Decidability Results for Sets with Atoms

Agostino Dovier[♭] Andrea Formisano[♮] Eugenio G. Omodeo[♯]

Abstract

Formal Set Theory is generally concerned with *pure* sets and decidability results for some classes of formulas are well-known. In practical applications, instead, it is common to assume the existence of a number of primitive objects (also called *atoms*), that can be members of sets but behave differently from them. If these entities are viewed as member-less, standard extensionality axioms must be revised. However, decidability results can be proved via reduction to the pure case and direct goal-driven algorithms have been already proposed. An alternative approach of modeling atoms that allows to retain original extensionality was proposed by Quine: atoms are self-singletons. In this paper we face the satisfiability problem in this case: we show the decidability of $\exists^*\forall$-formulas and we develop a goal-driven unification algorithm.

Keywords: Set-hyperset theories, Satisfiability problem, Syllogistics, Prenex sentences, Quantifier elimination, Unification.

1 Introduction

With his proof [13] that the classical set theory ZFC is complete w.r.t. $\forall^*\exists$- / $\exists^*\forall$-sentences, Gogol was one of the few forerunners of a very fruitful stream of research on computable set theory initiated twenty years ago.

Reverse logic revealed that Gogol's decidability result does not depend on strong assumptions (such as the axiom of choice or the subset axioms [12]), while algorithmic analysis assessed that his $\exists^*\forall$-provability problem belongs to the NP-complete complexity class, cf. [16].

Generalizations of Gogol's result consist in

- treating larger and larger collections of formulas in set theory [4, 6], by naïve reference to a standard and well-understood universe of sets such as the von Neumann hierarchy, or its sub-hierarchy of the hereditarily finite sets first investigated by Ackermann;
- seeking a syllogistic decomposition of the input formula r (cf., e.g., [5, 17]), so as to fully classify those set assignments for the existential variables that make r true;

[♭] Università di Verona, Dipartimento di Informatica. E-mail: dovier@sci.univr.it
[♮] Università di Perugia, Dipartimento di Matematica e Informatica. E-mail: formis@dipmat.unipg.it
[♯] Università di L'Aquila, Dipartimento di Matematica Pura ed Applicata. E-mail: omodeo@univaq.it

- adjusting the decision algorithm to, or enhancing it [3], under, varied axiomatic assumptions, e.g. by shifting the focus from regular set theory to hyperset theory [18].

Applications of Gogol's result include the design of a unification algorithm for sets that nicely fits into Constraint Logic Programming with Sets [11]. The latter algorithm was shown in [10] to be generalizable to a variety of set-theoretic contexts, and in particular to a hyperset context [7].

Formal Set Theory is generally concerned with *pure* sets. Namely, members of sets are sets themselves. $\emptyset$ denotes typically the unique set devoid of elements. In practical applications, it is common to assume the existence of a number of primitive objects (also called *atoms*), that can be members of sets but, generally, are not supposed to contain elements. Problems related to sets with atoms (also called *hybrid sets*) can be reduced to the pure case using rather standard reductions (see, e.g. [15]). However, when developing a constraint solver to deal with the hybrid case, practical considerations lead to a direct solution of the problems. This is the approach adopted, for instance, in [11] where the language $CLP(\mathcal{SET})$ is presented: atoms are there viewed as standard first-order terms that can not contain elements. A third approach to the handling of hybrid sets is justified by the following citation:

> *Thereupon the question arises how to interpret '$y \in z$' where z is an individual. The convention that first suggests itself and has commonly been adopted in the literature is that in such a case '$y \in z$' is simply false for all y; individuals do not have members. But there is a different convention that proves much more convenient $\cdots$ none of the utility of class theory is impaired by counting an individual, its unit class, the unit class of that unit class, and so on, as one and the same thing.* ([21], pp. 30–31)

In this paper we tune up a decision algorithm for $\exists^*\forall$-sentences, already adapted in [18] from [13] to very weak set/hyperset theories devoid of atoms, to sets with atoms. Atoms are treated here as self-singleton sets, as suggested above and this constitutes a novelty; indeed, to date the work in computable set theory treats atoms as member-less entities (which are not sets, and hence differ from $\emptyset$). One can see as a drawback in the traditional approach that it forcibly brings into play 'colored' sets in order to ensure a smooth treatment of unification, thus disrupting both elegance and intuitive appeal of the extensionality axiom.

We will discover, by the end, that the set unification algorithm of [8] can be painlessly adapted to sets with self-singleton atoms. This result discloses a viable alternative to the approach adopted in {log} for handling atoms in a logic programming language with sets. Indeed, it can be seen as a new foundational contribution to the design of theories suitably supporting programming with sets.

In Section 2 we describe the weak set theory needed for the scope of the work. In the subsequent Section 3 we show how to develop a decidability test for a class of formulas of the language used. Then, in Section 4 we develop a goal-driven unification algorithm for our theory. In Section 5 we draw some conclusions.

2 Set Axioms

In this section we will describe an axiomatization suitable for the purposes of the research carried on in this paper. This axiomatization can be considered as

- a subtheory of standard Zermelo-Skolem-Fraenkel (cf. [12, 14]),
- a subtheory of Aczel's hyperset theory (cf. [1, 2]), and
- a tiny theory of sets ultimately based on atoms.

Despite being rather weak, the theories to be specified will be able to 'judge' every $\exists^*\forall$-sentence r, by entailing either r or $\neg r$ as a theorem.

We will characterize atoms by a monadic predicate ur. Atoms are viewed here, in agreement with [21], as self-singletons (i.e., entities a fulfilling the identity $a = \{a\}$) in a universe which, save for what regards them, is well-founded by membership.

