International Journal of Mathematics, Statistics and Operations Research
Vol. 6; Number 1; 2026, pp. 81-98
https://doi.org/10.47509/IJMSOR.2026.v06i01.04

ARF INDIA
Academic Open Access Publishing
www.arfjournals.com

US-Completeness Results on the Uniqueness of Solutions for Satisfiability Problems

HUDRY, Olivier[a]

[a] *Télécom Paris, Institut polytechnique de Paris, 91123 Palaiseau, France*

ABSTRACT
This paper studies the complexity of some problems related to the satisfiability of Boolean formulas and consisting in asking about the *uniqueness* of a solution. More precisely, we show that the following decision problems: Unique Satisfiability (U-SAT), Unique k-Satisfiability (U-k-SAT) for $k \geqslant 3$ and Unique One-in-Three Satisfiability (U-1-3-SAT) have equivalent complexities by establishing explicit reductions relating each problem to each other; these reductions are polynomial (with respect to the binary size of the transformed instances) and keep the uniqueness of the solution when the solution is unique. As a consequence, these problems are all *US*-complete, and hence are co-*NP*-hard and belong to *DP*.

KEYWORDS
Complexity Theory, classes *DP* and *US*, *US*-completeness, Uniqueness of Solution, Boolean Satisfiability

1. Introduction

In the theory of complexity (see for instance [3], [5], [11], [20] or [28] for a general presentation of this topic), *decision problems* are problems stating a question that admits only the answer YES or NO; the question can be set in the very general following form: "Given an instance I and a property $\mathcal{P}$ on I, is $\mathcal{P}$ true for I?" where $\mathcal{P}$ can be expressed as: "Is there an appropriate entity – which can be a set, a function, a number..., such an entity depending on the problem – satisfying a given characteristic?". In our study, we add the extra word "unique" to the latter question: "Is there a *unique* entity satisfying the given characteristic?". Inside this framework, we study the complexity status of the uniqueness issue of some variations of the well-known problem Satisfiability (SAT).

The complexity of uniqueness of solutions, which may be seen as a part of the wider question of the number of solutions of a problem, had been studied earlier in some papers (see, e.g., [27], [36] or, for SAT problems: [7], [8], [12], [21], [25], [26], [30], [31], and, for graph theoretical problems: [16], [17], [18], [19]).

The Satisfiability problems considered in this paper are depicted in Section 2. In Section 3, we recall the necessary background about the theory of complexity. We

CONTACT Author[a]. Email:olivier.hudry@telecom-paris.fr

Article History
Received: 18 February 2026; Revised: 04 April 2026; Accepted: 15 April 2026; Published: 30 June 2026

To cite this paper
HUDRY, Olivier (2026). US-Completeness Results on the Uniqueness of Solutions for Satisfiability Problems. *International Journal of Mathematics, Statistics and Operations Research.* 6(1), 81-98.

study the complexity of the Satisfiabilty problems when we consider the question of the uniqueness of a solution in Section 4: we prove that the problems called below U-SAT, U-1-3-SAT and U-k-SAT ($k \geqslant 3$) have equivalent complexities by providing explicit polynomial reductions between them; consequently, they are *US*-complete and thus they also belong to *DP* and they are co-*NP*-hard (or also *NP*-hard with respect to Turing reductions, see below, Subection 3.2). We present some concluding remarks and open problems in Section 5.

2. Satisfiability Problems

We now recall the necessary definitions related to problems dealing with Boolean Satisfiability (for references about Satisfiability problems, see for instance [6], [24] or [32] and references inside; for an algorithmic point of view, see also [13], [34] or [35] for instance).

We consider a set $\mathcal{X}$ of n *Boolean variables* x_1, x_2, ..., x_n, *i.e.* variables whose values are only TRUE or FALSE. The *complement* (or *negated variable*) $\overline{x}$ of a variable $x \in \mathcal{X}$ is a variable such that the Boolean value of $\overline{x}$ is TRUE if and only if the Boolean value of x is FALSE. A *literal* is any variable $x \in \mathcal{X}$ or its complement $\overline{x}$. Let $\overline{\mathcal{X}}$ denote the set of the complements of the variables of $\mathcal{X}$; then $\mathcal{X} \cup \overline{\mathcal{X}}$ is the set of the literals. A *clause* is any subset of $\mathcal{X} \cup \overline{\mathcal{X}}$. A *logical formula* is a set of clauses.

Equivalently, if $\wedge$ stands for the Boolean "and" and $\vee$ for the Boolean "or", then a logical formula $\mathcal{F}$ of m clauses $c_j = \{\ell_{j,1}, \ell_{j,2}, ..., \ell_{j,\kappa_j}\}$, where κ_j denotes the number of literals $\ell_{j,i}$ contained in c_j, can be stated as follows:

$$\mathcal{F} = \wedge_{1 \leqslant j \leqslant m}\ c_j \text{ with } c_j = \vee_{1 \leqslant i \leqslant \kappa_j}\ \ell_{j,i}.$$

This form is called the *normal conjunctive form.*

A *truth assignment* for $\mathcal{X}$ sets the variable x_i to TRUE and its complement to FALSE, or *vice-versa.* A truth assignment φ is said to *satisfy the clause* c_j (respectively *the formula* $\mathcal{F}$) if c_j (respectively every clause c_j of $\mathcal{F}$) contains at least one literal $\ell_{j,i}$ set to TRUE by φ. In other words, a truth assignment satisfies a clause or a formula if and only if it sets the clause or the formula to TRUE.

In the sequel, we assume that a given literal is not contained twice or more in any clause and that no clause contains a variable and its complement simultaneously. Moreover, we assume that, for any variable x, x or $\overline{x}$ (but not necessarily both) appears at least once in the clauses (in other words, there is no useless variable).

The following decision problems, for which the binary size of the instance is polynomially linked to $n + m$, are classical problems in complexity.

Problem SAT (Satisfiability):
Instance: A set $\mathcal{X}$ of n variables, a formula $\mathcal{F}$ made of m clauses over $\mathcal{X}$.
Question: Is there a truth assignment for $\mathcal{X}$ that satisfies $\mathcal{F}$?

The following problem is stated for any fixed integer $k \geqslant 1$ (though the case $k = 1$ is trivial):

Problem k-SAT (k-Satisfiability):
Instance: A set $\mathcal{X}$ of n variables, a formula $\mathcal{F}$ made of m clauses over $\mathcal{X}$, each clause containing exactly k different literals.
Question: Is there a truth assignment for $\mathcal{X}$ that satisfies $\mathcal{F}$?

Remark 2.1. *A variant of k-SAT consists in considering the instances with at most k literals instead of exactly k literals. It is easy to observe that the two statements are equivalent. Indeed, one possibility to transform a clause with less than k literals in order to obtain an equivalent clause with exactly k distinct literals proceeds as follows. Let* α_i*, for* $1 \leqslant i \leqslant k-1$*,* β *and* γ *be* $k+1$ *new variables. We add the following* $4(k-1)$ *clauses to the original ones:* $\{\overline{\alpha_i}, \beta, \gamma\}$*,* $\{\overline{\alpha_i}, \beta, \overline{\gamma}\}$*,* $\{\overline{\alpha_i}, \overline{\beta}, \gamma\}$*,* $\{\overline{\alpha_i}, \overline{\beta}, \overline{\gamma}\}$ *for* $1 \leqslant i \leqslant k-1$*; it is obvious that the only way to satisfy these clauses is to set the variables* α_i *to FALSE for* $1 \leqslant i \leqslant k-1$ *and any value to* β *and* γ*. Last, we replace any clause c containing* κ *literals with* $1 \leqslant \kappa \leqslant k-1$ *by* $c' = c \vee \alpha_1 \vee \alpha_2 ... \vee \alpha_{k-\kappa}$*. As all the variables* α_i *(*$1 \leqslant i \leqslant k-1$*) must take the Boolean value FALSE, it is clear that c and* c' *have the same Boolean value for any assignment.*

For the problems called 1-3-SAT and U-1-3-SAT below, we shall say that a formula is *1-3-satisfied* by an assignment if every clause of the formula contains exactly one literal set to TRUE by this assignment.

Problem 1-3-SAT (One-in-Three Satisfiability):
Instance: A set $\mathcal{X}$ of n variables, a formula $\mathcal{F}$ made of m clauses over $\mathcal{X}$, each clause containing exactly three different literals.
Question: Is there a truth assignment for $\mathcal{X}$ that 1-3-satisfies $\mathcal{F}$?

The last problem, $\overline{\text{SAT}}$, is obtained from SAT by considering the negation of the question set in SAT:

Problem $\overline{\text{SAT}}$ (co-Satisfiability, also noted co-SAT):
Instance: A set $\mathcal{X}$ of n variables, a formula $\mathcal{F}$ made of m clauses over $\mathcal{X}$.
Question: Is it true that no truth assignment for $\mathcal{X}$ satisfies $\mathcal{F}$?

