Semantics-based Logics over Hierarchical Nominative Data

M(N).S. Nikitchenko[1] and S.S. Shkilniak[1]

[1] Department of Theory and Technology of Programming
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Tel.: +38044 2590519
`nikitchenko@unicyb.kiev.ua`

Abstract. In the paper new logics oriented on hierarchical data are developed. Algebras of partial predicates over such data with special compositions as operations form a semantic base for constructed logics. Characteristic property of such logics is the usage of composite names in their languages. Semantic properties of these logics are studied; corresponding sequent calculi are defined, their soundness and completeness are proved for logics of renominative level.

Keywords: partial predicates, program semantics, nominative data, composition, logic, soundness, completeness.

Key Terms. Research, MathematicalModel, FormalMethods, MachineIntelligence

1 Introduction

Mathematical logic is widely used in formal program development, analysis, and verification. Still, some discrepancies can be admitted between traditional logic and problems to be solved. For example:

- semantics of programs is presented by partial functions, whereas in traditional logic total functions and predicates are usually considered;
- programming languages have a developed system of data types, whereas traditional logic prefers to operate with simple unstructured types (sorts);
- semantic aspects of programs prevail over syntactical aspects, whereas in traditional logic we have the inverse situation.

Discrepancies mentioned above complicate the usage of logic for program development and verification. In this paper we propose to take program models as an initial point and construct logics semantically based on such models.

To realize this idea we should first construct adequate models of programs. To tackle this problem we use composition-nominative approach to program formalization [1], which aims to construct a hierarchy of program models of various abstraction levels and generality. The main principles of the approach are the following.

- *Development principle* (from abstract to concrete): program notions should be introduced as a process of their development that starts from abstract understanding capturing essential program properties and proceeds to more concrete considerations.
- *Principle of integrity of intensional and extensional aspects*: program notions should be presented in the integrity of their intensional and extensional aspects. The intensional aspects in this integrity play a leading role.
- *Principle of priority of semantics over syntax*: program semantic and syntactical aspects should be first studied separately, then in their integrity in which semantic aspects prevail over syntactical ones.
- *Compositionality principle*: programs can be constructed from simpler programs (functions) with the help of special operations, called compositions, which form a kernel of program semantics structures.
- *Nominativity principle*: nominative (naming) relations are basic ones in constructing data and programs.

Here we have presented only principles relevant to the topic of the article; richer system of principles is developed in [2]. The above principles specify program models as *composition-nominative systems* (CNS) [1]. Such a system may be considered as a triple of simpler systems: composition, description, and denotation systems. A composition system defines semantic aspects of programs, a description system defines program descriptions (syntactical aspects), and a denotation system specifies meanings (referents) of descriptions. We consider semantics of programs as partial functions over class of data processed by programs; compositions are n-ary operations over functions. Thus, composition system can be specified as two algebras: data algebra and functional algebra.

Functional algebra is the main semantic notion in program formalization. Terms of this algebra define syntax of programs (descriptive system), and ordinary procedure of term interpretation gives a denotation system.

CNS can be used to construct formal models of various programming, specification, and database languages [1–4]. The program models presented by CNS are mathematically simple, but specify program semantics rather adequately; program models are highly parametric and can in a natural way represent programs of various abstraction levels; there is a possibility to introduce on a base of CNS the notion of special (abstract) computability and various axiomatic formalisms [5–7].

CNS are classified in accordance with levels of abstraction of their parameters: data, functions, and compositions. In this article levels of program models are induced by abstraction levels of data.

Data are considered at three levels: abstract, Boolean, and nominative. At the abstract level data are treated as "black boxes", thus no information can be extracted. At the Boolean level to abstract data new data considered as "white boxes" are added. Usually, these are logical values T (true) and F (false) from the set *Bool*. At the nominative level data are considered as "grey boxes", constructed of "black" and "white boxes" with the help of naming relations. The last level is the most interesting for programming. Data of this level are called *nominative data*. The class of *nominative data* over a set of names V and class of basic values W can be defined inductively or as the least fixed point of the recursive definition

$ND(V,W) = W \cup (V \xrightarrow{m} ND(V,W))$, where $V \xrightarrow{m} ND(V,W)$ is the class of partial multi-valued (non-deterministic) functions.

To present nominative data we use the form $d=[v_i \mapsto a_i \mid i \in I]$. *Nominative membership relation* is denoted by $\in$. Thus, $v_i \mapsto a_i \in d$ means that the value of v_i in d is defined and is equal to a_i.

The class $ND(V,W) \setminus W$ is called the class of *proper nominative data*, or *hierarchical nominative data*; data from the class $V \xrightarrow{m} W$ will be called *flat nominative data*.

Concretizations of nominative data can represent various data structures, such as records, arrays, lists, relations, etc. [1,4]. For example, a set $\{s_1, s_2, ..., s_n\}$ can be presented as nominative data $[1 \mapsto s_1, 1 \mapsto s_2, ..., 1 \mapsto s_n]$, where 1 is treated as a standard name. Thus, we can formulate the following *data representation principle*: program data can be presented as concretizations of nominative data.

The levels of data abstraction formulated above may be treated as data intensionals. They respectively specify three levels of semantics-based program models: abstract, Boolean, nominative. The models of each level constitute extensionals of that level intensional. Program models of abstract level are very poor (actually, only sequencing compositions can be defined). Program models of Boolean level are richer and permit to define structured programming constructs (sequence, selection, and repetition). This level is still too abstract and does not explicitly specify data variables. At last, models of nominative level permit to formalise compositions of traditional programming. This level (its intensional) involves variables of different types. Consider, for example, a simple educational programming language WHILE [8], which is based on three main syntactical components: arithmetic expressions, Boolean expression, and statements. States of WHILE programs are considered as partial functions from the set V of variables to the set Z of values and here are denoted by ${}^{V}Z (= V \to Z)$. Thus, semantics of these components is the following: arithmetic expressions specify functions of the type ${}^{V}Z \to Z$ (we call them partial quasiary functions), Boolean expressions define functions of the type ${}^{V}Z \to Bool$ (partial quasiary predicates), statements specify functions of the type ${}^{V}Z \to {}^{V}Z$ (partial bi-quasiary functions). Note that in our terminology ${}^{V}Z$ is a class of single-valued flat nominative data.

Example 1. Consider a Boolean expression $x<y$. Its semantics is formalized as a partial quasiary predicate $less : {}^{V}Z \to Bool$. This predicate is undefined on flat nominative data $[x \mapsto 5, u \mapsto 4]$ (we write $less([x \mapsto 5, u \mapsto 4])\uparrow$), is defined on $[x \mapsto 5, u \mapsto 4, y \mapsto 2]$ with value F (we write $less([x \mapsto 5, u \mapsto 4, y \mapsto 2])\downarrow = F$). Note that if a value of *less* is defined on some data, then the predicate is defined with the same value on any extension of this data. Thus, $less([x \mapsto 5, u \mapsto 4, y \mapsto 2, v \mapsto 4])\downarrow = F$, $x, u, y, v \in V$. This property is called *equitonicity* (a special case of monotonicity). A specific new composition is renomination $R^{v_1,...,v_n}_{x_1,...,x_n}$,

e.g. $(R^{x,y}_{y,v}(less))([x \mapsto 5, u \mapsto 4, y \mapsto 2, v \mapsto 4]) = less([x \mapsto 2, u \mapsto 4, y \mapsto 4, v \mapsto 4]) = T$.

More elaborated programming languages work with hierarchical nominative data. In such languages composite names like $x_1.x_2. \ldots .x_n$ are used to access data components. The details can be found in [2].

Having described program models of various abstraction levels, we can now start developing semantics-based logics which correspond to such models. Such logics will be called *composition-nominative logics* (CNL). Analysis of constructed program models shows that the main semantic notion of mathematical logic – the notion of predicate – can be defined at the Boolean level. At this level predicates are considered as partial functions from a class of abstract data A (with abstract intensional) to *Bool*. In this case such compositions as disjunction $\vee$, negation $\neg$, etc, can be defined. These compositions are derived from Kleene's strong connectives [9]. Thus, the main semantic objects are algebras of partial predicates of the type $<A\to Bool;\ \vee, \neg>$. The obtained logics may be called propositional logics of partial predicates. Such logics are rather abstract, therefore their further development is required at the nominative level. As was mentioned earlier, at this level we have two sublevels determined respectively by flat and hierarchical nominative data.

