

Introduction to Relational Logic

M.A.Malkov

Russian Research Center for Artificial Intelligence.

E-mail: "ma@malkov.msk.su".

Contents

1. Axioms of First Order Logic	6
1.1. Mathematical Logic Language	6
1.2. First Normal Form	9
1.3. Second Normal Form	12
1.4. Undefined Notions of Logic	13
2. Natural Set Theory	15
3. Finite Logic	21
3.1. Model Construction (Logic Programming)	21
3.1.1. Logic and Set Theory	22
3.1.2. Formulas without Negations. Iteration Stages	24
3.1.3. Ghost Effect	25
3.1.4. Hierarchy of Formulas	27
3.1.5. Non-monotone Calculations	28
3.1.6. Constants	32
3.1.7. Second Order Formulas	33
3.2. Theory Construction (Theorem Proving)	34
3.2.1. Resolution	35
3.2.2. Resolution with Factor	36
3.2.3. Generalized Resolution	38
3.2.4. Paramodulation	39
3.2.5. Generalized Paramodulation	39
3.2.6. Finite Descent	41
3.2.7. Generalized Finite Descent	43
3.3. Conclusion	45
References	45

Logic, to be exact, mathematical logic, is a science about construction of theories and their models. But logic is a theory too. It means there is an ordered set of logics and logic of $(n + 1)$ -th order allows to construct theory and model for logic of n -th order.

This definition is conventional in computational logic. In classic logic it is conventional to define logic as the calculus of valid formulas.

In spite of that both kind of logic have many common features and their integration allows to have more perfect science.

Such integration is the relational logic. It is based on three concepts.

According to the first concept, all mathematics objects are relations [1], because relations are convenient for computer processing. So logic formulas, being relations, have neither brackets, connectives, nor quantifiers. It means all formulas of classic logic can be present in such form and, vice versa, all computer-oriented relations can be present as formulas of classic logic. This accordance lets use below classic presentation of formulas (with brackets, connectives, and quantifiers) as more simple and clear.

Besides that, all formulas in the relational logic can be constructed from atoms (atomic formulas) without terms. It means an atom is present syntactically as a symbol with argument places filled by variable symbols. These symbols of atoms can be functional or predicate. A functional symbol has one or several additional places containing a function value. A predicate symbol has no functional dependences among its places.

But usage of atoms without terms complicates the logical formulas. So in the relational logic the terms exist, but they are introduced only for the shortened record of the formulas. And it appears, the terms can be created not only by function symbols, but predicate too. In this sense the relational logic is the extension of the classic one.

According to the second concept, all sets are non-transitive. It means, a set and its member have no common member.

Such sets can not be produced by one atom or a finite number of atoms. Therefore, it is the most natural to consider atoms as the natural numbers. So such sets are called natural.

The classic set theory is one-atomic, but its atom is not a natural number, its atom is the logic object "false". As it is known, from false we can deduce everything.

From the point of view of the relational logic, the classic set theory is inconsistent because of existence of transitive sets. In this sense the relational logic is not conforming with classic.

According to the third concept, the finite logic is defined as the logic of finite proofs and calculated relations.

The finite proofs are constructed by automated theorem proving [2]. The calculated relations are constructed by logic programming [3]. The theorem proving is used for computer-oriented theory construction, the logic programming is used for computer-oriented model construction for a given theory. The finite logic are also called the relational programming.

The relational programming ensures the higher level of the automatic programming. A description of a problem by means of logic becomes the program, ready for the execution.

The modern logic programming contains members to contradict classic logic, but all these inconsistencies can be removed.

In particular, in logic programming there are several kinds of negation. But at more close study of these kinds of negation we see that in reality there are some kinds of problems, differing by complexity of negation calculations. There are, naturally, problems that have uncalculated negations.

By its nature logic programming realizes jump from recursive functions to the recursive relations. In this sense the relational logic is a new branch of classic logic.

According to the three above concepts the "Introduction" consists of three parts.

In the first part we give the syntactical and compute-oriented forms of presentation of logic objects. We introduce the terms and the rules of logical description of theories and models. These rules are a part of an axiom system for the relational logic.

In the second part we construct the set of all models not depended on theories. For this the arithmetic set theory is used.

But in the relational logic the set theory is not a mathematical theory. So we use a metalogic for construction of this theory. As such metalogic we take classic logic.

In the third part we state fundamentals of the finite logic. The rules of computer construction of the theorem set and the models of a theory are formulated. To construct the theorem set we use inconsistent theories as a rule.

Lower the following symbols are used:

- constant as a symbol without any subscript,
- variable as a symbol with a subscript.

Constant is interpreted by a set, variable is interpreted by an arbitrary member of the set.

For example, N is the symbol of the sort interpreted as the set of natural numbers, $N_0, N_1, N_2...$ are symbols of variables, interpreted as arbitrary natural numbers.

As an exception, with the purpose of simplification, the subscripted variables can be replaced by constants, for example, N_i instead of N_{N_1} (here i is a constant, N_1 is a variable).

The structure of theories has special features too. A theory has undefined relations and basic axioms, setting properties of the undefined relations. A basic signature of a theory includes only symbols of undefined relations.

The defined relations can be recursive. The new symbols for the defined relations are entered without any limitations. It means, a full signature of any theory can be permanently refilled.

Except basic axioms the theories have theorems as deduced axioms and definitions as axioms too.

We used the next symbols for notation of negation, conjunction, disjunction, direct and back (at permutation of places) implications, logical equivalence, and existential and universal quantifiers (these logical connectives are listed in the order of their priorities):

$$\neg, \wedge, \vee, \rightarrow, \leftarrow, \rightleftarrows, \exists, \forall.$$

The quantifiers differ from other connectives only by priority. This contradicts the conventional rules, but allows considerably to reduce a number of parentheses.

These connectives are used in logic of any order.

1. Axioms of First Order Logic

The theory construction is based on the theory calculus and on the axiom calculus. The theory calculus is used for construction of a theory set, the axiom calculus is used for construction of an axiom set of a theory.

The model construction is also based on the calculus of all models (i.e. the set theory) and on the model calculus for a theory.

In this section the account of these calculi is stated partly. This account is based on the first and second normal forms.

In the first normal form all axioms of a theory should be present. These axioms include theorems and definitions. The first normal form is basic in the theory and axiom calculi.

In the second normal form all positive definitions of a theory should be present. The second form is basic in the model calculus for a theory.

As a rule, a definition consists of two parts joint by symbol of logical equivalence. The left part is a defined object, the right part is a defining formula.

Replacing logical equivalence by back implication we get a positive definition. Replacing logical equivalence by direct (usual) implication we get a negative definition.

We need both definitions in the axiom calculus. We need only positive definitions in the model calculus for a theory.

A positive definition allows to construct a model of a defined object. Negative definition allows to construct a complement to a model. But there exists a way to construct the model complement without using the negative definition.

In section 1.1 we give rules to present logic objects in syntactical form.

In section 1.2 and 1.3 we give rules to construct the first and second normal form for the computer-oriented presentation of logic objects.

In section 1.4 we give some axioms to set properties of undefined relations. These properties are present in the computer-oriented form, but the axioms are present in the classic form.

1.1. Mathematical Logic Language.

The relational logic signature consists of an enumerable set of sorts S and enumerable set of relation symbols R . Sets of constant symbols and function symbols are empty.

Arbitrary members of S are S_0, S_1, S_2, \dots , arbitrary members of R are R_0, R_1, R_2, \dots . The arbitrary members are meta-variables, which can be replaced (through substitution) by concrete members.

Besides sets S and R , the alphabet of the language includes a set of variables. The arbitrary members of this set are symbols $S_{0_0}, S_{0_1}, \dots, S_{1_0}, S_{1_1}, \dots, \dots$. Symbol S_{i_j} means j -th variable of i -th sort.

But the sort S_i of variable S_{i_j} is defined by place N_1 this variable occupies in relation R_1 :

$$S_i = s(R_1, N_1)$$

where function s is interpreted by $s : R \times N \rightarrow S$.

The alphabet of the language includes also parentheses, comma, and logic symbols of negation \neg , implication \rightarrow , and universal quantifier \forall . All words in the alphabet must be finite.

The set of all words in the alphabet is denoted W , arbitrary members of the set are denoted W_0, W_1, W_2, \dots . An arbitrary member is a meta-variable and belongs to a metalogic.

But this metalogic is the second order logic. It is naturally because the first order logic belongs to the set of the second order theories.

Now we can define an *atomic formula (atom)* as a word:

$$R_1(S_{1_k}, S_{2_l}, \dots, S_{n_m})$$

where $S_1 = s(R_1, 1), S_2 = s(R_1, 2), \dots, S_n = s(R_1, n)$, k, l, \dots, m are arbitrary natural numbers, among these numbers someone can be equal, n is a number of places in R_1 ,

The set of words being atoms is denoted A , arbitrary members of the set are denoted A_0, A_1, A_2, \dots .

We must define the relation \mathcal{F} "to be a *formula*". This definition is recursive¹:

$$\mathcal{F}(W_0) \dot{\Leftarrow} (\dot{\exists} A_0 W_0 = A_0) \dot{\vee} (\dot{\exists} W_1 \dot{\exists} W_2 W_0 = (\mathcal{F}(W_1) \rightarrow \mathcal{F}(W_2))) \dot{\vee} (\dot{\exists} W_1 W_0 = (\neg \mathcal{F}(W_1))) \dot{\vee} (\dot{\exists} W_1 W_0 = (\forall S_{1_1} \mathcal{F}(W_1)))$$

The definition uses the metalogic. The logical connectives, belonging to the metalogic, are marked by a point above the connectives. The symbol "=" belongs to the metalogic too.

This definition consists of two definitions.

If we replace logical equivalence by back implication, we get the positive definition. After reducing to conjunctive normal form (CNF) and using equation properties we have the next formulas:

$$\begin{aligned} &\mathcal{F}(A_0). \\ &\mathcal{F}(\mathcal{F}(W_1) \rightarrow \mathcal{F}(W_2)). \\ &\mathcal{F}(\neg \mathcal{F}(W_1)). \\ &\mathcal{F}(\forall S_{1_1} \mathcal{F}(W_1)). \end{aligned}$$

So we have inductive definition of "formula". Formula is:

$$\begin{aligned} &\text{atom}; \\ &(\mathcal{F}_1 \rightarrow \mathcal{F}_2), \text{ if } \mathcal{F}_1 \text{ and } \mathcal{F}_2 \text{ are formulas}; \\ &(\neg \mathcal{F}_1), \text{ if } \mathcal{F}_1 \text{ is a formula}; \\ &(\forall S_{1_1} \mathcal{F}_1), \text{ if } \mathcal{F}_1 \text{ is formula}. \end{aligned}$$

We have a negative definition, if we replace logical equivalence by direct implication:

$$\mathcal{F}(W_0) \dot{\rightarrow} (\dot{\exists} A_0 W_0 = A_0) \dot{\vee} (\dot{\exists} W_1 \dot{\exists} W_2 W_0 = (\mathcal{F}(W_1) \rightarrow \mathcal{F}(W_2))) \dot{\vee} (\dot{\exists} W_1 W_0 = (\neg \mathcal{F}(W_1))) \dot{\vee} (\dot{\exists} W_1 W_0 = (\forall S_{1_1} \mathcal{F}(W_1)))$$

We have a final part of the inductive definition:

The other formulas do not exist.

The set of all formulas is denoted \mathcal{F} too. The arbitrary members of this set are $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \dots$.

An important kind of formulas is a *valid formula*. The definition of the relation \mathcal{V} "to be a valid formula" is recursive, but we limit ourselves to the positive definition only:

$$\begin{aligned} &\mathcal{V}(\mathcal{F}_1 \rightarrow (\mathcal{F}_2 \rightarrow \mathcal{F}_1)). \\ &\mathcal{V}(((\mathcal{F}_1 \rightarrow (\mathcal{F}_2 \rightarrow \mathcal{F}_3)) \rightarrow ((\mathcal{F}_1 \rightarrow \mathcal{F}_2) \rightarrow (\mathcal{F}_1 \rightarrow \mathcal{F}_3))). \\ &\mathcal{V}(((\neg \mathcal{F}_1) \rightarrow (\neg \mathcal{F}_2)) \rightarrow (((\neg \mathcal{F}_1) \rightarrow \mathcal{F}_2) \rightarrow \mathcal{F}_1)). \\ &(\mathcal{V}(\forall S_{1_1} \mathcal{F}_1) \rightarrow \mathcal{F}_1) \\ &\text{free}(\mathcal{F}_1, S_{1_1}) \dot{\vee} \mathcal{V}(\forall S_{1_1} (\mathcal{F}_1 \rightarrow \mathcal{F}_2) \rightarrow (\mathcal{F}_1 \rightarrow (\forall S_{1_1,1} \mathcal{F}_2))) \\ &\mathcal{V}(\mathcal{F}_1) \dot{\wedge} \mathcal{V}(\mathcal{F}_1 \rightarrow \mathcal{F}_2) \dot{\rightarrow} \mathcal{V}(\mathcal{F}_2). \\ &\mathcal{V}(\mathcal{F}_1) \dot{\rightarrow} \mathcal{V}(\forall S_{1_1} \mathcal{F}_1). \end{aligned}$$

where relation $\text{free}(\mathcal{F}_1, S_{1_1})$ means formula \mathcal{F}_1 has a free occurrence of variable S_{1_1} .

The first five clauses are called axioms and last two clauses are called inference rules of logic. Besides, the negative definition is absent as a rule.

It is not well. As it is marked above, logic is not the calculus of valid formulas, but the science of construction of theories and their models. This science has its own system of axioms. These and only these axioms are logic's. At that time the logic axiom set is not completed.

We must note every theory can be present by a set of valid formulas. But such presentation is used very seldom. The relation logic does not use it.

Below we exclude valid formulas, if we do not mention them explicit.

We can shorten formulas, if we use next denotations for back implication, disjunction, conjunction and existential quantifier:

$$\begin{aligned} &(\mathcal{F}_1 \leftarrow \mathcal{F}_2) \dot{\Leftarrow} (\mathcal{F}_2 \rightarrow \mathcal{F}_1) \\ &(\mathcal{F}_1 \vee \mathcal{F}_2) \dot{\Leftarrow} ((\neg \mathcal{F}_1) \rightarrow \mathcal{F}_2) \\ &(\mathcal{F}_1 \wedge \mathcal{F}_2) \dot{\Leftarrow} (\neg((\neg \mathcal{F}_1) \vee (\neg \mathcal{F}_2))) \\ &(\mathcal{F}_1 \rightleftarrows \mathcal{F}_2) \dot{\Leftarrow} ((\mathcal{F}_1 \rightarrow \mathcal{F}_2) \wedge (\mathcal{F}_2 \rightarrow \mathcal{F}_1)) \\ &(\dot{\exists} S_{1_1} \mathcal{F}_1) \dot{\Leftarrow} (\neg(\forall S_{1_1} (\neg \mathcal{F}_1))) \end{aligned}$$

The further shortening can be obtained by decrease of a number of brackets. For this purpose we must use priority of logical connectives (see above).

But the most powerful shortening is obtained, if we use terms.

¹Errata. See *Relational Logic*, 2001, 2, p. 62

The absence of terms in the relational logic is explained by the absence of constant and function symbols. But it does not confine possibilities of the relational logic, since any constant is a one-place relation. Any symbol of a function is a relation symbol too. The relational symbol has one more place than the same symbol of the function. This additional place in the relation must be the last one.

Moreover, in addition to the classic logic, one relational symbol can correspond to several functions with several additional places.

It is necessary to point out, in the relational logic any formula of classic logic with any terms can be present.

Example 1.1. The formula $y = x + 1$ in the relational logic is represented as $\exists N_2 + (N_1, N_2, N_3) \wedge 1(N_2)$, where N is the sort of the natural numbers, N_1 replaces x , N_3 replaces y , the variable N_2 is bound. \square

In the given example the simple expression is replaced by another, which is more complex and less clear.

Therefore, it is necessary to introduce terms for reduction and simplification of complex expressions.

The following statement solves this problem:

$$R_1(\dots, R_0(\dots, -, \dots), \dots) \Leftrightarrow \exists S_{0_0} R_1(\dots, S_{0_0}, \dots) \wedge R_0(\dots, S_{0_0}, \dots).$$

In this notation the atom $R_0(\dots, -, \dots)$ is a term, the symbol “-” in R_0 is an anonymous variable. This variable gives a value of the term.

The notation allows to give syntactical definition of terms, similar to the definition in classic logic:

- a variable is a term,

- if t_1, \dots, t_n are terms and R_1 is a relation with $n+1$ places, $R_1(t_1, \dots, t_{i-1}, -, t_i, \dots, t_n)$ is a term too.

In this definition, with the purpose of simplification, the term sort problem is not touched up.

The set of all terms is denoted by t , arbitrary members of this set are denoted by t_0, t_1, t_2, \dots .

The formula $R_1(t_1, \dots, t_n)$ is also called atom.

It is necessary to mark, a term in an atom with negation:

$$\neg R_1(\dots, R_0(\dots, -, \dots), \dots) \Leftrightarrow \forall S_{0_0} \neg R_1(\dots, S_{0_0}, \dots) \vee \neg R_0(\dots, S_{0_0}, \dots)$$

differs from a term in an atom without negation.

The given notation of terms is changed, if there are functional symbols.

A symbol of a relation is called functional, if this relation has functional dependencies, and predicate, if it does not contain such dependencies.

Functional dependence means property of atoms

$$\neg R_1(\dots, t_1, \dots) \vee \neg R_1(\dots, t_2, \dots) \vee t_1 = t_2.$$

As a rule, the places for function values are last in relations.

If a value of an anonymous variable is a function value, the denotation of a term becomes ambiguous:

$$R_1(\dots, R_0(\dots, -, \dots), \dots) \Leftrightarrow (\exists S_{0_0} R_1(\dots, S_{0_0}, \dots) \wedge R_0(\dots, S_{0_0}, \dots)) \Leftrightarrow \forall S_{0_0} R_1(\dots, S_{0_0}, \dots) \vee \neg R_0(\dots, S_{0_0}, \dots)$$

but terms in atoms without and with negations become identical.

It is easy to show, the formula

$$(\exists S_{0_0} R_1(\dots, S_{0_0}, \dots) \wedge R_0(\dots, S_{0_0}, \dots)) \rightarrow \forall S_{0_0} R_1(\dots, S_{0_0}, \dots) \vee \neg R_0(\dots, S_{0_0}, \dots).$$

is deduced from the uniqueness of a functional value $R_0(\dots, S_{0_1}, \dots) \wedge R_0(\dots, S_{0_2}, \dots) \rightarrow S_{0_1} = S_{0_2}$, and the formula

$$(\exists S_{0_0} R_1(\dots, S_{0_0}, \dots) \wedge R_0(\dots, S_{0_0}, \dots)) \leftarrow \forall S_{0_0} R_1(\dots, S_{0_0}, \dots) \vee \neg R_0(\dots, S_{0_0}, \dots).$$

is deduced from the existence of a function value $\exists S_{0_0} R_0(\dots, S_{0_0}, \dots)$.

The ambiguity of term denotation is eliminated by usage of expressions with a universal quantifier in the first normal form (see section 1.2) and by usage of expressions with an existential quantifier in the second normal form (see section 1.3).

Usage of terms considerably simplifies formulas. But our purpose is the maximal approaching to language of the first order classic logic.

To reach this purpose we shall define a function symbol (a constant is considered as a special case of a function).

A functional symbol becomes a function symbol, if the relation contains one functional dependence. The place for the function value is last.

Terms that contain only function symbols and variables, we shall call functions.

For such terms the anonymous variable always is the last, and so it can be omitted. If in terms (and atoms too) to admit also usage of infix and postfix notations, the presentation of such terms and formulas becomes identical in both kinds of logic.

New is that in classic logic terms and relations are independent.

In the relational logic terms and relations are dependent.

Besides, terms can contain predicate symbol. Usage of such terms allows to simplify formulas and to reduce a number of quantifiers in these formulas. This possibility should find broad applying.

Example 1.2. We shall demonstrate how to reduce the definition of the factorial in the relational logic to the definition in classic logic.

In the prefix form and without terms this definition is

$$\forall N_1 \forall N_2 ! (N_1, N_2) \Leftrightarrow \exists N_3 \exists N_4 \exists N_5 ! (N_3, N_4) \wedge 1(N_5) \wedge + (N_3, N_5, N_1) \wedge * (N_4, N_1, N_2).$$

where $!(N_1, N_2)$ is the prefix form of the factorial $N_1! = N_2$.

