CHAPTER
SEVENTEEN

FUZZY LOGIC

Slumber not in the tents of your fathers. The world is advancing. Advance with it.

—Giuseppe Mazzini

CHAPTER HIGHLIGHTS

Fuzzy logic provides PLCs with the ability to make "reasoned" decisions about a process. In this chapter, we will introduce you to the basics of fuzzy logic, including fundamental concepts and historical origins. We will demonstrate how fuzzy logic can be used in practical applications to provide real-time, logical control of a process. When you finish this chapter, you will have learned about the advanced applications of PLCs. You will then be ready to learn how to connect PLCs through local area networks.

17-1 INTRODUCTION TO FUZZY LOGIC

Fuzzy logic is a branch of artificial intelligence that deals with reasoning algorithms used to emulate human thinking and decision making in machines. These algorithms are used in applications where process data cannot be represented in binary form. For example, the statements "the air feels cool" and "he is young" are not discrete statements. They do not provide concrete data about the air temperature or the person's age (i.e., the air is at 65°F or the boy is 12 years old). Fuzzy logic interprets vague statements like these so that they make logical sense. In the case of the cool air, a PLC with fuzzy logic capabilities would interpret both the level of coolness and its relationship to warmth to ascertain that "cool" means somewhere between hot and cold. In straight binary logic, hot would be one discrete value (e.g., logic 1) and cold would be the other (e.g., logic 0), leaving no value to represent a cool temperature (see Figure 17-1).

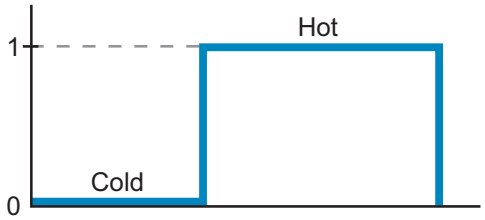


Figure 17-1. Binary logic representation of a discrete temperature value.

In contrast to binary logic, fuzzy logic can be thought of as *gray logic*, which creates a way to express in-between data values. Fuzzy logic associates a **grade**, or *level,* with a data range, giving it a value of 1 at its maximum and 0 at its minimum. For example, Figure 17-2a illustrates a representation of a cool air temperature range, where 70°F indicates perfectly cool air (i.e., a grade value of 1). Any temperature over 80°F is considered hot, and any temperature below 60°F is considered cold. Thus, temperatures above 80°F and below 60°F have a value of 0 cool, meaning they are not cool at all. Figure 17-2b shows another representation of the cool temperature range, where the dotted line shows that hot and cold temperatures are not cool. At 65°F, the fuzzy logic algorithm considers the temperature to be 50% cool and 50% cold, indicating a level of coolness. Below 60°F, the fuzzy logic algorithm considers the temperature to be cold.

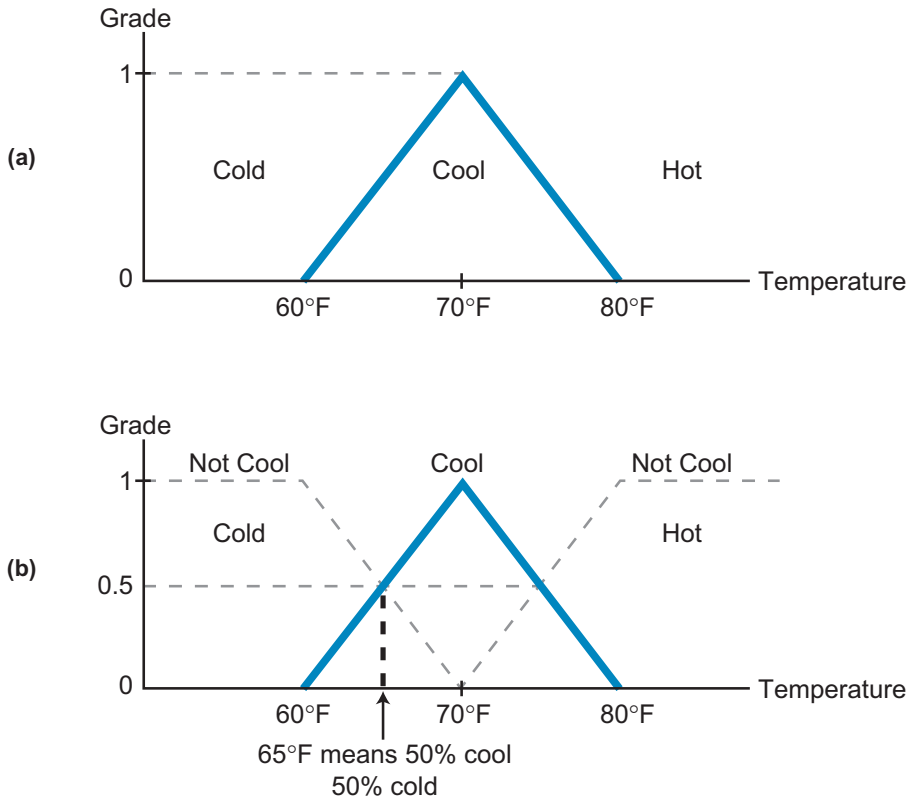
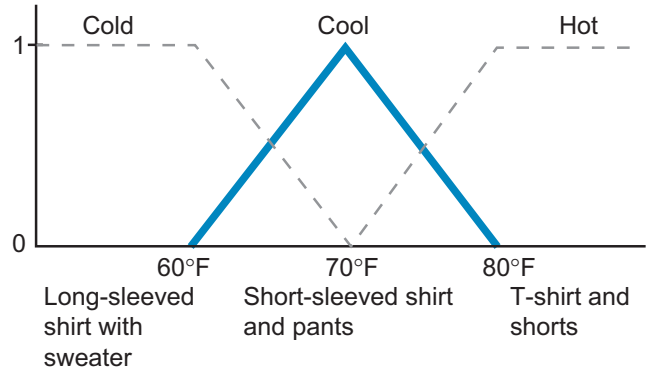


Figure 17-2. (a) Cool air temperature range with **(b)** dotted lines showing not cool range.

In real life, this fuzzy logic temperature algorithm can be associated with the decision you make about the type of clothing you wear at different times of the year. The type of clothing is based on the temperature (input) and its grade representation. As shown in Figure 17-3, at 70°F, you may only need a short-sleeved shirt and pants. However, as the temperature drops to 65°F, you may decide to wear a long-sleeved shirt instead of a short-sleeved one. Moreover, if the input is 25% cool and 75% cold (62.5°F), then you may decide to add another layer, a jacket, based on the temperature and its value of coolness. As we will explain later, a fuzzy system's output may be based on several inputs, not just one, like temperature. In this situation, the output decision is made using the knowledge base represented in the fuzzy logic graph.

Fuzzy logic requires knowledge in order to reason. This knowledge, which is provided by a person who knows the process or machine (the expert), is stored in the fuzzy system. For example, if the temperature rises in a temperature-regulated batch system, the expert may say that the steam valve needs to be turned clockwise a "little bit." A fuzzy system may interpret this expression as a 10-degree clockwise rotation that closes the current valve opening by 5%. As the name implies, a description such as a "little bit" is a fuzzy description, meaning that it does not have a definite value.



IF temperature is 70°F (grade 1–100% cool),
THEN wear short-sleeved shirt and long pants

IF temperature is 65°F (0.5 cold, 0.5 cool),
THEN wear long-sleeved shirt and long pants

IF temperature is 62.5°F (0.25 cool, 0.75 cold),
THEN wear long-sleeved shirt with a sweater and long pants

Figure 17-3. Fuzzy logic graph illustrating clothing choices based on temperature.

EXAMPLE 17-1

Figure 17-4 illustrates one representation of age (i.e., young, middle age, and old) based on the number of years a person has been alive. In this representation, the exact moment that someone passes the age of 35, he or she is considered middle-aged. Illustrate **(a)** a fuzzy logic representation of this same set of ages, and **(b)** how the representation would change if the age was divided into four ranges: young (up to 35 years), middle age (35–55 years), mature (45–65 years), and old (more than 65 years).

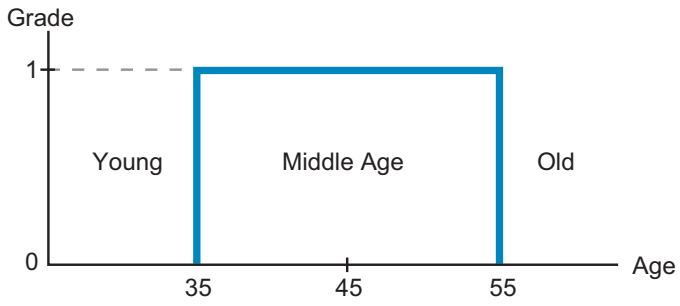


Figure 17-4. Age representation graph.

SOLUTION

(a) Figure 17-5 shows a triangular fuzzy representation that describes the age ranges. In this graph, a person who is 45 years old is perfectly middle-aged, while a person who is 50 years old is 50% middle-aged and 50% old.

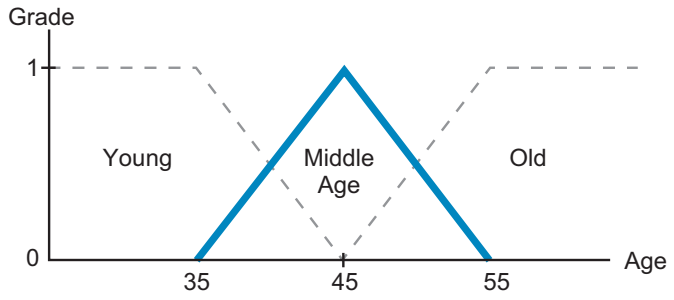


Figure 17-5. Fuzzy logic age ranges.

(b) Figure 17-6 illustrates the fuzzy logic representation for the four age groups: young, middle age, mature, and old. In this chart, a person who is 50 years old is 50% middle-aged and 50% mature.

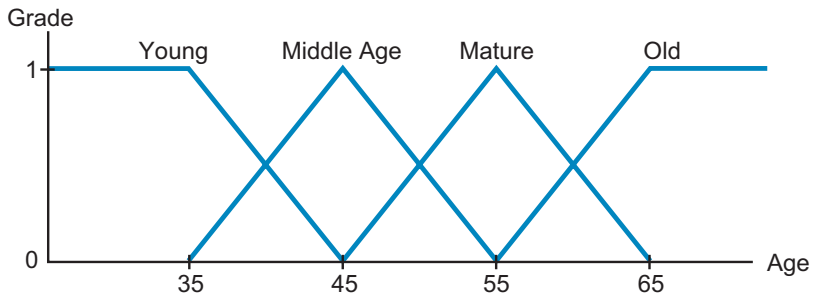


Figure 17-6. Fuzzy logic graph using four age groups.

17-2 HISTORY OF FUZZY LOGIC

Fuzzy logic has existed since the ancient times, when Aristotle developed the *law of the excluded middle*. In this law, Aristotle pointed out that the middle ground is lost in the art of logical reasoning—statements are either true or false, never in-between. When PLCs were developed, their discrete logic was based on the ancient reasoning techniques. Thus, inputs and outputs could belong to only one set (i.e., ON or OFF); all other values were excluded. Fuzzy logic breaks the law of the excluded middle in PLCs by allowing elements to belong to more than just one set. In the cool air example, the 65°F temperature input belonged to two sets, the cool set and the cold set, with grade levels indicating how well it fit into each set.

The origins of fuzzy logic date back to the early part of the twentieth century when Bertrand Russell discovered an ancient Greek paradox that states:

> A Cretan asserts that all Cretans lie. So, is he lying? If he lies, then he is telling the truth and does not lie. If he does not lie, then he tells the truth and, therefore, he lies.

In either case—that all Cretans lie or that all Cretans do not lie—a contradiction exists, because both statements are true and false. Russell found that this same paradox applied to the set theory used in discrete logic. Statements must either be totally true or totally false, leading to areas of contradiction.

Fuzzy logic surmounted this problem in classical logic by allowing statements to be interpreted as both true and false. Therefore, applying fuzzy logic to the Greek paradox yields a statement that is both true and false: Cretans tell the truth 50% of the time and lie 50% of the time. This interpretation is very similar to the idea of a glass of water being half empty or half full. In fuzzy logic the glass is both—50% full and 50% empty. Even as the amount of water decreases, the glass still retains percentages of both conditions.

Around the 1920s, independent of Bertrand Russell, a Polish logician named Jan Lukasiewicz started working on multivalued logic, which created fractional binary values between logic 1 and logic 0. In a 1937 article in *Philosophy of Science,* Max Black, a quantum philosopher, applied this multivalued logic to lists (or sets) and drew the first set of fuzzy curves, calling them *vague sets.* Twenty-eight years later, Dr. Lofti Zadeh, the Electrical Engineering Department Chair at the University of California at Berkeley, published a landmark paper entitled "Fuzzy Sets," which gave the name to the field of fuzzy logic. In this paper, Dr. Zadeh applied Lukasiewicz's logic to all objects in a set and worked out a complete algebra for fuzzy sets. Due to this groundbreaking work, Dr. Zadeh is considered to be the father of modern fuzzy logic.

Around 1975, Ebrahim Mamdani and S. Assilian of the Queen Mary College of the University of London (England) published a paper entitled "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," where the feasibility of fuzzy logic control was proven by applying fuzzy control to a steam engine. Since then, the term *fuzzy logic* has come to mean mathematical or computational reasoning that utilizes fuzzy sets.

17-3 FUZZY LOGIC OPERATION

Figure 17-7 illustrates a fuzzy logic control system. The input to the fuzzy system is the output of the process, which is entered into the system via input interfaces. For example, in a temperature control application, the input data would be entered using an analog input module. This input information would

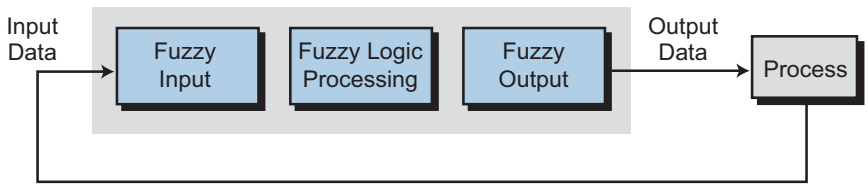


Figure 17-7. Fuzzy logic control system.

then go through the fuzzy logic process, where the processor would analyze a database to obtain an output. Fuzzy processing involves the execution of IF...THEN rules, which are based on the input conditions. An input's grade specifies how well it fits into a particular graphic set (e.g., too little, normal, too much). Note that input data, as shown in Figure 17-8, may also be represented as a count value ranging from 0 to 4095 or as a percentage of error deviation. If the fuzzy logic system utilizes an analog input that has a count range from 0 to 4095, the graphs representing the input will cover the span from 0 to 4095 counts. Furthermore, the analog input information (0–4095 counts) may represent an error range, from –50% to +50%, of a process.

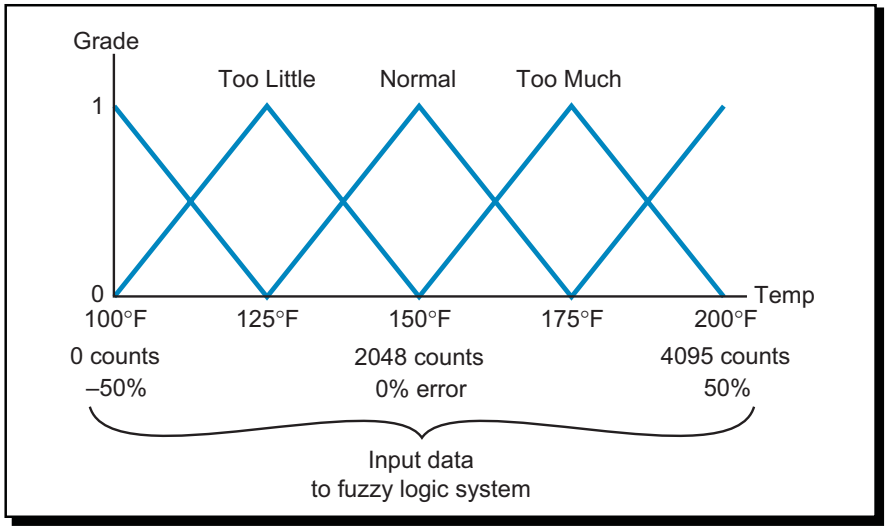


Figure 17-8. Input data to a fuzzy logic system represented as counts and percentages.

The output of a fuzzy controller is also defined by grades, with the grade determining the appropriate output value for the control element. The output of the fuzzy logic system in Figure 17-9, for example, controls a steam valve,

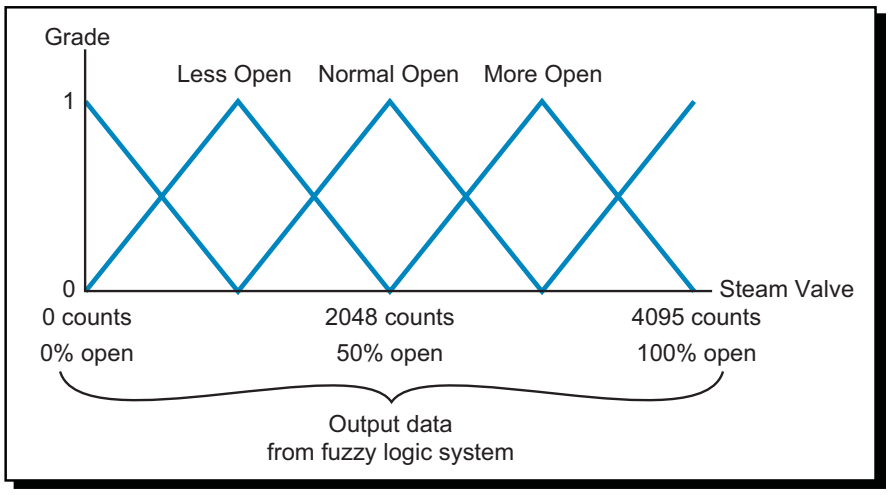


Figure 17-9. Output data from a fuzzy logic system represented as counts and percentages.

which opens or closes according to its grade on the output chart. Figure 17-10 illustrates a fuzzy logic cooling system chart with both input and output grades, where the horizontal axis is the input condition (temperature) and the vertical axis is the output (air-conditioner motor speed). In this chart, a single input can trigger more than one output condition. For example, if the input temperature is 137.5°F, then the temperature is part of two input curves—it is 50% too cool and 50% normal. Consequently, the input will trigger two outputs—the too cool input condition will trigger a less speed output, while the normal input will trigger a normal speed output condition. Since the fuzzy logic controller can have only one output, it completes a process called *defuzzification* (explained later) to determine the actual final output value.

The implementation and operation of a fuzzy logic control system is similar to the implementation of PID control using intelligent interfaces, where the module reads the input, processes the information, and provides an output.