Three kinds of logics can be constructed from program models at the flat nominative data level:

- logics, which use only partial quasiary predicates (pure predicate logic);
- logics, which use additionally partial quasiary functions (predicate-function logics);
- logics, which use also bi-quasiary functions (program logics).

The first type of logics will generalize classical pure predicate logics, the second type – classical predicate logic (with functions and equality), and the third type can present various logics, which use program constructs.

Here we give a short characteristic only to composition-nominative pure predicate logics; predicate-function logics are described in [3]; as to composition-nominative program logics some initial variants are presented in [3, 7].

From semantic point of view the main distinction of CNL from classical first-order logics is usage of partial quasiary predicates instead of total n-are predicates; this leads to algebras of quasiary predicates with compositions as operations. From syntactical point of view formulas of CNL are simply terms of algebras of quasiary predicates.

The main compositions that can be additionally specified at the nominative level are renomination $R^{v_1,\dots,v_n}_{x_1,\dots,x_n}$ (denoted also $R^{\bar{v}}_{\bar{x}}$) and quantification $\exists x$. These compositions use subject names as parameters. CNL of renominative level are based on algebras of the type $<{}^{V}\!A\to Bool;\ \vee, \neg,\ R^{\bar{v}}_{\bar{x}}>$, CNL of quantifier level – $<{}^{V}\!A\to Bool;\ \vee, \neg,\ R^{\bar{v}}_{\bar{x}},\ \exists x>$. Properties of these algebras determine calculi for corresponding logics.

Note, that renomination (primarily in syntactical aspects) is widely used in classical logic, lambda-calculus, and specification languages like Z-notation [10], B [11], TLA [12], etc. Here we will give explicit semantic definition of this operation (cf. with [13]).

To preserve properties of classical first-order logic we should restrict the class ${}^{V}\!A\to Bool$ of quasiary predicates. Namely, we introduce a class of equitone predicates and its different variations such as maxitotal equitone, local-equitone, equicompatible, and local-equicompatible classes [3]. Logics based on equitone and maxitotal equitone predicates are the "closest" generalization of classical first-order logic that preserve its main properties. These logics are called *neoclassical logics* [3].

The current article continues investigations of pure predicate logics over hierarchical nominative data initiated in [14]. Here we prove soundness (correctness) and completeness of the constructed logics. The distinctive feature of such logics is the usage of composite names of the form $x_1.x_2. \ldots .x_n$ as parameters of renomination and quantification compositions.

The article is structured as follows: the first section is introduction, in the second section operations over hierarchical data are introduced and their properties are studied, the third section is devoted to compositions over predicates. In the fourth section semantic models and corresponding languages of logics are described, and the fifth section is devoted to definition of sequent calculi for some of the described logics.

Notions not defined here we interpret in sense of [3].

2 Hierarchical Nominative Data

Class of hierarchical nominative data $ND(V, A)$ over classes of basic names V and basic values A is defined inductively:

1) $ND_0(V,A) = A$ – nominative data of rank 0;

2) $ND_{k+1}(V,A) = A \cup (V \xrightarrow{n} ND_k(V,A))$ – nominative data of rank less or equal to k+1.

Then $ND(V,A) = \bigcup\limits_{k \geq 0} (V \xrightarrow{n} ND_k(V,A))$.

Here $V \xrightarrow{n} ND_k(V,A)$ is the set of all finite single-valued mappings from V to $ND_k(V,A)$. Note, that we restrict nominative data to be single-valued mappings. This guaranties unambiguity of naming for data components. An empty nominative data has rank 0.

The set of *hierarchical nominative data* is defined as follows: $HD(V,A) = ND(V,A) \setminus A$.

The value of name u in data d is equal to $d(u)$, but we also write $u{:}d$ in style of denaming operation. For a composite name $u = y_1.y_2. \ldots .y_n$ notation $u{:}d$ means $y_n{:}(\ldots(y_2{:}(y_1{:}d))\ldots)$. We drop a component $x \mapsto u{:}\delta$, if $u{:}\delta$ is undefined.

Hierarchical data can be represented also as oriented trees with edges labeled by basic names and leafs labeled by basic values.

Any hierarchical data d can be represented as a flat nominative data with composite names – elements of the set V^+. These composite names are non-empty words in the alphabet V formed by concatenation "." of basic names along the path from the root to leafs in the tree representing d.

Example 2. Let $[x \mapsto [y \mapsto 1, z \mapsto 2], y \mapsto [x \mapsto 3, y \mapsto [x \mapsto 0, y \mapsto 0, z \mapsto 1]], z \mapsto 2, u \mapsto [x \mapsto [x \mapsto 0, u \mapsto 1], z \mapsto 3]]$ be hierarchical data. Its flat representation is

$[x.y \mapsto 1, x.z \mapsto 2, y.x \mapsto 3, y.y.x \mapsto 0, y.y.y \mapsto 0, y.y.z \mapsto 1, z \mapsto 2, u.x.x \mapsto 0, u.x.u \mapsto 1, u.z \mapsto 3]$.

Such representations are called *flat normal forms* (*FNF*) of hierarchical data. Due to the unambiguity of naming all (composite) names of FNF must be different;

moreover, they should be *incomparable*. Now it is possible to write $[x.y \mapsto \alpha, x.u \mapsto \beta, \ldots]$ in place of $[x \mapsto [y \mapsto \alpha, u \mapsto \beta, \ldots]]$.

Let us formulate some definitions and properties of hierarchical data used in further proofs. From now on, names are considered as composite names from V^+ unless explicitly stated that they belong to V.

A *prefix* of a word $u \in V^+$ is any word x such that $u = x.y$ for some $y \in V^*$. If $u \neq x$, we call x a *strict prefix*. We write $x \leq u$ $(x < u)$, if x is a prefix (strict prefix) of u. Words x and u are *comparable* $(x <> u)$, if $x \leq u$ or $u \leq x$; otherwise they are *incomparable* $(x \div u)$. Sets of names X and Y are *incomparable* $(X \div Y)$, if $x \div y$ for all $x \in X$ and $y \in Y$.

We call a composite name as a *full* name of d, if it coincides with some path from a root in the tree determined by d. If this path reaches a leaf, then the name is called *terminal*. We define the set of full names by $fn(d) = \{u \mid u{:}d\downarrow\}$; the set of terminal names by $tn(d) = \{u \mid u{:}d\downarrow \in A\}$.

Hierarchical data d_1 and d_2 are *disjoint*, if $x \div y$ for any $x \in tn(d_1)$ and $y \in tn(d_2)$. The union of disjoint data we denote by "+".

Parametric operation of *deletion* of data components, the names of which are comparable with given names $x_1,\ldots,x_n$, is defined via FNF as follows:

$$\|_{-x_1,\ldots,x_n}(d) = [u \mapsto a \in d \mid u \text{ is terminal and } x_1 \div u, \ldots, x_n \div u].$$

For basic data $a \in A$, $\|_{-x_1,\ldots,x_n}(a)$ is undefined.

In the sequel instead of $\|_{-x_1,\ldots,x_n}(d)$ we write $d\,\|_{-x_1,\ldots,x_n}$.

Example 3. Let d be a hierarchical data from example 2. Then:

$$d\,\|_{-x,\,u} = [y.x \mapsto 3, y.y.x \mapsto 0, y.y.y \mapsto 0, y.y.z \mapsto 1, z \mapsto 2];$$
$$d\,\|_{-x.z,\, y.y,\, z.y,\, u.x,u} = [x.y \mapsto 1, y.x \mapsto 3, u.x.x \mapsto 0, u.z \mapsto 3].$$

When using the symbol "+" we drop brackets "[" and "]", e.g. instead of $d\,\|_{-u} + [u \mapsto u{:}d\,\|_{-v}]$ we write $d\,\|_{-u} + u \mapsto u{:}d\,\|_{-v}$.

Proposition 1.

