PURL: A new polynomial-time solvable class of satisfiability[®]

J.R. Portillo[a], J.I. Rodrigues[b]

(a) *Universidad de Sevilla.*
E-mail: josera@us.es

(b) *Universidade do Algarve*
E-mail: jirodrig@ualg.pt

Abstract. In this work a new polynomial-time solvable class of satisfiability PURL (*PropUnit* RemoveLiterals) is presented, based on natural extensions of the known concepts of the 1-neighbourhood of a clause and removable literal. The algorithm RemoveLiterals is also shown which determines if a formula in PURL is satisfiable. The PURL class is a proper superset of all previously known polynomial-time solvable classes. The study of this class is motivated by the resolution of geometric problems.

Keywords. SAT, Satisfiability, CNF-formula, PropUnit, PURL.

The Satisfiability problem (SAT) is the determination of whether there exists a satisfying truth assignment for a given Boolean expression, usually in Conjunctive Normal Form (CNF). This problem is NP-complete; thus, there is no known polynomial-time algorithm for its solution. Due of the importance of SAT in logic, artificial intelligence, and operations research, considerable effort has been made to determine how to cope with this disappointing reality. Two approaches are: (1) the determination of whether algorithms for SAT exist which usually present a result in polynomial time; (2) the identification of special classes of SAT that can be solved in polynomial time. This paper is concerned with the second approach.

Several polynomial-time solvable classes of CNF formulas have been proposed. The Horn class [1, 2] and the 2-SAT class [3] has been known for a long time to be solved in linear time. Both these classes show up frequently in real-world problems.

Some generalizations of the Horn class have been studied, e.g., renamable Horn [4, 5], extended Horn [6, 7] and q-Horn [8, 9, 10]. Should also be considered the following classes: CC-balanced [11, 12], SLUR (Single Lookahead Unit Resolution) solvable [13, 14], *matched* formulas [14], and LinAut [14, 15, 16, 17].

[®] Partially supported by PAI FQM-0164 and BFM 2001-12474 projects, and by FSE, PRODEP program 5.3/ALG/194.019/01.

In accordance with [14], these are referred to as the well-known polynomial-time solvable classes. There is a strong relation between these classes. For example, the class SLUR was developed as a generalization of other classes including Horn, renamable Horn, extended Horn, and CC-balanced formulas. The relations are briefly explained in Table 1. Maaren [16] has shown that SLUR and LinAut are incomparable extensions of Horn formulas.

Horn	*renamable* Horn	
extended Horn	CC-*balanced*	$\subset$ SLUR
2-SAT	Horn	
q-Horn	*matched formulas*	$\subset$ LinAut

Table 1: Well-known polynomial-time solvable classes and relations of inclusion.

In Section 1 we introduce a new polynomial class of satisfiability, which we call PURL (*PropUnit* RemoveLiterals). The study of this class is motivated by the resolution of some geometric problems, e.g, the plane Single Bend Wiring problem (SBW) can be solved in polynomial time by reducing to 2-SAT [18]. Although the SBW problem in generalized surfaces is NP-complete [19, 20], SBW on the cylinder surface can be encoded in formulas which are in the PURL class, as we will shown in a further paper.

Section 2 is devoted to showing that PURL is a proper *superclass* of all the well-known polynomial-time solvable classes.

1 The PURL class of satisfiability

The terminology used in this paper follows that given by Franco and Gelder [14] and Goldberg [21]. Given a CNF formula $\mathcal{F}$ and a clause $C \in \mathcal{F}$, Goldberg defined the 1-neighbourhood of a clause C, $T_1(C)$ as the set of all assignments satisfying only one literal of C. If l is a literal in C, then we call the 1-neighbourhood of a clause C relative to literal l as the set of truth assignments for which l has value 1 and for wich all other literals have value 0 and this set is denoted as $T_1(l, C)$. Obviously, $T_1(l, C)$ is a subset of $T_1(C)$ and therefore these sets are equal only if C is a unit clause.