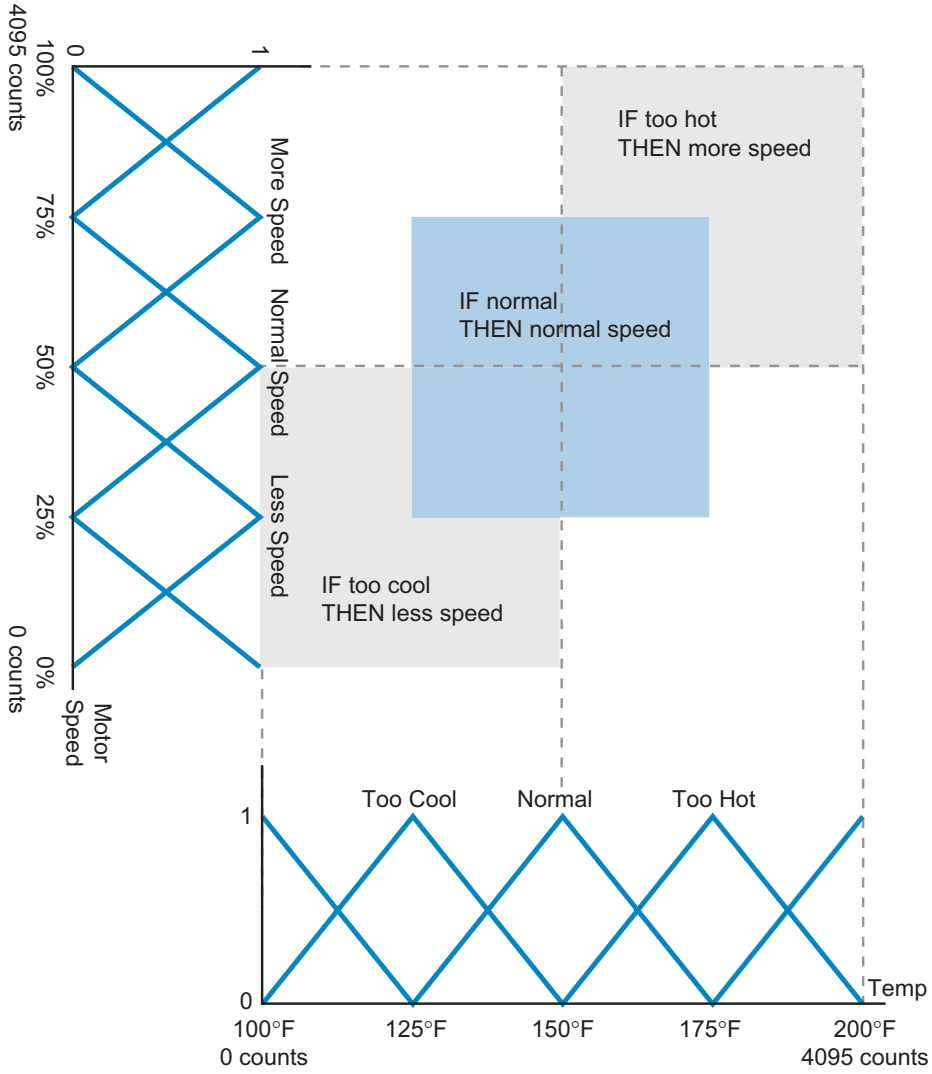


Figure 17-10. Fuzzy logic system chart showing both input and output grades.

However, fuzzy controllers are usually independent interfaces, which plug into the PLC rack and use the PLC's I/O system to communicate with the process under fuzzy control. In Chapter 8, we discussed the operation and interfacing of intelligent fuzzy logic modules.

17-4 FUZZY LOGIC CONTROL COMPONENTS

In this section, we will explain the main components of a fuzzy logic controller and also implement a simple fuzzy control program. The three main actions performed by a fuzzy logic controller are:

- fuzzification
- fuzzy processing
- defuzzification

As shown in Figure 17-11, when the fuzzy controller receives the input data, it translates it into a fuzzy form. This process is called **fuzzification**. The controller then performs **fuzzy processing**, which involves the evaluation of the input information according to IF…THEN rules created by the user during the fuzzy control system's programming and design stages. Once the fuzzy controller finishes the rule-processing stage and arrives at an outcome conclusion, it begins the **defuzzification** process. In this final step, the fuzzy controller converts the output conclusions into "real" output data (e.g., analog counts) and sends this data to the process via an output module interface. If the fuzzy logic controller is located in the PLC rack and does not have a direct or built-in I/O interface with the process, then it will send the defuzzification output to the PLC memory location that maps the process's output interface module.

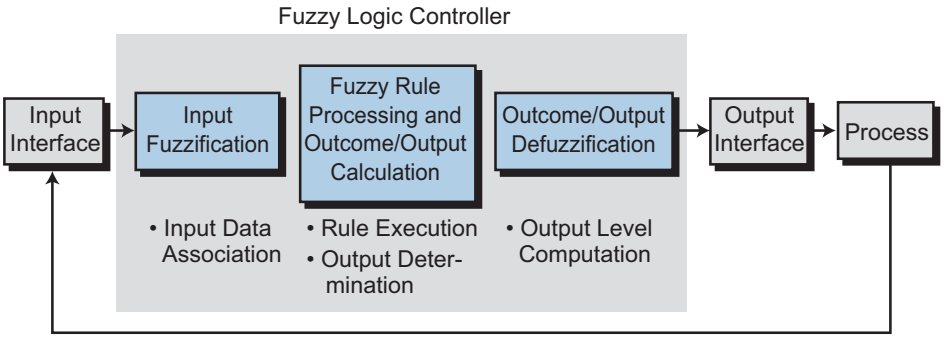


Figure 17-11. Fuzzy logic controller operation.

FUZZIFICATION COMPONENTS

The fuzzification process is the interpretation of input data by the fuzzy controller. Fuzzification consists of two main components:

- membership functions
- labels

Membership Functions. During fuzzification, a fuzzy logic controller receives input data, also known as the fuzzy variable, and analyzes it according to user-defined charts called **membership functions** (see Figure 17-12). Membership functions group input data into sets, such as temperatures that are too cold, motor speeds that are acceptable, etc. The controller assigns the input data a grade from 0 to 1 based on how well it fits into each membership function (e.g., 0.45 too cold, 0.7 acceptable speed). Membership functions can have many shapes, depending on the data set, but the most common are the S, Z, Λ, and Π shapes shown in Figure 17-13. Note that these membership functions are made up of connecting line segments defined by the lines' end points. Each membership function can have up to three line segments with a maximum of four end points. The grade

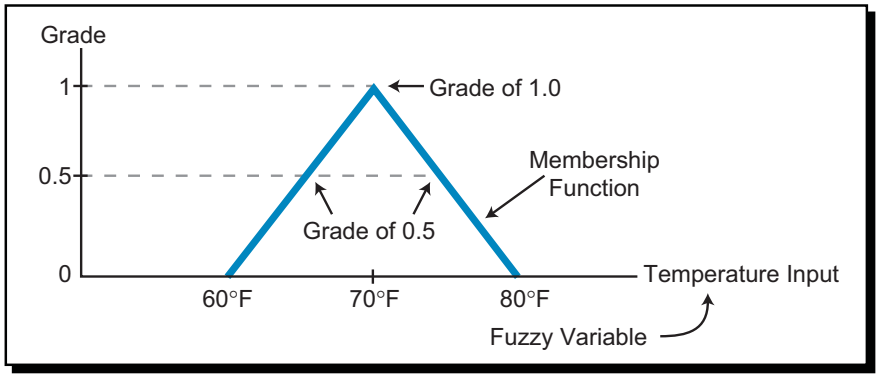


Figure 17-12. Membership function chart.

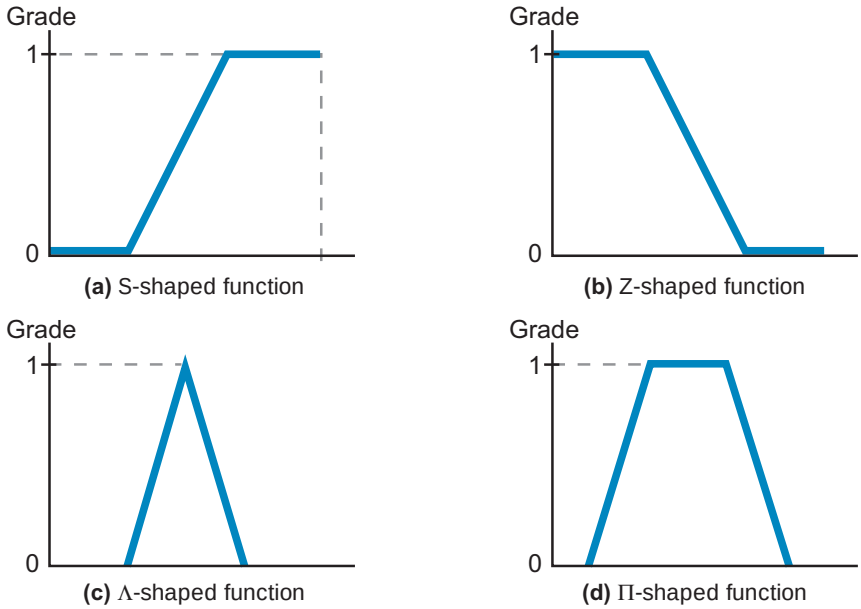


Figure 17-13. Membership function shapes: **(a)** S, **(b)** Z, **(c)** Λ, and **(d)** Π.

at each end point must have a value of 0 or 1. As shown in Figure 17-14, a membership function's shape does not have to be symmetrical; however, it must comply with the previously discussed specifications. Figure 17-15 illustrates some incorrect membership function shapes.

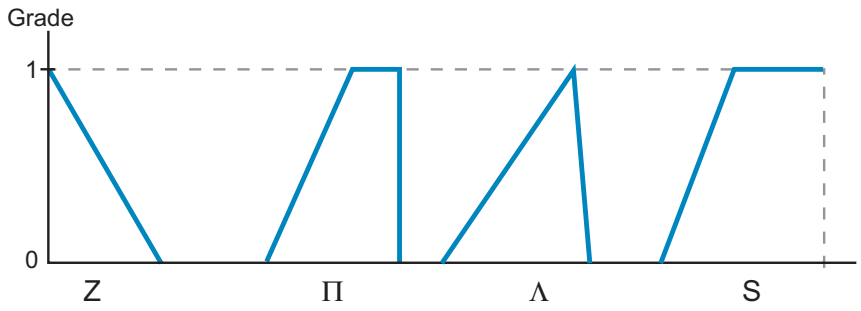


Figure 17-14. Asymmetrical membership functions.

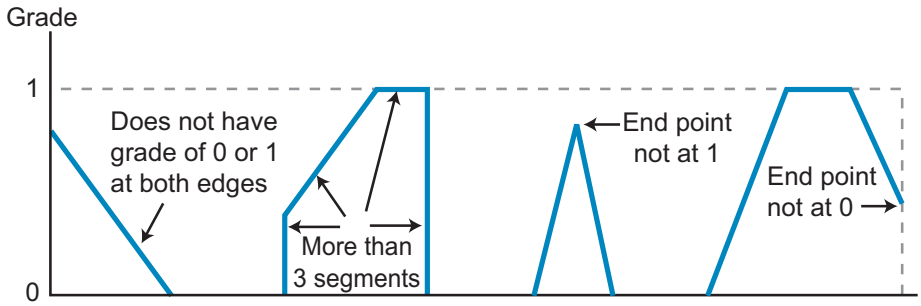


Figure 17-15. Incorrect membership function shapes.

Labels. Each fuzzy controller input can have several membership functions, with seven being the maximum and the norm, that define its conditions. Each membership function is defined by a name called a **label**. For example, an input variable such as temperature might have five membership functions labeled as cold, cool, normal, warm, and hot. Generically, the seven membership functions have the following labels, which span from the data range's minimum point (negative large) to its maximum point (positive large):

- NL (negative large)
- NM (negative medium)
- NS (negative small)
- ZR (zero)
- PS (positive small)
- PM (positive medium)
- PL (positive large)

Figure 17-16 illustrates an example of an input variable with seven Λ-shaped membership functions using all of the possible labels. A group of membership functions forms a **fuzzy set**. Figure 17-17 shows a fuzzy set with five membership functions. Although most fuzzy sets have an odd number of labels, a set can also have an even number of labels. For example, a fuzzy set may have four or six labels in any shape, depending on how the inputs are defined in relationship to the membership function.

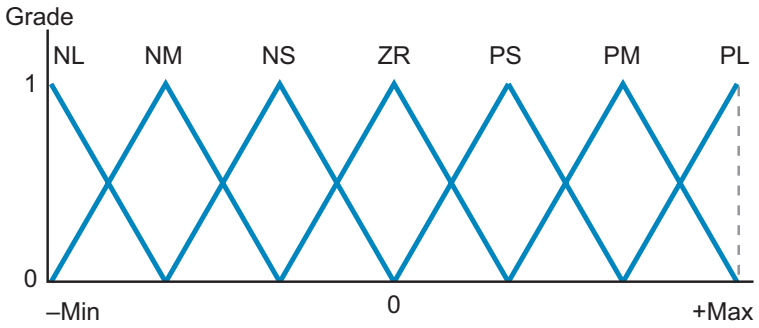


Figure 17-16. Fuzzy logic input using seven membership function labels.

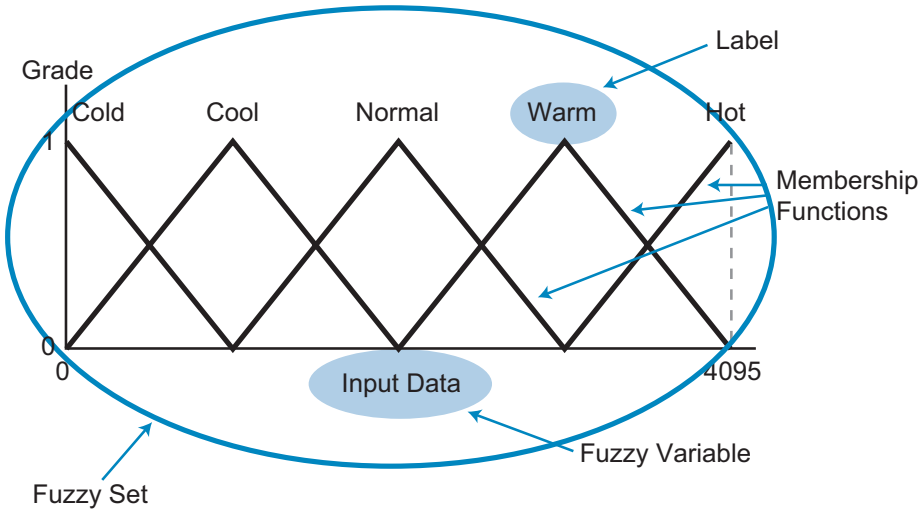


Figure 17-17. Fuzzy set with five membership functions.

FUZZY PROCESSING COMPONENTS

During fuzzy processing, the controller analyzes the input data, as defined by the membership functions, to arrive at a control output. During this stage, the processor performs two actions:

- rule evaluation
- fuzzy outcome calculation

Rule Evaluation. Fuzzy logic is based on the concept that most complicated problems are formed by a collection of simple problems and can, therefore, be easily solved. Fuzzy logic uses a reasoning, or *inferencing*, process composed of IF...THEN **rules**, each providing a response or outcome. Basically, a rule is activated, or *triggered*, if an input condition satisfies the IF part of the rule statement. This results in a control output based on the THEN part of the rule statement. In a fuzzy logic system, many rules may exist, corresponding to one or more IF conditions (see Figure 17-18). A rule may also have several input conditions, which are logically linked in either an AND or an OR relationship to trigger the rule's outcome (see Figure 17-19).

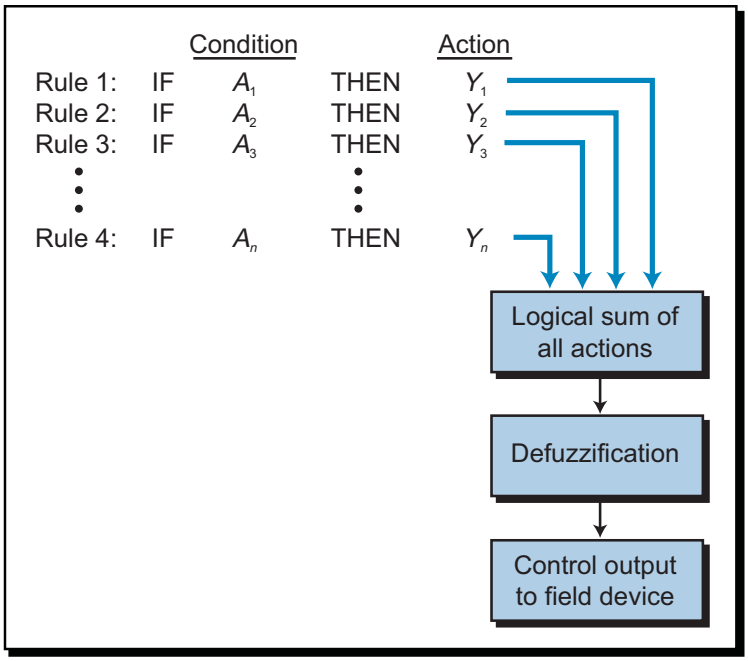


Figure 17-18. Multiple rules in a fuzzy system.

	Condition		Action
Rule 1:	IF	A_1 AND B_1 AND C_1	THEN Y_1
Rule 2:	IF	A_2 OR B_2	THEN Y_2
Rule 3:	IF	(A_3 AND B_3) OR C_3	THEN Y_3

Figure 17-19. Rules with multiple input conditions linked in AND and OR relationships.

