Lecture 1

Introduction
Constraint Programming

- Alternative approach to programming
- Combination of reasoning and computing
- **Constraint** on a sequence of variables: a relation on their domains
- **Constraint Satisfaction Problem (CSP)**: a finite set of constraints

Constraint programming approach:
- Formulate your problem as CSP
- Solve the chosen representation using
  - domain specific methods, or
  - general methods
Solving CSPs

- Determine whether it has a solution (is consistent)
- Find a solution
- Find all solutions
- Find an optimal solution
- Find all optimal solutions
Domain Specific vs. General Methods

Domain specific methods:
Special purpose algorithms (constraint solvers), for example
- Program for solving systems of linear equations
- Package for linear programming
- Implementation of the unification algorithm

General Methods:
- Constraint propagation algorithms
- Search methods
Applications

- Interactive graphic systems
  (to express geometric coherence for scene analysis)
- Operations research problems
  (various optimization problems)
- Molecular biology
  (DNA sequencing, construction of 3D models of proteins)
- Electrical engineering (location of faults in the circuits, computing the circuit layouts, testing the design, verification)
- Natural language processing
  (construction of efficient parsers)
- Computer algebra
  (solving and/or simplifying equations over various algebraic structures)
Some Recent Applications

- Generation of coherent music radio programs
- Software engineering: design recovery and code optimization
- Selecting and scheduling satellite observations
Outline (of Today's Lecture)

- Define formally Constraint Satisfaction Problems (CSPs)
- Modeling: representing a problem as CSP
- Clarify various aspects of modeling:
  - in general there are several natural representations
  - some representations straightforward, some non-trivial
  - some representations rely on a “background” theory
- Show the generality of the notion of a CSP
Constraint Satisfaction Problem (CSP)

Given:

- Variables \( Y := y_1, \ldots, y_k \)
- Domains \( D_1, \ldots, D_k \)

Constraint \( C \) on \( Y \): subset of \( D_1 \times \ldots \times D_k \)

Given:

- Variables \( x_1, \ldots, x_n \)
- Domains \( D_1, \ldots, D_n \)

Constraint Satisfaction Problem (CSP):

\( \{C; x_1 \in D_1, \ldots, x_n \in D_n\} \)

\( C \) – constraints, each on a subsequence of \( x_1, \ldots, x_n \)

\((d_1, \ldots, d_n) \in D_1 \times \ldots \times D_n\) is a solution to the CSP if for every constraint \( C \in \mathcal{C} \) on \( x_{i_1}, \ldots, x_{i_m} \)

\((d_{i_1}, \ldots, d_{i_m}) \in C\)
Example: SEND + MORE = MONEY

Replace each letter by a different digit so that

\[
\begin{align*}
\text{SEND} \\
+ \text{MORE} \\
\text{MONEY}
\end{align*}
\]

is a correct sum.

Unique solution:

\[
\begin{align*}
9567 \\
+ 1085 \\
10652
\end{align*}
\]

Variables: S, E, N, D, M, O, R, Y
Domains: [1..9] for S, M
[0..9] for E, N, D, O, R, Y
Alternatives for Equality Constraints

- 1 equality constraint:

\[
1000 \cdot S + 100 \cdot E + 10 \cdot N + D \\
+ 1000 \cdot M + 100 \cdot O + 10 \cdot R + E \\
= 10000 \cdot M + 1000 \cdot O + 100 \cdot N + 10 \cdot E + Y
\]

- 5 equality constraints:

\[
D + E = 10 \cdot C_1 + Y \\
C_1 + N + R = 10 \cdot C_2 + E \\
C_2 + E + O = 10 \cdot C_3 + N \\
C_3 + S + M = 10 \cdot C_4 + O \\
C_4 = M
\]

where \( C_1, \ldots, C_4 \in [0..1] \) “carry” variables
Alternatives for Disequality Constraints

- 28 disequality constraints:
  \[ x \neq y \text{ for } x, y \in \{S, E, N, D, M, O, R, Y\}, \ x < y \]

- 1 disequality constraint:
  \[ \text{all\_different}(S, E, N, D, M, O, R, Y) \]

- Modeling it as an IP (integer programming) problem:
  For \( x, y \in \{S, E, N, D, M, O, R, Y\} \) transform \( x \neq y \) to
  \[
  \begin{align*}
  x - y &\leq 10 - 11z_{x,y} \\
  y - x &\leq 11z_{x,y} - 1 \\
  \text{where } z_{x,y} &\in [0..1]
  \end{align*}
  \]
  Disadvantage: 28 new variables
N Queens

Place \( n \) queens on an \( n \times n \) chess board so that they do not attack each other.

