Installing and Using

NEFCON–I

Version 1.0

Detlef Nauck
Technical University of Braunschweig
Department of Computer Science
Bueltenweg 74 – 75
38106 Braunschweig, Germany
Tel: +49.531.391.3155
Fax: +49.531.391.5936
Email: nauck@ibr.cs.tu-bs.de

June 16, 1994

Acknowledgements

After we developed the idea of the NEFCON model we needed skillful programmers to implement our algorithms. The first version of NEFCON–I was implemented by Hermann–Josef Diekgerdes for his diploma thesis. Now Roland Stellmach is continuing the work, and he is currently preparing a new release of NEFCON–I for his diploma thesis, using an learning algorithm.

Both of them have done and are doing a great job, and their work helped us to improvee the definition of the NEFCON model.

Contents

Chapter 1

Introduction

NEFCON–I means **NE**ural **F**uzzy **CON**troller – based on **I**nterViews, and it is an X11 simulation environment to build and test neural fuzzy controllers based on the NEFCON model developed at the Department of Computer Science of the University of Braunschweig [Nauck, 1994b]. NEFCON–I is able to learn fuzzy sets and fuzzy rules by using a kind of reinforcement learning that is driven by a fuzzy error measure. To do this NEFCON–I communicates with another process, that implements a simulation of a dynamical process.

NEFCON–I can optimize the fuzzy sets of the antecedents and the conclusions of a given rule base, and it can also create a rulebase from scratch. The theoretical foundations of the NEFCON model are not discussed in this report. If you are interested please refer to our publications [Nauck and Kruse, 1992b, Nauck and Kruse, 1992a, Nauck et al., 1992, Nauck and Kruse, 1993, Nauck et al., 1993, Nauck, 1993, Nauck and Kruse, 1994, Nauck, 1994a, Nauck, 1994c]. Some of them you can also find on our FTP server (see Chapter 7).

If your are able to read German, you can download the diploma thesis of Hermann–Josef Diekgerdes [Diekgerdes, 1993], who wrote the first version of NEFCON–I. There you can find a short discussion of the theoretical foundations, and a comprehensive description of the data structures and algorithms used in the software.

For a detailed discussion of neural networks and neural fuzzy systems including the NEFCON model please refer to our new book [Nauck et al., 1994] which will be available in June '94.

Chapter 2

Installation

2.1 Required Configuration

NEFCON–I will run on any X–Window–System (X11R5), if you have InterViews 3.1 [Linton et al., 1992]. If you do not have InterViews you can get it by FTP[1]. You will have to recompile the software to run it on your machine. Thus you will need a complete X11R5 with include files and libraries, Xt include files and libraries, InterViews include files and libraries, and last but not least a C++ compiler (we have used GNU's g++ and SUN's CC). We have NEFCON–I running on Linux-PCs and on SUN-Workstations, and it should be no problem to compile it on other platforms.

2.2 Recompiling NEFCON–I

You should have obtained the sources in a single compressed tar file `nefcon_1.0.tar.gz`. When you unpack the file by `tar -xzvf nefcon_1.0.tar.gz` you will get a new directory `nefcon` where you will find the source files for NEFCON–I in the directory `nefcon/source`, and the three additional directories. The directory `nefcon/data` contains the examples, `nefcon/datagen` contains the data generator, and `nefcon/pendulum` contains the simulation of the inverted pendulum. The tar–file has been compressed by "gzip 1.2.4". If your tar command does not allow on the fly decompression use `gunzip` or `gzip -d` first, and then unpack with `tar -xvf`.

Now set the CPU environment variable (e.g.: type `setenv CPU SUN4` if you use a C–shell on a Sparcstation, or type `export CPU = LINUX`, if you use a Bourne shell on a Linux machine). Then enter the directory nefcon/source and type `ivmkmf -a` to create the Makefile from the Imakefile. Now the program can be compiled by typing `make`.

Remember, after having changed the Imakefile you always have to run `ivmkmf -a`, because otherwise your changes won't have any effect. Of course, you will have to recompile the program as well.

After compilation you will find the executable file `nefcon` in `nefcon/source/$CPU/`, where `$CPU` denotes your type of system. Move the file to your directory where you store your

[1]InterViews can be obtained by anonymous FTP from `interviews.stanford.edu` (IP address 36.22.0.175).

binaries, or create a symbolic link to it, so you can run it by just typing `nefcon`.

Now change to the directory `nefcon/pendulum/` and do the same to compile the pendulum simulation (run `ivmkmf -a` and then `make`). You will find the executable `pendulum` in the directory `nefcon/pendulum/$CPU/`.

After this change to the directory `nefcon/datagen/` and compile the data generator by entering `ivmkmf -a` and then `make`. This can only be done after you have compiled the pendulum, because some of its object files are used by the data generator. If you decided to change the default directory structure you have to edit the Imakefile. The executable file `datagen` will be in `nefcon/datagen/$CPU/`.

Chapter 3

Running NEFCON–I

3.1 Command line parameters

When you have started NEFCON–I it assumes that it can find files that contain knowledge bases in the directory `./data/`, or if it is not present it looks for them in the current directory `./`.

NEFCON–I will start with a German user interface by default. If you want to have an English interface please use `nefcon -e`. If your data files are not in `./data/` you can make NEFCON–I to use your own directory with the `-dir=<your_own_directory>` option.

NEFCON–I also knows some InterViews options like `-bg color`, and `-fg color`. All options can be set as X11 resources. You can find the names of these options in Table 3.1 below.

Option	Resource	Default	Purpose
`-dir=<directory>`	*DataDirectory	./data	loading and storing knowledge bases
`-p=<name>`	*protocol	protocol.nfc	file that contains the runtime protocol
`-e`	*english	off	using english (on) or german (off) texts
`-editorfont=<font>`	*NfcEditorFont*font	*helvetica-medium -r-normal–12*	
`-infofont=<font>`	*NfcInfoFont*font	*helvetica-bold -r-normal–12*	
	*PushButton*font	*times-bold -r-normal–12*	

Table 3.1: X11 resources and command line parameters

The default settings can be changed within the file `main.c`. For example, if you prefer English to German texts you should change the default setting of the `*english` option to `on`, so you don't need the `-e` option.

3.2 Using the Program

After you have started NEFCON–I the main window appears (see Fig. 3.1). Above it you see a copyright notice, that you can close by clicking on it with the left mouse button. Now you can start working.

