DTIC FILE COPY

(2)

AD-A222 991

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release: distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S): Preliminary TR/1/90	5. MONITORING ORGANIZATION REPORT NUMBER(S): R&D 5338-CC-03
6a. NAME OF PERFORMING ORGANIZATION: Brunel University; 6b OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: USARDSG(UK)
6c. ADDRESS (City, State, and ZIP Code): Uxbridge, Middlesex, UB8 3PH	7b. ADDRESS (City, State, and ZIP Code): Box 65, FPO NY 09150-1500
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: USARDSG (UK); 8b. OFFICE SYMBOL (If applicable): AMXSN-UK-RI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: DAJA-45-87-C-0003
8c. ADDRESS (City, State, and ZIP Code): Box 65, FPO NY 09150-1500	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 61102A; PROJECT NO. 1L161102BH57; TASK NO. 06; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

(U) Computer Assisted Analysis and Modelling of Structured Problems

12. PERSONAL AUTHOR(S): E Hadjiconstantinou, G Mitra

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
6th interim	FROM Dec 88 TO Mar 90	29 May 1990	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Boolean Algebra, Integer and Mixed Integer Linear Programs logical forms, propositional calculus, reformulation of logical forms to 0-1 MIP forms

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This sixth report provides an outline of (a) research that was carried out during this period, (b) the set of investigations already initiated and (c) future research directi

During this period Dr E Hadjiconstantinou joined the project as a part time research investigator

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: Dr. John Zavada	22b. TELEPHONE (Include Area Code): 01 409 4423; 22c. OFFICE SYMBOL: AMXSN-UK-RI

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE: Unclassified

(A)

1. Scientific Work

Our research efforts are focussed on developing a systematic procedure for transforming a set of logical conditions imposed on a mathematical optimisation model into an integer linear programming formulation. Through reformulation of logical forms into integer forms we support a uniform and powerful representation of a problem, consisting of a tightly interrelated closed system of choices.

We describe a systematic approach for transforming statements in Boolean Algebra into integer or mixed integer linear programmes. The method is particularly suitable as a modelling technique that allows logical relationships connecting variables and linear constraints to be modelled as integer and mixed integer programmes. We are preparing and illustrating a few example (logic) problems processed by this method which are set out to explain reformulation and modelling techniques.

2. Research Plans

In order to test out our mathematical ideas we are preparing a number of test models. From these we will design a syntax specification for presenting discrete optimization models in logical forms. We will also specify the target mixed integer programming forms which will be consistent with current CAMPS representation of linear forms. As indicated in the last report, overall research plans have a slightly modified goal which is to introduce a knowledge based systems shell as a vehicle for implementing the reformulation techniques. An internal specification for capturing this information and carrying out the analysis of bounds and generating the target mixed integer programmes, will also be prepared. We will then decide whether to use logic programming or one of the other knowledge representation methods to implement the system.

3. Administrative Change

Dr. C Lucas left as a full time investigator. We tried to find a part time investigator who could wrap up the rest of this research project. As stated earlier, a fall in the value of the dollar and Dr. Lucas's resignation has forced us to make the best use of research funding in this way. We were in search of a person already knowledgeable in this field of integer programming and who has sufficient enthusiasm for this research. We have been very fortunate in finding Dr. Eleni Hadjiconstantinou as a part time research investigator with all these attributes. She was a part time lecturer at Brunel University until the end of last year and is currently a full time lecturer at Imperial College, London University. She has already made some progress in preparing a preliminary report. We have worked together to put the project back on a reasonable time schedule.

4. Other Information

(a) We had planned to present a paper on this work at the Las Vegas, TIMS/ORSA meeting which was held on 7, 8, 9 May, 1990. At the last moment Dr. Hadjiconstantinou could not travel to USA as her visa application was not made in time to go through the normal procedures. We are planning to present this paper at a forthcoming mini symposium APMOD91.

(b) We are continuing with Dr. Lucas and with some support from Dr. Hadjiconstantinou with the preparation of our book on modelling. The book will be published by Academic Press and the support given by the US Army's European Research Office will be acknowledged.

5. Financial Annexe

See attached.

6. REFERENCES

1. Beaumont, N., A logical branch and bound algorithm. Paper presented at Australian Society for Operations Research Conference on Mathematical Programming, Melbourne, Australia (1986).

