On Some Refinement of Higman Embedding in Regular Approximations of Loops

Antonina Nepeivoda

Program System Institute of RAS
Pereslavl–Zalessky

Abstract. The paper studies an intransitive binary relation R given on words in traces generated by prefix-rewriting systems. R was introduced in 1988 by V.F. Turchin for regular approximation of semantics trees of programs. The relation satisfies the main property of quasi-orders: any infinite trace Φ_1, Φ_2, ... contains two words Φ_i, Φ_j s.t. $(\Phi_i, \Phi_j) \in R$ and $i < j$. We give a new simple constructive proof of the main property and estimate maximal length of a track with no pairs satisfying the relation. It is shown that on tracks generated by prefix-rewriting systems Higman's embedding can be expressed and studied through Turchin's embedding.

Keywords: Higman's lemma, prefix rewriting, almost well relation, computational complexity, termination, supercompilation

1 Introduction

Definition 1 *A transitive and reflexive relation $R \subset \Upsilon \times \Upsilon$ is a well quasi-order (wqo) iff every sequence $\{A_n\}$, $\forall n(A_n \in \Upsilon)$, s.t. $\forall i, j(i < j \Rightarrow (A_i, A_j) \notin R)$ is finite. The sequence $\{A_n\}$ that posesses the property $\forall i, j(i < j \Rightarrow (A_i, A_j) \notin R)$ is called a bad sequence.*

One of the most well-studied wqos is Higman–Kruskal embedding on terms. Now we only define the Higman relation; Kruskal relation generalizes the Higman embedding to trees instead of sequences.

Definition 2 *Given two words in an alphabet Υ, $A = a_1 a_2 \dots a_m$, $B = b_1 b_2 \dots b_n$, A is embedded in B in the sense of Higman relation ($A \trianglelefteq B$) iff A is a subsequence of B. This relation is also called a scattered subword relation.*

The Higman–Kruskal relation is a well quasi-order. This was established by Higman [4] (for sequences) and Kruskal [6] (for trees). This embedding is popular in program transformations for finding loop approximations in computations [9], [3]; it is combined with either additional annotations (e.g. in [5]) or other wqos (e.g. in [1]) to construct more precise approximation techniques. But since unexistence of infinite bad sequences is only essential for these needs, transitivity can be omitted (as in Turchin's relation which is used in [8]).

Definition 3 *A reflexive relation $R \subset \Upsilon \times \Upsilon$ is an almost well relation iff for any $\{A_n\}$, $\forall n(A_n \in \Upsilon)$, s.t. $\forall i,j(i<j \Rightarrow (A_i,A_j) \notin R)$ is finite. We also call the sequence $\{A_n\}$ with the property $\forall i,j(i<j \Rightarrow (A_i,A_j) \notin R)$ a bad sequence.*

This wqo property is independent of the sequence $\{A_n\}$: it may have an arbitrary nature. However, program transformations always operate with terms having some structure restrictions. We consider a special class of the sequences — sequences generated by a prefix rewriting system.

Definition 4 *Consider a tuple $\langle \Upsilon, \mathbf{R}, \Gamma_0 \rangle$, where Υ is an alphabet, $\Gamma_0 \in \Upsilon^+$ is an initial word and $\mathbf{R} \subset \Upsilon^+ \to \Upsilon^*$ is a set of rewrite rules. If the rules $R : R_l \to R_r$ can be only applied to words of the form $R_l\Phi$ (which begin with the prefix R_l) and generate the word $R_r\Phi$ then the tuple $\langle \Upsilon, \mathbf{R}, \Gamma_0 \rangle$ is a prefix rewriting grammar.*

If length of left-hand side R_l of every rule $R : R_l \to R_r$ is 1 (only the first letter of a word can be rewritten by the rule) then the prefix-rewriting grammar is called an alphabetic prefix rewriting grammar.

We call a track of a prefix rewriting grammar $\mathbf{G} = \langle \Upsilon, \mathbf{R}, \Gamma_0 \rangle$ a sequence $\{\Phi_i\}_{i=1}^n$, s.t. $\Phi_1 = \Gamma_0$ and $\forall i(i<n \Rightarrow \exists R(R : R_l \to R_r \,\&\, R \in \mathbf{R} \,\&\, \Phi_i = R_l\Theta \,\&\, \Phi_{i+1} = R_r\Theta)$.

Example 1 *Consider the following prefix rewriting grammar $\mathbf{G} = \langle \{a,b\}, \{R_1 : a \to bb, R_2 : aa \to b, R_3 : b \to aa\}, aabab \rangle$. Then the rule R_3 cannot be applied to aabaab, for it does not begin by b; either R_1 or R_2 can be applied — the only correct results of the applications are bbbaab and bbaab respectively.*

The prefix rewriting grammars are convenient to express stack operations and can be represented as stack automata [2]. V. F. Turchin used the similar notion of function stack transformations and introduced his relation for building a regular approximation of semantics trees of programs in the terms of changes in computational tracks [11]. To formalize these notions we use an approach presented in [8].

