

# *Leibniz System*

## *Development Tool for Logic-based Intelligent Systems*

## Chapter 2

---

### Getting Started

#### Installation

This section explains the system installation. If the system has already been installed, the user should ignore the discussion of this section and instead should proceed directly to the subsection *The First Time*.

Create a new, empty directory called *Leibniz*.

Copy the file *leibniz.zip* containing the entire system into the *Leibniz* directory. Unzip the file. At this time, the directory *Leibniz* contains the file *leibniz.zip* plus one file *makefile.install*, and it has five subdirectories: *Code*, *Makedata*, *Manual*, *Lsqcc*, and *Cutcc*. In turn, each of *Lsqcc* and *Cutcc* has two subdirectories *Code* and *Makedata*. The subdirectories contain the following files.

*Code*: *lbcc.code.zip* file of C source code of the *lbcc* compiler and of the related execution routines.

*Makedata*: *lbcc.makedata.zip* file of C source code and logic formulation for small example expert systems.

*Manual*: *lbcc.manual.zip* file of the manual in *.ps* and *.pdf* format. These files have been generated from Plain  $\text{T}_{\text{E}}\text{X}$  files.

---

Copyright 2006 by *Leibniz* Company, Plano, Texas

Subdirectory Code of Lsqcc: `lsqcc.code.zip` file of C source code of programs `lsqcc`, `optcc`, `pyrcc`, and `tstcc`.

Subdirectory Makedata of Lsqcc: `lsqcc.makedata.zip` file of data files of small example cases for program `lsqcc`.

Subdirectory Code of Cutcc: `cutcc.code.zip` file of C source code of program `cutcc`.

Subdirectory Makedata of Cutcc: `cutcc.makedata.zip` file of data files of small example cases for program `cutcc`.

All C source code follows the ANSI standard and thus can be compiled by virtually any C/C++ compiler.

## Unix/Linux Installation

It is assumed that the `gcc` compiler is available. The compilation and linking steps are accomplished by executing the single command `make -f makefile.install` in directory `Leibniz`. The following programs are created.

Code: `lbcc` compiler and object files for subsequent linking with user programs. The use of this code is discussed in Chapters 3 and 4.

Subdirectory Code of Lsqcc: programs `lsqcc`, `optcc`, `pyrcc`, and `tstcc`. The use of these programs is described in Chapter 6.

Subdirectory Code of Cutcc: program `cutcc`. The use of this program is described in Chapter 7.

## All Other Installations

For all non-Unix/Linux installations, the following steps are required.

First, go into each one of the lowest subdirectories of `Leibniz` and unzip the `.zip` file found there.

Second, in the Code subdirectories of Leibniz, Lsqcc, and Cutcc, use a suitable C/C++ compiler and carry out compilation and linking steps described in the `makefile.install` file of that subdirectory.

**CAUTION:** Compilation must be done first in subdirectory Code of Leibniz, then in subdirectory Code of Lsqcc, and finally in subdirectory Code of Cutcc.

MS Windows: Before compilation in subdirectory Code of Lsqcc, the file `featureswin.h` must be copied to the file `featuresunix.h`.

## The First Time

Here is an easy-to-follow recipe to get going.

**CAUTION:** The discussion in the remainder of this chapter and in subsequent chapters applies to Unix/Linux installations. For all other installations, for example, MS Windows or Macintosh, a suitable C/C++ compiler must be available, and one must carry out the compilation and linking steps described in the Unix/Linux `makefile` of the subdirectory. MS Windows: The “/” of paths in parameter files `leibnizparams.dat`, `lsqccparams.dat`, and `cutccparams.dat` must be replaced by “\\” since these paths are read by C code and since “\” is the escape character of C.

## lbcc Compiler

First, create a user directory having any name. We assume below that the name is Logic. Copy into Logic all files of Leibniz/Makedata.

