Chapter 6
Modelling and Types of Specifications

6.1 Introduction

The specifications of an optimization problem include all the constraints existing within it. The constraints have been catalogued generally according to their meaning, as Williams (2013) have already pointed out, although proposing a classification of constraints seems to be a rather complex task. The only classification that I consider 100% valid would be the one that refers to the sign of the constraints. There are constraints of three signs: $\leq$, $\geq$, and $=$. However, with this classification, we would not pay attention to the meaning of the specification. And on the other hand, not all specifications correspond to a single constraint. There are specifications, as we will see, that are modelled using more than one constraint.

Therefore, in this methodology, we will propose a classification in two levels. The upper level is determined by the way in which the specification is stated, whereby we would have two types of specifications:

A. Specification stated as a simple proposition
B. Specification stated as a compound proposition

In Sect. 5.3, dedicated to logical calculations, we state the concept of proposition and its typologies. It is important to bear in mind that a proposition in mathematical programming is defined as a mathematical semantic content that, when applied to a solution, can be assigned a truth value (true or false).

Remember also that there are two kinds of propositions:

– Simple: also called atomic, they express a statement that cannot be divided into other propositions because they do not employ any logical operator.

The original version of this chapter was revised. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-57250-1_9

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_6

– Compound: propositions composed of logical operators and one or more simple propositions. In the previous chapter, several examples of the formulation of compound propositions were already shown. In this chapter we will study its modelling.

Specifications Stated as Simple Propositions

The specifications that are formulated as simple propositions give rise to most of the constraints of optimization problems. Simple propositions can also be classified, here with greater difficulty and not exclusively, depending on the meaning or objective of the constraints or the intervening variables. In this methodology, we will define the following types of specifications stated as simple propositions:

Quantitative specification of selection

A specification based on the variables involved. Whenever logical activities are presented on sets of individual elements or binary logical calculations, it is necessary to analyze the existence of this type of specification, since in many cases they are *not explicitly described in the description of the system because they are understood.*

They can be of three types:

– Upper bound
– Lower bound
– Equality

Capacity specifications

Based on the semantics of element data, it is a specification catalogued by some authors. It is based on availability data of elements that have a resource character in the system. The two most common formats in which it is presented in a system are:

– Specification focused on the consumption of capacity: The capacity of the resource can be partially consumed.
– Specification focused on the contribution of capacity: The capacity of the resource is used to satisfy a fixed demand for capacity. In this case, the capacity is used in its entirety.

Supply of a demand

Based also on data semantics, it is the opposite of the capacity specification. It occurs when an element has an attribute that represents a quantity demanded that is necessary to cover or supply.

Bound imposition specifications

These are the most easily recognizable specifications in a system. It would be a typology regarding the way of stating the specification. There are two types:

– Imposition of maximums: these impose upper bounds on variables or functions of variables. Within these we could also include the specifications of capacity consumption.
– Imposition of minimums: these impose lower bounds on variables or functions of variables.

Allocation or balance specifications

These impose values on variables or functions of measurable variables. They also impose a balance between the inputs and outputs of a measurable element, collective or individual. Included in this typology are constraints on the distribution of stocks or those that impose equality between variable functions.

It is inevitable that sometimes certain specifications will be considered as belonging to more than one typology.

Specifications Stated as Compound Propositions

The specifications enunciated as compound propositions have already appeared in the chapter dedicated to the logical calculations of a system. They are specifications that are based on a logical language, using logical operators or a connective to define the specification. They are also usually easily identifiable in the system description. As we already mentioned, the modelling of logical propositions could involve the definition of logical calculations. In this chapter, we will see the modelling of any type of compound proposition.

The formulation of logic-based propositions has been studied mainly by Mitra et al. (1994), Williams (1995), and Williams (2009) proposing constraints using binary decision variables, particular propositions as disjunctive constraints but there is not a general method for formulating any compound propositions with any connective.

In this book we will see a general scheme for modelling compound propositions where the basic rules are based on the modelling of propositions already described by those authors.

The structure of this chapter is as follows: first (Sect. 6.2) we will analyze the elements on which the specifications fall. In Sects. 6.3 to 6.7, we will present the modelling of each type of simple specification (A.1 to A.5). Section 6.8 will deal with the modelling of compound propositions. Section 6.9 is devoted to objective functions, since objective functions can be considered as one more specification of the system. In Sect. 6.10, we will analyze the identification of specifications in the description of a system.

6.2 Elements on Which the Specification Falls on

When it comes to modelling a specification, the first task is to identify to which elements the specification is directed. If it is an individual specification, it will be directed to a particular element within a set, to an element that is not part of any set or to the system itself. If the specification is directed to a set of elements, the construction of the specification uses terms such as "*Every*" or "*A*" as an indeterminate article, without referring to one in particular but to any. This means that the specification will have to be mounted for each element of the set. We can even express a specification using the term "*all*." In this case the norm must also be mounted for each element individually if they are defined individually.

Illustration 6.1
In a system for assigning jobs to machines, with those elements defined as unitary, the specification that assigns each job to a machine could be defined as:

- *Each job must be assigned to a machine.*
- *A job will be assigned to a machine.*
- *All jobs must be assigned to a machine.*

It could also be that the specification does not apply to all elements of the set but only to those that meet a condition subject to the values of their data or their indices. This will be expressed using the mathematical symbol "such that" (*/*):

$\forall i/C_i>10$: . . . . (C_i is an attribute of the elements of the index set i)
$\forall i,j/i>j$: . . . .
$\forall i/i\geq 3$: . . . .

We should never express variables in the clause *such that*. As its value is not determined, we will not know whether or not the specification is applied:

$\forall i/x_i>10$: . . . . **Error!**

Whenever we try to define variables in the clause "such that," we must define a conditional logical proposition "IF. . .THEN. . ." that uses these variables as an input condition of the conditional. The logical propositions will be studied in Sect. 6.8. Let's look at a simple example:

$$\forall i/x_i > 10 : \alpha_i + \beta_i \leq 1 \Rightarrow \ \forall\, i : \text{IF}\, x_i > 10\, \text{THEN}\, \alpha_i + \beta_i \leq 1$$

It may also be common for specifications to be defined by the combination of several sets of elements. Again, the determination of the elements depends on the way in which they are alluded to in the statement of the specification, if in a determined way on some elements or in an indeterminate way, which is why all the combinations that define the specification should be considered. Let us look at an example:

Illustration 6.2: Distribution of Ham
A ham distribution company has designed a set of 20 delivery routes for distribution. The company has a portfolio of 350 clients. Each delivery route goes through a series of known customers. The company has ten vehicles for the distribution stage. Each vehicle has a given capacity or number of Iberian hams that it can transport.

The demand for ham that must be supplied to each customer is known. Each vehicle that delivers ham must choose a single route, since more than one would take too long.

Two vehicles cannot choose the same route.

If a vehicle delivers more than 50 Iberian hams to a customer, it should not deliver ham to any other customer.

We know the delivery cost of each route.

Table of Elements (Table 6.1)

Table 6.1 Table of elements of Illustration 6.2

Elements	SET	QN	Data				
			Name	Param	Type	Belonging	Value
Routes	$i = 1...20$	I_U	Customer_Route	RC_{ij}	B	S	...
			Cost	C_i	C	W	...
Customers	$j = 1...350$	I_U	Demand	D_j	I	S	...
				RC_{ij}			
Vehicles	$k = 1...10$	I_U	Capacity	K_k	I	S	...
Iberian hams	–	C_D		D_i; K_k			

In the Customer_Route attribute, the customers by which each route passes are annotated. It is therefore shared between routes and customers.

Decision Activities

Action: Deliver [Iberian hams to customers with vehicles]
Decision variables:
x_{kj} = Number of Iberian hams delivered with vehicle k to customer j.
$k = 1...10, j = 1...350$

Action: Choose [Routes for vehicles]
Decision variables:
α_{ik} =1 if I choose Route i for Vehicle k; 0 otherwise. $k = 1...10$, $i = 1...20$

Let us analyze two of the specifications of the system:

Specification 1: *Two vehicles cannot choose the same route*

The first question would be: which two vehicles? They are not determined, so we should consider the combination of every two vehicles from the ten vehicles. Obviously, since vehicles are individuals, this value (two vehicles) cannot represent a numeral of a collective element.

The second question would be: which route? And again, the answer is indeterminate, so we should consider them all.

Therefore, the specification is presented for each pair of vehicles and each route. For vehicles, we can define a second subscript k'. The specification is defined by a logical statement:

$$\text{Logical proposition} : \forall k, k', i : \text{NOT } (\alpha_{ik} = 1 \text{ AND } \alpha_{ik'} = 1) \tag{6.1}$$

This specification could also be written as follows: The number of vehicles that can choose each route must be at most 1:

$$\forall i : \sum_{k=1}^{10} \alpha_{ik} \leq 1 \tag{6.2}$$

Specification 2: If a vehicle delivers more than 50 hams to a customer, it should not deliver ham to any other customer:

"a vehicle": indeterminate, any vehicle
"a customer": indeterminate, any customer

"any other customer": the rest of customers is also indeterminate, any customer.

$$\text{Logical proposition} : \forall j, j'/j' <> j, k : \text{IF} x_{kj} > 50 \text{ THEN} x_{kj'} = 0 \tag{6.3}$$

The specification could also have been defined with the following equivalent statement: If a vehicle delivers more than 50 Iberian hams to a customer, the sum of Iberian hams to the rest of customers will be zero:

$$\text{Logical proposition} : \forall j, k : \text{IF} x_{kj} > 50 \text{ THEN} \sum_{j' \neq j} x_{kj'} = 0 \tag{6.4}$$

Both in (6.2) and in (6.4) the number of specifications is reduced, which a priori may contribute to improving the resolution times of the model.

6.3 Quantitative Specifications of Selection

Specifications are very common in modelling. They are relevant when we work with variables of logical decision and can also be enunciated with respect to a set of binary logical calculations.

This specification expresses a quantitative condition with respect to the set of binary variables. It is to establish the amount of choices that are going to be given or that can be given as a maximum or minimum.

Format: Sum of Selection options SIGN Value
General Expression:
$\alpha_1 + \alpha_2 + \ldots + \alpha_n \approx Value$
Regarding the SIGN ($\approx$):

$=$Exact imposition or imposition of equality
$\leq$ Imposition of upper bound
$\geq$Imposition of lower bound
Value =Quantification of the selection

The sum of binary variables expresses the set (or sets) of elements that are selection options. The most frequent case of this type of specification occurs between more than one set of individual elements, normally of unitary nature, which are combined in the events of the decision activity, although it is also possible to find it in a single set of elements.

As we will see in Sect. 6.10, this type of specification may not appear explicitly in the description of the system, so we will always have to do an analysis exercise of the quantitative selection rules when we have systems with logical activities.

Let us consider some examples of the specification in different scenarios:

Illustration 6.3: Case of a Set of Unitary Elements
Let us suppose there is a series of activities that make reference to the use of a unitary set of machines:

Elements: Machines $i = 1...n$ Unitary
Decision activities:
Use (Machines $i = 1...n$) $\Rightarrow \alpha_i = 1$ if I use machine i; 0 otherwise

Quantitative specifications of selection: In this system, specifications such as the following could be considered:

- *You must use (select) at least one machine:* $\sum_{i=1}^{n} \alpha_i \geq 1$
- *You must use three machines:* $\sum_{i=1}^{n} \alpha_i = 3$

Illustration 6.4: Case of Two Sets of Elements
Let us suppose there is a series of operators that must be assigned to a series of machines:

Elements: Machines i=1...n Unitary; Operators j=1...m Unitary
Decision activities:
Assign (Operators to Machines) $\rightarrow$ α_{ij}=1 if I assign operator j to machine i; 0 otherwise.
Quantitative specifications of selection:
Maximum two operators per machine: This specification falls on each machine of the system:

$$\forall i : \sum_{j=1}^{m} \alpha_{ij} \leq 2$$

An operator must be assigned only one machine: The specification refers to any operator, since it refers indeterminately to an operator. On the other hand, with machines it refers to the quantity of a machine.

$$\forall j : \sum_{i=1}^{n} \alpha_{ij} = 1$$

If we assume a power attribute (P) for the machines, we could find a specification that does not consider, as a selection option, all the elements of the set of machines:

A machine with power greater than 10000w will be assigned to operator 3:

$$j = 3 : \sum_{i/P>10000} \alpha_{ij} = 1$$

Illustration 6.5: Case of a Set with Choices About the Set Itself
Let us suppose there is a series of elements that must be assigned among them. Each element must be associated with another element of the same set:

Elements: Elements i=1...n Unitary;
Decision activity:

Assign [Elements to Elements] → α_{ij}=1 if I assign element i to element j; 0 otherwise.

Quantitative specifications of selection:

- Each element is associated with a different element of itself.
  $$\forall i : \sum_{j/j \neq i} \alpha_{ij} = 1$$
- Since i and j are indices of the same set and is a bi-univocal correspondence, the association of i with j is the same as that of j with i, therefore: $\forall i, j : \alpha_{ij} = \alpha_{ji}$

Illustration 6.6: Case of Two Sets That Share the Selection
Let us suppose that we have divided elements of the same functionality in the system into two sets due to data, but both share the same logical decision activity on which a quantitative selection rule is applied.

Elements:
Individuals i=1...n Unitary;
Groups_Type1 j=1...m_1 Unitary;
Groups_Type2 k=1...m_2 Unitary;
Decision activity:
Assign [Individuals to Groups_Type1 and Groups_Type2] →
→Events: Individuals to Groups_Type1; Individuals to Groups_Type2 →
→ α_{ij}=1 if I assign Individual i to Group_Type1 j; 0 otherwise.
→ β_{ik}=1 if I assign Individual i to Group_Type2 k; 0 otherwise.
Quantitative specifications of selection:
One individual in a group at most:
$$\forall i : \sum_{j=1}^{m_1} \alpha_{ij} + \sum_{k=1}^{m_2} \beta_{ik} \leq 1$$

6.4 Capacity Specifications

Capacity specifications are ones that appear in systems where there are elements acting as resources. For this, they must have an attribute of capacity or availability, either intrinsic to the element or shared with a measurable element, collective or individual. They usually appear in different ways that are synthesized in the following general format:

Format: *Capacity Consumption or Demand for Capacity* ≤ *Capacity Contribution*

Expression: Both the consumption or demand for capacity and the contribution of capacity can be fixed and/or variable. Capacity contribution and consumption must be expressed in the same unit of measure. In the same specification we can have both a fixed and a variable consumption or contribution at the same time.

Capacity consumption or demand		Capacity contribution	
Variable	Fixed	Variable	Fixed

Variable Capacity Consumption
This corresponds to the classic consumption format in a capacity specification.

$$\text{General expression of consumption} : c_1x_1 + c_2x_2 + \ldots + c_nx_n \qquad (6.5)$$

The capacity consumption is made up of a series of variables represented as x_1, x_2, …, x_n. Each activity performs a consumption that is given by the expression c_ix_i where c_i is the unit consumption of activity x_i (the amount of capacity consumed by each unit of value that variable x_i takes).

Fixed Capacity Consumption or Demand
This corresponds to an attribute of the element that demands that capacity. It appears in systems where a set of elements provide capacity of a measurable element that has a stock that corresponds to the fixed capacity demand. Decisions focus on the capacity contribution, not on the amount that is consumed or demanded, which is fixed.

Fixed Capacity Contribution
An element provides a given capacity, collected as an attribute. Decisions are focused on how much of that existing capacity is consumed. The specification falls on the proprietary element of that fixed capacity.

Variable Capacity Contribution
This usually appears in specifications where a fixed capacity is demanded. The general expression corresponds to an expression similar to (6.5):

General expression of contribution: $a_1y_1 + a_2y_2 + \ldots + a_my_m$

The variables y_j, $j = 1\ldots m$, represent activities that provide capacity, with a_j being the unit contribution.

There is a set of elements that provide capacity to an element that demands capacity. Decisions are focused on the contribution of capacity, not on the quantity that is consumed or demanded, which is constant. The system does not collect the amount of capacity consumed on each element that contributes capacity, but it collects the amount of capacity contributed.

This case is very similar to the next type of specification, the specification of demand contribution. In any case, they are treated here independently because semantically they are different. In the specification of demand contribution, a

Table 6.2 Most common formats in a capacity specification (Table 6.2)

Cases	Format	Expression
1	*Variable capacity consumption* $\leq$ *fixed capacity contribution*	$c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq K$
2	*Variable capacity consumption* $\leq$ *fixed and variable capacity contribution*	$c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq K$ $+a_1y_1 + a_2y_2 + \ldots + a_my_m$
3	*Fixed capacity consumption or demand* $\leq$ *variable capacity contribution*	$E \leq a_1y_1 + a_2y_2 + \ldots + a_my_m$

quantity of a measurable element is demanded since it is not possessed, and some elements can contribute to it. Here, we already have a quantity of a measurable element and capacity for this is requested. Basically, it is a semantic differentiation.

The most common formats of capacity specifications are collected in Table 6.2.

Let us analyze each case.

6.4.1 Case 1: Variable Capacity Consumption and Fixed Contribution

The relationship is established between the capacity consumption of an element and the fixed amount of capacity that the element has in the system.

Format: *Consumption* $\leq$ *Capacity*
General Expression: $c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq K$

This type of specification is usually not presented explicitly in the system description, as we will see in Sect. 6.10, but the modeller must detect them from the data of the table of elements.

Illustration 6.7: Production of Butter
We have already analyzed this problem amply. The specifications that are presented are only of capacity.

Table of Elements (Table 6.3)

Table 6.3 Table of elements

Elements	Set	QN	Data				
			Name	Param	Type	Belong	Value
Machines	$i = 1\ldots2$	I_M	Usage time	T_i	C (Min)	P	{6,3.5}
			Time consumed by 1 kg of butter j in machine i	TM_{ij}	C (Min)	C	{3m;3m; 3m;6m}
Butters	$j = 1\ldots2$	I_M	Profit	B_j	C ($)	P	...
				TM_{ij}			

Decision Activities

Action: Produce Butters
Decision variables: x_j = Amount of butter type j produced; $j = 1,2$;

Capacity Specifications

The Whipping Machine ($i = 1$) has a usage time of $T_1 = 6$ hours which is measurable.

Variables x_1 and x_2 consume TM_{11}=3 min. and TM_{12}= 3 min.

Expressing the specification in minutes: $3x_1 + 3x_2 \leq 360$

$$\text{Parametric} : \sum_{j=1}^{2} TM_{1j} x_j \leq T_1 \tag{6.6}$$

– The Pasteurization Machine ($i = 2$) also has a T_2 usage time:

$$\sum_{j=1}^{2} TM_{2j} x_j \leq T_2 \tag{6.7}$$

And the parameterized expression that collects (6.6) and (6.7) would be:

$$\forall i : \sum_{j=1}^{2} TM_{ij} x_j \leq T_i$$

Although in this type of specification it is usually more common for consumption to be carried out by measurable activities, logical activities can also participate as consumption, as in the following illustration.

Illustration 6.8: Management of a Warehouse
Management of a warehouse in which it is necessary to place a set of Pieces ($i = 1..n$) in a set of Shelves ($j = 1 \ldots m$). Each piece has a weight and a volume. Each shelf has a load capacity and a volume.

Capacity specifications:
Do not load a shelf with more weight than it can support.
Do not load a shelf with a total volume higher than its own.