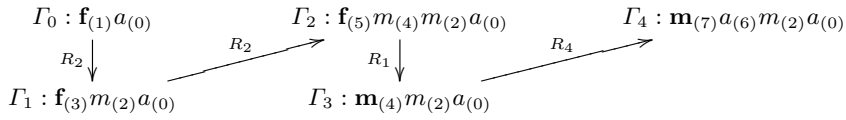
Definition 5 *Consider a track $\{\Phi_i\}_{i=1}^n$ of a prefix rewriting grammar $\mathbf{G} = \langle \Upsilon, \mathbf{R}, \Gamma_0 \rangle$. Let us mark all the letters of the words Φ_i in subsripts by time indexes (natural numbers enclosed in brackets) as follows. The i-th letter of Φ_1 is marked by $|\Gamma_0| - i$, where $|\Gamma_0|$ is the length of Γ_0; if the largest number that is used as a time index in the track segment $\{\Phi_i\}_{i=1}^k$ $(k<n)$ is M and Φ_{k+1} is generated from Φ_k by the rule $R : R_l \to R_r$ then the i-th letter of Φ_{k+1} $(i \le |R_r|$ where $|R_r|$ is the length of $R_r)$ is marked by the time index $M + |R_r| - i + 1$. Time indexes of all other letters in Φ_{k+1} remain the same as in Φ_k, since these letters are unchanged by R.*

We call this notation *time-indexing* and a track generated by $\mathbf{G}$ with the time-indexing notation is called *a computation.*

Example 2 *Consider the following prefix rewriting grammar $\mathbf{G_F}$ (Λ is the empty word).*

$$\begin{array}{lll} R_1 : f \to \Lambda & R_3 : m \to \Lambda & R_5 : a \to \Lambda \\ R_2 : f \to fm & R_4 : m \to ma & R_6 : a \to a \end{array}$$

Let the initial word Γ_0 be fa. The first segment of a computation can look as follows.



We use Greek capitals (Γ, Δ, Θ, Ψ, Φ) to denote words with time indexes (in this paper such words are also called *moments*). Additionally we assume that a word that is denoted by a greater index appears in a track after a word denoted with a lesser one. E.g. the moment Ψ_1 precedes (not necessarily immediately) Ψ_2.

The length of Δ is denoted as $|\Delta|$. $\Delta[k]$ denotes the k-th letter of Δ; $\Phi \approx \Psi \Leftrightarrow |\Phi| = |\Psi| \;\&\; \forall i(i \geq 1 \;\&\; i \leq |\Phi| \Rightarrow (\Phi[i] = a_{(n)} \;\&\; \Psi[i] = b_{(m)} \Rightarrow a = b))$.

Now we are ready to define the *Turchin relation* $\Gamma \preceq \Delta$.

Definition 6 $\Gamma \preceq \Delta \Leftrightarrow \Gamma = \Phi\Theta_0 \;\&\; \Delta = \Phi'\Psi\Theta_0 \;\&\; \Phi' \approx \Phi$.

In Example 2 there are four Turchin pairs: $\Gamma_0 \preceq \Gamma_1$, $\Gamma_0 \preceq \Gamma_2$, $\Gamma_1 \preceq \Gamma_2$, and at last $\Gamma_3 \preceq \Gamma_4$.

Our contributions are the following:

1. We show almost-wellness of the Turchin relation on time-indexed tracks of alphabetic prefix rewriting grammars in a finite alphabet. Thus we prove Turchin's theorem [11] in abstract terms [1]. From this proof we extract an upper bound of a bad sequence length for a grammar.
2. We prove that the bound of a bad sequence length is exact and construct a class of grammars in which this upper bound is achieved.
3. We construct a well-quasiorder that is a subset of Turchin's relation and generalize Turchin's theorem for a wider class of prefix rewriting systems.
4. From the upper bound for Turchin's embedding we construct this bound for Higman's embedding on prefix-rewriting grammar generated word sequences.

2 Turchin's Theorem

Let Υ be a finite alphabet and $\mathbf{G} = \langle \Upsilon, \mathbf{R}, \Gamma_0 \rangle$, $\mathbf{R} \subset \Upsilon \to \Upsilon^*$ an alphabetic prefix rewriting grammar. $\Delta[|\Delta|]$ — denoted as $\Delta[last]$ for better readability — is the last letter of Δ; Δ^- — Δ without the first letter.

Note that the grammar structure guarantees that if Φ precedes Ψ in a computation and $\forall j(\Phi[i] \neq \Psi[j])$ then a word of length no more than $|\Phi| - i + 1$

[1] The proof is independent from programming language features and order of computations.

appears somewhere between Φ and Ψ. Indeed, to get rid of $\Phi[i]$ it is necessary to get the word $\Phi[i]\Phi[i+1]\dots\Phi[last]$.

A rule $R : a \to R_r$ with the non-empty right-hand side is called *non-erasing* (NE-rule); the length of the right-hand side is called the length of the rule and is denoted $|R|$. The result of application of R to Δ is written as $R(\Delta[1])\Delta^-$.

Proposition 1 *If R is an NE-rule, $R(\Theta_0[1])\Theta_0^-$ precedes $R(\Theta_1[1])\Theta_1^-$ and $\exists i(\Theta_1^-[i] = \Theta_0^-[1])$, then $R(\Theta_0[1])\Theta_0^- \preceq R(\Theta_1[1])\Theta_1^-$.*

Proof. After applying R to Θ_0, all of the successor words have the form $\Psi[1]\Psi^-\Theta_0^-$ (because $\Theta_0^-[1]$ is preserved). Application of R to any of these words generates $R(\Psi[1])\Psi^-\Theta_0^-$, and $R(\Theta_0[1])\Theta_0^- \preceq R(\Psi[1])\Psi^-\Theta_0^-$.

