Information and Computation 252 (2017) 221–242

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Counting branches in trees using games

Arnaud Carayol [a,*,1], Olivier Serre [b,**,2]

[a] LIGM (CNRS & Université Paris Est), France
[b] IRIF (CNRS & Université Paris Diderot – Paris 7), France

ARTICLE INFO

Article history:
Received 21 January 2016
Received in revised form 16 September 2016
Available online 23 November 2016

Keywords:
Automaton on infinite trees
Two-player game
Cardinality constraint
Topologically large set

ABSTRACT

We study finite automata running over infinite binary trees. A run of such an automaton is usually said to be accepting if all its branches are accepting. In this article, we relax the notion of accepting run by allowing a certain quantity of rejecting branches. More precisely we study the following criteria for a run to be accepting:

(i) it contains at most finitely (resp. countably) many rejecting branches;
(ii) it contains infinitely (resp. uncountably) many accepting branches;
(iii) the set of accepting branches is topologically "big".

In all situations we provide a simple acceptance game that later permits to prove that the languages accepted by automata with cardinality constraints are always ω-regular. In the case (ii) where one counts accepting branches it leads to new proofs (without appealing to logic) of a result of Beauquier and Niwiński.

1. Introduction

There are several natural ways of describing sets of infinite trees. One is *logic* where, with any formula, one associates the set of all trees for which the formula holds. Another option is using *finite automata*. Finite automata on infinite trees (that extends both automata on infinite words and on finite trees) were originally introduced by Rabin in [1] to prove the decidability of the monadic second order logic (MSOL) over the full binary tree. Indeed, Rabin proved that for any MSOL formula, one can construct a tree automaton such that it accepts a non-empty language if and only if the original formula holds at the root of the full binary tree. These automata were also successfully used by Rabin in [2] to solve Church's synthesis problem [3], that asks for constructing a circuit based on a formal specification (typically expressed in MSOL) describing the desired input/output behaviour. His approach was to represent the set of all possible behaviours of a circuit by an infinite tree (directions code the inputs while node labels along a branch code the outputs) and to reduce the synthesis problem to emptiness of a tree automaton accepting all those trees coding circuits satisfying the specification. Since then, automata on infinite trees and their variants have been intensively studied and found many applications, in particular in logic. Connections between automata on infinite trees and logic are discussed *e.g.* in the excellent surveys [4,5].

* Corresponding author.
** Principal corresponding author.
E-mail addresses: Arnaud.Carayol@univ-mlv.fr (A. Carayol), Olivier.Serre@cnrs.fr (O. Serre).
[1] *Postal address*: IRIF, Université Paris Diderot – Paris 7, Case 7014, 75205 Paris Cedex 13, France.
[2] *Postal address*: LIGM, Cité Descartes, Bât Copernic – 5, bd Descartes, Champs sur Marne, 77454 Marne-la-Vallée Cedex 2, France.

http://dx.doi.org/10.1016/j.ic.2016.11.005
0890-5401/

Roughly speaking a finite automaton on infinite trees is a finite memory machine that takes as input an infinite node-labelled binary tree and processes it in a top-down fashion as follows. It starts at the root of the tree in its initial state, and picks (possibly nondeterministically) two successor states, one per child, according to the current control state, the letter at the current node and the transition relation. Then the computation proceeds in parallel from both children, and so on. Hence, a run of the automaton on an input tree is a labelling of this tree by control states of the automaton, that should satisfy the local constraints imposed by the transition relation. A branch in a run is accepting if the ω-word obtained by reading the states along the branch satisfies some acceptance condition (typically an ω-regular condition such as a Büchi or a parity condition). Finally, a tree is accepted by the automaton if there exists a run over this tree in which *every* branch is accepting. An ω-regular tree language is a tree language accepted by some tree automaton equipped with a parity condition.

