Two Crucial Cubic-Time Components of Polynomial-Maximal Decidable Boolean Languages*

Domenico Cantone[0000−0002−1306−1166]1, Pietro Maugeri[0000−0002−0662−2885]1, and Eugenio G. Omodeo[0000−0003−3917−1942]2

[1] Dept. of Mathematics and Computer Science, University of Catania, Italy
{domenico.cantone,pietro.maugeri}@unict.it
[2] Dept. of Mathematics and Earth Sciences, University of Trieste, Italy
eomodeo@units.it

Abstract. We continue our investigation aimed at spotting small fragments of Set Theory (in this paper, sublanguages of Boolean Set Theory) that might be of use in automated proof-checkers based on the set-theoretic formalism. Here we propose a method that leads to a cubic-time satisfiability decision test for the language involving, besides variables intended to range over the von Neumann set-universe, the Boolean operator $\cup$ and the logical relators $=$ and $\neq$. It can be seen that the dual language involving the Boolean operator $\cap$ and, again, the relators $=$ and $\neq$, also admits a decidable cubic-time satisfiability test; noticeably, the same algorithm can be used for both languages. Suitable pre-processing can reduce richer Boolean languages to the said two fragments, so that the same cubic satisfiability test can be used to treat the relators $\subseteq, \not\subseteq$ and the predicates '$\blacksquare = \varnothing$' and '$\textsc{Disj}(\blacksquare, \blacksquare)$', meaning 'the argument is empty' and 'the arguments are disjoint sets', along with their opposites '$\blacksquare \neq \varnothing$' and '$\neg\textsc{Disj}(\blacksquare, \blacksquare)$'. Those richer languages are 'polynomial maximal', in the sense that all languages strictly containing them have an NP-hard satisfiability problem.

Keywords: Satisfiability problem · Computable set theory · Boolean set theory · Expressibility · NP-completeness · Proof verification.

1 Introduction

The field named *Computable Set Theory* [5] pursued, with long-standing efforts, languages reconciling ease of symbolic management with high expressive power, so that an armory of reasoning tools—foremost, satisfiability testers for unquantified fragments of theories concerning sets and classes—could shape a friendly proof-development environment.

Long before the envisaged proof-verifier ÆtnaNova [17] came into being, a foundational quest aimed at carefully drawing the frontier between the decidable and the undecidable in set theory sparked interest in the field [15]. The language called Multi-Level Syllogistic and some extensions of it [11], whose satisfiability problems were shown to be NP-complete in [7], foreran that quest. In recent years, we undertook a similar investigation aimed at spotting 'small' fragments of set theory endowed with low-degree polynomial-time satisfiability tests, which hence promise to be useful in set-based proof technology. In [4] we reported about the complexity taxonomy of all the sublanguages of the theory called Boolean Set Theory ($\mathbb{BST}$ for short); here we treat in detail two core fragments or theories, $\mathsf{BST}(\cup, =, \neq)$ and $\mathsf{BST}(\cap, =, \neq)$, of that taxonomy and report about cubic-time tests for their satisfiability problems. Specifically, we will study two languages, consisting of all conjunctions of literals of the two forms

$$(=) \quad x_1 \star \cdots \star x_h = y_1 \star \cdots \star y_k$$
$$(\neq) \quad u_1 \star \cdots \star u_m \neq v_1 \star \cdots \star v_p,$$

where $h, k, m, p \geqslant 1$ holds, $\star$ stands for a fixed dyadic set operation, and x's, y's, u's, v's stand for set variables.

By instantiating $\star$ as $\cup$, we obtain the fragment $\mathsf{BST}(\cup, =, \neq)$; by instantiating it as $\cap$, we obtain the fragment $\mathsf{BST}(\cap, =, \neq)$.

The paper is organized as follows. In Section 2 we introduce an equivalence relation $\sim_\varphi$ between non-null subsets of the set of variables occurring in a formula φ of either fragment and elicit some of its key properties. Then, in Section 3, we present our main decidability results together with a cubic-time satisfiability test, stressing the commonalities between the two decidable theories. In Section4, we report about extending the presented results to richer languages within $\mathbb{BST}$. Then we draw conclusions.

2 The equivalence relation $\sim_\varphi$ and the $\sim_\varphi$-closure operator

Any given formula φ of $\mathsf{BST}(\star, =, \neq)$ can be represented by two sets:

$$\Phi_\varphi^{\oplus} := \Big\{ \{\{x_1, \cdots, x_h\}, \{y_1, \cdots, y_k\}\} \mid x_1 \star \cdots \star x_h = y_1 \star \cdots \star y_k \text{ is in } \varphi \Big\},$$
$$\Phi_\varphi^{\ominus} := \Big\{ \{\{u_1, \cdots, u_m\}, \{v_1, \cdots, v_p\}\} \mid u_1 \star \cdots \star u_m \neq v_1 \star \cdots \star v_p \text{ is in } \varphi \Big\}.$$

A central role will be played by the following relation $\sim_\varphi$ on $\mathscr{P}^+(\mathit{Vars}(\varphi))$, where $\mathit{Vars}(\varphi)$ is defined to be the collection of all variables occurring in φ, and $\mathscr{P}^+(S) := \mathscr{P}(S) \setminus \{\emptyset\}$ for any set S. Specifically, $\sim_\varphi$ is the intersection—hence the inclusion minimal—of all *equivalence relations* $\sim$ on $\mathscr{P}^+(\mathit{Vars}(\varphi))$ that satisfy the following closure conditions:

(Cl1) $\quad \Phi_\varphi^{\oplus} \subseteq \sim$,

(Cl2) if $A \sim B$, then $A \cup C \sim B \cup C$, for all $A, B, C \in \mathscr{P}^+(\mathit{Vars}(\varphi))$.

We will in fact prove, for any φ in $\mathsf{BST}(\star, =, \neq)$, that φ is satisfiable if and only if $\{u_1, \ldots, u_m\} \not\sim_\varphi \{v_1, \ldots, v_p\}$ holds for every literal of the form $(\neq)$ in φ. We will exploit a useful *closure operator* to test conditions of the form $Z_1 \not\sim_\varphi Z_2$. Specifically, we will prove that the set

$$\overline{Z} \coloneqq \bigcup\{W \mid W \sim_\varphi Z\},$$

called the $\sim_\varphi$*-closure of* Z (or, more simply, the *closure of* Z, when the relation $\sim_\varphi$ is understood), is the largest set $\sim_\varphi$-equivalent to Z and then will provide a quadratic algorithm for computing it. Accordingly, a set Z such that $\overline{Z} = Z$ will be said to be $\sim_\varphi$-closed or just closed.

