Chapter 8
Practical Projects Using FuzzyLisp

8.1 Introduction

In the last sections of the previous chapter we introduced some simple examples of practical fuzzy logic based applications where a FRBS always had two crisp inputs and a crisp output. In this chapter we shall discover models of a bigger complexity that, more than examples, are intended to be a guideline for more ambitious goals. These models, aside practical, pretend to be inspiring for the reader. Engineers and scientists will soon find some ideas for using FuzzyLisp in their respective fields of knowledge, but any attentive reader will find new routes of exploration in order to tackle complex problems from the real world. There are three practical projects in this chapter chosen from astronautics, astronomy and medicine. All of them are exposed in a clear and friendly way, and it is expected that any reader of this book will find them interesting.

Section 8.2 introduces a simplified version of the Moon landing problem where a physical model (that is, a set of traditional kinematics equations) describe the descent movement of the NASA Lunar Module and then a FRBS takes control in order to get a smooth touchdown on Earth's satellite, being thus a combined exercise of simulation systems and fuzzy control. In Sect. 8.3 a sophisticated FRBS architecture is introduced for speech synthesis where double stars are the excuse for showing how to automatically generate expert reports in natural language from a set of numerical data. This is the most complex project in the chapter, but suggests many real applications in practically every science and branch of technology, from biology to robotics. Finally, in Sect. 8.4, the reader is invited to an introduction to adaptive fuzzy models using floating singletons while developing a model to interpret spirometric results for detecting Chronic Obstruction Pulmonary Disease (COPD), having into account not only numerical data from a spirometry but also smoking. Floating singletons theory is introduced adequately before the model's development, being the last piece of fuzzy theory in this book.

L. Argüelles Méndez, *A Practical Introduction to Fuzzy Logic using LISP*,
Studies in Fuzziness and Soft Computing 327,
DOI 10.1007/978-3-319-23186-0_8

Since the complexity of these models, as already stated, has grown with respect to the models developed in the previous chapter, the same has taken place with the corresponding Lisp code for every project. All the code is shown in the chapter with the appropriate comments here and there, but now the reader does not need to type everything at the keyboard. The "Lisp vision" is already on the reader's mind and you will find the complete programs in the book's web site. Do enjoy these projects. Maybe they will be the ultimate key for future developments in your life.

8.2 Landing the Eagle: Simulation and Fuzzy Control in Engineering

Back in 1984 I bought a rather sophisticated (by those days) programmable electronic calculator. Its programming language, embedded in its ROM circuitry, was a tiny dialect of BASIC, one of the more simple computer languages in the 80s. Despite its limitations as a computer language, every manufacturer of small personal computers offered a version of BASIC in their systems, and the small calculator was not an exception. In order to gain some experience with the system and its implementation of BASIC I decided to write a program to simulate the landing of the Eagle module on the Moon. After more than thirty years the program still runs on the calculator, as can be seen in Fig. 8.1.

Since the RAM memory of the device was less than 2K (yes, the reader is reading well, that's about 0.002 MB of RAM) the model was rather simple and some assumptions were needed to develop it. First a vertical descending trajectory

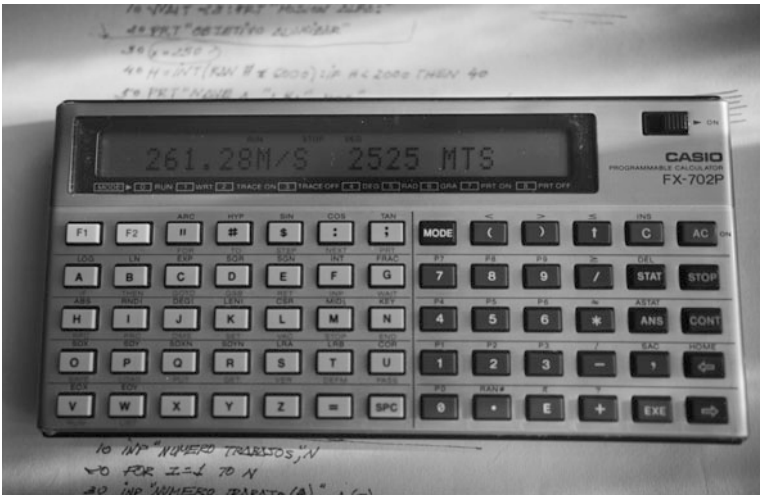

Fig. 8.1 A simple Moon landing simulator running on a 30+ years old programmable electronic calculator. Photograph by the author

was assumed: The simulated Lunar Module (LM) was initially placed at a random height between 2500 and 6000 m over the Moon surface and no control attitude was implemented. Also, the LM started its landing trip at an initial descending velocity of 300 m/s (that would be near Mach 1 speed in terrestrial atmosphere). Under these conditions, two kinematic equations describe the movement of the LM:

$$v = v_0 + g_m t$$
$$s = s_0 + v_0 t + 0.5\left(g_m t^2\right)$$

where v_0 is the initial velocity, s_0 the initial traversed space and then v and s are the resulting velocity and space after some elapsed time t. The Moon gravity acceleration, approximately equal to 1/6 of that of the Earth gravity is a constant g_m = 1.625 m/s^2. The variable mass of the LM due to fuel consumption is not considered in the model.

In this simulation, the user (the command pilot at the LM) must decide an upward thrust *p* in order to compensate both the descent speed of the LM and the pull of the Moon gravity for getting a smooth touchdown. It is assumed that a thrust *p* produces a positive and counteracting acceleration *a* on the LM. Under such conditions, and observing Fig. 8.2, the above equations transform into these ones:

$$v = v_0 + (g_m - a)t$$
$$h = h_0 - v_0 t - 0.5(g_m - a)t^2$$

The original simulation discretized descending time in 2 s intervals, that is, after a reading of height and velocity is made, the user decides how much thrust must be

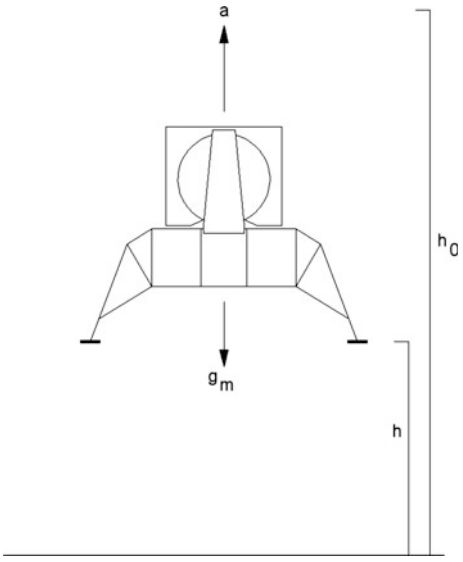

Fig. 8.2 A graphical representation of the LM descent simulation

Table 8.1 Kinematic expressions describing the LM's movement

Kinematic equations	Equivalent kinematic expressions in Lisp
$v = v_0 + 2(g_m - a)$	*(setq v (sub v (mul 2.0 (sub a 1.625))))*
$h = h_0 - 2v_0 - 2(g_m - a)$	*(setq h (add (sub h (mul 2.0 v)) (mul 2.0 (sub a 1.625))))*

applied, and then, that thrust is applied along 2 s in such a way that the above equations convert into the expressions shown in Table 8.1.

It is important to note that the thrust system (a rocket) actuates continuously in the simulation along 2 s and then waits for another new thrust command. In other words: after a thrust command is launched, nothing can stop the system actuation in the next 2 s. Fuel consumption is considered too: It is assumed that fuel consumption is directly proportional to thrust magnitude. The simulation has several possible outcomes:

- The LM gets a smooth touchdown. In this simulation, a smooth touchdown is considered when the LM final speed is less or equal than 5 m/s.
- The LM reaches the Moon at an excessive speed, that is, bigger than 5 m/s. A fatal crash is the ultimate result of this outcome.
- The LM runs out of fuel. In this case, the most probable scenario is to enter into the previous outcome after some free fall time. Neil Armstrong almost got into this outcome in the Apollo XI mission as stated in the previous chapter.

The translation of the original program written in BASIC to Lisp produces Code 8-1 as follows:

```
;code 8-1
(define (init-variables, fuel velocity height)
    (setq fuel 250.0) ;fuel units capacity
    (setq velocity 300.0) ;descent velocity in m/s
    ;generate a random height between 2500 and 6000 metres
    (while (<= height 2500)
       (setq height (rand 6001))
    )

    (println "Eagle landing on the Moon")
    (println "Fuel: " fuel " Velocity: " velocity " Height: "
             height)
    (list fuel velocity height)
)

;moon gravity, mg = 1.62519 m/s2
(define (update-variables x v h a)
    (setq v (sub v (mul 2.0 (sub a 1.625))))
    (setq h (add (sub h (mul 2.0 v)) (mul 2.0 (sub a 1.625))))
    (setq x (sub x (abs a)))
```

```
    (setq fuel x velocity v height h)
    (println "Fuel: " x " Velocity: " v " Height: " h)
    true
)

(define (simulate ,data fuel velocity height thrust)
    (setq data (init-variables))
    (setq fuel (first data) velocity (nth 1 data) height
            (nth 2 data))
    (while (and (> height 0.0) (> fuel 0.0))
        (print "\nRocket Thrust (0-25)?: ")
        (setq thrust (float (read-line)))
        (update-variables fuel velocity height thrust)
    );while end

    (if (<= fuel 0.0)
        (println "Mission failure: Out of rocket fuel.
                  Unavoidable LM crash")
    )
    (if (>= velocity 5.0)
        (println "Mission failure: Excessive velocity. LM
        crash")
    )
    (println "End simulation")
)
```

The listing of the program begins with the function *(init-variables)* that allows it to initialize the variables *fuel*, *velocity* and *height*. This last one is chosen randomly by means of the function *(rand)*, using a while loop that only ends when the initial height of the LM is placed between 2500 and 6000 m. Then the function prints these values and returns them as a list.

The function *(update-variables)* is the mathematical core in the simulation. It receives the variables representing the system status and then returns the updated values of these variables after using the expressions shown in Table 8.1. Finally, the function *(simulate)* puts it all together: Makes a call to *(init-variables)* and then enters the main event loop of the simulation where a thrust value is requested to the user and the system status is updated. The main event loop only ends when the LM reaches the Moon's surface or when the LM runs out of fuel. Finally, some *if* statements inform the user about the simulation outcome. Let us see an abbreviated run of the program:

> (simulate)
: Eagle landing on the Moon
Fuel: 250 Velocity: 300 Height: 5041
Rocket Thrust (0-25)?: 10

Fuel: 240 Velocity: 283.25 Height: 4491.25
Rocket Thrust (0-25)?:
...
...

Rocket Thrust (0-25)?: 1
Fuel: 45 Velocity: 3.75 Height: -0.12
End simulation

The reader can download Code 8-1 directly from the Book's website in order to become familiar with the program. Chances are that several crashes will be the initial outcomes of the simulation, including also some situations where all the fuel has been exhausted, especially when the LM is initially located near 6000 m high. It is not easy to land a spacecraft on the Moon!

Paraphrasing the words of John F. Kennedy at Rice University in 1962, "*We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard*", we have decided for this chapter to develop a fuzzy system able to get a smooth touchdown on the Moon surface. The next sections of this chapter will show how to build a FRBS that substitutes the human user in the previous LM simulation. Entering the realm of fuzzy control.

8.2.1 Fuzzy Control

In general, a control system is an arrangement of physical components designed to alter, to regulate, or to command, through a control action, another physical system so that it exhibits certain desired characteristics or behavior (Ross 2010). A good example of control system is the brake system in a car: the set of disks, brake pads, brake liquid, etc. is the "arrangement of physical components", while the car is the "another physical system" that must behave adequately (in this case to diminish its speed or to stop it completely). To perform the adequate "control action" we need yet a model, a complete "understanding" of the system that allows those physical components to perform the task. In the case of the car, the model for braking it is installed well inside our brain when we learn to drive. Usually in standard control theory these models are built by means of differential equations. Interestingly, traditional control engineers tended to treat their mathematical models of physical systems as exact and precise though they knew that the models were neither (Seising 2007). Take a falling body in terrestrial gravity as an example. Kinematics equations as the ones described in the previous section are exact if we forget the atmosphere and the mass of the falling body. If we take into account these factors, then differential equations are needed. If we consider also the shape of the solid, then aerodynamics enters into play and differential equations are no more an exact tool for describing the movement of the solid. Then we have no other option than using simulations at the wind tunnel. And this is a very expensive resource.

The fuzzy control model exposed in the next section is not only able to get a smooth touchdown on the Moon surface, but it would work also in the presence of an atmosphere, taking also into account the variable mass of the LM due to fuel consumption.

Fuzzy control systems satisfy completely Ross' definition, but the control action is obtained not by means of a set of mathematical equations but through the use of a Fuzzy Rule Based System (FRBS). In fact, a fuzzy control system is basically a FRBS where two especially important features are always present:

- The output of every fuzzy control system is an action.
- Every fuzzy control system uses time as a fundamental magnitude.

Control systems use sensors for measuring the desired information that describe the status of the system to control, such as velocity, temperature, density, humidity, etc. The obtained data from these sensors are the crisp inputs to a FRBS that, as shown in Fig. 7.6, ultimately produces a crisp output. In control systems this output represents an action that must be applied to the system to be controlled in order to modify o maintain its desired characteristics or behavior. Both classic control systems and fuzzy control systems result into an action, but while classic ones use mathematical equations, fuzzy based ones do use expert knowledge fuzzy rules.

After the action is applied on the system to control, the effects of the action cause changes in the behavior of the system. These changes, reflected in the system state variables, are again measured by the sensors and then a new input to the controller is created. This circular flow of information between sensor reading, controller and output action gives name to closed-loop control systems, the ones we are interested in this section. A pivotal design matter in closed-loop control systems is the concept of sampling time, the difference of time between two consecutive readings obtained from the sensors. While sampling time is usually of no importance in general FRBS, no fuzzy control system (FCS) could be developed without a clock whose ticks manage the readings of the sensors. In some cases the interval of time can extend to minutes or even hours while in other cases sampling time will be in the rank of milliseconds. The importance of sampling time will be evident after reflecting on the following example.

Imagine we are designing a fuzzy control system for an intelligent insulin pump for people suffering diabetes. Let us suppose that after a glucose reading obtained by a glucose meter the model suggests an action based in administering a subcutaneous quantity x of insulin to the patient. With a sampling time equal to 5 min another glucose reading will show about the same levels of glucose because the previously administrated insulin, including the more rapid acting types of this hormone, has not still started to show its physiological action. Then, after the second reading, the model recommends another subcutaneous quantity x of insulin. The net result is a patient receiving $2x$ units of insulin in 5 min. This probably will cause hypoglycemia on the patient and in extreme cases it could mean his/her exitus. On the other hand, a sampling time of, let us say, 1 h, probably would translate into weak control actions, producing a failure in lowering blood glucose levels towards normal values. This example suggests that determining an adequate

sampling time in a control system is one of the most important parameters of its design.

Fuzzy Control has produced such a quantity of successful results, patents and commercial products that by itself has already proven the advantages of the theory of fuzzy sets and fuzzy logic. Examples run from exact focus determination and correct exposure in cameras (Chen and Pham 2000) to automatic gearboxes in cars (Yamaguchi et al. 1993) to modern docking systems in spacecrafts (Dabney 1991), to name only a few. In fact, real practical applications of Fuzzy Logic are virtually endless today and Fuzzy Control is taught in every major college of engineering throughout the world.

In the previous chapter we already introduced an example of fuzzy control when designing an air-conditioner controller, but we did it while explaining the inner workings of a FRBS. An in depth discussion of Fuzzy Control is out the scope of this book, but the interested reader can easily find specific texts about this branch of engineering and mathematics, e.g. Passino and Yurkovich (1998). Even so, developing a control model for landing a simplified version of the LM on the Moon will show the reader more than a basic insight into fuzzy control.