A weak theory of sets can be based on the axioms of *extensionality*, *null set*, single-element *addition* and *removal*, which are (see, e.g., [18]):[1]

$$\begin{array}{ll} \textbf{(E)} & \forall v\,(v \in X \leftrightarrow v \in Y) \rightarrow X = Y, \\ \textbf{(N)} & \exists z\,\forall v\; v \notin z, \\ \textbf{(W)} & \exists w\,\forall v\,(v \in w \leftrightarrow v \in X \vee v = Y), \\ \textbf{(L)} & \exists \ell\,\forall v\,(v \in \ell \leftrightarrow v \in X \wedge v \neq Y). \end{array}$$

To inject atoms into a set theory, one has to restate **(E)** w.r.t. the formulation seen above. If atoms were viewed as entities devoid of members, then **(E)** should be revised, since an atom and the null set $\emptyset$, although distinct, would have the 'same' members. For us, the distinguishing feature of an atom a is the fulfillment of the condition $a = \{a\}$. An advantage is that **(E)** needs no change.

However something must be stated about well-foundedness (or non well-foundedness) of sets. A usual way to assert well-foundedness is via the *regularity* axiom:

$$\textbf{(R)} \qquad \exists r\,\forall v\,\big((r = X \vee r \in X) \wedge \neg(v \in r \wedge v \in X)\big)$$

We need to revise **(R)** because its formulation allows to entail $\neg\exists x\;\mathsf{ur}(x)$. The revised version $(\mathbf{R}')$ of **(R)**, and a *plenitude* axiom (ensuring that there exist atoms at will), are as follows:

$$\begin{array}{ll} (\mathbf{R}') & \exists r\,\Big(\big(r = X \vee (\neg\,\mathsf{ur}(r) \wedge r \in X)\big) \wedge (\forall v \in X)\big(v \in r \;\rightarrow\; \mathsf{ur}(v)\big)\Big)\,, \\ (\mathbf{D}_\in) & (\exists u \in u)(\forall v \in X)\;(u \notin v)\,. \end{array}$$

For a very weak, and yet useful, theory of sets with atoms, only the axioms **(E)**, **(N)**, **(W)**, and **(L)** need to be added.

Notice that if we were to postulate scarcity $\neg\exists x\;\mathsf{ur}(x)$ instead of plenitude, this new $(\mathbf{R}')$ would become equivalent to **(R)**. At any rate, even in the new context the rôle of **(R)** remains the one of forbidding the formation of genuine membership cycles:

[1] Capital letters denote universally quantified variables.

Lemma 2.1 *It follows from the definition of* ur *and from the axioms* **(E)**, **(N)**, **(W)**, **(L)**, *and* $(\mathbf{R}')$, *that for every natural number* n,

$$(\mathbf{A}^{(n)}) \qquad X_0 \in X_1 \in \cdots \in X_n \in X_0 \to \mathsf{ur}(X_0) \wedge X_0 = X_1 = \cdots = X_n.$$

Proof: (Sketch) After constructing (by **(N)**,**(W)**) the set $X_\star = \{X_0, \ldots, X_n\}$, one can treat the two cases $(\exists\, v \in X_\star)\mathsf{ur}(v)$ and $(\neg\exists\, v \in X_\star)\mathsf{ur}(v)$ separately, and $(\mathbf{R}')$ intervenes only in the latter. □

The axiom $(\mathbf{D}_{\notin})$, called *anti-diagonal* in [6] is needed for the decidability procedure:

$$(\mathbf{D}_{\notin}) \qquad \exists z\,(z \notin z \wedge (\forall x \in Y)(z \notin x))$$

This property follows from the other existing axioms:

Lemma 2.2 *The statement* $(\mathbf{D}_{\notin})$ *ensues from* **(E)**, **(N)**, **(W)**, **(L)**, *and* $(\mathbf{R}')$.

Proof: (Sketch) Given Y, we are to find a z such that both $z \notin z$ and $\neg\exists x$ s.t. $z \in x \wedge x \in Y$ hold. In view of $(\mathbf{A}^{(1)})$, it suffices to take $z = Y \setminus \{Y\}$. □

3 A Decision Technique for $\exists^*\forall$-Sentences on Sets

In this section we present a two-phase decision technique to test $\exists^*\forall$-sentences for satisfiability over sets with atoms. First, we introduce a technique that enables one to reduce the provability problem for $\exists^*\forall$-sentences to the same problem for (equivalent) formulas involving *bounded* quantifiers only. Secondly, in Sec. 3.2 we describe a decision algorithm for proving or refuting formulas of the latter kind.

3.1 Quantifier-bounding technique

Let us start by outlining a technique, introduced in [16, 18], for re-expressing any given sentence in the predicates $=$ and $\in$, of the form $\exists\, x_1 \cdots \exists\, x_n\, \forall\, y\, p$, with p devoid of quantifiers, constants, and functors, as a sentence which involves restricted universal quantifiers $\forall\, y \in x_j$ instead of the unrestricted $\forall\, y$.

The postulates on pure sets, as well as those on pure hypersets, were exploited in order to rewrite the given sentence into equivalent form

$$(\exists\, x_1, \ldots, x_n)\big(p_1 \wedge \cdots \wedge p_g \wedge (\forall\, y \in x_1)p_{g+1} \wedge \cdots \wedge (\forall\, y \in x_n)p_{g+n}\big)\,.$$

Here we will see how to adapt the same technique to sets with atoms.

