CENG424

Logic for Computer Science

Assignment 5 - Term Project

Zeynep Sıla Egül
e2380335@ceng.metu.edu.tr

Question 1

Pseudocode of the algorithm in the Relational Logic:

Relations:
State(state_id)
Symbol(symbol)
Transition(from_state, symbol, to_state, weight)
StringLength(n)
InitialState(state)
Path(position, current_state, accumulated_weight)
InputSymbol(position, symbol)

Rules:
Path(0, s, 0) :- InitialState(s)
Path(pos, to_state, new_weight) :-
    Path(pos, from_state, weight),
    Transition(from_state, ε, to_state, trans_weight),
    new_weight = weight + trans_weight,
    StringLength(len),
    pos <= len.
Path(pos+1, to_state, new_weight) :-
    Path(pos, from_state, weight),
    InputSymbol(pos, sym),
    Transition(from_state, sym, to_state, trans_weight),
    new_weight = weight + trans_weight,
    StringLength(len),
    pos < len.
MaxPath(weight) :-
    Path(final_pos, state, weight),
    StringLength(final_pos),
    NOT EXISTS p2, s2, w2: (
        Path(final_pos, s2, w2),
        w2 > weight
)

Question 2

Initialization:
InputSymbol(0, 'a')
InputSymbol(1, 'b')
InputSymbol(2, 'b')
StringLength(3)
State(q_0), State(q_1), State(q_2), State(q_3)
InitialState(q_0)
Symbol(a), Symbol(b), Symbol(ε)
Transition(q_0, a, q_1, 4)
Transition(q_0, b, q_2, 2)
Transition(q_0, a, q_3, 3)

Transition(q_0, b, q_3, 4)
Transition(q_1, b, q_1, 4)
Transition(q_1, ε, q_3, 2)
Transition(q_2, a, q_0, 2)
Transition(q_2, a, q_3, 4)
Transition(q_3, ε, q_2, 7)
Transition(q_3, b, q_1, 1)

Initial state:
Path(0, q_0, 0) :- InitialState(q_0)

InputSymbol(0, 'a') and/or ε:
Path1 (a):
Path(1, q_1, 4):
    Path(0, q_0, 0),
    InputSymbol(0, a),
    Transition(q_0, a, q_1, 4),
    new_weight = 0+4

Path2 (a):
Path(1, q_3, 3):
    Path(0, q_0, 0),
    InputSymbol(0, a),
    Transition(q_0, a, q_3, 3),
    new_weight = 0+3

InputSymbol(1, 'b') and/or ε:
Path1.1 (ab):
Path(2, q_1, 8):
    Path(1, q_1, 4),
    InputSymbol(1, b),
    Transition(q_1, b, q_1, 4),
    new_weight = 4+4

Path1.2.1 (aε):
Path(1, q_3, 6):
    Path(1, q_1, 4),
    Transition(q_1, ε, q_3, 2),
    new_weight = 4+2

Path1.2.2 (aεb):
Path(2, q_1, 7):
    Path(1, q_3, 6),
    InputSymbol(1, b),
    Transition(q_3, b, q_1, 1),
    new_weight = 6+1

Path2.1 (ab):
Path(2, q_1, 4):
    Path(1, q_3, 3),
    InputSymbol(1, b),
    Transition(q_3, b, q_1, 1),
    new_weight = 3+1

InputSymbol(2, 'b') and/or ε:
Path1.1.1 (abb):
Path(3, q_1, 12):
    Path(2, q_1, 4),
    InputSymbol(2, b),
    Transition(q_1, b, q_1, 4),
    new_weight = 8+4

Path1.1.2.1 (abε):
Path(2, q_3, 10):
Path(2, q_1, 8),
Transition(q_1, ε, q_3, 2),
new_weight = 8+2

Path1.2.2 (abεb):
Path(3, q_1, 11):
Path(2, q_3, 10),
Transition(q_3, b, q_1, 1),
new_weight = 10+1

Path1.2.2.1 (aεbb):
Path(3, q_1, 11):
Path(2, q_1, 7),
InputSymbol(2, b),
Transition(q_1, b, q_1, 4),
new_weight = 7+4

Path1.2.2.2.1 (aεbε):
Path(2, q_3, 9):
Path(2, q_1, 7),
Transition(q_1, ε, q_3, 2),
new_weight = 7+2

Path1.2.2.2.2 (aεbεb):
Path(3, q_1, 10):
Path(2, q_3, 9),
InputSymbol(2, b),
Transition(q_3, b, q_1, 1),
new_weight = 9+1

Path2.1.1 (abb):
Path(3, q_1, 8):
Path(2, q_1, 4),
InputSymbol(2, b),
Transition(q_1, b, q_1, 4),
new_weight = 4+4

Path2.1.2.1 (abε):
Path(2, q_3, 6):
Path(2, q_1, 2),
Transition(q_1, ε, q_3, 2),
new_weight = 4+2

Path2.1.2.2 (abεb):
Path(3, q_1, 7):
Path(2, q_1, 1),
InputSymbol(2, b),
Transition(q_3, b, q_1, 1),
new_weight = 6+1

Finding the maximum weighted path:
Path(3, q_1, 12) - abb
Path(3, q_1, 11) - abεb
Path(3, q_1, 11) - aεbb
Path(3, q_1, 10) - aεbεb
Path(3, q_1, 8) - abb
Path(3, q_1, 7) - abεb

```
MaxPath(12) :-
    Path(3, q1, 12),
    StringLength(3),
    NOT EXISTS p2, s2, w2: (
        Path(3, s2, w2),
        w2 > 12
    )
```

Question 3

1.

I need to convert Relational Logic relations into Prolog facts and Relational Rules into Prolog clauses. I also need to use Prolog's built-in predicates such as findall and max_list.

2.

These tools translate the Relational Logic representation into their internal format, where facts (relations) become base knowledge and rules become logical implications or constraints. They systematically explore the solutions defined by the logical rules through pattern matching between rules and facts, handle recursive definitions, and apply optimization criteria. The tools can automatically deduce new facts from existing ones using the provided rules, handle negation and aggregation, and efficiently maintain derived information. This automated reasoning process solves complex problems like path-finding task by processing the logical representation without requiring explicit algorithmic instructions.