Conveying Logical Conditions to MIP Solvers through an Algebraic Modeling Language

Robert Fourer, David M. Gay, Victor Zverovich

[4er,dmg,viz]@ampl.com
AMPL Optimization Inc.
www.ampl.com — +1 773-336-AMPL

INFORMS Annual Meeting
San Francisco — 9-12 November 2014
Session MC46, Advances in MIP Modeling Systems
Examples

Zero or range restrictions
Interactions between variables
General logical conditions
Piecewise-linear terms
General Approach

What you want to say
- General description
  - Combination of words and variables

Ways to say it in AMPL
- Linear formulation
  - Using integer variables
- “Not linear” formulation
  - Using integer variables and non-arithmetic operators
  - Not using integer variables

Transformations performed
- In AMPL before invoking the solver
- In the AMPL-solver interface
- In the solver (if at all)
Example 1: Zero or Range

What you want to say

- If $x$ isn’t zero then you want it to be at least $L$
  - where $x \geq 0$ is a variable and $L > 0$ is a constant

Ways to say it in AMPL

- Mixed-integer program
- Discontinuous domain
- Implication
- Disjunction
Example 1

Scheduling

Minimize number of workers needed

- How many workers are assigned to each schedule?
- If a schedule is used at all, at least $L$ workers must be assigned to it

Data: shifts in each schedule; least assignment $L$

```plaintext
set SHIFTS;
param Nsched;
set SCHEDS = 1..Nsched;
set SHIFT_LIST {SCHEDS} within SHIFTS;
param rate {SCHEDS} >= 0;
param required {SHIFTS} >= 0;
param least_assign >= 0;
```
Example 1

Case 1: Mixed-Integer Program

Zero-one variables and inequalities

```plaintext
var Work {SCHEDS} >= 0 integer;
var Use {SCHEDS} >= 0 binary;

minimize Total_Cost:
    sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
    sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use1 {j in SCHEDS}:
    least_assign * Use[j] <= Work[j];

subject to Least_Use2 {j in SCHEDS}:
    Work[j] <= (max {i in SHIFT_LIST[j]} required[i]) * Use[j];
```
Example 1

Case 1 (cont’d)

Solved by CPLEX

ampl: model sched1.mod;
ampl: data sched.dat;
ampl: option solver cplex;
ampl: let least_assign := 17;
ampl: solve;

Reduced MIP has 269 rows, 252 columns, and 1134 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 0 indicators.

Total (root+branch&cut) = 563.38 sec. (138138.56 ticks)

CPLEX 12.6.0.0: optimal integer solution; objective 267
24903192 MIP simplex iterations
3816760 branch-and-bound nodes
Example 1

Case 2: Discontinuous Domain

Union of a point and an interval

```
var Work {j in SCHEDS} integer in {0} union
  interval[least_assign, max {i in SHIFT_LIST[j]} required[i]];

minimize Total_Cost:
  sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
  sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];
```
Example 1

Case 2 (cont’d)

Transformed automatically

- AMPL processor . . .
  - adds auxiliary zero-one variables
  - generates appropriate constraints

Solved by CPLEX as a MIP

Reduced MIP has 269 rows, 252 columns, and 1134 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 0 indicators.
Total (root+branch&cut) = 342.49 sec. (85757.81 ticks)
CPLEX 12.6.0.0: optimal integer solution; objective 267
15087185 MIP simplex iterations
2306392 branch-and-bound nodes
Example 1

Case 2 (cont’d)

Same formulation as case 1

```
ampl: solexpand;

subject to (Work[1]+IUlb):
    Work[1] - 17*(Work[1]+b) >= 0;

subject to (Work[1]+IUub):
    -Work[1] + 100*(Work[1]+b) >= 0;

subject to (Work[2]+IUlb):

subject to (Work[2]+IUub):

. . . . . . .
```
**Example 1**

