Canonical Logic Programs are Succinctly Incomparable with Propositional Formulas

Yuping Shen and Xishun Zhao
Institute of Logic and Cognition
Department of Philosophy
Sun Yat-sen University
510275 Guangzhou, P.R. China
{shyping, hsszxs}@mail.sysu.edu.cn

Abstract

Canonical (logic) programs (CP) refer to the class of normal programs (LP) augmented with connective not not, and are equally expressive as propositional formulas (PF). In this paper we address the question of whether CP and PF are succinctly incomparable. Our main result shows that the PARITY problem only has exponential CP representations, while it can be polynomially represented in PF. In other words, PARITY separates PF from CP. Simply speaking, this means that exponential size blowup is generally inevitable when translating a set of PF formulas into a (logically) equivalent CP program (without introducing new variables). Furthermore, since it has been shown by Lifschitz and Razborov that there is also a problem which separates CP from PF (assuming P \nsubseteq NC1/poly), it follows that the two formalisms are indeed succinctly incomparable.

1 Introduction

The relationship between (logic) programs under answer set semantics (ASP) (Lifschitz 2008; Brewka, Eiter, and Truszczynski 2011) and propositional satisfiability (SAT) (Biere et al. 2009) gains a lot of attention in the literature. In 2006, Lifschitz and Razborov proved that an exponential size blowup is generally inevitable when translating a normal program (LP) to a (logically) equivalent set of propositional formulas (PF) (without introducing additional variables). More precisely, they showed that (a variant of the) P-complete problem PathSystem (PATH) has polynomial size LP representations, however, it cannot be polynomially represented in PF (assuming P \nsubseteq NC1/poly) (Lifschitz and Razborov 2006), i.e., PATH separates LP from PF.

As noted in (Lifschitz and Razborov 2006), PF can be considered as a special case of the class of (nondisjunctive) nested programs (NLP, without classical negation \neg) (Lifschitz 1999). Therefore, NLP is stronger than PF in terms of the succinctness criterion (or the “comparative linguistics” approach) (Gogic et al. 1995):

That is, we consider formalism A to be stronger than formalism B if and only if any knowledge base (KB) in B has an equivalent KB in A that is only polynomially longer, while there is a KB in A that can be translated to B only with an exponential blowup.

So the following footnote in (Lifschitz 2008) seems convincing:

...ASP appears to be stronger than SAT in the sense of the “comparative linguistics” approach to knowledge representation...

However, ASP involves many kinds of programs, the above statement probably needs further clarification. Particularly, the class of so-called (nondisjunctive) canonical programs (CP) (Lifschitz 1999; Lee 2005; Lee, Lifschitz, and Yang 2013), is a minimal form of ASP that is equally expressive as PF, but looks more likely not succinctly stronger. So a question naturally arises: Does there exist a problem that separates PF from CP? If there is such a problem, then CP and PF are succinctly incomparable (assuming P \nsubseteq NC1/poly).

In this paper we address the question and give a positive answer. Our main result shows that the PARITY problem separates PF from CP. Simply speaking, this means exponential size blowup is generally inevitable when translating a set of PF formulas into an equivalent CP program. The PARITY problem asks whether a binary string contains an odd number of 1’s, and it is well-known that (i) PARITY \in NC1/poly, i.e., it has polynomial PF representations; (ii) PARITY \notin AC0, i.e., it cannot be represented by polynomial size (boolean) circuits with constant depth and unbounded fan-in (Arora and Barak 2009; Jukna 2012).

To show PARITY separates PF from CP, we provide a procedure that simplifies every PARITY program \Pi into a shorter program \Pi’ s.t. \Pi’ is equivalent to its completion Comp(\Pi’) (Erdem and Lifschitz 2003). Such completions are essentially constant depth, unbounded fan-in circuits. According to PARITY \notin AC0, these circuits must be of exponential size, consequently, there are no polynomial size PARITY programs in CP. In the rest of the paper, we shall introduce the basic concepts, key steps of our proof and discuss the importance of succinctness research in the theory and practice of Knowledge Representation (KR).

