Computational semantics for monadic quantifiers*

Marcin Mostowski
e-mail:marcinmo@mail.uw.edu.pl

Abstract

The paper gives a survey of known results related to computational devices (finite and push–down automata) recognizing monadic generalized quantifiers in finite models. Some of these results are simple reinterpretations of descriptive—feasible correspondence theorems from finite–model theory. Additionally a new result characterizing monadic quantifiers recognized by push down automata is proven.

The aim of the work is presentation of the state of knowledge and main research problems in computational approach to finite interpretations of monadic generalized quantifiers. We will concentrate on purely logical approach – that is we will consider mainly structures without standard linear orderings. However proper results for linearly ordered structures will be mentioned when relevant.

Let us observe that monadic quantifiers – being the first natural class of quantifiers investigated from computational point of view – are not systematically treated in logical literature. In the first attempt [6] of surveying the subject of generalized quantifiers from computational point of view, monadic quantifiers are only shortly mentioned.

1 Generalities

By monadic generalized quantifiers we mean Lindström's quantifiers (see [5]) of types of the form $(1, 1, \ldots, 1)$; if the number of ones in the type signature is n then we say that the corresponding quantifier is a monadic

*The research reported here has been supported by the research grant of Polish National Commitee of Scientific Research (KBN) 433/H01/95/08.

n-ary quantifier. Syntactically it binds one variable in n formulae. More precisely for each n–ary monadic quantifier Q we define the logic $L(Q)$ as an extension of elementary (first order) logic. The set of $L(Q)$–formulae is obtained by adding to the standard construction rules the following:

if x is a variable and $\varphi_1, \ldots, \varphi_n$ are formulae then also $Qx(\varphi_1, \ldots, \varphi_n)$ is a formula.

Semantically n-ary monadic quantifier Q can be represented by closed on isomorphisms class K_Q of structures of the form $(U, R_1, \ldots, R_n)$, where $U \neq \emptyset$ and $R_i \subseteq U$, for $i = 1, 2, \ldots, n$.

The definition of the satisaction relation is extended by the following: for any model M, and a valuation $\mathbf{a}$ in M:
$M \models Qx(\varphi_1, \ldots, \varphi_n)[\mathbf{a}]$ if and only if $(|M|, \varphi_1^{M,x,\mathbf{a}}, \ldots, \varphi_n^{M,x,\mathbf{a}}) \in K_Q$,
where $|M|$ is the universe of M, and $\varphi^{M,x,\mathbf{a}}$ is the set defined by φ in M with respect to the variable x, under the valuation $\mathbf{a}$ – free in φ variables different than x are interpreted according to the valuation $\mathbf{a}$.[1]

The class K_Q can be represented by the set of nonempty words F_Q over the alphabet $A = \{a_0, a_1, \ldots, a_{2^n-1}\}$ such that

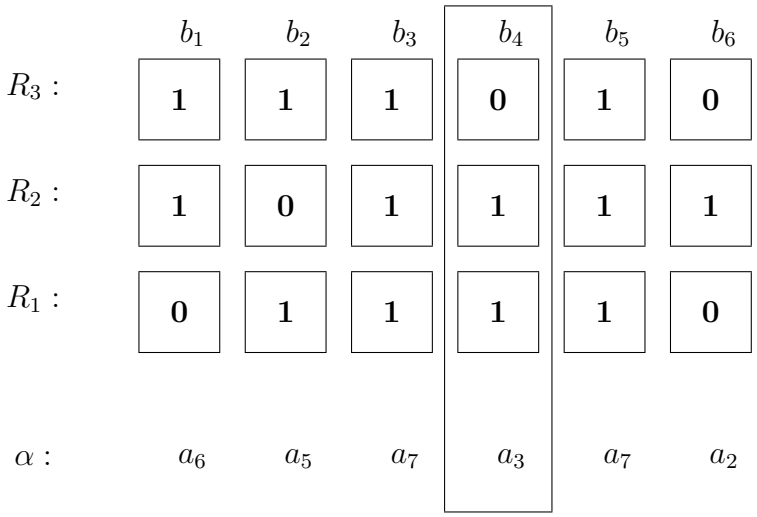
$\alpha \in F_Q$ if and only if there is $(U, R_1, \ldots, R_n) \in K_Q$ and a linear ordering $U = \{b_1, \ldots, b_k\}$ such that $lh(\alpha) = k$, and i-th character of α is a_j exactly when $b_i \in S_1 \cap \ldots \cap S_n$, where

$$S_l = \begin{cases} R_l & \text{if integer part of } j/2^{(l-1)} \text{ is odd} \\ U - R_l & \text{otherwise} \end{cases}$$

So defined intersections $S_1 \cap \ldots \cap S_n$ are called constituents of the proper model – the above defined constituent is also called j-th constituent. Then we can think of characters $a_0, \ldots, a_{2^n-1}$ as names of constituents. In other words our definition says that i–th character is a_j exactly when b_i belongs to j–th constituent. For identifying a constituent we should know which relation is taken positively and which is taken negatively. This can be calculated by writing down j in binary notation, and checking whether l–th digit from the rigth is 1 (positive) or 0 (negative).

