The Horn Non-Clausal Class and its Polynomiality

Gonzalo E. Imaz

Artificial Intelligence Research Institute (IIIA) - CSIC, Barcelona, Spain

email: {gonzalo}@iiia.csic.es

Abstract

The expressiveness of propositional non-clausal (NC) formulas is exponentially richer than that of clausal formulas. Yet, clausal efficiency outperforms non-clausal efficiency. Indeed, a major weakness of the latter is that, while Horn clausal formulas, along with Horn algorithms, are crucial for the high efficiency of clausal reasoning, no Horn-like formulas in non-clausal form had been proposed.

To overcome such weakness, we first define the hybrid class $\mathbb{H}_{\text{NC}}$ of Horn Non-Clausal (Horn-NC) formulas by adequately lifting the Horn pattern to NC form, and argue that $\mathbb{H}_{\text{NC}}$, along with future Horn-NC algorithms shall increase non-clausal efficiency just as the Horn class has increased clausal efficiency.

Secondly, we: (i) give the compact, inductive definition of $\mathbb{H}_{\text{NC}}$; (ii) prove that syntactically $\mathbb{H}_{\text{NC}}$ subsumes the Horn class but semantically both classes are equivalent; and (iii) characterize the non-clausal formulas belonging to $\mathbb{H}_{\text{NC}}$.

Thirdly, we define the Non-Clausal Unit-Resolution calculus, or URnc, and prove that it checks the satisfiability of $\mathbb{H}_{\text{NC}}$ in polynomial time. This fact, to our knowledge, makes $\mathbb{H}_{\text{NC}}$ the first characterized polynomial class in NC reasoning.

Finally, we prove that $\mathbb{H}_{\text{NC}}$ is linearly recognizable, and also that it is both strictly more succinct and exponentially richer than the Horn class.

We discuss that in NC automated reasoning, e.g. satisfiability solving, theorem proving, logic programming, etc., can directly benefit from $\mathbb{H}_{\text{NC}}$ and URnc and that, as a by-product of its proved properties, $\mathbb{H}_{\text{NC}}$ arises as a new alternative to analyze Horn functions and implication systems.

Field: Automated Reasoning in Propositional Logic.

Keywords: NNF; Unit Resolution; DPLL; Satisfiability; Theorem Proving; Logic Programming; Tractability; Horn functions; Implication Systems.

1 Introduction

The expressive power of non-clausal (NC) formulas is exponentially richer (formally specified in Section 5) than that of clausal formulas and they have found much use in heterogeneous fields and practical settings such as diagnosis [47], algebraic theorems [116], linear monadic decomposition [78], symbolic model checking [157], formal verification [57], supervisory control [40], circuit verification [56], linear quantifier elimination [123], quantified boolean formulas [61], constraint problems [154], theorem proving [126], ontologies [4], knowledge compilation [49], heuristics [10], satisfiability modulo theory [95], MaxSAT [110], encodings [121], modal logics [68], nested expressions [128], description logics [99], many-valued logics [159], learning [4], deductive databases [54], minimum unsatisfiability [33], compiling linear constrains [145], constraint handling rules [74], dynamic systems
DPLL, model generation, Horn clause verification, constraint logic programs, NC resolution, Skolemisation, numeric planning, stochastic search and hard problems.

On the other side, the Horn clausal formulas are pivotal in our approach. Such formulas can be read naturally as instructions for a computer and are central for deductive databases, declarative programming, automated reasoning and artificial intelligence. Modeling and computing with Horn languages have received a great deal of attention since 1943 and they are the core of countless contemporary research fields spread across many logics and a variety of reasoning problems.

Regarding specifically the conjunctive rôle of Horn formulas and Horn-SAT algorithms in propositional logic, its linearity for both satisfiability checking Horn formulas and simplifying non-Horn ones is likely its most significant, well-known virtue. The valuable contribution of Horn efficiency to the global efficiency of clausal reasoning is evidenced by the fact that highly-efficient DPLL solvers embed a Horn-SAT-like algorithm so-called Unit Propagation. Hence, finding polynomial clausal super-classes of the Horn class has been a key issue in the quest for improving clausal reasoning; some found classes are: hidden Horn, generalized Horn, Q-Horn, extended-Horn, SLUR, Quad, UP-Horn and more. The Related Work gives a chronological account of such clausal classes and their interrelations.

Despite those many clausal classes, so far no attempt is known to lift the Horn pattern to the non-clausal level, i.e. no NC class is recognized to be the NC analogous to the Horn class. Perhaps, the Horn pattern was considered inherently clausal and not extensible to non-clausal form. In this paper, we bridge that gap by introducing the hybrid class of Horn-NC formulas, which results by suitably amalgamating the Horn-clausal and the NC formulas, or equivalently, by appropriately lifting the Horn-clausal pattern to NC form.

Firstly, the syntactical Horn-NC pattern is established by lifting the clausal restriction: a formula is Horn if all its clauses have (any number of negative literals and) at most one positive literal, to the NC level, which gives rise to the recursive non-clausal restriction: an NC formula is Horn-NC if all its disjunctions have any number of negative disjuncts and at most one non-negative disjunct. Accordingly, $H_{NC}$ is the class of Horn-NC formulas. Note that $H_{NC}$ naturally subsumes the Horn class. After such simple definition of $H_{NC}$, we go deeper into the details and give its detailed definition compactly and inductively.

Secondly, we prove the relationships between $H_{NC}$ and the classes Horn and NC. Concretely, we show that $H_{NC}$ and the Horn class are related in that (i) $H_{NC}$ syntactically subsumes the Horn class but both classes are logically equivalent; and $H_{NC}$ and the NC class are related in that (ii) $H_{NC}$ contains all NCs whose clausal form is Horn. The Venn diagram in Fig. 1 below relates $H_{NC}$ to the classes Horn (H), NC (NC) and Clausal (C).

Thirdly, we provide Non-Clausal Unit-Resolution, or $UR_{NC}$, prove its completeness for $H_{NC}$ and that it allows to test the satisfiability of $H_{NC}$ in polynomial time. This claim makes $H_{NC}$, so far as we know, the first characterized tractable NC class. Such polynomiality allows to easily prove that Horn-NC-SAT is P-complete.

So the terms Horn-SAT algorithm and Unit Propagation will be used interchangeably.
Fourthly, we demonstrate the following additional properties of $\mathcal{H}_{\text{NC}}$: (i) $\mathcal{H}_{\text{NC}}$ is linearly recognizable, i.e. deciding whether an NC is Horn-NC takes linear time; (ii) $\mathcal{H}_{\text{NC}}$ is more succinct \cite{75} than the Horn class, i.e. some Horn-NC formulas are exponentially smaller than their equivalent Horn formulas; and (iii) syntactically, $\mathcal{H}_{\text{NC}}$ is exponentially richer than the Horn class, i.e. for each Horn formula there are exponentially many Horn-NCs.

We summarize and illustrate our aforementioned contributions through the NC formula $\varphi$ below, whose suffix notation will be detailed in Subsection 2 and wherein $A, B, \ldots$ and $\overline{A}, \overline{B}, \ldots$ are positive and negative literals, respectively, and $\phi_1, \phi_2$ and $\phi_3$ are NCs:

$$\varphi = \{ \land A \ (\lor \overline{B} \ \{ \land (\lor \overline{D} C) \ A \} \ (\lor \ \overline{\phi_1 \ \{ \land \phi_2 \ \overline{A} \}) \ B \}) \} \ \phi_3 \}$$

We will show that $\varphi$ is Horn-NC when $\phi_1, \phi_2$ and $\phi_3$ are Horn-NC and at least one of $\phi_1$ or $\phi_2$ is negative. In that case we will show that:

- $\varphi$ can be tested for satisfiability in polynomial time.
- $\varphi$ can be recognized as Horn-NC in linear time.
- $\varphi$ is logically equivalent to a Horn formula.
- Applying $\land/\lor$ distributivity to $\varphi$ yields a Horn formula.
- There are exponentially many Horn-NCs equivalent to $\varphi$.
- $\varphi$ is exponentially smaller than its equivalent Horn formula.

In a nutshell, $\mathcal{H}_{\text{NC}}$ gathers benefits from possessing a salient expressive power and from enjoying appealing properties for efficient reasoning. Hence, it is reasonable to expect that $\mathcal{H}_{\text{NC}}$ in tandem with fast future Horn-NC algorithms could be relevant in NC reasoning to the extent to which $\mathcal{H}_{\text{NC}}$ could be regarded the NC analogous to the Horn clausal class.

A direct benefit of this work is that $\mathcal{H}_{\text{NC}}$ and UR$_{\text{NC}}$ pave the way to develop NC DPLL reasoners able to emulate the efficiency of their clausal counterparts. We also argue that $\mathcal{H}_{\text{NC}}$ and UR$_{\text{NC}}$ allow logic programing and in general, knowledge-based systems: (i) enriching their syntax from simple Horn rules to Horn-NC rules where heads and bodies are NCs with slight restrictions; and (ii) answering queries with an efficiency comparable to the clausal efficiency as, both, Horn-NCs can be tested for satisfiability polynomially and have only one minimal model (as aforementioned, they are equivalent to a Horn formula).

As a by-product of its properties, $\mathcal{H}_{\text{NC}}$ can potentially draw interest from other fields such as Boolean functions and implicational (closure) systems. Indeed, $\mathcal{H}_{\text{NC}}$ can serve to analyze the Boolean functions representable by Horn formulas \cite{43}, since $\mathcal{H}_{\text{NC}}$ is equivalent to the Horn class, is easily recognizable and its formulas are much smaller than their clausal representations. Regarding implicational systems \cite{23}, instead of clausal implications related to Horn formulas, general NC implications related to Horn-NCs can be envisioned. Further applications using the $\mathcal{H}_{\text{NC}}$ properties are given in the related work.
The Future Work section outlines a substantial number of research directions which pivot on two research axes: (i) $H_{\text{NC}}$ and UR$_{\text{NC}}$ can be smoothly extended beyond propositional logic to richer logics and adapted to different reasoning problems; and (ii) our work can be used as a lever to develop the NC paradigm; for example, our definition of UR$_{\text{NC}}$ is the basis to conceive a new definition of Non-Clausal Resolution.

This paper continues as follows. Section 2 presents background and terminology. Section 3 defines $H_{\text{NC}}$ and relates $H_{\text{NC}}$ to the classes NC and Horn. Section 4 introduces UR$_{\text{NC}}$ and proves the tractability of $H_{\text{NC}}$. Section 5 demonstrates further properties of $H_{\text{NC}}$. Section 6 and 7 focus on related work and future work, respectively. The last section summarizes the main contributions.

2 Background, Notation and Terminology

This section presents background on propositional non-clausal logic (the reader is referred to e.g. [20] for a more complete background) and introduces our notation and terminology.

Definition 2.1. Positive literals are noted $\{A, B, \ldots\}$ and negative ones $\{\overline{A}, \overline{B}, \ldots\}$. $L$ is the set of literals. A clause is a disjunction of literals. A clause with at most one positive literal is Horn. A conjunction of clauses is a clausal formula. A conjunction of Horn clauses is a Horn formula. $C$ and $H$ are the sets of clausal and Horn formulas, respectively.

Definition 2.2. The NC alphabet is formed by the sets of: constants $\{T, F\}$, literals $L$, connectives $\{\neg, \lor, \land\}$ and auxiliary symbols: $\{,\}$, $\{\,\}$.

For the sake of readability of NC formulas, we employ: (1) the prefix notation because it requires only one connective $\lor$ or $\land$ per formula, while infix notation requires $k - 1$ ($k$ is the arity of $\lor$ and $\land$); and (2) two formula delimiters, $\{\ldots\}$ for conjunctions and $(\ldots)$ for disjunctions (Definition 2.3), to distinguish them inside nested NCs. The negation normal form formulas (NNFs) do not use the connective $\neg$. We give next our notation for NNFs.

Definition 2.3. The set $\mathbb{N}_{\text{NF}}$ of NNF formulas is defined in the usual way:

- $\{T, F\} \cup L \subset \mathbb{N}_{\text{NF}}$.
- If for all $i \in \{1, \ldots, k\}$, $\varphi_i \in \mathbb{N}_{\text{NF}}$ then $\{\land \varphi_1 \ldots \varphi_i \ldots \varphi_k\} \in \mathbb{N}_{\text{NF}}$.
- If for all $i \in \{1, \ldots, k\}$, $\varphi_i \in \mathbb{N}_{\text{NF}}$ then $\{\lor \varphi_1 \ldots \varphi_i \ldots \varphi_k\} \in \mathbb{N}_{\text{NF}}$.
  - $\{\land \varphi_1 \ldots \varphi_i \ldots \varphi_k\}$ and any $\varphi_i$ are called conjunction and conjunct, respectively.
  - $\{\lor \varphi_1 \ldots \varphi_i \ldots \varphi_k\}$ and any $\varphi_i$ are called disjunction and disjunct, respectively.
  - $\langle \diamond \varphi_1 \ldots \varphi_i \ldots \varphi_k \rangle$ stands for both $\{\lor \varphi_1 \ldots \varphi_i \ldots \varphi_k\}$ and $\{\land \varphi_1 \ldots \varphi_i \ldots \varphi_k\}$.

Example 2.4. $\varphi_1$ and $\varphi_3$ below are NNF while $\varphi_2$ and $\varphi_4$ are not.

- $\varphi_1 = (\lor \{\land \overline{A} \ T\} \{\land \ \lor \overline{A} \ C\} \ T \{\land \ D \ (\lor \ A \ \overline{B})\})$.
- $\varphi_2 = \{\land \ D \ \neg(\lor \{\land \overline{A} \ B\} \ F \ (\lor \ A \ C))\}$.
- $\varphi_3 = \{\land \ \varphi_1 \ (\lor \{\land \ A \ \overline{C}\} \overline{B}) \ (\lor \ \varphi_1 \ F \ A)\}$.
- $\varphi_4 = (\lor \ \varphi_2 \ \{\land \ A \ (\lor \ \varphi_1 \ \overline{D} \ \varphi_3)\} \ \varphi_2 \ B \ (\lor \ \varphi_1 \ \varphi_3)\}$

Definition 2.5. The set $\mathbb{N}_{\text{NC}}$ of NC formulas is built inductively as $\mathbb{N}_{\text{NF}}$ but over the whole NC alphabet, namely including also the connective $\neg$ (we omit the formal details).
Example 2.6. $\varphi_2$ and $\varphi_4$ in Example 2.4 which were not in $\mathbb{N}_{\text{NF}}$, are in $\mathbb{N}_{\text{C}}$.

Definition 2.7. Sub-formulas are inductively defined as follows. The unique sub-formula of an atom ($\{T, F\} \cup L$) is the atom itself. The sub-formulas of $\varphi$ are $\varphi$ itself more the sub-formulas of either (i) $\varphi'$, if $\varphi = \neg \varphi'$, or (ii) the $\varphi_i$'s, if $\varphi = \langle \circ \varphi_1 \ldots \varphi_i \ldots \varphi_k \rangle$.

Definition 2.8. NCs are representable by trees if: (i) each atom is a leaf node and each occurrence of connectives is an internal node; and (ii) the arcs are given by: (a) each sub-formula $\neg \varphi$ is an arc linking the nodes of $\neg$ and of the connective of $\varphi$; and (b) each sub-formula $\langle \circ \varphi_1 \ldots \varphi_i \ldots \varphi_k \rangle$ is a $k$-ary hyper-arc linking the nodes of $\circ$ and the nodes of, for every $i$, $\varphi_i$ if $\varphi_i$ is an atom and of its connective otherwise.

Example 2.9. The tree representing $\varphi_2$ in Example 2.4 is given in Fig. 2.

![Fig. 2. Tree of $\varphi_2$.](image)

There exist different, bi-dimensional graphical models [24, 93] Our approach also applies when NCs are modeled and implemented as directed acyclic graphs (DAGs), which are more general than trees and allow for important savings in both space and time.

Definition 2.10. An NC formula $\varphi$ is modeled by a DAG when each sub-formula $\varphi$ is modeled by a unique DAG $D_\varphi$ and each $\varphi$-occurrence by a pointer to (the root of) $D_\varphi$.

Example 2.11. Each sub-formula $\varphi_1$ to $\varphi_3$ in $\varphi_4$ from Example 2.4 requires one DAG and two sub-formula pointers.

An interpretation $I$ maps $\mathbb{N}_{\text{C}}$ onto $\{0, 1\}$ and maps constants as follows: $I(T) = I(\{\wedge\}) = 0$ and $I(F) = I(\{\vee\}) = 1$. As the interpretation of negated, conjunctive and disjunctive formulas is well-known, by space reasons we omit their formal definitions.

Definition 2.12. An interpretation $I$ is a model of $\varphi$ if $I(\varphi) = 1$. If $\varphi$ has a model then it is satisfiable and otherwise unsatisfiable. $\varphi$ and $\varphi'$ are equivalent, noted $\varphi \equiv \varphi'$, if $\forall I$, $I(\varphi) = I(\varphi')$. $\varphi'$ is logical consequence of $\varphi$, noted $\varphi \models \varphi'$, if $\forall I$, $I(\varphi) \leq I(\varphi')$.

Definition 2.13. We define: SAT, Horn-SAT, NC-SAT and Horn-NC-SAT are the problems of testing the satisfiability of clausal, Horn, NC and Horn-NC formulas, respectively.

Complexity. SAT [42] and NC-SAT are NP-complete, Horn-SAT is linear [55, 91, 117, 135, 60]. Here, the Horn-NC class is defined and proved that Horn-NC-SAT is polynomial.
Definition 2.14. Constant-free, equivalent formulas are straightforwardly obtained by applying to sub-formulas the next simplifying rules:

- Replace: \( \{ \land F \varphi \} \leftarrow F; \)  
  \( \{ \lor T \varphi \} \leftarrow T; \)  
  \( \{ \land T \varphi \} \leftarrow \varphi; \)  
  \( \{ \lor F \varphi \} \leftarrow \varphi \)

Remark. For simplicity and since constant-free, equivalent formulas are easily obtained, hereafter we will consider only constant-free formulas.

