SYNOPSIS
of the Ph.D. Thesis entitled

Counter automata and classical logics for data words

by

Amaldev Manuel
Institute of Mathematical Sciences, Chennai.

Proposed to be submitted to the
Board of Studies for Mathematical Sciences

In partial fulfilment of requirements
For the degree of

DOCTOR OF PHILOSOPHY
of
Homi Bhabha National Institute

July 2011

Synopsis

Introduction

The formalism of languages over finite alphabets is well-suited for abstracting sequential behaviour of computing systems such as execution traces of programs and plays of games. Hence the multipronged — algebraic, logical, automata-theoretic — study of languages over finite alphabets has contributed effectively to the verification of software and control systems. A standard approach is the following. The program or control system under scrutiny is abstracted as a finite state system and the specification is written in a specification language, very often a suitable logic. It is then checked that the finite state system obeys the specification by comparing the languages described by the system and the specification. A necessary premise for this method is the effectiveness of checking non-emptiness of the language defined by the system and checking satisfiability of the formula. In the case of finite state systems this is guaranteed by the notion of regularity, to which we will return shortly. The above approach to program verification, called model checking [CGP99], has been found very effective in the verification of stand-alone reactive systems like control systems in automobiles.

This thesis takes shape in an ongoing effort to find a suitable formalisms, both automata theoretic and symbolic, for languages over *infinite alphabets*. Infinite alphabets are an obvious way to abstract unboundedness often occurring in many areas of computer science. Natural examples are values of variables in programs, process ids in distributed computing, nonces in security protocols, attribute values in XML, keys in data bases etc. Apparent uses of such mechanisms are many fold [MRR+08], the single most important application being in verification.

Words over infinite alphabets

Let Σ be a finite alphabet and Γ be an infinite set in which membership and equality are decidable. We call finite sequences of elements of the set $\Sigma \times \Gamma$

data words. Formally a data word w is in $(\Sigma \times \Gamma)^*$.

The course of study of data languages so far has been driven by two important questions, which are (1) what is a suitable class of automata for recognizing data languages? (2) what is a suitable logical language for expressing data languages? The contributions of this thesis are to be seen in the light of these two questions which we discuss briefly below.

Automata for data words

We mentioned above that the effectiveness of finite state model checking is expediated by the presence of a class of languages captured by the notion of regularity. In the case of finite words regularity is synonymous with the confluence of the following properties: closure under boolean operations, low complexity of the decision problems such as membership and non-emptiness, alternate characterizations in terms of logics and algebras and robustness in terms of machine characterization in the sense that the restriction of determinism or the addition of alternation, two-way-ness or finite memory does not break the characterization.

An important question is whether there is a regular class of data languages. As of now the literature does not possess such a class. Of the above properties the decidability of non-emptiness problem plays a pivotal role in verification. Hence, unsurprisingly a good amount of time and energy has been invested in finding classes of automata with decidable non-emptiness problem.

The general approach, so far, for designing automata for data words has been to augment a finite state automaton with memory structures. This idea traces its origin to the dawn of automata theory in the fifties and sixties when an intensive study of automata with various memory structures such as stacks, queues, pushdowns, counters, tapes etc. was performed. Following this line, most important classes of automata known for data words employ structures such as registers, hash tables, counters, stacks, pointers etc.

Among the various automaton models proposed for data languages, two, Register automata and Data automata got particular attention. A Register automaton [KF94, DL09] is a finite state automaton equipped with a finite number of registers which can hold data values. The transitions depend on the state of the automaton as well as the register configuration. It is easy to observe that since the registers are only finitely many the automaton is unable to keep track of all the data values it has seen, thus incapable of recognizing the language "all data values occurring in the word are different". However register automata are akin to finite state automata in the sense that the string projections of the language accepted by a register automaton is regular. The nonemptiness problem of register