1) $d\,\|_{-z,u,x_1,\ldots,x_n} = d\,\|_{-z,x_1,\ldots,x_n}$, if $z \leq u$;

2) $d = d\,\|_{-x} + x \mapsto x{:}d$, if $x \in V$;

3) $(d\,\|_{-x})\|_{-x.y} = (d\,\|_{-x.y})\|_{-x} = d\,\|_{-x}$;

4) $(d_1 + d_2)\|_{-u} = d_1\|_{-u} + d_2\|_{-u}$;

5) $d\,\|_{-u.v} = d\,\|_{-u} + u \mapsto u{:}d\,\|_{-v}$;

6) $d \supseteq d\,\|_{-u} + u \mapsto u{:}d$.

For a composite u the property $d = d\,\|_{-u} + u \mapsto u{:}d$ may fail.

Example 4. Let $d = [u \mapsto 0, z.x \mapsto 0, z.y \mapsto 1]$. Then $d\,\|_{-u.v} = [z.x \mapsto 0, z.y \mapsto 1]$, and $d\,\|_{-u.v} + u.v \mapsto u.v{:}d = d\,\|_{-u.v} \neq d$, because $u{:}d$ is a basic value and $u.v{:}d$ is undefined.

Proposition 2. Let $u \div \{x_1,\ldots,x_n\}$, then $d\,\|_{-u,x_1,\ldots,x_n} = (d\,\|_{-u})\|_{-x_1,\ldots,x_n}$.

In particular, if $z \div u$, then $d\,\|_{-u,\,z} = (d\,\|_{-u})\|_{-z} = (d\,\|_{-z})\|_{-u}$.

In the general case we have that $d\,\|_{-u_1,\ldots,u_m,x_1,\ldots,x_n} = (d\,\|_{-u_1,\ldots,u_m})\|_{-x_1,\ldots,x_n}$ if $\{u_1,\ldots,u_m\} \div \{x_1,\ldots,x_n\}$.

Proposition 3. 1) $x : (d\,\|_{-z_1,\ldots,z_n} + x \mapsto h) = h$; in particular, $x : (x \mapsto h) = h$;

2) $x : (d\,\|_{-z_1,\ldots,z_n}) = x : d$, if $x \div \{z_1,\ldots,z_n\}$;

3) $x : (d\,\|_{-x,z_1,\ldots,z_n})\uparrow$.

Using the propositions 1–3, it is possible to represent $d\|_{-x_1,\ldots,x_n}$, where $x_1,\ldots,x_n\in V^{+}$, with an expression in some standard form, which uses only operations of deletion $\|_{-v_1,\ldots,v_m}$ with simple names $v_1,\ldots,v_m\in V$, union $+$, naming $y_1.y_2.\ldots.y_k\mapsto$ and denaming $u_1{:}u_2{:}\ldots u_l{:}$ (here $y_1,\ldots,y_k$, $u_1,\ldots,u_l\in V$). Details are omitted here.

Example 5. $d\|_{-z.x,\, z.y,\, u.x.y} = d\|_{-z,\, u} + z\mapsto z{:}d\|_{-x,\, y} + u\mapsto u{:}d\|_{-x} + u.x\mapsto x{:}u{:}d\|_{-y}$.

Operation of renomination $r^{v_1,\ldots,v_n}_{x_1,\ldots,x_n} : HD(V,A)\to HD(V,A)$ we define as follows:

$$r^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}(d) = d\,\|_{-v_1,\ldots,v_n} + v_1 \mapsto x_1 : d + \ldots + v_n \mapsto x_n : d\ .$$

Here all names $v_1,\ldots,v_n$ should be pairwise incomparable. We see that the result of renomination can be presented uniquely in the standard form.

Example 6. $r^{v.x,v.y,u.x.y}_{u.y,y.x,v.x}(d) = d\|_{-v,\, u} + v\mapsto v{:}d\|_{-x,\, y} + u\mapsto u{:}d\|_{-x} +$

$+ u.x\mapsto x{:}u{:}d\|_{-y} + v.x\mapsto y{:}u{:}d + v.y\mapsto x{:}y{:}d + u.x.y\mapsto x{:}v{:}d.$

Note, that renomination is monotone: if $d\subseteq h$, then $r^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}(d)\subseteq r^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}(h)$.

To present convolution of renominations we use the standard form.

Example 7. $r^{z}_{u}(r^{u.v}_{x}(d)) = r^{z}_{u}(d\,\|_{-u.v} + u.v \mapsto x : d) =$

$= r^{z}_{u}(d\,\|_{-u} + u \mapsto u : d\,\|_{-v} + u.v \mapsto x : d) =$

$d\,\|_{-u,z} + u.v \mapsto x : d + z \mapsto u : d\,\|_{-v} + z.v \mapsto x : d\ .$

Thus, situation for hierarchical data is more difficult than for flat data for which convolution of renominations can be presented as one new renomination [3].

Example 8. $r^{x.v}_{z.y}(r^{u.v,\, z}_{x,\, x.y}(d)) =$

$= r^{x.v}_{z.y}(d\,\|_{-u.z} + u \mapsto u : d\,\|_{-v} + u.v \mapsto x : d + z \mapsto y : x : d) =$

$= d\,\|_{-u.z,x} + u \mapsto u : d\,\|_{-v} + u.v \mapsto x : d + z \mapsto y : x : d + x \mapsto x : d\,\|_{-v} +$

$x.v \mapsto y : y : x : d.$

3 Compositions of Predicates over Hierarchical Data

From semantic point of view the notion of predicate is one of the basic concepts of logic.

By a *predicate* P on D we understand a single-valued partial function of the type $D\to Bool$. The truth and falsity domains of P are respectively $\boldsymbol{T}(P) = \{d\in D \mid P(d)\!\downarrow = T\}$ and $\boldsymbol{F}(P) = \{d\in D \mid P(d)\!\downarrow = F\}$. A predicate P is *irrefutable*, or partially true, if $\boldsymbol{F}(P) = \varnothing$.

Compositions determine universal methods of predicate construction; they form the kernel of logic of corresponding type.

At the *propositional* level data are treated as abstract, therefore predicates are interpreted as functions from A to *Bool*, where A is an abstract class. Basic propositional compositions are disjunction $\vee$ and negation $\neg$ (P, $Q\in A\to Bool$, $d \in A$):

$$(P \vee Q)(d) = \begin{cases} T, \text{ if } P(d)\downarrow = T \text{ or } Q(d)\downarrow = T, \\ F, \text{ if } P(d)\downarrow = F \text{ and } Q(d)\downarrow = F, \\ \quad \text{undefined in other cases.} \end{cases}$$

$$(\neg P)(d) = \begin{cases} T, \text{ if } P(d)\downarrow = F, \\ F, \text{ if } P(d)\downarrow = T, \\ \text{undefined, if } P(d)\uparrow. \end{cases}$$

At the *nominative* level data are constructed from a set of subject names and a class of subject values. In this work logics of partial predicates over hierarchical nominative data at renominative and quantifier level are investigated.

A function of the form $P : HD(V, A) \to Bool$ is called a *hierary predicate* on $HD(V, A)$. We denote the class of hierary predicates on $HD(V, A)$ by PrH^{V_A}.

The name $x \in V$ is *strictly unessential* for a hierary predicate P on $HD(V, A)$, if for arbitrary d, $\alpha \in HD(V, A)$ we have $P(d\|_{-x} + x \mapsto \alpha) = P(d\|_{-x})$. The notion of unessential name is an analogue of fresh name in classical and nominal logics [15].

A predicate $P : HD(V, A) \to Bool$ is called *equitone*, if for arbitrary $d, d' \in HD(V, A)$ conditions $d \subseteq d'$ and $P(d)\downarrow$ imply $P(d')\downarrow = P(d)$.