Lemma 1. *Let $Z \in \mathscr{P}^+(\mathit{Vars}(\varphi))$. Then the closure $\overline{Z}$ of Z, namely the set $\bigcup\{W \mid W \sim_\varphi Z\}$, is the largest subset of $\mathit{Vars}(\varphi)$ that is $\sim_\varphi$-equivalent to Z.*

Proof. Let W_1, W_2 be such that $W_1 \sim_\varphi Z$ and $W_2 \sim_\varphi Z$. By applying (Cl2) twice, we have: $W_1 \cup W_2 \sim_\varphi Z \cup W_2 \sim_\varphi Z$. In view of the finiteness of $\{W \mid W \sim_\varphi Z\}$, by induction it follows that

$$\Big(\bigcup\{W \mid W \sim_\varphi Z\}\Big) \sim_\varphi Z\,.$$

In addition, for every W' such that $W' \sim_\varphi Z$, we plainly have $W' \subseteq \bigcup\{W \mid W \sim_\varphi Z\}$. This yields that $\bigcup\{W \mid W \sim_\varphi Z\}$ is the largest set in $\mathscr{P}^+(\mathit{Vars}(\varphi))$ that is $\sim_\varphi$-equivalent to Z. □

The equivalence relation $\sim_\varphi$ captures which equalities among terms must be true in every model, if any, of a given formula φ: this is shown in the next lemma.

A *set assignment* M is any function sending a collection $\mathsf{dom}(M)$ of set variables into the von Neumann universe of all sets $\mathcal{V} \coloneqq \bigcup_{\alpha \in On} \mathcal{V}_\alpha$, resulting from the union of the levels $\mathcal{V}_\alpha \coloneqq \bigcup_{\beta<\alpha} \mathscr{P}(\mathcal{V}_\beta)$, with α ranging over the class On of all ordinal numbers, where $\mathscr{P}(\blacksquare)$ is the powerset operator.

Natural designation rules attach recursively a value to every term τ and to every literal of $\mathsf{BST}(\star, =, \neq)$ such that $\mathit{Vars}(\tau) \subseteq \mathsf{dom}(M)$, for any set assignment M:

$$\begin{aligned} M(\sigma \star \tau) &\coloneqq M\,\sigma \star M\,\tau\,; \\ M(\sigma = \tau) &\coloneqq \begin{cases} \mathsf{true} & \text{if } M\,\sigma = M\,\tau \\ \mathsf{false} & \text{otherwise;} \end{cases} \\ M(\sigma \neq \tau) &\coloneqq \neg M(\sigma = \tau)\,. \end{aligned}$$

Then we put, for conjunctions of literals ℓ_i:

$$M(\ell_1 \wedge \cdots \wedge \ell_k) \coloneqq M\ell_1 \wedge \cdots \wedge M\ell_k$$

when $\mathit{Vars}(\ell_i) \subseteq \mathsf{dom}(M)$, for $i = 1, \ldots, k$.

Given a conjunction φ and a set assignment M such that $\mathit{Vars}(\varphi) \subseteq \mathsf{dom}(M)$, we say that M *satisfies* φ, and write $M \models \varphi$, if $M\varphi = \mathsf{true}$. When M satisfies φ, we also say that M is a *model* of φ.

A conjunction φ is said to be *satisfiable* if it has some model, else *unsatisfiable*.

For convenience, in what follows we will use the shortening notations

$$M\{x_1, \ldots, x_k\} \coloneqq \{M\,x_1, \ldots, M\,x_k\},$$
$$\bigstar\{x_1, \ldots, x_k\} \coloneqq x_1 \star \cdots \star x_k,$$
$$\bigstar\{s_1, \ldots, s_k\} \coloneqq s_1 \star \cdots \star s_k,$$

where M represents a set assignment, $x_1, \ldots, x_k$ and $s_1, \ldots, s_k$ denote variables and sets, respectively, and the $\star$'s stand for alike symbols/operations '$\cup$' or '$\cap$'.

Lemma 2. *Let φ be any formula in $\mathsf{BST}(\star, =, \neq)$, and let M be any set assignment over $\mathit{Vars}(\varphi)$ satisfying φ. Then*

$$Z_1 \sim_\varphi Z_2 \implies \bigstar MZ_1 = \bigstar MZ_2,$$

for all $Z_1, Z_2 \in \mathscr{P}^+(\mathit{Vars}(\varphi))$.

Proof. In view of the fact that $\sim_\varphi$ is inclusion-minimal, it is sufficient to prove that the equivalence relation $\sim_M$ over $\mathscr{P}^+(\mathit{Vars}(\varphi))$ defined by

$$Z_1 \sim_M Z_2 \overset{\text{Def.}}{\iff} \bigstar MZ_1 = \bigstar MZ_2$$

satisfies the closure conditions (Cl1) and (Cl2).

Concerning (Cl1), if $\{L, R\} \in \Phi^{\oplus}_{\varphi}$ then $\bigstar ML = \bigstar MR$, so $L \sim_M R$ holds, proving $\Phi^{\oplus}_{\varphi} \subseteq \sim_M$.

As for (Cl2), let $A \sim_M B$ and $C \subseteq \mathit{Vars}(\varphi)$. Then $\bigstar MA = \bigstar MB$, and therefore

$$\bigstar M(A \cup C) = (\bigstar MA) \star (\bigstar MC) = (\bigstar MB) \star (\bigstar MC) = \bigstar M(B \cup C),$$

from which $A \cup C \sim_M B \cup C$ follows. □

We next prove the following key property, which will be used in the correctness proof of our fast algorithm presented in Section 3.1 for computing $\sim_\varphi$-closures.

Lemma 3. *Given a $\mathsf{BST}(\star, =, \neq)$-formula φ, let $Z \in \mathscr{P}^+(\mathit{Vars}(\varphi))$ be such that*

$$L \subseteq Z \iff R \subseteq Z, \qquad \text{for every } \{L, R\} \in \Phi^{\oplus}_{\varphi}. \tag{1}$$

Then, for all $W_1, W_2 \in \mathscr{P}^+(\mathit{Vars}(\varphi))$ such that $W_1 \sim_\varphi W_2$, we have

$$W_1 \subseteq Z \iff W_2 \subseteq Z. \tag{2}$$

Proof. Let φ and Z be as in the hypothesis. In view of the $\subseteq$-minimality of $\sim_\varphi$, it suffices to prove that the equivalence relation over $\mathscr{P}^+(\mathit{Vars}(\varphi))$ defined by

$$W_1 \sim_Z W_2 \quad \stackrel{\text{Def.}}{\Longleftrightarrow} \quad \big(W_1 \subseteq Z \iff W_2 \subseteq Z\big) \tag{3}$$

satisfies the closure conditions (Cl1) and (Cl2).

As for (Cl1), just from the hypothesis it follows that $L \sim_Z R$, for every $\{L, R\} \in \Phi_\varphi^\oplus$. Concerning (Cl2), let $A \sim_Z B$ and $C \subseteq \mathit{Vars}(\varphi)$, and assume that $A \cup C \subseteq Z$. Then, $A \subseteq Z$ and $C \subseteq Z$, so that by (3) we have $B \cup C \subseteq Z$. Symmetrically, it can be shown that $B \cup C \subseteq Z$ implies $A \cup C \subseteq Z$. Hence,

$$A \cup C \subseteq Z \iff B \cup C \subseteq Z$$

holds and, by (3), we readily have $A \cup C \sim_Z B \cup C$. The arbitrariness of A, B, and C yields that even the closure condition (Cl2) holds for $\sim_Z$.

Finally, from the $\subseteq$-minimality of $\sim_\varphi$, we have $\sim_\varphi \subseteq \sim_Z$. Therefore, if $W_1 \sim_\varphi W_2$, then $W_1 \sim_Z W_2$, which by (3) implies $W_1 \subseteq Z \iff W_2 \subseteq Z$. □

Further useful properties of the $\sim_\varphi$-closure operator and of the equivalence relation $\sim_\varphi$ are reported in the following lemma.

