An Efficient Logic-to-IP Translation Procedure

Tony Hürlimann

University of Fribourg, Switzerland

Institute of Informatics

Münchenwiler Conference, 2-3 Mai 2001, Departement of Informatics (Fribourg) and Research Group for Theoretical Computer Science and Logic (Bern) , NationalFond Nr. 2000-061454.00

Objectives

- **Logic to IP translation?**
- **A Mixed Example**
- **LPL and the Logical Operators**
- **A Translation Procedure**
- **Applications**

Logic to IP translation?

- **Boolean constraints can be translated to linear constraints containing 0–1 variables.**

Example: $((x \wedge \psi) \vee \zeta) \dot{\vee} \omega$

(1) Replacing $a \dot{\vee} \beta$ **by** $(a \vee \beta) \wedge (\overline{\alpha} \vee \overline{\beta})$ **gives**

$$((x \wedge \psi) \vee \zeta \vee \omega) \wedge (\overline{(\xi \wedge \psi)} \vee \zeta \vee \overline{\omega})$$

(2) Moving the NOT-operator inwards gives

$$((x \wedge \psi) \vee \zeta \vee \omega) \wedge (((\overline{\xi} \vee \overline{\psi}) \wedge \overline{\zeta}) \vee \overline{\omega})$$

(3) Moving the OR-operators inwards gives the following conjunctive normal form

$$(x \vee \psi \vee \zeta) \wedge (\xi \vee \psi \vee \omega) \wedge (\overline{\xi} \vee \overline{\psi} \vee \overline{\omega}) \wedge (\overline{\zeta} \vee \overline{\omega})$$

(4) Translating the clauses into linear constraints, is now straightforward. The resulting 0–1 IP is (where x**,** y**,** z**, and** w **are binary 0–1 variables):**

