Constraint Modelling and the Pursuit of the Holy Grail (2017/18)

Ian Miguel

University of St Andrews

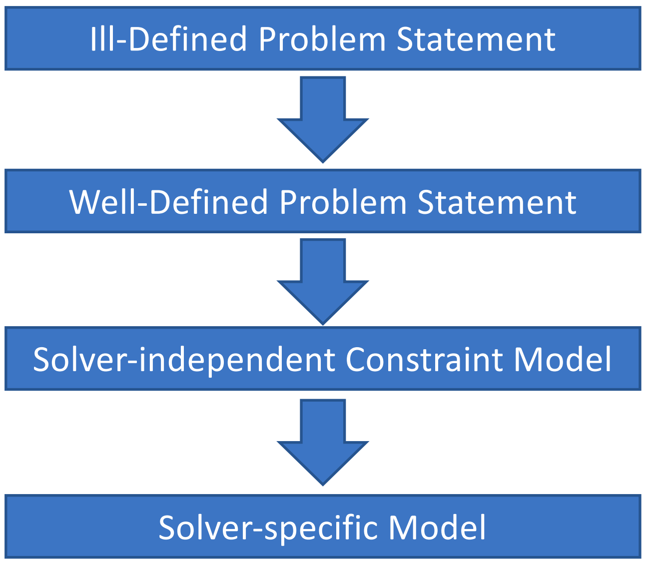
ijm@st-andrews.ac.uk

Constraint Modelling and the Quest

- Constraint Modelling: The preparation of the description of a problem of interest suitable for input to a particular constraint solver.
  - Constraint Modelling/Solving in the broad sense here: a formalism for describing and solving decision-making and optimisation problems.
  - Decision variables and restrictions on compatible assignments.
- **Crucial** to the quest for the Holy Grail.
  - Typically, there are many possible models for a given problem.
    - E.g. Nadel's 1990 study of the n-queens problem, many of the studies by Barbara Smith…
  - The right choice of model can make the difference between a solution swiftly obtained and being unable to solve the problem in a practical amount of time.

Modelling

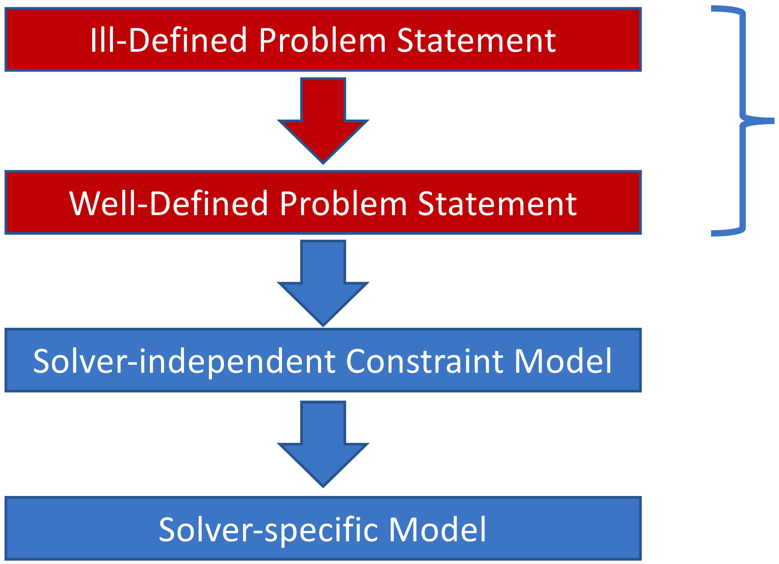
- The preparation of the description of a problem of interest suitable for input to a particular solver.



- A multi-stage process, starting from a possibly ill-defined idea of what the problem is.
- It is possible to engage with this process starting at any level.
- But the higher up a non-expert can begin and still obtain a solution to her problem efficiently and automatically, the closer we are to the grail.

