Trakhtenbrot theorem for classical languages with three individual variables

Mikhail Rybakov
Institute for Information Transmission Problems
Russian Academy of Sciences
Russia
Department of Mathematics
Tver State University
Russia
School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg
South Africa
m_rybakov@mail.ru

Dmitry Shkatov
School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg
South Africa
shkatov@gmail.com

ABSTRACT
We present a simple proof of Thrakhtenbrot's theorem for the classical predicate logic in the language with only three individual variables. Both forms of Thrakhtenbrot's theorem are established: we prove that the classical predicate logic $QCL$ over finite domains is not recursively enumerable in the language with only three individual variables and that the set of theorems of $QCL$ over arbitrary domains and the set of non-theorems of $QCL$ over finite domains, in the language with only three individual variables, form a recursively insep-arable pair of recursively enumerable sets. The techniques used here can be generalised to obtain similar results for non-classical predicate logics with further restrictions on their vocabularies.

CCS CONCEPTS
• Theory of computation → Logic;

KEYWORDS
Finite model theory, recursive enumerability, Thrakhtenbrot theorem, three-variable fragment.

1 INTRODUCTION
One of the most fundamental achievements of theoretical computer science in the 20th century, on a par with Alonzo Church's [5] and Alan Turing's [28] proofs of the undecidability of the classical first-order predicate logic $QCL$, is Boris Trakhtenbrot's [25] proof of the undecidability of the classical first-order predicate logic of finite domains $QCL_{fin}$. It led to the emergence of whole subfields of theoretical computer science such as finite model theory [12] and descriptive complexity [9], and it lies at the foundation of database theory [1].

Trakhtenbrot has also shown that the set of theorems of $QCL$ and the set of non-theorems of $QCL_{fin}$ are recursively inseparable [26]—this is the form in which "Trakhtenbrot's theorem" is often presented. More recently, similar results have been proven for other languages and corresponding classes of finite structures (see, e.g.,[8] and [3]).

Once a negative result like Church's and Trakhtenbrot's theorems has been obtained, the question naturally arises whether one can avoid the undesirable computational complexity of a formal language involved by restricting its expressive power. This, in particular, has lead to the emergence of a very broad and successful research area of "the classical decision problem" (see, e.g., [4]).

Two very natural ways of restricting the expressive power of a predicate language are, first, limiting the number of individual variables and, second, limiting the number and...
arity of predicate letters allowed in the construction of formulas. A number of results on the effects of such restrictions are known in the literature. Thus, it is known that $\Sigma$ is decidable in the language with two variables [7, 13], but is undecidable in the language with three variables [23]. It is also known that the intuitionistic predicate logic and most natural predicate modal logics are undecidable in the language with two individual variables [10], even if only a single monadic predicate letter is allowed in the construction of formulas [18]; decidable fragments of such logics are described in [30],—in most instances, they are equivalent to one-variable fragments, as studied in [21].

When it comes to computational properties of $\Sigma$ in restricted languages, the most widely known result is, probably, that the presence of a single binary and an unlimited number of monadic predicate letters, leaving out equality, makes $\Sigma$ not recursively enumerable [29]. (By contrast, the presence of a single binary predicate letter makes $\Sigma$ undecidable, as follows from [6].) No explicit proofs, as far as we know, are present in the literature on the inenumerability of $\Sigma$ in a language with a restricted number of individual variables. It does, however, follow from the facts that finitely presentable relation algebras are not recursively axiomatizable [2] and that the $\Sigma$ with three variables has the same expressive power as relation algebras [24] that $\Sigma$ with three variables is not recursively enumerable.

In the present paper, we give a very simple and concise proof of that result “from the first principles”, by encoding a not recursively enumerable problem about Turing machines using classical predicate formulas with only three individual variables. Moreover, we extend the proof to obtain a more general form of Trakhtenbrot’s theorem for such languages—namely, to prove that the set of theorems of $\Sigma$ in the language with three variables is recursively inseparable.

