Theoretical Computer Science ●●● (●●●●) ●●●–●●●

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Counter machines, Petri Nets, and consensual computation [☆]

Stefano Crespi Reghizzi [a,b,*], Pierluigi San Pietro [a,b]

[a] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy*
[b] *CNR IEEIT, Milano, Italy*

ARTICLE INFO

Article history:
Received 2 March 2015
Received in revised form 29 October 2015
Accepted 4 December 2015
Available online xxxx

Keywords:
Formal languages
Multi-counter machine
Real-time partially blind machine
Petri Net language
Petri Net normal form
Quasi-deterministic counter machine
Consensual language
Multiset machine
Modulo scheduling

ABSTRACT

We establish the relation between two language recognition models that use counters and operate in real-time: Greibach's partially blind machines operating in real time (RT-PBLIND), which recognize Petri Net languages, and the consensually regular (CREG) language model of the authors. The latter is based on synchronized computational threads of a finite automaton, where at each step one thread acts as the leader and all other threads as followers. We introduce two new normal forms of RT-PBLIND machines (and Petri Nets), such that counter operations are scheduled and rarefied, and transitions are quasi-deterministic, i.e., the finite automaton obtained by eliminating counter moves is deterministic. We prove that the CREG family can simulate any normalized RT-PBLIND machine, but it also contains the non-RT-PBLIND language $\{a^n b^n \mid n > 1\}^*$.

1. Introduction

Multi-Counter Machines (MCM) have been used since half a century to model integer computer programs [20,21] and to recognize formal languages, yet their theory is less established than, say, the theory of push-down machines and context-free languages. Several MCM types exist depending on the operations permitted on counters, on determinism, and on other constraints about reversals and spontaneous moves (see e.g., [11,13,17]). Other language families offer a double characterization by grammars and by machines, but only the most restricted MCM subfamilies (such as the one-counter languages) enjoy some sort of generative model.

Our nontraditional approach to MCM languages offers some promise to clarify the complex computations that have so far hindered their full understanding. Notably, we obtain the possibility to specify counter languages by means of regular expressions that describe interacting computational threads in a rather perspicuous way.

This paper focuses on a classical and important MCM family: the *real-time partially blind machines* [13] to be denoted by RT-PBLIND. Counters are nonnegative integers that can be tested for zero only in the final configuration; if the machine attempts to decrement a zero counter, it crashes. In terms of languages, such machines are equivalent to a natural family of Petri Nets firing sequences [13]. They are more powerful than the more popular reversal-bounded MCM [17]. The RT-

---

[☆] Work partially supported by MIUR project PRIN 2010LYA9RH-006.

[*] Corresponding author at: Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milano 20133, Italia.

E-mail addresses: stefano.crespireghizzi@polimi.it (S. Crespi Reghizzi), pierluigi.sanpietro@polimi.it (P. San Pietro).

http://dx.doi.org/10.1016/j.tcs.2015.12.004
0304-3975/

Please cite this article in press as: S. Crespi Reghizzi, P. San Pietro, Counter machines, Petri Nets, and consensual computation, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2015.12.004

PBLIND language family is closed with respect to union and intersection, concatenation, and alphabetic (direct or inverse) homomorphism; it includes also languages that have a nonsemilinear Parikh image.

The other model, the *consensually regular* languages (CREG), was introduced by the authors in [5] and further studied in [6,7,9,8,10]. Deferring its formal presentation to Sect. 3, we intuitively describe its functioning. Consider a nondeterministic real-time finite-state automaton (NFA), and classify each edge of the transition graph as *leader* or *follower*. To *consensually* recognize an input word w of length $|w|$, the NFA may need one or more computations, called threads, each of them accepting the word, and such that at each step $1 \le i \le |w|$, one, and only one, thread traverses a leader edge. Therefore at time i all remaining threads traverse a follower edge. In some sense, this *matching* discipline is similar to a token-passing scheduling policy, the temporary leader being the thread with the token. The standard finite-state recognition corresponds to the case of a thread that, for all moves, is a leader. The adjective *consensual* expresses the metaphor that the leader thread reads the current input character, but without the consent of all other threads the computation cannot proceed.

The number of threads i.e., the parallelism of the consensual device, is bounded by the input length, since a thread that at all times is a follower would be useless. The memory of the consensual device is encoded in the current states of all active threads, and can be represented by a multi-set of states, which motivates the name of *multiset machine*. Such machine could be rightly called a multi-counter machine, but we prefer to reserve this name to the classical models.

