UM-CRL-TR-10-85

THE UNIVERSITY OF MICHIGAN
COMPUTING RESEARCH LABORATORY

LOGIC AND THE CHALLENGE OF COMPUTER SCIENCE

YURI GUREVICH

CRL-TR-10-85

1079 East Engineering Bldg.
Ann Arbor, MI 48109

THE UNIVERSITY OF MICHIGAN
COMPUTING RESEARCH LABORATORY*

LOGIC AND THE CHALLENGE OF COMPUTER SCIENCE

YURI GUREVICH
CRL-TR-10-85

September 1985

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

*Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agency.

LOGIC AND THE CHALLENGE OF COMPUTER SCIENCE*

Yuri Gurevich**
Electrical Engineering and Computer Science Department
The University of Michigan, Ann Arbor, MI 48109-1109

Content:

Abstract. Now-a-days computer science is surpassing mathematics as the primary field of logic applications, but logic is not tuned properly to the new role. In particular, the following two features characterize many of the new applications: emphasis on finite structures and dynamic character of structures. These lectures address the challenges posed by the two features.

* This 10-hour series of lectures was a part of the Course on Computation Theory organized by the International Center for Mechanical Sciences in Udine, Italy in September-October 1984.

** Supported in part by NSF grants MCS 83-01022 and DCR 8503275.

§0. Introduction

These days computer science is characterized by an explosion of activities intimately related to logic. Consider for example formal languages. For years formal languages were a prerogative of logicians. But what is the most popular formal language of to-day? Is it a Hilbert type predicate calculus or is it the Genzen sequential calculus? Neither. Most popular formal languages of to-day are programming languages. Another kind of popular formal languages are database query languages. Some other formal languages emerge in artificial intelligence like languages for knowledge representation. Old discussions on names, denotations, types, etc. are suddenly revitalized to unprecedented magnitude.

The work on axiomatic semantics, logic programming and verification is related to classical proof theory, the work on computational complexity is related to classical theory of algorihtms. Even propositional logic is not left untouched by developments in computer science; for almost any number k between 3 and 20, there is a commercial circuit tester based on k-valued logic.

This is altogether a good news for logicians. Logic grows more relevant to computer science than any other part of mathematics. But the new applications call, we believe, for new developments in logic proper. First order predicate calculus and its usual generalizations are not sufficient to support the new applications. On the other hand, the new developments will most probably build on existing achievements of logic. In this connection it is worth to try to understand what made classical mathematical logic so successful?

Even though logic is an ancient subject, the origins of modern mathematical logic are closely related to the discovery of paradoxes and the subsequent crisis in foundation of mathematics [Kl]. In 1930 came the triumph of Gödel's completeness theorem. The syntax of first order predicate calculus and its semantics were proven to match perfectly. In addition the logic was restrictive enough to avoid paradoxes and expressive enough to provide a basis for Zermelo-Fraenkel set theory and resolve this way, to a large extent, the foundational crisis. This perfect match of syntax and semantics together with an adequate expressive power made first order logic an invaluable tool and a source of innumerable generalizations.

Extremely important features of first order logic are a formal language and a clear notion of models. The models are so-called first order structures or, simply, structures. (Some people object to the term "first order structure" on the ground that logic is first order rather than structures. This is a good point. But some structures are not first order, topological spaces for example; and we all know exactly what first order structures are.) This familiar pattern – a formal language with a clearly defined notion of a model – persists through familiar generalizations of first order logic.

(Let us mention another interesting feature of first order logic. Even though a consistent first order theory has usualy a multitide of models, the theory itself does not refer directly to different models; it "speaks" about the model of discourse.)

Classical logic facilitated numerous and impressive achievements. Let us mention only the Church-Turing thesis and the Gödel-Cohen resolution of the continuum hypothesis. It seems that we (the logicians) were somewhat hypnotized by the success of classical systems. We used first order logic where it fits well and where it fits not so well. We went on working on computability without paying an adequate attention to feasibility. One seemingly obvious but nevertheless important lesson is that different application may require formalizations of different kinds, that it is necessary to "listen" to the subject in order come up with a right formalization. (We philisophized on this topic in [Gu3].)

An important feature of many computer science structures is finiteness. Relational databases constitute an especially important example. Finiteness does not seem to be such a great novelty in classical logic. Nevertheless it poses a nontrivial challenge. Being so closely related to foundations of mathematics, classical logic is preoccupied with infinity. Many famous theorems collapse when only finite structures are allowed; among them are Gödel's Completeness Theorem, Craig Interpolation Theorem, Beth Definability Theorem and Substructure Preservation Theorem [Gu2].

Remark 1. Lyndon proved [Ly] that if a first-order formula $\varphi(P)$ is monotone in a predicate variable P (which means that the second order formula $P \subseteq P' \to [\varphi(P) \to \varphi(P')]$ is logically true) then $\varphi(P)$ is equivalent to a first-order formula $\psi(P)$ with only positive occurrences of P. We could not decide in [Gu2] the status of the Lyndon theorem in the finite case and conjectured a failure. The conjecture was recently confirmed [AG].

Variants of first order logic serve as standard relational query languages [Co, Ul], but the expressive power of first order logic is not sufficient for many purposes [AU]. On the other hand, second order logic is overly expressive. It expresses queries that are too hard to compute. Even existential second order formulas can express NP complete queries. Of course, the notion what is hard, may change from one application to another. One idea is to fix a reasonable complexity level, like polynomial time, and to devise an intermediate logic that "captures" this complexity level i.e. expresses exactly the queries of this complexity. The idea happen to be realizable to an extent. The pioneering papers include papers of Aho and Ullman [AU], Chandra and Harel [CH], Fagin [Fa1], Immerman [Im1] and Vardi [Va]. In particular, Immerman and Vardi proved that, in the presence of linear order, the least fixed point extension of first order logic captures polynomial time. Later Immerman "captured" log-space and a number of other natural complexity levels [Im2]. We have written on finite model theory and logic tailored for complexity at length in [Gu2]. Sections 1-9 of this paper constitute a kind of update of [Gu2] though they should be understandable without reading [Gu2] first. We also use this opportunity to make some remarks to different papers of ours which came to our mind after the papers were written.

Section 1 contains provisos and definitions that are used throughout sections 1-9. In particular, the notions of global relations and global functions are introduced; these notions provide convenient semantics for complexity tailored logics. In Sections 2-4 we consider some extensions of first order logic by additional constructs; in the presence of linear order the extended logics capture natural complexity classes. In Section 5 we consider two logics with an emphasis on functions rather than predicates; a linear order is built-in and the logics capture natural complexity classes. In Section 6 we consider some fixed point extensions of first order logic without assuming the presence of linear order. On the one side, the presence of linear order is natural if we want to treat structures as inputs to algorithms. On the other side, one may be interested in properties of structures that are independent of presentation. In Section 7 we consider those properties of structures with linear order that are preserved by any reordering. In Section 8, some evidence is given that certain familiar complexity classes cannot be captured by any logic. Circuit definability and topology on finite sets are briefly discussed in Section 9.

Remark 2. Several relevant issues are left out in this lectures. In particular, we do not discuss derivability in first order predicate calculus. The questions of expressibility and derivability are quite different. For

example, no first order formula φ expresses on finite graphs that (x,y) belongs to the transitive closure of the edge relation E. This is well known [Fa2, GV, Gu2], and remains true even if φ is allowed to use additional predicate symbols: just consider the case when all additional relations are trivial. On the other hand, the first order formula

$$\forall uv(Euv \to Tuv) \;\&\; \forall uvw(Euv \;\&\; Tvw \to Tuw) \to T(x,y)$$

is derivable frome the diagram of an arbitrary finite graph if and only if (x,y) belongs to the transitive closure of the edge relation E.

Another important feature of many computer science structures, which is harder to swallow, is their dynamic character. Mathematical structures – graphs, groups, topological spaces, etc. – do not change in time whereas computer science structures – databases, machines, etc. – often do. Considering time as a new dimension, a mathematician turns a dynamic situation into a static one. Complexity considerations may make such a transformation inadvisable in computer science. We would like to formalize the notion of dynamic structures and develop a logic of (finite) dynamic structure. Among other things it would provide an alternative semantics for usual imperative programming languages. In these lectures we only slightly touch the subject, see §10. First we argue there in favor of a computation theory that deals exclusively with finite machines, and then we discuss formalizing "real" finite machines as dynamic structures.

There are some logicians that consider computing a lower subject. There are former logicians that work in computing now and consider logic not very relevant to their new occupation. We happened to think that computer science badly needs what logicians are supposed to do best: logic. The situation seems to us reminiscent of that in the beginning of the century. Again we face most basic questions like what is a right logic and even what are right structures.

Acknowledgements. I thank the organizer - Dr. Egon Börger - and the host - International Center for Mechanical Sciences - for the invitation to lecture in Udine, and the listeners for their attention, good will and hard work. Special thanks are due to Dr. Klaus Ambos-Spies who faithfully recorded and subsequently prepared a good draft (of the bulk) of these lectures; I admit some editing, altering and reorganizing the material.

§1 Global relations and functions

This section contains provisos and definitions used in sections 1-9. We start with notions of global relations and global functions introduced first in [Gu1]; these notions will allow us to provide semantics for numerous logics.

Definition. Let K be a class of first-order structures of some signature σ. An r-ary K-*global relation* ρ assigns to each structure $S \in K$ an r-ary relation ρ^S on S. The relation ρ^S is the *specialization* of ρ on S. If K is the class of all structures of a signature σ we say that ρ is σ-global.

The notion of global relations generalizes Tarski's notion of sentential functions [Ta]. Sentential functions are global relations of arity zero. An r-ary global relation of signature σ can be viewed as a sentential function whose signature is an extension of σ by r additional individual constant; in this sense the domain of any global relation consists of structures.

Tarski's semantics for first-order logic can be conveniently formulated in terms of global relations (disallow function symbols for a moment). The meaning of a first-order formula φ with free individual variables $v_1, \ldots, v_r$ (in the lexicographical order) is a σ-global r-ary relation where σ is the signature of φ i.e. the set function and predicate symbols in φ (individual constants are function symbols of arity 0). The meaning is defined by an obvious induction.

Examples of global relations.

(1) Let GRAPH be the class of finite graphs (seen as structures with one binary relation that is irreflexive and symmetric). The following GRAPH-global relations are of arities 0, 1 and 2 respectively:

The graph is connected,

A node x has at most $\log n$ neighbors where n is the number of nodes,

Nodes x and y are connected.