Second, go into directory Logic. All steps below are done in that directory. Use any text editor and access the file `leibnizparams.machine`. Change the line

```
input directory = /home/Logic/
```

so that the complete path ending in directory Logic is specified. Access the file `makefile.machine`. Change the line

```
LEIBNIZ = /home/Leibniz/Code/
```

so that the complete path ending in `Leibniz/Code` is specified.

Third, type `make -f makefile.machine` and press return. The makefile copies the file `leibnizparams.machine` into the file `leibnizparams.dat`, then invokes the Leibniz compiler `lbcc` for compilation of the logic formulation of file `machine.log`. The output consists of a program file `machine.prg`, a transfer process file `leibniztrans.c`, and three 'include' files `machine.trs`, `leibnizdefs.h`, and `leibnizexts.h`. While `lbcc` is being executed, some information about the compile process is displayed. Ignore that information for the time being. Next, the makefile invokes the `gcc` compiler to compile the C code of files `machine.c`, `leibniztrans.c`, and `leibnizerroraction.c` and to link the resulting files with the object files of `Leibniz/Code`.

Fourth, type `machine` and press return to enter the expert system coded in `machine.c` and `machine.log`. Follow the instructions and provide answers to questions posed by the expert system. When done, scan the C code of `machine.c` and the logic formulation of `machine.log` to get an idea how the expert system analyzes various scenarios.

## lsqcc Program

For a first try of program `lsqcc` for learning logic, the user should proceed as follows.

First, create a directory having any name. We assume below that the name is `Learnlogic`. Copy into `Learnlogic` all files of the subdirectory of `Leibniz/Lsqcc/Makedata`.

Second, go into directory `Learnlogic`. Copy the file `lsqccparams.data1` into a new file called `lsqccparams.dat`. Access the latter file and change the line

```
training/testing directory = /home/Learnlogic/
```

so that the complete path ending in subdirectory `Learnlogic` is specified.

Third, execute program `lsqcc` of `Leibniz/Lsqcc/Code` while in directory `Learnlogic`. The program analyzes the data of the file `data1.trn`, extracts logic formulas, and computes certain distributions and probabilities. The logic formulas are outputted in the file `data1.sep`, while the distributions and probabilities are provided in the file `data1.dis`. The user may employ any text editor to look at these files.

The program `lsqcc` in `Learnlogic` also applies the formulas of `data1.sep` to data in the file `data1.tst` and outputs the results in the file `data1.vot`. The user may want to look at these two files. Details about the above files—`data1.trn`, `data1.sep`, `data1.dis`, `data1.tst`, and `data1.vot`—are provided in Chapter 6.

## cutcc Program

For a first try of program `cutcc` for transformation of data, the user should proceed as follows.

First, create a directory having any name. We assume below that the name is `Transform`. Copy into `Transform` all files of the subdirectory of `Leibniz/Cutcc/Makedata`.

Second, go into directory `Transform`. Copy the file `cutccparams.heart` into a new file called `cutccparams.dat`. Access the latter file and change the line

```
training/testing directory = /home/Transform/
```

so that the complete path ending in subdirectory `Transform` is specified.

Third, execute program `cutcc` of `Leibniz/Cutcc/Code` while in directory `Transform`. The program analyzes the rational data and set data of the file `heart.rtr`, determines certain cutpoints, and transforms the data of `heart.rtr` to logic data, which are placed into a file `heart.trn`. The latter file plus a second output file `heart.ptl` may be used as input files for program `lsqcc`.

Program `cutcc` produces additional output files that show the cutpoints. The files are `heart.cut` and `heart.vis`.

Finally, program `cutcc` also transforms an additional input file `heart.rts` with rational data and set data to a file with logic data, which is called `heart.tst`. The latter file is another input file for program `lsqcc`.

Details about the above files—`heart.rtr`, `heart.ptl`, `heart.cut`, `heart.vis`, `heart.rts`, and `heart.tst`—are provided in Chapter 7.

At this point, the user has enough insight into the *Leibniz System* to read the entire reference manual and then use the *Leibniz System* in conjunction with the book *Design of Logic-based Intelligent Systems*.