Counting multiplicity over infinite alphabets

Amal Dev Manuel and R. Ramanujam

The Institute of Mathematical Sciences, Chennai, India

{amal,jam}@imsc.res.in

Summary

- Motivation for infinite data.
- We need good data models, amenable to decidable verification.
- Crucial decision: operations and predicates on data.
- Our proposal: count data value occurrences (subject to constraints).
- Decidable automaton model.
- Interesting connections to logics.

Data in verification

- Semi-structured data: documents viewed as ranked / unranked trees with labels from finite domain.
- Software verification:
  - Control structures: Procedure calls, dynamic process creation.
  - Data structures: integers, lists, pointers.
  - Communication channels: unbounded buffers.
  - Parameters: Number of processes, communication delays.

A resourceful tale

Two processes: $\{r_i, s_i, t_i\}$ for request, start and terminate.

- Local property: in any computation, the i-projection is of the form: $(r_i s_i t_i)^*$.
- Global property: between any s_i and subsequent t_i, there is no s_j or t_j, where $j \neq i$.

What happens when the number of processes is either unknown, or changes during computation ?

Model checking infinite state systems

An active research area.

- A typical approach:
  - Describe system states by finite objects (strings).
  - Describe possible transitions by rewriting rules.
  - Devise algorithms for checking reachability.
- Model checking of linear time properties possible in many cases.
- Missing: reasoning about data across states (as above).
- Missing: A generic framework for branching time properties.

Decidability issues

- Parametrized verification: property refers to process actions indexed by process ID: requires an infinite alphabet.
- Apt and Kozen 1986: Parametrized verification is undecidable.
- Decidability obtained using network invariants, regular model checking.

Decidability issues

- Parametrized verification: property refers to process actions indexed by process ID: requires an infinite alphabet.
- Apt and Kozen 1986: Parametrized verification is undecidable.
- Decidability obtained using network invariants, regular model checking.

Emerson, Namjoshi 2005: indexed processes in CTL^*: decidability obtained by showing that the properties studied have constant cutoffs, using symmetry arguments.

A uniform framework

Similar considerations in dealing with semistructured data.
Enhance finitely labelled structures by data.

- One or more relations per node.
- Parameters:
  - Underlying structure.
  - Amount and structure of data at each node.
  - Operations and predicates on data.
  - Expressiveness of specification language.

Regular languages

Regular languages over finite alphabets: a robust notion.

- There does not seem to be a canonical notion of regular data languages.
- But we can mimic the regular languages framework.
- Some automata models have been studied.

Example languages

Some standard examples.

- No two a positions have same data value.
- There exist two a positions have same data value.
- For every a position, there exists a b position with the same data value.
- A process has to consume one given resource before requesting another.
- Every process requesting a resource is eventually granted.
- Only one process has the resource at any time.

Need for a theory

- We look for a decent theory of regular-like word and tree languages over infinite alphabets.
- Decent = decidable emptiness problem, with manageable complexity.
- Better, equivalent logical / algebraic characterizations.

Only equality comparisons on the infinite alphabet.

Data languages

- $(\Sigma \times D)$-labelled words, where Σ is finite and D is infinite.
- Data word language $L \subseteq \Sigma^*_\sim$,
- Data trees: the same notion, over unranked ordered trees.

Since we have only equality tests on values, positions in data words are partitioned into classes; similarly nodes in trees are equated.

Reasoning

- Books that have been re-edited:

$$\exists y.\ (x.isbn = y.isbn \land x.year \neq y.year)$$

- Unary keys: attribute A has distinct values:

$$\forall x, y.\ (x.A = y.A \implies x = y)$$

- Navigation: From node x we can access nodes y_1, y_2 via paths of type $p_1, p_2 \in R$ such that $y_1.B = y_2.B$.