3 The Horn-NNF and Horn-NC Classes

We give first the definition of \( \mathbb{H}_{\text{NNF}} \) and then that of \( \mathbb{H}_{\text{NC}} \). We will use the following abbreviations: \( \text{HNF} \) for Horn-NNF and \( \text{HNC} \) for Horn-NC.

3.1 The Horn-NNF formulas: \( \mathbb{H}_{\text{NNF}} \)

Definition 3.1. Negative formulas are NNF formulas having solely negative literals. \( \mathbb{N}_{\text{G}} \) is the set of negative NNF formulas.

Example 3.2. An example of negative formula is \( (\lor \{ \land \overline{A} \overline{C} \} \{ \land \overline{E} (\lor \overline{A} \overline{B}) \}) \).

Next we lift the the Horn pattern to the non-clausal level as follows:

Definition 3.3. An NNF is HNF all its disjunctions have any number of negative disjuncts and at most one non-negative disjunct. We denote \( \mathbb{H}_{\text{NNF}} \) the class of HNF formulas.

Clearly Horn formulas are HNF, namely \( \mathbb{H} \subset \mathbb{H}_{\text{NNF}} \).

Remark. As Definition 3.3 is not concerned with how NNFs are represented, our approach also applies when NNFs are represented by DAGs and not just by trees. Yet, for the sake of simplicity, we will use NNFs representable by trees throughout this article.

Example 3.4. Let us consider \( \varphi_1 \) and \( \varphi_2 \) below. As \( \varphi_1 \) has two disjuncts and only one is non-negative, \( \varphi_1 \) is HNF. Yet \( \varphi_2 \) has two non-negative disjuncts and so it is not HNF.

- \( \varphi_1 = (\lor \{ \land \overline{B} \overline{D} \} \{ \land C A \}) \)
- \( \varphi_2 = (\lor \{ \land \overline{B} D \} \{ \land C \overline{A} \}) \)

Example 3.5. Below we consider \( \varphi \) and \( \varphi' \) resulting from \( \varphi \) by switching its left-most \( \overline{A} \) for \( A \). All the disjunctions of \( \varphi \), i.e. \( (\lor \overline{A} C) \), \( (\lor A \overline{B}) \) and \( \varphi \) itself, have exactly one non-negative disjunct; so \( \varphi \) is HNF. Yet, \( \varphi' = (\lor A \phi) \), \( \phi \) being non-negative; so, as \( \varphi' \) has two non-negative disjuncts, \( \varphi' \) is not HNF.

- \( \varphi = (\lor \overline{A} \{ \land (\lor \overline{A} C) \} \{ \land D (\lor A \overline{B}) \}) \)
- \( \varphi' = (\lor A \{ \land (\lor \overline{A} C) \} \{ \land D (\lor A \overline{B}) \}) \)

Proposition 3.6. All sub-formulas of an HNF formula are HNF.

The proof follows immediately from Definition 3.3. Note that the converse does not hold since there are non-HNF formulas all of whose sub-formulas are HNF.

Towards a fine-grained definition of \( \mathbb{H}_{\text{NNF}} \), first we inductively specify HNF conjunctions (Lemma 3.7) and HNF disjunctions (Lemma 3.9), and then we compactly specify \( \mathbb{H}_{\text{NNF}} \) by embedding both specifications into an inductive function (Definition 3.12).

Conjunctions of Horn clausal formulas are Horn too, and a similar kind of Horn-like compliance also holds in NC form, viz. conjunctions of HNF formulas are HNF too.
**Lemma 3.7.** Conjunctions of HNF formulas are HNF as well, formally

\[ \{ \land \phi_1 \ldots \phi_i \ldots \phi_k \} \in \mathbb{H}_{\text{HNF}} \text{ iff } 1 \leq i \leq k, \phi_i \in \mathbb{H}_{\text{HNF}}. \]

The proof is straightforward.

**Example 3.8.** If $H_1$ is Horn and $\phi \in \mathbb{H}_{\text{HNF}}$, then $\{ \land H_1 (\lor D A) \phi \} \in \mathbb{H}_{\text{HNF}}$.

It is not hard to check that the definition of HNF disjunction in Definition 3.3 can be equivalently expressed in the next recursive manner: *an NNF disjunction is HNF if it has any number of negative disjuncts and one HNF disjunct*, which leads to:

**Lemma 3.9.** A disjunctive NNF $\phi = (\lor \phi_1 \ldots \phi_i \ldots \phi_k)$ with $k \geq 1$ disjuncts belongs to $\mathbb{H}_{\text{HNF}}$ iff it has one HNF and $k - 1$ negative disjuncts, formally

\[ (\lor \phi_1 \ldots \phi_i \ldots \phi_k) \in \mathbb{H}_{\text{HNF}} \text{ iff there is } i \text{ s.t. } \phi_i \in \mathbb{H}_{\text{HNF}} \text{ and for all } j \neq i, \phi_j \in \mathbb{N}_G. \]

**Proof.** If. Since the sub-formulas $\forall j, j \neq i, \phi_j$ have no positive literals, the non-negative disjunctions of $\phi = (\lor \phi_1 \ldots \phi_i \ldots \phi_k)$ are those in $\phi_i$ plus $\phi_i$ and $\phi_i$ themselves. Given that by hypothesis $\phi_i \in \mathbb{H}_{\text{HNF}}$ and that $\forall j, j \neq i, \phi_j$ have no positive literals, then $\phi = (\lor \phi_1 \ldots \phi_i \ldots \phi_k)$ satisfies Definition 3.3, and so $\phi \in \mathbb{H}_{\text{HNF}}$. Iff. It is done by contradiction: if (i) $\phi_i \notin \mathbb{H}_{\text{HNF}}$ or (ii) $\exists j, j \neq i, \phi_i, \phi_j \notin \mathbb{N}_G$, then $(\lor \phi_1 \ldots \phi_i \ldots \phi_k) \notin \mathbb{H}_{\text{HNF}}$. A similar proof is given in Theorem 3.31 and so, by space reasons, we omit the details.\[\blacksquare\]

From Lemma 3.9, we have: (A) non-recursive HNF disjunctions are Horn clauses; (B) NNF disjunctions with all negative disjuncts are HNF; and (C) NNF disjunctions with $k \geq 2$ non-negative disjuncts are not HNF. Next, we reexamine Examples 3.4 and 3.5 but this time bearing Lemma 3.9 in mind.

**Example 3.10.** Below we analyze $\phi_1$ and $\phi_2$ from Example 3.4.

- $\phi_1 = (\lor \{ \land B D \} \{ \land C A \})$.
  - Clearly $\{ \land B D \} \notin \mathbb{N}_G$ and by Lemma 3.7, $\{ \land C A \} \in \mathbb{H}_{\text{HNF}}$.
  - According to Lemma 3.9, $\phi_1 \in \mathbb{H}_{\text{HNF}}$.

- $\phi_2 = (\lor \{ \land B D \} \{ \land C \bar{A} \})$.
  - Clearly $\{ \land B D \} \notin \mathbb{N}_G$ and $\{ \land C \bar{A} \} \notin \mathbb{N}_G$; by Lemma 3.9, $\phi_2 \notin \mathbb{H}_{\text{HNF}}$.\[\blacksquare\]

**Example 3.11.** Let us consider again $\phi$ and $\phi'$ in Example 3.5 recalling that $\phi'$ results from $\phi$ by just switching its left-most $A$ for $\bar{A}$.

- By Lemma 3.9, $(\lor \bar{A} C) \in \mathbb{H}_{\text{HNF}}$ and $(\lor A \bar{B}) \in \mathbb{H}_{\text{HNF}}$.
- By Lemma 3.7, $\{ \land D (\lor A \bar{B}) \} \in \mathbb{H}_{\text{HNF}}$.
- By Lemma 3.7, $\phi = (\land (\lor \bar{A} C) \{ \land D (\lor A \bar{B}) \}) \in \mathbb{H}_{\text{HNF}}$.
- We have: $\phi = (\lor \bar{A} \phi)$. Since $\bar{A} \in \mathbb{N}_G$ and $\phi \in \mathbb{H}_{\text{HNF}}$, by Lemma 3.9, $\phi \in \mathbb{H}_{\text{HNF}}$.
- We have: $\phi' = (\lor A \phi)$. Since $A, \phi \notin \mathbb{N}_G$, by Lemma 3.9, $\phi' \notin \mathbb{H}_{\text{HNF}}$.\[\blacksquare\]

By merging Lemmas 3.7 and 3.9, $\mathbb{H}_{\text{HNF}}$ is compactly and inductively defined as follows.
Definition 3.12. We inductively define the set of formulas \( \mathcal{H}_{\text{NF}} \) exclusively from the rules below, wherein \( k \geq 1 \) and \( L \) is the set of literals.

1. \( L \subseteq \mathcal{H}_{\text{NF}} \).
2. If \( \forall i, \varphi_i \in \mathcal{H}_{\text{NF}} \) then \( \{ \varphi_1 \ldots \varphi_i \ldots \varphi_k \} \subseteq \mathcal{H}_{\text{NF}} \).
3. If \( \varphi_i \in \mathcal{H}_{\text{NF}} \) and \( \forall j \neq i, \varphi_j \in \mathcal{N}_G \) then \( (\lor \varphi_1 \ldots \varphi_i \ldots \varphi_k) \subseteq \mathcal{H}_{\text{NF}} \).

We prove next that \( \mathcal{H}_{\text{NF}} \) coincides with \( \mathcal{H}_{\text{HNF}} \), that is, Definition 3.12 corresponds to the recursive and compact definition of \( \mathcal{H}_{\text{HNF}} \). Besides, the specification in Definition 3.12 yields an optimal strategy to decide whether an NNF is HNF, which is proven in Subsection 5.1.

Theorem 3.13. We have: \( \mathcal{H}_{\text{NF}} = \mathcal{H}_{\text{HNF}} \).

Proof. We prove first \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \) and then \( \mathcal{H}_{\text{HNF}} \supseteq \mathcal{H}_{\text{NF}} \). The first relation \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \) is proven by structural induction, where (1) \( L \subseteq \mathcal{H}_{\text{NF}} \) holds trivially, as follows.

2. The non-recursive \( \mathcal{H}_{\text{NF}} \) conjunctions are literal conjunctions, which trivially verify Definition 3.3 and so are in \( \mathcal{H}_{\text{HNF}} \). Further, assuming that for an induction step \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \) holds and that \( \varphi_i \in \mathcal{H}_{\text{NF}}, 1 \leq i \leq k \), then any formula \( \{ \varphi_1 \ldots \varphi_i \ldots \varphi_k \} \) added in (2) to \( \mathcal{H}_{\text{NF}} \) belongs also to \( \mathcal{H}_{\text{HNF}} \) by Lemma 3.7. So \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \) holds.

3. Assuming that for a given recursive level \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \) holds, in the next recursion, only disjunctions \( \varphi \) in (3) are added to \( \mathcal{H}_{\text{NF}} \). But the condition of (3) and that of Lemma 3.9 are equal; so by Lemma 3.9 \( \varphi \) is in \( \mathcal{H}_{\text{HNF}} \) too. Therefore \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \) holds.

- \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \). Given that the structures to define \( \mathcal{H}_{\text{NF}} \) and \( \mathcal{H}_{\text{HNF}} \) in Definition 2.3 and Definition 3.12 respectively, are equal, the potential inclusion of each NNF formula \( \varphi \) in \( \mathcal{H}_{\text{NF}} \) is systematically considered. Further, the statement: if \( \varphi \in \mathcal{H}_{\text{NF}} \) then \( \varphi \in \mathcal{H}_{\text{HNF}} \), is proven by structural induction on the depth of formulas and by using a reasoning similar to that of the previous \( \mathcal{H}_{\text{NF}} \subseteq \mathcal{H}_{\text{HNF}} \) case and also by using Lemmas 3.7 and 3.9. ■

Example 3.14. We regard again \( \varphi \) and \( \varphi' \) from Example 3.11

- By (3), \( (\lor \overline{A} C) \in \mathcal{H}_{\text{HNF}}. \)
- By (2), \( (\land D (\lor A \overline{B})) \in \mathcal{H}_{\text{HNF}}. \)
- By (3), \( \varphi = (\lor \overline{A} \phi) \in \mathcal{H}_{\text{HNF}}. \)

\( \varphi' = (\lor A \phi) \notin \mathcal{H}_{\text{HNF}}. \)

Example 3.15. We now analyze a more complete NNF, concretely the one from the Introduction. Let us consider \( \varphi \) below wherein \( \phi_1, \phi_2 \) and \( \phi_3 \) are NCs:

\( \varphi = \{ \land A (\lor \overline{B} \{ \land (\lor \overline{B} \overline{C} A) (\lor \phi_1 \{ \land \phi_2 \overline{A} \}) B \}) \} \phi_3 \)

We check under which conditions verified by \( \phi_1, \phi_2 \) and \( \phi_3 \), \( \varphi \) is indeed HNF. The disjunctions of \( \varphi \) and the proper \( \varphi \) can be rewritten as follows:

- \( \omega_1 = (\lor \overline{B} \overline{C} A). \)
- \( \omega_2 = (\lor \phi_1 \{ \land \phi_2 \overline{A} \}). \)
- \( \omega_3 = (\lor \overline{B} \{ \land \omega_1 \omega_2 B \}). \)

\( \varphi = \{ \land A \omega_3 \phi_3 \}. \)

We analyze one-by-one such disjunctions and finally the proper \( \varphi \):
• $\omega_1$: Trivially, $\omega_1$ is Horn.
• $\omega_2$: $\omega_2$ is HNF if $\phi_1, \phi_2 \in H_{\text{NF}}$ and if at least one of $\phi_1$ or $\phi_2$ is negative.
• $\omega_3$: $\omega_3$ is HNF if $\omega_2 \in H_{\text{NF}}$ (as $\omega_1 \in H_{\text{NF}}$).
• $\phi$: $\phi$ is HNF if $\omega_2 \in H_{\text{NF}}$ (see previous line) and $\phi_3 \in H_{\text{NF}}$.

Recapitulating, the second ($\omega_2$) and fourth ($\phi$) conditions entail that $\phi$ is HNF only if $\phi_1$, $\phi_2$ and $\phi_3$ are HNF and if at least one of $\phi_1$ or $\phi_2$ is negative. Since the first condition is subsumed by Proposition 3.6, we can conclude that $\phi$ is HNF only if its sub-formulas are HNF and if at least one of $\phi_1$ or $\phi_2$ is negative.

3.2 Relating $H_{\text{NF}}$ to the Horn and NNF Classes

We prove that $H_{\text{NF}}$ and the Horn class are semantically equivalent and specify the NNF fragment that forms $H_{\text{NF}}$. To facilitate the reading of the section, the proofs of the theorems are relegated to Subsection 3.4. A new simple concept is introduced next.

Definition 3.16. For every $\varphi \in N_{\text{NF}}$, we define $cl(\varphi)$ as the unique clausal formula that results from applying $\vee/\wedge$ distributivity to $\varphi$ until a clausal formula, viz. $cl(\varphi)$, is obtained. We will call $cl(\varphi)$ the clausal form of $\varphi$.

Example 3.17. Applying $\vee/\wedge$ distributivity to $\varphi_1$ in Example 3.4, one obtains:

$$cl(\varphi_1) = \{\wedge (\vee \overline{B} C) (\vee \overline{B} A) (\vee \overline{D} C) (\vee \overline{D} A)\}.$$ 

Proposition 3.18. We have $\varphi \equiv cl(\varphi)$.

The proof is trivial. We next show, in Theorem 3.19, Corollary 3.20 and Theorem 3.21 that $cl(\varphi)$ allows to relate $H_{\text{NF}}$ to the classes Horn and NNF.

Theorem 3.19. The clausal form of all HNF formulas is Horn, i.e. $\forall \varphi \in H_{\text{NF}} : cl(\varphi) \in H$.

Proof. See Subsection 3.4.

Theorem 3.19 and Proposition 3.18 yield the next semantical characterization of $H_{\text{NF}}$.

Corollary 3.20. The classes $H_{\text{NF}}$ and $H$ are semantically equivalent: each formula in a class is equivalent to another formula in the other class.

Proof. On the one hand, by Proposition 3.18 and Theorem 3.19 for every HNF formula $\varphi$, we have $\forall \varphi \in H_{\text{NF}} : \varphi \equiv cl(\varphi) \in H$. On the other hand, $H \subset H_{\text{NF}}$.

Corollary 3.20 entails that $H_{\text{NF}}$ can represent the same Boolean functions that the Horn class, viz. the Horn functions [13] (see Section ??). The next theorem specifies which NNF formulas are included in $H_{\text{NF}}$.

Theorem 3.21. All NNF formulas $\varphi$ whose clausal form is Horn are HNF, namely:

$$\forall \varphi \in N_{\text{NF}} \text{ if } cl(\varphi) \in H \text{ then } \varphi \in H_{\text{NF}}.$$ 

Proof. See Subsection 3.4.
Example 3.22. For $\varphi_1$ and $\varphi_2$ from Example 3.1, $cl(\varphi_1) \in \mathcal{H}$ and $cl(\varphi_2) \notin \mathcal{H}$; only $\varphi_1$ is HNF. For $\varphi$ and $\varphi'$ from Example 3.5, $cl(\varphi) \in \mathcal{H}$ and $cl(\varphi') \notin \mathcal{H}$; so only $\varphi$ is HNF.

Next Theorem 3.23 puts together previous Theorems 3.19 and 3.21 and provides a concise definition of the class of HNF formulas.

