 (MG-ALIST (MG-MEANING-R (IF-FALSE-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 (IF-FALSE-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-TEMSTK-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))))
(LIST (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

18
(prove-lemma if-non-normal-false-state2-equals-final (rewrite)
  (implies
   (and (not (zerop n))
    (equal (car stmt) 'if-mg)
    (not (resources-inadequatep stmt proc-list)
      (list (length temp-stk)
        (p-ctrl-stk-size ctrl-stk))))
  (ok-mg-statement stmt r-cond-list name-alist proc-list)
  (ok-mg-vars-list-ok-in-p-state (mg-alist mg-state))
  (bindings (top ctrl-stk))
  (temp-stk)
  (ok-mg-statep mg-state r-cond-list)
  (ok-translation-parameters cinfo t-cond-list stmt proc-list code2)
  (signatures-match (mg-alist mg-state) name-alist)
  (ok-mg-def-plistp proc-list)
  (all-cars-unique (mg-alist mg-state))
  (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)
  (normal mg-state)
  (no-p-aliasing (bindings (top ctrl-stk)) (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))))
  (MG-EXPRESSION-FALSEP (IF-CONDITION STMT) MG-STATE)
  (not (normal (mg-meaning-r (if-false-branch stmt) proc-list mg-state (sub1 n))
    (list (length temp-stk)
      (p-ctrl-stk-size ctrl-stk))))
  (equal
   (P-STATE
    (TAG 'PC
     (CONS SUBR
      (IF
       (NORMAL (MG-MEANING-R (IF-FALSE-BRANCH STMT)
        PROC-LIST MG-STATE
        19
(SUB1 N)
  (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK))))
  (LENGTH
   (CODE
    (TRANSLATE
     (ADD-CODE
      (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
        (LIST (LIST 'PUSH-LOCAL
          (IF-CONDITION STMT))
        (LIST (LENGTH TEMP-STK)
          (P-CTRL-STK-SIZE CTRL-STK))))
        (LIST 'TEST-BOOL-AND-JUMP
          'FALSE
          (LABEL-CNT CINFO))))
      LABEL-ALIST CINFO)
    (ADD1 (LABEL-CNT CINFO))))
    T-COND-LIST
    (IF-TRUE-BRANCH STMT)
    PROC-LIST)
    (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
      (CONS 'DL
        (CONS (LABEL-CNT CINFO)
          '(NIL (NO-OP)))))))
    T-COND-LIST
    (IF-FALSE-BRANCH STMT)
    PROC-LIST))))
    (FIND-LABEL
     (CC (MG-MEANING-R (IF-FALSE-BRANCH STMT)
       PROC-LIST MG-STATE
       (SUB1 N)
       (LIST (LENGTH TEMP-STK)
         (P-CTRL-STK-SIZE CTRL-STK))))
     (LABEL-ALIST
      (TRANSLATE
       (ADD-CODE
        (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
          (LIST (LIST 'PUSH-LOCAL
            (IF-CONDITION STMT))
        (LIST (LENGTH TEMP-STK)
          (P-CTRL-STK-SIZE CTRL-STK))))
        (LIST (LIST 'TEST-BOOL-AND-JUMP
          'FALSE
          (LABEL-CNT CINFO))))
        (LABEL-ALIST CINFO)

20
(ADD1 (ADD1 (LABEL-CNT CINFO)))
  T-COND-LIST
  (IF-TRUE-BRANCH STMT)
  PROC-LIST)
  (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
  (CONS 'DL
  (CONS (LABEL-CNT CINFO)
    '(NIL (NO-OP))))))
  T-COND-LIST
  (IF-FALSE-BRANCH STMT)
  PROC-LIST)))
  (APPEND
  (CODE
  (TRANSLATE
  (ADD-CODE
    (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (LIST 'PUSH-LOCAL
        (LABEL-CNT CINFO)
      (APPEND
        (CODE (TRANSLATE
          (ADD1 (ADD1 (LABEL-CNT CINFO)))))
        T-COND-LIST
        (IF-CONDITION STMT)
        PROC-LIST)
        (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
        (CONS 'DL
        (CONS (LABEL-CNT CINFO)
          '(NIL (NO-OP))))))
        T-COND-LIST
        (IF-FALSE-BRANCH STMT)
        PROC-LIST)))
        (CONS (CONS 'DL
          (CONS (ADD1 (LABEL-CNT CINFO))
            '(NIL (NO-OP))))))
          T-COND-LIST
          (IF-FALSE-BRANCH STMT)
          PROC-LIST))
          (CONS (CONS 'DL
            (CONS (ADD1 (LABEL-CNT CINFO))
              '(NIL (NO-OP))))))
              CODE2))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (IF-FALSE-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 (IF-FALSE-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)
  (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))))

22
T-COND-LIST)))
  (MG-MAX-CTRL-STK-SIZE)
  (MG-MAX-TEMP-STK-SIZE)
  (MG-WORD-SIZE)
  ’RUN)))
((INSTRUCTIONS
  PROMOTE S
  (= (MG-MEANING-R STMT PROC-LIST MG-STATE N (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK))))
  (MG-MEANING-R (IF-FALSE-BRANCH STMT) PROC-LIST MG-STATE (SUB1 N)
    (LIST (LENGTH TEMP-STK) (P-CTRL-STK-SIZE CTRL-STK)))) 0)
SPLIT S S (S LEMMAS) (DIVE 2 2 2 1) (= F) TOP S PROVE (DIVE 1)
(REWITE IF-MEANING-R-2) TOP S)))

Event: Disable signatures-match-preserves-plistp.


Event: Disable simple-typed-literalp-ok-valuep.

Theorem: boolean-identifierp-not-false-expressionp
  (boolean-identifierp (b, mg-vars)
    (get-m-value (b, mg-vars) ≠ '(boolean-mg false-mg))
    mg-alistp (mg-vars))
  → (get-m-value (b, mg-vars) = '(boolean-mg true-mg))