The problems studied below, namely U-SAT, U-k-SAT ($k \geqslant 1$) and U-1-3-SAT, are obtained from the above problems SAT, k-SAT and 1-3-SAT respectivly by adding the word "unique" in the question:

Problem U-SAT (Unique Satisfiability):
Instance: A set $\mathcal{X}$ of n variables, a formula $\mathcal{F}$ made of m clauses over $\mathcal{X}$.
Question: Is there a unique truth assignment for $\mathcal{X}$ that satisfies $\mathcal{F}$?

As for k-SAT, the following problem is stated for any fixed integer $k \geqslant 1$ (though the case $k = 1$ is still trivial):

Problem U-k-SAT (Unique k-Satisfiability):
Instance: A set $\mathcal{X}$ of n variables, a formula $\mathcal{F}$ made of m clauses over $\mathcal{X}$, each clause containing exactly k different literals.
Question: Is there a unique truth assignment for $\mathcal{X}$ that satisfies $\mathcal{F}$?

Problem U-1-3-SAT (Unique One-in-Three Satisfiability):
Instance: A set $\mathcal{X}$ of n variables, a formula $\mathcal{F}$ made of m clauses over $\mathcal{X}$, each clause containing exactly three different literals.
Question: Is there a unique truth assignment for $\mathcal{X}$ that 1-3-satisfies $\mathcal{F}$?

The problem U-SAT has been studied as far back as 1982 ([7], [30]).

3. Recalls in Complexity Theory

We recall the notions of complexity that will be needed in the sequel. We refer the reader to, e.g., [1], [5], [11], [20] or [28] for more on this topic.

3.1. *Complexity classes P, NP, co-NP, NP-C*

As recalled in Section 1, a *decision problem* sets a question of the type "Given an instance I and a property $\mathcal{P}$ on I, is $\mathcal{P}$ true for I?", and has only two issues: YES or NO. The *class P* is the set of decision problems which can be solved by a *(deterministic) polynomial time algorithm*, and the *class NP* the set of problems which can be solved by a *nondeterministic polynomial time algorithm*. Quite often, the question associated with the property $\mathcal{P}$ of a problem belonging to *NP* is an existential question, *i.e.* a question of the form "Does there exist..."; it is the case for the problems studied here. From a practical point of view, a problem belongs to *NP* if we can check in polynomial time that a positive instance, *i.e.* an instance admitting the answer YES, indeed admits the answer YES, thanks to a clue (a *certificate*) provided by somebody else. The class co-*NP* is defined in a similar way as *NP*, but with checking the answer NO for negative instances (*i.e.*, instances admitting the answer NO) instead of YES for positive instances.

A *polynomial reduction* (also called *polynomial transformation*) from a decision problem D_1 into a decision problem D_2 is a transformation that maps any instance I_1 of D_1 into an instance of D_2 admitting the same answer as I_1; moreover, the complexity of the reduction must be upper-bounded by a polynomial with respect to the binary size of I_1 (all the sizes considered in the sequel are binary; as specified above, the binary size of an instance of SAT, k-SAT, 1-3-SAT, U-SAT, U-k-SAT, U-1-3-SAT and $\overline{\text{SAT}}$ can be bounded by a polynomial in $n + m$). When such a polynomial reduction from D_1 into D_2 exists, we shall write $D_1 \preceq D_2$ (the more formal notation $D_1 \preceq^p_m D_2$ is also used but, as there will be no ambiguity below, we prefer the easier notation $\preceq$). The relation $D_1 \preceq D_2$ involves that D_2 is at least as difficult as D_1 in the sense that any algorithm A_2 solving D_2 provides an algorithm solving D_1 with the same qualitative complexity (*i.e.*, up to polynomials). But for this, we must know an explicit polynomial reduction from D_1 to D_2: then it suffices to tranform any instance I_1 of D_1 into an instance I_2 of D_2 and then to apply A_2 to I_2 in order to obtain the answer admitted by I_1 (the same as for I_2). Just knowing that the relation $D_1 \preceq D_2$ is true without knowing an explicit reduction does not suffice with this respect.

A polynomial reduction φ is said to be *parsimonious* if, for decision problems in which we look for a certain entity (for satisfiability-like problems, it is the truth assignment), the number of researched entities is kept by φ: for any transformed instance I, I and $\varphi(I)$ admit the same number of solutions. When such a parsimonious reduction exists, of course it keeps uniqueness when the instance to be reduced admits a unique solution (though, for this purpose, we may also design reduction preseving unicity without being parsimonious). Below, we provide explicit parsimonious polynomial reductions between the studied problems.

S. Cook proved in [9] that SAT belongs to *NP* and is such that every other problem in *NP* can be polynomially reduced to it. Thus, in a sense, SAT is a "hardest" problem inside *NP*. Other problems share this property in *NP*; they are called *NP-complete problems*; denoting the class of *NP*-complete problems by *NP-C*, we have:

$$D \in \textit{NP-C} \Leftrightarrow (D \in \textit{NP} \text{ and } \forall\, D' \in \textit{NP}, D' \preceq D).$$

The way to show that a decision problem D is *NP*-complete is, once D is proved to be in *NP*, to choose some *NP*-complete problem D' and to polynomially reduce it to D:

$$D \in NP\text{-}C \Leftrightarrow (D \in NP \text{ and } \exists\, D' \in NP\text{-}C \text{ with } D' \preceq D).$$

From a practical viewpoint, the *NP*-completeness of a problem D implies that we do not know any polynomial algorithm in order to solve D, and that, under the assumption $P \neq NP$, which is widely believed to be true, no such algorithm exists: the time required to solve such a problem can grow exponentially with the size of the instance.

The *complement* of a decision problem D setting the question "Given I and $\mathcal{P}$, is $\mathcal{P}$ true for I?" is another decision problem, denoted $\overline{D}$ or co-D, admitting the same instances as D and setting the question "Given I and $\mathcal{P}$, is $\mathcal{P}$ false for I?". Thus, the problem called $\overline{\text{SAT}}$ above is in fact the complement problem of SAT. The class *co-NP* (respectively, *co-NP-C*, for *co-NP-complete*) is the class of the problems which are the complement of a problem in *NP* (respectively, *NP-C*).

3.2. *NP-hardness*

There exist several definitions of *NP-hardness* and of *co-NP-hardness* (which may lead to confusion, see Section 5).

The definition given by M. R. Garey and D. S. Johnson in [11], (and then reiterated in [20]) is based on *Turing reductions*. For problems which are not necessarily decision problems, there exists a *Turing reduction* from a problem π_1 to a problem π_2 if we can design an algorithm A_1 solving π_1 based on an algorithm A_2 solving π_2 as a subprogram in such a way that, if A_2 were a polynomial algorithm for π_2, then A_1 would be a polynomial algorithm for π_1; in particular, the number of calls to A_2 in order to solve an instance I_1 of π_1 must be upper-bounded by a polynomial in the size of I_1. In other terms, if we consider A_2 as an elementary operation, then π_1 would be polynomial. In this sense, π_2 is "at least as hard" as π_1. We note $\preceq_T$ the Turing reduction.

In [11], a problem π, which is not necessarily a decision problem, is *NP-hard* if there exists a *Turing reduction* from some *NP*-complete problem to π:

$$\pi \text{ is } NP\text{-hard} \Leftrightarrow \exists\, D \in NP\text{-}C \text{ with } D \preceq_T \pi.$$

We may define *co-NP-hardness* in a similar way from a co-*NP*-complete problem instead of an *NP*-complete problem but in fact, with these definitions, the notions of *NP*-hard and co-*NP*-hard coincide (see [11]). Still with these definitions, *NP*-complete problems are obviously *NP*-hard (and co-*NP*-hard), and co-*NP*-complete problems are *NP*-hard (and co-*NP*-hard) as well, since any algorithm solving a decision problem D may be performed to solve $\overline{D}$: it is indeed sufficient to consider the negation of the answer obtained for D. This definition of *NP*-hardness is often the one used in the field of combinatorial optimization (see for instance [4] or [23]).

In [29], a problem D, which is assumed to be a decision problem, is *NP-hard* (respectively, *co-NP-hard*) if there is a *polynomial* reduction from some *NP*-complete (respectively, co-*NP*-complete) problem to D:

$$D \text{ is } NP\text{-hard} \Leftrightarrow (D \text{ is a decision problem and } \exists\, D' \in NP\text{-}C \text{ with } D' \preceq D).$$

These definitions are used by several authors (see for instance [10], [15] or [37]). With these definitions, an *NP*-complete (respectively co-*NP*-complete) problem is exactly an *NP*-hard (respectively co-*NP*-hard) problem which belongs to *NP* (respectively to co-*NP*); here, *NP*-hardness and co-*NP*-hardness do not coincide if *NP* and co-*NP* are distinct.

