CENG 424

Logic For Computer Science

Fall 2024-2025

Assignment 5

Lök, Yusuf Sami

e2521748@ceng.metu.edu.tr

January 19, 2025

Question 1

Relation Definitions

transition(Q_0, S, Q_1, W): This relation represents a transition from state Q_0 to Q_1 with weight W by consuming symbol S.

path(Q_e, T, S, W): This relation represents the existence of a path with a list of transitions T, ending at state Q_e, with remaining string S, and cost of weight W.

Relational Logic Formulation

1. **Initialize relational logic items:**
   Create an empty list, `ListOfRelationalLogicItems`, to store all relational logic items.

2. **Initialize the starting path:**
   Define the starting path with the initial state s and the input string S:

$$\texttt{path}(s, \text{nil}, S, 0)$$

3. **Define transitions:**
   For each transition t in δ (transition relation of the NWFSM):

   (a) Find the corresponding weight w of transition t using w in the NWFSM.

   (b) Add the transition to `ListOfRelationalLogicItems`:

$$\texttt{transition}(Q_0, S, Q_1, w)$$

4. **Add the path extension rule:**
   If there exists a transition from Q_0 to Q_1 with symbol `sym` and weight w_0, and a path from Q_0 with remaining string `sym.r` and weight w, then the following relation holds:

$$\texttt{transition}(Q_0, \texttt{sym}, Q_1, w_0) \wedge \texttt{path}(Q_0, P_0, \texttt{sym.r}, w) \implies \texttt{path}(Q_1, P_0 \cup \{(Q_0, \texttt{sym}, Q_1)\}, \texttt{r}, w_0 + w)$$

5. **Add the goal statement:**
   The goal is defined as the path ending at a state Q with an empty remaining string and the highest total weight W:

$$\texttt{goal}(P, W) \iff \texttt{path}(Q, P, \texttt{nil}, W) \wedge \forall_{Q'} \forall_{P'} \big(\texttt{path}(Q', P', \texttt{nil}, W') \implies W \geq W'\big)$$

6. **Return the list:**
   Return `ListOfRelationalLogicItems`.

Question 2

Finding the Highest Weighted Path for abb on NWFSM N_1

1. **Add all transitions:**
   From the given NWFSM N_1, the transitions are:

$$\begin{array}{rl} \texttt{transition}(q_0, \texttt{a}, q_1, 4), & \texttt{transition}(q_0, \texttt{a}, q_3, 3), \\ \texttt{transition}(q_0, \texttt{b}, q_2, 2), & \texttt{transition}(q_0, \texttt{b}, q_3, 4), \\ \texttt{transition}(q_1, \epsilon, q_3, 2), & \texttt{transition}(q_1, \texttt{b}, q_1, 4), \\ \texttt{transition}(q_2, \texttt{a}, q_0, 2), & \texttt{transition}(q_2, \texttt{a}, q_3, 4), \\ \texttt{transition}(q_3, \epsilon, q_2, 7), & \texttt{transition}(q_3, \texttt{b}, q_1, 1) \end{array}$$

2. **Add initial path:**
   Define the starting path:

$$\texttt{path}(q_0, \text{nil}, \texttt{abb}, 0)$$

3. **Add path extension rule:**
   Using the transitions defined, apply the path extension rule:

$$\texttt{transition}(Q_0, \texttt{sym}, Q_1, w_0) \wedge \texttt{path}(Q_0, P_0, \texttt{sym.r}, w) \implies \texttt{path}(Q_1, P_0 \cup \{(Q_0, \texttt{sym}, Q_1)\}, \texttt{r}, w_0 + w)$$

4. **Process the string abb using resolution:**

   Premises:

$$\text{R1: } \texttt{transition}(Q_0, \texttt{sym}, Q_1, w_0) \wedge \texttt{path}(Q_0, P_0, \texttt{sym.r}, w) \quad \implies \texttt{path}(Q_1, P_0 \cup (Q_0, \texttt{sym}, Q_1), \texttt{r}, w_0 + w)$$

   Goal rule:

$$\begin{aligned} \text{G1: } \texttt{goal}(P, W) \iff & \texttt{path}(Q, P, \texttt{nil}, W) \wedge \\ & \forall Q' \forall P' \left(\texttt{path}(Q', P', \texttt{nil}, W') \implies W \geq W'\right) \end{aligned}$$