$$x + \psi + \zeta \geq 1,\ \xi + \psi + \omega \geq 1,$$

$$1 - \xi + 1 - \psi + 1 - \omega \geq 1, 1 - \zeta + 1 - \omega \geq 1$$

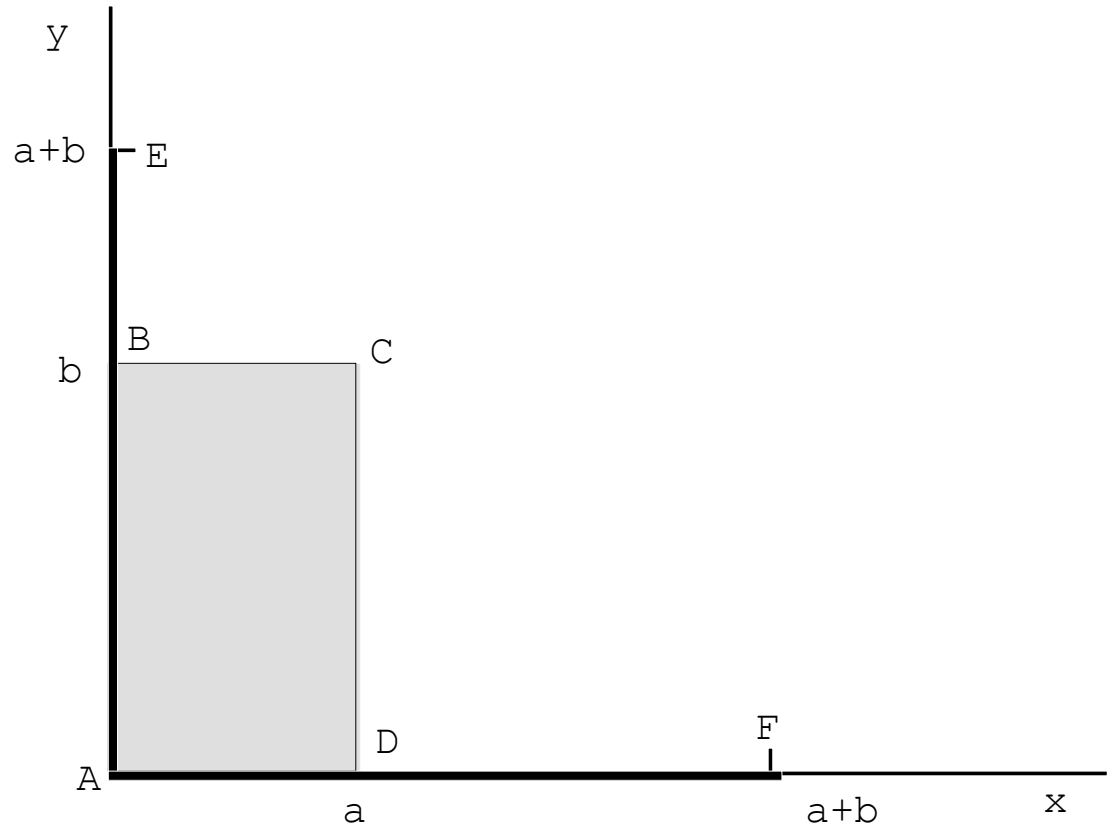
Advantages

- **We can apply powerful MIP-solvers.**
- **In the case where The Boolean constraints are Horn clauses, it can be solved using the Simplex without branching.**
- **We can mix logical and mathematical constraints.**

A Mixed Example

Problem:

A company produces two liquids at unknown quantities x and y. The liquids are stored in containers of capacity a and b.



Formulation using logical and mathematical operators:

$$(x \le \alpha \ \wedge \ \psi \le \beta) \ \vee \ (\xi \le \alpha + \beta \ \wedge \ \psi \le 0) \ \vee \ (\psi \le \alpha + \beta \ \wedge \ \xi \le 0)$$

Translated and formulated as IP model:

$$x - \alpha \le \beta(1 - \delta_1), \quad \psi - \beta \le \alpha(1 - \delta_2)$$

$$\psi \le (\alpha + \beta)\delta_1, \quad \xi \le (\alpha + \beta)\delta_2$$

LPL's logical operators

- **LPL is a modeling language which allows one to mix mathematical and logical operators in constraints.**

The first example:

```
VARIABLE  x, y, z, w BINARY;
CONSTRAINT  r : ((x AND y) OR z) XOR w;
```

The second example:

```
PARAMETER   a; b;
VARIABLE    x,y [0,a+b];
CONSTRAINT C:  (x>a -> y<=0) AND (y>b -> x<=0);
```

- **There are three type of operators:**

  mathematical: addition, multiplication, SUM, etc.

  relational: <, <=, >, >=, =, <>

  logical: ~ (not), and, or, xor, ->, <-, <->, nor, nand,
  indexed: and, or, xor, nor, nand, forall, exist,
  atleast, atmost, exactly

- **Examples:**

```
x AND y          both (x and y) are true.
x -> y                x implies y (if x is true then y  is true).
x NOR y          none of x and y is true.
OR{i} x[i]       at least one out of all x[i] is true.

ATLEAST(k){i} x[i]   at least k out of all x[i] are true.
ATMOST(k){i} x[i]    at most k out of all x[i] are true.
EXACTLY(k){i} x[i]   exactly k out of all x[i] are true.
```

Operator Reduction

Binary Connectors:

$$x \rightarrow \underset{\delta\varepsilon\phi}{\psi} = \leftarrow\xi \vee \psi \qquad x \leftarrow \underset{\delta\varepsilon\phi}{\psi} = \xi \vee \leftarrow\psi$$

$$x \leftrightarrow \underset{\delta\varepsilon\phi}{\psi} = (\xi \vee \leftarrow\psi) \wedge (\leftarrow\xi \vee \psi)$$

$$x \dot{\vee} \underset{\delta\varepsilon\phi}{\psi} = (\xi \vee \psi) \wedge (\leftarrow\xi \vee \leftarrow\psi)$$

$$x \text{ NAND } y \underset{\delta\varepsilon\phi}{=} \leftarrow(\xi \wedge \psi) \qquad x \text{ NOR } y \underset{\delta\varepsilon\phi}{=} \leftarrow(\xi \vee \psi)$$

Indexed Operators:

$$AND_{i=1}^{\nu}\xi_\iota \equiv \forall_{\iota=1}^{\nu}\xi_\iota \equiv AT\Lambda EA\Sigma T(\nu)_{\iota=1}^{\nu}\xi_\iota$$

$$OP_{\iota=1}^{\nu}\xi_\iota \equiv \exists_{\iota=1}^{\nu}\xi_\iota \equiv AT\Lambda EA\Sigma T(1)_{\iota=1}^{\nu}\xi_\iota$$