Variables: \( x_1, \ldots, x_n \)
Domains: [1..\( n \)]
Constraints:
- For \( i \in [1..n - 1] \) and \( j \in [i + 1..n] \)
  - \( x_i \neq x_j \) (rows)
  - \( x_i - x_j \neq i - j \) (South-West – North-East diagonals)
  - \( x_i - x_j \neq j - i \) (North-West – South-East diagonals)
Zebra Puzzle

A small street has five differently colored houses on it.

Five men of different nationalities live in them.

Each of them has a different profession, likes a different drink, and has a different pet animal.
Zebra Puzzle, ctd

The Englishman lives in the red house.
The Spaniard has a dog.
The Japanese is a painter.
The Italian drinks tea.
The Norwegian lives in the first house on the left.
The owner of the green house drinks coffee.
The green house is on the right of the white house.
The sculptor breeds snails.
The diplomat lives in the yellow house.
They drink milk in the middle house.
The Norwegian lives next door to the blue house.
The violinist drinks fruit juice.
The fox is in the house next to the doctor's.
The horse is in the house next to the diplomat's.

Who has the zebra and who drinks water?
Zebra Puzzle, ctd

25 Variables:
- red, green, white, yellow, blue
- english, spaniard, japanese, italian, norwegian
- dog, snails, fox, horse, zebra
- painter, sculptor, diplomat, violinist, doctor
- tea, coffee, milk, juice, water

Domains: [1..5]

Constraints:
- all_different(red, green, white, yellow, blue)
- all_different(english, spaniard, japanese, italian, norwegian)
- all_different(dog, snails, fox, horse, zebra)
- all_different(painter, sculptor, diplomat, violinist, doctor)
- all_different(tea, coffee, milk, juice, water)
Constraints, ctd

- The Englishman lives in the red house: 
  english = red
- spaniard = dog
- japanese = painter
- italian = tea
- The Norwegian lives in the first house on the left: 
  norwegian = 1
- green = coffee
- The green house is on the right of the white house: 
  green = white + 1
Constraints, ctd

- sculptor = snails
- diplomat = yellow
- milk = 3
- The Norwegian lives next door to the blue house:
  \(|\text{norwegian} - \text{blue}| = 1\)
- violinist = juice
- The fox is in the house next to the doctor's:
  \(|\text{fox} - \text{doctor}| = 1\)
- \(|\text{horse} - \text{diplomat}| = 1\)
Crossword Puzzles

Fill the crossword grid with words from
  - HOSES, LASER, SAILS, SHEET, STEER
  - HEEL, HIKE, KEEL, KNOT, LINE
  - AFT, ALE, EEL, LEE, TIE

Variables: \(x_1, \ldots, x_8\)

Domains: \(x_7 \in \{AFT, ALE, EEL, LEE, TIE\}\), etc.

Constraints: one per crossing
  \(C_{1,2} := \{(HOSES, SAILS), (HOSES, SHEET), (HOSES, STEER), (LASER, SAILS), (LASER, SHEET), (LASER, STEER)\}\)

etc.
Unique Solution

```
H O S E S
A T
H I K E
A L E E
L A S E R
E L
```
Qualitative Temporal Reasoning

The meeting ran non-stop the whole day.
Each person stayed at the meeting for a continuos period of time.
The meeting began while Mr Jones was present and finished while Ms White was present.
Ms White arrived after the beginning of the meeting.
Director Smith was also present, but he arrived after Jones had left.
Mr Brown talked to Ms White in the presence of Smith.
Could Jones and White possibly have talked during this meeting?
13 Temporal Relations (Allen 1983)

- A before B
- B after A
- A meets B
- B met-by A
- A overlaps B
- B overlapped-by A
- A starts B
- B started-by A
- A during B
- B contains A
- A finishes B
- B finished-by A
- A equals B
- A equals B
Composition Table

- Consider three events, A, B, and C
  Given the temporal relations between A, B and between B, C, what is the temporal relation between A and C?