Sometimes, more than one rule is triggered at a time in a fuzzy control process. In this case, the controller evaluates all the rules to arrive at a single outcome value and then proceeds to the defuzzification process. For instance, if two inputs are logically ANDed or ORed in several rules, then they will produce several outcomes, of which only one will be logically added

to determine the final outcome. Figure 17-20a illustrates an example of two fuzzy inputs, X_1 and X_2, and one fuzzy output, Y_1. The rules shown in Figure 17-20b represent four of nine possible rules that cover the two inputs. The four shown, however, cover the four possible triggering points for the two input readings, X_1 and X_2. Given the input values in Figure 17-20a, the inputs will trigger rule 1 because X_1 = ZR AND X_2 = NL. This will generate two outputs for Y_1 = NL, one at a grade of 0.6 (due to the input value of X_1) and the other at a grade of 0.75 (due to the value of X_2). In a fuzzy logic situation where a two-input rule with an AND relationship produces two outcome values, the controller will choose the outcome with the *lowest* grade, in this case 0.6NL. If the rule utilizes OR logic, the chosen outcome will be the one with the

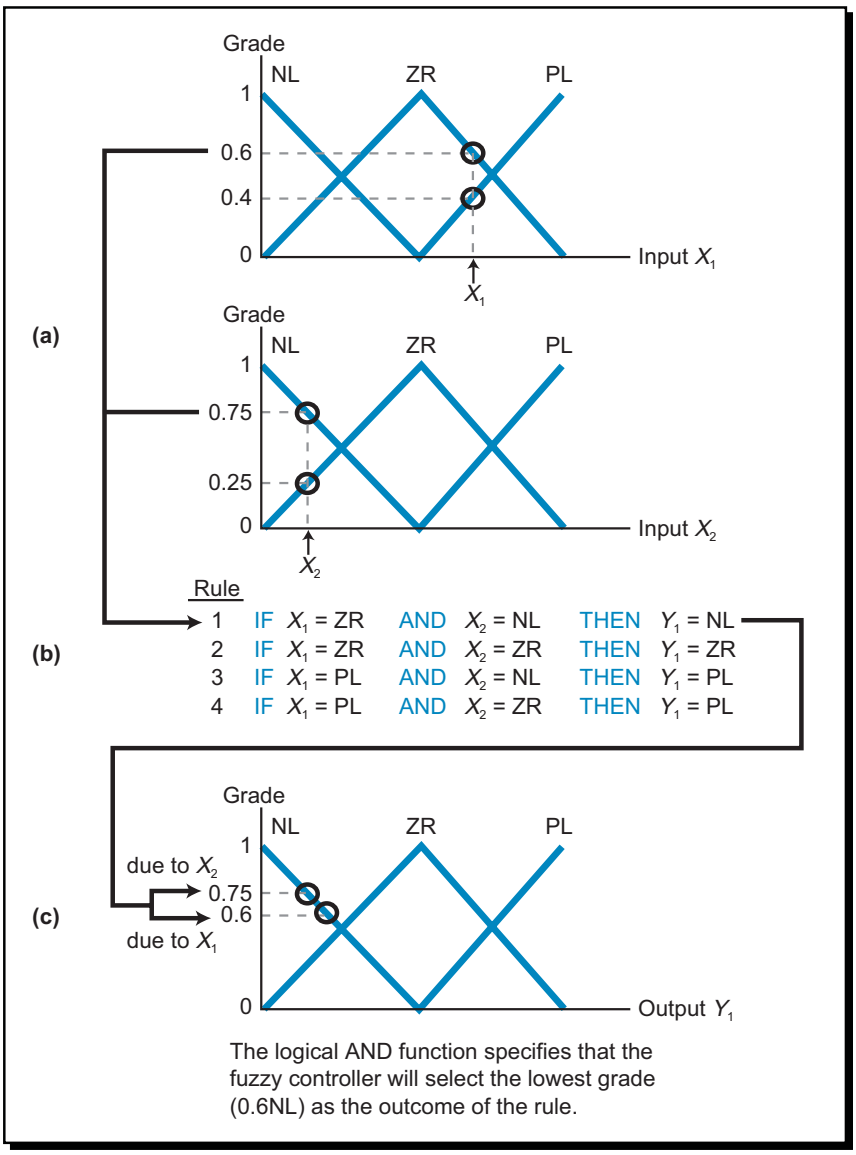


Figure 17-20. Fuzzy processing example showing **(a)** two fuzzy input values, **(b)** the four rules that they trigger, and **(c)** the resulting output.

largest grade. If rule 1 in Figure 17-20 had used an OR function instead of an AND function, then the controller would have selected the $Y_1 = 0.75$NL outcome, the largest of the two outcomes.

Different fuzzy logic controllers have different rule evaluation capabilities. The fuzzy logic controller from Omron Electronics shown in Figure 17-21, for example, is capable of handling eight inputs and four outputs, where each input can be represented by a maximum of seven membership functions for a total of 56 membership functions (8×7). The controller also allows a maximum of 128 programmed rules. Each rule can have up to eight input conditions (which can be logically ANDed or ORed) and two outcomes.

Figure 17-21. Omron's fuzzy logic controller in a PLC system.

Fuzzy logic rules with two inputs are often represented in matrix form to represent AND conditions. For example, Figure 17-22 illustrates a 3×3 matrix (9 rules) that uses two inputs, X_1 and X_2, and one output Y_1. One advantage of this matrix representation is that it makes it easy to represent all the rules for a system. A five-label system translates into a 5×5 matrix with 25 rules, while a seven-label system produces a 7×7 matrix with 49

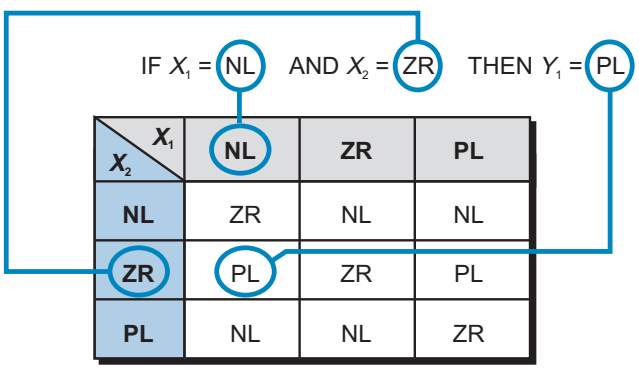


Figure 17-22. Fuzzy logic rule matrix.

rules. An even membership function combination (e.g., a system with 6 labels for one input and 4 labels for another) will have a 24-rule matrix. When more than three inputs are used, the matrix becomes more difficult to represent, since it becomes a three-dimensional matrix resembling a cube (three inputs). In this type of complicated system, the rules would be broken down into several two-dimensional matrices.

Fuzzy Outcome Calculations. Once a rule is triggered, meaning that the input data belongs to a membership function that satisfies the rule's IF statement, the rule will generate an output outcome. This fuzzy output is composed of one or more membership functions (with labels), which have grades associated with them. The outcome's membership function grade is affected by the grade level of the input data in its input membership function. In Figure 17-23a, the fuzzy input *FI* of 60% belongs to two membership functions, ZR and PS, corresponding to the grades of 0.6 and 0.4, respectively. These two grades will have an impact on the amount of the output (see Figure 17-23b) by intersecting the output membership functions at the same grade levels (0.6 and 0.4). However, the output membership function that is selected for the final output value depends on the user's programming of the IF...THEN rules.

For example, in Figure 17-24, the input triggers rules 3 and 4 because the input *FI* belongs to membership functions ZR and PS. These rules indicate that both fuzzy output action ZR and action PL must be applied to the process. These output actions will be applied at a value that corresponds to the grades generated in the input membership functions (i.e., output 0.6ZR and

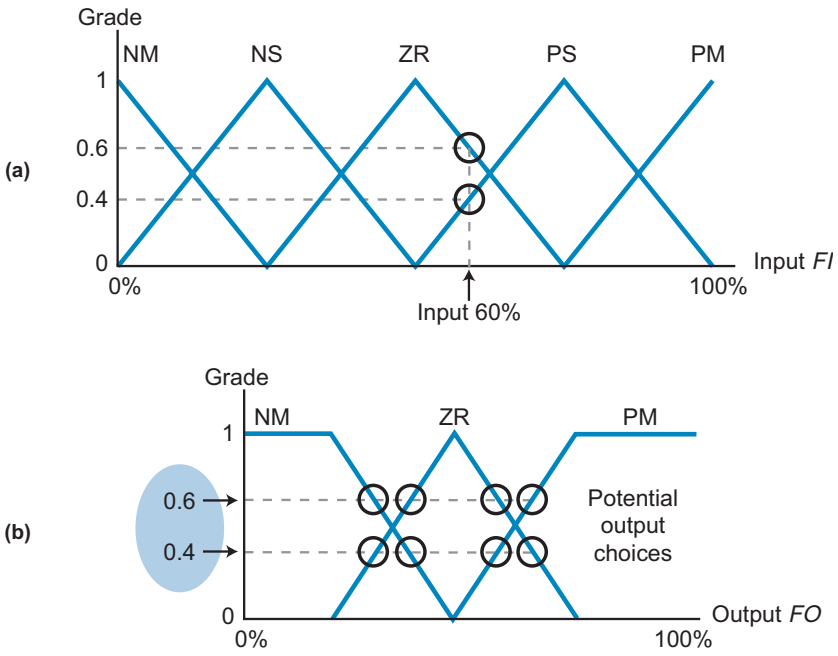


Figure 17-23. (a) Fuzzy input grades and **(b)** the resulting output grades.

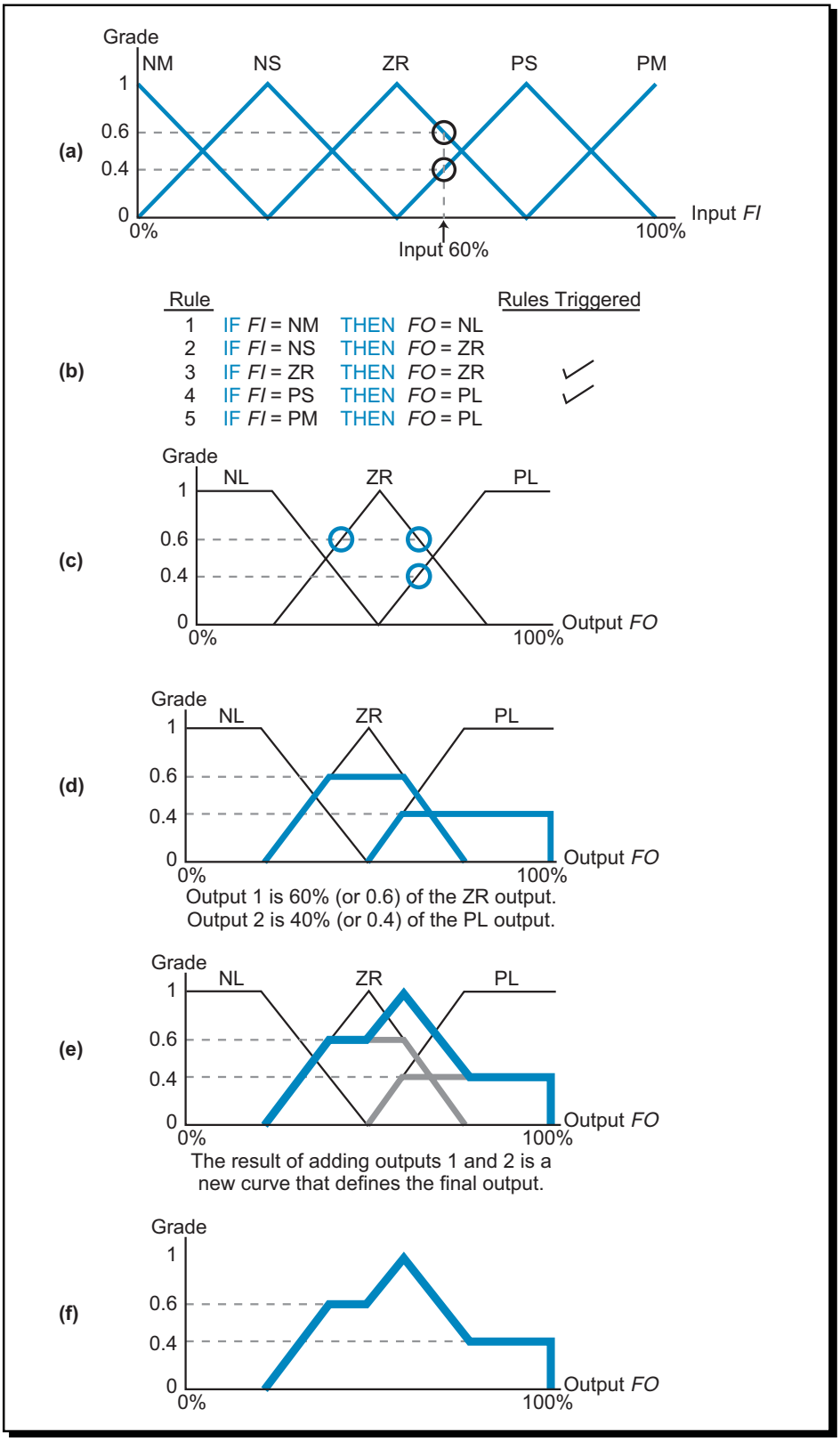


Figure 17-24. Fuzzy logic process: **(a)** inputs, **(b)** rules, **(c)** outputs, **(d)** output curves, **(e)** combined output curve, and **(f)** the output signal for the field device.

0.4PL). Note that the 0.6 grade is applied to output ZR and the 0.4 grade is applied to output PL because the user programmed the rules that way. Figure 17-24c shows these two outputs. To arrive at a final outcome value, the fuzzy logic controller logically adds both fuzzy outcomes to produce an aggregate outcome curve, illustrated in Figure 17-24e. The controller then generates an output signal (during defuzzification) that controls the process's field device (e.g., valve, motor, etc.) according to the input data (see Figure 17-24f).

A fuzzy logic controller may implement its output membership functions as noncontinuous functions that resemble spikes rather than geometrical shapes. Figure 17-25 illustrates an example of seven output membership functions represented as spikes and described by labels. Each label has a relationship to the output interface. For example, each label shown in Figure 17-25 corresponds to a value between 0 and 4095 counts. As another example, the three output membership functions presented in Figure 17-24 can be represented as noncontinuous spikes (see Figure 17-26), where the outcome grade levels specify 0.6 of ZR and 0.4 of PM.

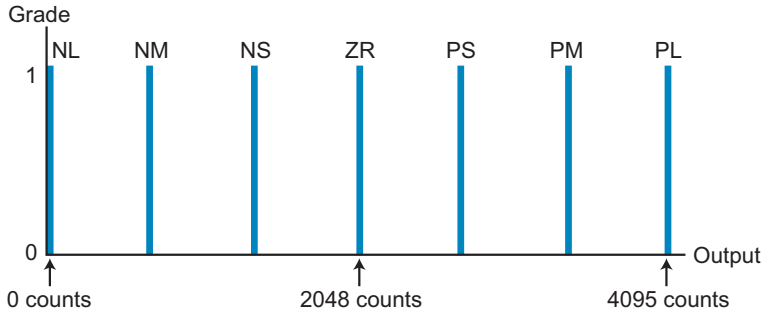


Figure 17-25. Output membership functions represented as noncontinuous functions.

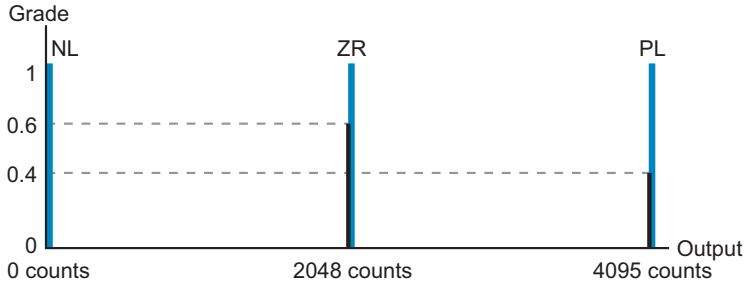


Figure 17-26. The three output membership functions from Figure 17-24 shown as spikes.

EXAMPLE 17-2

Figure 17-27 illustrates a three–membership function fuzzy set and the three rules that dictate the outcomes. For a fuzzy input (*FI*) of 37.5%, **(a)** indicate the triggered rules and the outcome membership functions selected and **(b)** illustrate the logical sum of the selected outputs.

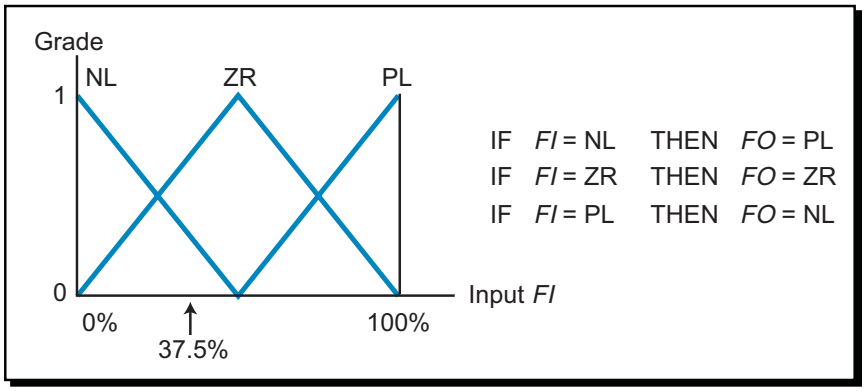


Figure 17-27. Three–membership function fuzzy set and its rules.

SOLUTION

(a) Figure 17-28a shows the two rules triggered (rules 1 and 2) by the 37.5% *FI* input, where *FI* intercepts the input membership functions at 0.25NL and 0.75ZR. Consequently, these rules trigger output values of 0.25PL and 0.75ZR, as shown in Figure 17-28b.

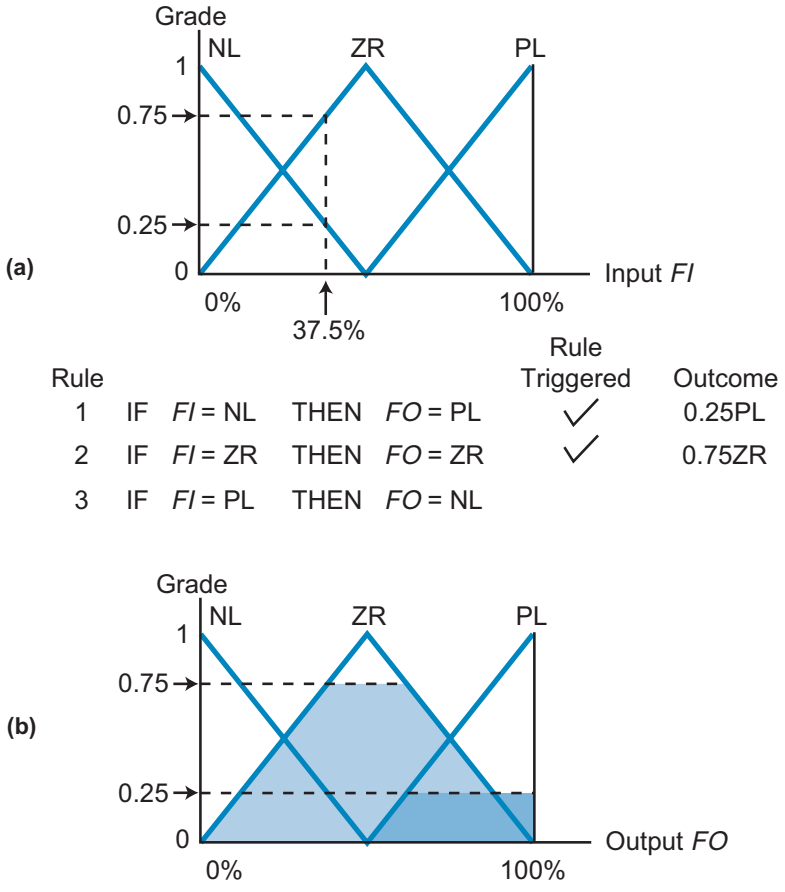


Figure 17-28. (a) Inputs triggered by the rules and **(b)** the resulting outputs.