For $n = 3$ this idea is ilustrated by the following picture:

[1]For the discussion of the general definition of Lindström quantifiers see e. g. [5] or [3].



Particularly, when $b_4 \notin R_1$ and $b_4 \in R_2 \cap R_3$, then the fourth character of the word α is a_3.

Let $\mathcal{C}$ be a class of recognizing devices, and $\mathcal{Q}$ be a class of monadic quantifiers. We say that $\mathcal{C}$ *accepts* $\mathcal{Q}$ *if and only if for each* $n > 0$ *and each* n*–ary monadic quantifier* Q ($Q \in \mathcal{Q} \Leftrightarrow \exists M \in \mathcal{C}(M$ *accepts* $F_Q)$.

Originally these concepts were introduced by Büchi (see [4]), who investigated models with fixed linear orderings. Of course when no ordering is fixed then classes F_Q are closed on permutations in the sense that if $\alpha \in F_Q$ and α' is obtained from α by arbitrary shuffling characters then $\alpha' \in F_Q$. Interpretations of monadic quantifiers in this spirit were first investigated in [1] and [2].

we have the following straightforward:

Lemma 1 *For each monadic quantifiers* Q_1, Q_2 *of the same type there are quantifiers being their intersection, union, and the complement of* Q_1*, defined by sets of words* $F_{Q_1} \cap F_{Q_2}$*,* $F_{Q_1} \cup F_{Q_2}$*, and* $A^* - F_{Q_1} - \{\varepsilon\}$ *respectively.*

If the class of quantifiers is closed on all the above operations then we will say that it is closed on boolean combinations.

Considering monadic quantifiers in presence of linear ordering can be understood as follows:

Instead of a monadic quantifier Q of type $(1, 1, \ldots, 1)$, we consider a generalized quantifier Q' of type $(2, 1, 1, \ldots, 1)$ (with the same number of ones), and we assume that in our language we have one binary predicate symbol

$<$, which is always interpreted as a linear ordering — what is equivalent to saying that we consider only models for the theory of linear ordering in terms of $<$. Then we interpret formulae of the form

$$Qx(\varphi_1, \ldots, \varphi_m)$$

as

$$Q'xy(x < y, \varphi_1, \ldots, \varphi_m).$$

A similar remark apply to the Ramsey quantifier Q^2, introduced by Macintyre (see [3]).

2 First Characterizations

The first characterization of quantifiers in finite models from computational point of view has been given by van Benthem.[2]

Theorem 1 (straightforward generalization of [2]) *Finite automata with only 1–loops accept the class of all monadic elementary definable quantifiers.*

Because of the following

Theorem 2 (see [9][3]) *Monadic quantifiers definable in monadic second order logic are elementary definable.*

we have also

Corollary 3 *Finite automata with only 1–loops accept the class of all monadic quantifiers definable in monadic second order logic.*

These results can be contrasted with the following:

Theorem 4 (Büchi, Ladner, see [4][4])

[2]Other older results mentioned in this paper are essentially reinterpretations of theorems about descriptive—feasible correspondences, originally formulated in terms of classes of models.

[3]This theorem follows also from the last theorem in [10], which says, among others, that monadic second–order logic in monadic vocabulary is semantically equivalent to elementary logic.

[4]This theorem follows from works of Büchi. However Ladner gave its reinterpretation in the spirit of finite–model theory.

1. *In presence of linear ordering star–free finite automata accept elementary definable monadic quantifiers.*

2. *In presence of linear ordering finite automata accept monadic quantifiers definable in monadic second order logic.*

The above theorem is essentially reinterpretation of Ladner's formulation, which was not given in terms of quantifiers, but definable classes of models. Reinterpretation of other results from finite–model theory in terms of quantifiers has been done in the first general survey devoted to generalized quantifiers in finite models by Makowsky and Pnueli [6].[5] In their paper they give also some new results, particularly, related to mutual interpretations of languages with generalized quantifiers in finite models.