Table of Elements (Table 6.4)

Decision Activities

Action: Assign Pieces to Shelves
Decision variables: $\alpha_{ij} = 1$ if I assign piece i to shelves j; 0 otherwise.

Table 6.4 Table of elements of illustration 6.8

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Pieces	$i = 1 \ldots n$	I_U	Weight	p_i	C	W	...
			Volume	v_i	C	W	...
Shelves	$j = 1 \ldots m$	I_U	Load	PE_j	C	W	...
			Volume	VE_j	C	W	...

Capacity Specifications

Do not load shelves with more weight than it can support: specification applied to each of the shelves

$$\forall j : \sum_{i=1}^{n} p_i \alpha_{ij} \leq \mathrm{PE}_j$$

Do not load a shelf with a total volume higher than its own: also applied to each of the shelves:

$$\forall j : \sum_{i=1}^{n} v_i \alpha_{ij} \leq \mathrm{VE}_j$$

6.4.2 Case 2: Variable Consumption with Fixed and Variable Capacity Contribution

Although it is less common, we can also find activities to increase capacity in addition to consumption. Let us take a look at the following illustration.

Illustration 6.9: Production of Pills
600 Kg of a certain drug is available to make large and small pills. We also have the possibility of buying more drugs at a price of $0.05/gram.
The big pills require 40 g and the small ones 30 g.
Each large pill provides a profit of $2 and a small one of $1.
The manufacturing capacity is 2000 pills.

Table of Elements (Table 6.5)

Decision Activities
Action: Produce [pills]
Decision variables: x_i = Number of pills produced of type i

Action: Buy [drug]
Decision variables: z = Amount of drug bought

Table 6.5 Table of elements of Illustration 6.9

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Drug	–	I_M	Availability	E	C	W	600 Kg
			Price	p	C	W	$0.05/gr
			Amount in pills	c_i	C	S	{40,30}
Pills	$i = 1,2$	C_I		c_i			
			Profit	b_i	C	W	{$2, $1}
Factory (System)	–	I_U	*Manufacturing capacity*	K	I	W	2000

Capacity Specifications

(Case 2) Drug capacity: relapses on the drug. There is a capacity contribution collected in the variable z.

$$\sum_{i=1}^{n} c_i x_i \leq E + z$$

(Case 1) Manufacturing capacity: applied in the system.

$$\sum_{i=1}^{n} x_i \leq K$$

6.4.3 Case 3: Fixed Capacity Demand and Variable Capacity Contribution

There are systems where elements with capacity data and therefore acting as a resource do not individually define a capacity consumption specification. The reason is that the problem can be modelled without there being activities that measure (discretely or continuously) the consumption of that resource. The capacity attribute is not measured because it is assumed that it has either not been used or used in its entirety. The system has an element that demands a capacity, and it is necessary that other elements provide that capacity. Sometimes this specification occurs because there is a capacity attribute that is assigned to each item of a collective element. Since the instances are not treated individually but collectively, the capacity attribute cannot be used for a capacity consumption specification, since this would be applied individually on each item. The capacity attribute is used as contribution. There are also cases where the capacity attribute is held by an individual element that acts as unitary.

The format of the specification would be:

Format: *Fixed capacity demand* $\leq$ variable *capacity contribution*

Let us first look at an example that illustrates the case of capacity data imputable to each item of a collective element (illustration 6.10), and a second illustration where the capacity attribute is possessed by individual elements. In this second illustration we will propose the two possible versions: measuring capacity through consumption and not measuring the capacity of each individual element, by using it logically in a capacity contribution specification (illustration 6.11).

Illustration 6.10: Celebration Hall

There is a celebration hall where a wedding will take place. The hall has tables of two sizes. There are tables of 8 seats and tables of 12 seats, 14 and 11 of them, respectively.

150 guests attend the wedding. The company must decide which tables to use so that the guests can sit down, minimizing the number of tables used.

Table of Elements (Table 6.6)

Since each type of table is formed by an identical set of items and the text does not refer individually to them, the tables are configured as collective. The same happens with the guests. We make explicit the seats as element although there are no decisions about the seats but so there is a demand for seats for the guests.

Regarding the tables, there are two sets of data on capacity: availability of tables and the number of seats. The availability is applied to the element globally or collectively, while the number of seats is capacity attribute regarding each item of the collective element, so that attribute will not be associated with a specification of capacity consumption.

Decision Activities

The system imposes the action of using tables as a decision activity. We do not need to decide where to locate the guests; we simply select tables and establish the specification of placing 150 guests, based on a certain value action corresponding to a specification of capacity demand.

Table 6.6 Table of elements of Illustration 6.10

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Tables	$i = 1,2$	C_D	Availability	N_i	I	W	{14,11}
			Seats	K_i	I	S	{8,12}
Seats	–	C_D		K_i			
Guests	–	C_D	Number	NG	I	W	150

Action: Use tables
Decision activities: x_i = Number of tables i used

Specifications

To seat the 150 guests: We can understand the specification as the guests demanding a capacity of 150 seats. The tables act as a resource for these places. Each type of table contributes a certain number of places:

Fixed capacity demand $\leq$ *Variable capacity contribution*

$$150 \leq 8x_1 + 12x_2$$

The capacities of the tables are being used as a contribution. The capacity refers to each item of the table.

Table availability: This attribute does give rise to a capacity consumption specification for each type of table

$$\forall i: \quad x_i \leq N_i$$

Objective Criterion

$$\text{Min} \quad x_1 + x_2$$

Illustration 6.11: Containers
There is a set of n containers each with a capacity of volume and a cost. There is a liquid product with an amount of C. The objective consists in the selection of containers at a minimum cost so that we can store the quantity C of the product.

Version 1: By Capacity Consumption
In this first version, we analyze the system as a system to store liquid in containers. Therefore, the capacity data will be measured in the specifications. The liquid is a measurable element by its own continuous quantity used partially.

Table of Elements (Table 6.7)

Decision Activities

Table 6.7 Table of elements of Illustration 6.11 – Version 1

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Containers	$i = 1...n$	I_U	Cost	c_i	C	W	...
			Capacity	K_i	C	W	...
Liquid product	–	I_M	Amount	Q	C	W	...

Action: Store liquid product in containers
Decision variables: x_i = amount of liquid product introduced in container i

We measure the liquid because it is the direct object of the action, "store liquid."

Specifications

Capacity consumption: In this scenario a capacity consumption specification is to be proposed for each container, to ensure that the quantity introduced does not exceed the capacity of the container:

$$\forall i : x_i \leq K_i \tag{6.8}$$

Balance specification: Although we have not yet seen the balance specifications, in the system, a balance specification is generated with the liquid availability data. The amount of liquid distributed by all the containers corresponds to the quantity Q.

$$\sum_{i=1}^{n} x_i = Q \tag{6.9}$$

Objective Criterion

Since we minimize the cost of the used or *selected* containers, it is necessary to calculate that qualifier for the containers, which becomes a logical calculation on the containers (if you store in a container, it is because you have selected it). If we had considered it as a decision activity, that is, on the one hand store and on the other select, the implicit relationship between the two activities cannot be ignored as a specification: If I store then I select.

Binary logical calculation: Selected container
Applied to: Containers i=$1 \ldots n$
Definition of logical variable: $\alpha_i = 1$ if I select container i; 0 otherwise
Logical proposition: $\forall i : {}_i = 1$ IF AND ONLY IF $x_i > 0 \Rightarrow$ Ref. S_V (The selection of a container has a cost so it is useless to select containers if you do not put anything in them [Sect. 5.3.3])

$$\Rightarrow \forall i : \text{IF} \quad x_i > 0 \quad \text{THEN} \quad \alpha_i = 1 \tag{6.10}$$

In spite of not having presented the modelling of logical propositions yet, we are going to write the constraints that are generated from modelling (6.10), since it will be necessary for the simplification process between versions. The constraints generated are:

$$\forall i: \quad x_i \leq K_i \alpha_i \tag{6.11}$$

where K_i acts as the upper bound of what can be stored in each container.

The objective function would be:

$$\text{Min} \quad \sum_{i=1}^{n} c_i \alpha_i \tag{6.12}$$

Version 2: By capacity contribution

In this second version, we will not introduce liquid into the containers, and therefore we will not measure at the specifications the capacity of the containers. The only thing we are going to do as a decision activity is select containers, making sure that the liquid fits. The liquid becomes unitary because we stop making a partial use of its quantity in the decision activities.

Table of Elements (Table 6.8)

Decision Activities

Action: Select [Containers]

Decision variables: $\alpha_i = 1$ if I select container i; 0 otherwise.

Specifications

1. *Capacity contribution*: The liquid element demands a capacity for its quantity. Each container i has a capacity contribution of value K_i.

$$Q \leq \sum_{i=1}^{n} K_i \alpha_i \tag{6.13}$$

Objective Criterion

Minimize costs:

Table 6.8 Table of elements of Illustration 6.11 – Version 2

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Containers	$i = 1..n$	I_U	Cost	c_i	C	W	...
			Capacity	K_i	C	W	...
Liquid product	–	I_U	Amount	Q	C	W	...

$$\text{Min} \quad \sum_{i=1}^{n} c_i \alpha_i \tag{6.14}$$

Relation Between Versions
Objective functions (6.12) and (6.14) are identical. Version 1 is equivalent to version 2 but logically larger. If we perform two simple operations, we can get to version 2 from version 1:

On the one hand, we have in Version 1:

- Modelling of the logical proposition (6.11): $\forall i: \quad x_i \leq K_i \alpha_i$
- Capacity specification (6.8): $\forall i: x_i \leq K_i$

It is only necessary to use (6.11). If I select the container (α_i=1) I cannot store more than its capacity, and if I do not select the container (α_i=0), then $x_i \leq 0 \Rightarrow x_i = 0$.

On the other hand, if we add the n constraints of (6.11), we obtain:

$$\sum_{i=1}^{n} x_i \leq \sum_{i=1}^{n} K_i \alpha_i \tag{6.15}$$

Substituting (6.9) with (6.15):

$$\sum_{i=1}^{n} x_i = Q \leq \sum_{i=1}^{n} K_i \alpha_i$$

This corresponds with (6.13), the only constraint that the first version had. In this way, it is guaranteed that at least quantity Q can be stored, and the use of decision variable x_i is not necessary.

6.5 Supply of a Demand

This is a specification which is analogous to the capacity contribution specification. It expresses a relationship between the supply of a measurable element, collective or individual, and the demand requested of that measurable element.

Format: *Measurable element supply* $\geq (=)$ *Measurable element demand*
General Expression:
The supply is considered variable in the specification. Demand is considered both fixed and variable, although the most common case is fixed.

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq D + d_1y_1 + d_2y_2 + \ldots + d_my_m \quad or$$
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = D + d_1y_1 + d_2y_2 + \ldots + d_my_m$$

The supply is made up of a series of variables represented as $x_1, x_2, \ldots x_n$

Each activity makes a contribution determined by the value of the expression: $a_i x_i$ where a_i is the unit contribution of the variable x_i (the amount supplied by each unit of the value that takes the variable x_i).

The quantity demanded is defined as D. The variables y_j represent activities that may exist in the system to increase demand (d_j would be the unit increase in demand for each activity y_j). The supply and the demand must be expressed in the same unit of measure.

Regarding the sign of the constraint, we can find cases where only $\geq$ is admissible, others where only $=$ is admissible, and others where both signs are valid. The validity is governed by the following rules:

- The expression with sign $\geq$ is correct when the supply involves a cost in the system or there are unit contributions other than 1.
- The expression with $=$ is correct as long as the unit contributions are all equal to 1.

If the unit contributions are different from 1, imposing equality could make the problem inadmissible because we would not obtain values of x_i that give equality, or we would exclude solutions that could be better for the objective function. Let us analyze this validity with an illustration:

Illustration 6.12: Buying from Providers

There is a simplified system of buying a product from two providers. It is necessary to buy 1000 units of the product. The purchase cost is €3 for both providers. (We can ignore other system features).

Table of Elements (Table 6.9)

Decision Activities

Action: Buy product from providers

Decision variables: x_i = Product units bought from provider i

Demand Supply Specification

There is a fixed demand $D = 1000$ units of the product for which we have the variable x_i as input. The unit contribution is 1. For each unit of product purchased, a unit of demand is provided.

In this case, the two signs would be valid:

$x_1 + x_2 \geq 1000$ Buying has a cost.

Table 6.9 Table of elements of Illustration 6.12

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Providers	$i = 1,2$	I_U	Cost	p_i	C	S	{$3, $3}
Product	–	C_D		p_i			
			Demand	D	I	W	1000

Table 6.10 Table of elements of Illustration 6.12 modified

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Provider 1	–	I_U	Product Cost	*p*	C	S	€3
Provider 2	–	I_U	Lot Cost	Pl	C	S	€22
Lot	–	C_I		Pl			
			Product units	*N*	I	S	8
Product	–	C_D		*p; N*			
			Demand	*D*	I	W	1000

$x_1 + x_2 = 1000$ Unitary contributions are 1.

Now we can modify the statement and incorporate the following information:

"Provider 2 does not serve single units but serves lots of 8 units at a price of $22."

This modifies the table of elements and the decision activity as follows:

Table of Elements (Table 6.10)

Decision Activities
Action: Buy product from provider 1 and lots from provider 2
Events: Product ⇒ provider 1; Lot ⇒ provider 2
Decision variables:
x = Product units bought from provider 1
y = Lots bought from provider 2

Demand Supply Specification

The two decision variables, x and y, act as input, but in this case, the unit contribution of the variable y is 8 (8 units of product are contributed for each lot).

$x + 8y \geq 1000$: Correct expression. Buying has a cost and the contributions are not all the units.

$x + 8y = 1000$: Incorrect expression. Solutions are excluded.

Obviously, if the system explicitly specifies that we must provide exactly the amount of demand, then we are obliged to use the equality sign.

6.6 Bound Imposition Specifications

Bound imposition specifications are usually the simplest to identify and model in a system. They establish lower or upper bounds for measurable activities or for variable functions (calculations). There are two types:

Upper bound: Imposition of maximum value
Lower bound: Imposition of minimum value

Both the capacity specifications and the demand contribution with sign $\geq$ can be understood as a particular case of bound specification. However, we have taken them out of this category because of the meaning they express. Bound specifications do not have to represent a concept of capacity consumption or demand supply but are impositions without defined semantics and of an explicit nature (they must be explicitly defined in the statement).

Illustration 6.13
For a system that generates the following decision variables:

x_i = *Number of product units purchased from provider Pi;* $i = 1 \ldots n$

The following bound specifications are defined:

1. Do not buy more than 50 units from provider 1: $x_2 \leq 50$.
2. Buy at least 10 units from provider 2: $x_1 \geq 10$.
3. Buy more than 20 units from provider 3: $x_1 > 20 \Rightarrow x_1 \geq 21$.

Illustration 6.14
System: A set of operators (O1, On) that work in the production of a set of substances (S1, Sm). The system measures the amount of substance produced by each operator. The number of kilos produced by operator 1 must be at least 1 kilo more than the kilos of operator 2.

Table of Elements (Table 6.11)

Decision Activities
Action: Produce substances using operators
Decision variables: x_{ij} = Kilos of Substance j produced by operator i

Specifications
Imposition: The number of kilos produced by operator 1 must be at least 1 kilo more than the kilos of operator 2:

Number of kilos produced by operator 1: Not included in the decision variables since it is an auxiliary calculation (y_1):

$$y_1 = \sum_{j=1}^{m} x_{1j}$$

Number of kilos produced by operator 2:

$$y_2 = \sum_{j=1}^{m} x_{2j}$$

Table 6.11 Elements of Illustration 6.14

Name	Set	QN
Operators	$i = 1 \ldots n$	I_U
Substances	$j = 1 \ldots m$	I_M

Bound specification: a lower limit is imposed on the kilos produced by the operator 1.

$y_1 \geq y_2 + 1$

6.7 Allocation, Balance, or Equilibrium Specifications

Allocation, balance, or equilibrium specifications are all of the specifications that define:

- An exact assignment to or imposition of a value on an integer or continuous variable or on variables functions.
- Balance or equilibrium between variables or functions of variables. This includes auxiliary calculations.

Their controversy is that they can cause equivalences between auxiliary calculation and decision activity.

Illustration 6.15
For a system that generates the following decision variables:

x_i = *Number of product units purchased from provider Pi;* $i = 1 \ldots n$

A. *Buy 10 units from provider 1:*
$x_1 = 10$
B. *The sum of quantities purchased from provider 1 and 2 must be 50:*
$x_1 + x_2 = 50$
C. *Buy a total of 500 units:*
$\sum_{i=1}^{n} x_i = 500$
D. *Buy the same from provider 1 as from 3:*
$x_1 = x_3$

As we have said, this type of specification makes certain decision variables cease to exist because they are given a certain value (Case A) or because they become auxiliary calculations (cases B, C, D). However, if we differentiate the description of elements and activities from the specifications, we can consider them as decision variables. In any case, the system does not vary, but its structure can simply be represented in several ways.

This type of specification has greater relevance when a relationship of flow balance is expressed. This happens when a directed graph G (N, A) formed by nodes (N) and arcs (A) is established as a scenario, where a concept that corresponds to a measurable element of the system flows through the graph.

The adjective of directed graphs (Bang-Jensen and Gutin, 2000) is important since the systems that are represented as non-directed graphs (formed by nodes and edges) have a completely different meaning. An undirected graph always has a

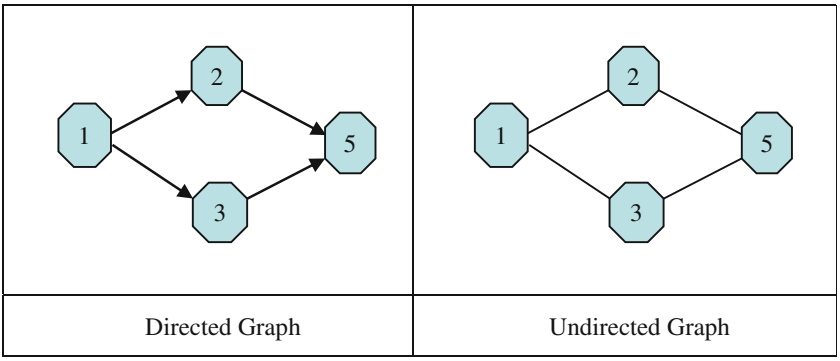


Fig. 6.1 Directed and undirected graphs

Table 6.12 Table of elements of a system with a directed graph

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Nodes	$i = 1...n$	I	Origin node	NO_{ik}	B	S	...
			Destination node	ND_{ik}	B	S	...
			Flow injection	I_i	C	S	...
			Flow demand	D_i	C	S	...
			Node Data	...	...	...	...
Arcs	$k = 1...m$	I		NO_{ik}; ND_{ik}			
			Arc Data	...	...	...	...
Flow	–	C/I_M		I_i; D_i			
Graph	–	I_U					

meaning of relationship or association between elements that are presented as nodes. These relationships are represented by its edges. Both nodes and edges can have associated data (Fig. 6.1).

A directed graph instead has a different meaning. The nodes are elements of the system, and the arcs are connections between nodes to enable the circulation of an added concept, the flow, in most cases. There are some scenarios where the direction has some meaning of relationship between the nodes, as dependency or offspring meanings. When there is flow, the flow is an element of the problem whose activity is to circulate through the arcs of the graph. This flow must be injected into the graph by the nodes and must also be extracted from the graph through the nodes.