(b) Figure 17-29 shows the logical sum that the fuzzy controller will perform. This logical sum is the result of geometrically adding the areas of the two outcomes (0.75ZR and 0.25PL) to form one graphic output, from which a final output (fuzzy output *FO*) will be selected during defuzzification. This output value will then be sent to the control field device.

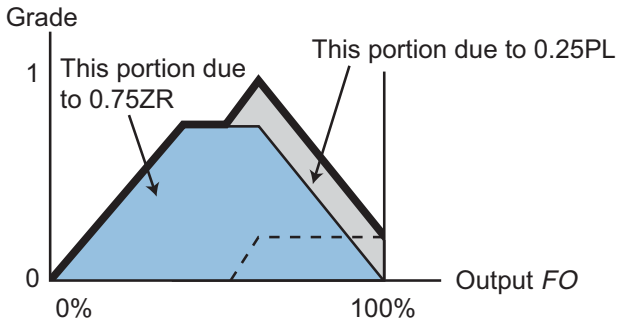


Figure 17-29. Outcome curve for Example 17-2.

EXAMPLE 17-3

Figure 17-30 illustrates two input fuzzy sets, one with five labels and the other with two labels, while Figure 17-31 shows one fuzzy output set with five labels. The rules that govern the system (as defined by the expert) are shown in Figure 17-31a in matrix form for a maximum of 10 possible combinations.

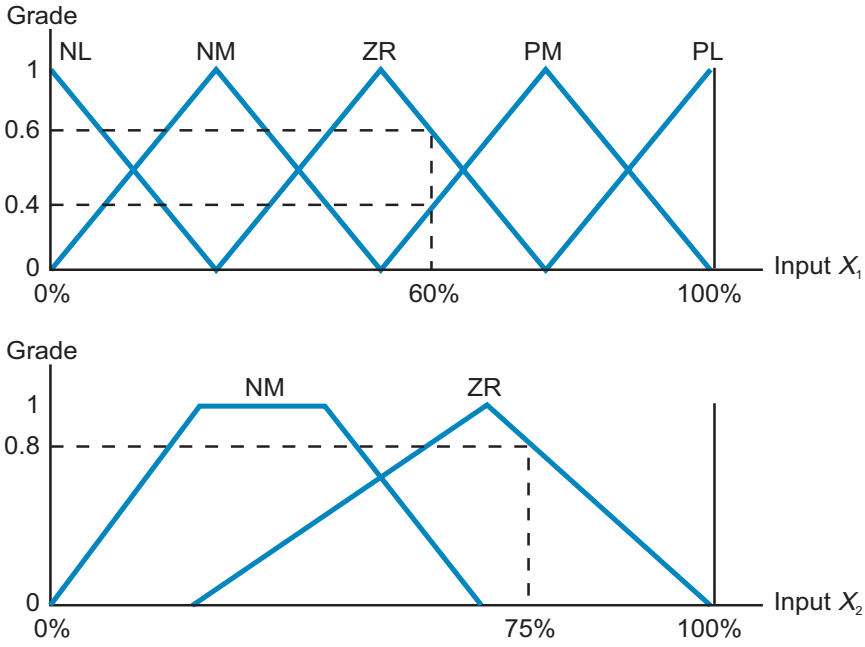


Figure 17-30. Fuzzy input sets for input X_1 and input X_2.

(a)

Output Rule Matrix

X_2 \ X_1	NL	NS	ZR	PS	PL
NM	NL	NM	NM	ZR	PM
ZR	NM	ZR	ZR	PM	PL

(b)

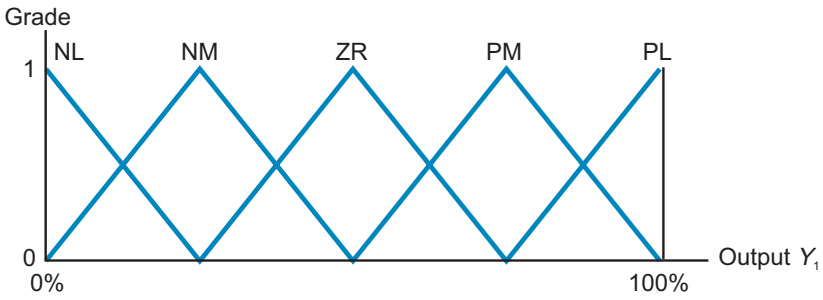


Figure 17-31. (a) Rule matrix and **(b)** fuzzy set.

(a) Indicate the rules that are triggered for the two input conditions X_1 = 60% and X_2 = 75%, as well as all the possible outcomes of the rules' triggered inputs. Also, indicate the outputs that will be selected. **(b)** Illustrate the selected outcomes and the logical outcome summation that will be used for defuzzification.

SOLUTION

(a) Figure 17-32 illustrates the two rules that will be triggered due to inputs X_1 and X_2. Input X_1 will intercept membership functions ZR and PM at grades 0.6 and 0.4, respectively. Input X_2 will intercept ZR at a

Output Rule Matrix

X_2 \ X_1	NL	NM	ZR	PM	PL
NM	NL	NM	NM	ZR	PM
ZR	NM	ZR	ZR	PM	PL

X_1 = 0.6ZR and 0.4PM
X_2 = 0.8ZR

IF X_1 = ZR AND X_2 = ZR
IF X_1 = PM AND X_2 = ZR

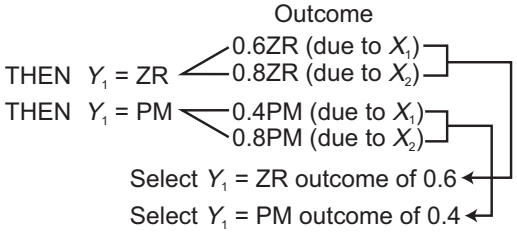


Figure 17-32. Triggered rules.

grade of 0.8. Figure 17-30 presented these grade levels. Because the rules are linked with AND functions, each rule will have two outputs, of which the one with the lowest value will be chosen for the logical sum of the outputs.

(b) Figure 17-33 illustrates the two selected outputs from the two triggered rules and the resulting output after the two rule outcomes are logically added.

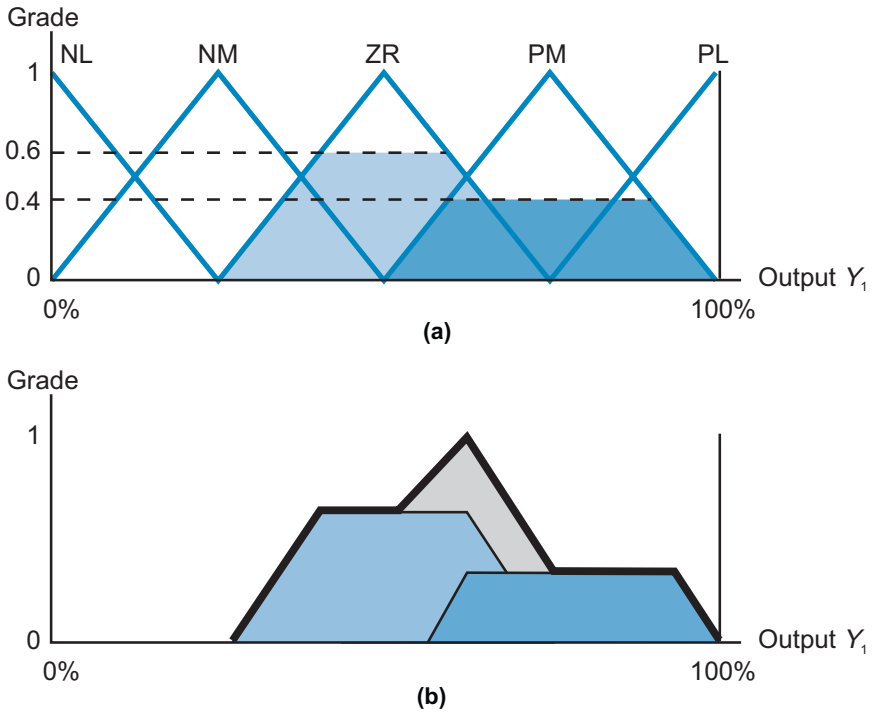


Figure 17-33. (a) Triggered outputs and **(b)** outcome curve.

DEFUZZIFICATION COMPONENTS

The final output value from the fuzzy controller depends on the defuzzification method used to compute the outcome values corresponding to each label. The defuzzification process examines all of the rule outcomes after they have been logically added and then computes a value that will be the final output of the fuzzy controller. The PLC then sends this value to the output module. Thus, during defuzzification, the controller converts the fuzzy output into a real-life data value (e.g., 1720 counts).

There are many defuzzification methods, but all are based on mathematical algorithms. The two most common defuzzification methods are:

- maximum value
- center of gravity

Maximum Value Method. The **maximum value method** bases the final output value on the rule output with the highest membership function grade. This method is mainly used with discrete output membership functions. Referring to Figure 17-26 (shown again in Figure 17-34), the maximum value defuzzification method would specify that the output value of 2048 counts be chosen as the final output value because it has the largest grade value. If two or more outcomes from two or more rules have the same grade level, then the controller will select the outcome that will be the final value based on criteria supplied by the user during the fuzzy application programming setup or system definition. Such criteria is determined by choosing either the left-most or right-most grade value of the two equal labels and their corresponding number of counts. The left-most criteria selects the lowest output (counts), while the right-most criteria selects the highest output value (highest counts).

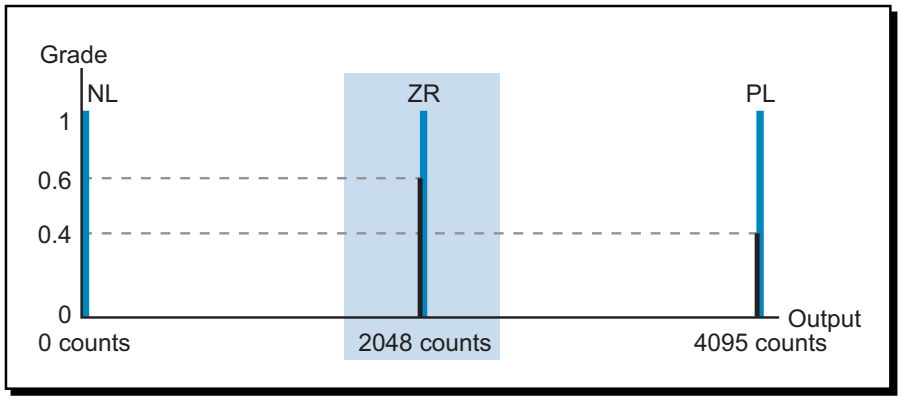


Figure 17-34. The maximum value method selects the largest output grade level.

Figure 17-35a illustrates the outcome of three rules. If the maximum value defuzzification method is used, ZR will be the final output value, meaning that the output of the controller will be 2340 counts. If the rules triggered two equal maximum grade values, as is the case in Figure 17-35b, then the controller would use the programmed criteria to select the appropriate output value. If this criteria specified the left-most maximum value, then the controller would chose label NM, which would provide an output of 1170 counts. Note that, during the defuzzification process, the fuzzy controller sends the actual output value (e.g., counts), not the grade value, to the output device. So, in Figure 17-35a, the output will be approximately 2340 counts. In Figure 17-35b, the left-most output will be approximately 1170 counts and the right-most output, if chosen, will be 3510 counts.

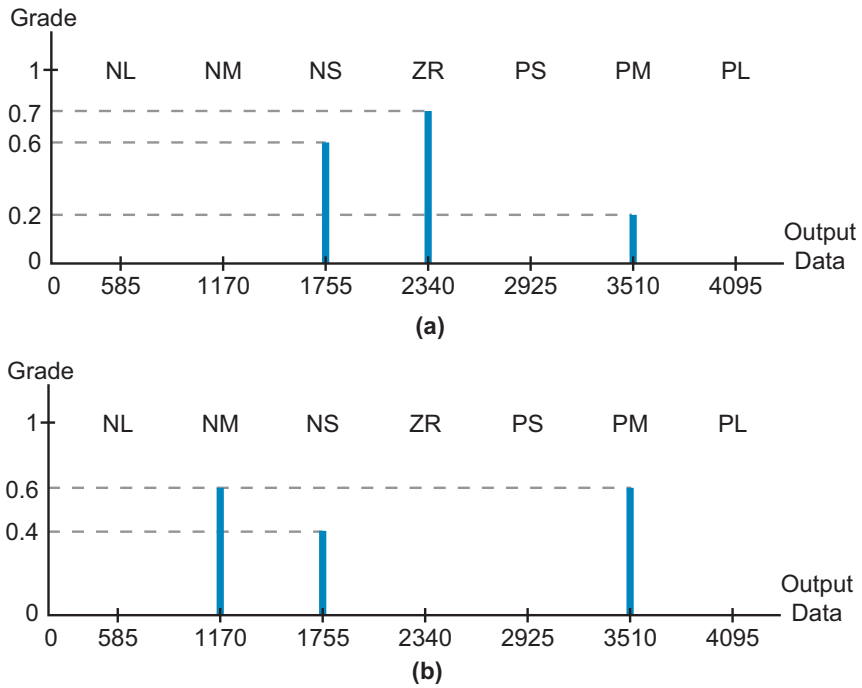


Figure 17-35. (a) Single maximum output value and **(b)** multiple maximum output values.

Center of Gravity Method. The **center of gravity method**, also referred to as "calculating the centroid," mathematically obtains the center of mass of the triggered output membership functions. Figure 17-36 illustrates the centroid calculation for the example previously illustrated in Figure 17-24. In mathematical terms, a **centroid** is the point in a geometrical figure whose coordinates equal the average of all the other points comprising the figure. This point is the center of gravity of the figure. In simple terms, the center of gravity for a fuzzy output is the output data value (as shown on the X-axis), that divides the area under the fuzzy membership function curve into two equal parts. The center of gravity method is the most commonly used defuzzification method because it provides an accurate result based on the weighted values of several output membership functions. The output value that is sent to the output interface module is the output data value at the intersection of the horizontal axis and the centroid.

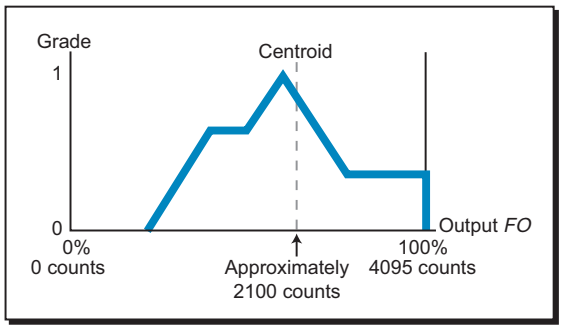


Figure 17-36. Centroid calculation of the output from Figure 17-24.

The center of gravity method applies to noncontinuous, or discrete, output membership functions, as well as continuous ones. In noncontinuous functions, the final output value that will be obtained for a seven-label output membership function (labels A through G) is expressed by the formula:

$$\text{Output data} = \frac{\sum_{n=\text{A}}^{n=\text{G}}\left[(FO_n)(FGrade_n)\right]}{\sum_{n=\text{A}}^{n=\text{G}} FGrade_n}$$

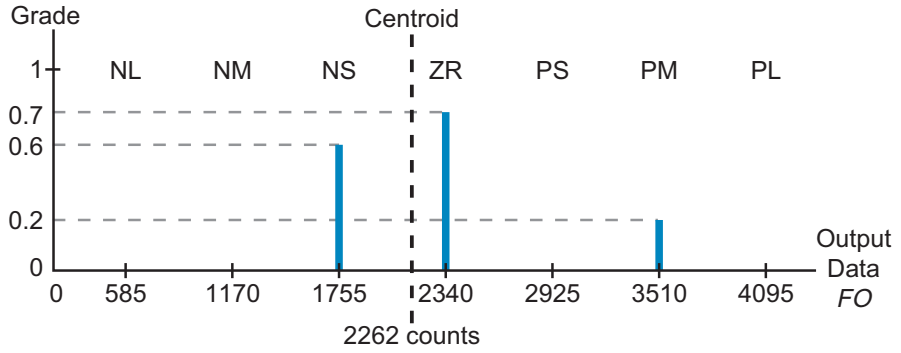
where:

Output data = the number of counts to be used for the output

FO = the fuzzy output in counts for labels n = A through G

$FGrade$ = the fuzzy grade level for levels n = A through G

Referring to our previous noncontinuous membership example, now shown in Figure 17-37, this equation implies that the final value of the output will be equal to the sum of each rule outcome's grade times its actual output data



$$\text{Final output value} = \frac{\sum_{n=\text{NL}}^{n=\text{PL}}\left[(FO_n)(FGrade_n)\right]}{\sum_{n=\text{NL}}^{n=\text{PL}} FGrade_n}$$

$$= \frac{(FO_{\text{NS}})(FGrade_{\text{NS}}) + (FO_{\text{ZR}})(FGrade_{\text{ZR}}) + (FO_{\text{PM}})(FGrade_{\text{PM}})}{FGrade_{\text{NS}} + FGrade_{\text{ZR}} + FGrade_{\text{PM}}}$$

$$= \frac{(1755)(0.6) + (2340)(0.7) + (3510)(0.2)}{0.6 + 0.7 + 0.2}$$

$$= \frac{3393}{1.5}$$

$$= 2262 \text{ counts}$$

Figure 17-37. Centroid calculation for the noncontinuous membership example.

value (i.e., counts) divided by the sum of the rule outcome grades. In this case, the fuzzy logic controller will decide to send an output of 2262 counts to the output interface after completing the center of gravity calculation. The fuzzy controller's output is less than it would have been using the maximum value method (the maximum output label is ZR, which is 2340 counts), indicating that the weighted value of the 0.6NS label pulls the value to the left (less counts). However, the output value is slightly balanced by the right-pulling action of the 0.2PM label.

Fuzzy controllers utilizing continuous membership functions and the center of gravity defuzzification method also use the previous summation equation to approximate the centroid value (see Figure 17-38). However, in this case, the controller uses approximate digitized values for each membership function to compute each of the points in the summation.