**Case 3: Implication**

**CPLEX indicator constraint**

```
var Work {SCHEDS} >= 0 integer;
var Use {SCHEDS} >= 0 binary;

minimize Total_Cost: 
  sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
  sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use {j in SCHEDS}:
  Use[j] = 1 ==> Work[j] >= least_assign else Work[j] = 0;
```
Example 1
Case 3 (cont’d)

Logic passed to solver
- AMPL writes “logical” constraints as expression trees
- AMPL-CPLEX driver “walks” the trees
  * detects indicator forms
  * converts to CPLEX library calls

Solved by CPLEX with MIP extensions

Reduced MIP has 143 rows, 252 columns, and 882 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 126 indicators.
Total (root+branch&cut) = 5936.45 sec. (1533625.65 ticks)
CPLEX 12.6.0.0: optimal integer solution; objective 267
250228203 MIP simplex iterations
29437722 branch-and-bound nodes
Example 1

Case 4: Disjunction

Logical constraint using “or” operator

```plaintext
var Work {j in SCHEDS} >= 0 integer;

minimize Total_Cost:
    sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
    sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use {j in SCHEDS}:
    Work[j] = 0 or Work[j] >= least_assign;

subject to Least_Use {j in SCHEDS}:
    Work[j] = 0 or
    least_assign <= Work[j] <= max {i in SHIFT_LIST[j]} required[i];
```
Example 1

Case 4 (cont’d)

Logic passed to solver

- AMPL writes “logical” constraints as expression trees
- AMPL-CPLEX driver “walks” the trees
  * looks for indicator forms

Rejected by CPLEX

```
ampl: option solver cplex;
ampl: solve;
CPLEX 12.5.0.1: logical constraint not indicator constraint.
```
Example 1

Case 4 (cont’d)

Logic passed to solver

- AMPL writes “logical” constraints as expression trees
- AMPL-CPLEX driver “walks” the trees
  * passes constraints as written to C++ “Concert” interface

Accepted and transformed to MIP by CPLEX

```
ampl: option solver ilogcp;
ampl: option ilogcp_options 'optimizer cplex mipdisplay 2';
ampl: solve;
Reduced MIP has 269 rows, 252 columns, and 1134 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 252 indicators.
<BREAK> (ilogcp)
Total (root+branch&cut) = 95272.30 sec. (23592380.69 ticks)
CPLEX 12.6.0.0: aborted, integer solution exists; objective 267
2.89e+009 MIP simplex iterations
351291725 branch-and-bound nodes
```
Example 1

Case 4 (cont’d)

Logic passed to a non-MIP solver

- AMPL writes “logical” constraints as expression trees
- AMPL-LocalSolver driver “walks” the trees

Accepted by LocalSolver

```ampl
ampl: option solver localsolver;
ampl: option localsolver_options 'timelimit 20';
ampl: solve;
LocalSolver 4.5: feasible solution
running time = 20 sec, nb iterations = 8566191, nb moves = 17132449
accepted = 9279 (0.0541604%), improving = 3949 (0.0230498%)
rejected = 17123170 (99.9458%), infeasible = 16367220 (95.5335%)
objective 269
```
Example 1

Transformations Performed

Discontinuous domain
- Transformed to MIP in AMPL
  * Solver’s semicontinuous option missed

Implication
- Passed through to MIP solver

Disjunction
- Transformed to MIP in solver
- Passed through to non-MIP solver
Example 2: Variable Interactions

What you want to say

- When two conditions both hold, there is a cost

Ways to say it in AMPL

- Forms involving $X[i] \times Y[j]$
  - **Case 1:** where $X$, $Y$ are binary (zero-one) variables
  - **Case 2:** where $X$ is binary and $Y$ is any variable
- Forms involving $X[i] = 1 \implies \ldots$ else $\ldots$
  - **Case 3:** where $X$ is binary and $\ldots$ are constraints
Example 2

Case 1

Sample model . . .