The constraints of flow balance mean that the flow to arrive at a node must be equal to the flow that leaves it.

In Chap. 3 (Tables 3.46, 3.47, and 3.48), we saw three ways of representing the table of elements of a directed graph. We use Table 3.47, incorporating the flow concept (Table 6.12):

As a directed graph, the data of injection or flow input and demand or flow output that each node can possess have been incorporated. A node will inject flow, demand flow, or do neither of these. On the other hand, we can also find systems where injecting or demanding flow is not an attribute but a decision activity. In that case, those data would disappear.

In the system, the sum of the flow injected must always be equal to the sum of the flow demanded.

In *Node Data* and *Arc Data*, all those specific data of the system associated with nodes and arcs would be represented, respectively.

Regarding the decision activities, each system will have specific activities, but there is always the flow circulation activity in common:

Action: Circulate [flow through the arcs]
Participating elements: Flow C/I_M; Arcs $k = 1...m$;
Quantification: Continuous
Events: Flow $\Rightarrow$ k=1...m
Decision variables: x_k = Flow that circulates through the arc k.

In general, the constraint of equilibrium or flow balance is expressed as follows:

Format: At each node of the graph: Sum of the Input Flow = Sum of the Output Flow
General expression:

$$\forall i : \sum_{k/\mathrm{ND}_{ik}=1} x_k + I_i = \sum_{k/\mathrm{NO}_{ik}=1} x_k + D_i$$

The flow x_k of all the incoming arcs in i ($k/\mathrm{ND}_{ik} = 1$) plus the flow I_i that injects the node is recorded as input flow.

In output flow, the flow x_k of all the outgoing arcs of i ($k/\mathrm{NO}_{ik} = 1$) plus the D_i flow demanded by the node is recorded.

We are going to differentiate two types of scenarios in which these restrictions are presented, and therefore we require the use of directed graphs:

- **Explicit Case**: the system itself is a directed graph on which an optimization problem is raised (shortest path, maximum flow, minimum cost flow, etc.).
- **Implicit Case**: this is the most interesting case. The system is not described as a graph. However, part of its activity can be represented by a directed graph. The condition for this is that there must be a **measurable element**, individual or collective, subject to activities in which unitary elements also participate. This acquires a greater meaning if a set of **time periods** participates in the system as elements. The complexity in the implicit case lies in the creation of the graph, although we will give a series of guidelines for its construction in Sect. 6.7.2.

6.7.1 Explicit Case

Within the problems associated with directed graphs, we can see one of the most known and applied, which is the shortest path problem. Let us look at an illustration of it:

Illustration 6.16: Shortest Path *Problem (Dijkstra 1959)*
There is a graph G (N, A) where each arc has a cost, obtaining the shortest path from a source node to a destination node. We model the problem for the graph of the figure, using Node 1 as the source node and Node 9 as destination.

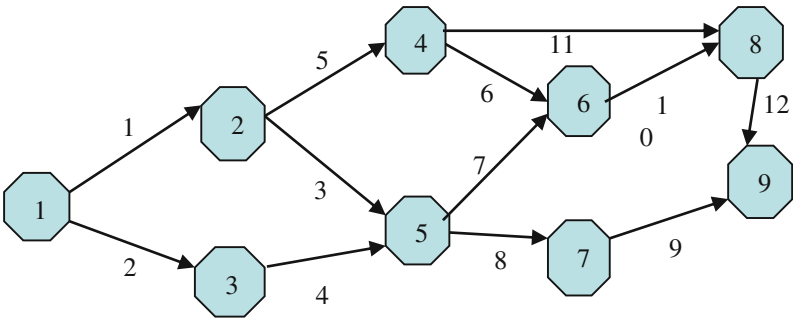


Fig. 1 Directed graph G(N,A)

Nodes and arcs have been labelled with a number.

Table of Elements

The problem uses a flow unit that will flow from node 1 to node 9. Therefore, node 1 injects a flow unit that requires node 9. Although we inject an integer amount of flow, it is not necessary to consider the flow as collective (discrete measurable), since in operative research it is demonstrated that by considering it as a continuous

Table 6.13 Table of elements of illustration 6.16

Elements	Set	QN	Data				
			Name	Param	Type	Belong	Value
Nodes	$i = 1...9$	I_U	Origin node	NO_{ik}	B	S	...
			Destination node	ND_{ik}	B	S	...
			Flow injection	I_i	C	S	{1,...,0,0}
			Flow demand	D_i	C	S	{0,...,0,1}
Arcs	$k = 1...12$	I_U		NO_{ik}; ND_{ik}			
			Cost	c_k	C	W	...
Flow	–	I_M		I_i; D_i			
Graph	–	I_U					

measurable, due to the property of unimodularity of its coefficients matrix, variables will always be integers in the optimal solution (Table 6.13).

Decision Activities

Action: Circulate flow through the arcs
Decision variables: x_k = Flow that circulates through the arc k.

Specifications

Flow balance:

$$\forall i : \sum_{k/\mathrm{ND}_{ik}=1} x_k + I_i = \sum_{k/\mathrm{NO}_{ik}=1} x_k + D_i$$

Objective Function

$$\text{Min} \sum_{k=1}^{12} c_k x_k$$

The shortest path problem does not need any additional specification. The specification of flow balance not only guarantees the conservation of the flow but also its continuity; therefore, the set of arcs of the solution guarantees a path.

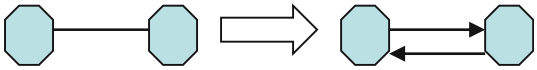
On the other hand, there are some problems associated with undirected graphs that have been modelled, transforming the graph into directed and introducing a flow concept in it. Examples include the Minimum spanning tree problem, MST (Graham and Hell 1985), or the Steiner problem (Hwang et al. 1992), as well as variants thereof. Let us consider the case of the MST problem.

Illustration 6.17: MST Problem
There is an undirected graph G (N, A), where each edge has a cost. We try to obtain the minimum cost spanning tree, that is, obtain a subgraph of G that connects all the nodes at a minimum cost.

To model this problem with the use of a directed graph, we proceed as follows:
Each edge is transformed into two arcs:

Table 6.14 Table of elements of Illustration 6.17

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Nodes	$i = 1...n$	$\mathrm{I_U}$	Origin node	NO_{ik}	B	C	...
			Destination node	ND_{ik}	B	C	...
			Flow injection	I_i	C	C	...
			Flow demand	D_i	C	C	...
Arcs	$k = 1...2m$	$\mathrm{I_U}$		NO_{ik}; ND_{ik}			
			Cost	c_k	C	P	...
Flow	–	$\mathrm{I_M}$		I_i; D_i			
Graph	–	$\mathrm{I_U}$					



$n-1$ flow units are included in the problem (n = number of nodes) that will send (inject) a node, labelled as root node, to the rest of nodes of the graph. Therefore, node i will have an injection $I_i = n-1$, with $D_i = 0$, and the $n-1$ remaining nodes $I_i = 0$ and $D_i = 1$.

In the objective function, we consider the cost of the arcs through which the flow has circulated, so that the cost is incurred if the flow has circulated, regardless of the amount of flow that has circulated. The cost of each arc is the cost of the associated edge.

Table of Elements (Table 6.14)

It is not necessary to use a binary attribute that identifies the root node, since it can be identified by the injection of $n-1$ flow units.

Decision Activities

Action: Circulate flow through the arcs.
Decision variables: x_k = Flow that circulates through the arc k. $k = 1...2m$.

Specifications

Flow balance:

$$\forall i: \sum_{k/\mathrm{ND}_{ik}=1} x_k + I_i = \sum_{k/\mathrm{NO}_{ik}=1} x_k + D_i$$

Objective Criterion

For the objective function, it is necessary to define a logical calculation on each arc to know whether or not the flow has circulated:

Binary logical calculation: Circulate flow through an arc.
Applied to: Arcs $k=1...2m$.
Definition of logical variable:
$\forall k$: $\alpha_k = 1$ if the flow circulates through arc k; 0 otherwise.
Logical proposition: $\forall k : \alpha_k = 1$ IF AND ONLY IF $x_k > 0 \Rightarrow$ Ref. $S_V \Rightarrow \forall k$: IF $x_k > 0$ THEN $\alpha_k = 1$.
The expression of costs would be as follows:

$$\text{Min} \sum_{k=1}^{2m} c_k \alpha_k$$

6.7.2 Implicit Case

The implicit case is an optional support tool for the identification of equilibrium specifications between variables of the system. It occurs in systems where a measurable element is subjected to activities in which other no measurable individual elements participate. These activities suppose injections or demands of the measurable element and even the transfer of quantities between individual elements. This becomes even more relevant if the time element participates in the system, that is, if there is a set of periods in which the activities occur. The activities contribute, demand, or simply move units of the element over time.

In a graph, only the movement of a measurable element can be represented. If there is more than one measurable element in a system, a graph must be made for each of them.

If there are no periods of time, there is only one implicit period in which the activities take place, as already mentioned in Chap. 3. However, what does need to happen to represent the problem as a graph is that other individual elements must participate in the measurement activities.

As we have said, it is not mandatory to design a directed graph to model these systems, but it is convenient. The graph will contain the activities of the system with respect to that measurable element and the relationships between them.

The construction procedure of the graph is the following:

Nodes
We will use a node for each individual element that intervenes in the flow of units of the measurable element in each period of time, except for the time element. By default, we can use all individual elements as nodes in each period of time and afterwards eliminate those that are not connected in the final graph. There are systems where an element only participates in a certain period, and therefore its use does not make sense in other periods.

The nodes can inject or demand flow. If they are known values, they correspond to data of those elements. The sum of the flow injected must always be equal to the demanded flow. It is also possible that the node injects or demands flow, but the amount is not determined. In this case, injecting or demanding flow corresponds to decision activities or calculations. By annotating this in the graph, we will represent the injection with a negative superscript $(-)$ and the demand with a positive superscript $(+)$.

Arcs
In the arc activities, simple calculations and data of the measurable element are represented.

We must analyze:

- The movements of units between elements.
- The elements that maintain units over time: since the nodes cannot store units, in order to respect the principle of flow balance, the units not subject to any activity in a node must also circulate over time to the node that represents the same

element in the next period. With this we achieve a circulation between nodes that is equivalent to the storage of units of the measurable element over time in that element.

Finally, once the graph has been designed, it will be necessary to explore which arcs and which auxiliary calculations define decision activities.

The design of the graph does not have to adapt to a single configuration. Depending on the interpretation of the activities over time, different designs can be generated.

From the constructed graph, a flow balance constraint is proposed on each node. This type of graph can always be refined and simplified, since nodes that have an input arc and an output arc can be discarded for the balance.

Table 6.15 Elements of Illustration 6.18

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Warehouses	$i = 1...m$	I_U	Stock	K_i	I	S	...
			Cost	C_{ij}	C	S	...
Customers	$j = 1...n$	I_U	Demand	D_j	I	S	...
			Cost	C_{ij}			
Product	–	C_D		K_i; D_j; C_{ij}			

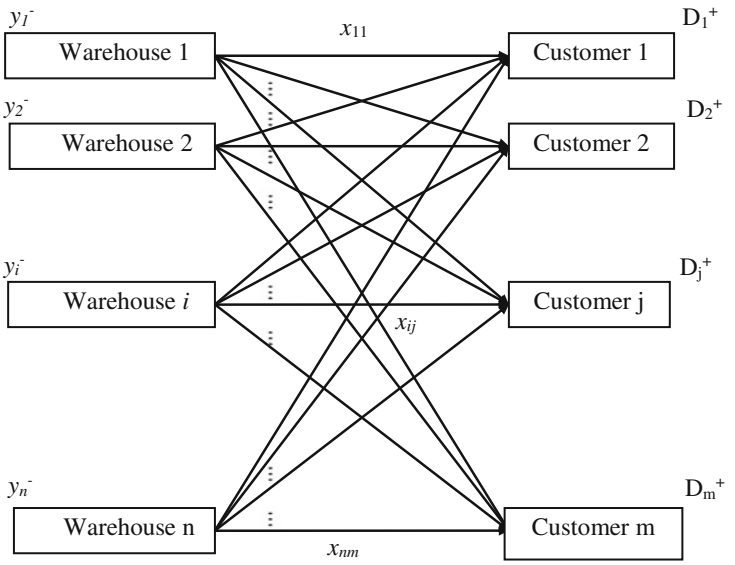


Fig. 6.2 Implicit directed graph of Illustration 6.18

This type of graph has always been used in some optimization problems, to turn them into problems associated with directed graphs. A clear example is the transport problem, which has been modelled as a minimum cost flow problem:

Illustration 6.18: Transportation Problem (Öztürk et al. 2015)
A company has m warehouses where its products are located. Each warehouse i (i = 1...m) has a stock of Ki units. There is a set of n customers (j = 1...n) with a demand Dj of product units. The company has to supply the product demand of the customers from the warehouses. The cost of sending a product from each warehouse i (i = 1...m) to each customer j is estimated in c_{ij}.

Table of Elements (Table 6.15)

We are facing a system that does not have periods of time. The measurable element is the product. We designed a graph (Fig. 6.2) with the *m* warehouses and the *n* customers:

The graph will collect the admissible movements of the product units (flow), which is produced from each warehouse to each customer. Nodes associated with warehouses inject an undetermined amount of flow (y_i for warehouse i). The nodes associated with customers demand a certain amount of flow, their product demand D_j. The arcs between each warehouse and each customer include the problem decision activities (x_{ij} = product units that are sent from warehouse i to customer j).

The equations of flow balance generated by the graph are:

$\forall i: \quad y_i = \sum_{j=1}^{m} x_{ij}$ The flow injection could be defined as an auxiliary calculation.

$\forall j: \quad \sum_{i=1}^{n} x_{ij} = D_j$ This corresponds with a specification of demand supply (unit contributions allow the use of the equality sign).

To finish formulating the transport problem, we would have to incorporate the capacity consumption specification at each warehouse:

$$\forall i: \quad \sum_{j=1}^{m} x_{ij} \leq K_i$$

Objective Function

Minimize associated costs. Each decision variable has a unit cost:

$$\text{Min} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

Let us now take a look at an illustration of the implicit case in a system that considers more than a period of time. Production planning problems are included in this type of case:

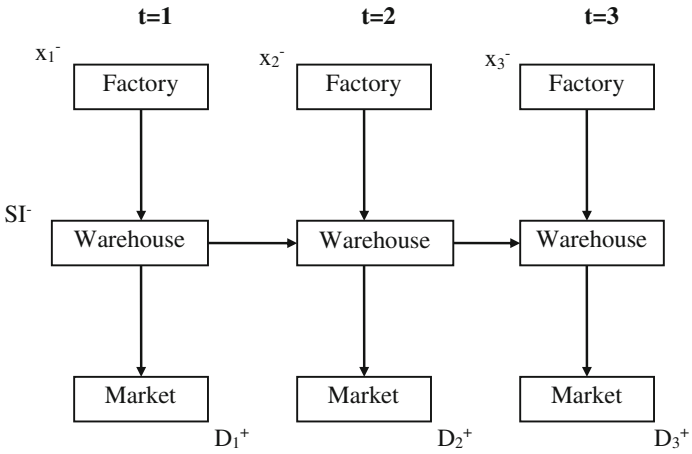


Fig. 6.3 Implicit directed graph of Illustration 6.19

Table 6.16 Elements of Illustration 6.19

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Factory	–	I_U					
Warehouse	–	I_U	Initial stock	SI	I	S	5
Market	–	I_U	Demand	D_t	I	S	{30,12,26}
Product	–	C_I		SI; D_t			
Months	$t = 1 \ldots 3$	I_U		D_t			

Illustration 6.19: Production Planning of a Product (Larrañeta et al. 1995)
System for the production planning of a factory that produces a product for which there is a market demand for the next three months of 30, 12, and 26 units. A warehouse is available to store the manufactured units. Initially, there is a stock in the warehouse of five units.

The system must determine the quantities produced in each period as well as the quantities stored (Fig. 6.3).

Table of Elements (Table 6.16)

The three individual elements participate in the three periods.

Since the warehouse can keep units from one period to the next, we join those nodes with arcs. Labelling flow movements (Fig. 6.4):

The nodes with an entry or injection and an exit or demand can simplify the labelling, as it is evident that for each node factory and each node market, it is fulfilled by flow balance (Fig. 6.5):

$$F_i = x_i \quad i = 1, 2, 3$$
$$E_i = D_i \quad i = 1, 2, 3$$

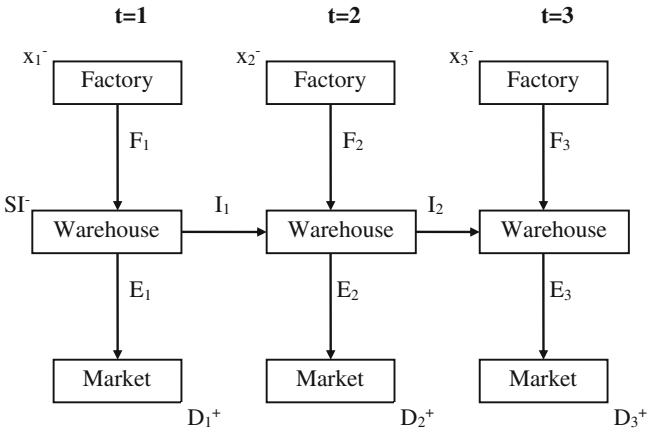


Fig. 6.4 Labelled graph of Illustration 6.19

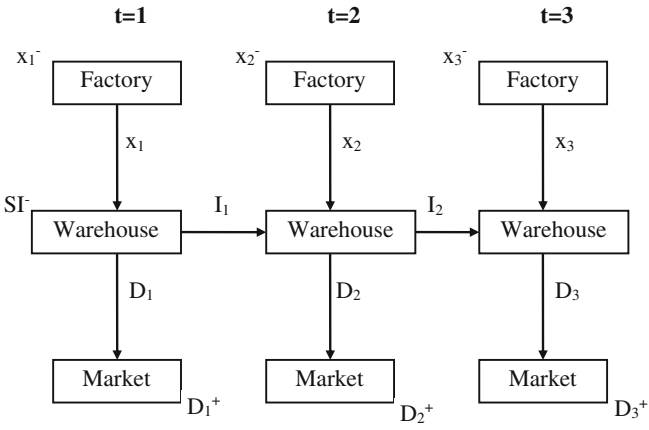


Fig. 6.5 Simplified labelled graph of Illustration 6.19

Therefore, we generate only the equations in the warehouse nodes:

Warehouse t=1: $x_1 + SI = D_1 + I_1$
Warehouse t=2: $x_2 + I_1 = D_2 + I_2$
Warehouse t=3: $x_3 + I_2 = D_3$
Generically: $\forall t: \quad I_{t-1} + x_t = D_t + I_t \quad (I_0 = SI)$

The flow balance equations define in themselves both the demand supply specification of the market element in each period and the auxiliary calculation of the

Table 6.17 Elements of Illustration 6.20

Elements	Set	QN	DATA				
			Name	Param	Type	Belonging	Value
Banquets	$i = 1...3$	I_U	Tablecloths	m_i	I	S	...
			Day	d_{it}	B	S	...
Storeroom	–	I_U	Stock	S	I	S	200
Market	–	I_U	Price	p	C	S	12
Basket	–	I_U					
Laundry	–	I_U					
Fast wash	–	I_U	Cost	cF	C	S	6
Slow wash	–	I_U	Cost	cL	C	S	4
Days	$t = 1...3$	I_U		d_{it}			
Tablecloths	–	C_I		m_i; S; p; cF; cL			

quantity stored in each period (the action of storing is not really a decision activity but an auxiliary calculation):

Picking up the excess of units of what stays in the warehouse:

$$I_1 = (\mathrm{SI} + x_1) - D_1$$
$$t = 2: \quad I_1 + x_2 \geq D_2$$
$$I_2 = (I_1 + x_2) - D_2$$
$$t = 3: \quad I_2 + x_3 \geq D_3$$
$$I_3 = (I_2 + x_3) - D_3$$
$$\forall t: \quad I_{t-1} + x_t \geq D_t \Rightarrow \forall t: \quad I_{t-1} + x_t - I_t = D_t$$

Finally, to reinforce the graph design, let us consider an example with more content:

Illustration 6.20: Food Service (Illustration 3.9.2)
A food service business has contracted three banquets for the next 3 days, requiring 150 clean tablecloths for the first banquet, 100 for the second, 140 for the third, and 130 for the fourth. Currently, it has 200 tablecloths in the storeroom, all of them clean, and they can buy what you need on the market every day at a cost of 12 m.u/tablecloth.

