Constraint-based
Information Integration

Craig Knoblock
University of Southern California

Thanks to Jose Luis for some of these slides
Constraint Satisfaction and Propagation for Integration

- Integrating data from multiple sources often involves reasoning about the information.
- Constraints provide a approach to expressing relationships and filtering data.
Part I: Integration Frameworks
  - Constraint satisfaction in SmartClients
  - Constraint propagation in Heracles

Part II: CSPs for Integrating Data
  - Constraint satisfaction for building identification from open source data
SmartClients [Torrens et al, 2002]

- Cast an integration problem as a Constraint Satisfaction Problem (CSP)
- Given a request, the server retrieves the required data and sends the data and the CSP to the client
- Client solves the CSP locally
  - Large complex problem transmitted in small amount of space
  - Provides fine-grained user interaction with the data
Example Problem

- I live in Bern, Switzerland, and would like to visit colleagues in Princeton and London. I would like to spend at least two days in each place, and will need to travel in the first two weeks of February.
  - 1st leg from Bern to Princeton: flights from ZRH/BSL/GVA to JFK/EWR/PHL on the dates from 1st to 10th February
  - 2nd leg from Princeton to London: flights from JFK/EWR/PHL to LGW/LHR/LCY on the dates from 4th to 12th February, and
  - 3rd leg from London to Bern: flights from LGW/LHR/LCY to ZRH/BSL/GVA on the dates from 6th to 14th February
A Constraint Satisfaction Problem (CSP) is:
- a set of variables each with
- a set of domain of values, and
- a set of constraints that define which combinations of variable values are allowed
- the task is to find a value for each variable such that all constraints are simultaneously satisfied

Algorithms and techniques for solving CSPs have been widely studied
Conjunctive Normal Form: Example

- As a CSP:
  - Each clause is a constraint, each literal is a variable with a \{true, false\} domain

\[
\{\neg x \lor \neg y \lor \neg z\} \land \{x \lor y \lor \neg z\} \land \\
\{\neg x \lor y\} \land \{\neg y \lor z\}
\]

- To Solve:
  - All clauses evaluate true
  - e.g.: $x = \text{false}$, $y = \text{true}$, $z = \text{true}$
Example CSP Graph for Travel

Bill's Agenda CSP

Travel CSP

John's Agenda CSP

Bill's Flights

GVA, ZRH

Outgoing Flight

Meeting Place
NYC, PHL

Start Meeting
Date + Time

End Meeting
Date + Time

Dep T_1

Arr T_1

Dep T_2

Arr T_2

Return Flight

Bill's Availability

Start Time

End Time

John's Availability

Start Time

End Time

Variables Arr T_i, Dep T_i, Start-Time and End-Time contain information about Dates and Times.
Formalization of Example: Variables

- $X = \{DT_0, \ldots, DT_{n-1}, AT_0, \ldots, AT_{n-1}, Airports_0, \ldots, Airports_n,$
  $Flights_0, \ldots, Flights_{n-1}, AirCrafts, Fares, Airlines, \ldots\}$ is a set of variables
- $DT_i$ and $AT_i$ represent the dates and times on which the traveler could depart and arrive respectively
- $Airports_i$ represents the possible airports near the departure for leg of the itinerary
- $Flights_i$ stands for the possible flights between the airports of $Airports_i$ and $Airports_{i+1}$
Formalization of Example:

Domains

- $D = \{D_1, \ldots, D_n\}$ is the set of domains
  - For variables $DT_i$ or $AT_i$: the domain contains all possible departure and arrival times for the leg $i$
  - For variables $Airports_i$: the domain is a set of airports for the departure of the leg $i$
  - For variables $Flights_i$: the domain is the set of possible flights from $Airports_i$ to $Airports_{i+1}$
  - For variables $AirCrafts$, $Fares$ and $Airlines$: the domain is the set of different aircrafts, the set of available fares or the set of airline companies respectively
Formalization of Example: Constraints

