Einführung in die Computerlinguistik – Endliche Automaten und Transduktoren in Prolog

Dozentin: Wiebke Petersen

17.6.2010

Prolog: the basics

- facts: state things that are unconditionally true of the domain of interest.

  ```
  human(sokrates).
  ```

- rules: relate facts by logical implications.

  ```
  mortal(X) :- human(X).
  ```

  - head: left hand side of a rule
  - body: right hand side of a rule
  - clause: rule or fact.
  - predicate: collection of clauses with identical heads.

- knowledge base: set of facts and rules
- queries: make the Prolog inference engine try to deduce a positive answer from the information contained in the knowledge base.

  ```
  ?- mortal(sokrates).
  ```

Prolog: some syntax

- facts: `fact.`
- rules: `head :- body.`
- conjunction: `head :- info1 , info2.`
- atoms start with small letters
- variables start with capital letters

Learning material: `http://www.learnprolognow.org/`

Übungsaufgabe

Nehmen Sie das Prolog-Programm `family.pl` von der Homepage. Es implementiert eine kleine Familiendatenbank. Erweitern Sie es um die Prädikate `grandfather/2`, `child/2` und `cousin/2`.

lists in Prolog

- Lists are recursive data structures: First, the empty list is a list. Second, a complex term is a list if it consists of two items, the first of which is a term (called first), and the second of which is a list (called rest).
- `[mary|[john|[alex|[tom|[]]]]]`
- simpler notation: `[mary,john,alex,tom]`
- Exercise: Write a predicate `member/2`.

Algorithms for finite automatons

```
function D-RECOGNIZE (tape, machine) returns accept or reject
  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
    if End of input has been reached then
      if current-state is an accept state then
        return accept
      else
        return reject
    elsif transition-table [current-state, tape[index]] is empty then
      return reject
   else
     current-state ← transition-table [current-state, tape[index]]
     index ← index + 1
end
```

Algorithms for finite automatons

function D-RECOGNIZE (*tape, machine*) **returns** accept or reject
  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
    if End of input has been reached **then**
      if current-state is an accept state **then**
        return accept
      else
        return reject
    elsif *transition-table [current-state, tape[index]]* is empty **then**
      return reject
    else
      current-state ← *transition-table [current-state, tape[index]]*
      index ← *index* + 1
end

```
% Finite state automaton.

fsa(Tape):-
    initial(S),
    fsa(Tape,S).

fsa([],S):- final(S).

fsa([H|T],S):-
    trans_tab(S,H,NS),
    fsa(T,NS).

% FSA transition table:
% trans_tab/3
% trans_tab(State, Input, New State)

trans_tab(1,a,1).
trans_tab(1,b,2).
trans_tab(2,a,2).

initial(1).
final(2).
```

Beispielprogramme (Homepage)

- `fsa.pl` Endlicher Automat, der die Sprache a^*ba^* erkennt. Aufruf mit `fsa(Eingabeliste).` (Bsp. `fsa([a,b,b,b,a]).`).
- `fst.pl` Endlicher Transduktor, der in Wörtern über dem Alphabet $\{a, b\}$ alle a's durch b's und alle b's durch a's ersetzt. Aufruf mit `fst(Eingabeliste,Ausgabelist).` (Bsp. `fst([a,b,b],[b,a,a]).`, oder `fst(L,[b,a,a]).` oder `fst([a,b,b],L).`). Das Programm erlaubt weder ϵ zu schreiben, noch ϵ zu lesen. Nichtdeterministische Transduktoren sind aber möglich. Erweitern Sie den Transduktor zum Beispiel um den Übergang `trans_tab_fst(1,a,c,1).`.
- `fst_2.pl` Endlicher Transduktor von Folie 16 vom 14.6.2010. Aufruf mit `fst(Eingabeliste,Ausgabeliste).` (Bsp. `fst([f,o,x,;s,#],L).`). Dieses Programm ist in der Lage auch Transduktoren mit ϵ-Übergängen zu verarbeiten. Versuchen Sie den Transduktor von Folie 17 vom 14.06.2010 zu programmieren. Sie müssen nur die Faktenbasis (Menge der Fakten ändern.