After the banquets, the tablecloths can go to the laundry basket or be sent to the laundry to be washed. The laundry offers the following washing services:

- *Fast: Clean tablecloths for the next day, at a cost of 6 m.u/tablecloth.*
- *Slow: Clean tablecloths in 2 days, at a cost of 4 m.u/tablecloth.*

Table of Elements (Table 6.17)

Next, we present a graph design. In the graph, we have excluded the possibility of washing slowly from the second and three periods and washing quickly from the fourth one.

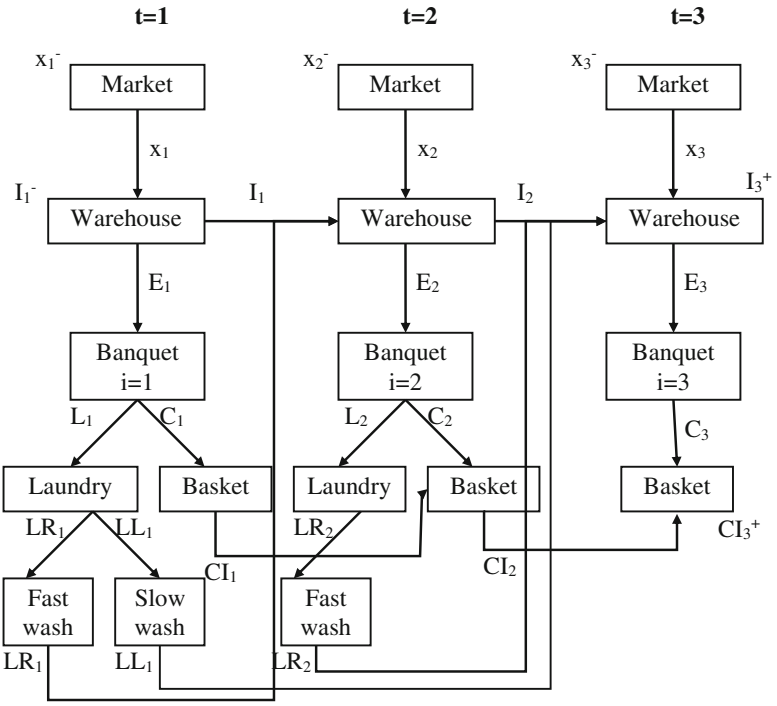


Fig. 6.6 Implicit directed graph of Illustration 6.20

On the other hand, since the injected units must be extracted from the graph, the warehouse and the laundry basket are taken as nodes that demand flow units in the last period of undetermined value.

For the flow balance we can exclude the market because it has a flow injection and a single exit arc. Similarly, the nodes that represent the fast wash and slow wash element have a single input and output and it is not necessary to propose the balance equation.

The labelling of arcs has been as follows:

x_t: Quantity of tablecloths purchased on day t
I_t: Quantity of tablecloths in store (clean tablecloths)
E_t: Quantity of tablecloths brought to the banquet $i = t$ ($E_t = D_t$)
L_t: Quantity of tablecloths sent to be washed on day t
C_t: Quantity of tablecloths taken to laundry basket on day t
CI_t: Quantity of tablecloths in basket
LR_t: Quantity of tablecloths washed quickly on day t
LL_t: Quantity of tablecloths washed slowly on day t (Fig. 6.6)

It would have been possible to use two elements to represent the tablecloth element: clean tablecloth and dirty tablecloth. This configuration would also have been correct in the system, but it was not necessary to make the distinction since in

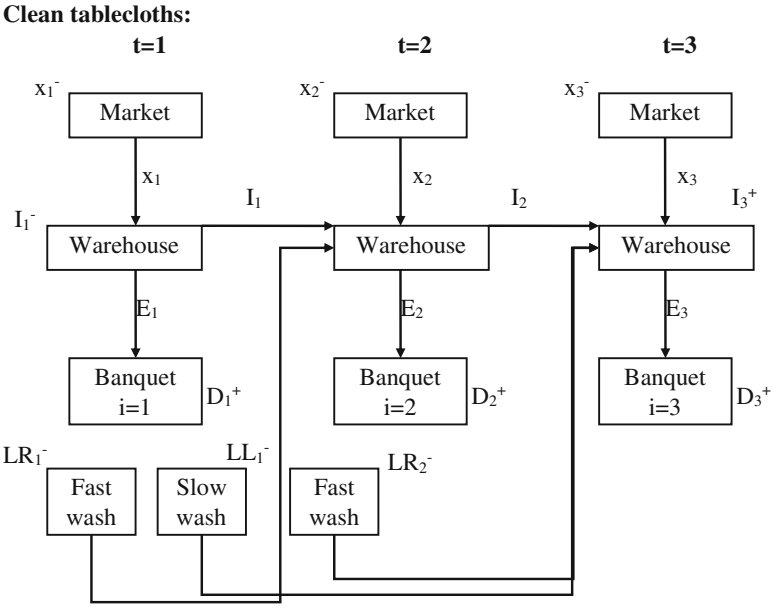


Fig. 6.7 Implicit directed graph for clean tablecloths

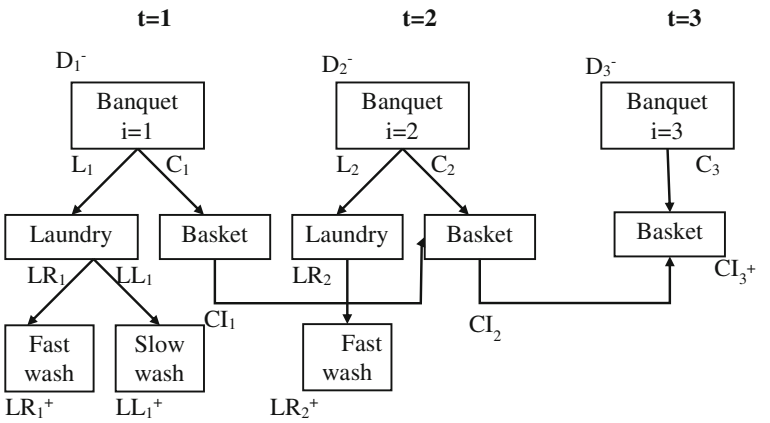


Fig. 6.8 Implicit directed graph for dirty tablecloths

the decision activities the concepts are not intermingled (only clean tablecloths are bought, only dirty tablecloths are washed, etc.). The system can work with a single concept.

However, if we had made the distinction in the table of elements, we should have designed two graphs, one for each measurable element. The flow of each graph should be related later. The design would be as follows:

Clean tablecloths (Fig. 6.7).

Dirty tablecloths (Fig. 6.8).

The tablecloth demand D_t becomes a demand for flow in the graph of clean tablecloths and injection of flow in the graph of dirty tablecloths. With the tablecloths to be washed (LR and LL), the opposite happens.

6.8 Modelling of Propositional Logic Specifications

At the beginning of the chapter, we assigned specifications to the nature of propositions, which may be simple or compound. The simple propositions are all the typologies that we have just studied. Compound propositions are those propositions that use logical operators or connectives (If . . . then; If and only if; Not; Or; And; Either . . . or) to relate simple propositions. In this section, we focus on the modelling of compound propositions. Compound propositions are a key aspect in the formulation of optimization problems of a certain depth.

Since compound propositions are the basis of propositional logic, we shall consider the propositional logic specification as that which is formulated as a compound proposition. When using operators, we are always representing a compound proposition.

We already saw in Chap. 5 that logical calculations were defined by compound propositions, and therefore these propositions can be considered as a propositional logic specification.

Let us look at some examples of specifications and logical calculations that give rise to compound propositions:

Illustration 6.21
For a product purchasing system with five suppliers, the following decision variables are generated:

xi = units of the product purchased from the supplier i. i = 1 . . . 5.

Specifications that we could define:

- *Logical proposition 1. – We cannot buy units from supplier 1 and supplier 2*:
  "NOT (x_1>0 AND x_2>0)"
- *Logical proposition 2. – If you buy more than 10 units from supplier 1 you cannot buy more than 5 units from supplier 3*:
  "IF x_1>10 THEN $x_3 \leq 5$"
- Logical proposition 3. – You must buy 25 units from only one of the suppliers:
  "EITHER x_1=25 OR x_2=25 OR x_3=25 OR x_4=25 OR x_5=25"

- Logical proposition 4. – If you buy more than 10 units from supplier 4 or supplier 5, you must buy 15 units from supplier 1:
  "IF x_4>10 OR x_5>10 THEN x_1=15"
- *Logical proposition 5. – If the system needs a logical calculation to know from which suppliers we have purchased units*:

  Binary logical calculation: Supplier provides units
  Applied to: Each supplier i=*1...5*
  Variables: $\alpha_i = 1$ if we buy units from supplier i; 0 otherwise. i=1...5
  Logical proposition: $\forall i: \quad \alpha_i = 1 \quad$ IF AND ONLY IF $x_i > 0$

The difficulty of modelling logical propositions lies not so much in obtaining the constraints that define it, which as we will see in the following sections is based on the application of some rules but on correctly stating the proposition.

We have already defined the concepts related to the propositional logic in Chap. 5, when we present the logical calculations. Now we present some of these concepts with the aim of structuring their modelling. For modelling, we propose a general scheme where some rules are based on the modelling of propositions already described by authors. We emphasize as a reference the modelling of propositions described by Williams (2013).

6.8.1 Simple Propositions and Logical Operators

Atomic or simple propositions are those that are defined without the use of any logical operator. The format in a lineal formulation would be:

Left part	Sign	Right part
X (Lineal function)	$<;\leq;=;\geq;>;\neq$	Independent term (Numeric value)

In mathematical programming, we will distinguish three types of simple propositions:

– Binary simple proposition: The lineal function from the left part only takes binary values (1; 0).
– Integer simple proposition: The lineal function from the left part only takes integer values.
– Continuous simple proposition: The lineal function from the left part takes continuous values.

This distinction is necessary for the modelling of compound propositions.

In mathematical programming, any valid or admissible solution must satisfy a truth result (T). Therefore, any restriction such as those defined in Sections 6.3, 6.4, 6.5, 6.6 and 6.7 would correspond to a simple proposition, with an admissible solution of the problem being one that satisfies a truth result when applied to the

Table 6.18 Truth tables of logical operators

Operator	Symbol	Semantic	Truth table
Negation	$\neg$	NOT (ϕ)	ϕ | $\neg\phi$ F | T T | F
Disjunction	$\vee$	ϕ OR ψ	ϕ ψ $\phi \vee \psi$ F F F F T T T F T T T T
Conjunction	$\wedge$; &	ϕ AND ψ	ϕ ψ $\phi \wedge \psi$ F F F F T F T F F T T T
Conditional	$\rightarrow$	IF ϕ THEN ψ	ϕ ψ $\phi \rightarrow \psi$ F F T F T T T F F T T T
Biconditional	$\leftrightarrow$	ϕ IF AND ONLY IF ψ	ϕ ψ $\phi \leftrightarrow \psi$ F F T F T F T F F T T T
Exclusive disjunction	$\oplus$	EITHER ϕ OR ψ	ϕ ψ $\phi \oplus \psi$ F F F F T T T F T T T F

Table 6.19 Equivalences between operators

Proposition	Equivalent proposition	Reference
$\phi \leftrightarrow \psi$	$\phi \rightarrow \psi$ $\psi \rightarrow \phi$	f_1
$\phi \bigoplus \psi$	$(\phi \wedge \neg (\psi)) \vee (\neg (\phi) \wedge \psi)$	f_2
$\phi \rightarrow \psi$	$\neg \psi \rightarrow \neg \phi$	f_3

simple proposition. When faced with composite propositions, where we use logical operators, the allowable solutions must satisfy the truth results of the operator's truth table. Let us see the truth tables of each operator:

ϕ *y* ψ are shown as logical propositions (Table 6.18).

Already in Chap. 5 devoted to logical calculations, we presented equivalences between some operators. Table 6.19 collects those equivalences in addition to another with the operator Exclusive disjunction ($\bigoplus$). The operator Biconditional ($\leftrightarrow$) as well as the operator Exclusive disjunction could be ignored thanks to these equivalences. However, for convenience, we will take a look at the modelling of those operators as well. From Table 6.19, we will label with references all the transformations or formulations that can be used in the modelling of propositions, in order to be able to reference the origin of the transformation in the text.

6.8.2 *Reduction of Signs*

For integer or continuous simple propositions, the group of signs is convenient to reduce it to the set ($\leq$; $=$; $\geq$), except for those to which the negation operator applies. In that case, for simplicity, reduction is not necessary.

Calling X the linear function of the proposition and V the independent term or numerical value of the right part of the simple proposition, the transformation of integer/continuous propositions is the following:

ξ is a small enough value to avoid in continuous propositions that the value less than V that the linear function could take is greater than $(V - \xi)$, in the case of $X < V$. The same applies for $X > V$.

We do not consider the case of binary atomic propositions in the reduction of signs, since the forms in which they can be presented are reduced to:

α as a binary linear function: $\alpha = 1$; $\alpha = 0$;

For convenience in modelling, in the case $\alpha = 0$, we can change the value to 1 using the following equivalence:

$\alpha = 0 \Rightarrow 1 - \alpha = 1$; $(1 - \alpha)$ is still a binary expression. [Reference f_7]

6.8.3 *Modelling Operators Individually*

First, we are going to analyze the modelling of connectives or logical operators individually, that is, we only consider compound propositions that do not have more than one different operator.

Table 6.20 Reduction of signs in simple propositions

Simple proposition	Reduction		
Sign	$X \in Z$	$X \in \Re$	Reference
$X < V$	$X \leq \lceil V \rceil - 1$	$X \leq V - \xi$	f_4
$X > V$	$X \geq \lfloor V \rfloor + 1$	$X \geq V + \xi$	f_5
$X \neq V$	$X < V$ OR $x > V$ $\Rightarrow$ Ref. f_4 y f_5 $\Rightarrow$ $(X \leq \lceil V \rceil - 1)$ OR $(X \geq \lfloor V \rfloor + 1)$	$X < V$ OR $X > V$ $\Rightarrow$ Ref. f_4 y f_5 $\Rightarrow$ $(X \leq V - \xi)$ OR $(X \geq V + \xi)$	f_6

Table 6.21 Increment and decrement parameters

	$V \in Z$		$V \notin Z$	
	$X \in Z$	$X \in \Re$	$X \in Z$	$X \in \Re$
$\mathbf{R^I}$	1	ξ	$\lceil V \rceil - V$	ξ
$\mathbf{R^D}$	1	ξ	$V - \lfloor V \rfloor$	ξ

To express the constraints resulting from modelling operators, we will distinguish between the type of value of the linear function (binary, integer, or continuous) and the sign ($\leq$; $=$; $\geq$) for the case of integer or continuous simple propositions.

We will denote with the variables X or Y the function of the left part of an integer or continuous simple proposition. The binary propositions will be expressed with a Greek letter (α, β, ω, δ, etc.).

In the modelling of operators, the integer or continuous simple propositions will only be differentiated in the increment or decrement parameters of the independent term V, as in the case of the reduction of signs (Table 6.20). Therefore, to simplify the notation, we are going to call the increment parameter R^I and the decrement parameter R^D. They will be defined as (Table 6.21):

ξ it will be a small enough value.

In the development of the modelling of some compound propositions, the definition of binary logical calculations is necessary, as we expressed in Section 5.2.3 of the previous chapter. These logical calculations serve to collect the result of simple propositions that are within the compound proposition. They will be collected in binary variables denoted by ω or by δ^1, δ^2, δ^3, and δ^4, when necessary. With these calculations we are going to ignore this semantic and mathematical definition. The proposition that defines them mathematically is integrated within the formulation of the operator.

On the other hand, for any integer or continuous atomic proposition, it will be necessary to obtain an upper bound and a lower bound of the linear function. The upper bound is a value that is never surmountable by the function. Equivalently, the lower bound is a value that can never be exceeded inferiorly by the linear function. Any value of dimension will be valid in the modelling, although adjusting the upper bound to the maximum of the linear function and the lower one to the minimum reduces the space of solutions and usually offers better behavior in the resolution. If

Table 6.22 Negation operator modelling

Sign	Model	Reference
NOT (X < V)	$X \geq V$	f_8
NOT (X ≤ V)	$X > V \Rightarrow f_5 \Rightarrow X \geq V + R^I$	f_9
NOT (X > V)	$X \leq V$	f_{10}
NOT (X ≥ V)	$X < V \Rightarrow f_4 \Rightarrow X \leq V - R^D$	f_{11}
NOT (X = V)	EITHER $(X < V)$ OR $(X > V) \Rightarrow^*$ $(X < V)$ OR $(X > V) \Rightarrow (X \leq V - R^D)$ OR $(X \geq V + R^I)$	f_{12}

*In that case, the exclusive disjunction coincides with the inclusive disjunction since the two propositions can never be fulfilled at the same time. The modelling would not have ended in case f_{12} since the connective OR would have to be modelled (Sect. 6.8.3.4)

Table 6.23 Nomenclature

Input Proposition	ϕ
Output Proposition	ψ
Binary functions	α; β
Integer/continuous functions	X; Y
Independent terms of integer/continuous functions	V; V_1; V_2

Table 6.24 Conditional operator modelling with binary input proposition

IF ϕ THEN ψ					
ϕ		ψ			
Type	Sign	Type	Sign	Model	Ref.
Binary	$\alpha = 1$	Binary	$\beta = 1$	$\alpha \leq \beta$	f_{13}
Binary	$\alpha = 1$	Integer/continuous	$X \leq V$	$X \leq V + (UB_X - V)(1-\alpha)$	f_{14}
Binary	$\alpha = 1$	Integer/continuous	$X \geq V$	$X \geq V\alpha + LB_X(1-\alpha)$	f_{15}
Binary	$\alpha = 1$	Integer/continuous	$X = V$	$X \leq V + (UB_X - V)(1-\alpha)$ $X \geq V\alpha + LB_X(1-\alpha)$	f_{16}

X is our integer or continuous function, we will denote its dimensions with the following parameters:

Upper bound of X: UB_X
Lower bound of X: LB_X

Table 6.25 Conditional operator modelling with integer/continuous input proposition

	IF ϕ THEN ψ						
	ϕ			ψ			
Ref	Type	Sign	Model	Type	Sign	Model	Ref.
f_{17}	Int/ Cont	$X \leq V_1$	$X \leq V_1 + (UB_X - V_1)(1-\omega)$ $X \geq (V_1 + R^I)(1-\omega) + LB_X\,\omega$	Int/ Cont	$Y \leq V_2$	$Y \leq V_2 + (UB_Y - V_2)(1-\omega)$	f_{20}
f_{18}	Int/ Cont	$X \geq V_1$	$X \geq V_1\omega + LB_X(1-\omega)$ $X \leq (V_1 - R^D)(1-\omega) + UB_X\,\omega$	Int/ Cont	$Y \geq V_2$	$Y \geq V_2\omega + LB_Y(1-\omega)$	f_{21}
f_{19}	Int/ Cont	$X = V_1$	$X \leq V_1 + (UB_X - V)(1-\omega)$ $X \geq V_1\omega + LB_X(1-\omega)$ $X \leq (V_1 - R^D)\delta_1 + UB_X\omega + UB_X\delta_2$ $X \geq (V_1 + R^I)\delta_2 + LB_X\omega + LB_X\delta_1$ $\delta_1 + \delta_2 = 1 - \omega$	Int/ Cont	$Y = V_2$	$Y \leq V_2 + (UB_Y - V_2)(1-\omega)$ $Y \geq V_2\omega + LB_Y(1-\omega)$	f_{22}

6.8.3.1 Negation Operator (NOT; ¬)

Negation operator modelling does not require any complex modelling exercise; it is just based on representing the opposite proposition. We show the case of whole or continuous propositions; for binary propositions, the application of the connective negation is something evident (Table 6.22).