For the problems studied here, our results show that U-SAT, U-k-SAT ($k \geqslant 3$) and U-1-3-SAT are co-*NP*-hard for both polynomial and Turing reductions, *i.e.* for both definitions. Hence they are also *NP*-hard with respect to Turing reductions, *i.e.* for the first definition. Their possible *NP*-hardness with respect to polynomial reductions, *i.e.*, for the second definition, is not known (see Section 5).

Completeness and hardness can be extended to classes other than *NP* or co-*NP*, based on the same principles. Membership to a complexity class (*P*, *NP*, co-*NP*...) provides an upper bound on the complexity of a problem (this problem is not more difficult than...), whereas a hardness result gives a lower bound (this problem is at least as difficult as...).

3.3. *Other complexity classes*

We now introduce other complexity classes, less usual but useful for the sequel. Some of them received different notations, or were shown to be the same as classes defined differently, hence the multiple notations (see, e.g., [1] or [14]).

The class L^{NP} [20] (also denoted by $P^{NP[O(\log n)]}$ or Θ_2 or still $P^{||NP}$...; as the notation L^{NP} is more intuitive, we will keep it in the following) contains the decision problems which can be solved by applying, with a number of calls which is logarithmic with respect to the size of the instance, a subprogram able to solve an appropriate problem in *NP* (usually, an *NP*-complete problem). The class P^{NP} [20] is defined similarly but with a number of calls which is polynomial; this class is also denoted by Δ_2 (or $\Delta_2 P$), which is the complexity class located just above *NP* and co-*NP* inside the polynomial hierarchy.

The class *DP* [30] (or also D^P or DIF^P [7]), for *Difference Polynomial Time*, is defined as the set of languages (or problems) $\mathcal{L}$ such that there are two languages $\mathcal{L}_1 \in NP$ and $\mathcal{L}_2 \in$ co-*NP* satisfying $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$. It is the same as the class denoted by BH_2, which is the complexity class located just above *NP* and co-*NP* inside the Boolean hierarchy (see [1] or [20]). This class is not to be confused with $NP \cap$ co-*NP* (see the warning in, e.g., [28]); actually, *DP* contains $NP \cup$ co-*NP* and is contained in L^{NP} (see Figure 1):

$$NP \cup \text{co-}NP \subseteq DP \subseteq L^{NP} \subseteq P^{NP}.$$

For instance, U-SAT belongs to *DP* ([30]; see also [28]). To show this result, one has to exhibit two languages, $\mathcal{L}_1 \in NP$ and $\mathcal{L}_2 \in$ co-*NP*, such that the set of YES instances of U-SAT is $\mathcal{L}_1 \cap \mathcal{L}_2$. For this, take $\mathcal{L}_1 = \{$formulas (or sets of clauses) such that there is at least one truth assignment satisfying them$\}$ and $\mathcal{L}_2 = \{$formulas (or sets of clauses) such that there is at most one assignment satisfying them$\}$. The same proof can be applied *mutatis mutandis* to U-1-3-SAT and, for every integer $k \geqslant 3$, to U-k-SAT (alternatively, we can use the equivalence between these problems established in Section 4).

By applying the same definition as for *NP* or co-*NP*, we may define *DP*-complete problems as the most difficult problems inside *DP*. More precisely, a decision problem

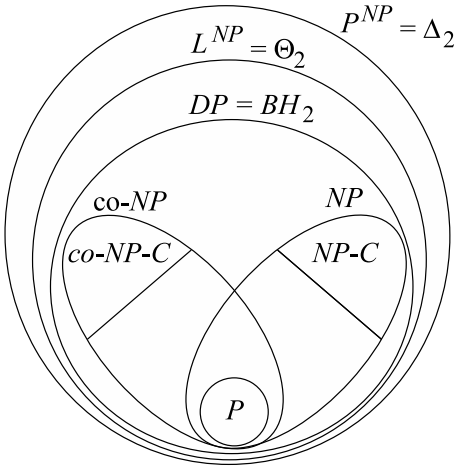

Figure 1. Some classes of complexity (under the assumption $NP \neq$ co-NP).

D is DP-complete if D belongs to DP and any problem belonging to DP can be polynomially reduced to D.

Finally, the class *US*, for *Unique Solution*, has been defined by A. Blass and Y. Gurevich in [7] as the class of decision problems solvable by a nondeterministic polynomial-time Turing machine such that the answer is "YES" if and only if exactly one computation path accepts, *i.e.*, when the entity that we are looking for is unique. As for the other classes, we may define the *US*-complete problems as the most difficult problems inside the class *US*. More precisely, with *US-C* denoting the set of *US*-complete problems:

$$D \in US\text{-}C \Leftrightarrow (D \in \ US \text{ and } \forall\, D' \in US, D' \preceq D).$$

The class *US* contains co-*NP* (see [1] and [7]). Moreover, if we consider that the uniqueness of the searched entity is the same as the existence of this entity together with the fact there are not several such entities, it appears that any problem of *US* belongs also to *DP*, as noticed by A. Blass and Y. Gurevich in [7] (see Figure 2):

$$\text{co-}NP \subseteq US \subseteq DP.$$

On the other hand, we do not know whether *US* contains *NP* (see [7] for a discussion about this).

4. Complexity results

As recalled at the end of Section 2, U-SAT has been studied as far back as 1982 ([7], [30]). It is not difficult to show the relation $\overline{SAT} \preceq$ U-SAT (see [7]), which proves that U-SAT is co-*NP*-hard (for both definitions and also *NP*-hard with respect to the first definition). In [7], A. Blass and Y. Gurevich stated a more accurate result: U-SAT is *US*-complete. On the other hand, U-1-SAT trivially belongs to P and it is proved in [12] that U-2-SAT also belongs to P (see also [2] for a polynomial method solving 2-SAT). In [8], problems also called U-k-SAT are studied, and more particularly

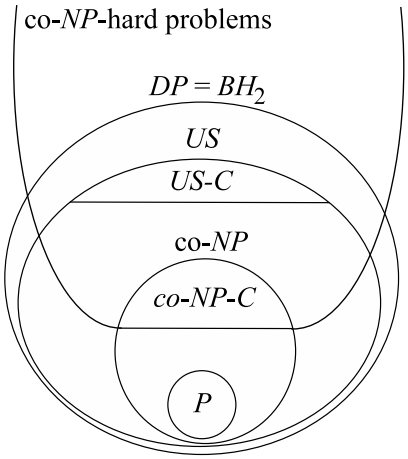

Figure 2. Classes of complexity *DP* and *US* (under the assumption $US \neq$ co-*NP*).

U-3-SAT, but it appears that they are versions where the given set of clauses has zero or one solution, which makes quite a difference with our problems.

The aim of this section is to show (Theorem 4.6) that U-k-SAT for $k \geqslant 3$ and U-1-3-SAT are also *US*-complete. For this, we explicit polynomial reductions establishing the following relations:

- U-SAT $\preceq$ U-3-SAT (Subsection 4.1);
- U-3-SAT $\preceq$ U-1-3-SAT $\preceq$ U-3-SAT (Subsection 4.2);
- U-$(k-1)$-SAT $\preceq$ U-k-SAT for $k \geqslant 4$ (Subsection 4.3);
- U-k-SAT $\preceq$ U-SAT for $k \geqslant 3$ (Subsection 4.3).

4.1. *U-SAT $\preceq$ U-3-SAT*

First, note that the usual proof of the complexity of 3-SAT from SAT ([9]; see also [11] and [22]) is not parsimonious and does not keep the potential uniqueness of a truth assignment.

Hence another proof, nonetheless based on the classic one.

Theorem 4.1. *There exists a polynomial reduction from U-SAT to U-3-SAT:*
U-SAT $\preceq$ U-3-SAT.

Proof. We consider any instance of U-SAT with n variables x_i and m clauses c_j.

- First, we keep all the variables x_i, $1 \leqslant i \leqslant n$.
- Then we create three new variables α, β, γ and seven clauses $\{\alpha, \beta, \gamma\}$, $\{\alpha, \beta, \overline{\gamma}\}$, $\{\alpha, \overline{\beta}, \gamma\}$, $\{\alpha, \overline{\beta}, \overline{\gamma}\}$, $\{\overline{\alpha}, \beta, \gamma\}$, $\{\overline{\alpha}, \beta, \overline{\gamma}\}$, and $\{\overline{\alpha}, \overline{\beta}, \gamma\}$. Note that these clauses can be simultaneously satisfied if and only if α, β and γ are set to TRUE. Only the variables α and β will be used in the sequel in order to transform the clauses c_j; the role of γ is just to allow the creation of the seven previous clauses with sizes equal to 3 and forcing the values of α and β.
- Let c_j $(1 \leqslant j \leqslant m)$ be any clause of the instance we start from, and let κ_j denote the size of c_j, with $\kappa_j \geqslant 1$: $c_j = \{\ell_{j,1}, \ldots, \ell_{j,\kappa_j}\}$. We transform c_j according to the value of its size κ_j.