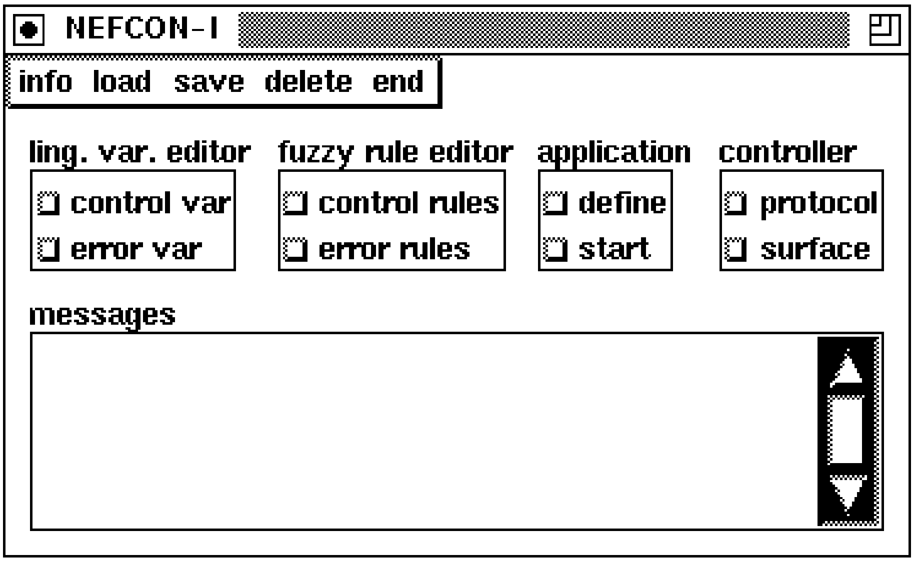

Figure 3.1: The main window of NEFCON–I

First you have to define a knowledge base, but for the beginning it might be better to play around with the examples we have prepared.

A knowledge base contains:

- a set of linguistic variables and their partitioning, that is a set of fuzzy sets
- a set of error variables that are linguistic variables as well, but they are only needed to define the fuzzy error. You always have to define the "fehler"–variable. "fehler" is German and means error.
- a set of control rules like
  IF angle is pos_big AND angle_velocity is pos_big THEN force is pos_big
- a set of error rules like
  IF angle is pos_zero AND angle_velocity is pos_zero THEN fehler is pos_zero

Because it is not very convenient to define the knowledge base by typing a lot of text, NEFCON–I offers two editors. The first one is the editor for the linguistic variables, and it starts if you click on one of the two **ling. var editor** buttons which are labelled **control var** and **error var**. The second editor is used to enter fuzzy rules, and it can be invoked by clicking either on the **control rules** button or the **error rules** button within the **fuzzy rule editor** box.

The Linguistic Variable Editor

The linguistic variable editor is used to define the necessary variables and their fuzzy sets. First you have to enter the control variables that describe the system you want to control. Click on the **control var** button in the **ling. var. editor** box, and the editor appears (see Fig. 3.2).

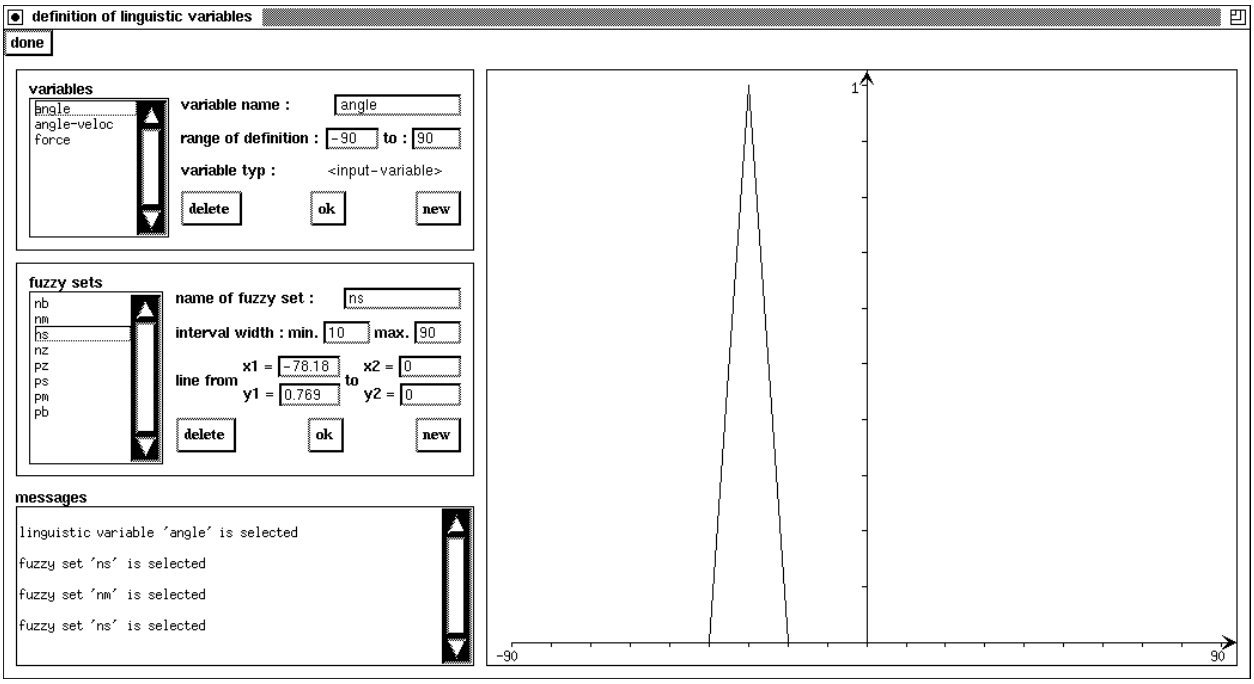

Figure 3.2: The linguistic variable editor

To define a new variable you first have to enter a variable name in the edit field next to **variable name** and to press Return (your mouse pointer has to be within the edit field to do this). Then enter the variables range of values, choose its type (input or output) by clicking on the selection next to **variable type**, and finally click on the **ok** button in the **variables** area. You may now define the fuzzy sets for this variable, or you can first enter the rest of the variables you need.

If you have already entered variables you can modify them by choosing their name in the variables list (double click on its name), and clicking **ok** when you are done. The **delete** button works as assumed, and the **new** button deletes the variable name, when you made a typo. You can also use `Control-D` and `Delete` to correct your errors within the edit fields.

Fuzzy sets are always defined for the selected variable. You have to specify a name, and the minmum and maximum width which has to be respected by the learning algorithm. The fuzzy set itself, is specified by lines which you may draw in the right hand window of the editor. Click where the first and where the last point of a line should be to draw it. You may draw as many lines as you wish, but the learning algorithm only accepts triangles for the input variables and, monotonous fuzzy sets (i.e. a single line in this case) for the output variable.

A better way to specify fuzzy sets is to enter the coordinates in the edit fields next to **line from** and **to**. Here it is easier to have exact values, which are hard to get by using the mouse. If you want to change or delete a alreay defined fuzzy set, select it from the list by double clicking its name. *Hint:* If you want to erase the lines of a fuzzy set, draw a line from left to right on the bottom of the drawing area.

When you have defined all variables and fuzzy sets, click on **done** and **exit**. If you click on **abort** instead, everything you have edited will be discarded.

To use the learning capabilities of NEFCON–I you also have to define fuzzy sets for the error signal. If you click on **error var** in the **ling. var. editor** area the linguistic variable editor opens and displays all the input variables, and the error variable "fehler" (German for error). You have to specify fuzzy sets for all these variables. Sorry, there is no possibility in the moment to copy the already defined fuzzy sets of the input variables. We will try to provider a nicer user interface in an updated release. But in fact you usually do not want to use the same fuzzy sets, because you can usually use less fuzzy sets for the error definition (see examples).