\[
\begin{align*}
\text{param } n &> 0; \\
\text{param } c \{1..n\} &> 0; \\
\text{param } d \{1..n\} &> 0; \\
\text{var } X \{1..n\} &\text{ binary;} \\
\text{var } Y \{1..n\} &\text{ binary;} \\
\text{minimize } \text{Obj:} & \quad \left( \sum_{j \text{ in } 1..n} c[j] \times X[j] \right) \times \left( \sum_{j \text{ in } 1..n} d[j] \times Y[j] \right); \\
\text{subject to SumX: } & \sum_{j \text{ in } 1..n} j \times X[j] \geq 2\times n + 3; \\
\text{subject to SumY: } & \sum_{j \text{ in } 1..n} j \times Y[j] \geq 2\times n + 3; \\
\text{subject to SumXY: } & \sum_{j \text{ in } 1..n} (X[j] + Y[j]) = n;
\end{align*}
\]
Example 2

Case 1 (cont’d)

Transformed in stages

- AMPL . . .
  * writes nonlinear expression tree
- AMPL interface . . .
  * multiplies out the product of linear terms
  * sends quadratic coefficient list to solver
- Solver . . .
Example 2

Case 1 (cont’d)

**CPLEX 12.5 transforms to quadratic MIP**

```plaintext
ampl: solve;

Repairing indefinite Q in the objective.

Total (root+branch&cut) = 1264.34 sec.

CPLEX 12.5.0: optimal integer solution within mipgap or absmipgap;
objective 290.1853405

23890588 MIP simplex iterations
14092725 branch-and-bound nodes
```

\[(n = 50)\]
Example 2

Case 1 (cont’d)

**CPLEX 12.6 transforms to linear binary IP**

```ampl
ampl: solve;
MIP Presolve added 5000 rows and 2500 columns.
Reduced MIP has 5003 rows, 2600 columns, and 10200 nonzeros.
Reduced MIP has 2600 binaries, 0 generals, and 0 indicators.

. . . . . .
Total (root+branch&cut) = 6.05 sec.
CPLEX 12.6.0: optimal integer solution; objective 290.1853405
126643 MIP simplex iterations
1926 branch-and-bound nodes
```
Example 2

Case 1a

Sample convex model . . .

```
param n > 0;
param c {1..n} > 0;
var X {1..n} binary;
minimize Obj:
   (sum {j in 1..n} c[j]*X[j])^2;
subject to SumX: sum {j in 1..n} j * X[j] >= 50*n+3;
```
Example 2

Case 1a (cont’d)

**CPLEX 12.5 solves as quadratic MIP**

```
AMPL: solve;

......

Cover cuts applied: 2
Zero-half cuts applied: 1

......

Total (root+branch&cut) = 0.42 sec.

CPLEX 12.5.0: optimal integer solution within mipgap or absmipgap;
objective 29576.27517

286 MIP simplex iterations
102 branch-and-bound nodes
```

(n = 200)
Example 2
Case 1a (cont’d)

CPLEX 12.6 transforms to linear binary IP

ampl: solve;
MIP Presolve added 39800 rows and 19900 columns.
Reduced MIP has 39801 rows, 20100 columns, and 79800 nonzeros.
Reduced MIP has 20100 binaries, 0 generals, and 0 indicators.

....... Cover cuts applied: 8
Zero-half cuts applied: 5218
Gomory fractional cuts applied: 6

....... Total (root+branch&cut) = 2112.63 sec.
CPLEX 12.6.0: optimal integer solution; objective 29576.27517
474330 MIP simplex iterations
294 branch-and-bound nodes
Example 2

Case 1: Transformations Performed

Convex quadratic binary
  ❖ Add $M_j(x_j^2 - x_j)$ to objective as needed to convexify
    ∗ done by CPLEX

Linear binary
  ❖ Define a (binary) variable for each term $x_i y_j$
  ❖ Introduce $O(n^2)$ new binary variables and constraints
    ∗ done by CPLEX

