VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Formalizace konstrukcí na konečných automatech v programu COQ

Formalization of structures on finite state machines in the COQ program

Bc. Petr Kožušník

Vedoucí práce: doc. Ing. Zdeněk Sawa, Ph.D.

Ostrava, 2022

Abstrakt

Práce demonstruje možný způsob formalizace automatů v jazyce COQ.

Klíčová slova

jazyk coq, deterministický konečný automat, formalizace

Abstract

The work demonstrates a possible way of formalization of automata in the COQ language.

Keywords

coq language, deterministic finite automaton, formalization

Obsah

Kapitola 1

Úvod

Coq je softwarový nástroj pro vytváření a ověřování formálních důkazů. Dokazovaná tvrzení a jednotlivé kroky důkazu se zapisují způsobem, který je v mnoha ohledech podobný zápisu programů v programovacím jazyce.

Cílem tohoto projektu je formalizovat některé základní konstrukce na konečných automatech (např. převod nedeterministického automatu na deterministický, konstrukce automatů pro různé jazykové operace, apod.) právě pomocí nástroje coq a následně jím také ověřit korektnost.

Pro pochopení způsobu formalizace automatu, je nezbytná alespoň zdánlivá znalost jazyka coq a jeho principů. Druhá kapitola je tak věnována úvodu do fungování tohoto dokazovacího systému.

Kapitola 2

Coq

Coq je imperativní programovací jazyk, který slouží zejména pro dokazování matematických tvrzení. Coq je rovněž označován jako takzvaný *proof assistant*, je tak zřejmé, že tento jazyk nutně obsahuje konstrukce pro zápis výroků.

Pro zachování korektnosti matematického dokazování je jasné, že nelze jakoukoliv formuli libovolně prohlásit za platnou.

Výroky je nutno dokazovat pomocí úsudků jíž dříve dokázaných. Samotný coq sám o sobě implicitně nezaručuje platnost ani význam běžně užívaných výrazů.

Ve standardní knihovně tak můžeme najít například definici záporu.

```
Definition not (A:Prop) := A -> False.
```

Nebo číselných množin.

```
Inductive nat : Set :=
| 0 : nat
| S : nat -> nat.
```

Poněkud zajímavější konstrukcí jsou takzvaný induktivní predikát, pomocí kterého je definována například sudost.

```
Inductive even : N -> Prop :=
  | even_0 : even 0
  | even_S n : odd n -> even (n + 1)
  with odd : N -> Prop :=
  | odd_S n : even n -> odd (n + 1)
```

Kapitola 3

Formalizace automatu

Tato kapitola ukazuje možný způsob formalizace automatu. Současně je nutné definovat pojmy s automaty související, jako je abeceda nebo jazyk.

3.1 Deterministický konečný automat

Automat je abstraktní výpočetní stroj, který je využíván v oblasti teoretické informatiky a diskrétní matematiky.

Automat se obecně sestává ze stavů a přechodů. Tedy z kolekce určitých prvků, kdy některý z nich je právě v kontextu a zákonitosti

Deterministický konečný automat je běžně definován jako následující uspořádaná pětice.
$(S, \Sigma, \sigma, s, F)$

- Q - konečná neprázdná množina stavů
- Σ - konečná neprázdná množina vstupních symbolů - abeceda
- $q0$ - počáteční stav, $q0 \in Q$
- F - množina koncových stavů, $F \subseteq Q$
- σ - přechodová funkce, $Q \times \Sigma \mapsto S$

Tuto pětice v coqu deklarujeme pomocí struktury.

```
(* Deterministic Finite Automaton *)
Structure DFA  (Sigma:Alphabet) : Type := createDFA {
                                Q:      Set;
                                delta:  Q -> Sigma -> Q;
                                q0:     Q;
                                F:      Q -> bool
}.
```

Dalším pojmem z této domény je jazyk, tedy konečná množina řetězců nad určitou abecedou.

Formálně tedy zapisujeme, že o konkrétních slovech (řetězcích) z abecedy Sigma lze rozhodnout zda do o jazyka patří, či nikoliv.