Please note: The range of the error variable is $[-1, 1]$, and it makes no sense to change this, even though the editor let's you do it. The learning algorithm can only work with error values in $[-1, 1]$.

The Rule Editor

After you have specified the necessary variables, you have to enter a rule base. After having clicked on the **control rules** button in the **fuzzy rule editor** area of the main window the rule editor appears (see Fig. 3.3).

There are three areas in the editor window, and in two of them you can edit or enter fuzzy rules.

1.) **Rule board**: Here you only can select rules for editing or deletion.

2.) **Rule input**: Here you can type in rules as text into the edit fields. This editor is more convenient, if you want to enter rules that do not use all the input variables in the antecedents, or when there is more then one output variable (which makes no sense in this version). You can change the ordering of the variables by clicking on their names.

3.) **Rule table**: Here you can simply define rules by clicking on a field and choosing a fitting fuzzy set for the output variable from the pop up list. If there are more than two input variables you can fix the values for some them by clicking on their names which will be shown above the table in this case. This way you can specify a rule base by entering several tables. You can change the variables that form the table by clicking on their names (top and left hand side of table).

After you have successfully defined the knowledge base leave the editor by clicking **done** and **exit**. Clicking on **abort** instead discards all your work.

Now you have to define a rule base that describes the fuzzy error. The idea is that you are usually able to tell how good or bad the current state of the controlled systems is, even if you do not know the exact control value to be applied. But you usually know the sign of the control values, because you know the desired state of the controlled system, and you know at least basically how it works. Let's take the pendulum as an example: If the pendulum

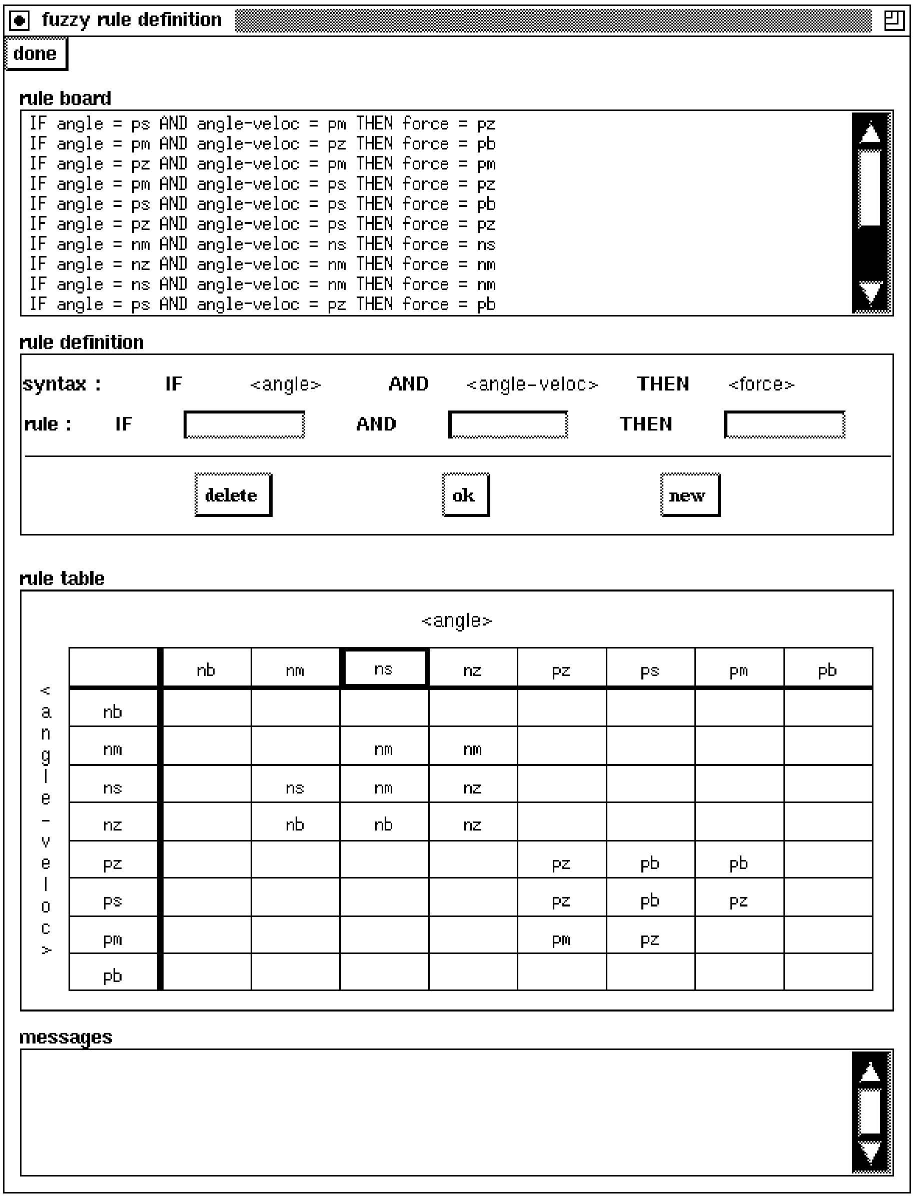

Figure 3.3: The fuzzy rule editor

falls to the right, you know this is a bad state, and that you have to push it to the left. If the pendulum is roughly vertical this is a good state, and you do not have to take an action. If the pendulum is tilted to the left side, but moves to its upright position, this is also a good state, which means the error is only small.

The sign of the error depends on the sign of the necessary output value. The error is negative, if the output has to be negative, and the error is positive, if the output has to be positive. For more information about the fuzzy error please check out the references and the examples.

You enter the rule base for the fuzzy error in the same way as the control rule base. But this time the variable "fehler" (error) is the output variable. To activate the rule editor to define the fuzzy error rule base click on **error rules** in the **fuzzy rule editor** area of the main window.

Defining an Application Program

NEFCON–I needs an application program from which it receives input values and to which it can pass on the control output. This application program has to read the control value from the standard input, and it has to write the input values for NEFCON–I to the standard output. NEFCON–I takes care of receiving and passing on the values, so you can easily write your own application. This can be a fancy graphical simulation of a plant you want to control, or just a simple calculation of a differential equation describing your dynamical system.