Lemma 1 *The maximum length of a word in a bad sequence is no more than*

$$|\Gamma_0| + \sum(|R_i| - 1)$$

where Γ_0 is the initial word, and $\sum(|R_i| - 1)$ runs through all NE-rules R_i.

Proof. To reuse (without quitting a bad sequence) any NE-rule R after its application to Θ_0 it is necessary to shorten a word down to the length $|\Theta_0| - 1$ or less. So the word total length decreases at least by 1, and its maximum cannot exceed the length of a word that is generated by applying all NE-rules exactly once (with no shortenings).

So Turchin's theorem becomes proven in a very simple way.

Proposition 2 (Turchin's theorem) *For every alphabetic prefix rewriting grammar $\mathbf{G}$ in a finite alphabet, computation generated by it does not contain infinite bad sequences (with respect to $\preceq$).*

Proof. Due to finiteness of Υ and existence of the upper bound of a word length in an bad sequence, there exist only a finite number of equivalence classes (with respect to $\approx$) of words that can appear in a bad sequence. When the length of a bad sequence exceeds the number of these classes, some two words in it fall into one class (and form a Turchin pair).

This proof is not only very simple but also is helpful in getting a first very rough upper bound for the maximum length of an bad sequence. It does not exceed the quantity of different words in Υ with the length bounded by $|\Gamma_0| + \sum(|R_i| - 1)$. Let us denote the number of letters in Υ as $card(\Upsilon)$. If $card(\Upsilon) > 1$ then the upper bound is

$$C_{Max} = card(\Upsilon)^{|\Gamma_0| + \sum(|R_i| - 1)}$$

This upper bound is sure rough, and, what is more, says nothing about when long bad sequences can appear and how to generalize the result in terms of arbitrary (i.e. not only alphabetic) prefix rewriting grammars. Little efforts allow to come closer to answers on these questions. And first of all, we must get rid of grammars' occasional features that can imply an unwanted bad sequence termination.

3 Annotated Prefix Rewriting Grammars

In this section we consider only grammars from a class $\mathfrak{G}$ that has two properties. First, if two right-hand sides of rules contain the same letter, then they coincide $(\exists i, j(R_r[i] \approx R'_r[j]) \Rightarrow i = j \ \& \ |R| = |R'| \ \& \ \forall i(i < |R| \Rightarrow R_r[i] \approx R'_r[i]))$. Second, every letter from a left-hand side can be rewritten into any of right-hand sides.

The class $\mathfrak{G}$ is a class of grammars that have ability to generate the longest bad sequences. Absence of coincidences in the different right-hand sides of rules forbids occasional terminations of bad sequences; extension by new rules allows to choose the least-terminating rule.

Definition 7 *A prefix rewriting grammar* $\mathbf{G}$*, s.t.* $\mathbf{R_G} : \Upsilon \to \Upsilon^*$ *is annotated or* $\mathfrak{G}$*-grammar iff*

1. *every two rules either have the same right-hand side or have no common letters in the right-hand sides;*
2. *for every rule* $R^a : a \to R_r$ *and* $b \in \Upsilon$ *the rule* $R^b : b \to R_r$ *is in* $\mathbf{R_G}$*.*

Lemma 2 *Every alphabetic prefix rewriting grammar* $\mathbf{G}$ *can be converted to a* $\mathfrak{G}$*-grammar* $\mathbf{G}'$ *so that every bad sequence generated by* $\mathbf{G}$ *after the conversion is also a bad sequence generated by* $\mathbf{G}'$*.*

Proof. Let us perform these two steps of transformation.

1. Replace every $R_r[i]$ in every right-hand side R_r by the pair $(a, 2^r * 3^{i-1})$, where r is the number of the rule in $\mathbf{R_G}$. All the letters in the initial word should be marked with $r = 0$.
2. Combine all the transformed right-hand sides with all left-hand sides of the form $(a, 2^x * 3^y)$, where x is not greater than the number of rules in $\mathbf{R_G}$, y is not greater than the length of the longest right-hand side. To do this step, add to every pair $a_1 \to \Phi$ and $a_2 \to \Psi$ the pair $a_1 \to \Psi$ and $a_2 \to \Phi$ while this action results with new rules.

Let the initial grammar $\mathbf{G}$ generate a bad sequence by means of a sequence of rules $\{R_i\}_{i=1}^n$. Let transform every R_i into the set $\{R'_{i,j}\}$ and Γ_0 to Γ'_0 using the algorithm above and then apply the sequence $\{R'_{i,j}\}$ to Γ'_0. Every two words in the generated track are incomparable because their projections on the first elements of the pairs $(a, 2^r * 3^{i-1})$ are incomparable.

Example 3 *Let us convert the grammar* $\mathbf{G_F}$ *into a* $\mathfrak{G}$*-grammar.*

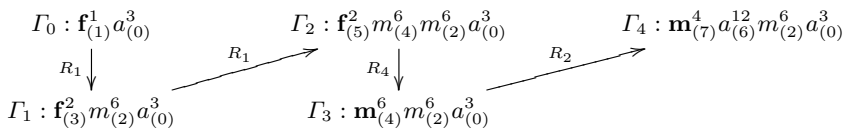
$\mathbf{G'_F}$*:*

$R_0 : x \to f^1 a^3$ $\quad$ $R_2 : x \to m^4 a^{12}$ $\quad$ $R_4 : x \to \Lambda$

$R_1 : x \to f^2 m^6$ $\quad$ $R_3 : x \to a^8$

x denotes an arbitrary letter from Υ.