The paper is structured as follows. In Section 2, we present preliminaries pertaining to computability theory. In Section 3, we introduce the syntax and semantics of the classical predicate logic. In Section 4, we present the proof that $\Sigma$ is not recursively enumerable in the language with three variables. Then, in Section 5, we present the proof of the recursive inseparability of the set of theorems of $\Sigma$ and non-theorems of $\Sigma$ in the language with three variables. We conclude in Section 6.

2 PRELIMINARIES

We briefly recall the notions, and introduce the notation we are going to use, pertaining to Turing Machines; for more background, we refer the reader to [11] and [22].

An alphabet is a finite set $\Sigma$ of symbols. A word over $\Sigma$ is a finite sequence of elements of $\Sigma$; the set of words over $\Sigma$ is denoted by $\Sigma^*$. We use $\epsilon$ to denote the empty sequence of symbols.

A Turing machine is a tuple $M = (Q, q_0, q_1, \Sigma, \delta)$, where $Q$ is a finite set of states; $q_0 \in S$ is the initial state; $q_1 \in S$ is the halting state; $\Sigma$ is an alphabet such that $s_0, s_1 \in \Sigma$ and $\{L, S, R\} \cap \Sigma = \emptyset$ (where $s_0$ is the blank symbol, and $s_1$ is the end tape marker symbol); and $\delta$ is the program. The program $\delta$ is a finite list of instructions of the form $q s \rightarrow q' s' D$, where $q \in Q - \{q_1\}$, $q' \in Q$, $s' \in S$, and $D \in \{L, S, R\}$, such that

- for every combination of $q \in Q - \{q_1\}$ and $s \in S$, the program $\delta$ contains exactly one instruction $q s \rightarrow q' s' D$ (i.e., $M$ is deterministic);
- if $q s_1 \rightarrow q' s' D \in \delta$, then $s' = s_1$ and $D \neq L$ (i.e., $M$ never overwrites the end tape marker and never moves its head to the left of the marker);
- if $q s \rightarrow q' s' D \in \delta$ and $s \neq s_1$, then $s' \neq s_1$ (i.e., $M$ never writes the end marker symbol over any other symbol).

A configuration of a Turing machine $M = (Q, q_0, q_1, \Sigma, \delta)$ is a tuple $(q, v, v')$, where $q \in Q$, $v$ is a word over $\Sigma$ beginning with $s_1$, and $v'$ is either an $\epsilon$ or a word over $\Sigma$ different from $\epsilon$ and not ending with $s_0$; the machine is understood to be scanning the last symbol of $v$. A configuration of the form $(q_1, v, v')$ is a halting configuration. A configuration $(q, v, u)$ yields configuration $(q', v', u')$ in one step, symbolically $(q, v, u) \rightarrow_M (q', v', u')$. If $v = ws$, for some $s \in S$, the instruction $q s \rightarrow q' s' D$ is in the program $\delta$, and

- if $D = S$, then $v' = ws'$ and $u' = u$;
- if $D = R$, then if $u \neq \epsilon$, then $v'$ is $ws'$ with the first symbol of $u$ appended to its end and $u'$ is $u$ with the first symbol removed; if, on the other hand, $u = \epsilon$, then $v' = ws' s_0$ and $u' = \epsilon$;
- if $D = L$, then $v' = w$ and, if $u \neq \epsilon$ or $s \neq s_0$, then $u'$ is $u$ with $s'$ attached to its beginning; if, on the other hand, $u = \epsilon$ and $s = s_0$, then $u' = \epsilon$.

A computation of $M$ on input $v$ is a sequence of configurations $\lambda = \lambda_0, \lambda_1, \ldots$ such that $\lambda_0 = (q_0, s_1, v)$ and

- either $\lambda$ is infinite and $\lambda_i \rightarrow_M \lambda_{i+1}$ holds for every $i \geq 0$, i.e., no configuration of $\lambda$ is a halting configuration,
- or $\lambda = \lambda_0, \lambda_1, \ldots, \lambda_k$, for some $k \geq 0$ (i.e., $\lambda$ is finite), $\lambda_i \rightarrow_M \lambda_{i+1}$ holds for every $i \in \{0, \ldots, k - 1\}$, and $\lambda_k$ is a halting configuration.