Atom elimination. Let $\Phi_0 \equiv \exists\, x_1 \cdots \exists\, x_n \forall\, y p$ be the input formula. In p we now admit constant symbols (denoting atoms). Let $a_1, \ldots, a_k$ be all the distinct constants occurring in p and let $x_{n+1}, \ldots, x_{n+k}$ be new distinct variables. Then a formula Φ_1 equivalent to Φ_0 can be defined as follows:[2]

$$\Phi_1 \quad \equiv_{\text{Def}} \quad \exists\, x_1 \cdots \exists\, x_{n+k}\, \forall\, y \quad \underbrace{\left(\begin{array}{l} p[a_1 \rightsquigarrow x_{n+1}, \ldots, a_k \rightsquigarrow x_{n+k}] \wedge \\ \bigwedge_{i=n+1}^{n+k} (x_i \in x_i \wedge \bigwedge_{j=i+1}^{n+k} x_i \neq x_j) \end{array}\right)}_{\tilde{p}}.$$

[2] By $p[x \rightsquigarrow y]$ we denote the formula obtained by replacing all occurrences of x with y in p.

Quantifier bounding. In this stage, by moving inward the universal quantifier, the formula Φ_1 is rewritten as to yield a formula Φ_2 of the form:

$$\Phi_2 \quad \equiv_{\text{Def}} \quad \exists\, x_1 \cdots \exists\, x_{n+k} \quad \left(\begin{array}{l} \bigwedge_i \tilde{p}[y \rightsquigarrow x_i] \wedge \\ \bigwedge_j (\forall y \in x_j) q_j \wedge \\ \bigwedge_h (\forall y \in y) r_h \wedge \\ \bigwedge_\ell (\forall y \notin y) s_\ell \end{array} \right)$$

where $\tilde{p}$ is the subformula of Φ_1 as defined in the previous stage. Observe that in the third conjunct of Φ_2 (i.e., $\bigwedge_h (\forall y \in y) r_h$), the constraint $y \in y$ forces y to be an atom, while the constraint $y \notin y$ ensures that y is not an atom in s_ℓ. Let us explain this rewriting process by means of a simple example (for the general description see also [18]).

Example 3.1 Consider the formula (i.e., let $n + k = 2$ in the above Φ_1):

$$\Phi_1 \equiv \exists x_1 \exists x_2 \forall y\, \tilde{p}$$

where $\tilde{p}$ is a Boolean combination of literals $u\, \pi\, v$ with u, v in $\{x_1, x_2, y\}$ and π a predicate symbol in $\{=, \in\}$. For any given assignment of values to the variables x_1 and x_2 satisfying Φ_1, the formula must hold for each possible values of y. A particular y may be equal either to x_1 or to x_2, or it may be the case that $y \in x_1$, or $y \in x_2$, and so on. We proceed by case analysis, first by splitting $\forall y\, \tilde{p}$ into various mutually exclusive sub-cases, and then by processing each of them. More precisely, we start by observing that (by the Boolean tautology $A \vee B \vee (\neg A \wedge \neg B)$) it holds:

$$\Phi_1 \equiv \exists x_1 \exists x_2 \forall y \Big(\; (y = x_1 \wedge \tilde{p}) \;\vee\; (y = x_2 \wedge \tilde{p}) \;\vee\; \underbrace{(y \neq x_1 \wedge y \neq x_2 \wedge \tilde{p})}_{p'} \; \Big)$$

Similarly, we have that

$$p' \equiv \Big(\; (y \in x_1 \wedge p') \;\vee\; (y \in x_2 \wedge p') \;\vee\; \underbrace{(y \notin x_1 \wedge y \notin x_2 \wedge p')}_{p''} \; \Big)$$

and moreover

$$p'' \equiv \Big(\; (y \in y \wedge p'') \;\vee\; (y \notin y \wedge p'') \; \Big)$$

To sum up, we have that

$$\Phi_1 \equiv \exists x_1 \exists x_2 \forall y \left(\begin{array}{l} (y = x_1 \wedge \tilde{p}) \vee \\ (y = x_2 \wedge \tilde{p}) \vee \\ (y \in x_1 \wedge y \neq x_1 \wedge y \neq x_2 \wedge \tilde{p}) \vee \\ (y \in x_2 \wedge y \neq x_1 \wedge y \neq x_2 \wedge \tilde{p}) \vee \\ (y \in y \wedge y \notin x_1 \wedge y \notin x_2 \wedge y \neq x_1 \wedge y \neq x_2 \wedge \tilde{p}) \vee \\ (y \notin y \wedge y \notin x_1 \wedge y \notin x_2 \wedge y \neq x_1 \wedge y \neq x_2 \wedge \tilde{p}) \end{array} \right)$$

By exploiting the properties of universal quantification, the formula Φ_1 can be proved equivalent to Φ_2 below:

$$\Phi_2 \equiv_{\text{Def}} \exists x_1 \exists x_2 \left(\begin{array}{l} \tilde{p}[y \rightsquigarrow x_1] \wedge \\ \tilde{p}[y \rightsquigarrow x_2] \wedge \\ (\forall y \in x_1)(y \neq x_1 \wedge y \neq x_2 \rightarrow \tilde{p}) \wedge \\ (\forall y \in x_2)(y \neq x_1 \wedge y \neq x_2 \rightarrow \tilde{p}) \wedge \\ (\forall y \in y)(y \notin x_1 \wedge y \notin x_2 \wedge y \neq x_1 \wedge y \neq x_2 \rightarrow \tilde{p}) \wedge \\ (\forall y \notin y)(y \notin x_1 \wedge y \notin x_2 \wedge y \neq x_1 \wedge y \neq x_2 \rightarrow \tilde{p}) \end{array} \right)$$

We will see in the following that the particular form of the 3rd–6th conjuncts will allow us to further simplify the formula $\tilde{p}$.