At the renominative level to propositional compositions we add renomination composition $R^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}$ defined by the formula

$$R^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}(Q)(d) = Q(r^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}(d)) = Q(d \|_{-v_1,\ldots,v_n} + v_1 \mapsto x_1 : d + \ldots + v_n \mapsto x_n : d).$$

Using vector notation, we can formulate the following properties of renomination:

R$\vee$) $R^{\bar{v}}_{\bar{x}}(P \vee Q) = R^{\bar{v}}_{\bar{x}}(P) \vee R^{\bar{v}}_{\bar{x}}(Q)$; R$\neg$) $R^{\bar{v}}_{\bar{x}}(\neg P) = \neg R^{\bar{v}}_{\bar{x}}(P)$.

The properties of R$\to$, R&, R$\leftrightarrow$ can be written down analogously.

RR) $R^{\bar{v}}_{\bar{x}}(R^{\bar{u}}_{\bar{y}}(P)(d) = P(r^{\bar{u}}_{\bar{y}}(r^{\bar{v}}_{\bar{x}}(d)))$ for each $d \in HD(V, A)$.

RSN) $R^{y,\bar{v}}_{z,\bar{x}}(P) = R^{\bar{v}}_{\bar{x}}(P)$, if $y \in V$ is strictly unessential for P.

RT) $R^{z,\bar{v}}_{z,\bar{x}}(P) = R^{\bar{v}}_{\bar{x}}(P)$ under condition $z \in V$.

In the case of equitone predicates for composite names we have:

RTE) $R^{u,\bar{v}}_{u,\bar{x}}(P) \cong R^{\bar{v}}_{\bar{x}}(P)$, where $\cong$ is weak equality.

At the *quantifier* level basic compositions are $\vee$, $\neg$, $R^{\bar{v}}_{\bar{x}}$, $\exists x$.

Contrary to traditional case quantified names can be composite; quantification is possible both over all hierarchical or only over basic data. In this work we consider quantification over hierarchical data. Composition of existential quantification is defined in the following way:

$$\exists x P(d) = \begin{cases} T, \text{ if there exists } \beta \in ND(V, A) : \; P(d \|_{-x} + x \mapsto \beta)\downarrow = T, \\ F, \text{ if } P(d \|_{-x} + x \mapsto \alpha)\downarrow = F \text{ for all } \alpha \in ND(V, A), \\ \quad \text{undefined in all other cases.} \end{cases}$$

Composition of universal quantification is defined by formula $\forall xP = \neg\exists x\neg P$.

Theorem 1. The class of equitone predicates over hierarchical data is closed under compositions $\vee$, $\neg$, $R^{\bar{v}}_{\bar{x}}$, $\exists x$, $\forall x$.

Main properties of compositions $\exists x$ and $\forall x$ are the following.

1. If x and y are incomparable then $\exists x\exists yP = \exists y\exists xP$ and $\forall x\forall yP = \forall y\forall xP$.
2. Absorption of external quantifier by internal with the same name:
$$\exists x\exists xP = \exists xP;\ \ \forall x\exists xP = \exists xP;\ \ \exists x\forall xP = \forall xP;\ \ \forall x\forall xP = \forall xP.$$
3. Absorption of external quantifier by internal with more general name:
$$\exists x.y\exists xP = \exists xP;\ \ \forall x.y\exists xP = \exists xP;\ \ \exists x.y\forall xP = \forall xP;\ \ \forall x.y\forall xP = \forall xP.$$
At the same time $\exists x\exists x.yP$, $\forall x\exists x.yP$, $\exists xP$, $\exists x.yP$ are all different;
$\exists x\forall x.yP$, $\forall x\forall x.yP$, $\forall xP$, $\forall x.yP$ are all different.
4. Absorption of the quantified name by more general upper name of renomination:
$$\exists x.y(R^{x,\bar{u}}_{z,\bar{v}}P) = R^{x,\bar{u}}_{z,\bar{v}}(P),\ \text{if}\ x.y \div \{z,\bar{u},\bar{v}\}.$$
5. Absorption of the upper name of renomination by more general quantifier:
$$R^{\bar{y},\bar{u}}_{\bar{z},\bar{v}}(\exists xP) = R^{\bar{u}}_{\bar{v}}(\exists xP),\ \text{if } x \text{ is a prefix of names from } \bar{y} \text{ and } x \div \{\bar{u}\}.$$
At the same time $\exists x.y(R^{x}_{z}P) \neq R^{x}_{z}(\exists x.yP)$ and $\exists x.y(R^{x}_{x.y.v}P) \neq R^{x}_{x.y.v}(P)$;
$$R^{x.y}_{u}(\exists xP) \neq \exists x(R^{x.y}_{u}(P)),\ R^{x.y}_{u}(\exists xP) \neq R^{x.y}_{u}(P),\ \exists x(R^{z}_{x.u}P) \neq R^{z}_{x.u}(\exists xP).$$
6. $\exists z(R^{\bar{u}}_{\bar{v}}P) = R^{\bar{u}}_{\bar{v}}(\exists zP)$, if $z \div \{\bar{u},\bar{v}\}$.

Properties 4–6 can be rephrased for universal quantification.

Let us note that some properties valid in classical logic fail for the class of equitone predicates over hierarchical data.

Example 9. Let predicate τ_x be defined by the following formula:
$$\tau_x(d) = \begin{cases} T, \text{ if } d(x)\downarrow\in A, \\ F, \text{ if } d(x)\downarrow\notin A, \\ \text{undefined in all other cases.} \end{cases}$$

It is clear that τ_x is equitone. By definitions of compositions $\exists x$ and $\forall xP$ we have that $\exists x.v\,\tau_x(d) = \forall x.v\,\tau_x(d) = F$ for each $d\in HD(V,A)$ such that $x\mapsto a\in d$, where $a\in A$. At the same time $\tau_x(d) = T$ for such d. So, $(\tau_x \to \exists x.v\,\tau_x)(d) = F$.

4 Semantic Models and Languages of Logics over Hierarchical Data

Semantic models of *composition-nominative logics over hierarchical nominative data* (*CNLH*) are predicate algebras with class $PrH^{V\text{-}A}$ of hierary predicates as carriers and class $\boldsymbol{C}$ of compositions as operations of algebras. The class $\boldsymbol{C}$ is determined by a level intensional; for a quantifier level $\boldsymbol{C}$ consists of compositions $\vee$, $\neg$, $R^{\bar{v}}_{\bar{x}}$, and $\exists x$; for renominative level these are $\vee$, $\neg$, and $R^{\bar{v}}_{\bar{x}}$. Thus, algebras of the form $AHD(V, A) = \,<PrH^{V\text{-}A};\ \vee, \neg, R^{\bar{v}}_{\bar{x}}, \exists x>$ are semantic base of constructed logics. With a fixed sets V and $\boldsymbol{C}$ such algebras are determined by the set A.

Alphabet of a language of quantifier level includes symbols of basic compositions, a set Ps of *predicate symbols*, and a set of *basic subject names (variables) V.*

The set Fr of formulas for a quantifier level is defined inductively:

1) every predicate symbol from Ps is an (atomic) formula;

2) if Φ and Ψ are formulas, then $\Phi\vee\Psi$ and $\neg\Phi$ are formulas;

3) if Φ is a formula, then $R_{\bar{x}}^{\bar{v}}\Phi$ is a formula;

4) if Φ is a formula, then $\exists x\Phi$ is a formula.

For CNLH of renominative level we drop item 4 in this definition.

Let *nm(*Φ) be the set of all names, which appear in the symbols of renomination and quantification in Φ.

To distinguish symbols of compositions from their interpretations we use for the latter bold font (only in the following definitions). Let $\boldsymbol{I}: Ps\to PrH^{V\_A}$ be a total single-valued *interpretation mapping*, then a pair (*AHD*(*V*, *A*), ***I***) is called *a model of CNLH language*. To simplify notation we will denote models as (*A*, ***I***) Interpretation $\boldsymbol{J}: Fr\to PrH^{V\_A}$ we define as follows:

1) $\boldsymbol{J}(p) = \boldsymbol{I}(p)$ for each $p\in Ps$;

2) $\boldsymbol{J}(\Phi\vee\Psi) = \boldsymbol{J}(\Phi)\vee\boldsymbol{J}(\Psi)$, $\boldsymbol{J}(\neg\Phi) = \neg(\boldsymbol{J}(\Phi))$;

3) $\boldsymbol{J}(R_{\bar{x}}^{\bar{v}}\Phi) = \boldsymbol{R}_{\bar{x}}^{\bar{v}}(\boldsymbol{J}(\Phi))$;

4) $\boldsymbol{J}(\exists x\Phi) = \exists\boldsymbol{x}(\boldsymbol{J}(\Phi))$.

