LAMBDATRAN

By: Ron Kaplan

The purpose of this package is to facilitate defining new LAMBDA words in such a way that a variety of other system packages will respond to them appropriately. A LAMBDA word is a word that can appear as the CAR of a function definition, like LAMBDA and NLAMBDA. New LAMBDA words are useful because they enable the user to define his or her own conventions about such things as the interpretation of arguments, and to build in certain defaults about how values are returned. For example, the DECL package defines DLAMBDA as a new LAMBDA word with unconventional arguments such as the following:

```
(DLAMBDA ((A FLOATP) (B FIXP) (RETURNS SMALLP))

(FOO A B))
```

In order for such an expression to be executable and compilable, a mechanism must be provided for translating this expression to an ordinary LAMBDA or NLAMBDA, with the special behavior associated with the arguments built into the function body. The LambdaTran package accomplishes this via an appropriate entry on DWIMUSERFORMS that computes the translation.

Besides executing and compiling, Interlisp applies a number of other operations to function definitions (e.g., breaking, advising), many of which depend on the system being able to determine certain properties of the function, such as the names of its arguments, their number, and the type of the function (EXPR, FEXPR, etc.). The LambdaTran package also provides new definitions for the functions FNTYP, ARGLST, NARGS, and ARGTYPE which can be told how to compute properties for the user's LAMBDA-words.

A new LAMBDA-word is defined in the following way:

1. Add the LAMBDA-word itself (e.g., the atom DLAMBDA) to the list LAMBDASPLST. This suppresses attempts to correct the spelling of the LAMBDA-word.

2. Add an entry for the LAMBDA-word to the association list LAMBDATRANFNS, which is a list of elements of the form: (*LAMBDA-WORD TRANFN FNTYP ARGLIST*), where (*LAMBDA-WORD* is the name of the LAMBDA-word (e.g., DLAMBDA).

TRANFN is a function of one argument that will be called whenever a real definition is needed for the LAMBDA-word definition. Its argument is the LAMBDA-word definition, and its value should be a conventional LAMBDA or NLAMBDA expression which will become the translation of the Lisp

LAMBDA-word form. The free variable FAULTFN is bound to the name of the function in which the LAMBDA-word form appeared (or TYPE-IN if the form was typed in).

FNTYP determines the function type of a definition beginning with LAMBDA-WORD. It is consulted if the definition does not already have a translation from which the function type may be deduced. If *FNTYP* is one of the atoms EXPR, FEXPR, EXPR*, or FEXPR*, then all definitions beginning with LAMBDA-word are assumed to have that type. Otherwise, *FNTYP* is a function of one argument that will be applied to the LAMBDA-word definition. Its value should be one of the above four function types.

ARGLIST determines the argument list of the definition if it has not already been translated (if it has, the *ARGLIST* is simply the *ARGLIST* of the translation). It is also a function of one argument, the LAMBDA-word definition, and its value should be the list of arguments for the function (e.g., (A B) in the DLAMBDA example above). If the LAMBDA-word definition is ill formed and the argument list cannot be computed, the function should return T. If an *ARGLIST* entry is not provided in the LAMBDATRANFNS element, then the argument list defaults to the second element of the definition.

As an example, the LAMBDATRANFNS entry for DLAMBDA is (DLAMBDA DECL EXPR DLAMARGLIST), where DECL and DLAMARGLIST are functions of one argument.

Note: if the LAMBDA-word definition has an argument list with argument names appearing either as literal atoms or as the first element of a list, the user should also put the property INFO with value BINDS on the property list of the LAMBDA-word in order to inform DWIMIFY to take notice of the names of the arguments when DWIMIFYing.