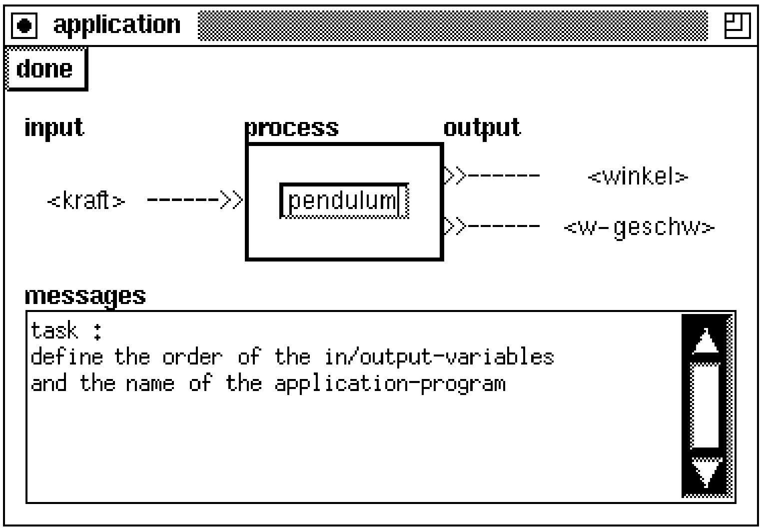

Figure 3.4: Defining an application for NEFCON–I

The application program must be started by NEFCON–I to establish the communication. To do this click on the **define** button in the **application** area of the main window. A window will pop up (see Fig. 3.4), where you have to enter the name of the program to be started, and the order in which the input and output values are being processed. Usually there is only one control value, so there is no problem here. But the order of the input variables that are passed from the application to NEFCON–I have to be defined correctly (the upper value is the first one). You can change the order of the variables by simply clicking on their names.

After you have defined the application leave this window by clicking **done** and **exit**. If you chose **quit** instead, your changes will be discarded. Now you may start the program by clicking on the **start** button in the **application** area of the main window.

If your application program does not has its own window, you don't have the possibility to stop it. It will end when you leave NEFCON–I. So it might be a good idea to write yourself a simple InterViews application that has a quit button.

The Controller Surface

Before you start the controller you might want to take a look at the protocol options of NEFCON–I. Click on the **protocol** button in the **controller** area of the main window and you see the pop up window in Fig. 3.5. There you can define the looks of the graphical controller surface, and whether you want to have a protocol file or not.

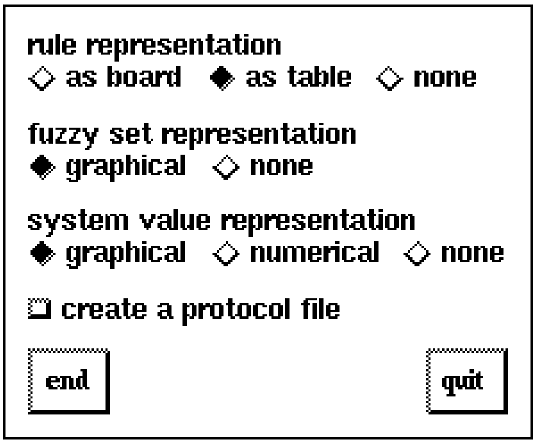

Figure 3.5: Defining the graphical presentations of NEFCON–I

In the first line you can define the form of the rule representation. You can have a table (default) which only makes sense, when you have two input values and a single control value. If you choose a "board" of rules, you will just get a list of rules, or you can decide to not display the rules at all.

In the second line you can specify whether to see graphical representations (default) of the fuzzy sets, or not to display them at all, and in the third line you can define the way the system values are to be displayed. You can display them as a graph (default), or just as numeric values, or not at all. The system values are the fuzzy error, and the values of the input and output variables.

At last you can decide whether you want to have a protocol file or not (default). This file will have the default name `protocol.nfc`, or the name you have chosen by the command line option, or by X11 resources, respectively. This file is useful if you want to take a look at the changes caused by the learning procedure, or if you want to draw nice control surface pictures with gnuplot (see Chapter 6). Old protocol files will be overwritten, so rename them if you want to keep them.

Leave the protocol specification window by clicking on **end**; **quit** discards your changes.

Now it is time to open the controller surface by clicking on **surface** in the **controller** area of the main window. Using the default protocol definitions it looks like Fig. 3.6.

In the upper right hand side of the controller window you can see the system values. This are the current values of the variables and the fuzzy error. If you choose a graphical representation, you will only see one variable, but you can switch around. In case of a numerical representation, all variables are displayed.

On the upper left hand side the rule base is depicted (in case you have entered one). It is either shown as a table or as a matrix like in Fig. 3.6. Below the rule base you can see the fuzzy sets of the linguistic variables. You can switch between the variables by clicking on the names in the list. During the learning process you can watch the fuzzy sets change.