$$NAND_{i=1}^{\nu}\xi_\iota \equiv \leftarrow(\forall_{\iota=1}^{\nu}\xi_\iota) \equiv \exists_{\iota=1}^{\nu}(\leftarrow\xi_\iota) \equiv AT\Lambda EA\Sigma T(1)_{\iota=1}^{\nu}(\leftarrow\xi_\iota) \equiv ATMO\Sigma T(\nu-1)_{\iota=1}^{\nu}\xi_\iota$$

$$NOP_{\iota=1}^{\nu}\xi_\iota \equiv \leftarrow(\exists_{\iota=1}^{\nu}\xi_\iota) \equiv \forall_{\iota=1}^{\nu}(\leftarrow\xi_\iota) \equiv AT\Lambda EA\Sigma T(\nu)_{\iota=1}^{\nu}(\leftarrow\xi_\iota) \equiv ATMO\Sigma T(0)_{\iota=1}^{\nu}\xi_\iota$$

$$\Xi OP_{\iota=1}^{\nu}\xi_\iota \equiv E\Xi AXT\Lambda(1)_{\iota=1}^{\nu}\xi_\iota$$

$$ATLEAST(k)_{i=1}^{\nu}(\xi_\iota) \equiv ATMO\Sigma T(\nu-\kappa)_{\iota=1}^{\nu}(\leftarrow\xi_\iota)$$

$$ATMO\Sigma T(\kappa)_{\iota=1}^{\nu}(\xi_\iota) \equiv AT\Lambda EA\Sigma T(\nu-\kappa)_{\iota=1}^{\nu}(\leftarrow\xi_\iota)$$

$$\leftarrow(AT\Lambda EA\Sigma T(\kappa)_{\iota=1}^{\nu}(\xi_\iota)) \equiv ATMO\Sigma T(\kappa-1)_{\iota=1}^{\nu}(\xi_\iota)$$

$$\leftarrow(ATMO\Sigma T(\kappa)_{\iota=1}^{\nu}(\xi_\iota)) \equiv AT\Lambda EA\Sigma T(\kappa+1)_{\iota=1}^{\nu}(\xi_\iota)$$

ωηερε $0 \le \kappa \le \nu$

Proposition Definitions

- **Mixing mathematical and logical constraints means to connect both parts in some way.**

- **This is done by Proposition Definitions, that is constraints of one of the three forms:**

$$p \leftrightarrow E \qquad p \rightarrow E \quad \text{or} \quad \leftarrow p \rightarrow E$$

Example: "Let p be true if and only if the quantity x of a product manufactured is strictly positive". This could be formulated as:

$$p \leftrightarrow \xi \neq 0$$

- **In LPL this can be formulated directly as variables. (No distinction is made between propositions and 0–1 variables):**

```
VARIABLE
  p BINARY : E;          (* which means: p → E *)
  q BINRAY ~: E;         (* which means: ←p → E *)
  r BINARY <->: E;       (* which means: p ↔ E *)
```

Now we are ready to define the translation procedure!

The Procedure: Overview

Translation in 8 steps:

Starting point: A constraint in LPL using the defined operators.

Step 1: Several logical operators are eliminated at parse time. This step is applied to *every* expression.

Step 2: Several operators are eliminated after parse-time. This step and the following steps are only applied to model constraints and proposition definitions as defined in Section 3.

Step 3: Some special constructs are replaced.

Step 4: The NOT-operator is pushed inwards within the expression.

Step 5: Complex constraints are broken into parts and new propositions are introduced in one of the form of (27).

Step 6: XOR and <-> are eliminated. The OR-operator is pushed inwards over the AND-operator (to generate a conjunctive normal form.

Step 7: Further special constructs are replaced.

Step 8: All remaining logical operators are eliminated and linear constraints are generated.

The result is a set of linear constraints together with additional 0–1 variables

Each step is done symbolically and intermediate results can be generated by LPL.

Step 1: Reduction at parse-time

The applied rules are:

```
2   X NOR Y          ::=   ~(X OR Y)
3   X NAND Y         ::=   ~(X AND Y)
4   AND{i} X[i]      ::=   FORALL{i} X[i]
5   OR{i} X[i]       ::=   EXIST{i} X[i]
6   XOR{i} X[i]      ::=   EXACTLY(1){i} X[i]
7   NOR{i} X[i]      ::=   ATMOST(0){i} X[i]
8   NAND{i} X[i]     ::=   ATMOST(#i-1){i} X[i]

10  X RelOp1 Y RelOp2 z ::= X RelOp1 Y AND Y RelOp2 z
      (for any RelOp1 or RelOp2 in [=,<>,<,<=,>,>=])
```

(#i means cardinality of set i).