Lemma 4. *Let $Z, Z_1, Z_2 \in \mathscr{P}^+(\mathit{Vars}(\varphi))$. Then*

(a) $Z \subseteq \overline{Z}$ *and* $Z \sim_\varphi \overline{Z}$;
(b) $\overline{Z} = \overline{\overline{Z}}$;
(c) *if* $Z_1 \sim_\varphi \overline{Z}_2$, *then* $Z_1 \subseteq \overline{Z}_2$;
(d) $Z_1 \sim_\varphi Z_2$ *if and only if* $\overline{Z}_1 = \overline{Z}_2$;
(e) *if* $Z_1 \subseteq Z_2$, *then* $\overline{Z}_1 \subseteq \overline{Z}_2$;
(f) $Z_1 \subseteq \overline{Z}_2$ *if and only if* $\overline{Z}_1 \subseteq \overline{Z}_2$;
(g) *if* $Z_1 \subseteq Z$ *or* $Z_2 \subseteq Z$ *holds and* $Z_1 \sim_\varphi Z_2$, *then* $Z \sim_\varphi Z \cup Z_1 \cup Z_2$.

Proof. Property (a) follows directly from Lemma 1.

Concerning (b): the transitivity of $\sim_\varphi$ yields $\overline{\overline{Z}} \sim_\varphi Z$ so that, by the definition of $\overline{Z}$, $\overline{\overline{Z}} \subseteq \overline{Z}$ holds. By (a) we have $\overline{Z} \subseteq \overline{\overline{Z}}$, therefore $\overline{Z} = \overline{\overline{Z}}$.

As for (c), if $Z_1 \sim_\varphi \overline{Z}_2$, then $Z_1 \subseteq \overline{\overline{Z}}_2 = \overline{Z}_2$.

Concerning (d), if $Z_1 \sim_\varphi Z_2$, then the transitivity of $\sim_\varphi$ yields $\overline{Z}_1 \sim_\varphi \overline{Z}_2$. Thus, by (c), we get $\overline{Z}_1 = \overline{Z}_2$. Conversely if $\overline{Z}_1 = \overline{Z}_2$, then $\overline{Z}_1 \sim_\varphi \overline{Z}_2$, thus by transitivity $Z_1 \sim_\varphi Z_2$ holds.

Regarding (e), let $Z_1 \subseteq Z_2$. Since by (a) $Z_2 \subseteq \overline{Z}_2$ and $Z_1 \sim_\varphi \overline{Z}_1$ hold, by (Cl2) we have

$$\overline{Z}_2 = Z_1 \cup (\overline{Z}_2 \setminus Z_1) \sim_\varphi \overline{Z}_1 \cup (\overline{Z}_2 \setminus Z_1) = \overline{Z}_1 \cup \overline{Z}_2\,.$$

Thus, by (c), we get the inclusion $\overline{Z}_1 \cup \overline{Z}_2 \subseteq \overline{Z}_2$, and therefore $\overline{Z}_1 \subseteq \overline{Z}_2$.

Concerning (f), if $Z_1 \subseteq \overline{Z}_2$, then by (e) and (b) we have $\overline{Z}_1 \subseteq \overline{\overline{Z}}_2 = \overline{Z}_2$. Conversely, if $\overline{Z}_1 \subseteq \overline{Z}_2$, then by (a) we have $Z_1 \subseteq \overline{Z}_1 \subseteq \overline{Z}_2$.

Finally, as for (g), suppose that we have $Z_1 \sim_\varphi Z_2$ and either $Z_1 \subseteq Z$ or $Z_2 \subseteq Z$ holds. By (Cl2), we have $Z \cup Z_1 \sim_\varphi Z \cup Z_2$. But $\{Z \cup Z_1, Z \cup Z_2\} = \{Z, Z \cup Z_1 \cup Z_2\}$, hence $Z \sim_\varphi Z \cup Z_1 \cup Z_2$ follows. □

3 Satisfiability in $\mathsf{BST}(\cup, =, \neq)$ and in $\mathsf{BST}(\cap, =, \neq)$

Below we will present a necessary condition that also suffices to ensure that a given formula in either $\mathsf{BST}(\cup, =, \neq)$ or $\mathsf{BST}(\cap, =, \neq)$ is satisfiable. Noticeably, this condition is essentially the same for both languages, so that the same algorithm can be used to test formulae of either language for satisfiability.

Theorem 1. *Let φ be a $\mathsf{BST}(\cup, =, \neq)$-formula. Then φ is satisfiable if and only if $L \not\sim_\varphi R$ (namely $\overline{L} \neq \overline{R}$) holds for every literal of the form $\bigcup L \neq \bigcup R$ in φ.*

Proof. (Necessity.) Let M be a model for φ. By way of contradiction, assume that there exists a literal $\bigcup L \neq \bigcup R$ such that $L \sim_\varphi R$. Then by Lemma 2 we would have $\bigcup ML = \bigcup MR$, a contradiction. Therefore for all literals $\bigcup L \neq \bigcup R$ we must have $L \not\sim_\varphi R$, completing the necessity part of the proof.

(Sufficiency.) Next, let us assume that, for each $\{L, R\} \in \Phi^{\ominus}_{\varphi}$, we have $L \not\sim_\varphi R$. We will construct a set assignment M that satisfies φ.

Let us assign a nonempty set $b_{\overline{V}}$ to each $\overline{V}$ such that $V \in \bigcup \Phi^{\ominus}_{\varphi}$ in such a way that the $b_{\overline{V}}$'s are pairwise distinct.[3] Then we define the set assignment M over $\mathit{Vars}(\varphi)$ by putting, for each $x \in \mathit{Vars}(\varphi)$,

$$Mx \coloneqq \{b_{\overline{V}} \mid x \notin \overline{V} \text{ and } V \in \bigcup \Phi^{\ominus}_{\varphi}\},$$

so that we have

$$\bigcup MU \coloneqq \{b_{\overline{V}} \mid U \not\subseteq \overline{V} \text{ and } V \in \bigcup \Phi^{\ominus}_{\varphi}\}, \tag{4}$$

for every $U \subseteq \mathit{Vars}(\varphi)$.

We first show that $\bigcup ML = \bigcup MR$ holds whenever $\{L, R\} \in \Phi^{\oplus}_{\varphi}$. Thus, let $\{L, R\} \in \Phi^{\oplus}_{\varphi}$ and let $b_{\overline{V}} \in \bigcup ML$, for some $V \in \Phi^{\ominus}_{\varphi}$ such that $L \not\subseteq \overline{V}$. Then $\overline{L} \not\subseteq \overline{V}$, by Lemma 4(f). Since $\{L, R\} \in \Phi^{\oplus}_{\varphi}$, then by (Cl1) we have $L \sim_\varphi R$, so that, by (d) and (f) of Lemma 4, $R \not\subseteq \overline{V}$ follows. Hence $b_{\overline{V}} \in \bigcup MR$, and by the arbitrariness of $b_{\overline{V}}$ we have $\bigcup ML \subseteq \bigcup MR$.