Self-singleton treatment. Consider a formula Φ_2 as obtained in the preceding step. Conjuncts of the kind $\tilde{p}[y \rightsquigarrow x_i]$ and $(\forall y \in x_j) q_j$ in Φ_2, are already in the desired form. In presence of self-singletons, the treatment of the conjuncts $(\forall y \in y) r_h$ and $(\forall y \notin y) s_\ell$ differs from the one presented in [18].

The elimination of both the quantifiers of the form $(\forall\, y \in y) r_h$ and $(\forall\, y \notin y) s_\ell$ crucially depends on the following lemma:

Lemma 3.2 (Diagonalization) *Sets with atoms fulfill both the law*

$$\mathbf{(Y_{\notin})} \qquad \begin{array}{l} \big(\bigwedge_{0<j\leq m} \bigwedge_{m<g\leq n} X_j \neq X_g \big) \rightarrow (\exists\, y \notin y) \big((\bigwedge_{0<i\leq n} y \notin X_i) \wedge \\ \qquad (\bigwedge_{0<i\leq n} y \neq X_i) \wedge (\bigwedge_{0<j\leq m} X_j \in y) \wedge (\bigwedge_{m<g\leq n} X_g \notin y) \big) , \end{array}$$

with m, n integers, $0 \leq m \leq n$, and the law

$$\mathbf{(Y_{\in})} \qquad (\exists\, y \in y) \big(\bigwedge_{0<i\leq n} X_i \neq y \wedge y \notin X_i \big) .$$

Proof: The proof is strongly based on the laws $\mathbf{(D_{\notin})}$ and $\mathbf{(D_{\in})}$. The only deviation worth of notice w.r.t. the similar result in [18] lies in the elimination of the forms $(\forall\, y \in y)\, r_h$. We know that r_h has the form $\bigwedge_{0<i\leq n}(X_i \neq y \wedge y \notin X_i) \rightarrow q'$, and since

$$(\forall\, y \in y) \left(\bigwedge_{0<i\leq n} y \notin X_i \rightarrow \bigwedge_{0<g\leq n} X_g \notin y \right)$$

holds, we can reduce $(\forall\, y \in y)\, r_h$ to a formula q'', obtainable from q' by replacing $X_g \in y$ by `false` for $g = 1, \ldots, n$. □

The final formula can be obtained from Φ_2 by exploiting the above lemma and in virtue of the form of r_h and s_ℓ, as described by the following algorithm:

1. Consider a formula of the form $(\forall y\, \xi\, y)\, (\bigwedge_i (y \neq x_i \wedge y \notin x_i) \rightarrow \varphi)$ where ξ is either $\in$ or $\notin$ and φ is an unquantified conjunction of literals on the variables $x_1, \ldots, x_{n+k}, y$ and predicate symbols $=$ and $\in$. W.l.o.g. we can assume that φ is in conjunctive normal form, namely $\varphi \equiv D_1 \wedge \cdots \wedge D_z$.

2. This allows us to distribute the $\wedge$ over $\forall$ and to consider separately formulas of the form

$$\psi \equiv \forall y \left(y \,\xi\, y \wedge \bigwedge_i (y \neq x_i \wedge y \notin x_i) \rightarrow D \right)$$

where D is a finite disjunction of positive and negative literals of the form $u \,\pi\, v$ with π in $\{=, \in\}$ and u, v variables from $x_1, \ldots, x_{n+k}, y$.

3. The consequent D in ψ can be simplified by exploiting the premises of the implication. There are several cases to take care of:
   (a) The literals of the form $y = y, y \neq y$ are replaced by true and false, respectively.
   (b) The literals of the form $y \in y, y \notin y, y \in x_i, y \notin x_i, y = x_i, y \neq x_i$: are replaced by true or false, depending on the premisses of the implication.
   (c) If at least one of the disjuncts of D has been replaced by true, then the whole ψ can be replaced by true. Otherwise, if all literals of D have been replaced by false, then ψ is replaced by false.
   (d) Any disjunct in D in which y does not occur (e.g., literals of the form $x_i \,\pi\, x_j$, with π is in $\{=, \in\}$), can be moved outside of the scope of the quantifier $\forall y$ in ψ.

   Let ψ' be the formula obtained from ψ by performing the steps (3a)–(3d). Moreover, let D' be the homologous of D in ψ'.
   (e) As in step (3c), if all literals in D' are false, we can replace ψ' by false, if one of them is true, by true.
   (f) At this point all literals in ψ' are of the form $x_i \in y$ or $x_i \notin y$. We consider separately the two possible cases $y \in y$ and $y \notin y$:

   $y \in y$**:** This means that y is forced to be an atom and thus $y = \{y\}$ holds. Each literal of the form $x_i \in y$ can be replaced by false, since the premises impose $x_i \neq y$. Therefore, if there are no literals of the form $x_i \notin y$ then ψ' is simply equivalent to false; otherwise, (only literals of the form $x_i \in y$ are in ψ') ψ' is provably equivalent to true by using the law $(\mathbf{Y}_\in)$ of Lemma 3.2.

   $y \notin y$**:** ψ' is of the form:

$$\forall y \left(y \notin y \wedge \bigwedge_i (y \neq x_i \wedge y \notin x_i) \rightarrow \left(\bigvee_{j \in \mathcal{F}_1} x_j \in y \vee \bigvee_{j \in \mathcal{F}_2} x_j \notin y \right) \right)$$

   where $\mathcal{F}_1$ and $\mathcal{F}_2$ are sets of indices among $\{1, \ldots, n+k\}$. There are two cases:
      i. If one between $\mathcal{F}_1$ and $\mathcal{F}_2$ is empty, it is immediate to check that the formula is false.
      ii. Otherwise, replace ψ' by the disjunction $\bigvee_{i \in \mathcal{F}_1, j \in \mathcal{F}_2} x_i = x_j$.