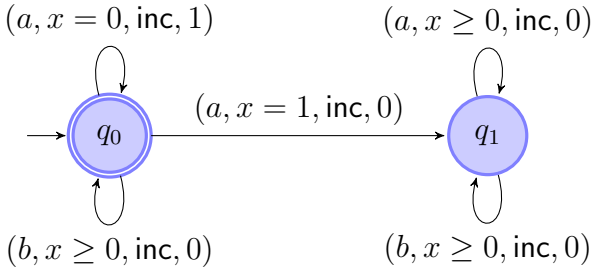

Figure 1: CCA accepting the language "All data values under a are distinct"

automata is NP-complete. A Data automaton [BMS$^+$06], is a composition of two finite state machines where regular properties over the entire word and over each class can be checked. They strictly subsume register automata in terms of the set of accepted languages. They are capable of accepting languages like "all data values under a are distinct", "every data value occurring under a occurs under b" etc. However their nonemptiness problem is of very high complexity (not known to be elementary). Both the above automata are not complementable.

Our approach to the automaton problem involves enhancing finite state automata with counters. Counters are a primitive and minimal computational device where the operations are increment, decrement and checking for zero. Yet it is long been known that automata with two counters are as powerful as Turing machines. Hence it is necessary to restrict the operations on the counters. There are standard restrictions in the literature. Some of them are (1) disallowing the decrement operation (2) removing the two-way branching on a zero test (3) allowing counter values to be negative etc.

We now briefly describe the class of machines we call Class Counting Automata. A class counting automaton $A = (Q, \Sigma, \Delta, I, F)$ is a finite state automaton with $|\Gamma|$-many counters where Q is the finite set of states, Δ is the transition function and $I \subseteq Q$ and $F \subseteq Q$ are the set of initial and final set of states. A configuration of the automaton is of the form (q, h) where $q \in Q$ and $h : \Gamma \to \mathbb{N}$ is a function holding the counter values. The transition of the automaton are of the form $(p, a, \varphi(x), u, q)$ where p, q are the entry and exit states of the transition, $\varphi(x)$ is a univariate linear inequality and u is from the set $\{\mathsf{inc}, \mathsf{reset}\}$. The intended semantics of the transition is that on a configuration (p, h) of the automaton on the pair (a, d) the transition $(p, a, \varphi(x), u, q)$ can be taken if $\varphi(h(d))$ is true. The resulting configuration will be (q, h') where h' is h for all but d where $h'(d) = h(d) + 1$ if u is inc and $h'(d) = 0$ if reset.

Theorem 1. *The nonemptiness problem of CCA is EXPSPACE-complete.*

Note that the complexity is to be contrasted with that of register automata (NP-complete) and that of data automata (not known to be elementary). The model checking problem of CCA is NP-complete. Addition of alternation or two-wayness leads to undecidability of the nonemptiness problem.

CCA are closed under union and intersection but they are not closed under complementation. The deterministic fragment is closed under complementation but is strictly less powerful. It is not known whether they subsume register automata.

We also study several extensions and restrictions of Class Counting automata that are equivalent to Register automata and Data automata.

Logics for data words

Various modal and classical logics are used for specifying properties over words over a finite alphabet. On the classical side, monadic second order logic and first order logic are the most important ones, while modal languages, very attractive due to their lower complexity and intuitiveness, include linear and branching time temporal logics.

For the purpose of verification the most important aspect regarding a logic is the decidability of the model checking and the satisfiability. A whole lot of other questions reduce to checking satisfiability, for example checking implication between two formulas, checking validity of a formula etc. From a practical point of view finite satisfiability problem ("is there a finite model satisfying the formula?") looks more attractive than the general problem.

