Constraint Programming (CP) and IBM CP Optimizer

Speaker: Xinbo Wang

UCDAVIS

Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Mathematical Programming (MP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer

Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Mathematical Programming (MP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer

What is constraint programming?

- **CP is an optimization technology which is complementary to Mathematical programming (e.g. ILP) taking a different approach to optimization, but sharing similarities.**

- **It is a relatively new technology developed in the computer science and artificial intelligence communities.**

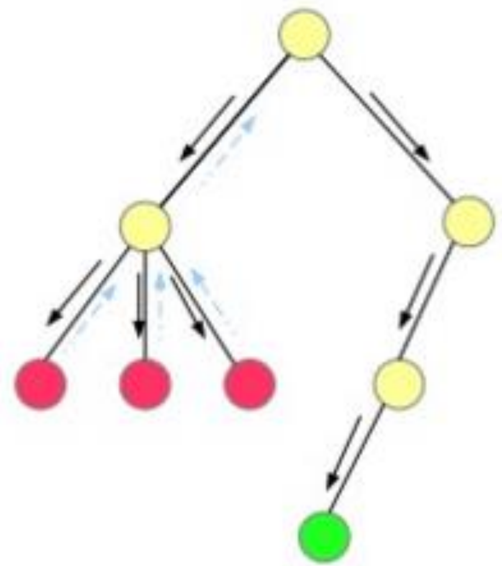
- **It has found an important role in scheduling and highly combinational problems (for ours).**

Applications

- Job shop scheduling
- Assembly line smoothing and balancing
- Cellular frequency assignment
- Airline crew rostering Nurse scheduling
- Shift planning
- Maintenance planning
- and scheduling
- Airport gate allocation and stand planning
- Production scheduling
- Transport scheduling
- Warehouse management
- Course timetabling

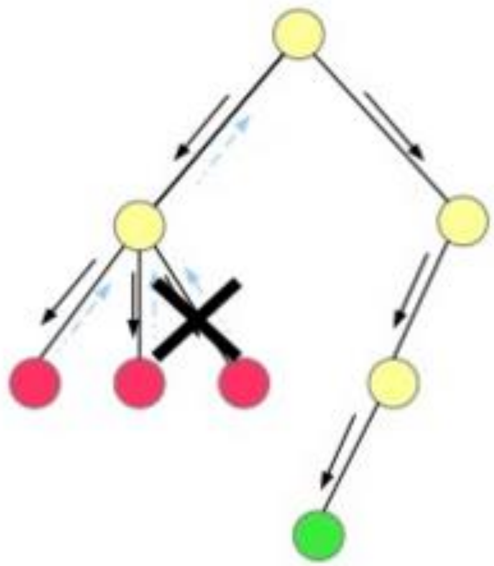
How Constraint Programming Works?

- CP is a *constructive* approach
- Values are assigned to variables one at a time to extend a partial solution to a complete solution
- At a point, it may be useless to further extend a partial solution as at least one constraint is already violated by the partial solution
  - The solver *backtracks* and tries a different value for a previously assigned variable
  - All possible assignments of values to variables can be examined in this way



How Constraint Programming Works?

- In CP, the basic search behaviour is tree search
- Including search space reduction via *domain filtering*
- Domain filtering
  - Before each value-variable assignment, *domain filtering* occurs
  - Each value of a variable which cannot be used in a solution (given the current partial assignment) can be removed
  - Each constraint type has a *specialized* algorithm which filters domains



Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Mathematical Programming (MP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer

What is Integer Linear Programming?

- An **integer programming** problem is a mathematical optimization or feasibility program in which some or all of the variables are restricted to be integers.
- The objective function and the constraints (other than the integer constraints) are linear.
- **Mixed integer linear programming** (MILP) involves problems in which only some of the variables are constrained to be integers, while other variables are allowed to be non-integers.
- **Zero-one linear programming** involves problems in which the variables are restricted to be either 0 or 1. Note that any bounded integer variable can be expressed as a combination of binary variables. For example, given an integer variable, x, the variable can be expressed :

$$x = x_1 + 2x_2 + 4x_3 + \ldots + 2^{\lfloor \log_2 U \rfloor} x_{\lfloor \log_2 U \rfloor + 1}.$$

Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Linear Programming (ILP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer

Comparison

- **CP works with the same concepts as mathematical programming: decision variables, objective function, and constraints.**
- **CP only discrete decision variables (integer or Boolean) vs MP discrete and continuous decision variables.**
- **CP logical constraints and arithmetic expressions (modulo, integer division, etc.) vs MP models only linear constraints or quadratic convex constraints.**
- **CP no limitation on the arithmetic constraints that can be set on decision variables vs MP specific to a class of problems whose solution space satisfies certain mathematical properties.**
- **Each optimization engine uses different techniques and algorithms to find feasible solutions and optimize them.**

A Tabular View

Constraint programming vs. mathematical programming

Feature	MP	CP
Relaxation	Yes	No
GAP measure	Yes	No
Optimality proof	Yes	Yes
Modeling limitations	Quadratic problems are limited to PSD (Positive Semi Definite) problems and Second Order Cone Programming (SOCP) problems	Discrete problems
Specialized constraints	No	Yes
Logical constraints	Yes	Yes
Theoretical grounds	Algebra	Graph theory and algorithmic
Modeler support	Yes	Yes
Model and run	Yes	Yes

Benefits of constraint programming

- **Solve time tabling problems and sequencing problems.**

- **An alternative to mathematical programming for allocation problems that have a slow convergence.**

- **Constraint programming has native support for:**
  - ✓ **Nonlinear costs or constraints**
  - ✓ **Logical constraints and statements**
  - ✓ **Constraints on and between interval variables**
  - ✓ **Compatibility or incompatibility constraints**
  - ✓ **More useful features**

Expressions and Constraints

- **Arithmetic constraints**
  - ✓ **x + y, x – y, x * y, x / y, x div y, x % y**
  - ✓ **min, max, abs, log, exp etc.**
  - ✓ **Piecewise linear functions**

- **Relational constraints**
  - ✓ **x == y, x != y, x <= y, x < y, lb <= x <= ub**

- **Logical constraints**
  - ✓ **!c, c||d, c && d,**
  - ✓ **c => d , c => d else e**
  - ✓ **c and d are relational or conditional constraints**