Correctness of this rewriting step follows from $(\mathbf{Y}_{\notin})$ of Lemma 3.2. As an example of how $(\mathbf{Y}_{\notin})$ enters into play, consider the simple case of the formula:

$$\forall y\ (y \notin y \wedge (y \neq x_1 \wedge y \notin x_1) \wedge (y \neq x_2 \wedge y \notin x_2) \rightarrow x_1 \in y \vee x_2 \notin y)$$

Two cases:

- if $x_1 = x_2$ then the r.h.s. of the implication become equivalent to $x_1 \in y \vee x_1 \notin y$, hence to true.
- Otherwise, if $x_1 \neq x_2$, by $(\mathbf{Y}_{\notin})$ the whole formula is equivalent to false

3.2 Decision technique for sentences with restricted universal quantifiers

A satisfiability decision procedure for finite conjunctions of formulas

$$(\forall y_1 \in w_1) \cdots (\forall y_m \in w_m)\ p$$

where $m \geq 0$, $y_1, \ldots, y_m$ are variables distinct from one another and distinct from the variables w_j, and p is a propositional combination of atomic formulas of the two kinds $u = v$, $u \in v$ (u, v variables), is described in [6]. We will indicate below the changes needed to adapt that procedure to sets with self-singleton atoms. As in Sec. 3.1, we can also start with constants representing atoms in the input formula r and eliminate them at the very beginning.

From now on, we indicate by r the result of atom elimination and by $x_1, \ldots, x_n$ the distinct free variables in r. Moreover, we put $x_0 \equiv_{\text{Def}} \emptyset$.

Let us begin by considering the case of r quantifier-free. W.l.o.g., we can assume that r is a conjunction of literals $u = v$, $u \neq v$, $u \in v$, and $u \notin v$, where each u and v stands for either a variable or $\emptyset$. We determine the finest equivalence relation $\sim$ between the symbols $x_0, x_1, \ldots, x_n$ such that

- $x_i \sim x_j$ when $x_i = x_j$ belongs to r;
- $x_{j_0} \sim x_{i_0}$ when there is a membership cycle $x_{i_0} \in x_{i_1} \in \cdots \in x_{i_h} \in x_{i_0}$ in r, and $x_{j_0} \in x_{j_1}$, $x_{j_1} \in x_{j_2}$, $\ldots$, $x_{j_{k-1}} \in x_{j_k}$, and $x_{j_k} \in x_{i_0}$ belong to r, for some h and k.

The satisfiability test consists in checking that r contains no literals $u \neq v$ with $u \sim v$; no literals $u \in v$ with $v \sim \emptyset$; and no pair $u \in v$, $u' \notin v'$ of literals with $u \sim u'$ and $v \sim v'$. If the test is successful, then sets $\mathcal{A}(x)$ can be assigned to variables so as to satisfy r, as follows: a directed graph $\mathcal{G}$ whose nodes are the $\sim$-classes and whose edges represent membership relations directly drawn from r is built up. The graph is acyclic save for self-loops; nodes with self loops are put in a 1–1 correspondence $N \mapsto a^{(N)}$ with atoms;[3] finally, one defines $\mathcal{A}$ as follows:

[3] In particular, if a variable x originates from the elimination of an atom b, and N is the $\sim$-class of x, then we take $a^{(N)} = b$.

$$\mathcal{A}[N] = \begin{cases} a^{(N)} & \text{if } N \text{ has a self-edge,} \\ \{a^{(N)}, \emptyset\} & \text{if } N \text{ has no entering edge and } \emptyset \text{ is not in } N, \\ \{\mathcal{A}[M] \ : \ M \mathcal{G} N\} & \text{otherwise.} \end{cases}$$

Let us now move to the case when at least one conjunct of the input formula r is quantified. In order to decide r:

- We conjoin with r the condition

$$\bigwedge_{i=0}^{n-1} \bigwedge_{j=i+1}^{n} \left(x_i \neq x_j \rightarrow \bigvee_{h=1}^{n} (x_{n+h} \in x_i \leftrightarrow x_{n+h} \notin x_j) \right),$$

  where $x_{n+1}, \ldots, x_{2 \cdot n}$ are new variables (witnessing differences among the sets x_i, x_j).

- By proceeding inside-out, we replace each sub-formula of the form $(\forall y \in w)p$ by

$$\bigwedge_{i=0}^{2 \cdot n} (x_i \in w \rightarrow p[y \rightsquigarrow x_i]).$$

- When no quantifiers are left, the disjuncts of a DNF-formula tautologically equivalent to the resulting r are tested for satisfiability as explained above, until a test leads to success or no disjunct remains.

The completeness proof of the method goes along the guidelines of [6].

A technique for obtaining an exhaustive decomposition of any given conjunction r of the form

$$\bigwedge_{j=1}^{g} p_j \wedge \bigwedge_{i=g+1}^{g+n} (\forall y \in x_i) p_i$$

has been designed in [18] for the case of pure set/hyperset theories. Intuitively, this technique amounts to exhaustively generating all possible interpretations satisfying r. Results are presented (cf. [18, 6]) in order to ensure that this search can be safely restrained within a finite portion of the search space.

A further advance in our work should consist in adapting the original technique to deal with self-singletons. As result we would obtain an effective method to rewrite the formula r as an (equivalent) finite disjunction. Such a disjunctive decomposition would encode all possible manners one can satisfy r and, ideally, it should be represented as a finite family of substitutions. This would give way to the introduction of an algorithm able to solve the unification problem for sets with atoms, through a finite family of templates encompassing all possible solutions to the given instance of the problem. Algorithms of this kind are basic components of any theory-based deduction system.