A fundamental result of Rabin is that ω-regular tree languages form a Boolean algebra [1]. The main technical difficulty in establishing this result is to show the closure under complementation. Since the publication of this result in 1969, it has been a challenging problem to simplify this proof. A much simpler one was obtained by Gurevich and Harrington in [6] making use of two-player perfect information games for checking membership of a tree in the language accepted by the automaton[3]: Éloïse (a.k.a. *Automaton*) builds a run on the input tree while Abélard (a.k.a. *Pathfinder*) tries to exhibit a rejecting branch in the run. Another fruitful connection between automata and games is for emptiness checking. In a nutshell the emptiness problem for an automaton on infinite trees can be modelled as a game where Éloïse builds an input tree together with a run while Abélard tries to exhibit a rejecting branch in the run. Hence, the emptiness problem for tree automata can be reduced to solving a two-player parity game played on a *finite* graph. Beyond these results, the tight connection between automata and games is one of the main tools in automata theory [4,8,9].

There are several levers on which one can act to define alternative families of tree automata/classes of tree languages. A first lever is *local* with respect to the run: it is the condition required for a branch to be accepting, the reasonable options here being all classical ω-regular conditions (reachability, Büchi, parity...). A second one has to do with the set of runs. The usual definition is existential: a tree is accepted if there exists an accepting run on that tree. Other popular approaches are universality, alternation or probabilistic transition functions. A third lever is *global* with respect to the run: it is the condition required for a run to be accepting. The usual definition is that *all* branches must be accepting for the run to be accepting but one could relax this condition by specifying *how many* branches should be accepting/rejecting. One can do this either by *counting* the number of accepting branches (*e.g.* infinitely many, uncountably many) or by counting the number of rejecting branches (*e.g.* finitely many, at most countably many): this leads to the notion of automata with cardinality constraints [10, 11]. As these properties can be expressed in MSOL [12], the classes of languages accepted under these various restrictions are always ω-regular. However, this logical approach does not give a tractable transformation to standard parity or Büchi automata. Another option is to use a notion of topological "bigness" and to require for a run to be accepting that the set of accepting branches is big [13,14]. Yet another option considered in [15–17] is to *measure* (in the usual sense of measure theory) the set of accepting branches and to put a constraint on this measure (*e.g.* positive, equal to one).

The idea of allowing a certain amount of rejecting branches in a run was first considered by Beauquier, Nivat and Niwiński in [10,11], where it was required that the number of accepting branches in a run belongs to a specified set of cardinals Γ. In particular, they proved that if Γ consists of all cardinals greater than some γ, then one obtains an ω-regular tree language. Their approach was based on logic (actually they proved that a tree language defined by such an automaton can be defined by a Σ^1_1 formula hence, can also be defined by a *Büchi* tree automaton) while the one we develop here is based on designing acceptance games. There is also work on the logical side with decidability results but that do not lead to efficient algorithms [12].

Our main contributions are to introduce (automata with cardinality constraints on the number of rejecting branches; automata with topological bigness constraints) or revisit (automata with cardinality constraints on the number of accepting branches) variants of tree automata where acceptance for a run allows a somehow negligible set of rejecting branches. For each model, we provide a game counterpart by means of an equivalent acceptance game and this permits to retrieve the classical (and fruitful) connection between automata and game. It also permit to argue that languages defined by those classes are always ω-regular. Moreover, in the case where one counts accepting branches we show that the languages that we obtain are always accepted by a *Büchi* automaton, which contrasts with the case where one counts rejecting branches where we exhibit a counter-example for that property.

The paper is organised as follows. Section 2 recalls classical concepts while Section 3 introduces the main notions studied in the paper, namely automata with cardinality constraints and automata with topological bigness constraints. Then, Section 4 studies those languages obtained by automata with cardinality constraints on the number of rejecting branches while Section 5 is devoted to those languages obtained by automata with cardinality constraints on the number of accepting branches. Finally, Section 6 considers automata with topological bigness constraints.

[3] Note that the idea of using games to prove this result was already proposed by Büchi in [7].