3 Divisibility Quantifiers and Finite Automata

For $n \geq 2$ by divisibility quantifier D_n we mean the quantifier of type (1) such that

$\mathsf{D}_n x\varphi$ if and only if the number of x satisfying φ is divisible by n.

Divisibility logic $L(\mathsf{D}_\omega)$ is elementary logic enriched by all quantifiers D_n, for $n \geq 2$. Divisibility logic is equivalent to the logic with quantifiers *counting modulo* (see e. g. [11]).

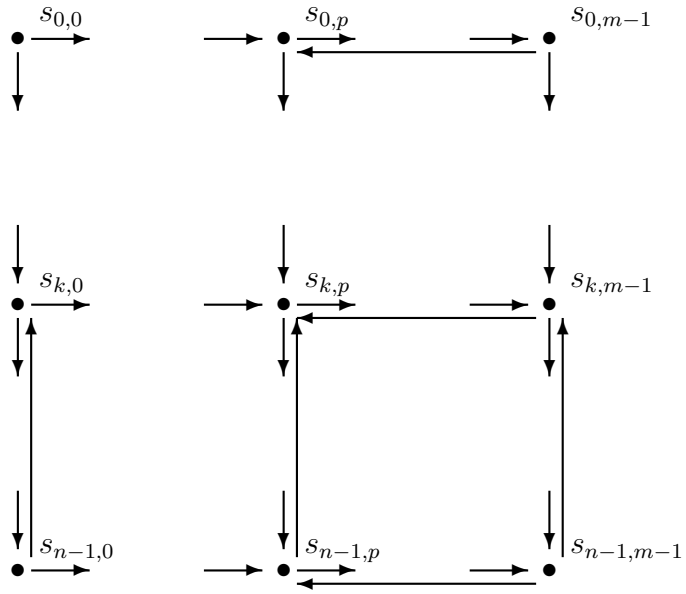
Theorem 5 ([8]) *Finite automata accept the class of all monadic quantifiers definable in $L(\mathsf{D}_\omega)$.*

Proof. We will discuss here only the simplest case of unary monadic quantifier Q such that F_Q is regular. Our alphabet A is $\{0, 1\}$. Following [8] we will describe the automaton recognizing F_Q.

We define the standard equivalence relation $\approx$ as follows $\alpha \approx \beta$ iff for every $\gamma \in \{0,1\}^*(\alpha\gamma \in F_Q$ iff $\beta\gamma \in F_Q)$. Let k be the smallest number such that $0^k \approx 0^i$, for some $i > k$, and n be the smallest such that $0^k \approx 0^n$ and $n > k$. Numbers p and m are defined in the same way taking 1 in the place of 0.

Let us consider the following automaton $\mathcal{A} = (S, \{0,1\}, \delta, s_{00}, F)$

[5] As far as I know the first paper dvoted to generalized quantifiers in finite models is [10].



where all horizontal arrows are labeled by 1, and all vertical arrows are labeled by 0. We will call this automaton: (n, m, k, p)-automaton.

A priori we cannot expect that $\mathcal{A}$ will be identical with the syntactic automaton $\mathcal{A}'$ defined by $\approx$. However we can define proper set F of accepting states in $\mathcal{A}$ by constructing a suitable homomorphism $f : \mathcal{A} \to \mathcal{A}'$, and defining F as $f^{-1}(F')$, where F' is the set of accepting states of $\mathcal{A}'$. So we obtain that $\mathcal{A}$ accepts F_Q.

For every state $s_{ij} \in F$ we define the formula ψ_{ij} such that ψ_{ij} expresses that:

"*there are exactly j x such that $\varphi(x)$ and exactly i x such that $\neg\varphi(x)$*", if $i < k$ and $j < p$,

"*the number of x such that $\varphi(x)$ minus p is equivalent to $j - p$ modulo $m - p$ and there are exactly i x such that $\neg\varphi(x)$*", if $i < k$ and $j \geq p$,

"*there are exactly j x such that $\varphi(x)$ and the number of x such that $\neg\varphi(x)$ minus k is equivalent to $i - k$ modulo $n - k$*"[6], if $i \geq k$ and $j < p$,

"*the number of x such that $\varphi(x)$ minus p is equivalent to $j - p$ modulo $m - p$ and the number of x such that $\neg\varphi(x)$ minus k is equivalent to $i - k$ modulo $n - k$*", if $i \geq k$ and $j \geq p$.