2. Berghel, H., Crossword compilation with horn clauses. The Computer Journal, 30 (2), 183-188, 1987

3. Blair, C.E., Jeroslow, R.H., and Lowe, J.K., Some results and experiments in programming techniques for propositional logic. Computers and Operations Research, 13 (5), 633-645, 1986.

4. Brearley, A.L., Mitra, G., and Williams, H.P., Analysis of Mathematical Programming Models Prior to Applying the Simplex Algorithm, Math. Prog., vol 8, pp 54-83.

5. Darby-Dowman, K., Lucas, C., Mitra, G., and Yadegar, J., Linear, Integer, Separable and Fuzzy Programming Problems: A unified approach towards reformulation, J.Op.Res.Soc., vol 39, No. 2, 161-171, 1988.

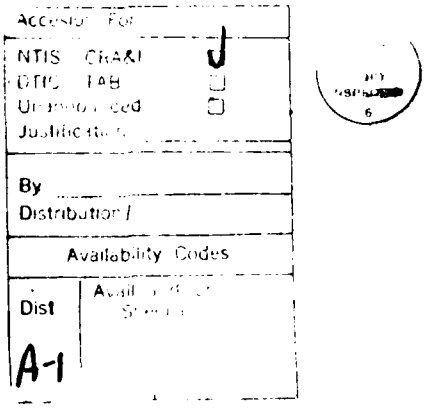
6. Lucas, C. and Mitra, G., Computer Assisted Mathematical Programming Modelling System: CAMPS, User Manual, Brunel University and NAG Ltd., 1988.

7. Williams, H.P., Linear and integer programming applied to the propositional calculus. International Journal of Systems Research and Information Science, 2, 81-100, 1987.

8. Williams, H.P. and McKinnon, K.I.M., Constructing integer programming models by predicate calculus, presented to the 13th International Mathematical Programming Symposium, TOKYO, 1988.

9. Wilson, J.M., Crossword compilation using integer programming, The Computer Journal, vol 32, No. 3, 1989.

kusarep.gm



TRANSFORMATION OF LOGICAL EXPRESSIONS INTO A SYSTEM OF INTEGER LINEAR CONSTRAINTS: AN APPROACH TOWARDS AN AUTOMATIC CONVERSION TO DISCRETE PROGRAMMING MODELS.

E. Hadjiconstantinou, Imperial College, London
G. Mitra, Brunel University, Uxbridge, London

This paper presents a systematic procedure for transforming a set of logical conditions imposed on a model into an integer linear formulation. ILP supports a uniform and powerful representation of a problem, consisting of a tightly interrelated closed system of choices, as a system of linear constraints with an objective function. It supports direct representation of arbitrary Boolean expressions. The method adopted to achieve this is to introduce binary variables (hereafter called logical variables) and re-express logical relationships amongst constraints in terms of " simple " constraints and logical variables i.e. generating logical constraints (a logical constraint is a logical combination of "simple" constraints).

Propositional or Statement Calculus

By a " statement " we define a declarative sentence. For example,

"Athens is the capital of Greece"

and

"Five is an even number"

are statements. This type of statement, about which it is possible to say that it is either *true* or *false* but not both, is called a **proposition** (for our purpose, propositions and statements are synonymous words). A proposition can take the **truth value** either true or false i.e. the truth value of a true proposition is TRUE (abbreviate to T) and the truth value of a false proposition is FALSE (abbreviate to F). No other value is permitted and the calculus of propositions thus refers to a two-valued logic. The above two propositions are true and false respectively.

Propositional calculus enables further propositions to be formed by modifying a simple proposition with the word *not* or by connecting propositions with the words *and*, *or*, *if ... then* (or *implies*) and *if and only if*. These five words are called propositional or logical **connectives** and can be used to build **compound propositions** from

given simple propositions. More generally, they can be used to construct more complicated compound propositions from compound propositions by applying them repeatedly. The connective structure of a compound proposition is described in terms of its constituent **individual propositions** (that is , statements which contain neither connectives nor any other proposition as a component part). The connectives used here are given below with their usual interpretation:

- **Negation**: a proposition which is modified by the word "not" is called the negation of the original proposition.
- **Conjunction**: a compound proposition formed by inserting the word "and" between two propositions.
- **Disjunction**: when two propositions are combined disjunctively by inserting the word "or" between them, the resulting compound proposition is a disjunction.

