

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

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                                      ;;
;;                                                                                      ;;
;;                                EXACT-TIME IF CASE                                ;;
;;                                                                                      ;;
;;                                                                                      ;;
;;                                                                                      ;;
;;                                                                                      ;;

```

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))
  ∧ (caddr (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-cnt (cinfo)))))
                                                                    label-alist (cinfo),
                                                                    1 + (1 + label-cnt (cinfo))),
                                                                    cond-list,
                                                                    if-true-branch (stmt),
                                                                    proc-list),
                                                                    list (list ('jump,
                                                                    1 + label-cnt (cinfo)),

```

```

list ('d1,
      label-cnt (cinfo),
      nil,
      ' (no-op) )),
cond-list,
if-false-branch (stmt),
proc-list,
list (list ('d1, 1 + label-cnt (cinfo), nil, ' (no-op) )))

```

THEOREM: if-meaning-r-2

```

(car (stmt) = 'if-mg)
→ (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)
   = if n ≈ 0 then signal-system-error (mg-state, 'timed-out)
     elseif ¬ normal (mg-state) then mg-state
     elseif resources-inadequatep (stmt, proc-list, sizes)
     then signal-system-error (mg-state, 'resource-error)
     elseif mg-expression-falsep (if-condition (stmt), mg-state)
     then mg-meaning-r (if-false-branch (stmt),
                        proc-list,
                        mg-state,
                        n - 1,
                        sizes)
     else mg-meaning-r (if-true-branch (stmt),
                        proc-list,
                        mg-state,
                        n - 1,
                        sizes) endif)

```

THEOREM: if-false-branch-doesnt-halt

```

((n ≠ 0)
 ∧ (car (stmt) = 'if-mg)
 ∧ normal (mg-state)
 ∧ 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-false-branch (stmt),
                        proc-list,
                        mg-state,
                        n - 1,
                        sizes))
   = 'run)

```

THEOREM: if-true-branch-doesnt-halt

```

((n ≠ 0)
 ∧ (car (stmt) = 'if-mg)
 ∧ normal (mg-state)

```

$$\begin{aligned}
& \wedge (\neg \text{mg-expression-falsep}(\text{if-condition}(stmt), mg\text{-state})) \\
& \wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, proc\text{-list}, mg\text{-state}, n, sizes))) \\
\rightarrow & (\text{mg-psw}(\text{mg-meaning-r}(\text{if-true-branch}(stmt), \\
& \quad proc\text{-list}, \\
& \quad mg\text{-state}, \\
& \quad n - 1, \\
& \quad sizes)) \\
& = \text{'run})
\end{aligned}$$

THEOREM: if-code-rewrite1

$$\begin{aligned}
& ((\text{car}(stmt) = \text{'if-mg}) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list})) \\
\rightarrow & (\text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), code2) \\
& = \text{append}(\text{code}(\text{translate}(\text{add-code}(\text{translate}(\text{make-cinfo}(\text{append}(\text{code}(cinfo), \\
& \quad \text{list}(\text{list}(\text{'push-local}, \\
& \quad \text{if-condition}(stmt)), \\
& \quad \text{'(fetch-temp-stk)}, \\
& \quad \text{list}(\text{'test-bool-and-jump}, \\
& \quad \text{'false}, \\
& \quad \text{label-cnt}(cinfo)))), \\
& \quad \text{label-alist}(cinfo), \\
& \quad 1 + (1 + \text{label-cnt}(cinfo))), \\
& \quad t\text{-cond-list}, \\
& \quad \text{if-true-branch}(stmt), \\
& \quad proc\text{-list}), \\
& \quad \text{list}(\text{list}(\text{'jump}, \\
& \quad 1 + \text{label-cnt}(cinfo)), \\
& \quad \text{cons}(\text{'dl}, \\
& \quad \text{cons}(\text{label-cnt}(cinfo), \\
& \quad \text{'(nil} \\
& \quad \quad \text{(no-op)}))))), \\
& \quad t\text{-cond-list}, \\
& \quad \text{if-false-branch}(stmt), \\
& \quad proc\text{-list}), \\
& \quad \text{cons}(\text{cons}(\text{'dl}, \\
& \quad \text{cons}(1 + \text{label-cnt}(cinfo), \\
& \quad \text{'(nil (no-op)}))), \\
& \quad code2)))
\end{aligned}$$