Theorem: if-initial-step3-true-test-equals-true-state1
  ((n ≠ 0)
    (car (stmt) = ‘if-mg)
    (∼ resources-inadequatep (stmt,
      proc-list,
      list (length (temp-stk),
        p-ctrl-stk-size (ctrl-stk))))
    ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
    mg-vars-list-ok-in-p-state (mg-alist (mg-state),
      bindings (top (ctrl-stk)),
      temp-stk)
    ok-mg-statep (mg-state, r-cond-list)
    ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2)
    signatures-match (mg-alist (mg-state), name-alist)
    ok-mg-def-plistp (proc-list)
    all-cars-unique (mg-alist (mg-state))
    (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)
∧ normal (mg-state)
∧ no-p-aliasing (bindings (top (ctrl-stk)), 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))))))
∧ (¬ mg-expression-falsep (if-condition (stmt, mg-state)))
→ (p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 2)),
    ctrl-stk,
    push (rget (untag (value (if-condition (stmt),
        bindings (top (ctrl-stk))))),
    map-down-values (mg-alist (mg-state),
        bindings (top (ctrl-stk),
        temp-stk)),
    map-down-values (mg-alist (mg-state),
        bindings (top (ctrl-stk),
        temp-stk)),
    translate-proc-list (proc-list),
    list (list ('c-c,
        mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
    MG-MAX-CTRL-STK-SIZE,
    MG-MAX-TEMP-STK-SIZE,
    MG-WORD-SIZE,
    'run))
= map-down (mg-state,
    proc-list,
    ctrl-stk,
    temp-stk,
    tag ('pc,
        cons (subr,
        length (code (make-cinfo (append (code (cinfo),
            list (list ('push-local,
                if-condition (stmt)),
                '(fetch-temp-stk),
                list ('test-bool-and-jump,
                false,
                label-cnt (cinfo))))),
            label-alist (cinfo),
            1 + (1 + label-cnt (cinfo))))))),
24
(prove-lemma if-nonnormal-true-state2-equals-final (rewrite)
    (implies
        (and (not (zerop n))
            (equal (car stmt) 'if-mg)
            (not (resources-inadequatep stmt proc-list)
                (list (length temp-stk)
                    (p-ctrl-stk-size ctrl-stk)))))
        (ok-mg-statement stmt r-cond-list name-alist proc-list)
        (mg-vars-list-ok-in-p-state (mg-alist mg-state)
            (bindings (top ctrl-stk)))
        (temp-stk)
        (ok-mg-statep mg-state r-cond-list)
        (ok-translation-parameters cinfo t-cond-list stmt proc-list code2)
        (signatures-match (mg-alist mg-state) name-alist)
        (ok-mg-def-plistp proc-list)
        (all-cars-unique (mg-alist mg-state))
        (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)
        (normal mg-state)
        (no-p-aliasing (bindings (top ctrl-stk)) (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)))
            (not (MG-EXPRESSION-FALSEP (IF-CONDITION STMT) MG-STATE))
            (not (normal (mg-meaning-r (if-true-branch stmt) proc-list mg-state (sub1 n)
                (list (LENGTH TEMP-STK)
                    (P-CTRL-STK-SIZE CTRL-STK))))))
            (equal
                (P-STATE
                    (TAG 'PC
                        (CDNS SUBR
                            (IF
                                (NORMAL (MG-MEANING-R (IF-TRUE-BRANCH STMT)
                                    PROC-LIST MG-STATE
                                    (SUB1 N)
                                    (LIST (LENGTH TEMP-STK)
                                        (P-CTRL-STK-SIZE CTRL-STK)))))))))
(LENGTH
  (CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
  (LIST (LIST 'PUSH-LOCAL
    (IF-CONDITION STMT))
    '(FETCH-TEMP-STK)
    (LIST 'TEST-BOOL-AND-JUMP
    'FALSE
    (LABEL-CNT CINFO))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST))
  
(FIND-LABEL
  (FETCH-LABEL
(CC (MG-MEANING-R (IF-TRUE-BRANCH STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LABEL-ALIST
  (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
  (LIST (LIST 'PUSH-LOCAL
    (IF-CONDITION STMT))
    '(FETCH-TEMP-STK)
    (LIST 'TEST-BOOL-AND-JUMP
    'FALSE
    (LABEL-CNT CINFO))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST))
  
(APPEND
(CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
  (LIST (LIST 'PUSH-LOCAL
    (IF-CONDITION STMT))
    '(FETCH-TEMP-STK)
    (LIST 'TEST-BOOL-AND-JUMP
    'FALSE
    (LABEL-CNT CINFO))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
  26
(IF-TRUE-BRANCH STMT)
PROC-LIST))
(CONS
  (LIST 'JUMP (ADD1 (LABEL-CNT CINFO))))
(CONS
  (CONS 'DL
    (CONS (LABEL-CNT CINFO)
      '(NIL (NO-OP))))
  (APPEND
   (CODE
    (TRANSLATE
     (NULLIFY (TRANSLATE (MAKE-CINFO NIL (LABEL-ALIST CINFO)
       (ADD1 (ADD1 (LABEL-CNT CINFO))))
     T-COND-LIST)
    (IF-TRUE-BRANCH STMT)
    PROC-LIST)))
  (CONS (CONS 'DL
    (CONS (ADD1 (LABEL-CNT CINFO))
      '(NIL (NO-OP))))
    CODE2))))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (IF-TRUE-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 (IF-TRUE-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)
\( (\text{MG-WORD-SIZE}) \)
\( '\text{RUN} \)
\( (\text{P-STATE}) \)
\( (\text{TAG 'PC}) \)
\( (\text{CONS SUBR}) \)
\( (\text{IF}) \)
\( (\text{NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N}) \)
\( (\text{LIST (LENGTH TEMP-STK)}) \)
\( (\text{P-CTRL-STK-SIZE CTRL-STK)}) \))
\( (\text{LENGTH (CODE (TRANSITIZE CINFO T-COND-LIST STMT PROC-LIST)})\) \)
\( (\text{FIND-LABEL}) \)
\( (\text{FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N}) \)
\( (\text{LIST (LENGTH TEMP-STK)}) \)
\( (\text{P-CTRL-STK-SIZE CTRL-STK)}) \))
\( (\text{LABEL-ALIST (TRANSITIZE CINFO T-COND-LIST STMT PROC-LIST)})\) \)
\( (\text{APPEND (CODE (TRANSITIZE CINFO T-COND-LIST STMT PROC-LIST)}) \)
\( \text{CODE2})) \))
\( \text{CTRL-STK} \)
\( (\text{MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R STMT PROC-LIST MG-STATE N}) \)
\( (\text{LIST (LENGTH TEMP-STK)})\)
\( (\text{P-CTRL-STK-SIZE CTRL-STK)}) \))
\( (\text{BINDINGS (TOP CTRL-STK)}) \)
\( \text{TEMP-STK} \)
\( (\text{TRANSITIZE-PROC-LIST PROC-LIST}) \)
\( \text{LIST} \)
\( \text{LIST 'C-C} \)
\( (\text{MG-COND-TO-P-NAT (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N}) \)
\( (\text{LIST (LENGTH TEMP-STK)}) \)
\( (\text{P-CTRL-STK-SIZE CTRL-STK)}) \))
\( \text{T-COND-LIST)}) \))
\( (\text{MG-MAX-CTRL-STK-SIZE}) \)
\( (\text{MG-MAX-TEMP-STK-SIZE}) \)
\( (\text{MG-WORD-SIZE}) \)
\( '\text{RUN}))) \))
\( ((\text{INSTRUCTIONS PROMOTE S}) \)
\( (\text{=} (\text{MG-MEANING-R STMT PROC-LIST MG-STATE N}) \)
\( (\text{LIST (LENGTH TEMP-STK)}) \)
\( (\text{P-CTRL-STK-SIZE CTRL-STK)}) \))
\( (\text{MG-MEANING-R (IF-TRUE-BRANCH STMT) PROC-LIST MG-STATE}) \)
\( \text{(SUB1 N)}) \)
\( (\text{LIST (LENGTH TEMP-STK)}) \)
\( (\text{P-CTRL-STK-SIZE CTRL-STK)}) \))
\( 28 \)
THEOREM: if-find-labelp-lemma3

