Journal of Logical and Algebraic Methods in Programming 95 (2018) 1–16

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Optimised determinisation and completion of finite tree automata

John P. Gallagher [a,b,*], Mai Ajspur [c], Bishoksan Kafle [d]

[a] Roskilde University, Denmark
[b] IMDEA Software Institute, Madrid, Spain
[c] IT University of Copenhagen, Denmark
[d] The University of Melbourne, Australia

ARTICLE INFO

Article history:
Received 5 March 2017
Received in revised form 28 September 2017
Accepted 31 October 2017
Available online 6 November 2017

ABSTRACT

Determinisation and completion of finite tree automata are important operations with applications in program analysis and verification. However, the complexity of the classical procedures for determinisation and completion is high. They are not practical procedures for manipulating tree automata beyond very small ones. In this paper we develop an algorithm for determinisation and completion of finite tree automata, whose worst-case complexity remains unchanged, but which performs far better than existing algorithms in practice. The critical aspect of the algorithm is that the transitions of the determinised (and possibly completed) automaton are generated in a potentially very compact form called product form, which can reduce the size of the representation dramatically. Furthermore, the representation can often be used directly when manipulating the determinised automaton. The paper contains an experimental evaluation of the algorithm on a large set of tree automata examples.

1. Introduction

A recognisable tree language is a possibly infinite set of trees that are accepted by a finite tree automaton (FTA). FTAs and the corresponding recognisable languages have desirable properties such as closure under Boolean set operations, and decidability of membership and emptiness.

In the paper we will give a brief overview of the relevant features of FTAs, but the main goal of the paper is to focus on two operations on FTAs, namely *determinisation* and *completion*. These operations play a key role in the theory of FTAs, for example in showing that recognisable tree languages are closed under Boolean operations. Potentially, they also play a practical role in systems that manipulate sets of terms, but their complexity has so far discouraged their widespread application.

In the paper we develop an optimised algorithm that performs determinisation and optionally completion, and analyse its properties. The most critical aspect of the optimisation is a compact representation of the set of transitions of the determinised automaton, called *product transitions*. Experiments show that the algorithm performs well, though the worst case remains unchanged. We also discuss applications of finite tree automata that exploit the determinisation algorithm.

* Corresponding author at: Roskilde University, Denmark.
E-mail address: jpg@ruc.dk (J.P. Gallagher).

https://doi.org/10.1016/j.jlamp.2017.10.004
2352-2208/

In Section 2 the essentials, for our purposes, of finite tree automata are introduced, including the notion of product transitions. The operations of determinisation and completion are defined. Section 3 presents the optimised algorithm for determinising an FTA. It is developed in a series of stages starting from the textbook algorithm for determinisation. In Section 3 it is shown how the algorithm can be optimised and output transitions in product form. The performance of the algorithm is analysed in Section 4. In Section 5 we discuss the combination of determinisation and completion of an FTA and show that the performance of the algorithm generating product form is as effective when generating a complete determinised automaton. Section 6 reports on the performance of the algorithm on a large number of example tree automata. Section 7 discusses potential applications of the algorithm to problems in program analysis and verification, and also to tree automata problems including checking inclusion and universality. Section 8 contains a discussion of related work and finally in Section 9 we summarise and discuss further work and applications.

2. Preliminaries

A *finite tree automaton* (FTA) is a quadruple $\langle Q, Q_f, \Sigma, \Delta\rangle$, where

1. Q is a finite set called *states*,
2. $Q_f \subseteq Q$ is called the set of accepting (or final) states,
3. Σ is a set of function symbols (called the *signature*) and
4. Δ is a set of *transitions*.