We assume that $M$ accepts the input if it reaches the configuration $\lambda_1$ and rejects the input if it reaches a configuration $\lambda_j$ such that $\lambda_j \neq \epsilon$.

A set $A$ of words over an alphabet $\Sigma$ is recursive if its characteristic function

$$\chi_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \in \Sigma^* - A \end{cases}$$
We assume that the language of the classical predicate logic is computable; equivalently, if there exists a Turing machine accepting every \( x \in A \) and rejecting every \( x \notin A \).

A set \( A \) of words is recursively enumerable if \( A = f(\mathbb{N}) \), for some computable function \( f \); equivalently, \( A \) is recursively enumerable if \( A = \text{Dom}(f) \), for some computable function \( f \).

Sets \( X \) and \( Y \) are recursively inseparable if \( X \cap Y = \emptyset \) and there exists no recursive set \( A \) such that \( X \subseteq A \) and \( A \cap Y = \emptyset \).

Every Turing machine \( M \) can be represented as a string of symbols in a suitable alphabet; we refer to such a string as an encoding of \( M \).

It is well-known [16] that the set of encodings of Turing machines that do not terminate on \( \epsilon \) is not recursively enumerable.

### 3 CLASSICAL PREDICATE LOGIC

We assume that the language of the classical predicate logic contains a countable supply of \( n \)-ary predicate letters \( P_1^n, P_2^n, \ldots \), for any \( n \in \mathbb{N} \), a countable supply of individual variables \( x_1, x_2, \ldots \), the Boolean connectives \( \land, \lor, \rightarrow, \leftrightarrow \), and \( \exists \) and \( \forall \) as defined in the usual way; when parentheses are left out, \( \land \) and \( \lor \) are understood to bind tighter than \( \rightarrow \) and \( \leftrightarrow \). We usually write atomic formulas in prefix notation; for some predicate letters, however, we use infix notation.

A model is a pair \( \mathfrak{M} = (D, I) \), where \( D \) is a non-empty set called the domain of \( \mathfrak{M} \), and \( I \) is an interpretation function assigning to each \( n \)-ary predicate letter \( P^n \) a subset \( I(P^n) \) of \( D^n \). A model \( \mathfrak{M} = (D, I) \) is finite if \( D \) is a finite set.

An assignment \( g \) is a function assigning to each individual variable \( x \) an element \( g(x) \) of \( D \).

The truth of formulas \( \varphi \) in a model \( \mathfrak{M} \) under the assignment \( g \) is defined inductively, as follows:

- \( \mathfrak{M} \models \varphi \) if \( (g(x_1), \ldots, g(x_n)) \in I(P) \);
- \( \mathfrak{M} \models \varphi_1 \land \varphi_2 \) if \( \mathfrak{M} \models \varphi_1 \) and \( \mathfrak{M} \models \varphi_2 \);
- \( \mathfrak{M} \models \forall x \varphi_1 \) if \( \mathfrak{M} \models \varphi'_1 \) for every assignment \( g' \) such that \( g' \) differs from \( g \) in at most the value of \( x \).

We say that formula \( \varphi \) is true in a model \( \mathfrak{M} \) and write \( \mathfrak{M} \models \varphi \) if \( \mathfrak{M} \models \varphi \) holds for every assignment \( g \).

We denote by QCL the set of formulas true in every model and by QCL\(_{\text{fin}}\) the set of formulas true in every finite model.

We denote by QCL(3) and QCL\(_{\text{fin}}(3)\) the subsets of QCL and QCL\(_{\text{fin}}\), respectively, containing formulas with at most three distinct individual variables, free or bound.

### 4 RECURSIVE INENUMERABILITY

In this section, we reduce the problem of non-termination of a Turing Machine \( M \) on the empty word \( \epsilon \) to satisfiability in QCL\(_{\text{fin}}\) using formulas with only three individual variables.