Register automata

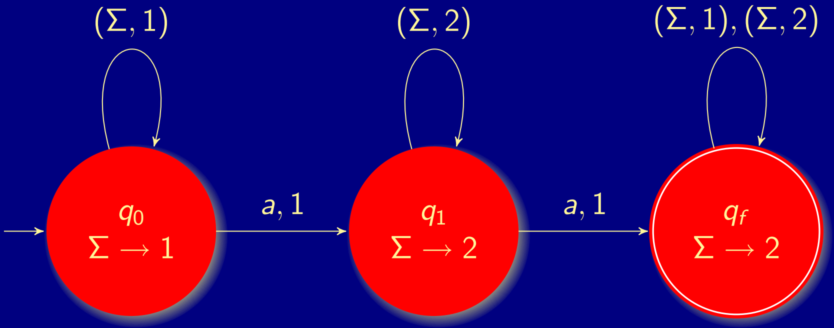
k-register automata: upon reading $(a, v) \in (\Sigma \times D)$, one can check in which register value v occurs, can store v into a register.
Let $L \subseteq \Sigma^*_\sim$. Define:

$$Proj(L) = \{a_1 \dots a_n \mid \exists (a_1, v_1) \dots (a_n, v_n) \in L\}$$

- If L is recognized by a k-RA M, then $Proj(L)$ is regular.
- From M one can construct a word automaton M' of size $|M| 2^{O(k^2)}$.
- Proof idea: Consider the matrix $\{=, \neq\}^{k \times k}$ for keeping track of equal registers; guess (in)equalities on the fly.

Example

All data values occurring with letter a are distinct.



Results

Non-emptiness for register automata is decidable.

- There are subtle differences between register automata models. In some, data values can occur in more than one register; in some they cannot.
- In the former, the problem is PSpace-complete; in the latter it is NP-complete.
- The mode is not expressive: local properties, like every projection is of the form $(r_i s_i t_i)^*$, cannot be expressed.

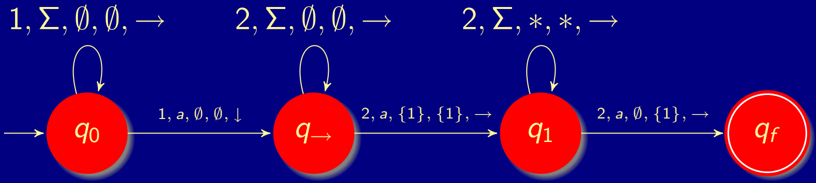
Pebble automata

Upon reading $(a, v) \in (\Sigma \times D)$,

- check which pebbles are under the head;
- check which pebbles mark positions with v
- can lift highest pebble, with head reverting to previous pebble, and place new pebble.

Example

There are at least two positions with a having the same data value.



Data logics

$FO(+1, <, \oplus 1, \sim)$:

- atomic predicates $P_a(x)$, for $a \in \Sigma$.
- $+1$ for successor position, $<$ order on positions.
- $\sim$ same value relation, $\oplus 1$ for class successor.

$EMSO(+1, <, \oplus 1, \sim)$, similarly.
Models: data words.

Examples

Consider $FO(+1, <, \sim)$:

- Every position labelled with a has a distinct value:

$$\forall x, y.(P_a(x) \wedge P_a(y) \wedge x \neq y) \implies \neg(x \sim y)$$

.

- Complement of the language above: words containing two positions labelled with a having the same data value:

$$\exists x, y.(P_a(x) \wedge P_a(y) \wedge x \neq y \wedge x \sim y)$$

.

- Inclusion dependence: every position labelled with a has a value which appears under a position labelled with b:

$$\forall x.\exists y.(P_a(x) \implies (P_b(y) \wedge x \sim y))$$

.

Examples

Sequences over $\{0,1\}$ with the same subsequence of 0-values and 1-values:

- All 0's have distinct values; similarly for 1's.
- There is a bijection between 0-values and 1-values.
- For every pair of 0-positions $x < y$ and every 1-position z with $x \sim z$, there exists a 1-position z' such that $z < z'$ and $y \sim z'$.

Needs 3 variables, accepted by 2-PA.
Earlier examples, plus:

$$\forall x, y, z.(P_0(x) \wedge P_0(y) \wedge x < y \wedge P_1(z) \wedge x \sim z)$$

$$\implies (\exists x.(P_1(x) \wedge x \sim y \wedge z < x))$$

.

Undecidability

$FO^3(S, \sim)$ is undecidable (and therefore $FO^3(<, \sim)$ since S is definable from $<$ when we can use 3 variables).

- PCP reduction: Given instance I over alphabet Σ, let Σ' consist of two disjoint copies of Σ.

Regular $\cap$ EqualSequences

Expressive power

- $\mathit{FO}(+1, <, \sim)$ is incomparable with register automata.
- $\mathit{FO}(+1, <, \sim)$ is strictly included in pebble automata.
- The two-variable fragment is a decidable fragment, but almost all natural extensions are undecidable.

E.g. $\mathit{FO}^2(+1, <, \sim, \preceq)$ with a linear order on data values.