Each function symbol $f \in \Sigma$ has an arity $n \geq 0$, written $\mathsf{ar}(f) = n$. Function symbols with arity 0 are called *constants*. Q and Σ are disjoint. $\mathsf{Term}(\Sigma)$ is the set of *ground terms* (also called *trees*) constructed from Σ where $t \in \mathsf{Term}(\Sigma)$ iff $t \in \Sigma$ is a constant or $t = f(t_1, \ldots, t_n)$ where $\mathsf{ar}(f) = n$ and $t_1, \ldots, t_n \in \mathsf{Term}(\Sigma)$. Similarly $\mathsf{Term}(\Sigma \cup Q)$ is the set of terms/trees constructed from Σ and Q, treating the elements of Q as constants. Each transition in Δ is of the form $f(q_1, \ldots, q_n) \to q$, where $\mathsf{ar}(f) = n$ and $q, q_1, \ldots, q_n \in Q$.

To define acceptance of a term by the FTA $\langle Q, Q_f, \Sigma, \Delta\rangle$ we first define a *context* for the FTA. A context is a term from $\mathsf{Term}(\Sigma \cup Q \cup \{\bullet\})$ containing exactly one occurrence of $\bullet$ (which is a constant not in Σ or Q). Let c be a context and $t \in \mathsf{Term}(\Sigma \cup Q)$; $c[t]$ denotes the term resulting from the replacement of $\bullet$ in c by t. A term $t \in \mathsf{Term}(\Sigma \cup Q)$ can be written as $c[t']$ if t has a subterm t', where c is the context resulting from replacing that subtree by $\bullet$.

The binary relation $\Rightarrow$ represents one step of a (bottom-up) run for the FTA. It is defined as follows; $c[l] \Rightarrow c[r]$ iff c is a context and $l \to r \in \Delta$. The reflexive, transitive closure of $\Rightarrow$ is denoted $\Rightarrow^*$.

A run for $t \in \mathsf{Term}(\Sigma)$ exists if $t \Rightarrow^* q$ where $q \in Q$. The run is *successful* if $q \in Q_f$ and in this case t is *accepted* by the FTA. There may be more than one state q such that $t \Rightarrow^* q$ and hence FTAs are sometimes called NFTAs, where N stands for non-deterministic. A tree automaton R defines a set of terms, that is, a tree language, denoted $L(R)$, as the set of all terms that it accepts. We also write $L(q)$ to be the set of terms t such that $t \Rightarrow^* q$ in a given FTA.

Definition 1. An FTA $\langle Q, Q_f, \Sigma, \Delta\rangle$ is called *bottom-up deterministic* if and only if Δ contains no two transitions with the same left hand side. A bottom-up deterministic FTA is abbreviated as a DFTA.

Runs of a DFTA are deterministic in the following sense; for every context c and term of form $c[t]$ there is at most one term $c[t']$ such that $c[t] \Rightarrow c[t']$. It follows that for every $t \in \mathsf{Term}(\Sigma)$ there is at most one $q \in Q$ such that $t \Rightarrow^* q$. As far as expressiveness is concerned we can limit our attention to DFTAs.[1] For every FTA R there exists a DFTA R' such that $L(R) = L(R')$.

Definition 2. An automaton $R = \langle Q, Q_f, \Sigma, \Delta\rangle$ is *complete* if for all n-ary functions $f \in \Sigma$ and states $q_1, \ldots, q_n \in Q$, there exists a state $q \in Q$ such that $f(q_1, \ldots, q_n) \to q \in \Delta$.

It follows that in a complete FTA every term t has at least one run and furthermore in a complete DFTA each t has a run to exactly one state. Thus a complete DFTA defines a partition of $\mathsf{Term}(\Sigma)$, namely $\{L(q) \mid q \in Q\} \setminus \{\emptyset\}$.

Definition 3. Let Σ be a signature and "*any*" a state. We define Δ^{Σ}_{any} to be the following set of transitions.

$$\{f(\overbrace{any, \ldots, any}^{n \text{ times}}) \to any \mid f^n \in \Sigma\}$$

Clearly, given an FTA $\langle Q, Q_f, \Sigma, \Delta\rangle$ with $any \in Q$ and $\Delta^{\Sigma}_{any} \subseteq \Delta$, there is a run $t \Rightarrow^* any$ for any $t \in \mathsf{Term}(\Sigma)$, that is, $L(any) = \mathsf{Term}(\Sigma)$.

[1] We do not deal here with top-down deterministic FTA, which are strictly less expressive than FTAs.