Expressions and Constraints

- **Reification**
  - ✓ **Relational or logical constraints can be used in a value context, where they evaluate to 0 or 1**

- **Examples**
  - ✓ **Arithmetic: max(0, abs(load[i] - cap))**
  - ✓ **Relational: wid * hei * depth * density <= maxLoad**
  - ✓ **Logical: end[i] <= start[j] || start[j] <= end[i]**
  - ✓ **Reification: spill == (load[i] > cap)**

Expressions and Constraints

- **Count expression**
  - ✓ count(dvar int[] x, int c)
  - ✓ Evaluates the number of variables in x with value c
  - ✓ e.g. Count the number of nurses allocated to ward 5
    - count(wardAllocation, 5) >= 3

- **Element expression**
  - ✓ (int[] a)[dvar int x] OR (dvar int[] a)[dvar int x]
  - ✓ Evaluates to the xth member of a
  - ✓ e.g. travel == 2 * distFromPittsburgh[holidayTown]
  - ✓ travel and holidayTown are variables

Expressions and Constraints

- **All Different**
  - ✓ **allDifferent(dvar int[] x)**
  - ✓ **All variables in x must take different values**
  - ✓ **e.g. The rank (visit priority) of each city is different**
    - **allDifferent(rankOfVisit)**

- **Allowed / Forbidden assignments**
  - ✓ **allowedAssignments({<a,b,c>} A, dvar int[3] x)**
  - ✓ **The assignments to x must fit with a tuple of A**
  - ✓ **forbiddenAssignments is the negation of this**

Expressions and Constraints

- **Bin packing constraint**
  - ✓ pack(dvar int[m] ld, dvar int[n]x, int[n] sz, dvar int c)
  - ✓ ld[i] == sum(j) (x[j] == i) * sz[j]
    - c is the number of containers used
- **Inverse constraint**
  - ✓ inverse(dvar int[n] x, dvar int[n] y)
  - ✓ x[i] == j <=> y[j] == i --- link primary and dual models
- **Lexicographic ordering constraint**
  - ✓ lex(dvar int[n] x, dvar int[n] y) --- break symmetries

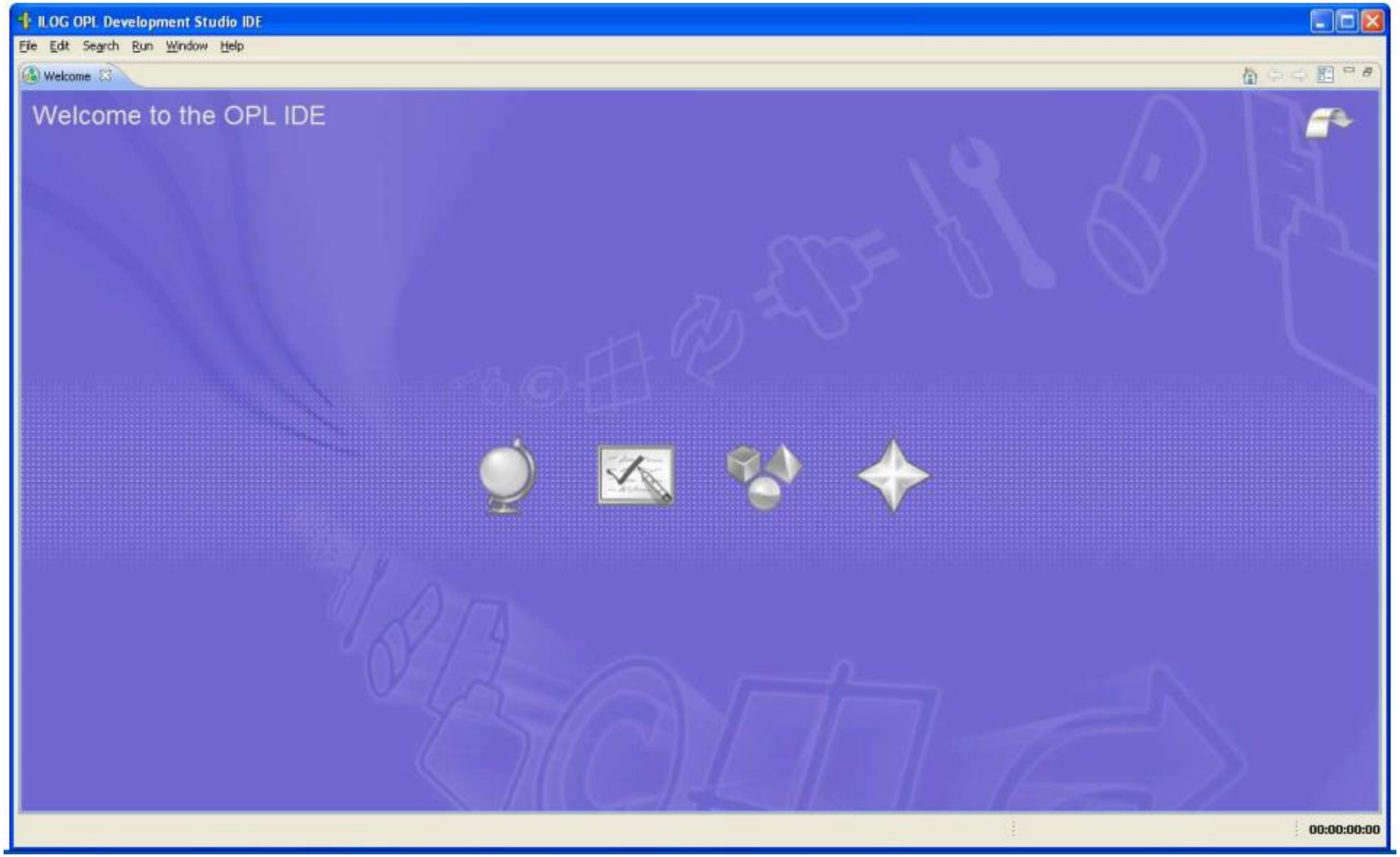
Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Linear Programming (ILP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer

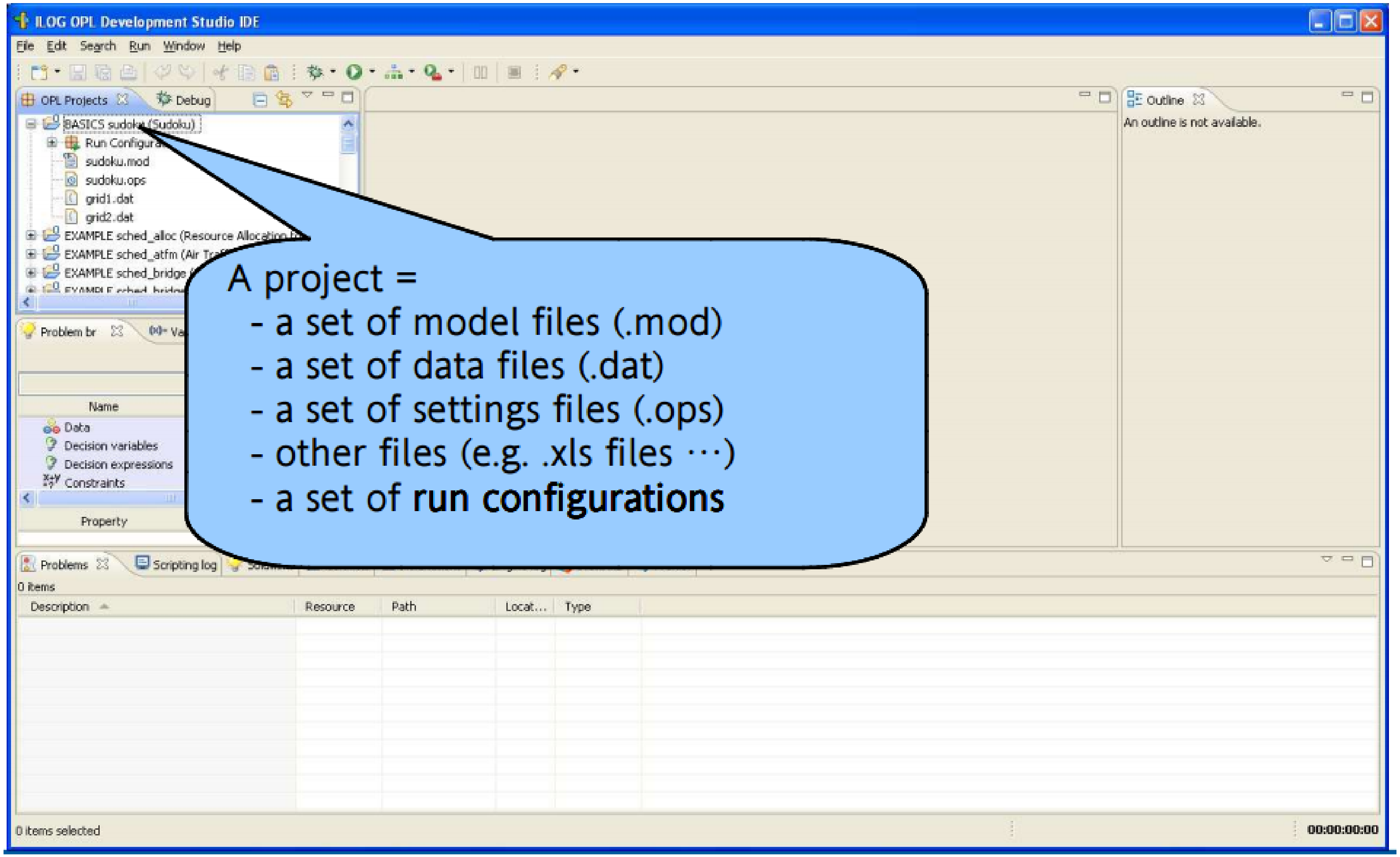
What is CP Optimizer

- A Constraint Programming engine with an emphasis on modelling and automatic search
- Available as a toolkit in C++ , Java, .NET
  - ✓ C++ is the native language and allows more possibilities, like writing incremental custom constraints, and fully controlling the search process
- Available as an engine inside ILOG OPL IDE
  - ✓ ILOG: name of the company (acquired by IBM)
  - ✓ OPL: optimization programming language
  - ✓ Higher level modelling and data manipulation