  There are two meanings of the "or" connective: the *inclusive or* i.e. at least one disjunct is true and the *exclusive or* which is true if at least one disjunct is true but not both are true. The latter operation is also known as "non-equivalence".
- **Condition or implication**: a compound proposition of the form "if ... then ..."; the proposition immediately following "if" is the *antecedent* and the proposition immediately following "then" is the *consequent*. Thus, the antecedent "implies" the consequent.
- **Equivalence**: two propositions are equivalent when they have the same truth value i.e. a biconditional proposition is obtained from two propositions by using the words "if and only if".

The choice of symbols for the connectives is obtained from Boolean Algebra and is as follows:

"~" means "not"
"." means "and"
"$\vee$" means "inclusive or"
"$\equiv$" means "exclusive or"
"$\rightarrow$" means "implies"
"$\leftrightarrow$" or "$\equiv$" means "if and only if"

It is convenient to represent arithmetic variables by small letters x, y, z, etc. and propositions by capital letters from the middle part of

the alphabet "P", "Q", etc. (if it is an arbitrary proposition, it is known as a **propositional variable**). Thus, P, Q, ... may be used to represent

(a) individual propositional variables, e.g. "it will rain on day x"
(b) describe an action or option or yes/no decision
e.g. "product i is manufactured",
(c) level of activity, e.g. "x=1",
(d) linear restrictions i.e. (in)equalities involving LP (or IP) variables, e.g. " $3x + 4y \leq z$",
(e) compound propositions.

For example, if P represents the proposition "It is raining today", Q the proposition "Today is clear", R the proposition "Yesterday was cloudy" and S the proposition "Yesterday was raining" then we have the following compound propositions:

$\sim P$	stands for It is **not** raining today"
$Q \vee P$	stands for "Today is clear **or** today is raining"
$P \leftrightarrow R$	stands for " If, and only if, yesterday was cloudy today it is raining"
$(R \rightarrow P) \vee Q$	stands for "**Either** today is clear **or if** yesterday was cloudy **then** it is raining today"
$\sim R \,.\, Q$	stands for "Yesterday was **not** cloudy **and** today is clear"

To avoid an excess of parentheses in writing compound propositions in symbolic form, we will consider the above connectives in the following conventional order of precidence in descending order:

negation "~"
conjunction "."
disjunction "v"
implication "$\rightarrow$"
equivalence "$\leftrightarrow$"

For example,

$R \,.\, S \rightarrow P$ means $(R \,.\, S) \rightarrow P$; $\sim R \,.\, Q$ means $(\sim R) \,.\, Q$

Given that the individual propositions must be TRUE or FALSE the compound propositions will be true or false depending upon the connectives used. If the truth values of the individual propositions are known, then the truth value of the compound proposition can be determined in a mechanical way by means of **truth tables**. It is

conventional in Boolean Algebra to equate the value *F* with 0 and *T* with 1.

The connectives are defined in terms of the truth values of propositions P and Q in Table 1:

P	Q	~P	P . Q	P v Q	P → Q	P ↔ Q
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

Several other connectives may be similarly defined for combining two or more propositions. Table 2 contains a list of the propositional connectives that we need for our purposes.

NO	Name of connnective	Symbd	Meaning of connective	Other common words
2.1	negation	~P	not P	
2.2	conjunction	P . Q	P and Q	Both P and Q / P but Q
2.3	inclusive disjunction	P v Q	P or Q	Either P or Q / P unless Q / at least one of P or Q
2.4	non-equivalence	P ≡ Q	P exclusive or Q	Exactly one of P or Q is true
2.5	implication	P → Q	if P then Q	P implies Q / P is a sufficient condition for Q
2.6	equivalence	P ↔ Q	P if and only if Q	P iff Q / P is a necessary and sufficient condition for Q
2.7	joint denial	~(P v Q)	P nor Q	Neither P nor Q / None of P Q is true
2.8	non-conjunction	~(P . Q)	P nand Q	

Table 2: Propositional connectives

It is possible to define all propositional connectives in terms of a subset of them. For example, they can all be defined in terms of the set {.,v,~} so that a given expression can be converted into a "**normal form**". Such a subset is known as a **complete set of connectives**. This is accomplished by replacing a certain expression by another "equivalent" expression involving other connectives. Two expressions are said to be "**equivalent**" if and only if their thuth values are the

same, e.g. $P \rightarrow Q$ is the same as $\sim P \vee Q$. Then we have some useful pairs of equivalent expressions, which are sufficient for our purposes, given in Table 3, where P, Q and R are all propositions.