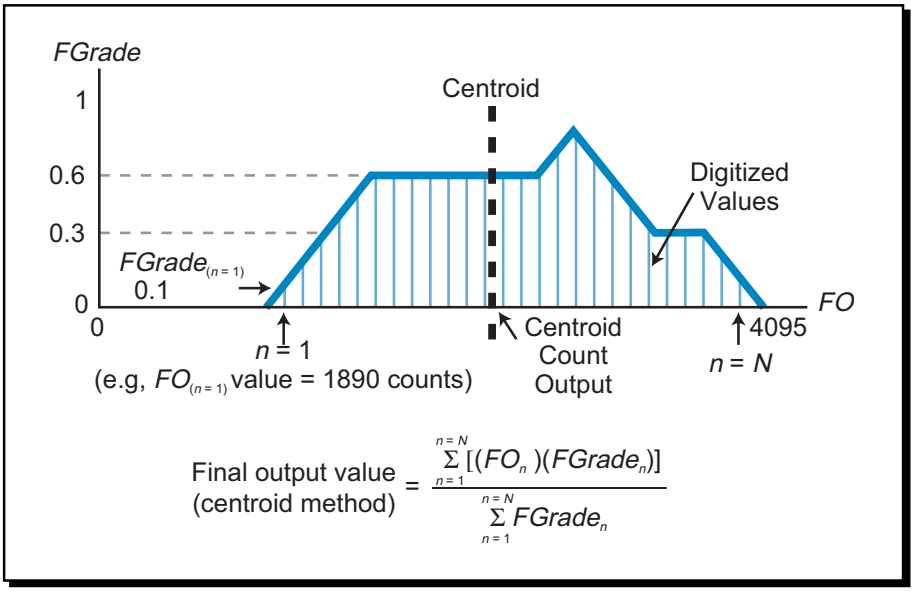


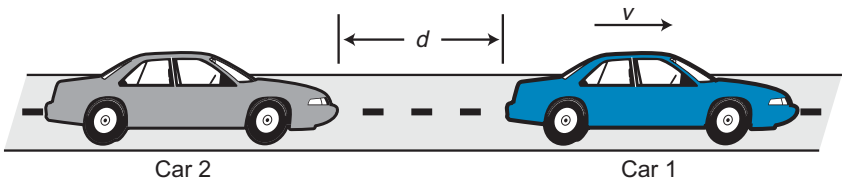
Figure 17-38. Centroid value approximation.

EXAMPLE 17-4

Figure 17-39 illustrates two cars separated by a distance *d*, which can range between 0 and 120 feet. Car 1 travels at a speed of *v*, ranging from 0 to 80 mph. Depending on the speed and distance, car 2 has several braking options (*B*) ranging from light to hard if car 1 slows down or stops.

(a) Create a Λ-shaped fuzzy set that contains three membership functions for each input: distance between cars (short, normal, long), car speed (low, normal, high), and braking strength (light, normal, hard). **(b)** Establish a set of rules for the braking output as a function

of the speed and distance. Illustrate these rules in matrix form. **(c)** Using the center of gravity method, calculate the value of the outcome if car 1 is traveling at 65 mph and the distance between car 1 and car 2 is 45 feet.



d = distance between cars
v = velocity (speed) of car 1
B = braking strength (function of *d* and *v*)

Figure 17-39. Example 17-4.

SOLUTION

(a) Figure 17-40 illustrates the three fuzzy sets, each with three membership functions. Figures 17-40a and 17-40b illustrate the two input fuzzy sets, distance and speed. The distance fuzzy set ranges from 0 to 120 feet and the speed fuzzy set ranges from 0 to 80 mph. Figure 17-40c shows the output fuzzy set (braking strength), whose output ranges from light braking to hard braking.

(b) The fuzzy system's rules are composed of IF…THEN statements defining all possible outcomes. For example:

IF the distance between the two cars is long
and the speed is normal, THEN brake lightly.

This rule implies that normal braking will be applied if the speed is normal and the distance between the cars is long. Using the fuzzy sets illustrated in Figure 17-40, this rule can be expressed as:

IF *d* = PL AND *v* = ZR THEN *B* = NL

Table 17-1 lists the rules for this fuzzy system, based on distance and speed. Remember that the user defines the rules of the system based on the desired outcome, according to his or her experience and knowledge. For example, a regular driver would tend to brake either normally or hard if the distance was short and the speed was high. However, a NASCAR driver might brake very lightly or not at all if the front car slows down, even though the NASCAR driver's distance may be very short and speed very high.

Figure 17-41 illustrates these rules in matrix form. This matrix configuration allows you to see a large number of rules and their outcomes at a glance.

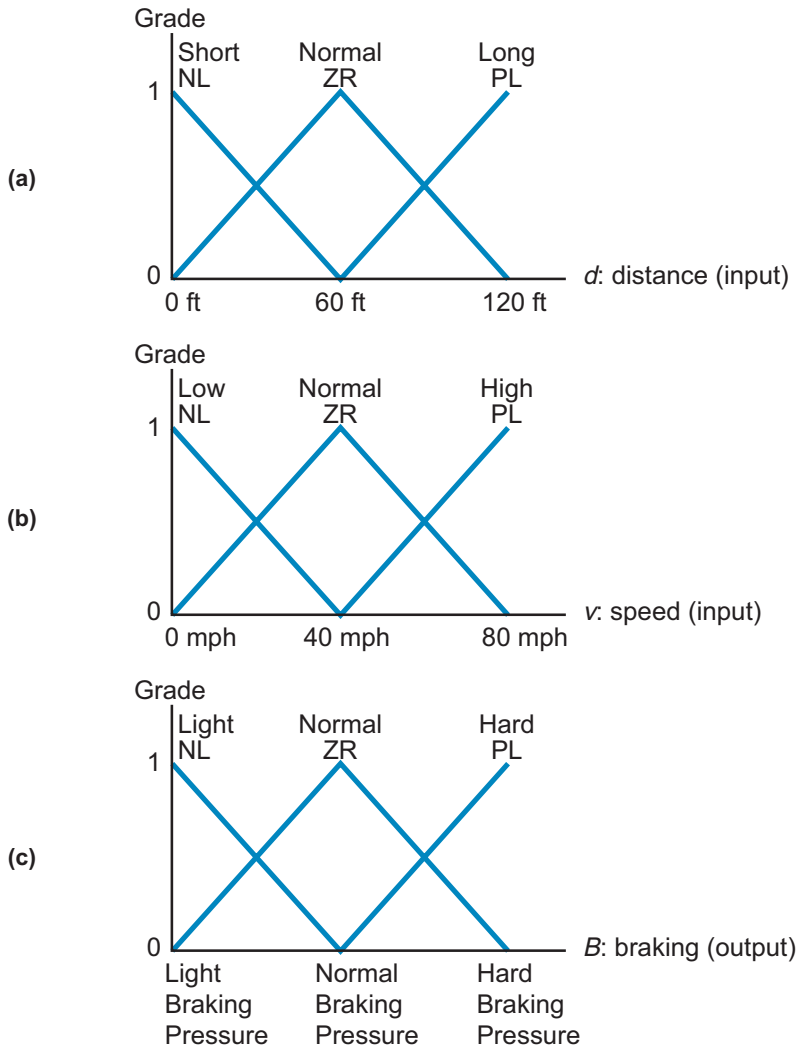


Figure 17-40. The fuzzy input sets—**(a)** distance and **(b)** speed—and **(c)** the fuzzy output set, braking.

	Fuzzy Rules
1	IF *d* = NL AND *v* = NL THEN *B* = ZR
2	IF *d* = NL AND *v* = ZR THEN *B* = PL
3	IF *d* = NL AND *v* = PL THEN *B* = PL
4	IF *d* = ZR AND *v* = NL THEN *B* = NL
5	IF *d* = ZR AND *v* = ZR THEN *B* = ZR
6	IF *d* = ZR AND *v* = PL THEN *B* = PL
7	IF *d* = PL AND *v* = NL THEN *B* = NL
8	IF *d* = PL AND *v* = ZR THEN *B* = NL
9	IF *d* = PL AND *v* = PL THEN *B* = ZR

Table 17-1. The fuzzy system's rules.

Distance *d* / Speed *v*	Short NL	Normal ZR	Long PL
Low NL	ZR (Brake Normal)	NL (Brake Light)	NL (Brake Light)
Normal ZR	PL (Brake Hard)	ZR (Brake Normal)	NL (Brake Light)
High PL	PL (Brake Hard)	PL (Brake Hard)	ZR (Brake Normal)

Figure 17-41. Fuzzy logic rule matrix.

(c) Figures 17-42a and 17-42b illustrate the graphs for the inputs d = 45 ft and v = 65 mph. Each input triggers (crosses) two membership functions—input d crosses membership functions ZR and NL; input v crosses membership functions PL and ZR. Thus, these inputs trigger four rules, rules 2, 3, 5, and 6, as were shown in Table 17-1.

Note that the inputs to these rules are connected logically by AND functions, meaning that the rules' outputs will correspond to the smallest input grade value. For example, rule 2 will be triggered because the 45-foot distance input crosses the NL (short distance) membership function (IF d = NL...) and the 65 mph speed input crosses the ZR (normal speed) membership function (...AND v = ZR). The grades for each input to rule 2 are as follows: distance = 0.25NL and speed = 0.375ZR. In other words, the 45-foot distance is 25%

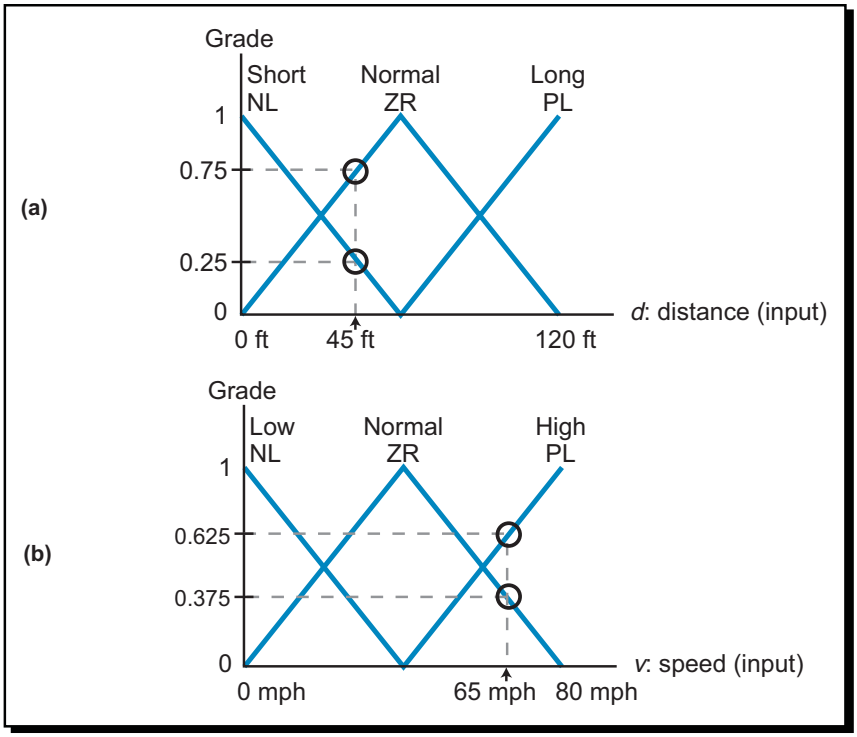


Figure 17-42. Graphs for **(a)** distance and **(b)** speed inputs.

short and the 65 mph speed is 37.5% normal. The rule implies two braking outputs (…THEN *B* = PL) based on these inputs, one at 0.25 (due to distance) and the other at 0.375 (due to speed). Because of the AND condition, however, the fuzzy controller will select the smallest outcome, 0.25 (see Figure 17-43). Figure 17-44a shows the outcome summary for all four rules, including the rule outcome selected (i.e., the smallest outcome because of the AND rule logic). Figure 17-44b illustrates all four triggered outcomes. Note that both rules 2 and 3 have 0.25PL outcomes; the dotted line represents the addition of these two outcomes. Figure 17-44c shows the logical sum of all the outcomes and the approximate output result, which was obtained using the center of gravity defuzzification method. By visually inspecting the output, the braking strength will be at approximately 70% normal and 30% hard.

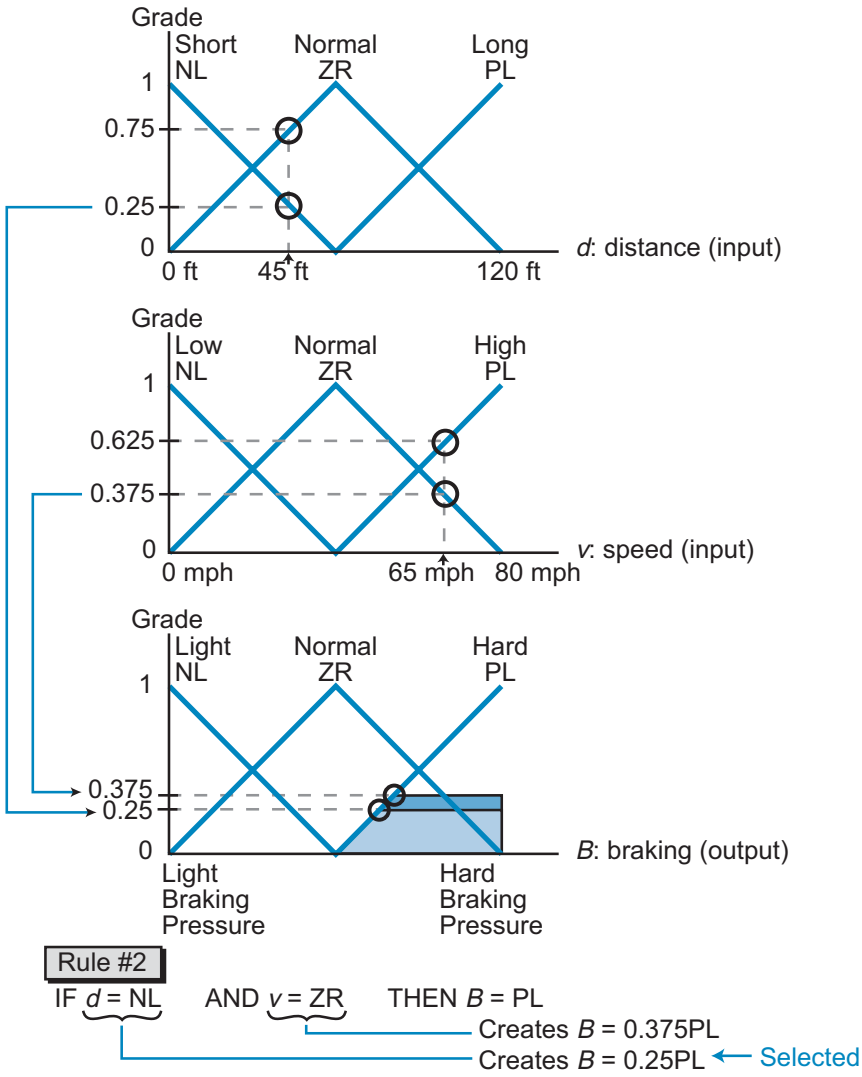


Figure 17-43. Output calculation.

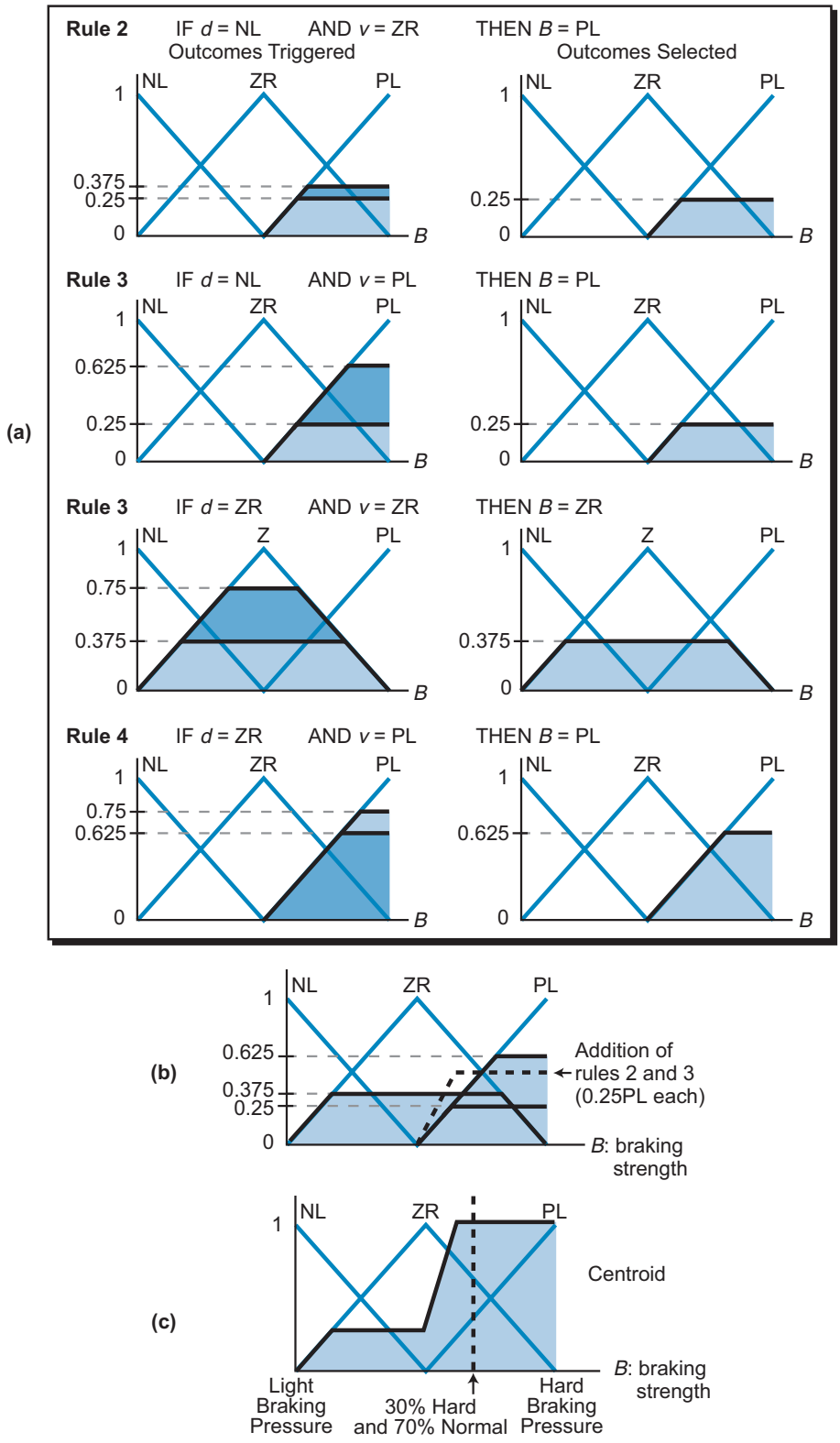


Figure 17-44. (a) Outcome summary for all four rules, **(b)** illustration of the outputs, and **(c)** defuzzification of the result.