((car (stmt) = 'if-mg)
 ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧ ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2))
→ (¬ find-labelp (1 + label-cnt (cinfo),
       code (translate (make-cinfo (append (code (cinfo),
             list (list (push-local,
               if-condition (stmt)),
             (fetch-temp-stk),
             list (test-bool-and-jump,
               false,
               label-cnt (cinfo)))),
             label-alist (cinfo),
             1 + (1 + label-cnt (cinfo)))),
       t-cond-list,
       if-true-branch (stmt),
       proc-list)))
'(FETCH-TEMP-STK)
  (LIST 'TEST-BOOL-AND-JUMP
'FALSE
(LABEL-CNT CINFO))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO)))))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST)))
  (FIND-LABEL
(FETCH-LABEL
(CC (MG-MEANING-R (IF-TRUE-BRANCH STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))))
(LABEL-ALIST
  (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (LIST 'PUSH-LOCAL
  (IF-CONDITION STMT))
  '(FETCH-TEMP-STK)
  (LIST 'TEST-BOOL-AND-JUMP
  'FALSE
  (LABEL-CNT CINFO))))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO)))))))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST)))
(APPEND
(CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
  (LIST (LIST 'PUSH-LOCAL
  (IF-CONDITION STMT))
  '(FETCH-TEMP-STK)
  (LIST 'TEST-BOOL-AND-JUMP
  'FALSE
  (LABEL-CNT CINFO))))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO))))))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST)))
(CONS
(LIST 'JUMP (ADD1 (LABEL-CNT CINFO))))
(CONS 'DL
(CONS (LABEL-CNT CINFO)
  '(NIL (NO-OP)))))
(APPEND
(CODE
  (TRANSLATE
   (NULLIFY (TRANSLATE (MAKE-CINFO NIL
    (LABEL-ALIST CINFO)
    (ADD1 (ADD1 (LABEL-CNT CINFO))))))
  T-COND-LIST
  (IF-TRUE-BRANCH STMT)
  PROC-LIST))
  T-COND-LIST
  (IF-FALSE-BRANCH STMT)
  PROC-LIST))
(CONS (CONS 'DL
  (CONS (ADD1 (LABEL-CNT CINFO))
    '(NIL (NO-OP)))))
CODE2))))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (IF-TRUE-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 (IF-TRUE-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))
    (P-STATE
      (TAG 'PC

32
(CONS SUBR
 (PLUS
 (LENGTH
 (CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
 (LIST (LIST 'PUSH-LOCAL
 (IF-CONDITION STMT))
 'FETCH-TEMP-STK)
 (LIST 'TEST-BOOL-AND-JUMP
 'FALSE
 (LABEL-CNT CINFO))))
 (LABEL-ALIST CINFO)
 (ADD1 (ADD1 (LABEL-CNT CINFO))))
 T-COND-LIST
 (IF-TRUE-BRANCH STMT)
 PROC-LIST)))
 (ADD1
 (ADD1
 (LENGTH
 (CODE
 (TRANSLATE
 (MAKE-CINFO NIL
 (LABEL-ALIST CINFO)
 (LABEL-CNT (TRANSLATE (MAKE-CINFO NIL
 (LABEL-ALIST CINFO)
 (ADD1 (ADD1 (LABEL-CNT CINFO))))
 T-COND-LIST
 (IF-TRUE-BRANCH STMT)
 PROC-LIST)))))
 T-COND-LIST
 (IF-FALSE-BRANCH STMT)
 PROC-LIST))))))))
 CTRL-STK
 (MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R (IF-TRUE-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 (IF-TRUE-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 (IF-TRUE-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)))

((INSTRUCTIONS PROMOTE
  (DIVE 1)
  X
  (S LEMMAS)
  (DIVE 1 1 2)
  (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (REWRITE IF-CODE-REWRITE2)
  UP
  (REWRITE GET-LENGTH-CAR)
  S UP X UP X
  (DIVE 1)
  X UP S X
  (S LEMMAS)
  (DIVE 1 2 1)
  (REWRITE DEFINEDP-CAR-ASSOC)
  NX
  (DIVE 2)
  (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (REWRITE IF-CODE-REWRITE2)
  UP
  (REWRITE FIND-LABEL-APPEND)
  (DIVE 2)
  X
  (DIVE 1)
  X
  (DIVE 1)
  (REWRITE FIND-LABEL-APPEND)
  (DIVE 2)
  X UP
  (REWRITE PLUS-0-REWRITE2)
  S TOP
  (DIVE 1)
  TOP
  (S LEMMAS)
  (S-PROP NULLIFY)
  (S LEMMAS)
Theorem: if-normal-true-state2-step2-equals-final

\[ ((n \neq 0) \land (\text{car}(\text{stmt}) = 'if-mg) \land (\neg \text{resources-inadequate}(\text{stmt}, \text{proc-list}, \text{list(length(temp-stk)}, \text{p-ctrl-stk-size(ctrl-stk)}))) \land \text{ok-mg-statement}((\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \land \text{mg-vars-list-ok-in-p-state}(\text{mg-alist}(\text{mg-state}), \text{bindings}(\text{top(ctrl-stk)}), \text{temp-stk})) \land \text{ok-mg-statep}(\text{mg-state}, \text{r-cond-list}) \land \text{ok-translation-parameters}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \land \text{signatures-match}(\text{mg-alist}(\text{mg-state}), \text{name-alist}) \land \text{ok-mg-def-plistp}(\text{proc-list}) \land \text{all-cars-unique}(\text{mg-alist}(\text{mg-state})) \land (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) = \text{append}(\text{code}(\text{translate}(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \text{code2})) \land \text{user-defined-procp}(\text{subr}, \text{proc-list}) \land \text{normal}(\text{mg-state}) \land \text{no-p-aliasing}(\text{bindings}(\text{top(ctrl-stk)}), \text{mg-alist}(\text{mg-state})) \land (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{list(length(temp-stk)}, \text{p-ctrl-stk-size(ctrl-stk)}))) \land (\neg \text{mg-expression-falsep}(\text{if-condition}(\text{stmt}, \text{mg-state}))) \]
\[ \land \text{normal(mg-meaning-r(if-true-branch(stmt), proc-list, mg-state, } \]
\[ n - 1, \]
\[ \text{list(length(temp-stk), p-ctrl-stk-size(ctrl-stk))))} \]
\[ \rightarrow (p-step(p-state(tag('pc, cons(subr,}
\[ \text{length(code(translate(make-cinfo(append(code(cinfo),}
\[ \text{list(list('push-local,} \]
\[ \text{if-condition(stmt)),} \]
\[ \text{'(fetch-temp-stk),} \]
\[ \text{list('test-bool-and-jump,} \]
\[ \text{'false,} \]
\[ \text{label-cnt(cinfo))},} \]
\[ 1 + (1 + \text{label-cnt(cinfo))},} \]
\[ t-cond-list,} \]
\[ \text{if-true-branch(stmt),} \]
\[ proc-list)))))} \]
\[ + (1 + (1 + \text{length(code(translate(make-cinfo(nil,}} \]
\[ \text{label-alist(cinfo),} \]
\[ \text{label-cnt(translate(make-cinfo(cinfo),}} \]
\[ t-cond-list,} \]
\[ \text{if-true-branch(stmt),} \]
\[ proc-list)))))},} \]
\[ ctrl-stk,} \]
\[ \text{map-down-values(mg-alist(mg-meaning-r(if-true-branch(stmt),} \]
\[ proc-list,} \]
\[ mg-state,} \]
\[ n - 1,} \]
\[ \text{list(length(temp-stk),} \]
\[ p-ctrl-stk-size(ctrl-stk))))},} \]
\[ \text{bindings(top(ctrl-stk)),} \]
\[ temp-stk),} \]
\[ \text{translate-proc-list(proc-list),} \]
\[ \text{list(list('c-c,}} \]
\[ \text{mg-cond-to-p-nat(cc(mg-meaning-r(if-true-branch(stmt),} \]
\[ proc-list,} \]
\[ mg-state,} \]
\[ t-cond-list,} \]
\[ \text{if-true-branch(stmt),} \]
\[ proc-list)))))})))},} \]
\[ 36} \]
\[ n - 1, \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))))}, \]
\[ t\text{-cond-list}))}, \]
\[ \text{MG-MAX-CTRL-STK-SIZE,} \]
\[ \text{MG-MAX-TEMP-STK-SIZE,} \]
\[ \text{MG-WORD-SIZE,} \]
\[ 'run)) \]
\[ = \text{p-state (tag ('pc,} \]
\[ \text{cons (subr,} \]
\[ \text{if normal (mg-meaning-r (stmt,} \]
\[ \text{proc-list,} \]
\[ \text{mg-state,} \]
\[ n,} \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))))} \]
\[ \text{then length (code (translate (cinfo,} \]
\[ \text{t-cond-list,} \]
\[ \text{stmt,} \]
\[ \text{proc-list)))} \]
\[ \text{else find-label (fetch-label (cc (mg-meaning-r (stmt,} \]
\[ \text{proc-list,} \]
\[ \text{mg-state,} \]
\[ n,} \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))))}, \]
\[ \text{label-alist (translate (cinfo,} \]
\[ \text{t-cond-list,} \]
\[ \text{stmt,} \]
\[ \text{proc-list)))}, \]
\[ \text{append (code (translate (cinfo,} \]
\[ \text{t-cond-list,} \]
\[ \text{stmt,} \]
\[ \text{proc-list))},} \]
\[ \text{code2}) \text{endif),} \]
\[ \text{ctrl-stk,} \]
\[ \text{map-down-values (mg-alist (mg-meaning-r (stmt,} \]
\[ \text{proc-list,} \]
\[ \text{mg-state,} \]
\[ n,} \]
\[ \text{list (length (temp-stk),} \]
\[ \text{p-ctrl-stk-size (ctrl-stk))))}, \]
\[ \text{bindings (top (ctrl-stk)),} \]
\[ \text{temp-stk),} \]
\[ 37 \]
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: if-clock-false-normal
  ((car (stmt) = 'if-mg)
   ∧ (n ≠ 0)
   ∧ normal (mg-state)
   ∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)))
   ∧ mg-expression-falsep (if-condition (stmt), mg-state)
   ∧ normal (mg-meaning-r (if-false-branch (stmt),
                             proc-list,
                             mg-state,
                             n - 1,
                             sizes)))
   → (clock (stmt, proc-list, mg-state, n) = (5 + clock (if-false-branch (stmt), proc-list, mg-state, n - 1)))