Using terms instead of variables N_4 and N_5 , we have:

$$\forall N_1 \forall N_2 ! (N_1, N_2) \Leftrightarrow \exists N_3 + (N_3, 1(-), N_1) \wedge * (! (N_3, -), N_1, N_2).$$

Symbols of functions are “+”, “*” and “!”. Removing the last argument in the atoms with these symbols and using the infix and postfix forms, we receive:

$$\forall N_1 \forall N_2 N_1! = N_2 \Leftrightarrow \exists N_3 N_3 + 1 = N_1 \wedge N_3! * N_1 = N_2.$$

This definition can be present as positive and negative:

$$\forall N_1 \forall N_2 N_1! = N_2 \leftarrow \exists N_3 N_3 + 1 = N_1 \wedge N_3! * N_1 = N_2.$$

$$\forall N_1 \forall N_2 N_1! = N_2 \rightarrow \exists N_3 N_3 + 1 = N_1 \wedge N_3! * N_1 = N_2.$$

The positive definition can be present as:

$$\forall N_1 \forall N_2 \forall N_3 N_1! = N_2 \vee N_3 + 1 \neq N_1 \vee N_3! * N_1 \neq N_2.$$

This definition can be simplified, using properties of inequalities in a disjunction:

$$\forall N_3 (N_3 + 1)! = N_3! * (N_3 + 1).$$

The received definition coincides with those in classic logic.

The negative definition can be present as:

$$\forall N_1 \forall N_2 \exists N_3 N_1! \neq N_2 \vee N_3 + 1 = N_1 \wedge N_3! * N_1 = N_2.$$

Using properties of inequalities in a disjunction, again we have

$$\exists N_3 N_3! * N_3 = (N_3 + 1)!.$$

Using properties of equalities in a conjunction, we receive the atomic formula:

$$\exists N_3 N_3! * N_3 = (N_3 + 1)!.$$

This formula is a particular case of positive definition.

Additionally we must define the factorial at zero point.

In the relational logic the factorial at zero point is defined by the formula:

$$\forall N_1 (\exists N_0 0(N_0) \wedge ! (N_0, N_1)) \Leftrightarrow \exists N_0 1(N_0) \wedge = (N_1, N_0) \quad \text{or} \quad \forall N_1 0! = N_1 \Leftrightarrow N_1 = 1.$$

The positive definition looks like:

$$\forall N_1 0! = N_1 \vee N_1 \neq 1.$$

Using properties of inequalities in a disjunction, we receive the same result, as in classic logic:

$$0! = 1.$$

The same result we shall receive for the negative definition.

Therefore, the negative definition is superfluous. This conclusion is fair for all primitive recursive functions. \square

It follows from the given example, in the classic logic definitions are frequently based on intuition of mathematicians. The relational logic allows to justify correctness of classic definition.

Example 1.3. We shall demonstrate how a predicate can be used as a term.

Let the relation A be interpreted by $\{ \langle 12, 5, 7 \rangle, \langle 9, 15, 6 \rangle, \langle 12, 15, 8 \rangle \}$ and the relation B be interpreted by $\{6, 7\}$.

Then $A(N_1, N_2, B(-))$ is interpreted by $\{ \langle 12, 5 \rangle, \langle 9, 15 \rangle \}$. \square

1.2. First Normal Form.

As it was specified above, the mathematical logic is a science to construct theories and their models.

There are many ways of the presentation of the first order theory set. The axioms of the relational logic must ensure only one of this ways.

One way of such presentation is to remove all valid formulas from the theories of the first order. In this sense the problems of classic and relational logics are diametrically opposite.

Initial information for construction of the first order theories is the set of the axioms of these theories.

There are many ways of the presentation of the axioms. This ambiguity is eliminated in the relational logic by reduction to the first normal form.

The first normal form is based on $\forall\exists$ CNF (a prenex conjunctive normal form with universal quantifiers preceded to existential quantifiers).

The reduction to $\forall\exists$ CNF is realized by a sequence of steps in accordance with the following rule.

Rule 1.4 of the reduction to $\forall\exists$ CNF:

- the transformation of the axioms to the closed formulas by addition of universal quantifiers for all free variables;
- the transformation to the prenex form with all quantifiers in the beginning of the formula;
- the reduction to $\forall\exists$ -form, i.e. the partition of one sentence onto some sentences with universal quantifiers preceded to existential quantifiers;
- the reduction (if there are existential quantifiers) to the disjunctive normal form (DNF), at reduction the properties of equalities (for simplification DNF) are used;
- the transference of existential quantifiers into \wedge -clauses (an \wedge -clause is a literal with variables bound by existential quantifiers or a sequence of such literals, joint by conjunctions, a literal is an atom without or with negation);
- the reduction (with usage of the equality properties) to a conjunctive normal form (CNF); if there are existential quantifiers, their scope (an \wedge -clause) is considered as monolithic, not destroyed at the reduction to CNF.

So the reduction to $\forall\exists$ CNF is finished.

If existential quantifiers are absent, we have a quantifier prefix and conjunction of \vee -clauses (a \vee -clause is a literal with variables not bound by existential quantifiers or a sequence of such literals joint by disjunctions).

If there are existential quantifiers, we have a universal quantifier prefix and conjunctions of clauses (a clause is either a \vee -clause, an \wedge -clause, or a sequence of such clauses joint by disjunctions). Every \wedge -clause into the clause has an existential quantifier prefix.

In $\forall\exists$ CNF all quantifiers (therefore, all variables) can be absent. If constants are absent too, $\forall\exists$ CNF turns into a propositional formula. It is not forbidden, but better not to include such formulas in the first order theories.

Rule 1.5 of the reduction to the first normal form:

- the reduction to $\forall\exists$ CNF;
- the separation of each clause of CNF into the isolate sentence;
- the introduction of the new identifications of variables: all variables, bound by universal quantifiers, become x_1, x_2, x_3, \dots , variables, bound by existential quantifiers, become a_1, a_2, a_3, \dots , and the subscript numeration starts anew in each \wedge -clause;
- all quantifiers are removed.

As a result of reduction to the first normal form each formula becomes a quantifier-free clause, i.e. a sequence of one \vee -clause and some \wedge -clauses, connected by disjunctions.

All steps above are particularized in the textbooks of classic logic. The exceptions are the reduction to $\forall\exists$ -form and the transference of existential quantifiers in \wedge -clauses. Let these steps be considered in more detail.

In the general case before the reduction to the $\forall\exists$ -form, any sequence of quantifiers in the prenex form can be present as:

$$\forall x_{11} \forall x_{12} \dots \exists y_{11} \exists y_{12} \dots \dots \forall x_{n1} \forall x_{n2} \dots \exists y_{n1} \exists y_{n2} \dots \mathcal{F}_0$$

where \mathcal{F}_0 is a quantifier-free formula with variables $x_{11}, x_{12}, \dots, \dots, x_{n1}, x_{n2}, \dots, y_{11}, y_{12}, \dots, \dots, y_{n1}, y_{n2}, \dots$.

This formula is divided into some sentences in $\forall\exists$ -form. The first of them is

$$\forall x_{11} \forall x_{12} \dots \exists y_{11} \exists y_{12} \dots R_1(x_{11}, x_{12}, \dots, y_{11}, y_{12}, \dots)$$

where R_1 is a relation defined through a line-up of another relations:

$$R_1(x_{11}, x_{12}, \dots, y_{11}, y_{12}, \dots) \Leftrightarrow \forall x_{21} \forall x_{22} \dots R_2(x_{11}, x_{12}, \dots, x_{21}, x_{22}, \dots, y_{11}, y_{12}, \dots)$$

$$R_2(x_{11}, x_{12}, \dots, x_{21}, x_{22}, \dots, y_{11}, y_{12}, \dots) \Leftrightarrow$$

$$\exists y_{21} \exists y_{22} \dots R_3(x_{11}, x_{12}, \dots, x_{21}, x_{22}, \dots, y_{11}, y_{12}, \dots, y_{21}, y_{22}, \dots)$$

...

$$R_{2n-2}(x_{11}, x_{12}, \dots, \dots, x_{n1}, x_{n2}, \dots, y_{11}, y_{12}, \dots, \dots, y_{n-1,1}, y_{n-1,2}, \dots) \Leftrightarrow \exists y_{n1} \exists y_{n2} \dots \mathcal{F}_0.$$

It is easy to show, each of these definitions is divided into two sentences in $\forall\exists$ -form.

Example 1.6. The axiom of arithmetic about existence of zero is:

$$\exists_1 N_1 \forall N_2 N_2' \neq N_1.$$

where " ' " - a symbol of a successor operation.

For reduction to $\forall\exists$ -form we enter the new relation 0:

$$0(N_1) \Leftrightarrow \forall N_2 N_2' \neq N_1.$$

As a result we have 4 sentences in $\forall\exists$ -form:

$$\forall N_1 \forall N_2 0(N_1) \rightarrow N_2' \neq N_1.$$

$$\forall N_1 \exists N_2 0(N_1) \leftarrow N_2' \neq N_1.$$

$$\exists N_1 0(N_1).$$

$$\forall N_1 \forall N_2 \neg 0(N_1) \vee \neg 0(N_2) \vee N_1 = N_2.$$

Thus, the axiom of existence of zero is simultaneously a definition of zero as a constant. \square

Example 1.7. To illustrate a transference of existential quantifiers into \wedge -clauses, we consider definition of the unordered pair in the classic set theory (the reduction to the clause form is well for every theory, consistent or inconsistent, symbolic or semantic):

$$\{X_1, X_2\} = X_3 \Leftrightarrow \forall X_4 Pr(X_4) \vee (X_4 \in X_3 \Leftrightarrow X_4 = X_1 \vee X_4 = X_2)$$

where X_1, X_2, X_3, X_4 are classes, Pr is the relation "to be proper class".

The reduction of this definition to the first normal form has the following sequence of transformations.

The addition of universal quantifiers:

$$\forall X_1 \forall X_2 \forall X_3 \{X_1, X_2\} = X_3 \Leftrightarrow \forall X_4 Pr(X_4) \vee (X_4 \in X_3 \Leftrightarrow X_4 = X_1 \vee X_4 = X_2).$$

The transformation to the prenex form:

$$\forall X_1 \forall X_2 \forall X_3 \exists X_4 \{X_1, X_2\} = X_3 \vee \neg Pr(X_4) \wedge (X_4 \notin X_3 \Leftrightarrow X_4 = X_1 \vee X_4 = X_2)$$

$$\forall X_1 \forall X_2 \forall X_3 \forall X_4 \{X_1, X_2\} \neq X_3 \vee Pr(X_4) \vee (X_4 \in X_3 \Leftrightarrow X_4 = X_1 \vee X_4 = X_2)$$

The transformation to $\forall\exists$ -form changes nothing.

The reduction to DNF of the first of these formulas:

$$\forall X_1 \forall X_2 \forall X_3 \exists X_4 \{X_1, X_2\} = X_3 \vee (\neg Pr(X_4) \wedge X_4 \notin X_3 \wedge X_4 = X_1) \vee$$

$$(\neg Pr(X_4) \wedge X_4 \notin X_3 \wedge X_4 = X_2) \vee (\neg Pr(X_4) \wedge X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2).$$

The transformation with usage of properties of equality:

$$\forall X_1 \forall X_2 \forall X_3 \exists X_4 \{X_1, X_2\} = X_3 \vee \neg Pr(X_1) \wedge X_1 \notin X_3 \vee$$

$$\neg Pr(X_2) \wedge X_2 \notin X_3 \vee X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2.$$

Also at this transformation the next formula is used:

$$\neg Pr(X_4) \wedge X_4 \in X_3 \Leftrightarrow X_4 \in X_3.$$

The transference of the existential quantifier to \wedge -clauses:

$$\forall X_1 \forall X_2 \forall X_3 \{X_1, X_2\} = X_3 \vee (\neg Pr(X_1) \wedge X_1 \notin X_3) \vee (\neg Pr(X_2) \wedge X_2 \notin X_3) \vee$$

$$(\exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2).$$

The reduction of this sentence to CNF:

$$\forall X_1 \forall X_2 \forall X_3 (\{X_1, X_2\} = X_3 \vee \neg Pr(X_1) \vee \neg Pr(X_2) \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2) \wedge$$

$$(\{X_1, X_2\} = X_3 \vee \neg Pr(X_1) \vee X_2 \notin X_3 \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2) \wedge$$

$$(\{X_1, X_2\} = X_3 \vee X_1 \notin X_3 \vee \neg Pr(X_2) \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2) \wedge$$

$$(\{X_1, X_2\} = X_3 \vee X_1 \notin X_3 \vee X_2 \notin X_3 \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2).$$

The partition of the sentences into separate formulas:

$$\forall X_1 \forall X_2 \forall X_3 \{X_1, X_2\} = X_3 \vee \neg Pr(X_1) \vee \neg Pr(X_2) \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2.$$

$$\forall X_1 \forall X_2 \forall X_3 \{X_1, X_2\} = X_3 \vee \neg Pr(X_1) \vee X_2 \notin X_3 \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2.$$

$$\forall X_1 \forall X_2 \forall X_3 \{X_1, X_2\} = X_3 \vee X_1 \notin X_3 \vee \neg Pr(X_2) \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2.$$

$$\forall X_1 \forall X_2 \forall X_3 \{X_1, X_2\} = X_3 \vee X_1 \notin X_3 \vee X_2 \notin X_3 \vee \exists X_4 X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2.$$

The reduction to CNF of the second source formula:

$$\begin{aligned} & \forall X_1 \forall X_2 \forall X_3 \forall X_4 (\{X_1, X_2\} \neq X_3 \vee Pr(X_4) \vee X_4 \in X_3 \vee X_4 \neq X_1) \wedge \\ & (\{X_1, X_2\} \neq X_3 \vee Pr(X_4) \vee X_4 \in X_3 \vee X_4 \neq X_2) \wedge \\ & (\{X_1, X_2\} \neq X_3 \vee Pr(X_4) \vee X_4 \notin X_3 \vee X_4 = X_1 \vee X_4 = X_2). \end{aligned}$$

The transformation with usage of properties of equality:

$$\begin{aligned} & \forall X_1 \forall X_2 \forall X_4 (Pr(X_1) \vee X_1 \in \{X_1, X_2\}) \wedge (Pr(X_2) \vee X_2 \in \{X_1, X_2\}) \wedge \\ & (X_4 \notin \{X_1, X_2\} \vee X_4 = X_1 \vee X_4 = X_2) \end{aligned}$$

Also at transformation the next equivalence is used

$$Pr(X_4) \vee X_4 \notin X_3 \Leftrightarrow X_4 \notin X_3.$$

The division of sentences of this CNF into separate formulas:

$$\begin{aligned} & \forall X_1 \forall X_2 Pr(X_1) \vee X_1 \in \{X_1, X_2\}. \\ & \forall X_1 \forall X_2 Pr(X_2) \vee X_2 \in \{X_1, X_2\}. \\ & \forall X_1 \forall X_2 \forall X_4 X_4 \notin \{X_1, X_2\} \vee X_4 = X_1 \vee X_4 = X_2). \end{aligned}$$

The change of variable denotation (in both initial formulas):

$$\begin{aligned} & \forall x_1 \forall x_2 \forall x_3 \{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee \neg Pr(x_2) \vee \exists a_1 a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \forall x_1 \forall x_2 \forall x_3 \{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee x_2 \notin x_3 \vee \exists a_1 a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \forall x_1 \forall x_2 \forall x_3 \{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee \neg Pr(x_2) \vee \exists a_1 a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \forall x_1 \forall x_2 \forall x_3 \{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee x_2 \notin x_3 \vee \exists a_1 a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \quad \forall x_1 \forall x_2 Pr(x_1) \vee x_1 \in \{x_1, x_2\}. \\ & \quad \forall x_1 \forall x_2 Pr(x_2) \vee x_2 \in \{x_1, x_2\}. \\ & \quad \forall x_1 \forall x_2 \forall x_3 x_3 \notin \{x_1, x_2\} \vee x_3 = x_1 \vee x_3 = x_2). \end{aligned}$$

The removal of quantifiers:

$$\begin{aligned} & \{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee \neg Pr(x_2) \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee x_2 \notin x_3 \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee \neg Pr(x_2) \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee x_2 \notin x_3 \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \quad Pr(x_1) \vee x_1 \in \{x_1, x_2\}. \\ & \quad Pr(x_2) \vee x_2 \in \{x_1, x_2\}. \\ & \quad x_3 \notin \{x_1, x_2\} \vee x_3 = x_1 \vee x_3 = x_2. \end{aligned}$$

The reduction to the first normal form is finished.

The received system of the formulas is more complex than the initial formula, but it ensures a uniqueness of presentation. Besides, after enough experience these formulas are easily interpreted. But the main thing is that the reduction to the first normal form is a necessary condition of computer formalization. \square

1.3. Second Normal Form.

The second normal form is destined for constructing models of defined objects in a theory.

The reduction to this form is defined by a sequence of steps in accordance with the following rule.

Rule 1.8 of reduction to the second normal form:

- the reduction of positive definition to the first normal form, and the defined relation must be in the beginning of each formula;
- the replacement of \wedge -clauses in the sentences by new relations and the addition of positive definitions for these relations (such definitions do not contain variables, bound by existential quantifiers);
- the transformation of all positive definitions into back implication with the defined relation as the left part and with negation of the other part of the definition as the right part of back implication.

On it the construction of the second normal form is finished. The right part of back implication contains an \wedge -clause. All variables of this clause absent in the left part, are connected by implicit existential quantifiers.

Below, for obviousness, we use the old denotation of variables in the second normal form.

Example 1.9. We shall present in the second normal form a positive definition of the unordered pair:

$$\begin{aligned} & \{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee \neg Pr(x_2) \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \\ & \{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee x_2 \notin x_3 \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2. \end{aligned}$$

$$\{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee \neg Pr(x_2) \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2.$$

$$\{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee x_2 \notin x_3 \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2.$$

We shall replace the \wedge -clause with variable a_1 by the relation $New(x_1, x_2, x_3)$ and add positive definition of this \wedge -clause:

$$New(x_1, x_2, x_3) \vee x_4 \notin x_3 \vee x_4 = x_1 \vee x_4 = x_2.$$

$$\{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee \neg Pr(x_2) \vee New(x_1, x_2, x_3).$$

$$\{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee x_2 \notin x_3 \vee New(x_1, x_2, x_3).$$

$$\{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee \neg Pr(x_2) \vee New(x_1, x_2, x_3).$$

$$\{x_1, x_2\} = x_3 \vee x_1 \notin x_3 \vee x_2 \notin x_3 \vee New(x_1, x_2, x_3).$$

These formulas are transformed into back implication form:

$$New(x_1, x_2, x_3) \leftarrow x_4 \in x_3 \wedge x_4 \neq x_1 \wedge x_4 \neq x_2.$$

$$\{x_1, x_2\} = x_3 \leftarrow Pr(x_1) \wedge Pr(x_2) \wedge \neg New(x_1, x_2, x_3).$$

$$\{x_1, x_2\} = x_3 \leftarrow Pr(x_1) \wedge x_2 \in x_3 \wedge \neg New(x_1, x_2, x_3).$$

$$\{x_1, x_2\} = x_3 \leftarrow x_1 \in x_3 \wedge Pr(x_2) \wedge \neg New(x_1, x_2, x_3).$$

$$\{x_1, x_2\} = x_3 \leftarrow x_1 \in x_3 \wedge x_2 \in x_3 \wedge \neg New(x_1, x_2, x_3).$$

In the first of these formulas the variable x_4 is connected by an existential quantifier, but in logic programming such variables have not any special identification. The other rule acts here: all the variables met only in the right part are connected by implicit existential quantifiers.

The absence of special identifications for a variable, bound by existential quantifiers, allows to return to old identifications (in the first normal form this is impossible):

$$New(X_1, X_2, X_3) \leftarrow X_4 \in X_3 \wedge X_4 \neq X_1 \wedge X_4 \neq X_2.$$

$$\{X_1, X_2\} = X_3 \leftarrow Pr(X_1) \wedge Pr(X_2) \wedge \neg New(X_1, X_2, X_3).$$

$$\{X_1, X_2\} = X_3 \leftarrow Pr(X_1) \wedge X_2 \in X_3 \wedge \neg New(X_1, X_2, X_3).$$

$$\{X_1, X_2\} = X_3 \leftarrow X_1 \in X_3 \wedge Pr(X_2) \wedge \neg New(X_1, X_2, X_3).$$

$$\{X_1, X_2\} = X_3 \leftarrow X_1 \in X_3 \wedge X_2 \in X_3 \wedge \neg New(X_1, X_2, X_3).$$

These formulas have the next interpretation.

In the first formula the relation New for each X_1 and X_2 contains the list of X_3 and every X_3 has one or more members not equal to X_1 and X_2 .

We can use New , but not complement to New , in the other formulas. E.g. in the last formula we can construct set $X_1 \in X_3 \wedge X_2 \in X_3$ and remove all its members, met in New (it is a set difference).

The given example is only demonstration, because all its relations are infinite. \square

As it follows from the second normal form, the right part of the formulas contains only conjunctions of literals. The disjunction of literals is contained indirectly in these formulas, through operation of back implication.