17-5 FUZZY LOGIC CONTROL EXAMPLE

In this section, we will implement fuzzy logic rules to control the speed of a conveyor in an automated packaging system. The objective of this application is to synchronize two conveyors so that parts and packaging boxes are positioned correctly, regardless of the part and package box positions and the speed of conveyor.

SYSTEM DESCRIPTION AND OPERATION

Figure 17-45 shows the two-conveyor packaging system. The parts travel on conveyor A, pass onto the connecting conveyor, and then go to conveyor B, where they are boxed before going to the wrapping machine. The photoelectric sensors PE1 and PE2 detect the presence of a part and initiate a count to determine the part's position from encoder 1. PE3 and PE4 detect the presence of a box and determine its position based on the count inputs from encoder 2.

The control objective is to adjust the speed of conveyor B so that the packaging boxes arrive at the same time as the parts, meaning that they meet at the connecting conveyor. The process information required to implement this control is:

- the offset between the part and the packaging box
- the rate of change of the offset

The parts on conveyor A travel at random intervals, but at a constant speed. The boxes on conveyor B occur at regular intervals, and the speed of conveyor B can be controlled. Figure 17-46 shows the block diagram of the complete PLC system, including I/O and the fuzzy logic controller. The photoelectric sensors will be used in the PLC program to detect when to start timing and computing the data from the encoders. A section of the PLC's

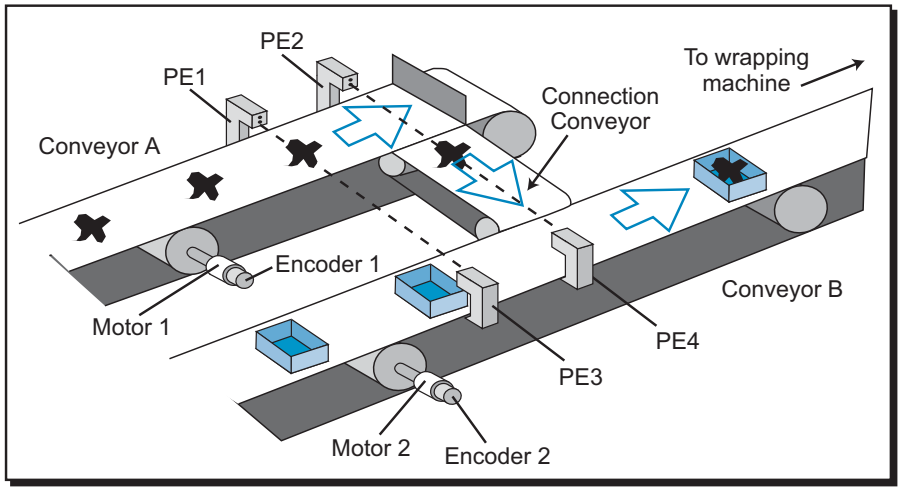


Figure 17-45. Two-conveyor packaging system.

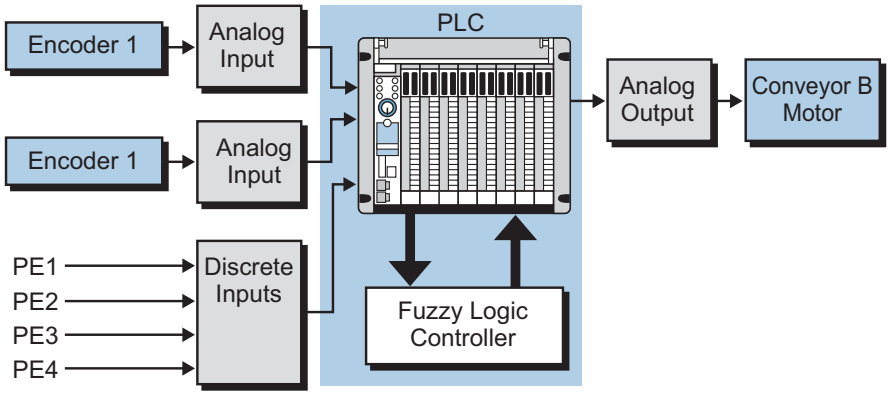


Figure 17-46. Block diagram of the PLC system.

main program must adjust the two fuzzy inputs, the part/box offset and the rate of change of the offset, so that the data is centered around a value of 2048 counts. The reason for this is that the range of the fuzzy set will be 0 to 4095 analog counts; the count value of 2048 is the middle membership function.

If a box is present at PE3 and a part is present at PE1, conveyor B should run at the same speed as conveyor A (the reference speed set initially by the operator). If the box is at PE3 but the part is behind PE1 (see Figure 17-47),

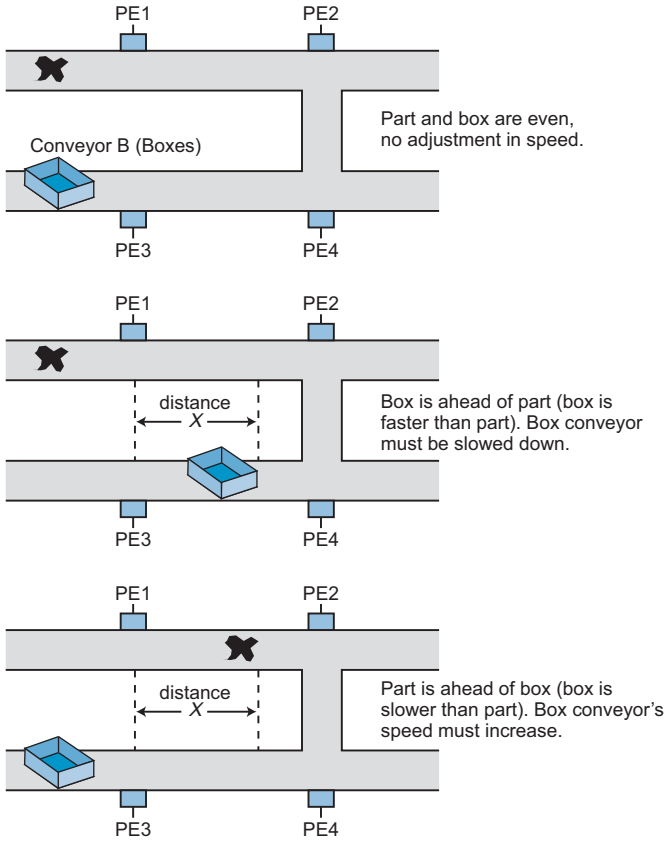


Figure 17-47. Conveyor B may slow down or speed up.

the system will slow conveyor B until the part is at PE1, at which time the fuzzy controller will indicate an increase in the speed of conveyor B so that it will catch up with conveyor A. The distance traveled by the box is calculated, using the input data from encoder 2, as the difference between the time the box passes PE3 and the time part passes PE1.

Figure 17-48 shows a flowchart of the steps that must occur to pass correct input information to the fuzzy processor. The value at position *X* provides the part/box offset data. This value is calculated as:

$$X = (\text{Encoder 1 counts}) - (\text{Encoder 2 counts})$$

The rate of change of the offset is calculated as the difference between the current offset reading (X_n) and the previous one ($X_{(n-1)}$):

$$\Delta X = X_n - X_{(n-1)}$$

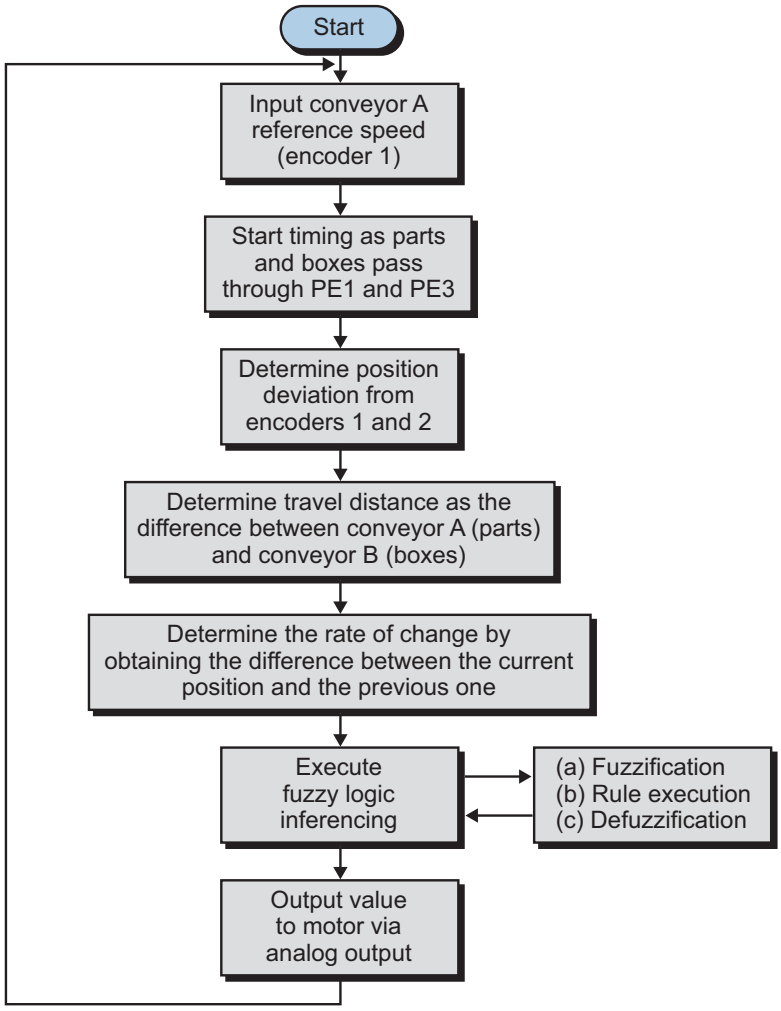


Figure 17-48. Fuzzy system flowchart.

MEMBERSHIP FUNCTIONS AND RULE CREATION

To provide enhanced resolution and accuracy, this system uses a five–membership function (five-label) fuzzy sets for the two inputs and a seven–membership function fuzzy set for the output. Figure 17-49 shows these three fuzzy sets. The offset input is named *X* (deviation between part and box) and the offset rate of change input is named *ΔX* (rate of change of deviation). The fuzzy set for the output is named *S* (speed), which corresponds to the motor speed of conveyor B. Note that the range of each fuzzy input and output

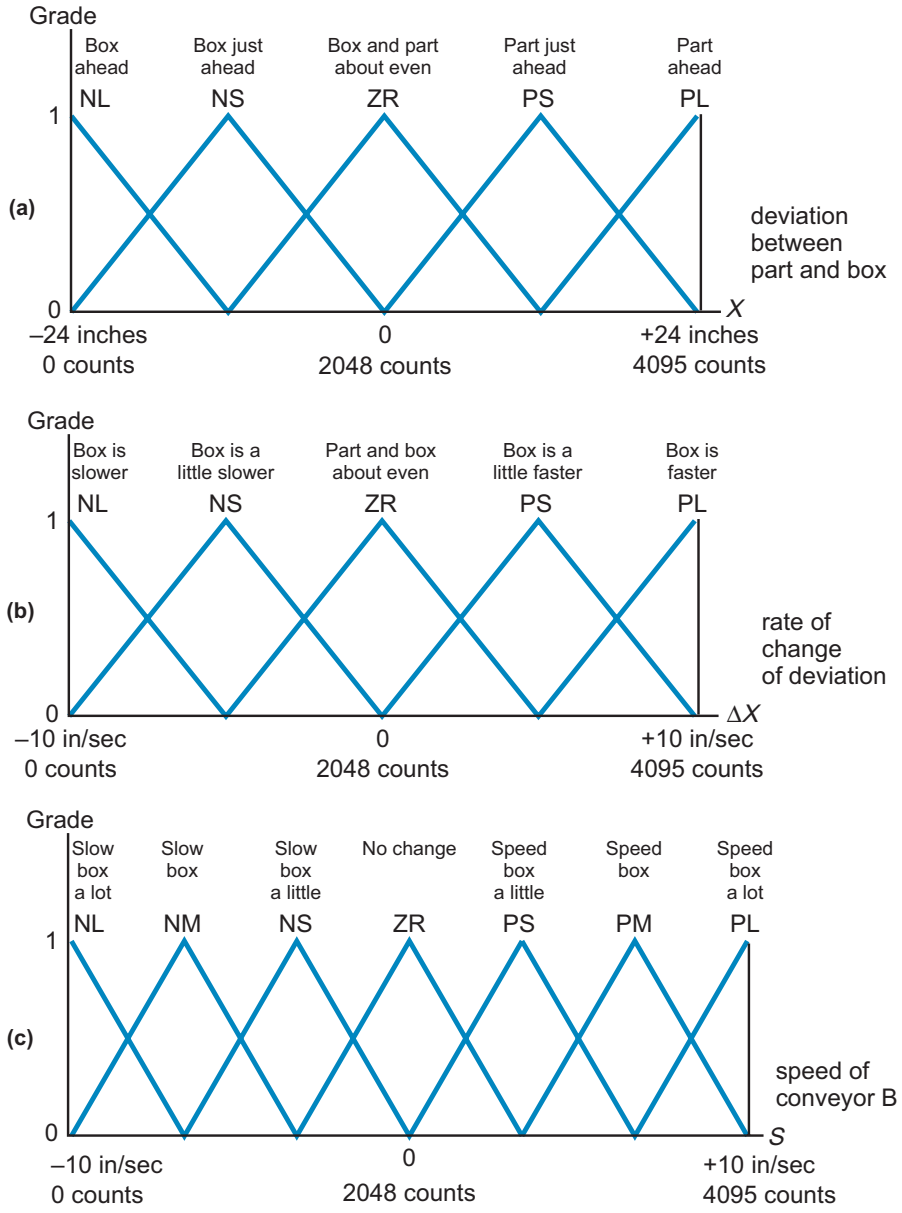


Figure 17-49. Three fuzzy sets used for the conveyor example: **(a)** deviation between part and box (input), **(b)** rate of change of deviation (input), and **(c)** speed of conveyor B (output).

variable is from 0 to 4095 counts. This corresponds to a range of ±24 inches for the deviation between the part and box positions, a range of ±10 inches/second for the rate of change of the offset, and a range of ±10 inches/second for the speed of the box conveyor.

The fuzzy logic database for this system contains 25 rules. Figure 17-50 shows a matrix of the rules, describing the desired output according to the deviation between the part and the box and the rate of change of deviation. This matrix includes a description of the rule inputs and outputs, as well as their respective membership function labels. Table 17-2 lists the actual rules that will be entered into the fuzzy controller in IF…THEN form.

Deviation *X* / Rate Δ*X*	**NL** Box is ahead	**NS** Box is just ahead	**ZR** About even	**PS** Part is just ahead	**PL** Part is ahead
NL Box is slower	NM Slow the box	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box	PL Speed up the box a lot
NS Box is a little slower	NM Slow the box	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box	PL Speed up the box a lot
ZR About even	NM Slow the box	NS Slow the box a little	ZR No change	PS Speed up the box a little	PM Speed up the box
PS Box is a little faster	NL Slow the box a lot	NM Slow the box	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box
PL Box is faster	NL Slow the box a lot	NL Slow the box a lot	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box

Figure 17-50. Fuzzy logic rule matrix.

Conveyor System Rules	
1 IF *X* = NL AND Δ*X* = NL THEN *S* = NM	**14** IF *X* = ZR AND Δ*X* = PS THEN *S* = NS
2 IF *X* = NL AND Δ*X* = NS THEN *S* = NM	**15** IF *X* = ZR AND Δ*X* = PL THEN *S* = NS
3 IF *X* = NL AND Δ*X* = ZR THEN *S* = NM	**16** IF *X* = PS AND Δ*X* = NL THEN *S* = PM
4 IF *X* = NL AND Δ*X* = PS THEN *S* = NL	**17** IF *X* = PS AND Δ*X* = NS THEN *S* = PM
5 IF *X* = NL AND Δ*X* = PL THEN *S* = NL	**18** IF *X* = PS AND Δ*X* = ZR THEN *S* = PS
6 IF *X* = NS AND Δ*X* = NL THEN *S* = NS	**19** IF *X* = PS AND Δ*X* = PS THEN *S* = PS
7 IF *X* = NS AND Δ*X* = NS THEN *S* = NS	**20** IF *X* = PS AND Δ*X* = PL THEN *S* = PS
8 IF *X* = NS AND Δ*X* = ZR THEN *S* = NS	**21** IF *X* = PL AND Δ*X* = NL THEN *S* = PL
9 IF *X* = NS AND Δ*X* = PS THEN *S* = NM	**22** IF *X* = PL AND Δ*X* = NS THEN *S* = PL
10 IF *X* = NS AND Δ*X* = PL THEN *S* = NL	**23** IF *X* = PL AND Δ*X* = ZR THEN *S* = PM
11 IF *X* = ZR AND Δ*X* = NL THEN *S* = PS	**24** IF *X* = PL AND Δ*X* = PS THEN *S* = PM
12 IF *X* = ZR AND Δ*X* = NS THEN *S* = PS	**25** IF *X* = PL AND Δ*X* = PL THEN *S* = PM
13 IF *X* = ZR AND Δ*X* = ZR THEN *S* = ZR	

Table 17-2. Fuzzy system rules.

Once the fuzzy controller receives the inputs, it will determine the final output value based on a logical addition of the selected outcomes. The outcome calculation may be very complex, due to the large number of rules. Remember, however, that the entire fuzzy logic analysis—fuzzification, rule execution, and defuzzification—is based on user-specified criteria for desired outputs based on photoelectric and encoder input data.

EXAMPLE 17-5

Figure 17-51 illustrates the part/box input membership functions for a conveyor system with two input readings, where the deviation between the part and the box is 9 inches and the rate of change is about –3.75 inches per second. This means that the part is just ahead of the box because the box is a little slower than the part. Figure 17-52 illustrates this situation.

(a) Determine which rules are triggered, indicate which outcomes are selected, and plot the output membership functions. **(b)** Illustrate the logical sum of the selected outputs and indicate an approximate output using the center of gravity method.