(2) Let GROUP be the class of finite groups. The following GROUP-global relations are of arities 0, 1 and 3 respectively:

The group is abelian,

The index of the subgroup, generated by an element x, is at most $\log n$ where n is the number of elements,

The subgroup, generated by elements x and y, contains an element z.

Even though a global relation ρ is, formally speaking, a function defined on a class of structures, we think about it as a relation on the structure of discourse. For example, we may speak about the negation of ρ. If ρ is binary, we may speak about the transitive closure of ρ. We take the same local attitude toward global functions defined below.

Definition. Let K be a class of structures of a fixed signature. A K-*global function* f *of type* $\text{Universe}^r \to \text{Universe}$ assigns to each $S \in K$ an r-ary function f^S that, given an r-tuple of elements of S, produces an element of S.

First-order terms denote global functions. In the obvious way, global relations and global functions of types $\text{Universe}^r \to \text{Universe}$ provide semantics for first-order logic with function symbols.

We will keep the notion of a global function informal (and very general) and will define formally only global functions of specific types. In particular, an r-ary global relation is a global function of type $\text{Universe}^r \to \text{Bool}$ where Bool = {False, True}.

Definition. Let K be a class of structures of a fixed signature. A K-*global function* f *of type* $\text{Universe}^p \to \text{Universe}^q$ assigns to each $S \in K$ a function f^S that, given a p-tuple of elements of S, produces a q-tuple of elements of S. We say that f, as well as each specification f^S of f, is p-ary and q-coary. The notion of a K-global partial function f of type $\text{Universe}^p \to \text{Universe}^q$ is an obvious generalization; f itself is total (defined on the whole K) but its specifications may be partial.

Examples of other types of global function:

(1) $(\text{Unverse}^p \to \text{Bool}) \to (\text{Unverse}^q \to \text{Bool})$,
(2) $[\text{Universe}^p \times (\text{Power-set(Universe)})^q] \to \text{Bool}$.

Global functions of type (2) form a realm where meanings of second order formulas live.

Proviso 1. A structure is -- unless the contrary is said explicitly -- a finite (first order) structure whose universe is an initial segment of natural numbers. This proviso is in force in the rest of this section and in sections 2-9.

Proviso 2. The domain of a global relation comprises all structures of a fixed signature unless the contrary follows from the context.

In order to discuss the decision problem for a global relation, we need to agree on a standard way to represent structures as inputs for computing devices. To simplify the exposition, we chose to represent structures by means of several input tapes. One input tape, called the universe tape, represents the universe $\{0,1,...,n-1\}$ of a given structure S; it is of length n, its end-cells are specially marked but the intermediate cells are all blank. If R is a basic relation of S or the graph of a basic function of S then R is represented by a special tape of length n^r where r is the arity of R; for all elements $x_0, ..., x_{r-1}$, the cell number $\sum x_i \cdot n^i$ contains 1 if $R(x_{r-1},...,x_0)$ is true and contains 0 otherwise.

Definition. Let ρ be an r-ary K-global relation. An instance of the decision problem for ρ is a structure (S,x) where S belongs to K and x is a r-tuple of elements of S; the corresponding question is whether $\rho(x)$ holds in S (in other words, whether x belongs to ρ^S).

Theorem 1 [Fa1]. A global relation is definable by an existential second-order formula if and only if it is recognizable by a polynomial time bounded nondeterministic Turing machine.

Thus, existential second order logic captures nondeterministic polynomial time. We skip the proof of Theorem 1.

It is easy to see that every first-order definable global relation is log-space (and therefore polynomial time) recognizable. The converse is not true. For example, the global relation "The cardinality of the universe is even" is log-space computable but not first order definable. But some natural extensions of first order logic express exactly log-space (respectively polynomial time) recognizable global relations.

Definition. Let L be first order logic or an extension of first order logic by additional constructs. (A number of such extension will be defined in subsequent sections.) $L^{+<}$ is the extension of L by means of a logical constant < (like first order logic with equality is the extension of first order logic by means of a logical constant =) which is interpreted on each structure S as the restriction of the usual order of natural numbers to the universe of S.

§2 Transitive closures

Aho and Ullman [AU] noticed that the relational calculus, a standard relational query language and a variant of first order logic, is not closed under the transitive closure and suggested extending the relational calculus by least fixed points. Immerman [Im2] explored turning the transitive closure itself into a logical construct. The resulting extensions of first order logic are explained in this section. The use of two-way multihead automata allowed us to simplify somewhat the proofs.

In this section, we identify an arbitrary 2r-ary relation R over some $U=\{0,...,n-1\}$ with the binary relation $\{(x,y)\in U^r\times U^r:$ the concatenation of tuples x,y belongs to R$\}$; this defines naturally the *transitive closure* TC(R) of R. The *transitive closure* TC(ρ) of a 2r-ary global relation ρ is defined in the local fashion: for each structure S in the domain of ρ, the specialization of TC(ρ) in S is the transitive closure of ρ^S.

Lemma 1. If a 2r-ary global relation ρ is nondeterministic log-space recognizable then so is TC(ρ).

Proof. Let S be a structure in the domain of ρ, R be the specialization of ρ, and a,b be r-tuples of elements of S. The desired algorithm is:

```
begin
    x:=a;
    repeat
        guess y;
        if (x,y)∈R then x:=y
    until x=b;
    halt with output YES
end. ⊥
```

We define a logic FO+TC. The syntax of FO+TC is the extension of the syntax of first-order logic by:

Transitive Closure Formation Rule. Let r be a positive integer and $\varphi(x,y)$ be a well-formed formula where x and y are r-tuples of distinct individual variables (all 2r variables are distinct). Then $TC_{x,y}\varphi(x,y)$ is a well-formed predicate, and $[TC_{x,y}\varphi(x,y)](x,y)$ is a well-formed formula.

The construct $TC_{x,y}$ binds the 2r individual variables in the new

predicate (but of course the additional occurrences of these variables in the tail of the formula are free). $\varphi(x,y)$ may have additional free individual variables. A more explicit notation for the new predicate is $TC_{x,y}\varphi(w,x,y)$ where w is the list of those additional variables. The new formula $[TC_{x,y}\varphi(w,x,y)](x,y)$ means that (x,y) belongs to the transitive closure of the relation $R_w=\{(x,y): \varphi(w,x,y)\}$. The global function semantics for first-order logic naturally extends to logic FO+TC; again the meaning of a formula with r free individual variables is a global r-ary relation.

Remark. A simplified notation $TC\varphi(x,y)$ for the formula $[TC_{x,y}\varphi(x,y)](x,y)$ is deficient. Try to express $[TC_{x,y}\varphi(x,y)](s,t)$ in the simplified notation. Suppose for simplicity that x and y are single variables and $\varphi(x,y)$ is an atomic formula P(x,y,x). Consider for example s=t=x, or s=fx and t=y.

Positive and negative occurrences of a predicate $P=TC_{x,y}\psi(x,y)$ in a formula φ are defined by the obvious induction; in particular, if φ is $TC_{u,v}\Phi(u,v)$ then every positive (respectively negative) occurrence of P in $\Phi(u,v)$ remains so in φ. The extension FO+TC+< of FO+TC is defined with respect to §1. Viewing 0 an 1 as logical constants yields a further extension FO+TC+<+{0,1}.

Theorem 1. Let ρ be a global relation. The following are equivalent:

(1) ρ is nondeterministic log-space recognizable,

(2) ρ is definable by an FO+TC+< formula φ such that every occurrence of a predicate of the form $TC_{u,v}\psi(u,v)$ in φ is positive,

(3) ρ is definable by a FO+TC+<+{0,1} formula $[TC_{x,y}\psi(x,y)](s,t)$ where ψ is first order.

Proof. (3) → (2). The constants 0 and 1 are definable in FO+<.

(2) → (1). Without loss of generality, one may suppose that only first order subformulas can be negated in the defining formula: use the usual duality laws for first order logic. Then an easy induction shows that every subformula of the defining formula is nondeterministic log-space recognizable. The case of TC is taken care in Lemma 1.

To prove the implication (1) → (3), suppose that ρ is recognizable in nondeterministic log-space. According to the Appendix, there is a nondeterministic 2-way multihead automaton A that recognizes ρ. Let

formula Next(w,x,y) and tuples Initial, Final be as in the Appendix. Then the desired FO+TC+<+{0,1} formula is $[TC_{x,y}Next(w,x,y)](Initial,Final)$. ⊥

Remark. One can easily get rid of the logical constant 1 as follows. Let Next'(x,y) be the disjunction: Next(x,y), or x=Zero & y=Initial, or x=Final & y=Zero. Hero Zero is a tuple of zeroes of the appropriate length. Then the desired formula is $[TC_{x,y}Next(w,x,y)](Zero,Zero)$.

Definition. The *deterministic version* of a binary relation R is the relation $\{(x,y)$: $(x,y)\in R$ and there is no $z\neq y$ with $(x,z)\in R\}$. The *deterministic transitive closure* DTC(R) of R is the transitive closure of the deterministic version of R.

The definition of the deterministic transitive closure extends naturally to the case of a 2r-ary relation R over some $U=\{0,...,n-1\}$ because we have identified R with a binary relation over U^r. The deterministic transitive closure DTC(ρ) of a 2r-ary global relation ρ is again defined in the local fashion: for each structure S in the domain of ρ, the specialization of DTC(ρ) in S is the deterministic transitive closure of ρ^S.

Lemma 2. If a 2r-ary global relation ρ is nondeterministic log-space recognizable then so is DTC(ρ).

Proof. Let S be a structure in the domain of ρ, R be the deterministic version of ρ^S, and a,b be r-tuples of elements of S. R is the graph of a partial function f on $\{0,1,...,n-1\}^r$. Compute $f^k a$ for k=1,2, etc. and halt when b comes along or NOT-DEFINED is returned or k reaches n. If b has come along then return YES, otherwise return NO. ⊥

The definition of an extension FO+DTC of first-order logic is similar to the definition of FO+TC. Just change "TC" to "DTC", and "transitive closure" to "deterministic transitive closure".

Theorem 2. Let ρ be a global relation. The following are equivalent:
(1) ρ is log-space recognizable,
(2) ρ is definable in FO+TC+<,
(3) ρ is definable by a FO+TC+<+{0,1} formula $[TC_{x,y}\psi(x,y)](s,t)$ where ψ is first order.

Proof is similar to that of Theorem 1. ⊥

§3. Least fixed points

In this section we define the extension FP+LFP of first order logic by the least fixed point operator and prove Immerman-Vardi's theorem [Im1, Va] that FO+LFP+< captures polynomial time. Again, the use of two-way multihead automata allowes certain simplification.