1.4. Undefined Notions of Logic.

After reducing axioms of all the first order theories to the first normal form and all positive definitions to the second normal form we can construct axioms of the first order logic.

While constructing these axioms we use the experience accumulated in the computational logic.

Basic in the computational logic is the experience of constructing programs for theorem proving. All further account uses this experience.

It is known, any computer program, including theorem proving program, can be present by means of logic programming as sets of formulas, and each formula is an axiom. As the theorem prover consists of a very large number of formulas, the number of axioms of the first order logic is great and is measured by thousands. Only some of them will be given below, and without reduction to $\forall\exists\text{CNF}$. This will allow to increase the number of the axioms a little more. A special series of articles will be dedicated to a full exposition of the first order logic axioms.

First of all, the logic axioms include the four axioms of arithmetic [4]. Therefore, undefined relations of the logic are the sort N , interpreted as the set of natural numbers, and the successor function " ' ".

Besides, there are 4 undefined notions. They are: *Relations*, *Functions*, *Theories* and *Models*.

The four-place *Relations* defines sorts of variables met in a relation of a theory.

The first argument *Relations* contains the code of a theory, the second argument - the code of a

relation, the third argument - the serial number of place in the relation (numbering starts with 0) and the last argument - the code of a sort. The function (and constant) symbols are present in *Relations* as relation symbols.

The three-place *Functions* defines the number of places for function values in a functional relation. The first place of *Functions* is for code of a theory, the second - for code of a relational symbol of the theory, the third - for the number of places for function values in the relation. These places must be last in relations.

The six-place relation *Theories* defines axioms of each theory, parts of these axioms, symbols of these parts, and kinds of these symbols. The places of *Theories* contain:

- the code of a theory;
- the serial number of an axiom in this theory;
- the serial number of a part of the axiom (any axiom has some parts, part 0 is the \vee -clause, part 1 is the first \wedge -clause, part 2 is the second \wedge -clause etc.);
- the serial number of a symbol in the part;
- the symbol code;
- the symbol kind.

There are 6 kinds of symbols:

- 0 - a symbol of a relation without negation;
- 1 - a symbol of a relation with negation;
- 2 - a symbol of a function or a predicate into a relation;
- 3 - a symbol of a variable bound by a universal quantifier;
- 4 - a symbol of a variable bound by an existential quantifier in $\forall\exists$ -prenex forms
- 5 - a symbol of a variable bound by an existential quantifier in $\exists\forall$ -prenex forms

The code of a variable is its subscript. This code is the positive number, because code 0 is for anonymous variables. Variables, bound by existential quantifiers are numbered anew in each \wedge -clause.

The last undefined relation *Models* lists formulas for constructing theory models. It has 5 places:

- the code of a model;
- the serial number of a formula in this model;
- the serial number of a symbol in the formula;
- the symbol code;
- the symbol kind.

Kinds 4 and 5 are absent, since the type of a quantifier is defined automatically in this case.

Example 1.10. As an example we again consider the first axiom of the unordered pair:

$$\{x_1, x_2\} = x_3 \vee \neg Pr(x_1) \vee \neg Pr(x_2) \vee a_1 \in x_3 \wedge a_1 \neq x_1 \wedge a_1 \neq x_2.$$

The interpretation of *Relations* and *Theories* for the given axiom is present in tabular form.

Relations				Theories						
Theory	Relation	Argument place	Sort	Theory	Axiom	Part	No	Symbol	Kind	
<i>NBG</i>	=	0	X	<i>NBG</i>	<i>Pair - 1</i>	0	0	=		
		1	X				1	{ }		
	{ }	0	X				2	x_1		
		1	X				3	x_2		
		2	X				4	x_3		
		2	X				5	<i>Pr</i>	\neg	
	<i>Pr</i>	0	X				6	x_1		
		0	X				7	<i>Pr</i>	\neg	
	\in	\in	1				X	8	x_2	
			0				X	0	\in	
1			X	1	a_1					
2			X	2	x_3					
3			X	3	=	\neg				
4			X	4	a_1					
5			X	5	x_1					
6			X	6	=	\neg				
7	X	7	a_1							
8	X	8	x_2							

For obviousness the codes in columns are substituted by the corresponding identifications, and repeated values (in the left columns) are removed. As it follows from the right table, there is one-to-one transformation between classic and computer (without brackets and logical symbols) presentations of the formulas in the first normal form. \square

The properties of undefined relations are set by the corresponding axioms of logic. Some of these axioms are indicated below.

Axiom 1.11. $Relations(N'_1,) \rightarrow Relations(N_1,)$.

The theories are encoded since 0 in succession. It allows automatically to assign codes to the new theories. The absent arguments in this axiom are the anonymous variables, the values of these variables are skipped. Generally relation with anonymous variables is interpreted as a set (table) with removed places (columns) for anonymous variables.

Axiom 1.12. $Relations(N_1, N'_2,) \wedge N_2 \geq 3 \rightarrow Relations(N_1, N_2,)$.

The symbols of the relations in each theory are encoded since 3 in succession. The code 0 is reserved for a symbol of equality "=", code 1 - for a symbol of " \leq ", code 2 - for a symbol " \preceq ".

Axiom 1.13. $Relations(N_1, N_2, N'_3,) \rightarrow Relations(N_1, N_2, N_3,)$.

Places in relations are numbered since 0 in succession.

Axiom 1.14. $Relations(, , N'_1) \rightarrow Relations(, , N_1)$.

Sorts in relations are numbered since 0 in succession.

Axiom 1.15. $Relations(N_1, N_2, N_3, N_4) \wedge Relation(N_1, N_2, N_3, N_5) \rightarrow N_4 = N_5$.

This means *Relations* is functional.

Axiom 1.16. $Relations(N_1, N_2,) \leftarrow Functions(N_1, N_2,)$.

The codes of functions should be present in *Relations*.

Axiom 1.17. $Relations(N_1, N_2,) \Leftrightarrow Theories(N_1, , , N_2, 0) \vee Theories(N_1, , , N_2, 1)$

Codes of the relations, introduced in *Relations*, should be present in *Theories*, and on the contrary, codes of the relations in *Theories* should be present in *Relations*.

The similar axioms are present for the other undefined relations too. Besides, the additional axioms provide a uniqueness of presentation of the theories, including the absence of isomorphism and subsumption.

Definition 1.18. The first order theory axiom is named *subsumed*, if it received from the other axiom at:

- the substitution of one or several variables by terms;
- the joint of the arbitrary formulas by disjunction.

Definition 1.19. An axiom is named *auto-subsumed*, if it contains superfluous literals (in \vee -clause) or superfluous \wedge -clauses.

In particular, an axiom is auto-subsumed, if it has duplicate literals (in \vee -clause) or duplicate \wedge -clauses.

Theorem 1.20. *A literal in a \vee -clause is superfluous, if this literal has local variables (i.e. variables located only in this literal) and there is an substitution of these local variables such that substituted literal becomes equal to any literal of the \vee -clause.*

Proof: The axiom without the superfluous literal is deduced from the axiom with the superfluous literal. Therefore, this literal is really superfluous. \square

2. Set Theory

Except atoms to be the natural numbers, there are two atoms which are logic objects f ("false") and t ("true"). These objects are 0-place relations.

The 0-place relations have no variable. So they are prime propositions [5].

It means the term "variable propositions" is not valid, but the term "undefined propositions" is valid. Such propositions are pseudo-variables and the collection A of two atoms f and t is pseudo-set. Then A_0, A_1, A_2, \dots are arbitrary members of A or symbols of pseudo-variables. The pseudo-variables can not be connected by quantifiers. Only this differs them from variables.

The classic set theory has neither set nor class, containing both atoms f and t , since atom t becomes universal class. So the classic set theory is monatomic. Its atom is f .

Below we treat atoms only as the natural numbers. We call sets generated by them as natural sets. The theory of natural sets we call *NS*-theory.

Note, any natural number is a relation in the relational logic. This relation is interpreted by a set with one member. But this member is the natural number too. Therefore, this member is a relation interpreted by a set with one member, etc., till infinity.

So we must separate terms “relation” and “set”: the relation is an object of theories, the set and its members - objects of models. It means there is the relation 0 (an object of a theory) and there is the atom 0 (an object of a model). They are different objects, but below we do not introduce new identifications to differ them.

Therefore, in the relational logic there are theories and there are models, and there is no theory of models. But the theory of models, as well as the theory of sets, exists in metalogic. We shall use classic logic as such metalogic.

The set theory is called arithmetic (*AS*) if its objects constructed from the set of natural numbers by a finite number of the operations of direct product and power set. *AS*-theory is a part of *NS*-theory (see below).

Now we construct *AS*-theory by means of classic logic.

In theory *AS* there are collections that are not sets. So we introduce the term “class”. This term includes both sets and not sets.

Undefined symbols of the theory *AS* are predicate symbol “ \in ” (the symbol of the relation of membership), one-place function symbol “ $'$ ” (the symbol of the succession), two-place function symbol “ $\langle \quad \rangle$ ” (the symbol of the ordered pair) and one sort of variables “ X ”, interpreted as collection of all classes. In the relational logic this sort does not exist.

It is necessary to mark, the definition of the classic ordered pair uses non-arithmetic sets. For example, the pair $\langle 0, \{0\} \rangle$ is defined as set $\{\{0\}, \{0, \{0\}\}\}$. But this set and its member $\{0, \{0\}\}$ have common member $\{0\}$. Therefore, in the relational logic the ordered pair is undefined notion.

We begin from the definition of the other notions of theory *AS*.

Definition 2.1. $Atomic(X_1) \Leftrightarrow \forall X_2 X_2 \notin X_1$.

The *atomic* class $Atomic(X_1)$ does not contain members. But it is not an empty class, because the empty class is a nonexistent class.

The nonexistent class can be denoted by the symbol f (false). But the symbols t and f are not objects of the theory *AS*, since they are easily replaced by formulas and these formulas are not to contain these symbols. So symbols t and f are not used below, and we always mean the empty class as the nonexistent class.

Corollary 2.2. $Atomic(X_1) \vee X_2 \notin X_1 \Leftrightarrow X_2 \notin X_1$.
 $\neg Atomic(X_1) \wedge X_2 \in X_1 \Leftrightarrow X_2 \in X_1$.

These formulas are an example of auto-subsumption, since the left part of the formulas contains a superfluous literal.

Definition 2.3. $M(X_1) \Leftrightarrow (\exists X_2 X_2 \in X_1) \wedge \exists X_2 X_1 \in X_2$.

The *set* $M(X_1)$ is defined as a class containing other classes as members and being a member of some other class.

Corollary 2.4. $M(X_1) \wedge X_2 \in X_1 \wedge X_1 \in X_3 \Leftrightarrow X_2 \in X_1 \wedge X_1 \in X_3$.
 $\neg M(X_1) \vee X_2 \notin X_1 \vee X_1 \notin X_3 \Leftrightarrow X_2 \notin X_1 \vee X_1 \notin X_3$.

Definition 2.5. $Pr(X_1) \Leftrightarrow \forall X_2 X_1 \notin X_2$.

The *proper* class $Pr(X_1)$ is not a member of other classes.

Corollary 2.6. $Pr(X_1) \vee X_1 \notin X_2 \Leftrightarrow X_1 \notin X_2$.
 $\neg Pr(X_1) \wedge X_1 \in X_2 \Leftrightarrow X_1 \in X_2$.

Theorem 2.7. *If the proper class X_1 is a member of X_2 , then X_1 can be removed from X_2 :*

$$Pr(X_1) \wedge X_1 \in X_2 \rightarrow X_1 \notin X_2.$$

Proof: The theorem follows from the definition of Pr . \square

Inverse is not true:

Theorem 2.8. *It is impossible to add the proper class X_2 to the class X_1 as a member:*

$$\neg(Pr(X_1) \wedge X_1 \notin X_2 \rightarrow X_1 \in X_2).$$

Proof: The theorem follows from the existence of Pr . \square

Both theorems are true in the classic set theory too.

Further we shall define the notion of equality of classes.

Definition 2.9. $Atomic(X_1) \vee Atomic(X_2) \vee (X_1 = X_2 \Leftrightarrow \forall X_3 X_3 \in X_1 \Leftrightarrow X_3 \in X_2)$

If the classes that are not atomic consist of identical members, these classes are *equal*.

The equality of atomic classes is defined in more complex way, since the atomic classes are divided into *ordered pairs* and *atoms* (natural numbers):

Axiom 2.10 (OP). $Atomic(\langle X_1, X_2 \rangle)$.

Definition 2.11. $Atom(X_1) \Leftrightarrow Atomic(X_1) \wedge \forall X_2 \forall X_3 X_1 \neq \langle X_2, X_3 \rangle$.

Corollary 2.12. $Atomic(X_1) \Leftrightarrow Atom(X_1) \vee \exists X_2 \exists X_3 X_1 = \langle X_2, X_3 \rangle$.

The ordered pairs are *equal*, if their first and second elements are equal:

Definition 2.13. $\langle X_1, X_2 \rangle = \langle X_3, X_4 \rangle \Leftrightarrow X_1 = X_3 \wedge X_2 = X_4$.

We shall define *ordered n-tuple*:

Definition 2.14. $\langle X_1, \dots, X_n \rangle = \langle \langle X_1, \dots, X_{n-1} \rangle, X_n \rangle$.

Corollary 2.15. $\langle X_{1,1}, \dots, X_{1,n} \rangle = \langle X_{2,1}, \dots, X_{2,n} \rangle \Leftrightarrow X_{1,1} = X_{2,1} \wedge \dots \wedge X_{1,n} = X_{2,n}$.

Ordered *n*-tuples are equal, if their *i*-th elements are equal.

The atom can be defined more detail with help of the axioms of arithmetic. There are 4 axioms.

The first axiom of arithmetic states the successor function is injective:

Axiom 2.16 (Ar1). $Atom(X'_1) \neq Atom(X'_2) \vee Atom(X_1) = Atom(X_2)$.

The second axiom is 0 definition:

Definition 2.17 (Ar2). $0 = X_1 \Leftrightarrow \forall X_2 X'_2 \neq X_1$.

It means there is no class preceded 0.

Now it is possible to give a recursive definition of the *atom*:

Definition 2.18. $Atom(X_1) \Leftrightarrow X_1 = 0 \vee \exists X_2 Atom(X_2) \wedge X_1 = X'_2$.

As it follows from this definition:

- 0 is an atom;
- if X_2 is an atom, then X'_2 is an atom too;
- other atoms do not exist.

The third axiom of arithmetic put up the order into the class of atoms:

Axiom 2.19 (Ar3). $\neg Atom(X_1) \vee X_1 \leq X'_1$.

The fourth axiom removes all the atoms between X_1 and X'_1 :

Axiom 2.20 (Ar4). $\neg Atom(X_1) \vee \neg Atom(X_2) \vee X'_1 \leq X_2 \vee X_2 \leq X_1$.

As it follows from two last axioms and from transitivity of an order, the order is linear:

$$\neg Atom(X_1) \vee \neg Atom(X_2) \vee X_1 \leq X_2 \vee X_2 \leq X_1.$$

In more detail about the axioms of arithmetic see [4].

We should add several axioms to the axioms of arithmetic.

Axiom 2.21 (Ar5). There is a class, which members are the atoms and only atoms:

$$\exists X_1 \forall X_2 Atom(X_2) \Leftrightarrow X_2 \in X_1.$$

So we have the definition of the class *N* as the *class of all atoms* (in arithmetic this class is undefined):

Definition 2.22. $N = X_1 \Leftrightarrow \forall X_2 Atom(X_2) \Leftrightarrow X_2 \in X_1$.

Further we define the relation of *including* for classes.

Definition 2.23. $X_1 \subset X_2 \Leftrightarrow \neg Atomic(X_1) \wedge X_1 \neq X_2 \wedge \forall X_3 X_3 \notin X_1 \vee X_3 \in X_2$.

So the class X_1 is included in the class X_2 , if all members X_1 are members X_2 . The class X_1 is also named subclass X_2 .

As it follows from this definition, if the class X_2 contains *n* members, then a number of its subclasses equals $2^n - 2$, since the empty class and (for symmetry) class X_2 are removed from subclasses X_2 .

Definition 2.24. $P(X_1) = X_2 \Leftrightarrow \neg Atomic(X_1) \wedge \forall X_3 X_3 \subset X_1 \Leftrightarrow X_3 \in X_2$.

The class X_2 of *all subsets* of the class X_1 is defined. The class X_2 is also named *power*.

The class X_1 should not be atomic, since if this condition is removed, then $P(X_1) = X_2$ will be true for any atomic classes X_1 and X_2 .

Definition 2.25. $X_1 \times X_2 = X_3 \Leftrightarrow \neg Atomic(X_1) \wedge \neg Atomic(X_2) \wedge \forall X_4 Pr(X_4) \vee (X_4 \in X_3 \Leftrightarrow \exists X_5 \exists X_6 X_4 = \langle X_5, X_6 \rangle \wedge X_5 \in X_1 \wedge X_6 \in X_2)$.

This is the definition of the *direct product*.

Further we shall specify the notion of a proper class and set, as well as the notion of the atomic class.

Definition 2.26. $Pr(X_1) \Leftrightarrow$

$$X_1 = N \vee (\exists X_2 Pr(X_2) \wedge X_1 = P(X_2)) \vee \exists X_2 \exists X_3 Pr(X_2) \wedge Pr(X_3) \wedge X_1 = X_2 \times X_3.$$

This is the additional definition of the *proper* classes. The proper class is:

N ;

a power class of a proper class;

a class of a direct product of proper classes.

Other proper classes do not exist.

Theorem 2.27. *The proper classes have no common members:*

$$\neg Pr(X_1) \vee \neg Pr(X_2) \vee X_3 \notin X_1 \vee X_3 \notin X_2 \vee X_1 = X_2.$$

Proof: We shall use the rule of the generalized induction (see section 3.2.7).

According to this rule some statement is deduced, if the following conditions take a place:

- the relation used in the statement is defined recursively (in this case - Pr);

- the statement is deduced before the beginning of the recursion;

- from a deducibility of the statement for some stage of recursion we have the deducibility for the following stage of recursion.

The difference from the usual rule of induction is that a step of recursion is substituted by a stage, i.e., a group of steps.

For example, at the first stage of a recursion we received $P(N)$ and $N \times N$, at the second stage - $P(P(N))$, $P(N \times N)$, $N \times P(N)$, $P(N) \times N$, $N \times N \times N$, $P(N) \times N \times N$ and $N \times N \times P(N)$.

The given theorem is a valid formula at $X_1 = X_2 = N$, i.e. before the beginning of the recursion.

Let all proper classes have no common members at a stage n . Let X_1 and X_2 ($X_1 \neq X_2$) be proper classes, one of these proper classes (for example, X_1) is received at a stage $n + 1$, the other one - at a stage $\leq n$. We shall show, these classes have no common members too.

Let X_1 and X_2 have a common member X_3 and be received at the stage $n + 1$. We shall prove, this assumption leads to inconsistency.

It is clear, X_3 is not an atom.

If X_3 is a set, then there are proper classes X_4 and X_5 , received at the stage n and X_3 is the subset of every of them. It means, both classes X_4 and X_5 have all members of X_3 as common members. This contradicts to the given conditions.

If X_3 is an n -tuple, then X_1 is a direct product of X_{1i} and X_2 is direct product of X_{2i} , $1 \leq i \leq m$. Every X_{1i} and every X_{2i} are N or a power set.

If X_{1j} is N , then X_{2j} is N too. So $X_{1j} = X_{2j}$. If X_{1k} is a power class, then X_{2k} is a power class too and both classes are received at a stage $\leq n$. But these classes have a common member and again it contradicts to the given conditions.

But X_1 can be received at the stage $n + 1$ and X_2 can be received at a stage $\leq n$. Repeating the same arguments we shall come to the same inconsistency. The proof is complete. \square

Corollary 2.28. $M(X_1) \Leftrightarrow \exists X_2 Pr(X_2) \wedge X_1 \subset X_2$

So any set is a subclass of only one proper class. Other sets do not exist.

The proved theorem allows to expand the domain of the successor function.

Definition 2.29. $X_0 \neq \langle X_1, \dots, X_n \rangle \vee (X_0' = X_{n+1} \Leftrightarrow Pr(X_{n+1}) \wedge X_0 \in X_{n+1})$.

The *successor function of an n -tuple* is a proper class, including this n -tuple.

Definition 2.30. $\neg M(X_1) \vee (X_1' = X_2 \Leftrightarrow Pr(X_2) \wedge X_1 \subset X_2)$.

The *successor function of a set* is a proper class, including this set.

Definition 2.31. $\neg Pr(X_1) \vee X_1' = X_1$.

The *successor function of a proper class* is this proper class.

So the successor function becomes defined everywhere (existence axiom 2.43 ($Ar0$) for this function is given below).

