EVENT: Start with the library "c-prog2".

THEOREM: loop-translation-2
(car (stmt) = 'loop-mg)
→ (translate (cinfo, cond-list, stmt, proc-list)
  = discard-label (add-code (translate (make-cinfo (append (code (cinfo),
          list (list ('dl,
                  label-cnt (cinfo),
                  nil,
                  '(no-op))))),
        cons (cons ('leave,
                  1 + label-cnt (cinfo)),
                  label-alist (cinfo)),
              1 + (1 + label-cnt (cinfo))'),
        cond-list,
        loop-body (stmt),
        proc-list),
        list (list ('jump, label-cnt (cinfo)),
        list ('dl,
        1 + label-cnt (cinfo),
        nil,
        '(push-constant
        (nat 2)),
        '(pop-global c-c)))))

THEOREM: loop-meaning-r-2
(car (stmt) = 'loop-mg)
→ (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)
  = if n ≃ 0 then signal-system-error (mg-state, 'timed-out)
    elseif ¬ normal (mg-state) then mg-state
    elseif resources-inadequatep (stmt, proc-list, sizes)
    then signal-system-error (mg-state, 'resource-error)
    else remove-leave (mg-meaning-r (stmt,
          proc-list),
          mg-meaning-r (loop-body (stmt),
          proc-list,
          mg-state),
          mg-state,
EVENT: Disable loop-meaning-r-2.

THEOREM: loop-body-doesnt-halt
\[(\text{car (stmt)} = \text{'loop-mg})
\land \text{normal (mg-state)}
\land (\neg \text{resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)})))
\rightarrow (\text{mg-psw (mg-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, sizes))}
\land \text{mg-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, sizes)})
\]

THEOREM: loop-sub1-body-doesnt-halt
\[(\text{car (stmt)} = \text{'loop-mg})
\land \text{normal (mg-state)}
\land (\neg \text{resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)})))
\land \text{normal (mg-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, sizes)})))
\rightarrow (\text{mg-psw (mg-meaning-r (stmt, proc-list, mg-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, sizes)), mg-state, n - 1, sizes)})
\]

THEOREM: clock-equivalence-loop-case
\[(\text{car (stmt)} = \text{'loop-mg})
\land \text{normal (mg-state)}
\land (\neg \text{resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)})))
\rightarrow (\neg \text{resource-errorp (mg-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, sizes)})))
\]

THEOREM: loop-body-exact-time-hyps
\[(\text{n} \nless 0)
\land (\neg \text{resources-inadequatep (stmt, proc-list, sizes)})
\]
\[
\begin{align*}
\text{list (length (temp-stk),} \\
\hspace{1cm} \text{p-ctrl-stk-size (ctrl-stk)))}
\end{align*}
\]
\[
\land (\text{car (stmt) = 'loop-mg)}
\land \text{ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)}
\land \text{ok-mg-def-plistp (proc-list)}
\land \text{ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2)}
\land \text{ok-mg-statep (mg-state, r-cond-list)}
\land \text{cond-subsetp (r-cond-list, t-cond-list)}
\land (\text{code (translate-def-body (assoc (subr, proc-list), proc-list)})
\hspace{1cm} = \text{append (code (translate (cinfo, t-cond-list, stmt, proc-list)), code2)})
\land \text{user-defined-proc (subr, proc-list)}
\land \text{plstp (temp-stk)}
\land \text{lstp (ctrl-stk)}
\land \text{mg-vars-list-ok-in-p-state (mg-alist (mg-state),}
\hspace{1cm} \text{bindings (top (ctrl-stk)), temp-stk)}
\land \text{no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))}
\land \text{signatures-match (mg-alist (mg-state), name-alist)}
\land \text{normal (mg-state)}
\land \text{all-cars-unique (mg-alist (mg-state))}
\land (\neg \text{resource-errorp (mg-meaning-r (stmt,}
\hspace{1cm} \text{proc-list, mg-state,}
\hspace{1cm} \text{n,}
\hspace{1cm} \text{list (length (temp-stk),}
\hspace{1cm} \text{p-ctrl-stk-size (ctrl-stk))))})
\rightarrow (\text{ok-mg-statement (loop-body (stmt),}
\hspace{1cm} \text{cons ('leave, r-cond-list),}
\hspace{1cm} \text{name-alist,}
\hspace{1cm} \text{proc-list)}
\land \text{ok-mg-def-plistp (proc-list)}
\land \text{ok-translation-parameters (make-cinfo (append (code (cinfo),}
\hspace{1cm} \text{list (cons ('dl,}
\hspace{1cm} \text{cons (label-cnt (cinfo),}
\hspace{1cm} \text{'(nil}
\hspace{1cm} \text{(no-op))))),}
\hspace{1cm} \text{cons (cons ('leave,}
\hspace{1cm} \text{1 + label-cnt (cinfo)),}
\hspace{1cm} \text{label-alist (cinfo)),}
\hspace{1cm} \text{1 + (1 + label-cnt (cinfo))),}
\hspace{1cm} t-cond-list,}
\hspace{1cm} \text{loop-body (stmt),}
\hspace{1cm} \text{proc-list),}
\hspace{1cm} 3)
\end{align*}
\]
\[
\begin{align*}
\text{cons} &\left(\text{list} \left( \text{\texttt{\textasciitilde}jump}, \text{\texttt{label-cnt}} \left( \text{\texttt{cinfo}} \right) \right)\right), \\
&\text{cons} \left( \text{\texttt{\textasciitilde}dl}, \\
&\text{cons} \left( 1 + \text{\texttt{label-cnt}} \left( \text{\texttt{cinfo}} \right), \\
&\left( \text{\texttt{nil}} \\
&\left( \text{\texttt{push-constant}} \\
&\left( \text{\texttt{nat} 2}) \right) \right) \right) \right), \\
&\text{cons} \left( \left( \text{\texttt{pop-global} \ c-c}, \\
&\text{\texttt{code2}} \right) \right) \right)
\end{align*}
\]

\[\land\text{ok-mg-statep} \left( \text{\texttt{mg-state}}, \text{cons} \left( \left( \text{\texttt{\textasciitilde}leave}}, \text{\texttt{r-cond-list}} \right) \right) \right)\]

\[\land\text{cond-subsetp} \left( \text{cons} \left( \left( \text{\texttt{\textasciitilde}leave}}, \text{\texttt{r-cond-list}} \right), \text{\texttt{t-cond-list}} \right) \right)\]

\[= \text{append} \left( \text{code} \left( \text{translate} \left( \text{\texttt{make-cinfo}} \left( \text{\texttt{append}} \left( \text{code} \left( \text{\texttt{cinfo}}, \\
&\text{list} \left( \text{cons} \left( \left( \text{\texttt{\textasciitilde}dl}}, \\
&\text{cons} \left( \text{\texttt{label-cnt}} \left( \text{\texttt{cinfo}} \right), \\
&\left( \text{\texttt{nil}} \\
&\left( \text{\texttt{no-op}} \right) \right) \right) \right) \right) \right), \\
&\text{\texttt{t-cond-list}}, \\
&\text{\texttt{loop-body}} \left( \text{\texttt{stmt}}, \\
&\text{\texttt{proc-list}} \right) \right) \right), \\
\text{cons} \left( \text{\texttt{\textasciitildejump}}, \text{\texttt{label-cnt}} \left( \text{\texttt{cinfo}} \right) \right)\right), \\
\text{cons} \left( \text{\texttt{\textasciitildedl}}, \\
\text{cons} \left( 1 + \text{\texttt{label-cnt}} \left( \text{\texttt{cinfo}} \right), \\
\left( \text{\texttt{nil}} \\
\left( \text{\texttt{push-constant}} \\
\left( \text{\texttt{nat} 2}) \right) \right) \right) \right), \\
\text{cons} \left( \left( \text{\texttt{pop-global} \ c-c}, \\
\text{\texttt{code2}} \right) \right) \right) \right)\]

\[\land\text{user-defined-procp} \left( \text{\texttt{subr}}, \text{\texttt{proc-list}} \right)\]

\[\land\text{plistp} \left( \text{\texttt{temp-stk}} \right)\]

\[\land\text{listp} \left( \text{\texttt{ctrl-stk}} \right)\]

\[\land\text{mg-vars-list-ok-in-p-state} \left( \text{\texttt{mg-alist}} \left( \text{\texttt{mg-state}} \right), \\
\text{\texttt{bindings}} \left( \text{\texttt{top}} \left( \text{\texttt{ctrl-stk}} \right) \right) \right)\]

\[\land\text{no-p-aliasing} \left( \text{\texttt{bindings}} \left( \text{\texttt{top}} \left( \text{\texttt{ctrl-stk}} \right) \right), \text{\texttt{mg-alist}} \left( \text{\texttt{mg-state}} \right) \right)\]

\[\land\text{signatures-match} \left( \text{\texttt{mg-alist}} \left( \text{\texttt{mg-state}} \right), \text{\texttt{name-alist}} \right)\]

\[\land\text{normal} \left( \text{\texttt{mg-state}} \right)\]

\[\land\text{all-cars-unique} \left( \text{\texttt{mg-alist}} \left( \text{\texttt{mg-state}} \right) \right)\]

\[\land\left( \neg \text{resource-errorp} \left( \text{\texttt{mg-meaning-r}} \left( \text{\texttt{loop-body}} \left( \text{\texttt{stmt}}, \\
\text{\texttt{proc-list}}, \\
\text{\texttt{mg-state}}, \\
\text{\texttt{n}} - 1 \right) \right) \right)\right)\]