THEOREM: if-code-rewrite2

$$\begin{aligned}
& ((\text{car}(stmt) = \text{'if-mg}) \\
& \wedge \text{ok-mg-statement}(stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list})) \\
\rightarrow & (\text{append}(\text{code}(\text{translate}(cinfo, t\text{-cond-list}, stmt, proc\text{-list})), code2) \\
& = \text{append}(\text{code}(\text{translate}(\text{make-cinfo}(\text{append}(\text{code}(cinfo),
\end{aligned}$$

```

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-false-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)),

```

```

                                p-ctrl-stk-size (ctrl-stk))))
  ∧ mg-expression-falsep (if-condition (stmt), mg-state)
→ (ok-mg-statement (if-false-branch (stmt),
                                r-cond-list,
                                name-alist,
                                proc-list)
  ∧ ok-translation-parameters (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),
                                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),
                                '(nil (no-op)))),
                                code2))
  ∧ (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),
                                1 + (1 + label-cnt (cinfo))),
                                t-cond-list,
                                if-true-branch (stmt),
                                proc-list),

```

$$\begin{aligned}
& \text{list}(\text{list}(' \text{jump}, \\
& \quad 1 + \text{label-cnt}(\text{cinfo}), \\
& \quad \text{cons}(' \text{dl}, \\
& \quad \quad \text{cons}(\text{label-cnt}(\text{cinfo}), \\
& \quad \quad \quad '(\text{nil} \\
& \quad \quad \quad \quad (\text{no-op}))))), \\
& \quad t\text{-cond-list}, \\
& \quad \text{if-false-branch}(\text{stmt}), \\
& \quad \text{proc-list}), \\
& \text{cons}(\text{cons}(' \text{dl}, \\
& \quad \text{cons}(1 + \text{label-cnt}(\text{cinfo}), \\
& \quad \quad '(\text{nil} \ (\text{no-op}))), \\
& \quad \text{code2})) \\
\wedge \quad & (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{if-false-branch}(\text{stmt}), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))))))
\end{aligned}$$

THEOREM: if-true-branch-hyps

$$\begin{aligned}
& ((n \neq 0) \\
& \wedge \quad (\text{car}(\text{stmt}) = ' \text{if-mg}) \\
& \wedge \quad \text{ok-mg-statement}(\text{stmt}, r\text{-cond-list}, \text{name-alist}, \text{proc-list}) \\
& \wedge \quad \text{ok-translation-parameters}(\text{cinfo}, t\text{-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \\
& \wedge \quad (\text{code}(\text{translate-def-body}(\text{assoc}(\text{subr}, \text{proc-list}), \text{proc-list})) \\
& \quad = \text{append}(\text{code}(\text{translate}(\text{cinfo}, t\text{-cond-list}, \text{stmt}, \text{proc-list})), \\
& \quad \quad \text{code2})) \\
& \wedge \quad \text{user-defined-procp}(\text{subr}, \text{proc-list}) \\
& \wedge \quad \text{normal}(\text{mg-state}) \\
& \wedge \quad (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \quad \text{p-ctrl-stk-size}(\text{ctrl-stk})))))) \\
& \wedge \quad (\neg \text{mg-expression-falsep}(\text{if-condition}(\text{stmt}), \text{mg-state})) \\
\rightarrow \quad & (\text{ok-mg-statement}(\text{if-true-branch}(\text{stmt}), \\
& \quad r\text{-cond-list}, \\
& \quad \text{name-alist}, \\
& \quad \text{proc-list}) \\
& \wedge \quad \text{ok-translation-parameters}(\text{make-cinfo}(\text{append}(\text{code}(\text{cinfo}), \\
& \quad \quad \text{list}(\text{list}(' \text{push-local}, \\
& \quad \quad \quad \text{if-condition}(\text{stmt}))),
\end{aligned}$$

```

                                '(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-a
                                                                1 + (1
                                                                t-cond-list,
                                                                if-true-branch(stm
                                                                proc-list))),
                                                                t-cond-list,
                                                                if-false-branch(stmt),
                                                                proc-list))),
                  cons(cons('dl,
                              cons(1 + label-cnt(cinfo),
                                    '(nil
                                      (no-op))))),
                          code2))))))
^ (code(translate-def-body(assoc(subr, proc-list), 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),
                                                  1 + (1 + label-cnt(cinfo))),
t-cond-list,
if-true-branch(stmt),
proc-list))),
cons(list('jump, 1 + label-cnt(cinfo)),
      cons(cons('dl,