(a) For $\kappa_j = 1$, we create the clause $\{\ell_{j,1}, \overline{\alpha}, \overline{\beta}\}$.
(b) For $\kappa_j = 2$, we create the clause $\{\ell_{j,1}, \ell_{j,2}, \overline{\alpha}\}$.
(c) For $\kappa_j = 3$, we keep the clause $\{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}\}$.
(d) For $\kappa_j > 3$, we create a new variable $y_{j,1}$ and the clauses $\{y_{j,1}, \ell_{j,3}, \ldots, \ell_{j,\kappa_j}\}$, $\{y_{j,1}, \overline{\ell_{j,1}}, \overline{\alpha}\}$, $\{y_{j,1}, \overline{\ell_{j,2}}, \overline{\alpha}\}$, and $\{\overline{y_{j,1}}, \ell_{j,1}, \ell_{j,2}\}$. As the first clause has size $\kappa_j - 1$, we iterate this process until this clause has size 3: at each step we create one variable of type $y_{j,h}$ for an appropriate integer h (related to the number of the current iteration) and four clauses, three of them (at least) with size 3. At the end, we keep only the clauses of size 3. More formally, in this step, we create $\kappa_j - 3$ new variables $y_{j,i}$ $(1 \leqslant i \leqslant \kappa_j - 3)$ and the following $3\kappa_j - 8$ clauses:
- for $1 \leqslant i \leqslant \kappa_j - 3$, $\{y_{j,i}, \overline{\ell_{j,i+1}}, \overline{\alpha}\}$
- for $2 \leqslant i \leqslant \kappa_j - 3$, $\{y_{j,i}, \overline{y_{j,i-1}}, \overline{\alpha}\}$
- for $2 \leqslant i \leqslant \kappa_j - 3$, $\{\overline{y_{j,i}}, y_{j,i-1}, \ell_{j,i+1}\}$
- $\{y_{j,1}, \overline{\ell_{j,1}}, \overline{\alpha}\}$
- $\{\overline{y_{j,1}}, \ell_{j,1}, \ell_{j,2}\}$
- $\{y_{j,\kappa_j-3}, \ell_{j,\kappa_j-1}, \ell_{j,\kappa_j}\}$.

Example. Assume that we have $\kappa_j = 5$ and $c_j = \{x_{j,1}, \overline{x_{j,2}}, \overline{x_{j,3}}, x_{j,4}, x_{j,5}\}$. First, we create the clauses $\{y_{j,1}, \overline{x_{j,3}}, x_{j,4}, x_{j,5}\}$, which will not be kept, $\{y_{j,1}, \overline{x_{j,1}}, \overline{\alpha}\}$, $\{y_{j,1}, x_{j,2}, \overline{\alpha}\}$, and $\{\overline{y_{j,1}}, x_{j,1}, \overline{x_{j,2}}\}$. Then, in order to transform $\{y_{j,1}, \overline{x_{j,3}}, x_{j,4}, x_{j,5}\}$, we create $\{y_{j,2}, x_{j,4}, x_{j,5}\}$, $\{y_{j,2}, \overline{y_{j,1}}, \overline{\alpha}\}$, $\{y_{j,2}, x_{j,3}, \overline{\alpha}\}$ and $\{\overline{y_{j,2}}, y_{j,1}, \overline{x}_{j,3}\}$. At the end, c_j has been transformed into seven clauses of size 3, namely: $\{y_{j,1}, \overline{x_{j,1}}, \overline{\alpha}\}$, $\{y_{j,1}, x_{j,2}, \overline{\alpha}\}$, $\{\overline{y_{j,1}}, x_{j,1}, \overline{x_{j,2}}\}$, $\{y_{j,2}, x_{j,4}, x_{j,5}\}$, $\{y_{j,2}, \overline{y_{j,1}}, \overline{\alpha}\}$, $\{y_{j,2}, x_{j,3}, \overline{\alpha}\}$ and $\{\overline{y_{j,2}}, y_{j,1}, \overline{x}_{j,3}\}$ thanks to the extra variables $y_{j,1}$ and $y_{j,2}$.

We now prove that this reduction keeps the answer. We first **(1)** consider the answer YES, then **(2)** the answer NO.

(1) Assume that the answer to the instance of U-SAT is YES: there is a unique way of satisfying all the clauses of the instance of U-SAT. We keep the same truth assignment for the variables x_i, $1 \leqslant i \leqslant n$; we set $\alpha = \beta = \gamma =$ TRUE; for all $j \in \{1, \ldots, m\}$ with $\kappa_j > 3$, we set $y_{j,1} =$ TRUE if and only if at least one of $\ell_{j,1}$ and $\ell_{j,2}$ is equal to TRUE, $y_{j,2} =$ TRUE if and only if at least one of $y_{j,1}$ and $\ell_{j,3}$ is equal to TRUE, and so on until y_{j,κ_j-3}: for $2 \leqslant i \leqslant \kappa_j - 3$, $y_{j,i} =$ TRUE if and only if at least one of $y_{j,i-1}$ and $\ell_{j,i+1}$ is equal to TRUE. It is quite straightforward to check that this new assignment satisfies the set of clauses that we have just constructed for U-3-SAT.

Is it the only way? Assume that two assignments, $\mathcal{A}_1$ and $\mathcal{A}_2$, satisfy the instance of U-3-SAT.

First, we prove that any assignment $\mathcal{A}$ which satisfies the instance of U-3-SAT also satisfies, when restricted to the variables x_i, the instance of U-SAT.

(a) Consider a clause of size 1: $c_j = \{\ell_{j,1}\}$. Because we necessarily have $\mathcal{A}(\alpha) = \mathcal{A}(\beta) =$ TRUE, $\{\ell_{j,1}, \overline{\alpha}, \overline{\beta}\}$ is necessarily satisfied by assigning the value TRUE to $\ell_{j,1}$, thus satisfying c_j.

(b) Consider a clause of size 2: $c_j = \{\ell_{j,1}, \ell_{j,2}\}$. Because we necessarily have $\mathcal{A}(\alpha) =$ TRUE, if $\{\ell_{j,1}, \ell_{j,2}, \overline{\alpha}\}$ is satisfied by the values given to $\ell_{j,1}$ and $\ell_{j,2}$, then this is also true for c_j.

(c) The clauses of size 3 in U-SAT need no explanation.

(d) Last, consider a clause $c_j = \{\ell_{j,1}, \ldots, \ell_{j,\kappa_j}\}$ in U-SAT with $\kappa_j > 3$ which would not be satisfied by the restriction of $\mathcal{A}$ to the initial variables x_i $(1 \leqslant i \leqslant n)$: $\forall\ 1 \leqslant i \leqslant \kappa_j$, $\mathcal{A}(\ell_{j,i}) =$ FALSE. Then we show by induction on i that all the variables $y_{j,i}$ must take the value FALSE: $\forall\, 1 \leqslant i \leqslant \kappa_j$, $\mathcal{A}(y_{j,i}) =$ FALSE. Indeed, because the

clause $\{\overline{y_{j,1}}, \ell_{j,1}, \ell_{j,2}\}$ in U-3-SAT is satisfied and since all the variables $\ell_{j,i}$ are set to FALSE, we must have $\mathcal{A}(y_{j,1}) = \text{FALSE}$; similarly, because of the clause $\{\overline{y_{j,2}}, y_{j,1}, \ell_{j,3}\}$ in U-3-SAT, we must have $\mathcal{A}(y_{j,2}) = \text{FALSE}$; and so on until the last new variable of type y: $\mathcal{A}(y_{j,\kappa_j-3}) = \text{FALSE}$. More formally, assume, for the induction hypothesis at the order i, that, for some i with $i < \kappa_j$, we have $\mathcal{A}(y_{j,i}) = \text{FALSE}$; observe that it is the case for $i = 1$, as shown above; then, because of the clause $\{\overline{y_{j,i+1}}, y_{j,i}, \ell_{j,i+2}\}$ in U-3-SAT, we must have $\mathcal{A}(y_{j,i+1}) = \text{FALSE}$: the induction hypothesis is restored at the order $i+1$. But then the clause $\{y_{j,\kappa_j-3}, \ell_{j,\kappa_j-1}, \ell_{j,\kappa_j}\}$ in U-3-SAT is not satisfied, a contradiction.

Therefore, the two assignments $\mathcal{A}_1$ and $\mathcal{A}_2$ give the same values to the variables x_i, $1 \leqslant i \leqslant n$, for otherwise we would have two ways of satisfying the instance of U-SAT, which would contradict our assumption on uniqueness. Thus it is the same for all the literals $\ell_{j,i}$ involved in the clauses: $\forall\ 1 \leqslant j \leqslant m$, $\forall\ 1 \leqslant i \leqslant \kappa_j$, $\mathcal{A}_1(\ell_{j,i}) = \mathcal{A}_2(\ell_{j,i})$. Since the values of α, β, γ are forced, $\mathcal{A}_1$ and $\mathcal{A}_2$ can differ only on the variables of type y.