3.1	$\sim\sim P = P$	
3.2	$P \equiv Q = (\sim P . Q) \vee (P . \sim Q)$	Exclusion
3.3	$\sim(P \vee Q) = \sim P . \sim Q$	De Morgan's Law
3.4	$\sim(P . Q) = \sim P \vee \sim Q$	
3.5	$P \rightarrow Q = \sim P \vee Q$	Implication
3.6	$P \leftrightarrow Q = (P \rightarrow Q) . (Q \rightarrow P) = \sim P . \sim Q \vee P . Q$	
3.7	$P \rightarrow Q . R = (P \rightarrow Q) . (P \rightarrow R)$	
3.8	$P \rightarrow Q \vee R = (P \rightarrow Q) \vee (P \rightarrow R)$	
3.9	$P . Q \rightarrow R = (P \rightarrow R) \vee (Q \rightarrow R)$	
3.10	$P \vee Q \rightarrow R = (P \rightarrow R) . (Q \rightarrow R)$	
3.11	$P . (Q \vee R) = (P . Q) \vee (P . R)$	Distributive Law
3.12	$P \vee (Q . R) = (P \vee Q) . (P \vee R)$	
3.13	$P . P \vee Q = P$	

Table 3 . Equivalent logical expressions

Connection with Boolean Algebra

Since all compound propositions may be expressed in terms of the three connectives "~", ".", "v" (i.e. converted into normal form) which correspond to the three Boolean operations (negation, Boolean product and Boolean sum, respectively) and also all axioms of Boolean algebra are satisfied, then it is posiible to show that the algebra of propositions is a Boolean algebra. It is then easy to apply any theorems of the latter and methods of simplification for expressing and manipulating logical relationships. For example, smaller complete set of connectives can be found by writting equivalence in normal forms which are defined below.

By De Morgan's Laws, conjunction may always be expressed in terms of negation and disjunction $P . Q = \sim(\sim P \vee \sim Q)$ - let P and Q be simple propositions. Therefore all conjunctions may be removed leaving an expression entirely in ~ and v, so that ~, v is a complete set of ~~objectives~~ connectives.

Definition: A compound proposition R is said to be in a *conjunctive normal form* if R has the form $R_1 . R_2 \ldots R_n$ where each R_i (i=1, ... n) is a disjunction of individual propositions or the negations of individual

propositions e.g. R = (P v ~Q) . (~P v Q) where P and Q represent simple propositions.

Definition: A compound proposition R is said to be in a disjunctive normal form if R has the form R_1 v R_2 ... R_n where each R_i (i=1, ..., n) is a conjunction of individual propositions or the negations of individual propositions e.g. R = (P . ~Q) v (~P . Q) where P and Q represent simple propositions.

Any expression can be transformed into a normal form. This is accomplished easily by using the equivalent statements given in Table3.

By using useful concepts and convenient methods provided by Propositional Calculus and Boolean Algebra to deal with logical relationships, our purpose is to develop an approach towards modelling logical conditions in terms of 0-1 integer variables and (in)equalities of Mathematical Programming (MP).

Representation of Boolean statements

We wish to transform an arbitrary (Boolean) statement in the propositional calculus into a system of integer linear constraints so that the logical equivalence of the transformed expressions is maintained. The resulting system of constraints clearly must have the same truth table as the original statement i.e. the truth or falsity of the statement is represented by the satisfaction or not of the corresponding linear equations and inequalities.

In order to explain the transformation process and the underlying principles more clearly, we will distinguish the following two cases at this stage:

(i) connecting logical variables

(ii) logically relating linear form constraints

Case (i)

Let P_i denote a Boolean or propositional variable which may take values TRUE (T) or FALSE (F). P_i represents an individual proposition, action, option or decision. Imposing logical conditions in a model requires the introduction of a 0-1 integer variable attached to each type of action (or option) that is envisaged. We adopt the convention of using the Greek letter "δ" for this variable, known as **decision**

variable, and agree that it takes the value of 1 if the action is realised (or the option adopted) and 0 otherwise i.e. define

$\delta_i = 1$ if and only if proposition P_i is TRUE

$\delta_i = 0$ if and only if proposition P_i is FALSE

The logical conditions linking these different actions will be written in the form of linear constraints acting on the associated decision variables.