Modelling

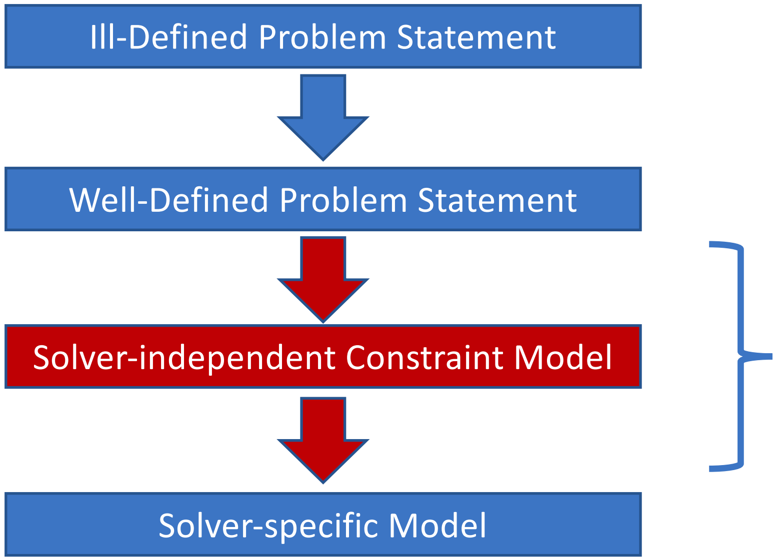
- The preparation of the description of a problem of interest suitable for input to a particular solver.



- We will set aside the process of knowledge elicitation/constraint acquisition.

Modelling

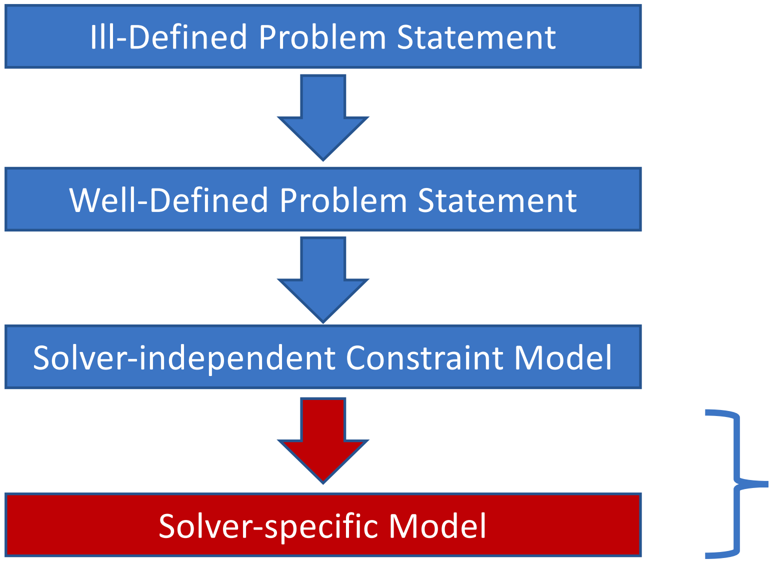
- The preparation of the description of a problem of interest suitable for input to a particular solver.



- MiniZinc, Essence Prime, OPL, …
- Primitive variables (Booleans, integers, perhaps sets) and collections.
- Quantification/comprehension.
- Arithmetic, logical operators, global constraints.
- Not focused on a particular solver.

Modelling

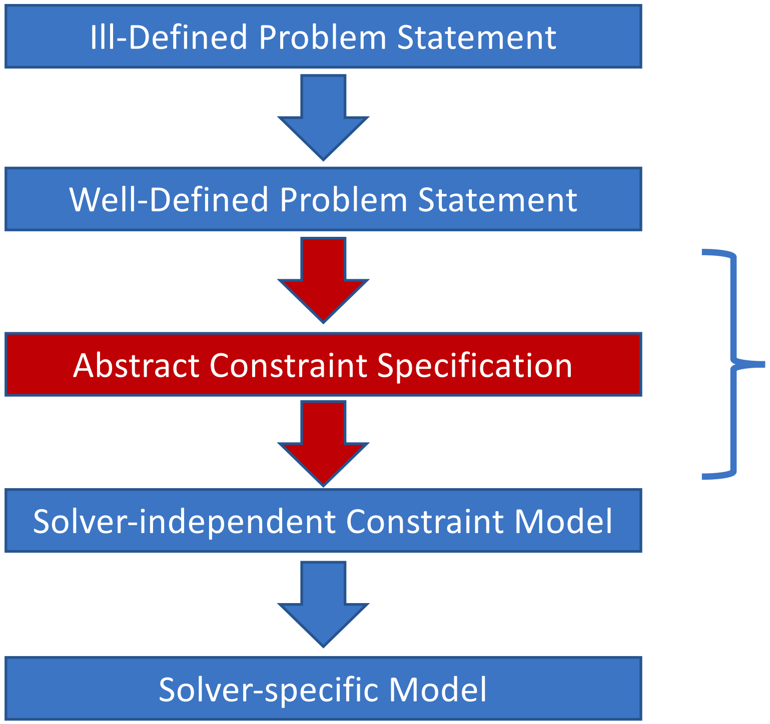
- The preparation of the description of a problem of interest suitable for input to a particular solver.