Suppose that they differ on $y_{j,1}$ for some j, with, say, $\mathcal{A}_1(y_{j,1}) = \text{TRUE}$ and $\mathcal{A}_2(y_{j,1}) = \text{FALSE}$. Then, together with the fact $\mathcal{A}_1(\alpha) = \mathcal{A}_2(\alpha) = \text{TRUE}$, the two clauses $\{y_{j,1}, \overline{\ell_{j,1}}, \overline{\alpha}\}$, $\{y_{j,1}, \overline{\ell_{j,2}}, \overline{\alpha}\}$ yield $\mathcal{A}_2(\ell_{j,1}) = \mathcal{A}_2(\ell_{j,2}) = \text{FALSE}$, while the clause $\{\overline{y_{j,1}}, \ell_{j,1}, \ell_{j,2}\}$ yields $\mathcal{A}_1(\ell_{j,1}) = \text{TRUE}$ or $\mathcal{A}_1(\ell_{j,2}) = \text{TRUE}$: $\mathcal{A}_1$ and $\mathcal{A}_2$ differ on $\ell_{j,1}$ or $\ell_{j,2}$, a contradiction. So $\mathcal{A}_1(y_{j,1}) = \mathcal{A}_2(y_{j,1})$. Because of this, $\mathcal{A}_1$ and $\mathcal{A}_2$ must assign the same value to $y_{j,2}$ too: $\mathcal{A}_1(y_{j,2}) = \mathcal{A}_2(y_{j,2})$. Indeed, since α must be set to TRUE, if $y_{j,1}$ receives the value TRUE, then $y_{j,2}$ must also receive the value TRUE because of the clause $\{y_{j,2}, \overline{y_{j,1}}, \overline{\alpha}\}$; if $y_{j,1}$ receives the value FALSE, then $y_{j,2}$ must receive the same value as $\ell_{j,3}$ because of the clauses $\{y_{j,2}, \overline{\ell_{j,3}}, \overline{\alpha}\}$ and $\{\overline{y_{j,2}}, y_{j,1}, \ell_{j,3}\}$; so the value assigned to $y_{j,2}$ is forced and thus must be the same by $\mathcal{A}_1$ and by $\mathcal{A}_2$. By induction, we show that it is the same for the other variables $y_{j,i}$. Indeed, if, for some index $i < \kappa_j - 3$, $y_{j,i}$ is set to TRUE, then it must be the same for $y_{j,i+1}$ because of the clause $\{y_{j,i+1}, \overline{y_{j,i}}, \overline{\alpha}\}$; otherwise the value taken by $y_{j,i+1}$ must be the same as the one assigned to $\ell_{j,i+2}$ because of the clauses $\{y_{j,i+1}, \overline{\ell_{j,i+2}}, \overline{\alpha}\}$ and $\{\overline{y_{j,i+1}}, y_{j,i}, \ell_{j,i+2}\}$; so the value assigned to $y_{j,i+1}$ is also forced and thus must be the same by $\mathcal{A}_1$ and by $\mathcal{A}_2$.

In conclusion, $\mathcal{A}_1$ and $\mathcal{A}_2$ assign the same values to all the variables $y_{j,i}$. So $\mathcal{A}_1$ and $\mathcal{A}_2$ are the same and there is only one solution for the instance of U-3-SAT.

(2) If the answer to the instance of U-SAT is NO, then either there are at least two solutions, which give at least two solutions to U-3-SAT, as previously shown, or there is no solution to U-SAT, in which case there is no solution to U-3-SAT, since otherwise one solution to U-3-SAT would also give, by restriction to the variables x_i, one solution to U-SAT, as seen above. Thus the answer to the instance of U-3-SAT is also NO.

In order to conclude, note that the complexity of the reduction is polynomially related to the number of variables and clauses constructed by the reduction. For any j with $1 \leqslant j \leqslant m$, c_j contains at most n literals (since any clause is assumed not to contain several times a same literal nor a variable and its complement simultaneously): $\kappa_j \leqslant n$. In addition to the variables x_i ($1 \leqslant i \leqslant n$) and to α, β and γ, the transformation of c_j with $\kappa_j > 3$ requires $\kappa_j - 3$ extra variables $\ell_{j,i}$ and constructs $3(\kappa_j - 3) + 1$ new clauses; for a clause c_j with $\kappa_j \leqslant 3$, we construct only one clause, without additional variable. In all cases, the number of new clauses per old clause can be bounded by $3n$. Thus, the total number of variables can be upper-bounded by $n+3+m(n-3)$ and the number of clauses by $3nm$. Obviously, these two quantities can be upper-bounded by a polynomial of $n+m$, hence the polynomiality of the transformation, which completes

the proof. □

Remark 4.1. *Though it is not the main point here (we just need a transformation keeping uniqueness), it is easy to see that the reduction provided in the proof of Theorem 4.1 is in fact parsimonious. Indeed, the proof shows that the Boolean values assigned to the extra variables α, β, γ and $y_{j,i}$ are forced as soon as the Boolean values of the variables x_i are fixed. Hence a one-to-one correspondance between the assignments for any instance of U-SAT and the assignments for the corresponding instance of U-3-SAT.*

4.2. *U-3-SAT* $\preceq$ *U-1-3-SAT* $\preceq$ *U-3-SAT*

We now consider the problem U-1-3-SAT. We first recall the poynomial reduction given by T. J. Schaefer (see [33] or [11]) in order to transform 3-SAT into 1-3-SAT and then we show that this reduction keeps uniqueness.

Consider an instance of 3-SAT. For each clause $c_j = \{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}\}$ of this instance, T. J. Schaefer creates, for the instance of 1-3-SAT, six variables:
- a_j, b_j, d_j, e_j, f_j and g_j

and the following five clauses (one containing the Boolean constant FALSE):
- $c_{j,1} = \{\ell_{j,1},\ a_j,\ e_j\}$
- $c_{j,2} = \{\ell_{j,2},\ b_j,\ e_j\}$
- $c_{j,3} = \{a_j,\ b_j,\ f_j\}$
- $c_{j,4} = \{d_j,\ e_j,\ g_j\}$
- $c_{j,5} = \{\ell_{j,3},\ d_j,\ \text{FALSE}\}$.

The last clause, $c_{j,5}$, does not fit the form specified above for 1-3-SAT, since each clause should contain three literals and no Boolean constant. So we adapt T. J. Schaefer's reduction in order to avoid the constant FALSE. For this, we introduce three new variables α, β, γ, which are added to the previous variables, and three new clauses $\{\alpha, \beta, \gamma\}$, $\{\alpha, \overline{\beta}, \overline{\gamma}\}$ and $\{\overline{\alpha}, \overline{\beta}, \gamma\}$, which are added to the $5m$ clauses $c_{j,i}$. It is then straightforward to check that there is one and only one way of 1-3-satisfying the new clauses: one must set $\alpha = \text{FALSE}$, $\beta = \text{TRUE}$, $\gamma = \text{FALSE}$. Thus each clause $c_{j,5} = \{\ell_{j,3}, d_j, \text{FALSE}\}$, $1 \leqslant j \leqslant m$, can be replaced by $c_{j,5} = \{\ell_{j,3}, d_j, \alpha\}$, in order to obtain a set of clauses of size 3 which can be 1-3-satisfied if and only if there is a truth assignment satisfying the clauses c_j, $1 \leqslant j \leqslant m$, from which we started. This is what we do in the sequel and thus we assume that the clauses $c_{j,5}$ $(1 \leqslant j \leqslant m)$ has the form $\{\ell_{j,3}, d_j, \alpha\}$.

Now, note that if $\ell_{j,1} = \ell_{j,2} = \ell_{j,3} = \text{FALSE}$, then there is no truth assignment for a_j, b_j, d_j, e_j, f_j and g_j 1-3-satisfying the five clauses $c_{j,i}$, $1 \leqslant i \leqslant 5$. Indeed, the clause $c_{j,5}$ implies $d_j = \text{TRUE}$, which implies $e_j = \text{FALSE}$ by the clause $c_{j,4}$, which in turn implies $a_j = b_j = \text{TRUE}$ by the clauses $c_{j,1}$ and $c_{j,2}$, and then $c_{j,3}$ contains at least two true literals.

Then, assume that c_j is satisfied: at least one of the three literals $\ell_{j,1}$, $\ell_{j,2}$ or $\ell_{j,3}$ is set to TRUE, which leads to seven possibilities. Table 1 shows, for each one of these possibilities, which truth values should be assigned to the extra variables a_j, b_j, d_j, e_j, f_j and g_j in order to 1-3-satisfy the clauses $c_{j,i}$ for $1 \leqslant i \leqslant 5$, while Table 2 specifies, still for these seven possibilities, which literal 1-3-satisfies each clause $c_{j,i}$ for $1 \leqslant i \leqslant 5$.