Case (ii)

A "**logical constraint**" is defined as a logical combination of "simple" constraints and it is formed from the following syntax:

if **antecedent** then **consequent**

where

antecedent must be a binary condition (0-1)

and

consequent can represent either a binary condition or level of activity or a linear form constraint.

In this case, 0-1 **indicator** variables (antecedent) are introduced and linked to some of the continuous variables in the problem to distinguish between certain states (consequent). The truth or falsity of a linear inequality is represented by a 0-1 indicator variable δ_i such that

$\delta_i = 1$ if and only if the ith linear constraint is satisfied

$\delta_i = 1$ if and only if the ith linear constraint is violated

Standard transformations of logical conditions into MP *constraints*

Using the conventional symbols and meaning of propositional connectives for logical operators, given in Table 2, and the set of equivalent statements, given in Table 3, we give below some standard form transformations of compound propositions into linear algebraic forms so that the two expressions are logically equivalent. We distinguish the transformations into two groups, T1 and T2, depending on the meaning of proposition under consideration (case (i) or (ii) above, respectively).

Transformations T1 are applied to propositions P_i (i=1, n) which represent individual propositional variables. We also define δ_i (i=1, n) 0-1 decision variables such that

δ_I = 1 if and only if proposition P_I is true
δ_I = 1 if and only if proposition P_I is false

T1	Statement	is transformed to	Constraint
1.1	$\sim P_1$		$\delta 1 = 0$
1.2	$P_1 \vee P_2$		$\delta_1 + \delta_2 \geq 1$
1.3	$P_1 \veebar P_2$		$\delta 1 + \delta_2 = 1$
1.4	$P_1 \cdot P_2$		$\delta_1 = 1, \delta_2 = 1$
1.5	$\sim(P_1 \vee P_2)$		$\delta 1 = 0, \delta_2 = 0$
1.6	$\sim(P_1 \cdot P_2)$		$\delta_1 + \delta_2 \leq 1$
1.7	$P_1 \rightarrow \sim P_2$		$1 - \delta_2 \geq \delta_1$
1.8	$P_1 \rightarrow P_2$		$\delta_1 - \delta_2 \leq 0$
1.9	$P_1 \leftrightarrow P_2$		$\delta_1 - \delta_2 = 0$
1.10	$P_1 \rightarrow P_2 \cdot P_3$		$\delta_1 \leq \delta_2, \delta_1 \leq \delta_3$
1.11	$P_1 \rightarrow P_2 \vee P_3$		$\delta_1 \leq \delta_2 + \delta_3$
1.12	$P_1 \cdot P_2 \rightarrow P_3$		$\delta_1 + \delta_2 - \delta_3 \leq 1$
1.13	$P_1 \vee P_2 \rightarrow P_3$		$\delta_1 \leq \delta_3, \delta_2 \leq \delta_3$
1.14	$P_1 \cdot (P_2 \vee P_3)$		$\delta_1 = 1, \delta_2 + \delta_3 \geq 1$
1.15	$P_1 \vee (P_2 \cdot P_3)$		$\delta_1 + \delta_2 \geq 1, \delta_1 + \delta_3 \geq 1$

The general forms of the relations 1.3, 1.4, 1.11 and 1.12 may be stated as

1.16	$P_1 \vee P_2 \vee \ldots P_n$		$\delta_1 + \delta_2 \ldots + \delta_n \geq 1$
1.17	$P_1 \veebar P_2 \veebar \ldots P_n$		$\delta_1 + \delta_2 \ldots + \delta_n = 1$
1.18	$P_1 \cdot P_2 \ldots P_k \rightarrow P_{k+1} \vee P_{k+2} \vee \ldots P_n$		$(1-\delta_1) + \ldots + (1-\delta_k) + \delta_{k+1} + \ldots + \delta_n \geq 1$
1.19	"at least k alternatives are TRUE"		$\delta_1 + \delta_2 \ldots + \delta_n \geq k$
1.20	"exactly k alternatives are TRUE"		$\delta_1 + \delta_2 \ldots + \delta_n = k$
1.21	"at most k alternatives are TRUE"		$\delta_1 + \delta_2 \ldots + \delta_n \leq k$

Transformations T2, imposed in the form of implication constraints, are applied to a proposition P which can represent either:

- some level of activity denoted by the LP continuous decision variables *x*, *y* or *z*; *L* and *U* represent the finite lower and upper bounds, respectively, on the activity, or
- a linear form (in)equality denoted by $\Sigma_j a_j x_j \rho b$ where ρ is an (in)equality relation of the form "≤", "≥", or "="; *L* and *U* represent the finite lower and upper bounds, respectively, on the value $\Sigma_j a_j x_j \rho - b$ may take in an optimal solution - how these values are obtained will be discussed later in an example; ε is a small number such that $\varepsilon \leq 1$.