THEOREM: if-clock-false-nonnormal
  ((car (stmt) = 'if-mg)
   ∧ (n ≠ 0)
   ∧ normal (mg-state)
   ∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)))
   ∧ mg-expression-falsep (if-condition (stmt), mg-state)
   ∧ (¬ normal (mg-meaning-r (if-false-branch (stmt),
                              proc-list,
                              mg-state,
                              n - 1,
                              sizes))))
   → (clock (stmt, proc-list, mg-state, n) = (4 + clock (if-false-branch (stmt), proc-list, mg-state, n - 1)))

THEOREM: if-clock-true-normal
  ((car (stmt) = 'if-mg)
   ∧ (n ≠ 0)
\[ \text{normal (mg-state)} \]
\[ \land (\neg \text{resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes))}) \]
\[ \land (\neg \text{mg-expression-falsep (if-condition (stmt, mg-state)}) \]
\[ \land \text{normal (mg-meaning-r (if-true-branch (stmt), proc-list, mg-state, n - 1, sizes))} \]
\[ \rightarrow (\text{clock (stmt, proc-list, mg-state, n}) \]
\[ = (5 + \text{clock (if-true-branch (stmt, proc-list, mg-state, n - 1))}) \]

**Theorem:** if-clock-true-nonnormal

\[ ((\text{car (stmt) = 'if-mg}) \]
\[ \land (n \not\equiv 0) \]
\[ \land \text{normal (mg-state)} \]
\[ \land (\neg \text{resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes))}) \]
\[ \land (\neg \text{mg-expression-falsep (if-condition (stmt, mg-state)}) \]
\[ \land (\neg \text{normal (mg-meaning-r (if-true-branch (stmt), proc-list, mg-state, n - 1, sizes))}) \]
\[ \rightarrow (\text{clock (stmt, proc-list, mg-state, n}) \]
\[ = (3 + \text{clock (if-true-branch (stmt, proc-list, mg-state, n - 1))}) \]

```
;; These are the schemata for the four IF cases.
```

**Theorem:** if-false-normal-exact-time-schema

\[ ((\text{stmt-time} = (5 + \text{false-time})) \]
\[ \land (\text{p (initial, 4) = false-state1}) \]
\[ \land (\text{p (false-state1, false-time) = false-state2}) \]
\[ \land (\text{p-step (false-state2) = final}) \]
\[ \rightarrow (\text{p (initial, stmt-time) = final}) \]

**Theorem:** if-false-nonnormal-exact-time-schema

\[ ((\text{stmt-time} = (4 + \text{false-time})) \]
\[ \land (\text{p (initial, 4) = false-state1}) \]
\[ \land (\text{p (false-state1, false-time) = false-state2}) \]
\[ \land (\text{false-state2 = final}) \]
\[ \rightarrow (\text{p (initial, stmt-time) = final}) \]

**Theorem:** if-true-normal-exact-time-schema

\[ ((\text{stmt-time} = (5 + \text{true-time})) \]
\[ \land (\text{p (initial, 3) = true-state1}) \]
\[ \land (\rightarrow (\text{p (initial, stmt-time) = final}) \]
∧ (p(true-state1, true-time) = true-state2)
∧ (p(true-state2, 2) = final))
→ (p(initial, stmt-time) = final)

THEOREM: if-true-nonnormal-exact-time-schema
((stmt-time = (3 + true-time))
∧ (p(initial, 3) = true-state1)
∧ (p(true-state1, true-time) = true-state2)
∧ (true-state2 = final))
→ (p(initial, stmt-time) = final)

(prove-lemma if-exact-time-lemma (rewrite)
 (IMPLIES
   (AND (NOT (ZEROP N))
    (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
      (LIST (LENGTH TEMP-STK)
        (P-CTRL-STK-SIZE CTRL-STK))))
    (EQUAL (CAR STMT) 'IF-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-PROCOP 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))))))

40
(IMPLIES ;; the true branch
  (AND
    (OK-MG-STATEMENT (IF-TRUE-BRANCH STMT)
      R-COND-LIST NAME-ALIST PROC-LIST)
    (OK-MG-DEF-PLISTP PROC-LIST)
    (OK-TRANSLATION-PARAMETERS
      (MAKE-CINFO (APPEND (CODE CINFO)
                (LIST (LIST 'PUSH-LOCAL
                  (IF-CONDITION STMT))
                  ' (FETCH-TEMP-STK)
                  (LIST 'TEST-BOOL-AND-JUMP
                    ' FALSE
                    (LABEL-CNT CINFO))))
                (LABEL-ALIST CINFO)
                (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (IF-TRUE-BRANCH STMT)
      PROC-LIST)
    (CONS
      (LIST ' JUMP (ADD1 (LABEL-CNT CINFO)))
    (CONS
      (CONS ' DL
        (CONS (LABEL-CNT CINFO)
          '(NIL (NO-OP))))
      (APPEND
        (CODE
          (TRANSLETE
            (NULLIFY (TRANSLETE (MAKE-CINFO NIL
              (LABEL-ALIST CINFO)
            (ADD1 (ADD1 (LABEL-CNT CINFO))))
          T-COND-LIST
          (IF-TRUE-BRANCH STMT)
          PROC-LIST))
        T-COND-LIST
        (IF-FALSE-BRANCH STMT)
        PROC-LIST))
      (CONS (CONS ' DL
        (CONS (ADD1 (LABEL-CNT CINFO))
          '(NIL (NO-OP))))
      CODE2))))
    (OK-MG-STATEP MG-STATE R-COND-LIST)
    (COND-SUBSETP R-COND-LIST T-COND-LIST)
    (EQUAL
      (CODE (TRANSLETE-DEF-BODY (ASSOC SUBR PROC-LIST))

