Satisfiability Problem in Composition-Nominative Logics of Quantifier-Equational Level

Mykola S. Nikitchenko[1] and Valentyn G. Tymofieiev[1]

[1] Department of Theory and Technology of Programming
Taras Shevchenko National University of Kyiv
64, Volodymyrska Street, 01601 Kyiv, Ukraine
nikitchenko@unicyb.kiev.ua
tvalentyn@univ.kiev.ua

Abstract. We investigate algorithms for solving the satisfiability problem in composition-nominative logics of quantifier-equational level. These logics are algebra-based logics of partial predicates constructed in a semantic-syntactic style on the methodological basis, which is common with programming; they can be considered as generalizations of traditional logics on classes of partial predicates that do not have fixed arity. We show the reduction of the problem in hand to the satisfiability problem for classical first-order predicate logic with equality. The proposed reduction requires extension of logic language and logic models with an infinite number of unessential variables. The method developed in the paper enables us to use existent satisfiability checking procedures also for quantifier composition-nominative logic with equality.

Keywords: Composition-nominative logics, partial predicates, partial logics, first-order logics, satisfiability, validity.

Key Terms. Research, MathematicalModel, FormalMethods, MachineIntelligence.

1 Introduction

Last years the interest to the satisfiability problem [1] has risen due to practical value it has obtained in such areas as program verification, synthesis, analysis, testing, etc. [2–5]. In this paper we address the satisfiability problem in the context of the *composition-nominative approach* [6], which aims to construct a hierarchy of logics of various abstraction and generality levels on the methodological basis, which is common with programming. The main principles of the approach are principles of *development from abstract to concrete*, *priority of semantics*, *compositionality*, and *nominativity*.

These principles specify a hierarchy of new logics that are semantically based on algebras of predicates. Predicates are considered as partial mappings from a certain class of data D into the class of Boolean values *Bool*. Operations over predicates are called *compositions*. They are treated as predicate construction tools. Data classes are considered on various abstraction levels, but the main attention is paid to the class of *nominative data*. Such data consist of pairs *name–value*. Nominative data can represent various data structures such as records, arrays, lists, relations, etc. [6, 7]; this fact explains the importance of the notion of nominative data. In the simplest case nominative data can be considered as partial mappings from a certain set of names (variables) V into a set of basic (atomic) values A. These data are called *nominative sets*; their class is denoted ^{V}A. Nominative sets represent program states for simple programming languages (see, for example, [6, 8]). Partial predicates and functions over ^{V}A are called *quasiary*, their classes are denoted $Pr^{V,A}= {}^{V}A \xrightarrow{p} Bool$ and $Fn^{V,A}= {}^{V}A \xrightarrow{p} A$ respectively. Partial mappings of type $^{V}A \xrightarrow{p} {}^{V}A$ are called *bi-quasiary*. Such mappings represent program semantics for simple programming languages; therefore their class is denoted $Prg^{V,A}$. From this follows that semantic models of programs and logics are mathematically based on the notion of nominative set (nominative data in general case). This fact permits to integrate models of programs and logics and represent them as hierarchy of composition-nominative models [9, 10]. Logics developed within such approach are called *composition-nominative logics* (CNL) because their predicates and functions are defined on classes of nominative data, and logical connectives and quantifiers are formalized as predicate compositions.

CNL can be considered as generalization of classical predicate logic but for all that many methods developed within classical logic can also be applied to CNL. Here we confirm this statement for the satisfiability problem in CNL. In this paper we consider composition-nominative logic of quantifier-equational level and construct an algorithm that reduces the satisfiability problem in this logic to the same problem in classical first-order predicate logic with equality. The reduction proposed requires the logic language to be extended with an infinite number of unessential variables.

The paper is structured in the following way. In section 2 we give an overview of the composition-nominative logics classification; then in section 3 we give formal definitions of the logics that we consider in this paper, and define the satisfiability problem. In section 4 we describe the reduction method for solving the satisfiability problem. In section 5 we discuss related work. In section 6 we summarize our results and formulate directions for future investigations.

Proofs are omitted here and will be provided in an extended version of the paper.

Notions and notations not defined in the paper are understood in the sense of [10].

2 Classification of Composition-Nominative Logics

Classification of *composition-nominative logics* is based on classification of their parameters: data, predicates, and compositions. The main semantic notion of mathematical logic – the notion of predicate – can be defined as a partial function from a data class D to $Bool$. For the most abstract level of data consideration such compositions as disjunction $\vee$, negation $\neg$, etc., can be defined. These compositions are derived from Kleene's strong connectives [11] when partiality of predicates is taken into consideration. Thus, the main semantic objects for logics of this level are algebras of partial predicates of the type $<D \xrightarrow{p} Bool; \vee, \neg>$. 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. At this level we have two sublevels determined respectively by flat and hierarchic nominative data.

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

1. pure quasiary predicate logics based on algebras with one sort: $Pr^{V,A}$;
2. quasiary predicate-function logics based on algebras with two sorts: $Pr^{V,A}$ and $Fn^{V,A}$;
3. quasiary program logics based on algebras with three sorts: $Pr^{V,A}$, $Fn^{V,A}$, and $Prg^{V,A}$.

For logics of pure quasiary predicates we identify renominative, quantifier, and quantifier-equational levels.