---

\*Corresponding author.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
2 Background

Canonical Programs

The following notations are adopted from (Lifschitz 1999; Lee 2005). A rule element $e$ is defined as

$$e := \top \mid \bot \mid x \mid \text{not} \ x \mid \text{not not} \ x$$

in which $\top$, $\bot$ are 0-ary connectives, $x$ is a (boolean) variable (or an atom) and not is a unary connective.\footnote{By (Lifschitz 1999), not not $x$ can be replaced by not $x$.} A (nondisjunctive canonical) rule is an expression of the form

$$H \leftarrow B$$

where the head $H$ is either a variable or the connective $\bot$, and the body $B$ is a finite set of rule elements. A canonical program $\Pi$ is a finite set of rules. E.g., the following is a canonical program:

$$\begin{align*}
x_1 &\leftarrow \text{not not} \ x_1, \\
x_3 &\leftarrow \text{not} \ x_1, \text{not} \ x_2, \\
x_2 &\leftarrow \text{not not} \ x_2, \\
x_3 &\leftarrow x_1, x_2.
\end{align*}$$

(2)

A canonical program $\Pi$ is normal if it contains no connectives not not. A normal program $\Pi$ is basic if it contains no occurrences of connective not.

The satisfaction relation $|=\,$ between a set of variables $I$ and a rule element is defined as follows:

1. $I|=\top$ and $I\neq\bot$.
2. $I|=x$ iff $I|=\text{not} \ x$ iff $x \in I$.
3. $I|=\text{not} \ x$ iff $x \notin I$.

Say $I$ satisfies a set of rule elements $B$ if $I$ satisfies each rule element in $B$. We say $I$ is closed under a program $\Pi$ if $I$ is closed under every rule in $\Pi$, i.e., for each rule $H \leftarrow B \in \Pi$, $I|=H$ whenever $I|=B$. Let $\Pi$ be a basic program and let $Cn(\Pi)$ denotes the minimal set (in terms of set inclusion) closed under $\Pi$, we say $I$ is an answer set of $\Pi$ if $I = Cn(\Pi)$. Note that a basic program has exactly one answer set.

The reduct $\Pi^l$ of a program $\Pi$ w.r.t. $I$ is a set of rules obtained from $\Pi$ via: (i) Replacing each not not $x$ with $\top$ if $I|=x$, and with $\bot$ otherwise; (ii) Replacing each not $x$ with $\top$ if $I\neq\bot$, and with $\bot$ otherwise. Observe that $\Pi^l$ must be a basic program. We say $I$ is an answer set of $\Pi$ if $I = Cn(\Pi^l)$, i.e., $I$ is an answer set of $\Pi^l$.\footnote{By (Lifschitz 1999), not not $x$ can be replaced by not $x$.} E.g., the following single rule program:

$$x \leftarrow \text{not not} \ x$$

(3)

has two answer sets $\emptyset$ and $\{x\}$.

For a set of rule elements $B$, define $\text{var}(B) = \{e \in B : e$ is a variable$\}$. E.g., $\text{var}\{x_1, \text{not} \ x_2, \text{not not} \ x_3\} = \{x_1\}$. The signature $\text{sig}(\Pi)$ of a program $\Pi$ is the set of all involved variables in $\Pi$. By $\text{Ans}(\Pi)$ we denote the set of all answer sets of $\Pi$. E.g., if $\Pi$ is (2), then $\text{sig}(\Pi) = \{x_1, x_2, x_3\}$ and $\text{Ans}(\Pi) = \{\{x_1, x_2, x_3\}, \{x_1\}, \{x_2\}, \{x_3\}\}$. The size $|\text{Pi}|$ of $\Pi$ is the number of its rules. As a convention, $\Pi_n$ refers to a program with signature $\{x_1, \ldots, x_n\}$, i.e., $\text{sig}(\Pi_n) = \{x_1, \ldots, x_n\}$.

It is easy to see that by using the form (3) and appropriate constraints of the form $\bot \leftarrow B$, it is easy to give an arbitrary set of answer sets over $\text{sig}(\Pi_n)$, in other words, CP has exactly the same expressive power as PF.