8.2.2 *Controlling the Eagle*

As we stated at the beginning of Sect. 7.7, Buzz Aldrin was providing a continuous readout of LM altitude and velocity to Neil Armstrong in order to get a smooth touchdown of the Eagle on the Moon. It seems then natural to use these parameters as the input data for developing the LM fuzzy control model. Let us see the corresponding linguistic variables for each of them.

The first one, named *height*, is composed by seven fuzzy sets with the following linguistic labels: near-zero, very-small, medium, medium-high, high, very-high and extra-high. Just note that the range of this linguistic variable extends from 0 to 8000 m. It's a huge range as can be seen in Fig. 8.3, so in order to include it completely in this page the two rightmost fuzzy sets have not been drawn at scale, as suggested by the obliquely dashed lines. You also have probably realized that the supports of the corresponding fuzzy sets are shorter and shorter as we approximate

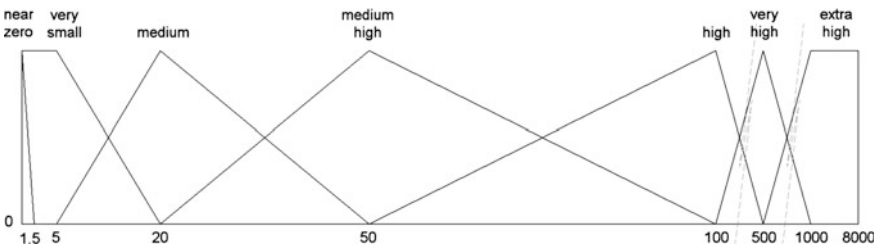

Fig. 8.3 Linguistic variable representing height of LM over the Moon's surface

to the left of the linguistic variable until arriving to a very short support fuzzy set named "near-zero". This is so because the control becomes critical when we approach the Moon surface and we shall need to have an excellent linguistic description of the model when developing its rules.

In Code 8-2a we can see this linguistic variable translated into Lisp expressions:

```
;code 8-2a
(setq h1 '(near-zero 0 0 0 1.5))
(setq h2 '(very-small 0 0 5 20))
(setq h3 '(medium 5 20 20 50))
(setq h4 '(medium-high 20 50 50 100))
(setq h5 '(high 50 100 100 500))
(setq h6 '(very-high 100 500 500 1000))
(setq h7 '(extra-high 500 1000 8000 8000))
(setq height '(h1 h2 h3 h4 h5 h6 h7))
```

The following input linguistic variable, named *velocity*, is composed again by seven fuzzy sets named very-very-small, very-small, small, medium, quick, very-quick and top (meaning top velocity). The range extends from −1.0 to 300 m/s.

As can be seen in Fig. 8.4, the fuzzy sets belonging to *velocity* are arranged in a similar way as in *height*, showing a skew towards the left of the universe of discourse where they are defined. At first it can seem strange that the leftmost fuzzy set has a support defined between −1 and 1 m/s (for clarity, the label of this fuzzy set, "very-very-small" has been omitted in the figure). This is so to cover those possible cases where the LM has suffered from excessive thrust in the final touchdown phase and is not descending but in fact is elevating over the lunar soil. The corresponding Lisp code for this linguistic variable is shown in Code 8-2b:

```
;code 8-2b
(setq v0 '(very-very-small -1.0 0.0 0.0 1.0))
(setq v1 '(very-small 0.0 0.0 0.0 5.0))
(setq v2 '(small 0.0 5.0 5.0 10.0))
```

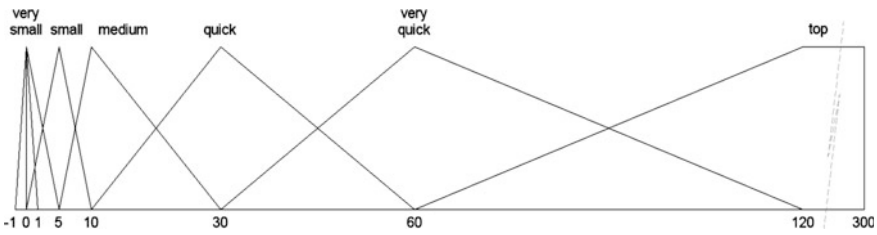

Fig. 8.4 Linguistic variable representing descent velocity of LM

```
(setq v3 '(medium 5.0 10.0 10.0 30.0))
(setq v4 '(quick 10.0 30.0 30.0 60.0))
(setq v5 '(very-quick 30.0 60.0 60.0 120.0))
(setq v6 '(top 60.0 120.0 300.0 300.0))
(setq velocity '(v0 v1 v2 v3 v4 v5 v6))
```

The output of the fuzzy control model is thrust, that is, how much gas must be ultimately expelled by the nozzle of the LM in order to modify its descent velocity. As previously assumed, a thrust p produces a positive acceleration a on the LM. The set of singletons for describing a suitable range of positive accelerations is named *thrust* and is composed by the following singletons: no-thrust (0), thrust-gently (0.6), maintain (1.625), small-minus (2.5), small (5), medium (10), strong (15) and very-strong (25), where the figures enclosed into parenthesis indicate the resulting acceleration associated to every single action.

The singleton "maintain" has a special meaning in this arrangement of singletons: The Moon gravity causes a downwards acceleration exactly equal to 1.625 m/s^2 on a falling body, so a thrust p producing an upwards acceleration a = 1.625 m/s^2 will result into a body moving with no acceleration, that is, a body moving at a constant velocity in the Moon's gravitational field. This singleton is extremely convenient for linguistically describing the expert rules of the fuzzy control model for the LM.

Additionally, it should be noted that the chosen range of accelerations would produce a relatively comfortable descending experience to our simulated team of astronauts. Having into account that 1 g (terrestrial) = 9.8 m/s^2, this means that the maximum thrust of our model is approximately equivalent to 2.55 g, less than the usual 3.0 g acceleration produced by the past Shuttle missions and even less than the stark trust of the already historical Saturn V rocket, at 4.0 g (Fortescue et al. 2011). Also important in the last phase of descent are the singletons "no-thrust" and "thrust-gently". Figure 8.5 shows the intervening singletons.

The corresponding Lisp code for this set of singletons is given by the Lisp expressions in Code 8-2c:

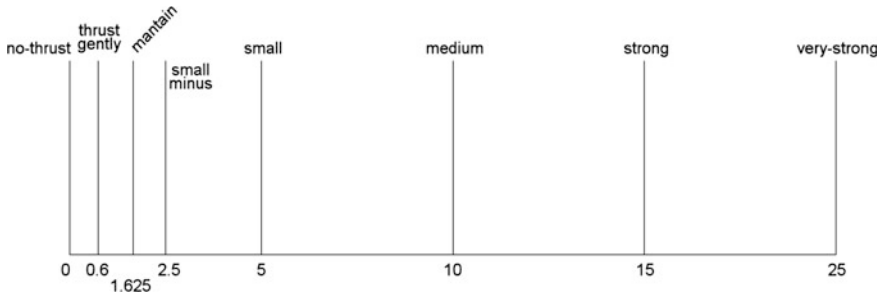

Fig. 8.5 Output singletons representing lunar module thrust

```
;code 8-2c
(setq thrust '(
    (no-thrust 0.0)
    (thrust-gently 0.6)
    (maintain 1.625) ;equivalent to moon gravity
    (small-minus 2.5)
    (small 5.0)
    (medium 10.0)
    (strong 15.0)
    (very-strong 25.0))
)
```

For building the rules of the LM fuzzy control model we combine every fuzzy set from the linguistic variable *height* with every fuzzy set contained in the linguistic variable *velocity*, associating an adequate singleton representing thrust to every rule. This results into a set of 49 expert rules that completely describes the system. Code 8-2d shows the corresponding Lisp code:

```
;code 8-2d
(setq rules-LEM '((height velocity thrust)
    (near-zero very-very-small thrust-gently AND-product)
    (near-zero very-small thrust-gently AND-product)
    (near-zero small small AND-product)
    (near-zero medium maintain AND-product)
    (near-zero quick medium AND-product)
    (near-zero very-quick very-strong AND-product)
    (near-zero top very-strong AND-product)

    (very-small very-very-small thrust-gently AND-product)
    (very-small very-small maintain AND-product)
    (very-small small small-minus AND-product)
    (very-small medium small AND-product)
    (very-small quick small AND-product)
    (very-small very-quick strong AND-product)
    (very-small top very-strong AND-product)

    (medium very-very-small thrust-gently AND-product)
    (medium very-small thrust-gently AND-product)
    (medium small maintain AND-product)
    (medium medium medium AND-product)
    (medium quick medium AND-product)
    (medium very-quick strong AND-product)
    (medium top very-strong AND-product)
```

```
    (medium-high very-very-small thrust-gently
    AND-product)
    (medium-high very-small maintain AND-product)
    (medium-high small maintain AND-product)
    (medium-high medium medium AND-product)
    (medium-high quick medium AND-product)
    (medium-high very-quick strong AND-product)
    (medium-high top very-strong AND-product)

    (high very-very-small thrust-gently AND-product)
    (high very-small thrust-gently AND-product)
    (high small maintain AND-product)
    (high medium medium AND-product)
    (high quick medium AND-product)
    (high very-quick medium AND-product)
    (high top very-strong AND-product)

    (very-high very-very-small no-thrust AND-product)
    (very-high very-small no-thrust AND-product)
    (very-high small no-thrust AND-product)
    (very-high medium maintain AND-product)
    (very-high quick maintain AND-product)
    (very-high very-quick medium AND-product)
    (very-high top very-strong AND-product)

    (extra-high very-very-small no-thrust AND-product)
    (extra-high very-small no-thrust AND-product)
    (extra-high small no-thrust AND-product)
    (extra-high medium maintain AND-product)
    (extra-high quick medium AND-product)
    (extra-high very-quick strong AND-product)
    (extra-high top strong AND-product))
)
```

When the LM is located at high altitude the rules simply try to moderate its descent velocity. If altitude becomes smaller and velocity is still near its maximum the model will react applying very strong thrust. Soon the LM falls at approximately uniform speed under the influence of the singleton named “maintain”. The most interesting phase of flight happens when the LM approaches the Moon surface. Here the first 28 expert rules of the model show their power, especially those that contain the fuzzy sets “very-very-small”, “very-small” and “small” from the linguistic variable *velocity*. In fact, under normal descent conditions, only twelve rules get the LM on the Moon.

This is interesting because the number of rules in the model could be simplified. As an example, when the LM is located at or near top height, the following rules could be eliminated:

```
(very-high very-very-small no-thrust AND-product)
(very-high very-small no-thrust AND-product)
(very-high small no-thrust AND-product)
(extra-high very-very-small no-thrust AND-product)
(extra-high very-small no-thrust AND-product)
(extra-high small no-thrust AND-product)
```

The reader can play with the model, trying to eliminate those rules that "never work", thus obtaining a more efficient set of rules, but make no mistake: the real world is full of instances where things that would never happen ultimately do happen. Depending on how critical a control application is robustness is far better than efficiency.

After building the knowledge database of the model is easy to code the simulation and fuzzy control program for the LM because we only need to take the fundamentals of code 8-1 and connect it with the fuzzy model description. Code 8-3 shows the resulting Lisp program for completing our Eagle landing simulation model:

```
;code 8-3
(define (update-variables x v h a)
   (setq v (sub v (mul 2.0 (sub a 1.625))))
   (setq h (add (sub h (mul 2.0 v)) (mul 2.0 (sub a 1.625))))
   (setq x (sub x (abs a)))
   (list h v x);returns current height, velocity and fuel
)

(define (simulate, fuel v h force list-data n)
   ;initialize variables:
   (setq n 0)
   (setq force 0.0)
   (setq fuel 500.0);fuel capacity
   (setq v 300.0);descent velocity in m/s

   (setq h 0.0)
   ;generate a random height between 2500 and 6000 metres:
   (while (<= h 2500)
      (setq h (rand 6001))
   )

   (println "Eagle landing on the Moon")
   (while (and (> h 0.0) (> fuel 0.0))
```

```
        (println "Fuel: " fuel " Height: " h " Velocity: " v)
       (print "\nPress return to continue... ")
       (read-line)

       (if (<= v -1.0)
        (setq force 0.0);if LEM elevates at v>1 m/s, apply no
        thrust
         (setq force (fl-inference rules-LEM h v))
       );end if
        (print "\nThrust by fuzzy model: " force)
       (print "\nPress return to continue... ")
       (read-line)

       (setq list-data (update-variables fuel v h force))
       (setq h (first list-data))
       (setq v (nth 1 list-data))
       (setq fuel (last list-data))
       (++ n)
    );while end

    (if (<= fuel 0.0)
       (println "Mission failure: Out of rocket fuel.
                 Unavoidable LEM crash")
    )
    (if (>= v 5.0)
       (println "Mission failure: Excessive velocity. LEM
       crash")
    )
    (println "End simulation. Final velocity: " v " n: " n)
)
```

Only these few lines of code put together all the parts:

```
(if (<= v -1.0)
 (setq force 0.0);if LEM elevates at v>1 m/s, apply no
 thrust
 (setq force (fl-inference rules-LEM h v))
);end if
```

This conditional construction manages the internals of the simulation. If the LM velocity is less than −1.0 m/s, that is, if the LM is elevating, then no thrust is applied and the model will wait until the Moon gravitational field produces again a positive descending speed. Otherwise, a call to the function *(fl-inference)* is made and the crisp output of the fuzzy control module is transferred to the variable force. Trying the program at the Lisp prompt is easy, pressing the return key for every iteration:

```
> (simulate)
: Eagle landing on the Moon
Fuel: 500 Height: 3000 Velocity: 300
Press return to continue...

Thrust by fuzzy model: 15
Press return to continue...
Fuel: 485 Height: 2480.25 Velocity: 273.25

...

Press return to continue...
Thrust by fuzzy model: 2.432348398
End simulation. Final velocity: 0.8998958331 n: 40
```

When the initial height is h_0 = 3000 m the final landing velocity on the Moon obtained by the fuzzy control mode is less than 1.0 m/s, thus meeting the NASA real standards for the LM (Klump 1971). Other initial height values can result in slightly higher final landing speeds. The reader is encouraged to improve the model using two strategies: modifying the knowledge database (fuzzy sets and/or set of fuzzy rules) or improving the resolution of sampling-time from 2 to 1 s. Changing the sampling time will need a redesign of the knowledge database. Moreover, the sampling time used on the Apollo missions for the LM was bounded between 1 and 2 s and some lags in the systems had to be had into account. I think the use of fuzzy control theory to the complete LM landing procedures on the Moon is fascinating and a project on this matter would be a technical delight to the interested reader. For this undertaking it is nice to know that NASA provides enough documentation on the Internet (NASA 2015).

8.2.3 Interpreting Results

The first thing to note, aside the final velocity at the end of the simulation, is the number of iterations produced in the main event loop. When falling from 3000 m high, the number of iterations is n = 40, so, having into account every loop pass equals to 2 s this results into a descent flight time of almost one minute and a half.

