Masaryk University
Faculty of Informatics

Transformation of Nondeterministic Büchi Automata to Slim Automata

Bachelor's Thesis

Pavel Šimovec

Brno, Fall 2020

Replace this page with a copy of the official signed thesis assignment and a copy of the Statement of an Author.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Pavel Šimovec

Advisor: RNDr. František Blahoudek, Ph.D.; doc. RNDr. Jan Strejček, Ph.D.

Acknowledgements

ack

Abstract

abstract

Keywords

keyword1, keyword2, …

Contents

List of Figures

1 Introduction

2 Preliminaries

In this section we define a Büchi automaton, as slim automata are specially constructed Büchi automata.

2.1 Büchi Automaton

A Büchi automaton is a theoretical finite-state machine used to define ω-languages. It decides which infinitely long words (ω-words) belong to its language.

A *transition-based Büchi automaton* (*TBA*) is a quintuple $A = (\Sigma, Q, q_i, \Delta, \Gamma)$, where

- Σ is a finite *alphabet*,
- Q is a finite set of *states*,
- $q_0 \in Q$ is the initial state of A.
- We write the set of *transitions* as $\Delta \subseteq Q \times \Sigma \times Q$. Intuitivelly, a transition (s, a, t) directionally connects two states inside Q with a letter from alphabet Σ.
- $\Gamma \subseteq \Delta$ is a set of *accepting transitions*.

An *alphabet* is a set of letters. An *ω-word* $w \in \Sigma^\omega$ is an infinite sequence of letters. A *language* is a set of ω-words.

A *run* r of A is an infinite sequence of transitions $r = (q_0, a_0, q_1)(q_1, a_1, q_2)(q_2, a_2, q_3) \ldots \in \Delta^\omega$ such that $q_0 = q_i$. A run of TBA is *accepting* iff it contains infinitely many accepting transitions.

Finally, we define the *language* $L_A \in \Sigma^\omega$ recognized by the automaton A. An ω-word $w \in \Sigma^\omega$ belongs to L_A iff there exists an accepting run of A over the word w.

2.2 Generalized Büchi Automaton

A *transition-based Generalized Büchi automaton* (TGBA) $A = (\Sigma, Q, q_i, \Delta, G)$ is a modified TBA, where $G \in 2^\Delta$ is set containing sets of accepting conditions. A run of TGBA is *accepting* iff it contains infinitely many accepting transitions *for each* subset of G.

2.3 Slim GFM Büchi Automaton WIP

We define *slim GFM*[1] *Büchi automaton* (slim automaton) through its construction which is based on breakpoint construction.

Construction Let us fix Büchi Automaton $A = (\Sigma, Q, q_i, \Delta, \Gamma)$. We can write Δ as a function $\hat{\delta} : Q \times \Sigma \to 2^Q$ with $\hat{\delta} : (q, a) \to \{q' \in Q | (q, a, q') \in \Delta\}$, which can be lifted to sets, using the deterministic transition function $\delta : 2^Q \times \Sigma \to 2^Q$ with $\delta : (S, a) \to \bigcup_{q \in S} \hat{\delta}(q, a)$.

We also define an operator, ndet, that translates deterministic transition functions $\delta : R \times \Sigma \to R$ to relations using

ndet: $(R \times \Sigma \to R) \to 2^{(R \times \Sigma \to R)}$ with $\delta \to \{(q, a, q') | q' \in \delta(\{q\}, a)\}$.

This is just an easy means to move back and forth between functions and relations, and helps one to visualize the maximal number of successors. We next define the variations of subset and breakpoint constructions that are used to define limit deterministic?? *semi-deterministic* GFM automata(semi-deterministic automata) - which we use in our evaluation for comparison - and the slim automata we construct.