Renominative logics [10] are most abstract among the above-mentioned logics. The main composition for these logics is the composition of renomination (renaming), which is a total mapping $R^{v_1,...,v_n}_{x_1,...,x_n}: Pr^{V,A} \xrightarrow{t} Pr^{V,A}$. Intuitively, given a quasiary predicate P and a nominative set d, the value of $R^{v_1,...,v_n}_{x_1,...,x_n}(P)(d)$ is evaluated in the following way: first, a new nominative set d' is constructed from d by changing the values of the names $v_1,...,v_n$ in d to the values of the names $x_1,..., x_n$ respectively; then predicate P is applied to d'. The obtained value of P (if it was evaluated) will be the result of $R^{v_1,...,v_n}_{x_1,...,x_n}(P)(d)$. For simplicity's sake we will also use the simplified notation $R^{\bar{v}}_{\bar{x}}$ for renomination composition. The basic composition operations of renominative logics are $\vee$, $\neg$, and $R^{\bar{v}}_{\bar{x}}$.

At the *quantifier* level, all basic (object) values can be used to construct different nominative sets to which quasiary predicates can be applied. This allows one to introduce the compositions of quantification $\exists x$ in style of Kleene's strong quantifiers. The basic compositions of logics of the quantifier level are $\vee$, $\neg$, $R^{\bar{v}}_{\bar{x}}$, and $\exists x$.

At the *quantifier-equational* level, new possibilities arise for equating and differentiating values using special 0-ary compositions, i.e., parametric equality predicates $=_{xy}$. Basic compositions of logics of the quantifier-equational level are $\vee$, $\neg$, $R^{\bar{v}}_{\bar{x}}$, $\exists x$, and $=_{xy}$.

All specified logics (renominative, quantifier, and quantifier-equational) are based on algebras which have only one sort: a class of quasiary predicates.

For quasiary predicate-function logics we identify function level and function-equational levels.

At the *function* level, we have extended capabilities of formation of new arguments for functions and predicates. In this case it is possible to introduce the superposition composition $S^{\bar{x}}$ (see [6, 10]), which formalizes substitution of functions into predicate. It also seems natural to introduce special 0-ary compositions, called denaming functions $'x$. Given a nominative set, $'x$ yields a value of the name x in this set. Introduction of such functions allows one to model renomination compositions with the help of superposition. The basic compositions of logics of the function level are $\vee$, $\neg$, $S^{\bar{x}}$, $\exists x$, and $'x$.

At the function-equational level a special equality composition = can be introduced additionally [10]. The basic compositions of logics of the function-equational level are $\vee$, $\neg$, $S^{\bar{x}}$, $\exists x$, $'x$, and $=$. At this level different classes of first-order logics can be presented.

This means that two-sorted algebras (with sets of predicates and functions as sorts and above-mentioned compositions as operations) form a semantic base for first-order CNL.

The level of *program logics* is quite rich. First, program compositions should be defined that describe the structure of programs. In the simplest case these are:

1. assignment composition AS^x: $Fn^{V,A} \xrightarrow{t} Prg^{V,A}$,
2. composition of sequential execution $\bullet$: $Prg^{V,A} \times Prg^{V,A} \xrightarrow{t} Prg^{V,A}$,
3. conditional composition IF: $Pr^{V,A} \times Prg^{V,A} \times Prg^{V,A} \xrightarrow{t} Prg^{V,A}$,
4. cycling composition WH: $Pr^{V,A} \times Prg^{V,A} \xrightarrow{t} Prg^{V,A}$.

Then we should define compositions specifying program properties. Here we only mention a composition which formalizes the notion of assertion in *Floyd-Hoare logic*. From a semantic point of view an assertion scheme of the form $\{P\}prog\{Q\}$ may be considered as composition FH, which given two quasiary predicates P (precondition), Q (postcondition), and a bi-quasiary function (a program) *prog* produces new quasiary predicate denoted by $FH(P, prog, Q)$. At this level we obtained a three-sorted predicate-function-program algebra. Classes of terms of this algebra may be considered as sets of formulas (or their components) of corresponding logics.

Having described classification of composition-nominative logics we can formulate a task of investigation of logics presented in this classification. For many of such logics axiomatic calculi were constructed and their properties were investigated [10, 12].

In this paper we will consider the satisfiability problem for logics of quantifier-equational level. This problem for logics of the previous levels (propositional, renominative, and quantifier) was considered in [13]. We choose a reduction method that reduces the satisfiability problem of composition-nominative logic to the satisfiability in classical logic. To simplify this reduction we will use an intermediate logic with unessential variables. Thus, we will define three logics of quantifier-equational level: composition-nominative logic, logic with unessential variables, and classical first-order logic.

3 Formal Definitions of Logics of Quantifier-Equational Level

At first, we describe a general mechanism of specifying composition-nominative logics and then provide definitions for the logics considered in this paper. To do this we should specify three logic components that reflect the semantic-syntactic scheme of logic definition:

− *semantic component*: a class of algebras of quasiary predicates that forms a semantic base for a logic. In our case we consider algebras of the form

$AQE(V, A)$=<$Pr^{V,A}$, $\vee, \neg$, $R_{\bar{x}}^{\bar{v}}$, $\exists x$, $=_{xy}$> for various sets of atomic values A (recall that $Pr^{V,A} = {}^{V}A \xrightarrow{p} Bool$ is a class of partial predicates over ${}^{V}A$);

- *syntactic component*: a logic language specified by a class of logic formulas. This class is determined by the logic signature Σ, which includes the infinite set of names V, a set Ps of predicate symbols and a set Cs of composition symbols; the set of formulas $Fr(\Sigma)$ is constructed inductively over the set of atomic formulas $AFr(\Sigma)$ with the help of symbols of compositions;
- *interpretational (denotational) component*: a parametric total mapping that prescribes to a formula its meaning as a predicate. Parameters are algebra $AQE(V, A)$ and interpretation for atomic formulas $I: AFr(V,Ps) \xrightarrow{t} Pr^{V,A}$ called σ-interpretation. A pair $(AQE(V, A), I)$ is called a model of the logic. Given a model $M = (AQE(V, A), I)$ an interpretational mapping for each formula Φ specifies its meaning as a quasiary predicate in $AQE(V, A)$ denoted Φ_M. Usually models are represented in simplified form, say $J=(V, A, I)$, called π-interpretations; then the meaning of the formula is denoted Φ_J.