When running the simulation the user can numerically observe the actual height, descent velocity and remaining fuel. After some simulations are made it is difficult to have a clear picture of the system's behavior, so obtaining a chart where these magnitudes are plotted is a good way to gain more knowledge of the system. NewLisp incorporates a complete set of graphical functions but since this would translate into more complex code in the book we shall use another strategy: to slightly transform Code 8-3 in order to redirect the program's output to disk instead to the console and then use a standard spreadsheet program to plot the file contents. Code 8-4 shows the transformation to be done inside the function *(simulate)*:

```
;code 8-4
(device (open "LEM-OUT.csv" "write"))
(println "Fuel;Height;Velocity;Thrust")
(println fuel ";" h ";" v ";" force)

(while (and (> h 0.0) (> fuel 0.0))
    (if (<= v -1.0)
      (setq force 0.0)
      (setq force (fl-inference rules-LEM h v))
    )
    (setq list-data (update-variables fuel v h force))
    (setq h (first list-data))
    (setq v (nth 1 list-data))
    (setq fuel (last list-data))
    (println fuel ";" h ";" v ";" force)
    (++ n)
);while end
(close (device))
```

As the reader can see, it is only a question of conveniently using the functions *(open)*, *(print)* and *(close)*. Thus, the values of fuel, height, velocity and thrust are written to a file named LEM-OUT.csv that can easily be imported in many spreadsheet applications. From this point, graphically representing the output data is straightforward. Figure 8.6 shows thrust along descent time from the fuzzy control model:

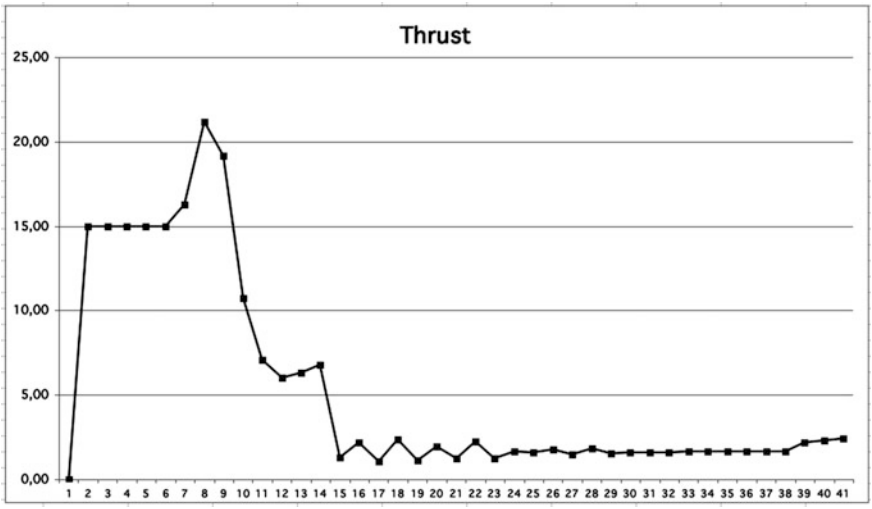

Fig. 8.6 Thrust along time as calculated by the LM fuzzy control descent model

As can be seen the model starts the descent flight applying no thrust and then, just in the second main loop pass of the simulation it produces an interval of continuous thrust at p = 15, raising it until reaching a peak about p = 22 and then initiating a smooth thrust descent that ends at values of p around 6-7. Later it still falls until stabilizing at p values between 1 and 3. The effects of this trust on the rest of LM system variables can be appreciated on Figs. 8.7, 8.8 and 8.9.

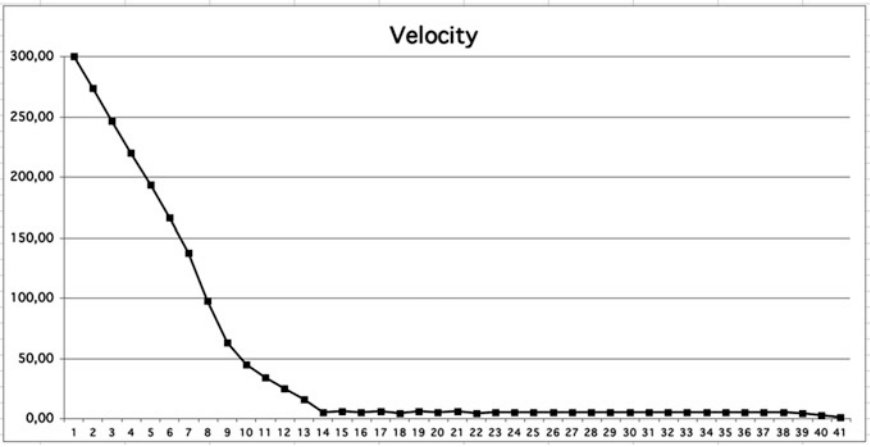

Fig. 8.7 LM obtained descent velocity along time

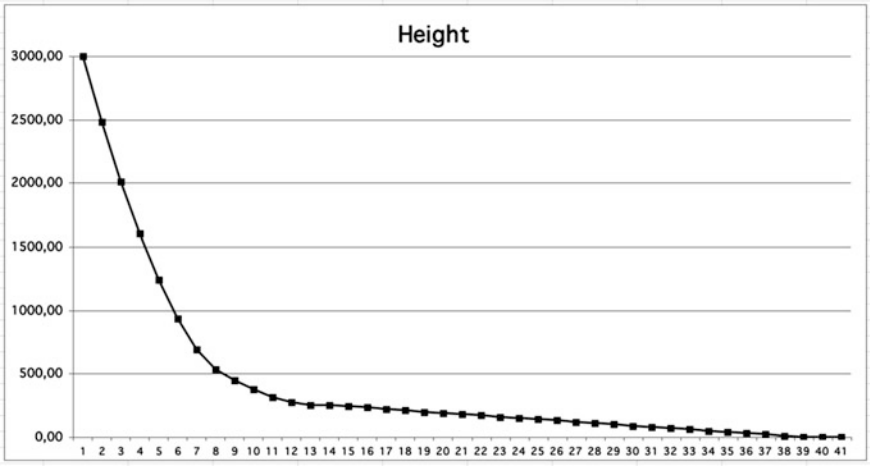

Fig. 8.8 Height over the Moon surface along descent time

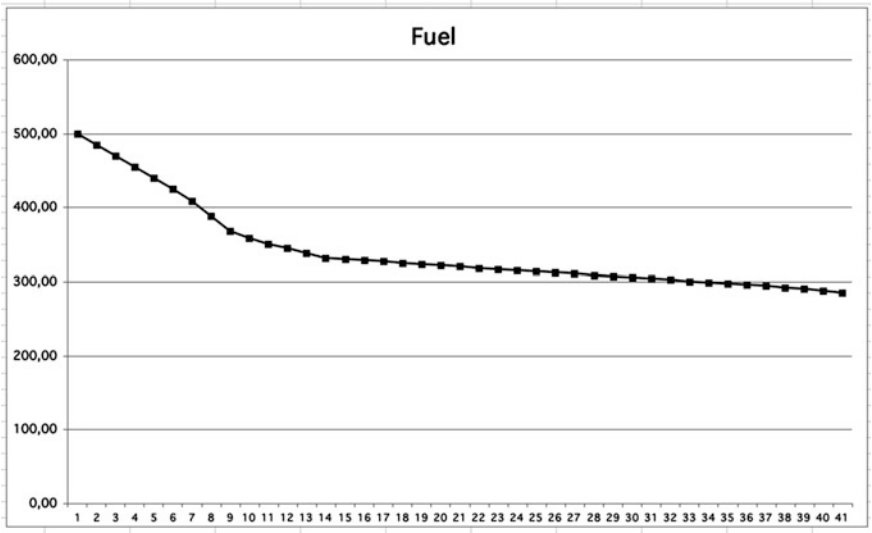

Fig. 8.9 Fuel consumption in fuel units along descent time

It is interesting to note how the control model gets initially a quick decrease of descending velocity and then it limits itself to maintain a virtually constant and moderate descent speed. A final slight increase in thrust produces a smooth, soft lunar landing. Fuel consumption is relatively low because the initial height. Consumption increases, obviously, for higher initial heights.

8.3 Double Stars in Astronomy: Speech Synthesis

When we observe the night sky from a site free from light pollution we can enjoy a wonderful sight formed by thousands of stars. It is not a generally known fact, but about half of them are double stars, that is, a set of two or more stars that orbit around a common center of masses. At the naked eye these systems appear as a single point of light, but using a telescope many of them can be split through the eyepiece, thus revealing their true double nature. These celestial objects result from the gravitational collapse of interstellar molecular clouds composed mainly by hydrogen that, due to initial inhomogeneities in density in the cloud, suffer a process of fragmentation. During this process, sections of the cloud collapse locally, forming protostars that eventually evolve into a set of two or more stars that orbit their common centre of masses with periods ranging from a few tens to a few million of years. Since aside metallicity, the main parameter for the life evolution of a star is its initial mass, such processes determine not only the separation and rotational period of the system, but also the spectral type of the components (Ostlie and Carrol 2006).

In strong contrast to optical pairs, i.e., apparent double stars that are simply formed due to line-of-sight coincidence, visual binaries are true gravitationally bounded pairs located in our galaxy. When observed through a number of years, they show different values of relative angle and angular separation between the stellar components, resulting into a curve when these parameters are plotted (Argyle 2004). In this section we are especially interested in visual binaries whose angular separation is equal or more to 1 arcsecond. With the exception of extremely good observing sites in the world such as the Canary Islands or Hawaii, this is usually the practical resolution limit of an optical telescope placed on Earth due to the atmosphere steadiness (well, lack of it), usually known in observational astronomy as "seeing".

Speech synthesis is, simply put, a computer-generated simulation of human speech that requires two phases: the first one consists in the grammatical generation of text on a certain subject matter and the second one is the audible generation of the obtained text. We are interested in the grammatical generation of double star reports using as input several intrinsic numerical parameters of this type of celestial objects. These parameters are: Separation (d), Visual magnitude of components (Mv_1, Mv_2) and Spectral class of every star ($s1$, $s2$). The aperture D in inches of the telescope is also taken into account to produce a human-like observing report. Expressing it formally, we are going to design a fuzzy logic based model F such as:

$$F(d, Mv_1, Mv_2, s1\ s2, D) \rightarrow Text\ report$$

As an example, when using Albireo, Beta Cygni, as the target double star ($d = 34.3$ arcseconds, $Mv_1 = 3$, $Mv_2 = 5.5$, $s_1 = \text{K3}$, $s_2 = \text{B9}$) observed through a four inches aperture telescope, the report generation obtained from our model F will be:

"Albireo is a really open double star that can be very easily split using any observing instrument. Difference of coloration is rather easy to observe and from an aesthetic point of view is a jewel in the sky. There is a medium difference of magnitude between components. The primary is a bright star, while the secondary is medium bright"

In order to obtain this type of speech synthesis we shall use a concatenation approach, that is, we shall use a fixed pattern of text containing several empty "pockets" where variable text will be instantiated depending on the numerical parameters from a given double star. The text structure will be the following one:

"**str1** is a **str2** double that can be **str3** split using **str4**. Difference of coloration is **str5** and from an aesthetic point of view is **str6**. There is a **str7** difference of magnitude between components. The primary is a **str8** star, while the secondary is **str9**"

The first variable string of text, **str1**, is the name of the double star. The rest of variable strings, **str2**, **str3**, **str4**, **str5**, **str6**, **str7**, **str8** and **str9** will be obtained from numerical values. Some of the numerical parameters of a double star produce a direct translation into text, such as angular distance between components or their visual magnitude. However some other information as difficulty to observe the celestial object in its true double nature or the perceived beauty of the stellar system at the eyepiece are subjective. Fuzzy logic based models are, as we already know, very well suited to this task (Argüelles and Trivino 2013).

In Chap. 7 we have used the Takagi-Sugeno (TK) model for explaining Fuzzy Rule Based Systems (Takagi and Sugeno 1985). Although the TK model was initially developed for Fuzzy Control Systems, it is of general application in FRBS when the required output is not an action but a numerical description about a certain system. Until now we have built models with two crisp inputs and a crisp output. It is time now to see how we can aggregate multiple inputs in FRBS and how we can use numerical output as new input to other FRBS. For building our intended double star speech synthesis system we shall use the architecture shown in Fig. 8.10.

The meaning of the complete set of numerical parameters (enclosed in boxes in the figure) is as follows:

- Separation: When astronomers speak about separation between stars in a double system they are not using kilometers or miles, but arcseconds, that is, an angular separation expressed in seconds of arc. Every degree has 60 min of arc, each of them divided again into 60 s of arc. Thus an arcsecond equals to 1/3600th of a degree. This angular distance is certainly small and is approximately equivalent to the view obtained from a one Euro coin located 5 km away from us. Measuring double stars was a frenetic astronomical activity back in centuries XVIII and XIX, and still today many amateur astronomers continue to systematically measure doubles stars. There is an important reason behind this task: Traditionally, the only way to discover the mass of stars is by means of observing double stars (Argyle 2004).
- Primary magnitude: In astronomy, the term "magnitude" refers to bright. The smaller the number, the brighter a star is perceived by an observer's eye. As a reference, our Sun has a −27 magnitude, being obviously the brighter celestial

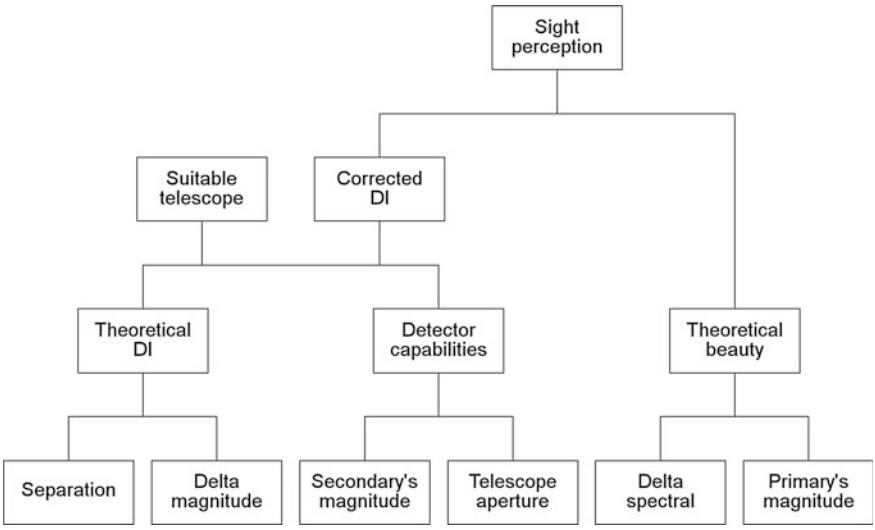

Fig. 8.10 Logical architecture for building double star speech synthesis reports

object as seen from Earth. Sirius, the brighter star from the North Hemisphere aside the Sun has a magnitude of −1.4. The scale is logarithmic and every unit in magnitude scale means about 2.512 times difference in bright from two stars. Under a really dark observing place the naked eye limit is established around magnitude 6.5, that is, fainter stars are not visible without using an observing instrument such a pair of binoculars or a telescope. To make things a bit more complex, astronomers distinguish between absolute and apparent magnitude. We are concerned, as catalogues of double stars are, only to apparent magnitude, that is, the bright of a star observed from Earth. In double stars terminology "primary magnitude" refers to the brightest star of the pair.

- Secondary magnitude: As the reader can infer from the above, this refers to the apparent magnitude of the fainter star of the pair.
- Delta magnitude: It is the numerical subtraction between primary and secondary magnitude values.
- Telescope aperture: The diameter of a telescope is the most important numerical parameter of these observing instruments because it determines both their maximum theoretical angular resolution and their limiting magnitude. As stated several paragraphs above, the usual practical observing resolution limit for a telescope placed on Earth is about one arcsecond. Observatory first class instruments such as the 10.4 m GRANTECAN Telescope in Canary Islands can observe celestial objects as faint as magnitude 25. The Hubble space telescope, entirely free from atmospheric limitations, is able to reach magnitude 31, although it must be said that observations made with these telescopes are made with electronic detectors.
- Delta spectral: We could extend pages about the concept of spectral class in stars but suffice is to know that when we speak about spectral class we are referring mainly to the temperature of the atmosphere of a star, that is, the temperature it has in its external surface. From the hottest to the colder stars astronomers have created a crisp classification based on capital letters: O, B, A, F, G, K, M (every astronomy student knows the mnemonics to learn it: "Oh, Be A Fine Girl, Kiss Me"). With the discovery of even colder, Brown Dwarfs stars the International Astronomical Union, IAI, has extended this classification to the letters L, T (the students have extended the mnemonics, too: "Late Tonight") but we shall not include them in our FRBS model. Since this is a crisp classification and Astronomy is a very traditional science, the observation of a continuum of spectral classes in stars had brought a division of ten subclasses for every spectral class, numerated from 0, the hottest star in the subclass, to 9, the colder one. As an example, the G spectral class is divided into G0, G1, G2, G3, G4, G5, G6, G7, G8 and G9 (our Sun is a G2 star, by the way). The hotter a star is, the bluer it seems to our eyes while on the other hand the colder a star is, the redder it appears rendered in the sky. In other words: O and B stars appears blue to us, while K and M stars seem orange-red and red to our eyes, depending also on the color sensitivity of the observer. With "Delta spectral" we mean the numerical distance in spectral classes between the components of a double star.