Let $3^Q := \{(S, S') | S' \subset S \subseteq Q\}$ and $3^Q_+ := \{(S, S') | S' \subseteq S \subseteq Q\}$. We define the subset notation for the transitions and accepting transitions as $\delta_S, \gamma_S : 2^Q \times \Sigma \to 2^Q$ with

$\delta_S : (S, a) \to \{q' \in Q | \exists q \in S.(q, a, q') \in \Delta\}$ and

$\gamma_S : (S, a) \to \{q' \in Q | \exists q \in S.(q, a, q') \in \Gamma\}$

We define the raw breakpoint transition $\delta_R : 3^Q \times \Sigma \to 3^Q_+$ as $((S, S'), a) \to (\delta_S(S, a), \delta(S', a) \cup \gamma_S(S, a))$. In this construction, we follow the set of reachable states (first set) and the states that are reachable while passing at least one of the accepting transitions (second set). To turn this into a breakpoint automaton, we reset the second set to the empty set when it equals the first; the transitions where we reset the second set are exactly the accepting ones. The breakpoint automaton $D = (\Sigma, 3^Q, (q_i, \emptyset), \delta_B, \gamma_B)$ is defined such that, when $\delta_R : ((S, S'), a) \to (R, R')$, then there are three cases:

1. if $R = \emptyset$, then $\delta_B((S, S'))$ is undefined (or, if a complete automation is preferred, maps to a rejecting sink),

1. Good for Markov decision processes [+zdroj]

2. else, if $R \neq R'$, then $\delta_B((S, S'), a) \to (R, R')$ is a non-accepting transition,

3. otherwise $\delta_B, \gamma_B : \delta_B((S, S'), a) \to (R, \emptyset)$ is an accepting transition.

Finally, we define transitions $\Delta_{SB} \subseteq 2^Q \times \Sigma \times 3^Q$ that lead from a subset to a breakpoint construction, and $\gamma_{2,1} : 3^Q \times \Sigma \to 3^Q$ that promote the second set of a breakpoint construction to the first set as follows. (?? probably a different text will be necessary here, following enumerate is WIP)

1. $\Delta_{SB} = \{(S, a, (S', \emptyset)) | \emptyset \neq S' \subseteq \delta_S(S, a)\}$ are non-accepting transitions,

2. if $\delta_S(S', a) = \gamma_S(S, a) = \emptyset$, then $\gamma_p((S, S'), a)$ and $\gamma_w((S, S'), a)$ are undefined, and

3. otherwise $\gamma_p : ((S, S'), a) \to (\delta_S(S', a) \cup \gamma_S(S, a), \emptyset)$ and $\gamma_w : ((S, S'), a) \to (\delta_S(S', a), \emptyset)$ are accepting transitions.

...? (I guess I could define semi-deterministic automaton, as it will be used in evaluation for comparison)

2.4 Slim automaton

We present 2 of possible variants of slim automata:

$S = (\Sigma, 3^Q, (q_i, \emptyset), ndet(\delta_B) \cup ndet(\delta_p), ndet(\delta_B) \cup ndet(\gamma_p))$ is slim.

$W = (\Sigma, 3^Q, (q_i, \emptyset), ndet(\delta_B) \cup ndet(\delta_w), ndet(\delta_B) \cup ndet(\gamma_w))$ is weak slim.

(Jen takova poznamka... Dava vlastne *weak slim* smysl? Pokud to chapu, tak tady v puvodnim clanku definuji *slim* spis jako vlastnost, nez jako nazev.. Takze *weak slim* co davam do seminatoru by stale mel byt *slim*...)

3 To Delete Chapter

3.1 Büchi Automaton

A nondeterministic Büchi automaton (BA) is a tuple $A = (\Sigma, Q, q_0, \Delta, \Gamma)$, where

- Σ is a finite alphabet
- Q is finite set of states
- $q_0 \in Q$ is the initial state
- $\Delta \subseteq Q \times \Sigma \times Q$ are transitions
- $\Gamma \subseteq \Delta$ are accepting transitions

run A run r of A on $w \in \Sigma^\omega$ is an ω-word $r_0, w_0, r_1, w_1, ...$ in $(Q \times \Sigma)^\omega$ such that $r_0 = q_0 \wedge \forall i > 0, (r_{i-1}, w_{i-1}, r_i) \in \Delta$

inf(r) We write $inf(r) \subseteq \Delta$ for the set of transitions that appear infinitely often in the run r.

accepting run A run r is accepting if $inf(r) \cap \Gamma \neq \emptyset$

language The language $L_A \subseteq \Sigma^\omega$ is recognized by A.
$\forall w \in L_A \exists r$ on w such that r is accepting.