The first segment of the computation done in Example 2 after the conversion looks like



The length of the bad sequence beginning from Γ_0 now is 2, because the initial f^2 now differs from the f^1 generated by R_1 (so $\Gamma_0 \not\preceq \Gamma_1$).

The described operation of transforming a grammar into the $\mathfrak{G}$-type is idempotent: implemented repeatedly, it results with a $\mathfrak{G}$-grammar that is equivalent up to renaming with the first generated $\mathfrak{G}$-grammar. The number of the rules in the annotated grammar $\mathbf{G}'$ is greater than the number of the rules in $\mathbf{G}$ at most by $\sum |R_i|$ times.

In respect to $\mathfrak{G}$-grammars' properties, expressions denoting rules (R, R' etc.) are also used in these grammars to denote only the right-hand parts of rules (or equivalence classes of rules up to the left-hand sides).

4 Estimating the Maximal Length of an bad sequence

Now we formulate the main theorem of the paper.

Theorem 1 *If the length of the right-hand side of every $\mathfrak{G}$-grammar rule is no more than k ($k > 1$), and there are N rules in the grammar, and the initial word is Γ_0, then the maximal length of a bad sequence is*

$$C'_{Max} = |\Gamma_0| * \frac{k^N - 1}{k - 1}.$$

The remainder of this section is a proof of the Theorem 1. To do this, we describe features of Turchin pairs in $\mathfrak{G}$-grammar generated computations. For the sake of this description we introduce an auxiliary notion of *rule cancellation.*

Let R be a NE-rule applied to $\Delta[1]$. $R(\Delta[1])$ *is canceled* in Δ_2 iff $\Delta_2[1] = \Delta_1^-[1] \vee \forall i, j(\Delta_2[i] \neq \Delta_1^-[1])$. Saying informally, R become canceled in such Δ_2 that neither the letters generated by the R application nor descendants of these letters occur in Δ_2. E.g. in Example 2 neither the first nor the second application of R_2 are not canceled in Γ_4: though Γ_4 contains no children of the second use of R_2, it contains a result of their transformation ($m_{(7)}a_{(6)}$ is a descendant of $m_{(4)}$).

We know that applying a NE-rule without its cancellation results in a Turchin pair (Section 2). It turns out that all $\mathfrak{G}$-grammar-generated bad sequences end by Turchin pairs of this sort (or Λ).

Theorem 2 *In $\mathfrak{G}$-grammars any bad sequence ends either by Λ or by a pair of the form $R(a)\Theta_0$, $R(b)\Psi\Theta_0$, where R is a NE-rule.*

Proof. Let us consider a pair $\Phi_1\Theta_0, \Phi_2\Psi\Theta_0$ such that $\Phi_1\Theta_0 \preceq \Phi_2\Psi\Theta_0$, $(\Phi_1 \approx \Phi_2)$, and the track segment before $\Phi_2\Psi\Theta_0$ is a bad sequence. According to the properties of the class $\mathfrak{G}$, $\Phi_1[1]$ and $\Phi_2[1]$ must be generated by the different applications of the same NE-rule $R : x \to R_r$, and if $\Phi_1[1] \approx R_r[i]$, then necessarily $\Phi_2[1] \approx R_r[i]$. Let us denote the prefix $R_r[1]_{(x+i-2)}R_r[2](x+i-3)...R_r[i-1]_{(x)}$ by $R^{(i-1)}$ and turn back to the two applications of R. The result of the former must be of the form $R^{(i-1)}_{(k_1)}\Phi_1\Theta_0$, the result of the latter — of the form $R^{(i-1)}_{(k_2)}\Phi_2\Psi\Theta_0$ (k_1 and k_2 are the time indices). They form a Turchin pair so coincide with $\Phi_1\Theta_0$ and $\Phi_2\Psi\Theta_0$ (which are, under our assumption, the first Turchin pair).

So we have proved $\Phi_1 = R(\Phi'_1[1])\Phi'^{-}_1$, $\Phi_2 = R(\Phi'_2[1])\Phi'^{-}_2$ $(\Phi'^{-}_1 \approx \Phi'^{-}_2)$. Let Φ'^{-}_1 be not empty. Then $\exists R', j(\Phi'_1[2] \approx R'_r[j]\ \&\ \Phi'_2[2] \approx R'_r[j])$, and $\Phi'_1[2] \neq \Phi'_2[2]$. Let us denote the prefix $R'_r[1]_{(x+j-2)}R'_r[2]_{(x+j-3)}...R'_r[j-1]_{(x)}$ by $R'^{(j-1)}_{(x)}$ and turn back to R' applications that generate $\Phi'_1[2]$ and $\Phi'_2[2]$. They look like $R'^{(j-1)}_{(l_1)}\Phi'^{-}_1\Theta_0$ and $R'^{(j-1)}_{(l_2)}\Phi'^{-}_2\Psi\Theta_0$ and form the Turchin pair. This contradicts the choice of $\Phi_1\Theta_0$ and $\Phi_2\Psi\Theta_0$.

Hence $\Phi_1\Theta_0 = R(\Phi'_1[1])\Theta_0$ and $\Phi_2\Psi\Theta_0 = R(\Phi'_2[1])\Psi\Theta_0$.