```

$$\begin{aligned}
& \text{cons}(\text{label-cnt}(\text{cinfo}), \\
& \quad '(\text{nil} \text{ no-op}))), \\
& \text{append}(\text{code}(\text{translate}(\text{nullify}(\text{translate}(\text{make-cinfo}(\text{nil}, \\
& \quad \text{label-alist}(\text{cinfo}), \\
& \quad 1 + (1 + \text{label-cnt}(\text{cinfo}), \\
& \quad \text{t-cond-list}, \\
& \quad \text{if-true-branch}(\text{stmt}), \\
& \quad \text{proc-list}))), \\
& \quad \text{t-cond-list}, \\
& \quad \text{if-false-branch}(\text{stmt}), \\
& \quad \text{proc-list}))), \\
& \text{cons}(\text{cons}('dl, \\
& \quad \text{cons}(1 + \text{label-cnt}(\text{cinfo}), \\
& \quad '(\text{nil} \\
& \quad \quad (\text{no-op}))), \\
& \quad \text{code2}})))))) \\
\wedge \quad & (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{if-true-branch}(\text{stmt}), \\
& \quad \text{proc-list}, \\
& \quad \text{mg-state}, \\
& \quad n - 1, \\
& \quad \text{list}(\text{length}(\text{temp-stk}), \\
& \quad \text{p-ctrl-stk-size}(\text{ctrl-stk}))))))
\end{aligned}$$

THEOREM: if-initial-step1

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

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

THEOREM: boolean-value-maps-down

$$\begin{aligned}
& (\text{boolean-identifierp } (b, \text{ mg-vars}) \\
& \wedge \text{ mg-vars-list-ok-in-p-state } (\text{mg-vars}, \text{ bindings}, \text{ temp-stk}) \\
& \wedge \text{ mg-alistp } (\text{mg-vars}) \\
& \wedge \text{ no-p-aliasing } (\text{bindings}, \text{ mg-vars}) \\
& \wedge \text{ all-cars-unique } (\text{mg-vars}) \\
\rightarrow & \text{ (rget (untag (value (b, bindings)),} \\
& \quad \text{map-down-values (mg-vars, bindings, temp-stk))}
\end{aligned}$$

$$= \text{mg-to-p-simple-literal}(\text{get-m-value}(b, \text{mg-vars}))$$

THEOREM: if-find-labelp-lemma1

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

THEOREM: if-initial-step3-false-test

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

$$\begin{aligned}
& \text{mg-state,} \\
& n, \\
& \text{list (length (temp-stk),} \\
& \quad \text{p-ctrl-stk-size (ctrl-stk))))) \\
\wedge & \text{mg-expression-falsep (if-condition (stmt), mg-state))} \\
\rightarrow & (\text{p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 2)),} \\
& \quad \text{ctrl-stk,} \\
& \quad \text{push (rget (untag (value (if-condition (stmt),} \\
& \quad \quad \text{bindings (top (ctrl-stk))}),} \\
& \quad \text{map-down-values (mg-alist (mg-state),} \\
& \quad \quad \text{bindings (top (ctrl-stk)),} \\
& \quad \quad \text{temp-stk)),} \\
& \quad \text{map-down-values (mg-alist (mg-state),} \\
& \quad \quad \text{bindings (top (ctrl-stk)),} \\
& \quad \quad \text{temp-stk)),} \\
& \text{translate-proc-list (proc-list),} \\
& \text{list (list ('c-c,} \\
& \quad \text{mg-cond-to-p-nat (cc (mg-state), t-cond-list))),} \\
& \text{MG-MAX-CTRL-STK-SIZE,} \\
& \text{MG-MAX-TEMP-STK-SIZE,} \\
& \text{MG-WORD-SIZE,} \\
& \text{'run))} \\
= & \text{p-state (tag ('pc,} \\
& \quad \text{cons (subr,} \\
& \quad \text{length (code (translate (make-cinfo (append (code (cinfo),} \\
& \quad \quad \text{list (list ('push-local,} \\
& \quad \quad \text{if-condition (stmt)),} \\
& \quad \quad \text{'(fetch-temp-stk),} \\
& \quad \quad \text{list ('test-bool-and-jump,} \\
& \quad \quad \text{'false,} \\
& \quad \quad \text{label-cnt (cinfo))}),} \\
& \quad \text{label-alist (cinfo),} \\
& \quad \text{1 + (1 + label-cnt (cinfo))),} \\
& \quad \text{t-cond-list,} \\
& \quad \text{if-true-branch (stmt),} \\
& \quad \text{proc-list))})} \\
& \quad + 1)), \\
& \text{ctrl-stk,} \\
& \text{map-down-values (mg-alist (mg-state),} \\
& \quad \text{bindings (top (ctrl-stk)),} \\
& \quad \text{temp-stk),} \\
& \text{translate-proc-list (proc-list),} \\
& \text{list (list ('c-c,} \\
& \quad \text{mg-cond-to-p-nat (cc (mg-state), t-cond-list))),}
\end{aligned}$$

MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: if-initial-step4-false-test

(($n \neq 0$)
 $\wedge$ (car(*stmt*) = 'if-mg)
 $\wedge$ ($\neg$ resources-inadequatep(*stmt*,
proc-list,
list(length(*temp-stk*),
p-ctrl-stk-size(*ctrl-stk*))))
 $\wedge$ ok-mg-statement(*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 $\wedge$ mg-vars-list-ok-in-p-state(mg-alist(*mg-state*),
bindings(top(*ctrl-stk*)),
temp-stk)
 $\wedge$ ok-mg-statep(*mg-state*, *r-cond-list*)
 $\wedge$ ok-translation-parameters(*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
 $\wedge$ signatures-match(mg-alist(*mg-state*), *name-alist*)
 $\wedge$ ok-mg-def-plistp(*proc-list*)
 $\wedge$ all-cars-unique(mg-alist(*mg-state*))
 $\wedge$ (code(translate-def-body(assoc(*subr*, *proc-list*), *proc-list*))
= append(code(translate(*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
 $\wedge$ user-defined-procp(*subr*, *proc-list*)
 $\wedge$ normal(*mg-state*)
 $\wedge$ no-p-aliasing(bindings(top(*ctrl-stk*)), mg-alist(*mg-state*))
 $\wedge$ ($\neg$ resource-errorp(mg-meaning-r(*stmt*,
proc-list,
mg-state,
n,
list(length(*temp-stk*),
p-ctrl-stk-size(*ctrl-stk*))))))
 $\wedge$ mg-expression-falsep(if-condition(*stmt*), *mg-state*))
 $\rightarrow$ (p-step(p-state(tag('pc,
cons(*subr*,
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*),

```

1 + (1 + label-cnt (cinfo))),
t-cond-list,
if-true-branch (stmt),
proc-list)))
+ 1)),
ctrl-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 (add-code (translate (make-cinfo (append (code (cinfo),
list (list ('push-local,
if-condition (st
' (fetch-temp-stk
list ('test-bool-a
'false,
label-cnt (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))

```

```

(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

```

```

(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
  (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))))
  (CONS (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))))
    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) (DIVE 1 1) (REWRITE IF-TRANSLATION-2)
UP UP (S LEMMAS) UP (S-PROP ADD-CODE) (S LEMMAS) (REWRITE GET-LENGTH-CAR)
S UP X UP X (DIVE 1) X UP S-PROP X TOP (S LEMMAS) (S-PROP ADD-CODE) PROVE)))
```

```
(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)
      (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
```

```

        (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
    (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-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)))

```

```

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)
  (REWRITE IF-MEANING-R-2) TOP S)))

```

EVENT: Disable signatures-match-preserves-plistp.

EVENT: Disable caddr-length-plistp-2.