Theorem 1. *[21] If $\mathcal{F}$ is a satisfiable propositional formula then there is a clause C and a truth assignment $\tau \in T_1(C)$ for which $\mathcal{F}(\tau) = 1$.*

Given a CNF formula $\mathcal{F}$ and a clause C of $\mathcal{F}$, a literal l in C is said to be removable if every truth assignment $\tau \in T_1(l, C)$ sets the formula to false, i.e.,

$\mathcal{F}(\tau) = 0$. Removing this literal of clause C, we obtain a new CNF formula $(\mathcal{F}\backslash\{C\}) \cup (\{C\backslash\{l\}\})$. It is easy to prove that both formulas are equivalent:

Theorem 2. *Let $\mathcal{F}$ be a CNF formula , C a clause of $\mathcal{F}$, and l a removable literal of C. Thus, $(\mathcal{F}\backslash C) \cup (\{C\backslash\{l\}\})$ is satisfiable if and only if $\mathcal{F}$ is satisfiable.*

By repeating this process a new formula $\mathcal{F}'$ is obtained which is equivalent to $\mathcal{F}$ without removable literals. Hence, $\mathcal{F}$ is satisfiable if and only if $\Box \notin \mathcal{F}'$.

For an instance $\mathcal{F}$ of SAT, determining if all literals in a clause are removable is equivalent to determining the non-satisfiability of $\mathcal{F}$. However, since SAT is NP-complete, the detection of the removable literals is also NP-complete.

Given a clause C of a CNF formula $\mathcal{F}$, $C = [l, l_1, \cdots, l_t]$, we denote $\{l, \overline{l_1}, \cdots, \overline{l_t}\}$ by $\tau_0(l, C)$. The partial truth assignment $\tau_0(l, C)$ is a subset of every truth assignment τ of $T_1(l, C)$. In this sense, $\tau_0(l, C)$ is the *minimal* truth assignment of the 1-neighbourhood of C relative to l. Furthermore, $\tau_0(l, C) := \bigcap_{\tau \in T_1(l,C)} \tau$.

Definition 1. *A literal l in a clause C of a propositional formula $\mathcal{F}$ is p-*removable *if the algorithm $PropUnit(\mathcal{F} \cup \mathcal{U}(\tau_0(l, C)))$ returns a propositional formula with a null clause.*

The operator $\mathcal{U}(\tau_0(l, C))$ returns a formula with a set of unit clauses, where each clause contains a literal of partial assignment $\tau_0(l, C)$. It is defined by its compatibility with the inputs of algorithm PropUnit.

A version of the algorithm PropUnit and a strategy to run it in linear time in the number of clauses and literals can be found in Dalal and Etherington [22].

All p-removable literals are removable, however the inverse is not true. Thus, if l is a p-removable literal of C, $\mathcal{F}$ and $(\mathcal{F}\backslash\{C\}) \cup \{C\backslash\{l\}\}$ are equivalent formulas by Theorem 2.

The algorithm RemoveLiterals, shown below, removes the p-removable literals by searching for these literals in each clause. It returns a propositional formula $\mathcal{F}'$, equivalent to $\mathcal{F}$ but whitout p-removable literals in any of its clauses. Since the detection of a p-removable literal is made in linear time (due to the algorithm PropUnit), therefore the algorithm RemoveLiterals runs in polynomial time.

```
Algorithm RemoveLiterals (p-removable)
Require: $\mathcal{F}$, CNF-formula
Ensure: $\mathcal{F}, \tau$, formula without $p$-removable literals and a truth assignment
  repeat
    remlitfree = true
    for all $C \in \mathcal{F}$ do
      for all $l \in C$ do
        $(\mathcal{F}', \tau') := PropUnit(\mathcal{F} \cup \mathcal{U}(\tau_0(l, C)))$
```

```
          if $\square \in \mathcal{F}'$ then
            $C := C \setminus [l]$, remlitfree=false
          end if
          if $C = \square$ then
            $\mathcal{F} = \{\square\}$
          end if
        end for
      end for
    until $\mathcal{F} = \{\ \}$ OR $\mathcal{F} = \{\square\}$ OR remilitfree=true
    return $(\mathcal{F}, \tau)$
```

If the formula returned by the algorithm has a null clause then $\mathcal{F}$ is not satisfiable. Otherwise, nothing can be concluded. However, in some cases, if a propositional formula has no null clauses then the formula is satisfiable. Obviously, this occurs if all the removable literals are p-removable.