4
list (length (temp-stk),
   p-ctrl-stk-size (ctrl-stk)))))

THEOREM: cond-list-superset-preserves-ok-mg-statement
(subset (cond-list1, cond-list2)
   ∧ ok-mg-statement (stmt, cond-list1, name-alist, proc-list))
   → ok-mg-statement (stmt, cond-list2, name-alist, proc-list)

THEOREM: adding-leave-preserves-statement-okness-begin-case
ok-mg-statement (begin-body (stmt),
   cons (‘leave, append (when-labels (stmt), cond-list)),
   name-alist,
   proc-list)
   → ok-mg-statement (begin-body (stmt),
   append (when-labels (stmt), cons (‘leave, cond-list)),
   name-alist,
   proc-list)

THEOREM: adding-leave-preserves-statement-okness
ok-mg-statement (stmt, cond-list, name-alist, proc-list)
   → ok-mg-statement (stmt, cons (‘leave, cond-list), name-alist, proc-list)

THEOREM: loop-sub1-body-exact-time-hyps
((n ≠ 0)
   ∧ (¬ resources-inadequatep (stmt, proc-list,
   list (length (temp-stk),
   p-ctrl-stk-size (ctrl-stk))))
   ∧ (car (stmt) = ‘loop-mg)
   ∧ ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
   ∧ ok-mg-def-plistp (proc-list)
   ∧ ok-translation-parameters (cinfo, t-cond-list, stmt, proc-list, code2)
   ∧ ok-mg-statep (mg-state, r-cond-list)
   ∧ cond-subsetp (r-cond-list, t-cond-list)
   ∧ (code (translate-def-body (assoc (subr, proc-list), proc-list))
      = append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
      code2))
   ∧ user-defined-procp (subr, proc-list)
   ∧ plistp (temp-stk)
   ∧ listp (ctrl-stk)
   ∧ mg-vars-list-ok-in-p-state (mg-alist (mg-state),
   bindings (top (ctrl-stk)),
   temp-stk)
   ∧ no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
   ∧ signatures-match (mg-alist (mg-state), name-alist)
   ∧
∧ normal (mg-state)
∧ all-cars-unique (mg-alist (mg-state))
∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))))
∧ normal (mg-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
→ (ok-mg-statement (stmt, cons (*leave, 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-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, list (length (temp-stk), p-ctrl-stk-size (ctrl-stk)))))
∧ cond-subsetp (cons (*leave, r-cond-list), t-cond-list)
∧ (code (translate-def-body (assoc (subr, proc-list), proc-list)) = append (code (translate (cinfo, t-cond-list, stmt, proc-list)), code2))
∧ user-defined-procp (subr, proc-list)
∧ plistp (temp-stk)
∧ listp (ctrl-stk)
∧ mg-vars-list-ok-in-p-state (mg-alist (mg-meaning-r (loop-body (stmt), proc-list, mg-state, n - 1, list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))), bindings (top (ctrl-stk)),

6
Theorem: \( \text{loop-clock-nonnormal-nonleave} \)

\[
\text{temp-stk) } \land \text{no-p-aliasing (bindings (top (ctrl-stk))),} \\
\text{mg-alist (mg-meaning-r (loop-body (stmt)),} \\
\text{proc-list,} \\
\text{mg-state,} \\
\text{n} - 1, \\
\text{list (length (temp-stk),} \\
p-ctrl-stk-size (ctrl-stk))}}) \\
\land \text{signatures-match (mg-alist (mg-meaning-r (loop-body (stmt)),} \\
\text{proc-list,} \\
\text{mg-state,} \\
\text{n} - 1, \\
\text{list (length (temp-stk),} \\
p-ctrl-stk-size (ctrl-stk))}}), \\
\text{name-alist} \\
\land \text{normal (mg-meaning-r (loop-body (stmt)),} \\
\text{proc-list,} \\
\text{mg-state,} \\
\text{n} - 1, \\
\text{list (length (temp-stk),} \\
p-ctrl-stk-size (ctrl-stk))}}) \\
\land \text{all-cars-unique (mg-alist (mg-meaning-r (loop-body (stmt)),} \\
\text{proc-list,} \\
\text{mg-state,} \\
\text{n} - 1, \\
\text{list (length (temp-stk),} \\
p-ctrl-stk-size (ctrl-stk))}}), \\
\land \text{\( \neg \) resource-errorp (mg-meaning-r (stmt,} \\
\text{proc-list,} \\
\text{mg-meaning-r (loop-body (stmt),} \\
\text{proc-list,} \\
\text{mg-state,} \\
\text{n} - 1, \\
\text{list (length (temp-stk),} \\
p-ctrl-stk-size (ctrl-stk))}}), \\
\text{n} - 1, \\
\text{list (length (temp-stk),} \\
p-ctrl-stk-size (ctrl-stk))}))
\]

\[\text{THEOREM: loop-clock-nonnormal-nonleave} \]

\[(\text{car (stmt) = 'loop-mg) } \land \text{ (n \neq 0) } \land \text{ normal (mg-state) } \land \text{ \( \neg \) resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes))}\]
\begin{align*}
&\land \ (\neg \text{normal}(\text{mg-meaning-r}(\text{loop-body}(\text{stmt}), \\
&\quad \text{proc-list}, \\
&\quad \text{mg-state}, \\
&\quad n-1, \\
&\quad \text{sizes}))) \\
&\land \ (\text{cc}(\text{mg-meaning-r}(\text{loop-body}(\text{stmt}), \text{proc-list}, \text{mg-state}, n-1, \text{sizes})) \neq \text{'}\text{leave}') \\
&\rightarrow \ (\text{clock}(\text{stmt}, \text{proc-list}, \text{mg-state}, n) \\
&\quad = \ (1 + \text{clock}(\text{loop-body}(\text{stmt}, \text{proc-list}, \text{mg-state}, n-1))))
\end{align*}

THEOREM: loop-clock-normal
((\text{car}(\text{stmt}) = \text{'}\text{loop-mg}) \\
\land \ (n \neq 0) \\
\land \ \text{normal}(\text{mg-state}) \\
\land \ (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{sizes})))) \\
\land \ (\text{normal}(\text{mg-meaning-r}(\text{loop-body}(\text{stmt}), \text{proc-list}, \text{mg-state}, n-1, \text{sizes})))) \\
\rightarrow \ (\text{clock}(\text{stmt}, \text{proc-list}, \text{mg-state}, n) \\
\quad = \ (1 + ((1 + \text{clock}(\text{loop-body}(\text{stmt}, \text{proc-list}, \text{mg-state}, n-1)) \\
\quad + \text{clock}(\text{stmt}, \\
\quad \quad \text{mg-meaning-r}(\text{loop-body}(\text{stmt}), \\
\quad \quad \quad \text{proc-list}, \\
\quad \quad \quad \text{mg-state}, \\
\quad \quad \quad n-1, \\
\quad \quad \quad \text{sizes}), \\
\quad \quad n-1)))))

THEOREM: loop-clock-nonnormal-leave
((\text{car}(\text{stmt}) = \text{'}\text{loop-mg}) \\
\land \ (n \neq 0) \\
\land \ \text{normal}(\text{mg-state}) \\
\land \ (\neg \text{resource-errorp}(\text{mg-meaning-r}(\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{sizes})))) \\
\land \ (\neg \text{normal}(\text{mg-meaning-r}(\text{loop-body}(\text{stmt}), \\
\quad \text{proc-list}, \\
\quad \text{mg-state}, \\
\quad n-1, \\
\quad \text{sizes})))) \\
\land \ (\text{cc}(\text{mg-meaning-r}(\text{loop-body}(\text{stmt}), \text{proc-list}, \text{mg-state}, n-1, \text{sizes})) = \text{'}\text{leave}') \\
\rightarrow \ (\text{clock}(\text{stmt}, \text{proc-list}, \text{mg-state}, n) \\
\quad = \ (3 + \text{clock}(\text{loop-body}(\text{stmt}, \text{proc-list}, \text{mg-state}, n-1))))

THEOREM: loop-step-initial-equals-state1
((\text{car}(\text{stmt}) = \text{'}\text{loop-mg}) \\
\land \ \text{ok-mg-statement}(\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list})