- Library embedded in a host language, e.g. Choco, Gecode.
- Solver with own input language, e.g. Minion.
- Beyond constraint solvers:
  - SAT, MIP, local search, …

Obtaining High Quality Models

Obtaining High Quality Models: Abstraction



- How do we obtain high quality models from a well-defined problem statement?
- One approach is to insert an extra step to allow the user to write abstract constraint specifications.
  - E.g. Zinc, Essence.
  - Abstraction: capturing the problem **without committing** to modelling decisions.
  - By providing types that match frequently occurring combinatorial structure, such as (multi)set, function, relation, partition.

Example: The Social Golfers Problem

- Problem Statement:
  - In a golf club there are a number of golfers who wish to play together in **g** groups of size **s**.
  - Find a schedule of play for **w** weeks such that no pair of golfers play together more than once.
- In fact, this is an example of a combinatorial design, well studied in mathematics.

Example: Social Golfers Essence Specification

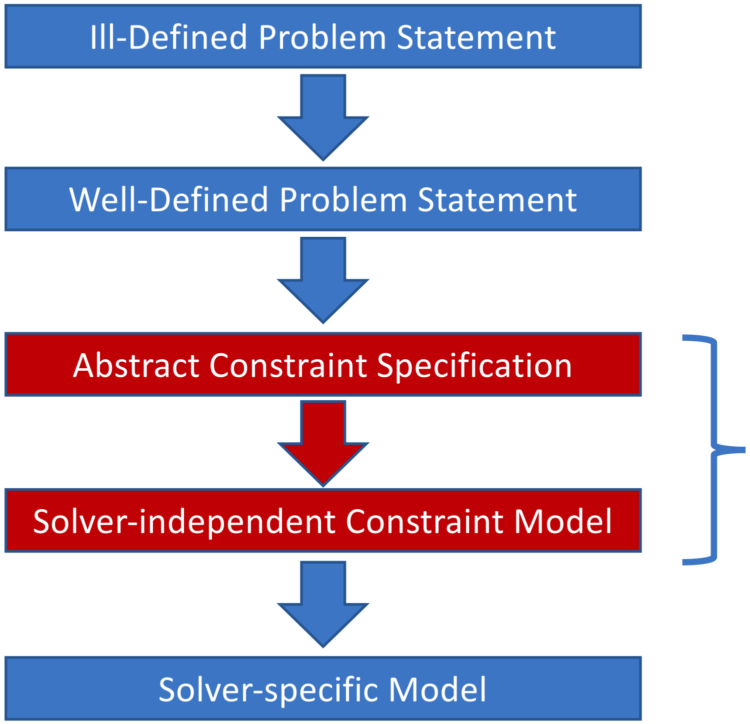
```
given w, g, s : int(1..)
letting Golfers be new type of size g * s

find sched : set (size w) of partition
  (regular, numParts g, partSize s) from Golfers

such that
forAll g1, g2 : Golfers, g1 < g2 .
  (sum week in sched .
   toInt(together({g1, g2}, week))) <= 1
```

- We have not committed to a model of the schedule (a set of partitions).
- We can produce different models of this specification automatically and transparently to the user.
- See Conjure/Savile Row tutorial by Akgun & Nightingale at CP'18.

Obtaining High Quality Models: Abstraction



- The process of refining a solver-independent model from an abstract specification can be automated.
  - Encodes modelling expertise.
  - How to model a function, partition etc.
  - Model selection heuristic/from training instances.
- Further advantages:
  - detect and break **symmetry** as it is introduced.
  - Streamlining.
  - Neighbourhood Generation for Local Search.