Theorem 3.23. The next statement holds:

$\forall \varphi \in \mathcal{N}_{\text{NF}} : \varphi \in \mathcal{H}_{\text{HNF}} \iff cl(\varphi) \in \mathcal{H}.$

Fig. 3 exemplifies Theorem 3.23. The classes $\mathcal{H}_{\text{HNF}}, \mathcal{H}, \mathcal{N}_{\text{NF}}$ and $\mathcal{C}$ are depicted and each $\varphi$ linked to its $cl(\varphi)$ with either a red or a blue line. Red lines: $cl(\varphi_1)$ to $cl(\varphi_3)$ are in $\mathcal{H}$; so $\varphi_1$ to $\varphi_3$ are HNF. Blue lines: $cl(\varphi_1)$ to $cl(\varphi_3)$ are not in $\mathcal{H}$; so $\varphi_1$ to $\varphi_3$ are not HNF.

Theorem 3.23 gives indeed a concise definition of $\mathcal{H}_{\text{HNF}}$ but recognizing HNFs through Theorem 3.23 is unfeasible as computing $cl(\varphi)$ takes exponential time and space. However the detailed syntactical Definition 3.12 of $\mathcal{H}_{\text{HNF}}$ will allow us (Section 5) to design an algorithmic strategy to recognize HNFs in linear time.

3.3 The Horn-NC Formulas

Next define $\mathcal{H}_{\text{NC}}$ by simply applying De Morgan’s laws to Definition 3.3 of $\mathcal{H}_{\text{NF}}$.

Definition 3.24. An NC formula $\varphi$ is in $\mathcal{H}_{\text{NC}}$ if: (A) all its disjunctions under the scope of no $\neg$-connective or of an even number of $\neg$-connectives have at most one non-negative disjunct; and (B) all its conjunctions under the scope of an odd number of $\neg$-connectives have at most one non-positive conjunct.

Example 3.25. One can check that $\varphi_2$ from Example 2.4 is HNC.

Definition 3.26. We define the set of formulas $\mathcal{H}_{\text{NC}}$ exclusively from the rules below:

- $\mathcal{H}_{\text{NF}} \subset \mathcal{H}_{\text{NC}}$.
- If $(\lor \varphi_1 \ldots \varphi_{k-1} \varphi_k) \in \mathcal{H}_{\text{NC}}$ then $\neg\{\land \neg\varphi_1 \ldots \neg\varphi_{k-1} \neg\varphi_k\} \in \mathcal{H}_{\text{NC}}$.
- If $(\land \varphi_1 \ldots \varphi_{k-1} \varphi_k) \in \mathcal{H}_{\text{NC}}$ then $\neg(\lor \neg\varphi_1 \ldots \neg\varphi_{k-1} \neg\varphi_k) \in \mathcal{H}_{\text{NC}}$.

Theorem 3.27. We have that: $\mathcal{H}_{\text{NC}} = \mathcal{H}_{\text{NC}}$.

Proof. It follows from Definition 3.24 of $\mathcal{H}_{\text{NC}}$ and Definition 3.26 of $\mathcal{H}_{\text{NC}}$, indeed both result from applying De Morgan’s laws to $\mathcal{H}_{\text{NF}}$. ■
Example 3.28. Applying De Morgan’s laws to the HNF \( \{ \land C \land (\lor A B) \land (\lor A C) \} \) yields \( \varphi_2 \) in Example 2.4 (ruling out \( F \)); so \( \varphi_2 \) is HNC.

Definition 3.29. For every \( \varphi \in \mathbb{N}_C \), we define \( cl^*(\varphi) \) as the unique clausal formula that results from applying De Morgan’s laws and \( \land / \lor \) distributivity to \( \varphi \) until a clausal formula, viz. \( cl^*(\varphi) \), is obtained.

Theorem 3.30 syntactically and semantically characterizes \( \mathbb{H}_{\mathbb{N}_C} \) just as Theorems 3.19, Theorem 3.21 and Corollary 3.20 characterized \( \mathbb{H}_{\mathbb{N}_{\mathbb{N}_F}} \).

Theorem 3.30. We have that the next relationships between \( \mathbb{H}_{\mathbb{N}_C} \) and \( \mathbb{N}_C \) and \( \mathbb{H} \):

- \( \forall \varphi \in \mathbb{N}_C : \varphi \in \mathbb{H}_{\mathbb{N}_C} \iff cl^*(\varphi) \in \mathbb{H} \).
- \( \mathbb{H}_{\mathbb{N}_C} \) and \( \mathbb{H} \) are semantically equivalent.

Proof. As Definition 3.29 of \( cl^*(\varphi) \) is obtained by just applying De Morgan’s laws to Definition 3.16 of \( \mathbb{H} \), the statements in Theorem 3.30 follow immediately from the same ones proved for \( \mathbb{H}_{\mathbb{N}_{\mathbb{N}_F}} \) in Theorem 3.19, Theorem 3.21 and Corollary 3.20 respectively.

3.4 Formal Proofs

Before proving Theorems 3.19 and 3.21, Theorem 3.31 is required. So, we prove successively: Theorem 3.31, Theorem 3.19 and Theorem 3.21.

Theorem 3.31. Let \( \varphi \) be an NNF disjunction \( \lor \varphi_1 \ldots \varphi_i \ldots \varphi_k \). \( cl(\varphi) \in \mathbb{H} \iff \varphi \) has \( k-1 \) negative disjuncts and one disjunct s.t. \( cl(\varphi_1) \in \mathbb{H} \), formally:

\[
cl(\lor \varphi_1 \ldots \varphi_i \ldots \varphi_k) \in \mathbb{H} \iff (1) \exists i, s.t. \cl(\varphi_i) \in \mathbb{H} \text{ and (2) } \forall j, j \neq i, \varphi_j \in \mathbb{N}_G.
\]

Proof. If-then. By refutation: let \( cl(\lor \varphi_1 \ldots \varphi_i \ldots \varphi_k) \in \mathbb{H} \) and prove that if (1) or (2) are violated, then \( cl(\varphi) \notin \mathbb{H} \).

- (1) \( \exists i \text{ s.t. } \cl(\varphi_i) \notin \mathbb{H} \).
  - If we take the case \( k = 1 \), then \( \varphi = \varphi_1 \).
  - But \( \cl(\varphi_1) \notin \mathbb{H} \) implies \( cl(\varphi) \notin \mathbb{H} \).

- (2) \( \exists j, j \neq i, \varphi_j \notin \mathbb{N}_G \).
  - Suppose that, besides \( \varphi_i \), one \( \varphi_j \), \( j \neq i \), has positive literals too.
  - We take a simple case, concretely \( k = 2 \), \( \varphi_1 = A \) and \( \varphi_2 = B \).
  - So, \( \varphi = (\lor \varphi_1 \varphi_2) = (\lor A B) \), which implies \( cl(\varphi) \notin \mathbb{H} \).

- Only-If. For simplicity and without loss of generality, we assume that \( (\lor \varphi_1 \ldots \varphi_i \ldots \varphi_{k-1}) = \varphi^- \in \mathbb{N}_G \) and \( \varphi_k \in \mathbb{H}_{\mathbb{N}_{\mathbb{N}_F}} \), and prove:

\[
cl(\varphi) = cl(\lor \varphi_1 \ldots \varphi_i \ldots \varphi_{k-1} \varphi_k) = cl(\lor \varphi^- \varphi_k) \in \mathbb{H}.
\]

- To obtain \( cl(\varphi) \), one must obtain first \( cl(\varphi^-) \) and \( cl(\varphi_k) \), and so

  - (i) \( cl(\varphi) = cl(\lor \varphi^- \varphi_k) = cl(\lor cl(\varphi^-) cl(\varphi_k)) \).

- By definition of \( \varphi^- \in \mathbb{N}_G \),
(ii) $cl(\varphi^-) = \{ \land D_1^- \ldots D_{m-1}^- D_m^- \}$; the $D_i^-$’s are negative clauses.

- By definition of $\varphi_k \in H_{\text{HNF}}$,

(iii) $cl(\varphi_k) = H = \{ \land h_1 \ldots h_{n-1} h_n \}$; the $h_i$’s are Horn clauses.

- By (ii) to (iii),

$$cl(\varphi) = cl((\lor \{ \land D_1^- \ldots D_{m-1}^- D_m^- \} \land h_1 \ldots h_{n-1} h_n \})).$$

- Applying $\lor/\land$ distributivity to $cl(\varphi)$ and noting $C_i = (\lor D_1^- h_i)$,

$$cl(\varphi) = cl((\land \{ \land C_1 \ldots C_i \ldots C_n \} \lor \{ \land D_2^- \ldots D_{m-1}^- D_m^- \} H) \} ).$$

- Since the $C_i = (\lor D_1^- h_i)$’s are Horn clauses,

$$cl(\varphi) = cl(\{ \land H_1 \lor \{ \land D_2^- \ldots D_{m-1}^- D_m^- \} H \} ).$$

- For $j < m$ we have,

$$cl(\varphi) = cl(\{ \land H_1 \ldots H_{j-1} H_j \lor \{ \land D_{j+1}^- \ldots D_{m-1}^- D_m^- \} H \} ).$$

- For $j = m$, $\varphi = H_1 \ldots H_m H \} = H' \in H$.

Hence $cl(\varphi) \in H$.

**Theorem 3.19.** We have $\forall \varphi \in H_{\text{HNF}} : cl(\varphi) \in H$.

**Proof.** We consider Definition 3.12 of $H_{\text{HNF}}$. The proof is done by structural induction on the depth $r(\varphi)$ of any $\varphi$ in $H_{\text{HNF}}$, formally defined below, where $\ell_i$ is a literal:

$$r(\varphi) = \begin{cases} 0 & \text{if } \varphi = \langle \odot \ell_1 \ldots \ell_{k-1} \ell_k \rangle \text{ or } \varphi = \ell. \\ 1 + \max\{r(\varphi_1), \ldots, r(\varphi_k)\} & \varphi = \langle \odot \varphi_1 \ldots \varphi_{k-1} \varphi_k \rangle. \end{cases}$$

- **Base Case:** $r(\varphi) = 0$.
  
  If $r(\varphi) = 0$ clearly $\varphi \in H$, and so $cl(\varphi) \in H$.

- **Induction hypothesis:** $\forall \varphi$, $r(\varphi) \leq n$, $\varphi \in H_{\text{HNF}}$ entails $cl(\varphi) \in H$.

- **Induction proof:** $r(\varphi) = n + 1$. By Definition 3.12 lines (2) and (3), we have:

(2) $\varphi = \{ \land \varphi_1 \ldots \varphi_i \ldots \varphi_k \} \in H_{\text{HNF}}$, where $k \geq 1$.

- By definition of $r(\varphi)$, $r(\varphi) = n + 1$ entails $1 \leq i \leq k$, $r(\varphi_i) \leq n$.

- By induction hypothesis, $\varphi_i \in H_{\text{HNF}}$ and $r(\varphi_i) \leq n$ entail $cl(\varphi_i) \in H$.

- It is obvious that, $cl(\varphi_i) = \{ \land cl(\varphi_1) \ldots cl(\varphi_i) \ldots cl(\varphi_k) \}$.

- Therefore, $cl(\varphi) = \{ \land H_1 \ldots H_i \ldots H_k \} = H \in H$.

(3) $\varphi = (\lor \varphi_1 \ldots \varphi_i \ldots \varphi_{k-1} \varphi_k) \in H_{\text{HNF}}$, where $k \geq 1$.

- By Definition 3.12 line (3), $0 \leq i \leq k - 1$, $\varphi_i \in H_{\text{HNF}}$ and $\varphi_k \in H_{\text{HNF}}$.

- By definition of $r(\varphi)$, $r(\varphi) = n + 1$ entails $r(\varphi_k) \leq n$.

- By induction hypothesis, $d(\varphi_k) \leq n$ and $\varphi_k \in H_{\text{HNF}}$ entail $cl(\varphi_k) \in H$. 

12
are straightforward, [UR] is quite elaborate and so is presented progressively. Albeit the latter has at least one negative disjunct, and so, assigning 0 to all its propositions satisfies 
Hence, to avoid such simple case we will consider that the input 

We define 

\[ H \]

as follows:

\[ H \]

\[ NC \] Unit-Resolution and the Tractability of \( H_{\mathcal{NC}} \)

We define \textit{Non-Clausal Unit Resolution}, noted \( UR_{\mathcal{NC}} \) and prove that, both, \( UR_{\mathcal{NC}} \) is complete for \( H_{NC} \) and that the satisfiability of \( H_{NC} \) is tested polynomially. The calculus \( UR_{\mathcal{NC}} \) encompasses the main rule, called \( UR \), and several simplification rules. Albeit the latter are straightforward, \( UR \) is quite elaborate and so is presented progressively.

\textbf{Remark.} If \( \varphi \) is a disjunctive HNF with \( k \geq 2 \) disjuncts, then, by Definition \ref{def:conjunctiveHNF}, \( \varphi \) has at least one negative disjunct, and so, assigning 0 to all its propositions satisfies \( \varphi \). Hence, to avoid such simple case we will consider that the input \( \varphi \) is a conjunctive HNF.
At first, we introduce UR just for almost-clausal HNFs. Assume HNF formulas with the almost-clausal pattern shown below and in which ℓ and ℓ' are any literal and its negated one, and the ϕ's and the φ's are formulas:

\[ \{ \land \varphi_1 \ldots \varphi_{l-1} \ell \varphi_{l+1} \ldots \varphi_{i-1} (\lor \phi_1 \ldots \phi_j \ell \phi_{j+1} \ldots \phi_k) \varphi_{i+1} \ldots \varphi_n \} \]

These formulas are almost-clausal because if the ϕ’s and φ’s were clauses and literals, respectively, then they would be clausal. Trivially such formulas are equivalent to the next ones: \( \{ \land \varphi_1 \ldots \varphi_{l-1} \ell \varphi_{l+1} \ldots \varphi_{i-1} (\lor \phi_1 \ldots \phi_j \phi_{j+1} \ldots \phi_k) \varphi_{i+1} \ldots \varphi_n \} \). Thus, one can derive the simple inference rule on the left, below. By noting \( D(\bar{t}) = (\lor \phi_1 \ldots \phi_j \phi_{j+1} \ldots \phi_n) \), such rule can be concisely written as indicated on the right:

\[ \frac{\ell \land (\lor \phi_1 \ldots \phi_j \ell \phi_{j+1} \ldots \phi_k)}{\ell \land (\lor \ell D(\bar{t}))} \text{ UR} \]

We now extend our analysis from formulas with the previous pattern \( \ell \land (\lor \ell D(\bar{t})) \) to those with pattern \( \ell \land (\lor C(\bar{t}) D(\bar{t})) \) where \( C(\bar{t}) \) is the maximal sub-formula linked to \( \ell \) that becomes false when \( \bar{t} \) is false, i.e. \( C(\bar{t}) \) is the maximal sub-formula equivalent to a conjunction \( \bar{t} \land \psi \). For instance, if \( \varphi \) has the sub-formula \( (\lor \varphi_1 \{ \land \varphi_1 \{ \land \bar{t} (\lor \phi_2 \bar{A}) \} \varphi_3 \}) \varphi_2 \), we take \( C(\bar{t}) = \{ \land \varphi_1 \{ \land \bar{t} (\lor \phi_2 \bar{A}) \} \varphi_3 \} \) because: (a) \( C(\bar{t}) \) verifies \( \bar{t} \land \psi = \bar{t} \land \{ \land \varphi_1 (\lor \phi_2 \bar{A}) \} \varphi_3 \); and (b) no bigger sub-formula verifies (a). It is easily checked that, the augmented UR for the pattern \( \ell \land (\lor C(\bar{t}) D(\bar{t})) \) is \( \ell \land (\lor C(\bar{t}) D(\bar{t})) \vdash D(\bar{t}) \).

**Example 4.1.** Let us consider the formula below where the \( \phi_i \)'s are HNF formulas:

\[ \varphi = \{ \land A (\lor \bar{C} \{ \land \bar{A} (\lor E \bar{D}) \}) \phi_1 \} (\lor D \{ \land \phi_2 \bar{A} \}) \varphi_3 \].

If we take the left-most \( \bar{A} \), then \( \varphi \) has a sub-formula with pattern \( A \land (\lor C(\bar{A}) D(\bar{A})) \), in which \( C(\bar{A}) = \{ \land A (\lor E \bar{D}) \} \) and \( D(\bar{A}) = (\lor \bar{C} \phi_1) \). By applying UR to \( \varphi \), we obtain:

\[ \varphi' = \{ \land A (\lor \bar{C} \phi_1) (\lor D \{ \land \phi_2 \bar{A} \}) \} \varphi_3 \].

The pattern of the almost-clausal HNFs can be rewritten \( \{ \land \Pi (\lor C(\bar{t}) D(\bar{t})) \Pi' \} \), where \( \Pi \) and \( \Pi' \) represent concatenations of formulas, i.e. \( \Pi = \varphi_1 \ldots \varphi_{i-1} \) and \( \Pi' = \varphi_{i+1} \ldots \varphi_k \). Then it is not hard to check that the general nested HNFs to which the rule UR can indeed be applied, must have the next syntactical pattern:

\[ \{ \land \Pi_0 \ell \{ \land \Pi_1 \ldots \{ \land \Pi_k \{ \lor C(\bar{t}) D(\bar{t}) \} \Pi'_k \} \ldots \Pi'_1 \} \Pi'_0 \} \]

where the \( \Pi_j \)'s and \( \Pi'_j \)'s are concatenations of HNF formulas, namely \( \Pi_j = \varphi_{j_1} \ldots \varphi_{j_{i-1}} \) and \( \Pi'_j = \varphi_{j_{i+1}} \ldots \varphi_{j_n} \), and the rule UR can be expressed as follows:

\[ \frac{\ell \land \{ \land \Pi_1 \ldots \{ \land \Pi_k \{ \lor C(\bar{t}) D(\bar{t}) \} \Pi'_k \} \ldots \Pi'_1 \}}{\{ \land \Pi_1 \ldots \{ \land \Pi_k \{ \lor C(\bar{t}) D(\bar{t}) \} \Pi'_k \} \ldots \Pi'_1 \} \text{ UR} \]

**UR means:** if the input has a literal \( \ell \) conjunctively linked to a sub-formula with the pattern of the right conjunct in the numerator, then the latter can be replaced with the formula in the denominator. In practice, applying UR is equivalent to remove \( C(\bar{t}) \). If we

---

2 The notation \( \langle \odot \varphi_1 \ldots \varphi_k \rangle \) was introduced in Definition 4.3 bottom.
denote $\Pi$ the right conjunct of the numerator and by $\Pi \cdot (\lor C(\bar{\ell}) \; D(\bar{\ell}))$ that $(\lor C(\bar{\ell}) \; D(\bar{\ell}))$ is a sub-formula of $\Pi$, then UR can be expressed:

$$\frac{\ell \land \Pi \cdot (\lor C(\bar{\ell}) \; D(\bar{\ell}))}{\Pi \cdot D(\bar{\ell})}$$

$\text{UR}$

Example 4.2. Let us consider that the following formula is HNF:

$$\varphi = \{ \land (\lor C \phi_1) \; (\lor A \{ \land (\lor A \overline{C}) \; (\lor \phi_2 \{ \land \overline{B} \; A \}) \; C \}) \; A \}.$$ 

Its associated DAG (tree, in this case) is depicted in Fig. 5, left.

