Lists

- Constructors
  - [] Empty list constant
  - . Constructor functor

- Example
  - .(a, .(b, .(c, [])))
  - [a, b, c] (syntactic sugar)

- Tail notation:
  - [a | [b, c]]
  - [a, b | [c]]

  Head::a Tail::[a]
Lists

Examples

- \texttt{member(X, [X | T])}.
- \texttt{member(X, [H | T]) \leftarrow member(X, T)}.

- \texttt{sorted([], \% empty list is sorted}.
- \texttt{sorted([X]), \% singleton is sorted}
- \texttt{sorted([A, B | T]) \leftarrow A =< B, \text{sorted}([B | T]).}

\% compound list is sorted if first two elements are in order and \% remainder of list (after first element) is sorted

- \texttt{append([], A, A).}
- \texttt{append([L | T], A, [H | L]) \leftarrow append(T, A, L).}

\% no notion of input or output parameters

Tic-Tac-Toe Example

- 3x3 grid
- Two Players:
  - X (computer)
  - O (human)
- Fact \texttt{X (n)} indicates a movement by X
  - E.g. \texttt{X(5), X(9)}
- Fact \texttt{O (n)} indicates a movement by O
  - E.g. \texttt{O(1), O(6)}
Tic-Tac-Toe Example

• Winning condition

<table>
<thead>
<tr>
<th>ordered_line(1, 2, 3). ordered_line(4, 5, 6).</th>
</tr>
</thead>
<tbody>
<tr>
<td>ordered_line(7, 8, 9). ordered_line(1, 4, 7).</td>
</tr>
<tr>
<td>ordered_line(2, 5, 8). ordered_line(3, 5, 9).</td>
</tr>
<tr>
<td>ordered_line(1, 5, 9). ordered_line(3, 5, 7).</td>
</tr>
<tr>
<td>line(A, B, C) :- ordered_line(A, B, C).</td>
</tr>
<tr>
<td>line(A, B, C) :- ordered_line(A, C, B).</td>
</tr>
<tr>
<td>line(A, B, C) :- ordered_line(B, A, C).</td>
</tr>
<tr>
<td>line(A, B, C) :- ordered_line(B, C, A).</td>
</tr>
<tr>
<td>line(A, B, C) :- ordered_line(C, A, B).</td>
</tr>
<tr>
<td>line(A, B, C) :- ordered_line(C, B, A).</td>
</tr>
</tbody>
</table>

Tic-Tac-Toe Example

move(A) :- good(A), empty(A).

\[ \text{Strategy: good moves} \]

full(A) :- x(A).
full(A) :- o(A).
empty(A) :- not full(A).

\% strategy:

1. good(A) :- win(A).
2. good(A) :- block_win(A).
3. good(A) :- split(A).
4. good(A) :- block_split(A).
5. good(A) :- build(A).

{ Ordered List of Choices }
Tic-Tac-Toe Example

1. \( \text{win}(A) :- x(B), x(C), \text{line}(A, B, C). \)
2. \( \text{block_win}(A) :- o(B), o(C), \text{line}(A, B, C). \)
3. \( \text{split}(A) :- x(B), x(C), \text{different}(B, C), \)
   \( \quad \text{line}(A, B, D), \text{line}(A, C, E), \text{empty}(D), \text{empty}(E). \)
   \( \text{same}(A, A). \)
   \( \text{different}(A, B) :- \text{not same}(A, B). \)
4. \( \text{block_split}(A) :- o(B), o(C), \text{different}(B, C), \)
   \( \quad \text{line}(A, B, D), \text{line}(A, C, E), \text{empty}(D), \text{empty}(E). \)
5. \( \text{build}(A) :- x(B), \text{line}(A, B, C), \text{empty}(C). \)

\[
\begin{array}{c|c|c|c}
1 & 2 & 3 \\
4 & x & 0 \\
7 & 8 & x \\
\end{array}
\]

Imperative Control Flow

The cut

- Prolog has a number of explicit control flow features
- Known as the cut
  - This is a zero-argument predicate that always succeeds
  - It commits the interpreter to the unification made between the parent goal and the left-hand side of the current rules

- Example

\[
\begin{align*}
\text{member}(X, [X|T]) . \\
\text{member}(X, [H|T]) & :- \text{member}(X, T) . \\
\text{member}(X, [X|T]) & :- ! . \\
\text{member}(X, [H|T]) & :- \text{member}(X, T) .
\end{align*}
\]

If this rule succeeded, do not try to use the following ones
Imperative Control Flow

- Alternative
  \[
  \text{member}(X, [X|T]) \quad \text{member}(X, [H|T]) : \text{not}(X=H), \text{member}(X, T).
  \]

- How does \text{not} work?
  \[
  \text{not}(P) \quad \text{not}(P) : \text{call}(P), !, \text{fail}.
  \]

  - \text{call} attempts to satisfy the goal \(P\).
  - \text{fail} always fails.

Prolog Database Manipulation

- Two built-in predicates can be used to modify the database of known facts
  - \text{assert}(P) adds a new fact.
    - \text{E.g. assert(parent(kevin, john))}
  - \text{retract}(P) removes a known fact.
    - \text{E.g. retract(parent(kevin, john))}
Backward Chaining in Prolog

- Backward chaining follows a classic depth-first backtracking algorithm.
- Example
  - Goal: Snowy(C)

Infinite Regression

edge(a, b), edge(b, c), edge(c, d),
edge(d, e), edge(b, e), edge(d, f),
path(X, Y) :- path(X, Z), edge(Z, Y).
path(X, X).

Goal

\[
\begin{align*}
X_1 &= a, Y_1 = a \\
X_2 &= X_1, Y_2 = Y_1, Z_1 = ? \\
X_3 &= X_2, Y_3 = Y_2, Z_3 = ? \\
&\ldots
\end{align*}
\]
• Read
  – Rest of Scott Sect. 11.3.1

• *Guide to Prolog Example*, Roman Barták
  – Go through all the examples