Linear mixed
  ❖ Go to Case 2 . . .
Example 2

Case 2

Alternative quadratic model . . .

```plaintext
param n > 0;
param c {1..n} > 0;
param d {1..n} > 0;
var X {1..n} binary;
var Y {1..n} binary;
var Ysum;

minimize Obj:
    (sum {j in 1..n} c[j]*X[j]) * Ysum;

subj to YsumDefn: Ysum = sum {j in 1..n} d[j]*Y[j];

subject to SumX: sum {j in 1..n} j * X[j] >= 2*n+3;
subject to SumY: sum {j in 1..n} j * Y[j] >= 2*n+3;
subject to SumXY: sum {j in 1..n} (X[j] + Y[j]) = n;
```
Example 2

Case 2 (cont’d)

**CPLEX 12.5 rejects as nonconvex**

```ampl
ampl: solve;
CPLEX 12.5.0: QP Hessian is not positive semi-definite.
```
Example 2

Case 2 (cont’d)

**CPLEX 12.6 transforms to linear MIP**

```ampl
ampl: solve;
MIP Presolve added 100 rows and 50 columns.
Reduced MIP has 104 rows, 151 columns, and 451 nonzeros.
Reduced MIP has 100 binaries, 0 generals, and 0 indicators.
.......  
Total (root+branch&cut) = 0.17 sec.
CPLEX 12.6.0: optimal integer solution; objective 290.1853405
7850 MIP simplex iterations
1667 branch-and-bound nodes
```
Example 2

Case 2: Transformations Performed

Linear mixed

- Introduce a (general) variable $y_{sum} = \sum_{j=1}^{n} d_j y_j$
- Define a (general) variable for each term $x_i y_{sum}$
- Introduce $O(n)$ new variables and constraints
  * done by CPLEX with help from the modeler
Example 2

Case 2: Well-Known Approach

Many refinements and generalizations


Case 3

Model with “indicator” constraints . . .

```
param n > 0;
param c {1..n} > 0;
param d {1..n} > 0;
var X {1..n} binary;
var Y {1..n} binary;
var Z {1..n};

minimize Obj: sum {i in 1..n} Z[i];

subj to ZDefn {i in 1..n}:
    X[i] = 1 ==> Z[i] = c[i] * sum {j in 1..n} d[j]*Y[j]
    else Z[i] = 0;

subject to SumX: sum {j in 1..n} j * X[j] >= 2*n+3;
subject to SumY: sum {j in 1..n} j * Y[j] >= 2*n+3;
subject to SumXY: sum {j in 1..n} (X[j] + Y[j]) = n;
```
Example 2

Case 3 (cont’d)

**CPLEX 12.6 transforms to linear MIP**

```
ampl: solve;
Reduced MIP has 53 rows, 200 columns, and 2800 nonzeros.
Reduced MIP has 100 binaries, 0 generals, and 100 indicators.
........
Total (root+branch&cut) = 5.74 sec.
CPLEX 12.6.0: optimal integer solution within mipgap or absmipgap;
    objective 290.1853405
377548 MIP simplex iterations
95892 branch-and-bound nodes
```
Example 2

Case 3: Transformations Performed

Linear mixed

- Define a (general) variable for each term $x_i \sum_{j=1}^{n} d_j y_j$
- Introduce $O(n)$ new variables
- Introduce $O(n)$ new indicator constraints
  - no actual transformation required
Example 3: General Logic

What you want to express

- Conditions involving disjunction, implication, etc.
- Conditions jointly involving many variables
- Nonstandard numerical relations and functions

Ways to say it in AMPL

- and, or, not, ==>, <=>
- alldiff, numberof
- <, >, floor, round, count, atmost
Example 3

Optimal Arrangement

Maximize adjacency preferences satisfied

```plaintext
param nPeople integer > 0;
set PREFS within {i1 in 1..nPeople, i2 in 1..nPeople: i1 <> i2};
var Sat {PREFS} binary;
var Pos {1..nPeople} integer >= 1, <= nPeople;
maximize NumSat: sum {(i1,i2) in PREFS} Sat[i1,i2];
subject to OnePersonPerPos:
    alldiff {i in 1..nPeople} Pos[i];
subject to SatDefn {(i1,i2) in PREFS}:
    Sat[i1,i2] = 1 <==> Pos[i1]-Pos[i2] = 1 or Pos[i2]-Pos[i1] = 1;
subject to SymmBreaking:
    Pos[1] < Pos[2];
```
Example 3

