Chemistry Studio
An Intelligent Tutoring System: Problem Solving

B.Tech Project
Ashish Gupta (Y8140)
Akshay Mittal (Y8056)

Mentored By:
Prof. Amey Karkare, IIT Kanpur
Dr. Sumit Gulwani, MSR Redmond
Dr. Ashish Tiwari, SRI
Objective

- Building a system aimed at helping a student in their learning process
- Generate solutions and explanations in accordance with the interest and knowledge of the student.
- Hint Generation and Problem Generation

**Target Users:** High school students (Grade 9 to 12)
Basic Intuition

- **Scenario:** A professor gives a problem in the domain of Periodic Table to the class.

- **Students Response**
  - Pre-knowledge of Basic Facts
  - Theorems & Relations

- **System Response**
  - In-built knowledge of Basic Facts
  - Breaks the complex into smaller problems
  - Builds up explanation (using *rules*)

- System tries to simulate the thinking of the student and is in a better position to guide the student.
Intermediate Logic Component

- Identification of major entities in the Periodic Table
- Intermediate representation between the NLP and Problem Solving component.
- Terms: Predicates, Functions, Variables

<table>
<thead>
<tr>
<th>Unary Predicates</th>
<th>Unary Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alkali Metals</td>
<td>First Ionization Energy</td>
</tr>
<tr>
<td>Alkaline Earth Metals</td>
<td>Atomic Radius</td>
</tr>
<tr>
<td>Transition Metals</td>
<td>Ionic Radius</td>
</tr>
<tr>
<td>Metalloids</td>
<td>Atomic Number</td>
</tr>
<tr>
<td>Non-Metals</td>
<td>Group Number</td>
</tr>
<tr>
<td>Halogens</td>
<td>Metallic Character</td>
</tr>
</tbody>
</table>
Prolog Database

- Construct a set of facts and rules in the logic programming language Prolog
- Build a directed dependency graph
  - Nodes – Predicates
  - Edges – Dependency between predicates
Prolog Database (contd.)

- Basic Facts about each element
- For each predicate, discover its relation with all possible predicates and facts

<table>
<thead>
<tr>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>data(atomicNumber, element, groupNumber, periodNumber, atomicMass, FIE, EA)</td>
</tr>
<tr>
<td>data(1, 'H', 1, 1, 1.008, 1312, 72)</td>
</tr>
<tr>
<td>data(2, 'He', 18, 1, 4.003, 2372, 0)</td>
</tr>
<tr>
<td>data(3, 'Li', 1, 2, 6.941, 520, 60)</td>
</tr>
</tbody>
</table>

1) firstIonizationEnergy(X, Y, '+') :- trendPeriod(X, Y, '='), !, trendGroup(X, Y, '+'), !.
2) firstIonizationEnergy(X, Y, '+'):- trendGroup(X, Y, '='), trendPeriod(X, Y, '-'), !.
Yield Prolog (YP)

- Yield Prolog is an implementation that enables Prolog to be directly embedded into C#.

- Used as an interface to process the input XML file, query the prolog database and outputting the result.

- Advantage: Unifies the power of procedural and declarative programming.
  - No API standing between our code and Yield Prolog
  - Mix Prolog-style predicates with ordinary arrays, file I/O, GUI calls and all your own classes

- Generation of Prolog Database to YP using YP Compiler
Problem Solving Component

- XML File Generation
- Prolog Goal Generation
- Yield Prolog (C#) Goal Generation
- Compile & Execution of Goal Code
- Derivation Tree for Solution
Recursive unification of child nodes

And of clauses is represented as a comma-separated list in the prolog goal

Example-

- Same(Group(Ca), $1)
- And(And(Same(2, Group($1)), Same(3, Period($1))), (AtomicNumber($1), $2))

Demo: Questions 1-5
Max/Min Query Planning

- Many questions (a *property* & a *domain*)
- Challenge: Extra conditions to be satisfied
- Passed as *Filter* conditions
- Restrict Domain (happens in *Interface*)

*Demo:* Questions 12-14
ForAll Query Planning

- **Trivial Approach**
  - Scan all elements of Periodic Table to find binding

- **Our Approach**
  - Assert the antecedents with new constants
  - Satisfy the goal with consequents containing new constants

### Database

- $A(x), A2(x) \rightarrow P(x)$
- $A1(x), A2(x) \rightarrow R(x)$
- $R(x) \rightarrow Q(x)$
- $P(x), Q(x) \rightarrow B(x)$

### Goal

- $\forall x : A(x), A1(x), A2(x) \rightarrow B(x)$
- **Assertions**
  - $A(a), A1(a), A2(a)$
- **New Goal**
  - $B(a)$

Demo: Questions 15
Trend Query Planning

- Properties vary along a direction
- Encapsulate directions
  - Basic knowledge which a student has related to a direction
  - Assert that knowledge and retract after solving
- Find the varying trend or property or direction

Demo: Questions 6-9

```prolog
assertFact('down') :- assertz(trendAtomicNumber('p', 'q', '-')), assertz(trendGroup('p', 'q', '=')).
assertFact('right') :- assertz(trendAtomicNumber('p', 'q', '-')), assertz(trendPeriod('p', 'q', '=')).

retractFact('down') :- retract(trendAtomicNumber('p', 'q', '-')), retract(trendGroup('p', 'q', '=')).
retractFact('right') :- retract(trendAtomicNumber('p', 'q', '-')), retract(trendPeriod('p', 'q', '=')).
```
Order Querying

- Rearrange the elements w.r.t. the value of a certain property
- Merge Sort

*Demo: Questions 10-11*
Challenges Faced

• Combining two or more prolog goals to build a more general unified goal

• Understanding the intricacies of the YP compiler
  ◦ Asserted rules and facts treated in separate predicate store as the existing predicates
  ◦ Dynamic passing of predicates in the yield prolog environment not supported
  ◦ Successfully modified the compiler to fix the above bugs

• On the fly compilation of the dynamically generated code from PHP script
Statistical Evaluation

- **Evaluation Metric:** Number of questions solved by the system versus the number of predicate rules defined.

- Solving 70 questions out of the 100 collected questions.
  - Using existing 5 templates and approximately 20 predicate rules
  - Collected Set is not representative of the entire domain

- As the predicate rules and templates increase, the system would be able to achieve higher question solving ability

- In future, proof-length with *ease of understanding* can be used to gauge the efficacy
Derivation Tree

- System emulates the thinking of the student

- At each step of derivation, the logical reasoning for that step can be accumulated to give an overall reasoning.

- Can have multiple successful derivations for a single goal
  - Represent different solutions to a particular problem
  - Important to let the user decide which solution to prefer

- Use the existing knowledge-base of the user to simulate which solutions would be more attractive to the user.
  - For example: Solution using Concept X and Concept Y
  - Give the user the choice to see any of the derivation
Future Work

- Hint Generation
- Efficient Query Planning
- Problem Generation
  - Use the depth of the proof as a heuristic to measure hardness
  - Assigning certain initial predicates a score of hardness
  - Use few basic templates and conjunction operators
- Future Interactive System
  - Modeling the initial knowledge of the student
- Handling Exceptions and Conflicting Rules
- Similarity Metric
  - Variation within an $\varepsilon$-range
Thank you …

Questions ?