Theorem 2.32. *If X_2 is a member of the proper class X_1 , then X_2 and X_1 have no common member:*

$$\neg Pr(X_1) \vee X_2 \notin X_1 \vee X_3 \notin X_2 X_3 \notin X_1.$$

Proof: Again we use the rule of the generalized induction.

For $X_1 = N$ the theorem is true.

Let the theorem be true at a stage n and let the proper class X_1 be received at a recursion stage $n + 1$. We must show, any member X_1 has no common member with X_1 .

It is true, if X_1 is a direct product of proper classes.

Let X_1 be power of the proper class X_4 received at the recursion stage n . And let X_1 and its member X_2 have the common member X_3 . We shall prove, this assumption goes to the inconsistency.

It follows from the given assumption,

$$X_2 \subset X_4 \quad X_3 \subset X_4 \quad X_3 \in X_2 \quad X_3 \in X_4.$$

Therefore, X_3 is a member of X_4 and all members X_3 belong to X_4 . But this contradicts the initial assumption. The proof is complete. \square

Corollary 2.33. $X_1 \notin X_2 \vee X_2 \notin X_3 \vee X_1 \notin X_3$.

This corollary has three wordings:

- the membership relation is non-transitive, i.e. from $X_1 \in X_2$ and $X_2 \in X_3$ we can not deduce $X_1 \in X_3$;

- the class X_3 and its member X_2 have no common member X_1 ;

- the class X_2 and its member X_1 can not be members of the other class X_3 .

So the statements "a set is non-transitive" and "a set and its member have no common members" are the same.

The given corollary is deduced from the theorem. Otherwise the proper class, which subset is X_3 , and a member X_2 of this proper class would have a common member X_1 .

According to the axiom of regularity, in the classic set theory there is a member of a class and this member does not contain common members with this class. In the given corollary existence is replaced by universality.

Corollary 2.34. $\forall X_1 \neg (\forall X_2 X_2 \notin X_1 \vee X_2 \subset X_1)$.

The transitive sets do not exist. Otherwise this set and its member would have common members.

Theorem 2.35. *The class can not contain n -tuple with an element to equal this class:*

$$\langle \dots, X_1, \dots \rangle \notin X_1.$$

The theorem has other wording - no proper class X_2 contains X_1 with a member $\langle \dots, X_1, \dots \rangle$.

Proof: We use again the rule of generalized induction.

For $X_2 = N$ the theorem is obvious.

Let the theorem be true at a recursion stage n . We must prove, the theorem is true at a stage $n + 1$.

Let the proper class X_2 be received on a recursion stage $n + 1$. Also let the class X_2 contains X_1 with the member $\langle \dots, X_1, \dots \rangle$. We shall show, these assumptions goes to the inconsistency.

There is a proper class X_3 and $X_2 = P(X_3)$. This class is created on the stage n and X_1 is its subset. Therefore, all members X_3 are n -tuples and one of the n -tuples is $\langle \dots, X_1, \dots \rangle$. It means $X_3 = \dots \times X_4 \times \dots$ where X_4 is a proper class of previous stages.

The class X_4 contains the member X_1 , but X_1 contains the member $\langle \dots, X_1, \dots \rangle$. We have inconsistency: the proper class, created on the stage below n , contains X_1 with a member $\langle \dots, X_1, \dots \rangle$. The proof is complete. \square

Theorem 2.36. *Infinite descending \in -sequence does not exist:*

$$\neg (\dots \wedge X_3 \in X_2 \wedge X_2 \in X_1).$$

It is an analog of the funded theorem.

Proof: This sequence can not be a proper class. It is deduced from the negative definition of proper classes. So this sequence can not be a set or atomic class at all. \square

Theorem 2.37. *Finite \in -cycle does not exist:*

$$\neg (X_1 \in X_2 \wedge X_2 \in X_3 \wedge \dots \wedge X_k \in X_1).$$

Proof: It is proved by generalized induction. \square

We go on definitions.

Definition 2.38. $X_1 \cap X_2 = X_3 \Leftrightarrow \forall X_4 X_4 \in X_3 \Leftrightarrow X_4 \in X_1 \wedge X_4 \in X_2$.

The *intersection* of classes is defined. The existence of this operation follows from axiom 2.46 (B2, see below).

Definition 2.39. $\overline{X}_1 = X_2 \Leftrightarrow \forall X_3 X_3 \notin X_1' \vee (X_3 \in X_2 \Leftrightarrow X_3 \notin X_1)$

The class X_2 is a *complement* of X_1 , if both classes have no common members, and every member of proper class included X_1 belongs to one of the classes X_1 or X_2 . The existence of this operation is set by the axiom 2.47 (B3, see below).

The defined complement has all the same properties, as classic defined. In particular we have $\overline{\overline{X}_1} = X_1$.

Definition 2.40. $X_1 \cup X_2 = \overline{\overline{X_1} \cap \overline{X_2}}$.

The *union* of classes is defined as a complement of intersection of complements of these classes.

Definition 2.41. $\{X_1, X_2\} = X_3 \Leftrightarrow \forall X_4 Pr(X_4) \vee (X_4 \in X_3 \Leftrightarrow X_4 = X_1 \vee X_4 = X_2)$.

The *unordered pair* is defined.

Axiom 2.42 (P). $\forall X_1 \forall X_2 \exists X_3 (Atom(X_1) \wedge Atom(X_2) \vee X_1' = X_2' \wedge \neg Pr(X_1) \wedge \neg Pr(X_2)) \rightarrow \{X_1, X_2\} = X_3$.

It is the existence axiom for the unordered pair. The pair arguments can be either atoms, n -tuples from the same proper class, or sets from the same proper class.

If both arguments of an unordered pair are proper classes, the unordered pair is not a function: any atomic class becomes the value of the “function”. If only one argument is a proper class, this argument becomes fiction (see theorem 2.7).

So the existence axiom is true not for all values of arguments.

Axiom 2.43 (Ar0). $\forall X_0 \forall X_1 \exists X_2 Pr(X_0) \wedge (X_1 \in X_0 \vee X_1 = X_0) \rightarrow X_2 = X_1'$.

The existence axiom for the successor function. This function becomes a collection of functions. Each member of this collection has its own proper class X_0 as the range of definition. The general successor function has no class to be a range of definition.

Note a proper class can be a member of range of definition. But such a member can be removed from the range of definition (see theorem 2.7).

Axiom 2.44 (B0). $\forall X_1 \forall X_2 \forall X_3 \forall X_4 \exists X_5 Pr(X_1) \wedge Pr(X_2) \wedge (X_3 \in X_1 \vee X_3 = X_1) \wedge (X_4 \in X_2 \vee X_4 = X_2) \rightarrow X_5 = \langle X_3, X_4 \rangle$.

It is the existence axiom of the ordered pair. This function is also a collection of functions and any member of the collection has $X_1 \times X_2$ to be a range of definition.

Axiom 2.45 (B1). $\forall X_0 \exists X_1 \forall X_2 \forall X_3 Pr(X_0) \wedge (X_3 \in P(X_0) \vee X_3 = X_0) \wedge X_2 \in X_3 \rightarrow \langle X_2, X_3 \rangle \in X_1$.

The existence of \in -relation is stated for each proper class X_0 . For a given X_0 \in -relation has $X_0 \times P(X_0)$ as the range of definition.

Axiom 2.46 (B2). $\forall X_1 \forall X_2 \exists X_3 (\exists X_4 X_4 \in X_1 \wedge X_4 \in X_2) \rightarrow X_3 = X_1 \cap X_2$

The operation “ \cap ” exists, if the intersection of X_1 and X_2 is not empty.

Axiom 2.47 (B3). $\forall X_1 \exists X_2 M(X_1) \rightarrow X_2 = \overline{X_1}$.

The complement exists for any set.

Axiom 2.48 (B4). $\forall X_1 (\exists X_2 \exists X_3 \neg Pr(X_2) \wedge \langle X_2, X_3 \rangle \in X_1) \rightarrow \exists X_2 \forall X_3 \neg Pr(X_3) \vee (X_3 \in X_2 \Leftrightarrow \exists X_4 \langle X_3, X_4 \rangle \in X_1)$.

The range of definition exists for all classes, if members of these classes are the ordered pairs.

Axiom 2.49 (B5). $\forall X_0 \forall X_1 \exists X_2 \forall X_3 \forall X_4 Pr(X_0) \wedge \neg Atomic(X_1) \wedge X_3 \in X_1 \rightarrow (\langle X_3, X_4 \rangle \in X_2 \Leftrightarrow X_4 \in X_0)$.

There exists a class X_2 of ordered pairs with the second fictive elements. Each member of non-atomic class X_1 is the first element of this ordered pairs.

Axiom 2.50 (B6). $\forall X_1 (\exists X_2 \exists X_3 \exists X_4 \langle X_2, X_3, X_4 \rangle \in X_1) \rightarrow \exists X_2 \forall X_3 \forall X_4 \forall X_5 (\langle X_3, X_4, X_5 \rangle \in X_2 \Leftrightarrow \langle X_4, X_5, X_3 \rangle \in X_1)$.

The cyclical permutation of the members in triples exists for any class X_1 , if members of this class are triples.

Axiom 2.51 (B7). $\forall X_1 (\exists X_2 \exists X_3 \exists X_4 \langle X_2, X_3, X_4 \rangle \in X_1) \rightarrow \exists X_2 \forall X_3 \forall X_4 \forall X_5 (\langle X_3, X_4, X_5 \rangle \in X_2 \Leftrightarrow \langle X_3, X_5, X_4 \rangle \in X_1)$.

The permutation of two last members in triples exists for all classes, if members of these classes are triples.

On it the construction of the theory AS is finished.

We shall apply this theory to model construction.

In the relational logic the set of the models is very limited, so some theories have no model, for example, the classic set theory. Such theories in the relational logic are inconsistent.

The consistent theory should have one or more models.

Definition 2.52. The theory is *arithmetic*, if it has the unique model (except isomorphic). The theory is *algebraic*, if it has several non-isomorphic models.

The algebraic theories require the extension of the theory AS for including collections of the models. This extension is realized by introduction of algebraic classes.

We recall, all considered till now classes are arithmetic, since the theory AS is theory of arithmetic sets and classes.

Definition 2.53. The *algebraic class* is a collection of arithmetic classes, if this algebraic class is not arithmetic.

So the algebraic class can contain proper classes as (nonempty) members (the arithmetic class can contain proper classes only as empty members, see above).

Definition 2.54. The *universal class of order 0* is an algebraic class with every arithmetic class as a member. The *universal class of order 1* is a collection of every arithmetic and algebraic classes as a member (because it is a universal class).

The universal class of order 0 met by above as a collection of arithmetic classes X . All models of arithmetic theories belong to this class.

The universal class of order 1 has all collections of the models. Every algebraic theory has one of these collections.

If we investigate models of a set of algebraic theories, we need the universal class of order 2. This class has all arithmetic and algebraic classes and all collections of algebraic classes. All models of every set of algebraic theories belong to this class.

There are universal classes of order 3 and more, but the necessity of their practical usage is doubtful.

Thus, beginning from non-transitive sets we can construct universal classes of the unrestricted power. But these classes have the other very strict limitations such as in the typed theory.

The collection of universal classes of all orders (finite and infinite) forms the natural set theory.

In conclusion we shall mark, the basic defect of the theory AS is next: though the complement to a set is a set (in contrast to the classic theory), but it is bad for interpretation of negation of a relation. This defect is not removed even by powerful means of classic logic because negation of relations used domains instead of proper classes. But these domains are not proper classes as a rule.

3. Finite Logic.

The finite logic is a science of constructing an axiom set of theories and their models of potential infinity.

The finite logic is the relational programming. Basic problem of this logic is a computer construction of theories (knowledge bases) and their models (data bases). It is supposed, every knowledge can be present as an axiom. If these axioms are in the first normal form, they are a program, ready for execution on computers to construct theories. If these axiom are definitions and present in the second normal form, they are a program, ready for execution on computers to construct models.

Section 3.1 is dedicated to model construction by iteration rule.

Section 3.2 is dedicated to theory construction by resolution, paramodulation and finite descent rules.

3.1. Model Construction (Logic Programming).

The model construction for a theory is a model construction of all relations of the theory. The models of undefined relations are given sets, the models of defined relations are calculated with the help of positive definitions of the relations.

Calculation of an explicit defined relation is simple. Calculation of a recursive defined relation is iterative.

Before the beginning of an iteration we should have an initial set of n -tuples for calculated relation. The set can be empty. At the first iteration this set is used for calculation of the right part of the relation definition. The calculation result is joined with the initial set and becomes a new set. This new set is used for calculation of the right part of the relation definition at the second iteration, the calculation result is joined with this set. The resulting set is again used for calculation of the right part of the definition at the third iteration etc. Calculations are finished, if the fixed point is reached, i.e. when other iterations do not change the calculated set.

The recursive relation calculation is alike the recursive function calculation. But the recursive relation definitions have negations and iterations become more complex. There are cases when iterations

become impossible for potential infinite sets.

The investigation of such cases is one of the basic problems for the finite logic, but below these problems are not considered.

Below we shall suppose, all formulas (axioms) of theories are in the first normal form and all formulas (definitions) of models are in the second normal form.

3.1.1. Logic and Set Theory.

We use either logic or set theory notation for a theory or model construction.

Usage of the logic notation we shall name as the logic way, usage of the set theory notation - as the set theory way.

These two ways seem to be one-to-one. For example:

Set theory way	Logic way
$X_1 \cap X_2$	$X_1(x_1) \wedge X_2(x_1)$
$X_1 \cup X_2$	$X_1(x_1) \vee X_2(x_1)$
$\overline{X_1}$	$\neg X_1(x_1)$.

Below it will be shown, so simple dependence is absent.

Let us formulate the ground rules of the logic way.

Rule 3.1. Interpretation of a variable sort is the range of corresponding variables.

Below this set we shall call the *domain* or the domain of the variable. The *place domain* is a domain of variable put in a place of a relation. The *relation domain* is a direct product of place domains for all places in the relation.

Rule 3.2. The interpretation of the relation $R_1(X_1, \dots, X_n)$ is the set R_1 of n -tuples $\langle X_1, \dots, X_n \rangle$:

$$R_1(X_1, \dots, X_n) \rightarrow \langle X_1, \dots, X_n \rangle \in R_1.$$

We use the same identifications for the relation and set in hope, this does not become a source of misunderstanding.

The interpretation of conjunctions will be realized by the several rules.

If we have

$$R_1(X_1, \dots, X_n) \wedge R_2(X_1, \dots, X_n)$$

interpretation of this conjunction is an intersection of sets

$$R_1 \cap R_2.$$

But the interpretation of the conjunction

$$R_1(X_1, \dots, X_n) \wedge R_2(X_{n+1}, \dots, X_{2n}) \quad (1)$$

is completely unexpected. It is interpreted by direct product of sets:

$$R_1 \times R_2.$$

In set theory there is no simple means for interpretation of the conjunction. Usage of set identifications without arguments limits possibilities of this theory very much.

In the relational logic there is the following rule for conjunctions and disjunctions:

Rule 3.3. The conjunction (disjunction) of relations is always interpreted as intersection (joint) of sets, but the number of places and terms in these places must be the same for both relations.

The reduction to such form is regulated by the next rule:

Rule 3.4. Before interpretation the number of places and their fillings must be equalized in relations at conjunction and disjunction. Such equalization is realized by transpositions of places and addition of fictive places.

In particular, the equalization of the formula (1) is:

$$R_1(X_1, \dots, X_{2n}) \wedge R_2(X_1, \dots, X_{2n}).$$

Rule 3.5. The addition of a fictive place in the relation is interpreted as a direct product of a set, corresponding to this relation, and a domain, corresponding to added argument.

Therefore, the formula (1) is interpreted as a direct product of R_1 and R_2 .

The interpretation of the disjunction

$$R_1(X_1, \dots, X_n) \vee R_2(X_{n+1}, \dots, X_{2n}) \quad (2)$$

is more unexpected: if the interpretations of R_1 and R_2 are finite, but their domains are infinite, the interpretation of the disjunction is infinite too.

It is a union of two sets:

- a direct product of R_1 and the relation domain for R_2 ,
- a direct product of R_2 and the relation domain for R_1 .

Rule 3.6. The interpretation for negation of a relation is a complement of the interpretation of the relation. The complement is defined on the domain of the relation.

Further we shall define interpretation for existential and universal quantifiers. It is supposed, all necessary intersections, unions and complements of sets are made for the relations in scope of quantifiers. It means, the scope of quantifiers becomes one relation. This relation corresponds to a resultant set.

Rule 3.7. The existential quantifier is interpreted by a set. In a source set the element, corresponding to the bound variable, is removed from every n -tuple.

Rule 3.8. The universal quantifier is interpreted by a set too. In a source set:

- every group of n -tuples is removed, if n -tuples in the group differ one from others only by values of bound variables and if sets of these values are not equal to the domains of these variables;
- in remaining n -tuples all elements of the bound variables are removed.

We use the second normal form to construct models. The basic part of the form is back implication. The interpretation of back implication is special.

Rule 3.9. The back implication has the next interpretation:

- the right part of the back implication is interpreted by the set obtaining after execution of conjunctions, negations and existential quantifiers in this part (variables, met only in the right part, are bound by implicit existential quantifiers);
- before calculation, the left part of back implication is interpreted by the corresponding set;
- after calculation the set of the left part joins the set of the right part, and this result is the interpretation of back implication.

The important part of formulas are terms.

Rule 3.10. The terms are interpreted in accordance with their definition:

$$R_1(\dots, R_0(\dots, -, \dots), \dots) \Leftrightarrow \exists S_{0_0} R_1(\dots, S_{0_0}, \dots) \wedge R_0(\dots, S_{0_0}, \dots).$$

where R_0 is a term.

Example 3.11. The relation $R_1(x_1, x_2, x_1 + x_2)$ defines as $\exists x_0 R_1(x_1, x_2, x_0) \wedge +(x_1, x_2, x_0)$. The conjunction of the relations R_1 and “+” is interpreted by the set R_1 after removing n -tuples with the third element does not equal the sum of the first two elements. Then the third elements are removed too.

If $R_1 = \{ \langle 0, 1, 1 \rangle, \langle 1, 2, 2 \rangle \}$, interpretation of the relation $R_1(x_1, x_2, x_1 + x_2)$ is the set $\{ \langle 0, 1 \rangle \}$. \square

The special rules exist for new (defined) sorts.

Rule 3.12. The relation, corresponding to a new sort, has one or more arguments of old sorts and one argument of new sort. All these arguments put up connection between them.

Example 3.13. The relation $C(R_1, R_2, C_1)$ is the sort C of complex numbers, R_1 and R_2 are variables of the old sort R of real numbers, C_1 is a variable of the new sort C . The relation C is defined by the formula:

$$C(R_1, R_2,)$$

In this definition we have an anonymous variable. This definition means, complex numbers are defined for all values of R_1 and R_2 .

Now we can define addition of complex numbers:

$$C_1 + C_2 = C_3 \Leftrightarrow \forall R_1 \dots \forall R_4 C(R_1, R_2, C_1) \wedge C(R_3, R_4, C_2) \wedge C(R_1 + R_3, R_2 + R_4, C_3)$$

In this formula the relation C puts up connection between complex and real variables. \square

Definition 3.14. The *power sort* is an $(n + 1)$ -place relation setting one-to-one for every sequence of n sorts and the new sort.

If the sequence of sorts is interpreted as domain of any relation, the power sort is interpreted as the set of all subsets of this domain. Since each such subset interprets as some relation, the power sort contains all relations on its domain.

The power sort is a powerful tool to construct new sorts in the logic order 2 and higher.

Some rules define interpretation of the theories.

Rule 3.15. Each theory is the relation interpreted as the set of this theory axioms (both independent and dependent).

Rule 3.16. The axiom, subsumed by the other axiom, is removed from the set of the axioms.

Rule 3.17. The axiom of a theory, deduced without usage of the other axioms of the theory, is a valid formula and must be removed.

From above it follows, the logic way is more preferable than the set theory way, since it grants more powerful expressive means, than the latter. The theory of sets should be used only for interpretations of logical formulas.

3.1.2. Formulas without Negations. Iteration stages.

We construct models on the basis of a set of the formulas. At constructing we use the special rules and these rules are axioms of the first order logic.

These rules are essentially simplified, if negation is absent in the right part of the formulas.

If the formulas are recursive, the rules of construction of the models in logic programming will be realized by iteration, i.e. as a sequence of stages and every stage has a group of steps. The step is a one-time calculation by one formula, the stage is one-time calculation of all formulas.

Rule 3.18. Iteration of formulas without negation:

- before the beginning to construct the model, initial subsets of constructed sets should be present;
- at the first stage all formulas are executed one time and the sets, containing the initial subsets, join the result of execution;
- at the following stages all formulas again are one-time executed and the new sets, containing the sets of previous stage, join the result;
- the process of iteration ends, if the fixed point is reached, i.e. the further stages do not add new members in the constructed sets.

