<ΛFFIRM>HANDCODED..12 5-Oct-81 16:49:58

(FILECREATED "25-Sep-81 19:34:58" <AFFIRM>HANDCODED..12 6183

previous date: "18-Nov-80 18:0[illegible]" <AFFIRM>HANDCODED..10)

(PRETTYCOMPRINT HANDCODEDCOMS)

(RPAQQ **HANDCODEDCOMS** [(FNS ALL\Boolean Equal\BUILTIN\Interface Equal\Integer IH\Boolean IH\Induction
IfThenElse\BUILTIN\Interface NE\BUILTIN\Interface Prop\Induction
SOME\Boolean)
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
(ADDVARS (NLAMA)
(NLAML SOME\Boolean ALL\Boolean)
(LAMA])

(DEFINEQ

1

```
(ALL\Boolean
  [NLAMBDA (var%  ex% )                                    (* R.Erickson "22-Sep-80 15:30")

          (* * We have to be an NLAMBDA and rebind Assumed/Denied when evaluating ex, since otherwise var might conflict with
          A/D)


    var% ←(EVAL var% )
    (if (LISTP var% )
        then                                               (* user slipped in a nilary constant.)
             (AffirmError <"Not a variable:" (Shorten var% :Operator)
                          >))
    ex% ←(PROG (Assumed Denied)
               (RETURN (EVAL ex% )))
    (if ex% :Operator=QOP
        then (if (FASSOC var%  ex% :given) or (FASSOC var%  ex% :find) or var%  ~MEMB ex% :free
                 then ex%
                 else (create Qexpression
                              given ←(<<var% > ! ex% :given>)
                              find ←(for v in ex% :find collect (AddArg var%  v))
                              free ←(REMOVE var%  ex% :free)
                              expr ← ex% :expr))
      else (create Qexpression
                   given ←(<<var% >>)
                   find ← NIL
                   free ←(REMOVE var%  (Frees ex% ))
                   expr ← ex% ])
```

2

```
(Equal\BUILTIN\Interface
  [LAMBDA (a1 a2 TooManyArguments)                         (* D.Musser "24-Jun-79 12:22")
    (if a1:1='ExpressionWithType and a2:1='ExpressionWithType and TooManyArguments=NIL
        and (EQUAL a1:3 a2:3) and (Report Equal\BUILTIN\Interface 1 interface)
      then (ExpressionWithType <(GetEqualOp a1:3)
                                  a1:2 a2:2>
                                Boolean)
      elseif NIL])
```

3

```
(Equal\Integer
  [LAMBDA (i j)                                            (* D.Musser "30-Aug-79 14:37")
    (if (EQUAL i j) and (Report Equal\Integer 1 axiom)
        then TRUE
      elseif (FIXP i) and (FIXP j)
        then FALSE
      elseif <'Equal\Integer i j>])
```

4

```
(IH\Boolean
  [LAMBDA (val target)                                     (* R.Erickson "30-Oct-80 16:40")

          (* * created by IH1OP. appears in propositions. Handcoded, takes the place of a dummy Boolean rule)


    (if (Report IH\Boolean 1 defn.) and ((FIXP target) or (LITATOM target) and ~(Extension target))
        then                                               (* test is to make sure we have valid nodeid, not just a
```

```
                                                    formal parameter from LHS (which happens when print type
                                                    Boolean))
        (PROG (node var)
              (node←(GetNode target))
              (if node and (L-CASE node:trans:command)='employ
                    and (Descendant? node:prop# CurrentNode:prop#)
                  then var←node:trans:parameters:1:Arg1
                       (RETURN (ComputeInductionExpression (ActualExprAt target)
                                                           var val))
                else (AffirmError "IH can't be invoed, it's undefined.")))
    elseif <IH2OP val target>])
```

5

```
(IH\Induction
  [LAMBDA (val)                                          (* R.Erickson "17-Jun-80 18:01")

          (* * see IH\Builtin this form occurs only in schemas.)


    (if ExpandInductors
        then <IH2OP val IndTarget>
      else <IH1OP val>])
```

6

```
(IfThenElse\BUILTIN\Interface
  [LAMBDA (b a1 a2 TooManyArguments)                     (* D.Musser " 7-Nov-79 12:44")
    (if b:1='ExpressionWithType and b:3=Boolean and a1:1='ExpressionWithType
          and a2:1='ExpressionWithType and TooManyArguments=NIL and (EQUAL a1:3 a2:3)
          and (Report IfThenElse\BUILTIN\Interface 1 interface)
        then (ExpressionWithType <'IfThenElse b:2 (if a2:3='Boolean
                                                      then <'IfThenElse a1:2 TRUE FALSE>
                                                    else a1:2)
                                   (if a2:3='Boolean
                                       then <'IfThenElse a2:2 TRUE FALSE>
                                     else a2:2)
                                   >
                                  a2:3)
      elseif NIL])
```

7

```
(NE\BUILTIN\Interface
  [LAMBDA (a1 a2 TooManyArguments)                       (* D.Musser "17-Aug-79 16:21")
    (if a1:1='ExpressionWithType and a2:1='ExpressionWithType and TooManyArguments=NIL
          and (EQUAL a1:3 a2:3) and (Report NE\BUILTIN\Interface 1 interface)
        then (ExpressionWithType <NOTOP <(GetEqualOp a1:3)
                                         a1:2 a2:2>>
                                 Boolean)
      elseif NIL])
```

8

```
(Prop\Induction
  [LAMBDA (val)                                          (* R.Erickson "11-Jun-80 21:11")
    (if ExpandInductors
        then (ComputeInductionExpression CurrentPropn IndVar val)
      else <PROPOP val>])
```

9

```
(SOME\Boolean
  [NLAMBDA (var% ex% )                                   (* R.Erickson "22-Sep-80 15:32")

          (* * We rebind Assumed/Denied when evaluating ex. See All\Boolean, Qexpression.)


    var% ←(EVAL var% )
    (if (LISTP var% )
        then                                             (* user slipped in a nilary constant.)
             (AffirmError <"Not a variable:" (Shorten var% :Operator)
                          >))
    ex% ←(PROG (Assumed Denied)
               (RETURN (EVAL ex% )))
    (if ex% :Operator=QOP
```

```
        then (if (FASSOC var% ex% :given) or (FASSOC var% ex% :find) or var% ~MEMB ex% :free
                 then ex%
                 else (create Qexpression
                              find ←(<<var% > ! ex% :find>)
                              given ←(for v in ex% :given collect (AddArg var% v))
                              free ←(REMOVE var% ex% :free)
                              expr ← ex% :expr))
      else (create Qexpression
                   find ←(<<var% >>)
                   given ← NIL
                   free ←(REMOVE var% (Frees ex% ))
                   expr ← ex% ])
)
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML SOME\Boolean ALL\Boolean)

(ADDTOVAR LAMA )
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (524 5995 (ALL\Boolean 536 . 1610) (Equal\BUILTIN\Interface 1614 . 2029) (Equal\Integer
2033 . 2329) (IH\Boolean 2333 . 3332) (IH\Induction 3336 . 3619) (IfThenElse\BUILTIN\Interface 3623 .
4296) (NE\BUILTIN\Interface 4300 . 4717) (Prop\Induction 4721 . 4962) (SOME\Boolean 4966 . 5992)))))
STOP
```