For $A$ and the right-most $\overline{A}$, the formulas in UR are the following:

- $\Pi = (\lor A \{ \land (\lor A \overline{C}) \; (\lor \phi_2 \{ \land \overline{B} \; A \}) \; C \})$
- $(\lor C(\overline{A}) \; D(\overline{A})) = (\lor \phi_2 \{ \land \overline{B} \; A \})$
- $C(\overline{A}) = \{ \land \overline{B} \; A \}$ and $D(\overline{A}) = \phi_2$.

Then, applying UR to $\varphi$ leads $\varphi'$ below and whose tree is the right one in Fig. 5:

$$\varphi' = \{ \land (\lor C \phi_1) \; (\lor A \{ \land (\lor A \overline{C}) \; (\lor \phi_2 \; C \}) \; A \}$$

\square

- **Simplification Rules.** The simplification rules given below must accompany UR.

The first two rules simplify formulas by (upwards) propagating $(\lor)$ from sub-formulas to formulas; $\varphi$ is the input HNF, and as before, $\varphi \cdot \phi$ means that $\phi$ is a sub-formula of $\varphi$:

$$\frac{\varphi \cdot (\lor \phi_1 \ldots \phi_{i-1} (\lor \phi_{i+1} \ldots \phi_k)}{\varphi \cdot (\lor \phi_1 \ldots \phi_{i-1} \phi_{i+1} \ldots \phi_k)} F_\lor$$
$$\frac{\varphi \cdot \{ \land \varphi_1 \ldots \varphi_{i-1} (\lor \varphi_{i+1} \ldots \varphi_k \}}{\varphi \cdot (\lor \varphi \) F_\land$$

The next two rules remove redundant connectives; the first one removes a connective $\circ$ if it is applied to a single formula, i.e. $(\circ \phi_1)$, and the second one removes a connective that is inside another equal connective:

$$\frac{\varphi \cdot (\circ_1 \varphi_1 \ldots \varphi_{i-1} (\circ_2 \phi_1 \ldots \phi_k) \varphi_{i+1} \ldots \varphi_k)}{\varphi \cdot (\circ_1 \varphi_1 \ldots \varphi_{i-1} \phi_1 \varphi_{i+1} \ldots \varphi_k) \circ \phi}$$
$$\frac{\varphi \cdot (\circ_1 \varphi_1 \ldots \varphi_{i-1} (\circ_2 \phi_1 \ldots \phi_n \varphi_{i+1} \ldots \varphi_k)) \circ_1 = \circ_2}{\varphi \cdot (\circ_1 \varphi_1 \ldots \varphi_{i-1} \phi_1 \ldots \phi_n \varphi_{i+1} \ldots \varphi_k) \circ \circ}$$
Example 4.3. We continue with Example 4.2. Picking the left-most \( \overline{A} \) (colored blue in Fig. 5, right), we have \( C(\overline{A}) = \overline{A} \), \( D(\overline{A}) = (\lor \{ \land (\lor \overline{A} C) \land \phi_2 \} \land C) \), and \( \Pi = (\lor C(\overline{A}) D(\overline{A})) \). By applying UR to \( \varphi' \), the obtained formula is given in Fig. 6, left. After two applications of \( \odot \phi \) and one of \( \odot \odot \), one gets the formula associated with the right tree in Fig. 6. Finally, two applications of UR to the two pairs \( C \) and \( \overline{C} \), and \( A \) and \( \overline{A} \), and then one application of \( F \land \) leads the calculus UR nc to derive \( \lor \).

![Diagram](image_url)

**Fig. 6. Example 4.3**

Definition 4.4. We define UR nc as the calculus formed by the UR rule and the above described simplification rules, i.e. \( UR_{nc} = \{ UR, F \lor, F \land, \odot \phi, \odot \odot \} \).

Lemma 4.5. An HNF \( \varphi \) is unsatisfiable iff UR nc applied to \( \varphi \) derives \( \lor \).

The proof is rather straightforward and contains some tedious details, and so it is omitted by space reasons.

Lemma 4.6. HNF-SAT is polynomial.

*Proof.* The number of UR and simplification rules performed is at most the number of literals and connectives in \( \varphi \), respectively. On the other hand, it is not difficult to find data structures to execute polynomially each rule in UR nc. Hence, the lemma holds. \( \blacksquare \)

Proposition 4.7. HNF-SAT is P-complete.

*Proof.* It follows straightforwardly from HNF-SAT is polynomial, by Lemma 4.6 and HNF-SAT includes Horn-SAT, which is P-complete \([46, 31]\). \( \blacksquare \)

Final Remark. We believe that it will be not hard to find data structures to devise an HNF-SAT algorithm based on UR nc with linear complexity. Since the HNF-SAT algorithm is NC Unit-Propagation, on its basis, effective NC DPLL solvers can be developed.

4.1 Beyond NC Unit-Resolution

(1) NC Hyper-Unit-Resolution. The given definition of the NC Unit-Resolution rule can be extended to obtain NC Hyper-Unit-Resolution (HUR). Assume that the \( \varphi \) contains a conjunction of \( \ell \) with two sub-formulas \( (\lor C(\overline{\ell}) D(\overline{\ell})) \) and \( (\lor C(\overline{\ell}^2) D(\overline{\ell}^2)) \), where \( \overline{\ell} \) denotes a specific occurrence of \( \overline{\ell} \). The simultaneous application of NC unit-resolution with two sub-formulas is formally expressed as follows:

\[
\ell \land \Pi^1 \succ (\lor C(\overline{\ell}^1) D(\overline{\ell}^1)) \land \Pi^2 \succ (\lor C(\overline{\ell}^2) D(\overline{\ell}^2)) \\
\Pi^1 \succ D(\overline{\ell}^1) \land \Pi^2 \succ D(\overline{\ell}^2)
\]
If the sub-formula $\Pi^i \gg (\lor C(\overline{\ell}) \ D(\overline{\ell}))$ is denoted $(\Pi, CD(\overline{\ell}))^i$, then HUR for $k$ sub-formulas is formally expressed as follows:

$$\ell \land (\Pi, CD(\overline{\ell}))^1 \land \ldots \land (\Pi, CD(\overline{\ell}))^i \land \ldots \land (\Pi, CD(\overline{\ell}))^k \quad \text{HUR}$$

Since the $\overline{\ell}$'s are pairwise different, so are the sub-formulas $CD^i$ and $D^i$ in HUR. However, the formulas $\Pi^i$ are not necessarily different as it is shown next:

**Example 4.8.** Let us reconsider also the formula in previous Example 4.2:

$$\phi = \{ \land (\lor C \phi_1) (\lor \overline{\ell}^i \ \{ \land (\lor \overline{\ell}^i \ \overline{C}) (\lor \phi_2 \ \{ \land B \overline{A}^i ) \ C \}) \} \ A \}.$$  

One can apply HUR with $A$ and the three literals $\overline{\ell}^i$. The formula $\Pi$ in the numerator of HUR is the same for the three literals, so it is noted $\Pi^{1,2,3}$ below, but the formulas $(\lor C(\overline{A}) \ D(\overline{A}))$ are different and are as follows:

$$- \Pi^{1,2,3} = (\lor \overline{A}^i \ \{ \land (\lor \overline{A}^i \ \overline{C}) (\lor \phi_2 \ \{ \land B \overline{A}^i ) \ C \})$$

- $(\lor C(\overline{A}^1) \ D(\overline{A}^1)) = \Pi^{1,2,3}$

- $(\lor C(\overline{A}^2) \ D(\overline{A}^2)) = (\lor C(\overline{A}^2) \ D(\overline{A}^2))$

- $(\lor C(\overline{A}^3) \ D(\overline{A}^3)) = (\lor \phi_2 \ \{ \land \overline{A}^3 \})$

Applying NC Hyper Unit-Resolution yields: $\{ \land (\lor R \phi_1) (\lor \{ \land \overline{\ell} \ (\lor \overline{\ell} \ \overline{R} \ (\lor \phi_2 \ R) \}) P \}$

After simplifying: $\{ \land (\lor C \phi_1) \overline{C} \phi_2 \ C \ A \}$. Clearly, a simple NC unit-resolution deduces $(\lor)$. Altogether, the rule HUR accelerates considerably the proof of unsatisfiability.

**Remark.** An NC hyper unit-resolution rule even more general than HUR can be devised to handle simultaneously $k \geq 1$ unit-clauses, $\ell_k$, so that for each $\ell_k$, $i \geq 1$ sub-formulas $\Pi^{k,i} \gg (\lor C(\neg \ell^k,i) \ D(\neg \ell^k,i))$ can be covered. That is, one can compute simultaneously $k \geq 1$ applications of HUR (but for space reasons, we omit its formal definition).

**2 NC Local-Unit-Resolution.** The UR rule can also be applied to sub-formulas in order to accelerate the searching of proofs, i.e. to allow for shorter proofs. Thus, if a sub-formula $\phi$ of $\varphi$ has the pattern of UR, UR can be applied to $\phi$. Namely, applying UR to any sub-formula of $\varphi$ with pattern $\phi = \ell \land \Pi \cdot (\lor C(\overline{\ell}) \ D(\overline{\ell}))$ can be authorized and $\phi$ replaced with $\ell \land \Pi \cdot D(\overline{\ell})$. So, Local NC Unit-Resolution, LUR, is formalized as follows:

$$\varphi \cdot (\ell \land \Pi \cdot (\lor C(\overline{\ell}) \ D(\overline{\ell}))) \quad \text{LUR}$$

**LUR means:** if the input $\varphi$ has a sub-formula with pattern $\ell \land \Pi \cdot (\lor C(\overline{\ell}) \ D(\overline{\ell}))$, the latter can be replaced with $\ell \land \Pi \cdot D(\overline{\ell})$, or analogously, the formula $C(\overline{\ell})$ can be eliminated.

**Example 4.9.** Consider again Example 4.2. Its sub-formula $\{ \land (\lor \overline{A} \ C) \ (\lor \phi_2 \ \{ \land \overline{B} \overline{A} \}) \ C \}$ has the pattern $\ell \land \Pi \cdot (\lor C(\overline{\ell}) \ D(\overline{\ell}))$, and so LUR can be applied to $\varphi$ as follows:

$$\varphi \cdot (C \land (\overline{\ell} \ D(\overline{\ell}))) \Rightarrow \varphi \cdot (C \land (\overline{A} \ D(\overline{\ell}))).$$

After applying LUR, one obtains:

$$\{ \land (\lor C \phi_1) \ (\lor \overline{A} \ \{ \land \overline{A} \ (\lor \phi_2 \ \{ \land \overline{B} \overline{A} \}) \ C \}) \ A \}.$$
Now applying HUR with $A$ and the first two $\overline{A}$'s, one determines that $\varphi$ is unsatisfiable. □

**Proposition 4.10.** If applying $LUR$ to any $NNF$ formula $\varphi$ results in $\varphi'$ then $\varphi$ and $\varphi'$ are logically equivalent.

The proof follows straightforwardly from the soundness of UR.

Observe that HUR and LUR habilitate shorter proofs. For instance, for Example 4.2 only one LUR and one HUR are enough to derive $(\lor)$. Thus, their suitable management can make increase the overall deductive efficiency.

### 4.2 Application to Logic Programming

The introduction of NC formulas in the field of logic programming, concretely within answer set programming, was done in [111]. Programs containing NCs in their bodies and heads are called nested programs and have been employed in many issues e.g. normal forms [30], extension to other logics [122], allowing for highly-expressive predicates (aggregates) [34], etc. However, nested definite programs, the equivalent to classical definite programs, have not been characterized. We show that the HNF formulas are indeed the base of the nested definite programs.

**Definition 4.11.** An HNF rule is an expression $\Pi^+ \rightarrow \Phi$ wherein $\Pi^+$ is an NNF having only positive literals and $\Phi$ is an arbitrary HNF. An HNF program is a set of HNF rules.

**Example 4.12.** The nest rule is an HNF rule because its body is a positive NNF and its head is an HNF:

$$\{\land A D \} \ (\lor \ E \ \{\land D C \}) \ \rightarrow \ \{\land \ (\lor \ \overline{A} C) \ \{\land D \ (\lor \ A \ \overline{B}) \} \}$$

**Proposition 4.13.** An HNF logic program is an HNF formula.

*Proof.* Clearly a rule $(\lor \ \neg \Pi^+ \ \Phi)$ satisfies Definition 3.3 and hence so does a conjunction of them, i.e. an HNF logic program. ■

**Lemma 4.14.** Let $P$ be a proposition set, $Lp$ an HNF logic program and $\varphi$ any arbitrary NNF. Deciding whether $P \land Lp \models \varphi$ is polynomial.

*Proof.* By Lemma 4.6 one can polynomially check whether $P \land Lp$ is satisfiable. If so, by applying $UR_{nc}$ one can polynomially determine the positive literals that follow logically from $P \land Lp$, i.e. its minimal model. Finally, one can also polynomially check whether, for such minimal model, $\varphi$ evaluates to 1 and so whether $P \land Lp \models \varphi$. ■

Given Corollary 3.20 each HNF program is equivalent to a clausal definite program, thus one can say that the HNF programs are the nested definite programs. On the other side, Example 4.12 visualizes the potentiality of HNF to enrich declarative rules in logic programming. As mentioned previously, we think that future work will allow to find data-structure to linearly test the satisfiability of $HNF$. This would imply that the complexity of propositional HNF logic programs is linear, that is, expressiveness can be significantly augmented while keeping the same linear complexity of clausal logic programming [46, 31, 155]. Besides, HNF logic programs are much smaller than their equivalent Horn programs.
5 Additional Properties of $H_{NC}$

This section proves three additional properties of $H_{NC}$.

5.1 Linear Recognition of $H_{NC}$

We present a recognition algorithm, $HNF$, to determine whether an NNF is HNF and also prove its correctness and linearity. Checking whether an NC $\varphi$ is HNC is performed by first translating $\varphi$ into an equivalent NNF $\varphi'$, and then, by applying $HNF$ to $\varphi'$. That translation can be achieved in linear-time by merely pushing in the negations.

- **HNF’s Principle.** We call disjunctions and conjunctions of literals, e.g. $(\lor A \lor B)$, flat-formulas. $HNF$ uses two fresh literals, $\ell^+$ and $\ell^-$, treating them as positive and negative, respectively, and replaces flat-formulas of the input with $\ell^+$ or $\ell^-$ as described below:

  1. The strategy to bottom-up replace flat-formulas is as follows. In a given step, $HNF$ replaces the flat-formulas with $\ell^+$ or $\ell^-$. So, sub-formulas containing only literals and flat-formulas are flattened and so recursively replaced in the next step.

  2. Below in (a)-(e), we enumerate the five possible cases encountered to replace a given flat-formula $\phi$ with either $\ell^+$ or $\ell^-$. $pos(\phi)$ is the number of positive literals of $\phi$. If case (a) holds, then $HNF$ halts and signals that the input is non-HNF.

- If $\phi$ is a clause, then $HNF$ operates as follows:

  (a) $pos(\phi) > 1 \rightarrow$ halts and returns non-HNF.

  (b) $pos(\phi) = 1 \rightarrow$ replaces $\phi$ with $\ell^+$.

  (c) $pos(\phi) = 0 \rightarrow$ replaces $\phi$ with $\ell^-$.

- If $\phi$ is a conjunction, then $HNF$ acts as follows:

  (d) $pos(\phi) > 0 \rightarrow$ replaces $\phi$ with $\ell^+$.

  (e) $pos(\phi) = 0 \rightarrow$ replaces $\phi$ with $\ell^-$.