- \( C = \{C_1, \ldots, C_n\} \) is the set of constraints
  - Two types of constraints:
    - Those imposed by the user’s preferences
    - Those imposed by flight schedules
  - There are constraints on the variables \( \text{Flights}_i, \text{Airports}_i, \text{DT}_i \) and \( \text{AT}_i \) that guarantee the flight is compatible with the airports, departure times and arrival times
  - A binary constraint in between \( \text{AT}_i \) and \( \text{DT}_{i-1} \) takes into consideration that the flight for leg\(_{i+1}\) departs after the flight for leg\(_i\) arrives
  - User preferences are expressed by means of constraints between Flight\(_i\) variables and Aircrafts, Airlines, Fares, and other variables
Constraint Graph for Flights
Architecture for SmartClients

Database with information for the CSP → SERVER (Generation of CSP) → Initial specification → smart agent → CLIENT (Resolution of CSP)
Pros and Cons

**Pros**
- Elegant approach that exploits past work on CSPs
- Minimizes the data retrieval and supports complex reasoning and integration of the data

**Cons**
- Assumes that all data can be retrieved before any reasoning about the data
- In the travel planning, assumes that prices are the same on any date and there are no issues with flight availability
Heracles Constraint Integration Framework

- Framework for building integrated applications
- Interleaves planning and information gathering
- Uses a constraint reasoner to decide what sources to query and to integrate the results
The Travel Assistant
Dynamically Updates Slots as Information Becomes Available
Supports Informed Choices
Changes Propagate Throughout
User Can Specify High-Level Preferences
Constraint Networks for Integrating Information

- Components:
  - Representation of the variables
  - Representation of constraints
  - Hierarchical template representation
  - Constraint propagation and cycle detection
Constraint Networks for Integrating Information

- Components:
  - Representation of the variables
  - Representation of constraints
  - Hierarchical template representation
  - Constraint propagation and cycle detection
Constraint Variables

- Constraint network consists of a set of variables such as:
  - MeetingStartTime
  - MeetingLocation

- Each variable depends on a set of ancestors.

- Variables are related by constraints that determine the possible values of a solution.
Constraint Networks for Integrating Information

Components:

- Representation of the variables
- Representation of constraints
- Hierarchical template representation
- Constraint propagation and cycle detection
Constraint Representation

- Constraints are computable components:
  - Local calculations based on Xquery
    - MeetingStartTime + MeetingDuration --> MeetingEndTime
  - Web and Database Wrappers
    - ITN: DepartureAirport, ArrivalAirport, Date --> Flights
    - Yahoo Weather: City, Date --> Weather prediction
  - External Programs (Outlook, Planners, etc)
    - Outlook Calendar: Date --> Meetings
Constraint Structure

Constraint

- Arguments: input and output variables
- Call:
  - Construct table with inputs and corresponding calls (http requests, SQL queries, etc) to sources (wrappers, DBs, etc) [using XML Query]
  - Calls are executed and results stored in an table
- Output
  - Restructure source results into desired output [using XML Query]
Drive or Take a Taxi?

DepartureDate: Sep 30, 2000
ReturnDate: Oct 2, 2000
DepartureAirport: LAX

1. **FindClosestAirport**
   - OriginAddress
   - DestinationAddress
   - GetDistance: 15.1 miles

2. **computeDuration**
   - Duration: 3 days

3. **getParkingRate**
   - ParkingRate: $7.00/day
   - ParkingTotal: $21.00
   - ParkingTotal: $23.00

4. **SelectModeToAirport**
   - ModeToAirport: Drive

Get Taxi Fare:
- TaxiFare: $23.00
Constraint Networks for Integrating Information

- Components:
  - Representation of the variables
  - Representation of constraints
  - Hierarchical template representation
  - Constraint propagation and cycle detection
Hierarchically-Partitioned Constraint Networks

- Template:
  - Groups related variables and constraints
  - Organizes information for computation and presentation to user

- Templates organized hierarchically
  - Template decomposed into subtemplates
  - Choose among alternative subtemplates
Template Structure

Template
- Arguments: input and output variables
- Variables: name, type, default values
- Constraints
- Expansions: alternative subtemplate calls
- GUI specification
Partitioned Constraint Network

Who

Company

Subject

Dest. Addr.