- (Allen 1983) defines a 13 × 13 table:
  Example: if A overlaps B and B before C, then A before C
  This yields entry allen(overlaps, before, before)
  (In total 409 entries)
The Composition Table, Part 1

<table>
<thead>
<tr>
<th></th>
<th>before</th>
<th>after</th>
<th>meets</th>
<th>met-by</th>
<th>overlaps</th>
<th>overl.-by</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>before</td>
<td>TEMP</td>
<td>before</td>
<td>before</td>
<td>before</td>
<td>before</td>
</tr>
<tr>
<td>after</td>
<td>TEMP</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
</tr>
<tr>
<td>meets</td>
<td>before</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
</tr>
<tr>
<td>met-by</td>
<td>before</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
</tr>
<tr>
<td>overlaps</td>
<td>before</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
</tr>
<tr>
<td>overl.-by</td>
<td>before</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
</tr>
<tr>
<td>starts</td>
<td>before</td>
<td>after</td>
<td>before</td>
<td>after</td>
<td>before</td>
<td>before</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>before</th>
<th>after</th>
<th>meets</th>
<th>met-by</th>
<th>overlaps</th>
<th>overl.-by</th>
</tr>
</thead>
<tbody>
<tr>
<td>started-by</td>
<td>before</td>
<td>after</td>
<td>overlaps contains finished-by</td>
<td>met-by</td>
<td>overl.-by contains finished-by</td>
<td></td>
</tr>
<tr>
<td>during</td>
<td>before</td>
<td>after</td>
<td>before</td>
<td>after</td>
<td>before</td>
<td>overl.-by</td>
</tr>
<tr>
<td>contains</td>
<td>before</td>
<td>after</td>
<td>overlaps contains finished-by</td>
<td>overl.-by started-by contains</td>
<td>overl.-by started-by contains</td>
<td></td>
</tr>
<tr>
<td>finishes</td>
<td>before</td>
<td>after</td>
<td>overl.-by started-by contains</td>
<td>overl.-by started-by contains</td>
<td></td>
<td></td>
</tr>
<tr>
<td>finished-by</td>
<td>before</td>
<td>after</td>
<td>overl.-by started-by contains</td>
<td>overl.-by started-by contains</td>
<td></td>
<td></td>
</tr>
<tr>
<td>equals</td>
<td>before</td>
<td>after</td>
<td>overl.-by started-by contains</td>
<td>overl.-by started-by contains</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Foundations of Constraint Programming  
Introduction  
23
## The Composition Table, Part 2

<table>
<thead>
<tr>
<th></th>
<th>starts</th>
<th>started-by</th>
<th>during</th>
<th>contains</th>
<th>finishes</th>
<th>finished-by</th>
<th>equals</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>before</td>
<td>before</td>
<td>before</td>
<td>before</td>
<td>before</td>
<td>before</td>
<td>before</td>
</tr>
<tr>
<td>after</td>
<td>during</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>after</td>
</tr>
<tr>
<td>meets</td>
<td>meets</td>
<td>meets</td>
<td>before</td>
<td>before</td>
<td>before</td>
<td>meets</td>
<td></td>
</tr>
<tr>
<td>met-by</td>
<td>during</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>met-by</td>
<td>met-by</td>
<td>met-by</td>
</tr>
<tr>
<td>overlaps</td>
<td>overlaps</td>
<td>overlaps</td>
<td>overlaps</td>
<td>overlaps</td>
<td>overlaps</td>
<td>overlaps</td>
<td></td>
</tr>
<tr>
<td>over.-by</td>
<td>during</td>
<td>after</td>
<td>after</td>
<td>after</td>
<td>over.-by</td>
<td>over.-by</td>
<td>over.-by</td>
</tr>
<tr>
<td>starts</td>
<td>starts</td>
<td>starts</td>
<td>during</td>
<td>before</td>
<td>during</td>
<td>starts</td>
<td></td>
</tr>
</tbody>
</table>
Representation as CSP

- 5 events:
  - M (meeting)
  - J (Jones's presence)
  - B (Brown's presence)
  - S (Smith's presence)
  - W (White's presence)

- 10 variables, each associated with an ordered pair of events and each with a domain:
  \[ TEMP := \{ \text{before, after, meets, met-by, overlaps, overlapped-by, starts, started-by, during, contains, finishes, finished-by, equals} \} \]
  \[ REAL-OVERLAP := TEMP - \{ \text{before, after, meets, met-by} \} \]
Representation as CSP, ctd

- **Constraints:**
  - $x_{J, M} \in \{\text{overlaps, contains, finished-by}\}$
  - $x_{M, W} \in \{\text{overlaps}\}$
  - $x_{M, S} \in \text{REAL-OVERLAP}$
  - $x_{J, S} \in \{\text{before}\}$
  - $x_{B, S} \in \text{REAL-OVERLAP}$
  - $x_{B, W} \in \text{REAL-OVERLAP}$
  - $x_{S, W} \in \text{REAL-OVERLAP}$
  - $x_{J, B}, x_{J, W}, x_{M, B} \in \text{TEMP}$

- **Final question**
  If the constraint $x_{J, W} \in \text{REAL-OVERLAP}$ is added, is the CSP consistent?
Allen's Temporal Constraints