From all this, we can conclude that there is a truth assignment for $\mathcal{X}$ satisfying simultaneously the m clauses that we started from if and only if there is a truth assignment for $\mathcal{X} \cup \{a_j, b_j, d_j, e_j, f_j, g_j : 1 \leqslant j \leqslant m\}$ such that each of the $5m$ clauses $c_{j,i}$,

	$\ell_{j,1}$	$\ell_{j,2}$	$\ell_{j,3}$	a_j	b_j	d_j	e_j	f_j	g_j
1	T	T	T	F	F	F	F	T	T
2	T	T	F	F	F	T	F	T	F
3	T	F	T	F	T	F	F	F	T
4	F	T	T	T	F	F	F	F	T
5	T	F	F	F	T	T	F	F	F
6	F	T	F	T	F	T	F	F	F
7	F	F	T	F	F	F	T	T	F

Table 1. The seven possibilities to 1-3-satisfy the clauses $c_{j,i}$ for $1 \leqslant i \leqslant 5$ (T stands for TRUE and F for FALSE).

	$c_{j,1}$	$c_{j,2}$	$c_{j,3}$	$c_{j,4}$	$c_{j,5}$
1	$\ell_{j,1}$	$\ell_{j,2}$	f_j	g_j	$\ell_{j,3}$
2	$\ell_{j,1}$	$\ell_{j,2}$	f_j	d_j	d_j
3	$\ell_{j,1}$	b_j	b_j	g_j	$\ell_{j,3}$
4	a_j	$\ell_{j,2}$	a_j	g_j	$\ell_{j,3}$
5	$\ell_{j,1}$	b_j	b_j	d_j	d_j
6	a_j	$\ell_{j,2}$	a_j	d_j	d_j
7	e_j	e_j	f_j	e_j	$\ell_{j,3}$

Table 2. The literals 1-3-satisfying the clauses $c_{j,i}$ for $1 \leqslant i \leqslant 5$.

$1 \leqslant j \leqslant m$ and $1 \leqslant i \leqslant 5$, is 1-3-satisfied. As moreover the reduction is polynomial, this is sufficient to prove that 1-3-SAT is *NP*-complete.

In fact, we can show that T. J. Schaefer's reduction is parsimonious and thus leads to Theorem 4.2 below:

Theorem 4.2. *There exists a polynomial reduction from U-3-SAT to U-1-3-SAT:* $U\text{-}3\text{-}SAT \preceq U\text{-}1\text{-}3\text{-}SAT$.

Proof. Let us show that T. J. Schaefer's reduction is parsimonious. For this, observe that, as nocited above, we must set α to FALSE, β to TRUE and γ to FALSE in order to 1-3-satisfy the three clauses involving these variables and only them. Moreover, once the values of ℓ_1, ℓ_2, ℓ_3 are given, there is *only one way* (the one provided by Table 1) to extend this truth assignment to the new six variables in such a way that the new five clauses are 1-3-satisfied. Indeed, as α must be set to FALSE, note the equality $d_j = \overline{\ell_{j,3}}$ because of clause $c_{j,5}$; once a_j, b_j, d_j and e_j have been assigned a value, then f_j and g_j are forced because of clauses $c_{j,3}$ and $c_{j,4}$ respectively. More precisely, let us consider the seven cases.

- Line 1. For $\ell_{j,1} = \ell_{j,2} = \ell_{j,3} =$ TRUE, then a_j, b_j, d_j, e_j must be FALSE because of clauses $c_{j,1}$, $c_{j,2}$, $c_{j,5}$, while f_j and g_j must be TRUE because of clauses $c_{j,3}$ and $c_{j,4}$.
- Line 2. For $\ell_{j,1} = \ell_{j,2} =$ TRUE, $\ell_{j,3} =$ FALSE, then a_j, b_j, e_j must be FALSE because of clauses $c_{j,1}$, $c_{j,2}$, $d_j = \overline{\ell_{j,3}} =$ TRUE, while f_j and g_j must respectively be TRUE and FALSE because of clauses $c_{j,3}$ and $c_{j,4}$ respectively.
- Line 3. For $\ell_{j,1} =$ TRUE, $\ell_{j,2} =$ FALSE, $\ell_{j,3} =$ TRUE, then a_j and e_j must be FALSE because of $c_{j,1}$, then b_j must be TRUE because of $c_{j,2}$, $d_j = \overline{\ell_{j,3}} =$ FALSE, while f_j and g_j must respectively be FALSE and TRUE because of $c_{j,3}$ and $c_{j,4}$

respectively.

- Line 4. For $\ell_{j,1}$ = FALSE, $\ell_{j,2}$ = TRUE, $\ell_{j,3}$ = TRUE, then b_j and e_j must be FALSE because of $c_{j,2}$, then a_j must be TRUE because of $c_{j,1}$, $d_j = \overline{\ell_{j,3}}$ = FALSE, while f_j and g_j must respectively be FALSE and TRUE because of $c_{j,3}$ and $c_{j,4}$ respectively.
- Line 5. For $\ell_{j,1}$ = TRUE, $\ell_{j,2}$ = FALSE, $\ell_{j,3}$ = TRUE, then a_j and e_j must be FALSE because of $c_{j,1}$, then b_j must be TRUE because of $c_{j,2}$, $d_j = \overline{\ell_{j,3}}$ = TRUE, while f_j and g_j must be FALSE because of $c_{j,3}$ and $c_{j,4}$ respectively.
- Line 6. For $\ell_{j,1}$ = FALSE, $\ell_{j,2}$ = TRUE, $\ell_{j,3}$ = FALSE, then b_j and e_j must be FALSE because of $c_{j,2}$, then a_j must be TRUE because of $c_{j,1}$, $d_j = \overline{\ell_{j,3}}$ = TRUE, while f_j and g_j must be FALSE because of $c_{j,3}$ and $c_{j,4}$ respectively.
- Line 7. For $\ell_{j,1} = \ell_{j,2}$ = FALSE and $\ell_{j,3}$ = TRUE, then $d_j = \overline{\ell_{j,3}}$ = FALSE. If we set a_j = TRUE, then e_j = FALSE because of $c_{j,1}$, b_j = TRUE because of $c_{j,2}$, and $c_{j,3}$ contains at least two true literals. So we must have a_j = FALSE, and the rest is forced: e_j = TRUE because of $c_{j,1}$ and b_j = FALSE because of $c_{j,2}$.

So, there is a one-to-one correspondance between the solutions that satisfy the clauses c_j (when such solutions exist) and the ones that 1-3-satisfy the associated clauses $c_{j,i}$. Thus the number of solutions for the instance of 3-SAT we started from is equal to the number of solutions for the constructed instance of 1-3-SAT: the reduction is parsimonious. In particular, there is a unique solution to the instance of 3-SAT if and only if there is a unique solution to the instance of 1-3-SAT, *i.e.*, the answer to the initial instance, now seen as an instance of U-3-SAT, is YES if and only if the answer to the corresponding instance for U-1-3-SAT is YES.

Moreover, as noticed above, the reduction is polynomial.

Hence the result: U-3-SAT $\preceq$ U-1-3-SAT. □

Theorem 4.3. *There exists a polynomial reduction from U-1-3-SAT to U-3-SAT:* U-1-3-SAT $\preceq$ U-3-SAT.

Proof. Consider any instance I of U-1-3-SAT with n variables x_i, $1 \leqslant i \leqslant n$, and m clauses c_j, $1 \leqslant j \leqslant m$, of the form $\{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}\}$.

We keep all the variables x_i, $1 \leqslant i \leqslant n$ and we create three new variables α, β, γ with the seven clauses $\{\alpha, \beta, \gamma\}$, $\{\alpha, \beta, \overline{\gamma}\}$, $\{\alpha, \overline{\beta}, \gamma\}$, $\{\alpha, \overline{\beta}, \overline{\gamma}\}$, $\{\overline{\alpha}, \beta, \gamma\}$, $\{\overline{\alpha}, \beta, \overline{\gamma}\}$, and $\{\overline{\alpha}, \overline{\beta}, \gamma\}$. Obviously, these clauses can be simultaneously satisfied if and only if we set α, β and γ to TRUE. In the sequel, only α will be used in the other clauses.

Let $c_j = \{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}\}$ be any clause of I, $1 \leqslant j \leqslant m$. We keep c_j and add three new clauses: $\{\overline{\ell_{j,1}}, \overline{\ell_{j,2}}, \overline{\alpha}\}$, $\{\overline{\ell_{j,1}}, \overline{\ell_{j,3}}, \overline{\alpha}\}$, $\{\overline{\ell_{j,2}}, \overline{\ell_{j,3}}, \overline{\alpha}\}$. Let I' be the instance of 3-SAT that we obtain.