- **Pseudo-Code.** The required data-structure is: An NNF is implemented by associating to each sub-formula $\varphi$: (1) a pointer $[\varphi]$ that points to the address where the linked structure of symbols constituting $\varphi$ begins, i.e. to the root of DAG $D_\varphi$. $D_\varphi$ is stored only once and each $\varphi$-occurrence in the NNF is replaced with $[\varphi]$; and (2) an integer $pos(\varphi)$ that, when $\varphi$ is flattened, is set to the number of positive literals in $\varphi$. The pseudo-code of $HNF$ is given above; the right column indicates cases (a)-(e) in the above $HNF$’s principle corresponding to the line in question.

$$HNF([\varphi]).$$

If $[\varphi] = (\lor \ell_1 \ldots \ell_i \ldots \ell_k)$ then do:

- $pos(\varphi) \leftarrow$ number of positive-literals in $\varphi$.
- if $pos(\varphi) > 1$ then halt and return non-HNF. : (a)
- if $pos(\varphi) = 1$ then return($\ell^+$). : (b)
- if $pos(\varphi) = 0$ then return($\ell^-$). : (e)
If \( \varphi = \{ \land \ell_1 \ldots \ell_i \ldots \ell_k \} \) then do:
\[
\text{pos}(\varphi) \leftarrow \text{number of positive-literals in } \varphi.
\]
\[
\text{if } \text{pos}(\varphi) > 0 \text{ then return}(\ell^+).
\]
\[
\text{if } \text{pos}(\varphi) = 0 \text{ then return}(\ell^-).
\]
\[
\text{else for } 1 \leq i \leq k, \varphi_i \neq \ell_i \text{ do: } \varphi_i \leftarrow \text{HNF}([\varphi_i]).
\]
\[
\text{Return } \text{HNF}([\varphi]).
\]

Example 5.1. Next, we run HNF on \( \varphi \) below. Its flat-formulas are \( \phi_1 = (\lor \overline{A} \ E), \)
\[
\varphi = \{ \land \overline{C} \ (\lor \overline{A} \ E) \ (\lor (\land \overline{F} \overline{C} \overline{E}) \ (\land A B \}) \}.
\]
\( \phi_2 = \{ \land \overline{F} \overline{C} \} \) and \( \phi_3 = \{ \land A B \} \), and their associated replacements are:
\[
1. \text{pos}(\phi_1) = 1 \rightarrow \text{case (b)} \rightarrow \phi_1 \text{ is replaced with } \ell^+.
2. \text{pos}(\phi_2) = 0 \rightarrow \text{case (e)} \rightarrow \phi_2 \text{ is replaced with } \ell^-.
3. \text{pos}(\phi_3) = 2 \rightarrow \text{case (d)} \rightarrow \phi_3 \text{ is replaced with } \ell^+.
\]
- After the replacements, one obtains: \( \{ \land \overline{C} \ell^+ (\lor (\land \ell^- \overline{E}) \ell^+) \} \).
- Replacing its flat-formula \( (\lor \ell^- \overline{E}) \) with \( \ell^- \) yields: \( \{ \land \overline{C} \ell^+ (\lor \ell^- \ell^+) \} \).
- Replacing its flat-formula \( (\lor \ell^- \ell^+) \) with \( \ell^+ \) yields: \( \{ \land \overline{C} \ell^+ \ell^+ \} \).
- Here, HNF returns \( \ell^+ \), indicating that \( \varphi \) is non-negative HNF.

Example 5.2. We now run HNF on the formula: \( \{ \land \overline{C} \ (\lor \{ \land \ A \overline{C} \ E \}) \} \).
- Replacing \( \{ \land \ A \overline{C} \} \) with \( \ell^+ \) leads to \( \{ \land \overline{C} (\lor \ell^+ E) \} \).
- As \( \text{pos}(\lor \ell^+ E) > 1 \), condition (a) holds and HNF halts returning non-HNF.

Theorem 5.3. Correctness of HNF: \( \varphi \) is non-HNF iff HNF(\( \varphi \)) returns non-HNF.

Proof. We denote \( \text{nn} (\phi_{\text{init}}) \) the number of non-negative disjuncts of any initial disjunctive sub-formula \( \phi_{\text{init}} \) of the input \( \varphi \).
- Steps (b) to (e). A flat-formula \( \phi \) is replaced with \( \ell^+ \) iff \( \phi \) has some positive literal. Assume that \( \phi' \) is the father of \( \phi \). As both \( \ell^+ \) and \( \phi \) are non-negative, \( \text{nn}(\phi') = \text{nn}(\phi) \).
- Sub-formulas are recursively flattened, all initial \( \text{nn}(\phi_{\text{init}}) \) are recursively preserved. Hence if \( \phi_{\text{flat}} \) is the initial \( \phi_{\text{init}} \) after having been flattened, then \( \text{pos}(\phi_{\text{flat}}) = \text{nn}(\phi_{\text{init}}) \).
- Step (a). Assume that (a) sometime holds. If HNF halts and returns non-HNF, then some \( \text{pos}(\phi_{\text{flat}}) > 1 \) is encountered. But \( \text{pos}(\phi_{\text{flat}}) = \text{nn}(\phi_{\text{init}}) \) and \( \text{pos}(\phi_{\text{flat}}) > 1 \) entail that \( \exists \phi_{\text{init}} \), \( \text{nn}(\phi_{\text{init}}) > 1 \), and so, by Definition 3.3, the input \( \varphi \) is not HNF. Hence, if HNF returns non-HNF, \( \varphi \) is non-HNF. On the contrary, supposing that the hypothesis (a) is never met, then by following a reasoning dual to the previous, one obtains that if HNF returns \( \ell^+ \) or \( \ell^- \), then \( \varphi \) is HNF.

Theorem 5.4. Linearity of HNF: HNF(\( \varphi \)) ends in time \( O(\text{size}(\varphi)) \).

Proof. The running time of HNF is the sum of the times for building and scanning (applying cases (a)-(e)) the DAG associated to \( \varphi \). Building: pointers are initialized and counters allocated; both in constant time; so building is linear. Scanning: (i) arcs are traversed at most once upwards and once downwards. (ii) when a sub-formula is flattened, HNF counts its positive literals and set \( \text{pos}(\phi) \) accordingly, which is done at most once and takes \( O(k) \) time, \( k \) being the arity of \( \phi \); so (ii) is linear. (iii) replacing a flat-formula with \( \ell^+ \) or \( \ell^- \) is done at most once and in constant time. Hence, the overall cost is linear.

20
5.2 Succinctness and Syntactical Richeness

Succinctness was introduced in [75] and is given in Definition 5.5. It measures spatial efficiency of formula classes for knowledge modeling and has been used in many works.

**Definition 5.5.** Let $X$ and $Y$ two formula classes. $X$ is at least as succinct as $Y$, noted $X \leq Y$, iff for every $\varphi \in Y$ there exists $\varphi' \in X$ s.t. $\varphi \equiv \varphi'$ and the size of $\varphi'$ is polynomially bounded in that of $\varphi$. $X$ is strictly more succinct than $Y$ iff $X \leq Y$ and $Y \not\leq X$.

**Proposition 5.6.** $H_{NC}$ is strictly more succinct than $H$.

**Proof.** Clearly $H_{NC} \leq H$ because $H_{NC} \supseteq H$. To prove that $H \not\leq H_{NC}$, it is enough to take any DNF $\varphi \in H$ with $k$ conjuncts and $n$ literals per conjunct. Thus $cl(\varphi) \equiv \varphi \in H$ and the size of $cl(\varphi)$ is $k^n$.

We now deal with syntactical richness and consider that class $X$ is richer than $Y$ simply if $X$ subsumes $Y$. As formulas express interrelations among variables, richer classes having more formulas supply more syntactical options to express interrelations among variables and so have superior syntactical capabilities for knowledge modeling. Clearly $H_{NC}$ is richer than $H$, but we check that it is even exponentially richer.

**Definition 5.7.** Let $X$ and $Y$ be two formula classes. $X$ is exponentially richer than $Y$ iff:

(i) $X \supset Y$.
(ii) $\forall \varphi' \in Y$ there are exponentially many $\varphi \in X$ s.t. $\varphi \equiv \varphi'$.

**Proposition 5.8.** $H_{NC}$ is exponentially richer than $H$.

The proof is straightforward. Note, however, that the previous proposition does not imply that $H_{NF}$ is exponentially richer than $H$.

6 Related Work

Given that $H_{NC}$ results from merging the NC and Horn formulas, we review the main literature relative to both of kind of formulas.

6.1 Non-Clausal Reasoning

**Logic Programming.** This area has been discussed in Subsection 4.2.

**Satisfiability.** Given the growing efficiency of clausal DPLL solvers, enhancing NC reasoning tended toward translating NCs into equisatisfiable clausal ones via Tseitin-like [148] translators, and then, applying DPLL solvers. Yet, the benefits of translations were questioned [146, 121, 56], which advocated both, developing approaches computing original NCs and incorporating structural information in the search. On the other side, a number of approaches for NC satisfiability exist: NC Resolution [119], NC DPLL [125, 73, 146, 94], path dissolution and anti-links [132], connection calculi [126], regular, connected tableaux [80], NC tableau-like with reductions [76], SLD-resolution [141] and general matings [93]. The experimental results showed the relative good performance of DPLL. Although several NC DPLL solvers have been implemented, NC unit resolution had not been published. On the other side, a rule called NC unit-dissolution is introduced in [120], but this is done
within the framework of other general rule (Dissolution) and it is not formalized. Our work led to formally define NC unit-resolution for the first time, prove its completeness for \( H_{\text{NC}} \) and it allowed to demonstrate the polynomiality of \( H_{\text{NC}} \).

**Incomplete Solvers.** For clausal problems, GSAT [140] demonstrated that an incomplete solver can solve many hard problems much more efficiently than complete solvers. Several authors have generalized these techniques to NCs. Sebastiani [136] suggested how to modify GSAT to be applied to NCs. Kautz et al. [96] introduced DAGSat for clausal formulas, but allowed handling of NCs without an increase in size. Later, Stachniak [142] introduced polWSAT an evolution of WalkSAT [139] to handle NCs. The authors in [118] present a new way of expressing the scoring function for evaluating partial interpretations and their solver outperforms previous complete and incomplete NC solvers. Yet, as HNCs are polynomial, designing incomplete NC solvers for them has limited interest. URnc could be relevant to develop incomplete NC-SAT solvers.

**Quantified Boolean Formulas.** An extension of the SAT problem increasingly present in research and applications, is featuring universal and existential quantifiers for boolean variables giving rise to the QBFs. In particular, developing efficient solvers for QBFs in NC form is a concern of contemporary research as evidenced by the reported performances, e.g. [113, 61, 14]. However, to the best of our knowledge, no results presented here concerning \( H_{\text{NC}} \) and URnc have been initialized for QBFs in NC.

**Max-SAT.** Some reasoning problems have been intensively studied for clausal but hardly for NC, and sometimes, even no true NC definition exists. MaxSAT is an example. The reached efficiency for MaxSAT [108] contrasts with the nonexistence of a proper definition for NC-MaxSAT. A recent work [110] defines NC-MaxSAT as the problem of, given \( k \) NCs \( \{\varphi_1, \varphi_2, \ldots, \varphi_k\} \), determining the greatest \( i, i \leq k \), of simultaneously satisfiable formulas. In such definition the only envisaged formula inter-dependence is the one derived from the shared literals which is, however, rather clausal-like. The proper NC inter-dependence is the one derived from the connexions among formulas and sub-formulas. Applications of clausal Unit Propagation include its use for improving the lower-bound during the solving of MaxSAT [109]. Thus URnc seems essential to compute efficiently NC-MaxSAT.

**Theorem Proving.** Automated reasoning in NC and FOL has been an active research field since the 1960s and is still the object of considerable research and progress, e.g. [80, 85, 126, 63]. On the other side, DPLL for clausal FOL has already been developed [18, 17]. Extending NC FOL with equality logic is a former [13] as well a recent [124] concern. Although our approach has been initially oriented towards propositional logic, it is extensible to FOL. Concretely, we plan to propose a FOL NC DPLL by lifting URnc to FOL (Future Work), which means, moreover, to generalize to clausal approach in [18, 17].

**Resolution.** In the seminal work by N. Murray [119], NC Resolution for propositional and FOL was defined in a combined functional and classical manner and has been applied to different contexts, e.g. [13, 141]. Each proper NC Resolution is followed by some functional-logical simplifications of the formula. The completeness of NC Resolution *under a linear restriction* was proved by the authors in [80]. Although NC Resolution was specified in 1982, paradoxically no NC Unit-Resolution version had been published. NC Hyper-resolution had not been defined either to our knowledge. The available definition of NC Resolution suffers form some drawbacks e.g. not clearly defining the potential resolution steps or requiring complex completeness proofs [80]. We plan to present a new
definition of NC Resolution (see future work).

**Knowledge Compilation.** It consists in translating an original formula into another equivalent one during the off-line phase. Whereas the original and target formulas are semantically equivalent, the target class present desired syntactical conditions permitting computing the clausal entailment test polynomially. This way, the on-line exponential complexity, usually present during query-answering, is confined to the off-line phase and later amortized over all on-line queries answered polynomially. NNF sub-classes, such as DNNF, d-NNF or the well-known BDD and OBDD [50], were introduced as target ones by A. Darwiche [48, 49]. \( H_{\text{NC}} \) is polynomial for satisfiability checking, i.e. for clausal entailment, but \( H_{\text{NC}} \) is incomplete since not all NNFs are of course equivalent to, and so representable by, an HNC. So, \( H_{\text{NC}} \) does not qualify as a target class. Yet in [64], the class formed by the disjunctions of Horn formulas, or \( H-[\lor] \), is introduced as a target class. Clearly, the class of disjunctions of HNCs, or \( H_{\text{NC}}-[\lor] \), subsumes \( H-[\lor] \) and is strictly more succinct, likewise \( H_{\text{NC}} \) is strictly more succinct than \( H \). Towards using \( H_{\text{NC}}-[\lor] \) as a target class, the next questions should be analyzed: how to compile knowledge into \( H_{\text{NC}}-[\lor] \); which is its complexity for consistency, clausal entailment, implicants, etc. [50]; and finally, its succinctness should be compared with existing target classes.

### 6.2 Horn Formulas

Due to the ubiquity of Horn formulas in knowledge representation and automated reasoning, and by space reasons, our related work is unavoidably incomplete. It focuses first on their presence in knowledge representation and then in automated reasoning.

#### 6.2.1 Knowledge Representation.

**Propositional logic.** As alluded to in the introduction, finding polynomial clausal super-classes of the Horn class has been a key issue on increasing clausal efficiency since 1978 [107]. Most of such extensions are presented below chronologically along with their relationships. Our research has allowed to lift the Horn formulas to NC and prove that tractable reasoning also exists in NC, but we think that this is just a first outcome and that some Horn extensions pointed out below can also be NC lifted. For instance, one can conceive the renomable-HNC formulas as those NCs whose clausal form is renomable-Horn.

- **Unit Propagation.** The so-called Unit Propagation was proved complete for Horn-SAT in [84, 70, 117]. Further publications [90, 55, 135] get this universally understood. The linearity of Horn algorithms [55, 91, 117, 135, 72, 60] gave rise to efficient implementation of Unit Propagation. Deciding if a model of a Horn formula is unique is also linear [129].

- **Hidden Horn.** A given formula is hidden Horn [107, 7] if a change in the polarity of some variables leads to an equivalent Horn formula. Recognizing hidden Horn formulas is linear [7, 37] and since Horn-SAT is linear, so is SAT-checking them.

- **\( S_0 \), Hierarchy \( \varphi_k \).** Another Horn extension was proposed in [160], called \( S_0 \), and later Generalized Horn in [87], and successively broadened in [6, 69]. Recognizing and SAT-checking \( S_0 \) formulas are quadratic [6, 69]. Reference [69] defines recursively a hierarchy \( \Sigma_0 \subseteq \Sigma_1 \ldots \subseteq \Sigma_k \ldots \) over the non-clausal classes \( \Sigma_k \), where \( \Sigma_0 \) are the Horn formulas and \( \Sigma_1 \) is \( S_0 \). This hierarchy was strengthened in [130] whose classes \( \Pi_k \) subsume the classes \( \Sigma_k \). SAT-checking \( \Sigma_k \) and \( \Pi_k \) formulas is polynomial but recognizing the renamings of \( \Sigma_k \) formulas, \( k > 1 \), is \( \mathcal{NP} \)-complete [62].
• **Q·Horn.** The Q-Horn formulas \[28\] subsume the Horn, hidden Horn and the Quadratic (2-CNF) formulas. Q-Horn formulas are recognized and SAT-checked linearly \[28, 29\].

• **Extended Horn.** The extended-Horn formulas were introduced in \[38\] based on a theorem in \[36\]. Their definition relies on Unit Propagation and matrix multiplication. Hidden extended-Horn formulas are defined in the same way as hidden Horn from Horn. SAT-checking extended-Horn formulas is linear; however, no polynomial recognition of the extended-Horn formulas is known.

• **Hierarchy Ω.** The hierarchy \(Ω\) of classes proposed in \[45\] contains the classes Horn, reverse-Horn (clauses have at most one negative literal) and hidden-Horn. The classes in \(Ω\) are recognized and SAT-checked polynomially.

• **SLUR.** The definition of the SLUR (Single Lookahead Unit Resolution) formulas \[134, 15\] is based on an algorithm rather than on properties of formulas. SAT-checking a formula \(ϕ\) is divided in two phases. First unsatisfiability is sought by applying Unit Propagation (UP). If such seeking fails, then \(ϕ\) is assumed to be satisfiable and a satisfying assignment is guessed using UP-lookahead to avoid obviously false assignments. The class SLUR contains those formulas where the algorithm succeeds in the first phase or finds a satisfying assignment in the second phase. The class SLUR subsumes the hidden-Horn, extended-Horn and other classes \[144\]. SLUR and Q-Horn are incomparable \[67\]. The SLUR formulas are efficiently SAT-checked but recognizing them is coNP-complete \[153\].

• **Quad.** The class Quad \[44\] includes the Horn and the Quadratic (2-CNF) formulas. Quad is not subsumed by the hierarchy \(Ω\), and Q-Horn and Quad are incomparable. Recognizing and SAT-checking Quad formulas take quadratic time.

• **Autarkies.** If a partial assignment satisfies all those clauses we thinks of \(ϕ\) affected by it, i.e., no new clauses are created, the resulting formula is an autarky, e.g. the simplest autarkies are the pure literals. Therefore, \(ϕ\) is satisfiable iff its autarky is. The class of autarkies include Q-Horn and is incomparable with the class SLUR \[151\]. Finding the specific linear autarkies is polynomial \[104\] and there exist partial autarkies that applied repeatedly result in a unique, autarky-free formula.

• **Matched.** The matched formulas were analyzed initially in \[147\] in order to provide a benchmark for testing some previously published formulas but were considered rather useless. However, although the class matched is incomparable with Q-Horn and SLUR, w.r.t. frequency of occurrence on random formulas, reference \[67\] verified that matched is far more common than both those classes together. Matched formulas are always satisfiable and trivially SAT-checkable.