Definition 2. *(PURL) Let $\mathcal{F}$ be a propositional formula and $\mathcal{F}'$ the equivalent formula without clauses with p-removable literals. $\mathcal{F}$ is in the PURL class if the following property is verified:* $\mathcal{F}$ is satisfiable if and only if $\square \notin \mathcal{F}'$

The satisfiability of a formula in PURL can be determined in polynomial time using the algorithm RemoveLiterals. Unfortunely, although RemoveLiterals is a complete algorithm, it is not reliable. The reliability is not guaranteed for general formulas, only for those in PURL class.

Theorem 3. *If every removable literal in the clauses of a propositional formula is p-removable then the formula is in the PURL class.*

2 PURL AND OTHER POLYNOMIAL CLASSES OF SATISFIABILITY

In this section we will prove that PURL is a strict superset of the well-known polynomial-time solvable classes. In particular, we prove that SLUR and LinAut are proper subclasses of PURL. It remains to be demonstrated if PURL is the polynomial superclass conjectured by Gu *et al.* [23].

We start by showing that 2-SAT is a subclass of PURL. It suffices to verify that in every non-satisfiable CNF formula of 2-SAT there is a clause for which all literals are p-removables. We gives the following lemma without proof due to the space limitation.

Lemma 1. *Let $\mathcal{F}$ be a non-empty 2-SAT formula for which $\Box \notin \mathcal{F}$, l is a literal of a clause of $\mathcal{F}$, and $(\mathcal{F}', \tau') := PropUnit(\mathcal{F} \cup \{[l]\}$. Hence: (a) $\Box \in \mathcal{F}$ or $\mathcal{F}' \subsetneq \mathcal{F}$; (b) In case of $\mathcal{F}' \subsetneq \mathcal{F}$, $\mathcal{F}$ is satisfiable if and only if $\mathcal{F}'$ is satisfiable .*

We now consider a non-satisfiable 2-SAT propositional formulas $\mathcal{F}$ and $\mathcal{F}_m \subset \mathcal{F}$ with the minimum of clauses, i.e., every subformula of $\mathcal{F}_m$ is satisfiable.

If all clauses of $\mathcal{F}_m$ have fewer than two literals, then $\Box$ is in $\mathcal{F}'$, $(\mathcal{F}', \tau') := PropUnit(\mathcal{F}_m)$. Let C be a clause in $\mathcal{F}_m$ with two literals, $C = [u, v]$. Thus, $\tau_0(u, C) = \{u, \overline{v}\}$ and $(\mathcal{F}', \tau') := PropUnit(\mathcal{F}_m \cup \mathcal{U}(\tau_0(u, C)))$, hence $\mathcal{F}' \subsetneq \mathcal{F}_m$ and $\mathcal{F}_m$ is not satisfiable. By Lemma 1, if $\Box \notin \mathcal{F}'$ then $\mathcal{F}'$ is not satisfiable either, which contradicts the hypothesis of $\mathcal{F}_m$ being minimal. Hence, $\Box$ is in $\mathcal{F}'$ and u is a literal p-removable. We now consider $C' := C \backslash \{u\}$. By taking $(\mathcal{F}'', \tau'') := PropUnit((\mathcal{F}_m \backslash C) \cup \{C'\})$, therefore $\Box \in \mathcal{F}''$ and, as a consequence, the literal $v \in C'$ is p-removable.

The RemoveLiterals algorithm now returns a formula with a null clause. It is concluded that a 2-SAT propositional formula is satisfiable if and only if the equivalent formula without p-removable literals does not have a null clause.

Theorem 4. *2-SAT is a subclass of the PURL class.*

2.1 SLUR is a proper subclass of PURL

The SLUR [13, 14] polynomial class of satisfiability is peculiar because its definition is based in a non-deterministic algorithm whereas another well-known polynomial-time solvable classes are defined by characteristics or propierties of clauses and literals of their formulas.