Example:

$x \leq \psi \leq \zeta$ **gives** $(x \leq \psi) \wedge (\psi \leq \zeta)$

Step 2: Reduction after parse-time

The Rules are:

```
1   X -> Y                  ::=   ~X OR Y
1a  X <- Y                  ::=   X OR ~Y
11  FORALL{i} X[i]          ::=   ATLEAST(#i){i} X[i]
12  EXIST{i} X[i]           ::=   ATLEAST(1){i} X[i]
13  ATMOST(k){i} X[i]       ::=   ATLEAST(n-k){i} ~X[i]
14  EXACTLY(0){i} X[i]      ::=   ATMOST(0){i}  X[i]
15  EXACTLY(#i){i} X[i]     ::=   ATLEAST(#i){i} X[i]
```

Step 3: Some special constructs are replaced.

Step 4: Push NOT inwards

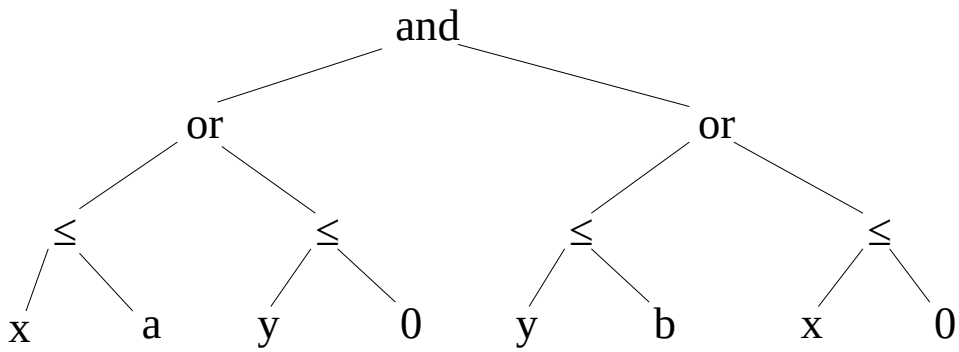
The rules are:

```
20  ~(~X)                        ::= X
21  ~(X AND Y)                   ::= ~X OR ~Y
22  ~(X OR Y)                    ::= ~X AND ~Y
23  ~(X <-> Y)                   ::= X XOR Y
24  ~(X XOR Y)                   ::= X <-> Y
25  ~(X,Y,Z)                     ::= ~X , ~Y , ~Z
26  ~(X [<=,>=,<,>,=,<>] Y)      ::= X [>,<,>=,<=,<>,=] Y
27  ~(ATLEAST(k){i} X[i])        ::= ATLEAST(n-k+1){i} ~X[i]
29  ~(EXACTLY(k){i} X[i])        ::= ATLEAST(n-k+1){i} ~X[i]
                                     OR ATLEAST(k+1){i} X[i]
29a ~IF(c,x,y)                   ::= IF(a,~x,~y)
30  ~c  (c is a const. expr)     ::= (no replacement)
31  ~x  (x is an binary var)     ::= (no replacement)
32  ~z  (z is any other var)     ::= z<=0
33  ~Z  (Z is an var expr)       ::= Z<>0
34  X<>Y                         ::= X<Y OR X>Y
```

Step 5: Split a constraint

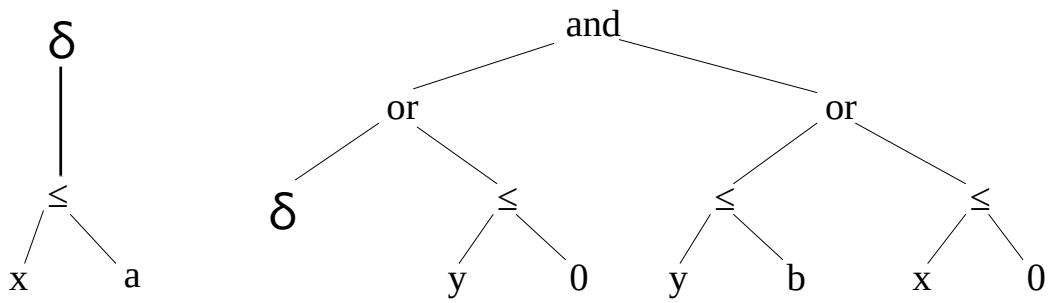
A complicated constraint is split into several ones and additional proposition definitions are introduced.

$$(x \le \alpha \vee \psi \le 0) \ \wedge \ (\psi \le \beta \vee \xi \le 0)$$



is transformed into

$$(\delta \vee \psi \le 0) \ \wedge \ (\psi \le \beta \vee \xi \le 0), \ \delta \rightarrow x \le a$$



This step is the essential part of the whole transformation process.