For example if the main component is B1 and the companion is B8, its Delta spectral, B8–B1 equals to 7.

- Theoretical DI (Difficulty Index), TDI: When using an optical telescope for observing double stars, some of them are more easy to split at the eyepiece into their components that others. Two numerical quantities describe mainly the degree of difficulty to visually get the split: angular separation and delta magnitude. Fuzzy logic has proven to be a valuable tool for assessing such a difficulty by means of a number in the closed interval [0–100] (Argüelles 2011). As we can see in Fig. 8.10, the theoretical DI builds on two previous parameters. In the Computational Theory of Perceptions (Zadeh 1999, 2001) such an element from the architecture shown is called a second order perception. All the previous parameters, which can be obtained directly by means of measurable data, are called first order perceptions. This and the next parameters are called second order perceptions.
- Detector capabilities: The Theoretical DI for a double star is intrinsic to the system and it does not dependent on the used observing instrument. Depending on the telescope's aperture and how dim is the secondary star in the pair, a factor K is obtained from a dedicated FRBS.
- Corrected DI, CDI: This parameter is obtained by the simple equation *Corrected DI = K x Theoretical DI*. This corrected DI has into account the aperture of the used telescope in the observation of a given double star.
- Suitable telescope aperture: This parameter is the output result of a dedicated FRBS that takes into account (inputs) both the Theoretical DI and the Detector capabilities. It will give the user a recommendation about a suitable size of telescope to observe a given double star.
- Theoretical beauty: Build upon the primary magnitude and delta spectral of the celestial objects under study, the theoretical, intrinsic beauty of a double star is obtained by another FRBS using the expert knowledge of seasoned double star observers.
- Sight perception (final sight): This is obtained by the last FRBS in the model, combining Theoretical beauty and Corrected Difficulty Index.

As we can appreciate, all the architecture shown in Fig. 8.10 can be described as a dance of numerical values that starts with first order perceptions and evolves into second order perceptions by means of several FRBS. After all the computations are made then it will be time for translating numbers into natural language descriptions.

8.3.1 Generating Suitable Linguistic Variables

Angular separation is the first parameter that comes to mind when speaking about double stars. In our model we deal with a numerical range from 1 to 100 arcseconds. When the angular separation between two stars in a double is big, it does not matter a difference of one or 2 arcseconds for getting an easy split at the eyepiece,

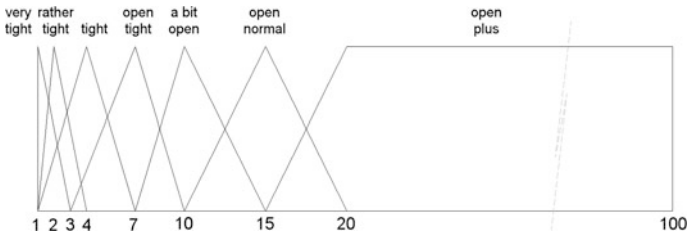

Fig. 8.11 Graphical representation of the linguistic variable "separation"

but when dealing with tight pairs, a difference of only 0.1 arcseconds can become critical. Wilhelm Struve, one of the greatest double stars observers from all times, experienced this phenomenon while making a classification for doubles in his *Catalogus Novus* (1827) depending on separation: Type I for doubles with less than 4″ separation, Type II for doubles between 4″ and 8″, Type III between 8″ and 16″ and Type IV for separations between 16″ and 32″. He realized the problems arising for such a sharp classification and again divided Type I into *Vicinae*, *Pervicinae* and *Vicinissimae*. This is, ultimately, the reason because the linguistic variable representing angular separation "compresses" its fuzzy sets towards left, as shown in Fig. 8.11.

Expressed into Lisp, this linguistic variable is expressed by Code 8-5a:

```
;code 8-5a: angular separation in double stars
(setq s1 '(very-tight 1.0 1.0 1.0 3.0))
(setq s2 '(rather-tight 1.0 2.0 2.0 4.0))
(setq s3 '(tight 1.0 4.0 4.0 7.0))
(setq s4 '(open-tight 3.0 7.0 7.0 10.0))
(setq s5 '(a-bit-open 7.0 10.0 10.0 15.0))
(setq s6 '(open-normal 10.0 15.0 15.0 20.0))
(setq s7 '(open-plus 15.0 20.0 100.0 100.0))
(setq separation '(s1 s2 s3 s4 s5 s6 s7))
```

An important remark on the used Lisp structure for representing fuzzy sets must be made now. Maybe it has been an itch for the reader from chapter five of this book. Let us have a look at the first fuzzy set in Code 8-5a:

```
(setq s1 '(very-tight 1.0 1.0 1.0 3.0))
```

The question is: Why not to use the following, simpler structure:

```
(setq s1 '(1.0 1.0 1.0 3.0))
```

After all, it would have allowed a more compact and efficient set of Lisp expressions not only for representing fuzzy sets and linguistic variables, but also an

easier work on programming FuzzyLisp functions. Powerful answer: Flexibility and adaptability are one of the most important factors not only in programming but also in Artificial Intelligence. At first sight, the symbol `very-tight` seems superfluous, but if we reflect a bit, we shall soon realize that as a symbol, it can point to practically anything in Lisp. It can point to an atom, a list, a list of lists, a file, an image…, you name it. Describing fuzzy sets in this form we always have a handle available for extending their functionality. Since we are now interested in linguistic descriptions of double star maybe it will not be a bad idea that these symbols point to string of text, as shown in Code 8-5b:

```
;code 8-5b
;associated linguistic descriptions to fuzzy sets in
separation:
(setq very-tight "very tight")
(setq rather-tight "rather-tight")
(setq tight "tight")
(setq open-tight "open-tight")
(setq a-bit-open "bit open")
(setq open-normal "open")
(setq open-plus "really open")
```

In this way, every fuzzy set from the linguistic variable "separation" not only stores a membership function, but also it has associated a textual description. This will be the key for speech synthesis later in this chapter.

The linguistic variable "delta-magnitude" is formed by four fuzzy sets, as shown in Fig. 8.12.

Translated into Lisp, its fuzzy sets and the associated linguistic descriptions are shown in Code 8-5c:

```
;code 8-5c: delta magnitude in double stars:
(setq d1 '(very-small 0.0 0.0 0.0 1.0))
(setq d2 '(medium 0.0 1.25 1.25 2.5))
(setq d3 '(rather-big 1.0 2.5 2.5 4.0))
(setq d4 '(very-big 2.5 5.0 9.0 9.0))
```

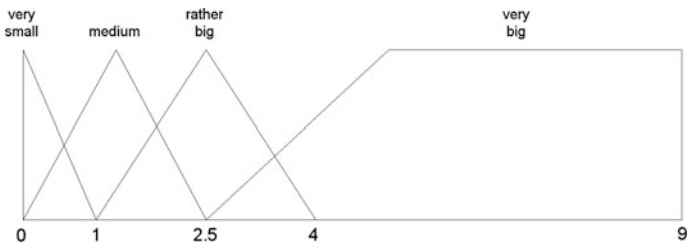

Fig. 8.12 Graphical representation of the linguistic variable "delta-magnitude"

```
(setq delta-m '(d1 d2 d3 d3 d4))
;associated linguistic descriptions:
(setq very-small "very small")
(setq medium "small")
(setq rather-big "medium")
(setq very-big "rather big")
```

Now, for obtaining the Theoretical Difficulty Index, we need first a suitable set of singletons, shown in Code 8-5d:

```
;code 8-5d: output singletons for calculating Theoretical
DI
(setq actions-di '(
      (very-easy 0.0)
      (rather-easy-plus 10.0)
      (rather-easy 20.0)
      (something-easy 40.0)
      (something-difficult 60.0)
      (rather-difficult 75.0)
      (very-difficult 100.0))
)
```

And finally, Code 8-5e shows the expert rules to complete the knowledge base for the Theoretical Difficulty Index:

```
;code 8-5e: Expert rules
(setq rules-separation '((separation delta-m actions-di)
     (very-tight very-small rather-difficult AND-product)
     (very-tight medium very-difficult AND-product)
     (very-tight rather-big very-difficult AND-product)
     (very-tight very-big very-difficult AND-product)

     (rather-tight very-small something-difficult AND-
     product)
     (rather-tight medium rather-difficult AND-product)
     (rather-tight rather-big very-difficult AND-product)
     (rather-tight very-big very-difficult AND-product)

     (tight very-small something-easy AND-product)
     (tight medium something-difficult AND-product)
     (tight rather-big rather-difficult AND-product)
     (tight very-big very-difficult AND-product)

     (open-tight very-small rather-easy AND-product)
     (open-tight medium something-easy AND-product)
```

```
        (open-tight rather-big something-difficult AND-
        product)
        (open-tight very-big rather-difficult AND-product)

        (a-bit-open very-small very-easy AND-product)
        (a-bit-open medium rather-easy AND-product)
        (a-bit-open rather-big something-easy AND-product)
        (a-bit-open very-big something-difficult AND-product)

        (open-normal very-small very-easy AND-product)
        (open-normal medium very-easy AND-product)
        (open-normal rather-big rather-easy AND-product)
        (open-normal very-big rather-easy AND-product)

        (open-plus very-small very-easy AND-product)
        (open-plus medium very-easy AND-product)
        (open-plus rather-big very-easy AND-product)
        (open-plus very-big rather-easy-plus AND-product))
)
```

Trying this FRBS at the Lisp prompt is easy. For the double star Castor, *Alpha Geminorum*, with separation $d = 2''$, primary magnitude $Mv_1 = 2.5$ and secondary magnitude $Mv_2 = 3.5$ we shall have:

> (fl-inference rules-separation 2 1)
: 79.09

This resulting value indicates us that a double star with these parameters for angular separation and difference of apparent magnitudes is a bit hard to split at the eyepiece.

Before describing the rest of linguistic variables of the complete model for speech synthesis we should comment that when complexity in programming increases, "divide and conquer" is an excellent strategy. It is thus suggested to split the knowledge database and the final FuzzyLisp application into different files, as suggested in Fig. 8.13.

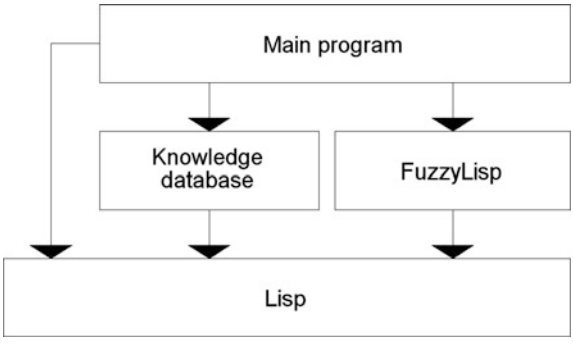

Fig. 8.13 Software engineering structure for dealing with complex models

As an example, we can name as "dstar-knowledge-db.lsp" a file where all the definitions of fuzzy sets, associated linguistic descriptions, linguistic variables and expert rules are stored and then have another file named, for example, "speech-dstar-main.lsp" where all the inferences and user interface will be implemented. Aside the comments, the first two lines of such a file should be the following ones:

```
;speech-dstar-main.lsp: A program for speech synthesis on
;double stars
(load "fuzzylisp.lsp")
(load "dstar-knowledge-db.lsp")
```

In this way, the main program can make direct calls to Lisp, to the knowledge database or to FuzzyLisp functions. Naturally, Lisp is always at the foundation base of the application, but using this structure the code is simpler to write and understand, resulting into less programming errors and thus becoming a better programming approach. After having said this, we shall continue describing the knowledge database for double stars. Now it is the turn for the FRBS that describes the Detector Capabilities of the optical system.

When we speak about "detector capabilities" we are referring to the sensibility of the human retina to light. If the secondary component of a double star is faint and the telescope's aperture is small then a given double star will be always hard to split at the eyepiece, especially if the separation between the components of a double star is small. Thus, we design another FRBS to obtain a real number K in the range [1,3]. This K value will be later multiplied to the already obtained Theoretical Difficulty Index, TDI, resulting into the so named "Corrected Difficulty Index", CDI. Let us go with the linguistic variables for this FRBS. Figure 8.14 shows the linguistic variable "secondary magnitude" describing the bright of the fainter component in a double star.

Code 8-6a shows the translation to Lisp code of this linguistic variable and the linguistic descriptions associated to its fuzz sets:

```
;code 8-6a: secondary-magnitude LV
(setq m21 '(very-bright -1.5 -1.5 -1.5 1.375))
(setq m22 '(bright -1.5 1.375 1.375 4.25))
```

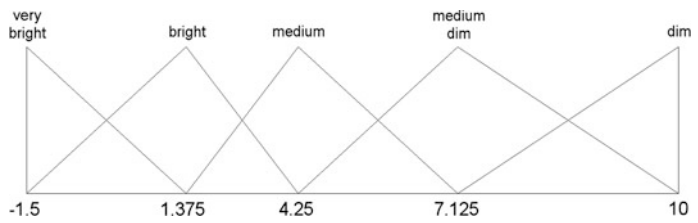

Fig. 8.14 Graphical representation of the linguistic variable "secondary-magnitude"

```
(setq m23 '(medium 1.375 4.25 4.25 7.125))
(setq m24 '(medium-dim 4.25 7.125 7.125 10.0))
(setq m25 '(dim 7.125 10.0 10.0 10.0))
(setq secondary-magnitude '(m21 m22 m23 m24 m25))
;associated linguistic descriptions to fuzzy sets in
;secondary-magnitude:
(setq very-bright "very bright")
(setq bright "bright or rather bright")
(setq medium "medium bright")
(setq medium-dim "dim")
(setq dim "rather or very dim")
```

The second linguistic variable in this FRBS is the aperture of the used telescope, expressed by its optics diameter (lens or mirror) in inches, the usual unit for expressing aperture in observational astronomy. Figure 8.15 shows it.