4 Unification Algorithm for Sets with Atoms

In this section we develop a unification algorithm for the theory considered in Sec. 2. We assume the standard notions of first-order term, substitution, mgu, etc. [20]. In the context of axiomatic set theory, two terms s and t denoting sets are said to be unifiable if there is a substitution σ such that $s\sigma$ and $t\sigma$ represents the same set. In this case we say that σ is a solution of $s = t$. However, determining whether two terms s, t which denote sets are unifiable, amounts to establishing whether the existential closure of the formula $\forall y(y \in s \leftrightarrow y \in t)$ is valid. Under the assumptions that we are about to make, the latter sentence is easily brought to the $\exists^*\forall$-form considered above, so that it can be submitted to the decision algorithm already seen. Instead, in what follows, we directly face the unification problem by developing a goal-driven unification algorithm.

To denote finite sets we fix a first-order signature $\Sigma = \{\emptyset, \{\cdot \mid \cdot\}, \mathsf{c}_1, \mathsf{c}_2, \dots\}$, where each c_i is assumed to fulfill $\mathsf{c}_i = \{\mathsf{c}_i\}$, i.e., to be self-singleton. If x and y are terms denoting sets, then $\{x \mid y\}$ denotes the set whose elements are 'x' plus those of the set y (cf. the axiom (**W**)). As a notational convenience, we denote the term $\{t_1 \mid \{t_2 \mid \cdots \{t_n \mid t\}\}\}$ by $\{t_1, t_2, \dots, t_n \mid t\}$, and simply by $\{t_1, t_2, \dots, t_n\}$ when t is $\emptyset$. When $n = 0$, $\{t_1, t_2, \dots, t_n \mid t\}$ is simply t. Moreover, t is said to be the tail of the term $\{t_1, t_2, \dots, t_n \mid t\}$, provided that $\{\cdot \mid \cdot\}$ is not the main functor of t. In the algorithm, uppercase letters represent variables; N stands for a newly generated variable; r, s, t, possibly subscripted, stand for generic terms; and c, d represent constant symbols drawn from among the c_is. With $t[X]$ we denote a term having X as subterm.

We will make use of the monadic predicate symbol ur to state that certain terms designate atoms. Thus, $\mathsf{ur}(t)$ implies $t = \{t\}$. For instance, if X is a variable, $\mathsf{ur}(X)$ is satisfiable; $\mathsf{ur}(\{c, d\})$, $\mathsf{ur}(\emptyset)$ are false formulas.

A substitution σ is a *solution* of a system E if all the equations of E are simultaneously satisfied by σ. A system E is said to be in *solved form* if it consists only of equations of the form $X = t$, with X not occurring in t nor elsewhere in E. A system in solved form uniquely identifies a substitution which is trivially a solution of itself.

The unification algorithm Unify described in Fig. 1 consists of mutually exclusive rewriting rules that are non-deterministically applied to E until E reaches a solved or fail is obtained. Briefly,

- Action (1) removes obvious identities.
- Action (2) is but a rearrangement aimed at making the formulation of the other rewriting rules simpler.
- Action (3) performs occur-check, to ensure well-foundedness. The presence of atoms allows many more solution possibilities than the standard (well-founded) case. For instance, $X = \{X, \{X\}\}$ admits $X = c$ as a solution for any atom c, since $\{c, \{c\}\} = \{c, c\} = \{c\} = c$. Action (3) also expands some equations involving atom variables.
- Action (4) applies substitution.