Problem Representation and Succinctness

A (binary) string is a finite sequence of bits from $\{0, 1\}$. A string $w$ of length $n$ (i.e., $w \in \{0, 1\}^n$) defines a subset of variables $\{x_1, \ldots, x_n\}$. E.g., $1010$ stands for $\{x_1, x_3\}$. Therefore, a set of variables $I$ and a string $w$ can be regarded as the same. A problem (or language) $L$ is a set of strings.

**Definition 2.1** (Problem Representation). A problem $L$ can be represented in a class of programs (or formulas, etc) $C$ (i.e., $L \in C$), if there exists a sequence of programs $(\Pi_n)$ $(n = 1, 2, \ldots)$ in $C$ that computes $L$, i.e., for every string $w \in \{0, 1\}^n$,

$$w \in L \iff w \in \text{Ans}(\Pi_n).$$

Say $L$ has polynomial representations in $C$ (i.e., $L \in \text{Poly}-C$), if $L \in C$ and $|\Pi_n|$ is bounded by a polynomial $p(n)$.

Recall that the PARITY problem is the set of binary strings with an odd number of 1’s. BY PARITY, we denote PARITY strings of length $n$. Clearly, (2) represents PARITY$_3$ since its answer sets are $111, 001, 010$ and $000$.

**Definition 2.2** (Succinctness (Gogic et al. 1995; French et al. 2013)). Let $C, C'$ be two classes of programs such that for every problem $L$, $L \in C \Rightarrow L \in C'$. Say $C$ is at least as succinct as $C'$ (i.e., $C \preceq C'$), if for every problem $L$,

$$L \in \text{Poly}-C \Rightarrow L \in \text{Poly}-C'.$$

If $L \in \text{Poly}-C$ but $L \notin \text{Poly}-C'$ (i.e., $C \not\preceq C'$), then $L$ separates $C$ from $C'$. If $C' \preceq C$ and $C \not\preceq C'$, then $C$ is strictly more succinct than $C'$ (i.e., $C' \prec C$). Moreover, $C, C'$ are succinctly incomparable if there is a problem $L$ that separates $C$ from $C'$, and vice versa (i.e., $C \not\preceq C'$ and $C' \not\preceq C$).

PARITY Representations and Completion

Note that (2) suggests a “pattern” for PARITY$_n$: The first part of the program (e.g., the “not not” rules in (2)) generates all possible strings of length $n - 1$, the second part identifies the last bit to form an odd string of length $n$. Following the pattern, we can give a sequence of exponential size CP programs for PARITY, i.e., PARITY$_{\leq \text{CP}}$.

On the other hand, it is a textbook result that PARITY has polynomial size PF representations with classical connectives $\{\land, \lor, \neg\}$, i.e., PARITY $\in$ NC$^1$/poly (or Poly-PF). Remind that the size of a formula is the number of its connectives. E.g., $(x_1 \land \neg x_2) \lor (\neg x_1 \land x_2)$ represents PARITY$_2$, and the formula can be recursively applied for arbitrary PARITY$_n$. It is also well-known that PARITY $\not\in$ AC$^0$, i.e., PARITY cannot be represented by a sequence of polynomial size circuits $\{C_n\}$ in which these circuits $C_n$ have unbounded fan-in and constant depth.

Simply speaking, a sequence of polynomial size PF formulas $\{\phi_n\}$ represents an AC$^0$ language, if the number of variables occur in a conjunction or disjunction of $\phi_n$ is unrestricted, and the distance of the longest path from the root to a leaf in the tree structure of each $\phi_n$ is fixed. E.g., PARITY cannot be represented by a sequence of polynomial size CNFs $\{\psi_n\}$ since $\{\psi_n\}$ represents an AC$^0$ language. For more details about circuits, please see (Arora and Barak 2009; Jukna 2012).
The completion Comp(Π) (Erdem and Lifschitz 2003) of a CP program Π consists of a set (or a conjunction) of PF formulas (we slightly abuse the connective ≡): (i) \( x \equiv B_1 \lor B_2 \lor \cdots \lor B_m \), where \( x \leftarrow B_1, \ldots, x \leftarrow B_m \) are all rules in Π with head x, and each \( B_i \) is the conjunction of rule elements in \( B_i \) with connective not replaced by \( \neg \); (ii) \( x \equiv \bot \), if \( x \) is not a head of any rule in Π; (iii) \( \neg B \), if a rule \( \bot \leftarrow B \) is in Π.