Assume first that the answer for I is YES: there is exactly one truth assignment, $\mathcal{A}_1$, 1-3-satisfying I. Keeping the same assignment for the variables x_i and setting $\mathcal{A}_1(\alpha) = \mathcal{A}_1(\beta) = \mathcal{A}_1(\gamma)$ = TRUE, we obviously satisfy all the clauses of I'. Is it possible to have a second assignment, $\mathcal{A}_2$, that satisfies I'? If yes, we have necessarily $\mathcal{A}_2(\alpha) = \mathcal{A}_2(\beta) = \mathcal{A}_2(\gamma)$ = TRUE, as for $\mathcal{A}_1$. Then, because of the clauses $\{\overline{\ell_{j,1}}, \overline{\ell_{j,2}}, \overline{\alpha}\}$, $\{\overline{\ell_{j,1}}, \overline{\ell_{j,3}}, \overline{\alpha}\}$, $\{\overline{\ell_{j,2}}, \overline{\ell_{j,3}}, \overline{\alpha}\}$, at most one of the three literals $\ell_{j,1}$, $\ell_{j,2}$ and $\ell_{j,3}$ can be set to TRUE by $\mathcal{A}_2$, but also at least one, because of the clause $\{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}\}$. So $\mathcal{A}_2$, when restricted to the variables x_i, also 1-3-satisfies the instance I of U-1-3-SAT, and $\mathcal{A}_2 \neq \mathcal{A}_1$ is impossible. So the answer for I' is also YES.

Assume now that the answer for I is NO. This can happen for two reasons. If there is no assignment 1-3-satisfying I, then there is no assignment satisfying the instance I'

of U-3-SAT, since such an assignment, when restricted to the variables x_i, would be an assignment 1-3-satisfying I, as we have already seen. If there are several assignments 1-3-satisfying I, then there are also several assignments satisfying I' for the same reason as before. So the answer for I' is also NO.

In conclusion, there is a YES answer for the instance of U-1-3-SAT if and only if there is a YES answer for the corresponding instance of U-3-SAT.

Moreover, the reduction is clearly polynomial with respect to the size of the instance I of U-1-3-SAT.

Hence the result: U-1-3-SAT $\preceq$ U-3-SAT. □

Remark 4.2. *Once again, the reduction from 1-3-SAT to 3-SAT depicted in the proof or Theorem 4.3 is parsimonious, which provides a little stronger result.*

4.3. *U-(k − 1)-SAT ⪯ U-k-SAT ⪯ U-SAT*

In this subsection, we consider U-k-SAT for $k \geqslant 3$ and, for $k \geqslant 4$, we prove the relation U-$(k-1)$-SAT $\preceq$ U-k-SAT.

Theorem 4.4. *For every integer $k \geqslant 4$, there exists a polynomial reduction from U-$(k-1)$-SAT to U-k-SAT:*

$$U\text{-}(k-1)\text{-}SAT \preceq U\text{-}k\text{-}SAT.$$

Proof. Consider any instance I of U-$(k-1)$-SAT with n variables x_i $(1 \leqslant i \leqslant n)$ and m clauses c_j $(1 \leqslant j \leqslant m)$ of size $k-1$.

We keep all the variables x_i of I $(1 \leqslant i \leqslant n)$.

We introduce k new variables $\alpha_1, \ldots, \alpha_k$ and $2^k - 1$ clauses: these are all the clauses of size k containing either α_1 or $\overline{\alpha_1}$, ..., α_k or $\overline{\alpha_k}$, with the exception of $\{\overline{\alpha_1}, \ldots, \overline{\alpha_k}\}$; since k is fixed, the number of clauses is a constant, so the transformation is polynomial with respect to the size of the instance which we start from. Note that these clauses can be simultaneously satisfied if and only if we set α_1, α_2, ..., α_k to TRUE. When building other clauses of size k, we shall use only α_1; the aim of the other variables $\alpha_2, \ldots, \alpha_k$ is simply to have the right size for the clauses.

The reduction is simple: besides the $2^k - 1$ aforementioned clauses, each clause $c_j = \{\ell_{j,1}, \ldots, \ell_{j,k-1}\}$ of I is transformed into $\{\ell_{j,1}, \ldots, \ell_{j,k-1}, \overline{\alpha}_1\}$. It is obvious that the answer for I is YES if and only if the answer is also YES for the resulting instance of U-k-SAT.

Moreover, as said above, the reduction is polynomial because k is fixed. Hence the result. □

We now consider the reduction from U-k-SAT to U-SAT.

Theorem 4.5. *For every integer $k \geqslant 3$, there exists a polynomial reduction from U-k-SAT to U-SAT:*

$$U\text{-}k\text{-}SAT \preceq U\text{-}SAT.$$

Proof. Simply use the identity reduction. □

Remark 4.3. *Here also, the reductions given in the proofs of Theorems 4.4 and 4.5 are in fact parsimonious.*

4.4. *US-completeness for U-Satisfiability Problems*

We have now proved that all our satisfiability problems are equivalent, up to polynomials, which leads to the next theorem:

Theorem 4.6. *The problems U-SAT, U-k-SAT for $k \geqslant 3$ and U-1-3-SAT are US-complete.*

Proof. Let us summarize the previous results:

- U-SAT $\preceq$ U-3-SAT (Theorem 4.1);
- U-3-SAT $\preceq$ U-1-3-SAT $\preceq$ U-3-SAT (Theorems 4.2 and 4.3);
- $\forall\, k \geqslant 4$, U-$(k-1)$-SAT $\preceq$ U-k-SAT (Theorem 4.4);
- $\forall\, k \geqslant 3$, U-k-SAT $\preceq$ U-SAT (Theorem 4.5).

Thanks to the transitivity of the polynomial reduction $\preceq$ (or, equivalently, by considering the composition of the previous reductions in order to obtain explicit polynomial reductions), these relations show that U-SAT, U-k-SAT ($k \geqslant 3$) and 1-3-SAT are pairwise polynomially reducible:

- $\forall\, k \geqslant 3$, U-SAT $\preceq$ U-k-SAT $\preceq$ U-SAT
- U-SAT $\preceq$ U-1-3-SAT $\preceq$ U-SAT
- $\forall\, k \geqslant 3$, 1-3-U-SAT $\preceq$ U-k-SAT $\preceq$ 1-3-U-SAT
- $\forall\, k \geqslant 3$, $\forall\, k' \geqslant 3$, U-k-SAT $\preceq$ U-k'-SAT.

As said above, U-SAT is known by [7] to be *US*-complete. Because of the previous relations, this involves that the problems U-SAT, U-k-SAT for $k \geqslant 3$ and U-1-3-SAT are also *US*-complete. □

Moreover, because of this *US*-completeness (or because of the mutual reductibility and the fact that U-SAT belongs to *DP* and is co-*NP*-hard), the problems U-SAT, U-k-SAT for $k \geqslant 3$ and U-1-3-SAT belong to *DP* and are co-NP-hard as well.

Observe that we can solve U-SAT thanks to any algorithm A solving SAT by performing A at most $2n+1$ times, where n still denotes the number of variables of the instance to be solved.

Indeed, let $I = (\mathcal{X}; c_1, c_2, ..., c_m)$ be any instance of U-SAT, where $\mathcal{X}$ is the set of the n Boolean variables x_i ($1 \leqslant i \leqslant n$) and c_1, c_2, ..., c_m are the clauses. The answer admitted by I (considered as an instance of U-SAT) is YES if and only if there exists an assignment satisfying I and this assignment is unique.

As I is also an instance of SAT, we may apply A to I. So, to know whether such an assignment exists and is unique, we first apply A to I, considered as an instance of SAT. If the answer provided by A is NO, then the answer admitted by I as an instance of U-SAT is NO as well and the process is over.

Now, assume that the answer returned by A is YES: at least one assignment satisfies I. We want to check whether this assignment is unique or not. There are at least two possible assignments to satisfy I if and only if there exists at least one variable x_i which can get TRUE or FALSE as its possible truth value. So, for any index i ($1 \leqslant i \leqslant n$), we first consider the TRUE value for x_i: we remove all the clauses containing x_i since they are satisfied by x_i, we remove $\overline{x_i}$ from the clauses containing this literal since $\overline{x_i}$ is set to FALSE, and we apply A to the resulting instance; if the answer is NO, then necessarily x_i must get FALSE as its truth value and we go to another variable. If the answer is YES, we now consider the FALSE value for x_i, we remove all the clauses containing $\overline{x_i}$, we remove x_i from the clauses containing this literal, and

we apply A to the resulting instance; if the answer is NO, then necessarily x_i must get TRUE as its truth value and we go to another variable; if the answer if YES, x_i can be set to TRUE or FALSE while I can still be satisfied in both cases and we thus may conclude that the assignment is not unique: the answer admitted by I as an instance of U-SAT is NO and we can stop.