Step 6: Generate a CNF

Generate a conjunctive normal form: The rules are:

```
40   x XOR y                  ::= (x OR y) AND (~x OR ~y)
41   x <-> y                  ::= (~x OR y) AND (x OR ~y)

42   x OR (y AND z)           ::= (x OR y) AND (x OR z)
43    (x AND y) OR z          ::= (x OR z) AND (y OR z)

45   x OR AND{i} y[i]         ::= AND{i} (x OR y[i])
46   AND{i} y[i] OR x         ::= AND{i} (y[i] OR x
47   OR{i} (x[i] AND y[i]}) ::= OR{i} x[i]
                                    AND OR{i} y[i]}
```

- **Step 4 must be applied again the introduced negations.**

Step 7 again replaces certain specific constructs.

Step 8: Generate linear constraints

Rules for constraints:

```
50  X AND Y AND ...            ::=  X, Y, ...
51  x OR y OR ...              ::=  x + y + ... >= 1
54  ATLEAST(k){i} x[i]         ::=  SUM{i} x[i] >= k
56  EXACTLY(k){i} x[i]         ::=  SUM{i} x[i]  = k
58  ~x  (x is a binary var)    ::=  x<=0  (constraint)
58a ~x  (x is a binary var)    ::=  1-x  (subexpression)
59  x   (x is a binary var)    ::=  x>=1  (expression)
```

Rules for proposition definitions:

```
60  p :  X AND Y AND ...      ::=  p : X , p : Y , ...
61  p :  x OR y OR ...        ::=  x + y + ... >= p
64  p :  ATLEAST(k){i} x[i]   ::=  SUM{i} x[i] >= k*p
66  p :  EXACTLY(k){i} x[i]   ::=  SUM{i} x[i]  = k*p
68  p :  x                    ::=  x >= p
69  p :  ~x                   ::=  1-x >= p

70  p :  ∑ ax <= b       ::=  ∑ ax-b <= U(∑ ax-b)*(1-p)
71  p :  ∑ ax >= b       ::=  ∑ ax-b >= L(∑ ax-b)*(1-p)
72  p :  ∑ ax = b        ::=  p : ∑ax >= b  ,  p : ∑ax <= b

80  p ~:  ∑ ax <= b      ::=  ∑ ax-b <= U(∑ ax-b)*p
81  p ~:  ∑ ax >= b      ::=  ∑ ax-b >= L(∑ ax-b)*p
82  p ~:  ∑ ax = b   ::=  p ~: ∑ax >= b , p ~: ∑ax <= b

83  p <->: X         ::=  p : X  ,  p ~: ~X
```

Applications

A Model: Energy Import [Mitra al. 1994]

Coal, gas and nuclear fuel can be imported in order to satisfy the energy demands of a country.

Three grades of gas and coal (low, medium, high) and one grade of nuclear fuel may be imported. The costs are known.

Furthermore, there are upper and lower limits in the imported quantities from a country. What quantities should be imported to meet the demand, if the import costs have to be minimized?

Three sets must be declared

I = {low, medium, high}	the quality grades of energy
J = {gas, coal}	non-nuclear energy sources
K = {GB FR IC}	countries from which energy is imported

With $i \in I$, $\varphi \in \vartheta$, $\kappa \in K$, the parameters are:

c_{ijk}	(non-nuclear) energy unit costs imported,
l_{ijk}, u_{ijk}	minimum and maximum amount of non-nuclear energy,
nc_k	nuclear energy unit costs,
nl_k, nu_k	minimum and maximum amount of nuclear energy,
e	desired energy amount to import (the demand),
lR_j , uR_j	minimum and maximum percentage of non-nuclear energy if coal and gas are imported (40–50, 20–30%),
lA, uA	minimum and maximum gas take if gas energy only is imported (50–60%).

The numerical variables are:

X_{ijk}	quantity of non-nuclear energy imported,
Y_k	quantity of nuclear energy imported.