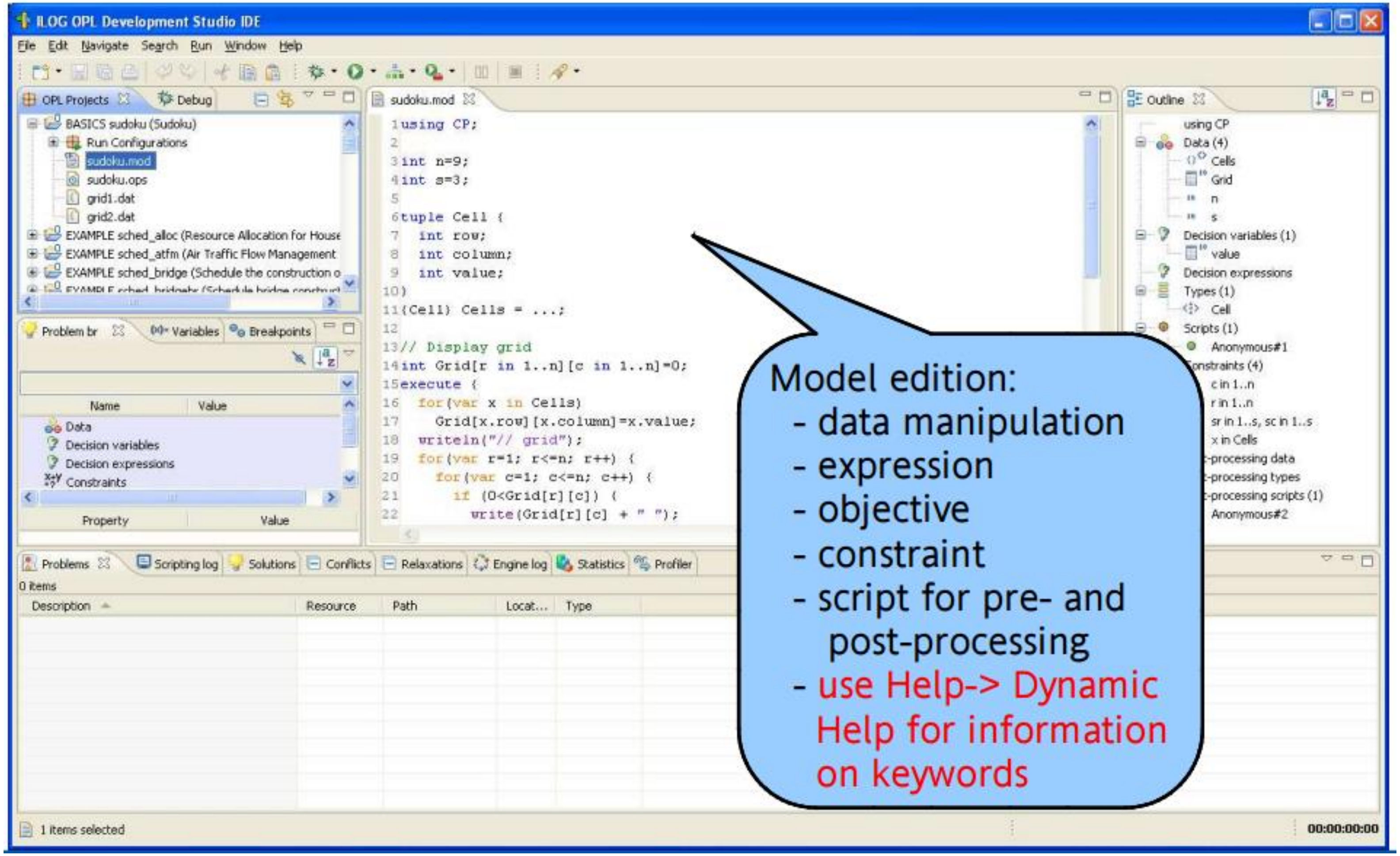
ILOG OPL IDE


ILOG OPL Development Studio IDE
File Edit Search Run Window Help
Welcome
Welcome to the OPL IDE
00:00:00:00

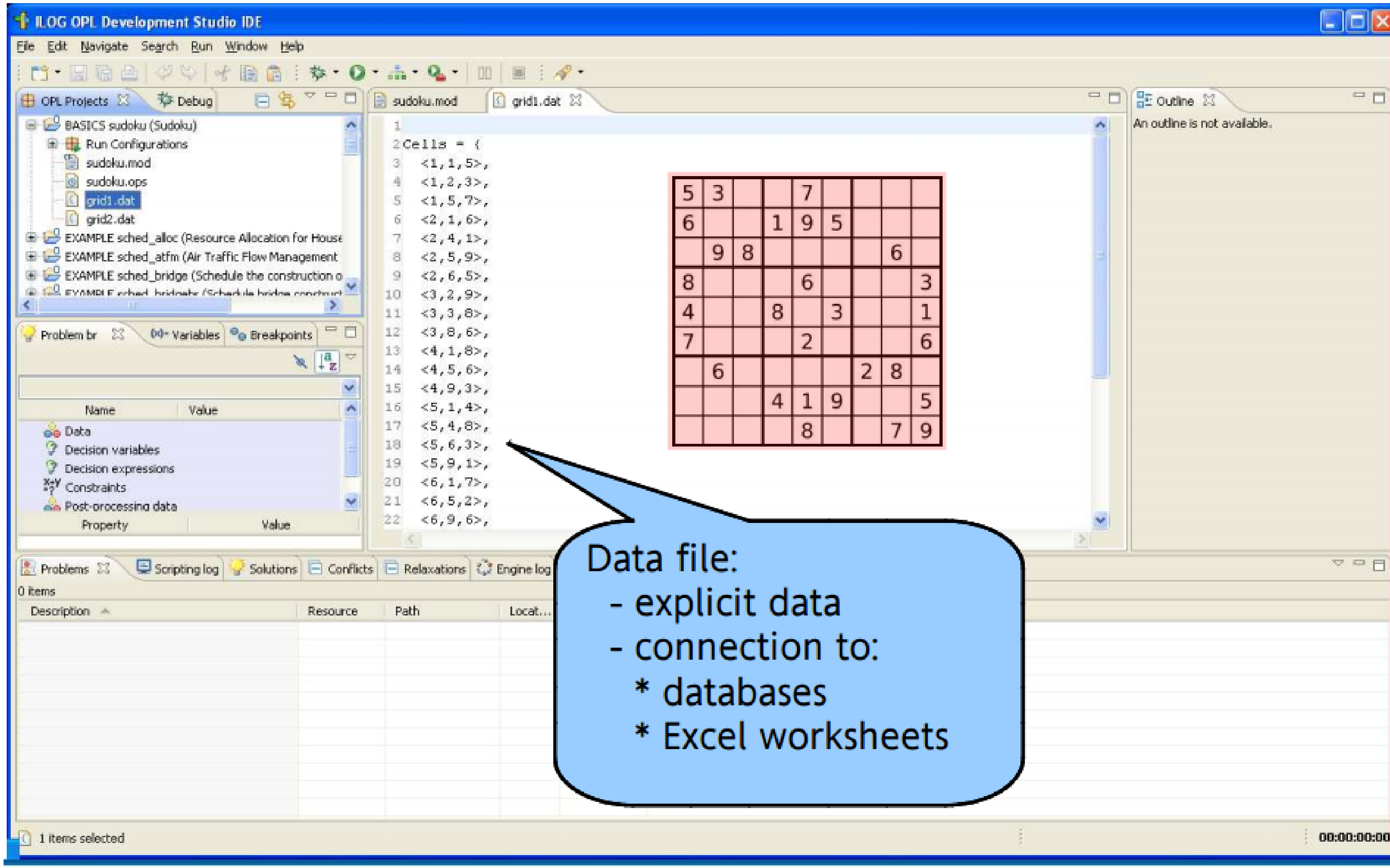
ILOG OPL IDE


ILOG OPL Development Studio IDE
File Edit Search Run Window Help
OPL Projects
Debug
BASICS sudoku (Sudoku)
Run Configura
sudoku.mod
sudoku.ops
grid1.dat
grid2.dat
EXAMPLE sched_alloc (Resource Allocation
EXAMPLE sched_atfm (Air Tra
EXAMPLE sched_bridge
Problem br
Name
Data
Decision variables
Decision expressions
Constraints
Property
Outline
An outline is not available.
Problems
Scripting log
0 items
Description
Resource
Path
Locat...
Type
0 items selected
00:00:00:00
A project =
- a set of model files (.mod)
- a set of data files (.dat)
- a set of settings files (.ops)
- other files (e.g. .xls files ⋯)
- a set of **run configurations**