6.8.3.2 Conditional Operator (IF ... THEN ... ; →)

The modelling of the conditional operator will be separated into two tables. In Table 6.24, we will present the modelling of the connective when the proposition of input of the condition is binary. In Table 6.25, we will deal with the modelling options when the input and output propositions are integer or continuous.

The nomenclature used in both Tables 6.24 and 6.25 is shown in Table 6.23.

Whenever possible, we should avoid the signs of equality in simple propositions. If the independent term corresponds to a lower bound of X, we can replace it with the sign $\leq$ ([Reference f_{LB}]). Similarly, if it corresponds to an upper bound, we can work with the sign $\geq$ ([Reference f_{UB}]).

Any combination of types of propositions not contemplated in the two previous tables can easily be deduced with the use of the equivalences Ref. f_3 and Ref. f_7.

The tables could have been further reduced, since we can change the sign $\geq$ to the sign $\leq$ simply by multiplying the proposition by -1. Even equality corresponds to two propositions of sign $\geq$ and $\leq$ with the connective AND, but I prefer this

representation to facilitate the obtaining of the mathematical formulation without having to change the propositions too much.

Let us take a look at some illustrations:

Illustration 6.22

We have x_1, x_2, x_3 integer variables ≥ 0.
We also have α_1 and α_2 binary variables.

IF α_1=1 THEN x_1+$x_2 \leq 10 \Rightarrow$ Ref. f_{14} [$\alpha = \alpha_1$; $X = x_1$+x_2; V=*10*] $\Rightarrow$
$\Rightarrow x_1 + x_2 \leq 10 + (UB_{x_1+x_2} - 10)(1 - \alpha_1)$
IF $\alpha_1 = 0$ THEN $\alpha_2 = 0 \Rightarrow$ Ref. $f_7 \Rightarrow$ IF $1-\alpha_1 = 1$ THEN $1-\alpha_2 = 1$
$\Rightarrow$ Ref. $f_{13} \Rightarrow 1 - \alpha_1 \leq 1 - \alpha_2$
IF $\alpha_2 = 0$ THEN $x_1 > 10 \Rightarrow$ Ref. $f_5 \Rightarrow$ IF $\alpha_2 = 0$ THEN $x_1 \geq 10 \Rightarrow$ Ref. $f_7 \Rightarrow$
$\Rightarrow$ IF $1-\alpha_2 = 1$ THEN $x_1 \geq 11 \Rightarrow$ Ref. $f_{15} \Rightarrow x_1 \geq 11\alpha_2 + LB_{x_1}(1 - \alpha_2)$

$\Rightarrow [LB_{x_1} = 0]$
$\Rightarrow x_1 \geq 11\alpha_2$
IF $x_1 > 5$ THEN $x_2 \leq 3 \Rightarrow$ Ref. $f_5 \Rightarrow$ IF $x_1 \geq 6$ THEN $x_2 \leq 3 \Rightarrow$ Ref. f_{18} ; $f_{20} \Rightarrow$

$x_1 \geq 6\omega + LB_{x_1}(1 - \omega)$ $\qquad$ $x_1 \geq 6\omega$

$\Rightarrow x_1 \leq 5(1 - \omega) + UB_{x_1}\omega \qquad \Rightarrow [LB_{x_1} = 0] \Rightarrow x_1 \leq 5(1 - \omega) + UB_{x_1}\omega$

$x_2 \leq 3 + (UB_{x_2} - 3)(1 - \omega)$ $\qquad$ $x_2 \leq 3 + (UB_{x_2} - 3)(1 - \omega)$

IF $x_1 + x_3 \geq 10$ THEN $\alpha_1 = 1 \Rightarrow$ Ref. f_3
$\Rightarrow$ IF NOT ($\alpha_1 = 1$) THEN NOT ($x_1 + x_3 \geq 10$) $\Rightarrow$
$\Rightarrow$ IF $\alpha_1 = 0$ THEN NOT (x_1+ $x_3 \geq 10$) $\Rightarrow$ Ref. $f_{11} \Rightarrow$ IF $\alpha_1 = 0$ THEN x_1+ $x_3 \leq 9 \Rightarrow$
Ref. $f_7 \Rightarrow$ IF $1-\alpha_1 = 1$ THEN x_1+$x_3 \leq 9 \Rightarrow$ Ref. $f_{14} \Rightarrow$
$\Rightarrow x_1 + x_3 \leq 9 + (UB_{x_1+x_3} - 9)(1 - (1 - \alpha_1)) \Rightarrow x_1 + x_3 \leq 9 + (UB_{x_1+x_3} - 9)\alpha_1$

Table 6.26 Biconditional connective modelling with binary propositions

ϕ IF AND ONLY IF ψ					
ϕ		ψ			
Type	Sign	Type	Sign	Model	Ref.
Binary	$\alpha = 1$	Binary	$\beta = 1$	$\alpha = \beta$	f_{23}
Binary	$\alpha = 1$	Integer/continuous	$X \leq V$	$X \leq V + (UB_X - V)(1-\alpha)$ $X \geq (V+R^I)(1-\alpha) + LB_X\,\alpha$	f_{24}
Binary	$\alpha = 1$	Integer/continuous	$X \geq V$	$X \geq V\alpha + LB_X\,(1-\alpha)$ $X \leq (V - R^D)(1-\alpha) + UB_X\,\alpha$	f_{25}
Binary	$\alpha = 1$	Integer/continuous	$X = V$	$X \leq V + (UB_X - V)(1-\alpha)$ $X \geq V\alpha + LB_X\,(1-\alpha)$ $X \leq (V - R^D)\delta_1 + UB_X\,\alpha + UB_X\delta_2$ $X \geq (V+R^I)\delta_2 + LB_X\,\alpha + LB_X\delta_1$ $\delta_1+\delta_2=1-\alpha$	f_{26}

Table 6.27 Biconditional connective modelling with integer/continuous propositions

	ϕ IF AND ONLY IF ψ						
	ϕ			ψ			
Ref.	Type	Sign	Model	Type	Sign	Model	Ref.
f_{27}	Int/ Cont	$X \leq V_1$	$X \leq V_1 + (UB_X - V_1)(1-\omega)$ $X \geq (V_1 + R^I)(1-\omega) + LB_X \omega$	Int/ Cont	$Y \leq V_2$	$Y \leq V_2 + (UB_Y - V_2)(1-\omega)$ $Y \geq (V_2 + R^I)(1-\omega) + LB_Y \omega$	f_{30}
f_{28}	Int/ Cont	$X \geq V_1$	$X \geq V_1\omega + LB_X(1-\omega)$ $X \leq (V_1 - R^D)(1-\omega) + UB_X \omega$	Int/ Cont	$Y \geq V_2$	$Y \geq V_2\omega + LB_Y(1-\omega)$ $Y \leq (V_2 - R^D)(1-\omega) + UB_Y\omega$	f_{31}
f_{29}	Int/ Cont	$X = V_1$	$X \leq V_1 + (UB_X - V)(1-\omega)$ $X \geq V_1\omega + LB_X(1-\omega)$ $X \leq (V_1 - R^D)\delta_1 + UB_X\omega + UB_X\delta_2$ $X \geq (V_1 + R^I)\delta_2 + LB_X\omega + LB_X\delta_1$ $\delta_1 + \delta_2 = 1 - \omega$	Int/ Cont	$Y = V_2$	$Y \leq V_2 + (UB_Y - V_2)(1-\omega)$ $Y \geq V_2\omega + LB_Y(1-\omega)$ $Y \leq (V_2 - R^D)\delta_3 + UB_Y\omega + UB_Y\delta_4$ $Y \geq (V_2 + R^I)\delta_4 + LB_Y\omega + LB_Y\delta_3$ $\delta_3 + \delta_4 = 1 - \omega$	f_{32}

IF $x1 - x2 = 5$ THEN $x3 \geq 1 \Rightarrow$ Ref. f19; f21 $\Rightarrow$

$$\Rightarrow \begin{cases} x_1 - x_2 \leq 5 + (UB_{x_1 - x_2} - 5)(1-\omega) \\ x_1 - x_2 \geq 5\omega + LB_{x_1 - x_2}(1-\omega) \\ x_1 - x_2 \leq 4\delta_1 + UB_{x_1 - x_2}\omega + UB_{x_1 - x_2}\delta_2 \\ x_1 - x_2 \geq 6\delta_2 + LB_{x_1 - x_2}\omega + LB_{x_1 - x_2}\delta_1 \\ \delta_1 + \delta_2 = 1 - \omega \\ x_3 \geq 1\omega + LB_{x_3}(1-\omega) \end{cases}$$

6.8.3.3 Biconditional Operator (IF AND ONLY IF; $\leftrightarrow$)

Following the same format as for the conditional, the modelling tables are the following (Tables 6.26 and 6.27):

6.8.3.4 Disjunction Operator (OR; $\vee$)

If there are two or more atomic propositions joined with the operator OR:
ϕ_i, i=1,2,. . ., n: ϕ_1 OR ϕ_2 OR . . . OR ϕ_n
Modelling follows two steps:

1. We define a logical calculation (ω_i) for each atomic proposition ϕ_i that is integer or continuous (not binary), to know when the proposition is fulfilled. However, it

is not necessary to control the two output values of the calculation, which would have been formulated as $\phi_i \leftrightarrow \omega_i = 1$. To simplify the modelling, we just need to pick up the value of ω_i when the proposition is not fulfilled:

$\forall i/\phi_i \in Z \vee \phi_i \in \Re$: IF NOT (ϕ_i)THEN $\omega_i = 0$

By equivalence f_3, we can also define it as:

$\forall i/\phi_i \in Z \vee \phi_i \in \Re$: IF $\omega_i = 1$ THEN ϕ_i [Reference f_{33}]

2. A quantitative selection specification is incorporated for the defined ω_i and the binary propositions ($i/\phi_i \in \{0,1\}$: $\alpha_i = 1$*):

$$\sum_{i/\phi_i \in \Re \vee \phi_i \in Z} \omega_i + \sum_{i/\phi_i \in \{0,1\}} \alpha_i \geq 1 \text{ [Reference } f_{34}]$$

where it is required that at least one proposition be fulfilled.

*: If the binary proposition were defined with value 0, by equivalence f_7, we transform it into value 1.

Illustration 6.23

We have x_1, x_2 continuous variables ≥ 0.

We also have α_1 and α_2 binary variables.

Proposition: $x_1 \leq 10 \ \vee \ x_2 \geq 4 \vee \ \alpha_1 = 1 \vee \ \alpha_2 = 0 \Rightarrow$ Ref. $f_{14} \Rightarrow$

$\Rightarrow x_1 \leq 10 \ \vee \ x_2 \geq 4 \vee \ \alpha_1 = 1 \vee \ (1 - \alpha_2) = 1$

Model:

1. Logical calculations [Ref. f_{33}]:

IF $\omega_1 = 1$ THEN $x_1 \leq 10 \Rightarrow$ Ref. $f_{14} \Rightarrow x_1 \leq 10 + (UB_{x_1} - 10)(1 - \omega_1)$

IF $\omega_2 = 1$ THEN $x_2 \geq 4 \Rightarrow$ Ref. $f_{15} \Rightarrow x_2 \leq 4\omega_2 + LB_{x_2}(1 - \omega_2) \Rightarrow LB_{x_2} = 0$

$\Rightarrow x_2 \leq 4\omega_2$

2. Quantitative selection specification [Ref. f_{34}]:

$\omega_1 + \omega_2 + \alpha_1 + (1 - \alpha_2) \geq 1$

6.8.3.5 Conjunction Operator (AND; ∧)

When we have a compound proposition where only the disjunction operator appears, it is not necessary to perform any modelling processes. Each atomic proposition corresponds to a restriction in the model.

Instead, the conjunction operator within compound proposals with more operators needs a modelling process, which we will see in Sect. 6.8.4.

6.8.3.6 Exclusive Disjunction Operator (EITHER . . . OR. . .;⊕)

If there are two or more atomic propositions joined with the operator $\bigoplus$:

ϕ_i , i =1,2,. . ., n: $\phi_1 \bigoplus \phi_2 \bigoplus \ldots \bigoplus \phi_n$

Modelling follows two steps:

1. Similar to step 1) of the connective DISJUNCTION (OR), but in this case the logical calculation must be defined as:
   $\forall i/\phi_i \in Z \vee \phi_i \in \Re: \ \phi_i \leftrightarrow \omega_i = 1$ [Reference f_{35}]
2. A quantitative selection specification is incorporated:
   $$\sum_{i/\phi_i \in \Re \vee \phi_i \in Z} \omega_i + \sum_{i/\phi_i \in \{0,1\}} \alpha_i = 1 \ \text{[Reference } f_{36}\text{]}$$

This means that one and only one atomic proposition can be fulfilled.

6.8.4 *Modelling Compound Propositions with Various Operators*

Compound propositions can join several atomic propositions using different operators. Examples can be the following:

Illustration 6.24: Compound Propositions with Several Operators
EITHER (($x_1 \geq 20$ AND $y_1 \leq 10$) OR $\alpha = 1$
(($x_1 \geq 20$ AND $y_1 \leq 10$) OR NOT ($x_3 \geq 20$))
IF (($x_1 \geq 20$ OR $y_1 \leq 10$) THEN NOT ($\alpha = 1$ AND $\beta = 1$)
(($x_1 \geq 20$ AND $y_1 \leq 10$) IF AND ONLY IF ($\alpha = 1$ OR $\beta = 1$)
. . . .

The modelling process of a compound proposition with several operators is done from the lowest level in the structure of the proposition to the highest level. The level is determined by the priority of the operators, according to the structure of parentheses. The lower the level, the higher the execution priority of the operator.

The process will always end with a proposition that has only one type of operator and that will be modelled as defined in Sect. 6.8.3.

We call ψ the proposition that is part of the original compound proposition and in which only one operator type appears. The modelling process of ψ depending on the operator is as follows.

6.8.4.1 Negation Operator (NOT;¬):

The result of the negation operator modelling replaces ψ with $\neg\psi$ in ϕ, but does not incorporate additional constraints into the model.

Illustration 6.25
ϕ: EITHER ($x_1 \geq 8$ AND $x_2 \leq 10$) OR NOT($y \geq 10$)
ψ= NOT($y \geq 10$)
Model of ψ: ⇒Ref. f_{11} ⇒ ($y \leq 9$) [we consider y as integer]
Result: EITHER ($x_1 \geq 8$ AND $x_2 \leq 10$) OR ($y \leq 9$)

6.8.4.2 Disjunction Operator (OR;$\vee$) and Exclusive Disjunction (EITHER... OR...;$\oplus$):

The step 1) of the exclusive disjunction operator described for the cases in which the operator appears individually is modelled (Sect. 6.8.3.6.). This is:

$\psi = (\psi_1 \vee \psi_2 \vee \ldots \vee \psi_i \vee \ldots)$
or
$\psi = (\psi_1 \oplus \psi_2 \oplus \ldots \oplus \psi_i \oplus \ldots)$
$\forall_i/\psi_i \in Z \vee \psi_i \in \mathfrak{R}: \ \psi_i \leftrightarrow \omega_i = 1$ [Reference f_{35}]

The constraint resulting from step 2) of the modelling process (Sect. 6.8.3.4 for disjunction operator and Sect. 6.8.3.6 for exclusive disjunction operator) is not incorporated as a constraint to the model, but instead replaces ψ in ϕ. With this we reduce operators of the original proposition ϕ.

Only for some compound propositions, the following expression for the OR operator may also be valid:

$\forall_i/\psi_i \in Z \vee \psi_i \in \mathfrak{R}: \ \psi_i \leftarrow \omega_i = 1$

Illustration 6.26
ϕ: EITHER ($x_1 \geq 8$ OR $x_2 \leq 10$) OR ($y \leq 9$) [x_1, x_2 integers]
$\psi = (x_1 \geq 8$ OR $x_2 \leq 10)$
Model of ψ:

$$\Rightarrow f_{35} \Rightarrow \omega_1 = 1 \text{IF AND ONLY IF} x_1 \geq 8 \Rightarrow$$
$$\Rightarrow f_{25} \Rightarrow x_1 \geq 8\omega_1 + \text{LB}x_1(1 - \omega_1) \tag{6.16}$$

$$\Rightarrow x_1 \leq 7(1 - \omega_1) + \text{UB}x_1\omega_1 \tag{6.17}$$

$$\Rightarrow f_{35} \Rightarrow \omega_2 = 1 \text{IF AND ONLY IF} x_2 \leq 10$$
$$\Rightarrow f_{24} \Rightarrow x_2 \leq 10 + (\text{UB}x_2 - 10)(1 - \omega_2) \tag{6.18}$$

$$\Rightarrow x_2 \geq 11(1 - \omega_2) + \text{LBx}_2\omega_2 \tag{6.19}$$

(6.16), (6.17), (6.18), and (6.19) are constraints that are incorporated into the model.

Result: $\Rightarrow f_{34} \Rightarrow$ EITHER ($\omega_1 + \omega_2 \geq 1$) OR ($y \leq 9$)

The modelling for this could be carried out as described in Sect. 6.8.3.6.

6.8.4.3 Conjunction Operator (AND; $\wedge$)

This operator had not been used individually for the obvious reasons that there was no need for any modelling exercise. However, within a proposal with more operators, it operates in a similar way to the OR and EITHER OR operators:

$\psi = (\psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_i \wedge \ldots \wedge \psi_n)$
$\forall i / \psi_i \in Z \vee \psi_i \in \Re : \ \psi_i \leftrightarrow \omega_i = 1$ [Reference f_{35}]

2. An expression ϕ_0 is created with the following format:

$$\phi_{0:} \sum_{i/\psi_i \in \Re \vee \psi_i \in Z} \omega_i + \sum_{i/\psi_i \in \{0,1\}} \alpha_i \geq n \quad [\text{Reference } f_{37}]$$

Replacing ψ with ϕ_0 in ϕ.