[6]We assume here that for each n, m, $n \equiv m (\mathrm{mod}\ 1)$.

That "*there are exactly j x such that $\varphi(x)$*" is expressible in $L(\mathsf{D}_\omega)$ follows from that it is expressible in elementary logic. That "*the number of x such that $\varphi(x)$ is equivalent to j modulo s*" can be expressed in $L(\mathsf{D}_\omega)$ by the following formula:

$\exists x_1 \ldots \exists x_j(\varphi(x_1) \wedge \ldots \wedge \varphi(x_j) \wedge \bigwedge_{1 \leq a < b \leq j}(x_a \neq x_b) \wedge \mathsf{D}_s y(y \neq x_1 \wedge \ldots \wedge y \neq x_j \wedge \varphi(y)))$.

We define ψ as the disjunction of all ψ_{ij} such that $s_{ij} \in F$. Observe that this construction is general and effective.

This ends the proof that each monadic quantifier Q such that F_Q is regular is definable in divisibility logic.

We recall here that a quantifier Q of type $(n_1, \ldots, n_k)$ is definable in a logic L if and only if there is L-formula without free variables $\varphi(P_1, \ldots, P_k)$ with $P_1, \ldots, P_k$ as only nonlogical concepts (P_i is n_i-ary predicate) such that $Q\mathbf{x}(\varphi_1, \ldots, \varphi_k)$ is semantically equivalent to $\varphi(\varphi_1, \ldots, \varphi_k)$, where substitution of formulae in place of predicates is defined in a natural way respective to proper variables from the sequence $\mathbf{x}$.

Therefore the second inclusion follows from the fact that each $L(\mathsf{D}_\omega)$–formula in monadic vocabulary is equivalent to a boolean combination of formulae described in the first part of the proof (see [8]).

This ends the proof.

By the above theorem monadic quantifiers definable in $L(\mathsf{D}_\omega)$ will be called regular quantifiers.

Let us observe that theorem 1 can be obtained as a simple corollary from the above reasoning. When considered quantifier Q is elementary definable we obtain the above described automaton $\mathcal{A}$ such that $n = k + 1$ and $m = p + 1$. In this case obviously $\mathcal{A}$ has only 1–loops.

The last theorem can be contrasted with the following

Theorem 6 (easily following from [11]) *In presence of linear ordering finite automata having syntactic monoids not containing non solvable groups accept the class of all monadic quantifiers definable in $L(\mathsf{D}_\omega)$.*

Of course there are regular sets having syntactic monoids, which contain nonsolvable groups. Then, in presence of linear ordering, monadic second order logic is stronger than divisibility logic. Without ordering we have the opposite relation, divisibility logic is stronger than monadic second order logic.

4 Push–Down Automata

We can describe monadic quantifiers also in another way. Let Q be n-ary monadic quantifier, then Q is uniquely determined by 2^n-ary relation R_Q on ω defined as

$R_Q = \{(s_0, \ldots, s_{2^n-1}) : \exists M = (U, R_1, \ldots, R_n) \in K_Q \forall j \in \{0, \ldots, 2^n - 1\}$ j-th constituent of M has cardinality $s_j\}$

On the other hand each 2^n-ary relation R on ω not containing the tuple $(0, 0, \ldots, 0)$ uniquely determines n-ary monadic quantifier Q such that $R = R_Q$. Therefore we can consider classes of monadic quantifiers $\mathcal{Q}$ determined by some classes $\mathcal{R}$ of relations on ω, in the sense that $\mathcal{Q} = \{Q_R : R \in \mathcal{R}\}$.

Van Benthem in [1] considered the class of quantifiers determined by relations elementary definable in the structure $(\omega, +)$, which he called additively definable quantifiers.

Theorem 7 ([1]) *Push Down Automata accept additively definable quantifiers of type* (1).

The proof of this theorem is based on the fact that additively definable relations are exactly semilinear relations, that is finite unions of so called linear relations of the form:

$S = \{(x_1, \ldots, x_m) \in \omega^m : \exists z_1, \ldots, z_k (x_1, \ldots, x_m) = (a_1, \ldots, a_m) +$ $z_1(b_{11}, \ldots, b_{1m}) + \ldots + z_k(b_{k1}, \ldots, b_{km})\}$.

Then quantifiers determined by semilinear relations will be called also *semilinear quantifiers*.

Van Benthem observed also that some semilinear (additively definable) quantifiers of arity > 1 cannot be recognized by PDA.

In this section we will consider also quantifiers of type (1) accepted by deterministic PDA. However firstly we will recall some basic concepts related to Push–Down Automata (PDA).

