AIMMS Modeling Guide - Algebraic Representation of Models

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com.

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
SOHO Fuxing Plaza No.388
Building D-71, Level 3
Madang Road, Huangpu District
Shanghai 200025
China
Tel.: ++86 21 5309 8733

Email: info@aimms.com
WWW: www.aimms.com

This documentation was typeset by AIMMS B.V. using LATEX and the LUCIDA font family.

Chapter 3

Algebraic Representation of Models

This chapter

In this chapter, the method of translating an explicit formulation to an AIMMS formulation is explained. A sequence of different representations of the same model demonstrates the use of *symbols* to represent data, the use of *index notation*, and the AIMMS *modeling language*.

References

The notation in this chapter is derived from standard mathematical notation. For the representation of models, you are referred to [Sc91] and [Wi90].

3.1 Explicit form

This section

In this section, the potato chips example from the previous chapter is revisited. The formulation below is usually referred to as the *explicit* form in standard algebraic notation. *Algebraic notation* is a mathematical notation, as are other notations such as matrix notation, or the AIMMS notation in Section 3.4. With the help of this example, the differences between several representations of the same model are illustrated.

Potato chips model

Variables:

X_p	*amount of plain chips produced [kg]*
X_m	*amount of Mexican chips produced [kg]*

Maximize:

$$2X_p + 1.5X_m \qquad \text{(net profit)}$$

Subject to:

$$2X_p + 4X_m \leq 345 \qquad \text{(slicing)}$$
$$4X_p + 5X_m \leq 480 \qquad \text{(frying)}$$
$$4X_p + 2X_m \leq 330 \qquad \text{(packing)}$$
$$X_p, X_m \geq 0$$

The above formulation is a correct representation of the problem description in mathematical form. However, it is *not a practical* mathematical description of the problem.

Unexplained numbers

The most obvious shortfall of the explicit form is that the numbers in the model are given without comment. While examining the model one must either look up or recall the meaning of each number. This is annoying and does not promote a quick understanding of the model. In larger models, it can cause errors to go unnoticed.

Possible improvements

It is better to attach a descriptive symbol to each number or group of numbers, plus a brief description for even further clarification. Entering these symbols into the model formulation instead of the individual numbers will lead to a model statement that is easier to understand. In addition, it paves the way for a more structured approach to model building. Specifically, if the values associated with a symbol change at a later stage, then the changes only need to be made at one place in the model. This leads to a considerable improvement in efficiency. These remarks give the motivation for *symbolic* model formulation.

3.2 Symbolic form

Separation between symbols and values

In the symbolic form, there is a separation between the symbols and their values. A model in symbolic form consists of the following building blocks:

- *symbols (parameters)*, representing data in symbolic form,
- *variables*, representing the unknowns, and
- *objective and constraints,* defining the relationships between symbols and variables.

The *data* is not a part of a symbolic model formulation. Values are assigned to the symbols when the model is solved. The data for the potato chips model can be found in Chapter 2.

Potato chips model

Parameters:

A_S	*available slicing time [min]*
A_F	*available frying time [min]*
A_P	*available packing time [min]*
N_p	*net profit of plain chips [$/kg]*
N_m	*net profit of Mexican chips [$/kg]*
S_p	*time required for slicing plain chips [min/kg]*
S_m	*time required for slicing Mexican chips [min/kg]*
F_p	*time required for frying plain chips [min/kg]*
F_m	*time required for frying Mexican chips [min/kg]*
P_p	*time required for packing plain chips [min/kg]*
P_m	*time required for packing Mexican chips [min/kg]*

Nonnegative variables:

X_p	*quantity of plain chips produced [kg]*
X_m	*quantity of Mexican chips produced [kg]*

Maximize:

$$N_p X_p + N_m X_m \qquad \text{(net profit)}$$

Subject to:

$$\begin{aligned} S_p X_p + S_m X_m &\le A_S && \text{(slicing time)} \\ F_p X_p + F_m X_m &\le A_F && \text{(frying time)} \\ P_p X_p + P_m X_m &\le A_P && \text{(packing time)} \\ X_p, X_m &\ge 0 \end{aligned}$$

This representation is evaluated and discussed below.