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)))

```

$$\begin{aligned}
&= \text{append}(\text{code}(\text{translate}(cinfo, t\text{-}cond\text{-}list, stmt, proc\text{-}list)), \\
&\quad code2)) \\
\wedge & \text{user-defined-procp}(subr, proc\text{-}list) \\
\wedge & \text{normal}(mg\text{-}state) \\
\wedge & \text{no-p-aliasing}(\text{bindings}(\text{top}(ctrl\text{-}stk)), mg\text{-}alist(mg\text{-}state)) \\
\wedge & (\neg \text{resource-errorp}(mg\text{-}meaning\text{-}r(stmt, \\
&\quad proc\text{-}list, \\
&\quad mg\text{-}state, \\
&\quad n, \\
&\quad \text{list}(\text{length}(temp\text{-}stk), \\
&\quad \quad p\text{-}ctrl\text{-}stk\text{-}size(ctrl\text{-}stk)))))) \\
\wedge & (\neg mg\text{-}expression\text{-}falsep(\text{if-condition}(stmt), mg\text{-}state)) \\
\rightarrow & (\text{p-step}(\text{p-state}(\text{tag}('pc, \text{cons}(subr, \text{length}(\text{code}(cinfo)) + 2)), \\
&\quad ctrl\text{-}stk, \\
&\quad \text{push}(\text{rget}(\text{untag}(\text{value}(\text{if-condition}(stmt), \\
&\quad \quad \text{bindings}(\text{top}(ctrl\text{-}stk)))), \\
&\quad \quad \text{map-down-values}(mg\text{-}alist(mg\text{-}state), \\
&\quad \quad \quad \text{bindings}(\text{top}(ctrl\text{-}stk)), \\
&\quad \quad \quad temp\text{-}stk)), \\
&\quad \text{map-down-values}(mg\text{-}alist(mg\text{-}state), \\
&\quad \quad \text{bindings}(\text{top}(ctrl\text{-}stk)), \\
&\quad \quad \quad temp\text{-}stk)), \\
&\quad \text{translate-proc-list}(proc\text{-}list), \\
&\quad \text{list}(\text{list}('c-c, \\
&\quad \quad mg\text{-}cond\text{-}to\text{-}p\text{-}nat(cc(mg\text{-}state), t\text{-}cond\text{-}list))), \\
&\quad MG\text{-}MAX\text{-}CTRL\text{-}STK\text{-}SIZE, \\
&\quad MG\text{-}MAX\text{-}TEMP\text{-}STK\text{-}SIZE, \\
&\quad MG\text{-}WORD\text{-}SIZE, \\
&\quad 'run)) \\
&= \text{map-down}(mg\text{-}state, \\
&\quad proc\text{-}list, \\
&\quad ctrl\text{-}stk, \\
&\quad temp\text{-}stk, \\
&\quad \text{tag}('pc, \\
&\quad \quad \text{cons}(subr, \\
&\quad \quad \quad \text{length}(\text{code}(\text{make-cinfo}(\text{append}(\text{code}(cinfo), \\
&\quad \quad \quad \quad \text{list}(\text{list}('push\text{-}local, \\
&\quad \quad \quad \quad \quad \text{if-condition}(stmt), \\
&\quad \quad \quad \quad \quad 'fetch\text{-}temp\text{-}stk, \\
&\quad \quad \quad \quad \quad \text{list}('test\text{-}bool\text{-}and\text{-}jump, \\
&\quad \quad \quad \quad \quad \quad 'false, \\
&\quad \quad \quad \quad \quad \quad \text{label-cnt}(cinfo))))), \\
&\quad \quad \text{label-alist}(cinfo), \\
&\quad \quad 1 + (1 + \text{label-cnt}(cinfo)))))),
\end{aligned}$$

t-cond-list))

```
(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
    (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))
      (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
      (CONS SUBR
        (IF
          (NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N
            (LIST (LENGTH TEMP-STK)
              (P-CTRL-STK-SIZE CTRL-STK))))
          (LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
          (FIND-LABEL
            (FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
              (LIST (LENGTH TEMP-STK)
                (P-CTRL-STK-SIZE CTRL-STK))))
            (LABEL-ALIST (TRANSLATE CINFO T-COND-LIST STMT
              PROC-LIST)))
            (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
              CODE2))))
          CTRL-STK
            (MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R STMT PROC-LIST MG-STATE N
              (LIST (LENGTH TEMP-STK)
                (P-CTRL-STK-SIZE CTRL-STK))))
              (BINDINGS (TOP CTRL-STK))
              TEMP-STK)
            (TRANSLATE-PROC-LIST PROC-LIST)
            (LIST
              (LIST 'C-C
                (MG-COND-TO-P-NAT (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
              (LIST (LENGTH TEMP-STK)
                (P-CTRL-STK-SIZE CTRL-STK))))
                T-COND-LIST)))
              (MG-MAX-CTRL-STK-SIZE)
              (MG-MAX-TEMP-STK-SIZE)
              (MG-WORD-SIZE)
              'RUN)))
          ((INSTRUCTIONS PROMOTE S
            (= (MG-MEANING-R STMT PROC-LIST MG-STATE N
              (LIST (LENGTH TEMP-STK)
                (P-CTRL-STK-SIZE CTRL-STK)))
              (MG-MEANING-R (IF-TRUE-BRANCH STMT)
                PROC-LIST MG-STATE
                (SUB1 N)
                (LIST (LENGTH TEMP-STK)
                  (P-CTRL-STK-SIZE CTRL-STK)))

```