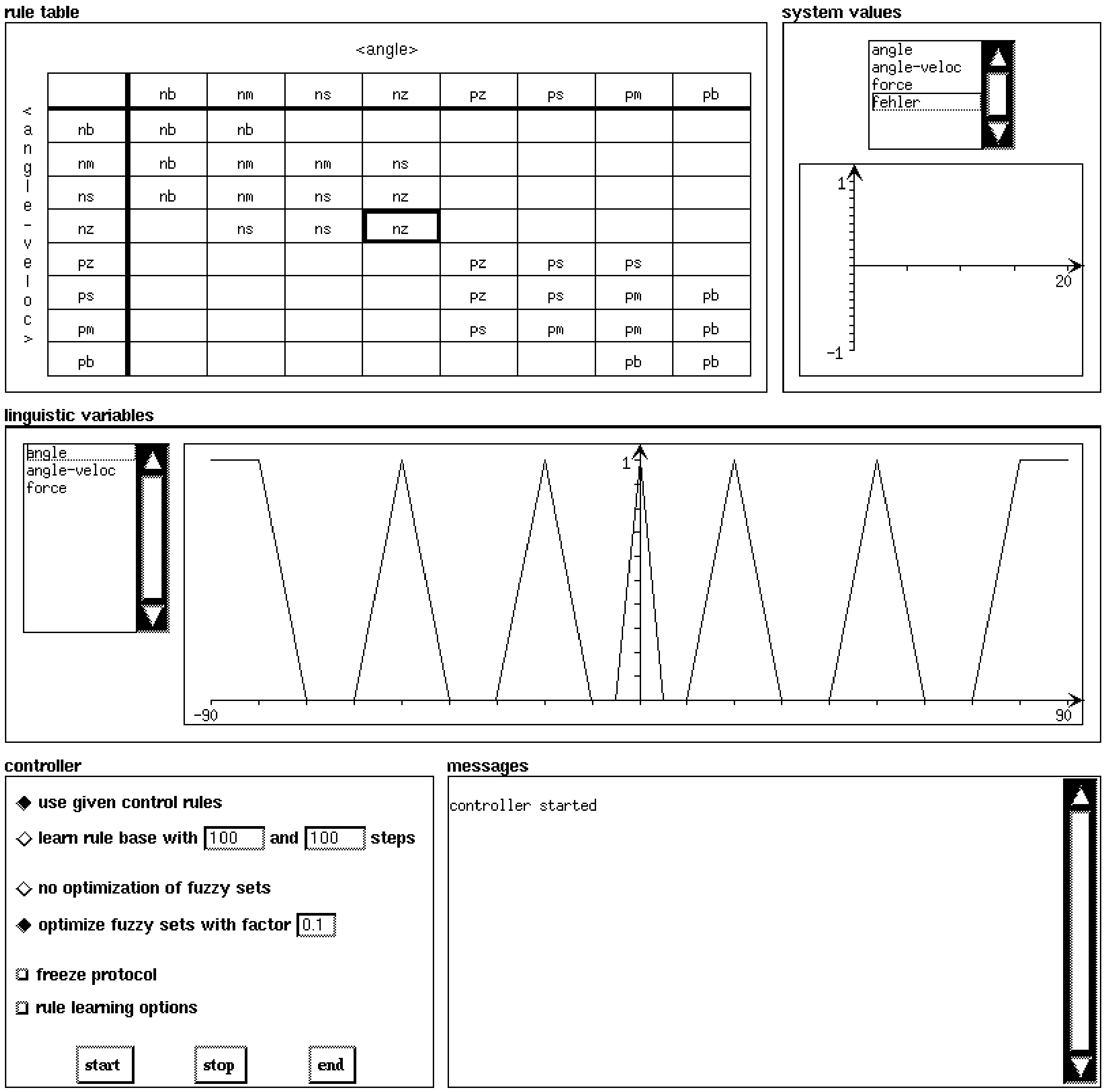

Figure 3.6: The main window of the neuro–fuzzy controller

Under the fuzzy sets you can see some system messages, and an area where you can define some learning options:

- **use given control rules**: If this is checked, the rule base that you have entered is used. The rules are not learned.
- **learn rule base with** X **and** Y **steps**: If you want to learn the fuzzy rules, you have to click on this option. The value of X determines the length of the first stage, and the value of Y the duration of the second stage of rule learning. In the first stage rules are deleted if they produce an negative output when a positive would be correct, and vice versa. During the second stage from rules with identical antecedents, but different conclusions, one rule is choosen (see our papers for details).

- **no optimizations of fuzzy sets**: As it says. Choose this if you have a working controller, or if you want to create data for graphs of control surfaces.

- **optimize fuzzy sets with factor** σ: The fuzzy sets will be learned (optimized) with learning rate σ. We recommend to keep the value of 0.1.

- **freeze protocol**: Clicking on this option stops the updating of the controller window. By this you can speed up the simulation. You can also gain additional speed, if you iconize your application simulation (at least if you use our pendulum). You can also toggle this option while the controller runs.

- **rule learning options**: When you click on this entry, a pop up window will open, where you can enter several options that guide the rule learning process (see Fig. 3.7). Rules are deleted during the learning process when they behave counterproductive, and when they are not used over a long time. This last aspect can be influenced by these options. You can specify a percentage value p, meaning that if a rule is not used in p percent of all cases, it has to be deleted after the rule learning phase. You can additionally specify that a rule has to be deleted, when it is not used at least once in x consecutive steps. The third value defines when a rule is considered as unused. Usually a rule is not used, when its degree of fulfillment is 0. But you might want to delete rules that are only fulfilled to a very small degree in all cases.

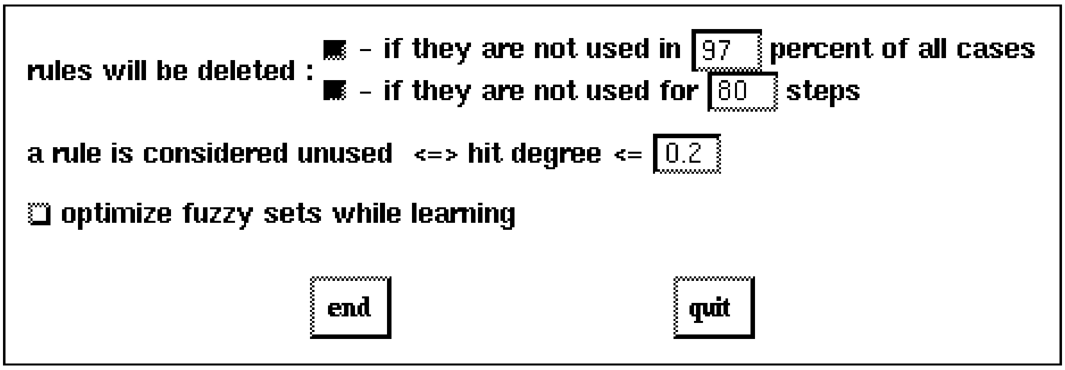

Figure 3.7: Rule learning options

When you have chosen appropriate options start the controller by clicking on the **start** button. After this start your application. The **stop** button stops the controller, and by clicking on **end** you close the controller window. The changes caused by the learning process are saved internally, but before you exit NEFCON–I by clicking on **end** in the main window (Fig. 3.1) remember to save the changes explictely using the **save** menu. You will not be notified of this! Once you click on **end** in the main window everything is lost (I know this is not user friendly, but there will be updates).

Leaving the controller window with **end** is safe. You can close the controller, make some changes to your rule base, or whatsoever, and resume learning.

At this moment the fuzzy set optimization works pretty well even though it is only allowed to change the width of the fuzzy sets. The learning of the rules also works for simple examples (pendulum), but we are not completely satisfied with it, because sometimes it fails. It is also not easy to get a working example with an application that has more than two state variables. But we are just working on it. Remember, this is a prototype. We are considering some new learning strategies, and we think future releases will be coming soon.

Chapter 4

The Application Program

At this moment, there exists only one application example; it is an animated implementation of the well known inverted pendulum example. To install it, you have to go through the same steps as you did, when you installed NEFCON–I (see Sect. 2.2).

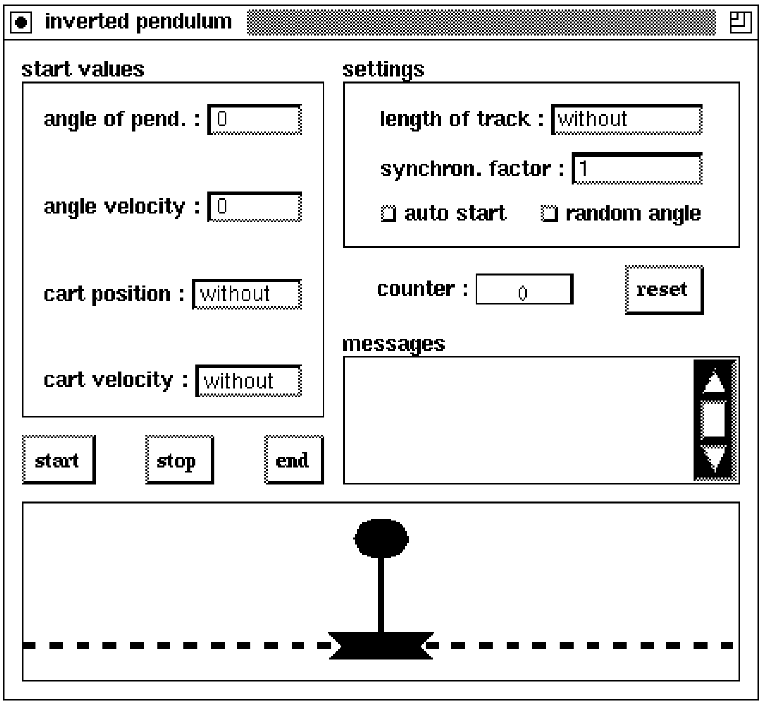

Figure 4.1: The inverted pendulum application

The surface of the existing application program is quite self explanatory (see Fig. 4.1). You can start the program, and play a little bit around with it, just to see how it behaves. It will have a German interface by default, and when you want to have the English version you have to invoke it by `pendulum -e`. When you start the simulation manually, it will send its outputs to your terminal, and it will wait for you to enter a force value to be applied to the cart.