Too many symbols

In this small example, eleven parameters and two variables are needed to generate a symbolic description of the model. Imagine a situation in which the number of production processes and the number of chip types are both in double figures. The number of constraints will be in the tens but the number of parameters will be in the hundreds. This is clearly unacceptable in practice. The way to compact the formulation is to use *index notation*, as explained in Section 3.3.

Meaningful names for symbols

It is worthwhile to note that the names of the symbols have not been chosen arbitrarily. Although they are short, they give more meaning than a number. For instance, the S which indicates the slicer in A_S (available slicing time) also indicates the slicer in S_p (time required for slicing plain chips). Furthermore, the A in A_S obviously denotes availability. It is important to choose the names of the symbols in a sensible way because it improves the clarity of the model. However, as observed earlier, there are quite a lot of symbols in the model statement above. The larger the model, the more inventiveness one requires to think of meaningful, unique names for all the identifiers. Again, index notation provides a way out, and thus, the naming of symbols will be reconsidered in the next section.

Separation of model and data

When the data is kept separate from the symbolic model statement, the model statement can describe a whole range of situations, rather than one particular situation. In addition, if changes occur in the data, these changes only have to be made in one place. So the separation of model and data provides greater flexibility and prevents errors when updating values.

3.3 Symbolic-indexed form

This section

Index notation is a technique for reducing the number of symbols and facilitating the naming of parameters. Consider the potato chip example using this new, compressed formulation.

Indicating an element of a set

According to Webster's dictionary [We67], one of the meanings of the word *index* is pointer. It points to, or indicates an element of a set. The terms, *set* and *index*, are elaborated further using the potato chips example.

Set of chip types

Recall the notation in the previous example, for instance: X_p "amount of plain chips produced." It is clear that the "p" indicates plain chips. So the "p" is used as an index, but it only points to a set with one element. The difficulty encountered in the previous section, where there were too many symbols, was caused by having all indices pointing only to single-element sets. When combining these sets with similar entities, the number of symbols can be reduced. The first set that seems logical to specify is a set of chip types:

$$I = \{\textit{plain}, \textit{Mexican}\}$$

Then one can state:

x_i	*amount of chips produced of type i [kg]*

Index notation

So the index i indicates an element of the set I, and the two decision variables are now summarized in one statement. It is customary to use *subscripts* for indices. Moreover, the mathematical shorthand for "i taken from the set I" is $i \in I$. The index i for every symbol referring to chip types in the model can be introduced to obtain four new parameter declarations.

Parameters:

n_i	*net profit of chips of type i [\$/kg]*
S_i	*time required for slicing chips of type i [min/kg]*
F_i	*time required for frying chips of type i [min/kg]*
P_i	*time required for packing chips of type i [min/kg]*

The number of parameters has been reduced from eleven to seven by adding one set. Note that indices do not need units of measurement. They just indicate certain entities—elements of a set.

Set of production processes

What is striking in the above list is the similarity of S_i, F_i, and P_i. All three symbols are for time requirements of different production processes. In a way, S, F, and P serve as indices pointing to single element sets of production processes. Because the processes all play a similar role in the model, one more general set an be introduced.

$$J = \{\textit{slicing}, \textit{frying}, \textit{packing}\}$$

An index j, pointing to members of J, can take over the role of S, F, and P. Now one symbol can summarize the six symbols S_p, S_m, F_p, F_m, P_p, P_m that were previously needed to describe the time required by the production processes.

r_{ij}	*time required by process j for chips of type i [min/kg]*

The index j can also be used to express the availabilities of the machines that carry out the processes.

a_j	*available processing time for process j [min]*

At this point two sets (I and J) and three parameters (a_j, n_i, r_{ij}) remain. The notation for the constraint specifications can also be compacted using indexing.

Summation operator

When looking at the first constraint, and trying to write it down with the notation just developed, the following expression can be obtained.

$$r_{mexican,slicing} x_{mexican} + r_{plain,slicing} x_{plain} \leq a_{slicing}$$

Obviously there is room for improvement. This is possible using the well-known summation operator; now used to indicate a summation over different elements of the set of chip types,

$$\sum_i r_{ij} x_i \leq a_j \quad \forall j$$

where $\forall j$ is shorthand notation meaning *for all elements j* (*in the set J*).