The given rule can be expanded into infinite sets, if the iteration is convergent to a fixed point. This fixed point is the enumerable set of all values of the calculated relation, and any member of this set can be calculated for a finite number of steps of iteration.

Example 3.19. It is required to calculate the factorial.

The calculation of a factorial is simplified, since each stage of iteration contains only one step.

Before the beginning of calculations we have a subset with a unique member $!(0, 1)$.

On the first step we receive one more member $!(1, 1)$ from the formula $(N_1 + 1)! = N_1! * (N_1 + 1)$.

On the second step, using the member $!(1, 1)$, we receive the third member $!(2, 2)$.

Each new step adds one member. The iteration is convergent.

The difference of the given iterative process from the similar process in usual programming is in a result of calculation: in the first case this result is a set of numbers, and in the second case it is only one number. Total time of calculations in both cases is the same. \square

The defect of the formulated rule of iteration is the next: the result of one stage of iteration depends on a sequence of execution of the formulas at this stage. If the fixed point is unique (for the formulas without negation it always is unique), the result will be identical for any sequence of execution.

Nevertheless, it is very important that the result of each stage of iteration did not depend on a sequence of execution of the formulas. It is possible, if at each stage at first to calculate (one-time) the right part of all formulas, and then to execute back implication.

Example 3.20. The positive definition of proper classes (see section 2) is:

$$\begin{aligned} Pr(N) \\ Pr(P(X_1) \leftarrow Pr(X_1)) \\ Pr(X_1 \times X_2) \leftarrow Pr(X_1) \wedge Pr(X_2) \end{aligned}$$

where Pr - symbol of the relation to be a proper class, P - symbol power set, " \times " - symbol of a direct product.

Before the beginning of iteration the set Pr contains one member N .

The result of execution of the first iteration stage, consisting of two steps, depends on a sequence of execution of two last formulas.

If executing the first of these two formulas, then the second, we shall receive the following result.

Step 1. $\{N, P(N)\}$.

Step 2. $\{N, P(N), N \times N, N \times P(N), P(N) \times N, P(N) \times P(N)\}$.

Executing the second formula, then the first, we shall receive the different result.

Step 1. $\{N, N \times N\}$.

Step 2. $\{N, N \times N, P(N), P(N \times N)\}$.

We shall receive a result not depending on a sequence of execution of the formulas at separate execution of the right and left parts of the formulas. But a number of steps doubles because of back implications:

Step 1. $\{P(N)\}$.

Step 2. $\{N \times N\}$.

Step 3. $\{N, P(N)\}$.

Step 4. $\{N, P(N), N \times N\}$.

All of these steps are simple and we can replace them by one stage:

Stage 1. $\{N, P(N), N \times N\}$. \square

In the next section we shall show, the calculation of the left parts of the formulas at the end of a stage is the only possible way.

3.1.3. Ghost Effect.

The existence of negations in the formulas has a number of features.

First of all we shall consider formulas with the right part including relations both with and without of negations.

Rule 3.21. Calculation with negations:

- conjunction of all relations without negations is calculated;
- conjunction of this calculated relation and the first relation with negation is calculated (by the rule of a set difference);
- conjunction of the latter calculated relation and the next relation with negation is calculated (by the rule of a set difference);
- the previous step is repeated for all relations with negation.

The rule of a set difference allows to calculate the relation with negation without construction of a complement.

Rule 3.22. Usage of a set difference:

- the conjunction of two relations, if one of these relations is without negation and the other with negation, is interpreted as a difference of interpretations of these relations, if a number of places and fillings of the places are identical in both relations.

If they are not identical, transpositions of arguments, and additions of fictive arguments are realized.

If the right part of a formula contains only relations with negation, the calculation of a complement is obligatory for one of the relations. For the remaining relations the rule of a set difference is again used.

Another feature of the formulas with negations is the ghost effect.

The ghost effect means dependence of the result of calculations from a sequence of these calculations.

Theoretically the ghost effect should be absent, because construction of the model should not depend on sequence of calculations. Nevertheless many theories of algorithms (by Markov, by Turing) are based on the dependence on the sequence of calculations. Only the theory of recursive functions is an exception. But logic programming must be an exception too.

In logic programming all formulas can be jointed in one formula with the help of conjunctions. But conjunction is commutative, therefore, the result of calculations should not depend on what formula is calculated first and what is last.

This rule is completely true for formulas without negations. If there are negations, the given rule is not always true, as this follows from the example below (the symbol “'” is used as an marker, not as a function).

Example 3.23. $a'(N_1) \leftarrow \neg a''(N_1) \wedge b'(N_1)$.
 $a''(N_1) \leftarrow \neg a'(N_1) \wedge b'(N_1)$.

Let sets a' and a'' be empty. The sets b' and b'' are not empty and are not changed during calculations.

After execution of the first formula, a' equals b' and a'' is empty. After execution of the second formula we have $a'' = b'' \setminus b'$ (" \setminus " - symbol of a set difference, the set difference is basic operation for calculation of negations without construction of complements, as it is mentioned above).

After re-execution of the first formula in a' will be added $b' \setminus b''$. This addition does not change a' and again $a' = b'$. Since a' was not changed, re-execution of the second formula changes nothing too. The fixed point is reached: $a' = b'$, $a'' = b'' \setminus b'$.

If we change a sequence of execution of the formulas and begin with the second formula, we shall receive the other fixed point: $a' = b' \setminus b''$, $a'' = b''$.

One more result we shall receive, if at first we execute the right parts of both formulas (in any sequence), and only after this we execute back implication.

After executing the first right part, we shall receive the set b' . After executing of the second right part we shall receive the set b'' . Both sets a' and a'' remain empty.

After executing the back implication, we shall receive $a' = b'$, $a'' = b''$.

At re-execution of the first right part we shall receive set $b' \setminus b''$. After execution of the second right part we shall receive set $b'' \setminus b'$.

The execution of back implication changes nothing. We have reached one more fixed point $a' = b'$, $a'' = b''$.

From the three results only the last is correct, because in this case the result of calculations does not depend on the sequence of formula execution. At calculation of the right parts the set a' and a'' do not vary, so they do not influence on the sequence of calculations. \square

Note, in the given example a' is defined through a'' , a'' is defined through a' . It can be, if the definitions are correct.

The initial definitions are really correct.

The next definitions are not correct:

$$\begin{aligned} a'(N_1) &\leftarrow \neg a''(N_1). \\ a''(N_1) &\leftarrow \neg a'(N_1). \end{aligned}$$

since both formulas coincide at reduction to CNF:

$$\begin{aligned} a'(N_1) \vee a''(N_1), \\ a''(N_1) \vee a'(N_1). \end{aligned}$$

But if the formula is one, definition should be one too. Therefore, one of these definitions should be removed, and the result of calculations can depend on what definition is removed.

The next definitions are not correct too:

$$\begin{aligned} a'(N_1) &\leftarrow \neg a''(N_1) \wedge b'(N_1). \\ a''(N_1) &\leftarrow \neg a'(N_1) \wedge b'(N_1). \end{aligned}$$

Both formulas coincide at reduction to CNF too:

$$\begin{aligned} a'(N_1) \vee a''(N_1) \vee \neg b'(N_1). \\ a''(N_1) \vee a'(N_1) \vee \neg b'(N_1). \end{aligned}$$

It is necessary to mark, in the given example we have used the monotonically increasing calculations (see below), though it is more correct to use non-monotone calculations (see example 3.31). Nevertheless, the results of calculations in both cases are the same. \square

If the definitions are correct, the next rule of iteration does not contain the ghost effect.

Rule 3.24. Iterations without the ghost effect:

- before the beginning of iterations initial sets must be given for each calculated set;
- at the first stage of iteration all the right parts of the formulas are calculated, then the back implications are calculated, as a result we have the new sets containing the initial sets;
- on the next stages all the right parts are again calculated, then the back implications are calculated too, and we have the new sets containing the previous sets;
- the process of calculation ends, if the fixed point is reached, i.e. all further stages do not add new members in calculated sets.

Below all iterations are executed without the ghost effect, and the step of iteration always means as a stage of iteration (but for one level of hierarchy, see next section).

3.1.4. Hierarchy of Formulas.

In logic programming calculations are effective, if spent time is minimal.

Effective calculations are the main problem of logic programming, because the calculations are the set operations and the more volume of these sets the greater spent time. Especially the time is great spent, if the sets are unordered.

If we set the identical order at calculation of intersections (joints) of sets, the time of calculations becomes minimal.

At more complex calculations, for example with the graphs, the whole series of ordering is used.

The efficiency of calculations increases not only at ordering of members of sets, but also at ordering of the formulas. Naturally, not any ordering is allowable. In particular, the ordering with ghost effect is forbidden.

Let us consider the basic rules of ordering.

Rule 3.25. The order of formulas without cycles:

- if R_i is a name (identification) of a relation in the left part of the formula, and R_j, R_k, \dots are names of the relations in the right part, we construct arcs $R_i \leftarrow R_j, R_i \leftarrow R_k, \dots$ (these arcs are interpreted as: " R_j is calculated before R_i, R_k is calculated before R_i, \dots ");

- if the result graph contains cycles, this rule is not used;
- the names, not meeting in the left part of the formulas, are removed;
- the order of the other names becomes the order of the formulas.

Since the formula set is finite this order allows to put hierarchy of the formulas.

After putting up hierarchy we calculate at beginning formulas of the first (upper) level of hierarchy, then the received results are used at calculation of the formulas of the second level, etc. And the calculation of the formulas ($i + 1$)-th level do not influence on the calculation of the formulas (i)-th level.

This hierarchy allows to put up linear preorder. At this order a number of the equivalence classes is equal to a number of hierarchy levels.

Unfortunately, such hierarchy of the formulas has a many problems. In particular, in this case all recursive definitions have no hierarchy.

Because the basic complication in calculations is introduced by negations, the special preorder can be set. This preorder forbids cycles only for relation with negation.

Rule 3.26. The order of formulas without cycles with marked arcs:

- the arcs $R_i \leftarrow R_j$, where R_j is the name of the relation in the right part of the formula, R_i is the name of the relation in the left part of the formula, are constructed for all formulas;
- if R_j has negation, the arc $R_i \leftarrow R_j$ is marked;
- if the result graph contains cycles with the marked arcs, this rule is not used;
- the names presented in a cycle with unmarked arcs, are considered as equivalent;
- the names not meeting in the left part of the formulas are removed;
- the order of the other names becomes the order of the formulas.

This order allows to put up hierarchy of the formulas too, if we put all vertices of the cycle without marked arcs on the same level of hierarchy.

According to this hierarchy, calculation of $\neg A$ is easy because calculation of A was completely finished on the previous level of hierarchy.

Note, both kinds of hierarchies provide the absence of ghost effects.

Example 3.27. We have formulas:

$$\begin{aligned} a(N_1, N_2) &\leftarrow d(N_1, N_2). \\ b(N_1, N_2) &\leftarrow e(N_1, N_2) \wedge \neg a(N_1, N_2). \\ c(N_1, N_2) &\leftarrow e(N_1, N_2) \wedge \neg b(N_1, N_2) \wedge \neg a(N_2, N_1). \end{aligned}$$

Before the beginning of iteration the sets a, b and c are empty.

The set of arcs $\{a \leftarrow d, b \leftarrow e, b \leftarrow a, c \leftarrow e, c \leftarrow b, c \leftarrow a\}$ does not contain a cycle. So we have hierarchy of names $c \leftarrow b \leftarrow a \leftarrow (d, e)$. After removing d and e we shall receive $c \leftarrow b \leftarrow a$. Therefore, on top level of hierarchy we have a (according to the first formula), then on the next level we have b (the second formula) and then c (the third formula). As a result we have $a = d$ (from the

first formula), $b = e \setminus d$ (from the second formula), $c = (e \cap d) \setminus I(d)$ (from the third formula). The function $I(d)$ realizes transposition of places in d . \square

3.1.5. Non-monotone Calculations.

In the previous rules of iteration all calculations were monotonically increasing, i.e. at each stage of iteration we added members in calculated sets.

The monotony is broken, if the graph of the relation names contains cycle with marked arcs.

In this case we must calculate complements to the sets, before these sets are constructed to the end. Such complements contain superfluous members and this members fall in calculated sets. Later on, step-by-step, superfluous members will be found and removed. But we have not a monotony of calculations.

It is possible to construct hierarchy of the formulas for non-monotone calculations too.

Rule 3.28. The order of arbitrary formulas:

- the arcs $R_i \leftarrow R_j$, where R_j - name of the relation in the right part of the formula, R_i - name of the relation in the left part of the formula, are constructed for all formulas;
- the result graph sets hierarchy of names, and all vertices, included in a cycle, present on the same level of hierarchy;
- the names not meeting in the left part of the formulas are removed;
- the hierarchy of the other names becomes hierarchy of the formulas.

Such hierarchy exists for all problems. The calculations are begun at the first (upper) level of the hierarchy. All relations, calculated at the given level, remain invariable at execution of all levels below. The ghost effect is absent, if it is absent on each level of the hierarchy.

In general case these calculations can not be executed, if we use two-valued logic. This logic can not find superfluous members in calculated sets. But these calculations can be executed, if we use standard 4-valued logic. These values are t (true), tf (as a rule it is true, but possible it is false), ft (as a rule it is false, but possible it is true) and f (false).

We must add one of these values to each n -tuple of a set as a new element. These values allow to find superfluous members.

But we must add one more value "-". This value means a member, absent in a set (i.e. the member of the complement to this set).

An absent member is false as a rule, but possible it can be true. So the absent member has value ft . It means, values "-" and ft are the same, but the member with value "-" is absent in the set and the member with value ft is present in the set.

The basic operations in this logic are negation, conjunction of positive relations, conjunction of positive and negative relations, disjunction, existential quantifier and back implication.

The tables of logical values for these operations are given below:

\neg	
t	f
tf	ft
ft	tf
f	t
-	tf

\wedge	t	tf	ft	f	-
t	t	tf	ft	f	-
tf	tf	tf	ft	f	-
ft	ft	ft	ft	f	-
f	f	f	f	f	f
-	-	-	-	f	-

$\wedge \neg$	t	tf	ft	f	-
t	f	ft	tf	t	tf
tf	f	ft	tf	tf	tf
ft	f	ft	ft	ft	ft
f	f	f	f	f	f
-	f	-	-	-	-

\exists	t	tf	ft	f	-
t	t	t	t	t	t
tf	t	tf	tf	tf	tf
ft	t	tf	ft	ft	ft
f	t	tf	ft	f	ft
-	t	tf	ft	ft	-

\leftarrow	t	tf	ft	f	-
t	t	t	t	t	t
tf	t	tf	$\frac{ft}{ft}$	$\frac{ft}{ft}$	tf
ft	t	tf	$\frac{ft}{ft}$	$\frac{ft}{ft}$	ft
-	t	tf	ft	ft	-

The first colon in these tables contains the values of the first argument of the operations. The first line in them (except table " \neg ") contains the values of the second argument of the operations.

The first two tables are clear.

The third table presents conjunctions of the positive and negative (with negation) relations. Values of the positive relation are the first argument of the operation, values of the negative one are the second argument.

The fourth table corresponds to existential quantifiers. It corresponds to disjunctions too.

This is because an existential quantifier removes values, corresponding to bound variables, from n -tuples. After removing we have some n -tuples differing only logical values, but only one of these n -tuples is allowable.

Disjunctions realize the joint of sets. At joint there are also n -tuples differing only logical values, but only one of these n -tuples is allowable.

In both cases we use the same rule of choice of one allowable logical value from several ones.

The last table " \leftarrow " is special.

This table should be copy of table " \exists ", because the right and left parts of " \leftarrow " must be joined. But this table is special as the back implication has unusual interpretation and this table must help to find superfluous (wrong) members at accommodation of new members in calculated sets. This is realized by the next rule:

- the values ft and tf must be symmetric in replacing one other.

This is because both values can be wrong and always the new value replaces the old value. To the end of calculation wrong values disappear and correct values replace wrong values in calculated sets (since correct values are generated again at each stage of iteration).

So, if the arguments equal tf and ft then result equals ft (this is new with respect to table " \exists "), and if the arguments equal ft and tf , then the result equals tf (this is not new).

But this rule resolves the problem only partly. We must add one more rule:

- value f replace value tf by ft .

This change only one value in table " \leftarrow ": if the arguments equal tf and f then the result equals ft .

Table " \leftarrow " (and table " \exists " too) has next property - value f is met as result only in the line next to last.

This line corresponds to value f as the first argument. It means this value never appears in calculated sets because it is absent before the iteration. So we can delete this line. But we can not delete the column corresponded to value f - this value can appear during the calculation of the right part of back implication (see table " $\wedge \neg$ ").

Now the wrong members in calculated sets have values only tf and ft . The members with value t never change this value.

In the other respects the last two tables are the same. The values of the last table are underlined if they differ from the values of the previous table.

Rule 3.29. Usage of 4-valued logic for non-monotone calculations:

- before the iteration all n -tuples of sets are added the logical value t ;

- the logical values are defined with the help of the tables " \neg ", " \wedge ", " $\wedge \neg$ ", " \exists " and " \leftarrow " during the iteration;

- after reaching a fixed point the members with logical value ft are removed and then the other logical values are removed from all n -tuples.

Thus, the 4-valued logic is used only during iterations. After ending iterations we shall return to the 2-valued logic.

Below, for a simplicity, stages of iteration are called steps.

Example 3.30. The graph of the formula

$$a(N_1) \leftarrow b(N_1) \wedge \neg a(N_1).$$

has a cycle with a marked arc.

Let a be empty before the beginning of calculations.

We must use the rule of non-monotone calculations.

Step 1. $a = b (tf)$.

Step 2. $a = b (ft)$.

Subsequently each i -th step of iteration repeats $(i - 2)$ -th step.

In these formulas the logical value is parenthesized.

The cycling can be interpreted as the unstable model with two fixed points (if the formula is correct):

$$a = \text{empty} \quad a = b$$

The formula, introduced in the given example, is not correct, because after reduction to CNF we have:

$$a(N_1) \vee \neg b(N_1).$$

Therefore, the initial formula becomes the formula without negation:

$$a(N_1) \leftarrow b(N_1).$$

The calculations of this formula give the result: $a = b$.

We shall receive the same result, if at cycling we select the maximal fixed point of two ones. \square

Below we again use the symbol " ' " as an marker.

Example 3.31. We have formulas, which graph contains a cycle with two marked arcs:

$$a'(N_1) \leftarrow b'(N_1) \wedge \neg a''(N_1).$$

$$a''(N_1) \leftarrow b''(N_1) \wedge \neg a'(N_1).$$

Before iteration the sets a' and a'' are empty.

Again we use the rule of non-monotone calculations.

Step 1. $a' = b'$ (tf).

$$a'' = b''$$
 (tf).

Step 2. $a' = (b' \cap b'')(ft) \cup (b' \setminus b'')(tf)$.

$$a'' = (b' \cap b'')(ft) \cup (b'' \setminus b')(tf).$$

Subsequently each i -th step of iteration repeats $(i - 2)$ -th step.

We must know, which of two fixed points

$$a' = b', \quad a'' = b''.$$

$$a' = b' \setminus b'', \quad a'' = b'' \setminus b'.$$

is correct.

The initial formulas can be present as

$$a'(N_1) \vee a''(N_1) \leftarrow b'(N_1).$$

$$a'(N_1) \vee a''(N_1) \leftarrow b''(N_1).$$

The graph of these formulas has no cycle. A fixed point

$$a' \cup a'' = b' \cup b''$$

is unique and reached for one step (stage) of iteration.

Only maximal of the two above-stated fixed points corresponds to this point:

$$a' = b', \quad a'' = b''.$$

The other fixed point is not correct.

The cycling which is a reason of the superfluous fixed point, is the result of a partial incorrectness of initial formulas.

Indeed, if we denote $c = a' \cup a''$ and $d = b' \cap b''$, we have (as independent) incorrect formula (see example 3.30):

$$c(N_1) \leftarrow d(N_1) \wedge \neg c(N_1).$$

This formula is a reason for change of logical value of c with tf on ft and back.

At removal of this incorrectness value c becomes tf and this agrees with received fixed point. \square

Example 3.31 is repetition of example 3.23. In example 3.23 we used the two-valued logic, in example 3.31 - the four-valued logic. In both cases the results are identical, though usage of the two-valued logic was incorrect.

It is necessary to mark, the given example is not an exception. In many non-monotone problems the two-valued logic gives the same result, as four-valued.