• **Minimally Unsatisfiable.** Several classes of unsatisfiable formulas are identified in \[105\]. A formula is minimally unsatisfiable if it is unsatisfiable and removing any clause results in a satisfiable formula \[53\]. Minimally unsatisfiable formulas are SAT-checked polynomially.

• **Various results \[152\].** The next results are provided in \[152\]: (i) none of the classes \(S_0\), Q-Horn and extended Horn subsumes any other; (ii) a new class is defined that subsumes \(S_0\); and (iii) originated in that new class, a hierarchy of classes is proposed that generalizes \(Ω\) (see hierarchy \(Ω\) above) and the classes \(S_0\) and Q-Horn. Recognizing and SAT-checking the newly described, hierarchical classes are both polynomial.

• **CC·balanced.** The CC-balanced formulas have been studied in \[41\] and the motivation for this class is when Linear Programming relaxations of SAT-checking have non-integer solutions. Recognizing and SAT-checking CC-balanced formulas are linear.
• UP-Horn. The UP-Horn formulas \[66\] include the Horn and the non-Horn formulas resulting in Horn formulas after applying the Unit Propagation. Thus, recognizing and SAT-checking UP-Horn formulas are polynomial.

• SLUR hierarchies. SLUR(i) \[153\] is a hierarchy on top of the class SLUR. Level \(i\)-th consists in making the SLUR algorithm to consider simultaneously \(i\) literals and their corresponding \(2^i\) assignments. SLUR*(i) \[15\] is a hierarchy more general than SLUR(i) and relies on allowing DPLL to backtrack at most \(i\) levels. The class SLUR\(_K\) is introduced in \[77\] and proved that it strongly subsumes previous hierarchies SLUR(i) and SLUR*(i).

• Various results \[3\]. Reference \[3\] presents some new polynomial fragments based on Unit Propagation: (i) the polynomial class in \[147\], formed by formulas whose variables may appear at most twice, is extended; (ii) some series of benchmarks from the DIMACS repository and from SAT competitions are shown to belong to UP-Horn; and (iii) some set relations between the classes Quad and UP-Horn are established.

Beyond Propositional Logic. Horn formulas are used in a large variety of artificial intelligence logics. Hence, we make reference just to some logics somewhat close to propositional logic and whose NC lifting is discussed in Future Work.

• Regular Many-Valued and Possibilistic Logics. The regular many-valued Horn formulas \[79\] and the possibilistic Horn formulas \[106, 58, 59\] have been lifted to NC in \[88, 89\].

• Linear Arithmetic Logic. In linear arithmetic logic \[8, 9\], an atom is a disequality, e.g. \(4x_1 + x_3 \neq 3\), or a weak inequality, e.g. \(3x_1 + x_4 \leq 10\). A Horn arithmetic clause is a disjunction of any number of disequalities and at most one weak inequality. An example of a Horn arithmetic clause is given below. Satisfiability of Horn arithmetic formulas is polynomial \[103\]. Future work section considers the extension of this class to NC form.

\[
(3x_1 + x_5 - 4x_3 \leq 7) \lor (2x_1 + 3x_2 - 4x_3 \neq 4) \lor (x_2 + x_3 + x_5 \neq 7)
\]

• Horn Constraint Systems. Different classes of numerical Horn formulas are studied in \[143\]. The authors consider the next setting: if \(x\) and \(a\) are vectors and \(A\) is a matrix, a system of constraints \(A \cdot x \geq a\) is called Horn constraint system if: (a) the entries in \(A\) are in \(\{0, 1, -1\}\); (b) each row of \(A\) contains at most one positive entry; (c) \(x\) is a real vector; and (d) \(a\) is a integral vector. A comprehensive account of a variety of theses Horn constraint systems and their corresponding complexity is given in \[158\]. Whether or not these classes can be extended to NC form is an open issue for us.

• Lukasiewicz Logic. The authors in \[27\] defined Horn formulas in the infinite-valued Lukasiewicz logic and proved that their satisfiability problem is \(\text{NP}\)-complete. However, it was proven to be polynomial for the 3-valued case \[25\]. Further tractable and intractable subclasses are given in \[26\]. Future work section regards this class to be extended to NC.

6.2.2 Automated Reasoning

The models of Horn formulas form a lattice\[3\] and this mathematical tool is also the underlying structure within a large number of domains: lattice theory, hypergraph theory, data bases, concept analysis, artificial intelligence, knowledge spaces, Semantic Web, etc. This

---

\[3\]A fact observed first by McKinsey \[115\] (see also \[87\]).
fact entails that many concepts and algorithms have been developed for Horn formulas in different realms and there exist bijective links between the elements belonging to different fields [23]. See [2, 156, 22] for comprehensive surveys. Thus, we give a short overview of some selected reasoning problems which are tighter connected with $H_{\text{NC}}$.

**Horn Functions.** Noting $B = \{0, 1\}$, a Boolean function $f_B$ is a mapping $B^n \rightarrow B$. $f_B$ is modeled by a formula $\varphi$ iff $f_B$ assumes 1 exactly on the 0-1 vectors corresponding to the models of $\varphi$ and 0 elsewhere. The next claims are well-known [81, 82, 43].

- $f_B$ is representable by numerous formulas.
- $f_B$ is representable by a clausal formula.
- $f_B$ represents a unique Boolean function.

If $f_B$ is a Horn function $f_H$ and is defined on $(x_1, x_2, \ldots, x_n)$, then $f_H$ fulfills [43]:

- The set of models of $f_H$ is closed under componentwise conjunction.
- $f_H$ is representable by a Horn formula.
- The true points of $f_H$ form a closure system on $\{x_1, x_2, \ldots, x_n\}$.

Given a $f_H$, obtaining its Horn formula is done in [21]. Horn functions are also representable by non-Horn formulas and to decide whether an arbitrary clausal formula represents a Horn function is co-NP-complete [43]. A prime implicant, fundamental in Boolean functions, of $f_B$ is a clause $C$: (i) whose models include those of $f_B$ and (ii) no sub-clause of $C$ satisfies (i). Any $f_B$ is representable by the conjunction of its prime implicants. Searching all prime implicants of any $f_B$ is NP-complete. The prime implicates of any $f_H$ are Horn clauses. A main application of Horn functions is finding the Horn representation of any $f_H$ with minimal size, which has several applications, e.g. compacting knowledge bases [83]. The steps to find the required prime implicants of any $f_B$ are: (1) obtaining the prime implicants of $f_B$ and (2) finding which prime implicants can be discarded. The minimization problem is NP-hard, for general and also for Horn functions [82]. Horn functions are a sub-class of other more general Boolean functions such as the renamable-Horn and the Q-Horn functions [43] representable by Horn-renamable and Q-Horn formulas, respectively (see Subsection 6.2.1).

Given that Horn functions [43] are representable by a Horn formula and Corollary 3.20 states that both $H_{\text{NC}}$ and the Horn class are semantically equivalent, $H_{\text{NC}}$ can serve for the analysis of Horn functions. In fact, $H_{\text{NC}}$ turns out to be an attractive option to the Horn class taking into account that, as proved in Section 5, $H_{\text{NC}}$ is linearly recognizable and its HNC formulas can be even exponentially smaller than their clausal Horn representations.

**Implicational (Closure) Systems.** They are used to express strong implications between attributes in databases (functional dependencies) and relational data [51], but are relevant in many domains. We refer to [2, 156, 22] for recent overviews. A formal context $\mathcal{K}$ is a 3-tuple $(G, M, I)$, where $G$ are objects, $M$ are attributes and $I$ is a binary table indicating which objects have which of the attributes [71]. A formal concept is a 2-tuple $(O_i, A_i)$ where $O_i$ is a set of objects and $A_i$ is a set of attributes and means that every object in $O_i$ has every attribute in $A_i$. If an object is not in $O_i$ then there is an attribute in $A_i$ which that object does not have; and for every attribute that is not in $A_i$ there is an object in $O_i$ that does not have that attribute. An implication $A \rightarrow B$ means each object having all attributes in $A$ also have all attributes in $B$. An implication system $\Sigma$ defined on a set $S$ is a binary relation on $\mathcal{P}(S)$ and consists of a set of implication rules $A \rightarrow B$, where $A, B \in \mathcal{P}(S)$. The family $\mathcal{F}$ that satisfies all implication rules in $\Sigma$ is:
An alternative to knowledge compilation has been developed

Horn Approximations.

such as maintaining the minimum model of an HNC, are worthwhi le issues to be tackled.

for the clausal scenario when clauses are inserted or deleted from a (un)satisfiable formula, because both it is polynomial and contains Horn-SAT. The men tioned problems studied

node in a

positive-literal node is reachable; (iii) a negative

∧

nodes in unit clauses are reachable; (ii) a negative-litera l node is reachable if its dual

conjunct nodes of a reachable non-negative

ϕ

be roughly defined on the DAG of any

ϕ

getting

clause of length

q

traversal of the HNF system. A polynomial algorithm based on URnc could be devised for

problem in directed hypergraphs.

Assuming that a Horn formula is satisfiable and a new

ϕ

is obtained via the operator

ϕ

A

is P-complete via P-completeness of the reachability

and Horn-SAT can be proved to be

implication systems. Horn formulas can be mapped into a dire cted hypergraph e.g., \[12\]. There are tight connections between Horn functions and implication systems \[23\]. Implication systems can be viewed from a logic- based perspective as follows. \(A \rightarrow B\) holds in \(\mathcal{K}\) if each object that belongs to attributes in \(A\) also belongs to attributes in \(B\). Any \(\Sigma\) is complete for \(\mathcal{K}\) if, for all \(A \rightarrow B\): \(\mathcal{K} \vdash A \rightarrow B\) iff \(\Sigma \vdash A \rightarrow B\). \(\Sigma\) and \(\Sigma'\) are equivalent if \(\phi_\Sigma(X) = \phi_{\Sigma'}(X)\). \(A \rightarrow B\) follows from \(\Sigma\), noted \(\Sigma \vdash A \rightarrow B\), if \(\Sigma \cup A \rightarrow B\) is equivalent to \(\Sigma\). Then, we have \[156\]: \(\Sigma \vdash A \rightarrow B\) iff \(B \subseteq \phi_\Sigma(A)\). The possibility of managing implications by inference systems is one of its outstanding features. Equivalent systems are obtained with inference rules whose aim is to turn the original system into an equivalent system fulfilling some desired properties. Thus, many methods to compact implication system have been

proposed and the first was formed by the Armstrong rules \[5\].

Implication systems with the HNF syntax can be contemplated. Classical implications can be extended with HNF implications as those of Definition 4.11 and Example 4.12. This would allow for compacter representations of implication systems, which is one of their desired properties, viz. optimality. However, compacter syntactical forms have more complex structures and so defining inferences with HNF implications makes difficult to ensure their completeness. Another interesting issue is that of directness. Directness with HNF implications amounts to, given any arbitrary set \(X\), find \(\phi_\Sigma(X)\) with just one traversal of the HNF system. A polynomial algorithm based on URnc could be devised for getting \(\phi_\Sigma(X)\) in an HNF direct implication-system, and likely even with linear complexity.

**Directed Hypergraphs.** A directed hypergraph is an pair \((E, \mathcal{H})\), \(E\) being a vertex set and \(\mathcal{H}\) a hyper-edge set. Hypergraphs have been applied in many domains including implication systems. Horn formulas can be mapped into a directed hypergraph e.g., \[12\] and Horn-SAT can be proved to be \(P\)-complete via \(P\)-completeness of the reachability problem in directed hypergraphs. Assuming that a Horn formula is satisfiable and a new clause is inserted, the satisfiability question is answered in \(O(1)\) time and inserting a new clause of length \(q\) in \(O(q)\) amortized time \[11\]. If \(m\) clauses of an unsatisfiable Horn formula are deleted, the time needed to maintain the satisfiability state and its minimum model is \(O(n \times m)\), \(n\) being the number of variables \[10\]. We think that a reachability notion can be roughly defined on the DAG of any \(\varphi \in \mathcal{H}_{\mathcal{NC}}\) as follows: (i) initially only positive-literal nodes in unit clauses are reachable; (ii) a negative-literal node is reachable if its dual positive-literal node is reachable; (iii) a negative \(\lor\)-node (resp. \(\land\)-node) is reachable if all (resp. one) of its negative disjunct nodes are reachable; and (iv) the non-negative disjunct node in a \(\lor\)-node is reachable if all its negative disjunct nodes are reachable; and (v) all conjunct nodes of a reachable non-negative \(\land\)-node are reachable. Thus \(\varphi\) is unsatisfiable iff the node \(F\) is reachable on the DAG of \(\varphi\). The satisfiability of \(\mathcal{H}_{\mathcal{NC}}\) also is \(P\)-complete because both it is polynomial and contains Horn-SAT. The mentioned problems studied for the clausal scenario when clauses are inserted or deleted from a (un)satisfiable formula, such as maintaining the minimum model of an HNC, are worthwhile issues to be tackled.

**Horn Approximations.** An alternative to knowledge compilation has been developed
in [137, 98, 138, 114] and consists in compiling a formula $\varphi$ into two Horn formulas $H_{lb}$ and $H_{ab}$ so that many queries performed under $\varphi$ can be performed under $H_{lb}$ and $H_{ab}$ polynomially. Thus, the goal is finding $H_{lb}$ and $H_{ab}$ whose models $\mathcal{M}(H_{lb})$ and $\mathcal{M}(H_{ab})$ verify $\mathcal{M}(H_{lb}) \subseteq \mathcal{M}(\varphi) \subseteq \mathcal{M}(H_{ab})$. Then one disposes of: (i) sound approximation: $\mathcal{M}(H_{lb}) \models \varphi$ implies $\varphi \models \varphi$, and (ii) complete approximation: $\mathcal{M}(H_{lb}) \not\models \varphi$ implies $\varphi \not\models \varphi$. To maximize these concepts, the tightest approximations must be sought, which are: (i) $H_{glb}$ s.t. $\mathcal{M}(H_{glb}) \subseteq \mathcal{M}(H_{lb}) \subseteq \mathcal{M}(\varphi)$ for all $H_{lb}$; and (ii) $H_{lab}$ s.t. $\mathcal{M}(\varphi) \subseteq \mathcal{M}(H_{lab}) \subseteq \mathcal{M}(H_{ab})$ for all $H_{ab}$. Finding Horn approximations is mildly harder than solving the original clause entailment, i.e. it is $\text{P}^{\text{NP}[O(\log n)]}$-hard [35]. Determining HNFs $\varphi_1$ and $\varphi_2$ verifying the requirements for $H_{glb}$ and $H_{lab}$ is clearly harder than determining the proper $H_{glb}$ and $H_{lab}$ as clausal formulas have simpler syntactical structure than NC formulas. However, some simple HNFs can be considered to start with, e.g. a conjunction of disjunctions formed by negative literals and one conjunction of positive literals, which amounts to compact Horn clauses sharing the same negative literals. Besides, as HNFs are much smaller than their Horn clausal counterparts and also clause entailment (unsatisfiability) with HNFs is polynomial, HNFs will allow accelerating on-line answering.

**Abduction.** Abduction is the problem of, given a Horn function $f_H$ and a assumption set $S$ of propositions, generating an explanation for a literal $\ell$, which is a subset $E \subseteq S$ verifying: (i) $f_H \land E$ is satisfiable, and (ii) $f_H \land E \models \ell$. The complexity of abduction depends on the representation of $f_H$: if $f_H$ is represented by its characteristic models, then abduction is polynomial [97], wheras it is NP-complete if $f_H$ is represented by OBDDs [86]. The complexity of abduction when Boolean functions are presented by INC formulas should be studied, but it is likely polynomial because the above conditions (i) and (ii) are checked polynomially, taking into account that $H_{bNC}$ is polynomial for satisfiability testing.

**Petri Nets.** There are tight relations between rule-based systems and Petri Nets [161, 39, 112]. Petri Nets are used to verify properties of rule-based systems such as inconsistency, redundancy and circularity rules. A transition $T$ having several input places $P_1, P_2, \ldots, P_{j-1}$ and several output places $P_j, P_{j+1}, \ldots, P_k$ is modeled by rule (1) below:

$$(1) \quad P_1 \land \ldots \land P_{j-1} \rightarrow P_j \land \ldots \land P_k$$

$$(2) \quad T_1 \lor \ldots \lor T_{j-1} \rightarrow T_j \land \ldots \land T_k$$

A place $P$ preceded by several transitions $T_1, T_2, \ldots, T_{j-1}$ and followed by several transitions $T_j, T_{j+1}, \ldots, T_k$ is modeled by rule (2) above. As rules (1)-(2) are not Horn, a normalization process [161] must be applied, which of course significantly increases the size of the obtained knowledge base. However, both rules are indeed HNF according to Definition [1, 11] and so, they can be polynomially tackled with $\text{UR}_{\text{NC}}$-based inferences.

## 7 Future Work

Our presented approach can be extended smoothly to heterogeneous logics, accommodated to different reasoning settings, and used as a base towards developing the NC paradigm. Future research, that is likely to receive our attention, is divided into two axes.

### 7.1 $H_{bNC}$ and $\text{UR}_{\text{NC}}$ for Several Logics

We will define $H_{bNC}$ and $\text{UR}_{\text{NC}}$ for several logics and validate each $\text{UR}_{\text{NC}}$ by proving its completeness for its corresponding $H_{bNC}$. Then, based on $\text{UR}_{\text{NC}}$, we will design Horn-NC-SAT algorithms, which are useful e.g. in DPLL reasoners.
**Propositional logic.** Since $H_{NC}$ should play in NC a rôle similar to the rôle played by Horn in clausal efficiency, worthy research efforts remain to devise a highly-efficient Horn-NC-SAT algorithm. As Horn-SAT is linear, proving that Horn-NC-SAT is also linear would suggest that reaching clausal efficiency in some NC settings is conceivable. We have shown that Horn-NC-SAT is polynomial and we believe that its linearity can be proved.