Analogously one can prove $\bigcup MR \subseteq \bigcup ML$, so $\bigcup ML = \bigcup MR$ holds, as we intended to show.

Next we prove that $\bigcup ML \neq \bigcup MR$ holds, whenever $\{L, R\} \in \Phi^{\ominus}_{\varphi}$. Thus, let $\{L, R\} \in \Phi^{\ominus}_{\varphi}$, so that by our assumption we have $L \not\sim_\varphi R$. Hence, by Lemma 4(d), $\overline{L} \neq \overline{R}$. W.l.o.g., let us assume that $\overline{L} \not\subseteq \overline{R}$. Since $R \subseteq \overline{R}$ (by Lemma 4(a)) and plainly $R \in \bigcup \Phi^{\ominus}_{\varphi}$, then $b_{\overline{R}} \notin \bigcup MR$, by (4). On the other hand, by Lemma 4(f), $L \not\subseteq \overline{R}$, hence $b_{\overline{R}} \in \bigcup ML$, again by (4), and therefore $\bigcup ML \neq \bigcup MR$, concluding the proof of the theorem. □

In a dual manner, one can also prove the following result:

Theorem 2. *Let φ be a $\mathsf{BST}(\cap, =, \neq)$-formula. Then φ is satisfiable if and only if $L \not\sim_\varphi R$ (namely $\overline{L} \neq \overline{R}$) holds for every literal of the form $\bigcap L \neq \bigcap R$ in φ.*

[3] For definiteness, such $b_{\overline{V}}$'s can be drawn from the collection $\{\{\emptyset\}, \{\{\emptyset\}\}, \{\{\{\emptyset\}\}\}, \dots\}$ of Zermelo's non-zero numerals.

3.1 A cubic-time satisfiability test for $\mathsf{BST}(\cup,=,\neq)$ and for $\mathsf{BST}(\cap,=,\neq)$

Theorems 1 and 2 imply that any $\mathsf{BST}(\star,=,\neq)$-formula φ, where $\star \in \{\cup,\cap\}$, is satisfiable if and only if $\overline{L} \neq \overline{R}$ holds for every literal $\bigstar L \neq \bigstar R$ in φ. Hence, they yield straight decision procedures for the theories $\mathsf{BST}(\star,=,\neq)$, with $\star \in \{\cup,\cap\}$.

The next step will then be to provide a quadratic algorithm for computing the closure $\overline{Z}$ of any input $Z \in \mathscr{P}^+(\mathit{Vars}(\varphi))$, namely, the largest set in $\mathscr{P}^+(\mathit{Vars}(\varphi))$ which is $\sim_\varphi$-equivalent to Z.

In Algorithm 1 below, we provide a high-level specification of the function Closure, intended to compute the closure $\overline{Z}$ of any given $Z \in \mathscr{P}^+(\mathit{Vars}(\varphi))$, for a $\mathsf{BST}(\star,=,\neq)$-formula φ. After proving its correctness, we will illustrate a lower-level implementation whose time complexity is *quadratic* (as opposed to the cubic-time complexity which would ensue from a naive implementation).

Algorithm 1 Satisfiability tester

Require: A $\mathsf{BST}(\star,=,\neq)$-formula φ represented by the sets of pairs $\Phi_\varphi^\oplus$ and $\Phi_\varphi^\ominus$.
Ensure: Is φ satisfiable?
1: **for each** $\{L,R\}$ in $\Phi_\varphi^\ominus$ **do**
2: **if** Closure(L,$\Phi_\varphi^\oplus$) = Closure(R,$\Phi_\varphi^\oplus$) **then**
3: **return false**;
4: **return true**;

1: **function** Closure(Z,$\Phi_\varphi^\oplus$)
Input: a set Z and the set $\Phi_\varphi^\oplus$
Output: the $\sim_\varphi$-closure $\mathcal{Z}$ of Z
2: $\mathcal{Z} \leftarrow Z$;
3: **while** there exists $\{L,R\} \in \Phi_\varphi^\oplus$ such that $L \subseteq \mathcal{Z} \iff R \not\subseteq \mathcal{Z}$ **do**
4: $\mathcal{Z} \leftarrow \mathcal{Z} \cup L \cup R$;
5: **return** $\mathcal{Z}$;

We can now prove quite easily the correctness of the function Closure.

Lemma 5. *The function* Closure *computes closures correctly.*

Proof. Given a $\mathsf{BST}(\star,=,\neq)$-formula φ, with input a set $Z \subseteq \mathit{Vars}(\varphi)$ and a collection $\Phi_\varphi^\oplus$ the while-loop of the function Closure plainly terminates within a number $k \leqslant |\Phi_\varphi^\oplus|$ of iterations. Let $\mathcal{Z}_i$ be the value of the variable $\mathcal{Z}$ after i iterations, so that $\mathcal{Z}_0 = Z$. Preliminarily, we prove by induction on $i = 0,1,\ldots,k$ that $\mathcal{Z}_i \sim_\varphi Z$ and $Z \subseteq Z_i$.

The base case $i = 0$ is trivial.

Next, let $\{L,R\} \in \Phi_\varphi^\oplus$ be the pair selected by the while-loop during its i-th iteration, with $i \geqslant 1$. Hence, we have:

$$L \sim_\varphi R, \quad L \subseteq \mathcal{Z}_{i-1} \iff R \not\subseteq \mathcal{Z}_{i-1}, \quad \text{and} \quad \mathcal{Z}_i = \mathcal{Z}_{i-1} \cup L \cup R.$$

Thus, by inductive hypothesis and Lemma 4(g), we have

$$Z \sim_\varphi \mathcal{Z}_{i-1} \sim_\varphi \mathcal{Z}_{i-1} \cup L \cup R = \mathcal{Z}_i \quad \text{and} \quad \mathcal{Z}_i = \mathcal{Z}_{i-1} \cup L \cup R,$$

from which $Z \sim_\varphi \mathcal{Z}_i$ and $Z \subseteq \mathcal{Z}_i$ follow. Hence, by induction, we have $Z \sim_\varphi \mathcal{Z}^{\mathrm{f}}$ and $Z \subseteq \mathcal{Z}^{\mathrm{f}}$, where $\mathcal{Z}^{\mathrm{f}} := \mathcal{Z}_k$ is the final value of the variable $\mathcal{Z}$ returned by the execution of $\textsc{Closure}(Z, \Phi_\varphi^{\oplus})$.

In addition, the termination condition for the while-loop yields that $L \subseteq \mathcal{Z}^{\mathrm{f}} \iff R \subseteq \mathcal{Z}^{\mathrm{f}}$, for all $\{L, R\} \in \Phi_\varphi^{\oplus}$. Thus, from Lemma 3, it follows that

$$W_1 \subseteq \mathcal{Z}^{\mathrm{f}} \iff W_2 \subseteq \mathcal{Z}^{\mathrm{f}}, \tag{5}$$

for all $W_1, W_2 \in \mathscr{P}^+(\mathit{Vars}(\varphi))$ such that $W_1 \sim_\varphi W_2$.