This thesis focuses on classical logics on data words. For this purpose, data words can be respresented as a first order structure $w = ([n], \Sigma, <, +1, \sim)$ extending the corresponding representation for words due to Büchi. Here $[n]$ denotes the set $\{1, \ldots, n\}$, and Σ stands for the unary relations indicating the alphabet labelling. The binary relations $<, +1$ are interpreted as the natural linear order and successor relations on the set $[n]$. The binary relation $\sim$ denotes the equivalence relation on $[n]$ given by the data values based on equality. That is to say, $i \sim j$ if $d_i = d_j$. In addition if we have a linear order $<_\Gamma$ on the data values then this uniquely defines a total preorder $<_p$ (a total preorder is a reflexive, transitive and total binary relation) on the positions $[n]$. We denote the successor relation of $<_p$ by $+1_p$. In the following we denote linear orders and their successor relations by $<_{l_1}, <_{l_2}, \ldots$ and by $+1_{l_1}, +1_{l_2}, \ldots$.

It is easy to see that satisfiability and finite satisfiability of first order logic on data words, $\mathrm{FO}(\Sigma, <, \sim)$ is undecidable. The problems remain undecidable even for the fragment $\mathrm{FO}^3(\Sigma, <, \sim)$, the set of formulas which uses at most 3

variables. Hence for decidability one has to look for suitable restrictions which are sufficiently expressive. Two-variable fragment is a natural candidate. It is known that satisfiability problem for first order logic with two variables is decidable [Mor75, GKV97]. Since it is not expressible in FO^2 that a binary relation R is a linear order (or an equivalence relation or a preorder), the above theorem does not imply satisfiability of FO^2 over data words or over ordered data words. In a landmark paper [BMS$^+$06] it was shown that,

Theorem 2 ([BMS$^+$06]). *Finite satisfiability of* $\mathrm{FO}^2(\Sigma, <_{l_1}, +1_{l_1}, \sim)$ *is decidable.*

Note that $+1_{l_1}$ is not definable in terms of $<_{l_1}$ using two-variables over words. This prompts us to add both the order and successor relations of the linear order to the vocabulary. [As a side note, it is also the case that $+1_{l_1}$ is not definable in terms of $<_{l_1}$ and $\sim$ using two-variables over words.] Decidability holds even when $<_{l_1}$ is the ordinal ω. Status of the infinite satisfiability problem is not known.

However, the theorem fails for ordered data words;

Proposition 1 ([BMS$^+$06]). *Finite satisfiability problems of* $\mathrm{FO}^2(\Sigma, <_{l_1}, +1_{l_1}, <_{p_2})$ *and* $\mathrm{FO}^2(\Sigma, <_{l_1}, +1_{l_1}, +1_{p_2})$ *are undecidable. In fact, undecidability persists even when the equivalence classes of* $<_{p_2}$ *are of size atmost* 2.

This implies that in the presence of a total order on data values to get back decidability either $<_{l_1}$ or $+1_{l_1}$ has to be dropped from the vocabulary. The former case was undertaken in [SZ10] where it was shown that $\mathrm{FO}^2(\Sigma, <_{l_1}, <_{p_2}, +1_{p_2})$ is decidable. We consider the latter case when the preorder is in fact a linear order (in the case of data words it corresponds to the scenario when all the data values are distinct) and show that,

Theorem 3. *Finite satisfiability of* $\mathrm{FO}^2(\Sigma, +1_{l_1}, +1_{l_2})$ *is decidable.*

Proposition 2. *Finite satisfiability of* $\mathrm{FO}^2(\Sigma, <_{l_1}, +1_{l_1}, <_{l_2}, +1_{l_2})$ *is undecidable.*

Note that this line of work is interesting on its own [Ott98]. Our proof is automata theoretic and makes use of Presburger automata. Concurrently, it was shown that removing at least one successor relation also results in decidability [SZ10], that is;

Theorem 4 ([SZ10]). *Finite satisfiability of* $\mathrm{FO}^2(\Sigma, <_{l_1}, +1_{l_1}, <_{l_2})$ *is decidable.*

This raises the question whether FO^2 is decidable if one of the order relations is absent from the vocabulary. This question is answered in the positive. In fact, a more general theorem is proved which says that $\mathrm{FO}^2\,(\Sigma, +1_{l_1}, <_{p_2}, +1_{p_2})$ is decidable where $+1_{l_1}$ is a successor of a linear order and $<_{p_2}, +1_{p_2}$ are a total preorder and its successor relation where the equivalence classes of the preorder is bounded by a constant. Note that it is to be contrasted with Proposition 1.