Example 3.32. We have formulas, which graph contains a cycle of three marked arcs:

$$a'(N_1) \leftarrow b'(N_1) \wedge \neg a''(N_1)$$

$$a''(N_1) \leftarrow b''(N_1) \wedge \neg a'''(N_1)$$

$$a^{p'''}(N_1) \leftarrow b'''(N_1) \wedge \neg a'(N_1)$$

Let the sets a' , a'' and a''' be empty before the beginning of iteration.

$$\begin{aligned} \text{Step 1. } a' &= b'(tf), \\ a'' &= b''(tf), \\ a''' &= b'''(tf). \end{aligned}$$

$$\begin{aligned} \text{Step 2. } a' &= (b' \cap b'')(ft) \cup (b' \setminus b'')(tf), \\ a'' &= (b'' \cap b''')(ft) \cup (b'' \setminus b''')(tf), \\ a''' &= (b''' \cap b')(ft) \cup (b''' \setminus b')(tf). \end{aligned}$$

Subsequently each i -th step of iteration repeats $(i - 2)$ -th step. Indeed, at step 3 we have:

$$a' = (b' \cap (b'' \cap b'''))(tf) \cup (b' \setminus (b'' \cap b'''))(tf) \cup (b' \cap (b'' \setminus b'''))(ft) \cup (b' \setminus (b'' \setminus b'''))(tf)$$

But $b' \cap (b'' \cap b''') \cup b' \setminus (b'' \cap b''') = b'$ and $b' \setminus (b'' \setminus b''')$ is a subset of b' . Besides, $b' \cap (b'' \setminus b''')$ is a subset of b' too, and at disjunctions value tf replaces value ft (see table “ \exists ”).

So $a' = b'$. But $a'' = b''$ and $a''' = b'''$ by analogy.

As a result we have two fixed points:

$$\begin{aligned} a' &= b' & a'' &= b'' & a''' &= b''' \\ a' &= b' \setminus b'' & a'' &= b'' \setminus b''' & a''' &= b''' \setminus b' \end{aligned}$$

The first point is maximal.

We can present the initial formulas as:

$$\begin{aligned} a'(N_1) \vee a''(N_1) &\leftarrow b'(N_1) \\ a''(N_1) \vee a'''(N_1) &\leftarrow b''(N_1) \\ a'''(N_1) \vee a'(N_1) &\leftarrow b'''(N_1) \end{aligned}$$

As it follows from these formulas, only the maximal fixed point satisfies them. \square

In the considered examples a maximal fixed point is always correct. It is not incidental.

If we have a cycling at calculations, the members of some subsets alternately accept logical values tf and ft . Therefore, each of such subsets R_1 is a result of iterations of the incorrect formula:

$$R_0(x_1, \dots, x_n) \leftarrow R_1(x_1, \dots, x_n) \wedge \neg R_0(x_1, \dots, x_n)$$

After correction the graph of this formula does not contain a cycle:

$$R_0(x_1, \dots, x_n) \leftarrow R_1(x_1, \dots, x_n)$$

As a result all cyclings disappear and each pair of fixed points of cycling is replaced by maximal of them.

Rule 3.33. Iteration with cycling:

- at cycling (by changing values tf on ft and back) each pair of the fixed points, appearing at cycling, is replaced by maximal of them.

The following example demonstrates, the series (in the order of hierarchy levels) and parallel (simultaneously for all formulas) calculations give the identical results.

Example 3.34. We have three-level formulas (see example 3.27):

$$\begin{aligned} a(N_1, N_2) &\leftarrow d(N_1, N_2). \\ b(N_1, N_2) &\leftarrow e(N_1, N_2) \wedge \neg a(N_1, N_2). \\ c(N_1, N_2) &\leftarrow e(N_1, N_2) \wedge \neg b(N_1, N_2) \wedge \neg a(N_2, N_1). \end{aligned}$$

Before the beginning of iteration the sets a , b and c are empty.

In example 3.27 the solution was constructed sequentially, in order of levels. It ensured a monotony of calculations.

If the calculations are executed in parallel (i.e. one-time calculations executed for all formulas of all levels on each step of iteration), the monotony disappears.

Below we use the operation $I(X_1)$ transposing elements in each pair of the set X_1 .

$$\text{Step 1. } a = d(t). \quad b = e(tf). \quad c = e(tf).$$

$$\text{Step 2. } a = d(t). \quad b = (e \cap d)(ft) \cup (e \setminus d)(tf). \quad c = e(ft).$$

$$\text{Step 3. } a = d(t). \quad b = (e \cap d)(ft) \cup (e \setminus d)(tf). \quad c = (e \cap d) \cap I(d)(ft) \cup ((e \cap d) \setminus I(d))(tf) \cup (e \setminus d)(ft).$$

The fixed point is reached: $a = d$, $b = e \setminus d$, $c = (e \cap d) \setminus I(d)$. This solution coincides with the solution in example 3.27.

All members of the set a were calculated at the first step.

The set b had the superfluous members on the first step and these members were removed on the second step.

The set c had superfluous members on the first step too, but all members, not only superfluous, were removed on the second step. So from e more members, than it is necessary, was removed, because

b had superfluous members. On the third step the set b was corrected and extra removed members were returned to set c . \square

Theorem 3.35. *If domains are finite and if the series iteration converges, the parallel iteration converges to the same solution.*

Proof: The series iteration means construction of a solution in order of hierarchy levels. The parallel iteration means one-time calculations of all formulas (all levels) on each step of the iteration. At the series iteration the calculated sets are always monotonically increasing, at the parallel iteration these sets can increase non-monotonically.

The proof is based on property of the member regeneration during the iteration: if the member x_1 appears on a step i from some set of members, the member x_1 will reappear while this set exists.

We shall use the induction rule.

The theorem is obvious to the single-level formulas.

Let the theorem be true for n levels and the solution is reached for k of steps. We must prove, the theorem is true for $(n + 1)$ -th level.

If a formula of this level does not enter into a cycle, the right part of this formula is not contain superfluous members after the step k . Therefore, the left part of this formula is not contain superfluous members after execution of the step $k + 1$ (and all consequent steps on the given level) too.

But if a formula enters into a cycle, the right part of this formula will contain superfluous members. These members appears in the left parts of the formula on the previous steps of iteration, and then they generate a chain of superfluous members.

With each new step of iteration a new member can appear at the end of the chain, but in the beginning of the chain any member can be removed (as a result of correction). The appearance of new members in the chain is finite, since the domains are finite. Therefore, removing of members in the beginning of the chain always ends by removing of all chain. Since after step k the new chains do not appear, on some step of iteration all the chains will be removed.

The normal members can be temporary replaced by superfluous members: if in a right part of the formulas the superfluous member has value tf , and the normal member has value ft , disjunction of these members (in the moment they must appear in the left part of the formulas) will have value tf . But as soon as a superfluous member will be removed the normal member appears in the left part (because of regeneration). As soon as all the chains will be removed, all normal members will be restored. The further calculations become monotonically increasing, without superfluous members. The theorem is proved. \square

Corollary 3.36. If the graph of the formulas does not contain a cycle with the marked arcs, the cycling (replacing value tf on ft and back) does not appear.

The given corollary is true for infinite domains too.

3.1.6. Constants.

Constants are special terms. We must use them with caution.

Example 3.37. The formula

$$a(1, 2) \leftarrow b(3, 4)$$

contains only constants.

According to "usual" interpretation the set a contains a member $\langle 1, 2 \rangle$ and to it the member $\langle 3, 4 \rangle$ from b is added. But this interpretation is wrong, because $\langle 3, 4 \rangle$ can be absent.

We have the true interpretation, if we remove constants from the formula:

$$a(N_1, N_2) \leftarrow 1(N_1) \wedge 2(N_2) \wedge 3(N_3) \wedge 4(N_4) \wedge b(N_3, N_4)$$

In this formula conjunction $3(N_3) \wedge 4(N_4) \wedge b(N_3, N_4)$ becomes a proposition (N_3 and N_4 are bound by implicit existential quantifiers). The proposition is true, if the set b contains a member $\langle 3, 4 \rangle$, or is false otherwise. In the first case the set a adds the pair $\langle 3, 4 \rangle$, in the second case this set remains invariable.

Thus, the formula of the example is the mixture of the theories of order 0 and 1. \square

Though the theory of the second order usually includes the theory of the first order, the mixing of theories of the order 0 and 1 is not recommended.

It means, we must know before calculations, if propositions are true or false and accordingly correct formulas.

In particular, in the example 3.37 before iteration we must add (or not add) the pair $\langle 3, 4 \rangle$.

Rule 3.38 of the proposition removal:

- the right part of formulas must be without propositions.

This rule is not used for formulas without the right part. These formulas are interpreted as members of initial sets.

But it is natural to remove these formulas. Then propositions will be absent in all formulas. It means initial sets must be given implicitly.

In particular, at the factorial definition formula $0! = 1$ must be removed from the definition. The other formula will allow to calculate not only factorial, but also gamma-function (depending on the initial set).

After removing all propositions (in the right parts of the formulas too) we exclude the possibility of error because of constants in positive literals.

But the error remains for constants in negative literals.

Example 3.39. $W(S_1) \leftarrow \neg SP(S_1, 1)$.

This example from [7], the relation $SP(S_1, P_1)$ defines parts P_1 provided by supplier S_1 . One supplier S_1 can have some parts, one part can be present at several suppliers. We note $\neg SP(S_1, 1)$ those suppliers that have not part 1. It is necessary to define the set $W(S_1)$ of these suppliers.

One interpretation is the construction of complement to set $SP(S_1, 1)$ (for domain $S \times P$). This interpretation is wrong: W will include the suppliers that have part 1, if they have other parts too.

We shall receive the right formula, if we remove the constant:

$$W(S_1) \leftarrow \neg SP(S_1, P_1) \wedge 1(P_1).$$

Therefore, at first we must calculate negation of SP , and only then substitute P_1 by 1. Negation of SP contains for each supplier the list of parts not to be provided by this supplier.

But there is a more simple way, if we use set difference. By this way we shall list supplies that have part 1. We denote this list $W0(S_1)$ and construct complement to $W0$ (for domain S). \square

Rule 3.40 of calculation of negative literals with constants:

- before the calculation of a negative literal, we must calculate this literal as positive (it means only lines with the given constants are left), then remove the columns corresponded to the given constants, and construct the complement to the received set.

3.1.7. Second Order Formulas.

The formulas of the second order were met above very often. Moreover, all the rules above can be present as formulas of the second order and the set of such formulas is an axiomatic of the first order finite logic (this logic is a member of the set of the second order theories).

In classic logic it is supposed, we come to the second order formulas, if values of variables of these formulas are relation names. It is not well.

Below we have two examples with such variables. But only one of these examples has the second order formulas.

Example 3.41. A number N_1 of arguments in a relation R_1 of a theory T_1 is given by the function $arity(T_1, R_1) = N_1$.

It is known (see section 1.4), in the relational logic the undefined notion $Relations(T_1, R_1, N_1, S_1)$ exists. Values of a variable R_1 are relation names in a theory T_1 . Values of a variables N_1 are the serial numbers of places in the relation R_1 , values of a variable S_1 are codes of sorts for these places.

The predicates and functions do not differ in $Relations$. Therefore, a number of places, extracted from $Relations$, should be decreased on 1 for functions. For separating functions among relations we use the notion $Functions(T_1, R_1, N_1)$. This notion gives a number N_1 of places containing function value for a relation R_1 of a theory T_1 . The relation R_1 is a function, if this relation has only one such place. This place is always the last. The relation is functional (but not a function), if R_1 has several such places.

The definition of $arity$ is put by the formula:

$$arity(T_1, R_1) = N_1 \Leftrightarrow \neg Functions(T_1, R_1, N_1) \wedge Relations(T_1, R_1, N_1) \wedge \neg Relations(T_1, R_1, N_1 + 1) \vee Relations(T_1, R_1, N_1 + 1) \wedge \neg Relations(T_1, R_1, N_1 + 2).$$

This definition consists of two parts.

In the first part we define a number of places for relations not being functions: from the set $Relations$ we extract a member with a maximal value N_1 at a given R_1 . The value N_1 is maximal, if a member with the value $N_1 + 1$ does not exist.

In the second part we define a number N_1 of arguments for functions. In *Relations* the value $N_1 + 1$ exists, but the value $N_1 + 2$ does not exist. \square

In this example we use the second order logic and the first order formulas.

Example 3.42. The definition of a direct product looks like:

$$R_1 \times R_2 = R_3 \Leftrightarrow \forall x_1 \dots \forall x_n R_1(x_1, \dots, x_m) \wedge R_2(x_{m+1}, \dots, x_n) \rightarrow R_3(x_1, \dots, x_n)$$

where $m = \text{arity}(R_1)$, $n = m + \text{arity}(R_2)$.

The positive definition

$$R_1 \times R_2 = R_3 \leftarrow \forall x_1 \dots \forall x_n R_1(x_1, \dots, x_m) \wedge R_2(x_{m+1}, \dots, x_n) \rightarrow R_3(x_1, \dots, x_n)$$

allows to construct set “ \times ” of members $\langle R_1, R_2, R_3 \rangle$. At iteration for every R_1 and R_2 we seek (in *Theories*) R_3 to equal direct product of R_1 and R_2 .

But we must solve completely another problem - to construct the set R_3 , instead of set “ \times ”.

To construct set R_3 , the formula is needed with the left part equal to R_3 .

This formula can be received from the negative definition of the direct product:

$$R_3(x_1, \dots, x_n) \leftarrow R_1 \times R_2 = R_3 \wedge R_1(x_1, \dots, x_m) \wedge R_2(x_{m+1}, \dots, x_n).$$

It is follows from this formula, the direct product $R_1(x_1, \dots, x_m)$ and $R_2(x_{m+1}, \dots, x_n)$ is added to $R_3(x_1, \dots, x_n)$ if $R_1 \times R_2 = R_3$.

So the positive definition of the direct product should be removed, since it does not take part in calculations of direct products. \square

This example demonstrates, model construction in the second order logic uses, as a rule, negative definitions. The second normal form should be replaced by a new normal form.

Now we can define the second order formulas.

Definition 3.43. *The second order formula* contains a variable without list of places and the same variable with list of places (in parentheses).

It is necessary to mark, the dots are very powerful tool for construction of the second order formulas.

In example 3.42 semantics of dots is trivial. But this semantics is very complicated in more complex problems. The computer formalizing of this semantics is an important problem of the second order logic.

3.2. Theory Construction (Theorem proving).

As it was pointed above the theory construction includes theory calculus and axiom calculus of a theory.

The theory calculus will be considered in the special series of articles. Below theory construction means only the axiom calculus of a theory. More than that, below we shall consider only the axiom calculus of an inconsistent theory.

Any theory becomes inconsistent, if the axiom system of the theory is added by negation of a theorem of the theory. The proof of inconsistency of such theory is simultaneously the proof of the theorem.

The proof of inconsistency of a theory is a part of axiom calculus. If the empty (i.e. not existing) axiom is generated as a new axiom (theorem), inconsistency of the theory becomes proved. The empty axiom as the empty set are interpreted by logic object “false”.

The proved theorem should be in the first normal form. It means, the quantifiers are implicit and in $\forall\exists$ -prenex form, i.e. existential quantifiers must be after universal quantifiers.

So negation of proved theorem will be in $\exists\forall$ -form and we must reduce it to $\forall\exists$ -form. But we can use $\exists\forall$ -form without the reduction if negation of the theorem is a sentence in the first normal form. We shall show this.

According section 1.2 the new relation W should be added to reduce to $\forall\exists$ -form:

$$\begin{aligned} W(x_i, \dots, x_j) &\leftarrow \forall x_{j+1}, \dots, x_k \mathcal{F}(x_i, \dots, x_k) \\ W(x_i, \dots, x_j) &\rightarrow \forall x_{j+1}, \dots, x_k \mathcal{F}(x_i, \dots, x_k) \\ &\exists x_i, \dots, x_j W(x_i, \dots, x_j) \end{aligned}$$

where $\mathcal{F}(x_i, \dots, x_k)$ is negation of the theorem, x_i, \dots, x_j - variables bound by existential quantifiers (usually these variables are denoted by a_i, \dots, a_j), x_{j+1}, \dots, x_k - variables bound by universal quantifiers.

The first sentence can not be used in the proving. The resolution of the last two sentences gives $\mathcal{F}(a_i, \dots, a_j, x_{j+1}, \dots, x_l)$, i.e. again negation of the proved theorem but in $\forall\exists$ -form. The last sentence means that a_i, \dots, a_j do not depend on x_{j+1}, \dots, x_k .

So we can remove all three sentences if we remember this property of a_i, \dots, a_j . For that we use notation c_i, \dots, c_j instead of a_i, \dots, a_j .

At deduction we shall watch for moving of these variables in the deduced sentences. These sentences are called the successors of negation of the proved theorem.

For example, if we have got $x_m = c_i$ or $x_m \neq c_i$ in any successor,

each of these relations becomes inconsistent because of the implicit quantifiers $\exists c_i \forall x_m x_m = c_i$ and $\exists c_i \forall x_m x_m \neq c_i$.

The next inference rules are used in the axiom calculus of an inconsistent theory:

- the resolution rule (analogue of the modus ponens);
- the paramodulation rule (analogue of the equation axioms);
- the finite descent rule (analogue of the induction).

The first two rules are two-place operations: from two axioms they deduce one new axiom. The last rule is one-place.

There is a lot of the other inference rules which are one-place operations.

All one-place operations execute transformations of axioms. As it is generally accepted, these operations are not included as steps of proofs. All these operations are used after execution of two-place operations and they reduce the result of two-place operations to a normal form.

3.2.1. Resolution

The resolution rule uses substitutions, unifications and disjunctions of contrary pairs.

Definition 3.44. The *substitution* is a finite set $\{x_1/t_1, \dots, x_n/t_n\}$ where x_i is a variable not to bind by an existential quantifier, t_i is a term that differs from x_i , all x_i are different. The substitution without members is called empty and is denoted ϵ .

We denote θ the set of all substitutions.

Definition 3.45. The *substitution application*

$$\mathcal{F}_1\theta_1$$

is the result of simultaneous replacement of all variables x_i in \mathcal{F}_1 on t_i in accordance with the substitution θ_1 .

Rule 3.46 of substitution composition. Let:

- the substitution θ_0 be a composition of substitutions θ_1 and θ_2 , i.e. $\mathcal{F}_1\theta_1\theta_2 \Leftrightarrow (\mathcal{F}_1\theta_0)$, where $\theta_1 = \{x_{11}/t_{11}, \dots, x_{1n}/t_{1n}\}$, $\theta_2 = \{x_{21}/t_{21}, \dots, x_{2m}/t_{2m}\}$.

Then:

- θ_0 is received from a set

$$\{x_{11}/(t_{11}\theta_2), \dots, x_{1n}/(t_{1n}\theta_2), x_{21}/t_{21}, \dots, x_{2m}/t_{2m}\}$$

by deletion of all members $x_{1i}/(t_{1i}\theta_2)$, for which $t_{1i}\theta_2 = x_{1i}$, and all members x_{2j}/t_{2j} , for which $t_{2j} \in \{x_{11}, \dots, x_{1n}\}$.

Rule 3.47 of unification of two relations. This rule consists of a sequence of the following steps:

Step 1. We take the beginning of the relations as the given positions (in these positions the symbols of the relations are present), and we take the empty substitution as an initial substitution θ_1 ($\theta_1 = \epsilon$). Further the given positions vary and can be disposed in each relation at different distances from the beginning of the relations. A number of positions in the relations changes after substitutions.

Step 2. We seek the next position with different symbols. Unification is finished and substitution θ_1 is sought for, if such position does not exist.

Step 3. If one of the symbols in the found position is a variable x_i , then

- the term t_i , beginning with the corresponding position in the other relation and not containing x_i , produces the substitution $\{x_i/t_i\}$;

- the given positions become the position after the variable in one relation and after the term in the other relation;

- all variable x_i (since the given position) are replaced by a term t_i in both relations;

- the composition of substitutions θ_1 and $\{x_i/t_i\}$ is constructed, this composition becomes the new value of a substitution θ_1 ;

- go to step 2.

The unification does not exist, if both symbols in the found position are not variables or the term t_i contains x_i .

It is necessary to point out, in this rule variables are x -variables, i.e. variables, bound by universal quantifier.

Definition 3.48. The *unificator* of two relations R_1 and R_2 is the substitution received by the rule of unification.

For terms the rule of unification and the definition of unificator are analogously.

Definition 3.49. The *contrary pair* is two literals differing only by signs.

Contrary pairs have important properties:

- an \vee -clause becomes a valid formula, if this \vee -clause has literals of a contrary pair.
- an \wedge -clause becomes negation of a valid formula, if this \wedge -clause has literals of a contrary pair.

Definition 3.50. Literals in two axioms (one in each axiom) are *marked*, if after unification these literals become the contrary pair.

Definition 3.51. Variables, bound by existential quantifiers and met only in terms of the unificator, are *a-variables*.

If in marked literals we meet the variables, bound by existential quantifiers, then they are *a-variables*.

Definition 3.52. The *member* of the axiom is an \wedge -clause or a literal from \vee -clause of this axiom. Every axiom is a set of members connected by disjunctions.