A push–down automaton is of the form $\mathcal{A} = (A, \Sigma, S, s_0, F, \delta)$, where A is an input alphabet, Σ – stack alphabet, S – a set of internal states, $s_0 \in S$ is an initial state, $F \subseteq S$ is a set of accepting states, and finally

$$\delta : (A \cup \{\varepsilon\}) \times S \times \Sigma \longrightarrow P(S \times \Sigma^*)$$

is a transition function.

The function δ describes the behaviour of $\mathcal{A}$ in the following sense:

When reading a word the automaton $\mathcal{A}$ pops an element σ from the top of the stack, reads a character $a \in A$ (or nothing in a case of so called ε–moves), and is in a state $s \in S$, then it chooses $(s', \sigma') \in \delta(a, s, \sigma)$, and changes its internal state to s', it shifts the head into the next character, and pushes the word σ' into the top of the stack.

If $\delta(\varepsilon, s, \sigma) = \emptyset$, for each $s \in S, \sigma \in \Sigma$ (there are no ε–moves), and $\delta(a, s, \sigma)$ always contains not more then one element, then we say that $\mathcal{A}$ is deterministic push down automaton, or shortly deterministic PDA, otherwise it is called nondeterministic PDA.

When $\mathcal{A}$ is deterministic PDA then the transition function δ can be described as a partial function

$$\delta : A \times S \times \Sigma \longrightarrow S \times \Sigma^*$$

An input word $\alpha \in A^*$ is accepted by $\mathcal{A}$ if and only if starting its work with its head on the first character of α, being in the iternal state s_0, and having the stack empty, the automaton $\mathcal{A}$ ends reading α in a state belonging to F. If additionally the stack is empty at the end, then we say that $\mathcal{A}$ accepts α by empty stack.

A set $X \subseteq A^*$ is recognized by $\mathcal{A}$ if for each $\alpha \in A^*$:

$$\alpha \in X \text{ if and only if } \mathcal{A} \text{ accepts } \alpha.$$

Deterministic PDA–s recognize by arbitrary stack larger class of sets then the class recognized by empty stack, e. g. let $L = \{0^n 1^k : k \leq n, k, n \in \omega\}$. L is acceptable deterministically, but not by empty stack. The reason is that each PDA after reading many symbols 0 and making the stack empty cannot memorize how many symbols 1 can be still accepted.

Let us observe that all the above concepts are defined in such the way that forgeting about stacks and stack alphabets we obtain adequate definitions of corresponding concepts related to finite automata.

Let $a, b, c, d \in \omega$, by (a, b, c, d)–linear quantifier $\mathsf{Lin}_{(a,b,c,d)}$ we mean the quantifier of type (1) such that $\mathsf{Lin}_{(a,b,c,d)} = Q_R$, where

$$(*) \qquad R = \{(x, y) \in \omega^2 : \exists z \in \omega \;\; (x, y) = (c, d) + z(a, b)\}$$

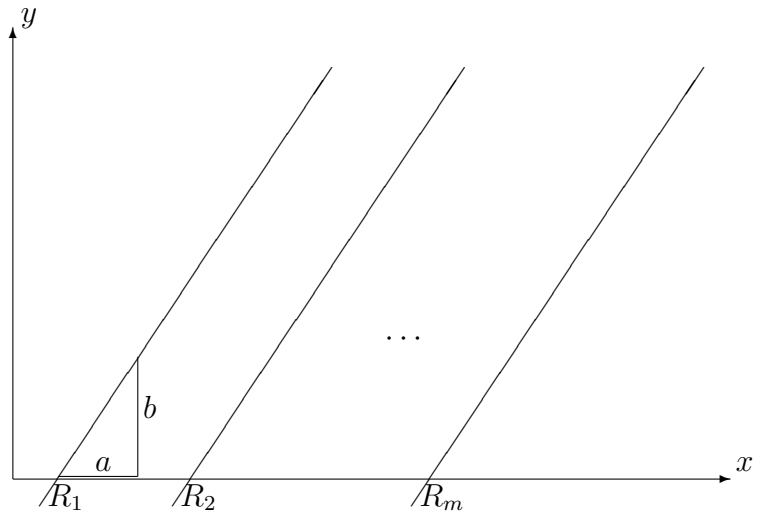
We say that a quantifier Q of type (1) is *almost linear* iff for some $a, b, c_1, \ldots, c_k, d_1, \ldots, d_k \in \omega$ and some finite relation $R_0 \subseteq \omega^2$:

$$Q = Q_{R_0} \cup \mathsf{Lin}_{(a,b,c_1,d_1)} \cup \ldots \cup \mathsf{Lin}_{(a,b,c_k,d_k)}.$$

In other words almost linear quantifiers are those semilinear quantifiers, which have at most one nontrivial vector (a, b) in its defining relation.