Theorem 5. *Finite satisfiability of* $\mathrm{FO}^2(\Sigma, +1_{l_1}, <_{p_2}, +1_{p_2})$ *is decidable when classes of* $<_{p_2}$ *are of size at most* k*.*

For the proof, the notion of data automata are generalized so that they accept ordered data words. A translation from the above logic to these automata is established and finally the non-emptiness of these automata are shown to be decidable by reduction to reachability problem in vector addition systems. Since it is definable in FO^2 that $<_{p_2}$ is a linear order, this implies the answer to the previous question.

Corollary 1. *Finite satisfiability of* $\mathrm{FO}^2(\Sigma, +1_{l_1}, <_{l_2}, +1_{l_2})$ *is decidable.*

Though it is decidable, it is proved that the problem is as hard as reachability in vector addition systems.

Organization of the thesis

In Chapter 1 we motivate and introduce the notion of data languages. We set out the goals regarding designing the automata. In particular we mention what is expected of regular data languages.

In Chapter 2 Register automata and Data automata are introduced and major facts about them are briefly surveyed. We introduce the model of Class Counting automata and give examples. Some extensions of Class counting automata are also detailed.

In Chapter 3 we study the non-emptiness problem of class counting automata and its variants.

In Chapter 4 we introduce first order logic over data words and show basic undecidability results. The landmark results on two-variable logic over data words are outlined.

In Chapter 5 it is shown that two-variable logic with two successor relations is decidable.

In Chapter 6 two-variable logic on k-bounded ordered data words is studied. A number of undecidability results are also proved here.

In Chapter 7 we conclude by summarize by a comparison of automaton models in terms of expressiveness and complexity of nonemptiness problems. The complexity of satisfiability problems of the logics is discussed.

Publications

1. Amaldev Manuel, R. Ramanujam. Counting multiplicity over infinite alphabets. In *RP*, Volume 5797 of LNCS, pages 141-153, 2009.

2. Amaldev Manuel, R. Ramanujam. Class counting automata on data words. IJFCS, Volume 22(4), pages 863-882, 2011.

3. Amaldev Manuel. LTL with a sub-order. Proceedings of ESSLLI, pages 21-28, 2009.

4. Amaldev Manuel, R. Ramanujam. Automata over infinite alphabets. In *Modern Applications of Automata Theory*, D'Souza and Shankar eds., World Scientific, 2011 (to appear).

5. Amaldev Manuel. Two variables and two successors. In *MFCS*, Volume 6281 of LNCS, pages 513-524, 2010.

Submitted

1. Amaldev Manuel, Thomas Zeume. Two-Variable logic with a linear successor and a preorder. (submitted)

References

[BMS+06] Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *LICS*, pages 7–16. IEEE Computer Society, 2006.

[CGP99] Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[DL09] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

[GKV97] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[KF94] Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[Mor75] M. Mortimer. On languages with two variables. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 21:135–140, 1975.

[MRR+08] Anca Muscholl, R. Ramanujam, Michaël Rusinowitch, Thomas Schwentick, and Victor Vianu. *Beyond the Finite: New Challenges in Verification and Semistructured Data*. Dagstuhl Seminar 08171 Proceedings. 2008.

[Ott98] Martin Otto. Two variable first-order logic over ordered domains. *J. Symb. Log.*, 66:685–702, 1998.

[SZ10] Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations. In *CSL*, volume 6247 of *LNCS*, pages 499–513, 2010.