A logic defined according to this scheme is denoted $L(\Sigma)$.

3.1 Algebras of Quasiary Predicates of Quantifier-Equational Level

Semantic base of composition-nominative logics is specified by classes of data, predicates, and compositions. The latter are determined by the abstraction level of logic under consideration and are the same for all logics of the level. As was formulated earlier, for the logics of quantifier-equational level (QE-level) the class of compositions consists of basic propositional connectives, renomination composition, quantifiers, and equality predicate. The compositions (except propositional connectives) are parametric with parameters from an infinite set of names V.

Therefore we consider the following set of composition symbols:

$Cs_{QE}(V) = \{\vee, \neg\} \cup \{ R_{\bar{x}}^{\bar{v}} \mid \bar{v} = (v_1,...,v_n),\ \bar{x} = (x_1,...,x_n),\ \bar{v}$ is a list of distinct names, $v_i, x_i \in V$ for all $i \in \{1,...,n\}$, $n \geq 0\} \cup \{\exists x \mid x \in V\} \cup \{=_{xy} \mid x,y \in V\}$.

For the sake of simplicity we will write $Cs_{QE}(V) = \{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy}\}$.

Given an algebra $AQE(V, A)$=<$Pr^{V,A}$, $\vee, \neg$, $R_{\bar{x}}^{\bar{v}}$, $\exists x$, $=_{xy}$> we now define interpretation of composition symbols. Again, for simplicity's sake we will use the same notations for compositions (as operations in the algebra) and their symbols.

In definitions of compositions we will use the following notation:

- $p(d)\downarrow$ means that a predicate p is defined on data d ;
- $p(d)\downarrow = b$ means that a predicate p is defined on data d with a Boolean value b;
- $p(d)\uparrow$ means that a predicate p on d is undefined;
- for nominative data representation we use the form $d = [v_i \mapsto a_i \mid i \in I]$. *Nominative membership relation* is denoted by $\in_n$. Thus, $v_i \mapsto a_i \in_n d$ means that the value of v_i in d is defined and is equal to a_i; this can be written in another form as $d(v_i)\downarrow = a$.

Propositional compositions are defined by the following formulas (p, $q\in Pr^{V,A}$, $d\in {}^{V}A$):

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

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

Unary renomination composition $R_{\bar{x}}^{\bar{v}}$ is a mapping $R_{\bar{x}}^{\bar{v}}: Pr^{V,A}\xrightarrow{t} Pr^{V,A}$, where $\bar{v}=(v_1,...,v_n)$ and $\bar{x}=(x_1,...,x_n)$ are lists of names from a set V; names from $\bar{v}$ are called upper names of renomination composition and should be distinct, $n\geq 0$. Please note that $R_{\bar{x}}^{\bar{v}}$ is a parametric composition which represents a class of renomination compositions with different parameters, which are elements of V. This composition is defined by the following formula ($p\in Pr^{V,A}$, $d\in {}^{V}A$):

$$(R_{x_1,...,x_n}^{v_1,...,v_n}\ p)\,(d)=p([v\mapsto a\in_n d\mid v\notin\{v_1,...,v_n\}]\,\nabla\,[v_i\mapsto d(x_i)\mid d(x_i)\downarrow, i\in\{1,...,n\}]).$$

The ∇ operation is defined as follows: if d_1 and d_2 are two nominative sets, then $d=d_1\nabla d_2$ consists of all named pairs of d_2 and only those pairs of d_1, whose names are not defined (do not have values) in d_2.

Unary parametric composition of existential quantification $\exists x$ with the parameter $x\in V$ is defined by the following formula ($p\in Pr^{V,A}$, $d\in {}^{V}A$):

$$(\exists x\ p)(d)=\begin{cases} T, \quad \text{if } b\in A \text{ exists}: \ p(d\nabla x\mapsto b)\downarrow=T,\\ F, \quad p(d\nabla x\mapsto a)\downarrow=F \text{ for each } a\in A,\\ \qquad\text{undefined in other cases.}\end{cases}$$

Here $d\nabla x\mapsto a$ is a shorter form for $d\nabla[x\mapsto a]$.

Finally, *null-ary parametric equality composition* $=_{xy}$ (x, $y\in V$) is defined as follows:

$$=_{xy}(d)=\begin{cases} T, \text{ if } d(x)\downarrow,\ d(y)\downarrow \text{ and } d(x)=d(y),\\ T, \text{ if } d(x)\uparrow \text{ and } d(y)\uparrow,\\ F \quad \text{otherwise}\end{cases}$$

Now we will give definitions for all logics with a fixed infinite set of names V and a fixed set of predicate symbols Ps. Note that according to the tradition elements of V are also called *variables*. As semantic components for all logics are the same, we need to define only syntactic and interpretational components.

3.2 Composition-Nominative Logic $L_{QE}(\Sigma_{QE})$ of Quantifier-Equational Level

1. *Syntactic component.* A tuple Σ_{QE}= (V, {$\vee$, $\neg$, $R_{\bar{x}}^{\bar{v}}$, $\exists x$, $=_{xy}$}, Ps) is called *a signature* of composition-nominative logic of QE-level. Taking into consideration that a set of composition symbols is determined by the set of variables V, we will use for a signature a simplified notation (V, Ps). Language of $L_{QE}(\Sigma_{QE})$ is represented by a class of formulas Fr_{QE} (V, Ps), which is defined inductively:

- If $P\in Ps$ then $P\in Fr_{QE}(V, Ps)$. Such formulas are called atomic and belong to the class $AFr_{QE}(V, Ps)$ of atomic formulas.
- If $x, y\in V$ then $=_{xy}\in Fr_{QE}(V, Ps)$. Such formulas are called atomic and belong to the class $AFr_{QE}(V, Ps)$ of atomic formulas.
- If Φ, $\Psi\in Fr_{QE}(V, Ps)$ then $(\Phi\vee\Psi)\in Fr_{QE}(V, Ps)$ and $\neg\Phi\in Fr_{QE}(V, Ps)$.
- If $\bar{v}=(v_1,...,v_n)$, $\bar{x}=(x_1,...,x_n)$, $\bar{v}$ is a list of distinct variables, v_i, $x_i\in V$ for all $i\in\{1,...,n\}$, $n\geq 0$, $\Phi\in Fr_{QE}(V, Ps)$ then $R_{\bar{x}}^{\bar{v}}\Phi\in Fr_{QE}(V, Ps)$.
- If $x\in V$, $\Phi\in Fr_{QE}(V, Ps)$ then $\exists x\Phi\in Fr_{QE}(V, Ps)$.

Note, that predicate symbols and symbols of null-ary compositions are atomic formulas.

2. *Interpretational component.* Let $AQE(V, A)$=<$Pr^{V,A}$, $\vee, \neg$, $R_{\bar{x}}^{\bar{v}}$, $\exists x$, $=_{xy}$> be an algebra of quasiary predicates of quantifier-equational level. In this algebra composition symbols obtain their interpretations as operations over predicates. In particular, atomic formulas for null-ary compositions $=_{xy}$ are interpreted as equality predicates in this algebra. Thus, we need to specify interpretation mappings for predicate symbols only. This is done with a mapping $I_{QE}^{Ps}:Ps\xrightarrow{t}Pr^{V,A}$ called a σ-interpretation. Having the interpretational mapping for predicate symbols, we can compositionally construct interpretational mapping for all formulas. A pair ($AQE(V, A)$, I_{QE}^{Ps}) is called a model for $L_{QE}(\Sigma_{QE})$. A model is determined by a tuple J_{QE}^{Ps}=(V, A, I_{QE}^{Ps}) called π-interpretation. In simplified form interpretations will be denoted J. For interpretation J and a formula Φ the meaning of Φ is denoted Φ_J.

3.3 Composition-Nominative Logic $L_{QEU}(\Sigma_{QEU})$ of Quantifier-Equational Level with Unessential Variables

Unessential variables play a role of additional memory and are used for "storing" values during formula transformations. We assume that a set U of unessential variables is an infinite subset of V ($U\subseteq V$). Informally speaking, logic with unessential variables is a logic L_{QE} (Σ_{QE}) with restriction on interpretations of predicate symbols specified by the set U.

1. *Syntactic component*. A tuple Σ_{QEU} =(V, U, {$\vee$, $\neg$, $R_{\bar{x}}^{\bar{v}}$, $\exists x$, $=_{xy}$}, Ps) is called a signature of CNL of QE-level with unessential variables. A class of formulas for L_{QEU} is Fr_{QEU} (V, U, Ps)= Fr_{QE} (V, Ps) .

2. *Interpretational component.* Let $AQE(V, A) = \langle Pr^{V,A}, \vee, \neg, R^{\bar{v}}_{\bar{x}}, \exists x, =_{xy}\rangle$ be an algebra of quasiary predicates of quantifier-equational level. By calling variables from U unessential we actually put a restriction on interpretations of predicate symbols. This restriction asserts that in σ-interpretation $I^{Ps}_{QEU} : Ps \xrightarrow{t} Pr^{V,A}$ for every $P \in Ps$ and for every $d \in {}^{V}A$ the value of $I^{Ps}_{QEU}(P)(d)$ does not depend on values of variables from the set U in d. Formally, for every $d \in {}^{V}A$ the values $I^{Ps}_{QEU}(P)(d)$ and $I^{Ps}_{QEU}(P)(d \setminus\setminus U)$ should either be equal or be undefined simultaneously. Here $d \setminus\setminus U = \{v \mapsto a \in_n d \mid v \notin U\}$. A π-interpretation will be denoted $J^{Ps}_{QEU} = (V, U, A, I^{Ps}_{QEU})$. Indexes may be omitted if they are clear from the context.

This completes a formal definition of logic $L_{QEU}(\Sigma_{QEU})$.

3.4 Classical First-Order Predicate Logic $L_{QECL}(\Sigma_{QECL})$ with Equality

A definition of classical logic differs from definitions of CNL because it is oriented not on quasiary but on n-ary predicates.

1. *Syntactic component.* A tuple $\Sigma_{QECL} = (V, \{\vee, \neg, \exists x, =\}, Ps, arity)$ is called a signature of a classical logic with equality (here $arity: Ps \xrightarrow{t} \{0,1,2, \ldots\}$ is a function that for each predicate symbol yields its arity). A signature in a simplified form is denoted $(V, Ps, arity)$. The language $Fr_{QECL}(V, Ps, arity)$ is defined inductively:

- If $P \in Ps$, $arity(P)=n$, and $x_1, ..., x_n \in V$, then $P(x_1, ..., x_n) \in Fr_{QECL}(V, Ps, arity)$. Such formulas are called atomic and belong to the class $AFr_{QECL}(V, Ps, arity)$ of atomic formulas.
- If $x, y \in V$ then $x=y \in Fr_{QECL}(V, Ps, arity)$. Such formulas are called atomic and belong to the class $AFr_{QECL}(V, Ps, arity)$ of atomic formulas.
- If $\Phi, \Psi \in Fr_{QECL}(V, Ps, arity)$ then $(\Phi \vee \Psi) \in Fr_{QECL}(V, Ps, arity)$ and $\neg\Phi \in Fr_{QECL}(V, Ps, arity)$.
- If $x \in V$, $\Phi \in Fr_{QECL}(V, Ps, arity)$ then $\exists x\Phi \in Fr_{QECL}(V, Ps, arity)$.