Definition. Let F be a unary operation on a partially ordered set. If $Fx=x$ then x is a *fixed point* of F. If $Fx=x$ and $\forall y(Fy=y \rightarrow x \leq y)$ then x is the *least fixed point* LFP(F) of F. If $Fx \leq Fy$ for all $x \leq y$ then F is *monotone*.

Definition. A partial ordered set is *complete* if every subset has the least upper bound and the greatest lower bound.

For example, the set of relations of a fixed arity on a fixed domain is a complete partially ordered set with respect to the inclusion. The following theorem is well-known.

Theorem 1. A monotone unary operation F on a complete partially ordered set D has a least fixed point.

Proof. Let $g0=\min(D)$ and each $g(i+1)=F(gi)$. By monotonicity, the function g is increasing (though not necesssarily strictly increasing). Hence there is i with $gi=g(i+1)$; let m the minimal among such i. Obviously, gm is a fixed point of F. Given a fixed point y of F, prove by induction that each $gi \leq y$. Hence $gm=LFP(F)$. ⊥

Lemma 1. Let F be a σ-global function of type

$$(\text{Power-set(Universe)})^r \rightarrow (\text{Power-set(Universe)})^r$$

which is monotone on any σ-structure. Let LFP(F) be the σ-global r-ary relation that assigns to each σ-structure the least fixed point of F^S. Suppose that F is polynomial time computable i.e. there is a polynomial time algorithm that, given a σ-structure S and an r-ary predicate P on S, computes F(S). Then LFP(F) is polynomial time recognizable.

Proof. Compute $P_0=\varnothing$, $P_1=F(P_0)$, $P_2=F(P_1)$, etc. until you come across $P_m=P_{m+1}$. Now check whether the given r-tuple belongs to P_m. ⊥

The syntax of logic FO+LFP is the result of augmenting the syntax of

first-order logic by:

LFP Formation Rule. Let r be a positive integer, x be an r-tuple $x_1,\ldots,x_r$ of individual variables, P be an r-ary predicate variable, and $\varphi(P,x)$ be a well-formed formula. If $\varphi(P,x)$ is positive in P (i.e. all free occurrences of P in $\varphi(P,x)$ are positive) then $LFP_{P;x}\varphi(P,x)$ is a well-formed predicate and $[LFP_{P;x}\varphi(P,x)](x)$ is a well-formed formula.

The construct $LFP_{P;x}$ binds the predicate variable P and the individual variables $x_1,\ldots,x_r$ (but of course the additional occurrences of these individual variables in the tail of the new formula are free). If Q is a predicate variable different from P then every positive (respectively, negative) occurrence of Q in $\varphi(P,x)$ remains positive (respectively, negative) in the new predicate and formula.

Remark. Substituting an r-tuple t of terms for x in $[LFP_{P;x}\varphi(P,x)](x)$ gives a formula $[LFP_{P;x}\varphi(P,x)](t)$. We suppose that substitution is one of the formation rules.

Remark. A simplified notation $LFP_P\varphi(P,x)$ for $[LFP_{P;x}\varphi(P,x)](x)$ is deficient: just try to express $[LFP_{P;x}\varphi(P,x)](t)$ in the simplified notation.

To be on the safe side, let us emphasize that logic FO+LFP allows interleaving LFP with propositional connectives (including negation) and quantifiers; in particular, one can negate an LFP formula then use the LFP formation rule again, etc.

The formula $\varphi(P,x)$ may have addittional free individual variables; let w be the list of the additional variables. The meaning of the predicate $LFP_{P;x}\varphi(P,x,w)$ is the least fixed point of the operator $F_w(P)=\{x: \varphi(P,x,w)\}$ on the set of r-place relations ordered by inclusion. Since the formula $\varphi(P,x,w)$ is positive in P, the operator F_w is monotone and therefore has a least fixed point. The global function semantics for first-order logic naturally extends to FO+LFP.

Theorem. Let ρ be a global relation. The following are equivalent:
(1) ρ is polynomial time recognizable,
(2) ρ is definable in logic FO+LFP+<,
(3) ρ is definable by a FO+LFP+<+{0,1} formula $[LFP_{P;x}\psi(P,x)](s,t)$

where ψ is first order.

Proof. The implications (3)→(2) and (2)→(1) are obvious. To prove the implication (1)→(3), suppose that ρ is polynomial time recognizable. Accorrding to the Appendix, there is two-way multihead finite automaton A that recognizes ρ. Let the formula Next(w,x,y) and the tuples Initial, Final be as in the Appendix. It is easy to write down first order formulas Existential(x) and Universal(x) asserting that the internal state of M in configuration x is existential and, respectively, universal. Let

$$\begin{aligned}\text{Accepted}(w,\_) = \text{LFP}_{P;x}[\ & x=\text{Final, or} \\ & \text{Universal}(x) \ \&\ \forall y(\text{Next}(w,x,y) \rightarrow P(y)), \text{ or} \\ & \text{Existential}(x) \ \&\ \exists y(\text{Next}(w,x,y) \ \&\ P(y))\ ].\end{aligned}$$

The desired FO+LFP formula is Accepted(w,Initial). ⊥

§4. Branching quantifiers

We turn now to an extension of first-order logic by branching (or Henkin) quantifiers whose introduction was motivated by considerations quite distant from computer science [He].

Let us start with an example. The expression

$$\begin{bmatrix} \forall x_1 \exists y_1 \\ | \quad\quad | \\ \forall x_2 \exists y_2 \end{bmatrix} \varphi(x_1,x_2,y_1,y_2) \qquad (4.1)$$

means that for all x_1,x_2 there are y_1, y_2 such that y_1 depends only on x_1, y_2 depends only on x_2, and $\varphi(x_1,x_2,y_1,y_2)$. In other words, there are functions $Y_1(x_1)$, $Y_2(x_2)$ such that $\varphi(x_1,x_2,Y_1(x_1),Y_2(x_2))$.

In general, a branching quantifier is [Wa] a partially ordered set of expressions $\forall x$ and $\exists y$; an existentially quantified variable y depends exactly on those universally quantified variables x that $\forall x$ precedes $\exists y$ in the partial order.

Theorem 1. For any global relation ρ the following are equivalent:

(1) ρ is NP,

(2) ρ is expressible by an existential second-order formula,

(3) ρ is expressible by a formula $Q\varphi$ where Q is a branching quantifier and φ is a first-order formula.

Proof. The equivalence (1) $\leftrightarrow$ (2) is Theorem 1 in §1, the equivalence (2) $\leftrightarrow$ (3) is proved in [Wa]. $\perp$

In the rest of the section we describe a few results from [BIG1]. The only novelty is the direct proof of Theorem 4 below. A branching quantifier Q is called *mighty* if there is a first-order formula φ such that the global relation $Q\varphi$ is NP-complete under polynomial time reductions.

Theorem 2. The quantifier (4.1) is mighty.

Proof. The idea is to express 3-colorability of a graph with individual constant 0,1 and 2. The desired φ is the conjunction of the formulas:

$x_1 = x_2 \rightarrow y_1 = y_2$,

$y_1 = 0$ or $y_1 = 1$ or $y_1 = 2$,

$\mathrm{Edge}(x_1,x_2) \rightarrow x_1 \neq x_2$. $\perp$

Note that, in the proof of Theorem 2, the existentially quantified variables range, in effect, over $\{0,1,2\}$. Let α, β, γ range over $\{0,1\}$, and μ range over $\{0,1,2\}$.

Theorem 3. The quantifiers

$$\begin{bmatrix} \forall x\, \exists \alpha \\ \forall y\, \exists \beta \\ \forall z\, \exists \gamma \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \forall x\, \exists \alpha \\ \forall y\, \exists \mu \end{bmatrix} \quad \text{are mighty.}$$

We omit the proof of Theorem 3. The branching quantifiers

$$\begin{bmatrix} \forall x_1 \forall x_2 \ldots \forall x_m\, \exists \alpha \\ \forall y_1 \forall y_2 \ldots \forall y_n\, \exists \beta \end{bmatrix}$$

will be called *narrow Henkin quantifiers*. In the rest of this section, x and y are tuples of individual variables, and $\mathrm{NH}(x,\alpha;y,\beta)$ is the corresponding narrow Henkin quantifier. Without loss of generality, x and y always have

the same length: just pad the shorter tuple. Let $ENH(x,\alpha;y,\beta)$ be the equality bound version of $NH(x,\alpha;y,\beta)$. $ENH(x,\alpha;y,\beta)\varphi(x,y,\alpha,\beta)$ means

$$NH(x,\alpha;y,\beta)\,[\,(x=y \rightarrow \alpha=\beta)\ \&\ \varphi(x,y,\alpha,\beta)\,],$$

i.e. there is a unary function f from the universe to {0,1} such that for all x and y, $\varphi(x,y,f(x),f(y))$. Let u, v be single individual variables. An arbitrary $NH(x,\alpha;y,\beta)\varphi(x,y,\alpha,\beta)$ is equivalent to

$$ENH(xu,\alpha;yv,\beta)[(u=0\ \&\ v=1 \rightarrow \varphi(x,y,\alpha,\beta)].$$

Let FO+NH be the extension of first-order logic by narrow Henkin quantifiers. Positive and negative occurrences of a subformula $\psi=NH(x,\alpha;y,\beta)\Psi(x,y,\alpha,\beta)$ in a formula φ are defined by the obvious induction; in particular, any positive (respectively negative) occurrence of ψ in $\Phi(u,v,\gamma,\delta)$ remains so in $NH(u,\gamma;v,\delta)\Phi(u,v,\gamma,\delta)$. Abbreviate "nondeterministic log-space" as "Nlog-space".

Theorem 4. For a global relation ρ the following are equivalent:

(1) ρ is co-Nlog-space recognizable,
(2) ρ is expressible by an FO+NH+< formula with only positive occurrences of branching quantifiers,
(3) ρ is expressible by an FO+NH+< formula $ENH(x,\alpha;y,\beta)\varphi(x,y,\alpha,\beta)$ with a first order φ.

Proof. (1)→(3). Suppose that ρ is co-Nlog-space recognizable and ρ' is the complement of ρ (on each relevant structure). According to the Appendix, there is a two-way multihead nondeterministic finite automaton A that recognizes ρ'. Let formula Next(w,x,y) and tuples Initial and Final be as in the Appendix. The desired formula expresses nonacceptance by A:

$$ENH(x,\alpha;y,\beta)[(x=Initial \rightarrow \alpha=1)\ \&$$
$$(\alpha=1\ \&\ Next(w,x,y) \rightarrow \beta=1)\ \&\ (y=Final \rightarrow \beta=0)].$$

The implication (3) → (2) is trivial.