For renominative level we drop item 4.

Predicate $\boldsymbol{J}(\Phi)$, which is the value of a formula Φ interpreted on $\boldsymbol{A}=(A,\boldsymbol{I})$, we denote by $\Phi_{\boldsymbol{A}}$. A formula Φ is partially true on $\boldsymbol{A}=(A,\boldsymbol{I})$ (denoted by $\boldsymbol{A}\models\Phi$), if $\Phi_{\boldsymbol{A}}$ is partially true (irrefutable) predicate. Φ is everywhere (partially) true, or irrefutable (denoted by $\models\Phi$), if Φ is partially true on every model of a language.

A formula Ψ is a *logical consequence* of a formula Φ ($\Phi\models\Psi$), if formula $\Phi\to\Psi$ is irrefutable. Ψ is *a weak logical consequence* of Φ ($\Phi\,||\!\!=\Psi$), if for each $\boldsymbol{A}=(A,\boldsymbol{I})$ the condition $\boldsymbol{A}\models\Phi$ implies $\boldsymbol{A}\models\Psi$.

Formulas Φ and Ψ are *logically equivalent* ($\Phi\sim\Psi$), if $\Phi\models\Psi$ and $\Psi\models\Phi$. Formulas Φ and Ψ are *logically strictly equivalent* ($\Phi\sim_{TF}\Psi$), if $\boldsymbol{T}(\Phi_A)=\boldsymbol{T}(\Psi_A)$ and $\boldsymbol{F}(\Psi_A)=\boldsymbol{F}(\Phi_A)$ for each AS $\boldsymbol{A}$. The relation of logical consequence can be extended to arbitrary sets $\Gamma,\Delta\subseteq Fr$. Δ is a logical consequence of Γ in the model $\boldsymbol{A}$ ($\Gamma_A\models\Delta$) if for all $d\in HD(V,A)$ the condition $\Phi_A(d)\downarrow=T$ for all $\Phi\in\Gamma$ implies that it is impossible that $\Psi_A(d)\downarrow=F$ for all $\Psi\in\Delta$. Δ is a logical consequence of Γ ($\Gamma\models\Delta$), if $\Gamma_A\models\Delta$ for all model $\boldsymbol{A}=(A,\boldsymbol{I})$. Relation $\models$ is reflective but not transitive.

For CNLH the following statements hold.

Theorem 2 (semantic equivalence). Suppose that Φ' is obtained from Φ by substitution of some occurrences of $\Phi_1,\dots,\Phi_n$ with $\Psi_1,\dots,\Psi_n$ respectively. If $\Phi_1\sim\Psi_1$, ..., $\Phi_n\sim\Psi_n$, then $\Phi\sim\Phi'$.

Theorem 3 (semantic equivalence, strong form). Suppose that Φ' is obtained from Φ by substitution of some occurrences of $\Phi_1,\dots,\Phi_n$ with $\Psi_1,\dots,\Psi_n$ respectively. If $\Phi_1\sim_{TF}\Psi_1,\dots,\Phi_n\sim_{TF}\Psi_n$, then $\Phi\sim_{TF}\Phi'$.

Theorem 4 (substitution of equivalents). Suppose that $\Phi\sim\Psi$. Then $\Phi,\Gamma\models\Delta\Leftrightarrow\Psi,\Gamma\models\Delta$ and $\Gamma\models\Delta,\Phi\Leftrightarrow\Gamma\models\Delta,\Psi$.

A name $x\in V$ is *strictly unessential* for Φ ($x\in sun(\Phi)$), if x is strictly unessential for a predicate $\Phi_{\boldsymbol{A}}$ for every $\boldsymbol{A}=(A,\boldsymbol{I})$.

Proposition 4. Let $y\in sun(\Phi)$. Then $\exists x\Phi\sim_{TF}\exists yR_y^x\Phi$.

For each $p{\in}Ps$ the set of strictly unessential subject names is fixed by a total function $\nu: Ps{\to}2^V$. For CNLH we postulate infinity of the set $V_T = \bigcap_{p\in Ps} \nu(p)$ of *totally strictly unessential names*.

The following properties of formulas are representations of corresponding semantic properties of predicate algebras.

RsN) $R^{y,\bar{v}}_{z,\bar{x}}(\Phi) \sim_{TF} R^{\bar{v}}_{\bar{x}}(\Phi)$, if $y{\in}sun(\Phi)$.

RT) $R^{z,\bar{v}}_{z,\bar{x}}(\Phi) \sim_{TF} R^{\bar{v}}_{\bar{x}}(\Phi)$, if $z{\in}V$; in particular, $R^{z}_{z}(\Phi) \sim_{TF} \Phi$.

R∨) $R^{\bar{v}}_{\bar{x}}(\Phi \vee \Psi) \sim_{TF} R^{\bar{v}}_{\bar{x}}(\Phi) \vee R^{\bar{v}}_{\bar{x}}(\Psi)$.

R¬) $R^{\bar{v}}_{\bar{x}}(\neg\Phi) \sim_{TF} \neg R^{\bar{v}}_{\bar{x}}(\Phi)$.

Generalizing R∨ and R¬, we get RR∨ and RR¬.

RR∨) $R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi \vee \Psi)...) \sim_{TF} R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi)...) \vee R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Psi)...)$.

RR¬) $R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(...R^{\bar{w}}_{\bar{z}}(\neg\Phi)...)) \sim_{TF} \neg R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(...R^{\bar{w}}_{\bar{z}}(\Phi)...))$.

Similarly, we can write down the properties R&, R→, R↔, RR&, RR→, RR↔.

RR_C) $R^{\bar{v}}_{\bar{x}}(R^{\bar{u}}_{\bar{y}}(\Phi_A)(d) = \Phi_A(r^{\bar{u}}_{\bar{y}}(r^{\bar{v}}_{\bar{x}}(d)))$ for each $\boldsymbol{A} = (A, \boldsymbol{I})$, $d{\in}HD(V, A)$.

ANQ) $\exists x.y(R^{x,\bar{u}}_{z,\bar{v}}\Phi) \sim_{TF} R^{x,\bar{u}}_{z,\bar{v}}(\Phi)$ and $\forall x.y(R^{x,\bar{u}}_{z,\bar{v}}\Phi) \sim_{TF} R^{x,\bar{u}}_{z,\bar{v}}(\Phi)$, if $x.y \div \{z,\bar{u},\bar{v}\}$.

ANR) $R^{\bar{y},\bar{u}}_{\bar{z},\bar{v}}(\exists x\Phi) \sim_{TF} R^{\bar{u}}_{\bar{v}}(\exists x\Phi)$ and $R^{\bar{y},\bar{u}}_{\bar{z},\bar{v}}(\forall x\Phi) \sim_{TF} R^{\bar{u}}_{\bar{v}}(\forall x\Phi)$, if x is a prefix of all names in $\bar{y}$ and $x \div \{\bar{u}\}$.

R∃) $R^{\bar{v}}_{\bar{x}}(\exists y\Phi) \sim_{TF} \exists y(R^{\bar{u}}_{\bar{v}}\Phi)$, if $y \div \{\bar{u},\bar{v}\}$.

R∃∃) $R^{\bar{v}}_{\bar{x}}(\exists y\Phi) \sim_{TF} \exists z R^{\bar{v}}_{\bar{x}}(R^{y}_{z}(\Phi))$ if $z{\in}V_T$ and $z{\notin}nm(R^{\bar{v}}_{\bar{x}}(\exists y\Phi))$.

Similarly, we can formulate R∀ and R∀∀. Properties R∃, R∃∃, R∀, R∀∀ can be generalized to RR∃, RR∃∃, RR∀, RR∀∀; R∨ and R¬ to RR∨ and RR¬.

For equitone predicates RT can be changed to RTE:

RTE) $R^{u,\bar{v}}_{u,\bar{x}}(\Phi) \sim R^{\bar{v}}_{\bar{x}}(\Phi)$; in particular $R^{u}_{u}(\Phi) \sim \Phi$.

