Chapter 1: Constraints

CONSTRAINT PROGRAMMING COURSE

M1 IDO

What Is a Constraint?

A constraint is a logical formula relating different variables, built upon a fixed language. Each variable has a domain (instantiation set) from which it takes its values. A constraint denotes a set of solutions (the satisfying assignments) for a fixed logical interpretation.

Example: The constraint X + Y = 2 over the domain $\mathbb{N}$ has three solutions: {X=0, Y=2}, {X=1, Y=1}, and {X=2, Y=0}. A constraint restricts the values that its variables can simultaneously take.

Application Domains

- Combinatorial problems (puzzles, games, ...)
- Planning problems (e.g., timetabling)
- Scheduling problems (e.g., task-machine assignment)
- Packing and placement problems (e.g., fitting objects into a bounded space)

Diagnosis and verification

All these problems may additionally involve an objective function f to be optimised (minimising space, time, number of machines, etc.).

How It Works

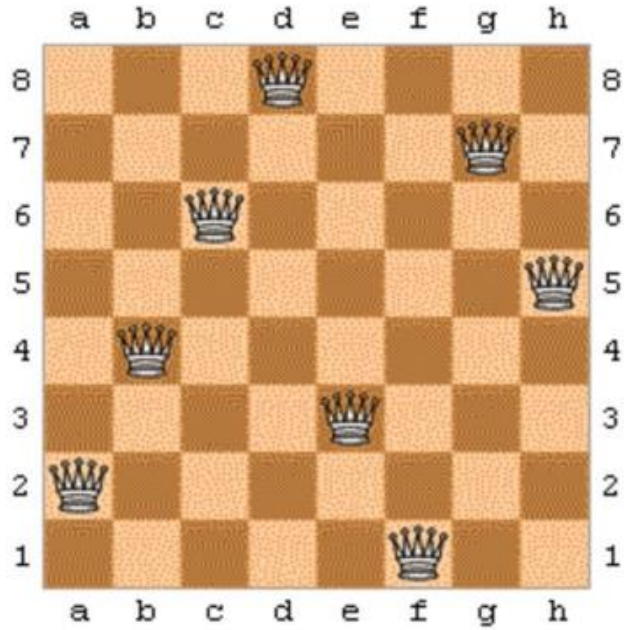
- The programmer uses constraints (logical formulae) to model the problem.
- The Prolog interpreter can call constraint solvers to decide whether a constraint has a solution.
- Different constraint solvers exist for different domains ("constraint systems"): arithmetic, finite domains, ...
- Difficulty: in general, these solvers are not complete.

Motivating Example – The N-Queens Problem

The goal is to place n queens on an n × n chessboard so that no queen threatens another.

Two queens threaten each other if they share a row, column, or diagonal.

A standard chessboard has 8 rows and 8 columns.



Syntax of a Constraint

Constraints are defined based on first-order logic, using a signature (F, P) where F is a set of functions (and constants) symbols and P is a set of predicate symbols.

Simple constraint: a predicate applied to terms.

Constraint: a conjunction of simple constraints $C = c_1 \wedge c_2 \wedge \ldots \wedge c_k$.

- **Example:** $X \geq 42 \wedge X = Y + 2$

Special constraints: true (always satisfied) and false (never satisfied).

A constraint is a formula of first-order logic, normally without negation, disjunction, or explicit quantifiers.

System of constraints

A constraint domain D is a set such as the integers or the reals.

A system of constraints is given by (F, P, D) and an interpretation of the symbols in F and P.

Example – linear numeric constraint system:

F = {+, −, 0, 1, ...}

P = {=, ≤, <, ≥, >, ≠}

D = $\mathbb{N}$

Interpretations: standard arithmetic.

Other constraint systems: real numbers, Herbrand constraints, finite domain constraints, ...

Semantics of Constraints

Assignment: a partial function mapping variables to domain values.

An assignment θ violates a simple constraint if it renders it false; it violates a constraint if it violates at least one simple constraint.

An assignment is consistent for a constraint if it does not violate it.

Solution: a total and consistent assignment.

Example: $X \geq 42 \wedge X = Y + 2$ has a solution $\theta = \{X \leftarrow 43, Y \leftarrow 41\}$.

This is exactly the semantics of first-order logic.

Satisfiability and Equivalence

A constraint is satisfiable if it has at least one solution.

For $C = c_1 \wedge ... \wedge c_k$, we define the set $\Omega(C) = \{c_1, ..., c_k\}$.

A constraint c_1 implies c_2 if every solution of c_1 is also a solution of c_2.

Two constraints are equivalent if they have the same solution set.

Extensional Definition of Constraints (Enumeration)

A k-ary constraint on X_{i1}, ..., X_{ik} defined in extension enumerates all permissible k-tuples from $D(X_{i1}) \times ... \times D(X_{ik})$.

- $(x, y) \in \{(0,0), (1,0), (2,2)\}$
- $(x, y, z) \in \{(1,2,3), (2,3,4)\}$

Intentional Definition (Comprehension)

A k-ary constraint on $X_1, ..., X_k$ defined in comprehension is given mathematically, e.g. as an equation or a linear inequality.

- Logical: $x = 1 \vee y = 3,\ x = 2 \Rightarrow y = 4$
- Arithmetic: $x > y,\ z = 2x + 3y - 5$
- Global (complex): all-different($x_1, ..., x_n$)

Arity of Constraints

- Unary: involves one variable only, e.g. x = 4.
- Binary: involves two variables, e.g. x + y = 9.
- n-ary: involves *n* variables. The term is used when the number of variables is not fixed in advance (e.g. all-different).

Operations on Constraints

- **Solution finding**
  - Variables: $X \in \{1,2,3\}$, $Y \in \{1,2,3\}$
  - Constraint: $X \neq Y$
  - → Solution: X=1, Y=2 (among others)
- **Satisfiability testing**
  - $X \in \{1,2\}$, $Y \in \{1,2\}$
  - Constraints: $X < Y$ AND $Y < X$
  - → UNSATISFIABLE (detected immediately)
- **Implication checking**
  - Store: $X \in \{4,5,6\}$
  - Query: Does $X > 3$ hold in all solutions?
  - → YES, it is implied ($\min(X) = 4 > 3$)
- **Simplification**
  - $X \in \{1,2,3\}$, $Y \in \{1,2,3\}$,
  - Constraint: $X < Y$
  - → After AC: $X \in \{1,2\}$, $Y \in \{2,3\}$ (X=3 has no Y>3; Y=1 has no X<1)
- **Projection**
  - Constraints over X, Y, Z: $X + Y = Z$, $Z \in \{3,4\}$
  - Project onto {X, Y}:
  - → $X + Y \in \{3,4\}$ (Z is eliminated)
- **Optimisation**
  - Minimise: $X + Y$
  - Subject to: $X \in \{1..5\}$, $Y \in \{1..5\}$, $X \neq Y$
  - → Optimal: X=1, Y=2 (or X=2,Y=1), cost = 3