Lemma 2. *Let $\mathcal{F}$ be a propositional formula of the SLUR class of satisfiability. If $\mathcal{F}$ is not satisfiable then a unit clause with a p-removable literal exists.*

Proof. As $\mathcal{F}$ is an instance of SLUR, the algorithm does not returns *give up*. Therefore the answer is a truth assignment satisfying $\mathcal{F}$ or the message *not satisfiable*. Moreover, SLUR returns *not satisfiable* if, considering $(\mathcal{F}', \tau) := PropUnit(\mathcal{F})$, then $\Box \in \mathcal{F}'$, i.e., for a unit clause $C = [u]$, $PropUnit(\mathcal{F} \cup \{[u]\} = PropUnit(\mathcal{F})$. Hence, u is a p-removable literal. $\Box$

The previous Lemma implies that, if $\mathcal{F}$ is in SLUR and not satisfiable, then $RemoveLiterals(\mathcal{F})$ returns a formula with a null clause. i.e, , SLUR is a subclass of PURL. In order to show that it is a strict subclass, it is enough to consider the propositional formula $\{[1, \overline{2}, 4], [1, 2, 5], [\overline{1}, \overline{3}, 6], [\overline{1}, 3, 7]\}$ wich is in the PURL class, since it does not have p-removable literals and is satisfiable. By choosing the sequence of variables 4, 5, 6 and 7, SLUR returns a message *give up* as the answer. Hence $\mathcal{F}$ cannot be an instance of SLUR class. This lead us to the following result:

Theorem 5. *SLUR is a proper subclass of PURL.*

2.2 LinAut is a proper subclass de PURL

Let $\mathcal{F}$ be a propositional formula in LinAut class. Therefore there is a family of pairwise disjoint subformulas of $\mathcal{F}$, $\{\mathcal{F}_1, \cdots, \mathcal{F}_k\}$, for which $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_k = \mathcal{F}$. Each formula $\mathcal{F}_i$ is recursively defined as follows: A linear autarky exists $\mathbf{x} \in Q^n$ and an asociated partial assignment τ_i, $\mathcal{F}_i := \{C \in \mathcal{F} : C(\tau_i) = 1\}$, and $\mathcal{F} := \mathcal{F} \backslash \mathcal{F}_i$.

If $\mathcal{F}$ is not satisfiable, $\mathcal{F}_k$ is a non-satisfiable 2-SAT formula. We have already seen that 2-SAT is a subclass of PURL, and hence $RemoveLiterals(\mathcal{F}_k)$ returns a formula with a null clause, as does $RemoveLiterals(\mathcal{F})$. As a consequence, LinAut is a subclass of PURL.

Theorem 6. *LinAut is a proper subclass of PURL.*

It only remains to be proved that these two classes are different. To this end, it is sufficient to consider the 3-SAT propositional formula $\mathcal{F} = \{[1,2,\overline{4}], [\overline{1},2,4],$ $[2,\overline{3},\overline{4}]$, $[\overline{2},3,\overline{4}]$, $[1,2,6]$, $[\overline{1},2,\overline{6}]$, $[2,5,\overline{6}]$, $[\overline{2},\overline{5},\overline{6}]$, $[1,2,8]$, $[\overline{1},2,\overline{8}]$, $[2,\overline{7},\overline{8}]$, $[\overline{2},7,\overline{8}]$, $[1,2,\overline{0}]$, $[\overline{1},2,0]$, $[2,\overline{9},\overline{0}]$, $[\overline{2},9,\overline{0}]$, $[3,4,6]$, $[\overline{3},4,\overline{6}]$, $[4,\overline{5},\overline{6}]$, $[\overline{4},5,\overline{6}]$, $[3,4,8]$, $[\overline{3},4,\overline{8}]$, $[4,\overline{7},\overline{8}]$, $[\overline{4},7,\overline{8}]$, $[3,4,0]$, $[\overline{3},4,\overline{0}]$, $[4,9,\overline{0}]$, $[\overline{4},\overline{9},\overline{0}]$, $[5,6,\overline{8}]$, $[\overline{5},6,8]$, $[6,7,\overline{8}]$, $[\overline{6},\overline{7},\overline{8}]$, $[5,6,\overline{0}]$, $[\overline{5},6,0]$, $[6,\overline{9},\overline{0}]$, $[\overline{6},9,\overline{0}]$, $[7,8,0]$, $[\overline{7},8,\overline{0}]$, $[8,\overline{9},\overline{0}]$, $[\overline{8},9,\overline{0}]\}$, and its equivalent without p-removable literals, $\mathcal{F}' = \{[2], [3],$ $[\overline{5}], [\overline{6}], [7], [\overline{8}], [\overline{0}]\}$. It easy to prove that $\mathcal{F}$ is in PURL, but not in LinAut.