Case 1

CP solvers handle this directly

```ampl
ampl: model photo.mod;
ampl: data photo11.dat;
ampl: option solver ilogcp;
ampl: solve;
ilogcp 12.6.0: optimizer cp
ilogcp 12.6.0: optimal solution
Solution time = 57.880664s
8837525 choice points, 8432821 fails, objective 12

ampl: option solver gecode;
ampl: solve;
gecode 3.7.3: optimal solution
589206448 nodes, 294603205 fails, objective 12
```

(11 people, 20 preferences)
Case 1 (cont’d)

**CPLEX won’t transform to MIP**

```ampl
ampl: model photo.mod;
ampl: data photo11.dat;
ampl: option solver ilogcp;
ampl: option ilogcp_options 'optimizer cplex';
ampl: solve;
ilogcp 12.6.0: optimizer cplex
Error: unsupported expression: IloAllDiffI (34)
```
Example 3

Case 2

Alternative formulation

```
param nPeople integer > 0;
set PREFS within {i1 in 1..nPeople, i2 in 1..nPeople: i1 <> i2};

var Sat {PREFS} binary;
var Pos {1..nPeople} integer >= 1, <= nPeople;

maximize NumSat: sum {(i1,i2) in PREFS} Sat[i1,i2];

subject to OnePersonPerPos {i in 1..nPeople, j in i+1..nPeople}:
   Pos[i] != Pos[j];

subject to SatDefn {(i1,i2) in PREFS}:
   Sat[i1,i2] = 1 <==> Pos[i1]-Pos[i2] = 1 or Pos[i2]-Pos[i1] = 1;

subject to SymmBreaking:
```
Example 3

Case 2 (cont’d)

**CPLEX transforms to linear IP**

```ampl
ampl: model photo.mod;
ampl: data photo11IP.dat;
ampl: option solver ilogcp;
ampl: option ilogcp_options 'optimizer cplex';
ampl: solve;
ilogcp 12.6.0: optimizer cplex
Reduced MIP has 253 rows, 209 columns, and 614 nonzeros.
Reduced MIP has 144 binaries, 65 generals, and 220 indicators.
........
Total (root+branch&cut) = 3.12 sec.
optimal solution
7822 nodes, 102980 iterations, objective 12
```
Example 3

Transformations Performed

All-different, less-than
  ❖ Passed through to CP solver

Not-equal, less-than-or-equal
  ❖ Transformed to IP in solver
Example 4: **Piecewise-Linear**

*What you want to say*

- Costs are linear but with changing slopes

*How to say it in AMPL*

- `<<breakpoints; slopes>>` variable
Example 4

Transportation with Concave Costs

Supplies, demands, cost parameters

```plaintext
set ORIG;   # origins  
set DEST;   # destinations

param supply {ORIG} >= 0;  # availabilities at origins  
param demand {DEST} >= 0;  # requirements at destinations

param limit1 {i in ORIG, j in DEST} >= 0;   # breakpoints
param limit2 {i in ORIG, j in DEST} >= limit1[i,j];

param rate1 {i in ORIG, j in DEST} >= 0;   # slopes
param rate2 {i in ORIG, j in DEST} <= rate1[i,j];
param rate3 {i in ORIG, j in DEST} <= rate2[i,j];
```
Example 4
Transportation with Concave Costs