Nutno připomenout, že mapping na Prop, označuje výrok, tedy typ, který lze pomocí logických spojek skládat do formulí.

Kapitola 4

Operace nad automaty

```
Lemma transition_nil: forall q1 q2, transition q1 nil q2 <-> q1 = q2.
```

Tato sekce ukazuje možný formální zápis množinových operací nad automaty.

4.1 Sjednocení automatů

Automat vzniklý sjednocením automatů jiných je takový, který přijímá taková slova, jež jsou přijímána alespoň jedním z nich. Tedy sjednocením automatu A1 rozpoznávající jazyk L1 a automatu A2, rozpoznávající jazyk L2, vznikne automat A, který rozpoznává slova jež patří do jazyka L1 anebo L2.

Konstrukce takového automatu je následující. Počátečním stavem A je uspořádaná dvojice počátečních stavů A1, A2.

```
Let q0_a := q0 Sigma Automaton_a.
Let q0_b := q0 Sigma Automaton_b.
Let q0 := pair q0_a q0_b.
```

Přechodová funkce.

```
Definition delta_union (q: Q)(a: Sigma) : Q :=
  let (q1, q2) := q in
  let q1':= delta_a q1 a in
  let q2':= delta_b q2 a in
    pair q1' q2'.
```

Koncový stav tvoří takové uspořádané dvojice, kdy některý členů je koncovým stavem předešlého automatu.

```
Definition qend_union (q: Q) : bool :=
  let (q1, q2) := q in
  orb (qend_a q1)(qend_b q2).
```

Samotný sjednocený automat se pak zkonstruuje zápisem.

```
Definition Union_automata: DFA Sigma :=
  createDFA Sigma Q delta_union q0 qend_union.
```

Konstrukce automatu průniku by pak nebyla příliš odlišná. Ostatně jediným rozdílem je množina konečných stavů. Kdy oba prvky z uspořádané dvojice musí být koncevým stavem.

```
Definition Intersection_automata (a: DFA Sigma)(b: DFA Sigma) : DFA Sigma :=
  createDFA Sigma Q delta_union q0 qend_intersection.
```

Příkladem složitější konstrukce může být následující formule $A1$ přejde slovem w ze stavu $q1$ do stavu $q1'$ a zároveň $A2$ přejde slovem w ze stavu $q2$ do stavu $q2'$ právě tehdy a jen tehdy když $A3$ vznikající sjednocením $A1 \cup A2$ přejde slovem w ze stavu $(q1, q2)$ do stavu $(q1', q2')$.
Pomocí značně zjednodušeného matematického zápisu lze formuli vyjádřit takto.

$$(q1q2)w = (q1', q2') \equiv (q1w = q1' \wedge q2w = q2')$$

Je zřejmé, že pro přesnější zápis je nutné blíže specifikovat funkci přechodu při přijetí slova.

```
Inductive transition: Q -> Word Sigma -> Q -> Prop :=
|trans_empty: forall q, transition q nil q
|trans_ind : forall q q' q'' h t, delta q h = q' -> transition q' t q'' -> transition q (h::t)
```

Poněkud přesnější zápis již v jazyce coq má následující podobu.

```
Lemma Union_transition : forall Sigma (Automaton_a Automaton_b : DFA Sigma),
  let automaton_union := Union_automata Sigma Automaton_a Automaton_b in
  let Q1 := Q Sigma Automaton_a in
  let Q2 := Q Sigma Automaton_b in
      forall (w: Word Sigma)(q1 q1': Q1)(q2 q2': Q2),
        transition Sigma automaton_union (pair q1 q2) w (pair q1' q2') <->
        (transition Sigma Automaton_a q1 w q1' /\ transition Sigma Automaton_b q2 w q2').
```

Kapitola 5

Závěr

Práce přibližuje základní práci v prostředí dokazovacího nástroje coq. Výstupem je ukázka možné formalizace deterministického konečného automatu.