ILOG OPL IDE


ILOG OPL Development Studio IDE
File Edit Navigate Search Run Window Help
OPL Projects
Debug
BASICS sudoku (Sudoku)
Run Configurations
sudoku.mod
sudoku.ops
grid1.dat
grid2.dat
sudoku.mod
1 using CP;
3 int n=9;
4 int s=3;
6 tuple Cell {
7 int row;
8 int column;
9 int value;
10 }
11 {Cell} Cells = ...;
13 // Display grid
14 int Grid[r in 1..n][c in 1..n]=0;
15 execute {
16 for(var x in Cells)
17 Grid[x.row][x.column]=x.value;
18 writeln("// grid");
19 for(var r=1; r<=n; r++) {
20 for(var c=1; c<=n; c++) {
21 if (0<Grid[r][c]) {
22 write(Grid[r][c] + " ");
Outline
Problems
Scripting log
Solutions
Conflicts
Relaxations
Engine log
Statistics
Profiler
0 items
Description
Resource
Path
Type
1 items selected
00:00:00:00
Model edition:
- data manipulation
- expression
- objective
- constraint
- script for pre- and post-processing
- use Help-> Dynamic Help for information on keywords

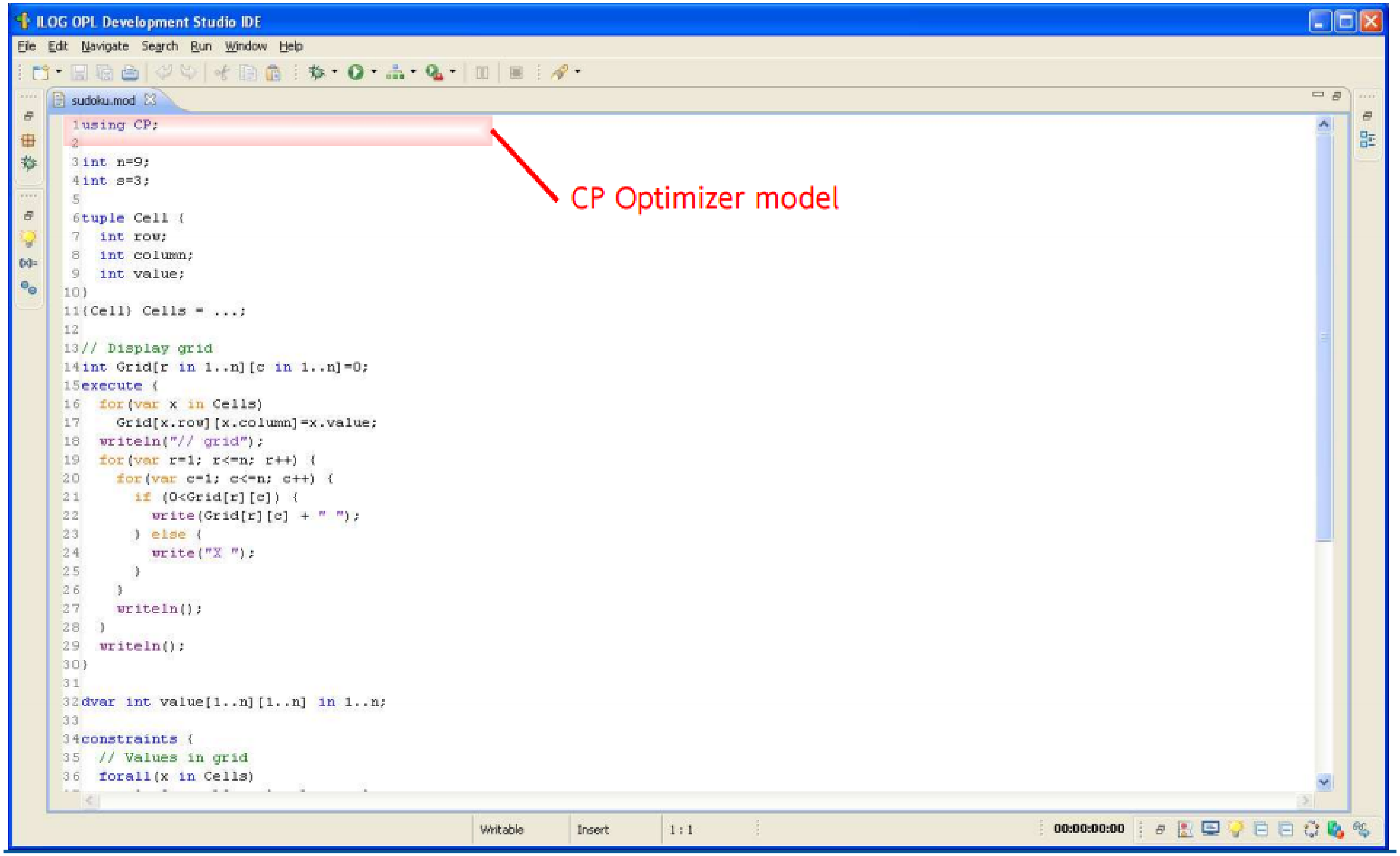
ILOG OPL IDE


ILOG OPL Development Studio IDE
File Edit Navigate Search Run Window Help
OPL Projects
Debug
BASICS sudoku (Sudoku)
Run Configurations
sudoku.mod
sudoku.ops
grid1.dat
grid2.dat
EXAMPLE sched_alloc (Resource Allocation for House
EXAMPLE sched_atfm (Air Traffic Flow Management
EXAMPLE sched_bridge (Schedule the construction o
Problem br
Variables
Breakpoints
Name
Value
Data
Decision variables
Decision expressions
Constraints
Post-processing data
Property
Value
sudoku.mod
grid1.dat
Cells = {
<1,1,5>,
<1,2,3>,
<1,5,7>,
<2,1,6>,
<2,4,1>,
<2,5,9>,
<2,6,5>,
<3,2,9>,
<3,3,8>,
<3,8,6>,
<4,1,8>,
<4,5,6>,
<4,9,3>,
<5,1,4>,
<5,4,8>,
<5,6,3>,
<5,9,1>,
<6,1,7>,
<6,5,2>,
<6,9,6>,
Outline
An outline is not available.
Data file:
- explicit data
- connection to:
* databases
* Excel worksheets
Problems
Scripting log
Solutions
Conflicts
Relaxations
Engine log
0 items
Description
Resource
Path
Locat...
1 items selected
00:00:00:00