The constraint to attain the desired amount of energy is:

$$\sum_{i \in I} \sum_{\varphi \in \vartheta} \sum_{k \in K} X_{ijk} = \varepsilon$$

The objective is to minimize overall energy import costs:

MIN: $$\sum_{i \in I} \sum_{\varphi \in \vartheta} \sum_{\kappa \in K} c_{ijk} \cdot \Xi_{i\varphi\kappa} + \sum_{\kappa \in K} v\chi_{\kappa} \cdot \Psi_{\kappa}$$

Applications

In addition, three further conditions must be fulfilled:

1 **The supply condition: Each country can supply *either* up to three (non-nuclear) low or medium grade fuels *or* nuclear fuel and one high grade fuel.**

2 **The environmental restriction: Environmental regulations require that nuclear fuel can be used only if medium and low grades of gas and coal are excluded.**

3 **The energy mixing condition: If gas is imported then either the amount of gas energy imported must lie between 40–50% and the amount of coal energy must be between 20–30% of the total energy imported or the quantity of gas energy must lie between 50–60% and coal is not imported.**

We define four propositions:

P_{ijk} : $l_{ijk} \leq X_{ijk} \leq u_{ijk}$

N_k : $nl_k, \leq Y_k \leq nu_k$

Q_j : $e \cdot \lambda P_\varphi \leq \sum_{\iota \in I} \sum_{\kappa \in K} \Xi_{\iota\varphi\kappa} \leq \varepsilon \cdot \upsilon P_\varphi$

R : $e \cdot \lambda A \leq \sum_{\iota \in I} \sum_{\kappa \in K} \Xi_{\iota,\gamma\alpha\varsigma,\kappa} \leq \varepsilon \cdot \upsilon A$

The three condition can now be formulated as:

$$\mathrm{ATMOST}(3)_{i\in I,\ \varphi\in\vartheta|\iota<>\eta\iota\gamma\eta} \Pi_{\iota\varphi\kappa} \Xi\mathrm{OP}\ (N_\varphi \mathrm{AN}\Delta\ \Xi\mathrm{OP}_{\varphi}\ \Pi_{\eta\iota\gamma\eta,\ \varphi\kappa})$$

$$\phi o \rho\ \alpha\lambda\lambda\in K$$

$$\mathrm{OR}_k\ N_j \rightarrow \mathrm{NOP}_{\iota,\ \varphi,\kappa|\iota<>\eta\iota\gamma\eta}\ \Pi_{\iota\varphi\kappa}$$

$$\mathrm{OR}_{ik}\ P_{i,gas,k} \rightarrow \mathrm{AN}\Delta_{\varphi}\ \Theta_\varphi \Xi\mathrm{OP}\ P\ \mathrm{AN}\Delta\ \mathrm{NOP}_{\iota\kappa}\ \Pi_{\iota,\chi o\alpha\lambda,\kappa}$$

Applications

Formulation in LPL:

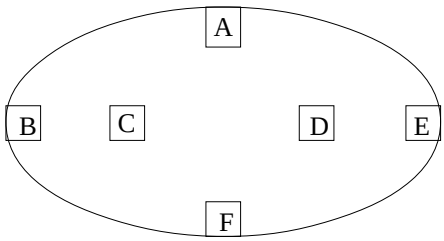
```
....
VARIABLE
 x{i,j,k};
 y{k};
 P{i,j,k} BINARY : l <= x <= u;
 N{k} BINARY : nl <= y <= nu;
 Q{j} BINARY : e*lR <= SUM{i,k} x <= e*uR;
 R BINARY : e*lA <= SUM{i,k} x[i,'gas',k] <= e*uA;

CONSTRAINT
 Cost: SUM{i,j,k} c*x + SUM{k} nc*y;
 ImportReq: SUM{i,j,k} x + SUM{k} y = e;

 SuplCond{k} : ATMOST(3){i,j|i<>'high'} P
               XOR (N AND XOR{j} P['high',j,k]);
 Environ :  OR{k} N  ->  NOR{i,j,k|i<>'high'} P ;
 AltMix  : OR{i,k} P[i,'gas',k] ->  AND{j}Q
           XOR R AND NOR{i,k} P[i,'coal',k];
MINIMIZE Cost;
END
```