In order to prove that $\mathcal{Z}^{\mathrm{f}} = \overline{Z}$, we observe that, since $Z \sim_\varphi \overline{Z}$ and $Z \subseteq \mathcal{Z}^{\mathrm{f}}$, by (5) we have $\overline{Z} \subseteq \mathcal{Z}^{\mathrm{f}}$. Moreover, by Lemma 4(a),(d) and since $Z \sim_\varphi \mathcal{Z}^{\mathrm{f}}$, we readily get $\mathcal{Z}^{\mathrm{f}} \subseteq \overline{\mathcal{Z}^{\mathrm{f}}} = \overline{Z}$. The latter inclusion, together with the previously established one $\overline{Z} \subseteq \mathcal{Z}^{\mathrm{f}}$, implies $\mathcal{Z}^{\mathrm{f}} = \overline{Z}$, i.e., $\mathcal{Z}^{\mathrm{f}}$ is the closure of Z, proving that the call to $\textsc{Closure}(Z, \Phi_\varphi^{\oplus})$ computes the closure $\overline{Z}$ of Z correctly. □

Theorems 1 and 2, together with Lemma 5 and Lemma 4(d), readily yield that Algorithm 1 is a valid satisfiability test for formulae in the languages $\mathsf{BST}(\cup, =, \neq)$ and $\mathsf{BST}(\cap, =, \neq)$.

A quadratic implementation of the function Closure

Next, we provide a quadratic implementation of the function Closure, which, for a given $\mathsf{BST}(\star, =, \neq)$-formula φ, takes as input the collection $\Phi_\varphi^{\oplus}$ and a set $Z \in \mathscr{P}^+(\mathit{Vars}(\varphi))$ of which one wants to compute the closure $\overline{Z}$. As internal data structures, the function Closure uses: a doubly linked list Ripe of sets of the form $(L \cup R) \setminus \mathcal{Z}$, where $\{L, R\} \in \Phi_\varphi^{\oplus}$ and $\mathcal{Z}$ is the internal variable whose value will converge to $\overline{Z}$ at termination; a doubly linked list Unripe of pairs of form $\big\langle (L \setminus \mathcal{Z}), (R \setminus \mathcal{Z}) \big\rangle$, with $\{L, R\} \in \Phi_\varphi^{\oplus}$; and an array Aux of m lists of pointers to nodes either in Ripe or in Unripe, where m is the number of the distinct variables $x_1, \ldots, x_m$ in φ, intended to allow fast retrieval of nodes in the lists Ripe and Unripe.

We will express the complexity of the main procedure in Algorithm 1 and of our efficient implementation of the function Closure in terms of the four quantities m, n, p, q, where $m = |\mathit{Vars}(\varphi)|$ is the number of distinct set variables in φ, $n = |\varphi|$ is the size of φ, $p = |\Phi_\varphi^{\oplus}|$ is the number of literals of the form $(=)$ in φ, and $q = |\Phi_\varphi^{\ominus}|$ is the number of literals of the form $(\neq)$ in φ. Plainly, we have $m, p, q \leqslant n$.

Much as in [6], we can index the variables in $\mathit{Vars}(\varphi)$ from 1 to $m = |\mathit{Vars}(\varphi)|$, so that every subset A of $\mathit{Vars}(\varphi)$ can be represented as a Boolean array of size m such that any set variable x_i belongs to A if and only if $A[i] = 1$. In fact, it is possible to build such an index and initialize accordingly all the arrays corresponding to the collections of set variables present in $\bigcup \Phi_\varphi^{\oplus} \cup \bigcup \Phi_\varphi^{\ominus}$ in $\mathcal{O}(m(p+q)+n)$ time, even starting from a formula φ in plain text, yet to be parsed. Specifically, for each literal ℓ in φ of the form $\bigstar L = \bigstar R$ or $\bigstar L \neq \bigstar R$, we let π_L and π_R be pointers to the sets L and R, respectively. Then, while parsing the formula φ, we construct the collection of pairs

```
1: function Closure(Z,Φ⊕φ)
2:     Z ← Z;
3:     for each {L, R} ∈ Φ⊕φ do
4:         if L ∪ R ⊈ Z then
5:             if L ⊆ Z or R ⊆ Z then Ptr ← Add(Ripe, (L ∪ R) \ Z);
6:                 ▷ Ptr is a pointer to the node just added to the list Ripe ◁
7:             else Ptr ← Add(Unripe, ⟨(L \ Z), (R \ Z)⟩);
8:             for each index i such that x_i ∈ L ∪ R do Add(Aux[i], Ptr);
9:     while Ripe is not empty do
10:        S ← Extract(Ripe);    ▷ Extracts the first set in Ripe
11:        for each index i such that x_i ∈ S do
12:            Z ← Z ∪ {x_i};
13:            for each pointer Ptr in Aux[i] do
14:                if Ptr is in Ripe then    ▷ Ptr points to the set Ptr.Data
15:                    Ptr.Data ← Ptr.Data \ {x_i};
16:                    if Ptr.Data = ∅ then Remove(Ripe, Ptr);
17:                else    ▷ Ptr points to the pair Ptr.Data in the list Unripe
18:                    L_Ptr ← Ptr.Data.Left ← Ptr.Data.Left \ {x_i};
19:                    R_Ptr ← Ptr.Data.Right ← Ptr.Data.Right \ {x_i};
20:                    if L_Ptr = ∅ or R_Ptr = ∅ then
21:                        Remove(Unripe, Ptr);
22:                        if L_Ptr ≠ ∅ then Add(Ripe, L_Ptr)
23:                        else if R_Ptr ≠ ∅ then Add(Ripe, R_Ptr)
24:                Remove(Aux[i], Ptr);
25:    return Z;
```

$$\Pi \coloneqq \left\{ \langle x, \pi_A \rangle \mid x \in A \in \bigcup \Phi^{\oplus}_{\varphi} \cup \bigcup \Phi^{\ominus}_{\varphi} \right\}$$

and sort it in $\mathcal{O}(n)$ time, according to the first components, using the lexicographic sorting algorithm of strings of varying length described in [1, Algorithm 3.2 (pp. 80–84)].

Having sorted Π, we can easily collect the $m \leqslant n$ distinct variables of φ and index them using the integers $1, \ldots, m$. By means of such an indexing, in $\mathcal{O}(m(p+q))$ time (where $p = |\Phi^{\oplus}_{\varphi}|$ and $q = |\Phi^{\ominus}_{\varphi}|$) we can represent as an m-bit array each set of variables in $\bigcup \Phi^{\oplus}_{\varphi} \cup \bigcup \Phi^{\ominus}_{\varphi}$, and accordingly represent $\Phi^{\oplus}_{\varphi}$ and $\Phi^{\ominus}_{\varphi}$ as lists of pairs of m-bit arrays. Such preliminary encoding phase can be carried out in $\mathcal{O}(m(p+q)+n)$ time.

We make use of the auxiliary functions Add(List, S) and Remove(List, Ptr) to add S to List (S can be either a set or a pair of sets) and to remove the node pointed to by Ptr from List, respectively. Since the two lists we use, namely Ripe and Unripe, are maintained as doubly linked lists, both operations can be performed in $\mathcal{O}(1)$ time. The function Add returns also a pointer to the newly inserted node. Finally, we use the function Extract(List) to access in $\mathcal{O}(1)$ time the pointer to the first node of List while removing it.