Definition 3.53. The marked members of two axioms are the members with marked literals.

If a unificator has no *a*-variable, the resolution is usual. In this case marked members are literals.

Rule 3.54 of resolution. For two axioms:

- variables, bringing to collisions, are renumbered;
- literals only from \vee -clause of axioms are *marked*;
- the marked literals are removed from both axioms;
- the rest of both axioms are joined by a disjunction;
- the unificator is applied to the received formula.

This formula is named *resolvent* of given axioms. As a rule, the resolvent is the new axiom.

Theorem 3.55. *The resolution is a valid formula.*

Proof: Let two axioms be given:

$$\begin{aligned} L_1(\dots) \vee \mathcal{F}_1 \\ L_2(\dots) \vee \mathcal{F}_2 \end{aligned}$$

where L_1 and L_2 are marked literals, \mathcal{F}_1 and \mathcal{F}_2 are the other part of the axioms. Below parentheses and dots are removed.

These two axioms can be joined with the conjunction:

$$(L_1 \vee \mathcal{F}_1) \wedge (L_2 \vee \mathcal{F}_2)$$

Using the distributivity of disjunction and conjunction we shall receive after applying of the unificator θ_1 :

$$(L_1 \wedge \mathcal{F}_2 \vee \mathcal{F}_1 \wedge L_2 \vee \mathcal{F}_1 \wedge \mathcal{F}_2)\theta_1$$

We remove the literal $L_1 \wedge L_2$ since it is negation of a valid formula.

Again we use the distributivity of disjunction and conjunction:

$$\begin{aligned} ((L_1 \vee \mathcal{F}_1) \wedge (L_1 \vee \mathcal{F}_1 \vee \mathcal{F}_2) \wedge (L_1 \vee L_2 \vee \mathcal{F}_1) \wedge (L_1 \vee L_2 \vee \mathcal{F}_2) \wedge \\ (\mathcal{F}_2 \vee \mathcal{F}_1) \wedge (\mathcal{F}_2 \vee \mathcal{F}_1) \wedge (\mathcal{F}_2 \vee L_2 \vee \mathcal{F}_1) \wedge (\mathcal{F}_2 \vee L_2))\theta_1 \end{aligned}$$

Removing \vee -clauses with the contrary pair or subsumed by the initial axioms $(L_1 \vee \mathcal{F}_1, L_2 \vee \mathcal{F}_2)$ and other \vee -clauses we shall receive single \vee -clause:

$$(\mathcal{F}_1 \vee \mathcal{F}_2)\theta_1$$

Therefore, the formula

$$(L_1 \vee \mathcal{F}_1) \wedge (L_2 \vee \mathcal{F}_2) \rightarrow (\mathcal{F}_1 \vee \mathcal{F}_2)\theta_1$$

is valid. \square

3.2.2. Resolution with Factor.

Definition 3.56. The *factor* is a result of applying of the unification to two or more literals in an axiom.

If literals have the common unificator, these literals become the same after the unificator is applied. So several literals become a one literal.

Obviously the factor is a valid formula. This formula is used to simplify axioms.

But the resolution with the factor is complex for the axiom generation. So it is not used, if it can be replaced. But there are problems what are insoluble without factors.

Example 3.57. Let two axioms be:

$$\begin{aligned} P(x_1, x_2) \vee P(x_2, x_3) \vee Q(x_3, x_4) \vee Q(x_4, x_1) \\ \neg P(x_1, x_2) \vee \neg P(x_2, x_1). \end{aligned}$$

In both axioms the first two literals have common unificators. After applying these unificators we have:

$$\begin{aligned} P(x_1, x_1) \vee Q(x_1, x_4) \vee Q(x_4, x_1) \\ \neg P(x_1, x_1). \end{aligned}$$

The resolvent of these axioms is

$$Q(x_1, x_4) \vee Q(x_4, x_1) \quad (1)$$

Let one more axiom be added to the initial axioms:

$$\neg Q(x_1, x_2) \vee \neg Q(x_2, x_1) \quad (2)$$

The literals have common unificators in both axiom (1) and (2). Applying these unificators we get:

$$\begin{aligned} \neg Q(x_1, x_1) \\ Q(x_1, x_1) \end{aligned}$$

The resolvent of these axioms is the empty axiom. We cannot deduce this empty axiom without factors. \square

In this example the factor is used before the resolution. But it is more effective to use the resolution before the factor, if we use a computer.

For that we must introduce a notion.

Definition 3.58. The resolvent with the *informative literal* is a resolvent of two axiom with an additional literal. This literal is the positive part of the contrary pair of the resolution after applying the unificator. The informative literal separates literals of axioms being the operands of the resolution. Before the informative literal we put the literals of the axiom with the positive literal of the contrary pair, after the informative literal we put the literals of the other axiom.

Example 3.59. Again we have three axioms of example 3.57:

$$\neg P(x_1, x_2) \vee \neg P(x_2, x_1).$$

$$Q(x_1, x_2) \vee Q(x_3, x_4)$$

$$P(x_1, x_2) \vee P(x_2, x_3) \vee Q(x_3, x_4) \vee Q(x_4, x_1)$$

or, after the renumeration (to escape the variable collision)

$$P(x_1, x_2) \vee P(x_2, x_3) \vee Q(x_3, x_4) \vee Q(x_4, x_1) \quad (1)$$

$$\neg P(x_5, x_6) \vee \neg P(x_6, x_5) \quad (2)$$

$$Q(x_7, x_8) \vee Q(x_8, x_7) \quad (3)$$

The resolvent of axioms (1) and (2) with the informative literal is:

$$P(x_2, x_3) \vee Q(x_3, x_4) \vee Q(x_4, x_1) \vee [P(x_1, x_2)] \vee \neg P(x_2, x_1) \quad (4)$$

In this formula the informative literal is taken in square brackets. The resolvent of axioms (3) and (4) is

$$P(x_2, x_3) \vee Q(x_4, x_1) \vee [P(x_1, x_2)] \vee \neg P(x_2, x_1) \vee [Q(x_3, x_4)] \vee \neg Q(x_4, x_3) \quad (5)$$

This resolvent becomes empty because $P(x_2, x_3)$ and $\neg P(x_2, x_1)$ unify one with other and with $[P(x_1, x_2)]$, also $Q(x_4, x_1)$ and $Q(x_3, x_4)$ unify one with other and with $[Q(x_3, x_4)]$. The common unificator for all P equals $\{x_2/x_1, x_3/x_1\}$ and for all Q equals $\{x_3/x_1, x_4/x_1\}$. Both unificators do not contradict one another. \square

Rule 3.60 of resolution with factor:

- for every resolution the resolvent is added the informative literal;

- the resolvent becomes the empty axiom, if this resolvent is partitioned in some groups and every group includes one informative literal and has a common unificator for all literals of this group (the common unificators must be uncontradicted).

So we use the factorization only when the resolvent can be empty.

Example 3.61. We have two axioms:

$$P(x_1, x_2) \vee P(x_2, x_3) \vee P(x_3, x_1).$$

$$\neg P(x_1, x_2) \vee \neg P(x_2, x_3) \vee \neg P(x_3, x_1).$$

We must renumerate variables:

$$P(x_1, x_2) \vee P(x_2, x_3) \vee P(x_3, x_1). \\ \neg P(x_4, x_5) \vee \neg P(x_5, x_6) \vee \neg P(x_6, x_4).$$

The resolvent of these axioms is:

$$P(x_2, x_3) \vee P(x_3, x_1) \vee [P(x_1, x_2)] \vee \neg P(x_2, x_6) \vee \neg P(x_6, x_1).$$

The common unificator for the resolvent equals $\{x_2/x_1, x_3/x_1, x_6/x_1\}$.

So this resolvent is empty. \square

3.2.3. Generalized Resolution.

If unificator of marked literals contains a -variables, the usual resolution rule is replaced by the generalized resolution rule.

Rule 3.62 of generalized resolution. For two axioms:

- variables that leads to collisions are renumbered;
- literals from any part of axioms are marked, if their unificator has "a"-variable;
- both axioms are jointed by conjunction;
- the received formula is reduced to DNF;
- the existential quantifiers for all a -variables are added in the beginning of the received formula;
- unificator of marked literals is applied to this formula;
- the received formula is reduced to the first normal form;
- sentences, subsumed by the other sentences, are removed.

The reduction to DNF and the addition of explicit existential quantifiers are needed to apply the unificator to all formulas.

Theorem 3.63. *The generalized resolution is a valid formula.*

Proof: small The proof follows from the definition. Transformations in this definition used valid formulas. \square

We shall enter the following identifications:

E_1 and E_2 - marked members of two axioms (a member is a literal without variables bound by existential quantifiers or an \wedge -clause);

\mathcal{F}_1 and \mathcal{F}_2 - the rest of the axioms after removing marked members (they consist of members, connected by disjunction);

According to the rule of generalized resolution we join both axioms by conjunction:

$$(E_1 \vee \mathcal{F}_1) \wedge (E_2 \vee \mathcal{F}_2)$$

Using the valid formula:

$$E_1 \vee \mathcal{F}_1 \vdash (E_1 \vee \mathcal{F}_1) \wedge \mathcal{F}_2 \Leftrightarrow \mathcal{F}_2$$

we can replace the conjunction of axioms by:

$$(E_1 \vee \mathcal{F}_1) \wedge E_2 \vee \mathcal{F}_2$$

Then DNF of this formula is (after applying the unificator θ_1):

$$(\underline{E_2 \wedge \mathcal{F}_1'} \vee E_2 \wedge \mathcal{F}_1'' \vee \mathcal{F}_2)\theta_1$$

In this formula $E_1 \wedge E_2$ is deleted because it is negation of a valid formula. The formula F_1 is replaced by $F_1' + F_1''$, where members of \mathcal{F}_1' have a -variables common with E_2 . We underline \wedge -clause in this formula. Then we move the existential quantifiers of a -variables to the \wedge -clause.

After reducing to the first normal form we have two sentences:

$$(E_2 \wedge \mathcal{F}_1' \vee \mathcal{F}_1'' \vee \mathcal{F}_2)\theta_1 \tag{R1} \\ (E_2 \wedge \mathcal{F}_1' \vee E_2 \vee \mathcal{F}_2)\theta_1$$

The last sentence is subsumed by the second axiom. So we have only sentence (R1).

By analogy we can have one more sentence:

$$(E_1 \wedge \mathcal{F}_2' \vee \mathcal{F}_2'' \vee \mathcal{F}_1)\theta_1 \tag{R2}$$

It means the resolution depends on the order of arguments.

Usually the first argument is an axiom of the theory, the second argument is a successor of negation of the proved theorem, and the resolvent is the successor, too. Then we have only one resolvent (R1).

The generalized resolution is effective at realization on computers because of integration a considerable amount of information in one axiom. Conventional usage of Skolem functions [6] disintegrates this information.

The generalized resolution is just irreplaceable if we use the induction rule.

3.2.4. Paramodulation.

The equality is a special relation. It has properties of a function: $x_1 = x_2 \wedge x_1 = x_3 \rightarrow x_2 = x_3$. But it is never used as a function. It is defined in each theory differently, but its properties are the same for all theories, i.e. these properties are valid formulas in the logic with equality.

The properties of equality are used to reduce a number of literals in the axioms and deduce new axioms.

Two valid formulas are used to reduce a number of literals:

$$\begin{aligned}\mathcal{F}_1[x_1] \vee x_1 \neq t_1 &\Leftrightarrow \mathcal{F}_1[t_1] \\ \mathcal{F}_1[x_1] \wedge x_1 = t_1 &\Leftrightarrow \mathcal{F}_1[t_1]\end{aligned}$$

where \mathcal{F}_1 is an arbitrary formula containing a free variable x_1 , t_1 is an arbitrary term that does not contain x_1 .

The rule of paramodulation is used to deduce the new axioms.

Rule 3.64 of paramodulation. If we have:

- two axioms;
- a literal with equation in \vee -clause of one of these axioms (for example, in the first), this equality we shall call marked;
- the left or the right part in this equality (for example, the left part of the equality);
- a literal in any part of the second axiom, this literal we shall call marked;
- a (marked) symbol disposed after the first symbol of this literal;
- a unificator of the (marked) term that begins with the marked symbol, and term that is the left part of the equality (it is supposed, unificator exists and does not contain a -variables, i.e. the left part of marked equality and marked term do not contain a -variables).

Then

- variable are renumbered to avoid collisions;
- marked equality is removed from the first axiom;
- marked term of the second axiom is replaced by the right part of removed equality;
- both changed axioms are joined by disjunction in a formula;
- the unificator is applied to this formula.

Modulation in this rule means a change of the second axiom. The prefix of "para" underlines that a number of literals in this axiom increased.

Theorem 3.65. *Paramodulation is a valid formula in logic with equality.*

Proof: Let two axioms be given:

$$\begin{aligned}t_1 = t_2 \vee \mathcal{F}_1 \\ L_2[t_0] \vee \mathcal{F}_2\end{aligned}$$

where t_1 and t_2 are terms, $L_2[t_0]$ is a marked literal with a marked term t_0 , \mathcal{F}_1 and \mathcal{F}_2 are remaining parts of the axioms.

Let x_k be a variable which is not met in the given axioms. Then the next formula is valid:

$$L_2[t_0] \vee \mathcal{F}_2 \Leftrightarrow L_2[x_k] \vee \mathcal{F}_2 \vee x_k \neq t_0$$

Thus, the second axiom can be substituted by the formula

$$L_2[x_k] \vee \mathcal{F}_2 \vee x_k \neq t_0$$

Let equality in the first axiom and inequality in this formula be marked literals. Then the unificator of these literals contains a substitution x_k/t_2 and all substitutions generated by the unificator θ_1 of the terms t_0 and t_1 .

Using the rule of the resolution and the substitution x_k/t_2 , we shall receive

$$(\mathcal{F}_1 \vee L_2[t_2] \vee \mathcal{F}_2)\theta_1.$$

Therefore, the formula

$$(t_1 = t_2 \vee \mathcal{F}_1) \wedge (L_2[t_0] \vee \mathcal{F}_2) \rightarrow (\mathcal{F}_1 \vee L_2[t_2] \vee \mathcal{F}_2)\theta_1$$

is valid. The proof is completed. \square

3.2.5. Generalized Paramodulation.

The generalized paramodulation is a resolution with a generalized unification. This resolution can be usual or generalized.

But before we should enter some notions.

Definition 3.66. A term being an argument of a relation is *primary*. A term being an argument of a primary term is *secondary*.

Rule 3.67 of the generalized unification. If we have:

- two literals;
- two positions (one in each literal) till to them these literals are unified by the usual or generalized unification;
- these positions must be in any point of secondary terms, if these terms are the first arguments, or in any point of primary terms, if these terms are the other arguments.

Then:

- we construct the inequality, both parts of which are the terms, beginning in the given positions.
- the inequality becomes the \wedge -clause, if this inequality has variables bound by existential quantifier, we construct \wedge -clause in accordance with the generalized resolution rule (below we call these clauses the inequality, too).

As a result we have both a unificator and a set of inequalities. But we have many results of the generalized unification for the given two literals. If we apply the usual unification, the result is unique.

The positions for unification must be in the secondary terms if these terms are the first arguments. Otherwise we shall have several equivalent proofs.

Rule 3.68 of generalized paramodulation (resolution with generalized unification). If we have:

- a contrary pair.

Then:

- we construct the \vee -clause from the inequalities generated at the unification;
- we construct the resolvent using the rule of generalized resolution;
- disjunction of this resolvent and the \vee -clause forms the terminal resolvent.

Rule 3.69 of one-place paramodulation. If we have:

- an axiom;
- an inequality in this axiom.

Then:

- the generalized paramodulation is used with the contrary pair formed by the left and right parts of this inequality.

Here we give the examples taken from [6].

For simplification in these examples we drop the literals inherited by previous steps of proving. So \wedge -clauses are absent.

Example 3.70. The axioms are given:

1. $x_1 + x_2 = x_2 + x_1$.
2. $x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$.
3. $(x_1 + x_2) - x_2 = x_1$.
4. $(x_1 - x_2) + x_3 = (x_1 + x_3) - x_2$.

We shall prove the theorem $(x_1 - x_2) + x_3 = x_1 + (x_3 - x_2)$.

Negation of the theorem and steps of the proof are the next.

5. $(c_1 - c_2) + c_3 \neq \underline{c_1 + (c_3 - c_2)}$.

We have underlined the part that will be the right part of inequality as a result of generalized unification on the next step of proving. Below we use underlining with the same purpose.

- 6.(4,5) $\underline{(c_1 + c_3)} - c_2 \neq \underline{c_1 + (c_3 - c_2)}$.

It is the first step of the proof. In parentheses we give the numbers of axioms used by the paramodulation.

7. (4r,6) $x_1 + x_3 \neq c_1 + c_3 \vee (x_1 - c_2) + x_3 \neq c_1 + (c_3 - c_2)$

We have added the letter "r" in the parentheses. It means the order of unification is changed: the right part of (4) unified with the left part of (6). In the previous step the order of unification is not change: we have unified the left part of (4) with the left part of (5).

We have two inequalities as a result of the generalized unification.

The first inequality does not seem natural, because it can be replaced by two substitutions x_1/c_1 and x_3/c_3 . But this inequality is needed for the next paramodulations.

- 8.(1,7) $(c_3 - c_2) + c_1 \neq c_1 + (c_3 - c_2)$

9.(1,8) Empty. \square

Example 3.71. The same axioms and negation of the theorem $(x_1 - x_2) - (x_3 - x_2) = x_1 - x_3$ are given. We must construct the proof.

1. $x_1 + x_2 = x_2 + x_1$.
2. $x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$.
3. $(x_1 + x_2) - x_2 = x_1$.
4. $(x_1 - x_2) + x_3 = (x_1 + x_3) - x_2$.
5. $(c_1 - c_2) - (c_3 - c_2) \neq c_1 - c_3$.
6. (3,5) $(c_1 - c_3) + (c_3 - c_2) \neq \underline{c_1 - c_2}$.
7. (1,6) $(c_3 - c_2) + (c_1 - c_3) \neq \underline{c_1} - c_2$.
8. (4,7) $c_3 + (c_1 - c_3) \neq \underline{c_1}$.
9. (1,8) $(c_1 - c_3) + c_3 \neq \underline{c_1}$.
10. (4,9) $(c_1 + c_3) - c_3 \neq c_1$.
11. (3,10) Empty. \square

Example 3.72. The axioms and negation of the theorem $(x_1 + x_2) - (x_3 + x_2) = x_1 - x_3$ are given:

1. $x_1 + x_2 = x_2 + x_1$.
2. $x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$.
3. $(x_1 + x_2) - x_2 = x_1$.
4. $(x_1 - x_2) + x_3 = (x_1 + x_3) - x_2$.
5. $(c_1 + c_2) - (c_3 + c_2) \neq c_1 - c_3$.

Proof.

6. (3,5) $(c_1 - c_3) + (c_3 + c_2) \neq \underline{c_1} + c_2$.
7. (2,6) $(c_1 - c_3) + c_3 \neq \underline{c_1}$.
8. (4,7) $(c_1 + c_3) - c_3 \neq c_1$.
9. (3,8) Empty. \square

3.2.6. Finite Descent.

The *finite descent* rule is computer oriented variant of the induction rule.

Let the literal $L_1[a_1]$ be received on some step of a proof, where $L_1[a_1]$ means the literal L_1 contains the variable a_1 , bound by existential quantifier. This variable will supply the literal L_1 to be included in all next steps of the proof.

Let the literal $L_1[a'_1]$ be received on the other step of a proof. So an \wedge -clause contains $L_1[a_1] \wedge L_1[a'_1]$.

Therefore, one part of the induction rule is executed. For execution of the other part of this rule we shall remove the literal $L_1[a'_1]$ from the \wedge -clause, replace all variables a_1 by constant 0 and go on the proof.

This proof procedure is the finite descent rule, since the subformula $L_1[a_1] \wedge L_1[a'_1]$ are replaced by the subformula $L_1[0]$. It is the descent from value a_1 to value 0.

Definition 3.73. The *inductive pair* in an \wedge -clause is two literals one of them containing the variable a_i , and the other one containing in the same position variable a'_i , in the rest of the positions the symbols are the same. The values of a_i are the natural numbers, and a_i is the *inductive variable*.

Rule 3.74 of finite descent. If it is given:

- \wedge -clause in the axiom;
- two literals in this \wedge -clause which become an inductive pair after a unificator applying.

Then:

- the second literal of inductive pair is removed from \wedge -clause;
- the inductive variable is replaced by 0 in the other literals of \wedge -clause.

Example 3.75. The axioms of arithmetic are given:

1. $x'_1 \neq 0$.
2. $x_1 = x_2 \vee x'_1 \neq x'_2$.

We must prove the theorem $x'_1 \neq x_1$.