   Derivation:

   1. $\texttt{transition}(q_0, \texttt{a}, q_1, 4)$ [Premise]
   2. $\texttt{transition}(q_0, \texttt{a}, q_3, 3)$ [Premise]
   3. $\texttt{transition}(q_1, \texttt{b}, q_1, 4)$ [Premise]
   4. $\texttt{transition}(q_3, \texttt{b}, q_1, 1)$ [Premise]
   5. $\texttt{transition}(q_1, \epsilon, q_3, 2)$ [Premise]
   6. $\texttt{transition}(q_3, \epsilon, q_2, 7)$ [Premise]
   7. $\texttt{path}(q_0, \text{nil}, \texttt{abb}, 0)$ [Initial fact]
   8. $\texttt{path}(q_1, (q_0, \texttt{a}, q_1), \texttt{bb}, 4)$ [R1: 7,1]
   9. $\texttt{path}(q_3, (q_0, \texttt{a}, q_3), \texttt{bb}, 3)$ [R1: 7,2]
   10. $\texttt{path}(q_1, (q_0, \texttt{a}, q_1), (q_1, \texttt{b}, q_1), \texttt{b}, 8)$ [R1: 8,3]
   11. $\texttt{path}(q_1, (q_0, \texttt{a}, q_3), (q_3, \texttt{b}, q_1), \texttt{b}, 4)$ [R1: 9,4]
   12. $\texttt{path}(q_1, P_1, \texttt{nil}, 12)$ [R1: 10,3] where $P_1 = (q_0, \texttt{a}, q_1), (q_1, \texttt{b}, q_1), (q_1, \texttt{b}, q_1)$
   13. $\texttt{path}(q_1, P_2, \texttt{nil}, 8)$ [R1: 11,3] where $P_2 = (q_0, \texttt{a}, q_3), (q_3, \texttt{b}, q_1), (q_1, \texttt{b}, q_1)$
   14. $\texttt{path}(q_3, P_3, \texttt{nil}, 14)$ [R1: 12,5] where $P_3 = P_1 \cup (q_1, \epsilon, q_3)$
   15. $\texttt{path}(q_2, P_4, \texttt{nil}, 21)$ [R1: 14,6] where $P_4 = P_3 \cup (q_3, \epsilon, q_2)$

   Weight Derivation using Goal Rule:

   Let's compare all paths ending with nil:

   (a) $\texttt{path}(q_1, P_1, \texttt{nil}, 12)$ from line 12

   (b) $\texttt{path}(q_1, P_2, \texttt{nil}, 8)$ from line 13

(c) $\texttt{path}(q_3, P_3, \texttt{nil}, 14)$ from line 14

(d) $\texttt{path}(q_2, P_4, \texttt{nil}, 21)$ from line 15

Applying G1, comparing weights: $21 > 14 > 12 > 8$

Therefore: $\texttt{goal}(P_4, 21)$

The highest weighted path is: $(q_0, \texttt{a}, q_1), (q_1, \texttt{b}, q_1), (q_1, \texttt{b}, q_1), (q_1, \epsilon, q_3), (q_3, \epsilon, q_2)$ with total weight 21.

Question 3

I have chosen the Prolog programming language for this purpose.

1. Changes to the Output Format:

- The transitions should be represented as facts, such as:

```
transition(q0, a, q1, 4).
```

- The paths and weight relations should be defined as rules (using `:-` in Prolog) that can be recursively processed to evaluate possible paths.
- For empty transitions, a separate rule needs to be defined to change states without consuming symbols.
- The goal condition must be written as a rule to select the maximum weight path from all possible solutions.

2. How Logic Programming Tools Work:

Logic programming tools, such as Prolog, use a declarative approach, where the problem is expressed in terms of relations and logical rules. These tools automatically:

- Use facts and rules to generate all possible solutions.
- Apply backward and forward chaining to explore different combinations of transitions and paths.
- Solve the problem by searching for a path that satisfies the goal condition.
- Prolog uses a depth-first search (DFS) strategy to traverse the possible paths, and the goal condition is evaluated by unifying the paths and comparing their weights.