The worst case is the one for which we must try TRUE and FALSE for all the variables. This case requires $2n + 1$ calls to A, as claimed above.

Similar processes can be performed for U-1-3-SAT and k-SAT for $k \leqslant 3$ with the following adaptation when the value of a variable x is fixed: in order to keep the right number of variables inside each clause, we do not remove x or $\overline{x}$ (but we still remove the clauses which can be removed), but we replace x or $\overline{x}$ by a variable of which the value is forced, as we did previously (see Remark 2.1 and the proofs above).

5. Conclusion

The previous theorems show that the problems U-SAT, U-k-SAT for $k \geqslant 3$ and U-1-3-SAT are equivalent, in the sense that they are pairwise related by polynomial reductions, and thus they are all *US*-complete. As a consequence, they are also co-*NP*-hard (for both definitions specified in Subsection 3.2) and they belong to *DP*. An interesting question is: are they *DP*-complete? The question remains open:

Open problem 1. Are U-SAT, U-k-SAT for $k \geqslant 3$ and U-1-3-SAT *DP*-complete?

In [28], C. H. Papadimitriou states that "U-SAT is not believed to be *DP*-complete". Note that, if U-SAT is *DP*-complete, then U-1-3-SAT and U-k-SAT for $k \geqslant 3$ will also be *DP*-complete, and conversely.

In [7], A. Blass and Y. Gurevich wonder whether U-SAT is *NP*-hard. Though they do not specify the definition that they consider for *NP*-hardness, we can guess that it is the definition based on polynomial reductions and not on Turing reductions. On the other hand, if we consider the definition of *NP*-hardness given by M. R. Garey and D. S. Johnson in [11], based on Turing reductions, then U-SAT, U-1-3-SAT and U-k-SAT ($k \geqslant 3$) are also *NP*-hard. If we consider the definition of *NP*-hardness based on polynomial reductions, then the question remains open:

Open problem 2. Are U-SAT, U-k-SAT for $k \geqslant 3$ and U-1-3-SAT *NP*-hard with respect to the definition based on polynomial reductions?

Anyway, with their definition of *NP*-hardness, A. Blass and Y. Gurevich show in [7] that U-SAT is *NP*-hard if and only if U-SAT is *DP*-complete. They are still more precise by stating that the following five questions are equivalent:

- (1) Is U-SAT *DP*-complete?
- (2) Is *DP* included in *US*?
- (3) Is *DP* equal to *US*?
- (4) Is U-SAT *NP*-hard (surely with respect to the definition based on polynomial reductions, see above)?
- (5) Is *NP* included in *US*?

Thanks to the polynomial reductions provided in Section 4, we can set the same questions with U-k-SAT ($k \geqslant 3$) or U-1-3-SAT instead of U-SAT.

References

[1] Aaronson, S., Habryka, O. and Kuperberg G. (2026), Complexity Zoo, MediaWiki, https://complexityzoo.net.

[2] Aspvall B., Plass M. F. and Tarjan R. E. (1979), "A linear-time algorithm for testing the truth of certain quantified Boolean formulas", *Information Processing Letters* 8, 121–123.

[3] Atallah M. J. (ed.) (1998), *Algorithms and Theory of Computation Handbook*, CRC Press, Boca Raton.

[4] Ausielle G., Marchetti-Sapccamela A., Crescenzi P., Gambosi G., Protasi M. and Kann V. (1999), *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*, Springer, Berlin.

[5] Barthélemy J.-P., Cohen G. and Lobstein A. (1996), *Algorithmic Complexity and Communication Problems*, University College of London, London.

[6] Biere A., van Maaren H. and Walsh T. (eds) (2021), *Handbook of Satisfiability*, IOS Press, Amsterdam.

[7] Blass A. and Gurevich Y. (1982), "On the unique satisfiability problem", *Information and Control* 55, 80-88.

[8] Calabro C., Impagliazzo R., Kabanets V. and Paturi R. (2008), "The complexity of Unique k-SAT: an isolation lemma for k-CNFs", *Journal of Computer and System Sciences* 74, 386–393.

[9] Cook S. A. (1971), "The complexity of theorem-proving procedures", *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, 151–158.

[10] Cormen T. (2000), "Algorithmic complexity", in: Rosen K. H. (ed.), *Handbook of discrete and combinatorial mathematics*, 1077–1085, CRC Press, Boca Raton.

[11] Garey M. R. and Johnson D. S. (1979), *Computers and Intractability, a Guide to the Theory of NP-Completeness*, Freeman, New York.

[12] Hansen P. and Jaumard B. (1985), "Uniquely solvable quadratic Boolean equations", *Discrete Applied Mathematics* 12(2), 147-154.

[13] Hansen T. D., Kaplan H., Zamir O. and Zwick U. (2019), "Faster k-SAT Algorithms using Biased-PPSZ", *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing* (STOC '19), Phoenix, USA.

[14] Hemachandra L. A. (1989), "The strong exponential hierarchy collapses", *Journal of Computer and System Sciences* 39, 299–322.

[15] Hemaspaandra L. (2000), "Complexity classes", in: Rosen K. H. (ed.) *Handbook of discrete and combinatorial mathematics*, 1085–1090, CRC Press, Boca Raton.

[16] Hudry O. and Lobstein A. (2019), "Unique (optimal) solutions: complexity results for identifying and locating-dominating codes", *Theoretical Computer Science* 767, 83–102.

[17] Hudry O. and Lobstein A. (2019), "Complexity of unique (optimal) solutions in graphs: Vertex Cover and Domination", *Journal of Combinatorial Mathematics and Combinatorial Computing* 110, 217–240.

[18] Hudry O. and Lobstein A. (2022), "On the complexity of determining whether there is a unique Hamiltonian cycle in a graph", *WSEAS Transactions on Mathematics* 21, 433–446.

[19] Hudry O. and Lobstein A. (2023), "Some complexity considerations on the uniqueness of graph colouring", *WSEAS Transactions on Mathematics* 22, 483–493.

[20] Johnson D. S. (1990), "A catalog of complexity classes", in: van Leeuwen J. (ed.), *Handbook of theoretical computer science, Vol. A: Algorithms and complexity*, Chapter 2, Elsevier, Amsterdam.

[21] Juban L. (1999), "Dichotomy theorem for the generalized unique satisfiability problem", in: Ciobanu G. and Paun G. (eds.) *Fundamentals of computation theory* (FCT'99), LNCS 1684, Berlin, 327-337.

[22] Karp R. M. (1972), "Reducibility Among Combinatorial Problems", in: Miller R. E. and Thatcher J. W. (eds.), *Complexity of Computer Computations*, Plenum, New York, 85–103.

[23] Korte B. and Vygen J. (2018), *Combinatorial Optimization: Theory and Algorithms*,

Springer, Berlin.
[24] Marek V. (2009), *Introduction to Mathematics of Satisfiability*, Chapman and Hall/CRC Studies in Informatics, New York.
[25] Matthews W. and Paturi R. (2010), "Uniquely Satisfiable k-SAT Instances with Almost Minimal Occurrences of Each Variable", in: Strichman O., Szeider S. (eds), *Theory and Applications of Satisfiability Testing, Proceedings of the 13th International Conference SAT 2010*, Edinburgh, UK, 369–374, Springer, Berlin, Heidelberg.
[26] Minoux M. (1992), "The Unique Horn-Satisfiability problem and quadratic Boolean equations", *Annals of Mathematics and Artificial Intelligence* 6, 253-266.
[27] Papadimitriou C. H. (1984), "On the complexity of unique solutions", *Journal of the Association for Computing Machinery* 31, 392-400.
[28] Papadimitriou C. H. (1994), *Computational Complexity*, Addison-Wesley, Reading.
[29] Papadimitriou C. H. and Steiglitz K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Hoboken, New Jersey.
[30] Papadimitriou C. H. and Yannakakis M. (1982), "The complexity of facets (and some facets of complexity)", *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, San Francisco, USA.
[31] Papadimitriou C. H. and Yannakakis M. (1984), "The complexity of facets (and some facets of complexity)", *Journal of Computer and System Sciences* 28, 244–259.
[32] Petke J. (2015), *Bridging Constraint Satisfaction and Boolean Satisfiability*, Springer, Heidelberg, New York, Dordrecht, London.
[33] Schaeffer T. J. (1978), "The complexity of satisfiability problems", *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, 216–226.
[34] Scheder D. (2024), "PPSZ is better than you think", *TheoretiCS* 3, Art. 5, 1–37.
[35] Schöning U. and Toran J. (2013), *The Satisfiability Problem: Algorithms and Analyses*, Lehmanns Media, Berlin.
[36] Valiant L. G. and Vazirani V. V. (1986), "NP is as easy as detecting unique solutions", *Theoretical Computer Science* 47 (1), 85-93.
[37] Vazirani V. V. (2003), *Approximation algorithms*, Springer, Berlin.