Let us prove that Turchin pairs that end $\mathfrak{G}$-grammar bad sequences can appear only through reusing a NE-rule without its cancellation.

Proposition 3 *If Φ_1 and $\Phi_2$form the first Turchin pair in a $\mathfrak{G}$-grammar computation then both $\Phi_2[1]$ and $\Phi_1[1]$ coincide up to the time indices with some $R_r[1]$, and the application of the corresponding $R : x \to R_r$ that generates Φ_1 is not canceled in Φ_2.*

Proof. Let R be canceled somewhere between the two applications of R to generate Φ_1 and Φ_2. Let $\Phi_1 = R(\Theta_0[1])\Theta^-_0$, $\Phi_2 = R(\Psi[1])\Psi^-\Theta^-_0$. In the segment from Φ_1 to Φ_2 there is the moment Θ^-_0. This implies that $\Theta^-_0[1]$ is to be transformed and cannot be in $R(\Psi[1])\Psi^-\Theta^-_0$. Contradiction.

Now it is possible to present a strategy of building the maximal bad sequence by a $\mathfrak{G}$-grammar and to estimate the length of this bad sequence.

An application of a rule R_1 is called worse than an application of R_2 iff it allows to construct a longer maximal bad sequence.

Lemma 3 *If some NE-rule is canceled (or never used) then its application to Θ_0 is worse than shortening of Θ_0 (applying the rule of the form $x \to \Lambda$).*

Proof. Let R be a NE-rule that is canceled or never used. No word in the track segment from $R(\Theta_0[1])\Theta^-_0$ to Θ^-_0 (the cancellation of $R(\Theta_0[1])$) can terminate the bad sequence with a word after Θ^-_0. The proof is by contradiction. Let a Turchin pair exist on this segment. Let the first member of the Turchin pair be $R'(a)\Theta_1$, and the second — $R'(b)\Psi\Theta_1$. $\Theta^-_0[1]$ remains unchanged until R cancellation, therefore cannot occur in $R'(a)$. So $\Theta^-_0[1]$ occurs in Θ_1, but then the word $R'(b)\Psi\Theta_1$ cannot contain it, because $\Theta^-_0[1]$ is erased after the moment Θ^-_0. So a shortening cannot be worse than an application of an unused R.

Therefore the worst strategy of building bad sequences by a $\mathfrak{G}$-grammar is to apply NE-rules until each of them is used exactly once, and then to shorten until a first cancellation (and so on). Thus all the maximal bad sequences that retain a suffix Θ_0 must end by the word Θ_0.

Using this strategy we can find the maximal bad sequence length by induction.

Theorem 3 *If NE-rules are applied in an order $R_0, R_1, ..., R_N$ to the initial word of the length 1, the maximal length of a bad sequence is*

$$C'_{Max} = 1 + |R_0| * (1 + |R_1| * (... * (1 + |R_N|)...))$$

Proof. If only one unused (or canceled) NE-rule remains at a moment Θ_0 then the length of the bad sequence segment that starts by Θ_0 and ends by Θ_0^- is $|R| + 1$ (one application of R and $|R|$ shortenings).

Suppose we know how to find the maximal bad sequence length in the case of N unused (or canceled) rules. Let $N + 1$ unused (or canceled) NE-rules remain. Application of an arbitrary R to Θ_0 results in $R(\Theta_0[1])\Theta_0^-$. Let us build the maximal bad sequence segment starting from $R(\Theta_0[1])\Theta_0^-$ that preserves $R(\Theta_0[1])^-\Theta_0^-$ (the length of this bad sequence segment is known under the assumption and is denoted by $C(N)$). This segment ends by the word $R(\Theta_0[1])^-\Theta_0^-$. Again let us build the maximal bad sequence segment from this moment that preserves $R(\Theta_0[1])^{--}\Theta_0^-$, and so on, until the computation reaches the moment Θ_0^-. Every letter in the right-hand side of R generates $C(N)$ words in the bad sequence, and the application of R generates one more additional word. So the total increment is $1 + |R| * C(N)$. If all the rules are of the same length, the upper bound from Theorem 1 is exact in the class $\mathfrak{G}$.

If the initial Γ_0 is not of the length 1 then C_{Max} must be multiplied by $|\Gamma_0|$. C_{Max} reaches the maximum if the rules in the sequence $R_0, R_1, ..., R_N$ are ordered by non-increasing length. Thus the algorithm of building the longest bad sequence by $\mathfrak{G}$-grammars can be formulated.

1. While unused (or canceled) NE-rules remain, apply the longest of them.
2. Shorten until a first cancellation.

Note that the sum runs over all distinct right-hand sides: if several rules of the form $a_1 \to R_r$, ... , $a_n \to R_r$ exist, the length of R_r is counted only once in C_{Max}. This prevents an exponential growth of the bad sequence length when converting to a $\mathfrak{G}$-grammar.

Example 4 *Let us estimate the maximal length of an bad sequence in the $\mathfrak{G}$-grammar $\mathbf{G}'_{\mathbf{F}}$ (Example 2). The length of the initial word is 2, the length of the two NE-rules — R_1 and R_2 — is also 2, and the length of the last rule is 1. So the maximal bad sequence length is $2 * (1 + 2 * (1 + 2 * (1 + 1))) = 22$.*

Now let us build the bad sequence explicitly. For readability we rename the same letters with different superscripts to the different ones.