2. *Interpretational component.* Let $AQE(V, A)=\langle Pr^{V,A}, \vee, \neg, R^{\bar{v}}_{\bar{x}}, \exists x, =_{xy}\rangle$ be an algebra of quasiary predicates of QE-level (for classical logic we assume that A is nonempty). Note that the renomination composition is present as operation in this algebra, though it is not explicitly used in classical logic. Formulas of the language are interpreted as predicates in this algebra. Atomic formula $x=y$ is interpreted as a predicate $=_{xy}$. To give an interpretation of atomic formulas of the form $P(x_1, ..., x_n)$ we need to specify an interpretational mapping for predicate symbols. In case of classical logic it is specified by a mapping $I^{Ps}_{NAr} : Ps \xrightarrow{t} \bigcup_{n \geq 0} (A^n \xrightarrow{t} Bool)$ such that $I^{Ps}_{NAr}(P) \in A^n \xrightarrow{t} Bool$ if $arity(P) = n$ for $P \in Ps$. This mapping interprets predicate symbols as total n-ary predicates. Thus, π-interpretations have the form $J^{Ps}_{CLE} = (V, A, arity, I^{Ps}_{NAr})$. Such π-interpretation J^{Ps}_{CLE} (or simply J) for every

atomic formula $P(x_1, ..., x_n)$ defines its meaning in $Pr^{V,A}$ as a predicate $P(x_1,..., x_n)_J$ such that $P(x_1, ..., x_n)_J(d) = I^{Ps}_{NAr}(P)(d(x_1), \dots, d(x_n))$ for every $d \in {}^VA$; if one of the values $d(x_1), \dots, d(x_n)$ is not defined then $P(x_1, ..., x_n)_J$ is undefined on d. Let us note that in classical logic d is called *variable valuation* or *variable assignment*. The meaning Φ_J of a complex formula $\Phi \in Fr_{QECL}(Ps, V, arity)$ is defined in a usual way.

For all three logics derived compositions (such as conjunction &, universal quantification $\forall x$, negated equality $\neq_{xy}$ etc.) are defined in a traditional way. In the sequel we consider formulas in their traditional form using infix operations and brackets; brackets can be omitted according to common rules for the priorities of operations (priority of the binary disjunction is weaker than priory of unary operations). We will also consider a more general case for I^{Ps}_{NAr} permitting *partial n*-ary predicates as values of predicate symbols, thus, $I^{Ps}_{NAr}(P) \in A^n \xrightarrow{p} Bool$; still, this generalization does not affect the satisfiability problem due to monotonicity of considered compositions under predicate extensions [13].

To simplify notation we will often omit parameters of logic signatures and write simply L_{QE}, L_{QEU}, and L_{QECL}; for classes of formulas we use notations Fr_{QE}, Fr_{QEU}, and Fr_{QECL}; formulas of these classes will be called QE-, QEU-, and CL-formulas, π-interpretations in L_{QE}, L_{QEU}, L_{QECL} will also be called QE-, QEU-, CL-interpretations respectively.

3.5 Satisfiability Problem

For all three logics the definition of satisfiability can be given in the same way.

A formula Φ is called *satisfiable in a π-interpretation J* if there is $d \in {}^VA$ such that $\Phi_J(d){\downarrow}= T$. We shall denote this by $J \mid\approx \Phi$. A formula Φ is called *satisfiable* if there exists an interpretation J in which Φ is satisfiable. We shall denote this as $\mid\approx \Phi$. We call formulas Φ and Ψ *equisatisfiable* if they are either both satisfiable or both not satisfiable (i.e., unsatisfiable). When needed we will underline the corresponding logic in the satisfiability sign $\mid\approx$, e.g. $\mid\approx_{QE}$, $\mid\approx_{QEU}$, or $\mid\approx_{QECL}$.

Satisfiability of a formula is related to its validity. A formula Φ is called *valid in a π-interpretation J* if there is no $d \in {}^VA$ such that $\Phi_J(d){\downarrow}= F$. We shall denote this as $J \models \Phi$, which means that Φ is not refutable in J. A formula Φ is called *valid* if $J \models \Phi$ for every interpretation J. We call formulas Φ and Ψ *equivalent* if $\Phi_J = \Psi_J$ for every interpretation J.

Due to possible presence of a nowhere defined predicate (which is a valid predicate) we do not have in CNL the property that Φ is satisfiable if Φ is valid (which holds for classical first-order logic). But reduction of satisfiability to validity still holds in CNL: formula Φ is satisfiable in a π-interpretation J iff $\neg\Phi$ is not valid in J.

4 Reduction of Satisfiability Problem for $L_{QE}(\Sigma_{QE})$

The problem discussed in this paper is to check whether $|\approx_{QE}\ \Phi$ holds given an arbitrary formula $\Phi\in Fr_{QE}(W, Ps)$; here we choose W as an initial set of variables in the considered logic. Our main aim is to transform this QE-formula Φ to an equisatisfiable formula Φ_{CL} of classical first-order predicate logic with equality so that we can use existent methods for solving this problem developed for classical logic. To carry out necessary equivalent transformations we need to consider Φ in an intermediate logic – CNL of QE-level with unessential variables – extending the initial set of variables W with a set U of unessential variables ($W\cap U=\varnothing$). For these needs we will consider a logic L_{QEU} with the signature $\Sigma_{QEU}=(V, U, \{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy}\}, Ps)$, where $V=W\cup U$. Within L_{QEU} we transform Φ to a formula Φ_{UR} being in a special normal form; then the latter formula is translated to its classical counterpart Φ_{CL}.

The overall circular reduction scheme is grounded on following statements.