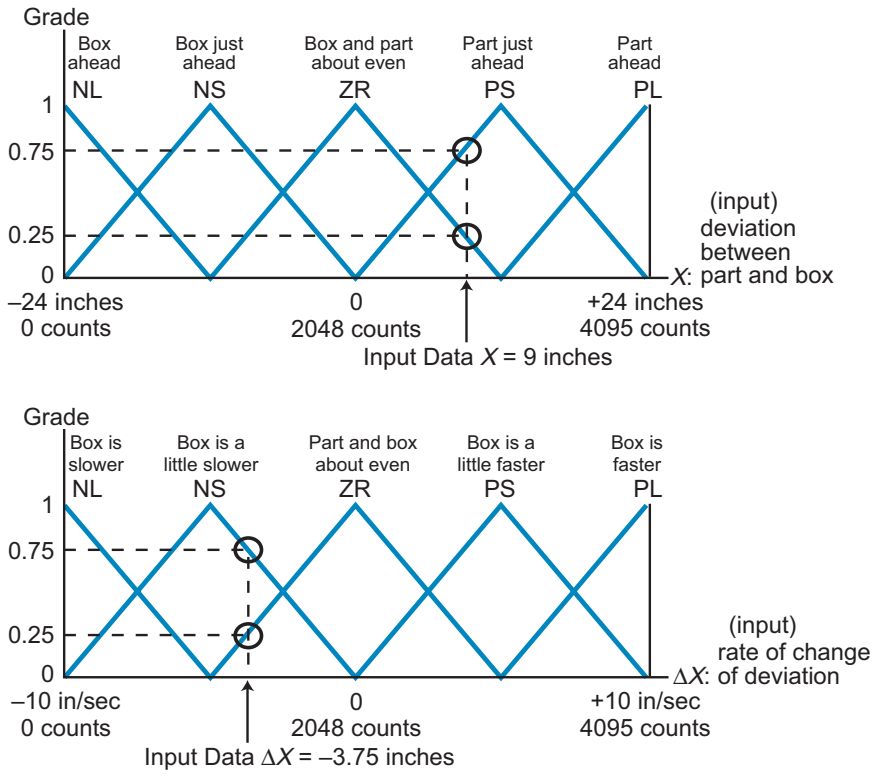


Figure 17-51. Part/box input membership functions.

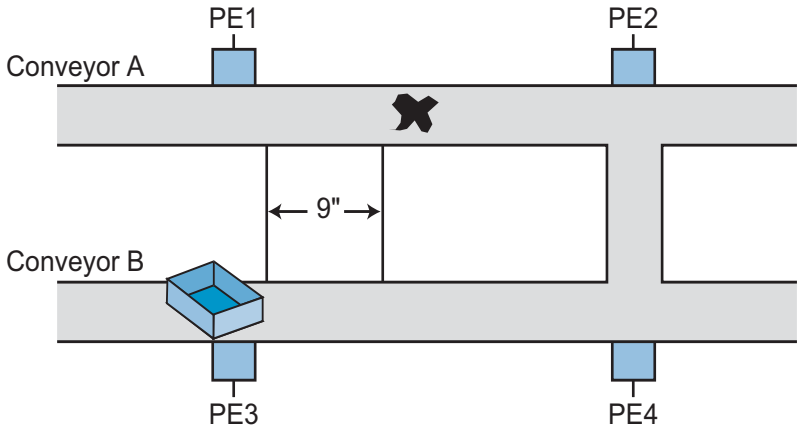


Figure 17-52. Part/box configuration.

SOLUTION

(a) Figure 17-53 shows the four rules that will be triggered by the input reading, along with the selected outcomes and their graphical representation. Note that the outcomes selected are the lowest of the two possible outcomes due to the AND logical link. Note that two of the rules generate a 0.25PS output. The bold line in Figure 17-53b indicates the sum of both outputs (0.25PS × 2 = 0.5PS).

(a)

				Outcomes	Selected
IF X = ZR	AND ΔX = ZR	THEN S = ZR		0.25 due to X 0.25 due to ΔX	0.25ZR
IF X = ZR	AND ΔX = NS	THEN S = PS		0.25 due to X 0.75 due to ΔX	0.25PS
IF X = PS	AND ΔX = ZR	THEN S = PS		0.75 due to X 0.25 due to ΔX	0.25PS
IF X = PS	AND ΔX = NS	THEN S = PM		0.75 due to X 0.75 due to ΔX	0.75PM

(b)

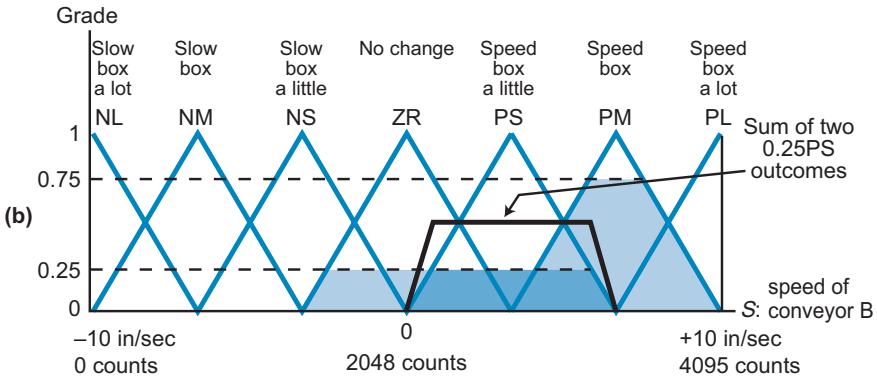


Figure 17-53. (a) Triggered rules and **(b)** their output graphic.

(b) Figure 17-54 shows the logical sum of all the rules' actions. The centroid for this output is located at approximately 2990 counts, which increases the conveyor speed approximately 4.6 inches/second, so that the box can catch up with the part.

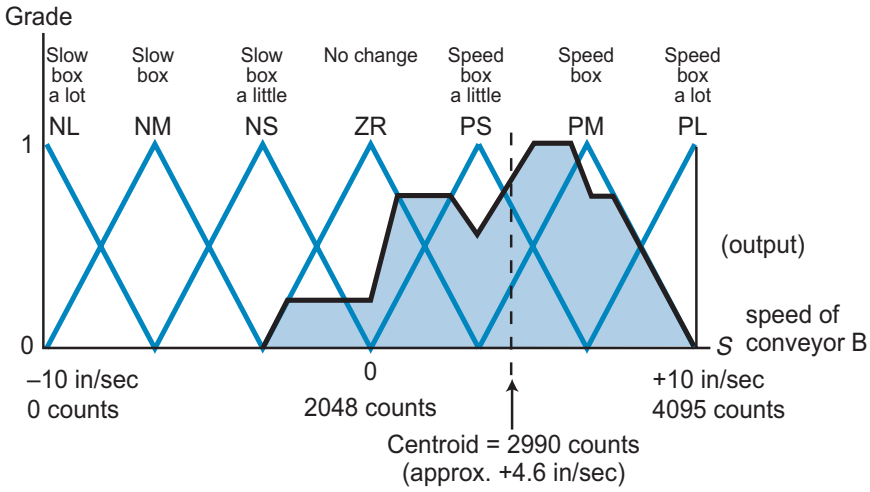


Figure 17-54. Logical sum of the rules' actions and the corresponding centroid.

17-6 FUZZY LOGIC DESIGN GUIDELINES

The guidelines presented in this section provide you with the proper procedures for designing an effective fuzzy logic control system. Although some of these design guidelines are similar to those used in standard PLC systems, others are design requirements that are specific to fuzzy logic systems. The basic elements for the successful implementation of a fuzzy logic control system include:

- control objectives
- control system configuration
- input/output determination
- fuzzy inference engine design

CONTROL OBJECTIVES

Fuzzy logic can be applied to virtually any type of control system, but it is especially suited for applications that rely heavily on human intuition and experience. The primary objective of applying fuzzy logic to an existing process is to improve the overall process and to automate tasks that previously required human judgment. In a new system, the primary objective of using fuzzy logic is usually to implement control that cannot be implemented

using standard control methods. A system designer should not use fuzzy logic control just because it is available. Rather, he or she should use it because it will enhance the system. Otherwise, the outcome may not be enhanced; it may just become confusing.

Typical applications of fuzzy logic involve batching systems and temperature control loops, where process control involves "tweaking" the output based on judgments about input conditions. For example, a temperature control loop application typically requires a knowledgeable operator who can regulate the control element based on decisions such as "if the temperature is a little high but all other inputs are OK, then turn the steam valve a little clockwise." This rationale lends itself to fuzzy logic control.

CONTROL SYSTEM CONFIGURATION

The control objective may lead you to one of several types of system configurations where fuzzy logic can be implemented. Fuzzy logic does not have to be applied only in dedicated fuzzy control applications. It can also be used as a complementary system that supports other more conventional control methods. When used in this manner, the system is said to be a conventional, fuzzy, hybrid control system.

Figure 17-55 illustrates a typical process control system controlled by a PID loop. A fuzzy logic controller could enhance this PID system by regulating the steam volume based on the tank jacket temperature and the batch temperature (see Figure 17-56). If the jacket temperature decreases

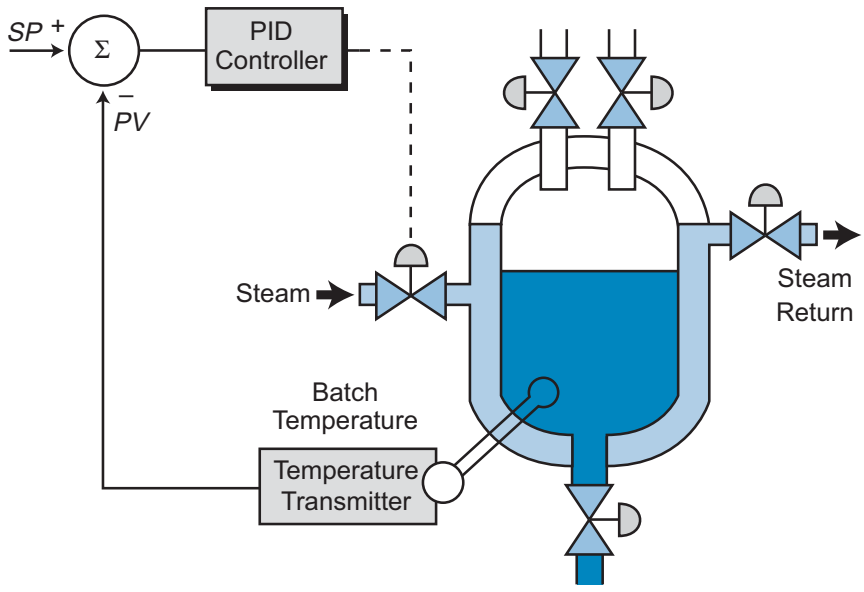


Figure 17-55. PID-controlled heating system.

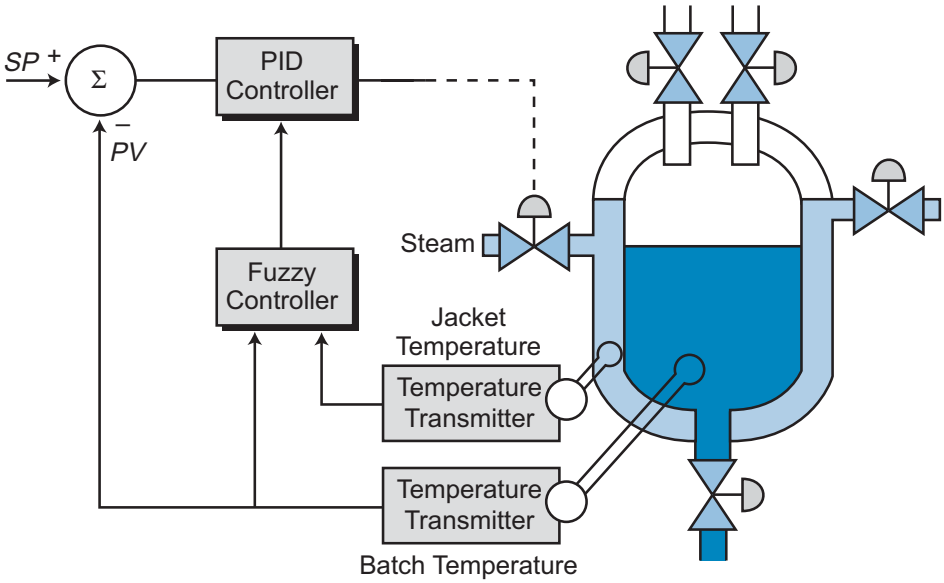


Figure 17-56. PID-controlled heating system with a fuzzy logic controller.

before the batch temperature, the fuzzy controller can take corrective action by suggesting an increase in the steam volume going into the jacket. This operation is similar to cascade control utilizing a fuzzy controller for the inner loop (secondary loop). The fuzzy controller's responsibility is to maintain a proper ratio between the jacket and batch temperatures. Figure 17-57 illustrates several other fuzzy logic system block diagrams, including a pure fuzzy control system.

INPUT/OUTPUT DETERMINATION

Once you have selected the fuzzy system configuration that is appropriate for the control objective, you must determine which inputs and outputs will be used in the fuzzy logic controller. The input conditions, or fuzzy input variables, must be able to be expressed by IF…THEN statements. That is, the input conditions to the fuzzy controller must be able to trigger conditional rules, meaning that they specify one or more output conditions. Inputs should be selected according to the process situations they describe. In other words, if you select two inputs that have little to do with each other, the outcomes that they generate will not be as precise or intuitive as the outcomes generated by inputs that deal with the same process element. For example, referring to Figure 17-56, the batch temperature and jacket temperature both relate to the regulation of the steam valve output. By analyzing these two inputs together, the fuzzy controller can make a precise decision about how much to adjust the steam valve. An analysis of two other unrelated inputs, such as batch temperature and liquid level, would not provide such an informed decision.

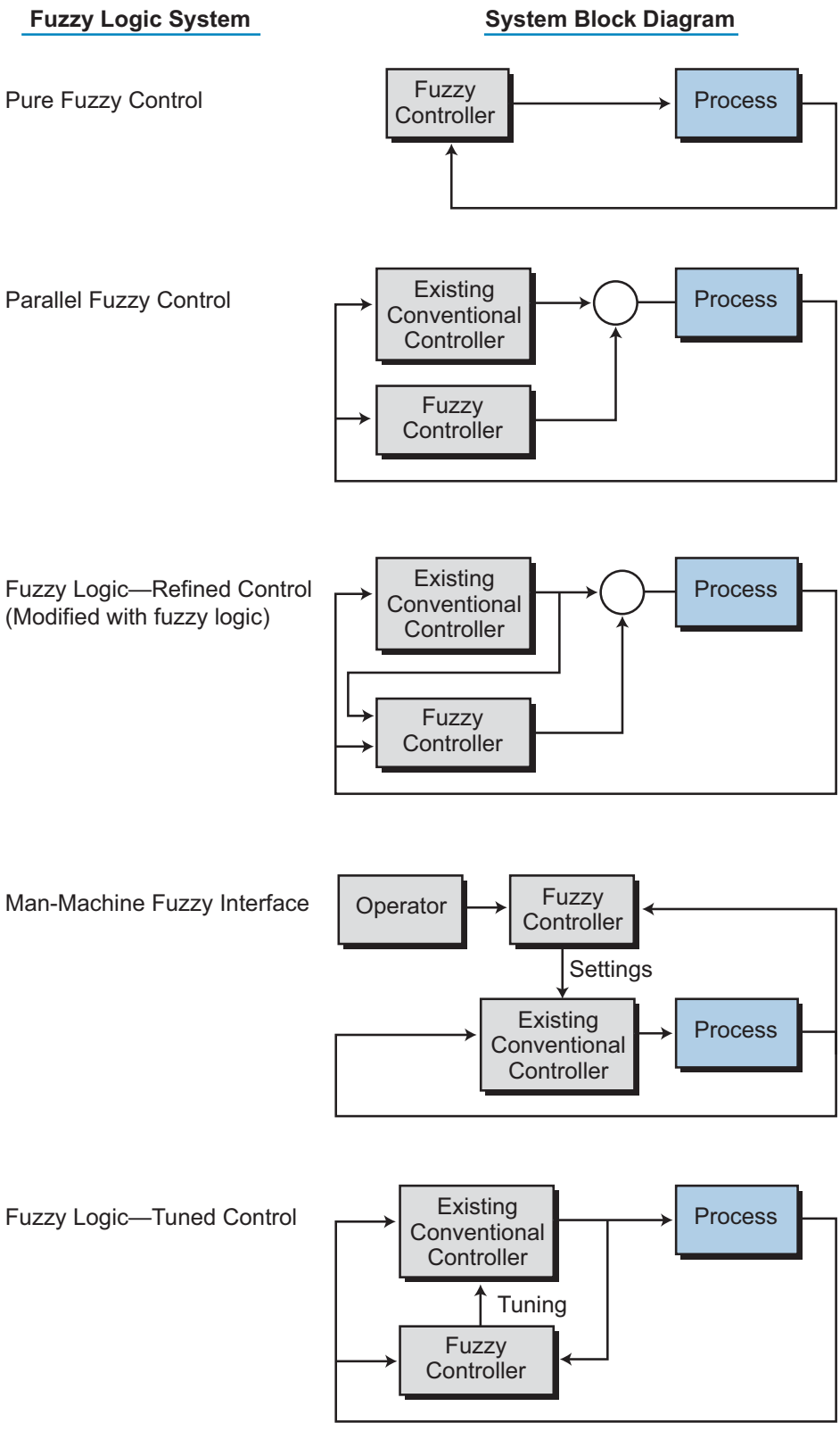


Figure 17-57. Various types of fuzzy logic systems and their block diagrams.

FUZZY INFERENCE ENGINE

The selection of the fuzzy inference engine encompasses the determination of how the fuzzification process will take place (e.g., the number and form of membership function, etc.), how the rules determine an outcome, and how the fuzzy controller implements the defuzzification.

Fuzzification. The fuzzification process, which utilizes the membership functions defined by the user, assigns a grade to each fuzzy input received. This grade determines the level of outcome that will be triggered. Therefore, the shape of the fuzzy set's membership functions is important, since the shape determines the input signals' grades, which are mapped on the output membership function.