(2)→(1). Without loss of generality, we may suppose that only first order subformulas of the defining formula can be negated: just use the usual duality laws of first order logic. Then every subformula of the defining formula is co-Nlog-space recognizable. It suffices to prove that if $\varphi(x,y,\alpha,\beta)$ is co-Nlog-space then so is $\psi=ENH(x,\alpha;y,\beta)\varphi(x,y,\alpha,\beta)$. Thus, suppose that M' is a log-space bounded nondeterministic Turing machine

that recognizes the negation $\varphi'(x,y,\alpha,\beta)$ of $\varphi(x,y,\alpha,\beta)$. We have

$$\psi \leftrightarrow \exists f\ \forall x \forall y\ \varphi(x,y,fx,fy) \leftrightarrow \exists f\ \forall x \forall y\ \text{not}\,\varphi'(x,y,fx,fy) \leftrightarrow$$
$$\exists f\ \Pi_{x,y}\ \text{not}\,\Sigma_{\varphi'(x,y,\alpha,\beta)}(fx=\alpha\ \&\ fy=\beta) \leftrightarrow$$
$$\exists f\ \Pi_{x,y}\ \Pi_{\varphi'(x,y,\alpha,\beta)}(fx\neq\alpha \text{ or } fy\neq\beta) \leftrightarrow \exists f\ \Pi_{\varphi'(x,y,\alpha,\beta)}(fx\neq\alpha \text{ or } fy\neq\beta).$$

For every x, view "fx=1" as a propositional variable. Then ψ asserts satisfiability of the propositional formula $\Pi_{\varphi'(x,y,\alpha,\beta)}(fx\neq\alpha \text{ or } fy\neq\beta)$ in variables "fx=1". Recall that a literal is a propositional variable or the negation of such.

Fact [Kr]. A conjunction C of binary disjunctions of literals is unsatisfiable if and only if there are a propositional variable p and a circle $a_1 \to a_2 \to \ldots \to a_m \to a_1$ of literals such that each implication $a_i \to a_{i+1}$ as well as the implication $a_m \to a_1$ is equivalent to a member of C and both p and the negation of p are in the circle.

Now we are ready to describe a log-space bounded nondeterministic Turing machine M that recognizes the negation of ψ. Step-by-step M guesses a circle of literals that witnesses unsatisfiablity of the propositional formula $\Pi_{\varphi'(x,y,\alpha,\beta)}(fx\neq\alpha \text{ or } fy\neq\beta)$. Having guessed an implication $fx=\alpha \to fy\neq\beta$, M uses the given machine M' to establish $\varphi'(x,y,\alpha,\beta)$. $\perp$

§5. Function logics

Restricted to finite structures, primitive recursiveness coincide with log-space computability [Gu1] and recursiveness coincide with polynomial time computability [Guf1, Sa]. These results are discussed in this section. We give a new and simpler recursive schema that together with primitive recursive means suffices to construct all recursive functions.

We start with setting a frame. Recall that the universe of every structure is an initial segment of natural numbers. In this section, we have three additional provisos:

(1) Every structure contains at least two elements. (Alternatively, one may assume existence of an extra universe Bool={False, True}.)

(2) Individual constants 0, End and a unary function symbol Successor are logical constants (like equality is a logical constant in first-order logic with equality). In every structure, 0 denotes itself, End denotes the maximal number in the universe, and Successor denotes the partial function $\lambda x(x+1)$. We will not count the three logical constants as members of any signature.

(3) A certain (possibly empty) signature σ is fixed. Every structure is a σ-structure.

In this section, a (global) function means a partial (σ-global) function of type $\text{Universe}^p \to \text{Universe}^q$ for some nonnegative integer p (the arity) and some positive integer q (the coarity). If $t_1,...,t_k$ are tuples of elements then $(t_1,...,t_k)$ will denote the concatenation of tuples $t_1,...,t_k$ rather than a k-tuple of tuples.

Global functions are viewed as their specializations on the structure S of discourse. Let $U=\{0,...,n-1\}$ be the universe of S. The *value* of a nonempty tuple $(x_{k-1},...,x_1,x_0)$ of elements of U is the number $\sum_{i<k} x_i \cdot n^i$. If one views elements of U as n-ary digits then any nonempty tuple is an n-ary notation for its value.

Definition. The *initial functions* are:

(1) For every nonnegative p, the constant p-ary functions with values 0 or End.

(2) For every positive p, the p-ary p-coary successor function. Given a p-tuple of value $V<n^k-1$, the function produces the tuple of value V+1; it is not defined on the p-tuple of value n^k-1. We will denote the successor of a tuple t as t+1.

(3) For all $p \geq q \geq 1$ and every sequence $1 \leq i_1 \leq i_2 \leq ... \leq i_q \leq p$, the corresponding p-ary q-coary projection function. For example, if p=4, q=2 and $i_1=2$, $i_2=4$ then projection of (0,1,2,3) is (1,3).

(4) The basic σ-functions, and the characteristic functions of basic σ-predicates. (Individual constants are functions of arity 0 and coarity 1.)

The *composition* $g(h_1(x),...,h_k(x))$ of functions is defined in the

obvious way. It is required that all functions h_i have the same arity and $arity(g) = coarity(h_1) + \ldots + coarity(h_k)$.

As usual, *primitive recursion* schema is the schema

$$f(x,Zero) = g(x),\quad f(x,t+1) = h(x,t,f(x,t)) \qquad (5.1)$$

which defines a new function f by means of given functions g and h of the same coarity. Here Zero is the tuple of zeroes of the appropriate length.

Definition. A global function is *primitive recursive* if it belongs to the closure of initial global functions under compositions and primitive recursions. A global relation is *primitive recursive* if so is its characterictic function.

Theorem 1. A global function f is primitive recursive if and only if it is log-space computable.

We skip the proof of Theorem 1.

The calculus of primitive recursive functions can be made closer to programming languages. For example, the construct

$$y:=F_0(u);\ \text{for } s:=F_1(u) \text{ to } F_2(u) \text{ do } y=F_3(s,u,y)$$

produces a primitive recursive function y=G(u) provided the given functions F_i are primitive recursive. On the other hand, the primitive recursion schema (5.1) can be expressed as:

$$y:=g(x);\ \text{for } s:=Zero \text{ to } t-1 \text{ do } y:=h(x,s,y)].$$

Similarly, given primitive recursive functions F and G, the construct

$$v:=F_0(u);\ \text{while } F_1(u,v)=0 \text{ do } v=F_2(u,v)$$

produces a primitive recursive function v=G(u) provided the functions F_i are primitive recursive; and the primitive recursion schema (5.1) can be expressed as:

$$(s,y):=(Zero,g(x));\ \text{while } s<t \text{ do } (s,y):=(s+1,h(x,s,y)).$$

The classical Herbrand-Gödel-Kleene definition of recursive functions can be naturally adapted to the case of our global functions; it turns out that a global function is recursive iff it is polynomial time computable. Moreover, recursive functions form the closure of primitive recursive functions under a single additional recursion schema. Two schemas are specified for this purpose in [Gu1]. Let us introduce a simpler schema:

$$f(x,Zero)=g(x),\quad f(x,t+1)=h(x,f(\alpha x,t),f(\beta x,t)) \qquad (5.2)$$

which defines a new function f by means of given functions g, h, α and β.

Theorem 2. A global function is polynomial time computable if and only if it belongs to the closure of primitive recursive functions by the recursion schema (5.2).

Proof. The "only if" implication is clear. To prove the "if" implication, let RECFUN be the closure of initial primitive recursive functions by composition and recursion schemas (5.1), (5.2), and let RECREL be the class of relations with characteristic functions in RECFUN. It suffices to prove that an arbitrary polynomial time recognizable global relation ρ belongs to RECREL because a polynomial time computable function can be recovered from its graph by primitive recursive means. According to the Appendix, there is an alternating two-way multihead automaton A that accepts a structure (S,w) of the appropriate signature if and only if $\rho(w)$ holds in S. Let tuples Initial and Final be as in the Appendix.

Without loss of generality, every configuration of A has at most two next configurations. There are primitive recursive functions α and β such that if y codes a configuration then $\alpha(w,y)$ and $\beta(w,y)$ code the next configurations; if there is only one next configuration then $\alpha(w,y)=\beta(w,y)$. Without loss of generality, every internal state of A is either existential or universal; the deterministic states (with only one next configuration) can be counted either way. We say that a configuration is existential (respectively universal) if so is the corresponding internal state. There is a primitive recursive function E such that if y codes an existential (respectively universal) configuration then Ey equals 0 (respectively 1). Schema (5.2) allows us to define an auxiliary function Ac(w,y,t):

$$\begin{aligned} Ac(w,y,Zero) &= \text{If } y=Final \text{ then } 1 \text{ else } 0,\\ Ac(w,y,t+1) &= \text{If } Ey=0 \text{ then } \max\{Ac(\alpha(w,y),t), Ac(\beta(w,y),t)\}\\ &\quad \text{else } \min\{Ac(\alpha(w,y),t), Ac(\beta(w,y),t)\}. \end{aligned}$$

But $\rho(w) \leftrightarrow \exists t[Ac(w,Initial,t)=1]$. $\perp$

§6. Inductive fixed points

We discuss an extension FO+IFP of first order logic by means of what we call inductive fixed points. It is seemingly a far more liberal extension than FO+LFP but, in the case of finite structures, its expressive power coincide with that of FO+LFP. The section is based on paper [GS].

In this section, we are interested in logics without the built-in linear order. On the one hand, it is natural to have the linear order around: as inputs for computing devices, structures should be represented in some way. One the other hand, one is interested often in properties of structures that are independent of representation; let us call such properties invariant. One way to ensure invariance of a property is to express it in a logic that does not distinguish between different representations. For example, FO+LFP sentences express only invariant properties.

All FO+LFP expressible properties are polynomial time recognizable. The contrary is not true: "The cardinality of the universe is even" is not expressible in FO+LFP [CH]. However, some polynomial time complete properties are expressible in FO+LFP [Im2]. We doubt that there is a reasonable logic that expresses exactly polynomial time recognizable invariant properties: see the next section. In this connection we are interested in natural extensions of first order logic that express big chunks of polynomial time recognizable invariant properties. In any case, fixed point extensions of first order logic seem to be important enough to be studied for their own sake.