This gives us a proof of the weaker form of Trakhtenbrot’s theorem for languages with only three individual variables.

We begin by introducing auxiliary formulas that define, among other things, a linear partial ordering \( P(x, y) \) with the associated “immediate successor” relation \( \text{Next}(x, y) \) and the minimal element \( \text{Min}(x) \). In these formulas, \( \equiv \) is a binary predicate letter that will be interpreted as a congruence relation (i.e., it is not an identity symbol, which is absent from the languages we consider). Let

\[
\begin{align*}
\alpha_0 & = \forall x \epsilon \land \forall x \forall y (x \approx y \rightarrow y \approx x) \\
\alpha_1 & = \forall x \not\approx P(x, y) \\
\alpha_2 & = \forall x \forall y \forall z (P(x, y) \land P(y, z) \rightarrow P(x, z)) \\
\alpha_3 & = \forall x \forall y (P(x, y) \land \not\approx y \rightarrow x \approx y) \\
\alpha_4 & = \forall x \forall y (\text{Next}(x, y) \leftrightarrow P(x, y) \land \not\approx z (P(x, z) \land P(z, y))) \\
\alpha_5 & = \forall x (\text{Min}(x) \leftrightarrow \forall y (x \not\approx y \rightarrow P(x, y))).
\end{align*}
\]

Let \( \alpha \) be the conjunction of formulas \( \alpha_0 \) through \( \alpha_5 \).

We now encode the set of Turing machines that do not terminate on \( \epsilon \) using predicate formulas of three individual variables.

In the encoding, we use the following predicate letters, with the corresponding intuitive meaning:

\[
\begin{align*}
C(x, y) & = \text{at step } x, M \text{ is scanning cell number } y; \\
Q_k(x) & = \text{at step } x, M \text{ is in state } q_k; \\
S_k(x, y) & = \text{at step } x, \text{ cell number } y \text{ contains symbol } s_k.
\end{align*}
\]

First, we describe the initial configuration of \( M \) on \( \epsilon \):

\[
\varphi_{\text{start}} = \exists x (\text{Min}(x) \land C(x, x) \land Q_0(x) \land S_1(x, x) \land \forall y (x \not\approx y \rightarrow S_0(x, y)))
\]

Next, we say that, at any step of a computation, \( M \) is scanning a unique cell, is in a unique state, and that every cell of \( M \)'s tape contains a unique symbol:

\[
\begin{align*}
\varphi_C & = \forall x \exists y (C(x, y) \land \forall z (C(x, z) \rightarrow z \approx y)); \\
\varphi_Q & = \forall x \bigvee_{k \neq j} (Q_k(x) \land \lnot Q_j(x)); \\
\varphi_S & = \forall x \forall y \bigvee_{k \neq j} (S_k(x, y) \land \lnot S_j(x, y)).
\end{align*}
\]

We now describe \( M \)'s instructions, which have the form \( I = q_0a_1 \rightarrow q ka_1D \), where \( D \in \{ S, R, L \} \). We, thus, have three cases to consider.
Case $D = S$.

$$q_I = \forall x \forall y (Q_I(x) \land C(x, y) \land S_I(x, y) \rightarrow$$

$$\exists z \text{Next}(x, z) \land \forall z (\text{Next}(x, z) \rightarrow Q_z(z) \land C(z, y) \land S_I(z, y) \land$$

$$\forall z (z \neq y \rightarrow$$

$$\land (S_I(x, z) \rightarrow \forall y (\text{Next}(x, y) \rightarrow S_I(y, z))))).$$

Case $D = R$.

$$q_I = \forall x \forall y (Q_I(x) \land C(x, y) \land S_I(x, y) \rightarrow$$

$$\exists z \text{Next}(x, z) \land \forall z (\text{Next}(x, z) \rightarrow$$

$$Q_z(z) \land \forall x (\text{Next}(y, x) \rightarrow$$

$$C(z, x) \land S_I(z, y) \land$$

$$\forall z (z \neq y \rightarrow$$

$$\land (S_I(x, z) \rightarrow \forall y (\text{Next}(x, y) \rightarrow S_I(y, z))))).$$