Click on **start** to start the simulation. The program will output the current angle, and the current angular velocity (both will be 0 if you do not change the default values). Now enter a 0 as the force to be applied, and you will receive the new values for angle and angular velocity. Go on to enter zeros, and after a few cycles you will see the pendulum to fall over.

To create a new knowledge base, you only have to know the domains of the variables and their order for the communication with NEFCON–I (see Tab. 4.1).

kind of variable	name of variable	range of variable
input	force	$[-1000, 1000]$ Newton (but $[-30, 30]$ is reasonable)
output 1	angle	$[-90, 90]$ degrees
output 2	angle velocity	as calculated, typically $[-300, 300]$ degrees per second
output 3	position of cart	depends on x entered: $[-\frac{x}{2}, +\frac{x}{2}]$ meters, or unused (without) by default
output 4	velocity of cart	as calculated, typically $[-5, 5]$ meters per second, or unused (without) by default

Table 4.1: Domains and order of the variables of the pendulum application

The meaning of the buttons and options of the pendulum interface are explained in the following list:

- **Button "start"**: Start the simulation.
- **Button "stop"**: Stop the simulation.
- **Button "end"**: Abort the program.
- **Option "angle of pend."**: The angle at which the simulation is (re)started.
- **Option "angle velocity"**: The angle velocity for the (re)start of the simulation.
- **Option "cart position"**: Position of the pendulum at the (re)start of the simulation (not used by default: without). A value of 0 denotes the middle of the track.
- **Option "cart velocity"**: Initial velocity of the cart when the simulation is (re)started (not used by default: without).
- **Option "length of track"**: Length of the track on which the cart is able to move along (not used by default: without).
- **Option "sync. factor"**: This option determines how often the state of the pendulum is updated before a new force value is accepted. If you increase this factor controlling the pendulum gets much harder, because your controller can not react immediately. The default value is 1, only positive integers are allowed.

- **Check box "auto start"**: If you check this option the simulation will automatically restarted with the start values, if the pendulum has been dropped.
- **Check box "random angle"**: If you check this option there will be always used a random angle to (re)start the pendulum.

If you want to write your own application, it must use the standard input/output to communicate with NEFCON–I. However, you don't have to worry about communication itself. Just write the state variables to the standard output, and wait for the control value supplied by NEFCON–I to the standard input of your application.

If you want to use the pendulum simulation together with NEFCON–I, you have to invoke it from within NEFCON–I. The invokation mechanism does currently not allow it to specify any parameters or options. If you want to have an English interface for the pendulum create a small shell script containing just the command `pendulum -e`. Name this script e.g. `pend`, make it executable, and invoke it instead of pendulum. Alternatively you can change the default settings in the file `main.c`, or specify the X11 resource `*english`, and set it to "on".

Chapter 5

Examples

Change to your NEFCON directory, and start the program. Click on **load** and **control var**, and a dialog window will pop up (see Fig. 5.1). Assuming that the directory `data` exists in your current directory, you will see a list of files including `exam1.lv`. Click on this name, and then on the OK button, and the linguistic variables are loaded. Now in the same manner load the error variables in `exam1_e.lv`, the control rules in `exam1.rg`, and the error rules in `exam1_e.rg`.

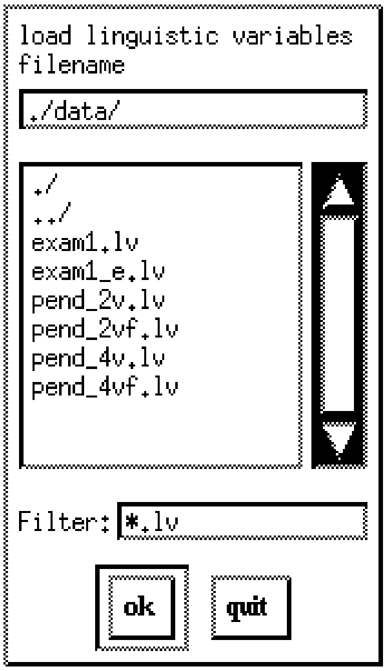

Figure 5.1: The dialog window for loading files

After this click on **define** to specifiy the application program, and fill in the name `pendulum` (the program has to be somewhere in your path). Then click on **done** and **exit**. Now start the application by clicking on **start**, and start the controller interface by clicking on **surface**.

Now click on **start** in the controller interface, and double click on **angle** in the linguistic values area and the system values area. You will see the initial fuzzy sets for the angle of the pendulum, and the rule base. In the system values area the angle will be displayed graphically, once the controller is started. The controller is now ready, and it waits for the data of the application to arrive.

Now click on **auto start** and **random angle** in the pendulum interface, and start the pendulum by clicking on **start**. The pendulum will drop for a few times because of the insufficient initialization of the fuzzy sets. But in a few seconds (depends on your hardware) NEFCON–I will be able to balance it. To be also able to control extreme angles, you have to "disturb" the pendulum: Click on the edit field of **angle**, delete the current value, and enter an angle between -20 and $+20$. The pendulum will again fall over, and depending on the random start angles it will be balanced again after a few trials. Continue to "push" the pendulum, and alternately enter positive and negative angles. After some time NEFCON–I should be able to cope with angles between -15 and $+15$ degrees. At this time the fuzzy sets of the angle should look like in Fig. 5.2.

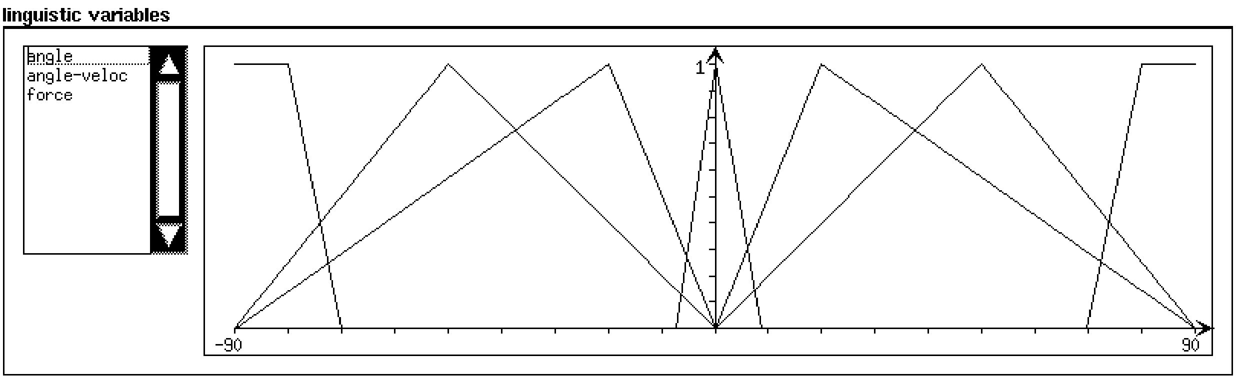

Figure 5.2: The fuzzy sets of angle after training

You may now stop the simulation, by first stopping the pendulum, and then stopping the controller. The changes in the fuzzy sets will be stored, and you can save them once you have left the controler interface by clicking on "end". To save the fuzzy sets click on "save" and "control var" in the main window, and enter a file name or select one by clicking on it.