The ratio a/b will be called the ratio of the corresponding almost linear relation.

For geometrical illustration of what does it mean *almost linear*, let us consider a relation $S \subseteq \omega^2$ being a finite union of relations of the form $(*)$ with the same ratio a/b. Allowing z to vary over the set of reals we can think of S as a set of points on the euclidean plane. Then almost linear relation S can be defined (modulo a finite set) as a finite union of straight lines $R_1, \ldots, R_m$ parallel each to other.



The above picture suggests that complements of almost linear relations are never almost linear, their finite unions are almost linear only if they have the same ratio, and their intersections are either almost linear or finite.

Of course deterministic PDA–s recognize by empty stack all regular quantifiers, but what more? We will show that they recognize also almost linear quantifiers. However the intersection of a regular set and a set recognized by deterministic PDA (by empty stack) is still recognized by deterministic PDA (by empty stack). We claim that nothing more (between permutation closed sets) is recognized by deterministic PDA–s by empty stack.

Theorem 8 *The class of all quantifiers recognized by deterministic PDA–s by empty stack is the union of the following:*

1. *regular quantifiers,*

2. *intersections of regular and almost linear quantifiers.*

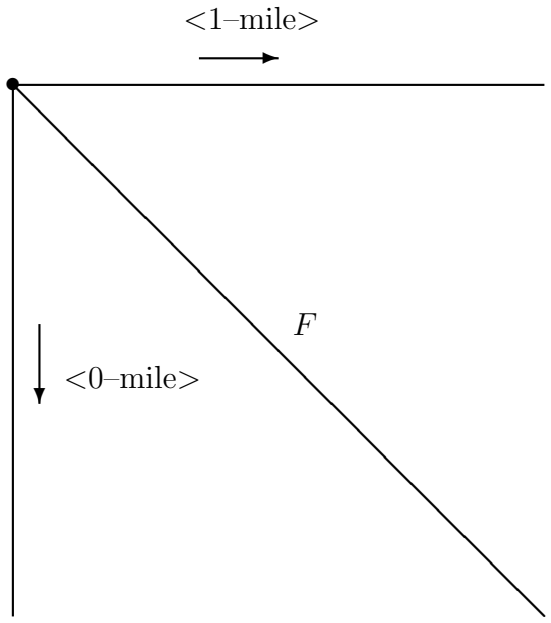
Proof Firstly we will show that almost linear quantifiers are recognized by deterministic PDA–s. We start with the simplest linear case without

exceptions. Let R be defined as above for some fixed $a, b, c, d \in \omega$. Let Q be Q_R. Now we will describe the deterministic PDA recognizing F_Q.

We define the stack alphabet S as {<0–mile>, <1–mile>}.

<0–mile> contains a steps, and <1–mile> contains b steps.

The behaviour of our PDA can be described by the following picture:



Reading a word α we go to right – when 1 is read b times, and down – when 0 is read a times.

We count miles poping and pushing them as follows:

- when we have passed one mile and the stack is empty then we push it,
- when we have passed one mile and the stack contains a mile of the same kind then we push it,
- when we have passed one mile and the stack contains a mile of another kind then we pop the mile from the stack.

0–miles can be counted after reading 0 c–times, 1–miles can be counted after reading 1 d–times.

The word α is accepted if after reading it we are staying at the line F — it means that the stack is empty.

The general case allowing exceptions can be obtained by enriching our construction by some states and instructions not modifying the stack.

Remark Let us observe that a similar picture can be used to justify Van Benthem's theorem 7.

Let $Q' = Q_{R'}$ be defined by a semilinear R', that is being a finite union of linear relations S such that for some $a_1, b_1, \ldots, a_k, b_k, c, d \in \omega$

$$S = \{(x, y) \in \omega^2 : \exists z_1, \ldots, z_k (x, y) = (c, d) + z_1(a_1, b_1) + \ldots + z_k(a_k, b_k)\}.$$

In this case for recognizing Q_S we can use the same idea constructing nondeterministic PDA with k kinds of 0–miles, and k kinds of 1–miles. Our PDA will choose nondeterminstically what kind of miles have to be used for measuring distances.