Then, the following formula describes M’s program:

$$\varphi_{\text{prog}} = \bigwedge_{i \in b} q_i$$

We also define the formula asserting that $M$ reaches a halting configuration:

$$\varphi_{\text{stop}} = \exists x Q_i(x).$$

Let $\text{Congr}$ be the formula asserting that $\approx$ is a congruence relation with respect to the predicate letters we use, i.e., the conjunction of formulas

$$\forall x \forall y (x \approx y \rightarrow (P(x) \rightarrow P(y)))$$

and

$$\forall x \forall y (x \approx y \rightarrow ((S(x, z) \rightarrow S(z, y)) \land (S(x, z) \rightarrow S(y, z))),$$

where $P$ ranges over the set of the unary, and $S$ binary, predicate letters we use.

Finally, let

$$\varphi_M = \alpha \land \text{Congr} \land \varphi_{\text{start}} \land \varphi_C \land \varphi_Q \land \varphi_S \land \varphi_{\text{prog}}.$$
5 RECURSIVE INSEPARABILITY

We now extend the methods used in the previous section to prove the strong version of Trakhtenbrot’s theorem for the classical logic with three individual variables, i.e., to show that the set of theorems of QCL and the set of non-theorems of QCL_{fin} in the language with only three individual variables form a recursively inseparable pair of recursively enumerable sets.

**Theorem 5.1.** There is no recursive set \( X \) such that QCL(3) \( \subseteq X \subseteq \text{QCL}_{fin}(3) \).

**Proof.** It is known (see, e.g., [14, Theorem 3.3]) that there do exist recursively inseparable pairs of recursively enumerable sets. Let \( A \) and \( B \) be such a pair. We may assume without a loss of generality that \( A, B \subseteq \mathbb{N} \). Since both \( A \) and \( B \) are recursively enumerable, the function \( f \) defined by

\[
  f(n) = \begin{cases} 
    0, & \text{if } n \in A, \\
    1, & \text{if } n \in B, \\
    \text{undefined otherwise}
  \end{cases}
\]

is partial recursive. Let \( M \) be the Turing machine that computes \( f \). We assume that the input \( n \in \mathbb{N} \) is represented in unary using, say, the symbol \( s_2 \), i.e., \( s_2 \) is written in the cells 1 through \( n \) at the beginning of the computation. We assume that \( M \), given \( n \in A \), halts with \( s_0 \) written in cell number 1; if, on the other hand, \( n \in B \), then \( M \) halts with \( s_2 \) written in cell number 1.

Define the formula \( \varphi_n \) describing the initial configuration of \( M \) on input \( n \), as follows. First, inductively define the following sequence of formulas:

\[
\begin{align*}
  N_0(x, z) & = \forall y \ (P(z, y) \rightarrow S_0(x, y)); \\
  N_{2m+1}(x, y) & = \exists z \ (\text{Next}(y, z) \land S_2(x, z)) \\
  & \land N_{2m}(z, x)); \\
  N_{2m+2}(x, z) & = \exists y \ (\text{Next}(y, z) \land S_2(y, x)) \\
  & \land N_{2m+1}(x, y)).
\end{align*}
\]

Then, let

\[
  \varphi_n = \exists x \ (\text{Min}(x) \land C(x, x) \land Q_0(x) \land S_1(x, x) \land N_n(x, x)).
\]

Let the formulas \( \alpha, \text{Congr}, \varphi_C, \varphi_Q, \varphi_S \), and \( \varphi_{prog} \) be as defined in Section 4. Then, the following formula describes the computation of \( M \) on \( n \):

\[
  \varphi_{M,n} = \alpha \land \text{Congr} \land \varphi_n \land \varphi_C \land \varphi_Q \land \varphi_S \land \varphi_{prog}.
\]

Finally, we describe the final configuration of \( M \) when starting with input \( n \in A \):

\[
  \varphi_A = \exists x \ (Q_1(x) \land \exists y \exists z \ (\text{Min}(y) \land \text{Next}(y, z) \land S_0(x, z))).
\]

Before proceeding, we prove two auxiliary lemmas.