$\mathbf{G'_F}$:
$R_0 : x \to ab$ $R_2 : x \to ef$
$R_1 : x \to cd$ $R_3 : x \to g$
$R_4 : x \to \Lambda$

Rules' names in the computation are dropped.

$\Gamma_0 :$ $\mathbf{a}_{(1)}b_{(0)}$	$\Gamma_{11} : \mathbf{b}_{(0)}$
$\Gamma_1 :$ $\mathbf{c}_{(3)}d_{(2)}b_{(0)}$	$\Gamma_{12} : \mathbf{c}_{(13)}d_{(12)}$
$\Gamma_2 :$ $\mathbf{e}_{(5)}f_{(4)}d_{(2)}b_{(0)}$	$\Gamma_{13} : \mathbf{e}_{15}f_{(14)}d_{(12)}$
$\Gamma_3 :$ $\mathbf{g}_{(6)}f_{(4)}d_{(2)}b_{(0)}$	$\Gamma_{14} : \mathbf{g}_{16}f_{(14)}d_{(12)}$
$\Gamma_4 :$ $\mathbf{f}_{(4)}d_{(2)}b_{(0)}$	$\Gamma_{15} : \mathbf{f}_{(14)}d_{(12)}$
$\Gamma_5 :$ $\mathbf{g}_{(7)}d_{(2)}b_{(0)}$	$\Gamma_{16} : \mathbf{g}_{(17)}d_{(12)}$
$\Gamma_6 :$ $\mathbf{d}_{(2)}b_{(0)}$	$\Gamma_{17} : \mathbf{d}_{(12)}$
$\Gamma_7 :$ $\mathbf{e}_{(9)}f_{(8)}b_{(0)}$	$\Gamma_{18} : \mathbf{e}_{(19)}f_{(18)}$
$\Gamma_8 :$ $\mathbf{g}_{(10)}f_{(8)}b_{(0)}$	$\Gamma_{19} : \mathbf{g}_{(20)}f_{(18)}$
$\Gamma_9 :$ $\mathbf{f}_{(8)}b_{(0)}$	$\Gamma_{20} : \mathbf{f}_{(18)}$
$\Gamma_{10} : \mathbf{g}_{(11)}b_{(0)}$	$\Gamma_{21} : \mathbf{g}_{(21)}$

Application of any rule R_1–R_4 to Γ_{21} results with a Turchin pair.

If the alphabet Υ contains at least two letters, the upper bound provided by Theorem 1 is more exact than the upper bound from Section 2 (extracted from the proof of Turchin's theorem). Truly, let all NE-rules be of the length k and their quantity be N. Then the former bound C_{Max} can be written as

$$C_{Max} = |\Upsilon|^{|\Gamma_0|+\sum(|R_i|-1)} = card(\Upsilon)^{|\Gamma_0|} * card(\Upsilon)^{(k-1)*N} = card(\Upsilon)^{|\Gamma_0|} * (card(\Upsilon)^{k-1})^N.$$

If $card(\Upsilon) \geq 2$ then $|\Gamma_0| < card(\Upsilon)^{|\Gamma_0|}$ and $k \leq card(\Upsilon)^{k-1}$.

5 Further Properties of Turchin's Relation

To be applicable in program transformation, Turchin's relation must have a transitive almost well subset, which allows to construct almost well intersections of it with other almost well relations. It seems reasonable to apply the Infinite Ramsey Theorem to get a proof of this fact but there are some subtle points that emerge from the fact that we do not operate with all sequences but only with grammar-generated ones.

Example 5 *Suppose we have an arbitrary track and colour all the pairs $\langle \Phi, \Psi \rangle$ such that Φ precedes Ψ in two colors. All the pairs such that $\Psi \preceq \Phi$ are coloured green and all others are coloured red. By the Infinite Ramsey Theorem there exists an infinite set such that all the pairs it contains are of the same color [12]. If the color is red, it may seem that the contradiction with Turchin theorem is achieved, but consider the following sequence. On every i-th step all letters are erased and the word of i a-s followed by a single b is put down. No two words in this sequence form a Turchin pair. If we replace the immediate erasing of i*

letters by i erasings of a single letter and do the same with the non-erasing rules then we receive a correct prefix grammar-generated track

$\Gamma_0 : a_{(2)}b_{(1)}c_{(0)}$ $\quad$ $\Gamma_7 : \ b_{(3)}c_{(0)}$
$\Gamma_1 : b_{(1)}c_{(0)}$ $\quad$ $\Gamma_8 : \ c_{(0)}$
$\Gamma_2 : c_{(0)}$ $\quad$ $\Gamma_9 : \ b_{(6)}c_{(0)}$
$\Gamma_3 : b_{(3)}c_{(0)}$ $\quad$ $\Gamma_{10} : a_{(7)}b_{(6)}c_{(0)}$
$\Gamma_4 : a_{(4)}b_{(3)}c_{(0)}$ $\quad$ $\Gamma_{11} : a_{(8)}a_{(7)}b_{(6)}c_{(0)}$
$\Gamma_5 : a_{(5)}a_{(4)}b_{(3)}c_{(0)}$ $\quad$ $\Gamma_{12} : a_{(9)}a_{(8)}a_{(7)}b_{(6)}c_{(0)}$
$\Gamma_6 : a_{(4)}b_{(3)}c_{(0)}$ $\quad$ $\ldots$ $\quad$ $\ldots$

