Model the traffic light system as CSP in Boolean constraint domain.
Traffic Light System

Right Turn
Forward
Left Turn

“Stop” Sensor

“Go” Sensor

Traffic Light System

Sensor

“Stop”

Sensor

“Go”
Describe the problem, usually in natural language.

Design a CSP model for representing the problem.

Implement the representation:
- Declare the unknowns in the representation as variables;
- Create the constraints imposed on those variables;

Let Solver search for the solution.
ILOG Solver 4.4

- ILOG Solver is a C++ library which can tackle various constraint domains.
  - e.g. Integer, Boolean, Set, Real etc.

- Object-orientation and constraint programming.

- http://www.ilog.com
An IlcManager

- It handles input, output, memory allocation, and other general services.
- First step in exploiting Solver in your application is to create a manager

```c
int main() {
    IlcManager m(IlcNoEdit);
    // declare unknowns in the problem
    // declare constraints in the problem
    // start the solution search
    m.end();
    return 0;
}
```
Constrained Integer Variables are variables that can only take integer values, and it’s domain consists only of integers.

- E.g. $D(X) = \{1,2,3,4,\ldots,n\}$

- Each domain has a minimum and maximum domain value.

- A variable is said to be **instantiated** if the domain consists of only one value.
The constructor declares a constrained integer variable,

\[
\text{IlcIntVar(IlcManager } m, \text{ IlcInt min, IlcInt max, char* name=0);} \\
\]

For example...

\[
\text{IlcIntVar var = IlcIntVar}(m, 0, 100, "X1"); \\
\text{IlcIntVar var}(m, 10, 20);
\]
Properties of Constrained Variables

- Min/Max values of variable domains can be retrieved from the constrained integer variable itself.

  ```cpp
  m.out() << var.getMin() << endl;
  m.out() << var.getMax() << endl;
  ```

- A constrained variable has a value only when it is bounded (instantiated) otherwise an error is raised,

  ```cpp
  if(var.isBound())
      m.out() << "value: " << var.getValue() << endl;
  ```
Variable can be printed directly

\texttt{m.out() \texttt{<< var \texttt{<< endl;}}}

Output

\texttt{> x1[0..100]}
\texttt{> x1[0..40 42..89]}
Defining Expressions

- You can formulate expressions on variables using the operators:
  - Adds +
  - Minus -
  - Multiplies *
  - Divides /
  - Absolute value \( \text{IlcAbs}(\text{IlcIntExp}) \)
  - ... etc

- E.g. \( \text{IlcAbs}(X + Y) \) is an \( \text{IlcIntExp} \).
You can formulate constraints on expressions using the operators:

- equality ==
- less than or equal to <=
- less than <
- greater than or equal to =>
- greater than >
- not equal to !=

E.g. $\text{IlcAbs}(X + Y) > 23$ is an $\text{IlcConstraint}$. 
For a constraint to be taken into account, you must add that constraint to its manager.

\[
\text{IlcManager: :add(IlcConstraint)}
\]

E.g. Post a constraint \( X + Y > 10 \),

\[
\text{IlcIntVar } X(m, 0, 5, "X");
\]
\[
\text{IlcIntVar } Y(m, 3, 8, "Y");
\]
\[
\text{m. add(X+Y>10)};
\]
Solution can be generated by the following procedures:

\[
\text{IlcIntVarArray vars}(m, 2, X, Y);
\text{m.add(IlcGenerate(vars));}
\text{m.nextSolution();}
\text{m.out() \&\& vars \&\& endl;}
\]

Output

\[\{X[3] \ Y[8]\}\]
4-Queens Problem

- To place 4 Queens on a 4 x 4 chess board such that none of them can capture any others.
4-Queens Problem

- **Variables**
  - $X_i, Y_i$ where $i,j = 0,1,2,3$

- **Domains**
  - $D(X_i) = D(Y_i) = \{0,1,2,3\}$

- **Constraints**
  - $X_i \neq X_j$ if $i \neq j$, $i,j = 0,1,2,3$
  - $Y_i \neq Y_j$ if $i \neq j$, $i,j = 0,1,2,3$
  - $|X_i - X_j| \neq |Y_i - Y_j|$ if $i \neq j$, $i,j = 0,1,2,3$
int main() {
   IlcManager m(IlCNoEdit);
   IlcIntVarArray X(m, 4, 0, 3);
   IlcIntVarArray Y(m, 4, 0, 3);

   X[0].setName("X0");
   Y[0].setName("Y0");
   // ... set variables’ name

   for(int i=0; i<3; i++)
      for(int j=i+1; j<4; j++){
         m.add(X[i] != X[j]);
         m.add(Y[i] != Y[j]);
         m.add(IlCAbs(X[i]-X[j])!=IlCAbs(Y[i]-Y[j]));
      }

   // to be continued...
}
// . . .

IlcIntVarArray vars(m, 8);
for(int i=0; i<4; i++){
    vars[2*i]=X[i];
    vars[2*i+1]=Y[i];
}

m.add(IlcGenerate(vars));
m.nextSolution();
m.out << vars << endl;
Any Questions...