**Proposition 2.1.** The completion Comp(Π) of an arbitrary canonical program Π is a constant depth, unbounded fan-in circuit whose size is polynomially bounded by |Π|.

It is well-known that an answer set of Π is also a model of its completion, but the inverse generally does not hold. E.g., the completion \( \{ x_1 \equiv \neg x_2 \lor (x_2 \land x_1), x_2 \equiv \neg x_1 \lor (x_1 \land x_2) \} \) of the PARITY 2 program:

\[
x_1 \leftarrow \neg x_2,
\]
\[
x_2 \leftarrow \neg x_1,
\]
\[
x_1 \leftarrow x_2, \text{ not } x_1,
\]
\[
x_2 \leftarrow x_1, \text{ not } not x_2,
\]

has an even string model 11, i.e., (4) is not completion-equivalent. In the next section, we will show how to simplify a PARITY program to be completion-equivalent.

### 3 Simplifying PARITY Programs

Let B be a finite set of rule elements built on signature \( V = \{ x_1, \ldots, x_n \} \). By \( S(B) \) we denote the set \( \{ I \subseteq V : I \models B \} \). E.g., let \( V = \{ x_1, x_2, x_3, x_4 \} \), and \( B = \{ x_2, \text{not } x_3, \text{not } not x_4 \} \), then \( S(B) = \{ \{ x_1, x_2, x_4 \}, \{ x_2, x_4 \} \} = \{ 1101, 0101 \} \). We say B is consistent if there is a set of variables I s.t. \( I \models B \). We say B covers a variable \( x \in V \) iff \( x \in B \) or not \( x \in B \) or not not \( x \in B \). If B covers every variable in V then B fully covers V. E.g., \( B = \{ x_1, \text{not } x_2, \text{not } not x_3 \} \) fully covers V = \{x_1, x_2, x_3\}. Clearly, B is consistent and fully covers V iff \( S(B) \) contains a unique string.

In the following we assume that all rules \( x \leftarrow B \) have consistent body B and \( x \notin \text{var}(B) \). It is easy to see that the assumption does not affect the generality.

**Lemma 3.1.** Let \( \Pi_n \) be a PARITY_n program. If there is a rule \( x \leftarrow B \) in \( \Pi_n \) s.t. not \( x \in B \) and \( S(B) \) contains a unique even string, then dropping \( x \leftarrow B \) from \( \Pi_n \) results in a PARITY_n program \( \Pi_n' \).

E.g., by Lemma 3.1, (4) can be simplified to:

\[
x_1 \leftarrow \neg x_2,
\]
\[
x_2 \leftarrow \neg x_1.
\]

**Lemma 3.2.** Let \( \Pi_n \) be a PARITY_n program. If there is a rule \( x \leftarrow B \) in \( \Pi_n \) s.t. not \( x \in B \) and \( S(B) \) contains a unique odd string, then dropping \( not not x \) from B results in a PARITY_n program \( \Pi_n' \).

### Standard Programs and the Main Theorem

A PARITY_n program \( \Pi_n \) is standard if for each rule \( x \leftarrow B \in \Pi_n \), not \( x \notin B \) whenever \( S(B \cup \{ x \}) \) contains a unique string. E.g., (2) and (5) are standard programs. By Lemma 3.1 and 3.2, it is not hard to see:

**Proposition 3.1.** Let \( \Pi_n \) be a PARITY_n program. Then there is a standard PARITY_n program \( \Pi_n' \) s.t. \( |\Pi_n'| \leq |\Pi_n| \).

Moreover, note that (2) and (5) are completion-equivalent PARITY programs. This is guaranteed by the following proposition:

**Proposition 3.2.** Let \( \Pi_n \) be a standard PARITY_n program. Then \( \Pi_n \) is equivalent to its completion Comp(\Pi_n).