41
(APPEND
 (CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
 (LIST (LIST 'PUSH-LOCAL
 (IF-CONDITION STMT))
 'FETCH-TEMP-STK)
 (LIST 'TEST-BOOL-AND-JUMP
 'FALSE
 (LABEL-CNT CINFO))))
 (LABEL-ALIST CINFO)
 (ADD1 (ADD1 (LABEL-CNT CINFO))))
 T-COND-LIST
 (IF-TRUE-BRANCH STMT)
 PROC-LIST))
 (CONS
 (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
 (CONS
 (CONS 'DL
 (CONS (LABEL-CNT CINFO)
 '(NIL (NO-OP))))
 (APPEND
 (CODE
 (TRANSLATE
 (NULLIFY (TRANSLATE (MAKE-CINFO NIL
 (LABEL-ALIST CINFO)
 (ADD1 (ADD1 (LABEL-CNT CINFO))))
 T-COND-LIST
 (IF-TRUE-BRANCH STMT)
 PROC-LIST))
 T-COND-LIST
 (IF-FALSE-BRANCH STMT)
 PROC-LIST))
 (CONS (CONS 'DL
 (CONS (ADD1 (LABEL-CNT CINFO))
 '(NIL (NO-OP)))))
 CODE2))))))
 (USER-DEFINED-PROC 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 (IF-TRUE-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 ;; true-state1
  (TAG 'PC
  (CONS SUBR
  (LENGTH (CODE (MAKE-CINFO (APPEND (CODE CINFO)
  (LIST (LIST 'PUSH-LOCAL
  (IF-CONDITION STMT))
  '(FETCH-TEMP-STK)
  (LIST 'TEST-BOOL-AND-JUMP
  'FALSE
  (LABEL-CNT CINFO))))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO)))))))
  T-COND-LIST
  (CLOCK (IF-TRUE-BRANCH STMT) ;; true-time
  PROC-LIST MG-STATE
  (SUB1 N)))))
  (P-STATE;; true-state2
  (TAG 'PC
  (CONS SUBR
  (IF
  (NORMAL (MG-MEANING-R (IF-TRUE-BRANCH STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))
  (LENGTH
  (CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
  (LIST (LIST 'PUSH-LOCAL
  (IF-CONDITION STMT))
  '(FETCH-TEMP-STK)
  (LIST 'TEST-BOOL-AND-JUMP
  'FALSE
  (LABEL-CNT CINFO)))))))}
(LABEL-ALIST CINFO)
(ADD1 (ADD1 (LABEL-CNT CINFO)))
  T-COND-LIST
  (IF-TRUE-BRANCH STMT)
  PROC-LIST))
(FIND-LABEL
  (FETCH-LABEL
    (CC (MG-MEANING-R (IF-TRUE-BRANCH STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))
    (LABEL-ALIST
     (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
        (LIST (LIST 'PUSH-LOCAL
          (IF-CONDITION STMT))
          '(FETCH-TEMP-STK)
          (LIST 'TEST-BOOL-AND-JUMP
            'FALSE
            (LABEL-CNT CINFO)))))
        (LABEL-ALIST CINFO)
        (ADD1 (ADD1 (LABEL-CNT CINFO)))
      T-COND-LIST
      (IF-TRUE-BRANCH STMT)
      PROC-LIST))
      (APPEND
        (CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
          (LIST (LIST 'PUSH-LOCAL
            (IF-CONDITION STMT))
            '(FETCH-TEMP-STK)
            (LIST 'TEST-BOOL-AND-JUMP
              'FALSE
              (LABEL-CNT CINFO)))))
          (LABEL-ALIST CINFO)
          (ADD1 (ADD1 (LABEL-CNT CINFO)))
        T-COND-LIST
        (IF-TRUE-BRANCH STMT)
        PROC-LIST))
        (CONS
          (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
        CONS
        (CONS 'DL
          (CONS (LABEL-CNT CINFO)
            '(NIL (NO-OP))))))
(APPEND
  (CODE
    (TRANSLATE
      (NULLIFY (TRANSLATE (MAKE-CINFO NIL
        (LABEL-ALIST CINFO)
        (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (IF-TRUE-BRANCH STMT)
      PROC-LIST))
    T-COND-LIST
    (IF-FALSE-BRANCH STMT)
    PROC-LIST))
    (CONS (CONS 'DL
      (CONS (ADD1 (LABEL-CNT CINFO))
        '((NIL (NO-OP)))))
      CODE2))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (IF-TRUE-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 (IF-TRUE-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 (IF-FALSE-BRANCH STMT)
        R-COND-LIST NAME-ALIST PROC-LIST)
      (OK-MG-DEF-PLISTP PROC-LIST)
      (OK-TRANSLATION-PARAMETERS
        'RUN))))
(ADD-CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (LIST 'PUSH-LOCAL
  (IF-CONDITION STMT))
'(FETCH-TEMP-STK)
(LIST 'TEST-BOOL-AND-JUMP
  'FALSE
  (LABEL-CNT CINFO)))))
(LABEL-ALIST CINFO)
(ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST)
(LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO))))
(CONS 'DL
  (CONS (LABEL-CNT CINFO)
    '(NIL (NO-OP))))))
T-COND-LIST
(IF-FALSE-BRANCH STMT)
PROC-LIST
(CONS (CONS 'DL
  (CONS (ADD1 (LABEL-CNT CINFO))
    '(NIL (NO-OP)))
  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
        (ADD-CODE
          (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
            (LIST (LIST 'PUSH-LOCAL
              (IF-CONDITION STMT))
            '(FETCH-TEMP-STK)
            (LIST 'TEST-BOOL-AND-JUMP
              'FALSE
              (LABEL-CNT CINFO)))))
            (LABEL-ALIST CINFO)
            (ADD1 (ADD1 (LABEL-CNT CINFO))))
          T-COND-LIST
          (IF-TRUE-BRANCH STMT)
          PROC-LIST)
        PROC-LIST)))
  46)
(LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))))
(CONS 'DL
(CONS (LABEL-CNT CINFO)
   '(NIL (NO-OP)))))))
T-COND-LIST
(IF-FALSE-BRANCH STMT)
PROC-LIST))
(CONS (CONS 'DL
   (CONS (ADD1 (LABEL-CNT CINFO))
     '(NIL (NO-OP)))))
CODE2))
(USER-DEFINED-PROC 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 (IF-FALSE-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 ;; false-state1
   (TAG 'PC
   (CONS SUBR
     (LENGTH
     (CODE
     (ADD-CODE
     (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO))
     (LIST (LIST 'PUSH-LOCAL
       (IF-CONDITION STMT))
       '(FETCH-TEMP-STK)
       (LIST 'TEST-BOOL-AND-JUMP
       'FALSE
       (LABEL-CNT CINFO)))))
     (LABEL-ALIST CINFO))
   47
(ADD1 (ADD1 (LABEL-CNT CINFO))))
  T-COND-LIST
   (IF-TRUE-BRANCH STMT)
   PROC-LIST)
  (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
    (CONS 'DL
      (CONS (LABEL-CNT CINFO)
        '(NIL (NO-OP))))))")
  T-COND-LIST)
  (CLOCK (IF-FALSE-BRANCH STMT) ;; false-time
    PROC-LIST MG-STATE
    (SUB1 N)))
  (P-STATE;; false-state2
  (TAG 'PC
    (CONS SUBR
      (IF
        (NORMAL (MG-MEANING-R (IF-FALSE-BRANCH STMT)
          PROC-LIST MG-STATE
          (SUB1 N)
          (LIST (LENGTH TEMP-STK)
            (P-CTRL-STK-SIZE CTRL-STK))))
        (LENGTH
          (CODE
            (TRANSLATE
              (ADD-CODE
                (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                  (LIST (LIST 'PUSH-LOCAL
                    (IF-CONDITION STMT))
                    '(FETCH-TEMP-STK)
                    (LIST 'TEST-BOOL-AND-JUMP
                      'FALSE
                      (LABEL-CNT CINFO))))))
                (LABEL-ALIST CINFO)
              (ADD1 (ADD1 (LABEL-CNT CINFO))))
            T-COND-LIST
            (IF-TRUE-BRANCH STMT)
            PROC-LIST)
            (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
              (CONS 'DL
                (CONS (LABEL-CNT CINFO)
                  '(NIL (NO-OP))))))
            T-COND-LIST
            (IF-FALSE-BRANCH STMT)
            PROC-LIST)))))
(FIND-LABEL
 (FETCH-LABEL
  (CC (MG-MEANING-R (IF-FALSE-BRANCH STMT)
   PROC-LIST MG-STATE
   (SUB1 N)
   (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))
  (LABEL-ALIST
   (TRANSLATE
    (ADD-CODE
     (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
       (LIST (LIST 'PUSH-LOCAL
         (IF-CONDITION STMT))
         '(FETCH-TEMP-STK)
         (LIST 'TEST-BOOL-AND-JUMP 'FALSE
          (LABEL-CNT CINFO)))))
       (LABEL-ALIST CINFO)
       (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (IF-TRUE-BRANCH STMT)
      PROC-LIST)
      (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO))))
      (CONS 'DL
        (CONS (LABEL-CNT CINFO)
          '(NIL (NO-OP)))))))))
  T-COND-LIST
  (IF-FALSE-BRANCH STMT)
  PROC-LIST))
  (APPEND
   (CODE
    (ADD-CODE
     (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
       (LIST (LIST 'PUSH-LOCAL
         (IF-CONDITION STMT))
         '(FETCH-TEMP-STK)
         (LIST 'TEST-BOOL-AND-JUMP 'FALSE
          (LABEL-CNT CINFO)))))
       (LABEL-ALIST CINFO)
       (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (IF-TRUE-BRANCH STMT)
      PROC-LIST))
      (APPEND
       (CODE
        (ADD-CODE
         (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
           (LIST (LIST 'PUSH-LOCAL
             (IF-CONDITION STMT))
             '(FETCH-TEMP-STK)
             (LIST 'TEST-BOOL-AND-JUMP 'FALSE
              (LABEL-CNT CINFO)))))
           (LABEL-ALIST CINFO)
           (ADD1 (ADD1 (LABEL-CNT CINFO))))
          T-COND-LIST
          (IF-TRUE-BRANCH STMT)
          PROC-LIST))
          (APPEND
           (CODE
            (ADD-CODE
             (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
               (LIST (LIST 'PUSH-LOCAL
                 (IF-CONDITION STMT))
                 '(FETCH-TEMP-STK)
                 (LIST 'TEST-BOOL-AND-JUMP 'FALSE
                  (LABEL-CNT CINFO)))))
               (LABEL-ALIST CINFO)
               (ADD1 (ADD1 (LABEL-CNT CINFO))))
              T-COND-LIST
              (IF-TRUE-BRANCH STMT)
(CONS 'DL
  (CONS (LABEL-CNT CINFO)
    '(NIL (NO-OP))))))