The restriction that the universes of structures are necessarily initial segments of natural numbers may be dropped in this section (but structures are supposed to be finite).

The formulation of LFP formation rule in §3 had one ad hoc feature. In order to ensure that the operator $F(P)=\{x: \varphi(P,x)\}$ is monotone, $\varphi(P,x)$ was supposed to be positive in P. Unfortunately, replacing the positivity condition by the condition that F is monotone, results in an extension FO+LFP' of a first order logic that we would not like to call a logic: the set of FO+LFP'. formulas is undecidable [Gu2]. Fortunately, there is a good fixed point extension of first order logic which is even more liberal than FO+LFP'.

Definition. Let F be a unary operation on a complete partially ordered

set D. Let g0=min(D) and each g(i+1)=F(gi). F is *inductive* if gi$\leq$g(i+1) for every i. F is *inflationary* if X$\leq$F(X) for every XϵD. It is easy to see that if F is inductive then it has a unique fixed point of the form gi; this fixed point will be called the *inductive fixed point* IFP(F) of F.

Lemma 1. Let F be a unary operation on a comlete partially ordered set.

(a) If F is inflationary then it is inductive.

(b) The operation F'(X)=sup{X,F(X)} is inflationary; if F itself is inductive then IFP(F')=IFP(F).

(c) If F is monotone then it is inductive and LFP(F)=IFP(F).

Proof is clear. $\perp$

Examples. Consider the power set of U={0,1,2} ordered by inclusion.

(i) Define FX=X$\cup${the cardinality of X} if X$\neq$U, and FU=U. Then F is inflationary but not monotone. Moreover, F does not have a least fixed point: both {1} and {0,2} are fixed points of f but F$\emptyset\neq\emptyset$.

(ii) Define G=F exept G{1}=$\emptyset$. Then G is inductive but neither inflationary nor monotone.

(iii) The constant operation H(X)={0} is monotone but not inflationary.

Building on an idea of Livchak [Li], we defined a logic FO+IFP in [Gu2]. The syntax of FO+IFP is the extension of the syntax of first order logic by:

IFP Formation Rule. Let r be a positive integer, x be an r-tuple $x_1,\ldots,x_r$ of individual variables, P be an r-ary predicate variable, $\varphi(P,x)$ be a well-formed formula, and $\varphi'(P,x)$=[P(x) or $\varphi(P,x)$]. Then $IFP_{P;x}\varphi'(P,x)$ is a well-formed predicate and $[IFP_{P;x}\varphi'(P,x)](x)$ is a well-formed formula.

The meaning of the predicate $IFP_{P;x}\varphi'(P,x)$ is the inductive fixed point of the inflationary operator F(P)={x: $\varphi'(P,x)$}. The global function semantics for first order logic naturally extends to FO+IFP.

The statement (c) of Lemma 1 implies that FO+IFP is at least as expressive as FO+LFP'.

Theorem 1. The logics FO+LFP and FO+IFP have the same expressive power.

Theorem 1 is a consequence of a stronger theorem. Let Γ be a global function of the empty signature and of type

$$(\text{Power-set}(\text{Universe}))^{\Gamma} \times (\text{Power-set}(\text{Universe}))^{\Gamma} \times (\text{Universe})^{\Gamma} \longrightarrow \text{Bool}.$$

Given (an arbitrary universe and) two r-ary relation P_1, P_2 and an r-tuple x of elements, Γ produces a boolean value $\Gamma(P_1,P_2,x)$. Define an extension FO+Γ of first order logic in the obvious way. More specifically, extend the syntax of first order logic by means of the following formation rule: if x is an r-tuple of individual variables and $\varphi(x)$, $\psi(x)$ are well formed formulas then so is $\Gamma(\{x: \varphi(x)\}, \{x: \psi(x)\}, x)$. The semantics is clear. We suppose that Γ is monotone in both relational arguments (in every universe). In other words, the following second order formula is logically true:

$$[\, P_1 \subseteq P_3 \;\&\; P_2 \subseteq P_4 \,] \longrightarrow [\, \Gamma(P_1,P_2,x) \;\longrightarrow\; \Gamma(P_3,P_4,x) \,].$$

Treat Γ as a positive operator: every positive (respectively negative) occurence of a predicate symbol in $\varphi(x)$ or $\psi(x)$ remains so in $\Gamma(\{x: \varphi(x)\}, \{x: \psi(x)\}, x)$. It is easy to see that if an FO+Γ formula $\psi(Q,y)$ is positive in a predicate symbol Q then the operator $F(Q)=\{y: \psi(Q,y)\}$ is monotone; if this operator is also repetitive (i.e. the length of the sequence y of individual variables equals the arity of Q) then F has a least fixed point.

Theorem 2. There is an FO+Γ formula ψ such that $[\text{IFP}_{P;x}(P(x) \text{ or } \Gamma(P,\neg P,x))](x)$ is equivalent to $[\text{LFP}_{\ldots}\psi](\ldots)$.

To deduce Theorem 1 from Theorem 2, prove that every FO+IFP formula φ is equivalent to (i.e. defines the same global relation as) some FO+LFP formula. The proof proceeds by induction on φ. The only nontrivial case is when $\varphi=[\text{IFP}_{P;x}(P(x) \text{ or } \Phi(P,x))](x)$. Let $\Gamma(P,P',x)$ be the result of replacing all negative occurrences of P in Φ by a new predicate symbol P'. Then Γ is monotone in both relational variables and $\Phi(P,x)$ is equivalent to $\Gamma(P,\neg P,x)$. Now use Theorem 2.

Theorem 3. Every FO+IFP formula is equivalent to an FO+IFP formula φ such that φ is either first order or of the form $[\text{IFP}_{\ldots}\Phi](\ldots)$ where Φ is first order.

Theorems 2 and 3 immediately imply the analog of Theorem 3 for FO+LFP announced in [Im1].

Remark. Theorem 3 can be strengthen further: Φ can be taken to be a boolean combination of existential first order formulas [BöG]. A similar remarks applies to a number of other extensions of first order logic.

§7. Invariant global relations

In the previous section, we mentioned invariant properties of structures. Here we formalize the notion of invariant global relations and make a couple of remarks about invariant global relations.

We drop the assumtion that the universes of structures are initial segments of natural numbers (keeping the assumtion that structures are finite), but restrict interpretations of the binary predicate symbol $<$ to linear orders; structures with (an interpretation of) $<$ will be called ordered and structures without $<$ will be called unordered. This is an expository matter: each ordered structure is naturally isomorphic to a standardly ordered structure on an initial segment of natural numbers.

Let σ range over signatures without $<$. If ordered structures S and T of some signature $\sigma \cup \{<\}$ are equal except for the interpretations of $<$, we view S and T as representations of the same unordered structure S_0 of signature σ; we say that S_0 is the unordered version of S, and S, T are ordered versions of S_0, and T is the result of a reordering of S.

Definition. Let ρ be an r-ary global relation defined on all reorderings of an ordered structure S. Then ρ is *invariant* on S if for every reordering T of S and for every r-tuple x of elements of S, $\rho^S(x) \leftrightarrow \rho^T(x)$.

Definition. An r-ary K-global relation ρ is *abstract* if for every isomorphism f from a K-structure S onto a K-structure T and all elements $x_1, \ldots, x_r$ in S, $\rho^S(x_1, \ldots, x_r) \leftrightarrow \rho^T(fx_1, \ldots, fx_r)$.

Definition. Let K be a class of ordered structures of a fixed signature that is closed under reorderings. A K-global relation is *invariant* if it is (i) abstract, and (ii) invariant on every structure in K.

Corollary. Let K be a class of ordered structures, ρ be an r-ary

K-global relation, S range over K, S_0 be the unordered version of S, and x range over r-tuples of elements of S. Then ρ is invariant if and only if the boolean value of $\rho(x)$ in S depends only on the isomorphism type of (S_0,x).

Examples of invariant global relations are the center of an ordered group (no correlation is assumed between the order and the group structure) and the transitive closure of the edge relation of an ordered graph. Note that the definition and computation of an invariant relation may use the ordering of the universe, only the relation itself must not depend on the order. For example, the following algorithm computes the center of an ordered group:

```
C:=∅;
for x:=(the first element) to (the last element) do
begin
    flag:=1;
    for y:=(the first element) to (the last element) do
        if x·y≠y·x then flag:=0;
    if flag=1 then C:=C∪{x}
end.
```

Theorem 1. The decision problem whether a given first order sentence (that uses symbol <) yields an invariant global relation, is undecidable.

Proof. Let α range over first order sentences without symbol <. The validity of α on all finite structures is undecidable [Tr], hence the validity of α on all finite structures with at least two elements is undecidable. Let P be a unary predicate symbol that does not occur in α, and let β be a first order sentence of signature {P,<} asserting that < is an order and the first element belongs to P whereas the last element does not. It is easy to see that α is valid on all finite structures with at least two elements if and only if the disjunction (α or β) is invariant. $\perp$

Theorem 2. There is a first-order sentence φ such that the decision problem whether φ is invariant on a given ordered structure, is co-NP complete.

Proof. We reduce the 3-colorability problem, which is a known NP complete problem [GJ], to the complement of our problem. The 3-colorability problem remains NP complete if one considers only graphs G=(V,E) such that V={0,...,n-1} for some n>3 and E≠∅; order any such graph G in the standard way and call the result $G^<$.

The desired φ speaks about ordered graphs. It asserts that there are vertices $x<y$ such that the segments $\{v: v<x\}$, $\{v: x\leq v<y\}$ and $\{v: y\leq v\}$ constitute a 3-coloring of the graph. If a graph G is not 3-colorable then φ fails on any reordering of $G^<$ and therefore is invariant on $G^<$. If G is 3-colorable then φ holds on some reordering of $G^<$ and fails on another; in this case φ is not invariant on $G^<$. $\perp$

We conjecture that no logic (under some reasonable restrictions on the notion of logics, see the next section in this connection) expresses exactly invariant polynomial time recognizable global relations.

§8. Is there a logic for NP∩coNP or R ?

We give some evidence that no logic expresses exactly NP∩coNP global relations or exactly R (random polynomial time recognizable) global relations. The argument is an elaboration of a remark in [Gu1] and uses Sipser's result [Si1] that each of the two classes fails to have a complete problem (with respect to polynomial time reductions) under an appropriate oracle.

We suppose that every logic L determines a set of L-sentences and a satisfaction relation $\models_L$. A signature is associated with each L-sentence, and there is a Turing machine that, given a signature σ, generates all L-sentences of signature σ. The satisfaction relation $\models_L$ is a set of pairs (S,φ) where φ is an L-sentence, S is a structure and the signatures of φ, S coincide. An L-sentence φ *defines* the class $Mod(\varphi)=\{S: S\models_L\varphi\}$ of *models* of φ.