Example: Automated Symmetry Breaking

```
given w, g, s : int(1..)
letting Golfers be new type of size g * s

find sched : set (size w) of partition
  (regular, numParts g, partSize s) from Golfers

such that …
```

1	2	3	4
P1	P2	P3	P4

Symmetrically equivalent to:

1	2	3	4
P2	P4	P3	P1

- Consider modelling the outer set as a matrix with `w` entries.
  - Each represents a partition of golfers.
  - A set has no indices; a matrix does.
  - This modelling step **introduces symmetry**.
  - By recognizing this we can immediately add constraints to remove it.
    - No “detection” necessary.
    - Reasoning at the problem class level.
- See Conjure/Savile Row tutorial by Akgun & Nightingale at CP18.
- Symmetry can also be detected/broken in constraint models.
  - See Codish et al. ‘18, Mears et al. ‘09.

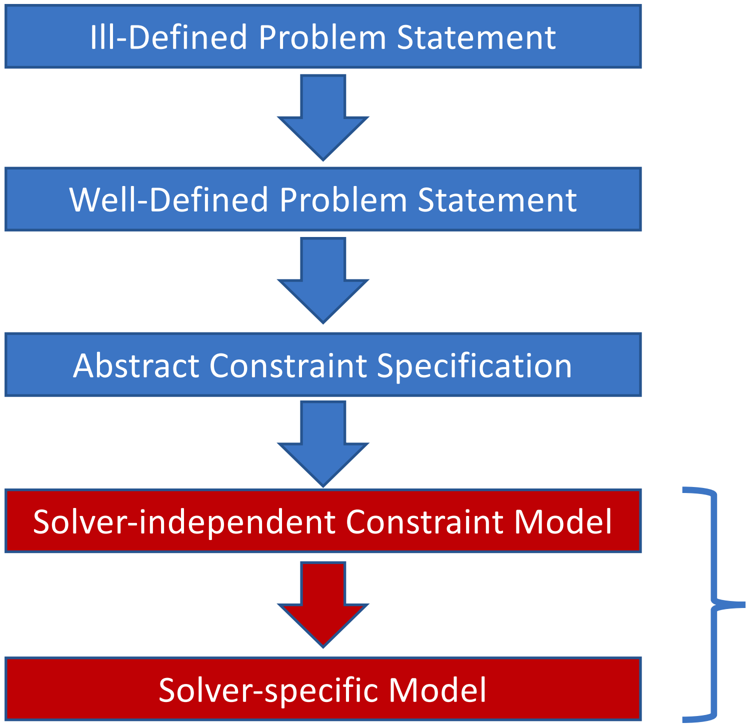
Example: Automatic Streamliner Generation

- Streamlining: addition of "uninferred" constraints designed to reduce significantly the search space while permitting at least one solution.
  - Effective streamliners found by hand – see Le Bras et al. CP'14, IJCAI'13.
- We can exploit the structure apparent in an abstract specification to generate powerful candidate streamliners automatically.
  - E.g. if we know we are looking for a relation we can try looking for relation that is symmetric, transitive, reflexive, etc.
  - E.g. 2: if we know we are looking for a function we can require it to be monotonically in/decreasing, or constrain the function domain or range.
  - Performing the same steps directly on a constraint model would require us to recognise the particular model of relation/function first.
- See Spracklen et al. CP'18, Wetter et al. CP'15.

Example: Neighbourhood Generation for Local Search

- Effective local search relies on high quality neighbourhoods to guide modifications to an active solution.
- Such neighbourhoods can be derived directly from the constraints in a model.
  - E.g. Bjordal et al. for MiniZinc, Constraints 2015.
- Powerful neighbourhoods can also be derived from the structure in an abstract specification.
  - E.g. move golfers between parts of a partition in the Social Golfers example.
  - While always maintaining the partition structure.
  - See Akgun et al. IJCAI’18, Attieh et al. ModRef’18.

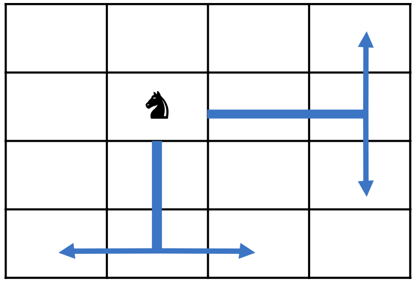
Obtaining High Quality Models: Reformulation



- Another approach is to transform or reformulate an initial model to improve its performance.
- Rewrite a set of constraints to obtain better propagation.
  - E.g. Clique of disequalities -> alldifferent.
  - See also: Leo et al. CP’13, Bessiere et al. IJCAI’07.
  - Implied constraints: Arafailova et al. This workshop.