Therefore $F_{Q'}$ as being a finite union of context free sets is itself context free.

end of Remark

Now let us assume that Q is accepted by some deterministic push down automaton $\mathcal{A}$. By Parikh's theorem there is a semilinear R', as described in the above remark, such that $Q = Q_{R'}$.

Firstly we assume that modulo finite number of exceptions R' is linear, that is $R' = S \cup R_0$, where R_0 is finite, and for some $a_1, b_1, \ldots, a_k, b_k, c, d \in \omega$

$$S = \{(x, y) \in \omega^2 : \exists z_1, \ldots, z_k (x, y) = (c, d) + z_1(a_1, b_1) + \ldots + z_k(a_k, b_k)\}.$$

Then we obtain two possible cases:

1. for some $i = 1, \ldots, k$ either $a_i = 0$ and $b_i \neq 0$ or $b_i = 0$ and $a_i \neq 0$, in this case Q is definable by divisibility quantifiers;
2. all $a_1, \ldots, a_k, b_1, \ldots, b_k$ are different than 0, in this case we show that Q is almost linear.

Let us consider in details only the case 2. The case 1 is simpler, and it can be analyzed similarly. We consider the simplest case when the set S is defined by three vectors:

$$S = \{(x, y) \in \omega^2 : \exists z_1, z_2 (x, y) = (c, d) + z_1(a_1, b_1) + z_k(a_2, b_2)\},$$

and no one parameter is equal to 0. The general argument differs only by straightforward technicalities.

Let us consider words:

$$\alpha_x = 0^c 1^d 0^{(xa_1)a_2} 1^{(xa_1)b_2}$$

$$\beta_x = 0^c 1^d 0^{(xa_2)a_1} 1^{(xa_2)b_1}$$

Now we will show that $a_1 b_2 = a_2 b_1$.

Let us assume that $a_1 b_2 < a_2 b_1$. Then for arbitrary large n we can chose x such that

$$k = x(a_2 b_1 - a_1 b_2) > n.$$

Then $\beta_x = \alpha_x 1^k$.

The automaton $\mathcal{A}$ after reading α_x is in accepting state with empty stack. It means that it cannot memorize x, and from this moment, not reading any zeros, it behaves like finite automaton (without stack). Chosing sufficiently large x we obtain the contradiction.

Therefore in the case 2 we prove that there are natural numbers $i, j > 0$ such that for some $x_1, \ldots, x_k$:

$$\begin{aligned} (a_1, b_1) &= x_1(i, j) \\ &\vdots \\ (a_k, b_k) &= x_k(i, j) \end{aligned}$$

Moreover we can choose $x_1, \ldots, x_k$ being relatively prime and i, j being greatest possible. Then we take $a = yi$, $b = yj$, and $c' = c + ix$, $d' = d + jx$, for some suficiently large x, e. g. $x = x_1 \ldots x_k z$, where y is the least multiplier of (i, j) greater than all exceptions and z is used to make c' and d' greater than all exceptions. For differentiating between particular (a_i, b_i), for $i = 1, \ldots, k$, we need only counting modulo y, what can be done without using the stack.

Therefore Q is $\mathsf{Lin}_{(a,b,c',d')} \cap Q'$ modulo some finite number of exceptions, for some regular Q'.

The general case can be obtained by a similar argument.

This ends the proof.

As the result we obtain the following:

Corollary 9 *The complement of a quantifier Q recognized by a deterministic PDA by empty stack is also recognized by deterministic PDA by empty stack if and only if Q is regular.*

Of course, by a trivial argument, the complement of a quantifier recognized by a deterministic PDA is also recognized by a deterministic PDA by arbitrary stack.

Corollary 10 *The class of quantifiers recognized by deterministic PDA–s by empty stack is closed on intersections.*

Let us observe that the assumption that accepting is by empty stack is essential here, because the class of quantifiers described above is not closed on complements. Additionally the union of an almost linear and a regular quantifier can be still recognized by a deterministic PDA but not by empty stack.

Similarly as in the case of nondeterministic PDA–s no natural descriptive class of all monadic quantifiers recognized by deterministic PDA–s is not known.

Finally let us mention another open problem. In the paper [7] it was observed that in the theory of $(\omega, +)$ divisibility quantifiers are eliminable. We do not know any other monadic quantifiers being eliminable in arithmetic of addition, which are not definable in divisibility logic. Are there any such quantifiers?