(1)	$t = t \wedge E$	$\mapsto$	E
(2.1)	$t = X \wedge E$, t is not a variable	$\mapsto$	$X = t \wedge E$
(2.2)	$t = c \wedge E$, t is neither a variable nor a constant d	$\mapsto$	$c = t \wedge E$
(3.1)	$X = \{t_0, \ldots, t_n \mid X\} \wedge E$	$\mapsto$	$X = \{t_0, \ldots, t_n \mid N\} \wedge E$
(3.2)	$X = \{t_0, \ldots, t_i[X], \ldots, t_n \mid t\} \wedge E$, t is a variable Y, $X \not\equiv Y$, or $\emptyset$	$\mapsto$	$\bigwedge_{i=0}^{n} X = t_i \wedge E \wedge$ $(i) Y = \emptyset \vee$ $(ii) Y = X$ Add $\mathsf{ur}(X)$ to U
(3.3)	$X = \{t_0, \ldots, t_n\} \wedge E$, $\mathsf{ur}(X) \in U$	$\mapsto$	$\bigwedge_{i=0}^{n} X = t_i \wedge E$
(3.4)	$X = \{t_0, \ldots, t_n \mid c\} \wedge E$, $\mathsf{ur}(X) \in U$	$\mapsto$	$\bigwedge_{i=0}^{n} X = t_i \wedge X = c \wedge E$
(4.1)	$X = Y \wedge E$, X occurs in $E, \mathsf{ur}(X), X \not\equiv Y$	$\mapsto$	$E[X/Y] \wedge X = Y$ Add $\mathsf{ur}(Y)$ to U
(4.2)	$X = c \wedge E$, X occurs in E,	$\mapsto$	$E[X/c] \wedge X = c$
(4.3)	$X = t \wedge E$, X occurs in E, X not in t and not $\mathsf{ur}(X)$	$\mapsto$	$E[X/t] \wedge X = t$
(5.1)	$c = \{t_1, \ldots, t_n \mid d\} \wedge E$, $c \not\equiv d, n \geq 0$	$\mapsto$	fail
(5.2)	$c = \emptyset \wedge E$	$\mapsto$	fail
(5.3)	$\emptyset = \{s \mid t\} \wedge E$	$\mapsto$	fail
(5.4)	$\{s \mid t\} = \emptyset \wedge E$	$\mapsto$	fail
(6.1)	$c = \{t_0, \ldots, t_n \mid Y\} \wedge E$	$\mapsto$	$\bigwedge_{i=0}^{n} c = t_i \wedge E \wedge$ $(i) Y = \emptyset \vee$ $(ii) Y = c$
	.....		
(6.2)	$c = \{t_0, \ldots, t_n \mid t\} \wedge E$, t is c or $\emptyset$	$\mapsto$	$\bigwedge_{i=0}^{n} c = t_i \wedge E$
(7.1)	$\{t_0, \ldots, t_m \mid X\} = \{t'_0, \ldots, t'_n \mid X\} \wedge E$	$\mapsto$	select arbitrarily i in $\{0, \ldots, n\}$; choose one among: (i) $\{t_1, \ldots, t_m \mid X\} = \{t'_0, \ldots, t'_{i-1}, t'_{i+1}, \ldots, t'_n \mid X\} \wedge t_0 = t'_i \wedge E$ (ii) $\{t_0, \ldots, t_m \mid X\} = \{t'_0, \ldots, t'_{i-1}, t'_{i+1}, \ldots, t'_n \mid X\} \wedge t_0 = t'_i \wedge E$ (iii) $\{t_1, \ldots, t_m \mid X\} = \{t'_0, \ldots, t'_n \mid X\} \wedge t_0 = t'_i \wedge E$ (iv) $\{t_1, \ldots, t_m \mid N\} = \{t'_0, \ldots, t'_n \mid N\} \wedge X = \{t_0 \mid N\} \wedge E$
	.....		
(7.2)	$\{t \mid s\} = \{t' \mid s'\} \wedge E$, $\mathsf{tail}(s)$ and $\mathsf{tail}(s')$ are not the same variable	$\mapsto$	(i) $s = s' \wedge t = t' \wedge E$ (ii) $\{t \mid s\} = s' \wedge t = t' \wedge E$ (iii) $s = \{t' \mid s'\} \wedge t = t' \wedge E$ (iv) $s = \{t' \mid N\} \wedge \{t \mid N\} = s' \wedge E$

Figure 1: Set unification rewriting rules

- Action (5) leads to failure when the empty set is compared to non-empty entities, or when two distinct atoms are compared.
- Action (6) deals with atoms. Notice that atoms behave like variables X for which $\mathsf{ur}(X)$ is assumed to hold.
- Action (7) is the non-deterministic treatment of set-set equalities (cf. [8, 11]).

We assume that there is a set U (the *store*) where facts of the form $\mathsf{ur}(X)$ are stored. At the beginning of the execution we assume that U is empty. The algorithm checks if X is stored in U before applying a substitution to it. Moreover, at the end of the process, we will use the store U to compute/filter the solutions. Precisely, we say that σ is a substitution *consistent* w.r.t. the store U if $\sigma(X) = \mathsf{c}_i$ (an atom) for each variable X s.t. $\mathsf{ur}(X)$ is in U.

Lemma 4.1 *Let E be a system and $E_1, \ldots, E_k$ be the equation systems non-deterministically obtained after $h \geq 0$ steps of* Unify*, and $U_1, \ldots, U_k$ be the corresponding stores. Then,*

1. *if σ is a solution of E_i, consistent w.r.t. the store U_i, then it is also a solution of E.*
2. *if σ is a solution of E, then, there exists i, $1 \leq i \leq k$, such that σ can be expanded to the variable of E_i that are not in E so that it is a solution of E_i consistent w.r.t. U_i.*

Proof: (Sketch) Correctness of cases (7.1), and (7.2) follows from [11]. Cases (1) and (2) hold trivially. Let us analyze the other cases:

(3) (3.1) For one side, if σ is a solution of $X = \{t_0, \ldots, t_n \,|\, X\}$, then $\sigma' = \sigma \cup \{N/\sigma(X)\}$ is clearly a solution of $X = \{t_0, \ldots, t_n \,|\, N\}$. For the other side, the result follows from the fact that $\{t_0, \ldots, t_n, t_0, \ldots, t_n \,|\, \sigma(N)\} = \{t_0, \ldots, t_n \,|\, \sigma(N)\}$, for any value of $\sigma(N)$.

(3.2) The unique case in which X is admitted to be a subterm of itself is when it is an atom. The rewriting rule reflects this fact. Here the constraint $\mathsf{ur}(X)$ is added to the store U.

(3.3), (3.4) The constraint $\mathsf{ur}(X)$ forces X to be an atom. The rewriting rule is accordance with this fact.

(4) This rule consists in substitution application. The checking and updating of the store allow to guarantee correctness.

(5) These are all failure steps. For instance, An atom c cannot be equivalent to $\{\ldots \,|\, d\}$ or to d, for any other atom d, by **(E)**. Similar considerations apply to the other three cases.

(6) Similarly to actions (3.3)/(3.4), if c is an atom any solution of it must satisfy the equations on the r.h.s.

□

The algorithm is fully non-deterministic and it is easy to find a sequence of actions leading to non-termination. However, as shown in [8, 11] for the unification algorithm presented there, a simple deterministic strategy (e.g., by adopting a stack strategy for the selection rules of equations introduced by action (7)) can be devised to ensure termination.