1. From $|\approx_{QE}\ \Phi$ follows $|\approx_{QEU}\ \Phi$ (lemma 1).
2. From $|\approx_{QEU}\ \Phi$ follows $|\approx_{QEU}\ \Phi_{UR}$ (lemma 2, 3).
3. From $|\approx_{QEU}\ \Phi_{UR}$ follows $|\approx_{QECL}\ \Phi_{CL}$ (lemma 4).
4. From $|\approx_{QECL}\ \Phi_{CL}$ follows $|\approx_{QEU}\ \Phi_{UR}$ (lemma 5).
5. From $|\approx_{QEU}\ \Phi_{UR}$ follows $|\approx_{QEU}\ \Phi$ (lemma 2).
6. From $|\approx_{QEU}\ \Phi$ follows $|\approx_{QE}\ \Phi$ (lemma 6).

Lemma 1. Let $\Phi\in Fr_{QE}(W, Ps)$. Then from $|\approx_{QE}\ \Phi$ follows $|\approx_{QEU}\ \Phi$.

Consider the transformation rules (T1-T9) of the form $\Phi_l \mapsto \Phi_r$, where Φ_l, $\Phi_r \in Fr_{QEU}(V,U)$.

T1) $R_{\bar{x}}^{\bar{v}} =_{xy} \mapsto =_{\tilde{x}\tilde{y}}$

T2) $R_{\bar{x}}^{\bar{v}} \neg\Phi \mapsto \neg R_{\bar{x}}^{\bar{v}}\Phi$

T3) $R_{\bar{x}}^{\bar{v}}(\Phi_1 \vee \Phi_2) \mapsto R_{\bar{x}}^{\bar{v}}\Phi_1 \vee R_{\bar{x}}^{\bar{v}}\Phi_2$

T4) $R_{x_1,\ldots,x_n,y_1,\ldots,y_m}^{v_1,\ldots,v_n,w_1,\ldots,w_m}\ R_{s_1,\ldots,s_n,z_1,\ldots,z_k}^{v_1,\ldots v_n,u_1,\ldots,u_k}\Phi \mapsto R_{\alpha_1,\ldots,\alpha_n,y_1,\ldots,y_m,\beta_1,\ldots,\beta_k}^{v_1,\ldots,v_n,w_1,\ldots,w_m,u_1,\ldots,u_k}\Phi$

T5) $R_{\bar{x}}^{\bar{v}}\ \exists y\ \Phi \mapsto \exists y\ R_{\bar{x}}^{\bar{v}}\Phi$, when $y\notin\{\bar{v}, \bar{x}\}$

T6) $R_{z,\bar{x}}^{y,\bar{v}} \exists y\Phi \mapsto \exists y\, R_{\bar{x}}^{\bar{v}}(\Phi)$

T7) $R_{y,\bar{x}}^{z,\bar{v}} \exists y\Phi \mapsto \exists u\ R_{y,\bar{x}}^{z,\bar{v}}\ R_u^y\ \Phi$, $u\in U$, u does not occur in the formula on the left hand side of the rule.

T8) $R_{\bar{q}}^{\bar{u}}\, P \mapsto R_{z,\bar{q}}^{z,\bar{u}}\, P$ (in case when vectors $\bar{u}, \bar{v}$ are empty this rule is represented as $P \mapsto R_z^z\ P$.

T9) $R^{u_1,...,u_i,...,u_j,...,u_n}_{q_1,...,q_i,...,q_j,...,q_n}P = R^{u_1,...,u_j,...,u_i,...,u_n}_{q_1...,q_j,...,q_i,...,q_n}P$

Here for the rule T1 $\tilde{x} = x(\bar{v}/\bar{x})$, $\tilde{y} = y(\bar{v}/\bar{x})$, for the rule T4 $\alpha_i = s_i(v_1,...,v_n, w_1,...,w_m \,/\, x_1,...,x_n, y_1,...,y_m)$, $\beta_j = z_j(v_1,...,v_n, w_1,...,w_m \,/\, x_1,...,x_n, y_1,...,y_m)$, where $r(b_1,...,b_q \,/\, c_1,...,c_q) = r$ if $r \notin \{b_1,...,b_q\}$, $r(b_1,...,b_q \,/\, c_1,...,c_q) = c_i$ if $r = b_i$ for some i.

The rule T4 represents explicitly the result of functional composition of parameters of two successive renominations.

The rule T7 permits to assume w.l.o.g. that all quantified variables in initial formula are different.

Lemma 2. Let $\Phi_l, \Phi_r \in Fr_{QEU}(V,U,Ps)$ be such formulas that Φ_r is a result of application of some T1-T9 rule to Φ_l. Then Φ_l and Φ_r are equisatisfiable in L_{QEU}.

A formula Φ is said to be in *unified renominative normal form* (URNF) if the following requirements are satisfied:

- the renomination composition is only applied in Φ to predicate symbols. It means that for every sub-formula of the form $R^{\bar{v}}_{\bar{x}}\Psi$ we have that $\Psi \in Ps$;
- for every pair of its renominative atoms $R^{\bar{u}}_{\bar{q}}P$ and $R^{\bar{w}}_{\bar{y}}Q$ we have that vectors $\bar{u}$ and $\bar{w}$ coincide; so, in all renominative atoms the lists of their upper names are the same;
- for every renominative atom $R^{\bar{v}}_{\bar{x}}P$ and every quantifier $\exists y$ that occurs in the initial formula Φ we have that $y \in \bar{v}$.

When formula is in *URNF* we call its atomic subformula $R^{\bar{v}}_{\bar{x}}P$ *a renominative atom* ($P \in Ps$). Note that if a formula is in *URNF* then every its subformula is in URNF as well.