Two variable FO

Why consider a two variable logic, at all ?

- More hope for decidability. Rich structure over words.
- Core XPath without attributes $= FO(+1, <)$ over trees [Gottlob et al '02, Marx '05].
- Core XPath with one attribute $\supseteq FO(+1, <, \sim)$.

Two variable logics

- FO^2 over graphs has finite model property [Mortimer '75]; is NEXPTIME complete [Graedel, Otto '99].
- Over words, FO^2 is equivalent to:
  - unary LTL and $\Sigma^2 \cap \Pi^2$ [Etessami, Vardi Wilke '02].
  - the variety DA [Therien, Wilke '98].
  - 2-way partially ordered DFA [Schwentick, Therien, Vollmer '01].

and is NEXPTIME complete.

Decidability

$FO^2(+1, <, \sim)$ is decidable [Bojanczyk, Muscholl, Schwentick, Segoufin, David '06].

- For each formula ϕ construct a data automaton that accepts $L(\phi)$.
- For each data automaton accepting L, construct a multicounter automaton that recognizes $str(L)$.
- 2-EXPTIME reduction.

Data automata

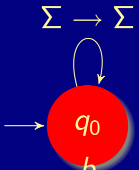
Data automaton (A, B): A, the base automaton, is a nondeterministic letter-to-letter transducer.
B, the class automaton, is an NFA.

- A outputs a word x over a finite alphabet.
- B checks, for each $\sim$-class, that the subword of x corresponding to the class is accepted.

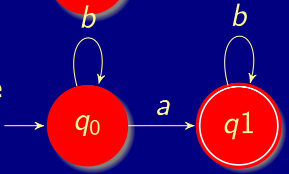
Example

Every data value occurring under *a* is distinct.

The transducer



The finite automaton



FO and data automata

Above, we saw that FO^2 definable data languages are recognizable by data automata.
But the converse does not hold.

- Consider the property: each class is of even length. This is not FO-definable.
- A prefix of second order existential quantifiers helps.
- Still not good enough; describing an accepting run needs a comparison of successive positions in the same class.

$EMSO^2(+1, <, \sim, \oplus 1) = \mathrm{DA}$.

FO to DA

Scott normal form: every formula equivalent to

$$\forall x \forall y . \chi \wedge \bigwedge_i \forall x \exists y . \chi_i$$

where the χ_i and χ are quantifier free, but over an extended signature with unary predicates.

- Hence equivalent to $\exists R_1 \ldots \exists R_m$ followed by a Scott formula.
- Careful rewriting to ensure that innermost conjuncts are all of the form base type or $x \sim y$, $x \neq y$, $x < y$ etc.
- Then construct data automata for each case, and use closure under intersection, union and renaming.

Multicounter automata

Finite automata $+$ positive counters.
Equivalent to Petri nets.

- No test for zero (except at the end).
- Acceptance by final state all counters $= 0$.

Emptiness decidable [Mayr, Kosaraju '84].
Not known to be elementary.

DA to multicounter automata

Show that $Proj(L)$ can be obtained as $Shuf(L') \cap R$.

- When $Proj(L)\{a^n b^n | n \geq 0\}$, each class contains one a and one b to its right; i.e. $Proj(L) = Shuf(\{ab\}) \cap a^* b^*$.
- Marked shuffle of n words: use n colours.
- When L is regular, $Shuf(L)$ is recognized by a multicounter automaton [Gischer '81].

Counter automata

Counter mechanisms in the context of unbounded data.

- Each "event type" (occurrence of a data vaue, occurrence of a letter - value pair, etc) needs its own counter.
- Hence we need unboundedly many counters.
- A restraint on counter operations: monotone counters.
- Can be incremented, reset or compared against constants.

The proposal

An automaton model for counting multiplicity of data values.

- The automaton includes a bag of infinitely many monotone counters, one for each possible data value.
- When it encounters a letter - data pair, say (a, d), the multiplicity of d is checked against a given constraint, and accordingly updated, the transition causing a change of state, as well as possible updates for other data as well.
- A bag is like a hash table, with elements of D as keys, and counters as hash values.
- Transitions depend only on hash values (subject to constraints) and not keys.