Starting Time

Ending Time

Dest Weather

Origin Weather

Distance

Travel Mode

Depart Time

Depart Airport

Arrival Time

Dist. to Airport

Flight Num

Parking Lot

Arrival Airport

Parking Rate

Mode to Airport

Taxi Fare

Partitioned Constraint Network
Template Hierarchy for the Travel Assistant

Trip
  1 AND
  ModeToDestination
    OR
    Drive OR Fly OR Taxi
  2 AND
    ModeHotel
    OR
    Hotel OR NoOvernight
  3 OR
    ModeNext
      OR
      Trip (Return Home) OR Trip (Return Office) OR Trip (New Leg) OR End Trip

ModeToAirport
  1 OR
  Drive OR Taxi

FlightDetail
  2 OR
  Drive OR Taxi

ModeFromAirport
  3 OR
  Drive OR Taxi
Dynamic Constraint Networks

Generalization of Constraint Networks

- Variables can be active or inactive
- Normal Constraints
  \[ x_1 = k_1 \land \ldots \land x_m = k_m \rightarrow x_n = k_n \]
- Activity constraints:
  \[ x_1 = k_1 \land \ldots \land x_m = k_m \rightarrow \text{active}(x_n) \]
- Inactive variables do not participate in the network, i.e., do not propagate constraints
Heracles: Template Selection

- Core network
  - Computes values of template selection vars
  - Always active

- Template selection variables
  - Inputs to activity constraints: determine the choice of subtemplates, i.e., which additional variables are active
Constraint Networks for Integrating Information

- Components:
  - Representation of the variables
  - Representation of constraints
  - Hierarchical template representation
  - Constraint propagation and cycle detection
Constraint Propagation

- **Approach**
  - When a variable is assigned a value, re-compute the value sets and assigned values of all dependent variables
  - Proceeds recursively until no values are changed or a cycle is detected

- **Core network**
  - Propagates all variables through the core network
  - Remaining variables are computing when a template is opened

- **Does not perform full CSP**
  - Less costly
  - Does not require all information in advance
  - Makes choices locally, so may fail to find optimal assignment
Cycle Detection

- Address cyclic *interactive* networks
  - Multiple input paths:
    - region/country/city vs. lat/long
  - Conversion rounding errors:
    - lat/long, temperature, ...

=> Cycle detection in constraint propagation
Interactive Cyclic Network

C3: Closest City

C1: Find City Center

C2: Copy

City (v1)

City center (v2)

Lat/Long (v3)
Interactive Cyclic Network

<table>
<thead>
<tr>
<th>Sequence of events</th>
<th>User time</th>
<th>Visited Variables</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>1</td>
<td>ø</td>
<td>LA</td>
</tr>
</tbody>
</table>

C1: Find City Center

C3: Closest City

C2: Copy

Lat/Long (v3)

City (v1)

City center (v2)
Interactive Cyclic Network

<table>
<thead>
<tr>
<th>Sequence of events</th>
<th>User time</th>
<th>Visited Variables</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>LA</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>v1</td>
<td>34N118W</td>
</tr>
</tbody>
</table>

C1: Find City Center
C2: Copy
C3: Closest City

City (v1) → City center (v2) → Lat/Long (v3)
Interactive Cyclic Network

<table>
<thead>
<tr>
<th>Sequence of events</th>
<th>User time</th>
<th>Visited Variables</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>1</td>
<td>Ø</td>
<td>LA</td>
</tr>
<tr>
<td>2)</td>
<td>1</td>
<td>v1</td>
<td>34N118W</td>
</tr>
<tr>
<td>3)</td>
<td>1</td>
<td>v1 v2</td>
<td>34N118W</td>
</tr>
</tbody>
</table>
C1: Find City Center

C2: Copy

C3: Closest City

City (v1)

City center (v2)

Lat/Long (v3)

Sequence of events | User time | Visited Variables | Value
---|---|---|---
1) | 1 | Ø | LA
4) | Blocked! | t(v3) = t(v1) \( \land v_1 \in \text{vis}(v3) \)
2) | 1 | v1 | 34N118W
3) | 1 | v1 v2 | 34N118W
Interactive Cyclic Network