Code 8-6b expresses this in Lisp language. Do note that for the fuzzy sets of this linguistic variable there are no associated linguistic descriptions. Instead we shall need descriptions for recommended telescope aperture later.

```
;code 8-6b: telescope aperture LV
(setq ap1 '(a-small 2.0 2.0 2.0 4.0))
(setq ap2 '(a-medium 2.0 4.0 4.0 9.0))
(setq ap3 '(a-large 4.0 9.0 12.0 12.0))
(setq aperture '(ap1 ap2 ap3))
```

Since from this FRBS we desire to obtain a real value K in the closed interval [0–3] we shall design the output singletons accordingly, as shown in Code 8-6c:

```
;code 8-6c: Singletons for calculating detector
capability
;(expressed by a real value K)
(setq singletons-di-plus '(
     (s-low 1.0)
     (s-almost-low 1.15)
     (s-medium-minus 1.25)
```

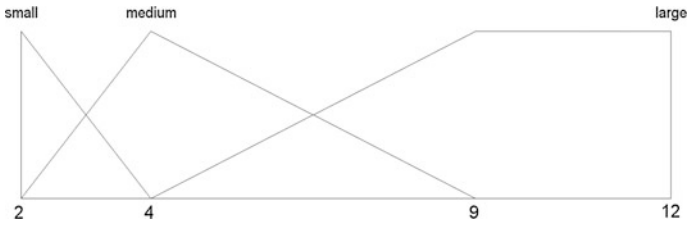

Fig. 8.15 Graphical representation of the linguistic variable “aperture”

```
      (s-medium 1.35)
      (s-medium-high 2.0)
      (s-high 3.0))
)
```

the resulting expert rules are shown in Code 8-6d:

```
;code 8-6d: Expert rules
(setq rules-di-plus
            '((secondary-magnitude aperture singletons-
             di-plus)
     (very-bright a-small s-low AND-product)
     (very-bright a-medium s-low AND-product)
     (very-bright a-large s-low AND-product)

     (bright a-small s-almost-low AND-product)
     (bright a-medium s-low AND-product)
     (bright a-large s-low AND-product)

     (medium a-small s-medium AND-product)
     (medium a-medium s-low AND-product)
     (medium a-large s-low AND-product)

     (medium-dim a-small s-medium-high AND-product)
     (medium-dim a-medium s-medium-minus AND-product)
     (medium-dim a-large s-low AND-product)

     (dim a-small s-high AND-product)
     (dim a-medium s-medium AND-product)
     (dim a-large s-almost-low AND-product))
)
```

Now, let us try this FRBS at the Lisp prompt. If we use a four inches aperture telescope for observing Castor (Mv_2 = 3.5), then we shall have:

> (fl-inference rules-di-plus 3.5 4)
: 1.0

When testing the previous FRBS, we obtained a TDI = 79.09 for the double star Castor. Taking into account the K = 1.0 value now obtained, the Corrected Difficulty Index will obviously be CDI = 79.09 × 1.0 = 79.09. The reason for maintaining the difficulty Index in this case is because Castor is a rather bright double star, so the detection capability of our retina is more than enough to observe Castor with a four inches aperture telescope. However, if we observe Castor through a 2.5 in. aperture telescope, things would change:

> (fl-inference rules-di-plus 3.5 2.5)
: 1.22

And then, with K = 1.22, the obtained CDI will be CDI = 79.09 × 1.22 = 96.76, showing that under such circumstances Castor becomes a lot harder to split because a 2.5 in. telescope gathers less stellar light (dimmer resulting image) and has less resolution than a four inches telescope.

For those cases where the resulting CDI exceeds 100.0 we shall use the following algorithm expressed in pseudo-code:

```
(calculate CDI = TDI x K)
(if obtained CDI > 100 then CDI = 100)
```

This algorithm is, as we know, certainly easy to implement, and we shall do it in the main program of this project.

For obtaining the Theoretical Beauty we shall use as inputs the Primary's magnitude, that is, the apparent visual magnitude of the brightest star in the pair, and its Delta Spectral, that is, the numerical distance between spectral lasses, as discussed in the previous section. It happens that the more contrast in coloration, that is, the bigger its delta spectral between components a double star has, the more beauty is perceived for an observer (Adler 2006). It also happens that when delta spectral is very small when the components are both bright, the resulting beauty at the telescope's eyepiece is easy to appreciate.

Translating into Lisp the concept of Primary's magnitude can't be easier. We only need a line of code for it:

```
;code 8-7a: primary magnitude LV
(setq primary-magnitude secondary-magnitude)
```

That is, we shall use the same fuzzy sets and associated linguistic descriptions as expressed by Code 8-6a (Fig. 8.14), assigning them to the symbol `primary-magnitude`. On the other hand, the linguistic variable "delta-spectral" contains five fuzzy sets defined on an universe of discourse ranging from 0 to 70, being equivalent to the numerical distance between the spectral class O0 and M9. Figure 8.16 shows graphically this linguistic variable.

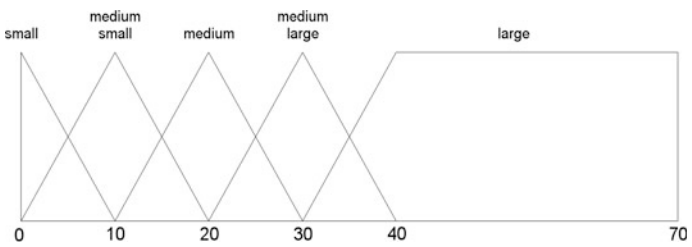

Fig. 8.16 Graphical representation of the linguistic variable "delta-spectral"

The Lisp representation of this linguistic variable is straightforward, as shown in Code 8-7b:

```
;code 8-7b: delta-spectral LV
(setq ds1 '(ds-small 0.0 0.0 0.0 10.0))
(setq ds2 '(ds-medium-small 0.0 10.0 10.0 20.0))
(setq ds3 '(ds-medium 10.0 20.0 20.0 30.0))
(setq ds4 '(ds-medium-large 20.0 30.0 30.0 40.0))
(setq ds5 '(ds-large 30.0 40.0 70.0 70.0))
(setq delta-spectral '(ds1 ds2 ds3 ds4 ds5))
;associated linguistic descriptions to fuzzy sets
;in delta-spectral:
(setq ds-small "virtually not existent")
(setq ds-medium-small "not easy at all to observe")
(setq ds-medium "neither easy not hard to observe")
(setq ds-medium-large "rather easy to observe")
(setq ds-large "easily appreciated")
```

We shall create seven output singletons for describing the Theoretical Beauty of the pair, expressed in the closed interval [0–100] as can be seen in Code 8-7c:

```
;code 8-7c: Singletons for calculating Theoretical Beauty
(setq singletons-Tbeauty '(
     (stb-top-class 100.0)
     (stb-high-plus 80.0)
      (stb-high 75.0)
     (stb-normal 50.0)
     (stb-low 25.0)
     (stb-uninteresting 0.0))
)
```

and now, the adequate expert rules for this FRBS are shown in Code 8-7d:

```
;code 8-7d: Expert rules
(setq rules-Tbeauty '((primary-magnitude delta-spectral
singletons-Tbeauty)
     (very-bright ds-small stb-top-class AND-product)
     (very-bright ds-medium-small stb-top-class AND-
     product)
     (very-bright ds-medium stb-high AND-product)
     (very-bright ds-medium-large stb-top-class AND-
     product)
     (very-bright ds-large stb-top-class AND-product)
```

```
        (bright ds-small stb-top-class AND-product)
        (bright ds-medium-small stb-high AND-product)
        (bright ds-medium stb-normal AND-product)
        (bright ds-medium-large stb-high-plus AND-product)
        (bright ds-large stb-top-class AND-product)

        (medium ds-small stb-high AND-product)
        (medium ds-medium-small stb-normal AND-product)
        (medium ds-medium stb-normal AND-product)
        (medium ds-medium-large stb-top-class AND-product)
        (medium ds-large stb-high-plus AND-product)

        (medium-dim ds-small stb-normal AND-product)
        (medium-dim ds-medium-small stb-low AND-product)
        (medium-dim ds-medium stb-low AND-product)
        (medium-dim ds-medium-large stb-uninteresting AND-
        product)
        (medium-dim ds-large stb-uninteresting AND-product)

        (dim ds-small stb-low AND-product)
        (dim ds-medium-small stb-low AND-product)
        (dim ds-medium stb-uninteresting AND-product)
        (dim ds-medium-large stb-uninteresting AND-product)
        (dim ds-large stb-uninteresting AND-product))
)
```

Following our example with Castor, its Theoretical Beauty will be obtained with the following expression at the Lisp prompt:

> (fl-inference rules-Tbeauty 2.5 1)
: 87.72

Observing again Fig. 8.10 we shall realize that there is only one FRBS to go. This one takes as inputs the obtained value of theoretical-beauty and the CDI value. The design of this FRBS is based on the idea that the theoretical-beauty value cannot be improved. That is, depending on the Difficulty Index it only can be worsened. If a double star is easy to observe the resulting Sight Perception will be the same as its theoretical-beauty value. On the other hand, if the CDI is high then the final Sight Perception will have a lower value than its corresponding theoretical-beauty. With these ideas in mind, we can create a linguistic variable for representing the theoretical-beauty, as shown in Fig. 8.17.

As usually, translating this arrangement of fuzzy sets into Lisp is easy, as shown in Code 8-8a:

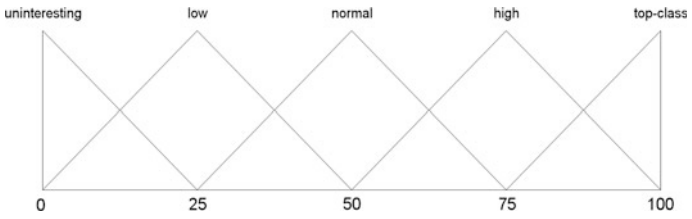

Fig. 8.17 Graphical representation of the linguistic variable "theoretical-beauty"

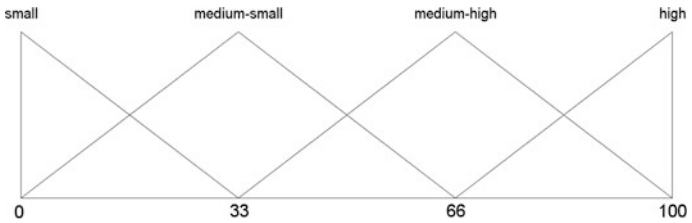

Fig. 8.18 Graphical representation of the linguistic variable "corrected DI"

```
;code 8-8a: theoretical-beauty LV
(setq tb1 '(tb-uninteresting 0.0 0.0 0.0 25.0))
(setq tb2 '(tb-low 0.0 25.0 25.0 50.0))
(setq tb3 '(tb-normal 25.0 50.0 50.0 75.0))
(setq tb4 '(tb-high 50.0 75.0 75.0 100.0))
(setq tb5 '(tb-top-class 75.0 100.0 100.0 100.0))
(setq theoretical-beauty '(tb1 tb2 tb3 tb4 tb5))
```

The other input linguistic variable, Corrected DI (CDI) can be modeled in an identical fashion, as expressed by Fig. 8.18.

Again, translating this LV into Lisp is immediate, as shown in Code 8-8b:

```
;code 8-8b: corrected-DI LV
(setq di1 '(di-small 0.0 0.0 0.0 33.0))
(setq di2 '(di-medium-small 0.0 33.0 33.0 66.0))
(setq di3 '(di-medium-high 33.0 66.0 66.0 100.0))
(setq di4 '(di-high 66.0 100.0 100.0 100.0))
(setq corrected-DI '(di1 di2 di3 di4))
```

Now we only need to create a suitable set of singletons for representing the intended output, that is, the sight-perception value, expressed also in the closed interval [0–100]. Code 8-8c shows it:

```
;code 8-8c: Singletons for calculating Sight Perception
(setq singletons-Sight '(
```

```
      (sgh-top-class 100.0)
      (sgh-high 75.0)
      (sgh-normal 50.0)
      (sgh-low 25.0)
      (sgh-uninteresting 0.0))
)
```

And finally, the rules for relating theoretical-beauty, CDI and sight perception are described in Code 8-8d:

```
;code 8-8d: Expert rules
(setq rules-Sight '((theoretical-beauty corrected-DI
singletons-Sight)
      (tb-uninteresting di-small sgh-uninteresting AND-
      product)
      (tb-uninteresting di-medium-small sgh-uninteresting
      AND-product)
      (tb-uninteresting di-medium-high sgh-uninteresting
      AND-product)
      (tb-uninteresting di-high sgh-uninteresting AND-
      product)

      (tb-low di-small sgh-low AND-product)
      (tb-low di-medium-small sgh-low AND-product)
      (tb-low di-medium-high sgh-uninteresting AND-product)
      (tb-low di-high sgh-uninteresting AND-product)

      (tb-normal di-small sgh-normal AND-product)
      (tb-normal di-medium-small sgh-normal AND-product)
      (tb-normal di-medium-high sgh-low AND-product)
      (tb-normal di-high sgh-low AND-product)

      (tb-high di-small sgh-high AND-product)
      (tb-high di-medium-small sgh-high AND-product)
      (tb-high di-medium-high sgh-normal AND-product)
      (tb-high di-high sgh-normal AND-product)

      (tb-top-class di-small sgh-top-class AND-product)
      (tb-top-class di-medium-small sgh-top-class AND-
      product)
      (tb-top-class di-medium-high sgh-high AND-product)
      (tb-top-class di-high sgh-high AND-product))
)
```

Table 8.2 Summarized results of theoretical difficult index, corrected difficulty index, theoretical beauty and sight perception for castor (Alpha Geminorum)

Theoretical DI	Corrected DI	Theoretical beauty	Sight perception
79.09	79.09	87.72	62.72

For testing this last FRBS, and for obtaining the final sight perception we shall use the obtained values of theoretical beauty = 87.72 and CDI = 79.09 at the Lisp prompt:

> (fl-inference rules-Sight 87.72 79.09)
: 62.72

Table 8.2 summarizes the obtained numerical values for the second order perceptions in our model.

8.3.2 Managing Linguistic Expressions

After obtaining the numerical values for TDI, CDI, Theoretical beauty and Sight perception it is time to develop the code for speech synthesis, taking numerical data as input and obtaining text string data as output. The key function to cross the bridge between numbers and text strings is named *(extract-description)* and is shown in Code 8-9:

```
;code 8-9
 (define (extract-description lv x,
           a-list n i location mu-value sub-list)
    (setq a-list (fl-lv-membership2? lv x))
    (setq n (length a-list) i 0 location 0 mu-value 0.0)
    (list n i)

    (while (< i n)
           (setq sub-list (nth i a-list));extracts sublist
           (if (> (last sub-list) mu-value)
           (begin
                  (setq mu-value (last sub-list))
                  (setq location i))
           );end if
           (++ i)
     );while end
     (eval (first (nth location a-list)))
)
```

This function takes a linguistic variable and a crisp value belonging to its universe of discourse as inputs and after the internal processing is done it returns the linguistic description (text string) associated to the fuzzy set with maximum membership degree for such x value. Let us try an example at the Lisp prompt in order to observe how it works in practice:

> (extract-description separation 2.7)
: "rather-tight"

While not a especially complex function by itself, it is one of the key components in the strategy for this fuzzy based model of speech synthesis. Now in Fig. 8.19 we can observe, selected with a thick vertical bar located at right in the selected boxes, what linguistic variables we are interested in for generating the final text report.