5 Examples

¿From our work and known proprieties of finite and push–down automata we obtain the following facts:

1. **Definable in $L(\mathsf{D}_\omega)$, regular quantifiers:**
   - **even**, **odd**;
   - counting modulo $\exists^{pq}$, for $q < p$, where $\exists^{p,q} x\varphi$ iff $\#\{x : \varphi\} \equiv q(mod\ p)$;[7]
   - $Qx(\varphi, \psi)$ iff $\#\{x : \varphi\} \equiv \#\{x : \psi\}(mod\ p)$.
2. **Recognized by deterministic PDA–s by empty stack, but not regular quantifiers:**
   - **Half**, **every other**, where **Half**$x\varphi$ iff $\#\{x : \varphi\} = \#\{x : \neg\varphi\}$;
   - **every third**, **every fourth**, ..., **every** n**–th**, where **every** n**–th** $x\varphi$ iff $n \cdot \#\{x : \varphi\} = \#\{x : x = x\}$;

[7] For a set A by $\#A$ we denote the cardinal number of A.

- Härtig quantifier, $Q_H x(\varphi, \psi)$ iff $\#\{x : \varphi\} = \#\{x : \psi\}$.

3. **Recognized by deterministic PDA–s by arbitrary stack, but not by deterministic PDA–s by empty stack:**

   - **majority**, where **majority** $x\varphi$ iff $\#\{x : \varphi\} > \#\{x : \neg\varphi\}$.

4. **Recognized by non deterministic PDA–s, but not by deterministic PDA–s:**

   - **two or three times more**, where the quantifier Q is definned by disjunction of $\mathsf{Lin}_{(1,2,0,0)}$ and $\mathsf{Lin}_{(1,3,0,0)}$,
     $Qx\varphi$ iff $\#\{x : \varphi\} = 2 \cdot \#\{x : \neg\varphi\}$ or $\#\{x : \varphi\} = 3 \cdot \#\{x : \neg\varphi\}$.

5. **Semilinear, but not recognized by PDA–s:**

   - $Qx(\varphi, \psi)$ iff $\#\{x : \varphi \wedge \neg\psi\} = \#\{x : \varphi \wedge \psi\} = \#\{x : \neg\varphi \wedge \psi\}$.

References

[1] J. van Benthem, **Essays in Logical Semantics**, D. Reidel Publishing Company 1986.

[2] J. van Benthem, *Towards a Computational Semantics*, in P. Gardenförs (ed.), **Generalized Quantifiers**, D. Reidel Publishing Company 1987, pp. 31–71.

[3] M. Krynicki and M. Mostowski *Quantifiers, Some Problems and Ideas*, in M. Krynicki, M. Mostowski, and L.W. Szczerba (eds.) **Quantifiers: Logics, Models and Computation** Volume I, Kluwer Academic Publishers, 1995, pp. 1–19.

[4] R. E. Ladner, *Application of Model Theoretic Games to Discrete Linear Orders and Finite Automata*, **Information and Control** 33 (1977), pp. 281–303.

[5] P. Lindström, *First order predicate logic with generalized quantifiers*, **Theoria** 32 (1966), pp. 186–195.

[6] J. A. Makowsky and Y. B. Pnueli *Computable quantifiers and logics over finite structures*, in M. Krynicki, M. Mostowski, and L.W. Szczerba (eds.) **Quantifiers: Logics, Models and Computation** Volume I, Kluwer Academic Publishers, 1995, pp. 313–357.

[7] M. Mostowski, *Divisibility quantifiers*, in **Bulletin of the Section of Logic** 20, no. 2 (1991), pp. 67–70.

[8] M. Mostowski, *The Logic of Divisibility*, to appear in **The Journal of Symbolic Logic**.

[9] M. Mostowski, *Kwantyfikatory rozgałęzione a problem formy logicznej*, (in Polish, Branched Quantifiers and the Problem of Logical Forms) in **Nauka i język**, Wydział Filozofii i Socjologii Uniwersytetu Warszawskiego, Warszawa 1994, pp. 201 – 241.

[10] J. Väänänen, *Remarks on generalized quantifiers and second–order logic*, in **Set Theory and Hierarchy Theory**, J. Waszkiewicz, A. Wojciechowska, and A. Zarach (eds.), Prace Naukowe Instytutu Matematyki Politechniki Wrocławskiej, Wrocław 1977, pp. 117–123.

[11] H. Straubing, D. Thérien, W. Thomas, *Regular Languages Defined with Generalized Quantifiers*, **Proc. 15th ICALP 88**, T. Lepistö and A. Salomaa (eds.) Lec. Notes in Comput. Sci. 317, Springer-Verlag, Berlin 1988, pp. 561–575.