Lemma 4.2 *The algorithm* Unify *can be implemented so as to ensure termination.*

Proof: The termination proof can be carried on by following the lines of the termination proof of the set unification algorithm described in [11]. □

Theorem 4.3 *Given a system E,* Unify(E) *always terminates either returning* fail *or a system in solved form. If it terminates with* fail *in all of the non-deterministic computation, then the system is unsatisfiable. Otherwise, the set of its solutions is represented by the systems $E_1, \ldots, E_k$ returned, together with their stores.*

Proof: Immediate by the Lemmas 4.1 and 4.2. □

Example 4.4 Consider the system

$$E \equiv X = \{X, Y\}, Y = \{a\}, Z = \{Z\}$$

One possible computation yields

$$E_1 \equiv X = a, Y = a$$

with the store

$$U = \mathsf{ur}(X), \mathsf{ur}(Y), \mathsf{ur}(Z)$$

Any substitution that extends the substitution induced by E_1 by mapping Z to a constant (an atom) is consistent w.r.t. the store and is also a solution of E.

5 Conclusions

Various approaches can be adopted in axiomatizing hybrid set/hyperset theories (i.e., set theories involving atoms). Consequently, *a priori*, the unification problem—as well as the decision problem—should be faced in different manners. On the one hand, as mentioned, reduction to the pure case constitutes the simpler (naïve) solution. On the other hand, a treatment involving the notion of colored set seems to be more viable, since it may offer greater practical advantage in real implementations. For instance, multisets with colors are shown (cf. [9]) to be suitable to implement P-systems (the abstract model behind membrane computing [19]).

In this paper we introduced an alternative treatment which models atoms as self-singletons, following a proposal of Quine. Both the decision problem, for $\exists^*\forall$-sentences, and the unification problem have been solved by slightly modifying techniques previously designed for the case of pure (or colored) sets/hypersets. Hence, we shown that each of the three different approaches to hybrid sets can be treated by exploiting essentially the same general techniques.

References

[1] P. Aczel. *Non-Well-Founded Sets.* CSLI Lecture Notes, vol. 14, Stanford, 1988.

[2] J. Barwise and L. Moss. *Vicious Circles.* CSLI Publications, vol. 60, Stanford, 1996.

[3] D. Bellè and F. Parlamento. Decidability of the $\forall^*\exists^*$-class in the membership theory NWL. *Proc. of Gödel'96-Logical Foundations of Mathematics, Computer Science and Physics-Kurt Gödel's Legacy.* Brno, Czech Republic. Lecture Notes in Logic, vol. 6, Springer-Verlag, 1996.

[4] D. Cantone, A. Ferro, and E. G. Omodeo. *Computable Set Theory.* International series of monographs on computer science. Clarendon Press, 1989.

[5] D. Cantone, S. Ghelfo, and E. G. Omodeo. The automation of syllogistic I. Syllogistic normal forms. *J. of Symbolic Computation*, 6(1):83–98, 1988.

[6] D. Cantone, E. G. Omodeo, and A. Policriti. *Set Theory for Computing.* Monographs in Computer Science, Springer-Verlag, 2001.

[7] A. Dovier, E. G. Omodeo, and A. Policriti. Solvable set/hyperset contexts: II. A goal-driven unification algorithm for the blended case. *Applicable algebra in engineering, communication and computing*, 9(4):293-332, 1999.

[8] A. Dovier, E. G. Omodeo, E. Pontelli, and G.-F. Rossi. {log}: A Language for Programming in Logic with Finite Sets. *J. of Logic Programming*, 28(1):1–44, 1996.

[9] A. Dovier, C. Piazza, and G.-F. Rossi. Multiset rewriting by multiset constraint solving. *Romanian Journal of Information Science and Technology*, 4(1–2), 2001.

[10] A. Dovier, A. Policriti, and G.-F. Rossi. A uniform axiomatic view of lists, multisets, and sets, and the relevant unification algorithms. *Fundamenta Informaticae*, 36(2/3):201–234, 1998.

[11] A. Dovier, C. Piazza, E. Pontelli, and G.-F. Rossi. Sets and constraint logic programming. *ACM Transaction on Programming Language and Systems*, 22(5):861–931, 2000.

[12] H. B. Enderton. *Elements of set theory.* Academic Press, New York, 1977.

[13] D. Gogol. The $\forall_n\exists$-completeness of Zermelo-Fraenkel set theory. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, pages 1–2, 1978.

[14] T. J. Jech. *Set theory.* Academic Press, New York, 1978.

[15] K. Kunen. *Set Theory. An Introduction to Independence Proofs.* Studies in Logic. North Holland, Amsterdam, 1980.

[16] E. G. Omodeo, F. Parlamento, and A. Policriti. Decidability of $\exists^*\forall$-sentences in Membership Theories. *Mathematical Logic Quarterly*, 42(1):41–58, 1996.

[17] E. G. Omodeo, F. Parlamento, and A. Policriti. A derived algorithm for evaluating ε-expressions over abstract sets. *J. of Symbolic Computation*, 15:673–704, 1993.

[18] E. G. Omodeo and A. Policriti. Solvable set/hyperset contexts: I. Some decision procedures for the pure, finite case. *Comm. Pure Appl. Math.*, 48(9-10):1123–1155, 1995. Special issue in honor of J. T. Schwartz.

[19] G. Pāun. Computing with Membranes. *J. of Computer and System Science*, 61(1):108–143, 2000.

[20] J. Siekmann. Unification Theory. *J. of Symbolic Computation* 7(3,4):207–274, 1989.

[21] W. V. Quine. *Set theory and its logic.* The Belknap Press of Harvard University Press, Cambridge, Massachussetts, revised edition, 3^{rd} printing, 1971.