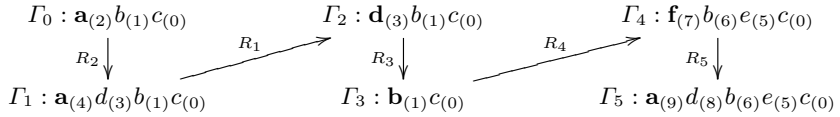
The infinite subsequence Γ_0, Γ_5, Γ_{12}, Γ_{21}, ... contains only red coloured pairs. Hovewer, there is a couple of infinite subsequences that contain only green coloured pairs. But the existence of the sequence Γ_0, Γ_5, Γ_{12}, Γ_{21}, ... shows that we cannot get the proof of their existence without considering grammar features.

Theorem 4 *The relation $\preceq$ is not a well quasi-order, but it contains a quasi-order that is well on all time-indexed tracks of alphabetic prefix-rewriting grammars.*

Proof. 1. To show the intransitivity consider the following grammar.

$\mathbf{G_{ABC}}$:
$R_1 : a \to \Lambda$ $\quad$ $R_3 : d \to \Lambda$ $\quad$ $R_5 : f \to ad$
$R_2 : a \to ad$ $\quad$ $R_4 : b \to fbe$

Let us compute from *abc*.



$\Gamma_0 \preceq \Gamma_1$ and $\Gamma_1 \preceq \Gamma_5$, but $\Gamma_0 \not\preceq \Gamma_5$.

2. Let us consider all infinite computations $\{\Phi_i\}_{i=1}^{\infty}$ s.t. $\exists N \, \forall i \, \exists j (i < j \ \& \ |\Phi_j| \leq N)$. Since the alphabet Υ is finite, there is only finite number of equivalence classes Q s.t. $\forall i, j (\Phi_i \in Q \ \& \ \Phi_j \in Q \Leftrightarrow |\Phi_i| \leq N \ \& \ \Phi_i \approx \Phi_j)$. At least one of these classes contains infinite number of elements. Let us denote this class by $\{\Phi'_i\}$. If for some i, j, k $\Phi'_i[k] \neq \Phi'_j[k]$ then the moments when $\Phi'_i[k]$ and $\Phi'_j[k]$ are generated (they are generated by the same NE-rule R) must form a Turchin pair. Thus, there is an infinite sequence of words of the form $R(a)\Phi$. Any two words from this sequence form a Turchin pair.
All other sequences satisfy a condition $\forall N \, \exists i_N \, \forall j (j > i_N \Rightarrow |\Phi_j| > N)$. For every N let us choose the first i_N such that $\forall j (j < i_N \Rightarrow \exists k (k \geq j \ \& \ |\Phi_k| < N))$. Therefore $|\Phi_{i_N - 1}| < N$ and $|\Phi_{i_N}| \geq N$, and Φ_{i_N} is generated from its predecessor by a NE-rule R, $|R| \geq 2$: $\Phi_{i_N} = R(\Phi_{i_N - 1})\Phi^-_{i_N - 1}$. $\Phi^-_{i_N - 1}$ is never changed, since $|\Phi_{i_N - 1}| < N$. All of the elements of $\{\Phi_{i_N}\}_{i=1}^{\infty}$ begin with the right-hand side of some NE-rule. Due to the finiteness of the number of the rules there is an infinite subsequence of $|\Phi_{i_N - 1}| < N$, such that all elements of the subsequence begin with the right-hand side of the same rule. Any two words from this subsequence form a Turchin pair.

Set T of all the pairs from the constructed infinite sequences is a wqo that is a subset of the Turchin relation.

Note that if $(\Phi, \Psi) \in T$ then both Φ and Ψ are generated directly by the some NE-rule. So if the Turchin condition is not checked after shortenings, it has no influence on almost wellness of the relation (or of its direct products with other almost well relations). This allows to extend Turchin's theorem on arbitrary prefix rewriting grammars with finite prefix lengths.

Theorem 5 *Turchin's relation is almost well on all time-indexed sequences of words generated by finite prefix rewriting grammars.*

Proof. Consider an application of a rule $a_1 a_2 \dots a_n \to b_1 b_2 \dots b_m$. It can be replaced by applications of the rules $a_1 \to \Lambda$, ..., $a_{n-1} \to \Lambda$, $a_n \to b_1 b_2 \dots b_m$. If all immediate moments after shortenings are ignored by the Turchin relation, then the modified sequence of words contains an bad sequence if and only if the initial sequence contains.

Turchin's theorem not only guarantees that Turchin pairs exist in every infinite computation but also implies that these pairs can be found in exponential time. In the case of Higman's pairs the upper bound is given by hyper Ackermann function even if a length of a word in computation on an i-th step is bounded by $|\Gamma_0| + i * k$ (k is a constant) [10] (for threshold results see [13]). Why does this complexity gap appear? Grammar features play a key role here. Without the grammar restrictions, Turchin pairs may not occur in an infinite computation even with a constant growth of word length (as it can be seen in the red subsequence from Example 5).