Symbolic-indexed formulation

The symbols defined above are used in the following *indexed* formulation of the potato chips problem with the actual numeric data omitted.

Indices:

i	*chip types*
j	*production processes*

Parameters:

a_j	*available processing time of process j [min]*
n_i	*net profit of chips of type i [$/kg]*
r_{ij}	*time requirements of type i and of process j [min/kg]*

Variables:

x_i	*amount of chips produced of type i [kg]*

Maximize:

$$\sum_i n_i x_i \qquad \text{(net profit)}$$

Subject to:

$$\sum_i r_{ij} x_i \leq a_j \qquad \forall j \qquad \text{(time limits)}$$

$$x_i \geq 0 \qquad \forall i$$

Reducing the number of statements

In previous statements of the potato chips model, there were always three constraints describing the limited availability of different production processes. In the symbolic indexed formulation, the use of the index j for production processes enables one to state just one *grouped* constraint, and add the remark "$\forall j$" (for all j). Thus, index notation provides not only a way to summarize many similar identifiers, but also to summarize similar equations. The latter are referred to as *constraint declarations*

Reducing the number of identifiers

In the previous section, it was noted that index notation would also be helpful in reducing the number of identifiers. Using indices of group parameters and variables has reduced the number of identifier descriptors from thirteen to four.

More meaningful names

As a result of reducing the number of identifiers, it is easier to choose unique and meaningful names for them. A name should indicate the common feature of the group. For algebraic notation, the convention is to choose single letter names, but this marginally improves the readability of a model. At most, it contributes to its compactness. In practical applications longer and more meaningful names are used for the description of identifiers. The AIMMS language permits the names of identifiers to be as long as you find convenient.

Expanding the model with set elements

Note that the size of a set can be increased without increasing the length of the model statement. This is possible because the list of set elements is part of the data and not part of the model formulation. The advantages are obvious. Specifically, the number of indexed identifiers and the number of indexed equations are not impacted by the number of set elements. In addition, as with the rest of the data, changes can be made easily, so index notation also contributes to the generality of a model statement. When symbolic notation is introduced there is separation between the model statement and the data. This separation is complete when index notation is used.

3.4 AIMMS form

This section

The last step is to represent the model using the AIMMS modeling language. This yields the advantages that error checks can be carried out, and that the software can activate a solver to search for a solution.

Models in AIMMS

By using the AIMMS Model Explorer, a model created in AIMMS is essentially a graphical representation. At the highest level there is a tree to structure your model in sections and subsections. At the leaves of the tree you specify your declarations and procedures. For each identifier declaration there is a form by which you enter all relevant attributes such as `index domain`, `range`, `text`, `unit`, `definition`, etc.

Example

Figure 3.1 gives you an idea on how the symbolic-indexed representation of the potato chips problem can be structured in the Model Editor. Note that in AIMMS, the full length descriptor of `ProcessTimeRequired`(p, c) replaces the r_{ij} which was used in the earlier mathematical formulation. Clearly, this improves the readability of the model. In AIMMS, symbols are still typically used for set indexing. The set of chips is given the index c and the set of processes, the index p. In the earlier mathematical representation, i and j were used for these sets respectively.

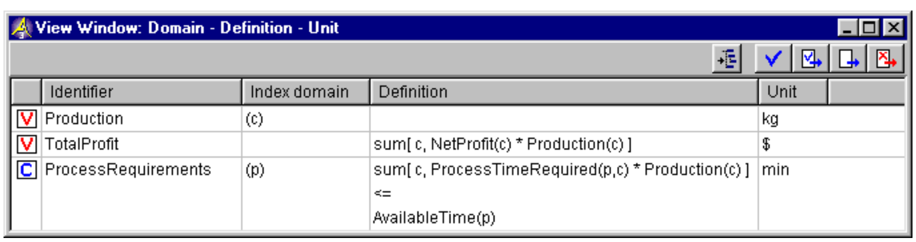
Figure 3.1: AIMMS Model representation of the potato chips model

Multiple views

The graphical tree representation of models inside the Model Explorer is just one way to view a model. In large-scale applications it is quite natural to want to view selected portions of your model. AIMMS allows you to create your own identifier selections and your own view windows. By dragging a particular identifier selection to a particular view window you obtain your own customized view. You may also edit your model components from within such a customized view.

