Minimization of Symbolic Automata

Jonathan Homburg

Based on "Minimization of Symbolic Automata" by Loris D'Antoni & Margus Veanes

Intuition

- A deterministic finite automaton (DFA) is defined over a finite alphabet. Characters from this alphabet are given to the DFA as input and also label the state transitions.
- A symbolic finite automaton (SFA) has inputs defined over a (possibly infinite) domain set and transitions labeled by a set of predicates over that domain.
  - e.g. an SFA might have $\mathbb{Z}$ as a domain and might have predicates $\text{odd}(x)$ and $\text{even}(x)$ for transitions.

Definition

An effective Boolean algebra is a tuple $A = (D, \Psi, [\![\cdot]\!], \bot, \top, \wedge, \vee, \neg)$ such that

- D is a recursively enumerable set of domain elements.
- Ψ is a recursively enumerable set of predicates closed under $\wedge, \vee, \neg$ and with $\bot, \top \in \Psi$
- $[\![\cdot]\!] : \Psi \to 2^D$ (called the denotation function) is recursively enumerable and has:
  - $[\![\bot]\!] = \varnothing$ and $[\![\top]\!] = D$
  - $\forall\ \varphi, \psi \in \Psi$, $[\![\phi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!]$, $[\![\phi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$, and $[\![\neg\phi]\!] = D \setminus [\![\varphi]\!]$

For $\varphi \in \Psi$, if $[\![\varphi]\!] \neq \varnothing$ then φ is said to be satisfiable and we write $\mathit{IsSat}(\varphi)$. It is generally required that satisfiability checking is decidable.

Definition

A symbolic finite automaton is a tuple $M = (Q, A, q^0, \Delta, F)$ such that

- Q is a finite set of states
- A is an effective Boolean algebra, called the alphabet
- $q^0 \in Q$ is the initial state
- $\Delta \subseteq Q \times \Psi_A \times Q$ is a finite set of transitions
- $F \subseteq Q$ is the set of final states

Most notation used for SFAs is directly analogous to those used for DFAs. e.g. $L(M), \delta(p, a), \overleftarrow{\Delta}(q)$

Properties of a Symbolic Automaton

Let $M = (Q, A, q^0, \Delta, F)$ be an SFA

- M is determinstic iff $\forall (p, \varphi, p'), (p, \varphi', q') \in \Delta$ we have $IsSat(\varphi \wedge \varphi')$ implies $q = q'$
- M is complete iff $\forall a \in D_A, p \in Q \; \exists q \in Q, \varphi \in \Psi_A$ such that $(p, \varphi, q) \in \Delta$ and $a \in [\![\varphi]\!]$
- M is clean iff $\forall (p, \varphi, q) \in \Delta$, p is reachable from q^0 and $IsSat(\varphi)$
- M is normalized iff $\forall p, q \in Q$ there is at most one transition from p to q

It is nearly always assumed that an SFA is clean and normalized. Four our purposes, it will generally be assumed that every SFA is deterministic and complete.

Minimality of SFAs

Definition

An SFA M is minimal if it is deterministic, complete, clean normalized, and for all $p, q \in Q$, $p = q$ if and only if $L_p(M) = L_q(M)$.

Theorem

If M and N are minimal SFAs such that $L(M) = L(N)$ then M and N are equivalent automata up to the relabeling of states and equivalence of predicates.

M-equivalence

Definition

Two states $p, q \in Q$ are M-equivalent, denoted $p \equiv_M q$, if $L_p(M) = L_q(M)$.

Theorem

If M is a clean, complete and deterministic SFA then $M_{/\equiv_M}$ is minimal and $L(M) = L(M_{/\equiv_M})$.