Note that all coefficients and variables in these problems will be integers quantities and ε may be taken as 1.
We define an indicator variable δ taking the value 1 or 0 to show the truth or falsity, respectively, of the proposition P.

T2	Statement	is transformed to	Constraint
2.1	$\delta = 1 \rightarrow x \geq L$		$x \geq L\ \delta$
2.2	$\delta = 0 \rightarrow x \leq U$		$x \leq U\ \delta$
2.3	$\delta = 1 \rightarrow \Sigma_j a_j x_j \leq b$		$\Sigma_j a_j x_j - b \leq U\ (1 - \delta)$
2.4	$\delta = 0 \rightarrow \Sigma_j a_j x_j > b + \varepsilon$		$\Sigma_j a_j x_j - b \geq (L - \varepsilon)\ \delta + \varepsilon$
2.5	$\delta = 1 \rightarrow \Sigma_j a_j x_j \geq b$		$\Sigma_j a_j x_j - b \geq L\ (1 - \delta)$
2.6	$\delta = 0 \rightarrow \Sigma_j a_j x_j < b + \varepsilon$		$\Sigma_j a_j x_j - b \leq (U + \varepsilon)\ \delta - \varepsilon$
2.7	$\delta = 1 \rightarrow \Sigma_j a_j x_j = b$		T2.3 ($\delta = 1 \rightarrow \Sigma_j a_j x_j \leq b$), T2.5 ($\delta = 1 \rightarrow \Sigma_j a_j x_j \geq b$)

A sequence of steps for the transformation of logical conditions

Having represented in the previous sections, compound propositions as (in)equalities, we now wish to model more complicated statements by further inequalities. As a result of the many different, but equivalent, forms any Boolean statement can take there are often different ways of generating the same or equivalent constraints.

One possible way would be to convert the desired Boolean expression into a conjunctive normal form i.e. remove implication, move negation inwards by applying De Morgan's laws and recursively distribute "v" over ".". This results in a conjunction of disjunctive terms, called **clauses**, where negation is only applied to individual propositions. Each clause is then transformed into a linear constraint (of type T1 16), resulting in a system of constraints, derived in this manner, which have to be satisfied invoking the logical "and" operation (an illustration of this method is presented in Example 1).
A more general and systematic procedure for converting a given logical condition imposed on a model into a set of integer linear constraints, is described below in the form of a sequence of transformations steps.

Step 1. Write explicitly the required condition, in words, in the form of a Boolean statement using known logical operators. Let S denote the required condition.

Step 2. Define simple or individual propositions P_i which are sufficient to describe S. Also define compound propositions Q_j in terms of P_i.

Step 3. Rewrite S in symbolic form, as a complicated compound proposition in terms of Q_j and propositional connectives. It may be advisable to use an equivalent form of S or reduce it to a form that represents a conjunction of compound propositions.

Step 4. Break S, if possible, into a set of simpler compound propositions-which are again written in terms of Q_j- (ideally as simple as the standard form propositions) to represent certain states of the problem. It may be necessary to simplify further some of these propositions by writing them into equivalent forms using Table 3.

Step 5. Assign δ_i 0-1 decision variables to the individual propositions P_i to represent the truth or falsity of P_i and define 0-1 indicator variables to relate logically the different states.

Step 6. Transform each compound proposition Q_j into a linear constraint using transformations T1.

Step 7. Combine the states of the problem by linking the indicator variables to the above constraints or to the original (probably continuous) variables of the problem in the form of implication constraints. The latter are then transformed into (in)equalities using transformations T2.

Illustrative Examples

To illustrate the conversion of a logical condition using the above tranformation procedure, we present the following examples.

Example 1.

If 3 or more of products {1 to 5} are made, or less than 4 of products {3 to 6, 8, 9} are made then at least 2 of products {7 to 9} must be made unless none of products {5 to 7} are made.
This example is taken from Williams [1977].

Let the above condition be denoted by S.