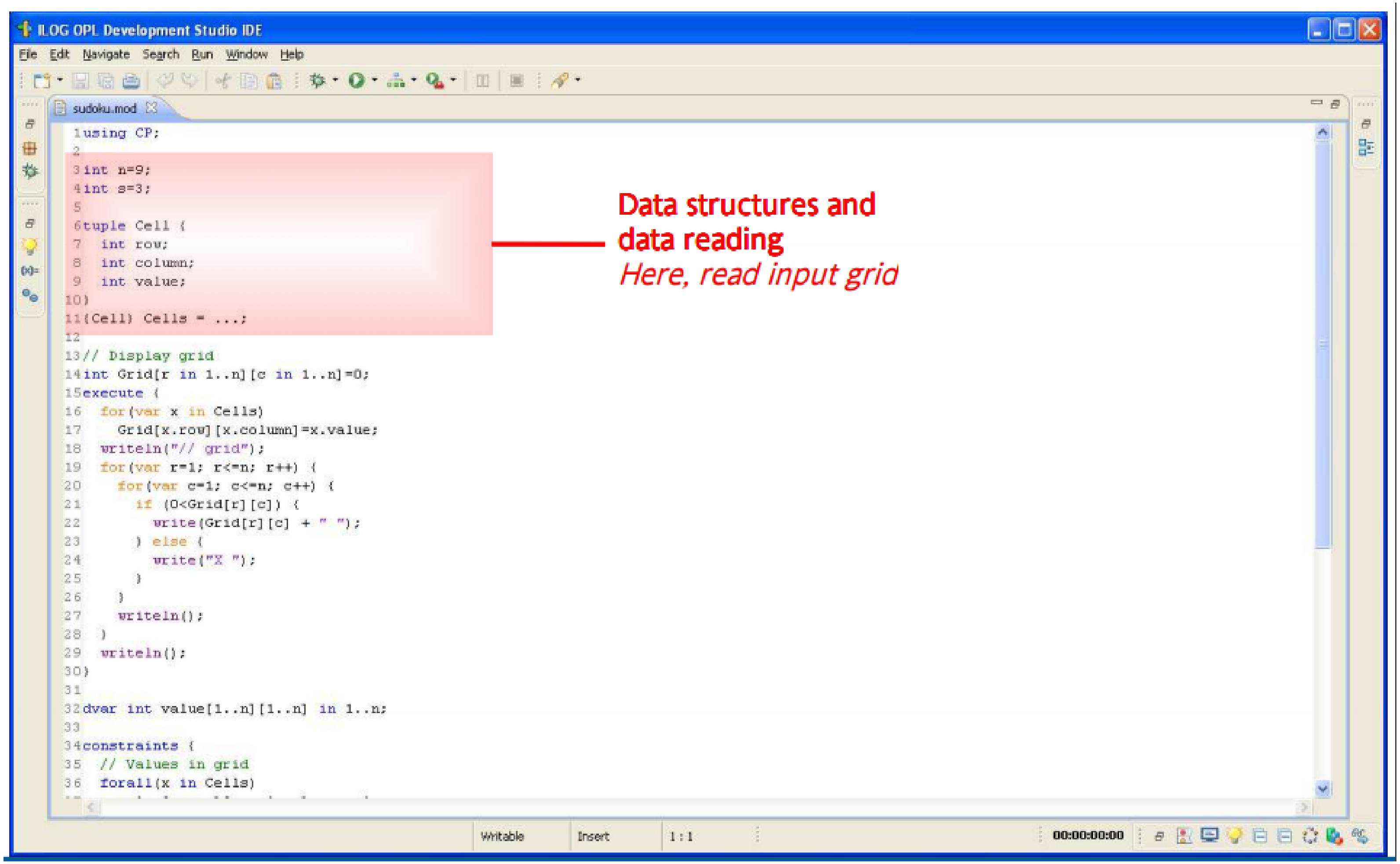
Overview of an CP Model using OPL

- **Top**
  - ✓ Data manipulation and pre-processing
    - declarative (expressions) and/or imperative (script)
  - ✓ Variable declarations
- **Middle**
  - ✓ Declarative model
    - objective (optional) and constraints
- **Bottom**
  - ✓ Post-processing of solutions
  - ✓ Declarative (expressions) and/or imperative (script)