3 Conclusions

This paper describes a new polynomial-time solvable class of satisfiability. This new class is a proper superset of all previously known polynomial-time solvable classes and can be used for resolving geometric problems. Further research will be devoted to the application of the results of this work to several geometric problems and to the study of hierarchies of polynomially solvable satisfiability problems using the PURL class.

References

[1] Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming* **3** (1984) 267–284.

[2] Itai, A., Makowsky, J.: On the complexity of Herbrand's theorem. *Working Paper* **243** (1982).

[3] Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.* **8**(3) (1979) 121–123.

[4] Lewis, H.: Renaming a set of clauses as a Horn set. *J. ACM* **25** (1978) 134–135.

[5] Aspvall, B.: Recognizing disguised NR(1) instances of the satisfiability problem. *J. Algorithms* **1** (1980) 97–103.

[6] Chandru, V., Hooker, J.N.: Extended Horn sets in propositional logic. *J. ACM* **38** (1991) 205–221.

[7] Del Val, A.: On 2SAT and renamable Horn. *AAAI'00 Proceedings of the 7th (U.S.) National Conference on Artificial Intelligence* (2000) 343–348.

[8] Scutella, M.G.: A note on Dowling and Gallier's top-down algorithm for propositional Horn satisfiability. *Journal Logic Programming* **8** (1990) 265–273.

[9] Boros, E., Hammer, P., Sun, X.: Recognition of q-Horn formulae in linear time. *Discrete Appl. Math.* (55) (1994) 1–13.

[10] Boros, E., Crama, Y., Hammer, P., Saks, M.: A complexity index for satisfiability problems. *SIAM J. Comput.* **23** (1994) 45–49.

[11] Conforti, M., Cornuejols, G.: A class of logic problems solvable by linear programming. *J. ACM* **42** (1995) 1107–1113.

[12] Trümper, K.: Alpha-balanced graphs and matrices and gf(3)-representability of matroids. *J. Comb. Theory Ser.* **B32** (1982) 112–139.

[13] Schlipf, J., Annexstein, F., Franco, J., Swaminathan, R.: On finding solutions for extended Horn formulas. *Inf. Process. Lett.* **54** (1995) 133–137.

[14] Franco, J., van Gelder, A.: A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Appl. Math.* **125**(2-3) (2003) 177–214.

[15] Kullmann, O.: Investigations on autark assignments. *Discrete Appl. Math.* **107**(1-3) (2000) 99–137.

[16] Maaren, H.V.: A short note on some tractable cases of the satisfiability problem. Information and Computation (158) (2000) 125–130.

[17] Kullmann, O.: Lean clause-sets: generalizations of minimally unsatisfiable clause-sets. *Discrete Appl. Math.* **130**(2) (2003) 209–249.

[18] Raghavan, R., Cohoon, J., Sahni, S.: Single Bend Wiring. J. Algorithms **7** (1986) 232–257.

[19] Portillo, J.R.: *Problemas de conexiones ortogonales.* PhD thesis, Universidad de Sevilla (2002).

[20] Garrido, M.A., Márquez, A., Morgana, A., Portillo, J.R.: Single bend wiring on surfaces. *Discrete Appl. Math.* **117**(1-3) (2002) 27–40.

[21] Goldberg, E.: Proving unsatisfiability of CNFs locally. *J. Autom. Reasoning* **28**(5) (2002) 417–434.

[22] Dalal, M., Etheringthon, D.: A hierarchy of tractable satisfiability problems. *Inf. Process. Lett.* **44**(4) (92) 173–180.

[23] Gu, J., Purdom, P., Franco, J., Wah, B. Algorithms for the Satisfiability (SAT) Problem: A survey (invited paper). In: Volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* AMS (1997) 19–151.