Illustration 6.27
ϕ: EITHER ($x_1 \geq 8$ AND $x_2 \leq 10$ AND $\beta = 0$) **OR** ($y \leq 9$) [x_1, x_2 integers; β binary]
ψ= ($x_1 \geq 8$ AND $x_2 \leq 10$ AND $\beta = 0$)
Model of ψ:

$$\Rightarrow f_{35} \Rightarrow \omega_1 = 1 \leftrightarrow x_1 \geq 8 \Rightarrow f_{25} \Rightarrow x_1 \geq 8\omega_1 + \text{LB}x_1(1 - \omega_1) \tag{6.20}$$

$$\Rightarrow x_1 \leq 7(1 - \omega_1) + \text{UB}x_1\omega_1 \tag{6.21}$$

$$\Rightarrow \omega_2 = 1 \leftrightarrow x_2 \leq 10 \Rightarrow f_{24} \Rightarrow x_2 \leq 10 + (\text{UB}x_2 - 10)(1 - \omega_2) \tag{6.22}$$

$$\rightarrow x_2 \geq 11(1 - \omega_2) + \text{LB}x_2\omega_2 \tag{6.23}$$

(6.20), (6.21), (6.22), and (6.23) are constraints that are incorporated into the model.

ϕ_0: $\omega_1 + \omega_2 + (1-\beta) \geq 3$
Result: $\Rightarrow f_{37} \Rightarrow$ EITHER ($\omega_1 + \omega_2 + (1-\beta) \geq 3$) OR ($y \leq 9$)

The modelling for this could again be carried out as described in Sect. 6.8.3.6.

6.8.4.4 Conditional and Biconditional Operators

The constraints resulting from operator modelling replace ψ.
Let $\pi_{1 \ldots} \pi_r$ be constraints resulting from operator modelling
Proposition (π_1 AND π_2 AND ... AND π_r) replaces ψ in ϕ.

Illustration 6.28
ϕ: EITHER $x_1 \geq 8$ OR (IF $x_1 + x_2 \geq 8$ THEN $\beta = 1$)
ψ = (IF $x_1 + x_2 \geq 8$ THEN $\beta = 1$)
Model of ψ:

$$\begin{aligned} &\Rightarrow f_3 \Rightarrow \text{IF}\beta = 0\,\text{THEN}\,x_1 + x_2 < 8 \\ &\Rightarrow f_7; f_4 \Rightarrow \text{IF}\,1 - \beta = 1\,\text{THEN}\,x_1 + x_2 \leq 7 \\ &\Rightarrow f_{14} \Rightarrow x_1 + x_2 \leq 7 + (\text{UB}_{x1+x2} - 7)(1 - \omega_1) \end{aligned} \tag{6.24}$$

(6.24) replaces ψ in ϕ:
EITHER $x_1 \geq 8$ OR $x_1 + x_2 \leq 7 + (\text{UB}_{x1\ +\ x2} - 7)(1 - \omega_1)$

The modelling for this could again be carried out as described in Sect. 6.8.3.6.

A couple of illustrations to express the complete process.

Illustration 6.29

$$\phi : \text{IF } (x_1 \geq 20 \text{ OR } y_1 \leq 10) \text{ THEN NOT } (\alpha = 1 \text{ OR } \beta = 0)$$

$$[x_1, y_1 \text{ integers}, \alpha \text{ and } \beta \text{ binaries}]$$

$$\Rightarrow \mathrm{f}_{7;}\ \mathrm{f}_{34} \Rightarrow \text{IF } (x_1 \geq 20 \text{ OR } y_1 \leq 10) \text{ THEN NOT } (\alpha + (1 - \beta) \geq 1)$$

$$\Rightarrow \mathrm{f}_{11} \Rightarrow \text{IF } (x_1 \geq 20 \text{ OR } y_1 \leq 10) \text{ THEN } \alpha + (1 - \beta) \leq 0$$

$$\Rightarrow \mathrm{f}_{35} \Rightarrow \omega_1 = 1 \text{ IF AND ONLY IF } x_1 \geq 20$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x_1 \geq 20\omega_1 + \mathrm{LB}x_1(1 - \omega_1) \tag{6.25}$$

$$\Rightarrow x_1 \leq 19(1 - \omega_1) + UBx_1\omega_1 \tag{6.26}$$

$$\Rightarrow \mathrm{f}_{35} \Rightarrow \omega_2 = 1 \text{ IF AND ONLY IF } y_1 \leq 10$$

$$\Rightarrow \mathrm{f}_{24} \Rightarrow y_1 \leq 10 + (\mathrm{UB}y_1 - 10)(1 - \omega_2) \tag{6.27}$$

$$\Rightarrow y_1 \geq 11(1 - \omega_2) + LBy_1\omega_2 \tag{6.28}$$

$$\Rightarrow \mathrm{f}_{34} \quad \Rightarrow \text{IF } \omega_1 + \omega_2 \geq 1 \text{ THEN } \alpha + (1 - \beta) \leq 0 \Rightarrow$$

$$\Rightarrow \text{IF } \omega_1 + \omega_2 \geq 1 \text{ THEN } \alpha - \beta \leq -1$$

$$\Rightarrow \mathrm{f}_{18} \Rightarrow \omega_1 + \omega_2 \geq \omega \tag{6.29}$$

$$\Rightarrow \omega_1 + \omega_2 \leq 2\omega \tag{6.30}$$

$$\Rightarrow \mathrm{f}_{20} \Rightarrow \alpha - \beta \leq -1 + 2(1 - \omega) \tag{6.31}$$

Therefore, the starting proposition is modelled with a total of seven constraints (6.25–6.31).

Illustration 6.30

$$\phi : \text{EITHER } (x_1 \geq 1 \text{ IF AND ONLY IF } \alpha = 1) \text{ OR } (x_2 \geq 1 \text{ AND } x_3 \geq 1)$$

$$(x_1, x_2, x_3 \geq 0 \text{ integers}, \alpha \text{ binary})$$

$$\Rightarrow \mathrm{f}_{35} \Rightarrow \omega_1 = 1 \text{ IF AND ONLY IF } x_2 \geq 1$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x_2 \geq \omega_1 \tag{6.32}$$

$$\Rightarrow x_2 \leq \mathrm{UB}x_2\ \omega_1 \tag{6.33}$$

$$\Rightarrow \mathrm{f}_{35} \Rightarrow \omega_2 = 1 \text{ IF AND ONLY IF } x_3 \geq 1$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x_3 \geq \omega_2 \tag{6.34}$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow x_3 \leq \ \mathrm{LB}x_3\omega_2 \tag{6.35}$$

$$\Rightarrow \mathrm{f}_{37} \Rightarrow \text{EITHER } (x_1 \geq 1 \text{ IF AND ONLY IF } \alpha = 1) \text{ OR } (\omega_1 + \omega_2 \geq 2)$$

$$\Rightarrow \mathrm{Ref.f}_{25} \ \Rightarrow x_1 \geq \alpha$$

$$\Rightarrow \mathrm{Ref.f}_{25} \ \Rightarrow x_1 \leq \ \mathrm{UB}_{x1}\ \alpha$$

$$\Rightarrow \phi : \text{EITHER } (x_1 \geq \alpha \ \text{ AND } x_1 \leq \mathrm{UB}_{x1}\ \alpha) \text{ OR } (\omega_1 + \omega_2 \geq 2)$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_3 = 1 \text{ IF AND ONLY IF } x_1 \geq \alpha$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x_1 - \alpha \ \geq -(1 - \omega_3) \tag{6.36}$$

$$\Rightarrow x_1 - \alpha \ \leq -(1 - \omega_3) + \mathrm{UB}_{\mathrm{x1}}\omega_3 \tag{6.37}$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_4 = 1 \text{ IF AND ONLY IF } x_1 \leq \mathrm{UB}_{x1}\ \alpha$$

$$\Rightarrow \mathrm{f}_{24} \Rightarrow x_1 - \mathrm{UB}_{x1}\alpha \leq \mathrm{UB}_{x1}(1 - \omega_4) \tag{6.38}$$

$$\Rightarrow x_1 - \mathrm{UB}_{x1}\alpha \ \geq (1 - \omega_5) - \mathrm{UB}_{x1}\omega_4 \tag{6.39}$$

$$\Rightarrow \mathrm{f}_{37} \ \Rightarrow \text{EITHER } (\omega_3 + \omega_4 \geq 2) \text{ OR } (\omega_1 + \omega_2 \geq 2)$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_5 = 1 \text{ IF AND ONLY IF } \omega_3 + \omega_4 \geq 2$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow \omega_3 + \omega_4 \geq 2\omega_5 \tag{6.40}$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow \omega_3 + \omega_4 \leq (1 - \omega_5) + 2\omega_5 \tag{6.41}$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_6 = 1 \text{ IF AND ONLY IF } \omega_1 + \omega_2 \geq 2$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow \omega_1 + \omega_2 \geq 2\omega_6 \tag{6.42}$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow \omega_1 + \omega_2 \leq (1 - \omega_6) + 2\omega_6 \tag{6.43}$$

$$\Rightarrow \mathrm{f}_{36} \Rightarrow \omega_5 + \omega_6 = 1 \tag{6.44}$$

(6.32) to (6.44) are incorporated as constraints to the model.

Regardless of this methodology, which is sufficient for the modelling of any proposition, we can also make use of the distributive law between propositions in order to present the concatenation of propositions in a different way:

If we have ϕ, ψ, and σ propositions, the distributive laws between expressions are defined as:

$\phi \vee (\psi \wedge \sigma) \equiv (\phi \vee \psi) \wedge (\phi \vee \sigma)$ [Reference f_{38}]
$\phi \wedge (\psi \vee \sigma) \equiv (\phi \wedge \psi) \vee (\phi \wedge \sigma)$ [Reference f_{39}]

It is also possible to divide conditional propositions when they are at the highest level of the compound proposition:

We have ϕ_1, ϕ_2, …, ϕ_n, ϕ_m propositions:

IF ϕ_1 v ϕ_2 v … v ϕ_n THEN $\phi_m \Rightarrow$
$\Rightarrow$ IF ϕ_1 THEN ϕ_m
$\Rightarrow$ IF ϕ_2 THEN ϕ_m [Reference f_{40}]

$\Rightarrow \ldots$
$\Rightarrow$ IF ϕ_n THEN ϕ_m
IF ϕ_m THEN $\phi_2 \wedge \phi_2 \wedge \ldots \wedge \phi_n \Rightarrow$
$\Rightarrow$ IF ϕ_m THEN ϕ_1
$\Rightarrow$ IF ϕ_m THEN ϕ_2 [Reference f_{41}]
$\Rightarrow \ldots$
$\Rightarrow$ IF ϕ_m THEN ϕ_n

6.8.5 *Data as Propositions*

Sometimes and whenever the specification refers to one or more sets of elements, we can propose propositions where the wording includes, among its atomic propositions, conditions on element data values. Let us take a look at some simple examples in the following illustration.

Illustration 6.31
There is a system for allocating distribution hubs to supermarkets. We have n hubs and m supermarkets. The distance between hubs and supermarkets and the demand of each supermarket is known. The system has the following specifications:

1. *If the distance between a supermarket and a hub exceeds 50Km, the supermarket cannot be assigned to the hub.*
2. *If a supermarket has a demand higher than 1000 kgs, it will be assigned two hubs.*
3. *If hub 2 is assigned a supermarket, the supermarket should be less than 1 km away.*
4. *If a supermarket assigned to a hub exceeds the distance of 30 km, the hub will be limited to a maximum of ten supermarkets.*

Table of Elements (Table 6.28)

Decision Activities
Action: Allocate hubs to supermarkets
Decision variables: $\alpha_{ij} = 1$ if I allocate Hub i to Supermarket j; 0 otherwise.
Specifications

Table 6.28 Elements of Illustration 6.31

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Hubs	$i = 1 \ldots n$	I_U	Distance	D_{ij}	C (km)	S	...
Supermarkets	$j = 1 \ldots m$	I_U		D_{ij}			
			Demand	M_j	C (Kgs)	W	...

The four specifications of the system are enunciated as conditional logical propositions. Let us take a look at this statement:

1. If the distance between a supermarket and a hub exceeds 50 km, the supermarket cannot be assigned to the hub.

   If we apply it to any supermarket and any hub:

$\forall i,j$: IF $D_{ij} > 50$ THEN $\alpha_{ij}=0$

2. If the supermarket has a demand higher than 1000 kgs, it will be assigned two hubs.

   If we apply it to any supermarket:

$$\forall j : \text{IF } M_j > 1000 \text{ THEN } \sum_{i=1}^{n} \alpha_{ij} = 2$$

3. If hub 2 is allocated to a supermarket, the supermarket must be less than 1 km away.

   If we apply it to any supermarket:

$\forall j$: IF $\alpha_{2j}=1$ THEN $D_{2j}< 1$

4. If a supermarket assigned to a hub exceeds the distance of 30 km, the hub will be limited to a maximum of 10 supermarkets.

   If we apply it to any supermarket and any hub:

$$\forall i,j : \text{IF } \alpha_{ij} = 1 \text{ AND } D_{ij} > 1000 \quad \text{THEN} \quad \sum_{j=1}^{m} \alpha_{ij} \leq 10$$

This casuistry does not imply an additional modelling exercise and the rules previously seen should not be followed. It is only necessary to extract the atomic propositions associated with data of the global proposition and include it as a condition of the elements on which the specification falls.

Let us call the data propositions with the term P_{At}, whether they are one or several data joined by operators.

We first distinguish the case of propositions with operators individually:

Let ϕ be a proposition of variables (Table 6.29).

Table 6.29 Model of propositions with data

Proposition	Modelling	Reference
$\phi \vee P_{At}$	$\forall$element/NOT(P_{At}): ϕ	f_{42}
$\phi \wedge P_{At}$	$\forall$element/P_{At}: ϕ	f_{43}
$\phi \oplus P_{At}$	$\forall$element/NOT(P_{At}): ϕ $\forall$element/P_{At}: NOT (ϕ)	f_{44}
IF P_{At} THEN ϕ	$\forall$element/P_{At}: ϕ	f_{45}
P_{At} IF AND ONLY IF ϕ	$\forall$element/P_{At}: NOT (ϕ) $\forall$element/NOT (P_{At}): ϕ	f_{46}

In the case of several operators, references f_{38}, f_{39}, f_{40}, and f_{41} must be used and operate as follows:

1. If the conditional operator or the biconditional operator does not exist in the upper level:
   1.1 If necessary, the distributive law (Ref. f_{38} and f_{39}) is applied until obtaining a union of propositions of the form:
   (ϕ_1 OR P_{At1}) AND (ϕ_2 OR P_{At2}) AND ...
   1.2. For each compound proposition united with the Operator AND, the following specification is created:
   Ref. $f_{42} \Rightarrow \forall$ element/NOT(P_{At1}): ϕ_1
2. If the conditional operator exists in the upper level:
   2.1. If necessary, apply Ref. f38 and f39 until obtaining a proposal of the form:

IF ϕ_1 OR P_{At1} OR (ϕ_2 AND P_{At2}) OR (ϕ_3 OR P_{At3}) OR ... THEN ψ
Ref. $f_{40} \Rightarrow$ IF ϕ_1 THEN $\psi \Rightarrow$ ...

IF PAt1THEN $\psi \Rightarrow$ Ref. f45 $\Rightarrow \forall$ element/PAt1: ψ
IF (ϕ2 AND PAt2) THEN $\psi \Rightarrow \forall$ element/ PAt2: IF ϕ2 THEN ψ
IF (ϕ3 OR PAt3) THEN $\psi \Rightarrow \forall$ element/ NOT(PAt3): IF ϕ THEN ψ
$\Rightarrow \forall$ element/P_{At3}: ψ
...

Next, we model the propositions of Illustration 6.31.

Illustration 6.32: Modelling the propositions of 6.31
(1)
$\forall i,j$: IF $D_{ij} > 50$ THEN $\alpha_{ij}=0$
$\Rightarrow f_{45} \Rightarrow \forall i,j/D_{ij} > 50$: $\alpha_{ij}=0$
(2)

$$\forall j : \text{IF } M_j > 1000 \quad \text{THEN} \quad \sum_{i=1}^{n} \alpha_{ij} = 2$$

$$\Rightarrow f_{45} \Rightarrow \forall j/M_j > 1000: \sum_{i=1}^{n} \alpha_{ij} = 2$$

(3)
$\forall j$: IF $\alpha 2j=1$ THEN $D2j< 1$
$\Rightarrow f_3 \Rightarrow \forall j$: IF NOT ($D2j< 1$) THEN NOT($\alpha 2j=1$) $\Rightarrow$ IF $D2j \geq 1$ THEN $\alpha 2j= 0$
$\Rightarrow f_{45} \Rightarrow \forall j/D2j \geq 1$: $\alpha 2j=0$
(4)

$$\forall i,j : \text{IF } \alpha_{ij} = 1 \text{ AND } D_{ij} > 1000 \quad \text{THEN} \quad \sum_{k=1}^{m} \alpha_{ik} \leq 10$$

$$\Rightarrow \forall i,j : \text{IF } \left(\forall i, j/D_{ij} > 1000 : \alpha_{ij} = 1\right) \quad \text{THEN} \quad \sum_{k=1}^{m} \alpha_{ik} \leq 10$$

$$\Rightarrow \forall i,j/D_{ij} > 1000 : \text{IF } \alpha_{ij} = 1 \text{ THEN} \quad \sum_{k=1}^{m} \alpha_{ik} \leq 10$$

$$\Rightarrow \text{Ref. } f_{14} \Rightarrow \forall i,j/D_{ij} > 1000 : \sum_{k=1}^{m} \alpha_{ik} \leq 10 + (m-10)\left(1-\alpha_{ij}\right)$$

6.8.6 Logical Propositions That Express Possibility

When the statement of a system refers to possibilities not subject to conditions, no specification is really being established unless some additional imposition is expressed (e.g., "you can buy at most 10 units"). In most cases, possibilities only serve to establish associations between elements to form activities. The verb we use when talking about possibilities is the verb "can." Let us look at an example:

"Provider A can supply units of product 1": The statement does not generate any constraint. It is established that provider A participates in the supply of product 1 action.

"Provider B can supply more than 50 units of product 2": The statement does not create any constraint. It is established that provider B participates in the supply of product 2 action.

If any limitation is included in the statement, then it may be necessary to establish a specification:

"Provider B can ***only*** supply product 1."

In those cases, it is necessary to model the specification by expressing the statement of impossibility, about what cannot be done: If the supplier can only supply product 1, then it cannot supply product 2.

If the possibility statement is part of a logical proposition because it has a condition or is described with any logical connective, then it is necessary to model that proposition in all cases. For the modelling of this type of logical proposition, it is necessary to rephrase the proposition to express it in negative. It is about converting the proposition of possibility into a proposition of impediment. Let us see some illustrations:

1. "*Provider A can supply units of product 1 if provider B does not supply units of that product*": When there is a simple proposition within the compound proposition that expresses possibility, we rephrase the statement to express it as an impediment:
   *"Provider A **cannot** supply units of product 1 if supplier B* ~~does not supply~~ *supplies units of that product"*
2. "Provider B can supply more than 50 units of product 2 if provider A supplies more than 10 units of product 1":
   *"Provider B **cannot** supply more than 50 units of product 2 if provider A supplies* ~~more than 10~~ *less than 11 units of product 1"*

The logical process is simple. The verb "can" expresses possibility. The opposite, "cannot," expresses that there is no possibility, but both are not disjunctive. Being

Table 6.30 Elements of Illustration 6.33

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tasks	$i = 1\ldots15$	I_M	Duration	D_i	C	W	...
Operators	$j = 1\ldots4$	I_U					

able to perform an action includes doing it and not doing it. Not being able to do it expresses only the option of not doing it. That is why it is necessary to model the expression that imposes a specification, which is the negative.