8
\[\begin{align*}
\wedge & \quad \text{ok-mg-def-plistp} \,(\text{proc-list}) \\
\wedge & \quad \text{ok-translation-parameters} \,(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \\
\wedge & \quad \left(\text{code} \,(\text{translate-def-body} \,(\text{assoc} \,(\text{subr}, \text{proc-list}), \text{proc-list}))ight) \\
& \quad = \quad \text{append} \,(\text{code} \,(\text{translate} \,(\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \text{code2})) \\
\wedge & \quad \text{user-defined-procp} \,(\text{subr}, \text{proc-list})) \\
\to & \quad \left(\text{p-step} \,(\text{map-down} \,(\text{mg-state}, \text{proc-list}, \text{ctrl-stk}, \text{temp-stk}, \text{tag} \,\left(\text{'pc}, \text{cons} \,(\text{subr}, \text{length} \,(\text{code} \,(\text{cinfo}))))), \text{t-cond-list})\right) \\
& \quad = \quad \text{map-down} \,(\text{mg-state}, \text{proc-list}, \text{ctrl-stk}, \text{temp-stk}, \text{tag} \,\left(\text{'pc}, \text{cons} \,(\text{subr}, \text{length} \,(\text{code} \,(\text{make-cinfo} \,(\text{append} \,(\text{code} \,(\text{cinfo}), \text{list} \,(\text{cons} \,\left(\text{'dl}, \text{cons} \,(\text{label-cnt} \,(\text{cinfo}), \text{'nil} \,(\text{no-op})))), \text{cons} \,(\text{cons} \,\left(\text{'leave}, \text{1 + label-cnt} \,(\text{cinfo})), \text{label-alist} \,(\text{cinfo})), \text{1 + (1 + label-cnt} \,(\text{cinfo}))))))\right), \text{t-cond-list})\right) \right) \\
& \quad \quad \text{Theorem: nonleave-nonnormal-body-meaning-preserved} \\
& \quad \quad \quad \left(\left(\text{n} \neq 0\right) \wedge \text{normal} \,(\text{mg-state}) \wedge \left(\text{car} \,(\text{stmt}) = \text{'loop-mg}\right) \wedge \neg \text{resource-errorp} \,(\text{mg-meaning-r} \,(\text{stmt}, \text{proc-list}, \text{mg-state}, \text{n}, \text{sizes})) \wedge \neg \text{normal} \,(\text{mg-meaning-r} \,(\text{loop-body} \,(\text{stmt}), \text{proc-list}, \text{mg-state}, \text{n} - 1, \text{sizes})) \wedge \text{cc} \,(\text{mg-meaning-r} \,(\text{loop-body} \,(\text{stmt}, \text{proc-list}, \text{mg-state}, \text{n} - 1, \text{sizes}))) \neq \text{'leave}) \\
& \quad \quad \quad \to \quad \text{mg-meaning-r} \,(\text{stmt}, \text{proc-list}, \text{mg-state}, \text{n}, \text{sizes}) \quad = \quad \text{mg-meaning-r} \,(\text{loop-body} \,(\text{stmt}, \text{proc-list}, \text{mg-state}, \text{n} - 1, \text{sizes})) \\
& \quad \quad \text{Theorem: loop-code-rewrite} \\
\end{align*}\]
\[
(car\ (stmt) = \text{'loop-mg})
\rightarrow (append\ (code\ (translate\ (cinfo,\ t\text{-}\cond\text{-}list,\ stmt,\ proc\text{-}list)),\ code2))
\]
\[
= append\ (code\ (translate\ (make\text{-}cinfo\ append\ (code\ (cinfo)),
list\ (cons\ ('d1,
cons\ (label\text{-}cnt\ (cinfo),
'(nil
(no\-op))))),
cons\ (cons\ ('\text{leave},
1 + label\text{-}cnt\ (cinfo)),
label\text{-}alist\ (cinfo)),
1 + (1 + label\text{-}cnt\ (cinfo))),
t\text{-}\cond\text{-}list,
loop\text{-}body\ (stmt),
proc\text{-}list)),
cons\ (list\ ('\text{jump},\ label\text{-}cnt\ (cinfo)),
cons\ (cons\ ('d1,
cons\ (1 + label\text{-}cnt\ (cinfo),
'(nil
(push\text{-}constant\ (nat\ 2))))),
cons\ ('(pop\text{-}global\ c\text{-}c),\ code2))))
\]

(prove-lemma loop\text{-}nonnormal\text{-}nonleave\text{-}state2\text{-}equals\text{-}final\ (rewrite)
(implies
(and\ (not\ (zerop\ N))
(equal\ (car\ STMT) \text{'loop-mg})
(normal\ mg\text{-}state)
(not\ (resource\text{-}errorp\ (mg\text{-}meaning\text{-}r\ stmt\ proc\text{-}list\ mg\text{-}state\ n)
(list\ (length\ temp\text{-}stk)
(p\text{-}ctrl\text{-}stk\text{-}size\ ctrl\text{-}stk))))))
(not\ (normal\ (mg\text{-}meaning\text{-}r\ (loop\text{-}body\ stmt)\ proc\text{-}list\ mg\text{-}state\ (sub1\ n)
(list\ (length\ temp\text{-}stk)\ (p\text{-}ctrl\text{-}stk\text{-}size\ ctrl\text{-}stk))))))
(not\ (equal\ (cc\ (mg\text{-}meaning\text{-}r\ (loop\text{-}body\ stmt)\ proc\text{-}list\ mg\text{-}state\ (sub1\ N)
(list\ (length\ temp\text{-}stk)\ (p\text{-}ctrl\text{-}stk\text{-}size\ ctrl\text{-}stk))))
'\text{leave}))
(equal
(p\text{-}state
(tag\ \text{\textquote{pc}}
(ccons\ subr
(if
(normal\ (mg\text{-}meaning\text{-}r\ (loop\text{-}body\ stmt)
proc\text{-}list\ mg\text{-}state
(sub1\ n))
(sub1\ n))

10
(LIST (LENGTH TEMP-STK)
   (P-CTRL-STK-SIZE CTRL-STK)))))
   (LENGTH
   (CODE
   (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (CONS 'DL
             (CONS (LABEL-CNT CINFO)
                  '(NIL (NO-OP))))))
             (CONS (CONS 'LEAVE
                     (ADD1 (LABEL-CNT CINFO)))
                     (LABEL-ALIST CINFO))
                 (ADD1 (ADD1 (LABEL-CNT CINFO))))
   T-COND-LIST
   (LOOP-BODY STMT)
   PROC-LIST)))
   (FIND-LABEL
   (FETCH-LABEL
   (CC (MG-MEANING-R (LOOP-BODY 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 (CONS 'DL
             (CONS (LABEL-CNT CINFO)
                  '(NIL (NO-OP))))))
             (CONS (CONS 'LEAVE
                     (ADD1 (LABEL-CNT CINFO)))
                     (LABEL-ALIST CINFO))
                 (ADD1 (ADD1 (LABEL-CNT CINFO))))
   T-COND-LIST
   (LOOP-BODY STMT)
   PROC-LIST)))
   (APPEND
   (CODE
   (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (CONS 'DL
             (CONS (LABEL-CNT CINFO)
                  '(NIL (NO-OP))))))
             (CONS (CONS 'LEAVE
                     (ADD1 (LABEL-CNT CINFO)))
                     (LABEL-ALIST CINFO))
                 (ADD1 (ADD1 (LABEL-CNT CINFO))))
   T-COND-LIST
   (LOOP-BODY STMT)
   PROC-LIST)))
   (APPEND
   (CODE
   (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
      (LIST (CONS 'DL
             (CONS (LABEL-CNT CINFO)
                  '(NIL (NO-OP))))))
             (CONS (CONS 'LEAVE
                     (ADD1 (LABEL-CNT CINFO)))
                     (LABEL-ALIST CINFO))
                 (ADD1 (ADD1 (LABEL-CNT CINFO))))
   T-COND-LIST
   (LOOP-BODY STMT)
   PROC-LIST)))
(CONS (CONS 'DL
(NIL (ADD1 (LABEL-CNT CINFO))
'(POP-GLOBAL C-C) CODE2))))

CTRL-STK
(MAP-DOWN-VALUES
(MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(BINDINGS (TOP CTRL-STK))
TRANSLATE-PROC-LIST PROC-LIST
(LIST
(LIST 'C-C
(MG-COND-TO-P-NAT (CC (MG-MEANING-R (LOOP-BODY 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;; final
(TAG 'PC
(CONS SUBR
(IF
(NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LIST (LENGTH CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
(FIND-LABEL
(FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LABEL-ALIST (TRANSLATE CINFO T-COND-LIST STMT
PROC-LIST))))
Theorem: loop-leave-body-meaning-preserved
((n ≠ 0)
∧ normal (mg-state)
∧ (car (stmt) = 'loop-mg)
∧ (¬ resource-errorp (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)))
∧ (cc (mg-meaning-r (loop-body (stmt), proc-list, mg-state, n − 1, sizes))
   = 'leave))
→ (mg-meaning-r (stmt, proc-list, mg-state, n, sizes)
   = set-condition (mg-meaning-r (loop-body (stmt),
   proc-list, mg-state, n − 1, sizes),
   'normal))