First we consider class NP∩coNP. Nondeterministic Turing machines M, N and a polynomial f will be said to *witness* that a class K of structures of some signature σ is NP∩coNP if for every n and every σ-structure S of cardinality n, (i) $S\in K$ iff M accepts S within time f(n), and (ii) S does not belong to K iff N accepts S within time f(n).

Definition 1. A logic L *captures* NP∩coNP if:

(i) For each L-sentence φ, the class $Mod(\varphi)$ is NP∩coNP; moreover, there is a Turing machine that, given an L-sentence φ, produces a triple (M, N, f) witnessing that $Mod(\varphi)$ is NP∩coNP; and

(ii) Every NP∩coNP class of structures of a fixed signature is

definable by an L-sentence.

Theorem 1. If a logic L captures NP∩coNP then NP∩coNP has a complete problem with respect to polynomial time reducibility.

Proof. Let σ be a signature comprising one unary predicate symbol. Fix a Turing machine A that generates all L-sentences of signature σ, and a Turing machine B that, given an L-sentence φ of signature σ, generates a triple witnessing that $Mod(\varphi)$ is NP∩coNP. Let Q be the set of tuples $(\alpha, \varphi, \beta, M, N, f, S, 1^{f(n)})$ such that (i) α is a computation of A, φ is an L-sentence generated by α, β is the computation of B on φ, (M, N, f) is the output of β, S is a σ-structure, n is the cardinality of S, $1^{f(n)}$ is a string of 1's of length n, and (ii) $S \models_L \varphi$.

The condition (i) is polynomial time checkable. The condition (ii) is NP (respectively coNP): guess a computation of M (respectively N) on S of length f(n) and verify that the computation is accepting. Thus the decision problem for Q is NP∩coNP. To show that this decision problem is NP∩coNP hard, we reduce to Q the decision problem for an arbitrary NP∩coNP class X of binary words. If w is a binary word $\alpha_1 \ldots \alpha_n$ let S_w be the σ-structure with universe $\{0, 1, \ldots, n\}$ and relation $\{i: \alpha_i = 1\}$. Since L captures NP∩coNP, there is an L-sentence φ with $Mod(\varphi) = \{S_w: w \in X\}$. Let α be a computation of A that outputs φ, β be the computation of B on φ, and (M, N, f) be the output of β. Obviously, $w \in X$ iff $S_w \in Mod(\varphi)$ iff $(\alpha, \varphi, \beta, M, N, f, S_w, 1^{f(n+1)})$ belongs to Q. ⊥

Theorem 1 contrasts with Sipser's result [Si1] that, relative to some oracle Δ, NP∩coNP does not posses a complete problem. (Certainly no logic expresses exactly the global relations that are NP∩coNP with respect to Δ because the proof of Theorem 1 relativizes.)

Conjecture. If there is a logic that captures NP∩coNP then something trastic happens like NP∩coNP=P or NP=coNP. (If necessary, restrict further the notion of a logic capturing NP∩coNP.)

One way to restrict the notion of a logic L capturing NP∩coNP, is to request that L-sentences are polynomial time recognizable.

Remark. The converse of Theorem 1 is true to the extent that, given an NP∩coNP complete problem Q, one can construct a set of "sentences" and a satisfaction relation that capture NP∩coNP. Define sentences of

signature σ as triples (M, f, σ) where M is a deteministic Turing machine and f is a polynomial. Say that a σ-structure S of cardinality n satisfies $\varphi = (M, f, \sigma)$ if M halts on input S within time f(n), and the result M(S) belongs to Q. (One can speak also about "formulas" $(M, f, \sigma, v_1, \ldots, v_r)$ with free individual variables $v_1, \ldots, v_r$; treat the variables as additional individual constant.)

Definition 1 and Theorem 1 generalize to other classes with well defined witnesses. We turn now to random polynomial time. Recall that a set K of strings in an alphabet Σ is R if and only if there are a deterministic Turing machine M and polynomials f,g such that for every n and every string $s \in \Sigma^*$ of length n the following are equivalent:

(i) There is a string t in $\{0,1\}^{g(n)}$ such that M accepts the pair (s,t) within time f(n), and

(ii) For at least one half of strings t in $\{0,1\}^{g(n)}$, M accepts the pair (s,t) within time f(n).

We say that (M,f,g) *witnesses* that K belongs to R. Without loss of generality, we may suppose that $fn \geq gn$ for all n. The definition obviously generalizes to the case when K is a class of structures of a fixed signature.

Definition 2. A logic L *captures* R if:

(i) For each L-sentence φ, $Mod(\varphi)$ is R; moreover, there is a Turing machine that, given an L-sentence φ and its signature, produces a triple (M,f,g) witnessing that $\{S: S \text{ satisfies } \varphi\}$ is R; and

(ii) Every R class of structures of a fixed signature is definable by an L-sentence.

Theorem 2. If a logic L captures R then R has a complete problem with respect to polynomial time reducibility.

Proof is similar to that of Theorem 1. $\perp$

Theorem 2 contrasts with Sipser's result [Si1] that, relative to some oracle, there is no complete problem for R with respect to polynomial time reducibility.

§9. Miscellany

9.1. A logic for log-space constructable sequences of circuits of bounded depth.

We suppose here that every signature comprises only predicate symbols, and every boolean circuit has a unique output gate. Recall that the universe of a structure of cardinality n is the segment $\{0,1,...,n-1\}$ of natural numbers.

Definition. Let σ be a signature, $U=\{0,1,...,n-1\}$, and A be the set of sentences $Q(i_1,...,i_r)$ where $Q\in\sigma$, r is the arity of Q and every $i_p\in U$. A boolean circuit C is *formatted* to σ and n if all input gates of C are labeled by elements of A. (C may have less than $|A|$ input gates.)

Definition. A circuit C, formatted to σ and n, *accepts* a σ-structure S of cardinality n if C outputs 1 when the inputs gates of C are set with respect to S (an input gate labeled $Q(i_1,...,i_r)$ gets value 1 if S satisfies $Q(i_1,...,i_r)$ and value 0 otherwise).

Definition. A a class K of σ-structures is *definable* by a sequence of circuits $C_1,C_2,...$ if every C_n can be formatted to σ and n in such a way that the formatted circuit accepts a σ-structures S of cardinality n if and only if $S\in K$.

Given a signature σ, a first-order σ-sentence φ and a natural number n, it is easy to construct a circut C_n formatted to σ and n in such a way that the depth of C_n is the logical depth of φ, and C_n accepts a σ-structure S of cardinality n iff S satisfies φ. Let σ_n be the extension of σ by individual constants 0, 1, ..., n-1. By induction on the logical depth, turn any sentence α of signature σ_n into a formatted circuit α_n. If α is atomic then α_n is the circuit comprising one gate labeled α. The cases of conjunction, disjunction and negation are obvious. If α is $\exists x\beta(x)$ (respectively $\forall x\beta(x)$) then join the circuits $\beta(0)_n$, $\beta(1)_n$, ..., $\beta(n-1)_n$ by an additional OR (respectively AND) gate. φ_n is the desired C_n.

The sequence of circuits, constructed in the previous paragraph, is very uniform. In particular, it is log-space constructable i.e. there is a log-space bounded Turing machine that, given the unary notation for n,

produces (the standard code for) C_n.

Let L_0 be a logic that captures exactly log-space recognizable global predicates of the empty vocabulary. L_0 can be the fragment of logic FO+DTC+< (see §3) whose formulas contain no individual constants, no function symbols and no predicate symbols except for <. L_0 can be the calculus of primitive recursive functions of the empty vocabulary (see §6); in this case the formulas are equations t=0. Let $FO+L_0$ be the extension of first-order logic by L_0. The formulas of $FO+L_0$ are built from first-order formulas and formulas in L_0 by first-order means (boolean connectives and quantifiers $\forall,\exists$); the semantics is obvious.

Theorem [GL]. Let K be a class of structures of some signature σ. The following are equivalent:

(i) K is definable by a log-space constructable sequence of circuits of bounded depth,

(ii) K is definable by a sentence in $FO+L_0$.

We skip the proof here. The theorem generalizes for many other complexity classes [GL].

9.2. A note on topology on finite sets.

There is a definite analogy between classes of unary global relations definable by sequences of bounded-depth polynomially-bounded-size circuits on one side and Borel subsets of the Cantor discontinuum on the other side. This analogy was exploited by Sipser in [Si2]. Reading Ajtai's paper [Aj], we found it useful to think in terms of Borel subsets of finite topological spaces. The definition of Borel subsets of finite topological spaces is given in this subsection.

Recall that a topology is T_1 [Ku] if all one-point subsets are closed; we are not interested here in topologies that are not T_1. Every finite T_1 topological space is discrete i.e. every subset is both closed and open. Thus the theory of finite T_1 topological spaces seems to be quite trivial. However, one may ask how many intersections and unions does it take to express a given point set in terms of sub-basic open sets. This leads to a generalization of the Borel hierarchy to finite topological spaces.

Definition. X_n is the topological space whose points are subsets of $\{0,1,...,n-1\}$ and whose sub-basis comprises the n point sets $\{P: i \epsilon P\}$.

The analogous definition for ω instead of $\{0,1,...,n-1\}$ would result in a topological space X_ω homeomorphic to the Cantor Discontinuum [Ku, §3, IX]. Borel subsets of X_ω form the closure of the sub-basis under complements, countable intersections and countable unions. This suggests the following:

Definition. A subset of X_n is Borel of level 0 if it is sub-basic. It is Borel of level $d>0$ if it is the intersection of at most n Borel sets of levels less than d or the union of at most n Borel sets of levels less than d or the complement of a Borel set of a level less than d.

There is an obvious connection between Borel point sets and boolean circuits with a unique output gate. Suppose that C is a circuit with n input gates labeled by integers $0,...,n-1$. In the obvious way, the input of C represents a point in Top(U). C is said to *accept* a point P if the corresponding output is 1. C is said to *recognize* the set $\{P:$ C accepts $P\}$.

Claim 1. Let F be a family of subsets of X_n, and d be a natural number. The following are equivalent:

(i) F is Borel of level d,

(ii) There is a circuit C with n input gates labeled by integers $0,...,n-1$ such that the depth of C is at most d, the fan-in of C-gates is bounded by n and C recognizes X.