An CP Model

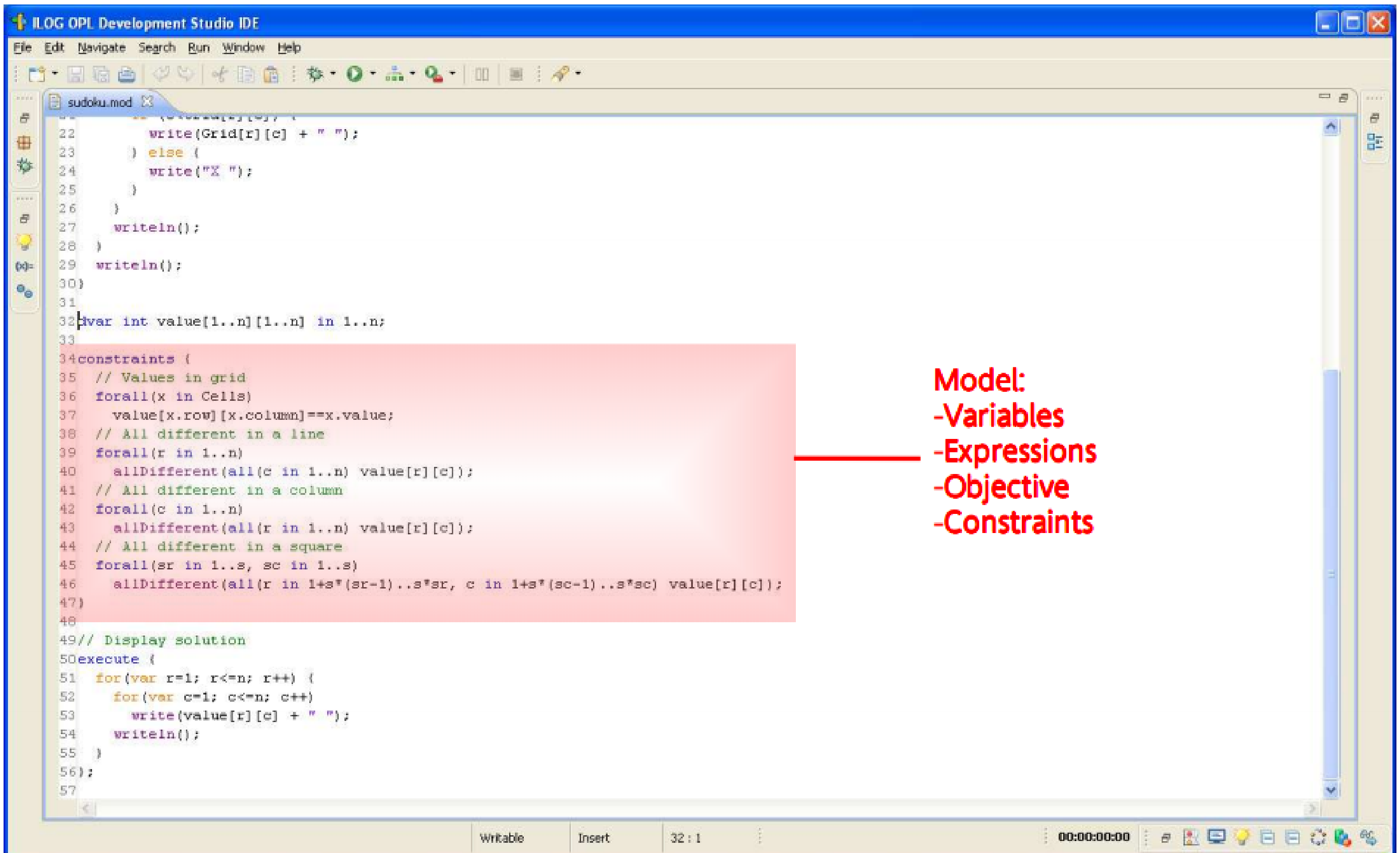

ILOG OPL Development Studio IDE
File Edit Navigate Search Run Window Help
sudoku.mod
using CP;
int n=9;
int s=3;
tuple Cell {
int row;
int column;
int value;
}
{Cell} Cells = ...;
// Display grid
int Grid[r in 1..n][c in 1..n]=0;
execute {
for(var x in Cells)
Grid[x.row][x.column]=x.value;
writeln("// grid");
for(var r=1; r<=n; r++) {
for(var c=1; c<=n; c++) {
if (0<Grid[r][c]) {
write(Grid[r][c] + " ");
} else {
write("X ");
}
}
writeln();
}
writeln();
}
dvar int value[1..n][1..n] in 1..n;
constraints {
// Values in grid
forall(x in Cells)
Writable
Insert
1 : 1
00:00:00:00
CP Optimizer model

An CP Model



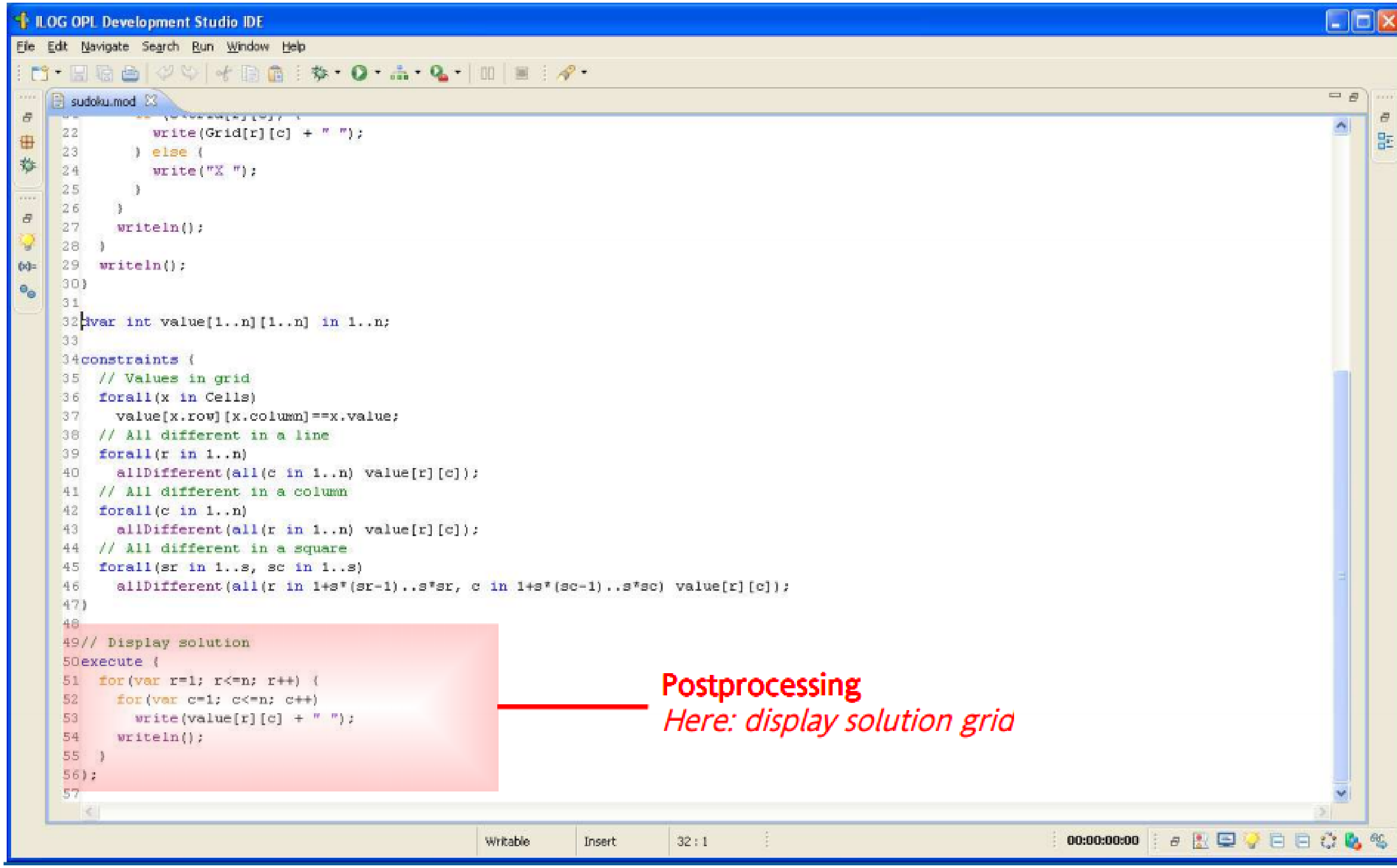
Data structures and
data reading
Here, read input grid