Some fuzzy controllers allow the user to choose the shape of the membership functions by trial and error, while others have predefined membership function shapes. When using trial and error to determine the function shapes in a closed-loop fuzzy control system, the input membership functions should begin with overlapped Λ-shaped labels (see Figure 17-58). This ensures smoother control for the first trial due to the coverage provided by the Λ shape and the overlapping at the minimum and maximum points, which creates a balance (i.e., when one label grade is 1, the other is 0). The number of labels, or membership functions, that will form the fuzzy set is also an important part of the system design. For example, if a fuzzy set has five

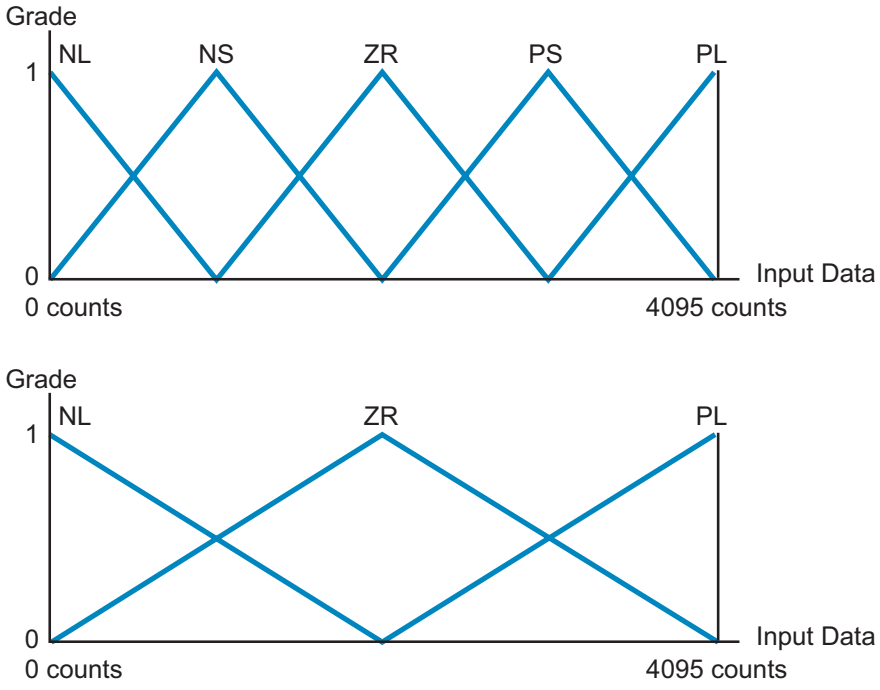


Figure 17-58. Fuzzy input sets using **(a)** five and **(b)** three membership functions.

labels covering the same input data range as a three-label fuzzy set (refer to Figure 17-58), the one with five labels will provide more fine-tuned control, especially if the output membership function also has five labels.

Although membership functions do not have to be symmetrical (see Figure 17-59), asymmetrical fuzzy sets should be carefully designed to ensure that they describe the fuzzy variable input properly. In Figure 17-59, the inner membership functions provide more sensitivity near the zero label (from NS to PS) than at the NL and PL labels (from NL to NS and from PS to PL). Asymmetrical membership functions are typically used in open-loop system applications.

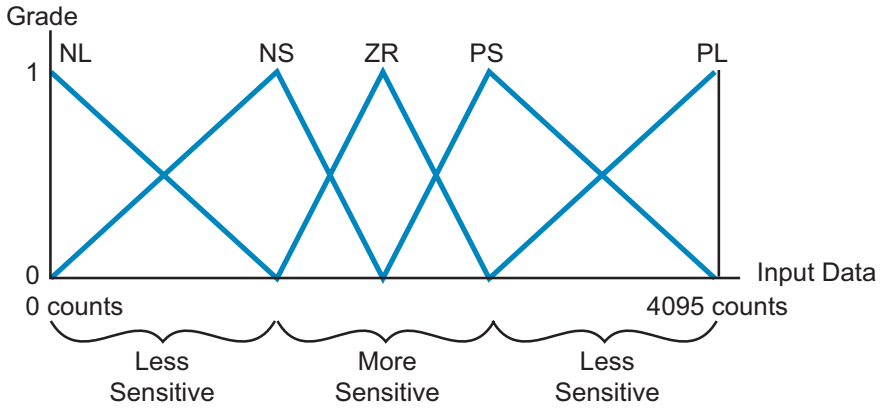


Figure 17-59. Asymmetrical fuzzy input set.

Sometimes, a membership function in a fuzzy set may not provide any sensitivity between two labels. As illustrated in Figure 17-60, the flat sections of the membership functions do not influence neighboring functions or the output. Therefore, the output will not change if the input variable falls in these regions.

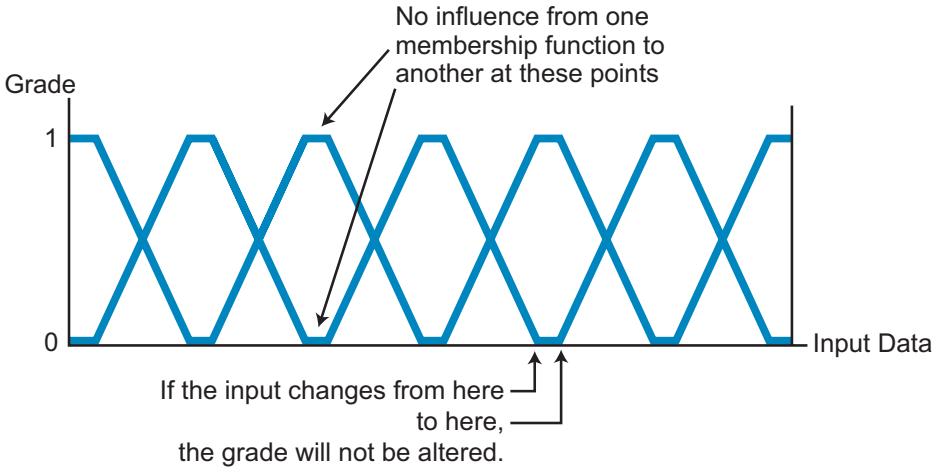


Figure 17-60. Fuzzy input set with no sensitivity at the flat sections between the labels.

Rule Decision Making and Outcome Determination. The easiest way to formulate the rules for a fuzzy logic controller is to first write them as IF…THEN statements that describe how the inputs affect the outcome. Some fuzzy controllers are capable of handling two outputs at the same time, thus allowing two rules to be combined. For example, the rules:

IF A = PS AND B = NS THEN C = ZR

IF A = PS AND B = NS THEN D = NS

can be combined into one rule:

IF A = PS AND B = NS THEN C = ZR and D = NS

This rule gives two outcomes, thus invoking two defuzzification processes, one for each controlling output. It is easiest, however, to create each rule individually (with only one outcome) and then combine them later. If at any point during the rule definition you are uncertain of the operational knowledge required for that particular rule, you should consult a knowledgeable operator so that he/she can provide you with more input information.

As mentioned earlier, you may or may not have a choice of output membership function shapes (Λ, Π, S, or Z). You also may or may not have a choice about whether the functions are continuous or discrete (see Figure 17-61).

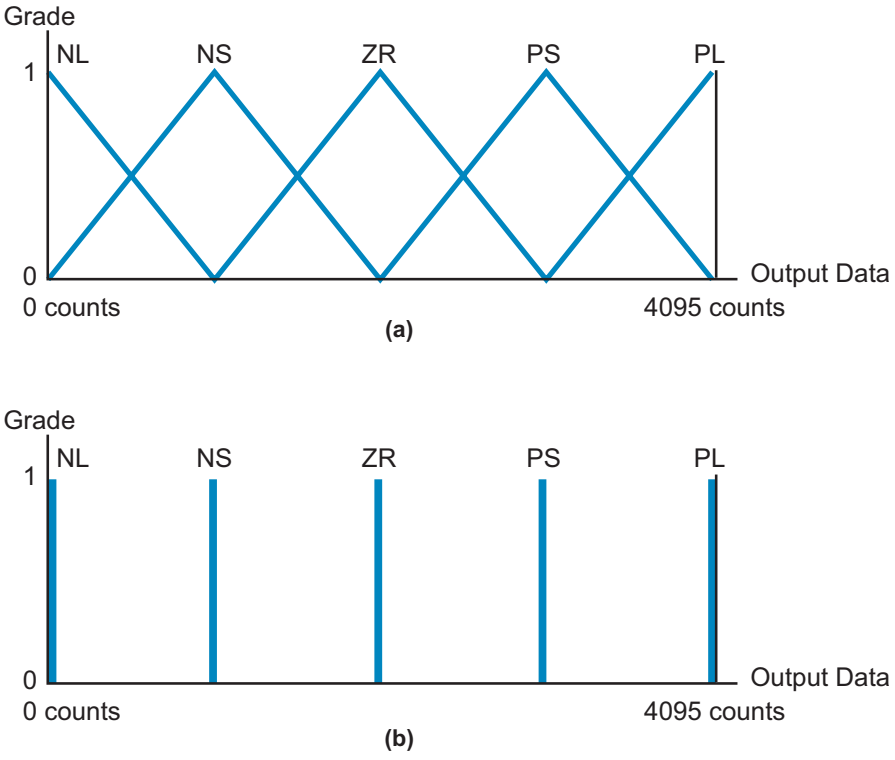


Figure 17-61. Fuzzy output sets with **(a)** continuous and **(b)** noncontinuous membership functions.

Remember that, before defuzzification occurs, the fuzzy controller adds all the outcomes based on the appropriate logic. If the rule contains a logical AND function, the controller will select the *lowest* output value; if the rule contains an OR function, the controller will select the *highest* output value.

If an application requires a highly accurate or smooth output, the rules should be designed so that an input condition triggers two or more rules. To do this, either the input membership functions must overlap or two input conditions must influence the same output (see Figure 17-62).

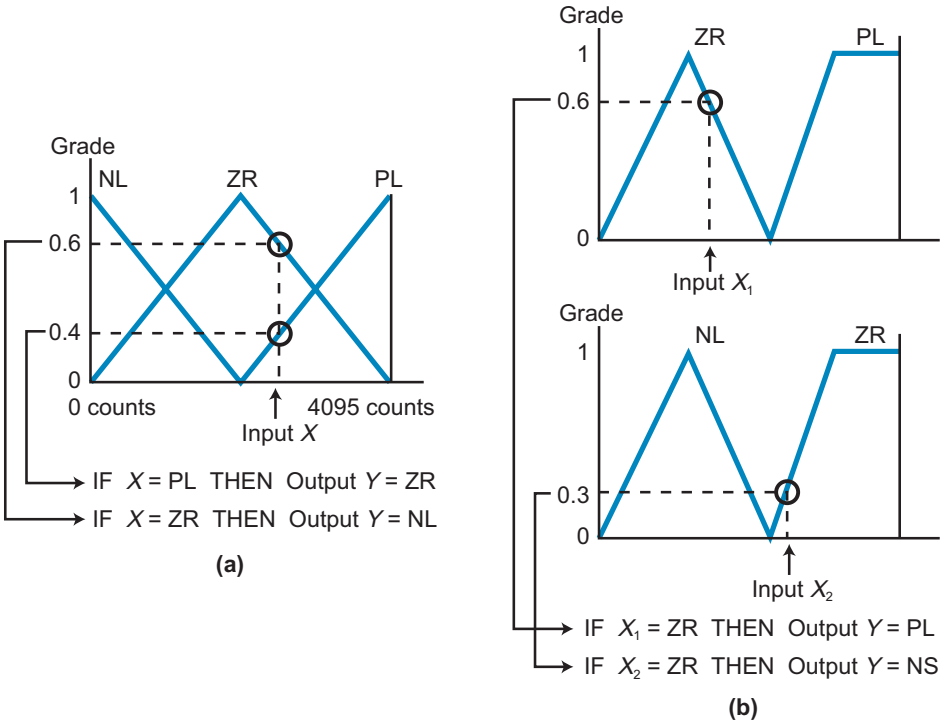


Figure 17-62. Two rules triggered by **(a)** one input in an overlapping membership function and **(b)** two inputs in two nonoverlapping membership functions.

Defuzzification. During the design of a fuzzy logic system, you may be required to choose a defuzzification method, especially if the output membership function is noncontinuous. Defuzzification methods include the center of gravity (centroid), the left-most maximum, and the right-most maximum (see Figure 17-63). If the selected defuzzification method is the center of gravity approach, the triggering rules must be arranged so that at least one rule is triggered at all times. Thus, there must always be an output from a rule. The controller will generate an error if there is no output due to a gap in input condition coverage.

Figure 17-64 illustrates two fuzzy input sets with four rules, which have a potential error condition due to improper coverage of the inputs by the rules defined. For instance, if the X_1 input intersects label ZR at the point where only ZR, and not PL, is triggered (shown as the gap in Figure 17-64a) and

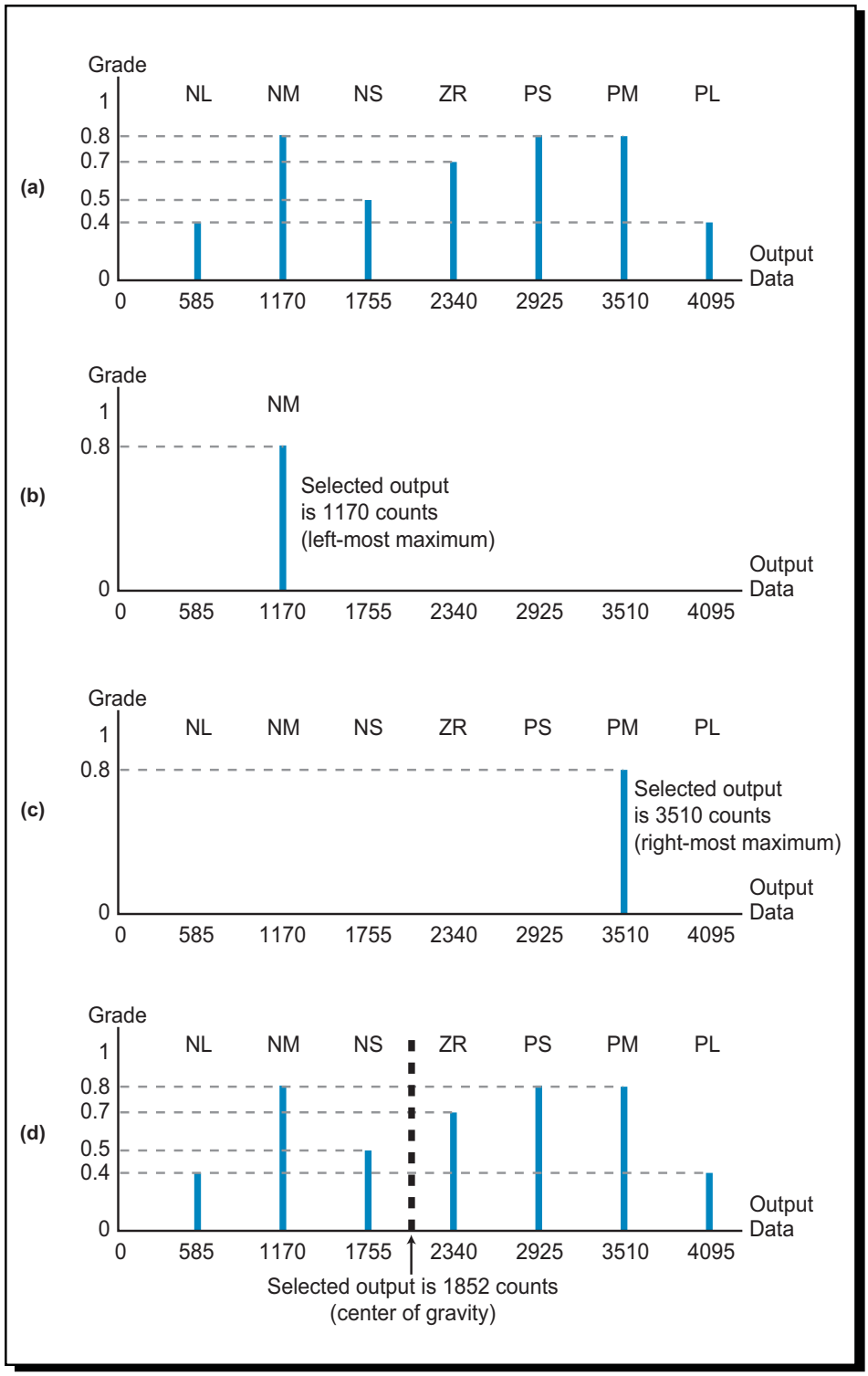


Figure 17-63. **(a)** Seven outputs with the final output selected using **(b)** the left-most maximum, **(c)** the right-most maximum, and **(d)** the center of gravity methods.

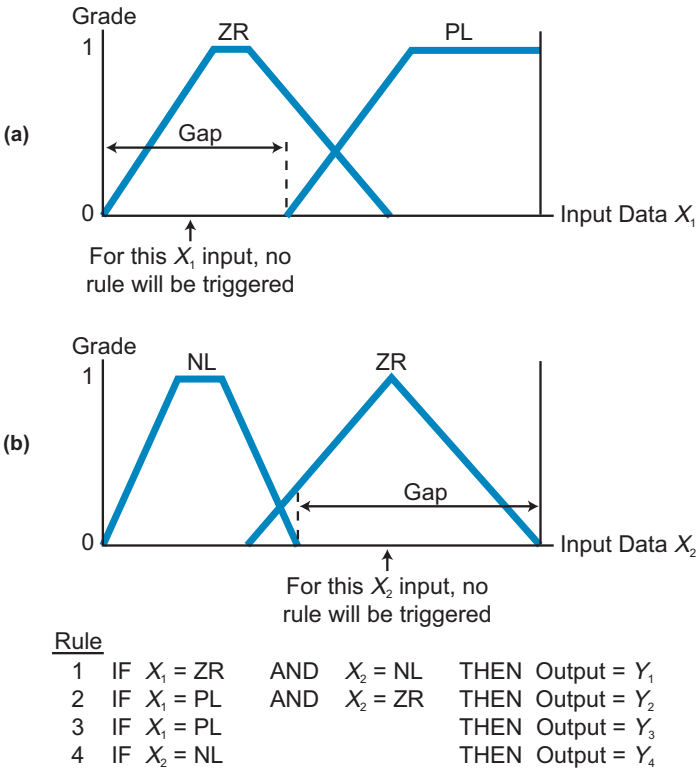


Figure 17-64. Improper coverage of inputs leading to an error condition.

input X_2 intersects label ZR anywhere in the gap area shown in Figure 17-64b, no rule will be triggered. Therefore, no output will be generated and an error will occur. Figure 17-65 shows another gap situation where a region with no sensitivity has no label (membership function); thus, no rule can be triggered. To avoid these potential error conditions, the rules should be designed so that there are no gaps in the rules.

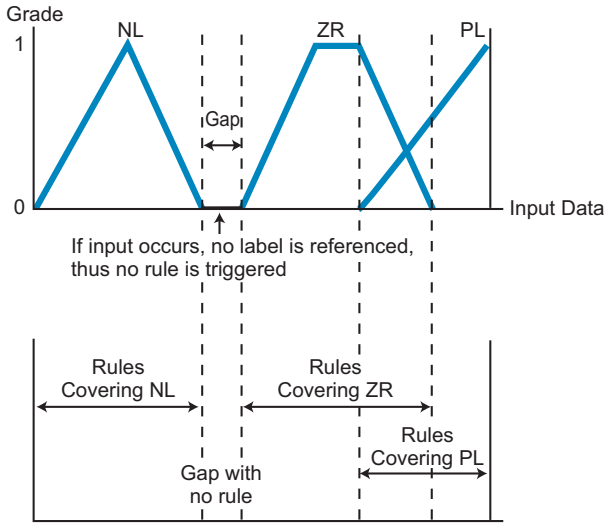


Figure 17-65. A gap in a fuzzy input set.

KEY TERMS

center of gravity method
centroid
defuzzification
fuzzification
fuzzy logic
fuzzy processing
fuzzy set
grade
label
maximum value method
membership function
rule

This page intentionally left blank.