The proof idea of Proposition 3.2 is that every standard PARITY_n program \( \Pi_n \) can be equivalently rewritten to \( \Pi_n' \) by replacing each \( x \in \text{var}(B) \) with not \( x \) for every rule body B in \( \Pi_n \). By the Lin-Zhao Theorem (Lin and Zhao 2004) or the (generalized) Fages Theorem (Erdem and Lifschitz 2003; You, Yuan, and Zhang 2003), \( \Pi_n \) is equivalent to its completion Comp(\Pi_n'). And then the Proposition follows from the fact that Comp(\Pi_n') = Comp(\Pi_n).

**Lemma 3.3** (Main Lemma). Let \( \Pi_n \) be a PARITY_n program. Then there is a PARITY_n program \( \Pi_n' \) s.t. \( \Pi_n' \) is equivalent to Comp(\Pi_n') and \( |\Pi_n'| \leq |\Pi_n| \).

**Theorem 3.1** (PARITY ̸∈ Poly-CP). PARITY has no polynomial size CP representations.

**Proof.** Assume the contrary that there is a sequence of PARITY_n programs \( \{ \Pi_n \} \) s.t. \( |\Pi_n| \) is bounded by a polynomial \( p(n) \). By Lemma 3.3, there is a sequence of completion-equivalent PARITY_n programs \( \{ \Pi'_n \} \) in which \( |\Pi'_n| \) is also bounded by the polynomial \( p(n) \). By Proposition 2.1, \( \{ \Pi'_n \} \) represents a language in AC^0. This contradicts PARITY ̸∈ AC^0.

**Corollary 3.1.** PARITY separates PF from CP.

**Corollary 3.2.** Suppose P ̸∈ NC^1/poly. Then CP and PF are succinctly incomparable.

### 4 Discussion and Conclusion

In this paper we show that the PARITY problem separates PF from CP, and the two formalisms are succinctly incomparable (assuming P ̸∈ NC^1/poly). Interestingly, our main result may at first appear counter-intuitive: the P-complete problem PATH has Poly-CP representations, while this does not hold for an “easy” problem PARITY. Actually, there is no contradiction. As noted in (Abiteboul, Hull, and Vianu 1995; Dantsin et al. 2001), a complete problem in a complexity class can be represented in a formalism C, does not imply that all problems in that class can be represented in C.

Generally speaking, the research of succinctness (Gogic et al. 1995; Coste-Marquis et al. 2004; Grohe and Schweikardt 2005; French et al. 2013) gives us a deeper understanding about KR formalisms, for it reveals their (in)abilities of concisely representing different problems under the condition that the encoded models are the same. This is particularly interesting if the formalisms are equally expressive and share the same reasoning complexity.

E.g., in addition to CP and NLP, programs with cardinality constraints and choice rules (CC, without classic negation ¬) (Simons, Niemelä, and Soininen 2002), (simple) definite causal theories^4 (S/DT) (Giunchiglia et al. 2004) and

\[4\] A theory is simple if each rule body is a conjunction of literals.
two-valued programs (TV) (Lifschitz 2012) are as expressive as PF and NP-complete for consistency checking. But they have a non-trivial succinct picture, see Fig. 1.

Besides the theoretical interests, succinctness also tells us something like “which for what is the best” in choosing KR formalisms for a given application. E.g., one should choose ASP instead of SAT (or DT) if the application involves reasoning about PATH or Transitive Closure\(^5\), because the former provides compact representations to avoid unnecessary overload. Recall that from the complexity viewpoint, even one extra variable may double the search space for intractable problems.

In future work we plan to establish the missing connections in Fig. 1, moreover, we will consider the succinctness of more expressive formalisms like programs with predicate symbols or higher-order atoms (Gebser, Schaub, and Thiele 2007).

Acknowledgement
We are grateful to the anonymous reviewers for their valuable comments. Thanks to Shiguang Feng, Yan Zhang, Jiankun He, Guangrui Dang and Xiaolong Liang for their helpful discussions. The research was partially supported by NSFC Grant 61272059, MOE Grant 11JJJD720020, NSSFC Grant 13&ZD186 and the Fundamental Research Funds for the Central Universities Grant 1409025.

References


\(^5\) An NL-complete problem. It is believed that NL \( \not\subseteq \text{NC}^1 / \text{Poly} \).