```

0)
S
(S LEMMAS)
(DIVE 2 2 2 1)
(= F)
TOP S-PROP
(DIVE 2 2 2 2 1)
(REWRITE NEW-CODE-APPENDED-TO-OLD1)
TOP
(S-PROP ADD-CODE NULLIFY)
S
(S LEMMAS)
(DIVE 2 2 2 2 2 2 2 1 1 1 3)
(REWRITE CODE-DOESNT-AFFECT-OTHER-FIELDS)
TOP
(S-PROP NULLIFY)
(S LEMMAS)
X
(S LEMMAS)
(DIVE 1)
(REWRITE IF-MEANING-R-2)
S TOP S)))

```

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))))

```

(prove-lemma if-normal-true-state2-step1 (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))
          (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-step
              (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)))
(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))
    (P-STATE
      (TAG 'PC

```

```

(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)))
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)

```

```

S
(DIVE 1)
(REWRITE FIND-LABELP-MONOTONIC-LESSP)
TOP S
(S-PROP NULLIFY)
S
(REWRITE LABEL-CNT-MONOTONIC3)
S PROVE
(S LEMMAS)
PROVE S
(DIVE 1)
(REWRITE IF-FIND-LABELP-LEMMA3)
TOP S
(REWRITE CAR-DEFINEDP-DEFINED-PROCP)))

```

THEOREM: if-normal-true-state2-step2-equals-final

```

(( $n \neq 0$ )
 $\wedge$  (car (stmt) = 'if-mg)
 $\wedge$  ( $\neg$  resources-inadequatep (stmt,
                                proc-list,
                                list (length (temp-stk),
                                        p-ctrl-stk-size (ctrl-stk))))
 $\wedge$  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 $\wedge$  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
 $\wedge$  ok-mg-statep (mg-state, r-cond-list)
 $\wedge$  ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2)
 $\wedge$  signatures-match (mg-alist (mg-state), name-alist)
 $\wedge$  ok-mg-def-plistp (proc-list)
 $\wedge$  all-cars-unique (mg-alist (mg-state))
 $\wedge$  (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 $\wedge$  user-defined-procp (subr, proc-list)
 $\wedge$  normal (mg-state)
 $\wedge$  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 $\wedge$  ( $\neg$  resource-errorp (mg-meaning-r (stmt,
                                      proc-list,
                                      mg-state,
                                      n,
                                      list (length (temp-stk),
                                              p-ctrl-stk-size (ctrl-stk))))))
 $\wedge$  ( $\neg$  mg-expression-falsep (if-condition (stmt), mg-state))

```

```

 $\wedge$    normal (mg-meaning-r (if-true-branch (stmt),
                                     proc-list,
                                     mg-state,
                                      $n - 1$ ,
                                     list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))))
 $\rightarrow$    (p-step (p-state (tag ('pc,
                               cons (subr,
                                       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),
                                                                                       1 + (1 + label-cnt (cinfo))),
                                                                                       t-cond-list,
                                                                                       if-true-branch (stmt),
                                                                                       proc-list))))))
                               + (1 + (1 + length (code (translate (make-cinfo (nil,
                                                                                       label-alist (cinfo),
                                                                                       label-cnt (translate (make-cinfo
                                                                                       t-cond-list,
                                                                                       if-true-l
                                                                                       proc-lis
                                                                                       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,
                                                                                        $n - 1$ ,
                                                                                       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 (if-true-branch (stmt),
                                                                                       proc-list,
                                                                                       mg-state,

```