The function Closure$(Z, \Phi^{\oplus}_{\varphi})$ comprises two phases: an initialization phase, lines 2–8, and a computation phase, lines 9–25.

For each m-bit array, we maintain a counter of its bits set to 1, so that emptiness tests can be performed in $\mathcal{O}(1)$ time. Plainly, unions, set differences,

and inclusion tests of sets represented as m-bit arrays can easily be performed in $\mathcal{O}(m)$ time. Also, membership tests and the operations of singleton addition and singleton removal can be performed in $\mathcal{O}(1)$ time.

Thus, the initialization phase of the function Closure (lines 2–8) can be performed in $\mathcal{O}(mp)$ time.

At the end of the initialization phase and at each subsequent step, the lists Ripe and Unripe contain only sets disjoint from $\mathcal{Z}$, and each of them has length at most $p = |\Phi^{\oplus}_{\varphi}|$. Specifically, the list Ripe contains the set $(L \cup R) \setminus \mathcal{Z}$, for all $\{L, R\} \in \Phi^{\oplus}_{\varphi}$ such that $L \cup R \not\subseteq \mathcal{Z}$ but either $L \subseteq \mathcal{Z}$ or $R \subseteq \mathcal{Z}$ holds. Instead, the list Unripe contains the pair $\langle L \setminus \mathcal{Z}, R \setminus \mathcal{Z}\rangle$, for all $\{L, R\} \in \Phi^{\oplus}_{\varphi}$ such that $L \not\subseteq \mathcal{Z}$ and $R \not\subseteq \mathcal{Z}$ both hold.

In view of the assignments at lines 15, 18, and 19, the disjointness property from $\mathcal{Z}$ of the sets $(L \cup R) \setminus \mathcal{Z}$ in Ripe and the sets $L \setminus \mathcal{Z}$ and $R \setminus \mathcal{Z}$ such that $\{L \setminus \mathcal{Z}, R \setminus \mathcal{Z}\}$ is in Unripe is maintained at each iteration of the **while**-loop at lines 9–24. Hence, at each extraction of a set S from the list Ripe at line 10, none of the set variables in S has already been selected and processed by the **for**-loop 11–24. Therefore, each set variable in $\mathit{Vars}(\varphi)$ is processed by the **for**-loop at lines 11–24 at most once, yielding that, in the overall, the **for**-loop 11–24 is executed at most m times, for a total of $\mathcal{O}(mp)$ time, since each execution of the **for**-loop 11–24, say relative to a set variable x_i, is dominated by the time taken by the internal **for**-loop 13–24, which is $\mathcal{O}(p)$. Indeed, (i) at the end of the initialization phase, the list Aux$[i]$ contains at most p pointers to nodes in the lists Ripe and Unripe; (ii) once a pointer in the list Aux$[i]$ is processed, it is then removed (line 24), so that it will never be processed again; and (iii) each line of the **for**-loop 13–24 can be executed in constant time.

Since each extraction at line 10 takes $\mathcal{O}(1)$ time and, as observed, the list Ripe has size at most p at the end of the initialization phase, it follows that the **while**-loop at lines 9–24 takes $\mathcal{O}(mp)$ time.

Thus, the overall complexity of the function Closure is $\mathcal{O}(mp)$ time. Since $m, p \leqslant n$, we have also that the function Closure takes quadratic time $\mathcal{O}(n^2)$ in the size n of the formula φ to be tested for satisfiability.

Finally, our satisfiability tester (Algorithm 1) checks at most q pairs $\{L, R\} \in \Phi^{\ominus}_{\varphi}$ in $\mathcal{O}(mpq)$ time, by computing the closures $\overline{L}$ and $\overline{R}$ by means of calls to the function Closure and comparing them. By taking into account also the preliminary encoding phase, which has a $\mathcal{O}(m(p+q)+n)$-time complexity, the overall complexity of Algorithm 1 is $\mathcal{O}(mpq+n)$, which is $\mathcal{O}(n^3)$ since $m, p, q \leqslant n$.

Concerning the space complexity of our satisfiability tester, it is immediate to check that all data structures used in Algorithm 1 and in the function Closure require $\mathcal{O}(mp)$ space, namely $\mathcal{O}(n^2)$ space since $m, p \leqslant n$.

Summarizing, we have proved the following result:

Theorem 3. *The satisfiability decision problem for the language* $\mathsf{BST}(\cup, =, \neq)$*, as well as for* $\mathsf{BST}(\cap, =, \neq)$*, can be solved in cubic time and quadratic space.*

Remark 1. It is not hard to see that our satisfiability tester (Algorithm 1) and its auxiliary function Closure can be refined in order that an explicit set assignment modeling the input conjunction φ is returned when φ is satisfiable.

4 Extensions of $\mathsf{BST}(\cup, =, \neq)$ and $\mathsf{BST}(\cap, =, \neq)$

In [4] we presented two notions called *existential expressibility* and $\mathcal{O}(f)$-*expressibility*, through which one can reduce, in constant or $\mathcal{O}(f)$ time, a set-theoretic formula of a richer language to an equisatisfiable formula belonging to a smaller language. Those notions rely on techniques for replacing formulae that involve operators or relators not belonging to the smaller language with convenient surrogates, only comprising operators of the smaller language.

Regarding the languages $\mathsf{BST}(\cup, =, \neq)$ and $\mathsf{BST}(\cap, =, \neq)$, we have:

(a) the literal $x = \varnothing$ is $\mathcal{O}(n)$-expressible in $\mathsf{BST}(\cup, =)$;
(b) the literal $x = \varnothing$ is $\mathcal{O}(n)$-expressible in $\mathsf{BST}(\cap, =)$;[4]
(c) the literal $x \subseteq y$ is existentially expressible in $\mathsf{BST}(\cup, =)$ and in $\mathsf{BST}(\cap, =)$;
(d) the literal $x \not\subseteq y$ is existentially expressible in $\mathsf{BST}(\cup, \neq)$ and in $\mathsf{BST}(\cap, \neq)$;
(e) the literal $\textsc{Disj}(x, y)$ is existentially expressible in $\mathsf{BST}(\cap, =\varnothing)$ and therefore, by (b), it is $\mathcal{O}(n)$-expressible in $\mathsf{BST}(\cap, =)$;
(f) the literal $\neg\textsc{Disj}(x, y)$ is existentially expressible in $\mathsf{BST}(\subseteq, \neq)$; therefore, by (c), it is existentially expressible in both of $\mathsf{BST}(\cup, =, \neq)$ and $\mathsf{BST}(\cap, =, \neq)$.