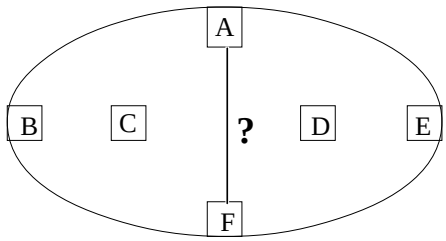
Application

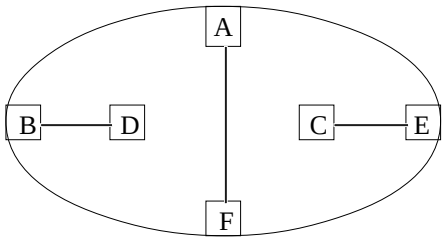
The resulting MIP model is:

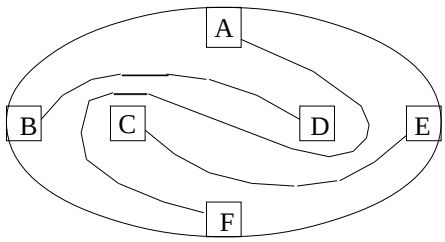
```
VARIABLE x{i,j,k}; y{k};   (*continuous *)
 P{i,j,k}; N{k}; Q{j}; R; X34{k}; X35{k}; (*binary*)

CONSTRAINT
 Pa{i,j,k} : x[i,j,k] <= l[i,j,k]*P[i,j,k];
 Pb{i,j,k} : x[i,j,k] <= u[i,j,k];
 Na{k}     : y[k] <= nl[k]*N[k];
 Nb{k}     : y[k] <= nu[k];
 Qa{j}     : SUM{i,k} x[i,j,k] >= e*MinR[j];
 Qb{j}     : SUM{i,k} x[i,j,k] <= e*MaxR[j];
 Ra        : SUM{i,k} x[i,1,k] >= e*MinA;
 Rb        : SUM{i,k} x[i,1,k] <= e*MaxA;

 SuplCondA{k}: X34[k] + N[k] >= 1;
 SuplCondB{k}: X34[k] + X35[k] >= 1;
 SuplCondC{k}: -X34[k] - N[k] - X35[k] >= -2;
 CX34{k}     : SUM{i,j|i<>'high'}P + 3*X34[k] <= 4;
 CX35a{k}    : SUM{j} P[3,j,k] - X35[k] >= 0;
 CX35b{k}    : SUM{j} P[3,j,k] + X35[k] <= 2;
 Environ{i,j,k,k1 in k|i<>'high'}
             : 1-N[k] + 1-P[i,j,k1] >= 1
 AltMix{j,i,k}  : 1-P[i,1,k] + Q[j] + R >= 1
 CX45  :  SUM{i,k} P[i,1,k] + 1-R + SUM{i,k} P[i,2,k] >= 1
 CX46  :  SUM{j} (1-Q) + 1-R + SUM{i,k} P[i,2,k] >= 1
 CX47{i,k,j,i1 in i,k1 in k}
             : 1-P[i1,1,k1] + Q[j] + 1-P[i,2,k] >= 1
```

Application (Intersection)









			A			
B		C		D		E
			F			

			1			
2		3		2		3
			1			

(Initial coloring)

Application (Intersection)

Model-Formulation:

The sets are:

I	columns, rows of the grid (= {1..7})
K	colour assigned to a square {1,2,3}

The parameters with $i, j \in I, \kappa \in K$ are:

$a_{i,j}$	initial assignment of colours to the square

The variables are:

$x_{k,i,j}$	=1 if square (i,j) has colour k (else 0 (no colour))

The constraints are:

(1) fixed variables: $x_{kij} = 1$ for all $\{i, j \in I, \kappa \in K | \alpha_{\iota\varphi\kappa} = \kappa\}$

(2) exactly one of the (7) neighbours of the initially coloured squares must get the same colour:

$XOR(x_{k,i,j+1}, \xi_{\kappa,\iota-1,\varphi+1}, \xi_{\kappa,\iota-1,\varphi}, \xi_{\kappa,\iota-1,\varphi-1}, \xi_{\kappa,\iota,\varphi-1}, \xi_{\kappa,\iota+1,\varphi-1}, \xi_{\kappa,\iota+1,\varphi}, \xi_{\kappa,\iota+1,\varphi+1})$

$\forall\{k, i, j | a_{i,j} = k\}$:

(3) if a square (other than the initially coloured) is coloured k than exactly two neighbours must be coloured k:

$x_{ijk} \rightarrow E\Xi AXT\Lambda\Psi(2)(\xi_{\kappa,\iota,\varphi+1}, \xi_{\kappa,\iota-1,\varphi+1}, \xi_{\kappa,\iota-1,\varphi}, \xi_{\kappa,\iota-1,\varphi-1},$
$, x_{k,i,j-1}, \xi_{\kappa,\iota+1,\varphi-1}, \xi_{\kappa,\iota+1,\varphi}, \xi_{\kappa,\iota+1,\varphi+1})$ $\forall\{k, i, j | a_{i,j} \neq k\}$:

(4) a square can be coloured with, at most, one colour:

$ATMOST(1)_k\, x_{k,i,j}$ $\forall\{i, j \in I\}$

(5) if two squares touching each other at a corner have the same colour, then the other two squares touching the same corner cannot have the same colour

$x_{k1,i,j} \wedge \xi_{\kappa1,\iota+1,\varphi+1} \wedge \xi_{\kappa2,\iota+1,\varphi} \rightarrow \leftarrow \xi_{\kappa2,\iota,\varphi+1}$

$\forall\{i, j \in I, k1, k2 \in K | ((k1 \neq k2) \wedge (i < |I|) \wedge (j < |I|))\}$

(6) The objective is to colour the minimal number of squares:

MINIMIZE $\sum_{k,i,j} x_{k,i,j}$

Application (Intersection)

Model-Formulation in LPL:

```
MODEL NoIntersectingLines "intersection problem";
SET i ALIAS j := /1:7/;   k=/1:3/;
PARAMETER a{i,j} =
  / 1 4  1, 7 4  1, 4 1  2, 4 5  2, 4 3  3, 4 7  3 /;
VARIABLE x{k,i,j} BINARY;
CONSTRAINT
 I{k,i,j|a[i,j]=k}: x[k,i,j]=1;
 N{k,i,j|a[i,j]=k}:
   XOR(x[k,i,j+1],x[k,i-1,j+1],x[k,i-1,j],x[k,i-1,j-1],
       x[k,i,j-1],x[k,i+1,j-1],x[k,i+1,j],x[k,i+1,j+1]);
 NN{k,i,j|a[i,j]<>k}:
   x[k,i,j] -> EXACTLY(2)(x[k,i,j+1],x[k,i-1,j+1],
                  x[k,i-1,j],x[k,i-1,j-1],x[k,i,j-1],
                  x[k,i+1,j-1],x[k,i+1,j],x[k,i+1,j+1]);
 E{i,j}: ATMOST(1){k} x[k,i,j];
 K{i,j,k1=k,k2=k|k1<>k2 AND i<#i AND j<#j}:
    x[k1,i,j] AND x[k1,i+1,j+1] AND x[k2,i+1,j] ->
    ~x[k2,i,j+1];
MINIMIZE pathL: SUM{k,i,j} x[k,i,j];

WRITE pathL;
WRITE{k,i,j} FORMAT 3:0 : if(x[k,i,j],1,' ');
END
```

Application (Intersection)

Model-Result in LPL:

The model contains:

294 binary variables,

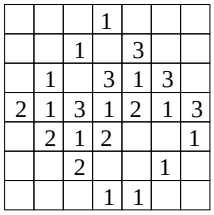
707 constraints, and

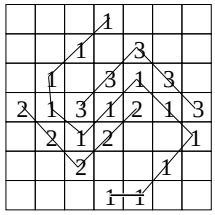
3,642 non-zeroes.

XA found a solution after 10 hours (no optimality prove).

LPL prints:

```
pathL    22.0000
PrintExpr{i,j}
  1 2 3 4 5 6 7
11     1
12    1
13   1    1
14   1  1  1
15    1     1
16         1
17      1 1
21
22
23
24 1     1
25   1  1
26    1
27
31
32       1
33      1  1
34    1     1
35
36
37
```





Conclusion

- **The translation procedure is useful for large mathematical models enriched by a few logical constraints.**

- **It is also useful for symbolic manipulation of mixed expressions.**

However

- **There is no claim here, that every problem, which *could* be formulated in this mixed form, should be translated into a MIP model.**

- **Furthermore, many improvements could be integrated.**

But

- **It is a good starting point and as far as I know the first and unique attempt which integrates symbolic model transformation consistently into a modeling language.**