THEOREM: loop-find-labelp-lemma1
((car (stmt) = 'loop-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 (cons ('dl,
       cons (label-cnt (cinfo),
       'nil
       (no-op)))))),
   cons (cons ('leave,
       1 + label-cnt (cinfo)),
   label-alist (cinfo)),
   1 + (1 + label-cnt (cinfo))),
   t-cond-list,
   loop-body (stmt),
   proc-list))))

THEOREM: loop-find-labelp-lemma2
((car (stmt) = 'loop-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 (label-cnt (cinfo), code (cinfo)))

(prove-lemma loop-leave-state2-step1-effect (rewrite)
 (IMPLIES
   (AND (NOT (ZEROP N))
    (NOT (RESOURCES-INADEQUATEP STMT PROC-LIST
                              (LIST (LENGTH TEMP-STK)
                              (P-CTRL-STK-SIZE CTRL-STK))))))

14
(EQUAL (CAR STMT) 'LOOP-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))
(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)))))
 (equal (cc (mg-meaning-r (loop-body stmt) proc-list mg-state (sub1 n)
 (list (length temp-stk)
 (p-ctrl-stk-size ctrl-stk)))))
 'leave))
 (equal
 (p-step
 (P-STATE
 (TAG 'PC
 (CDNS SUBR
 (IF (NORMAL (MG-MEANING-R (LOOP-BODY 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 (CONS 'DL
 (CONS (LABEL-CNT CINFO)
 '[NIL (NO-OP))))))))
 (CONS (CONS 'LEAVE

15
(ADD1 (LABEL-CNT CINFO))
  (LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO))))
  T-COND-LIST
  (LOOP-BODY STMT)
  PROC-LIST))
(FIND-LABEL
(FETCH-LABEL
  (CC (MG-MEANING-R (LOOP-BODY 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 (CONS 'DL
          (CONS (LABEL-CNT CINFO)
            '(NIL (NO-OP)))))))
      (CONS (CONS 'LEAVE
          (ADD1 (LABEL-CNT CINFO)))))
      (LABEL-ALIST CINFO))
      (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (LOOP-BODY STMT)
      PROC-LIST))
  (APPEND
    (CODE
      (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
        (LIST (CONS 'DL
          (CONS (LABEL-CNT CINFO)
            '(NIL (NO-OP)))))))
          (CONS (CONS 'LEAVE
            (ADD1 (LABEL-CNT CINFO)))))
      (LABEL-ALIST CINFO))
      (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (LOOP-BODY STMT)
      PROC-LIST))
    (CONS (LIST 'JUMP (LABEL-CNT CINFO))
    (CONS (CONS 'DL
      (CONS (ADD1 (LABEL-CNT CINFO))
        '(NIL (PUSH-CONSTANT (NAT 2))))
          (CONS 'POP-GLOBAL C-C CODE2))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (LOOP-BODY 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 (LOOP-BODY 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 (CONS 'DL
  (CONS (LABEL-CNT CINFO)
  '(NIL (NO-OP)))))
  (CONS (CONS 'LEAVE
  (ADD1 (LABEL-CNT CINFO)))
  (LABEL-ALIST CINFO))
  (ADD1 (ADD1 (LABEL-CNT CINFO))))
  T-COND-LIST
  (LOOP-BODY STMT)
  PROC-LIST))))
  2)))))
  CTRL-STK
  (PUSH '(NAT 2)
  (MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R (LOOP-BODY 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 'LEAVE T-COND-LIST)))
  (MG-MAX-CTRL-STK-SIZE)
  (MG-MAX-TEMP-STK-SIZE)
  (MG-WORD-SIZE)
  'RUN)))

((INSTRUCTIONS PROMOTE
  (DIVE 1)
  S
  (= (CC (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (CONS (LENGTH TEMP-STK)
      (CONS (P-CTRL-STK-SIZE CTRL-STK)
        'NIL))))
    'LEAVE
    0)
  S
  (S LEMMAS)
  (DIVE 1 1 1 2 2)
  (REWRITE FIND-LABEL-APPEND)
  (DIVE 2)
  X
  (DIVE 1)
  X UP TOP
  (DIVE 1)
  S
  (S LEMMAS)
  (DIVE 1 1 2)
  (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (DIVE 1 1)
  (REWRITE LOOP-TRANSLATION-2)
  UP
  (S LEMMAS)
  UP UP
  (S LEMMAS)
  (REWRITE GET-LENGTH-PLUS)
  X UP
  (S LEMMAS)
  UP X
  (DIVE 1)
\texttt{(prove-lemma loop-leave-state2-step2-effect (rewrite) (implies (and (not (zerop n)) (not (resources-inadequatep stmt proc-list) (list (length temp-stk)))) \texttt{TOP S)})


(P-CTRL-STK-SIZE CTRL-STK)))
(EQUAL (CAR STMT) 'LOOP-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-PROC SUBR PROC-LIST)
(PLISTP TEMP-STK)
(LISTP CTRL-STK)
(MG-VARS-LIST-OK-IN-P-STATE (MG-ALIST MG-STATE))
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
(NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
(MG-ALIST MG-STATE))
(SIGNATURES-MATCH (MG-ALIST MG-STATE) NAME-ALIST)
(NORMAL MG-STATE)
(ALL-CARS-UNIQUE (MG-ALIST MG-STATE))
(NOT (RESOURCE-ERRORP (MG-MEANING-R STMT PROC-LIST MG-STATE N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(equal (cc (mg-meaning-r (loop-body stmt) proc-list mg-state (sub1 n)
(list (length temp-stk)
(p-ctrl-stk-size ctrl-stk))))
'leave))
(equal
(p-step
(P-STATE
(TAG 'PC
(CONS SUBR
(PLUS
(LENGTH
(CODE (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL
(CONS (LABEL-CNT CINFO)
'((NIL (NO-OP))))))
(CONS (CONS 'LEAVE
(ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))

20
(ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST)))
  2))
CTRL-STK
(PUSH '(NAT 2)
(MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R (LOOP-BODY 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 'LEAVE T-COND-LIST)))
(MG-MAX-CTRL-STK-SIZE)
(MG-MAX-TEMP-STK-SIZE)
(MG-WORD-SIZE)
'RUN))
  (P-STATE;; final
  (TAG 'PC
  (CONS SUBR
  (IF
  (NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N)
(LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))
  (LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
  (FIND-LABEL
  (FETCH-LABEL (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
  (LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))
  (LABEL-ALIST (TRANSLATE CINFO T-COND-LIST STMT
PROC-LIST)))))
  (APPEND (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))
(CODE2))))
CTRL-STK
(MAP-DOWN-VALUES (MG-ALIST (MG-MEANING-R STMT PROC-LIST MG-STATE N
(LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))))
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
(TRANSLATE-PROC-LIST PROC-LIST)
(LIST
(LIST 'C-C
  (MG-COND-TO-P-NAT (CC (MG-MEANING-R STMT PROC-LIST MG-STATE N
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))
  T-COND-LIST)))
  (MG-MAX-CTRL-STK-SIZE)
  (MG-MAX-TEMP-STK-SIZE)
  (MG-WORD-SIZE)
  'RUN)))

((INSTRUCTIONS PROMOTE
  (DIVE 1)
  X
  (S LEMMAS)
  (DIVE 1 1 2)
  (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (REWRITE LOOP-CODE-REWRITE)
  UP
  (REWRITE GET-LENGTH-PLUS)
  X X UP X UP X
  (DIVE 1)
  X UP S-PROP X
  (S LEMMAS)
  S UP S
  (= (MG-MEANING-R STMT PROC-LIST MG-STATE N
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))
  (SET-CONDITION (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
      (P-CTRL-STK-SIZE CTRL-STK)))))
  'NORMAL)
  0)
SPLIT
  (ENABLE MG-COND-TO-P-NAT)
  S S
  (ENABLE LENGTH-CONS)
  (S LEMMAS)
  S
  (DIVE 1)
  (REWRITE LOOP-LEAVE-BODY-MEANING-PRESERVED)
  TOP S)))