An CP Model



UC DAVIS
UNIVERSITY OF CALIFORNIA

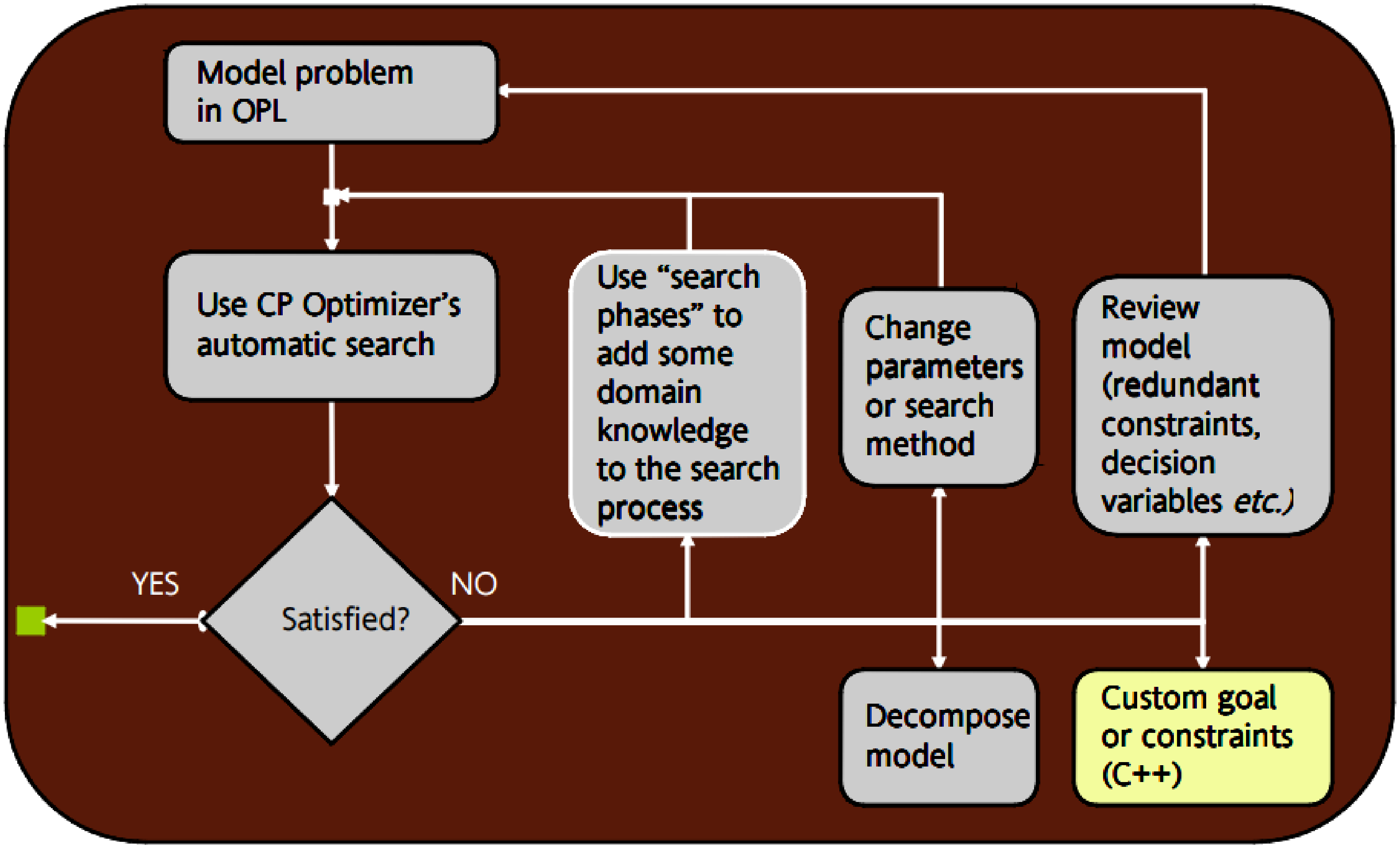
An CP Model


ILOG OPL Development Studio IDE
File Edit Navigate Search Run Window Help
sudoku.mod
        write(Grid[r][c] + " ");
      } else {
        write("X ");
      }
    }
    writeln();
  }
  writeln();
}
dvar int value[1..n][1..n] in 1..n;
constraints {
  // Values in grid
  forall(x in Cells)
    value[x.row][x.column]==x.value;
  // All different in a line
  forall(r in 1..n)
    allDifferent(all(c in 1..n) value[r][c]);
  // All different in a column
  forall(c in 1..n)
    allDifferent(all(r in 1..n) value[r][c]);
  // All different in a square
  forall(sr in 1..s, sc in 1..s)
    allDifferent(all(r in 1+s*(sr-1)..s*sr, c in 1+s*(sc-1)..s*sc) value[r][c]);
}
// Display solution
execute {
  for(var r=1; r<=n; r++) {
    for(var c=1; c<=n; c++)
      write(value[r][c] + " ");
    writeln();
  }
};
Postprocessing
Here: display solution grid
Writable
Insert
32 : 1
00:00:00:00

Search in CP Optimizer

- Automatic search is emphasized
  - ✓ Simpler, more maintainable, benefit from upgrades
- Search Phases
  - ✓ What group of variables to assign first
  - ✓ (optionally) define instantiation strategy
- Parameters
  - ✓ Inference levels and search control parameters
- Problem still hard?
  - ✓ Improve model
  - ✓ Simplify or relax specification
  - ✓ Decompose: CPLEX often useful here

Typical Use of CP Optimizer



UCDAVIS UNIVERSITY OF CALIFORNIA

Thank You !

xbwang@ucdavis.edu

UCDAVIS
UNIVERSITY OF CALIFORNIA