For logics of equitone predicates we introduce the notion of *primitive formula*. A formula $R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(...R^{\bar{w}}_{\bar{z}}(p)...))$ is primitive, if $p{\in}Ps$ and in renominations identical pairs of names are removed.

With every primitive $R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(...R^{\bar{w}}_{\bar{z}}(p)...))$ we connect an expression of the form $p(\alpha)$, where α represents a convolution of renominations $r^{\bar{u}}_{\bar{x}}(r^{\bar{v}}_{\bar{y}}(...r^{\bar{w}}_{\bar{z}}(\delta)...))$ given in the standard form, $\delta{\notin}V{\cup}Ps$ is a special symbol, which denotes arbitrary data. To take into account strictly unessential subject names, we delete all components that have $z{\in}\nu(p)$ as a prefix. An expression $p(\alpha)$ is called a *renominant* of the above primitive formula. The set of longest incomparable names occurred in a renominant is called its *naming scheme*.

Example 10. To construct the renominant of a primitive formula $R^{u.v}_{x}(R^{z}_{u}(q))$ we specify corresponding standard form of renomination convolution (see Example 7) obtaining renominant $q(d\,\|_{-u,z} + u.v \mapsto x : d + z \mapsto u : d\,\|_{-v} + z.v \mapsto x : d)$. Its naming scheme is $\{u.v, x, z.v\}$.

Now we point out basic properties of quantification compositions for CNLH.

Q1. $\exists x\exists y\Phi \sim_{TF} \exists y\exists x\Phi$ and $\forall x\forall y\Phi \sim_{TF} \forall y\forall x\Phi$, if x and y are incomparable.

Q2. $\neg\forall x\Phi \sim_{TF} \exists x\neg\Phi$ and $\neg\exists x\Phi \sim_{TF} \forall x\neg\Phi$.

Q3. $\exists x\Phi \sim_{TF} \forall x\exists x\Phi$, $\exists x\Phi \sim_{TF} \exists x\exists x\Phi$; $\forall x\Phi \sim_{TF} \forall x\forall x\Phi$, $\forall x\Phi \sim_{TF} \exists x\forall x\Phi$.

Q4. $\exists x\Phi \sim_{TF} \forall x.y\exists x\Phi$, $\exists x\Phi \sim_{TF} \exists x.y\exists x\Phi$; $\forall x\Phi \sim_{TF} \forall x.y\forall x\Phi$, $\forall x\Phi \sim_{TF} \exists x.y\forall x\Phi$.

Q5. $\exists x\Phi\vee\exists x\Psi \sim_{TF} \exists x(\Phi\vee\Psi)$ and $\forall x\Phi\&\forall x\Psi \sim_{TF} \forall x(\Phi\&\Psi)$.

Q6. $\exists x(\Phi\&\Psi) \models \exists x\Phi\&\exists x\Psi$ and $\forall x\Phi\vee\forall x\Psi \models \forall x(\Phi\vee\Psi)$.

Q7. $\exists y\forall x\Phi \models \forall x\exists y\Phi$; and not always $\forall x\exists y\Phi \models \exists y\forall x\Phi$.

Q8. $\Phi \,\|\!= \exists x\Phi$ and $\Phi \,\|\!= \forall x\Phi$.

Q9. $\models \forall x\,(\forall x\Phi\to\Phi)$ and $\models \exists x\,(\forall x\Phi\to\Phi)$; $\models \forall x\,(\Phi\to\exists x\Phi)$ and $\models \exists x\,(\Phi\to\exists x\Phi)$.

Properties Q2, Q3, Q5–Q9 are analogous to the corresponding properties of logics of quasiary predicates.

At the propositional level the properties of $\models$ for sets of formulas are identical to corresponding properties of logic of quasiary predicates [3].

Now we formulate basic properties of renomination compositions.

$\text{RTE}_{|-}$) $R^{u,\bar{v}}_{u,\bar{x}}(\Phi), \Gamma\ {}_A\!\models \Delta \Leftrightarrow R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma\ {}_A\!\models \Delta$.

$\text{RTE}_{-|}$) $\Gamma\ {}_A\!\models \Delta, R^{u,\bar{v}}_{u,\bar{x}}(\Phi) \Leftrightarrow \Gamma\ {}_A\!\models \Delta, R^{\bar{v}}_{\bar{x}}(\Phi)$.

$\text{RsN}_{|-}$) $R^{y,\bar{v}}_{z,\bar{x}}(\Phi), \Gamma\ {}_A\!\models \Delta \Leftrightarrow R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma\ {}_A\!\models \Delta$, where $y\in V$ is strictly unessential for Φ.

$\text{RsN}_{-|}$) $\Gamma\ {}_A\!\models \Delta, R^{y,\bar{v}}_{z,\bar{x}}(\Phi) \Leftrightarrow \Gamma\ {}_A\!\models \Delta, R^{\bar{v}}_{\bar{x}}(\Phi)$, where $y\in V$ is strictly unessential for Φ.

$\text{RR}\vee_{|-}$) $R^{\bar{u}}_{\bar{x}}(\ldots R^{\bar{w}}_{\bar{z}}(\Phi\vee\Psi)\ldots), \Gamma\ {}_A\!\models \Delta \Leftrightarrow R^{\bar{u}}_{\bar{x}}(\ldots R^{\bar{w}}_{\bar{z}}(\Phi)\ldots) \vee R^{\bar{u}}_{\bar{x}}(\ldots R^{\bar{w}}_{\bar{z}}(\Psi)\ldots), \Gamma\ {}_A\!\models \Delta$.

$\text{RR}\vee_{-|}$) $\Gamma, R^{\bar{u}}_{\bar{x}}(\ldots R^{\bar{w}}_{\bar{z}}(\Phi\vee\Psi)\ldots)\ {}_A\!\models \Delta \Leftrightarrow \Gamma, R^{\bar{u}}_{\bar{x}}(\ldots R^{\bar{w}}_{\bar{z}}(\Phi)\ldots) \vee R^{\bar{u}}_{\bar{x}}(\ldots R^{\bar{w}}_{\bar{z}}(\Psi)\ldots)\ {}_A\!\models \Delta$.

$\text{RR}\neg_{|-}$) $R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(\ldots R^{\bar{w}}_{\bar{z}}(\neg\Phi)\ldots)), \Gamma\ {}_A\!\models \Delta \Leftrightarrow \neg R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(\ldots R^{\bar{w}}_{\bar{z}}(\Phi)\ldots)), \Gamma\ {}_A\!\models \Delta$.

$\text{RR}\neg_{-|}$) $\Gamma, R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(\ldots R^{\bar{w}}_{\bar{z}}(\neg\Phi)\ldots))\ {}_A\!\models \Delta \Leftrightarrow \Gamma, \neg R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(\ldots R^{\bar{w}}_{\bar{z}}(\Phi)\ldots))\ {}_A\!\models \Delta$.

Properties $\text{RR}\to_{|-}$, $\text{RR}\to_{-|}$, $\text{RR}\&_{|-}$, $\text{RR}\&_{-|}$ are analogous.

$\text{R}\exists_{|-}$) $R^{\bar{v}}_{\bar{x}}(\exists y\Phi), \Gamma\ {}_A\!\models \Delta \Leftrightarrow \exists y R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma\ {}_A\!\models \Delta$ if $y \div \{\bar{u}, \bar{v}\}$.

$\text{R}\exists_{-|}$) $\Gamma\ {}_A\!\models \Delta, R^{\bar{v}}_{\bar{x}}(\exists y\Phi) \Leftrightarrow \Gamma\ {}_A\!\models \Delta, \exists y R^{\bar{v}}_{\bar{x}}(\Phi)$ if $y \div \{\bar{u}, \bar{v}\}$.

$\text{R}\exists\exists_{|-}$) $R^{\bar{v}}_{\bar{x}}(\exists y\Phi), \Gamma\ {}_A\!\models \Delta \Leftrightarrow \exists z R^{\bar{v}}_{\bar{x}}(R^{y}_{z}(\Phi)), \Gamma\ {}_A\!\models \Delta$.