T-COND-LIST
(IF-FALSE-BRANCH STMT)
PROC-LIST))
  (CONS (CONS 'DL
    (CONS (ADD1 (LABEL-CNT CINFO))
      '(NIL (NO-OP)))))
  CODE2)))))))

CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (IF-FALSE-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 (IF-FALSE-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))))))

(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)))))
(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)
(TRANSFORM-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)))))

(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 @TRUE-STATE1

(MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK

(TAG 'PC

(CONS SUBR

(LENGTH (CODE (MAKE-CINFO (APPEND (CODE CINFO) (LIST (LIST 'PUSH-LOCAL

(IF-CONDITION STMT))

'(FETCH-TEMP-STK)

(LIST 'TEST-BOOL-AND-JUMP

'FALSE

(LABEL-CNT CINFO))))))

(LABEL-ALIST CINFO)

(ADD1 (ADD1 (LABEL-CNT CINFO))))))))

T-COND-LIST))

(ADD-ABBREVIATION @TRUE-TIME

(CLOCK (IF-TRUE-BRANCH STMT))

52
PROC-LIST MG-STATE
(SUB1 N))

(ADD-ABBREVIATION @TRUE-STATE2
(P-STATE
(TAG 'PC
(CONS SUBR
(IF
(NORMAL (MG-MEANING-R (IF-TRUE-BRANCH STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))

(LENGTH
(CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (LIST 'PUSH-LOCAL
(IF-CONDITION STMT))
'(FETCH-TEMP-STK)
(LIST 'TEST-BOOL-AND-JUMP
'FALSE
(LABEL-CNT CINFO))))
(LABEL-ALIST CINFO)
(ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST))))

(FIND-LABEL
(FETCH-LABEL
(CC (MG-MEANING-R (IF-TRUE-BRANCH STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))

(LABEL-ALIST
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (LIST 'PUSH-LOCAL
(IF-CONDITION STMT))
'(FETCH-TEMP-STK)
(LIST 'TEST-BOOL-AND-JUMP
'FALSE
(LABEL-CNT CINFO))))
(LABEL-ALIST CINFO)
(ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
(IF-TRUE-BRANCH STMT)
PROC-LIST))
(APPPEND
  (CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
  (LIST (LIST 'PUSH-LOCAL
            (IF-CONDITION STMT))
            '(FETCH-TEMP-STK)
            (LIST 'TEST-BOOL-AND-JUMP
               'FALSE
               (LABEL-CNT CINFO))))
  (LABEL-ALIST CINFO)
  (ADD1 (ADD1 (LABEL-CNT CINFO))))
  (T-COND-LIST
   (IF-TRUE-BRANCH STMT)
   PROC-LIST))
  (CONS
   (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
  (CONS
   (CONS 'DL
   (CONS (LABEL-CNT CINFO)
  '(NIL (NO-OP))))
  (APPEND
   (CODE
    (TRANSLATE
     (NULLIFY (TRANSLATE (MAKE-CINFO NIL
      (LABEL-ALIST CINFO)
      (ADD1 (ADD1 (LABEL-CNT CINFO))))
      (T-COND-LIST
       (IF-TRUE-BRANCH STMT)
       PROC-LIST))
    T-COND-LIST
    (IF-FALSE-BRANCH STMT)
    PROC-LIST))
  (CONS (CONS 'DL
  (CONS (ADD1 (LABEL-CNT CINFO))
  '(NIL (NO-OP))))
  (CODE2))))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (IF-TRUE-BRANCH STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))
  (BINDINGS (TOP CTRL-STK))

54
(TRANSLATE-PROC-LIST PROC-LIST)

LIST

LIST 'C-C

(MG-COND-TO-P-NAT (CC (MG-MEANING-R (IF-TRUE-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 @FALSE-STATE1

(MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK

(TAG 'PC

(CDONS SUBR

(LENGTH

(CODE

(ADD-CODE

(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)

(LIST (LIST 'PUSH-LOCAL

(IF-CONDITION STMT))

'(FETCH-TEMP-STK)

(LIST 'TEST-BOOL-AND-JUMP

'FALSE

(LABEL-CNT CINFO)))

(LABEL-ALIST CINFO)

(ADD1 (ADD1 (LABEL-CNT CINFO))))

T-COND-LIST)