ω-regular language A language is ω-regular if it is accepted by BA.

deterministic automaton $A = (\Sigma, Q, q_0, \Delta, \Gamma)$ is deterministic if $(q, \rho, q^{'}), (q, \rho, q^{''}) \in \Delta \implies q^{'} = q^{''}$

complete automaton A is complete if, $\forall w \in \Sigma, \forall q \in Q, \exists (q, w, q^{'}) \in \Delta$. A word in Σ^ω has exactly one run in a deterministic, complete automaton.

nepouzivat ρ , kombinace ρ a q je spatna
zminit v intro

3.2 Markov Decision Processes

A Markov decision process (MDP) M is a tuple (S, A, T, Σ, L), where

- S is a finite set of states
- A is a finite set of actions
- $T : S \times A \to D(S)$, where $D(S)$ is set of probability distributions over S, is the probabilistic transition (partial) function
- Σ is an alphabet
- $L : S \times A \times S \to \Sigma$ is the labeling function of the set of transitions. For a state $s \in S$, $A(s)$ denotes the set of actions available in s.

run A run of M is an ω-word $s_0, a_1, ... \in A = S \times (A \times S)^\omega$ such that $Pr(s_{i+1}|s_i, a_{i+1}) > 0$ for all $i >= 0$. A finite run is a finite such sequence.

labeled run We define labeled run as $L(r) = L(s_0, a_1, s_1), L(s_1, a_2, s_2), ... \in \Sigma^\omega$.

paths We write $\Omega(M)(Paths(M))$ for the set of runs (finite runs) of M and $\Omega_s(M)(Paths_s(M))$ for the set of runs (finite runs) of M starting from state s. When the MDP is clear from the context we drop the argument M.

strategy A strategy in M is a function $\mu : Paths \to D(A)$ such that $supp(\mu(r)) \subseteq A(last(r))$, where $supp(d)$ is the support of d and $last(r)$ is the last state of r. Let Ω_μ^M denote the subset of runs Ω^M that correspond to strategy μ and initial state s. Let Π_M be the set of all strategies.

pure strategy We say that a strategy μ is pure if $\mu(r)$ is a point distribution for all runs $r \in Paths$.

behavior The behavior of an MDP M under a strategy μ with starting state s is defined on a probability space $(\Omega_s^\mu, F_s^\mu, Pr_s^\mu)$ over the set of infinite runs of μ from s.

3.3 [WIP]Good-for-MDP (GFM) Automata

Given an MDP M and an automaton $A = (\Sigma, Q, q_0, \Delta, \Gamma)$, we want to compute an optimal strategy satisfying the objective that the run of M is in the language of A.

semantic satisfaction probability for given automaton and strategy We define the semantic satisfaction probability for A and strategy μ from state s as:
$PSem_A^M(s, \mu) = Pr_s^\mu\{r \in \Omega_s^\mu : L(r) \in L_A\}$

semantic satisfaction probability for given automaton .
$PSem_A^M(s) = \sup_{\mu \in \Pi_M} PSem_A^M(s, \mu)$

syntactic variant of the acceptance condition When using automata for given analysis of MDPs, we need a syntactic variant of the acceptance condition.

product of MDP and automaton Given an MDP $M = (S, A, T, \Sigma, L)$ with initial state $s_0 \in S$ and automaton $A = (\Sigma, Q, q_0, \Delta, \Gamma)$, the product $M \times A = (S \times Q, (s_0, q_0), A \times Q, T^\times, \Gamma^\times)$ is an MDP augmented with an inital state $s_0 \in S$ and accepting transitions $\Gamma^\times$. The (partial) function $T^\times : (S \times Q) \times (A \times Q) \to D(S \times Q)$ is defined by
$$T^\times((s,q),(a,q^{'}))((s^{'},q^{'})) = \begin{cases} T(s,a)(s^{'}) & if(q, L(s,a,s^{'}), q^1) \in \Delta \\ undefined & otherwise \end{cases}$$

GFM Automata An automaton A is good for MDPs if, for all MDPs M, $PSYN_A^M(s_0) = PSEM_A^M(s_0)$ holds, where s_0 is the initial state of M.

3.4 to be defined

ω-word?, point distribution?, what is F_s^μ in 'pure strategy' paragraph?, TGBA, describe Semi-determistic as I am going to compare them with SBA

3.4.1 text

GF MDP, model checking

3.5 Algorithms

BP + both slim

4 Implementation

4.1 Technologies

4.2 Implementation inside Seminator

5 Evaluation

5.1 Alternative Algorithm

5.2 Different Implementation - ePMC

5.3 Semi-deterministic Automata

6 Conclusion