```

                                n - 1,
                                list (length (temp-stk),
                                      p-ctrl-stk-size (ctrl-stk))),
                                t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
= p-state (tag ('pc,
               cons (subr,
                    if normal (mg-meaning-r (stmt,
                                             proc-list,
                                             mg-state,
                                             n,
                                             list (length (temp-stk),
                                                    p-ctrl-stk-size (ctrl-stk))))
                    then length (code (translate (cinfo,
                                                  t-cond-list,
                                                  stmt,
                                                  proc-list)))
                    else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                    proc-list,
                                                                    mg-state,
                                                                    n,
                                                                    list (length (temp-stk),
                                                                           p-ctrl-stk-size (ctrl-stk))))),
                                     label-alist (translate (cinfo,
                                                             t-cond-list,
                                                             stmt,
                                                             proc-list))),
                                     append (code (translate (cinfo,
                                                             t-cond-list,
                                                             stmt,
                                                             proc-list)),
                                             code2)) endif)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (stmt,
                                         proc-list,
                                         mg-state,
                                         n,
                                         list (length (temp-stk),
                                               p-ctrl-stk-size (ctrl-stk))),
bindings (top (ctrl-stk)),
temp-stk),

```

```

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

```

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)

```

$$\begin{aligned}
& \wedge \text{normal}(mg\text{-state}) \\
& \wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, proc\text{-list}, mg\text{-state}, n, sizes))) \\
& \wedge (\neg \text{mg-expression-falsep}(\text{if-condition}(stmt), mg\text{-state})) \\
& \wedge \text{normal}(\text{mg-meaning-r}(\text{if-true-branch}(stmt), \\
& \quad \quad \quad proc\text{-list}, \\
& \quad \quad \quad mg\text{-state}, \\
& \quad \quad \quad n - 1, \\
& \quad \quad \quad sizes))) \\
\rightarrow & (\text{clock}(stmt, proc\text{-list}, mg\text{-state}, n) \\
& = (5 + \text{clock}(\text{if-true-branch}(stmt), proc\text{-list}, mg\text{-state}, n - 1)))
\end{aligned}$$

THEOREM: if-clock-true-nonnormal

$$\begin{aligned}
& ((\text{car}(stmt) = \text{'if-mg})) \\
& \wedge (n \neq 0) \\
& \wedge \text{normal}(mg\text{-state}) \\
& \wedge (\neg \text{resource-errorp}(\text{mg-meaning-r}(stmt, proc\text{-list}, mg\text{-state}, n, sizes))) \\
& \wedge (\neg \text{mg-expression-falsep}(\text{if-condition}(stmt), mg\text{-state})) \\
& \wedge (\neg \text{normal}(\text{mg-meaning-r}(\text{if-true-branch}(stmt), \\
& \quad \quad \quad proc\text{-list}, \\
& \quad \quad \quad mg\text{-state}, \\
& \quad \quad \quad n - 1, \\
& \quad \quad \quad sizes)))) \\
\rightarrow & (\text{clock}(stmt, proc\text{-list}, mg\text{-state}, n) \\
& = (3 + \text{clock}(\text{if-true-branch}(stmt), proc\text{-list}, mg\text{-state}, n - 1)))
\end{aligned}$$

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

THEOREM: if-false-normal-exact-time-schema

$$\begin{aligned}
& ((stmt\text{-time} = (5 + false\text{-time})) \\
& \wedge (\text{p}(initial, 4) = false\text{-state1}) \\
& \wedge (\text{p}(false\text{-state1}, false\text{-time}) = false\text{-state2}) \\
& \wedge (\text{p-step}(false\text{-state2}) = final)) \\
\rightarrow & (\text{p}(initial, stmt\text{-time}) = final)
\end{aligned}$$

THEOREM: if-false-nonnormal-exact-time-schema

$$\begin{aligned}
& ((stmt\text{-time} = (4 + false\text{-time})) \\
& \wedge (\text{p}(initial, 4) = false\text{-state1}) \\
& \wedge (\text{p}(false\text{-state1}, false\text{-time}) = false\text{-state2}) \\
& \wedge (false\text{-state2} = final)) \\
\rightarrow & (\text{p}(initial, stmt\text{-time}) = final)
\end{aligned}$$