Moore's Algorithm (DFA)

Let $M = (Q, \Sigma, q^0, \delta, F)$ be a DFA.
Define $\equiv_0$ on Q such that $p \equiv_0 q \iff p, q \in F$ or $p, q \notin F$
Recursively define $\equiv_i$ on Q such that

$$p \equiv_i q \iff p \equiv_{i-1} q \text{ and } \forall a \in \Sigma, \delta(p, a) \equiv_{i-1} \delta(q, a)$$

Continue until $\equiv_k$ and $\equiv_{k-1}$ are equivalent relations.
$M_{/\equiv_k}$ is minimal.

Moore's Algorithm (SFA)

Let $M = (Q, A, q^0, \Delta, F)$ be an SFA.
Define $\equiv_0$ on Q s.t. $p \equiv_0 q \iff (p, q) \in (F^c \times F) \cup (F \times F^c)$
Recursively define $\equiv_i$ on Q such that

$p \equiv_i q \iff p \equiv_{i-1} q$ or
$\exists (p, \varphi, p'), (q, \psi, q') \in \Delta$ s.t. $p' \equiv_{i-1} q'$ and IsSat$(\varphi \wedge \psi)$

Continue until $\equiv_k$ and $\equiv_{k-1}$ are equivalent relations
Define $\equiv_M$ on Q such that $p \equiv_M q \iff p \not\equiv_k q$
$M_{/\equiv_M}$ is minimal

Hopcroft's Algorithm (DFA)

input: DFA $M = (Q, \Sigma, q^0, \delta, F)$
$P = \{F, Q \setminus F\}$ // Q is partitioned
$W = \{F\}$ // Working set
while $W \neq \varnothing$ **do**
    remove any R from W
    for $a \in \Sigma$ **do**
        $S = \delta^{-1}(R, a)$ // states leading into R on a
        for $T \in P$ **do**
            if $T \cap S \neq \varnothing$ *and* $T \setminus S \neq \varnothing$ **then**
                Split(T, $T \cap S$, $T \setminus S$) // refines P, W.
                    Adds smaller of $\{T \cap S, T \setminus S\}$ to W
            end
        end
    end
end
Return $M_{/\equiv_P}$

Hopcroft's Algorithm (SFA)

input: SFA $M = (Q, A, q^0, \Delta, F)$
$P = \{F, Q \setminus F\}$ // Q is partitioned
$W = \{F\}$ // Working set
while $W \neq \varnothing$ **do**
  remove any R from W
  for $\varphi \in$ *Minterms(M)* **do**
    $S = \delta^{-1}(R, \varphi)$ // states leading into R on a
    for $T \in P$ **do**
      if $T \cap S \neq \varnothing$ *and* $T \setminus S \neq \varnothing$ **then**
        Split(T, $T \cap S$, $T \setminus S$) // refines P, W.
          Adds smaller of $\{T \cap S, T \setminus S\}$ to W
      end
    end
  end
end
Return $M_{/\equiv_P}$

Modified Hopcroft's Algorithm (SFA)

input: SFA $M = (Q, A, q^0, \Delta, F)$
$P = \{F, Q \setminus F\}$
$W = \{F\}$
while $W \neq \varnothing$ **do**
  remove any R from W
  $S = \overleftarrow{\Delta}(R)$ // set of all states going into R
  $\Gamma = \{S \ni p \mapsto \bigvee_{(p,\varphi,q)\in\Delta, q\in R} \varphi\}$ // $\Gamma(p)$ is disjunction of all predicates going into R
  while $\exists T \in P$ *with* $T \cap S \neq \varnothing$ *and* $\exists p_1, p_2 \in T$ *s.t.* *IsSat*$(\neg(\Gamma(p_1) \iff \Gamma(p_2)))$ **do**
    Let $a \in [\![\neg(\Gamma(p_1) \iff \Gamma(p_2))]\!]$
    $T_1 = \{p \in T \mid a \in \Gamma(p)\}$
    Split$(T,\ T_1,\ T \setminus T_1)$
  end
end
Return $M_{/\equiv_P}$ (Note: optional optimization steps are skipped)