Wrapping up, we have that (a), (c), (d), and (f) ensure that any $\mathsf{BST}(\cup, =\varnothing, \neq\varnothing, \neg\textsc{Disj}, \subseteq, \not\subseteq, =, \neq)$-formula can be reduced in linear time to an equisatisfiable $\mathsf{BST}(\cup, =, \neq)$-formula. Similarly (b), (c), (d), and (e), (f) ensure that any $\mathsf{BST}(\cap, =\varnothing, \neq\varnothing, \textsc{Disj}, \neg\textsc{Disj}, \subseteq, \not\subseteq, =, \neq)$-formula can be reduced in linear time to an equisatisfiable $\mathsf{BST}(\cap, =, \neq)$-formula. Therefore:

Lemma 6. *The satisfiability decision problem for either one of the languages* $\mathsf{BST}(\cup, =\varnothing, \neq\varnothing, \neg\textsc{Disj}, \subseteq, \not\subseteq, =, \neq)$, $\mathsf{BST}(\cap, =\varnothing, \neq\varnothing, \textsc{Disj}, \neg\textsc{Disj}, \subseteq, \not\subseteq, =, \neq)$ *can be solved in cubic time.*

Remark 2. In [4] it is also proved that $\mathsf{BST}(\cup, =\varnothing, \neq\varnothing, \neg\textsc{Disj}, \subseteq, \not\subseteq, =, \neq)$ and $\mathsf{BST}(\cap, =\varnothing, \neq\varnothing, \textsc{Disj}, \neg\textsc{Disj}, \subseteq, \not\subseteq, =, \neq)$ are polynomial-maximal, which means that every language strictly containing either one of those languages is endowed with an NP-hard satisfiability decision problem.

Related work and conclusions

In [4,6], we highlighted initial results on fragments of set theory endowed with polynomial-time satisfiability decision tests, potentially useful for automated proof verification and, more generally, in the symbolic manipulation of declarative specifications (cf., e.g., [18,9,8]). At the outset, we focused on 'Boolean Set Theory', $\mathbb{BST}$, namely the language of quantifier-free formulae that involves set-variables, the Boolean set operators $\cup, \cap, \setminus$, the Boolean relators $\subseteq, \not\subseteq, =, \neq$, and the predicates '$\blacksquare = \varnothing$' and '$\mathsf{Disj}(\blacksquare, \blacksquare)$', along with their opposites. That language, whose expressive power is greater than it may appear at first glance (cf. [3]), has an NP-complete satisfiability problem. In [4] we organized the fragments of $\mathbb{BST}$

[4] The proofs of (a) and of (c)–(f) appear in [4]; the proof of (b), which is new, is provided in Appendix A.

in a full complexity taxonomy which spots the 18 minimal NP-complete fragments, and the 5 maximal polynomial fragments. We then announced a study on sub-maximal polynomial fragments of $\mathbb{BST}$, which this paper has undertaken.

We plan to address the satisfiability problem for the theories $\mathsf{BST}(\cup, \not\subset, \neq)$ and $\mathsf{BST}(\cap, \not\subset, \neq)$ by suitably adapting to them the equivalence relation $\sim_\varphi$ and its related closure operator introduced in this paper for the fragments $\mathsf{BST}(\cup, =, \neq)$ and $\mathsf{BST}(\cap, =, \neq)$. In view of the equivalences

$$A = B \iff A \not\subset A \cup B \ \wedge \ B \not\subset A$$
$$A = B \iff A \cap B \not\subset A \ \wedge \ B \not\subset A,$$

it turns out that the theories $\mathsf{BST}(\cup, \not\subset, \neq)$ and $\mathsf{BST}(\cap, \not\subset, \neq)$ are extensions of $\mathsf{BST}(\cup, =, \neq)$ and of $\mathsf{BST}(\cap, =, \neq)$, respectively. Notice that the relation $A \not\subset B$, which stands for $A \not\subseteq B \ \vee \ A = B$, embodies a disjunction. It is therefore expected that the resulting satisfiability tests for $\mathsf{BST}(\cup, \not\subset, \neq)$ and $\mathsf{BST}(\cap, \not\subset, \neq)$ may have a complexity worse than cubic, though still polynomial.

We envisage a confluence of the line of research centered on satisfiability testers, to which this paper, along with [4,6,3], contributes, with another active line of research centered on set-unification algorithms (see, e.g., [10]). A quick satisfiability tester often is, in fact, less helpful than a solver able to produce a bunch of solution templates which cover, collectively, all possible models of a given formula; or, even better, a solver supplying a symbolic solution of maximum possible generality: e.g., the Büttner–Simonis unification algorithm for Boolean algebras [2], which produces, when a model exists, a most general unifier.

Another foreseeable confluence has to do with a long-standing line of research initiated by [12,13], concerning the cooperation among decision algorithms (see also, among many, [16]). In connection with the problem of combining decision algorithm, it is worth noticing that not just $\mathbb{BST}$ but even the much richer decidable theory MLS turns out to be 'convex' in the sense explained in [14].

Acknowledgements We gratefully acknowledge partial support from project "STORAGE—Università degli Studi di Catania, Piano della Ricerca 2020/2022, Linea di intervento 2", and from INdAM-GNCS 2019 and 2020 research funds.

We are also grateful to the anonymous reviewers for their helpful comments.

References

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, 1975.
2. W. Büttner and H. Simonis. Embedding boolean expressions into logic programming. *J. Symb. Comput.*, 4(2):191–205, 1987.
3. D. Cantone, A. De Domenico, P. Maugeri, and E. G. Omodeo. A quadratic reduction of constraints over nested sets to purely Boolean formulae in CNF. In F. Calimeri, S. Perri, and E. Zumpano, editors, *Proceedings of the 35th Italian Conference on Computational Logic - CILC 2020, Rende, Italy, October 13-15, 2020*, volume 2710 of *CEUR Workshop Proceedings*, pages 214–230. CEUR-WS.org, 2020.

4. D. Cantone, A. De Domenico, P. Maugeri, and E. G. Omodeo. Complexity assessments for decidable fragments of set theory. I: A taxonomy for the Boolean case. *Fundam. Informaticae*, 181(1):37–69, 2021.
5. D. Cantone, A. Ferro, and E. G. Omodeo. *Computable Set Theory*, volume 1 of *International Series of Monographs on Computer Science*. Clarendon Press, Oxford, UK, 1989.
6. D. Cantone, P. Maugeri, and E. G. Omodeo. Complexity assessments for decidable fragments of set theory. II: A taxonomy for 'small' languages involving membership. *Theor. Comput. Sci.*, 848:28–46, 2020.
7. D. Cantone, E. G. Omodeo, and A. Policriti. The automation of syllogistic. II: Optimization and complexity issues. *J. Autom. Reason.*, 6(2):173–188, June 1990.
8. M. Cristiá and G. Rossi. Solving quantifier-free first-order constraints over finite sets and binary relations. *J. Autom. Reason.*, 64(2):295–330, 2020.
9. A. Dovier, C. Piazza, and G. Rossi. A uniform approach to constraint-solving for lists, multisets, compact lists, and sets. *ACM Trans. Comput. Log.*, 9(3):15:1–30, 2008.
10. A. Dovier, E. Pontelli, and G. Rossi. Set unification. *Theor. Pract. Log. Prog.*, 6(6):645–701, 2006.
11. A. Ferro, E. G. Omodeo, and J. T. Schwartz. Decision procedures for elementary sublanguages of set theory. I: Multi-level syllogistic and some extensions. *Commun. Pure Appl. Math.*, 33(5):599–608, 1980.
12. G. Nelson and D. C. Oppen, Simplification by Cooperating Decision Procedures, *ACM Trans. Program. Lang. Syst.*, 1(2):245–257, 1979.
13. G. Nelson and D. C. Oppen, Fast Decision Procedures Based on Congruence Closure, *J. ACM*, 27(2):356–364, 1980.
14. D. C. Oppen, Complexity, Convexity and Combinations of Theories, *Theor. Comput. Sci.*, 12:291–302, 1980.
15. F. Parlamento and A. Policriti. Decision Procedures for Elementary Sublanguages of Set Theory. IX: Unsolvability of the decision problem for a restricted subclass of the Δ_0-formulas in Set Theory, *Comm. Pure Appl. Math.*, 41(2):221–251, 1988.
16. Deciding Combinations of Theories, J. ACM, 31(1):1–12, 1984.
17. J. T. Schwartz, D. Cantone, and E. G. Omodeo. *Computational Logic and Set Theory – Applying Formalized Logic to Analysis*. Springer, 2011. Foreword by Martin Davis.
18. F. Stolzenburg. Membership-constraints and complexity in logic programming with sets. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems, First International Workshop FroCoS 1996, Munich, Germany, March 26-29, 1996, Proceedings*, volume 3 of *Applied Logic Series*, pages 285–302. Kluwer Academic Publishers, 1996.