Example: Automatic Tabulation.

- Tabulation: aggregate a set of constraint expressions into a single table constraint.
  - To exploit efficient table constraint propagators that enforce generalised arc consistency.
  - Typically a stronger level of inference than is achieved for a logically equivalent collection of separate constraints.

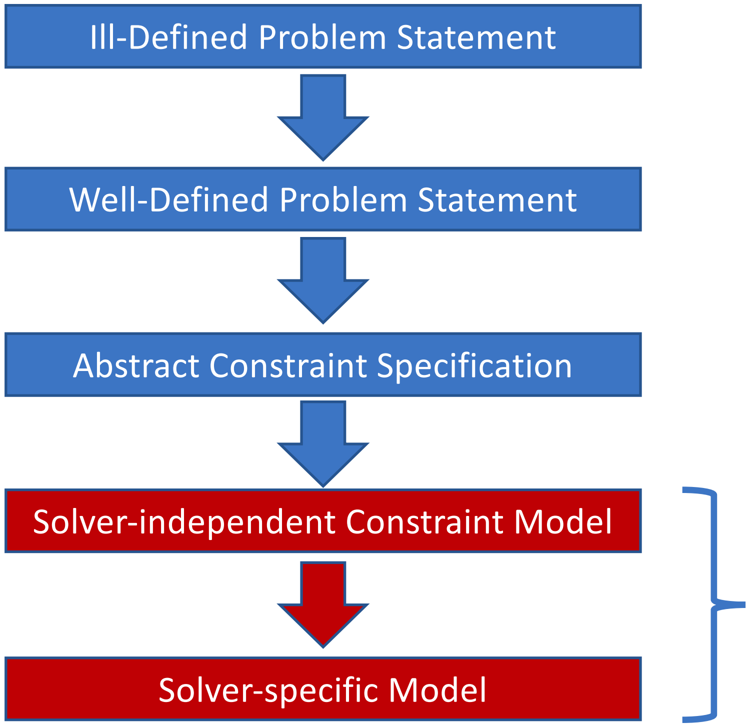


- Knight's Tour (linking position variables x and y):

$$(|x\%n - y\%n| = 1 \text{ and } |x/n - y/n| = 2) \text{ OR }$$
$$(|x\%n - y\%n| = 2 \text{ and } |x/n - y/n| = 1)$$

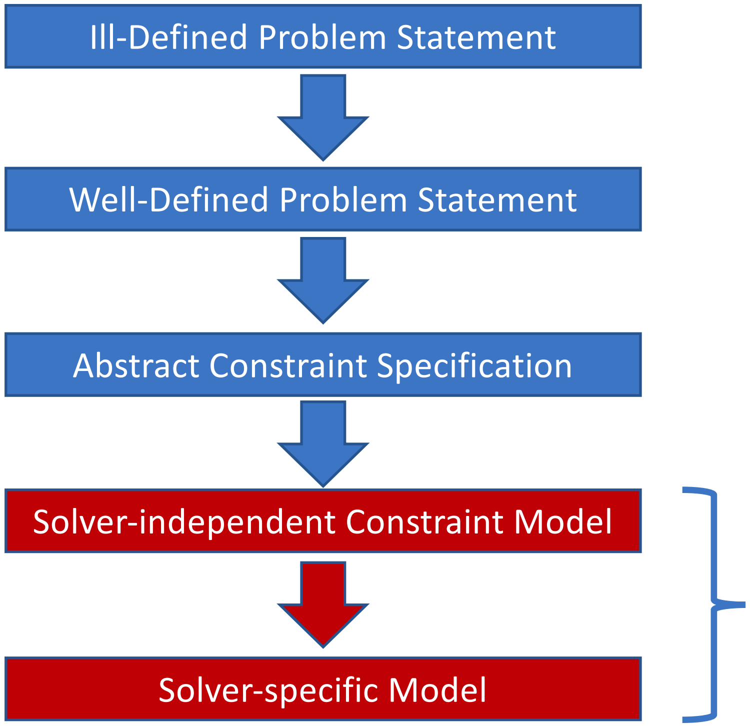
  - CP solvers generally treat occurrences of x as independent - poor propagation
- Identify promising sets of expressions to tabulate **heuristically**:
  - E.g. expressions with duplicate variables, or otherwise likely to propagate weakly.
- See Nightingale et al. CP'18, Dekker et al. Constraints '17, "strong" annotation in CPLEX Optimizer.