- **allen**: the composition table as a ternary relation (409 triples)

For each ordered triple $A, B, C$ of the events:
- a constraint $C_{A,B,C}$ on the variables $x_{A,B}, x_{B,C}, x_{A,C}$
- $C_{A,B,C} := \text{allen} \cap (D_{A,B} \times D_{B,C} \times D_{A,C})$
- where
  - $x_{A,B} \in D_{A,B}$
  - $x_{B,C} \in D_{B,C}$
  - $x_{A,C} \in D_{A,C}$
Qualitative Spatial Reasoning

Two houses are connected by a road. The first house is surrounded by its garden or is adjacent to its boundary while the second house is surrounded by its garden.

What can we conclude about the relation between the second garden and the road?
8 Spatial Relations

\[
\begin{array}{ccc}
\text{disjoint}(A,B) & \text{meet}(A,B) & \text{equal}(A,B) \\
\text{covers}(A,B) & \text{contains}(A,B) & \text{overlap}(A,B) \\
\text{coveredby}(B,A) & \text{inside}(B,A) &
\end{array}
\]

\[
\text{RCC8} := \{\text{disjoint, meet, equal, covers, coveredby,}
\text{contains, inside, overlap}\}
\]
## The Composition Table for RCC8

<table>
<thead>
<tr>
<th></th>
<th>disjoint</th>
<th>meet</th>
<th>equal</th>
<th>inside</th>
<th>covered by</th>
<th>contains</th>
<th>covers</th>
<th>overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>disjoint</td>
<td>RCC8</td>
<td>disjoint</td>
<td>disjoint</td>
<td>inside</td>
<td>covered by</td>
<td>disjoint</td>
<td>disjoint</td>
<td>overlap</td>
</tr>
<tr>
<td>meet</td>
<td>disjoint</td>
<td>meet</td>
<td>disjoint</td>
<td>inside</td>
<td>covered by</td>
<td>disjoint</td>
<td>disjoint</td>
<td>overlap</td>
</tr>
<tr>
<td></td>
<td>meet</td>
<td>inside</td>
<td>covered by</td>
<td>inside</td>
<td>covered by</td>
<td>disjoint</td>
<td>disjoint</td>
<td>overlap</td>
</tr>
<tr>
<td>equal</td>
<td>disjoint</td>
<td>meet</td>
<td>disjoint</td>
<td>inside</td>
<td>covered by</td>
<td>disjoint</td>
<td>disjoint</td>
<td>overlap</td>
</tr>
<tr>
<td>inside</td>
<td>disjoint</td>
<td>inside</td>
<td>inside</td>
<td>inside</td>
<td>inside</td>
<td>RJCC8</td>
<td>disjoint</td>
<td>overlap</td>
</tr>
<tr>
<td>covered by</td>
<td>disjoint</td>
<td>meet</td>
<td>covered by</td>
<td>inside</td>
<td>covered by</td>
<td>disjoint</td>
<td>disjoint</td>
<td>overlap</td>
</tr>
<tr>
<td>contains</td>
<td>disjoint</td>
<td>meet</td>
<td>covered by</td>
<td>contains</td>
<td>covered by</td>
<td>covers</td>
<td>contents</td>
<td>covers</td>
</tr>
<tr>
<td>covers</td>
<td>disjoint</td>
<td>meet</td>
<td>covers</td>
<td>inside</td>
<td>covered by</td>
<td>covers</td>
<td>contents</td>
<td>covers</td>
</tr>
<tr>
<td>overlap</td>
<td>disjoint</td>
<td>meet</td>
<td>covers</td>
<td>inside</td>
<td>covered by</td>
<td>covers</td>
<td>contents</td>
<td>covers</td>
</tr>
</tbody>
</table>

Foundations of Constraint Programming

Introduction
Representation as CSP

- **5 spatial objects**: H1, H2, G1, G2, R

- **10 variables with domains, each associated with an ordered pair of spatial objects**:
  - $x_{H1,G1} \in \{\text{inside, coveredby}\}$
  - $x_{H2,G2} \in \{\text{inside}\}$
  - $x_{H1,H2} \in \{\text{disjoint}\}$
  - $x_{H1,R} \in \{\text{meet}\}$
  - $x_{H2,R} \in \{\text{meet}\}$
  - $x_{G1,G2} \in \{\text{disjoint, meet}\}$
  - $x_{H1,G2} \in \{\text{disjoint, meet}\}$
  - $x_{G1,H2} \in \{\text{disjoint, meet}\}$
  - $x_{G1,R} \in \text{RCC8}$
  - $x_{G2,R} \in \text{RCC8}$
Constraints