Word recognition for an RT-PBLIND (Petri Net) language requires logarithmic space complexity, and the same property holds for the multi-set machine. Apart from this similarity, the two models look at first glance very different, but, in an effort to assess the power of CREG, we recently proved that some RT-PBLIND languages are consensually regular: the deterministic RT-PBLIND [8] languages, and the commutative languages that have a semilinear Parikh image [10]. Here we prove a much more general and unexpected result: the strict inclusion of nondeterministic RT-PBLIND (therefore also of Petri net) languages in CREG. While the two mentioned inclusions had been proved by special transformations of consensual languages, here we directly simulate an RT-PBLIND machine on a multiset machine. An RT-PBLIND move may be nondeterministic in two manners: by moving to multiple next states (as an NFA) and by performing different increments or decrements of counters. For the latter manner, we prove that any RT-PBLIND machine can be transformed to an equivalent RT-PBLIND machine, called *quasi-deterministic* (QD), such that the underlying finite-state automaton is deterministic. Then we show how to simulate any QD RT-PBLIND machine on a multiset machine. At last, the strict inclusion RT-PBLIND $\subset$ CREG and the incomparability of non-RT PBLIND machines and CREG are proved by means of witnesses.

To obtain the QD normal form, we reschedule each counter operation at an ordinal time, which belongs to a specific congruence class, modulo a large enough integer. Machine moves are in this way simplified (at most one counter is affected in one move) and counter operations are rarefied. Such transformations are not new for the more powerful MCM types that are allowed to test counters for zero [11], but in our case we had to develop a very thorough analysis.

We briefly mention similarities and differences of CREG with respect to some language families other than the multi-counter ones, which have been already considered.

Essential features of concurrent and parallel computations are captured in various, old and recent, formal models. Several models compose sequential threads, represented by strings, by means of the *interleaving* or *shuffle* operation. The concurrent regular expressions of [12] are an example; a more recent one is the work on *shuffled languages* [2]. To augment the effect of the pure shuffle operation on regular languages, various directions have been explored. Several authors, e.g., the two latter citations, add the transitive closure of the shuffle. Another possibility for enlarging the language family, exploits the synchronized shuffle operation, which exists in different versions, see [26]. Our consensual model is based on a lock-step synchronization called matching, that bears some similarity to a synchronized shuffle.

We notice another interesting analogy between the multiset consensual device and other devices based on multiple coordinated runs of a finite-state machine. Such devices have been proposed in different settings, as in the machine that recognizes the shuffle closure of regular languages [19]. A recent more complex device is the concurrent machine that recognizes the shuffled languages of [2]: such machine stores in its configuration an unbounded number of states, organized as a tree, that assigns a partial order to state activation. In contrast, the consensual device simply stores a multiset of states, and requires that all states consensually fire at each step.

At last, there is an enduring interest for models that are able to represent the commutation or partial commutation of words, starting from classical language families. Two recent examples are the restarting automata [23] and the context-free grammars enhanced with rules that interchange adjacent subtrees [22]. The consensual model is not entirely dissimilar and is able to recognize a subfamily of commutative languages [10].

To sum up, in our opinion the consensual model is not directly comparable with any existing proposals, and can be considered as a minimalist attempt to capture some language features pertinent to concurrent and parallel computations, using a simple yet intriguing extension to the finite-state model.

Concerning applications, our new normal forms may be interesting for proving properties of multi-counter machines and Petri Nets. Since consensual languages can be defined by finite automata or by regular expressions on an alphabet, which enriches the terminal alphabet with leader/follower roles, it becomes possible to define multi-counter and Petri Net languages using such established notations. We hope that this novel specification style will be convenient in some areas, such as formal specifications, where counter machines are used.

The paper is organized as follows. Sect. 2 deals with RT-PBLIND machines and Petri Nets. It defines the classical models and introduces two new normal forms: the rarefied and modulo-scheduled form, and the QD form. It describes the corresponding normal forms of Petri Nets. Sect. 3 defines the CREG languages and devices and recalls known properties. Sect. 4

Please cite this article in press as: S. Crespi Reghizzi, P. San Pietro, Counter machines, Petri Nets, and consensual computation, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2015.12.004