$\text{R}\exists\exists_{-|}$) $\Gamma\ {}_A\!\models \Delta, R^{\bar{v}}_{\bar{x}}(\exists y\Phi) \Leftrightarrow \Gamma\ {}_A\!\models \Delta, \exists z R^{\bar{v}}_{\bar{x}}(R^{y}_{z}(\Phi))$.

For $\text{R}\exists\exists_{|-}$ and $\text{R}\exists\exists_{-|}$ z is totally strictly unessential and $z\notin nm(R^{\bar{v}}_{\bar{x}}(\exists y\Phi))$.

Properties $\text{R}\forall_{|-}$, $\text{R}\forall_{-|}$, $\text{R}\forall\forall_{|-}$, $\text{R}\forall\forall_{-|}$ are analogous. Properties of type R∃, R∃∃, R∀, R∀∀ can be generalized to properties of type RR∃, RR∃∃, RR∀, RR∀∀.

5 The Sequent Calculus of Logics of Predicates over Hierarchical Data

For logics of equitone hierary predicates we will build a calculus of sequent type. We will consider here only logics of renominative level. Sequents are interpreted as sets of *labeled formulas* marked by one of two symbols – ${}_{\vdash}$ or ${}_{\dashv}$. Such sequents Σ are also denoted by ${}_{\vdash}\Gamma{}_{\dashv}\Delta$, where all formulas of Γ are labeled by the symbol ${}_{\vdash}$, of Δ – by the symbol ${}_{\dashv}$.

Sequent Σ is *closed*, if there exists Φ such that ${}_{\vdash}\Phi\in\Sigma$ and ${}_{\dashv}\Phi\in\Sigma$ or if there exist primitive φ and ψ with identical renominants such that ${}_{\vdash}\varphi\in\Sigma$ and ${}_{\dashv}\psi\in\Sigma$. Consequently, if ${}_{\vdash}\Gamma{}_{\dashv}\Delta$ is closed then $\Gamma \models \Delta$.

Derivation in the sequent calculus has the form of tree, the vertices of which are sequents. Such trees [3] are called sequent trees. A sequent tree is *closed*, if every its leaf is a closed sequent. A sequent Σ is *derivable*, if there is a closed sequent tree with root Σ. Sequent calculus is constructed in such a way that sequent ${}_{\vdash}\Gamma{}_{\dashv}\Delta$ has a derivation if and only if $\Gamma \models \Delta$.

Semantic properties of relation $\models$ have their syntactic analogues – sequent forms (rules). For renominative logics of equitone hierary predicates these forms are the following.

$${}_{\vdash}\vee\ \frac{{}_{\vdash}A,\ \Sigma \qquad {}_{\vdash}B,\ \Sigma}{{}_{\vdash}A\vee B,\ \Sigma} \qquad\qquad {}_{\dashv}\vee\ \frac{{}_{\dashv}A,\ {}_{\dashv}B,\ \Sigma}{{}_{\dashv}A\vee B,\ \Sigma}$$

$${}_{\vdash}\neg\ \frac{{}_{\dashv}A,\ \Sigma}{{}_{\vdash}\neg A,\ \Sigma} \qquad\qquad {}_{\dashv}\neg\ \frac{{}_{\vdash}A,\ \Sigma}{{}_{\dashv}\neg A,\ \Sigma}$$

$${}_{\vdash}\mathbf{RTE}\ \frac{{}_{\vdash}R_{\bar{x}}^{\bar{v}}(A),\Sigma}{{}_{\vdash}R_{u,\bar{x}}^{u,\bar{v}}(A),\Sigma} \qquad\qquad {}_{\dashv}\mathbf{RTE}\ \frac{{}_{\dashv}R_{\bar{x}}^{\bar{v}}(A),\Sigma}{{}_{\dashv}R_{u,\bar{x}}^{u,\bar{v}}(A),\Sigma}$$

${}_{\vdash}\mathbf{RR}\vee$

$$\frac{{}_{\vdash}R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(A)\ldots)\vee R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(B)\ldots),\Sigma}{{}_{\vdash}R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(A\vee B)\ldots)),\Sigma}$$

${}_{\dashv}\mathbf{RR}\vee$

$$\frac{{}_{\dashv}R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(A)\ldots)\vee R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(B)\ldots),\Sigma}{{}_{\dashv}R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(A\vee B)\ldots),\Sigma}$$

$${}_{\vdash}\mathbf{RR}\neg\ \frac{{}_{\vdash}\neg R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(A)\ldots),\Sigma}{{}_{\vdash}R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(\neg A)\ldots),\Sigma} \qquad\qquad {}_{\dashv}\mathbf{RR}\neg\ \frac{{}_{\dashv}\neg R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(A)\ldots),\Sigma}{{}_{\dashv}R_{\bar{x}}^{\bar{u}}(\ldots R_{\bar{z}}^{\bar{w}}(\neg A)\ldots),\Sigma}$$

Sequent calculus with basic sequent forms shown above we will call *RID-calculus*. For *RID-calculus* theorems of soundness and completeness hold.

Theorem 5 (soundness). Let sequent ${}_{\vdash}\Gamma{}_{\dashv}\Delta$ be derivable. Then $\Gamma \models \Delta$.

The proof can be conducted by induction over shape of a sequent tree for ${}_{\vdash}\Gamma{}_{\dashv}\Delta$.

For proving completeness we will use ***Hintikka's method of model sets***. The set $\boldsymbol{H}$ of labeled formulas with $W = nm(\boldsymbol{H})$ is a model set, if:

HCΦ) For every non-primitive formula Φ it is impossible that ${}_{\vdash}\Phi\in\boldsymbol{H}$ and ${}_{\dashv}\Phi\in\boldsymbol{H}$.

HCR) For primitive formulas φ and ψ with identical renominants it is impossible that ${}_{\vdash}\varphi, {}_{\dashv}\psi\in\boldsymbol{H}$ and it is impossible that ${}_{\vdash}\psi, {}_{\dashv}\varphi\in\boldsymbol{H}$.

H∨) If $\vdash\Phi\vee\Psi\in\boldsymbol{H}$, then $\vdash\Phi\in\boldsymbol{H}$ or $\vdash\Psi\in\boldsymbol{H}$; if $\dashv\Phi\vee\Psi\in\boldsymbol{H}$, then $\dashv\Phi\in\boldsymbol{H}$ and $\dashv\Psi\in\boldsymbol{H}$.
H¬) If $\vdash\neg\Phi\in\boldsymbol{H}$, then $\dashv\Phi\in\boldsymbol{H}$; if $\dashv\neg\Phi\in\boldsymbol{H}$, then $\vdash\Phi\in\boldsymbol{H}$.
HRT) If $\vdash R^{u,\bar{v}}_{u,\bar{x}}(\Phi)\in\boldsymbol{H}$, then $\vdash R^{\bar{v}}_{\bar{x}}(\Phi)\in\boldsymbol{H}$; if $\dashv R^{u,\bar{v}}_{u,\bar{x}}(\Phi)\in\boldsymbol{H}$, then $\dashv R^{\bar{v}}_{\bar{x}}(\Phi)\in\boldsymbol{H}$.
HR∨) If $\vdash R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi\vee\Psi)...)\in\boldsymbol{H}$, then $\vdash R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi)...)\vee R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Psi)...)\in\boldsymbol{H}$;
if $\dashv R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi\vee\Psi)...)\in\boldsymbol{H}$, then
$\dashv R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi)...)\vee R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Psi)...)\in\boldsymbol{H}$.
HR¬) If $\vdash R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\neg\Phi)...)\in\boldsymbol{H}$, then $\vdash\neg R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi)...)\in\boldsymbol{H}$;
if $\dashv R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\neg\Phi)...)\in\boldsymbol{H}$, then $\dashv\neg R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(\Phi)...)\in\boldsymbol{H}$.

Procedure of construction of a tree for Σ is split into stages. Every application of sequent form is performed only for the finite set of accessible formulas. At the beginning of every stage we perform the step of access: to the list of accessible formulas one formula from each of lists of $\vdash$-formulas and $\dashv$-formulas is added. We start the construction with a pair of first formulas from the lists.