The negation of the theorem is:

3. $c'_1 = c_1$.

We shall use the one-place inference rule: if $t = t'$, then $t' = t''$, where t is a term. It follows from the valid formula: functions are equal if their arguments are equal. So

$$3'. c_1'' = c_1' \wedge c_1' = c_1.$$

The finite descent rule gives:

$$3''. 0' = 0.$$

The proof has only one step (we point out in parentheses the steps using in the resolution);

4. (1,3'') Empty. \square

Example 3.76. The axioms of arithmetic are given:

$$1. x_1' \neq 0.$$

$$2. x_1 = x_2 \vee x_1' \neq x_2'.$$

$$3. x_1 + 0 = x_1.$$

$$4. x_1 + x_2' = (x_1 + x_2)'.$$

We like to prove the theorem $(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$. The negation of this theorem is added to the axioms of arithmetic:

$$5. (c_1 + c_2) + c_3 \neq c_1 + (c_2 + c_3)$$

Proof.

$$6.(2,5) ((c_1 + c_2) + c_3)' \neq \underline{(c_1 + (c_2 + c_3))}' \wedge (c_1 + c_2) + c_3 \neq c_1 + (c_2 + c_3)$$

$$7.(4r,6) (c_1 + c_2) + c_3' \neq (c_1 + (c_2 + c_3))' \wedge ((c_1 + c_2) + c_3)' \neq (c_1 + (c_2 + c_3))' \wedge (c_1 + c_2) + c_3 \neq c_1 + (c_2 + c_3)$$

$$8. (4r,7r) (c_1 + c_2) + c_3' \neq c_1 + (c_2 + c_3)' \wedge (c_1 + c_2) + c_3' \neq (c_1 + (c_2 + c_3))' \wedge ((c_1 + c_2) + c_3)' \neq (c_1 + (c_2 + c_3))' \wedge (c_1 + c_2) + c_3 \neq c_1 + (c_2 + c_3).$$

In this formula we can use the finite descent rule: the first literal $(c_1 + c_2) + c_3' \neq c_1 + (c_2 + c_3)'$ and the last literal $(c_1 + c_2) + c_3 \neq c_1 + (c_2 + c_3)$ form the inductive pair, if $(c_2 + c_3)' = c_2 + c_3'$. So at (generalized) unification of these literals, the terminal resolvent has the inequality $(c_2 + c_3)' \neq c_2 + c_3'$ as the additional literal.

Then we can use the finite descent rule:

$$8'. (c_1 + c_2) + 0 \neq \underline{c_1 + (c_2 + 0)} \wedge (c_1 + c_2) + 0' \neq (c_1 + (c_2 + 0))' \wedge ((c_1 + c_2) + 0)' \neq (c_1 + (c_2 + 0))' \vee c_2 + c_3' \neq (c_2 + c_3)'.$$

$$9. (3,8) c_1 + c_2 \neq c_1 + (c_2 + 0) \wedge (c_1 + c_2) + 0 \neq c_1 + (c_2 + 0) \wedge (c_1 + c_2) + 0' \neq (c_1 + (c_2 + 0))' \wedge ((c_1 + c_2) + 0)' \neq (c_1 + (c_2 + 0))' \vee c_2 + c_3' \neq (c_2 + c_3)'.$$

Now we can apply the one-place paramodulation rule to the left and right parts of the first inequality in 9:

$$9'. c_2 \neq c_2 + 0 \wedge (c_1 + c_2) + 0 \neq c_1 + (c_2 + 0) \wedge (c_1 + c_2) + 0' \neq (c_1 + (c_2 + 0))' \wedge ((c_1 + c_2) + 0)' \neq (c_1 + (c_2 + 0))' \vee c_2 + c_3' \neq (c_2 + c_3)'.$$

$$10.(3,9') c_2 + c_3' \neq (c_2 + c_3)'$$

11.(4,10) Empty. \square

Example 3.77. The axioms of arithmetic are given:

$$1. x_1' \neq 0.$$

$$2. x_1 = x_2 \vee x_1' \neq x_2'.$$

$$3. x_1 + 0 = x_1.$$

$$4. x_1 + x_2' = (x_1 + x_2)'.$$

We should prove the theorem $x_1 + x_2 = x_2 + x_1$.

The negation of this theorem is added to given axioms:

$$5. c_1 + c_2 \neq c_2 + c_1$$

The proof consists of the following steps.

$$6 (2,5) (c_1 + c_2)' \neq \underline{(c_2 + c_1)'} \wedge c_1 + c_2 \neq c_2 + c_1$$

$$7 (4r,6) c_1 + c_2' \neq (c_2 + c_1)' \wedge (c_1 + c_2)' \neq (c_2 + c_1)' \wedge c_1 + c_2 \neq c_2 + c_1$$

The first literal $c_1 + c_2' \neq (c_2 + c_1)'$ and the last literal $c_1 + c_2 \neq c_2 + c_1$ form the inductive pair if $(c_2 + c_1)' = c_2' + c_1$. So at (generalized) unification of these literals, the terminal resolvent has the inequality $(c_2 + c_1)' \neq c_2' + c_1$ as the additional literal. Then we can use the finite descent rule:

$$7' c_1 + 0 \neq \underline{0 + c_1} \wedge (c_1 + 0)' \neq (0 + c_1)' \vee (c_2 + c_1)' \neq c_2' + c_1$$

$$8. (3,7') c_1 \neq 0 + c_1 \wedge c_1 + 0 \neq 0 + c_1 \wedge (c_1 + 0)' \neq (0 + c_1)' \vee (c_2 + c_1)' \neq c_2' + c_1$$

$$9. (2,8) c_1' \neq (0 + c_1)' \wedge c_1 \neq 0 + c_1 \wedge c_1 + 0 \neq 0 + c_1 \wedge (c_1 + 0)' \neq (0 + c_1)' \vee (c_2 + c_1)' \neq c_2' + c_1$$

$$10. (4r,9r) 0 + c_1' \neq c_1' \wedge c_1' \neq (0 + c_1)' \wedge c_1 \neq 0 + c_1 \wedge c_1 + 0 \neq 0 + c_1 \wedge (c_1 + 0)' \neq (0 + c_1)' \vee (c_2 + c_1)' \neq c_2' + c_1$$

The literals $0 + c_1' \neq c_1'$ and $c_1 \neq 0 + c_1$ form the inductive pair. So we can use the finite descent rule:

$$10' 0 \neq 0 + 0 \wedge 0' \neq (0 + 0)' \wedge 0 + 0 \neq 0 + 0 \wedge (0 + 0)' \neq (0 + 0)' \vee (c_2 + c_1)' \neq c_2' + c_1$$

The inconsistent literal $0 + 0 \neq 0 + 0$ allows to remove \wedge -clause:

$$10'' (c_2 + c_1)' \neq \underline{c_2' + c_1}$$

$$11. (4r,10'') c_2 + c_1' \neq c_2' + c_1 \wedge (c_2 + c_1)' \neq c_2' + c_1$$

$$12. (2,11) (c_2 + c_1')' \neq (c_2' + c_1)' \wedge c_2 + c_1' \neq c_2' + c_1 \wedge (c_2 + c_1)' \neq c_2' + c_1$$

$$13. (4r,12r) c_2' + c_1' \neq (c_2 + c_1')' \wedge (c_2 + c_1')' \neq (c_2' + c_1)' \wedge c_2 + c_1' \neq c_2' + c_1 \wedge (c_2 + c_1)' \neq c_2' + c_1$$

The literals $c_2' + c_1' \neq (c_2 + c_1')'$ and $(c_2 + c_1)' \neq c_2' + c_1$ form the inductive pair. So we can use the finite descent rule once more:

$$13' \ (c_2 + 0)' \neq c_2' + 0 \wedge (c_2 + 0)' \neq (c_2' + 0)' \wedge c_2 + 0' \neq c_2' + 0$$

$$14. \ (3,13'r) \ (c_2 + 0)' \neq c_2' \wedge (c_2 + 0)' \neq c_2' + 0 \wedge (c_2 + 0)' \neq (c_2' + 0)' \wedge c_2 + 0' \neq c_2' + 0$$

The literal $(c_2 + 0)' \neq c_2'$ can be reduced to $c_2 + 0 \neq c_2$ (by the one-place paramodulation rule):

$$14'. \ c_2 + 0 \neq c_2 \wedge (c_2 + 0)' \neq c_2' + 0 \wedge (c_2 + 0)' \neq (c_2' + 0)' \wedge c_2 + 0' \neq c_2' + 0$$

15. (3,14') Empty.

The analogous proof in classic logic [8] uses the induction rule 3 times too, but corresponding theorems are formulated by intuition. In relational logic these theorems appear at proving. \square

3.2.7. Generalized Finite Descent.

The rule of induction is used basically for primitive recursive functions. These functions contain a recursive variable used as inductive in theorem proving.

But such inductive variable is absent, if we use recursive relations instead of recursive function.

As a rule, we construct recursive relations by iteration.

Before iteration we have a relation, which becomes initial approximation of required relation.

We construct the first approximation of the required relation at the first stage of the relation iteration (one stage includes several steps of iteration, see section 3.1.2).

Further, at the next stage of the iteration we construct the new approximation of the required relation.

The serial number of iteration stage is absent in received approximations. So this number can not be an inductive variable.

We could add a new variable for every approximation of recursive relation. The value of this variable becomes a number of iteration stage. Such variable could be used as inductive.

But there is a more simple way without an inductive variable. It put up the preorder - relations received at a stage n are less than the relations received at a stage $n + 1$. Then we can use the rule of infinite descent:

$$(\forall x_1 \mathcal{F}[x_1] \rightarrow (\exists x_2 \ x_2 \prec x_1 \wedge \mathcal{F}[x_2])) \rightarrow \forall x_1 \neg \mathcal{F}[x_1].$$

It means a formula does not deduce, if having the formula at any x_1 we deduce the same formula at $x_2 \prec x_1$.

But if all definitions in a theory are explicit or recursive, infinite descent becomes implicit finite.

In reality every inductive definition is recursive and consists of positive and negative definitions.

The positive definition consists of some formulas, a part of them are base of the inductive definition, the others are generating rules. But the negative definition consists of one formula in which transformed positive definitions are joined by disjunction (see section 1.1).

Using negation of a proving theorem we must show, every of these subformulas is inconsistent. For subformulas corresponding to generating rules, their inconsistency can be proved only by infinite descent. For subformulas corresponding to base of inductive definition, such restrictions are absent, but these subformulas are interpreted as descent to the base. So the proof of any property by infinite descent is always followed by the proof of this property by finite descent to the base.

Therefore, we shall name infinite descent as generalized finite descent.

Rule 3.78 of generalized finite descent. Let be given:

- a formula in the first normal form;
- \wedge -clause in this formula;
- subformula $L[a_1] \wedge L[a_2] \wedge a_2 \prec a_1$ in this \wedge -clause.

Then:

- this \wedge -clause can be removed (because it is a negation of a valid formula in logic with preorder).

Example 3.79. In the arithmetic set theory AS (see section 2) we have recursive definition of the proper class Pr :

$$Pr(x_1) \Leftrightarrow x_1 = N \vee (\exists x_2 \ x_1 = P(x_2) \wedge Pr(x_2)) \vee \exists x_2 \exists x_3 \ x_1 = x_2 \times x_3 \wedge Pr(x_2) \wedge Pr(x_3).$$

where P is the power-class function, " \times " - symbol of the direct products of classes (classes are divided into sets and proper classes, see section 2).

The positive definition of the proper class is divided into three formulas:

$$Pr(N).$$

$$Pr(P(x_1)) \vee \neg Pr(x_1).$$

$$Pr(x_1 \times x_2) \vee \neg Pr(x_1) \vee \neg Pr(x_2).$$

The first formula is a base of inductive definition, the other two formulas are generating rules.

Therefore, we have only one proper class N before iteration. We have two more proper classes $P(N)$ and $N \times N$ after the first stage. And we have 7 more proper classes $P(P(N))$, $P(N \times N)$, $N \times P(N)$, $P(N) \times N$, $N \times N \times N$, $P(N) \times N \times N$ and $N \times N \times P(N)$ after the second stage. All proper classes created at a given stage are equivalent, and all proper classes, created in previous stages, are less than they.

So this preorder is linear.

The formulas corresponding to the putting preorder must be added at reduction of recursive definition to the first normal form. It means the positive definition has more formulas:

$$x_1 \prec P(x_1) \vee \neg Pr(x_1).$$

$$x_1 \prec x_1 \times x_2 \vee \neg Pr(x_1) \vee \neg Pr(x_2).$$

$$x_2 \prec x_1 \times x_2 \vee \neg Pr(x_1) \vee \neg Pr(x_2).$$

$$x_1 \preceq x_2 \vee x_2 \preceq x_1$$

Using generalized finite descent we must prove the theorem $\langle x_1, x_2 \rangle \notin x_1 \vee \neg M(x_1) \vee \neg M(x_2)$, where M is the relation "to be set".

The next two axioms are lemmas. We give these lemmas without proving for simplification.

$$1. x_1 \notin x_2 \vee \neg M(x_2) \vee Pr(a_1) \wedge x_1 \in a_1 \wedge x_2 \subset a_1.$$

$$2. \langle x_1, x_2 \rangle \notin N.$$

One more axiom is the negative definition of proper classes. The first member of this axiom is a base of inductive definition, the other two members are inference rules.

$$3. \neg Pr(x_1) \vee x_1 = N \vee x_1 = P(a_1) \wedge Pr(a_1) \vee x_1 = a_1 \times a_2 \wedge Pr(a_1) \wedge Pr(a_2).$$

The next two axioms put up the preorder:

$$4. x_1 \prec P(x_1) \vee \neg Pr(x_1).$$

$$5. x_1 \prec x_1 \times x_2 \vee \neg Pr(x_1) \vee \neg Pr(x_2).$$

The last axiom is the negation of the theorem:

$$6. \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2).$$

Proof.

$$7. (1,6) \neg M(c_1) \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \vee$$

$$Pr(a_3) \wedge \langle c_1, c_2 \rangle \in a_3 \wedge c_1 \subset a_3 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2).$$

The first member of this formula must be removed because it has subformula $\neg M(c_1) \wedge M(c_1)$:

$$7'. Pr(a_3) \wedge \langle c_1, c_2 \rangle \in a_3 \wedge c_1 \subset a_3 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2).$$

$$8. (3,7') a_3 = N \wedge \langle c_1, c_2 \rangle \in a_3 \wedge c_1 \subset a_3 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_3) \vee a_3 = P(a_4) \wedge Pr(a_4) \wedge \langle c_1, c_2 \rangle \in a_3 \wedge c_1 \subset a_3 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_3) \vee$$

$$a_3 = a_4 \times a_5 \wedge Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_3 \wedge c_1 \subset a_3 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_3).$$

In this formula the marked literal $Pr(a_3)$ is replaced in the end of the \wedge -clause because after resolution marked literals become non-active. But we place active literals in the beginning of \wedge -clauses as a rule.

Using equation properties in \wedge -clauses we get:

$$8'. \langle c_1, c_2 \rangle \in N \wedge c_1 \subset N \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(N) \vee$$

$$Pr(a_4) \wedge \langle c_1, c_2 \rangle \in P(a_4) \wedge c_1 \subset P(a_4) \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(P(a_4)) \vee$$

$$Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5)$$

$$9. (2,8') Pr(a_4) \wedge \langle c_1, c_2 \rangle \in P(a_4) \wedge c_1 \subset P(a_4) \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(P(a_4)) \vee$$

$$Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5)$$

The subformula $Pr(a_4) \wedge Pr(P(a_4))$ allows to use the generalized finite descent if $a_4 \prec P(a_4)$. So at (generalized) unification the terminal resolvent has $a_4 \not\prec P(a_4)$ as the additional literal (see below). Then we can replace the first \wedge -clause by the same clause with this additional literal:

$$9'. a_4 \not\prec P(a_4) \wedge Pr(a_4) \wedge \langle c_1, c_2 \rangle \in P(a_4) \wedge c_1 \subset P(a_4) \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(P(a_4)) \vee$$

$$Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5)$$

$$10. (4,9') \neg Pr(a_4) \wedge a_4 \not\prec P(a_4) \wedge Pr(a_4) \wedge \langle c_1, c_2 \rangle \in P(a_4) \wedge c_1 \subset P(a_4) \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(P(a_4)) \vee$$

$$Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5)$$

The first \wedge -clause of the formula can be removed because this clause has the subformula $\neg Pr(a_4) \wedge Pr(a_4)$:

$$10'. Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5).$$

The subformula $Pr(a_4) \wedge Pr(a_4 \times a_5)$ allows to use the generalized finite descent if $a_4 \prec a_4 \times a_5$. So we can replace the \wedge -clause by the same clause with the additional literal $a_4 \not\prec a_4 \times a_5$:

$$10''. a_4 \not\prec a_4 \times a_5 \wedge Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5).$$

$$11. (5,10'') \neg Pr(a_4) \wedge a_4 \not\prec a_4 \times a_5 \wedge Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5) \vee$$

$$\neg Pr(a_5) \wedge a_4 \not\prec a_4 \times a_5 \wedge Pr(a_4) \wedge Pr(a_5) \wedge \langle c_1, c_2 \rangle \in a_4 \times a_5 \wedge c_1 \subset a_4 \times a_5 \wedge \langle c_1, c_2 \rangle \in c_1 \wedge M(c_1) \wedge M(c_2) \wedge Pr(a_4 \times a_5).$$

Both \wedge -clauses of this formula can be removed because they have subformulas $\neg Pr(a_4) \wedge Pr(a_4)$ and $\neg Pr(a_5) \wedge Pr(a_5)$:

11'. Empty. \square

It is necessary to mark that we use the generalized unification rule, but this rule can add only inequalities. We must show this rule can add other negation too.

Let $E[a_i]$ be an \wedge -clause and $L[a_i]$ be a literal with the same a -variable a_i . We must show

$$E[a_i] \Leftrightarrow L[a_i] \wedge E[a_i] \vee \neg L[a_i] \wedge E[a_i]$$

We can move the implicit existential quantifiers from \wedge -clauses of the right part in the beginning of this part:

$$(\exists a_i E[a_i]) \Leftrightarrow \exists a_i L[a_i] \wedge E[a_i] \vee \neg L[a_i] \wedge E[a_i]$$

Using the distributivity of disjunction and conjunction we get:

$$(\exists a_i E[a_i]) \Leftrightarrow \exists a_i E[a_i] \wedge (L[a_i] \vee \neg L[a_i])$$

In the last parentheses we have the valid formula. Removing this formula we have one more the valid formula $(\exists a_i E[a_i]) \Leftrightarrow \exists a_i E[a_i]$. In this formula all quantifiers are explicit.

Rule 3.80 of addition of literal in \wedge -clause. We have:

- \wedge -clause $L_1(\dots) \wedge \dots \wedge L_n(\dots)$;
- literal $L_0(\dots)$.

Then

- we can replace the given \wedge -clause by two \wedge -clauses

$$L_0(\dots) \wedge L_1(\dots) \wedge \dots \wedge L_n(\dots) \vee \neg L_0(\dots) \wedge L_1(\dots) \wedge \dots \wedge L_n(\dots).$$

3.3. Conclusion.

The account of the finite logic was made informally, since the formal account of even parts of the rules by means of logic programming would have many complex formulas, which are difficult for understanding. But all these rules will be obligatory formalized and put in a special section of the journal dedicated to publications of programs (and rules) by means of logic programming.

Theorem proving is given in informal way too. But a series of articles will be dedicated to the formal proving.

It is necessary also to mark, till now there is no computer realization of construction of the theories and models. Such programming systems as Prolog, Datalog and others have little in common with logic programming stated in this "Introduction".

Many problems of logic programming are stated for the first time. But it is principal, the account of logic programming was constructed in accordance with classic logic. It allowed to cut off rather vast investigations that are wrong or unimportant.

Almost all problems of the theorem proving (besides the usual resolution and paramodulation rules) are stated for the first time, too.

References

- [1]. E. Codd. A data base sublanguage founded on the relational calculus *Proc. 1971 ACM SIGFIDET workshop on data description, access and control*.
- [2]. A.Colmerauer, H.Kanouï, R.Pasero, P.Roussel, Un système de communication homme-machine en Français, *Tech. Rep.*,Groupe d'Intelligence Artificielle, université d'Aix Marseille II, Luminy, France, 1973.
- [3]. J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM* 1965, 12.
- [4]. M.A. Malkov. Relational logic and arithmetic. *Relational logic*, 2001, 1.
- [5]. M.A. Malkov. Relational propositional logic. *Relational logic*, 2001, 1.
- [6]. C.-L. Chang, R.C.-H. Lee. *Symbolic logic and mechanical theorem proving*. Academic press, New York (1973).
- [7]. C.J. Date. *An introduction to database systems*. Addison-Wesley (1977).
- [8]. E. Mendelson. *Introduction to mathematical logic*. Nostrand company (1964).