**Logic Programming.** This topic has been discussed in Subsection 4.2.

**Quantified Boolean Formulas (QBFs).** Building solvers for QBFs in NC is a contemporary research concern, e.g. [113, 61, 14]. We will extend the presented $H_{NC}$ to QBFs and define the QBF-Horn-NC class. Then, we will determine $UR_{NC}$ for QBF-NCs, basic for DPLL. The complexity of QBF-Horn-NC is an open question knowing that QBF-Horn is quadratic [32].

**Regular Many-Valued Logic and Possibilistic Logic.** Our approach has been extended to these logics in [88, 89].

**Linear-Arithmetic Logic.** In Related Work, Arithmetic-Horn formulas have been presented and mentioned that its associated SAT problem is polynomial [103]. Thus, we will define the Arithmetic-Horn-NC class. Then, we will try to prove that Arithmetic-Horn-NC-SAT is polynomial as its clausal counterpart is [103]. For that purpose, we will require first to stipulate $UR_{NC}$ for linear-arithmetic logic.

**Lukasiewicz Logic.** As evoked in Related Work, the satisfiability problem of the class $L_{\infty}$-Horn is $NP$-complete [27] and polynomial [25] for the 3-valued case, $L_3$-Horn. Our goal will be to lift $L_3$-Horn to NC, determining thus, the $L_3$-Horn-NC class, $L_3 \oplus H_{NC}$. Then we will analyze whether tractability is preserved in NC, that is, whether $L_3 \oplus H_{NC}$ is polynomial. For that, a former step is to define $UR_{NC}$ for $L_3$. Posteriorly, we will deal with the infinite-valued case.

### 7.2 Developing the NC Paradigm

**Tractability.** This research line will be resumed by uplifting some of the polynomial Horn-clausal extensions reviewed in Related Work. Concretely, we will first attempt to determine the renamable-Horn-NC and SLUR-NC classes. Indeed, uplifting polynomial clausal classes may be an open research direction towards amplifying NC tractability.

**Satisfiability.** It was empirically verified (Related Work) that DPLL outperforms the other published methods. $H_{NC}$ and $UR_{NC}$ are the base of NC DPLL solvers. Hence, we plan: (1) devising a fast Unit-Propagation based on $UR_{NC}$ and easing its suitable integration into the NC-DPLL structure; and (2) lifting polynomial classes surveyed in Related Work to NC form.

**MaxSAT.** As evoked in Related Work, NC MaxSAT has not been properly defined. At least the next three optimization factors can be considered, $\varphi$ being the input: (i) the number of satisfiable clauses in $cl(\varphi)$; (ii) the number of satisfiable sub-formulas in $\varphi$; and (iii) a weighted factor resulting from combining both previous factors. We will analyze such variants, compare them and solve NC-MaxSAT.

**Signed Many-Valued Logic.** Polarity of literals is the underlying characterization of the Horn pattern; yet, it does not apply in signed many-valued logic [19, 79]. However, a realistic possibility to make tractability appear in signed logic consists in extending polynomial Horn super-classes to signed logic. We think that the simplicity, wide range
and relevance of the SLUR formulas (Related Work) make them good candidates to begin with. Then, we will attempt to lift the found clausal subclasses to NC.

**Resolution.** N. Murray in the 1980s [119] proposed NC Resolution in a combined functional and classical manner. Thus, we plan to make the next contributions: (1) determining a new definition of NC Resolution, making it as deterministic as clausal Resolution is, and proving also its refutation completeness. We will pursue this objective following the principle employed in this article which led us to define URnc; and (2) NC Hyper-Resolution has not been defined yet, and thus we will attempt to bridge such gap.

**Q-Resolution.** Resolution QBFs is called Q-Resolution [100]. Thus, we plan lifting Q-Resolution to NC and so defining NC-Q-Resolution. Yet, in view of the aforementioned difficulties to prove the completeness of (non-quantified) NC-Resolution, it will be mandatory to start with simple conjunctions of hybrid CNF and DNF instances of QBF-NCs.

**Theorem Proving.** Future research pointed out so far, deals with propositional logic and with some of its extensions. As long-term research project, we plan to enlarge the scope of our approach to first-order logic (FOL). Nonetheless, attending to the complexity of reasoning with NC and FOL, we will begin with simple fragments such as those limited to monadic and binary predicates and having at most one function symbol.

## 8 Conclusion

Our first main contribution has been the definition of the hybrid Horn-NC class, $\mathcal{H}_{\text{NC}}$, obtained by adequately amalgamating both Horn and NC classes, or equivalently, by lifting the Horn pattern to NC. We first defined $\mathcal{H}_{\text{NC}}$ as the class of NCs whose disjunctions have at most one non-negative disjunct, and then, provided a syntactically detailed, compact, inductive definition of $\mathcal{H}_{\text{NC}}$. Furthermore, we proved its next properties:

1. $\mathcal{H}_{\text{NC}}$ is polynomial for satisfiability testing.
2. $\mathcal{H}_{\text{NC}}$ syntactically subsumes $\mathcal{H}$.
3. $\mathcal{H}_{\text{NC}}$ and $\mathcal{H}$ are semantically equivalent.
4. $\mathcal{H}_{\text{NC}}$ contains all NCs whose clausal form is Horn.
5. $\mathcal{H}_{\text{NC}}$ is linearly recognizable.
6. $\mathcal{H}_{\text{NC}}$ is strictly more succinct than $\mathcal{H}$.
7. $\mathcal{H}_{\text{NC}}$ is exponentially richer than $\mathcal{H}$.

Our second main contribution has been establishing the calculus Non-Clausal Unit-Resolution, URnc, and proving that it checks the satisfiability of $\mathcal{H}_{\text{NC}}$ in polynomial time, which makes $\mathcal{H}_{\text{NC}}$ the first determined tractable class in NC reasoning.

The advantages of $\mathcal{H}_{\text{NC}}$ and URnc include its clear potential to leverage NC reasoning: (i) $\mathcal{H}_{\text{NC}}$ and URnc are crucial to build NC DPLL satisfiability solvers and NC DPLL theorem provers capable of emulating the efficiency of their clausal counterparts; and (ii) a general NC reasoner supplied with URnc can polynomially decide the HNC fragment.

Also, we have highlighted that declarative languages and in general rule-based systems may enrich their languages considering NC formulas in their antecedents and consequents and which can be done with a query-answering efficiency comparable to clausal efficiency. As a by-product of the properties of $\mathcal{H}_{\text{NC}}$, it could draw interest from other fields such as knowledge compilation, Horn functions analysis or implication systems.
We plan to follow several future research directions: proving that satisfiability checking $H_{NC}$ is linear; exploring the scope of $H_{NC}$ and $UR_{NC}$ in NC logic programming; defining $H_{NC}$ and $UR_{NC}$ beyond propositional logic; founding new polynomial NC classes; obtaining a clausal-like definition of Non-Clausal Resolution; and developing the NC paradigm through a number of outlined ways.

Acknowledgments J. Buenabad-Chávez, Lluís Godo.

References

[1] A. Abouzied, D. Angluin, C. H. Papadimitriou, J. M. Hellerstein, and A. Silberschatz. Learning and verifying quantified boolean queries by example. In Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013, pages 49–60, 2013.

[2] K. V. Adaricheva, G. F. Italiano, H. Kleine Büning, and G. Turán. Horn formulas, directed hypergraphs, lattices and closure systems: related formalisms and applications (dagstuhl seminar 14201). Dagstuhl Reports, 4(5):1–26, 2014.

[3] B. Al-Saedi, O. Fourdrinyo, E. Grégoire, B. Mazure, and L. Saïs. About some UP-based polynomial fragments of SAT. In Annals of Mathematics and Artificial Intelligence, volume 79, pages 25–44. Springer International Publising Switzerland, 2017.

[4] J. Álvez, M. Hermo, P. Lucio, and G. Rigau. Automatic white-box testing of first-order logic ontologies. J. Log. Comput., 29(5):723–751, 2019.

[5] W. W. Armstrong. Dependency structures of data base relationships. In Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5-10, 1974, pages 580–583, 1974.

[6] V. Arvind and S. Biswas. An $O(n^2)$ algorithm for the satisfiability problem of a subset of propositional sentences in CNF that include all Horn sentences. Information Processing Letters, 1987.

[7] B. Aspvall. Recognising disguised NR(1) instances of the satisfiability problem. Journal of Algorithms, (1):97–103, 1980.

[8] G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Integrating boolean and mathematical solving: Foundations, basic algorithms, and requirements. In J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, and V. Sorge, editors, Artificial Intelligence, Automated Reasoning, and Symbolic Computation, Joint International Conferences, AISC 2002 and Calculus 2002, Marseille, France, July 1-5, 2002, Proceedings, volume 2385 of Lecture Notes in Computer Science, pages 231–245. Springer, 2002.

[9] G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani. A SAT based approach for solving formulas over boolean and linear mathematical propositions. In Automated Deduction - CADE-18, 18th International Conference on
G. Ausiello, P. G. Franciosa, D. Frigioni, and R. Giacchino. Decremental maintenance of reachability in hypergraphs and minimum models of horn formulae. In H. W. Leong, H. Imai, and S. Jain, editors, Algorithms and Computation, 8th International Symposium, ISAAC ’97, Singapore, December 17-19, 1997, Proceedings, volume 1350 of Lecture Notes in Computer Science, pages 122–131. Springer, 1997.

G. Ausiello and G. F. Italiano. On-line algorithms for polynomially solvable satisfiability problems. J. Log. Program., 10(1):69–90, 1991.

G. Ausiello and L. Laura. Directed hypergraphs: Introduction and fundamental algorithms - A survey. Theor. Comput. Sci., 658:293–306, 2017.

L. Bachmair and H. Ganzinger. Non-clausal resolution and superposition with selection and redundancy criteria. In Logic Programming and Automated Reasoning, International Conference LPAR’92, St. Petersburg, Russia, July 15-20, 1992, Proceedings, pages 273–284, 1992.

V. Balabanov and J. R. Jiang. Unified QBF certification and its applications. Formal Methods Syst. Des., 41(1):45–65, 2012.

T. Balyo, S. Gurský, P. Kučera, and V. Vlček. On hierarchies over the SLUR class. In International Symposium on Artificial Intelligence and Mathematics (ISAIM 2012), 2012.

C. W. Barrett and J. Donham. Combining SAT methods with non-clausal decision heuristics. Electron. Notes Theor. Comput. Sci., 125(3):3–12, 2005.

P. Baumgartner. Model evolution-based theorem proving. IEEE Intell. Syst., 29(1):4–10, 2014.

P. Baumgartner and C. Tinelli. The model evolution calculus as a first-order DPLL method. Artif. Intell., 172(4-5):591–632, 2008.

B. Beckert, R. Hähnle, and F. Manyà. The SAT problem of signed CNF formulas. In D. Basin, M. D’Agostino, D. Gabbay, S. Matthews, and L. Viganò, editors, Labelled Deduction, pages 59–80. Applied Logic Series, vol 17. Springer, Dordrecht, 2000.

M. Ben-Ari. Mathematical Logic for Computer Science, 3rd Edition. Springer, 2012.

K. Ben-Khalifa and S. Motameny. Horn representation of a concept lattice. Int. J. Gen. Syst., 38(4):469–483, 2009.

K. Bertet, C. Demko, J. Viaud, and C. Guérin. Lattices, closures systems and implication bases: A survey of structural aspects and algorithms. Theor. Comput. Sci., 743:93–109, 2018.

K. Bertet and B. Monjardet. The multiple facets of the canonical direct unit implicational basis. Theor. Comput. Sci., 411(22-24):2155–2166, 2010.

W. Bibel. On Matrices with Connections. J. ACM., 28(4):633–645, 1981.
[25] M. Bofill, F. Manyà, A. Vidal, and M. Villaret. The complexity of three-valued Lukasiewicz rules. In *MDAI*, pages 221–229, 2015.

[26] M. Bofill, F. Manyà, A. Vidal, and M. Villaret. New complexity results for Lukasiewicz logic. *Soft Computing*, 23 (7):2187–2097, 2019.

[27] S. Borgwardt, M. Cerami, and R. Peñaloza. Many-valued horn logic is hard. In *Proceedings of the First Workshop on Logics for Reasoning about Preferences, Uncertainty, and Vagueness, PRUV 2014*, co-located with 7th International Joint Conference on Automated Reasoning (IJCAR 2014), Vienna, Austria, July 23-24, 2014, pages 52–58, 2014.

[28] E. Boros, Y. Crama, and P. L. Hammer. Polynomial-time inference of all valid implications for Horn and related formulae. *Annals of Mathematics and Artificial Intelligence*, 1:21–32, 1990.

[29] E. Boros, Y. Crama, P. L. Hammer, and M. Saks. A complexity index for satisfiability problems. *SIAM Journal on Computing*, 23:45–49, 1994.

[30] A. Bria, W. Faber, and N. Leone. Normal form nested programs. *Fundam. Informaticae*, 96(3):271–295, 2009.

[31] F. Bry, N. Eisinger, T. Eiter, T. Furche, G. Gottlob, C. Ley, B. Linse, R. Pichler, and F. Wei. Foundations of rule-based query answering. In *Reasoning Web*, Third International Summer School 2007, Dresden, Germany, September 3-7, 2007, Tutorial Lectures, pages 1–153, 2007.

[32] H. Büning and U. Bubeck. QBF Theory. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability: Chapter 23*, pages 339–395. IOS Press, 2009.

[33] H. Büning and O. Kullman. Minimum Unsatisfiability and Autarkies. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability: Chapter 11*, pages 339–395. IOS Press, 2009.

[34] P. Cabalar, J. Fundinno, T. Schaub, and S. Schellhorn. Gelfond-zhang aggregates as propositional formulas. *Artif. Intell.*, 274:26–43, 2019.

[35] M. Cadoli and F. Scarcello. Semantical and computational aspects of horn approximations. *Artif. Intell.*, 119(1-2):1–17, 2000.

[36] R. Chandrasekaran. Integer programming problems for which a simple rounding type of algorithm works. In *Progress in Combinatorial Optimization. Academic Press Canada, Toronto, Ontario, Canada*, pages 101–106, 1984.

[37] V. Chandru, C. Coullard, P. Hammer, M. Montanuz, and X. Sun. On renamable Horn and generalised Horn functions. *Annals of Mathematics and Artificial Intelligence*, (1):33–47, 1990.

[38] V. Chandru and J. Hooker. Extended Horn sets in propositional logic. *Journal of ACM*, (38):205–221, 1991.
[39] E. Chavarria-Baez and X. Li. Structural error verification in active rule-based systems using petri nets. In MICAI-06, 2006.

[40] K. Claessen, N. Een, M. Sheeran, N. Sörenson, A. Voronov, and K. Akesson. SAT-Solving in Practice, with a Tutorial Example from Supervisory Control. Discrete Event Dyn. Syst., 19:495–524, 2009.

[41] M. Conforti, G. Cornuéjols, and K. Vusković. Balanced matrices. Discrete Mathematics, 306(19-20):2411–2437, 2006.

[42] S. Cook. The complexity of theorem-proving procedures. In Third ACM Symposium on theory of Computing, pages 151–158, 1971.

[43] Y. Crama and P. L. Hammer. Boolean Functions - Theory, Algorithms, and Applications, volume 142 of Encyclopedia of mathematics and its applications. Cambridge University Press, 2011.

[44] M. Dalal. An almost quadratic class of satisfiability problems. In W. Wahlster, editor, 12th European Conference on Artificial Intelligence, pages 355–359. John Wiley and Sons, Ltd., 1996.

[45] M. Dalal and D. Etherington. A hierarchy of tractable satisfiability problems. Information Processing Letters, (44):173–180, 1992.

[46] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. ACM Comput. Surv., 33(3):374–425, 2001.

[47] A. Darwiche. Model-based diagnosis using structured system descriptions. J. Artif. Intell. Res., 8:165–222, 1998.

[48] A. Darwiche. Compiling knowledge into decomposable negation normal form. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages, pages 284–289, 1999.

[49] A. Darwiche. Decomposable negation normal form. J. ACM, 48(4):608–647, 2001.

[50] A. Darwiche and P. Marquis. A knowledge compilation map. J. Artif. Intell. Res., 17:229–264, 2002.

[51] R. Dechter and J. Pearl. Structura identifiction in relational data. Artificial Intelligence, 58, 1992.

[52] A. del Val. Tractable databases: how to make propositional unit resolution complete through compilation. In Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR’94), pages 551–561, 1994.

[53] H. E. Dixon, M. L. Ginsberg, and A. J. Parkes. Generalizing boolean satisfiability I: background and survey of existing work. J. Artif. Intell. Res., 21:193–243, 2004.

[54] P. Doherty, J. Kachniarz, and A. Szalas. Meta-queries on deductive databases. Fundam. Inform., 40(1):7–30, 1999.
[55] W. Dowling and J. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, (3):267–284, 1984.

[56] R. Drechsler, T. Junttila, and I. Niemelä. Non-clausal SAT and ATPG. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability: Chapter 21*, pages 655–693. IOS Press, 2009.

[57] V. D’Silva, D. Kroening, and G. Weissenbacher. A survey of automated techniques for formal software verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7), 2008.

[58] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 419–513. New York: Oxford University Press, 1994.

[59] D. Dubois and H. Prade. Possibilistic Logic - An Overview. In J. W. D. M. Gabbay, J. H. Siekmann, editor, *Handbook of the History of Logic. Vol 9, Computational Logic*, pages 283–342. North-Holland, 2014.

[60] O. Dubois, P. André, Y. Boufkhad, and Y. Carlie. Chap. SAT vs. UNSAT, volume 26 of *Second DIMACS implementation challenge: cliques, coloring and usatisifiability*. *DIMACS Series in Discrete Mathematics and Theoretical Computer Sciences*, pages 415–436. American Mathematical Society, 1996.