**Lemma 5.2.** If \( n \in A \), then \( \varphi_{M,n} \rightarrow \varphi_A \in \text{QCL} \).

**Proof.** Assume that \( \varphi_{M,n} \rightarrow \varphi_A \notin \text{QCL} \). Thus, \( \mathcal{M} \models \varphi_{M,n} \) and \( \mathcal{M} \not\models \varphi_A \), for some model \( \mathcal{M} = \langle D, I \rangle \). We need to show that \( n \notin A \).

Since \( \mathcal{M} \models \text{Congr} \), the domain \( D \) of \( \mathcal{M} \) is divided up into equivalence classes with respect to \( \equiv(a) \); thus, let \( \overline{a} = \{ b \in D : \langle a, b \rangle \in I(\equiv) \} \) and \( \overline{D} = \{ \overline{a} : a \in D \} \). Since \( \mathcal{M} \models a_1 \land a_2 \land a_3 \), the relation \( I(P) \) is a total strict partial order on \( \overline{D} \) in the sense that, if we pick one element form each \( \overline{a} \in \overline{D} \), then \( I(P) \) is a total strict partial order on such a set. Since \( \mathcal{M} \models \text{Congr} \), we might think of this partial order as being defined on the elements of \( \overline{D} \). Since \( \mathcal{M} \models a_4 \), the said partial order is discrete.

Since \( \mathcal{M} \models \varphi_{M,n} \), there exists an element \( c_0 \in D \) such that \( \mathcal{M} \models \text{Min}(c_0) \).

Since \( \mathcal{M} \models \varphi_{M,n} \land \varphi_C \land \varphi_Q \land \varphi_S \land \varphi_{prog} \), there exists an infinite chain of elements \( c_0, c_1, \ldots \) such that \( \mathcal{M} \models \text{Next}(c_i, c_{i+1}) \), for every \( i \geq 0 \). The existence of the“next”element in this chain is guaranteed by the subformula \( \exists z \ (\text{Next}(x, z) \) of the formula \( \varphi_T \). As the set \( \{ c_0, c_1, \ldots \} \) with the relation \( I(P) \) is isomorphic to \( \langle \mathbb{N}, \lt \rangle \), we may associate with every such \( c_k \) the number \( k \in \mathbb{N} \).

Now, let \( \lambda = \lambda_0 \uparrow \lambda_1 \uparrow \ldots \) be the computation of \( M \) on \( n \). Since \( \mathcal{M} \models \varphi_{M,n} \land \varphi_C \land \varphi_Q \land \varphi_S \land \varphi_{prog} \), we can prove by routine induction on \( m \in \mathbb{N} \) that, if \( \lambda_m = \langle q_k, v_{s_1}, v \rangle \) is a configuration of \( \lambda \), then

- \( \mathcal{M} \models Q_k(a) \), for some \( a \in D \) such that \( a \in \overline{c}_m \);
- \( \mathcal{M} \models C(a, b) \), for some \( a, b \in D \) such that \( a \in \overline{c}_m \) and \( b \in \overline{c}_{|v|-1} \);
- \( \mathcal{M} \models S_i(a, b) \), for some \( a, b \in D \) such that \( a \in \overline{c}_m \) and \( b \in \overline{c}_{|v|-1} \).