Almost all the required LVs have been already coded into Lisp with three exceptions: The difficulty to split a double star (expressed from its Corrected DI), another variable for commenting a suitable telescope (when using the linguistic variable Telescope aperture" we were referring to the telescope we actually use, the model will tell us a telescope recommendation for observing a given pair), and finally, another linguistic variable for describing the perceived final sight at the eyepiece. Code 8-10 shows the required Lisp code:

```
;code 8-10: Additional linguistic variables:
;LV-exp-diff is for expressing how hard is to split a double
star:
```

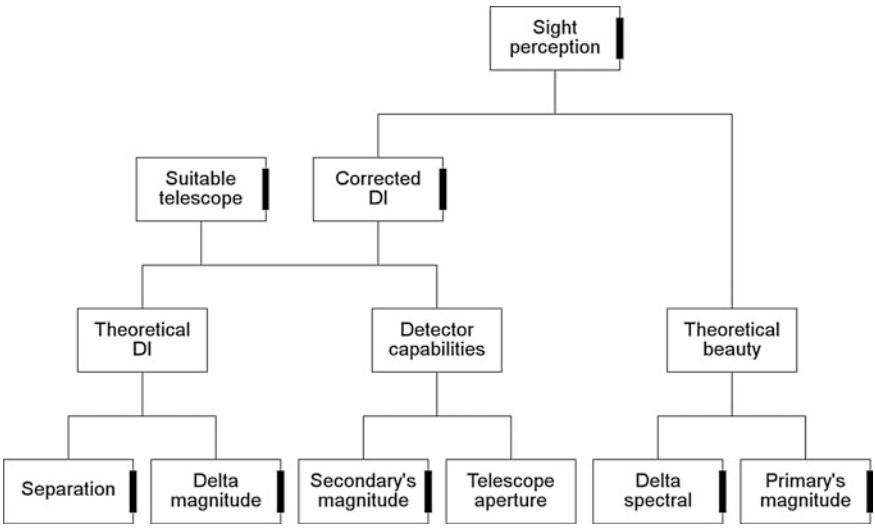

Fig. 8.19 Linguistic variables involved in the speech synthesis (those with a *vertical bar* at their right)

```
(setq exp-diff1 '(expd-very-easy 0.0 0.0 0.0 25.0))
(setq exp-diff2 '(expd-easy 0.0 25.0 25.0 50.0))
(setq exp-diff3 '(expd-not-hard 25.0 50.0 50.0 75.0))
(setq exp-diff4 '(expd-moderate 50.0 75.0 75.0 100.0))
(setq exp-diff5 '(expd-hard 75.0 100.0 100.0 100.0))
(setq LV-exp-diff
     '(exp-diff1 exp-diff2 exp-diff3 exp-diff4 exp-diff5))
;associated linguistic descriptions to fuzzy sets in
LV-exp-diff:
(setq expd-very-easy "very easily")
(setq expd-easy "easily")
(setq expd-not-hard "not especially hard to")
(setq expd-moderate "hard to")
(setq expd-hard "hardly")

;LV-exp-telescope is for expressing a suitable telescope:
(setq exp-tel1 '(expt-very-easy 0.0 0.0 0.0 25.0))
(setq exp-tel2 '(expt-easy 0.0 25.0 25.0 50.0))
(setq exp-tel3 '(expt-not-hard 25.0 50.0 50.0 75.0))
(setq exp-tel4 '(expt-moderate 50.0 75.0 75.0 100.0))
(setq exp-tel5 '(expt-hard 75.0 100.0 100.0 100.0))
(setq LV-exp-telescope
     '(exp-tel1 exp-tel2 exp-tel3 exp-tel4 exp-tel5))
;associated linguistic descriptions to fuzzy sets in
LV-exp-diff:
(setq expt-very-easy "any observing instrument")
(setq expt-easy "small telescopes")
(setq expt-not-hard "small and medium-size telescopes")
(setq expt-moderate "good telescopes under good seeing
skies")
(setq expt-hard "only first-class instruments under ex-
cepcional seeing")

;LV-final-sight is for expressing the final observed sight
at the eyepiece:
(setq fsight1 '(boring 0.0 0.0 0.0 20.0))
(setq fsight2 '(boring-plus 0.0 20.0 20.0 40.0))
(setq fsight3 '(snormal 20.0 40.0 40.0 60.0))
(setq fsight4 '(snormal-plus 40.0 60.0 60.0 80.0))
(setq fsight5 '(good 60.0 80.0 80.0 100.0))
(setq fsight6 '(very-good 80.0 100.0 100.0 100.0))
(setq LV-final-sight
     '(fsight1 fsight2 fsight3 fsight4 fsight5 fsight6))
;associated linguistic descriptions to fuzzy sets in
LV-final-sight:
```

```
(setq boring "a frankly uninteresting double")
(setq boring-plus "a not especially interesting pair")
(setq snormal "a rather normal double")
(setq snormal-plus "a beautiful pair")
(setq good "a top-class double")
(setq very-good "a jewel in the sky")
```

Now, for example:

> (extract-description LV-final-sight 87)
: "a top-class double"

We are certainly in the right track for the intended speech synthesis model, but there is still an important detail that we must solve: Spectral classes are given in a combination of letters and numbers, such as B4 or G2 and we require a numerical value for the "Delta spectral" component in the model. For solving this question, we shall first enumerate all the possible spectral classes, as shown in Code 8-11a:

```
;code 8-11a: Lisp symbol for representing spectral classes:
(setq s-classes '(
"O0""O1""O2""O3""O4""O5""O6""O7""O8""O9""B0""B1""B2""B3""B4"
"B5""B6""B7""B8""B9""A0""A1""A2""A3""A4""A5""A6""A7""A8""A9"
"F0"“F1""F2""F3""F4""F5""F6""F7""F8""F9""G0""G1""G2""G3""G4"
"G5""G6""G7""G8""G9""K0""K1""K2""K3""K4""K5""K6""K7""K8""K9"
"M0""M1""M2""M3""M4""M5""M6""M7""M8""M9"))
```

and now, Code 8-11b shows a function named *(find-class-diff)* that takes two spectral classes expressed as strings and then returns its numerical distance:

```
;code 8-11b: returns numerical distance between two spec-
tral classes
   (define (find-class-diff class1 class2, p q)
     (setq p (find class1 s-classes))
     (setq q (find class2 s-classes))
     (abs (- p q))
)
```

As can be seen, this is just another simple, yet powerful function. Let us try it:

> (find-class-diff "A2" "B3")
: 9

Almost all the ingredients for the final model are now ready. We still need an user input function for asking the required numerical dataset. Function *(data-in)* in Code 8-12 takes charge of it:

```
;code 8-12: Entering input data from the keyboard
(define (data-in, star-name d m1 m2 sp1 sp2 a)
(print "Name of the star: ")
            (setq star-name (read-line))
(print "Separation between components: ")
            (setq d (float (read-line)))
(print "Magnitude first component: ")
            (setq m1 (float (read-line)))
(print "Magnitude second component: ")
            (setq m2 (float (read-line)))
(print "Spectral class first component: ")
            (setq sp1 (read-line))
(print "Spectral class second component: ")
            (setq sp2 (read-line))
(print "Telescope aperture in inches: ")
            (setq a (float (read-line)))

      (list star-name d m1 m2 sp1 sp2 a)
)
```

This function is so simple that it does not need further explanation. Only to comment that it returns a list formed by the data gathered in the input process.

Now it is time to write another function for running all the FRBS in the model at a time. This function, named *(run-the-FRBS)* is shown in Code 8-13:

```
;code 8-13: running all the FRBS in the model
(define (run-the-FRBS, data-list sp-diff delta-mag DI K
DI-OK TB)
     (setq data-list (data-in))
     (setq sp-diff (find-class-diff (nth 4 data-list)
            (nth 5 data-list)))
     (setq delta-mag (sub (nth 3 data-list) (nth 2 data-
           list)))

     ;calculate Difficulty Index:
     (setq DI (fl-inference rules-separation (nth 1 data-
     list)
       delta-mag))

     ;calculate K factor:
     (setq K (fl-inference rules-di-plus (nth 3 data-list)
            (nth 6 data-list)))

     ;calculate corrected DI:
     (setq DI-OK (mul DI K))
```

```
      (if (> DI-OK 100.0)
             (setq DI-OK 100.0)
      )
      ;calculate theoretical Beauty:
      (setq TB (fl-inference rules-Tbeauty (nth 2 data-list)
      sp-diff))

      ;calculate final Beauty (Sight):
      (setq FB (fl-inference rules-Sight TB DI-OK))

      ;return numerical results as a list:
      (append data-list (list DI DI-OK TB FB))
)
```

The function starts calling *(data-in)* and then returns a list formed by all the required information for start the process of pure speech synthesis. The output is a list whose elements are structured in the following way: Name of the double star, separation, primary's magnitude, secondary's magnitude, spectral class of primary star, spectral class of secondary, used telescope aperture, TDI and CDI. Let us try it!

> (run-the-FRBS)
: Name of the star: Castor
Separation between components: 2
Magnitude first component: 2.5
Magnitude second component: 3.5
Spectral class first component: A1
Spectral class second component: A2
Telescope aperture in inches: 4
("Castor" 2 2.5 3.5 "A1" "A2" 4 79.09090909 79.09090909 87.7173913 62.7173913)

Now we are only one step away from finishing our speech synthesis model. The function (print-report) puts it all together, calling *(run-the-FRBS)* and then making successive calls to *(extract-description)* in order to finally compose the output linguistic report. Code 8-14 shows this main function:

```
;code 8-14: main function
(define (print-report,
           numerical-data sep mv1 mv2 spc1 spc2 apert
           TDI CDI TB FS str1 str2 str3 str4 str5 str6
           str7 str8 str9)
   (setq numerical-data (run-the-FRBS))
   (setq str1 (nth 0 numerical-data));get star's name
   (setq sep (nth 1 numerical-data))
   (setq mv1 (nth 2 numerical-data))
   (setq mv2 (nth 3 numerical-data))
```

```
    (setq spc1 (nth 4 numerical-data))
    (setq spc2 (nth 5 numerical-data))
    (setq apert (nth 6 numerical-data))
    (setq TDI (nth 7 numerical-data))
    (setq CDI (nth 8 numerical-data))
    (setq TB (nth 9 numerical-data))
    (setq FS (nth 10 numerical-data))

    ;obtain the strings:
    (setq str2 (extract-description separation sep))
    (setq str3 (extract-description LV-exp-diff CDI))
    (setq str4 (extract-description LV-exp-telescope CDI))
    (setq str5 (extract-description delta-spectral
       (find-class-diff spc1 spc2)))
    (setq str6 (extract-description LV-final-sight FS))
    (setq str7 (extract-description delta-m (abs (sub mv1
    mv2))))
    (setq  str8  (extract-description  primary-magnitude
    mv1))
    (setq str9 (extract-description secondary-magnitude
    mv2))

     ;finally, concatenate the output report:
     (println str1 " is a " str2 " double star that can be " str3
            " split using " str4 ". Difference of coloration
            is " str5 " and from an aesthetic point of view
            is "
            str6 ". There is a " str7 " difference of magnitude
            between components. The primary is a " str8 "
            star, while the secondary is " str9 "."
      );end println
)
```

Now let us enjoy its working at the Lisp prompt:

> (print-report)
:Name of the star: Castor
Separation between components: 2
Magnitude first component: 2.5
Magnitude second component: 3.5
Spectral class first component: A1
Spectral class second component: A2
Telescope aperture in inches: 4
Castor is a rather-tight double star that can be hard to split using good telescopes under good seeing skies. Difference of coloration is virtually not existent

Table 8.3 List of selected double stars for testing the model

Star name	Constellation	Sep	Mv_1	Mv_2	Sp_1	Sp_2
Sigma Orionis	Orion	12.9	4	7.5	O9	B2
Albireo	Cygnus	34.3	3	5.5	K3	B9
O. Struve 111	Orion	2.6	6	10	A5	B5
Rigel	Orion	9.4	0	7	B8	B5
Struve 747	Orion	36	5.5	6.5	B0	B1
Mintaka	Orion	52.8	2	6.5	O9	B2
Zeta Ori	Orion	2.5	1.9	4	O9	B0
Iota Ori	Orion	11	2.8	6.9	O5	B9
Eta Ori	Orion	1.5	3.1	3.4	B1	B2
Beta Cep	Cepheus	13	3.2	7.9	B2	A2
Delta Cyg	Cygnus	2.5	2.9	6.3	B9	F1
Lambda Ori	Orion	4	3.6	5.5	O8	B0
STF 422	Taurus	7	5.9	6.8	G9	K6
Epsilon Per	Perseus	9	2.9	7.6	B0	A2
Theta 1 Ori	Orion	9	6.4	7.9	B0	B0
Algieba	Leo	4.4	2.5	3.5	K0	G7
Theta Per	Perseus	20	4.1	9.9	F7	M1
Eta Lyrae	Lyra	28	4.4	9.1	B3	B3
Castor	Gemini	2	2.5	3.5	A1	A2
Wasat	Gemini	6.3	3.5	8	F0	F0

and from an aesthetic point of view is a beautiful pair. There is a small difference of magnitude between components. The primary is a bright or rather bright star, while the secondary is medium bright.

Depending on your computer, you can copy the report into a new text document and select the "Speech" option directly from the Operating System. You will hear the report directly from your computer.

The reader can play the model with double star data from catalogues easily downloadable from the Internet or with some example data as shown in Table 8.3.

8.4 Spirometry Analysis in Medicine: Floating Singletons

Chronic Obstruction Pulmonary Disease (COPD) is a severe condition caused by long-term exposure to lung irritants such as dust or micro granular material. Since many activities in industrial processes in quarries and other extractive industries generate micro particles (silicates, coal dust, etc.), the resulting atmospheres at many industrial facilities are aggressive to the human lungs. As a consequence, many people working in such environments end up developing COPD

(Santo-Tomas 2011). It is usual that the medical services at such companies carry out periodic revisions concerning the health status of their respective working forces. Spirometric analysis is a fundamental test in such medical revisions.

An spirometry is a clinical test where a device (spirometer) records the volume of air exhaled by a patient and plots it as a function of time, producing a curve that shows the lung function, specifically the volume and flow of air that can be exhaled. The patient inhales maximally and then exhales as rapidly and completely as possible. This technique is the most common one of the pulmonary function tests, being a suitable clinical test for detecting COPD. In fact, spirometry is the only medical test able to detect COPD years before shortness of breath develops (Hyatt et al. 2009). A typical spirometric curve can be observed in Fig. 8.20.

The most important numerical parameters obtained by means of a spirometry are the following ones:

- FVC: Forced Vital Capacity: It is the whole volume of air that can be exhaled by a patient after full inspiration in the test.
- FEV1: Forced Expiratory Volume in one second: It is the volume of air that can be exhaled in the first second of a spirometric test. Both FVC and FEV1 are measured in litres.
- TI: Tiffenau Index: It is a relationship between FEV1 and FVC values given by the following expression: $TI = 100 \times (FEV1/FVC)$

From the height, x, in metres and age, y, in years of an individual, theoretical values of FEV1 and FVC for men, in litres, can be obtained using the following expressions (Morris et al. 1971):

$$Theoretical\,FVC = 5.76x - 0.026y - 4.34 \tag{8-1}$$

$$Theoretical\,FEV1 = 4.3x - 0.026y - 2.49 \tag{8-2}$$

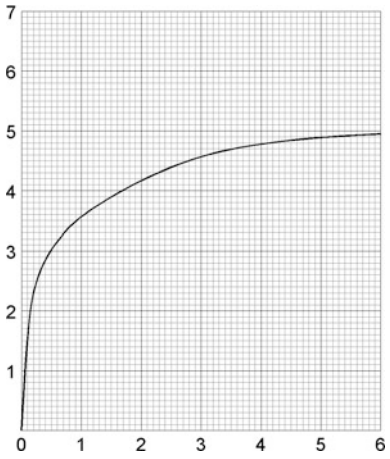

Fig. 8.20 A typical spirometry. The *horizontal axis* represents time in seconds, while the *vertical axis* represents litres of exhaled air. This spirometry show moderate COPD, with FEV1 = 3.6 l and FVC = 4.95 l (see text)

Comparing the experimental *FEV1* and *FVC* values obtained from a spirometry with the theoretical values, lung obstruction is detected applying the following, traditional, algorithm:

If ((*Observed FVC* > *80 % theoretical FVC*) and (*Observed FEV1* > *80 % theoretical FEV1*) and (*TI* > *80 %*))
then spirometry is normal
else the individual suffers an obstructive pulmonary disease.

Similar algorithms are given, for example, by Qaseem et al. (2011). However, expressions (8-1) and (8-2) are in fact equations representing two regression lines from a general population, thus being a "sharp" approximation for representing observed cases from the real world. Physicians know that the requirements for detecting COPD in workers exposed to ambient dust must be more strict than the simple application of the aforementioned algorithm, and test results must be evaluated in light of the patient's clinic history, physical examination and pertinent laboratory findings. Moreover, medical practice shows that smoking is a strong factor that worsens COPD. In this section of the chapter we are going to explore a sophisticated FRBS architecture that gathers the experience of physicians and also has into account the smoking factor. This architecture uses floating singletons (Argüelles 2013).