A Expressibility

In this appendix we highlight the reduction technique used in Section 4.

Most of our reductions are based on the standard notion of 'context-free' expressibility:

Definition 1 (Existential expressibility). *A formula $\psi(\boldsymbol{x})$ is said to be* existentially expressible *in a theory $\mathcal{T}$ if there exists a $\mathcal{T}$-formula $\Psi(\boldsymbol{x}, \boldsymbol{z})$ such that*

$$\models \quad \psi(\boldsymbol{x}) \iff (\exists \boldsymbol{z})\, \Psi(\boldsymbol{x}, \boldsymbol{z}),$$

where $\boldsymbol{x}$ and $\boldsymbol{z}$ stand for tuples of set variables.

We also devised a more general notion of 'context-sensitive' expressibility, embodying complexity-related information.

Definition 2 ($\mathcal{O}(f)$-expressibility). *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be any theories and $f\colon \mathbb{N} \to \mathbb{N}$ be a given map. A collection $\mathcal{C}$ of formulae is said to be* $\mathcal{O}(f)$-expressible from $\mathcal{T}_1$ into $\mathcal{T}_2$ *if there exists a map*

$$\langle\, \varphi(\boldsymbol{y}),\ \psi(\boldsymbol{x})\,\rangle \ \mapsto\ \Xi^{\psi}_{\varphi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \tag{6}$$

from $\mathcal{T}_1 \times \mathcal{C}$ into $\mathcal{T}_2$, where no variable in $\boldsymbol{z}$ occurs in either $\boldsymbol{x}$ or $\boldsymbol{y}$, such that the following conditions are satisfied:

(a) the mapping (6) *can be computed in $\mathcal{O}\big(f(|\varphi \wedge \psi|)\big)$-time,*
(b) if $\varphi(\boldsymbol{y}) \wedge \Xi^{\psi}_{\varphi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is satisfiable, so is $\varphi(\boldsymbol{y}) \wedge \psi(\boldsymbol{x})$,
(c) $\models \big(\varphi(\boldsymbol{y}) \ \wedge\ \psi(\boldsymbol{x})\big) \ \longrightarrow\ (\exists \boldsymbol{z}) \Xi^{\psi}_{\varphi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.

We are now ready to prove that the literal $x = \varnothing$ is $\mathcal{O}(n)$-expressible in $\mathsf{BST}(\cap, =)$, albeit not existentially expressible in $\mathsf{BST}(\cap, =, \neq)$.

Lemma 7. *The literal $x = \varnothing$ is not existentially expressibile in $\mathsf{BST}(\cap, =, \neq)$.*

Proof. Assume by way of contradiction that $x = \varnothing$ is existentially expressible in $\mathsf{BST}(\cap, =, \neq)$, so that there exists a $\mathsf{BST}(\cap, =, \neq)$-formula $\Psi(x, \boldsymbol{y})$ such that:

$$\models x = \varnothing \iff (\exists \boldsymbol{z})\Psi(x, \boldsymbol{z}). \tag{7}$$

Since $x = \varnothing$ is plainly satisfiable then so it is Ψ, therefore it is satisfied by the set assignment $x \overset{M}{\mapsto} \{b_V \mid x \in \overline{V}\}$ (dual to the assignment (4), and actually exploited in the proof of Theorem 2). Notice that M is such that, for each set variable $y \in \mathit{Vars}(\Psi)$, $My \neq \emptyset$ holds; and, since $x \in \mathit{Vars}(\Psi)$ we obtain

$$M \models (\exists \boldsymbol{z})\Psi \wedge x \neq \emptyset,$$

contradicting (7), whence our claim follows. □

Lemma 8. *The literal $x = \varnothing$ is $\mathcal{O}(|\varphi|)$-expressible from $\mathsf{BST}(\cap, =)$ into $\mathsf{BST}(\cap, =)$.*

Proof. We will show that the map

$$\langle\, \varphi(\boldsymbol{y})\,,\ x = \varnothing\,\rangle \mapsto \bigcap_{y\in\boldsymbol{y}} y \cap x = x, \tag{8}$$

defined for all $\varphi(\boldsymbol{y}) \in \mathsf{BST}(\cap, =)$, satisfies (a), (b), and (c), where f is the function sending each formula φ to its length $n = |\varphi|$.

Plainly the map (8) can be computed by collecting all variables occurring in the formula φ while scanning it, hence it can be computed in $\mathcal{O}(n)$-time.

Concerning (b), let M be a model for $\varphi(\boldsymbol{y}) \wedge \bigcap_{y\in\boldsymbol{y}} y \cap x = x$. Then we have $Mx = Mx \cap My$, namely $Mx \subseteq My$, for all $y \in \mathit{Vars}(\varphi)$. Put

$$M'v \coloneqq Mv \setminus Mx,$$

for every set variable $v \in \mathit{Vars}(\varphi) \cup \{x\}$. Plainly $M'x = Mx \setminus Mx = \emptyset$, so that $M' \models x = \varnothing$.

To see that $M' \models \varphi(\boldsymbol{y})$, consider any literal $\bigcap L = \bigcap R$ of φ, and note that:

$$\begin{aligned}
&\textstyle\bigcap ML = \bigcap MR && \text{(since } M \models \varphi\text{)}\\
\therefore\ &\textstyle(\bigcap ML \setminus Mx) \cup Mx = (\bigcap MR \setminus Mx) \cup Mx && \text{(since } Mx \subseteq My \text{ for all } y \in \mathit{Vars}(\varphi)\text{)}\\
\therefore\ &\textstyle\bigcap M'L = \bigcap M'R;
\end{aligned}$$

therefore M' models each literal of φ.

Finally, concerning (c), let M be a model for $\varphi(\boldsymbol{y}) \wedge x = \varnothing$. Then $Mx = \emptyset$ so that

$$\bigcap_{y\in\boldsymbol{y}} My \cap Mx = \bigcap_{y\in\boldsymbol{y}} My \cap \emptyset = \emptyset = Mx;$$

by the genericity of M, we readily get

$$\models \varphi(\boldsymbol{y}) \wedge x = \varnothing \longrightarrow \bigcap_{y\in\boldsymbol{y}} y \cap x = x,$$

whence the claim follows. □