Lemma 3. Given an arbitrary formula $\Phi \in Fr_{QEU}(V,U,Ps)$ we can construct its unified renominative normal form $urnf[\Phi]$ by applying rules T1-T9.

According to lemmas 2 and 3, we can think of a total multi-valued (non-deterministic) mapping $urnf : Fr_{QEU} \xrightarrow{tm} Fr_{QEU}$ that transforms in a satisfiability-preserving way every QEU-formula to its URNF.

In order to reduce the satisfiability problem in L_{QEU} to that of L_{QECL} we extend the set of basic values A with additional value ε. Informally, this value will represent undefined components of nominative sets.

We formalize the syntactical reduction $clf : Fr_{QEU}(V,U,Ps) \xrightarrow{t} Fr_{QECL}(V,Ps,arity)$ of QEU-formulas in unified renominative normal form to CL-formulas inductively as follows:

1. $clf[P] = P$
2. $clf[=_{xy}] = =_{xy}$
3. $clf[R^{v_1,...,v_n}_{x_1,...,x_n}P] = P(x_1,...,x_n)$
4. $clf[\neg\Phi] = \neg clf[\Phi]$

5. $clf[(\Phi_1 \vee \Phi_2)] = (clf[\Phi_1] \vee clf[\Phi_2])$
6. $clf[\exists x\Phi] = \exists x(x \neq e \,\&\, clf[\Phi])$, $e \in U$, *e* is a predefined variable.

Note that all applications of the 6-th rule introduce the same variable *e*; *e* is some predefined variable from *U* in the sense that it does not occur in URNF.

This reduction transforms the formula to the language of classical logic but preserves its satisfiability.

Given a formula $\Phi \in Fr_{QEU}(V, U, Ps)$ in unified renominative normal form we denote by $V_\Phi \subseteq V$ the set of all variables that occur as upper names in renominative atoms of Φ .

Lemma 4. Let Φ be a formula in unified renominative normal form, $\Phi \in Fr_{QEU}(V, U, Ps)$. Then from $|\approx_{QEU} \Phi$ follows $|\approx_{QECL} clf[\Phi]$.

Lemma 5. Let Φ be a formula in renominative normal form, $\Phi \in Fr_{QEU}(V, U, Ps)$. Then from $|\approx_{QECL} clf[\Phi]$ follows $|\approx_{QEU} \Phi$.

Lemma 6. Let $\Phi \in Fr_{QE}(W, Ps)$. Then from $|\approx_{QEU} \Phi$ follows $|\approx_{QE} \Phi$.

Lemmas 1-6 justify all reductions described in the article and the main theorem of the article.

Theorem. Let $\Phi \in Fr_{QE}(W, Ps)$. Then $|\approx_{QE} \Phi$ if and only if $|\approx_{QECL} urnf[clf[\Phi]]$.

The theorem states the reduction of satisfiability problem in composition-nominative logic of quantifier-equational level to the satisfiability problem in classical first-order logic with equality.

Let us illustrate the method proposed on a simple example.

Example. Consider the following QE-formula Φ with one predicate symbol *P* :

$$\Phi = P \,\&\, R_x^z(=_{zy} \,\&\forall z\neg P)$$

Let us construct its unified renominative normal form Φ_{UR} .

$\Phi = P \,\&\, R_x^z(=_{zy} \,\&\forall z\neg P) \mapsto$ / push the renomination down to predicate symbols/

$\mapsto P \,\&=_{xy} \&R_x^z(\forall z\, \neg P) \mapsto$ /renomination is removed due to T6/ $\mapsto$

$\mapsto P\&=_{xy} \&(\forall z\neg\, P) \mapsto$ /add R_z^z to *P* as the predicate occurs under $\forall z$ / $\mapsto$

$\mapsto P\&=_{xy} \&(\forall z\, \neg R_z^z\, P) \mapsto$ /unify renominative atoms / $\mapsto$

$\mapsto R_z^z P \,\&=_{xy} \&\forall z\, \neg R_z^z\, P = \Phi_{UR}$.

Note that we use derived transformation rules that handle compositions & and $\forall$.

Now $\Phi_{CL} = cnl[\Phi_{UR}] = P(z) \,\&\, x = y \,\&\, \forall z((z \neq e) \to \neg P(z))$. Formula Φ_{CL} is satisfiable in L_{CL}. That means that Φ is satisfiable in L_{QE}.

Indeed, let $J = (W, A, I)$ be such an interpretation that $W=\{x,y,z\}$, $A=\{1,2\}$. Let $I(P)(d)\downarrow = F$ if a pair $z \mapsto a \in_n d$ for some $a \in A$ and *T* in all other cases. In other words, the predicate *P* takes the value *T* on some data *d* if the variable *z* is undefined in *d*. Now we have that $\Phi_J([\, x \mapsto 1, y \mapsto 1\,])\downarrow = T$.

5 Related work

Many different aspects of the composition-nominative approach such as partiality, compositionality, nominativity, have long history of development, which is also reflected in works in the field of logic and computer science.

The importance of partiality, for example, was already being discussed in detail by the time of 80-ties [14], and many different approaches have emerged since that time. In [15, 16] there is a survey of some of those and a comparison of different formalisms. Partiality receives more and more attention nowadays, the support for partial functions is being introduced in theorem proving systems and validity checkers [17, 18].

Compositionality can be traced back to works of G. Frege; the history of this principle is presented in [19]. The importance of the compositionality principle grows due to the necessity of investigation and verification of complex systems [20, 21], in particular, concurrent systems [22]. Our approach takes compositionality as a basic principle, thus, the constructed formal languages are compositional by construction when we consider functions (predicates) as meanings of expressions (of formulas).