8.4.1 Introducing Floating Singletons

Basically speaking, floating singletons are a special type of singletons belonging to a first FRBS where their precise location on the real axis is only known at run-time. At design time some intervals are established in the first FRBS where these special singletons are allowed to float and later, the defuzzification of a complementary, second FRBS fixes their location. That is, a floating singleton, *sf*, has the ability to move inside an universe of discourse defined in a closed interval $[a,b]$. A set of traditional singletons s_i defines a set of $I_i = i - 1$ internal intervals in $[a,b]$ as can be observed in Fig. 8.21a, b:

We say that sf_i is a floating singleton by-left if:

$$s_{i-1} \leq sf_i \leq s_i \rightarrow sf_i \in I_{i+1} \tag{8-3}$$

with $sf_1 = s_1$. Conversely, sf_i is a floating singleton by-right if:

$$s_i \leq sf_i \leq s_{i+1} \rightarrow sf_i \in I_i \tag{8-4}$$

with $sf_n = s_n$ (see Fig. 8.21a, b). Now we can expose a more restricted definition: a floating singleton, sf_i, is a special type of singleton that has the ability to move inside its assigned interval I_i until run-time, that is, when the model is run at the

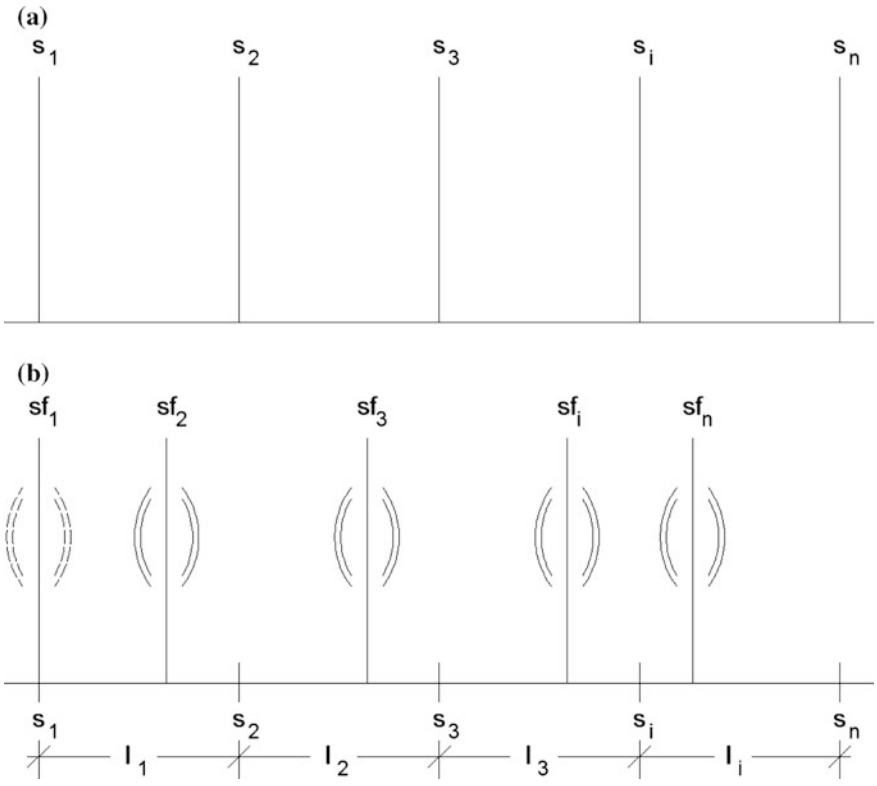

Fig. 8.21 **a**, **b** At *top*, a traditional set of singletons, s_i can be seen. At *bottom* a set of floating singletons sf_i "by-left" is shown. Note the position of traditional singletons s_i on the horizontal axis at *bottom*

computer. The floating nature of floating singletons is described by means of the existence of a second FRBS composed by fuzzy rules of the type:

$$\text{if p } is\ M_g\ and\ q\ is\ N_h\ then\ z_i = t_{i,j} \tag{8-5}$$

where $t_{i,j} \in [s_i, s_{i+1}]$ are a set of singletons $t_{i,j}$, expressed as consequents. Here we must remark that every singleton $t_{i,j}$ is defined on the interval I_i. This can be expressed graphically by means of the Fig. 8.22.

Now, the final position of every floating singleton sf_i comes from the defuzzification of the singletons t_i, expressed, as we already know, by Eq. 7-22. As can be seen, in this arrangement of models, inferences are made first for the second FRBS whose rules are given by (8-5), and then, the defuzzification of t_i is made for "fixing" the position of the floating singletons. Finally, after making the inferences for the first FRBS, the final defuzzification process is made.

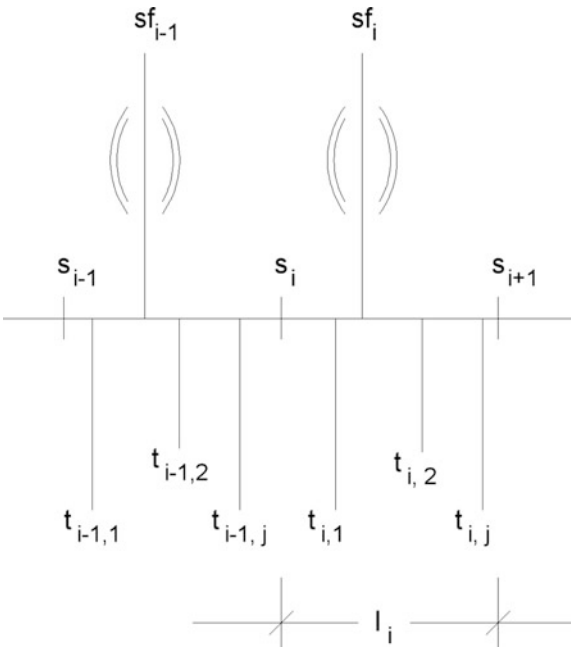

Fig. 8.22 Graphical interpretation of floating singletons sf_i, and their relationship with standard singletons $t_{i,j}$

8.4.2 Modeling Pulmonary Obstruction with FuzzyLISP

For developing a model representing pulmonary obstruction after obtaining the FVC and FEV1 values from a real spirometry, we shall first create two magnitudes, f1 and f2, expressed by:

$$f1 = 100(FVC\,from\,spirometry)/(theoretical\,FVC) \tag{8-6}$$

$$f2 = 100(FEV1\,from\,spirometry)/(theoretical\,FEV1) \tag{8-7}$$

where the theoretical FVC and theoretical FEV1 values are obtained from expressions (8-1) and (8-2), respectively. The following step is to create two linguistic variables, Pct-FVC and Pct-FEV1, for expressing all the possible values of f1 and f2. These linguistic variables will allow us to create a first FRBS composed by expert rules of the type:

$$if\ f1_i\ is\ Pct\text{-}FVC\ and\ f2_i\ is\ Pct\text{-}FEV\ then\ Obstruction\ is\ sf_i \tag{8-8}$$

The key point now is the definition of the floating singletons sf_i for describing pulmonary Obstruction. Since at design time they "float" we do not give fixed real numbers for them, but some intervals. Table 8.4a shows the floating singletons involved in our model:

As can be observed in the table, these floating singletons are expressed by a set of intervals I_i. Please do note also that the last floating singleton, "Extreme" is fixed at [100], that is, we are creating a model with floating singletons by-right. In this way, an instance from rules given by (8-8) could be:

$$if\ f1_i\ is\ Severe\ and\ f2_i\ is\ Normal\ then\ Obstruction\ is\ Important[65,85] \quad (8\text{-}9)$$

So "Important" is a floating singleton that can adopt numerical values between 65 and 85. Now, how do we get a fixed value for "Important" in such a way that we can make fuzzy logic inferences from the rules given by (8-8)? The answer comes from a second FRBS that takes into account smoking. The model will ask for the number of cigarettes/day and the number of years of smoking associated to the person passing the spirometric test. If he or she is not a smoker, both values will be null, and then Table 8.4a would convert to Table 8.4b.

Figure 8.23 shows the whole architecture of the intended model for describing pulmonary obstruction.

As can be seen, the second FRBS takes the number of smoking years and the number of cigarettes per day as crisp numerical inputs, ultimately generating an

Table 8.4a List of floating singletons for describing pulmonary obstruction

Floating singleton name	Floating interval I_i
Null	[0,15]
Appreciable	[15,65]
Important	[65,85]
Very-important	[85,100]
Extreme	[100,100]

Table 8.4b List of singletons for describing pulmonary obstruction in non smokers

Singleton name	Singleton defined at
Null	0
Appreciable	15
Important	65
Very-important	85
Extreme	100

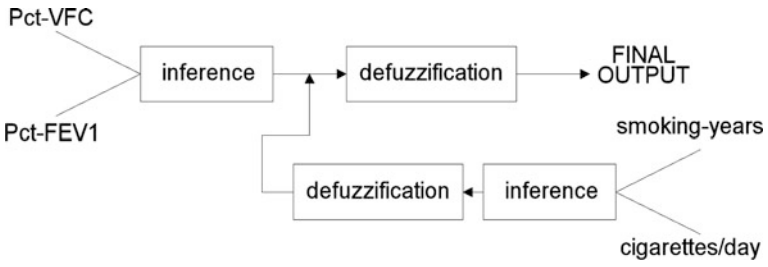

Fig. 8.23 FRBS architecture for spirometric and COPD analysis

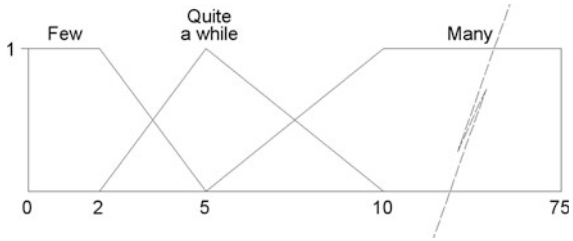

Fig. 8.24a Graphical representation of the linguistic variable "smoking-years".

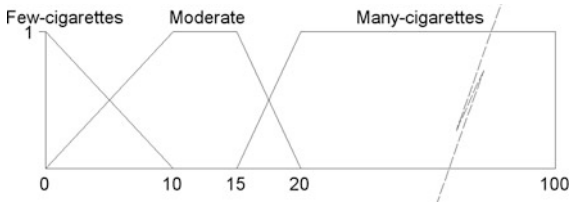

Fig. 8.24b Graphical representation of the linguistic variable "cigarettes-day"

output that converts into standard singletons, ready for being used in the defuzzification for the first FRBS. For representing the number of cigarettes/day and the number of years of smoking we shall create two linguistic variables named, respectively, "smoking-years" and "cigarettes-day". The former, composed by three fuzzy sets, is shown in Fig. 8.24a.

Figure 8.24b shows the graphical representation of the Linguistic Variable "cigarettes/day".

Translating these variables to Lisp is immediate, as can be appreciated in Code 8-15:

```
;code 8-15: Linguistic variables for floating
;singletons management
;smoking-years:
(setq sy1 '(Few 0.0 0.0 2.0 5.0))
(setq sy2 '(Quite-a-while 2.0 5.0 5.0 10.0))
(setq sy3 '(Many 5.0 10.0 75.0 75.0))
(setq smoking-years '(sy1 sy2 sy3))
;cigarettes-day:
(setq cd1 '(Few-cigarettes 0.0 0.0 0.0 10.0))
(setq cd2 '(Moderate 0.0 10.0 15.0 20.0))
(setq cd3 '(Many-cigarettes 15.0 20.0 100.0 100.0))
(setq cigarettes-day '(cd1 cd2 cd3))
```

Now we need to define the floating singletons as described in Table 8.4a. This is shown in code 8-16:

```
;code 8-16: Definition of all the floating singletons
intervals
;first floating singletons interval:
(setq Sf1 '(
     (sf1-left 0.0)
     (sf1-medium 8.0)
     (sf1-right 15.0))
)

;second floating singletons interval:
(setq Sf2 '(
     (sf2-left 15.0)
     (sf2-medium 40.0)
     (sf2-right 65.0))
)

;third floating singletons interval:
(setq Sf3 '(
     (sf3-left 65.0)
     (sf3-medium 75.0)
     (sf3-right 85.0))
)

;fourth floating singletons interval:
(setq Sf4 '(
     (sf4-left 85.0)
     (sf4-medium 92.5)
     (sf4-right 100.0))
)

;fifth floating singletons interval:
(setq Sf5 '(
     (sf5-left 100.0)
     (sf5-medium 100.0)
     (sf5-right 100.0))
)
```

For each of these intervals we shall need a dedicated set of expert rules joining the linguistic variables "smoking-years" and "cigarettes-day" in the antecedents as follows:

$$\textit{if } x_i \textit{ is } \text{smoking-years } \textit{and } y_i \textit{ is } \text{cigarettes-day } \textit{then singleton is } t_i \qquad (8\text{-}10)$$

In this way we shall be able to determine the floating singletons into standard singletons. Code 8-17 shows all the required expert rules:

```
;code 8-17: Expert rules for determining the floating
singletons:
(setq rules-s1 '((smoking-years cigarettes-day Sf1)
     (Few Few-cigarettes sf1-left AND-product)
     (Few Moderate sf1-left AND-product)
     (Few Many-cigarettes sf1-medium AND-product)

     (Quite-a-while Few-cigarettes sf1-left AND-product)
     (Quite-a-while Moderate sf1-medium AND-product)
     (Quite-a-while Many-cigarettes sf1-right AND-product)

     (Many Few-cigarettes sf1-medium AND-product)
     (Many Moderate sf1-right AND-product)
     (Many Many-cigarettes sf1-right AND-product))
)

(setq rules-s2 '((smoking-years cigarettes-day Sf2)
     (Few Few-cigarettes sf2-left AND-product)
     (Few Moderate sf2-left AND-product)
     (Few Many-cigarettes sf2-medium AND-product)

     (Quite-a-while Few-cigarettes sf2-left AND-product)
     (Quite-a-while Moderate sf2-medium AND-product)
     (Quite-a-while Many-cigarettes sf2-right AND-product)

     (Many Few-cigarettes sf2-medium AND-product)
     (Many Moderate sf2-right AND-product)
     (Many Many-cigarettes sf2-right AND-product))
)

(setq rules-s3 '((smoking-years cigarettes-day Sf3)
     (Few Few-cigarettes sf3-left AND-product)
     (Few Moderate sf3-left AND-product)
     (Few Many-cigarettes sf3-medium AND-product)

     (Quite-a-while Few-cigarettes sf3-left AND-product)
     (Quite-a-while Moderate sf3-medium AND-product)
     (Quite-a-while Many-cigarettes sf3-right AND-product)

     (Many Few-cigarettes sf3-medium AND-product)
     (Many Moderate sf3-right AND-product)
     (Many Many-cigarettes sf3-right AND-product))
)
```

```
(setq rules-s4 '((smoking-years cigarettes-day Sf4)
     (Few Few-cigarettes sf4-left AND-product)
     (Few Moderate sf4-left AND-product)
     (Few Many-cigarettes sf4-medium AND-product)

     (Quite-a-while Few-cigarettes sf4-left AND-product)
     (Quite-a-while Moderate sf4-medium AND-product)
     (Quite-a-while Many-cigarettes sf4-right AND-product)

     (Many Few-cigarettes sf4-medium AND-product)
     (Many Moderate sf4-right AND-product)
     (Many Many-cigarettes sf4-right AND-product))
)

(setq rules-s5 '((smoking-years cigarettes-day Sf5)
     (Few Few-cigarettes sf5-left AND-product)
     (Few Moderate sf5-left AND-product)
     (Few Many-cigarettes sf5-medium AND-product)

     (Quite-a-while Few-cigarettes sf5-left AND-product)
     (Quite-a-while Moderate sf5-medium AND-product)
     (Quite-a-while Many-cigarettes sf5-right AND-product)

     (Many Few-cigarettes sf5-medium AND-product)
     (Many Moderate sf5-right AND-product)
     (Many Many-cigarettes sf5-right AND-product))
)
```

Let us test at the Lisp prompt all these sets of rules for a person that started to smoke ten years ago and smokes seven cigarettes every day:

> (fl-inference rules-s1 10 7)
: 12.9

> (fl-inference rules-s2 10 7)
: 57.5

> (fl-inference rules-s3 10 7)
: 82

> (fl-inference rules-s4 10 7)
: 97.75

> (fl-inference rules-s5 10 7)
: 100

Table 8.4c List of singletons for describing pulmonary obstruction for x = 10 smoking years and y = 7 cigarettes/day

Singleton name	Singleton defined at
Null	12.9
Appreciable	57.5
Important	82
Very-important	97.75
Extreme	100

In this way, for x = 10 years and y = 7 cigarettes/day, the resulting singletons (we could say they have just finished their floating nature after inferences are made in the second FRBS) are shown in Table 8.4c.