22
(prove-lemma loop-normal-body-step-state2-equals-state3 (rewrite)
  (implies
   (and (not (zerop n))
        (not (resources-inadequately-stmt proc-list
             (list (length temp-stk)
                    (p-ctrl-stk-size ctrl-stk)))))
    (equal (car stmt) 'loop-mg)
    (ok-mg-statement stmt r-cond-list name-alist proc-list)
    (ok-mg-def-plistp proc-list)
    (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))))))
  (equal
    (normal (mg-meaning-r (loop-body stmt) proc-list mg-state (sub1 n))
             (list (length temp-stk)
                    (p-ctrl-stk-size ctrl-stk))))
    (p-step
     (p-state
      (tag 'pc
           (cons subr
                  (if
                   (normal (mg-meaning-r (loop-body 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 (CONS 'DL
                                    (CONS (LABEL-CNT CINFO)
                                          '(NIL (NO-OP)))))))
                             (CONS (CONS 'LEAVE
                                     (ADD1 (LABEL-CNT CINFO)))
                                     (LABEL-ALIST CINFO))
                             (ADD1 (ADD1 (LABEL-CNT CINFO))))
                             T-COND-LIST
                             (LOOP-BODY STMT)
                             PROC-LIST)))
                             (FIND-LABEL
                             (FETCH-LABEL
                             (CC (MG-MEANING-R (LOOP-BODY 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 (CONS 'DL
                                    (CONS (LABEL-CNT CINFO)
                                          '(NIL (NO-OP)))))))
                             (CONS (CONS 'LEAVE
                                     (ADD1 (LABEL-CNT CINFO)))
                                     (LABEL-ALIST CINFO))
                             (ADD1 (ADD1 (LABEL-CNT CINFO))))
                             T-COND-LIST
                             (LOOP-BODY STMT)
                             PROC-LIST)))
                             (APPEND
                             (CODE
                             (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                             (LIST (CONS 'DL
                                    (CONS (LABEL-CNT CINFO)
                                          '(NIL (NO-OP)))))))
                             (CONS (CONS 'LEAVE
                                     (ADD1 (LABEL-CNT CINFO)))
                                     (LABEL-ALIST CINFO))
                             (ADD1 (ADD1 (LABEL-CNT CINFO))))
                             T-COND-LIST
                             (LOOP-BODY STMT)
                             PROC-LIST)))))
                             (APPEND
                             (CODE
                             (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                             (LIST (CONS 'DL
                                    (CONS (LABEL-CNT CINFO)
                                          '(NIL (NO-OP)))))))
                             (CONS (CONS 'LEAVE
                                     (ADD1 (LABEL-CNT CINFO)))
                                     (LABEL-ALIST CINFO))
                             (ADD1 (ADD1 (LABEL-CNT CINFO))))
                             T-COND-LIST
                             (LOOP-BODY STMT)
                             PROC-LIST)))))
                             (APPEND
                             (CODE
                             (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                             (LIST (CONS 'DL
                                    (CONS (LABEL-CNT CINFO)
                                          '(NIL (NO-OP)))))))
                             (CONS (CONS 'LEAVE
                                     (ADD1 (LABEL-CNT CINFO)))
                                     (LABEL-ALIST CINFO))
                             (ADD1 (ADD1 (LABEL-CNT CINFO))))
                             T-COND-LIST
                             (LOOP-BODY STMT)
                             PROC-LIST)))))
                             (APPEND
                             (CODE
                             (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
                             (LIST (CONS 'DL
                                    (CONS (LABEL-CNT CINFO)
                                          '(NIL (NO-OP)))))))
                             (CONS (CONS 'LEAVE
                                     (ADD1 (LABEL-CNT CINFO)))
                                     (LABEL-ALIST CINFO))
                             (ADD1 (ADD1 (LABEL-CNT CINFO))))
                             T-COND-LIST
                             (LOOP-BODY STMT)
                             PROC-LIST))))
PROC-LIST))
(CONS (LIST 'JUMP (LABEL-CNT CINFO))
  (CONS (CONS 'DL
    (CONS (ADD1 (LABEL-CNT CINFO))
      'NIL (PUSH-CONSTANT (NAT 2)))))
  (CONS 'POP-GLOBAL C-C) CODE2)))))))
CTRL-STK
(MAP-DOWN-VALUES
  (MG-ALIST (MG-MEANING-R (LOOP-BODY 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 (LOOP-BODY 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))
(MAP-DOWN (MG-MEANING-R (LOOP-BODY STMT));; state3
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
  (P-CTRL-STK-SIZE CTRL-STK))
PROC-LIST CTRL-STK TEMP-STK
(TAG 'PC
  (CONS SUBR (LENGTH (CODE CINFO))))
T-COND-LIST)))
((INSTRUCTIONS
  PROMOTE (DIVE 1) S X (S LEMMAS) (DIVE 1 1 2) (REWRITE TRANSLATE-DEF-BODY-REWRITE)
  (REWRITE LOOP-CODE-REWRITE) UP (REWRITE GET-LENGTH-CAR) S UP (ENABLE LABELLEDP)
  X UP X (DIVE 1) X UP S-PROP X (S LEMMAS) (DIVE 1 2 1) (REWRITE DEFINEDP-CAR-ASSOC)
  NX (DIVE 2) (REWRITE TRANSLATE-DEF-BODY-REWRITE) (REWRITE LOOP-CODE-REWRITE) (DIVE 1)
  (REWRITE NEW-CODE-APPENDED-TO-OLD1) UP UP (S LEMMAS) (REWRITE FIND-LABEL-APPEND)
  (DIVE 2) X UP (REWRITE PLUS-O-REWRITE) S TOP S (DIVE 1)
Theorem: loop-never-leave

\[ ((\text{car} (\text{stmt}) = '\text{loop-mg}) \land \text{normal} (\text{mg-state})) \rightarrow (\text{cc} (\text{mg-meaning-r} (\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{sizes})) \neq '\text{leave}) \]

Theorem: loop-normal-body-state4-equals-final

\[ ((n \neq 0) \land (\neg \text{resources-inadequatep} (\text{stmt}, \text{proc-list}, \text{list} (\text{length} (\text{temp-stk}), \text{p-ctrl-stk-size} (\text{ctrl-stk})))) \land (\text{car} (\text{stmt}) = '\text{loop-mg}) \land \text{ok-mg-statement} (\text{stmt}, \text{r-cond-list}, \text{name-alist}, \text{proc-list}) \land \text{ok-mg-def-plistp} (\text{proc-list}) \land \text{ok-translation-parameters} (\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list}, \text{code2}) \land \text{ok-mg-statep} (\text{mg-state}, \text{r-cond-list}) \land \text{cond-subsetp} (\text{r-cond-list}, \text{t-cond-list}) \land (\text{code} (\text{translate-def-body} (\text{assoc} (\text{subr}, \text{proc-list}), \text{proc-list})) = \text{append} (\text{code} (\text{translate} (\text{cinfo}, \text{t-cond-list}, \text{stmt}, \text{proc-list})), \text{code2})) \land \text{user-defined-procp} (\text{subr}, \text{proc-list}) \land \text{plistp} (\text{temp-stk}) \land \text{listp} (\text{ctrl-stk}) \land \text{mg-vars-list-ok-in-p-state} (\text{mg-alist} (\text{mg-state}), \text{bindings} (\text{top} (\text{ctrl-stk})), \text{temp-stk}) \land \text{no-p-aliasing} (\text{bindings} (\text{top} (\text{ctrl-stk})), \text{mg-alist} (\text{mg-state})) \land \text{signatures-match} (\text{mg-alist} (\text{mg-state}), \text{name-alist}) \land \text{normal} (\text{mg-state}) \land \text{all-cars-unique} (\text{mg-alist} (\text{mg-state})) \land (\neg \text{resource-errorp} (\text{mg-meaning-r} (\text{stmt}, \text{proc-list}, \text{mg-state}, n, \text{list} (\text{length} (\text{temp-stk}), \text{p-ctrl-stk-size} (\text{ctrl-stk})))))) \land \text{normal} (\text{mg-meaning-r} (\text{loop-body} (\text{stmt}), \text{proc-list}, \text{mg-state}, n - 1, \text{list} (\text{length} (\text{temp-stk}), \text{p-ctrl-stk-size} (\text{ctrl-stk})))))) \rightarrow (\text{p-state} (\text{tag} ('\text{pc}, \text{cons} (\text{subr}, \ldots)))) \]
if normal (mg-meaning-r (stmt, 
proc-list, 
mg-meaning-r (loop-body (stmt), 
proc-list, 
mg-state, 
n − 1, 
list (length (temp-stk), 
p-ctrl-stk-size (ctrl-stk))), 
n − 1, 
list (length (temp-stk), 
p-ctrl-stk-size (ctrl-stk))))
then length (code (translate (cinfo, 
t-cond-list, 
stmt, 
proc-list)))
else find-label (fetch-label (cc (mg-meaning-r (stmt, 
proc-list, 
mg-meaning-r (loop-body (stmt), 
proc-list, 
mg-state, 
n − 1, 
list (length (temp-stk), 
p-ctrl-stk-size (ctrl-stk))), 
n − 1, 
list (length (temp-stk), 
p-ctrl-stk-size (ctrl-stk)))))
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-meaning-r (loop-body (stmt), 
proc-list, 
mg-state, 
n − 1, 
list (length (temp-stk), 
p-ctrl-stk-size (ctrl-stk))))),
\[
\begin{align*}
&\quad n - 1, \\
&\quad \text{list (length (temp-stk),} \\
&\quad \quad \text{p-ctrl-stk-size (ctrl-stk)))), \\
&\quad \quad \text{bindings (top (ctrl-stk),} \\
&\quad \quad \quad temp-stk), \\
&\quad \quad \text{translate-proc-list (proc-list),} \\
&\quad \quad \text{list (list ('}\text{c-c,} \\
&\quad \quad \quad \text{mg-cond-to-p-nat (cc (mg-meaning-r (stmt,} \\
&\quad \quad \quad \quad \text{proc-list,} \\
&\quad \quad \quad \quad \text{mg-meaning-r (loop-body (stmt),} \\
&\quad \quad \quad \quad \quad \text{proc-list,} \\
&\quad \quad \quad \quad \quad \text{mg-state,} \\
&\quad \quad \quad \quad \quad \quad n - 1, \\
&\quad \quad \quad \quad \quad \quad \text{list (length (temp-stk),} \\
&\quad \quad \quad \quad \quad \quad \quad \text{p-ctrl-stk-size (ctrl-stk)))), \\
&\quad \quad \quad \quad \quad \quad n - 1, \\
&\quad \quad \quad \quad \quad \quad \text{list (length (temp-stk),} \\
&\quad \quad \quad \quad \quad \quad \quad \text{p-ctrl-stk-size (ctrl-stk)))), \\
&\quad \quad \quad t-cond-list)))}, \\
&\quad \quad \text{ MG-MAX-CTRL-STK-SIZE,} \\
&\quad \quad \text{ MG-MAX-TEMPS-STK-SIZE,} \\
&\quad \quad \text{ MG-WORD-SIZE,} \\
&\quad \quad \text{'}\text{run)} \\
&= \quad \text{p-state (tag ('pc,} \\
&\quad \quad \text{cons (subr,} \\
&\quad \quad \quad \text{if normal (mg-meaning-r (stmt,} \\
&\quad \quad \quad \quad \text{proc-list,} \\
&\quad \quad \quad \quad \text{mg-state,} \\
&\quad \quad \quad \quad \quad n, \\
&\quad \quad \quad \quad \quad \text{list (length (temp-stk),} \\
&\quad \quad \quad \quad \quad \quad \text{p-ctrl-stk-size (ctrl-stk)))),} \\
&\quad \quad \quad \text{then length (code (translate (cinfo,} \\
&\quad \quad \quad \quad \quad \quad \quad \text{t-cond-list,} \\
&\quad \quad \quad \quad \quad \quad \quad \text{stmt,} \\
&\quad \quad \quad \quad \quad \quad \quad \text{proc-list))))} \\
&\quad \quad \quad \text{else find-label (fetch-label (cc (mg-meaning-r (stmt,} \\
&\quad \quad \quad \quad \quad \quad \quad \text{proc-list,} \\
&\quad \quad \quad \quad \quad \quad \quad \text{mg-state,} \\
&\quad \quad \quad \quad \quad \quad \quad \quad n,} \\
&\quad \quad \quad \quad \quad \quad \quad \quad \text{list (length (temp-stk),} \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \text{p-ctrl-stk-size (ctrl-stk)))),} \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \text{label-alist (translate (cinfo,} \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{t-cond-list,} \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{stmt,} \\
\end{align*}
\]
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),
lst (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: loop-normal-body-exact-time-schema
((stmt-time = (1 + body-time))
∧ (p-step (initial) = state1)
∧ (p (state1, body-time) = state2)
∧ (state2 = final))
→ (p (initial, stmt-time) = final)