Nominativity is also a fundamental aspect not only in computer science but in other branches of science as well, especially in philosophy. This topic requires a special treatment, but here we would like to mention nominal logic [23] only, which has similarities with the logic defined in this paper. Nominal logic addresses such special questions of nominativity as name bindings, swapping, and freshness. The predicates investigated in nominal logic should be equivariant (their validity is invariant under name swapping); in our work we consider general classes of partial predicates.

A thorough comparison of composition-nominative approach with other approaches that address compositionality, nominativity or allow reasoning about partial functions and predicates is by far beyond the scope of this paper, but still we would like to stress on the important differences. Our approach is based on algebras of partial predicates over nominative data, and especially, algebras of quasiary functions and predicates as opposed to traditional algebras of *n*-ary functions and predicates. It involves new compositions, in particular, renomination composition, which take into account nominative aspects of data structures. Composition-nominative approach also prescribes the semantic-syntactic style of logic definitions. This style simplifies construction and investigation of such logics.

6 Conclusions

This paper investigates the satisfiability problem for composition-nominative logic (CNL) of quantifier-equational level. As a main result we have shown that this problem can be reduced by using more powerful language to the satisfiability problem for classical predicate logic with equality. Thus, existent state-of-the-art methods and techniques for checking satisfiability in classical logics can also be applied to CNL.

Future work on the topic will include investigation of satisfiability problem for richer CNL of predicate-function level and for CNL over hierarchic nominative data. Hierarchic data permit to represent such complex structures as lists, stacks, arrays etc; thus, such logics will be closer to program models with more rich data types. Another direction is related with identification of classes of formulas in various types of CNL for which satisfiability problem can be solved efficiently. In particular, this concerns specialized theories, where some predicates have specific interpretations and several axioms shall hold for such interpretations. This is often referred to as satisfiability modulo theory (SMT) problem [24]. At last, prototypes of software systems for satisfiability checking in CNL should be developed.

References

1. Mendelson E.: Introduction to Mathematical Logic, 4th ed. Chapman & Hall, London (1997)
2. Kroening D., Strichman O.: Decision Procedures – an Algorithmic Point of View. Springer-Verlag, Berlin Heidelberg (2008)
3. Marques-Silva J.: Practical Applications of Boolean Satisfiability. In: Workshop on Discrete Event Systems (28-30 May 2008, Goteborg, Sweden), pp. 74--80 (2008)
4. Nieuwenhuis R., Oliveras A., Tinelli C.: Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). J ACM 53, pp. 937--977 (2006)
5. de Moura L., Bjørner N.: Satisfiability Modulo Theories: Introduction and Applications. COMMUN ACM 54 (9), pp. 69--77 (2011)
6. Nikitchenko, N.S.: A Composition Nominative Approach to Program Semantics. Technical Report IT–TR 1998-020, Technical University of Denmark (1998)
7. 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), pp. 221--231. Springer, London (1995)
8. Nielson H.R., Nielson F.: Semantics with Applications: A Formal Introduction. John Wiley & Sons Inc (1992)
9. Nikitchenko M.S.: Composition-nominative aspects of address programming. Kibernetika I Sistemnyi Analiz 6, pp. 24--35 (In Russian) (2009)
10. Nikitchenko M.S., Shkilnyak S.S.: Mathematical logic and theory of algorithms. Publishing house of Taras Shevchenko National University of Kyiv, Kyiv, (in Ukrainian) (2008)
11. Kleene, S. C.: Introduction to Metamathematics. Van Nostrand, New York (1952)
12. Shkilniak S. S.: First-order logics of quasiary predicates. Kibernetika I Sistemnyi Analiz 6, pp. 32--50, (in Russian) (2010)
13. Nikitchenko M.S., Tymofieiev V.G.: Satisfiability Problem in Composition-Nominative Logics. In: Proceedings of the Eleventh International Conference on Informatics INFORMATICS'2011, Roznava, Slovakia, November 16-18, pp. 75--80 (2011)
14. Blamey, S.: Partial Logic. In Gabbay D., Guenthner F. (eds.), Handbook of Philosophical Logic, Volume III, D. Reidel Publishing Company (1986)
15. Jones C. B.: Reasoning About Partial Functions in the Formal Development of Programs. ENTCS 145, pp. 3--25 (2006)
16. Owe O.: Partial Logics Reconsidered: A Conservative Approach. FORM ASP COMPUT 5, pp. 208--223 (1997)
17. Mehta F. A.: Practical Approach to Partiality A Proof Based Approach. LNCS, vol. 5256, pp. 238--257. Springer, Heidelberg (2005)
18. Berezin S., Barrett C., Shikanian I., Chechik M., Gurfinkel A., Dill D.L.: A Practical Approach to Partial Functions in CVC Lite. ENTCS 125, pp. 13--23 (2005)

19. Janssen T.M.V.: Compositionality. In van Benthem J., ter Meulen A. (eds.), Handbook of Logic and Language, pp. 417--473. Elsevier and MIT Press (1997)
20. de Roever, W.-P., Langmaack H., Pnueli A. (eds.): Compositionality: The Significant Difference. LNCS, vol. 1536, VIII. Springer, Heidelberg (1998)
21. Bjørner D., Eir A.: Compositionality: Ontology and Mereology of Domains. LNCS, vol. 5930, pp. 22--59. Springer, Heidelberg (2010)
22. Dams D., Hannemann U., Steffen M. (eds.): Concurrency, Compositionality, and Correctness, Essays in Honor of Willem-Paul de Roever. LNCS, vol. 5930. Springer, Heidelberg (2010)
23. Pitts, A. M.: Nominal Logic, A First Order Theory of Names and Binding. INFORM COMPUT 186, pp. 165--193 (2003)
24. Barrett C., Sebastiani R, Seshia S. A., Tinelli C.: Satisfiability Modulo Theories. In: Biere A., Heule M., van Maaren H., Walsh T. (eds.), Handbook of Satisfiability. IOS Press (2009)