Piecewise-linear objective

\[
\text{var} \quad \text{Trans} \{\text{ORIG,DEST}\} \geq 0;
\]

\[
\text{minimize} \quad \text{Total\_Cost}: \\
\quad \text{sum} \{i \in \text{ORIG}, j \in \text{DEST}\} \\
\quad \quad <<\text{limit1}[i,j], \text{limit2}[i,j]; \\
\quad \quad \text{rate1}[i,j], \text{rate2}[i,j], \text{rate3}[i,j]>> \text{Trans}[i,j];
\]

\[
\text{subject to \ Supply} \{i \in \text{ORIG}\}: \\
\quad \text{sum} \{j \in \text{DEST}\} \text{Trans}[i,j] = \text{supply}[i];
\]

\[
\text{subject to \ Demand} \{j \in \text{DEST}\}: \\
\quad \text{sum} \{i \in \text{ORIG}\} \text{Trans}[i,j] = \text{demand}[j];
\]
**Example 4**

**Case 1**

**AMPL transforms . . .**

```
ampl: option solver cplex;
ampl: solve;
Substitution eliminates 18 variables.|
21 piecewise-linear terms replaced by 87 variables and 87 constraints.
Adjusted problem:
90 variables:
    41 binary variables
    49 linear variables
79 constraints, all linear; 251 nonzeros
    33 equality constraints
    46 inequality constraints
1 linear objective; 49 nonzeros.
```

(3 origins, 7 destinations)
Example 4

Case 1 (cont’d)

AMPL-CPLEX interface transforms . . .

Reduced MIP has 15 rows, 49 columns, and 108 nonzeros. Reduced MIP has 0 binaries, 0 generals, 18 SOSs, and 0 indicators.

........

Total (root+branch&cut) = 0.13 sec.

CPLEX 12.6.0: optimal integer solution; objective 256100

501 MIP simplex iterations

388 branch-and-bound nodes
Example 4

Case 1 (cont’d)

. . . with SOS type 2 markers in output file

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>87</td>
<td>sos</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>18</td>
</tr>
<tr>
<td>S1</td>
<td>64</td>
<td>sos</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>35</td>
</tr>
<tr>
<td>S4</td>
<td>46</td>
<td>sosref</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-501</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>751</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-501</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>500</td>
</tr>
</tbody>
</table>
Case 2

AMPL sends untransformed representation

```
ampl: option pl_linearize 0;
ampl: option solver ilogcp;
ampl: option ilogcp_options 'optimizer cplex';
ampl: solve;
21 variables:
    18 nonlinear variables
    3 linear variables
10 constraints, all linear; 42 nonzeros
    10 equality constraints
1 nonlinear objective; 21 nonzeros.
```
**Example 4**

**Case 2 (cont’d)**

**CPLEX handles piecewise-linear terms directly**

Reduced MIP has 58 rows, 79 columns, and 187 nonzeros.
Reduced MIP has 13 binaries, 0 generals, 5 SOSs, and 26 indicators.

\[ \ldots \ldots \]
Total (root+branch&cut) = 0.03 sec.
Ilogcp 12.6.0: optimal solution
0 nodes, 35 iterations, objective 256100
Example 4
Transformations Performed

$pl\_linearize = 1$
- AMPL converts to general MIP formulation
- Interface converts to SOS2 formulation
- Solver’s *built-in piecewise-linear* features missed

$pl\_linearize = 0$
- AMPL conveys as nonlinear expression tree
- Interface passes piecewise-linearities to solver
Who Should Transform It?

*The AMPL user*

*The AMPL processor*

*The AMPL-solver interface*

*The solver*
The AMPL User

Advantages
- Can exploit special knowledge of the problem
- Doesn’t have to be programmed

Disadvantages
- May not know the best way to transform
- May have better ways to use the time
- Can make mistakes
The AMPL Processor

Advantages

- Makes the same transformation available to all solvers
- Has a high-level view of the problem

Disadvantages

- Is a very complicated program
- Can’t take advantage of special solver features
The AMPL-Solver Interface

Advantages

- Works on simplified problem instances
- Can use same ideas for many solvers, \textit{but also}
- Can tailor transformation to solver features

Disadvantages

- Creates an extra layer of complication
The Solver

**Advantages**

- Ought to know what’s best for it
- Can integrate transformation with other activities

**Disadvantages**

- May not incorporate best practices
- Is complicated enough already