In the second example we will try to learn fuzzy rules. The easiest way is to leave NEFCON–I and then to restart it. Alternatively you can also delete the control variables and the control rules using the **delete** menu. But we recommend to start over again to reset the random seed. Let's assume you have restarted NEFCON–I. Now load the control variables in the file `exam2.lv`, and again load the error variables from `exam1_e.lv`, and the error rules from `exam1_e.rg`. Do not load any control rules, however.

Now specify the application (pendulum) and start it. Then start the controller surface and click on **learn rule base with**, and enter 2000 as the first value and 3000 as the second value. Now click on **rule learning options**, and enter 100 as the percent value, when a rule has to be deleted. Unselect the option **if they are not used in** x **steps**, we will not use it in this example. Finally enter 0 in the edit field that specifies when a rule is considered as used, and click on **end**.

Now start the controller by clicking on **start**, and then start the pendulum, after you have clicked on **auto start** and **random angle**. During the first 2000 steps the pendulum will not necessary fall over, but it will be very unsteady. During the next 3000 steps the pendulum will fall over several times. After these 5000 steps a rule base will be displayed. It should look like the rulebase in Fig. 5.3, but there may be differences, because the evaluation of rules is done randomly during the second phase.

rule table

<angle>

<angle-veloc>	nb	nm	ns	nz	pz	ps	pm	pb
nb								
nm	nz	nm	nm	nb	nb	nz	nz	nb
ns	nz	ns	ns	nz				
nz	ns	nm	ns	nz				
pz					pz	pm	pm	pm
ps					ps	pm	pm	pm
pm	ps	pb	pz	pm	pb	pz	ps	pb
pb								

Figure 5.3: A rule base learned by NEFCON–I

NEFCON–I should be able to balance the pendulum immediately with this rule base, although it looks a little bit unusual. Try to "push" the pendulum again to learn to control extreme angles. After a few pushes the controller should be able to cope with angles between -15 and $+15$ degrees.

As already mentioned, the rule base looks a little bit unusual. You cannot expect NEFCON–I to learn a rule base like you would specify it. The result depends on the learning parameters (number of steps, options when a rule may be deleted because it is not used frequently enough, etc.), and on the error variables and error rules. Try to play around with these parameters, and observe the results.

Chapter 6

Generating Pictures of Control Surfaces

In the NEFCON–I distribution you will find a simple program that can be used to generate data for drawing control surfaces with software like GNUPLOT. This programm is called `datagen`, and it is a simple application that just outputs two values and then waits for an input value from standard input/output. You can specify two ranges of values and a step width for each of them. When you start the program it increments the values by their step width in two nested loops (see Fig. 6.1).

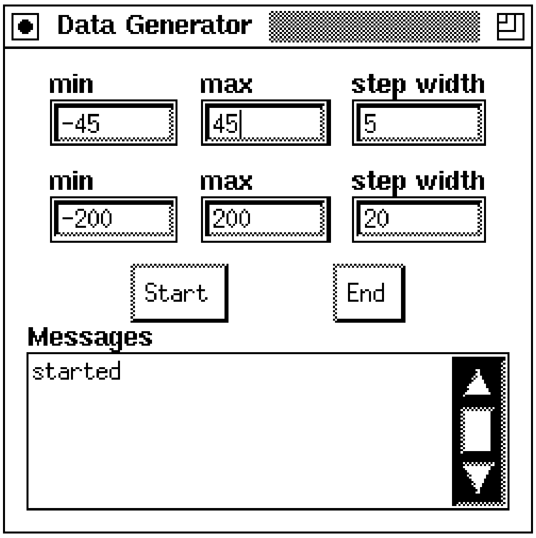

Figure 6.1: The application `datagen`

If you want to create a figure of a control surface you need software like GNUPLOT, i.e. a programm that can take data points and display them in 3–dimensional graphs. To create the necessary data, start NEFCON–I, load a rule base, and linguistic variables for a system with only two state variables (e.g. the inverted pendulum using only angle and angle velocity), and define `datagen` as the application, and start it.

Now click on **controller** and **protocoll** and activate the button next to the text **create a protocol file**. You probably also want to specify **none** for the rule representation and fuzzy set representation, and **numerical** for the system value representation. This way the graphical surface of the controller needs less time to be updated, and the software runs faster. Leave the dialog window by clicking on **end**.

After you have done this, start the surface of the controller. It is important that you click on **no optimization of fuzzy sets**! If you forget to do so, the learning process will be carried out, and your fuzzy sets will change during the run. But to get an accurate picture of the control surface created by the current rule base, this must not happen, of course. After you have made sure, that **no optimization of fuzzy sets** is selected, and **optimize fuzzy sets with factor** is not selected, you can start the controller.

Now enter the values into the data generator. If you use the provided examples of the inverted pendulum, enter -90 as the minimum values, and 90 as the maximum value in the first line. Use as step width of e.g. 9. In the second line enter -200, 200, and use a step widht of e.g. 20 This will produce $21 \cdot 21 = 441$ lines of data. Now start the data generator. You can observe the run in the system values. After the data generator is done, close the controller surface, what also closes the protocol file. Don't forget to rename the file, if you are going to do more runs with different rule bases, or it will be overwritten.

You will find the data in the file `protocol.nfc` in the current directory. Before and after the lines data the parameters of the fuzzy sets are printed. If you want to plot your data, you have to delete this information using your favourite text editor.

Each line of data contains a value for the angle, the angle velocity, the force calculated by NEFCON–I, and an error value (assuming you have used the rule base for the inverted pendulum).

If you want to use GNUPLOT[1] to create a 3–dimensional plot you have to enter a blank line each time the second variable has taken its maximum value. Let's say you have saved your edited file as `data.gpl`. Now start GNUPLOT and enter the following commands:

```
set param
splot "data.gpl" using 1,2,3
```

You will receive a 3–dimensional plot of the control surface. For other commands of GNUPLOT please refer to its documentation.

[1]You can obtain GNUPLOT and other GNU software by FTP from prep.ai.mit.edu (IP address 18.71.0.38)

Chapter 7

Where to Get More Information

If you can read German, get the diploma theses of Herman Dieckgerdes, where the implementation of NEFCON–I is described [Diekgerdes, 1993], and for the theoretical background please refer to our book [Nauck et al., 1994]. Otherwise you can get some of our papers which also describe some aspects of the theoretical background behind the NEFCON model. You can find these documents on our FTP server at `ftp@ibr.cs.tu-bs.de` (IP address 134.169.34.15) in the directory `pub/local/papers`. Get the file `INDEX` first, and check which documents you want to download.