THEOREM: loop-normal-body-exact-time-schema
((stmt-time = (3 + body-time))
∧ (p-step (initial) = state1)
∧ (p (state1, body-time) = state2)
∧ (p (state2, 2) = final))
→ (p (initial, stmt-time) = final)

THEOREM: loop-normal-body-exact-time-schema
((stmt-time = (1 + ((1 + body-time) + loop-sub1-time))))
\(\land (p('state1', 'body-time') = 'state2)\)
\(\land (p('state3', 'loop-sub1-time') = 'state4)\)
\(\land (p-step('initial') = 'state1)\)
\(\land (p-step('state2') = 'state3)\)
\(\land ('state4 = \text{final})\)
\(\rightarrow (p('initial', 'stmt-time') = \text{final})\)

(prove-lemma loop-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) 'loop-mg)
    (ok-mg-statement stmt r-cond-list name-alist proc-list)
    (ok-mg-def-plistp proc-list)
    (ok-translation-parameters cinfo t-cond-list stmt proc-list code2)
    (ok-mg-statep mg-state r-cond-list)
    (cond-subsetp r-cond-list t-cond-list)
    (equal (code (translate-def-body (assoc subr proc-list)
                 proc-list))
           (append (code (translate cinfo t-cond-list stmt proc-list))
                  code2))
    (user-defined-procp subr proc-list)
    (plistp temp-stk)
    (listp ctrl-stk)
    (mg-vars-list-ok-in-p-state (mg-alist mg-state)
     (bindings (top ctrl-stk))
     temp-stk)
    (no-p-aliasing (bindings (top ctrl-stk))
     (mg-alist mg-state))
    (signatures-match (mg-alist mg-state)
                      name-alist)
    (normal mg-state)
    (all-cars-unique (mg-alist mg-state))
    (not (resource-errorp (mg-meaning-r stmt proc-list mg-state n)
                        (list (length temp-stk)
                               (p-ctrl-stk-size ctrl-stk))))
    (implies
     (and
      (ok-mg-statement (loop-body stmt)
       (cons 'leave r-cond-list)))
     \text{prove-lemma loop-exact-time-lemma (rewrite)})
NAME-ALIST PROC-LIST)
(OK-MG-DEF-PLISTP PROC-LIST)
(OK-TRANSLATION-PARAMETERS
 (MAKE-CINFO (APPEND (CODE CINFO))
 (LIST (CONS 'DL
   (CONS (LABEL-CNT CINFO)
     '(NIL (NO-OP))))))
 (CONS (CONS 'LEAVE
   (ADD1 (LABEL-CNT CINFO))))
 (LABEL-ALIST CINFO))
 (ADD1 (ADD1 (LABEL-CNT CINFO))))
 T-COND-LIST
 (LOOP-BODY STMT)
 PROC-LIST
 (CONS (LIST 'JUMP (LABEL-CNT CINFO))
 (CONS (CONS 'DL
 (CONS (ADD1 (LABEL-CNT CINFO))
   '(NIL (PUSH-CONSTANT (NAT 2)))))
 (CONS '(POP-GLOBAL C-C) CODE2))))
 (OK-MG-STATEP MG-STATE
 (CONS 'LEAVE R-COND-LIST))
 (COND-SUBSETP (CONS 'LEAVE 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 (CONS 'DL
 (CONS (LABEL-CNT CINFO)
   '(NIL (NO-OP))))))
 (CONS (CONS 'LEAVE
 (ADD1 (LABEL-CNT CINFO))
 (LABEL-ALIST CINFO))
 (ADD1 (ADD1 (LABEL-CNT CINFO))))
 T-COND-LIST
 (LOOP-BODY STMT)
 PROC-LIST))
 (CONS (LIST 'JUMP (LABEL-CNT CINFO))
 (CONS (CONS 'DL
 (CONS (ADD1 (LABEL-CNT CINFO))
   '(NIL (PUSH-CONSTANT (NAT 2)))))
 (CONS '(POP-GLOBAL C-C) CODE2))))
 (USER-DEFINED-PROC SUBR PROC-LIST)
(PLISTP TEMP-STK)
(LISTP CTRL-STK)
(MG-VARS-LIST-OK-IN-P-STATE (MG-ALIST MG-STATE))
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
(NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
(MG-ALIST MG-STATE))
(SIGNATURES-MATCH (MG-ALIST MG-STATE)
NAME-ALIST)
(NORMAL MG-STATE)
(ALL-CARS-UNIQUE (MG-ALIST MG-STATE))
(NOT (RESOURCE-ERRORP (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))))))
(EQUAL
(P
(MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK ;; state1
(TAG 'PC
(CONS SUBR
(LIST (CONS 'DL
(CONS (LABEL-CNT CINFO)
'(NIL (NO-OP))))))
(CONS (CONS 'LEAVE
(ADD1 (LABEL-CNT CINFO)))
(LABEL-ALIST CINFO))
(ADD1 (ADD1 (LABEL-CNT CINFO))))))))
T-COND-LIST)
(CLOCK (LOOP-BODY STMT) PROC-LIST MG-STATE (SUB1 N))) ;; body-time
(P-STATE ;; state2
(TAG 'PC
(CONS SUBR
(IF
(NORMAL (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(LIST (CONS 'DL
(CODE
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
(LIST (CONS 'DL

32
(CONS (LABEL-CNT CINFO)
  '(NIL (NO-OP)))))
(CONS (CONS 'LEAVE
  (ADD1 (LABEL-CNT CINFO)))
  (LABEL-ALIST CINFO))
  (ADD1 (ADD1 (LABEL-CNT CINFO))))
T-COND-LIST
(LOOP-BODY STMT)
PROC-LIST))
  (FIND-LABEL
  (FETCH-LABEL
    (CC (MG-MEANING-R (LOOP-BODY 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 (CONS 'DL
        (CONS (LABEL-CNT CINFO)
        '(NIL (NO-OP)))))
        (CONS (CONS 'LEAVE
        (ADD1 (LABEL-CNT CINFO)))
        (LABEL-ALIST CINFO))
        (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (LOOP-BODY STMT)
      PROC-LIST))
    (APPEND
      (CODE
        (TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO)
          (LIST (CONS 'DL
          (CONS (LABEL-CNT CINFO)
          '(NIL (NO-OP)))))
          (CONS (CONS 'LEAVE
          (ADD1 (LABEL-CNT CINFO)))
          (LABEL-ALIST CINFO))
          (ADD1 (ADD1 (LABEL-CNT CINFO))))
      T-COND-LIST
      (LOOP-BODY STMT)
      PROC-LIST))
    (CONS (LIST 'JUMP (LABEL-CNT CINFO))
    (CONS (CONS 'DL
    (CONS (ADD1 (LABEL-CNT CINFO))
    '33
'(NIL (PUSH-CONSTANT (NAT 2))))
(CONS '(POP-GLOBAL C-C) CODE2)))))