THEOREM: if-true-normal-exact-time-schema

$$\begin{aligned}
& ((stmt\text{-time} = (5 + true\text{-time})) \\
& \wedge (\text{p}(initial, 3) = true\text{-state1})
\end{aligned}$$

$$\begin{aligned}
& \wedge \quad (p(\text{true-state1}, \text{true-time}) = \text{true-state2}) \\
& \wedge \quad (p(\text{true-state2}, 2) = \text{final})) \\
\rightarrow & \quad (p(\text{initial}, \text{stmt-time}) = \text{final})
\end{aligned}$$

THEOREM: if-true-nonnormal-exact-time-schema

$$\begin{aligned}
& ((\text{stmt-time} = (3 + \text{true-time})) \\
& \wedge \quad (p(\text{initial}, 3) = \text{true-state1}) \\
& \wedge \quad (p(\text{true-state1}, \text{true-time}) = \text{true-state2}) \\
& \wedge \quad (\text{true-state2} = \text{final})) \\
\rightarrow & \quad (p(\text{initial}, \text{stmt-time}) = \text{final})
\end{aligned}$$

```

(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-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                                     ;;; 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
        (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))))))
(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 (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-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 (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

```



```

        (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-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 (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
        (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)
        ;; false-state1

```

```

(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
   (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-TEMP-STK-SIZE)
      (MG-WORD-SIZE)
      'RUN))))
(EQUAL
  (P (MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK
    (TAG 'PC
      (CONS SUBR (LENGTH (CODE CINFO))))
      T-COND-LIST)
    (CLOCK STMT PROC-LIST MG-STATE N))
  (P-STATE;; final
    (TAG 'PC
      (CONS SUBR
        (IF

```

```

;; initial
;; stmt-time

```

```

(NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N
  (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK))))
  (LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
  (FIND-LABEL
(FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
  (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK))))
  (LABEL-ALIST (TRANSLATE CINFO T-COND-LIST STMT
    PROC-LIST)))
(APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
CODE2))))))
  CTRL-STK
  (MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R STMT PROC-LIST MG-STATE N
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
    (BINDINGS (TOP CTRL-STK))
    TEMP-STK)
  (TRANSLATE-PROC-LIST PROC-LIST)
  (LIST
    (LIST 'C-C
  (MG-COND-TO-P-NAT (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK))))
    T-COND-LIST)))
  (MG-MAX-CTRL-STK-SIZE)
  (MG-MAX-TEMP-STK-SIZE)
  (MG-WORD-SIZE)
  'RUN)))
((INSTRUCTIONS
  (ADD-ABBREVIATION @INITIAL
    (MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK
      (TAG 'PC
        (CONS SUBR (LENGTH (CODE CINFO))))
      T-COND-LIST))
  (ADD-ABBREVIATION @STMT-TIME
    (CLOCK STMT PROC-LIST MG-STATE N))
  (ADD-ABBREVIATION @FINAL
    (P-STATE
      (TAG 'PC
        (CONS SUBR
  (IF
    (NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N
      (LIST (LENGTH TEMP-STK)

```

```

(P-CTRL-STK-SIZE CTRL-STK)))
(LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
(FIND-LABEL
(FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(LABEL-ALIST (TRANSLATE CINFO T-COND-LIST STMT
PROC-LIST)))
(APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
CODE2))))
CTRL-STK
(MAP-DOWN-VALUES
(MG-ALIST (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
(TRANSLATE-PROC-LIST PROC-LIST)
(LIST
(LIST 'C-C
(MG-COND-TO-P-NAT (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
T-COND-LIST)))
(MG-MAX-CTRL-STK-SIZE)
(MG-MAX-TEMP-STK-SIZE)
(MG-WORD-SIZE)
'RUN))
(ADD-ABBREVIATION @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)

```

```

                                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)))
    (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))))
              (CONS (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))
(ADD-ABBREVIATION @FALSE-STATE1
  (MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK
    (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)
                (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

```

```

(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
  (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-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

```

```

(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-O-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-lemma3, 29
- if-initial-step1, 8
- if-initial-step2, 9
- if-initial-step3-false-test, 11
- if-initial-step3-true-test-equal, 23
- if-initial-step4-false-test, 13
- if-meaning-r-2, 2
- if-normal-true-state2-step2-equal, 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, 37
- 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
- ls-true-state1, 23

- 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