On the other hand, the Higman condition on computations imposed by $\mathfrak{G}$-grammars also appears to have exponential upper bound. The longest Higman bad sequence cannot be longer than the longest Turchin bad sequence because the latter relation is a subset of the former (up to time indexes). But in Turchin bad sequences constructed by the maximal bad sequence algorithm no Higman pairs appear, hence the maximal Higman bad sequence length is exactly the same as the maximal Turchin bad sequence length in the $\mathfrak{G}$-style computations. What is more, not only the worst-case sequence lengths but every bad sequence lengths (in the same $\mathfrak{G}$-style computation) for these two embeddings are equal.

Theorem 6 *A first Higman pair found in a $\mathfrak{G}$-grammar computation is the Turchin pair.*

Proof. Suppose that two elements Φ_1 and Φ_2 such that $\Phi_1 \trianglelefteq \Phi_2$ are of the forms $A_1 A_2 ... A_n \Theta_0$ and $B_1 A'_1 B_2 A'_2 ... B_n A'_n B_{n+1} \Theta_0$ respectively, and $\forall i (i \geq 1 \;\&\; i \leq n \Rightarrow A_i \equiv_S A'_i)$. If we turn back to the moments when $A_i[1]$ and $A'_i[1]$ were generated by $R : x \to R_r$, we receive words of the form $R^{(i-1)}_{(k_1)} A_i A_{i+1} ... A_n \Theta_0$ and $R^{(i-1)}_{(k_2)} A'_i B_{i+1} ... B_n A_n B_{n+1} \Theta_0$ ($R^{(i-1)}_{(k_j)}$ denotes the prefix $R_r[1]_{(k_j+i-2)} ... R_r[i-1]_{(k_j)}$). They form a Higman pair so $i = 1 = n$, or it contradicts the fact that Φ_1 and Φ_2 is the first generated Higman pair. But if $i = 1 = n$ then $\Phi_1 = A_1 \Theta_0$ and $\Phi_2 = A'_1 B_{n+1} \Theta_0$, so $\Phi_1 \preceq \Phi_2$.

Thus the intersection of Higman and Turchin conditions also has the exponential upper bound on prefix-rewriting grammar generated computations.

6 Conclusion and further work

The considered relation was initially presented by V. F. Turchin for finding loop approximations on configurations of function stacks in semantics trees paths. In general the relation remained unstudied. Now we have shown that this relation posesses the same properties as the Higman relation on prefix-rewriting grammar generated tracks, but is more "powerful" than Higman relation in the sense of admitting longer bad sequences. Now it is clear that Turchin's embedding is a safe and universal refinement for finding regular loop approximations in program analysis. The relation allows to observe not only a structure of computational states but also their histories. It is easy to be checked and can be applied to estimate lengths of bad sequences for other almost well embeddings on grammar-generated sequences.

Although Turchin's relation can be considered asa refinement for Higman relation, it is not clear yet if there exists an analogue of Turchin's relation for trees instead of sequences that is almost well on tracks of some practical class of rewriting systems. It would be interesting to consider this question as well.

Acknowledgments. The author is very grateful to A.P. Nemytykh who encouraged and directed the study and whose comments helped to significantly improve the text.

References

1. Albert, E., Gallagher, J., Gomez-Zamalla, M., Puebla, G.: Type-based Homeomorphic Embedding for Online Termination. In Journal of Information Processing Letters. Vol. 109(15) (2009), pp. 879–886.
2. Caucal, D.: On the Regular Structure of Prefix Rewriting. Theoretical Computer Science, vol. 106 (1992), pp. 61–86.
3. Bolingbroke, M. C., P. Jones, S.L., Vytiniotis, D.: Termination combinators forever. In Proceedings of the 4th ACM SIGPLAN Symposium on Haskell, Haskell 2011, Tokyo, Japan, pp. 23–34.
4. Higman, G.: Ordering by divisibility in abstract algebras. In Bulletin of London Math. Soc. Vol. 3(2) (1952), pp. 326–336.
5. Klyuchnikov, I., Romanenko, S. Proving the equivalence of higher-order terms by means of supercompilation. In Perspectives of Systems Informatics, LNCS, vol. 5947 (2010), pp. 193–205.
6. Kruskal, J. B.: Well-quasi ordering, the tree theorem, and Vazsonyi?s conjecture. In Transactions of the American Mathematical Society, 95 (1960), pp. 210–225.
7. Nash-Williams, C.St.J.A.: On well-quasi-ordering infinite trees. In Proc. Cambr. Philos. Soc. Vol. 61(1965), pp. 697–720.
8. Nemytykh, A.P.: The SCP4 supercompiler: general structure. Moscow, 2007. 152 p. (in Russian)

9. Sørensen, M.H., Glück, R.: An algorithm of generalization in positive supercompilation. In Logic Programming: Proc. of the 1995 International Symposium (1995), pp. 465–479.
10. Touzet, H.: A Characterisation of Multiply Recursive Functions with Higman's Lemma. In Information and Computation. Vol. 178 (2002), pp. 534–544.
11. Turchin, V.F.: The algorithm of generalization in the supercompiler. In Partial Evaluation and Mixed Computation (1988), pp. 341–353.
12. Veldman, W., Bezem, M.: Ramsey's theorem and the pigeonhole principle in intuitionistic mathematics. In Journal of London Mathematical Society, Vol. 47(2) (1993), pp. 193–211.
13. Weiermann, A.: Phase transition thresholds for some Friedman-style independence results. In Mathematical Logic Quarterly, Vol. 53 (2007), no. 1, pp. 4–18.