<table>
<thead>
<tr>
<th>Sequence of events</th>
<th>User time</th>
<th>Visited Variables</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 1 Ø LA</td>
<td>1</td>
<td>0</td>
<td>LA</td>
</tr>
<tr>
<td>4) Blocked!</td>
<td></td>
<td>t(v3) = t(v1) ∧ v1 ∈ vis(v3)</td>
<td></td>
</tr>
<tr>
<td>2) 1 v1 34N118W</td>
<td>1</td>
<td>v1</td>
<td>34N118W</td>
</tr>
<tr>
<td>3) 1 v1 v2 34N118W</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5) 2 Ø 40N70W</td>
<td>2</td>
<td>0</td>
<td>40N70W</td>
</tr>
</tbody>
</table>
**Interactive Cyclic Network**

<table>
<thead>
<tr>
<th>Sequence of events</th>
<th>User time</th>
<th>Visited Variables</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 1 Ø LA</td>
<td>1</td>
<td>Ø</td>
<td>LA</td>
</tr>
<tr>
<td>2) 1 v1 34N118W</td>
<td>1</td>
<td>v1</td>
<td>34N118W</td>
</tr>
<tr>
<td>3) 1 v1 v2 34N118W</td>
<td>1</td>
<td>v1, v2</td>
<td>34N118W</td>
</tr>
<tr>
<td>4) Blocked! t(v3) = t(v1) ( \wedge v1 \in \text{vis}(v3) )</td>
<td>2</td>
<td>Ø</td>
<td>40N70W</td>
</tr>
<tr>
<td>6) 2 v3 NY</td>
<td>2</td>
<td>v3</td>
<td>NY</td>
</tr>
</tbody>
</table>

**Diagram:**

- **C1: Find City Center**
- **C2: Copy**
- **C3: Closest City**
C1: Find City Center

C2: Copy

C3: Closest City

City (v1) ➔ City center (v2) ➔ Lat/Long (v3)

Sequence of events | User time | Visited Variables | Value
--- | --- | --- | ---
1) | 1 | Ø | LA
4) | Blocked! | \( t(v3) = t(v1) \land v1 \in \text{vis}(v3) \)
6) | 2 | v3 | NY
2) | 1 | v1 | 34N118W
7) | 2 | v3, v1 | 40N73W
3) | 1 | v1, v2 | 34N118W
5) | 2 | Ø | 40N70W
**Interactive Cyclic Network**

<table>
<thead>
<tr>
<th>Sequence of events</th>
<th>User time</th>
<th>Visited Variables</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 1 Ø LA</td>
<td>1</td>
<td>∅</td>
<td>LA</td>
</tr>
<tr>
<td>4) Blocked!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6) 2 v3 NY</td>
<td>2</td>
<td>v3</td>
<td>NY</td>
</tr>
<tr>
<td>2) 1 v1 34N118W</td>
<td>1</td>
<td>v1</td>
<td>34N118W</td>
</tr>
<tr>
<td>7) 2 v3 v1 40N73W</td>
<td>2</td>
<td>v3 v1</td>
<td>40N73W</td>
</tr>
<tr>
<td>3) 1 v1 v2 34N118W</td>
<td>1</td>
<td>v1 v2</td>
<td>34N118W</td>
</tr>
<tr>
<td>5) 2 Ø 40N70W</td>
<td>2</td>
<td>Ø</td>
<td>40N70W</td>
</tr>
<tr>
<td>8) Blocked!</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
    t(v3) = t(v1) \land v1 \in \text{vis}(v3)
\]

\[
    t(v3) = t(v2) \land v3 \in \text{vis}(v2)
\]
Discussion

- General framework for interleaving planning and information gathering
  - Retrieves information as needed
  - Gathers and integrates data in a uniform framework
  - Evaluates tradeoffs and selects among alternatives
  - Allows the users to explore alternatives
  - Supports a wide variety of information types: databases, web pages, images, video, etc.