(IF-TRUE-BRANCH STMT)

PROC-LIST)

(LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO))))

(CONS 'DL

(CONS (LABEL-CNT CINFO)

'(NIL (NO-OP))))))))))

T-COND-LIST))

(ADD-ABBREVIATION @FALSE-TIME

(CLOCK (IF-FALSE-BRANCH STMT)

PROC-LIST MG-STATE

(SUB1 N)))))

(ADD-ABBREVIATION @FALSE-STATE2

(P-STATE

55
(TAG 'PC
(CONS SUBR
  (IF
   (NORMAL (MG-MEANING-R (IF-FALSE-BRANCH STMT))
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))

(LIST (LENGTH
  (CODE
   (TRANSLATE
    (ADD-CODE
     (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
       (LIST (LIST 'PUSH-LOCAL
         'IF-CONDITION STMT))
       '(FETCH-TEMP-STK)
       (LIST 'TEST-BOOL-AND-JUMP
         'FALSE
         (LABEL-CNT CINFO))))
      (LABEL-ALIST CINFO)
      (ADD1 (ADD1 (LABEL-CNT CINFO)))))
     T-COND-LIST
     (IF-TRUE-BRANCH STMT))
    PROC-LIST)
    (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
      (CONS 'DL
        (CONS (LABEL-CNT CINFO)
          '(NIL (NO-OP)))))))
  T-COND-LIST
  (IF-FALSE-BRANCH STMT)
  PROC-LIST))
(FIND-LABEL
(FETCH-LABEL
(CC (MG-MEANING-R (IF-FALSE-BRANCH STMT))
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK)))))
(LABEL-ALIST
(TRANSLATE
  (ADD-CODE
   (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
     (LIST (LIST 'PUSH-LOCAL
       'IF-CONDITION STMT))
     (LABEL-CNT CINFO)))))
  (ADD1 (LABEL-CNT CINFO))))

56
'(FETCH-TEMP-STK)
   (LIST '
      TEST-BOOL-AND-JUMP
      'FALSE
      (LABEL-CNT CINFO)))
   (LABEL-ALIST CINFO)
   (ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
   (IF-TRUE-BRANCH STMT)
PROC-LIST)
   (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
      (CONS 'DL
      (CONS (LABEL-CNT CINFO)
      '(NIL (NO-OP))))))
T-COND-LIST
   (IF-FALSE-BRANCH STMT)
PROC-LIST))
(APPEND
   (CODE
   (TRANSLATE
   (ADD-CODE
   (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (LIST 'PUSH-LOCAL
      (IF-CONDITION STMT))
      '(FETCH-TEMP-STK)
      (LIST 'TEST-BOOL-AND-JUMP
      'FALSE
      (LABEL-CNT CINFO))))
       (LABEL-ALIST CINFO)
       (ADD1 (ADD1 (LABEL-CNT CINFO)))
T-COND-LIST
   (IF-TRUE-BRANCH STMT)
PROC-LIST)
   (LIST (LIST 'JUMP (ADD1 (LABEL-CNT CINFO)))
   (CONS 'DL
   (CONS (LABEL-CNT CINFO)
   '(NIL (NO-OP))))))
T-COND-LIST
   (IF-FALSE-BRANCH STMT)
PROC-LIST))
   (CONS (CONS 'DL
   (CONS (ADD1 (LABEL-CNT CINFO))
   '(NIL (NO-OP))))
   CODE2)))))
CTRL-STK

57
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (IF-FALSE-BRANCH STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))
  (BINDINGS (TOP CTRL-STK))
  TEMP-STK)
(TTRANSLATE-PROC-LIST PROC-LIST)
  (LIST
    (LIST 'C-C
      (MG-COND-TO-P-NAT (CC (MG-MEANING-R (IF-FALSE-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))
(PROMOTE
  (CLAIM (MG-EXPRESSION-FALSEP (IF-CONDITION STMT)
    MG-STATE)
    0)
  (DROP 19)
  (DEMOTE 19)
  (DIVE 1 1)
(PUSH TOP PROMOTE
  (CLAIM (EQUAL (P @INITIAL 4) @FALSE-STATE1)
    0)
  (CLAIM (NORMAL (MG-MEANING-R (IF-FALSE-BRANCH STMT)
      PROC-LIST MG-STATE
      (SUB1 N)
      (LIST (LENGTH TEMP-STK)
        (P-CTRL-STK-SIZE CTRL-STK)))))
    0)
  (CLAIM (EQUAL (P-STEP @FALSE-STATE2) @FINAL)
    0)
  (CLAIM (EQUAL @STMT-TIME
    (PLUS 5 @FALSE-TIME)))
    0)
  (DEMOTE 20 21 23 24)
(DROP

58
(GENERALIZE ((@FALSE-STATE2 FALSE-STATE2)
  (@FALSE-TIME FALSE-TIME)
  (@FALSE-STATE1 FALSE-STATE1)
  (@TRUE-STATE2 TRUE-STATE2)
  (@TRUE-TIME TRUE-TIME)
  (@TRUE-STATE1 TRUE-STATE1)
  (@FINAL FINAL)
  (@STMT-TIME STMT-TIME)
  (@INITIAL INITIAL)))

DROP
(USE-LEMMA IF-FALSE-NORMAL-EXACT-TIME-SCHEMA)
PROVE
(CONTRADICT 24)
(DIVE 1)
(REWRITE IF-CLOCK-FALSE-NORMAL)
TOP S
(CONTRADICT 23)
(DIVE 1)
(REWRITE IF-STEP-NORMAL-FALSE-STATE2-EQUALS-FINAL)
TOP S-PROP
(CLAIM (EQUAL @FALSE-STATE2 @FINAL) 0)
(CLAIM (EQUAL @STMT-TIME
  (PLUS 4 @FALSE-TIME))
  0)
(DEMOTE 20 21 23 24)
(GENERALIZE ((@FALSE-STATE2 FALSE-STATE2)
  (@FALSE-TIME FALSE-TIME)
  (@FALSE-STATE1 FALSE-STATE1)
  (@TRUE-STATE2 TRUE-STATE2)
  (@TRUE-TIME TRUE-TIME)
  (@TRUE-STATE1 TRUE-STATE1)
  (@FINAL FINAL)
  (@STMT-TIME STMT-TIME)
  (@INITIAL INITIAL)))

DROP
(USE-LEMMA IF-FALSE-NONNORMAL-EXACT-TIME-SCHEMA)
PROVE
(CONTRADICT 24)
(DIVE 1)
(REWRITE IF-CLOCK-FALSE-NONNORMAL)
TOP S
(CONTRADICT 23)
(DIVE 1)
(REWRITE IF-NON-NORMAL-FALSE-STATE2-EQUALS-FINAL)
TOP S-PROP
(CONTRADICT 21)
(DROP 20 21)
(DIVE 1)
(REWRITE P-ADD1-3)
(DIVE 1)
(REWRITE IF-INITIAL-STEP1)
UP
(REWRITE P-ADD1-3)
(DIVE 1)
(REWRITE IF-INITIAL-STEP2)
UP
(REWRITE P-ADD1-3)
(DIVE 1)
(REWRITE IF-INITIAL-STEP3-FALSE-TEST)
UP
(REWRITE P-ADD1-3)
(REWRITE P-0-UNWINDING-LEMMA)
(REWRITE IF-INITIAL-STEP4-FALSE-TEST)
TOP S-PROP SPLIT
(REWRITE IF-FALSE-BRANCH-HYPS)
(REWRITE IF-FALSE-BRANCH-HYPS)
(DIVE 1)
(REWRITE IF-FALSE-BRANCH-HYPS)
TOP S
(DIVE 1)
(REWRITE IF-FALSE-BRANCH-HYPS)
TOP S
(DROP 20)
(DEMOTE 19)
(DIVE 1 1)
PUSH TOP PROMOTE
(CLAIM (EQUAL (P @INITIAL 3) @TRUE-STATE1)
  0)
(CLAIM (NORMAL (MG-MEANING-R (IF-TRUE-BRANCH STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
  0)
(CLAIM (EQUAL (P @TRUE-STATE2 2) @FINAL)
  0)
(CLAIM (EQUAL @STMT-TIME (PLUS 5 @TRUE-TIME))
  0)
(demote 20 21 23 24)
(generalize ((@false-state2 false-state2)
   (@false-time false-time)
   (@false-state1 false-state1)
   (@true-state2 true-state2)
   (@true-time true-time)
   (@true-state1 true-state1)
   (@final final)
   (@stmt-time stmt-time)
   (@initial initial)))