Now, assume, for the sake of contradiction, that \( n \in A \). Then, for some \( h \geq 0 \), we have \( \lambda_h = \langle q_k, v_{s_1}, s_0 \rangle \), i.e., \( M \) halts with \( s_0 \) written in its cell number 1. Then,

\[
  \mathcal{M} \models Q_1(c_k) \land \exists y \exists z \ (\text{Min}(y) \land \text{Next}(y, z) \land S_0(c_k, z)).
\]

Since \( \mathcal{M} \models \neg \varphi_A \), this gives us a sought contradiction. \( \square \)

**Lemma 5.3.** If \( n \in B \), then \( \varphi_{M,n} \rightarrow \varphi_A \notin \text{QCL}_{fin} \).

**Proof.** Assume that \( \varphi_{M,n} \rightarrow \varphi_A \notin \text{QCL} \) such that \( \mathcal{M} \models \varphi_{M,n} \) and \( \mathcal{M} \not\models \varphi_A \). Let \( \lambda = \lambda_0 \uparrow \lambda_1 \uparrow \ldots \uparrow \lambda_h \) be the computation of \( M \) on \( n \). Let \( \mathcal{M} = \langle D, I \rangle \) be a model such that

\[
\begin{align*}
  D & = \{ 0, \ldots, h \}; \\
  I(P) & = \{(x, y) : x < y \}; \\
  I(\text{Next}) & = \{(x, y) : x + 1 = y \}; \\
  I(\text{Min}) & = \{(y)\}; \\
  I(\equiv) & = \{(x, y) : x = y \}; \\
  I(C) & = \{(x, y) : \text{in } \lambda_x, M \text{ is scanning cell } y \}; \\
  I(Q_k) & = \{(x, y) : \text{in } \lambda_x, M \text{ is in state } q_k \}; \\
  I(S_k) & = \{(x, y) : \text{in } \lambda_x, \text{ cell } y \text{ contains } s_k \}.
\end{align*}
\]
It is then straightforward to check that $\forall n \models \alpha \land \text{Congr} \land \phi_n \land \phi_C \land \phi_S \land \phi_{\text{pro}}$ and that $\forall n \nvdash \phi_A$. Therefore, $\phi_M \to \phi_A \notin \text{QCL}_{\text{fin}}$, as required. $\square$

We now proceed with the proof of the theorem.

Suppose, for the sake of contradiction, that there exists a recursive set $X$ such that $\text{QCL}(3) \subseteq X \subseteq \text{QCL}_{\text{fin}}(3)$. Then, the set

$$Y = \{ n : \phi_{M,n} \to \phi_A \in X \}$$

is a recursive set separating $A$ from $B$, i.e., $A \subseteq Y$ and $B \cap Y = \emptyset$.

Indeed, $Y$ is recursive since $X$ is. Next, assume that $n \in A$. Then, due to Lemma 5.2, $\phi_{M,n} \to \phi_A \in \text{QCL}$ and, thus, $\phi_{M,n} \to \phi_A \in X$; hence, $n \in Y$. Lastly, assume that $n \in B$. Then, due to Lemma 5.3, $\phi_{M,n} \to \phi_A \notin \text{QCL}_{\text{fin}}$ and, thus, $\phi_{M,n} \to \phi_A \notin X$; hence, $n \notin Y$.

Since, by assumption, $A$ and $B$ are recursively inseparable, we have arrived at a contradiction. $\square$

**Theorem 5.4.** The sets $\text{QCL}(3)$ and $\text{QCL}_{\text{fin}}(3)$ form a recursively inseparable pair of recursively enumerable sets.

**Proof.** It is well-known that $\text{QCL}$ and $\text{QCL}_{\text{fin}}$ are both recursively enumerable (see, e.g., [15]). Clearly, this implies that $\text{QCL}(3)$ and $\text{QCL}_{\text{fin}}(3)$ are both recursively enumerable. Due to Theorem 5.1, $\text{QCL}(3)$ and $\text{QCL}_{\text{fin}}(3)$ are recursively inseparable. $\square$

6 DISCUSSION

We have given a simple proof “from the first principles” of two forms of Trakhtenbrot’s theorem for classical languages with only three individual variables.

The technique used here is simple and flexible enough to be extended to non-classical predicate logics such as modal and intuitionistic predicate logics [20], where it is furthermore possible, using techniques from [18] (those techniques are based on those originally developed for propositional languages, as presented in, e.g., [17] and [19]), to show that inenumerability results analogous to those obtained here hold if the language contains only a single monadic predicate letter.

REFERENCES


Trakhtenbrot theorem for languages with three variables