Suppose that k stages of procedure have already been performed. On the stage k+1 we check whether all terminal nodes are closed. If yes, the procedure is completed positively, and we have got a closed sequent tree. If no, for every unclosed leaf ξ we undertake a next step of access, whereupon we finish building of finite subtree with a vertex ξ as follows.

We activate all accessible non-primitive formula ξ. Then to every active formula we apply the proper sequent form. We remove all repetitions of formulas in a sequent.

During the construction of sequent tree the following cases are possible:

1. Procedure is completed positively; we have the finite closed tree.
2. Procedure is completed negatively, or is not completed; we have a finite
or infinite unclosed tree. Such tree has at least one path all vertices of which are unclosed sequents. Such path $\wp$ is unclosed. Every formula of Σ will be in $\wp$ and will become accessible.

Theorem 6. Let $\wp$ be an unclosed path in sequent tree. Then there exists AS $\boldsymbol{A}=(A,\boldsymbol{I})$ and $\delta\in HD(V,A)$: $\vdash\Phi\in\boldsymbol{H}\Rightarrow\Phi_{\boldsymbol{A}}(\delta)\downarrow=T$ and $\dashv\Phi\in\boldsymbol{H}\Rightarrow\Phi_{\boldsymbol{A}}(\delta)\downarrow=F$.

The set $\boldsymbol{H}$ of labeled formulas of sequents of the path $\wp$ is a model set.

Let W be a combination of naming schemes of the set of renominants of primitive formulas of sequents of the path $\wp$. Such W includes longest incomparable names, which are involved in renominations of formulas of sequents of the path $\wp$.

We duplicate elements of W obtaining $A=\{\boldsymbol{u}\mid u\in W\}$; then put $\delta=[u\mapsto\boldsymbol{u}\mid u\in W]$.

We specify the values of basic predicates on δ and on data of the form $r^{\bar{u}}_{\bar{x}}(...r^{\bar{w}}_{\bar{z}}(\delta)...)$ in the following way:

- if $\vdash p\in\boldsymbol{H}$, then set $p_A(\delta)=T$; if $\dashv p\in\boldsymbol{H}$, then set $p_A(\delta)=F$;
- if $\vdash R^{\bar{u}}_{\bar{x}}(...R^{\bar{w}}_{\bar{z}}(p)...)\in\boldsymbol{H}$, then set $p_A(r^{\bar{u}}_{\bar{x}}(...r^{\bar{w}}_{\bar{z}}(\delta)...))=T$;
- if $\dashv R^{\bar{u}}_{\bar{x}}(R^{\bar{v}}_{\bar{y}}(...R^{\bar{w}}_{\bar{z}}(p)...)\in\boldsymbol{H}$, then set $p_A(r^{\bar{u}}_{\bar{x}}(...r^{\bar{w}}_{\bar{z}}(\delta)...))=F$.

In all other cases for $d\in HD(V,A)$ the value of $p_A(d)$ can be set arbitrarily, taking into account equitonicity and strict inessentiality of names.

Theorem holds for atomic and primitive formulas due to above definitions of basic predicates. Then the proof is carried out by induction over the complexity of a formula in accordance with construction of a model set.

Theorem 7 (completeness). Let $\Gamma \models \Delta$. Then a sequent ${}_{\vdash}\Gamma{}_{\dashv}\Delta$ is derivable.

Suppose contrary: $\Gamma \models \Delta$ and a sequent ${}_{\vdash}\Gamma{}_{\dashv}\Delta$ is not derivable. Then sequent tree δ for $\Sigma = {}_{\vdash}\Gamma{}_{\dashv}\Delta$ is not closed. Consequently, in δ there is unclosed path $\wp$. The set $\boldsymbol{H}$ of all labeled formulas of sequents of this path is a model set. According to the theorem 5 there exists AS $\boldsymbol{A} = (A, \boldsymbol{I})$ and $\delta \in HD(V, A)$ such that ${}_{\vdash}\Phi \in \boldsymbol{H} \Rightarrow \Phi_A(\delta)\downarrow = T$ and ${}_{\dashv}\Phi \in \boldsymbol{H} \Rightarrow \Phi_A(\delta)\downarrow = F$. Due to $\Sigma \subseteq \boldsymbol{H}$ we have ${}_{\vdash}\Phi \in \Sigma \Rightarrow \Phi_A(\delta)\downarrow = T$ and ${}_{\dashv}\Phi \in \Sigma \Rightarrow \Phi_A(\delta)\downarrow = F$. But it contradicts $\Gamma \models \Delta$.

6 Conclusions

In the paper new logics oriented on hierarchical data are developed. Algebras of partial predicates over such data with special compositions as operations form a semantic base for constructed logics. These logics may also be treated as generalization of classical logic. First of all, this generalization concerns types of predicates: while classical logic is semantically based on total n-ary predicates, we have constructed logics based on partial quasiary and hierary predicates, defined on special types of hierarchical nominative data. Importance of such data is explained by their representational power, which permits to model data structures of specification and programming languages. Characteristic feature of such languages is usage of composite names to access data components. The constructed logics also use composite names. Semantic properties of such logics have been studied; corresponding sequent calculi have been defined, their soundness and completeness have been proved for logics of renominative level. Authors plan to present more developed logics at hierarchical nominative level in forthcoming papers.

References

1. Nikitchenko, N.S.: A Composition-nominative Approach to Program Semantics. Technical Report IT–TR 1998-020, Technical University of Denmark, 103 p. (1998)
2. Nikitchenko, M.S.: Composition-nominative aspects of address programming. Cybernetics and Systems Analysis, No. 6, pp. 24-35 (2009) (In Russian). English translation: Springer New York, Volume 45, Number 6 / November, 2009.
3. Nikitchenko, M.S., Shkilniak, S.S.: Mathematical logic and theory of algorithms. Publishing house of National Taras Shevchenko University of Kyiv, 528 p. (2008) (in Ukrainian).
4. Basarab, I.A., Gubsky, B.V., Nikitchenko, N.S., Red'ko, V.N.: Composition models of databases. In: Eder, J., Kalinichenko, L.A. (eds.) East-West Database Workshop.– (Workshops in Computing Series). Springer, London, pp. 221-231 (1995)
5. Nikitchenko, N.S.: Abstract Computability of Non-deterministic Programs over Various Data Structures. In: Bjørner, D., Broy, M., Zamulin A.V. (eds.) Perspectives of System Informatics. LNCS, vol. 2244, pp. 471-484. Springer, Berlin (2001)
6. Shkilniak, S.S.: Relations of logical consequence in composition-nominative logics. Problems of Programming. Kyiv, No. 1, pp. 15–38, 2010 (In Ukrainian)

7. Nikitchenko, M.S., Shkilnyak, S.S., Omelchuk, L.L.: Formalisms for Specification of Programs over Nominative Data. In: Electronic computers and informatics (ECI 2006). Thesis of conference reports, pp. 134-139. Kosice, Herl'any, Slovakia (2006)
8. Nielson, H.R., Nielson, F.: Semantics with Applications: A Formal Introduction. John Wiley & Sons Inc. 252 p. (1992)
9. Kleene, S. C.: Introduction to metamathematics, Van Nostrand, New York (1952)
10. Woodcock, J.C.P., Davies, J.: Using Z: Specification, Refinement and Proof. Prentice Hall, 523 p. (1996)
11. Abrial, J.R.: The B-Book: Assigning programs to meanings. Cambridge University Press, 779 p. (1996)
12. Lamport, L.: Specifying Systems: The TLA$^+$ Language and Tools for Hardware and Software Engineers. Addison-Wesley (2002)
13. Lamport, L.: Substitution: Syntactic versus Semantic SRC Technical Note 1998-004 (March 1998)
14. Nikitchenko, M.S., Shkilnyak, S.S., Composition-nominative logics over hierarchical data. Problems of Programming. Kyiv, No. 2-3, pp. 48–57, 2010 (In Ukrainian)
15. Pitts, A.M.: Nominal logic, a first order theory of names and binding. Inf. Comput. 186(2), pp. 165-193 (2003)