Now we need to establish the linguistic variables Pct-FVC and Pct-FEV1, shown in Fig. 8.25. The fuzzy sets distribution in both of them is identical and only the linguistic labels do differ, as can be seen next, in the code, where the "fvc" and "fev" prefixes are used:

As usually, Code 8-18 put these definitions into Lisp:

```
;code 8-18, linguistic variables Pct-FVC and Pct-FEV1:
(setq fvc1 '(fvc-Severe 20.0 20.0 30.0 50.0))
(setq fvc2 '(fvc-Moderate 30.0 50.0 50.0 70.0))
(setq fvc3 '(fvc-Slight 50.0 70.0 70.0 85.0))
(setq fvc4 '(fvc-Normal 70.0 85.0 100.0 120.0))
(setq fvc5 '(fvc-Excellent 100.0 120.0 150.0 150.0))
(setq Pct-FVC '(fvc1 fvc2 fvc3 fvc4 fvc5))

(setq fev11 '(fev-Severe 20.0 20.0 30.0 50.0))
(setq fev12 '(fev-Moderate 30.0 50.0 50.0 70.0))
(setq fev13 '(fev-Slight 50.0 70.0 70.0 85.0))
(setq fev14 '(fev-Normal 70.0 85.0 100.0 120.0))
(setq fev15 '(fev-Excellent 100.0 120.0 150.0 150.0))
(setq Pct-FEV1 '(fev11 fev12 fev13 fev14 fev15))
```

In the first section of this book we learnt that Lisp is a great computer language where there is not a sharp frontier between code and data. A language that is even

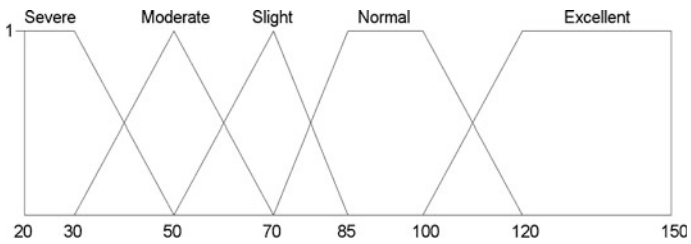

Fig. 8.25 Graphical representation of the linguistic variables "Pct-FVC" and "Pct-FEV1"

able to generate its own code. Now this is a pretty opportunity to demonstrate it. For obtaining the required inferences from the expert rules enunciated in (8-8) we now need that Lisp generate the code for describing the final singletons. It is time to introduce the most important function for our spirometric analysis model as shown in Code 8-19:

```
;code 8-19: generating code for final singletons
(define (create-floating-singletons years cgr-day,
                  s1 s2 s3 s4 s5 l1 l2 l3 l4 l5)
   (setq s5 (fl-inference rules-s5 years cgr-day))
   (setq l5 (list 'Extreme s5))
   (setq s4 (fl-inference rules-s4 years cgr-day))
   (setq l4 (list 'Very-Important s4))
   (setq s3 (fl-inference rules-s3 years cgr-day))
   (setq l3 (list 'Important s3))
   (setq s2 (fl-inference rules-s2 years cgr-day))
   (setq l2 (list 'Appreciable s2))
   (setq s1 (fl-inference rules-s1 years cgr-day))
   (setq l1 (list 'Null s1))
   (list l5 l4 l3 l2 l1)
)
```

Let us try the function *(create-floating-singletons)* using again the values x = 10 smoking years and y = 7 cigarettes/day from the previous example:

> (setq Obstruction (create-floating-singletons 10 7))
: ((Extreme 100) (Very-Important 97.75) (Important 82) (Appreciable 57.5) (Null 12.9))

Please, do appreciate how the list now stored in the symbol "Obstruction" is completely equivalent to the evaluation of the following Lisp code:

```
(setq Obstruction '(
      (Extreme 100)
      (Very-Important 97.75)
      (Important 82)
      (Appreciable 57.5)
      (Null 12.9))
)
```

Now we can finally describe in Code 8-20 the expert rules corresponding to the first FRBS, as expressed by (8-8):

```
;code 8-20
(setq rules-obstruction '((Pct-FVC Pct-FEV1 Obstruction)
      (fvc-Severe fev-Severe Extreme AND-product)
```

```
      (fvc-Severe fev-Moderate Extreme AND-product)
      (fvc-Severe fev-Slight Extreme AND-product)
      (fvc-Severe fev-Normal Important AND-product)
      (fvc-Severe fev-Excellent Important AND-product)

      (fvc-Moderate fev-Severe Extreme AND-product)
      (fvc-Moderate fev-Moderate Extreme AND-product)
      (fvc-Moderate fev-Slight Very-Important AND-product)
      (fvc-Moderate fev-Normal Important AND-product)
      (fvc-Moderate fev-Excellent Important AND-product)

      (fvc-Slight fev-Severe Extreme AND-product)
      (fvc-Slight fev-Moderate Very-Important AND-product)
      (fvc-Slight fev-Slight Appreciable AND-product)
      (fvc-Slight fev-Normal Appreciable AND-product)
      (fvc-Slight fev-Excellent Null AND-product)

      (fvc-Normal fev-Severe Extreme AND-product)
      (fvc-Normal fev-Moderate Important AND-product)
      (fvc-Normal fev-Slight Appreciable AND-product)
      (fvc-Normal fev-Normal Null AND-product)
      (fvc-Normal fev-Excellent Null AND-product)

      (fvc-Excellent fev-Severe Extreme AND-product)
      (fvc-Excellent fev-Moderate Appreciable AND-product)
      (fvc-Excellent fev-Slight Appreciable AND-product)
      (fvc-Excellent fev-Normal Null AND-product)
      (fvc-Excellent fev-Excellent Null AND-product))
)
```

All the ingredients for the design of a fuzzy-logic based model of spirometric analysis are already prepared. We now only need a final function that puts it all together. Such a function, named *(main)* incorporates a tiny user interface and is shown in Code 8-21:

```
;code 8-21. Main function for the spirometric analysis
model
(define (main, height age fvc fev1 tfvc tfev1 pctfvc pctfev1 TI
          years cigarettes Obstruction)
   (print "Height in metres: ") (setq height (float (read-
   line)))
   (print "Age in years: ") (setq age (float (read-line)))
   (print "Spirometry FVC in litres: ")
   (setq fvc (float (read-line)))
   (print "Spirometry FEV1 in litres: ")
```

```
                (setq fev1 (float (read-line)))
    (print "Years of smoking: ") (setq years (float (read-
    line)))
    (print "Number of cigarettes/day: ")
                (setq cigarettes (float (read-line)))

    ;calculate theoretical tfvc, tfev1 and then pctfvc and
    pctfev1:
    (setq tfvc (sub (sub (mul 5.76 height) (mul 0.026 age))
    4.34))
    (setq tfev1 (sub (sub (mul 4.3 height) (mul 0.026 age))
    2.49))
    (setq pctfvc (mul (div fvc tfvc) 100.0))
    (setq pctfev1 (mul (div fev1 tfev1) 100.0))
    (setq TI (mul (div fev1 fvc) 100.0))

    ;finally run the fuzzy model:
    (setq Obstruction (create-floating-singletons years
    cigarettes))
    (println "Pulmonary obstruction index: "

         (fl-inference rules-obstruction pctfvc pctfev1))

      (list tfvc tfev1 pctfvc pctfev1 TI);returns partial
      results
)
```

Let us try the model at the Lisp prompt with some real data from a spirometric test for a 32 years old man, 1.80 m height and with FVC = 4.82 l, FEV1 = 4.15 l spirometric results. We assume this individual is a ten years old smoker and smokes, as an average, 7 cigarettes/day:

> (main)
: Height in metres: 1.8
Age in years: 32
Spirometry FVC in litres: 4.82
Spirometry FEV1 in litres: 4.15
Years of smoking: 10
Number of cigarettes/day: 7
Pulmonary obstruction index: 12.9
(5.196 4.418 92.76366436 93.93390675 86.09958506)

Aside obtaining a Pulmonary Obstruction Index, *(main)* returns also intermediate results from the input data as a list. In this example these are: Theoretical FVC: 5.196, Theoretical FEV1: 4.418, %FVC: 92.76, %FEV1: 93.93 and Tiffenau

Index = 86.10. Let us quote the traditional algorithm mentioned some pages before in this chapter for evaluating spirometries:

"If ((*Observed FVC > 80 % theoretical FVC*) and (*Observed FEV1 > 80 % theoretical FEV1*) and (*TI > 80 %*))
then spirometry is normal
else the individual suffers an obstructive pulmonary disease"

Under such algorithm the spirometry in the example is normal, but the inclusion of the smoking parameters in the fuzzy model reveals what physicians know from experience: Even obtaining good spirometric results, smokers usually show a certain degree of Chronic Obstruction Pulmonary Disease. For long-time smokers a spirometry always reveals an obstructive pulmonary disease because smoking is the first cause of COPD in the world. The fuzzy model developed in this section helps physicians to reveal COPD before it shows in a spirometric test, resulting into a Pulmonary Obstruction Index bigger than zero (12.9 in this case).

8.5 As a Summary

This is a different summary with respect to summaries in the previous chapters. Aside the introduction to floating singletons only some paragraphs above (floating singletons can be understood as an advanced topic in this work), there have not been theoretical concepts in this chapter, so this section will be short. We have learnt how to design some medium-complexity projects, including examples of simulation/control, speech synthesis and a case of expert diagnosis system in medicine.

Now I would like to seize the reader's attention in order to remember some little material from the first chapter in Sect. 1.3 when we spoke about neurons in our brains and the incredible work developed by chemical reactions in the synapses when we think. We realized that thinking can be seen as an extremely quick exchange of electrons between atoms of chemical elements and molecules of neurotransmitters in a huge number of individual chemical reactions. In a similar way, artificial intelligence maybe will be only a sophisticated and relatively simple handling of arithmetic processes happening at really high speeds in a parallel architecture of processes. In this second decade of century XXI we are still learning the basics of the game, but even now, just after having read this book, we are ready to examine the number of simple algebraic operations required to operate the models developed in this book.

With this goal in mind we only require a line of Lisp code inserted in the FuzzyLisp function *(fl-set-membership?)*, the most "low-level" function in

Table 8.5 Number of calls to the function (*fl-set-membership?*) and the resulting number of simple arithmetic calls for running the fuzzy-logic based projects described in this chapter

	Moon landing	Castor	Spirometry
Function calls	27,440	900	520
Arithmetic calls	109,760	3600	2080

FuzzyLisp that, as the reader well knows, returns the membership degree of an element x in a fuzzy set *X*. That line of Lisp code reads as follows:

```
(setq *number-calls* (add *number-calls* 1))
```

where *number-calls* is a global variable that can be initialized at the Lisp prompt before running any of the projects developed in this chapter:

> (setq *number-calls* 0)
: 0

After running any of the projects, we could type *number-calls* at the Lisp prompt and the language will tell us how many times the function has been required in the computations. When the resulting membership degree is located in the open interval (0,1), the function *(fl-set-membership?)* requires two subtractions, one multiplication and one division, that is, as a superior bounding limit we can estimate that every call to this function requires four arithmetic operations. Armed with this trap we can calculate the whole number of arithmetic calculations for, (a) a Moon landing sequence from 3000 m high, (b) a linguistic report for the double star Castor, and c) the spirometry example described some paragraphs above. Results are given in Table 8.5.

The Moon landing project requires about 30 times more arithmetic calculations than the other ones because it is an iterative process at a 2 s sampling interval, but the important thing is that all the expert knowledge embedded in the fuzzy rules of the models and all the inferences made for running them are in fact reduced to a finite set of arithmetic operations.

Is this set of arithmetic operations "Artificial Intelligence"? Since Fuzzy Logic is a branch of AI we must concede that certainly it is. Now let us imagine for a moment the development of future complex artificial intelligence models based on fuzzy logic, neural networks, genetic algorithms and other AI strategies not even formulated yet. Now let us add computing power in the order of tens or hundred of millions of arithmetic operations per second concentrated in small hardware computing-units. Without doubt it will be "Artificial Intelligence" too. Let us try to see 50 years, or still better, two centuries into the future. The question evolves and now it is not a discussion about if it is or it is not Artificial Intelligence. The question, probably, will be "is Artificial Intelligence distinguishable from natural intelligence"?

Time will tell.

References

Adler, A.: Pretty double stars for everyone. In: Sky and Telescope. http://www.skyandtelescope.com/observing/objects/doublestars/3304626.html (2006). Acceded 2015

Argüelles, L.: Systemes de classification des etoiles doubles au moyen des techniques de logique floue. Obs. Trav. **78**, 2–6 (2011)

Argüelles, L.: Introducing floating singletons with an example application in clinic spirometry. In: 8th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2013) (2013)

Argüelles, L., Trivino, G.: I-Struve: Automatic linguistic descriptions of visual double stars. Eng. Appl. Artif. Intel. **26**(9), 2083–2092 (2013)

Argyle, R.: More than one sun. In: Observing and Measuring Double Stars. Springer, Berlin (2004)

Chen, C., Pham, T.: Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems. CRC Press, Boca Raton (2000)

Dabney, R.: Autonomous Rendezvous and Capture System Design. NASA (1991)

Fortescue, P., Swinered, G., Stark, J.: Spacecraft Systems Engineering. Wiley, New York (2011)

Hyatt, R., Scanlon, P., Nakamura, M.: Interpretation of Pulmonary Function Tests: A Practical Guide. Lippincott Williams & Wilkins (2009)

Klump, A.: Apollo Guidance, Navigation and Control. MIT (1971)

Morris, J.F. Koski, A., Johnson, L.C.: Spirometric standards for healthy, non-smoking adults. Am. Rev. Respir. Dis. **103**, 57–67 (1971)

NASA: Apollo Lunar Module Documentation. https://www.hq.nasa.gov/alsj/alsj-LMdocs.html. Acceded Feb 2015

Ostlie, D., Carroll, B.: An Introduction to Modern Astrophysics. Benjamin Cummings, San Francisco (2006)

Passino, K., Yurkovich, S.: Fuzzy Control. Addison Wesley, Reading (1998)

Qaseem, A., et al.: Diagnosis and management of stable chronic obstructive pulmonary disease: a clinical practice guideline update from the American College of Physicians, American College of Chest Physicians, American Thoracic Society, and European Respiratory Society. Ann. Intern. Med. **155**(3), 179–191 (2011)

Ross, T.: Fuzzy Logic with Engineering Applications. Wiley, New York (2010)

Santo-Tomas, L.H.: Emphysema and chronic obstructive pulmonary disease in coal miners. Curr. Opin. Pulm. Med. **17**(2), 123–125 (2011)

Seising, R.: The Fuzzification of Systems. Springer, Berlin (2007)

Yamaguchi, H., Narita, Y., Takahashi, H., Katou, Y.: Automatic transmission shift schedule control using fuzzy logic. SAE technical paper 930674 (1993)

Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybernet. **15**(1), 116–132 (1985)

Zadeh, L.A.: From computing with numbers to computing with words—from manipulation of measurements to manipulation of perceptions. IEEE Trans. Circuits Syst. **45** (1999)

Zadeh, L.A.: A new direction in AI: towards a computational theory of perceptions of measurements to manipulation of perceptions. AI Mag. **22**(1), 73–84 (2001)