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

... slim automata are specially constructed Büchi automata...

2 Preliminaries

In this section we define a Büchi automaton. We will need to know that on *alphabet* is a set of letters, an ω-*word* $w \in \Sigma^\omega$ is an infinite sequence of letters, and A *language* is a set of ω-words.

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 tuple $\mathcal{A} = (\Sigma, Q, q_i, \Delta, \Gamma)$, where

- Σ is a finite *alphabet*,
- Q is a finite set of *states*,
- $q_i \in Q$ is the initial state of $\mathcal{A}$.
- 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*.

A *run* r of $\mathcal{A}$ 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 $\mathcal{A}$. An ω-word $w \in \Sigma^\omega$ belongs to L_A iff there exists an accepting run of $\mathcal{A}$ over the word w.

2.2 Generalized Büchi Automaton

A *transition-based Generalized Büchi automaton* (TGBA) is a tuple $\mathcal{A} = (\Sigma, Q, q_i, \Delta, G)$ is a modified TBA, where $G = \{\Gamma_0, \Gamma_1, \ldots, \Gamma_{|G|-1}\} \subseteq 2^\Delta$ contains sets of accepting conditions and the rest is defined as for

TBA. A run of $\mathcal{A}$ is *accepting* iff it contains infinitely many accepting transitions *for each* subset of G. Büchi Automata can be seen as a special case with $|G| = 1$

2.3 Breakpoint Automaton

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

Construction Let us fix Büchi Automaton $\mathcal{A} = (\Sigma, Q, q_i, \Delta, \Gamma)$. We define the variations of subset and breakpoint constructions that are used to define semi-deterministic GFM automata (semi-deterministic automata) - which we use in our evaluation for comparison - and the slim automata we construct.

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

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

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

We define the raw breakpoint transition $\rho\colon 3^Q \times \Sigma \to 3^Q_+$ as $((S, S'), a) \to (\delta(S, a), \delta(S', a) \cup \gamma(S, a))$. 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 $\mathcal{D} = (\Sigma, 3^Q, (q_i, \emptyset), \delta_B, \gamma_B)$ is defined such that, when $\rho\colon ((S, S'), a) \to (R, R')$, 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),
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.

1. Good for Markov decision processes [+zdroj]

Breakpoint automata are insufficient to decide all GFM languages. On the other hand, semi-deterministic automata decide superset of such language. We are going to define a few more transitions on top of breakpoint construction which allow us to construct slim automata that decide exactly the class of GFM languages.

2.4 Slim Automata Construction

Let us we define transitions $\gamma_w, \gamma_p : 3^Q \times \Sigma \to 3^Q$ that promote the second set of a breakpoint construction to the first set as follows.

1. 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
2. otherwise $\gamma_p : ((S, S'), a) \to (\delta(S', a) \cup \gamma(S, a), \emptyset)$ and $\gamma_w : ((S, S'), a) \to (\delta(S', a), \emptyset)$

$\mathcal{S} = (\Sigma, 3^Q, (q_i, \emptyset), \Delta_p, \Gamma_p)$ is slim, when Δ_p is set of transitions generated by δ_b and γ_p, and Γ_p is set of accepting transitions, that is generated by γ_b and γ_p. (proof in text with original definition)

Alternatively, similarly defined using γ_w instead of γ_p, automaton $\mathcal{W} = (\Sigma, 3^Q, (q_i, \emptyset), \Delta_w, \Gamma_w)$ is slim. (no proof yet)

2.5 Slim Automaton Construction Generalized to TGBA

We want to construct a slim automaton from TGBA $\mathcal{T} = (\Sigma, Q, q_i, \Delta, G)$. One possibility is to *degeneralize* $\mathcal{T}$ and to use previously mentioned algorithm in section 2.3. Another way is to extend slim automaton construction to TGBA.

extended slim construction We need to make sure we go infinitely many times trough each accepting subset $g \in G$. To achieve this, we will go through each subset one by one, using original algorithm. We will keep track of levels($:= \{0, 1, \ldots, |G| - 1\}$) in the names of states. Let $|G|$ be number of levels and $i \in N, i < |G|$ the current level. At each level, we look at ith subset of G. We use same steps as in classic

breakpoint construction, but on each accepting transition the new state will be leveled up to $(i+1)\%|G|$, otherwise the target state has the same level. Our new automaton simulates $\mathcal{T}$, as it accepts a word iff it cycles through all levels. If $|G| = 0$, we return a trivially accepting automaton

We can use the core of previous construction and just to extend it with levels. Let

$P := 3^Q \times$levels and $P_+ := 3^Q_+ \times$levels

We define γ_i similarly like γ, we just use Γ_i instead of Γ

$\mathrm{up}(x)=(x+1) \mod |G|$

We define the raw generalized breakpoint transitions

$\delta_R : P \times \Sigma \to P+$ as $((S, S', i), a) \to (\delta(S, a), \delta(S', a) \cup \gamma_i(S, a), j)$

The generalized breakpoint automaton $\mathcal{D} = (\Sigma, 3^Q \times \mathcal{N}, (q_i, \emptyset, 0))$ is defined such that, when $\delta_R : ((S, S', i), a) \to (R, R', j)$, then there are three cases:

1. if $R = \emptyset$, then $\delta_B((S, S', i))$ is undefined,
2. else, if $R \neq R'$, then $\delta_B : ((S, S', i), a) \to (R, R', i)$ is a non-accepting transition,
3. otherwise $\delta_B, \gamma_B : \delta_B((S, S', i), a) \to (R, \emptyset, \mathrm{up}(i))$.

$\gamma_p : P \times \Sigma \to P$

1. if $\delta(S', a) = \gamma_i(S, a) = \emptyset, then \gamma_p((S, S', i), a)$ is undefined, and
2. otherwise $\gamma_p : ((S, S', i), a) \to (\delta(S', a) \cup \gamma_i(S, a), \emptyset, \mathrm{up}(i))$ is an accepting transition.

$\mathcal{S} = (\Sigma, P, (q_i, \emptyset, 0), ndet(\delta_B) \cup ndet(\gamma_p), ndet(\gamma_B) \cup ndet(\gamma_p))$ is 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