CTRL-STK
(MAP-DOWN-VALUES
(MG-ALIST (MG-MEANING-R (LOOP-BODY 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 (LOOP-BODY 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 STMT
(CONS 'LEAVE 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-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
(CONS 'LEAVE R-COND-LIST))
(COND-SUBSETP (CONS 'LEAVE 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)
(PLISTP CTRL-STK)
(MG-VARS-LIST-OK-IN-P-STATE
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
   PROC-LIST MG-STATE
   (SUB1 N)
   (LIST (LENGTH TEMP-STK)
   (P-CTRL-STK-SIZE CTRL-STK))))
 (BINDINGS (TOP CTRL-STK))
 TEMP-STK)
 (NO-P-ALIASING (BINDINGS (TOP CTRL-STK))
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 (SIGNATURES-MATCH
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 NAME-ALIST)
 (NORMAL (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 (ALL-CARS-UNIQUE
 (MG-ALIST (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 (NOT
 (RESOURCE-ERRORP
 (MG-MEANING-R STMT PROC-LIST
 (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N))
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)

35
(P-CTRL-STK-SIZE CTRL-STK)))))

(EQUAL
 (P (MAP-DOWN (MG-MEANING-R (LOOP-BODY STMT))
 PROC-LIST MG-STATE
 (SUB1 N))
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK)))
 PROC-LIST CTRL-STK TEMP-STK
 (TAG 'PC
 (CONS SUBR (LENGTH (CODE CINFO))))

(T-COND-LIST)
 (CLOCK STMT PROC-LIST)
 (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N))
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK)))
 (SUB1 N))
 (P-STATE ;; state4
 (TAG 'PC
 (CONS SUBR
 (IF
 (NORMAL (MG-MEANING-R STMT PROC-LIST
 (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N))
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK)))
 (SUB1 N))
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 (LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST)))
 (FIND-LABEL
 (FETCH-LABEL
 (CC (MG-MEANING-R STMT PROC-LIST
 (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N))
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK)))
 (SUB1 N))
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK)))
 (LABEL-ALIST (TRANSLATE 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-MEANING-R (LOOP-BODY STMT)
  PROC-LIST MG-STATE
  (SUB1 N)
  (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 N)
  (LIST (LENGTH TEMP-STK)
    (P-CTRL-STK-SIZE CTRL-STK))))
(BINDINGS (TOP CTRL-STK))
TEMP-STK)
  (TRANSLATE-PROC-LIST PROC-LIST)
  (LIST
  (LIST 'C-C
    (MG-COND-TO-P-NAT
      (CC (MG-MEANING-R STMT PROC-LIST
        (MG-MEANING-R (LOOP-BODY STMT)
        PROC-LIST MG-STATE
        (SUB1 N)
        (LIST (LENGTH TEMP-STK)
          (P-CTRL-STK-SIZE CTRL-STK)))
        (SUB1 N)
        (LIST (LENGTH TEMP-STK)
          (P-CTRL-STK-SIZE CTRL-STK))))
        (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))
     ;; stmt-time
(P-STATE;; final
(TAG 'PC
  (CONS SUBR
      (IF
        (NORMAL (MG-MEANING-R STMT PROC-LIST MG-STATE N

37
(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))))
            'RUN)))))
(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 @STATE1
 (MAP-DOWN MG-STATE PROC-LIST CTRL-STK TEMP-STK
 (TAG 'PC
 (CONS SUBR
 (LENGTH (CODE (MAKE-CINFO (APPEND (CODE CINFO)
 (LIST (CONS 'DL
 (CONS (LABEL-CNT CINFO)
 '(NIL (NO-OP))))))
 (CONS (CONS 'LEAVE
 (ADD1 (LABEL-CNT CINFO)))
 (LABEL-ALIST CINFO))
 (ADD1 (ADD1 (LABEL-CNT CINFO))))))
 T-COND-LIST))
 (ADD-ABBREVIATION @BODY-TIME
 (CLOCK (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N)))
39
(ADD-ABBREVIATION @STATE2
  (P-STATE
    (TAG 'PC
    (CONS SUBR
      (IF
        (NORMAL (MG-MEANING-R (LOOP-BODY 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 (CONS 'DL
                (CONS (LABEL-CNT CINFO)
                  '(NIL (NO-OP))))))))
            (CONS (CONS 'LEAVE
              (ADD1 (LABEL-CNT CINFO))))
          (LABEL-ALIST CINFO))
          (ADD1 (ADD1 (LABEL-CNT CINFO)))
          T-COND-LIST
          (LOOP-BODY STMT)
          PROC-LIST)))))
  (FIND-LABEL
    (FETCH-LABEL
      (CC (MG-MEANING-R (LOOP-BODY 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 (CONS 'DL
            (CONS (LABEL-CNT CINFO)
              '(NIL (NO-OP)))))))
          (CONS (CONS 'LEAVE
            (ADD1 (LABEL-CNT CINFO))))
        (LABEL-ALIST CINFO))
        (ADD1 (ADD1 (LABEL-CNT CINFO)))
        T-COND-LIST
        (LOOP-BODY STMT)
        PROC-LIST)))))
  (APPEND
    (CODE
      40)
(TRANSLATE (MAKE-CINFO (APPEND (CODE CINFO))
 (LIST (CONS 'DL
   (CONS (LABEL-CNT CINFO)
     '(NIL (NO-OP)))))
 (CONS (CONS 'LEAVE
   (ADD1 (LABEL-CNT CINFO))))
 (LABEL-ALIST CINFO))
 (ADD1 (ADD1 (LABEL-CNT CINFO))))
 T-COND-LIST
 (LOOP-BODY STMT)
 PROC-LIST))
 (CONS (LIST 'JUMP (LABEL-CNT CINFO))
 (CONS (CONS 'DL
 (CONS (ADD1 (LABEL-CNT CINFO))
   '(NIL (PUSH-CONSTANT (NAT 2))))
   (CONS '(POP-GLOBAL C-C) CODE2))))))))
 CTRL-STK
 (MAP-DOWN-VALUES
 (MG-ALIST (MG-MEANING-R (LOOP-BODY 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 (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 (MG-MAX-CTRL-STK-SIZE)
 (MG-MAX-TEMP-STK-SIZE)
 (MG-WORD-SIZE)
 'RUN))
 (ADD-ABBREVIATION @STATE3
 (MAP-DOWN (MG-MEANING-R (LOOP-BODY STMT)
 PROC-LIST MG-STATE
 (SUB1 N)
 (LIST (LENGTH TEMP-STK)
 (P-CTRL-STK-SIZE CTRL-STK))))
 41
PROC-LIST CTRL-STK TEMP-STK
(TAG 'PC
  (CONS SUBR (LENGTH (CODE CINFO))))
T-COND-LIST))
(ADD-ABBREVIATION @LOOP-SUB1-TIME
  (CLOCK STMT PROC-LIST
   (MG-MEANING-R (LOOP-BODY STMT)
    PROC-LIST MG-STATE
    (SUB1 N)
    (LIST (LENGTH TEMP-STK)
     (P-CTRL-STK-SIZE CTRL-STK))))
  (SUB1 N))))
(ADD-ABBREVIATION @STATE4
  (P-STATE
   (TAG 'PC
    (CONS SUBR
     (IF
      (NORMAL (MG-MEANING-R STMT PROC-LIST
       (MG-MEANING-R (LOOP-BODY STMT)
        PROC-LIST MG-STATE
        (SUB1 N)
        (LIST (LENGTH TEMP-STK)
         (P-CTRL-STK-SIZE CTRL-STK))))
       (SUB1 N)
       (LIST (LENGTH TEMP-STK)
        (P-CTRL-STK-SIZE CTRL-STK))))
       (LENGTH (CODE (TRANSLATE CINFO T-COND-LIST STMT PROC-LIST))))
     (FIND-LABEL
      (FETCH-LABEL
       (CC (MG-MEANING-R STMT PROC-LIST
        (MG-MEANING-R (LOOP-BODY STMT)
        PROC-LIST MG-STATE
        (SUB1 N)
        (LIST (LENGTH TEMP-STK)
         (P-CTRL-STK-SIZE CTRL-STK))))
        (SUB1 N)
        (LIST (LENGTH TEMP-STK)
         (P-CTRL-STK-SIZE CTRL-STK))))
       (LABEL-ALIST (TRANSLATE 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
   (LENGTH (CODE CINFO))))))
T-COND-LIST))

(MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(BINDINGS (TOP CTRL-STK)
TEMP-STK)
(TRANSLATE-PROC-LIST PROC-LIST)
(LIST
(LIST 'C-C
(MG-COND-TO-P-NAT
(CC (MG-MEANING-R STMT PROC-LIST
(MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK)))
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
T-COND-LIST)))
(MG-MAX-CTRL-STK-SIZE)
(MG-MAX-TEMP-STK-SIZE)
(MG-WORD-SIZE)
'RUN))
PROMOTE
(DEMOTE 19)
(DIVE 1 1)
PUSH TOP PROMOTE
(CLAIM (EQUAL (P-STEP @INITIAL) @STATE1)
0)
(CLAIM (NOT (NORMAL (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
(P-CTRL-STK-SIZE CTRL-STK))))
0)
(DROP 19)
(CLAIM (NOT (EQUAL (CC (MG-MEANING-R (LOOP-BODY STMT)
PROC-LIST MG-STATE
(SUB1 N)
(LIST (LENGTH TEMP-STK)
   (P-CTRL-STK-SIZE CTRL-STK))

'LEAVE))

0)
(CLAIM (EQUAL @STATE2 @FINAL) 0)
(CLAIM (EQUAL @STMT-TIME (ADD1 @BODY-TIME)) 0)
(DEMOTE 19 20 23 24)
DROP
(GENERALIZE ((@STATE4 STATE4)
   (@LOOP-SUB1-TIME LOOP-SUB1-TIME)
   (@STATE3 STATE3)
   (@STATE2 STATE2)
   (@BODY-TIME BODY-TIME)
   (@STATE1 STATE1)
   (@FINAL FINAL)
   (@STMT-TIME STMT-TIME)
   (@INITIAL INITIAL)))

(USE-LEMMA LOOP-NONNORMAL-NONLEAVE-EXACT-TIME-SCHEMA)

DEMOTE
(S-PROP AND OR NOT IMPLIES FIX ZEROP IFF NLISTP)
(CONTRADICT 24)
(DROP 19 23 24)
(DIVE 1)
(REWRITE LOOP-CLOCK-NONNORMAL-NONLEAVE)
TOP S
(CONTRADICT 23)
(DROP 19 20 23)
(DIVE 1)
(REWRITE LOOP-NONNORMAL-NONLEAVE-STATE2-EQUALS-FINAL)
TOP S-PROP
(CLAIM (EQUAL (P @STATE2 2) @FINAL) 0)
(CLAIM (EQUAL @STMT-TIME (PLUS 3 @BODY-TIME)) 0)
(DEMOTE 19 20 23 24)
DROP
(GENERALIZE ((@STATE4 STATE4)
   (@LOOP-SUB1-TIME LOOP-SUB1-TIME)
   (@STATE3 STATE3)
   (@STATE2 STATE2)
   (@BODY-TIME BODY-TIME)
   (@STATE1 STATE1)
   (@FINAL FINAL)
   (@STMT-TIME STMT-TIME)
   (P @STATE2 2))

44
((@INITIAL INITIAL)))

DROP
(USE-LEMMA LOOP-LEAVE-BODY-EXACT-TIME-SCHEMA)
PROVE
(CONTRADICT 24)
(DIVE 1)
(REWRITE LOOP-CLOCK-NONNORMAL-LEAVE)
TOP S-PROP
(CONTRADICT 23)
(DIVE 1)
(REWRITE P-ADD1-3)
(REWRITE P-ADD1-3)
(REWRITE P-O-UNWINDING-LEMMA)
(DIVE 1)
(REWRITE LOOP-LEAVE-STATE2-STEP1-EFFECT)
UP
(REWRITE LOOP-LEAVE-STATE2-STEP2-EFFECT)
TOP S-PROP
(DEMOTE 19)
(DIVE 1 1)
PUSH TOP PROMOTE
(CLAIM (EQUAL @STMT-TIME
          (ADD1 (PLUS (ADD1 @BODY-TIME)
                   @LOOP-SUB1-TIME)))
       0)
(CLAIM (EQUAL (P-STEP @STATE2) @STATE3)
       0)
(CLAIM (EQUAL @STATE4 @FINAL) 0)
(DEMOTE 19 20 22 23 24 25)
DROP
(GENERALIZE ((@STATE4 STATE4)
             (@LOOP-SUB1-TIME LOOP-SUB1-TIME)
             (@STATE3 STATE3)
             (@STATE2 STATE2)
             (@BODY-TIME BODY-TIME)
             (@STATE1 STATE1)
             (@FINAL FINAL)
             (@STMT-TIME STMT-TIME)
             (@INITIAL INITIAL)))

DROP
(USE-LEMMA LOOP-NORMAL-BODY-EXACT-TIME-SCHEMA)
PROVE
(CONTRADICT 25)
(DROP 19 20 22 23 24 25)

(DIVE 1)
(REWRITE LOOP-NORMAL-BODY-STATE4-EQUALS-FINAL)
TOP S-PROP
(CONTRADICT 24)
(DIVE 1)
(REWRITE LOOP-NORMAL-BODY-STEP-STATE2-EQUALS-STATE3)
TOP S-PROP
(CONTRADICT 23)
(DIVE 1)
(REWRITE LOOP-CLOCK-NORMAL
   ((($SIZES (LIST (LENGTH TEMP-STK)
   (P-CTRL-STK-SIZE CTRL-STK)))))
UP S
(USE-LEMMA LOOP-SUB1-BODY-EXACT-TIME-HYPS)
SPLIT
(CONTRADICT 21)
(DIVE 1)
(REWRITE LOOP-STEP-INITIAL-EQUALS-STATE1)
TOP S-PROP
(DROP 19)
(USE-LEMMA LOOP-BODY-EXACT-TIME-HYPS)
DEMOTE
(S-PROP AND OR NOT IMPLIES FIX ZERO P IFF NLISTP)))

EVENT: Make the library "c-loop".
Index

add-code, 1
adding-leave-preserves-statement
- okness, 5
- okness-begin-case, 5
all-cars-unique, 3, 4, 6, 7, 26
begin-body, 5
bindings, 3–7, 26, 28, 29
cc, 8, 9, 14, 26–29
clock, 8
clock-equivalence-loop-case, 2
code, 1, 3–6, 9, 10, 14, 26–29
cond-list-superset-preserves-ok
- mg-statement, 5
cond-subsetp, 3–6, 26
discard-label, 1
fetch-label, 27, 29
find-label, 27, 29
find-labelp, 14
label-alist, 1, 3, 4, 9, 10, 14, 27, 29
label-cnt, 1, 3, 4, 9, 10, 14
length, 3, 5–7, 9, 26–29
loop-body, 1–4, 6–10, 14, 26–28
loop-body-doesnt-halt, 2
loop-body-exact-time-hyps, 2
loop-clock-nonnormal-leave, 8
loop-clock-nonnormal-nonleave, 7
loop-clock-normal, 8
loop-code-rewrite, 10
loop-find-labelp-lemma1, 14
loop-find-labelp-lemma2, 14
loop-leave-body-exact-time-schema, 29
loop-leave-body-meaning-preserved, 13
loop-meaning-r-2, 1
loop-never-leave, 26
loop-nonnormal-nonleave-exact-time-schema, 29
loop-normal-body-exact-time-schema, 29
loop-normal-body-state4-equals-final, 26
loop-step-initial-equals-state1, 8
loop-sub1-body-doesnt-halt, 2
loop-sub1-body-exact-time-hyps, 5
loop-translation-2, 1
make-cinfo, 1, 3, 4, 9, 10, 14
map-down, 9
map-down-values, 28, 29
mg-alist, 3–7, 26, 28, 29
mg-cond-to-p-nat, 28, 29
mg-max-ctrl-stk-size, 28, 29
mg-max-temp-stk-size, 28, 29
mg-meaning-r, 1–3, 5–9, 14, 26–29
mg-psw, 2
mg-vars-list-ok-in-p-state, 3–5, 7, 26
mg-word-size, 28, 29
no-p-aliasing, 3–5, 7, 26
nonleave-nonnormal-body-meaning-preserved, 9
normal, 1–4, 6–9, 14, 26–28
ok-mg-def-plistp, 3, 5, 6, 9, 26
ok-mg-statement, 3, 5, 6, 8, 14, 26
ok-mg-statep, 3–6, 26
ok-translation-parameters, 3–6, 9, 14, 26
p, 29, 30
p-ctrl-stk-size, 3, 5–7, 26–29
p-state, 28, 29
p-step, 9, 29, 30
plistp, 3–6, 26
remove-label, 2
resources-inadequatep, 1, 3, 5, 26
set-condition, 14
signal-system-error, 1
signatures-match, 3–5, 7, 26
subset, 5

tag, 9, 27, 29
top, 3–7, 26, 28, 29
translate, 1, 3–6, 9, 10, 14, 26–29
translate-def-body, 3–6, 9, 26
translate-proc-list, 28, 29

user-defined-procp, 3–6, 9, 26

when-labels, 5