The model

- A **constraint** is a pair $c = (op, e)$, where $op \in \{<, =, \neq, >\}$ and $e \in N$.
- Define a *bag* to be a map $h : D \to N$.
- $Inst = \{\uparrow^+, \downarrow\}$, the set of *instructions*.
- $CC_A = (Q, \Delta, I, F)$, where: $\Delta \subseteq (Q \times \Sigma \times C \times Inst \times U \times Q))$, where U is a finite subset of $\mathbb{N}$.

Behaviour

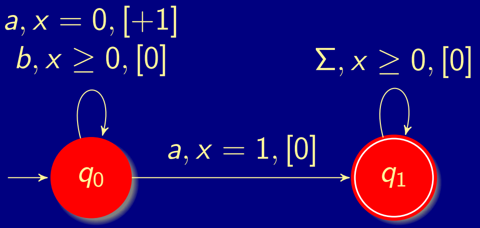
A configuration is a pair (q, h), where $q \in Q$ and $h \in B$. The initial configuration of A is given by (q_0, h_0), where $\forall d \in D, h_0[i](d) = 0$ and $q_0 \in I$.

- Given a data word $w = (a_1, d_1), \ldots (a_n, d_n)$, a run of A on w is a sequence $\gamma = (q_0, h_0)(q_1, h_1) \ldots (q_n, h_n)$ such that $q_0 \in I$ and for all $i, 0 \leq i < n$, there exists a transition $t_i = (q, a, c, \iota, n, q') \in \Delta$ such that $q = q_i$, $q' = q_{i+1}$, $a = a_{i+1}$ and:
  - $h_i(d_{i+1}) \models c$.
  - h_{i+1} is given by:

$$h_{i+1} = \left\{ \begin{array}{lll} h_i \oplus (d, n') & \text{if} & \iota = \uparrow^+, n' = h_i(d) + n \\ h_i \oplus (d, n) & \text{if} & \iota = \downarrow \end{array} \right\}$$

Example

All data values under a are distinct.



Examples

- The language $L_{\mathrm{fd(a)}} =$ *"Data values under a are all distinct"* is recognizable.
- The language "There exists a data value appearing at least twice under *a*" is recognizable.
- The language "All data values under *a* occur at most *n* times" is recognizable.
- The language " There exists a data value appearing under *a* occurring more than *n* times" is recognizable.
- The language $L_{\forall a, = n} =$ *"All data values under a occur exactly n times"* is not recognizable.

Decidability

Theorem

The emptiness problem of class counting automata is decidable.

- By reduction to the covering problem for Petri nets.
- The decision procedure runs in Expspace, and thus we have elementary decidability.
- The problem is complete for Expspace, by an easy reduction the other way as well.

CCA are closed under union and intersection, but not under complementation.

Extensions

The model admits many possible extensions.

- Instead of working with one bag of counters, the automaton can use several bags of counters, much as multiple registers are used in the register automaton.
- We can check for the presence of *any* counter (in each bag) satisfying a given constraint and updating it.
- The language of constraints can be strengthened: any syntax that can specify semilinear sets will do.
- Extensions like two-way movement and alternation lead to undecidability.

A comparison

- No two *a* positions have same data value: PA, DA, CCA, FO^2, but not RA.
- There exist two *a* positions having same data value: all formalisms.
- For every *a* position, there exists a *b* position with the same data value: PA, DA, CCA, FO^2, but not RA.
- A process has to consume one given resource before requesting another: PA, DA, CCA, FO^2, but not RA.
- Every process requesting a resource is eventually granted: PA, DA, CCA, FO^2, but not RA.
- Between two successive accesses to the resource by the same user, some other process has to access it: PA, DA, RA, but not FO^2 or CCA.

Comparison

- Non-emptiness decidable for RA, CCA, DA and FO^2, but not PA.
- Inclusion decidable only for FO^2.
- Membership efficient for RA, CCA, PA and FO^2, but not DA.
- PA and FO^2 are closed under complementation, CCA, PA and DA are not.

Automata vs Logics

Mostly incomparability results; better behaviour for PA than RA.

- No FO / MSO characterization for RA.
- 2-way APA = MSO; 2-way strong DPA = FO.
- Emptiness undecidable for weak 1-way PA.

Many questions

Data words have many potential applications.

- Applications to verification of parametrized systems ?
- This approach orthogonal to reachability based approaches.
- Ability to talk about data is very limited (no arithmetic).
- Find models with better complexities.
- Study the tradeoff between more expressive data access and complexity / decidability.

Clear need for decidable automata models and logics over data words and data trees.

A challenging topic with many potential applications in databases and system verification.