Let's take a look at an illustration based on a mathematical environment.

Illustration 6.33
There is a system of assigning workers to tasks. We have 15 tasks and 4 operators. The tasks have a duration time. Two specifications are established in the assignment:

- *An operator can carry out more than two tasks if he partially performs a task*
- *The working time of an operator may be more than 10 hours in the case of doing more than 3 tasks.*

Based on the description it is clear that the tasks are divisible in the system, because they can be partially carried out by several operators, so they have a measurable character. It is a statement that lacks a description of other norms and an objective; in this case we will only focus on the two specifications indicated.

Table of Elements (Table 6.30)

Decision Activities
Action: Assign [tasks to Operators]
Decision variables: x_{ij} = Amount of time of Task i assigned to Operator j.
Specifications

1. *An operator can carry out more than two tasks if he partially performs a task*

The specification refers to each operator j=1...4.

First, it is necessary to express the calculation of the number of tasks performed by an operator as an auxiliary calculation that will use a logical calculation to know if an operator has been assigned to each task:

Binary logical calculation: Operator assigned to task
Applied to: Each Operator j=1...4 and each task i=1...15
Variables: $\alpha_{ij} = 1$ if operator j is assigned to task i; 0 otherwise. i=1...15; j=1...4
Logical proposition: $\forall i, \forall j: \quad \alpha_{ij} = 1 \quad$ IF AND ONLY IF $\quad x_{ij} > 0$

The number of tasks performed by each operator can be expressed by an auxiliary calculation:

Auxiliary calculation: Number of tasks performed by each operator

Applied to: Each operator j=1…4
Variables: y_j = number of operator tasks j
Constraints that define the calculation:

$$\forall j: \quad y_j = \sum_{i=1}^{15} \alpha_{ij}$$

Second, we also have to create a logical calculation to know if an operator has partially carried out a task:

Binary logical calculation: Operator partially performs a task
Applied to: Each Operator j=1…4 and each task i=1…15
Variables:
$\beta_{ij} = 1$ if the operator j partially performs task i; 0 otherwise. i=1…15; j=1…4
Logical proposition:
$\forall i, \forall j: \quad \beta_{ij} = 1 \quad$ IF AND ONLY IF $\quad x_{ij} > 0 \quad$ AND $x_{ij} < D_i$

We could also create an auxiliary calculation for collecting the total number of partially performed tasks:

Auxiliary calculation: Number of tasks partially performed by an operator
Applied to: Each Operator j=1…4
Variables: z_j = number of partial tasks of the operator j
Constraints that define the calculation:

$$\forall j: \quad z_j = \sum_{i=1}^{15} \beta_{ij}$$

We return to the starting specification:

"*An operator can carry out more than two tasks if he partially completes a task"*

And we express it in negative:

"An operator cannot perform more than two tasks if he does not partially do any task"⇒ "If an operator does not partially do any task, he cannot perform more than two tasks"

Mathematically:

$\forall j: \quad$ IF $z_j = 0 \quad$ THEN $y_j \leq 2$

2. *The working time of an operator may be more than 10 hours in the case of doing more than three tasks.*

Working time can be collected in an auxiliary calculation:

Auxiliary calculation: Working time of an operator
Applied to: Each Operator j=1…4

Variables: w_j = Working time of operator j
Constraints that define the calculation:

$$\forall j: \quad w_j = \sum_{i=1}^{15} x_{ij}$$

We express the specification as an impediment:

"The working time of an operator cannot exceed 10 hours in the case of performing no more than three tasks" ⇒
⇒ "If you perform at most three tasks, the working time of an operator cannot exceed 10 hours"

Mathematically:

$$\forall j: \quad \text{IF } y_j \leq 3 \quad \text{THEN } w_j \leq 10$$

6.9 Objective Criterion

The objective function is the criterion that guides the search for solutions. Defining an objective function in the system results in the complete definition of an optimization problem. As we discussed in the introductory chapter, the illustrations will focus on problems with a single objective function. However, the typologies and modelling of the functions that we will explain below can also serve to develop multiobjective problems or simply to create a function that integrates diverse weighted functions.

Once a criterion has been defined, all the costs, positive or negative (profits), of the activities and calculations that participate in that function will be expressed in the objective function. Therefore, the objective function can be used a priori to identify decision activities or calculations, since any action that entails a cost will correspond to a decision activity or calculation.

The normal or most usual situation in a system is that the unit cost of an activity represented in a variable does not vary whatever the value of the variable. For example, let x be the decision activity associated with buying units from a supplier and let c be the cost of a unit. Typically, the cost c is the cost associated with the purchase of units, regardless of the value of x. The total cost of that activity will be cx. This will happen as long as the variable is binary, since it only takes 2 values (Value 0, no cost; Value 1, cost c). Therefore, the objective function is usually the simplest specification of the system in most cases. It is enough to identify which variables have cost data with respect to the proposed objective.

However, there are problems in which for a generic variable x, integer or continuous, the cost that is applied may depend on the value that variable x takes. The cases that may arise are:

1. The cost of variable x depends on the range of values on which the variable falls

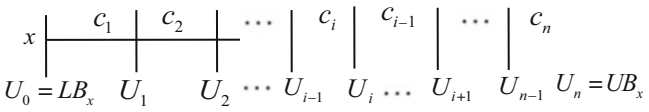


Fig. 6.9 Value intervals

2. The cost of variable x depends on the value that another variable takes
3. The cost depends on the deviation of the variable with respect to reference threshold.

We explain and illustrate below the modelling of each of the cases. Some of the ideas have been based on the modelling presented by Sarker and Newton (2007).

6.9.1 Cost According to Interval of Values

With a variable x, we define a set of n intervals $(U_{i-1}, U_i]$ i=1. . .n, $U_i \geq 0$ i=0. . .n, and a cost c_i associated with each interval (Fig. 6.9).

Since we need to know on what interval $(U_{i-1}, U_i]$, i=1 . . . n the variable x has fallen, it is necessary to define logical calculations, but first, to unify the modelling process, and it is also necessary to define the closed intervals for each cost value. The first interval corresponds to $[U_0, U_1]$. From the second interval, the first value is determined by U_{i-1}+A ($A = 1$ if x is integer, $A = \xi$ if x is continuous). To unify the intervals, we define n intervals $[U_{i-1}+B, U_i]$, where $B = 0$ if i=1, B=A if i> 1.

Binary logical calculation: x belongs to the interval $[U_{i-1}+\mathrm{B}, U_i]$
Applied to: Each interval i=1. . .n
Variables:

$$\alpha_i = \begin{cases} 1 & \text{if } x \in [U_{i-1} + B, U_i] \\ 0 & \text{otherwise} \end{cases}$$

Logical proposition:

$\forall i : \alpha_i = 1$ IF AND ONLY IF $x \geq U_{i\_1} + B$ AND $x \leq U_i$

Model:

$\Rightarrow \mathrm{f}_{35} \Rightarrow \forall i : \omega_{i1} = 1$ IF AND ONLY IF $x \geq U_{i\_1} + B$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x \geq (U_{i-1} + \mathrm{B})\omega_{i1} + \mathrm{LB}_x\,(1 - \omega_{i1}) \tag{6.45}$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x \leq (U_{i-1} + B - 1)(1 - \omega_{i1}) + \mathrm{UB}_x\,\omega_{i1} \tag{6.46}$$

$$\Rightarrow \mathrm{f}_{35} \Rightarrow \forall i : \omega_{i2} = 1 \quad \text{IF AND ONLY IF } x \leq U_i$$

$$\Rightarrow \quad \mathrm{f}_{24} \Rightarrow x \leq U_i + (\mathrm{UB}_x - U_i)(1 - \omega_{i2}) \tag{6.47}$$

$$\Rightarrow \quad \mathrm{f}_{24} \Rightarrow x \geq (U_{i+1})(1-\omega_{i2}) + \mathrm{LB}_x\omega_{i2} \tag{6.48}$$

$$\Rightarrow \mathrm{f}_{37} \Rightarrow \forall i : \alpha_i = 1 \text{ IF AND ONLY IF } \omega_{i1} + \omega_{i1} \geq 2$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow \forall i : \omega_{i1} + \omega_{i1} \geq 2\alpha_i \tag{6.49}$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow \forall i : \omega_{i1} + \omega_{i1} \leq 1 + \alpha_i \tag{6.50}$$

In addition, we need a non-binary logical calculation that collects the value of x in each interval in order to maintain the linearity when expressing the cost in the objective function. Only with the variables α_i,, the definition of the cost of x depending on the interval would be defined as:

$$\sum_{i=1}^{n} c_i \alpha_i x$$

So, we would have a non-linear expression. To avoid this, we must pick up the value of x, according to the interval on which it falls. The calculation would be:

Non-binary logical calculation: Collect the value of x in each interval $[U_{i-1}+B, U_i]$
Applied to: Each interval $i=1\ldots n$
Variables:

$$x_i = \begin{cases} x & \text{if } \alpha_{\mathrm{i}} = 1 \\ 0 & \text{if } \alpha_{\mathrm{i}} = 0 \end{cases}$$

Logical propositions: $\forall i$: IF $\alpha_i = 1$ THEN $x_i = x$
$\forall i$: IF $\alpha_i = 0$ THEN $x_i = 0$
Model:

$$\forall i : \mathrm{IF}\alpha_i = 1 \text{ THEN } \; x_i = x \Rightarrow \mathrm{f}_{16} \Rightarrow \forall i : x \leq x_i + \mathrm{UB}_x(1-\alpha_i) \tag{6.51}$$

$$\Rightarrow \forall i : x \geq x_i \tag{6.52}$$

$$\forall : \mathrm{IF}\alpha_i = 0 \text{ THEN } \; x_i = 0 \Rightarrow \mathrm{f}_7 \Rightarrow \forall i : \text{IF } 1 - \alpha_i = 1 \text{ THEN } \; x_i = 0$$

$$\Rightarrow \mathrm{f}_{16} \Rightarrow \forall i : x_i \leq U_i\alpha_i \tag{6.53}$$

$$\Rightarrow \mathrm{f}_{16} \Rightarrow \forall i : x_i \geq 0 \tag{6.54}$$

The expression of the objective function that collects the cost of the variable x would be defined as:

$$\sum_{i=1}^{n} c_i x_i$$

Simplification of Constraints
Proposition $\forall i : \alpha_i = 1$ IF AND ONLY IF $x \geq U_{i\_1} + B$ AND $x \leq U_i$ can be simplified by Ref. S_V taking advantage of the characteristics of the variables. The

variable x cannot fall onto more than one interval, without imposing it as a condition. That means that it would suffice to impose the relationship between α_i and x:

$$\text{Only variable } \alpha_i \text{ will take value } 1 : \sum_{i=1}^{n} \alpha_i = 1 \tag{6.55}$$

$$\forall i : \text{IF } \alpha_i = 1 \quad \text{THEN} \quad x \geq U_{i-1} + B \ \text{ AND } \ x \leq U_i$$

Model:

$$\Rightarrow \mathrm{f}_{41} \Rightarrow \forall i : \text{IF } \alpha_i = 1 \quad \text{THEN } \ x \geq U_{i-1} + B$$
$$\forall i : \text{IF } \alpha_i = 1 \quad \text{THEN } \ x \leq U_i$$

$$\Rightarrow \mathrm{f}_{15} \Rightarrow \forall i : x \geq (U_{i-1} + B) \ \ \alpha_i \tag{6.56}$$

$$\Rightarrow \mathrm{f}_{14} \Rightarrow \forall i : x \leq U_i \alpha_i + UB_x(1 - \alpha_i) \tag{6.57}$$

In addition to (6.55), (6.56), and (6.57), we should also include (6.51), (6.52), (6.53), and (6.54) to complete the modelling, although even (6.51) and (6.52) can also be substituted for a single equality function:

$$x = \sum_{i=1}^{n} x_i \tag{6.58}$$

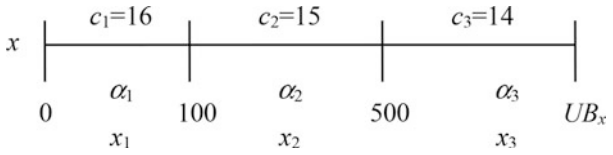
Since with the proposition "IF $\alpha_i = 0$ THEN $x_i = 0$", all x_i take value 0 minus one, that index i for which $\alpha_i = 1$. By imposing (6.58), the x_i that does not take value 0 will automatically take the value of x.

Illustration 6.34

In a purchase system, we have a supplier that offers the following prices for the product:

- *$16 if we buy a maximum of 100 pcs.*
- *$15 if we buy more than 100 pcs.*
- *$14 if we buy more than 500 pcs.*

If x is the variable associated with the activity of purchasing units of the product from that supplier, the cost of x is determined by the interval in which x falls:



The constraints generated by determining the cost would be:

$$x_1 \leq 100\alpha_1$$

By (6.53) $\Rightarrow x_2 \leq 500\alpha_2$

$$x_3 \leq UB_x\alpha_3$$

By (6.55) $\Rightarrow \alpha_1 + \alpha_2 + \alpha_3 = 1$

By (6.56) and (6.57)$\Rightarrow$

$$\left|\begin{array}{l} x \geq 1\alpha_1 \\ x \leq 100\alpha_1 + UB_x(1-\alpha_1) \\ x \geq 101\alpha_2 \\ x \leq 500\alpha_2 + UB_x(1-\alpha_2) \\ x \geq 501\alpha_3 \\ x \leq UB_x \end{array}\right.$$

By (6.58) $\Rightarrow x = x_1 + x_2 + x_3$

In the objective function, we would introduce the cost terms:

Min $\quad \ldots + 16x_1 + 15x_2 + 14x_3 + \ldots$

6.9.2 *Cost According to the Value of Another Variable*

Although the variable that determines the cost is integer or continuous, it will be reduced to depend on the value of one or more binary variables. The modelling is almost identical to the previous case.

Let y be the variable that determines the cost of x, y integer or continuous. Generally, the cost of x will depend on the range of values in which y falls (Fig. 6.10).

To know in which interval $[U_{i-1} + \mathrm{B}, U_i]$ $i = 1\ldots$n the value of y has fallen, we can just define a logical calculation for each interval in the following way:

Binary logical calculation: y belongs to the interval $[U_{i-1}+\mathrm{B}, U_i]$
Applied to: Each interval $i=1\ldots n$
Variables:

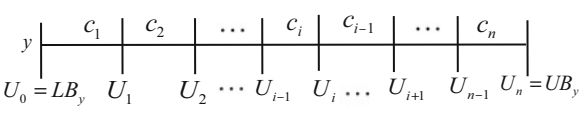


Fig. 6.10 Value intervals for variable y

$$\alpha_i = \begin{cases} 1 & \text{if } y \in [U_{i-1} + B, U_i] \\ 0 & \text{otherwise} \end{cases}$$

Logical propositions: As in the reduction of Sect. 6.9.1, it is not necessary to define the two values of α_i, but only impose that:
Only a α_i will take value 1:

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{6.59}$$

$\Rightarrow$ Ref. $S_V \Rightarrow \forall i$: IF $\alpha_i = 1$ THEN $y \geq U_{i\_1} + B$ AND $y \leq U_i$
Model:

$$\Rightarrow \mathrm{f}_{41} \Rightarrow \forall i : \text{IF } \alpha_i = 1 \quad \text{THEN } y \geq U_{i-1} + B$$
$$\forall i : \text{IF } \alpha_i = 1 \quad \text{THEN } y \leq U_i$$
$$\Rightarrow \mathrm{f}_{15} \Rightarrow \forall i : y \geq (U_{i-1} + B)\ \alpha_i \tag{6.60}$$
$$\Rightarrow \mathrm{f}_{14} \Rightarrow \forall i : y \leq U_i \alpha_i + UB_y(1 - \alpha_i) \tag{6.61}$$

Variables α_i will determine the cost of x:
With α_1=1, the cost of $x = c_1$
With α_2=1, the cost of $x = c_2$
...

If the starting variable y had been binary, the previous process would not be necessary, we would simply continue from this moment, since:

With $y = 1$ ($\alpha_1 = y$), the cost of $x = c_1$
With $y = 0 \Rightarrow \mathrm{f}_7 \Rightarrow 1-y = 1$ ($\alpha_2 = 1-y$), the cost of $x = c_2$

To model this process, it is sufficient to collect the value of x in one variable for each possible cost value:

Non-binary logical calculation: Collect the value of x according to α_i
Applied to: Each interval $i=1...n$
Variables:

$$x_i = \begin{cases} x & \text{if } \alpha_i = 1 \\ 0 & \text{if } \alpha_i = 0 \end{cases}$$

Logical propositions:
IF $\alpha_i = 1$ THEN $x_i = x$
IF $\alpha_i = 0$ THEN $x_i = 0$
Model: (identical to that expressed in 6.9.1) Resulting expressions: (6.51), (6.52), (6.53), and (6.54). Similarly, (6.51) and (6.52) can be reduced to (6.58).

The expression of the objective function that collects the cost of variable x would be defined as:

$$\sum_{i=1}^{n} c_i x_i$$

Illustration 6.35
In a purchase system, we have a product whose purchase cost depends on whether we have signed a contract with the supplier. That contract implies a cost C. The price of the product unit is $\$c_1$ with the signing of the contract and $\$c_2$ without the contract.

The system would have two decision activities:

- Buy product from the supplier
- Sign contract with supplier

This would generate the variables:

$x =$ Product units purchased from the supplier.
$\beta = 1$ If I sign a contract with the supplier; 0 otherwise.
The cost of x depends on the value taken by the variable β
$\beta = 1 \Rightarrow$ Cost of $x = c_1$
$\beta = 0 \Rightarrow 1 - \beta = 1 \Rightarrow$ Cost of $x = c_2$
Constraints generated are:
By (6.53): $x_1 \leq \text{UB}_x \beta$
$x_2 \leq \text{UB}_x (1 - \beta)$
By (6.58): $x = x_1 + x_2$

In the objective function we would include the cost of signing the contract and the cost of purchasing units:

Min $\ldots + C\beta + c_1 x_1 + c_2 x_2 + \ldots$

6.9.3 Costs Depending on the Deviation of the Variable

In some situations, a reference value can be imposed on the values of a variable, so that we can be interested in the approach of the variable to that reference value or we are interested in distancing from it. The distance from this reference is not imposed as a specification in the problem, but the variable has freedom, penalizing or rewarding the deviation on the reference value in the objective function.

The bonuses or penalties imposed affect the units deviated from the reference value.

The following table summarizes all the possibilities that may arise (Table 6.31):

Table 6.31 Cases of deviation

Deviation	Repercussion	Section
Excess	Penalty	6.9.3.1
	Bonus	6.9.3.2
Default	Penalty	6.9.3.3
	Bonus	6.9.3.4

6.9.3.1 Penalty by Excess

Given

U: Reference value (attribute or variable)
p: unit penalty (attribute)
Affected variable: x
Auxiliary variables:
x_d: Deviated units by default of x over U
x_e: Deviated units by excess of x over U

These variables come from defining a free auxiliary variable y, collecting the difference between U and x: $x + y = U$. In order to use variables ≥ 0, we perform the change of variables: $y = x_d - x_e$, $x_d \geq 0$ and $x_e \geq 0$, and the resulting constraint would be:

$$x + x_d - x_e = U$$

Although that expression would allow values to be given simultaneously to x_d and x_e, this would never occur even in the best case of the problem since the excess implies a cost, and therefore it is important that x_e be as low as possible.