Obtaining High Quality Models: Reformulation



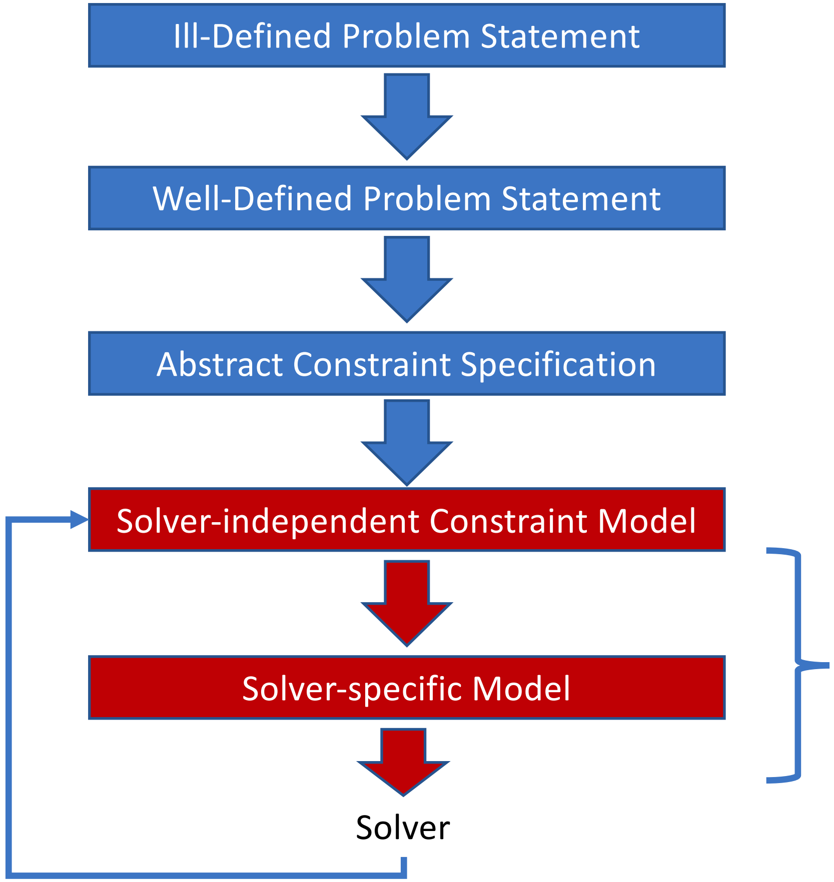
- Another approach is to transform or **reformulate** an initial model to improve its performance.
- Common Subexpression Elimination:
  - Simplest form: avoid flattening two identical subexpressions x + y to separate auxiliary variables.
  - More sophisticated: reformulate to reveal common subexpressions, associative-commutative matching.
- See Nightingale et al. AIJ '17, Conjure/Savile Row and MiniZinc tutorials CP'18.

Obtaining High Quality Models: Reformulation



- Another approach is to transform or reformulate an initial model to improve its performance.
- Dominance-breaking constraints.
- Generalises symmetry breaking.
  - Dominance relations describe pairs of assignments where one is at least as good as the other with respect to satisfiability or the objective function.
  - E.g. two items with same weight but differing value in a knapsack problem.
- See Guns et al. ModRef ’18, Mears & de la Banda IJCAI ‘15, Chu & Stuckey Constraints ‘15.

Obtaining High Quality Models: Feedback Loops



- Use a solver to process an initial model. Information gained used to strengthen the model.
- E.g. Savile Row uses Minion to enforce Singleton Arc Consistency
  - Domain reductions can trigger, e.g., common subexpression elimination.
  - See Nightingale et al. AIJ 2017.
- See also:
  - Leo & Tack IJCAI'15.
  - Zeighami et al. CP'18.
  - Simonis' tutorial ModRef'18.

Conclusions

- Automated modelling is central to the pursuit of the holy grail.
- Modelling is a conduit to other related disciplines:
  - SAT, SMT, MIP, ASP, local search, …
  - This brings new challenges in finding effective encodings to these formalisms.
  - But the flexibility this affords brings us closer to the grail.
- Modelling & Solving: a symbiosis.
- Can exploit Machine Learning in a variety of contexts:
  - Model, streamliner, reformulation selection…