Proof is clear. ⊥

Definition. A *global point* set π assigns a subset of X_n to each X_n. If there is a natural number d such that the specification of π on each X_n is Borel of level d then π is *Borel* (of level d).

Claim 2. The following are equivalent:

(i) π is Borel,

(ii) There is a bounded-depth polynomially-bounded-size sequence of circuits C_n such that each C_n recognizes the specialization of π on X_n.

Proof is clear.

Let P be a unary predicate symbol. A first order sentence $\varphi(P)$ in signature {P} defines a Borel global point set $\{P: \varphi(P)\}$ of level d where d is the logical depth of $\varphi(P)$. One obvious property of a first-order definable point set π is a symmetry: if P_1 and P_2 are subsets of X_n of the same cardinality then P_1 belongs to is the specialization of π on X_n if and only if P_2 does. There are symmetric Borel global point sets that are not first order definable [DGS, FKPS].

§10. Turing's thesis and dynamic structures

We argue in favor of a computation model based on finite computing devices (whereas classical computation model is based on potentially infinite computing devices like Turing machines). In order to formalize finite machines, a special class of dynamic structures is introduced. The section is based on [Gu4] and a forthcoming [BIG2].

Turing's thesis asserts that every algorithm can be simulated by an appropriate Turing machine. An implicit part of the thesis is that Turing machines are justifiable idealizations of real computing devices. But are they? The problem is that Turing tapes are potentially infinite; in that sense Turing machines are potentially infinite.

The answer depends on what reality one has in mind. To prove informally his thesis, Turing analized a routine computation of a human computer. He idealized a pile of paper sheets as a work tape. It is natural to suppose that more paper is available than a human computer can possibly use. In this situation potential infinity of the tape is quite justifiable. One can think about other situations where the idealization of potentially infinite machines is justifiable.

On the other hand, the resources of nonhuman computers are often explicitly bounded and the computer may easily run out of some resource (my Macintosh runs out of memory all the time). In such situations the idealization of computers as potentially infinite machines is not justifiable. (Imagine you buy a Turing machine and use it. After a while you may run out of tape. The table of your Turing machine does not tell you how and where to get an extension. Of course, the manual may contain some useful hints but this is not precisely an algorithm.)

We restrict our attention to digital (rather than analog) finite

computing devices that work in discrete time. The finiteness means here that the hardware is essentially fixed. Changing a bolt is not important but extending the primary memory (or adding a new processor to a multiprocessor machine) results in a new machine.

Now let hear our imaginary opponent.

Objection 1. A computer with a fixed hardwere is just a finite automaton.

Answer. This is true, but the classical theory of finite automata is not very relevant unless one deals with very small devices. Even in the case of a personal computer, the total number of different states is overwhelming. It is not feasible to describe the behavior of a real computer by a state transformation table or a regular expression. One has to take into account the internal structure of computers, and we are intending to do that.

Objection 2. A finite machine A is just a cut-off part of a potentially infinite machine M. For example, a Turing machine with a bounded tape is just a cut-off of a Turing machine with an unbounded tape. Accordingly, each computation of A is just an initial segment of a computations of M. The only new information, gained by studying finite machines, is knowledge of when they break down.

Answer. Some finite machines are just truncations of appropriate infinite machines, but others are not. One distinguishing feature of finite machines is that they may know their resources and use this knowledge. Think about MAXINT of a virtual Pascal machine: what does it correspond to in the case of an infinite machine? For another example, in which the use of resource bounds is crucial, consider operating systems.

Objection 3. Potentially infinite machines are needed to define the operational meaning of a program (say, a Pascal program). For example, every finite machine fails to compute the factorial of a sufficiently large natural number; so how can a finite machine give operational meaning to an algorithm for computing factorial?

Answer. The operational meaning of a program is given by a family of finite machines. Think for example about a family of virtual finite Pascal machines with different values of MAXINT. For another example, consider different implementations of UNIX.

More extensive answers to the objections can be found in [Gu4]. The question of adapting Turing's thesis to finite machines is discussed there too. Dynamic structures and families of dynamic structures seem to be the right terms for formulating a new thesis (new theses). Here we do not formulate new theses. We only define two very special classes SEQ and PAR of dynamic structures and give a couple of examples.

In both cases, static structures (and in particular the static parts of dynamic structures) are finite many-sorted first-order structures. Thus, a static structure consists of a finite number of nonempty finite domains (sorts, basic types) and a finite many basic relations and functions (of specified arities) between these domains. For technical convenience we allow functions of coarity greater than 1. The values of a function of coarity r are r-tuples of elements; such a function could be always replaced by r functions, its components, of coarity 1. The notion of first order terms is easily generalizable to the case of vector terms. The different basic types of a static structure are supposed to be disjoint and the basic functions are supposed to be typed: each argument place and each value place is assigned a particular basic type; as a result, vector terms are typed. Without loss of generality, there are no basic relations (only basic functions): just view relations as functions with values of type Bool={True, False}.

A dynamic structure S of class SEQ evolves in discrete time. The configurations (instantaneous states) of S are static structures of the same signature as the static part of S; the initial configuration of S is the static part of S. The evolution is governed by (a finite many of) transition rules. There are two kinds of transition rules. A *sequential internal* transition rule has the form

$$\text{If } s_1=s'_1 \text{ and } \ldots \text{ and } s_m=s'_m \text{ then } f(t):=u \qquad (10.1)$$

where the vector terms s_i, s'_i, t, u are all variable free. The rule updates the value of the basic function f at point t provided the conditions $s_i=s'_i$ are satisfied. A *sequential external* transiton rule has the form

$$\text{If } s_1=s'_1 \text{ and } \ldots \text{ and } s_m=s'_m \text{ then } f(t):=\text{INPUT} \qquad (10.2)$$

where the vector terms s_i, s'_i, t are variable free. It updates the value of the basic function f at point t provided the conditions $s_i=s'_i$ are satisfied. However, the new value comes from outside. It is a character in the input

alphabet which is one of the basic types.

A basic function f of S is *static* if S does not have any transition rule for updating f, otherwise f is *dynamic*. The two boolean values and the propositional connectives are typical static functions (the boolean values are 0-ary functions).

Consistent dynamic structure in SEQ are idealized sequential computing devices. It is argued in [BIG2] that SEQ dynamic structures satisfactory formalize arbitrary "real" sequential computers.

Example 1: A formalization of a Turing machine with one (inextensible) tape of length n as a dynamic structure S. For the sake of definiteness, we suppose that if the head of the machine is located in the leftmost (respectively rightmost) cell and is instructed to move left (respectively right) then its position remains unchanged. The static part of the desired dynamic structure has basic types Alphabet, Control, and Tape. Here Tape is the initial segment $\{0, 1, ..., n-1\}$ of natural numbers. Each element of type Alphabet or Control is a static basic function. The other static basic functions are 0, End whose values are the numbers 0, n-1 respectively, and Succesor, Predecessor of type Tape → Tape. Successor(i)=i+1 if i≠End, and Successor(End)=End; similarly Predecessor(i)=i-1 if i≠0, and Predecessor(0)=0. The dynamic basic functions are: Head of type Tape, State of type Control, and Content of type Tape → Alphabet. The transition rules reflect instructions of the machine. Every instruction yields three transition rules. For example, an instruction pa → qb(-1) yields:

If State=p and Content(Head)=a then State:=q,
If State=p and Content(Head)=a then Content(Head):=b,
If State=p and Content(Head)=a then Head:=Predecessor(Head). ⊥

Example 2: A partial formalization of a Von Neumann computer as a dynamic structures in SEQ. We suppose that machine words contain 32 bits; the first 8 bits may specify an operation code, the remaining 24 bits then constitute an address. Let Fetch, Store and Jump be operation codes for the respective operations. There is only one basic type Bit={0,1}. The two elements 0 and 1 of type Bit are static basic function. The other static basic functions include projections $x|0..7$ and $x|8..31$ of types $Bit^{32} \to Bit^{8}$ and $Bit^{32} \to Bit^{24}$ (with obvious meaning). The dynamic basic functions include PC (the program counter) of type Bit^{24}, Ac (the accumulator) of type Bit^{24}, and Me (the memory) of type $Bit^{24} \to Bit^{32}$. The transition rules include:

If Me(PC)|0..7 = Fetch then Ac:= Me(Me(PC)|8..31),
If Me(PC)|0..7 = Store then Me(Me(PC)|8..31):= Ac,
If Me(PC)|0..7 = Jimp then PC:= Me(PC)|8..31. ⊥

The definition of class PAR is similar to that of class SEQ but free variables are allowed to appear in 10.1 and 10.2. The new rules are called *parallel internal* and *parallel external* respectively. Consistent dynamic structure in SEQ are idealized parallel computing devices. It is argued in [BIG2] that PAR dynamic structures satisfactory formalize arbitrary "real" parallel computers.

We attempt to formalize families of real and virtual computing devices and to develop a logic of finite dynamic structures which is supposed to provide an alternative semantics for imperative programming languages.

Appendix. Two-way multihead automata

To accomodate standard representations of structures (see §1), our computing devices have in general several input tapes. One of these input tapes is the universe tape that contains the unary notation for the cardinality of the structure. We will ignore structures of cardinality 1 and will suppose that the end-cells of the universe tape are specially marked.

A two-way multihead automaton is as a Turing machine without any work tape. It seems to be a long known folklore that the recognition power of two-way multihead automata equals that of log-space bounded Turing machines.

Theorem 1. A global relation is recognizable in log-space by a deterministic (respectively nondeterministic, alternating) Turing machine if and only if it is recognizable by a deterministic (respectively nondeterministic, alternating) two-way multihead finite automaton.

Proof. The "if" implication is easy (and will not be used): record the current positions of the automaton heads on a work tape.

To prove the "only if" implication, suppose that a log-space bounded Turing machine M recognizes the global relation in question. Let n be the length of the universe tape. Without loss of generality, we can assume the following about M: there is only one work tape, on each step the work tape

head either writes or moves but not both, the work tape alphabet is {0,1} where 0 is also the blank, and the end cells of the work tape never hold zeroes.

Let u be the content of the initial segment of the current work tape up to and including the position of the head, v* be the content of the corresponding final segment, and v be the reverse of v*. The strings u and v are binary notations for some numbers that uniquely define the content of work tape. The symbol observed by the work tape head is exactly the parity of u (0 if u is even and 1 otherwise). If the work tape head changes 0 to 1 (respectively 1 to 0) then $u:=u+1$ (respectively $u:=u-1$) and v does not change. If the work tape head moves to the right then $u:=2u+\delta$ and $v:=(v-\delta)/2$ where δ is the parity of v. If the work tape head moves to the left then $u:=(u-\delta)/2$ and $v:=2v$ where δ is the parity of u.