In the objective function, the cost term $p\ x_e$ is included.

Illustration 6.36
Within a sales system, we have a customer to whom we offer the following prices for the product:

- *$16 for the first 100 units purchased*
- *$15 for units that exceed 100 units.*

The affected variable would be the quantity sold of product units to the customer, which we call x.

The reference value is $U = 100$
The company suffers a penalty for excess, $p = \$1$ ($16−$15)
Having defined x_d and x_e, this means that:

$$x + x_d - x_e = 100$$

In the objective function we incorporate the terms of the profit of the sale x and the penalty:

$$\text{Max} \quad \ldots + 16x - 1x_e$$

Being a function of maximizing, the penalty has a minus sign because it is a cost.

6.9.3.2 Bonus by Excess

Given

U: Reference value (attribute or variable)
b: unit bonus (attribute)
Affected variable: x
Auxiliary variables:
x_d: Deviated units by default of x over U
x_e: Deviated units by excess of x over U
Binary variables from logical calculations:
α_d: x_d>0 IF AND ONLY IF $\alpha_d = 1$
α_e: x_e>0 IF AND ONLY IF $\alpha_e = 1$

Constraints:

In this case it would not be worth imposing only $x + x_d - x_e = U$, since it interests the greatest possible value of x_e, so that x_d and x_e would grow to infinity simultaneously. Therefore, the logical calculations are defined to know if x_d and x_e have become positive and then it is imposed that both cannot be made positive simultaneously.

Simplifying the definition of the logical calculations to:

α_d: IF x_d>0 THEN $\alpha_d = 1$
α_e: IF x_e>0 THEN $\alpha_e = 1$

The resulting constraints of this process are:

$$x + x_d - x_e = U$$
$$x_d \leq U * \alpha_d$$
$$x_e \leq (UB_x - U*)\alpha_e$$
$$\alpha_d + \alpha_e \leq 1$$

* If the reference value U is a variable, we have to use another upper bound for x_d and x_e in the modeling process.

In the objective function, the profit term bx_e is incorporated and x would enter in the function with its base cost.

Illustration 6.37
Within a purchasing system, we have a supplier that offers the following prices for the product:

- *$16 for the first 100 units purchased*
- *$15 for units that exceed 100 units.*

The affected variable would be the purchased quantity of product units to the supplier, which we call x.

The reference value is U = 100

The bonus for excess is b = $1 ($16−$15)

Having defined x_d, x_e, α_d and α_e, this means that:

$x + x_d - x_e = 100$
$x_d \leq 100\alpha_d$
$x_e \leq (\mathrm{UB}_x - 100)\alpha_e$
$\alpha_d + \alpha_e \leq 1$

In the objective function we incorporate the terms of the cost of x and the bonus

Min $\ldots + 16x - 1x_e$

that in a minimizing function would have a minus sign because it is a profit.

6.9.3.3 Penalty by Default

Equivalent to Sect. 6.9.3.1.
Given

U: Reference value (attribute or variable)
p: Unit penalty (attribute)
Affected variable: x
Auxiliary variables:
x_d: Deviated units by default of x over U
x_e: Deviated units by excess of x over U

Constraint: $x + x_d - x_e = U$

The objective function incorporates the cost term px_d

Illustration 6.38
In a system of production and sale of product units we have signed with a customer to supply 1000 units per month, so that if we do not meet that supply we have a penalty of $P for each unit not delivered.

There is a default penalty with a reference number of $U = 1000$ units.

The affected variable would be the quantity supplied to the customer, which we call x.

The default penalty, $p = \$\mathrm{P}$

Having defined x_d, x_e this means that:

$x + x_d - x_e = 1000$

In the objective function we would incorporate the penalty:

Min $\ldots + Px_d + \ldots$

6.9.3.4 Bonus by Default

Equivalent to Sect. 6.9.3.2.

$$x + x_d - x_e = U$$
$$x_d \leq U\alpha_d$$
$$x_e \leq (UB_x - U)\alpha_e$$
$$\alpha_d + \alpha_e \leq 1$$

If the reference value U is a variable, we have to use another upper bound for x_d and x_e in the modeling process.

In the objective function, the benefit term bx_d is incorporated.

Illustration 6.39
After the Kyoto protocol, the state proposes bonuses on the gas emissions of our company in the case of not exceeding the A kg/year, meliorating with $A for each Kg/year deviated by default.

6.10 Identification of Specifications

The specifications of a system include the standards and operation regulations declared within it.

From the statement or description of the system, the first task for modelling involves the identifying of specifications. Extracting the specifications of a system consists of identifying all the declared norms, both those that are presented explicitly in the statement and those that are assumed from the nature of the elements and activities and do not have an explicit description.

Norms that appear explicitly in the description are easy to identify and one only has to look for verbs of imposition or logical propositions. Those that are found implicitly in a system, without there being the need to declare them, are specifications that are based on data of elements, quantitative selection rules, logical conditions between activities, impositions of flow balance, or bounds of measurable activities:

- Based on data: attributes that express a continuous or integer magnitude of intrinsic capacity, availability, or demand on a collective or measurable element always have a specification associated with capacity consumption, capacity contribution, demand or balance, depending on the system operation. These specifications may not be defined as such, only the attribute. The same happens with relational data between elements, for example, of incompatibility of some action, that probably define constraints regarding decision activities, but they are not made explicit because they are defined with the attribute itself.
- Quantitative selection rules: many systems assume without making explicit the norms that define the quantitative selection specifications for certain logic decision activities between sets of elements. Therefore, it will be necessary to analyze if there are selection rules on each of the elements that participate in the activity (Sect. 6.3).

- Logical conditions between activities: sometimes definitions of decision activities have an implicit relationship between them defined by a logical proposition. This does not mean that any activity represents a calculation, but some of the values of one variable condition the value of another.

When a variable defines a calculation, all its values are obtained from the values of other variables, or they are negligible values in the system. The logical conditions between activities also occur when we identify a logical calculation as a decision activity. By not defining it as a calculation, we ignore the logical proposition that defines it. This logical proposition cannot be ignored from the model, and it would be represented as a specification.

- Bounds of discrete measurable activities: they appear in measurable decision activities in which there is an upper bound of measurement of the activity, individually generally, but also jointly with other measurable activities, without this information being explicitly included in the statement.
- Flow balance constraints: in the system, equilibrium relationships between activities and calculations are established, when there are measurable elements and generally over a set of time periods, and these restrictions are assumed in the operation of the system without these relationships being explicit.

The modeller must analyze the following aspects in the identification of specifications:

– The data of elements that can refer to capacity, availability, or demand, fundamentally
– The selection rules in decision activities
– The decision activities identified in the system in case there is any implicit relationship between them or if any of them were really a logical calculation
– Balance relationships between variables
– The upper bound of discrete measurable activities

The description of a system should always avoid wrong interpretations, so it is desirable that the number of implicit specifications be as low as possible, with all the details indicated in the statement, although some are obvious.

Let's look at some examples of identifying specifications in systems.

Illustration 6.40: Assigning objects to positions (Romero and Romeijn **2005**)
There is a set of n objects and m positions, m> n. Each object has a weight. Each position has a maximum weight supported. It is about assigning objects to positions. There is a cost involved in assigning each object to each position. It is about minimizing the cost of the assignment.

Table of Elements (Table 6.32)

Decision Activities

Action: Assign objects to positions
Decision variables:

Table 6.32 Elements of Illustration 6.40

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Objects	$i = 1...n$	I_U	Weight	p_i	C	W	...
			Cost	c_{ij}	C	S	...
Positions	$j = 1...m$	I_U	Max_Weight	M_j	C	W	...
				c_{ij}			

$\alpha_{ij} = 1$ if I assign Object i to Position j; 0 otherwise. $i=1...n$, $j=1...m$

Specifications

The statement does not present any explicit specification. All specifications are given implicitly in the description:

Specifications I1. Based on data: each position has a capacity attribute, the maximum weight supported; therefore, it will be necessary to define a consumption specification in this case on each position.

Constraints:

$$\forall j : \sum_{i=1}^{n} p_i \alpha_{ij} \leq M_j$$

Specifications I2. Quantitative selection rules: the activities of the system are logic; therefore, it will be necessary to analyze which are the quantitative norms in the selection (Table 6.33):

The most logical analysis is to assume that an object occupies exactly one position. If it could occupy more than one position, it should have been specified in the statement. And that amount is mandatory and does not act as a higher level, since you must place all objects. Therefore, there is an implicit selection rule for each object. Regarding the positions, there is no rule.

Logically with any position, we can always impose as an upper bound all objects and as a lower bound no objects, but those specifications would not be necessary and therefore are not defined.

Table 6.33 Selection diagram

Elements selecting	Selectable elements	Type of Norm	Quantity	Constraints
Object $i = 1...n$	Positions	Upper bound	–	
		Lower bound	–	
		Equality	1	$\forall i : \sum_{j=1}^{m} \alpha_{ij} = 1$
Position $j = 1...m$	Objects	Upper bound	–	
		Lower bound	–	
		Equality	–	

Specifications I3. Logical conditions between activities: there are no relationships between activities, since there is only one activity.
Specifications I4. Bounds of discrete measurable activities: there are no discrete measurable activities.
Specifications I5. Flow balance constraints: they do not exist.

Illustration 6.41: Ham distribution
A ham distribution company has designed a set of 20 delivery routes for distribution. The company has a portfolio of 350 customers. Each delivery route goes through a series of known customers.

The company has ten vehicles for the distribution. Each vehicle has a given capacity or number of Iberian hams that it can transport.

The demand for ham is known from each customer and must be attended to. Each vehicle that delivers Iberian hams must choose a single route, because more than one would take too long. We know the delivery cost of each route.

Table of Elements (Table 6.34)

In the Route_Customer attribute, customers of each route are annotated. It is therefore shared between routes and customers.
Decision Activities

Action: Deliver Iberian hams with vehicles to customers.
Decision variables:
x_{kj} = Number of Iberian hams delivered with vehicle k to customer j.
k=1...10, j=1...350

Action: Choose routes for vehicles.
Decision variables:
α_{ik} =1 if I choose Route i for Vehicle k; 0 otherwise. k=1...10, i=1...20

Explicit Specifications
This statement does present explicit impositions:

– *"The demand for ham from each customer is known and **must be attended to**"*: It is an imposition of demand contribution. The specification refers to satisfying a

Table 6.34 Elements of Illustration 6.41

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Routes	$i = 1...20$	I_U	Route_Customer	RC_{ij}	B	S	...
			Cost	C_i	C	W	...
Customers	$j = 1...350$	I_U	Demand	D_j	I	S	...
				RC_{ij}			
Vehicles	$k = 1...10$	I_U	Capacity	K_k	I	S	...
Iberian hams	–	C_D		D_i; K_k			

Table 6.35 Selection diagram of decision activity "Choose routes for vehicles"

Elements selecting	Selectable elements	Type of norm	Quantity	Constraints
Vehicle $k = 1...10$	Routes	Upper bound	1	$\forall k : \sum_{i=1}^{20} \alpha_{ik} \leq 1$
		Lower bound	–	
		Equality	–	
Route $i = 1...20$	Vehicles	Upper bound	–	
		Lower bound	–	
		Equality	–	

demand, so if that imposition had not been explicitly specified, it would have been logical to identify it implicitly.

Constraints:

$$\forall j : \sum_{k=1}^{10} x_{kj} = D_j$$

Unitary contributions validate the equality sign

– *"Each vehicle that delivers Iberian hams **must choose a single route**"*: Explicitly, a selection rule is being presented for each vehicle in the activity of choosing a route (Table 6.35):

Note that it would be a mistake to assume that the route selection rule for each vehicle would be equal to 1, since we would assign a route to each vehicle. The specification states that a route is chosen for each vehicle that distributes, so we cannot consider that everyone will perform the delivery.

In addition, the phrase "each vehicle that delivers" brings light to an implicit specification existing in the problem, a logical condition between activities (I3).

Implicit Specifications

- **Specifications I1. Based on data:** each vehicle has a capacity attribute, the number of Iberian hams that can be transported; therefore it will be necessary to define a capacity consumption specification for each vehicle. Each customer also has a demand attribute that will give rise to a demand contribution specification, although we have already mentioned that it is given explicitly.

  Constraints: $\forall k : \sum_{j=1}^{350} x_{jk} \leq K_k$

- **Specifications I2. Quantitative selection rules:** the selection rule for each vehicle with respect to routes appears explicitly.
- **Specifications I3. Logical conditions between activities:** as mentioned, between the two decision activities there is a logical condition reflected in the phrase "Each vehicle that delivers ham slices must choose only one route." This phrase refers to the fact that in order for a vehicle to distribute ham, it must have a route assigned to it. If we do not assign a route, it will not distribute Iberian hams.

Logical proposition: If a vehicle delivers ham, then it must have been assigned a route.

(If a vehicle delivers ham to a customer then it must have been assigned a route)

Logical proposition with mathematical formulation:

$$\forall k,j\text{: IF } x_{kj} > 0 \text{ THEN } \sum_{i=1}^{20} \alpha_{ik} = 1$$

In addition to this, it is necessary to contemplate that it is not worth assigning any route, but only one that passes by the customer to whom it delivers:

Logical proposition: If a vehicle delivers Iberian hams to a customer, then the vehicle must have assigned a route that passes by that customer.

Logical proposition with mathematical formulation:

$$\forall k,j\text{: IF } x_{kj} > 0 \text{ THEN } \sum_{i/RC_{ij}=1} \alpha_{ik} = 1$$

This second logical proposition encompasses the previous one, since if it is fulfilled the previous one is fulfilled, so we can omit the first one.

- **Specifications I4. Bounds of discrete measurable activities:** Measurable activities do not have a given upper bound.
- **Specifications I5. Flow balance constraints:** The measurable activities participate in specifications that have already been reflected.

Illustration 6.42: Supermarket Allocation

There is a supermarket company that has several locations (j =1. . .6) to install a maximum of 3 product distribution centers.

The cost of installing a center in each location is established in CI_j m.u.

The Company has 30 supermarkets to be supplied from locations with distribution centers.

In addition, the following rules must apply in the system:

- *Each location with a distribution center can supply a maximum of ten supermarkets.*
- *For legal requirements, if the company installs a center in location 3 and another in location 5, it cannot install any in location 6.*

Objective Function

Minimize the cost of the problem taking into account that if the number of supermarkets assigned to a location is less than 8, it is penalized with a cost of F m.u.

Table of Elements (Table 6.36)

In the configuration carried out, the locations have been considered as unitary, since they are different, apart from the fact that they are referred to in a particular way. Distribution centers and supermarkets are considered collectives, since they are identical items, determined in the case of supermarkets and indeterminate in the case

Table 6.36 Elements of Illustration 6.42

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Locations	$j = 1...6$	I_U	Cost	CI_j	C	W	...
			Max_Supermarkets	MS	I	S	5
Distribution centers	–	C_I	Maximum quantity	M	I	W	3
Supermarkets	–	C_D	Quantity	S	I	W	30
				MS			

of distribution centers. And on the other hand, we can avoid referring to those instances in a particular way in the statement, so we only refer to numerals of the collective element.

Decision Activities
Action: Install distribution centers in locations.
Decision variables:
x_j = Number of distribution centers installed in location j. $j = 1...6$

Action: Supply supermarkets from locations.
Decision variables:
y_j =Number of supermarkets supplied from location j; $j = 1...30$
Explicit Specifications

The statement explicitly presents the following specifications:

E1. *"install a maximum of 3 product distribution centers":*
E2. *"30 supermarkets to be supplied from locations with distribution centers"*
E3. *"Each location with a distribution center can supply a maximum of 10 supermarkets"*
E4. *"If the company installs a center in location 3 and another in location 5, it cannot install any in location 6"*

The first three correspond to specifications that are based on data, although they are explicitly described. The fourth specification corresponds to a logical proposition.

Constraints:

E1. $\sum_{j=1}^{6} x_j \leq 3$
E2. $\sum_{j=1}^{6} y_j = 30$
E3. $\forall j : y_j \leq 10$
E4. IF y_3=1 and y_5=1 THEN y_6=0

Implicit Specifications

- **Specifications I1. Based on data:** as mentioned, the specifications that could be based on data have been made explicit.
- **Specifications I2. Quantitative selection rules:** there are no decision activities selected.
- **Specifications I3. Logical conditions between activities:** between the two activities there is a conditional relationship, which is mentioned in the following sentence:

"30 supermarkets to be supplied from locations with distribution centers"

We have considered as explicit the norm of supplying a total of 30 supermarkets, but in this sentence, we also allude to the conditional relationship between the two activities: in order for a location to supply supermarkets, it must have installed a distribution center.

Mathematically:

We are going to express it negatively because we are dealing with a proposition of possibility:

"A location can supply supermarkets if it has a distribution center"
"If a location does not have a center installed, then it cannot supply any supermarket"

$\forall j$: IF $x_j = 0$ THEN $y_j = 0$

- **Specifications I4. Bounds of discrete measurable activities:** in the problem we have two measurable activities. The first of the activities, installing centers in locations, carries an implicit type I4 specification, since the most sensible thing to do is to assume that a distribution center will be installed in a physical location at most. It would be strange to think that there may be more or other specifications.

  $\forall j$: $x_j \leq 1$

Regarding the second activity, supplying supermarkets, its measurement is not limited by a specific value.

- **Specifications I5. Flow balance constraints:** they do not exist.

References

Bang-Jensen, J., & Gutin, G. (2000). *Digraphs: Theory, algorithms and applications*. Berlin: Springer.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*, 269–271.

Graham, R. L., & Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing, 7*(1), 43–57.

Hwang, F. K., Richards, D. S., & Winter, P. (1992). *The Steiner tree problem* (Annals of discrete mathematics. 53). North-Holland: Elsevier.

Larrañeta, J., Onieva, L., & Lozano, S. (1995). *Métodos Modernos de Gestión de la Producción*. Madrid: Alianza Editorial.

Mitra, G., Lucas, C., & Moody, S. (1994). Tools for reformulating logical forms into zero-one mixed integer programs. *European Journal of Operational Research, 72*, 262–276.

Öztürk, Ö., Gazibey, Y., & Gerdan, O. (2015). The triple test algorithm to get feasible solution for transportation problems. *International Journal of Numerical Methods and Applications, 13*, 37–50.

Romero, D., & Romeijn, H. E. (2005). The generalized assignment Problem and Extensions. In D.-Z. Du & P. M. Pardalos (Eds.), *Handbook of combinatorial optimization* (Vol. 5, pp. 259–311). Boston: Springer Kluwer Academic Publishers.

Sarker, R. A., & Newton, C. S. (2007). *Optimization modelling. A practical approach*. New York: CRC Press.

Williams, H. P. (1995). Logic applied to integer programming and integer programming applied to logic. *European Journal of Operational Research, 81*, 605–616.

Williams, H. P. (2009). *Logic and integer programming* (pp. 71–103). New York: Springer.

Williams, H. P. (2013). *Model building in mathematical programming* (5th ed.). Wiley.. ISBN: 978-1-118-44333-0