[61] U. Egly, M. Seidl, and S. Woltran. A solver for qbf’s in negation normal form. *Constraints An Int. J.*, 14(1):38–79, 2009.

[62] T. Eiter, P. Kilpeläinen, and H. Mannila. Recognizing renamable generalized propositional formulas is NP-complete. *Discrete Applied Mathematics*, 59:23–31, 1995.

[63] M. Färber and C. Kaliszyk. Certification of nonclausal connection tableaux proofs. In *Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings*, pages 21–38, 2019.

[64] H. Fargier and P. Marquis. Extending the knowledge compilation map: Krom, horn, affine and beyond. In D. Fox and C. P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 442–447. AAAI Press, 2008.

[65] A. Farzan and Z. Kincaid. Strategy synthesis for linear arithmetic games. *Proc. ACM Program. Lang.*, 2(POPL):61:1–61:30, 2018.

[66] O. Fourdrinoy, E. Grégoire, B. Mazure, and L. Saïs. Reducing hard SAT instances to polynomial ones. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI 2007)*, pages 18–23, 2007.

[67] J. Franco and A. V. Gelder. A perspective on certain polynomial time solvable classes of satisfiability. *Discrete Applied Mathematics*, 125(3):177–214, 2003.

[68] J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, and F. Orejas. Invariant-free clausal temporal resolution. *J. Autom. Reasoning*, 50(1):1–49, 2013.
[69] G. Gallo and M. Scutellà. Polynomially solvable satisfiability problems. *Inform. Procc. Letters*, 29:221–227, 1988.

[70] G. Gallo and G. Urbani. Algorithms for testing the satisfiability of propositional formulae. *Journal of Logic Programming*, (7):45–61, 1989.

[71] B. Ganter and R. Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999.

[72] M. Ghallab and G. Escalada-Imaz. A linear control algorithm for a class of rule-based systems. *Journal of Logic Programming*, (11):117–132, 1991.

[73] E. Giunchiglia and R. Sebastiani. Applying the davis-putnam procedure to non-clausal formulas. In *AI*IA 99:Advances in Artificial Intelligence, 6th Congress of the Italian Association for Artificial Intelligence, Bologna, Italy, September 14-17, 1999, *Proceedings*, pages 84–94, 1999.

[74] C. D. Giusto, M. Gabbielli, and M. C. Meo. On the expressive power of multiple heads in CHR. *ACM Trans. Comput. Log.*, 13(1):6:1–6:19, 2012.

[75] G. Gogic, H. A. Kautz, C. H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20–25 1995*, 2 Volumes, pages 862–869. Morgan Kaufmann, 1995.

[76] G. Gutiérrez, I. P. de Guzmán, J. Martínez, M. Ojeda-Aciego, and A. Valverde. Satisfiability testing for boolean formulas using delta-trees. *Studia Logica*, 72(1):85–112, 2002.

[77] M. Gwynne and O. Kullmann. Generalising Unit-Refutation Completeness and SLUR via Nested Input Resolution. *Journal of Automatic Reasoning*, (52):31–65, 2014.

[78] M. Hague, A. W. Lin, P. Rümmer, and Z. Wu. Monadic decomposition in integer linear arithmetic. In N. Peltier and V. Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*, volume 12166 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2020.

[79] R. Hähnle. Complexity of Many-Valued Logics. In *Proceedings ISMVL 2001: 137-148*, 2001.

[80] R. Hähnle, N. V. Murray, and E. Rosenthal. Linearity and regularity with negation normal form. *Theor. Comput. Sci.*, 328(3):325–354, 2004.

[81] P. L. Hammer and A. Kogan. Horn functions and their dnfs. *Inf. Process. Lett.*, 44(1):23–29, 1992.

[82] P. L. Hammer and A. Kogan. Optimal compression of propositional horn knowledge bases: Complexity and approximation. *Artif. Intell.*, 64(1):131–145, 1993.

[83] P. L. Hammer and A. Kogan. Essential and redundant rules in horn knowledge bases. *Decis. Support Syst.*, 16(2):119–130, 1996.
[84] L. Henschen and L. Wos. Unit refutations and Horn sets. *Journal of the Association for Computing Machinery*, 21(4):590–605, 1974.

[85] B. Holen, E. B. Johnsen, and A. Waaler. Proof search for the first-order connection calculus in maude. In *Proceedings of the Seventh International Workshop on Rewriting Logic and its Applications, WRLA 2008, Budapest, Hungary, March 29-30, 2008*, pages 173–188, 2008.

[86] T. Horiyama and T. Ibaraki. Reasoning with ordered binary decision diagrams. *Discret. Appl. Math.*, 142(1-3):151–163, 2004.

[87] A. Horn. On sentences which are of direct unions of algebras. *J. Symb. Logic*, 16(1):14–21, 1951.

[88] G. E. Imaz. A First Polynomial Non-Clausal Class in Many-Valued Logic. *CoRR*, https://arxiv.org/abs/2110.12901:1–37, 2021.

[89] G. E. Imaz. The Possibilistic Non-Clausal Horn Knowledge Bases. *CoRR*, http://arxiv.org/abs/2111.07648:1–43 2021.

[90] A. Itai and J. Makowsky. On the complexity of Herbrand’s theorem. Technical Report 243, Dept. of Computer Science, Israel Inst. of Technology, 1982.

[91] A. Itai and J. Makowsky. Unification as a complexity measure for logic programming. *Journal of Logic Programming*, (4):105 – 177, 1987.

[92] E. K. Jackson and W. Schulte. Model generation for horn logic with stratified negation. In K. Suzuki, T. Higashino, K. Yasumoto, and K. El-Fakih, editors, *Formal Techniques for Networked and Distributed Systems - FORTE 2008, 28th IFIP WG 6.1 International Conference, Tokyo, Japan, June 10-13, 2008, Proceedings*, volume 5048 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.

[93] H. Jain, C. Bartzis, and E. Clarke. Satisfiability Checking of Non-clausal Formulas Using General Matings. In *SAT-2006*, pages 75–89, 2006.

[94] H. Jain and E. M. Clarke. Efficient SAT solving for non-clausal formulas using DPLL, graphs, and watched cuts. In *Proceedings of the 46th Design Automation Conference, DAC 2009, San Francisco, CA, USA, July 26-31, 2009*, pages 563–568, 2009.

[95] S. Jha, R. Limaye, and S. A. Seshia. Beaver: Engineering an efficient SMT solver for bit-vector arithmetic. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, pages 668–674, 2009.

[96] H. Kautz, B. Selman, and D. McAllester. Exploiting variable dependency in local search. In *Abstracts of the Poster Session of IJCAI-97*, 1997.

[97] H. A. Kautz, M. J. Kearns, and B. Selman. Reasoning with characteristic models. In R. Fikes and W. G. Lehnert, editors, *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993*, pages 34–39. AAAI Press / The MIT Press, 1993.
H. A. Kautz and B. Selman. Forming concepts for fast inference. In Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12-16, 1992, pages 786–793, 1992.

S. Klarman, U. Endriss, and S. Schlobach. Abox abduction in the description logic ALC. J. Autom. Reasoning, 46(1):43–80, 2011.

H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for quantified boolean formulas. Inf. Comput., 117(1):12–18, 1995.

A. Komuravelli, N. Björner, A. Gurfinkel, and K. L. McMillan. Compositional verification of procedural programs using horn clauses over integers and arrays. In R. Kaivola and T. Wahl, editors, Formal Methods in Computer-Aided Design, FM-CAD 2015, Austin, Texas, USA, September 27-30, 2015, pages 89–96. IEEE, 2015.

E. Kotelnikov, L. Kovács, M. Suda, and A. Voronkov. A clausal normal form translation for FOOL. In C. Benzmüller, G. Sutcliffe, and R. Rojas, editors, GCAI 2016. 2nd Global Conference on Artificial Intelligence, September 19 - October 2, 2016, Berlin, Germany, volume 41 of EPiC Series in Computing, pages 53–71. EasyChair, 2016.

M. Koubarakis and S. Skiadopoulos. Querying temporal and spatial constraint networks in PTIME. Artif. Intell., 123(1-2):223–263, 2000.

O. Kullmann. Investigations on autark assignments. Discrete Applied Mathematics, 107:99–137, 2000.

O. Kullmann. Lean clause-sets: generalizations of minimally unsatisfiable clause-sets. Discrete Applied Mathematics, 130(2):209–249, 2003.

J. Lang. Possibilistic logic: complexity and algorithms. In e. Gabbay D., Smets Ph, editor, Handbook of Defeasible Reasoning and Uncertainty Management System, pages 179–220. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001.

H. Lewis. Renaming a set of clauses as a Horn set. Journal of the ACM, (25):134–135, 1978.

C. M. Li and F. Manyà. MaxSAT, Hard and Soft Constraints. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, Handbook of Satisfiability: Chapter 9, pages 613–631. IOS Press, 2009.

C. M. Li, F. Manyà, and J. Planes. New inference rules for Max-SAT. Journal of Artificial Intelligence Research (JAIR), (1):134–135, 2007.

C. M. Li, F. Manyà, and J. R. Soler. A tableau calculus for non-clausal maximum satisfiability. In Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings, pages 58–73, 2019.

V. Lifschitz, L. R. Tang, and H. Turner. Nested expressions in logic programs. Ann. Math. Artif. Intell., 25(3-4):369–389, 1999.

38
[112] H. Liu, Q. Lin, L. Mao, and Z. Zhang. Dynamic adaptive fuzzy petri nets for knowledge representation and reasoning. IEEE Trans. Syst. Man Cybern. Syst., 43(6):1399–1410, 2013.

[113] F. Lonsing and A. Biere. Nenofex: Expanding NNF for QBF solving. In Theory and Applications of Satisfiability Testing - SAT 2008, 11th International Conference, SAT 2008, Guangzhou, China, May 12-15, 2008. Proceedings, pages 196–210, 2008.

[114] K. Makino and H. Ono. Deductive inference for the interiors and exteriors of horn theories. ACM Trans. Comput. Log., 13(3):23:1–23:16, 2012.

[115] J. McKinsey. The decision problem for some classes of sentences without quantifiers. Journal of Symbolic Logic, (8):61–76, 1943.

[116] A. Meier and V. Sorge. A new set of algebraic benchmark problems for SAT solvers. In In SAT, pages 459–466, 2005.

[117] M. Minoux. LTUR: A simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. Information Processing Letters, (29):1–12, 1988.

[118] R. Muhammad and P. J. Stuckey. A stochastic non-cnf SAT solver. In Q. Yang and G. I. Webb, editors, PRICAI 2006: Trends in Artificial Intelligence, 9th Pacific Rim International Conference on Artificial Intelligence, Guilin, China, August 7-11, 2006, Proceedings, volume 4099 of Lecture Notes in Computer Science, pages 120–129. Springer, 2006.

[119] N. Murray. Completely Non-Clausal Theorem Proving. Artificial Intelligence, 18(1):67–85, 1982.

[120] N. V. Murray and E. Rosenthal. On the relative merits of path dissolution and the method of analytic tableaux. Theor. Comput. Sci., 131(1):1–28, 1994.

[121] J. Navarro and A. Voronkov. Generation of Hard Non-Clausal Random Satisfiability Problems. In The Twentieth National Conference on Artificial Intelligence, pages 426–436, 2005.

[122] J. C. Nieves and H. Lindgren. Possibilistic nested logic programs. In Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012, September 4-8, 2012, Budapest, Hungary, pages 267–276, 2012.

[123] T. Nipkow. Linear quantifier elimination. J. Autom. Reasoning, 45(2):189–212, 2010.

[124] B. E. Oliver and J. Otten. Equality preprocessing in connection calculi. In Joint Proceedings of the 7th Workshop on Practical Aspects of Automated Reasoning (PAAR) and the 5th Satisfiability Checking and Symbolic Computation Workshop (SC-Square) Workshop, 2020 co-located with the 10th International Joint Conference on Automated Reasoning (IJCAR 2020), Paris, France, June-July, 2020 (Virtual), pages 76–92, 2020.

[125] J. Otten. On the advantage of a non-clausal davis-putnam procedure. Technical report, Technische Hochschule Darmstadt, Germany, 1997.
[126] J. Otten. A non-clausal connection calculus. In Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings, pages 226–241, 2011.

[127] D. Pearce. Equilibrium logic. Ann. Math. Artif. Intell., 47(1-2):3–41, 2006.

[128] D. Pearce, H. Tompits, and S. Woltran. Characterising equilibrium logic and nested logic programs: Reductions and complexity. Theory Pract. Log. Program., 9(5):565–616, 2009.

[129] D. Petrolani. A linear algorithm for unique Horn satisfiability. Information Processing Letters, 48(2):61–66, 1993.

[130] D. Petrolani. Hierarchies of polynomially solvable satisfiability problems. Annals of Mathematics and Artificial Intelligence, 17:339–357, 1996.

[131] A. Platzer. Differential dynamic logic for hybrid systems. J. Autom. Reason., 41(2):143–189, 2008.

[132] A. Ramesh, B. Beckert, R. Hähnle, and N. V. Murray. Fast subsumption checks using anti-links. J. Automated Reasoning, 18(1):47–83, 1997.

[133] E. Scala, P. Haslum, S. Thiébaux, and M. Ramírez. Subgoaling techniques for satisficing and optimal numeric planning. J. Artif. Intell. Res., 68:691–752, 2020.

[134] J. S. Schlipf, F. Annexstein, J. Franco, and R. Swaminathan. On finding solutions for extended Horn formulas. Information Processing Letters, (54):133–137, 1995.

[135] M. Scutellà. A note on Dowling and Gallier’s top-down algorithm for propositional Horn satisfiability. Journal of Logic Programming, (8):265–273, 1990.

[136] R. Sebastiani. Applying GSAT to non-clausal formulas (research note). J. Artif. Intell. Res., 1:309–314, 1994.

[137] B. Selman and H. A. Kautz. Knowledge compilation using horn approximations. In Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, July 14-19, 1991, Volume 2, pages 904–909, 1991.

[138] B. Selman and H. A. Kautz. Knowledge compilation and theory approximation. J. ACM, 43(2):193–224, 1996.

[139] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1, pages 337–343, 1994.

[140] B. Selman, H. J. Levesque, and D. G. Mitchell. A new method for solving hard satisfiability problems. In Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12-16, 1992, pages 440–446, 1992.

[141] Z. Stachniak. Non-Clausal Reasoning with Definite Theories. Fundamenta Informaticae, 48:1–26, 2001.

40
[142] Z. Stachniak. Going Non-clausal. In Fifth International Symposium on the Theory and Applications of Satisfiability Testing, pages 316–322, 2002.

[143] K. Subramani and J. Worthington. Feasibility checking in horn constraint systems through a reduction based approach. *Theor. Comput. Sci.*, 576:1–17, 2015.

[144] R. P. Swaminathan and D. K. Wagner. The arborescence realization problem. *Discrete Applied Mathematics*, (59):267–283, 1995.

[145] N. Tamura, A. Taga, S. Kitagawa, and M. Banbara. Compiling finite linear CSP into SAT. *Constraints An Int. J.*, 14(2):254–272, 2009.

[146] C. Thillault, F. Bacchus, and T. Walsh. Solving Non-Clausal Formulas with DPLL search. In Tenth International Conference on Principles and Practice of Constraint Programming, pages 663–678, 2004.

[147] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89, 1984.

[148] G. Tseitin. On the complexity of proofs in propositional logics. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning 2: Classical Papers on Computational Logic*, pages 466–483. Springer, 1983.

[149] H. Unno, Y. Satake, T. Terauchi, and E. Koskinen. Program verification via predicate constraint satisfiability modulo theories. *CoRR*, abs/2007.03656, 2020.

[150] H. Unno and T. Terauchi. Inferring simple solutions to recursion-free horn clauses via sampling. In C. Baier and C. Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2015.

[151] H. van Maaren. A short note on some tractable classes of the satisfiability problem. *Information and Computation*, 158(2):125–130, 2000.

[152] O. Čepek and P. Kučera. Known and new classes of generalized Horn formulae with polynomial recognition and SAT testing. *Discrete Applied Mathematics*, 149(1-3):14–52, 2005.

[153] O. Čepek, P. Kučera, and V. Vlček. Properties of SLUR formulae. In Springer, editor, *SOFSEM 2012: Theory and Practices of Computer Science. LNCS.*, volume 7147, pages 177–189. Bieliková M., Friedrich G., Gottlob G. Katzenbeisser S., Turán G., 2012.

[154] M. Veksler and O. Strichman. Learning general constraints in CSP. *Artif. Intell.*, 238:135–153, 2016.

[155] D. S. Warren. Top-down and bottom-up evaluation procedurally integrated. *Theory Pract. Log. Program.*, 18(3-4):706–721, 2018.

[156] M. Wild. The joy of implications, aka pure horn formulas: Mainly a survey. *Theor. Comput. Sci.*, 658:264–292, 2017.
[157] P. F. Williams, A. Biere, E. M. Clarke, and A. Gupta. Combining decision diagrams and SAT procedures for efficient symbolic model checking. *Computer Aided Verification (CAV), LNCS*. Vol. 1855, pages 124–138, 2000.

[158] P. J. Wojciechowski. *Analyzing Satisfiability and Refutability in Selected Constraint Systems*. PhD thesis, West Virginia University, 2019.

[159] Y. Xu, J. Liu, X. He, X. Zhong, and S. Chen. Non-clausal multi-ary $\alpha$-generalized resolution calculus for a finite lattice-valued logic. *Int. J. Comput. Intell. Syst.*, 11(1):384–401, 2018.

[160] S. Yamasaki and S. Doshita. The satisfiability problem for a class consisting of Horn sentences and some non-Horn sentences in propositional logic. *Inform. Control*, 59:1–12, 1983.

[161] S. J. H. Yang, J. J. P. Tsai, and C. Chen. Fuzzy rule base systems verification using high-level petri nets. *IEEE Trans. Knowl. Data Eng.*, 15(2):457–473, 2003.