Step 2.

Define the individual propositions P_i :"Product i is made" (i=1, ..., 9).
Also define compound propositions
Q_1 : " at least 3 of P_1, P_2, P_3, P_4,P_5 are TRUE "
Q_2 : " at most 3 of P_3, P_4, P_5, P_6, P_8, P_9 are TRUE"
Q_3: " $\sim(P_5 \vee P_6 \vee P_7)$ "
Q_4: " at least 2 of P_7, P_8, P_9 are TRUE"

Step 3.

Write S in symbolic form: $(Q_1 \vee Q_2) . \sim Q_3 \rightarrow Q_4$

Step 4.

Break S into simpler compound propositions to represent states of the problem: $Q_1 \vee Q_2$; $\sim Q_3$; Q_4

Step 5.

Assign decision variables δ_i to the propositions P_i such that
$\delta_i = 1$ if and only if product i is made i.e. proposition P_i is TRUE
$\delta_i = 0$ otherwise
Introduce indicator variables δ and δ' such that
$\delta = 1$ if and only if the proposition $Q_1 \vee Q_2$ is FALSE
$= 0$ if and only if Q_1 or Q_2 is TRUE
$\delta' = 1$ if and only if $\sim Q_3$ is FALSE
$= 0$ if and only if $\sim Q_3$ is TRUE

Step 6.

The T1 transformations are applied to propositions Q_i in order to convert them into linear constraints.
Using T1.19, proposition Q_1 can be represented by the inequality

$$\delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 \geq 3$$

Using T1.21, proposition Q_2 can be represented by the inequality

$$\delta_3 + \delta_4 + \delta_5 + \delta_6 + \delta_8 + \delta_9 \leq 3$$

Using T1.5, proposition Q_3 can be represented by the inequality

$$(1- \delta_5) = 1, (1- \delta_6) = 1, (1- \delta_7) = 1$$

Using T1.19, proposition Q_4 can be represented by the inequality

$$\delta_7 + \delta_8 + \delta_9 \geq 2$$

Step 7.

We relate logically the various states of the problem firslty by imposing the following conditions

$Q_1 \rightarrow \delta = 0$

$Q_2 \rightarrow \delta = 0$

$\sim Q_3 \rightarrow \delta' = 0$

$\delta = 0 \;.\; \delta' = 0 \rightarrow Q_4$

and then converting them into linear constraints by applying the transformations T2. Rewritting the above set of conditions, we have

$\delta = 1 \rightarrow \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 \leq 2$ (1)

$\delta = 1 \rightarrow \delta_3 + \delta_4 + \delta_5 + \delta_6 + \delta_8 + \delta_9 \geq 4$ (2)

$\delta' = 1 \rightarrow \delta_5 = 0 \;.\; \delta_6 = 0 \;.\; \delta_7 = 0$ (3)

$\delta = 0 \;.\; \delta' = 0 \rightarrow \delta_7 + \delta_8 + \delta_9 \geq 2$ (4)

Using T2.3, U may be taken as 3 (= 1+1+1+1+1-2). This gives the following constraint represnetation of (1):

$\delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 + 3\,\delta \leq 5$ (5)

Using T2.5, L may be taken as -4 (= 0+0+0+0+0+0-4). This gives the following constraint representation of (2):

$\delta_3 + \delta_4 + \delta_5 + \delta_6 + \delta_8 + \delta_9 \geq 4\,\delta$ (6)

Using T1.7, we impose condition (3) by the constraints

$1 - \delta_5 \geq \delta'$, $1 - \delta_6 \geq \delta'$, $1 - \delta_7 \geq \delta'$ (7)

To impose condition (4), we may introduce another indicator variable such that

$\delta'' = 1$ if and only if $\delta = 0 \;.\; \delta' = 0$

Condition (4) can then be replaced by the following conditions

$\delta = 0 \;.\; \delta' = 0 \rightarrow \delta'' = 1$ (8)

$\delta'' = 1 \rightarrow \delta_7 + \delta_8 + \delta_9 \geq 2$ (9)

Using T1.12, condition (8) is represented by

$\delta + \delta' + \delta'' \geq 1$ (10)

and using condition T2.5 condition (9) is represented by

$\delta_7 + \delta_8 + \delta_9 \geq 2\,\delta''$ (11)

The complete IP representation of the condition S is given by the constraints (5)-(7), (10)-(11).