Example

Figure 3.2 gives an example of a view in which the variables and constraints of the potato chips problem are listed, together with their index domain, definition and unit. Note that the AIMMS notation in the definition attributes resembles the standard algebraic index notation introduced in the previous section.


View Window: Domain - Definition - Unit

	Identifier	Index domain	Definition	Unit
V	Production	(c)		kg
V	TotalProfit		sum[c, NetProfit(c) * Production(c)]	$
C	ProcessRequirements	(p)	sum[c, ProcessTimeRequired(p,c) * Production(c)] <= AvailableTime(p)	min

Figure 3.2: An AIMMS view for the potato chips model

Data initialization

Data must be initialized and added to an AIMMS model formulation because the computer needs this data to solve the model. More than one such data set can be associated with a model, allowing for different versions of the same model. The data set for the potato chips problem is presented in the form of an text file. In most real-world applications such data would be read directly by AIMMS from a database.

3.5 Translating a verbal problem into a model

Getting verbal problem description

Throughout this book, the same sequence of steps will be used when translating a verbal problem description into an optimization model. It is assumed that a verbal problem description, posed in such a way that a model can be used to support the decision, is available. Of course, the translation from a real-life problem into a well-posed verbal problem statement is far from trivial, but this exercise is outside the scope of this book.

A general framework

The framework for analyzing a verbal problem is presented below. Such a framework has the advantage that it facilitates a structured approach.

When analyzing a problem in order to develop a model formulation the following questions need to be answered.

- *Which* sets *can be specified for indexing data and variables?*

Such sets have just been explained. The advantages mentioned in Section 3.3 justify the use of index notation throughout the remainder of this manual. Sets often appear in verbal problem descriptions as lists of similar entities, or as labels in tables, such as the production processes in Table 2.1.

- *What are the* decision variables *in the problem?*

Decision variables reflect a choice, or a trade-off, to be made. They are the unknowns to be determined by the model. In fact, the decision reflected in the decision variables is often the very reason for building the model.

- *What entity is to be* optimized*?*

In small examples, the *objective* is often stated explicitly in the problem description. In real-world problems, however, there may be many, possibly conflicting, objectives. In these cases, it is worthwhile experimenting with different objectives.

- *What* constraints *are there?*

Constraints can also include procedural checks on solutions to see if they are usable. A potential solution that does not satisfy all constraints is not usable. The two questions about the objective and constraints can often be answered simultaneously. It is strongly recommended that you specify the measurement *units* of the variables, the objective and the constraints. Since this is a way of checking the consistency of the model statement and can prevent you from making many mistakes.

Potato chips model

The answers to these questions for the potato chips problem have already been given implicitly. They are summarized here once more. The *sets* in the potato chips problem are given by the sets of production processes and types of chips. The *decision variables* are the amounts of both types of chips to be produced, measured in kilograms. The *objective* is net profit maximization, measured in dollars. The *constraints* are formed by the limited availability of the production processes, measured in minutes.

3.6 Summary

This chapter has shown a sequence of different representations of the same model in order to introduce the use of symbols, index notation and the AIMMS language. While using an *explicit* (with numeric values) form of standard algebraic notation may initially be the intuitive way to write down a model, this form is only suitable for the simplest of models. A superior representation is to replace numbers with symbols, thereby obtaining a *symbolic* model representation . Advantages of this representation are that you do not have to remember the meanings of the numbers and that the data, which does not influence the model structure, is separated from the model statement. Another refinement to model formulation is to specify index sets and to use indices to group identifiers and constraints. This yields the *symbolic-indexed* form. This form is recommended because it combines the advantages of the symbolic form with the compactness of index notation. Finally, the sequence of steps to translate a verbal description of a problem to a mathematical programming model was given.

Bibliography

[Sc91] L. Schrage, *Lindo: An optimization modeling system*, 4th ed., The Scientific Press, South San Francisco, 1991.

[We67] *Webster's seventh new collegiate dictionary*, G. & C. Merriam Company, Springfield, Massachusetts, 1967.

[Wi90] H.P. Williams, *Model building in mathematical programming*, 3rd ed., John Wiley & Sons, Chichester, 1990.