There you also find new releases of NEFCON–I in the directory `pub/local/nefcon` in the file `nefcon_x.y.tar`, where `x.y` is the release number. New releases will be announced in the newsgroups `comp.ai.fuzzy` and `comp.ai.neural-nets`. You will probably also find NEFCON–I on an FTP server near you. Please check the newsgroups and mailing lists like `fuzzy-mail`[1] for announcements.

If you have questions or comments concerning the NEFCON model or the software, please mail them to `nauck@ibr.cs.tu-bs.de` (Detlef Nauck). Questions concerning the program code will be usually forwarded to the developer who works on NEFCON–I (currently Roland Stellmach). You can send mail directly to him (`stellma@ibr.cs.tu-bs.de`), but please note that he is a student at our department, and he does not work full time here, i.e. answering your questions may take some time.

If you are going to use NEFCON–I please send an email to `nauck@ibr.cs.tu-bs.de`, so we can keep track of users. We will inform you about updates. We would also appreciate to hear from you, if you tried to use NEFCON–I to solve an applicational problem (successfully or not).

If you plan to put NEFCON–I on any FTP server, please feel free to do so. In this case please send us an email, so we can send updates to this site.

[1] If you can't read Usenet news, but you can receive email, you might want to join the fuzzy–mail list maintained by Marcus Herzog and Wolfgang Slany. Send an email to `listserver@vexpert.dbai.tuwien.ac.at` that contains only the line `SUBSCRIBE FUZZY-MAIL <your name>` in its message.

Chapter 8

Known Bugs

- You always have to define fuzzy sets for the error variables, even if you do not want to learn anything.
- The **new** buttons do not work as expected.
- Setting and selecting variables and fuzzy sets is sometimes not obvious.
- There is the word “fehler” instead of “error” in the English version.
- There is no manpage.

Bibliography

H. Diekgerdes (1993). *NEFCON-I: Entwurf und Implementierung einer Entwicklungsumgebung für Neuronale Fuzzy-Regler*. Diplomarbeit, Technische Universität Braunschweig.

M. A. Linton, P. R. Calder, J. A. Interrante, S. Tang and J. M. Vlissides (1992). *InterViews Reference Manual Version 3.1*. The Board of Trustees of the Leland Stanford Junior University.

D. Nauck (1993). *NEFCON-I: Eine Simulationsumgebung für Neuronale Fuzzy-Regler*. In *1. GI-Workshop Fuzzy-Systeme '93*, Braunschweig.

D. Nauck (1994a). *Building Neural Fuzzy Controllers with NEFCON–I*. In R. Kruse, J. Gebhardt and R. Palm, eds.: *Fuzzy Systems in Computer Science*. Vieweg–Verlag, Wiesbaden.

D. Nauck (1994b). *Modellierung Neuronaler Fuzzy–Regler (Modeling Neural Fuzzy Controllers)*. PhD thesis, Technische Universität Braunschweig.

D. Nauck (1994c). *Neural Fuzzy Systems*. In R. Kruse, J. Gebhardt and R. Palm, eds.: *Fuzzy Systems in Computer Science*. Vieweg–Verlag, Wiesbaden.

D. Nauck, F. Klawonn and R. Kruse (1992). *Fuzzy Sets, Fuzzy Controllers, and Neural Networks*. Wissenschaftliche Zeitschrift der Humboldt-Universität zu Berlin, R. Medizin, 41(4):99–120.

D. Nauck, F. Klawonn and R. Kruse (1993). *Combining Neural Networks and Fuzzy Controllers*. In E. P. Klement and W. Slany, eds.: *Fuzzy Logic in Artificial Intelligence (FLAI93)*, pages 35–46, Berlin. Springer–Verlag.

D. Nauck, F. Klawonn and R. Kruse (1994). *Neuronale Netze und Fuzzy–Systeme (Neural Networks and Fuzzy Systems, in German)*. Vieweg–Verlag, Wiesbaden.

D. Nauck and R. Kruse (1992a). *Interpreting Changes in the Fuzzy Sets of a Self-Adaptive Neural Fuzzy Controller*. In *Proc. Second Int. Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems (IFIS'92)*, pages 146–152, College Station, Texas.

D. Nauck and R. Kruse (1992b). *A Neural Fuzzy Controller Learning by Fuzzy Error Propagation*. In *Proc. Workshop of North American Fuzzy Information Processing Society (NAFIPS92)*, pages 388–397, Puerto Vallarta.

D. Nauck and R. Kruse (1993). *A Fuzzy Neural Network Learning Fuzzy Control Rules and Membership Functions by Fuzzy Error Backpropagation*. In *Proc. IEEE Int. Conf. on Neural Networks 1993*, pages 1022–1027, San Francisco.

D. Nauck and R. Kruse (1994). *NEFCON–I: An X–Window based Simulator for Neural Fuzzy Controllers*. In *Proc. IEEE Int. Conf. Neural Networks 1994 at IEEE WCCI'94*, Orlando.

Appendix — Data file formats

The file formats are given in a form that is similar to a Backus-Naur notation. Please note that each number has to be followed by a blank, and each string has to be followed by a newline.

File format for linguistic variables

*.lv	=	**cnt-LV** (**ling-var**)$^{cnt-lv}$
cnt-lv	=	number of linguistic variables
ling-var	=	**lv-name min max type** **cnt-fs** (**fuzzy-set**)$^{cnt-fs}$
lv-name	=	name of linguistic variable
min	=	lower border of domain
max	=	upper border of domain
type	=	0 for input variables, 1 for output variables
cnt-fs	=	number of fuzzy sets
fuzzy-set	=	**fs-name** [**borders**] **cnt-iv** (**interval**)$^{cnt-iv}$
fs-name	=	name of fuzzy set
borders	=	'%' **lower-b upper-b**
lower-b	=	minimal support width
upper-b	=	maximale support width
cnt-iv	=	number of intervals
interval	=	**x1 y1 isin1 x2 y2 isin2**
x1	=	lower interval border
y1	=	membership degree of lower interval border
isin1	=	1 if the lower border is an element of the interval, 0 otherwise
x2	=	upper interval border
y2	=	membership degree of upper interval border
isin2	=	1 if the upper border is an element of the interval, 0 otherwise

File format of linguistic rules

*.rg	=	**cnt-rule** (**rule**)$^{cnt-rule}$
cnt-rule	=	number of linguistic rules
rule	=	**cnt-ant cnt-con**,
		(**expr**)$^{cnt-ant}$ (**expr**)$^{cnt-con}$
cnt-ant	=	number of expressions in the antecedents of a rule
cnt-con	=	number of expressions in the conclusions of a rule
expr	=	**lv-name fs-name**
lv-name	=	name of linguistic variable
fs-name	=	name of fuzzy set