Example 2 : Crossword Compilation

This problem has a logical structure; it can be formulated in terms of Boolean Algebra and then converted into an integer programming system of constraints. The objective is to fill in an nxn full puzzle with complete interlocking using words from a given lexicon.

Define the following sets

I the set of rows ($i \in I$)

M the set of columns ($m \in M$)

J the set of letters of the alphabet ($j \in J$)

and let $n = |M| = |I|$. Given also is a lexicon of n-letter words.

This example is taken from Wilson [1989].

To formulate the problem, the following set of logical conditions have to be modeled:

C1. Each cell of the nxn matrix must be occupied by **exactly one** letter of the alphabet.

C2. **If** cell (i, m) is occupied by letter j **then at least (n-1)** cells (i,m'), $m' \neq m$ must be occupied by letters $j' \in J_1$ **and at least** (n-1) cells (i', m), $i' \neq i$ must be occupied by letters $j'' \in J_2$ where

J_1 : set of letters which by virtue of the lexicon could appear in cells (i, m'), $m' \neq m$ given that letter j appears in cell (i, m).

J_2 : set of letters which by virtue of the lexicon could appear in cells (i', m), $i' \neq i$ given that letter j is in cell (i, m).

To convert the above conditions into linear constraints, apply the transformatio procedure.

Step 2.

Define the individual propositions

P_{imj} : " Cell (i, m) is occupied by letter j " for all $i \in I$, $m \in M$ and $j \in J$

and compound propositions

Q_1: " at least (n-1) of $P_{im'j'}$ are TRUE " where $m' \neq m$, $j' \in J_1$

Q_2: " at least (n-1) of $P_{i'mj''}$ are TRUE " where $i' \neq i$, $j'' \in J_2$

Step 3.

Write conditions C_1 and C_2 in symbolic form:

$C_1 : P_{im"A"} \equiv P_{im"B"} \equiv \ldots \equiv P_{im"Z"}$ for all $i \in I$ and $m \in M$

$C_2 : P_{imj} \rightarrow Q_1 \,.\, Q_2$ for all $i \in I$ and $m \in M$

Quite clearly, using 3.7 of Table 3, C_2 is equivalent to the statement:

$(P_{imj} \rightarrow Q_1) \,.\, (P_{imj} \rightarrow Q_2)$.

Step 5.

Assign decision variables δ_{imj} to propositions P_{imj} such that

$\delta_{imj} = 1$ if and only if letter j is placed in cell (i, m)

$= 0$ otherwise

Step 6.

Applying T1 transformations to propositions Q_1 and Q_2, we obtain the following linear constraints.

Using T1.19, Q_1 can be represented by the inequality

$$\Sigma_{m' \neq m,\ j' \in J1}\ \delta_{im'j'} \geq n-1 \qquad \text{for all i, j, m.}$$

and Q_2 can be represented by the inequality

$$\Sigma_{i' \neq i,\ j'' \in J2}\ \delta_{i'mj''} \geq n-1 \qquad \text{for all i, j, m.}$$

Step 7.

Using transformation T1.3, condition C_1 can be represented by the inequality

$$\Sigma_{j \in J}\ \delta_{imj} = 1 \qquad \text{for all i and m} \qquad (1)$$

Using the equivalent form of C_2, we can replace it by the imposing the following pair of conditions:

$$\delta_{imj} = 1 \rightarrow \Sigma_{m' \neq m,\ j' \in J1}\ \delta_{im'j'} \geq n-1 \qquad \text{for all i, j, m} \qquad (2)$$

$$\delta_{imj} = 1 \rightarrow \Sigma_{i' \neq i,\ j'' \in J2}\ \delta_{i'mj''} \geq n-1 \qquad \text{for all i, j, m} \qquad (3)$$

(2) and (3) can be converted into inequalities using transformation T2.5. L may be taken as -(n-1) giving the following representations of (2) and (3), respectively:

$$\Sigma_{m' \neq m,\ j' \in J1}\ \delta_{im'j'} \geq (n-1)\ \delta_{imj} \qquad \text{for all i, j, m} \qquad (4)$$

$$\Sigma_{i' \neq i,\ j'' \in J2}\ \delta_{i'mj''} \geq (n-1)\ \delta_{imj} \qquad \text{for all i, j, m} \qquad (5)$$

Thus, condition C_1 is represented by constraint (4) and condition C_2 is represented by constraints (4)-(5).