- $S_3$: the composition table as a ternary relation (193 triples)
- For each ordered triple $A, B, C$ of the objects:
  - a constraint $C_{A,B,C}$ on the variables $x_{A,B}, x_{B,C}, x_{A,C}$
  - $C_{A,B,C} := S_3 \cap (D_{A,B} \times D_{B,C} \times D_{A,C})$
  - where
    - $x_{A,B} \in D_{A,B}$
    - $x_{B,C} \in D_{B,C}$
    - $x_{A,C} \in D_{A,C}$
Constrained Optimization Problem (COP)

- Given:
  - a CSP
    \[ \mathcal{P} := \langle C ; x_1 \in D_1, \ldots, x_n \in D_n \rangle \]
  - an objective function
    \[ \text{obj} : \text{Sol} \to \mathcal{R} \]

- \((\mathcal{P}, \text{obj})\) a constrained optimization problem (COP)

- Task: Find a solution \(d\) to \(\mathcal{P}\) for which the value \(\text{obj}(d)\) is optimal (maximal)
Example: Knapsack Problem

Given a knapsack of a fixed volume and $n$ objects, each with a volume and a value. Find a collection of these objects with maximal total value that fits in the knapsack.

Representation as a COP:

Given: knapsack volume $v$; volumes $a_1, \ldots, a_n$; values $b_1, \ldots, b_n$

Variables: $x_1, \ldots, x_n$

Domains: $\{0,1\}$

Constraint:
\[ \sum_{i=1}^{n} a_i \cdot x_i \leq v \]

Objective function:
\[ \sum_{i=1}^{n} b_i \cdot x_i \]
Example: Golomb Ruler

Golomb ruler with \( m \) marks: an ordered sequence of \( m \) natural numbers such that the distance between any two elements in this sequence is unique.

The largest element of a Golomb ruler is its length.

An optimum Golomb ruler with \( m \) marks: a Golomb ruler with \( m \) marks with a minimal length.
Optimum Golomb Ruler with 5 Marks

A Golomb ruler with 5 marks: \(0, 1, 4, 9, 11\)

The distances are:
- for elements one apart: 1, 3, 5, 2
- for elements two apart: 4, 8, 7
- for elements three apart: 9, 10
- for elements four apart: 11

In fact, this is an optimum Golomb ruler with 5 marks.
The largest known optimum Golomb ruler has 21 marks and is of length 333.
Representations as a COP

- Pair: two numbers $i, j$ such that $1 \leq i < j \leq m$
- Pairs $i, j$ and $k, l$ are
  - different if $i \neq k$ or $j \neq l$
  - disjoint if $i \neq k$ and $j \neq l$

Representation 1

Variables: $x_1, \ldots, x_m$

Domains: $\mathbb{N}$

Constraints:
- $x_i < x_{i+1}$ for $i \in [1..m-1]$
- $x_j - x_i \neq x_l - x_k$ for all different pairs $i, j$ and $k, l$

Objective function: $-x_m$
Representations as a COP, ctd

Representation 2
Constraints:
- $x_i < x_{i+1}$ for $i \in [1..m - 1]$
- $x_j - x_i \neq x_j - x_k$ for all disjoint pairs $i, j$ and $k, l$

Representation 3
Variables: $x_1, ..., x_m, z_{ij}$ for each pair $i, j$
Domains: $\mathbb{N}$ for $x_1, ..., x_m$
- $\mathbb{N} \setminus \{0\}$ for $z_{ij}$
Constraints:
- $z_{ij} = x_j - x_i$ for each pair $i, j$
- $z_{ij} \neq z_{kl}$ for all different pairs $i, j$ and $k, l$

Representation 4
Replace the disequality constraints by a single all_different on $z_{ij}$. 
Different Representations as CSP

Less Contrived Examples

- A Microcode Label Assignment Problem
  - CSP representation: 187 finite integer domain variables
  - IP representation: 2024 Boolean variables

- A Packing Problem
  - CSP representation: 7 finite integer domain variables, 2 constraints
  - IP representation: 42 Boolean variables, 18 constraints

- A Golf Scheduling Problem
  - CP representation: 176 variables
  - IP representation 1: 2574 variables
  - IP representation 2: 592 variables
Objectives (of Today's Lecture)

- Define formally Constraint Satisfaction Problems (CSPs)
- Modeling: representing a problem as CSP
- Clarify various aspects of modeling:
  - in general there are several natural representations
  - some representations straightforward, some non-trivial
  - some representations rely on a “background” theory
- Show the generality of the notion of a CSP