Since the length of the work tape is bounded by a multiple of log n, the numbers u and v are bounded by some n^k. Thus,

$$u = \sum_{i<k} \alpha_i n^i \quad \text{and} \quad v = \sum_{i<k} \beta_i n^i$$

where $\alpha_i, \beta_i < n$ for each i. The desired two-way multihead automaton A represents u and v by 2k heads on the universe tape. Using a few auxiliary heads, A is able to compute the parities of u,v and to perform the operations $u:=u\pm1$, $u:=2u+\text{parity}(v)$, $u:=[u-\text{parity}(u)]/2$, and the same operations with v. (Concerning the operations $u:=u-1$ and $v:=v-1$ recall that the end cells of the work tape never hold zeroes and therefore u and v are always positive.)

Some internal states of A code the internal states of M, in addition A has some auxiliary internal states. When A is the internal state p coding an internal state q of M, the configuration of A codes a configuration of M; if q is existential (respectively universal) then so is p. Every auxiliary internal state of A is deterministic. It is easy to see that A faithfully simulates M and that A accept a given structure iff M accept it. ⊥

Corollary. A global relation is polynomial time recognizable if and only if it is recognizable by an alternating two-way multihead finite automaton.

Proof. Polynomial time equals alternating log-space [CKS]. ⊥

Suppose that a two-way multihead automaton A recognizes a global

relation ρ. Represent the position of a head h on a tape of length n^p by a p-tuple $x_{h0}, \ldots, x_{h(p-1)}$ with the intended interpretation $\sum x_{hi} \cdot n^i$. Here n is the length of the universe tape and each x_{hi} is a natural number $<n$. Further, represent the j-th internal state of the automaton by a q-tuple $y_0, \ldots, y_{q-1}$ where q is the number of internal states, $y_j=1$ and $y_i=0$ for $i \neq j$. Thus there is an r such that every configuration of A is represented by an r-tuple of natural numbers $<n$. Without loss of generality, we may assume that A has a unique accepting configuration and that both in the initial and in the accepting configuration of A all heads are in the leftmost positions. Then the r-tuples Initial and Final, representing the initial and the final configurations respectively, consist of zeroes and ones.

Claim. There is an FO+< formula Next satisfying the following. Let S be a structure in the domain of ρ, w be a k-tuple of elements of S where k is the arity of ρ, and x, y be r-tuples of elements of the unverse of S. Then Next(w,x,y) holds in S if and only if x,y represent configurations of A on input (S,w) and A is able to go from configuration x to configuration y in one step.

Proof. The desired formula Next is a conjunction where each conjunct describes (in the obvious way) one instruction of A. (The variables w appear since there are reading heads on the corresponding input tapes.) ⊥

Remark. The formula Next is especially simple if one uses the successor function (rather than order) and individual constants 0 and End. If the universe is $\{0, \ldots, n-1\}$ then End is interpreted as n-1. To make the successor function total, define Successor(End)=0 or Successor(End)=End.

In the rest of Appendix, a global function is a partial σ-global funcion of type $\text{Universe}^p \to \text{Universe}^q$ for some σ,p,q; such global function assigns to each σ-structure S a p-ary q-coary operation on the universe of S.

Theorem 3. A global function is computable by a deterministic log-space bounded Turing machine if and only if it is computable by a deterministic two-way multihead automaton.

Proof is essentially the same proof as that of Theorem 1. If the simulated device outputs a character in a configuration x then the simulating device outputs the same character in the configuration that represents x. ⊥

References

Aj M. Ajtai, "Σ^1_1-formulae on finite structures", Annals of Pure and Applied Logic 24 (1983), 1-48.

AG M. Ajtai and Y. Gurevich, "Monotone versus positive", Research Report RJ 4697, IBM, San Jose, May 1985, To appear. (AMS Abstracts, 1985).

AU A. V. Aho and J. D. Ullman, "Universality of data retrieval languages", 6th Symp. on Principles of Programming Languages, Association for Computing Machinery, 1979, 110-117.

BGK A. Blass, Y. Gurevich and D. C. Kozen, "A zero-one law for logic with a fixed-point operator", Technical report CRL-TR-38-84, University of Michigan, Sep 1984. Information and Control, to appear.

BlG1 A. Blass and Y. Gurevich, "Henkin quantifiers and complete problems", Technical report CRL-TR-39-84, University of Michigan, Sep 1984, To appear in Annals of Pure and Applied Logic.

BlG2 A. Blass and Y. Gurevich, "A new thesis", In preparation. (AMS Abstracts, August 1985, p. 317.)

BÖG E. Börger and Y. Gurevich, "Hierarchy theorems for extensions of first order logic" (a tentative title), in preparation.

CKS A. K. Chandra, D. C. Kozen and L. J. Stockmeyer, "Alternation", J. of Association for Computing Machinery 28 (1981), 114-133.

Ch A. Church, "An unsolvable problem of elementary number theory", American Journal of Mathematics 58 (1936), 345-363.

Co E. F. Codd, "Relational completeness of database sublanguages", in "Database systems" (ed. R. Rustin), Prentice-Hall, 1972, 65-98.

DGS L. Dennenberg, Y. Gurevich and S. Shelah, "Cardinalities defined by constant depth polynomial size circuits", Information and Control, to appear.

Eh A. Ehrenfeucht, "An application of games to the completeness

problem for formalized theories",
Fundamenta Mathematica 49 (1961), 129-141.

Fa1 R. Fagin, "Generalized first-order spectra and polynomial time recognizable sets", in "Complexity of computation" (ed. R. Karp), SIAM-AMS Proc. 7 (1974), 43-73.

Fa2 R. Fagin, "Monadic generalized spectra", Zeitschrift für Math. Logik und Grundlagen der Mathematik 21 (1975), 89-96.

FKPA R. Fagin, M. Klawe, N. J. Pippenger, and L. Stockmeyer, "Bounded depth polynomial size circuits for symmetric functions", Theoretical Computer Science, April 1985, 239-250.

Ga R. Gandy, "Church's thesis and principles for mechanisms", in: "The Kleene Symposium" (ed. J. Barwise et al.), North-Holland, 1980, 123-148.

GV H. Gaifman and M. Vardi, "A simple proof that connectivity of finite graphs is not first-order definable", Bulletin of European Assoc. for Theoretical Computer Science, June 1985, 43-45.

GJ M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to the theory of NP completeness", Freeman, 1979.

Gu1 Y. Gurevich, "Algebras of feasible functions", 24h Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1983, 210-214.

Gu2 Y. Gurevich, "Toward logic tailored for computational complexity", in "Computataion and Proof Theory" (Ed. M. M. Richter et al.), Springer Lecture Notes in Math. 1104 (1984), 175-216.

Gu3 Y. Gurevich, "Monadic second-order theories", in "Model-theoretical logics" (ed. J. Barwise and S. Feferman), Springer-Verlag, 1985.

Gu4 Y. Gurevich, "Reconsidering Turing's thesis (toward more realistic semantics of programs)", Technical report CRL-TR-36-84, University of Michigan, Sep 1984, Journal of Symbolic Logic, to appear.

GL Y. Gurevich and H. R. Lewis, " A logic for constant-depth circuits",

Information and Control 61 (1984), 65-74.

GS Y. Gurevich and S. Shelah, "Fixed-point extensions of first order logic", 26th Annual Symposium on Foundation of Computer Science, IEEE Computer Society Press, to appear.

He L. Henkin, "Some remarks on infinitely long formulas", in "Infinitistic Methods, Warsaw, 1961, 167-183.

Im1 N. Immerman, "Relational queries computable in polynomial time", 14th Symposium on Theory of Computing, Association for Computing Machinery, 1982, 147-152.

Im2 N. Immerman, "Languages which capture complexity classes", 15th Symposium on Theory of Computing, Association for Computing Machinery, 1983, 347-354.

Kl S. C. Kleene, "Introduction to metamathematics", D. Van Nostrand, New York, Toronto, 1952.

Kr M. R. Krom, "The decision problem for a class of first-order formulas in which all disjunctions are binary", Zeitschrift für math. Logik und Grundlagen der Mathematik 13 (1967), 15-20.

Ku K. Kuratowski, "Topology", volume 1, Academic Press, 1966.

Li A. Livchak, "The relational model for process control", Automatic Documentation and Mathematical Linguistics 4 (1983), 27-29.

Sa V. Y. Sazonov, "Polynomial computability and recursivity in finite domains", Elektronische Informationverarbeitung und Kybernetik 16 (1980), 319-323.

Si1 M. Sipser, "On relativization and the existence of complete sets", ICALP 1982, 523-531.

Si2 M. Sipser, "Borel sets and circuit complexity", 15th ACM Symposium on Theory of Computing (1983), 61-69.

Ta A. Tarski, "Some notions and methods on the borderline of algebra and metamathematics", Proceedings of 1950 International Congress of Mathematicians in Cambridge, MA, American Mathematical Society, Rhode Island, 1952, 705-720.

Tr B. A. Trakhtenbrot, "Impossibility of an algorithm for the decision problem on finite classes", Doklady 70 (1950), 569-572.

Tu A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem", Proc. of London Mathematical Society 2, no. 42 (1936), 230-236, and no. 43 (1936), 544-546.

Ul J. D. Ullman, "Principles of database systems", Computer Science Press, 1982.

Va M. Vardi, "Complexity of relational query languages", 14th Symposium on Theory of Computing, Association for Computing Machinery, 1982, 137-146.

Wa W. Walkoe, "Finite partially-ordered quantification", Journal of Symbolic Logic 35 (1970), 535-555.

AIIM SCANNER TEST CHART # 2

Spectra

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Times Roman

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Century Schoolbook Bold

4 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

6 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

8 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

10 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

News Gothic Bold Reversed

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Bodoni Italic

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Greek and Math Symbols

4 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

6 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

8 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

10 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

White

Black

Isolated Characters

e	m	1	2	3	a
4	5	6	7	o	°
8	9	0	h	I	B

A4 Page 6543210

MESH HALFTONE WEDGES

65

85

100

110

133

150

A4 Page 6543210

RIT ALPHANUMERIC RESOLUTION TEST OBJECT, RT-1-71

PRODUCED BY GRAPHIC ARTS RESEARCH CENTER

ROCHESTER INSTITUTE OF TECHNOLOGY, ONE LOMB

MEMORIAL DRIVE, ROCHESTER, NEW YORK 14623