drop
(use-lemma if-true-normal-exact-time-schema)
prove
(contradict 24)
(dive 1)
(rewrite if-clock-true-normal)
top s
(contradict 23)
(drop 20 21 23)
(dive 1)
(rewrite p-add1-3)
(rewrite p-add1-3)
(rewrite p-0-unwinding-lemma)
(dive 1)
(rewrite if-normal-true-state2-step1)
up
(rewrite if-normal-true-state2-step2-equals-final)
top s-prop
(claim (equal @true-state2 @final) 0)
(claim (equal @stmt-time (plus 3 @true-time)) 0)
(demote 20 21 23 24)
(generalize ((@false-state2 false-state2)
   (@false-time false-time)
   (@false-state1 false-state1)
   (@true-state2 true-state2)
   (@true-time true-time)
   (@true-state1 true-state1)
   (@final final)
   (@stmt-time stmt-time)
   (@initial initial)))

drop
(use-lemma if-true-nonnormal-exact-time-schema)
prove
(CONTRADICT 24)
(DIVE 1)
(REWRITE IF-CLOCK-TRUE-NONNORMAL)
TOP S
(CONTRADICT 23)
(DIVE 1)
(REWRITE IF-NONNORMAL-TRUE-STATE2-EQUALS-FINAL)
TOP S-PROP
(CONTRADICT 21)
(DIVE 1)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-O-UNWINDING-LEMMA)
(DIVE 1 1)
(REWRITE IF-INITIAL-STEP1)
UP
(REWRITE IF-INITIAL-STEP2)
UP
(REWRITE IF-INITIAL-STEP3-TRUE-TEST-EQUALS-TRUE-STATE1)
UP S SPLIT
(REWRITE IF-TRUE-BRANCH-HYPS)
(REWRITE IF-TRUE-BRANCH-HYPS)
(DIVE 1)
(REWRITE IF-TRUE-BRANCH-HYPS)
TOP S
(DIVE 1)
(REWRITE IF-TRUE-BRANCH-HYPS)
TOP S))}

Event: Make the library "c-if".
Index

add-code, 2, 3, 5, 6, 14
all-cars-unique, 1, 9–11, 13, 23, 35

bindings, 8–14, 23, 24, 35–37
boolean-identifierp, 1, 10, 23
boolean-identifierp-not-false-expressionp, 23
boolean-value-maps-down, 10

cc, 9, 10, 12, 14, 24, 37, 38
clock, 38, 39
code, 1, 3–14, 23, 24, 29, 35–37

fetch-boolean-identifier-ins-okp, 1
fetch-label, 37
find-label, 37
find-labelp, 11, 29
get-m-value, 11, 23

if-clock-false-nonnormal, 38
if-clock-false-normal, 38
if-clock-true-nonnormal, 39
if-clock-true-normal, 38
if-code-rewrite1, 3
if-code-rewrite2, 3
if-condition, 1–7, 9–14, 24, 29, 35, 36, 38, 39
if-condition-boolean-identifierp, 1
if-false-branch, 2–8, 36, 38
if-false-branch-doesnt-halt, 2
if-false-branch-hyps, 4
if-false-nonnormal-exact-time-schema, 39
if-false-normal-exact-time-schema, 39
if-find-labelp-lemma1, 11
if-find-labelp-lemma2, 11
if-false-normal-exact-time-schema, 39
if-find-labelp-lemma1, 11
if-find-labelp-lemma2, 11
if-initial-step1, 3
if-initial-step2, 9
if-initial-step3-false-test, 11
if-initial-step3-true-test-equa

ls-true-state1, 23
if-initial-step4-false-test, 13
if-meaning-r-2, 2
if-normal-true-state2-step2-equals-final, 35
if-translation-2, 1
if-true-branch, 1–8, 11, 12, 14, 29, 36, 39
if-true-branch-doesnt-halt, 2
if-true-branch-hyps, 6
if-true-nonnormal-exact-time-schema, 40
if-true-normal-exact-time-schema, 39

label-alist, 1, 3–5, 7, 8, 11–14, 24, 29, 36
label-cnt, 1–8, 11–14, 24, 29, 36
length, 1, 4, 6, 8–14, 23, 24, 35–38

make-cinfo, 1, 3–5, 7, 8, 11, 12, 14, 24, 29, 36
map-down, 9, 14, 25
map-down-values, 9, 10, 12, 14, 24, 36, 37
mg-alist, 1, 8–14, 23, 24, 35–37
mg-alistp, 10, 23
mg-cond-to-p-nat, 9, 10, 12, 14, 24, 37, 38
mg-expression-falsep, 2, 3, 5, 6, 12, 13, 24, 35, 38, 39
mg-max-ctrl-stk-size, 9, 10, 12–14, 24, 37, 38
mg-max-temp-stk-size, 1, 9, 10, 12–14, 24, 37, 38
mg-meaning-r, 2, 3, 5, 6, 8–10, 12, 13, 24, 35–39
mg-psw, 2, 3
mg-to-p-simple-literal, 11
mg-vars-list-ok-in-p-state, 1, 8–11, 13, 23, 35

63
mg-word-size, 9, 10, 12–14, 24, 37, 38

no-p-aliasing, 10, 11, 13, 24, 35
normal, 2, 4, 6, 8, 9, 11, 13, 24, 35–39
nullify, 4, 7, 8

ok-mg-def-plistp, 8, 9, 11, 13, 23, 35
ok-mg-statement, 1, 3–6, 8, 9, 11, 13, 23, 29, 35
ok-mg-statep, 1, 8, 9, 11, 13, 23, 35
ok-translation-parameters, 4–7, 11, 13, 23, 29, 35

p, 39, 40
p-ctrl-stk-size, 5, 6, 8–13, 23, 24, 35–38
p-state, 9, 10, 12–14, 24, 37, 38
p-step, 9, 10, 12, 14, 24, 37, 39
push, 9, 10, 12, 24

resource-errorp, 2, 3, 5, 6, 8–10, 12, 13, 24, 35, 38, 39
resources-inadequatep, 2, 8, 9, 11, 13, 23, 35
rget, 10, 12, 24

signal-system-error, 2
signatures-match, 1, 9, 11, 13, 23, 35
small-naturalp, 1
tag, 9, 10, 12, 14, 24, 36, 37
top, 8–14, 23, 24, 35–37
translate, 1–9, 11–14, 24, 29, 35–37
translate-def-body, 4–9, 11, 13, 23, 35
translate-proc-list, 9, 10, 12, 14, 24, 36, 38
type, 1
untag, 1, 10, 12, 24
user-defined-procp, 4, 6, 8, 9, 11, 13, 24, 35
value, 1, 9, 10, 12, 24