Compact Propositional Encoding of First-Order Theories

Deepak Ramachandran and **Eyal Amir**
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{dramacha,eyal}@cs.uiuc.edu

Abstract

In this paper we present polynomial-time algorithms that translate First-Order Logic (FOL) theories to smaller propositional encodings than achievable before in polynomial time. For example, we can sometimes reduce the number of propositions to $O(|P| + |C|)$, or $O(|P|^k \cdot \log |P|)$, for $|P|$ predicates of arity k and $|C|$ constant symbols. The guarantee depends on availability of some graphical structure in the FOL representation. Our algorithms accept all FOL theories, and preserve soundness and completeness (sometimes requiring the Domain Closure Assumption). Our experiments show significant speedup in inference with a SAT solver on real-world problems. Our results address a common approach that translates inference and decision problems that originate in FOL into propositional logic, later applying efficient SAT solvers. Standard translation techniques result in very large propositional encodings ($O(|P||C|^k)$ for predicates of arity k) that are often infeasible to solve. Our approach scales up inference for many objects, and has potential applications in planning, probabilistic reasoning, and formal verification.

1 Introduction

A *propositionalization* of a theory in First-Order Logic (FOL) is a set of propositional sentences that is satisfiable iff the original theory is satisfiable. We cannot translate arbitrary FOL theories to propositional logic because FOL is only semi-decidable. However, when possible, it is often advantageous to do so because we can use optimized, efficient SAT solvers to solve the resulting SAT problem. Propositionalization is used frequently in Planning (Kautz and Selman 1996), Relational Data Mining (Krogel *et al.* 2003), and Formal Verification (Kropf 1999).

Current propositional encodings (*naive prop.*) are based on (Gilmore 1960). They create a propositional symbol for every ground atomic formula, yielding a representation with $O(|P||C|^k)$ propositional variables, for $|P|$ predicates of maximum arity k and $|C|$ constants symbols. The intuition is that any element in our universe may influence our system, so we must check all the combinations. This results in prohibitively large propositional encodings even for moderate applications. Other propositional encodings follow from decidability results for classes of FOL theories (Börger *et al.* 1996), and do not assume finite domains, but they always result in representations of *super-exponential* size in $|P|$ and $|C|$.

In this paper we present a novel, systematic approach to translating FOL to propositional logic. Many times we assume the Domain Closure Assumption (DCA; The domain is finite and of known cardinality), but for some classes of FOL theories we do not need to assume it. Our approach leverages structure in the FOL formulation to provide significantly more compact propositional encodings.

Our algorithm generates propositional encodings of FOL as follows. It starts by using the DCA to reduce the FOL theory into one of two classes: (a) monadic FOL (*MFOL*), in which only arity 1 predicates (and constants) are allowed; and (b) exists-forall class (*EAFOL*) (aka the *Bernays-Schöenfinkel-Ramsey* class (see (Börger *et al.* 1996))), in which existential quantifiers occur before universal ones (all arity is allowed for predicates, with equality, but no functions). Then, it groups axiom sets into a tree of partitions following the approach of (Amir and McIlraith 2005; Amir 2001). Then, it translates each partition separately, using only a restricted set of constants that depends on the structure of the partitioning. Finally, it combines the translated parts into a single propositional theory.

We can leverage the structure of the FOL formulation to reduce the number of propositions from $O(|P||C|^k)$ to $O(|P|^k \cdot c^k \cdot p \cdot \log |P|)$ when each partition includes at most p predicates of arity k and at most c constant symbols. Under different conditions we get a different term: $O(|P|+|C|)(3^p+pc)$, which can be as low as $O(|P|+|C|)$ propositions. These results are significant because they lead to a uniform speed-up in the resulting SAT problem. For example, in different experiments we reduced the number of propositions from $1,000,000$ to $20,000$, allowing us to solve (in under 10 minutes) problems that are infeasible otherwise.

The rest of the paper is organized as follows: Section 2 gives some preliminary definitions. Section 3 presents an overview of our methods. In Sections 4 and 5, which form the core of our work, we describe the algorithms that use partitioning-based methods to create efficient propositional encodings. Section 6 discusses our experimental results. We omit proofs of our theorems in this paper, for lack of space.

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

2 Preliminaries

We assume familiarity with the standard definitions of First Order Logic. Also, recall that formula F is in *Prenex* form if all of its quantifiers appear before all of its literals, in which case $Matrix(F)$ denotes F without its quantifiers.

A logical theory is a set of formulae. For logical theory τ, $L(\tau)$ is its signature and $\mathcal{L}(\tau)$ is its language. $L_{pred}(\tau)$ and $L_{const}(\tau)$ are the set of predicate symbols and constant symbols respectively of τ. In the rest of the paper we restrict our attention to languages with no function symbols.

A formula in prenex form $\tau = Q.M$, is in the *monadic* class if the arity of all its predicates is exactly one. It is in the *EAFOL class* if $Q \in \exists^*\forall^*$(this is usually called the *Bernays-Schöenfinkel-Ramsey* class in Mathematical Logic and is known to be decidable (Börger *et al.* 1996)).

We use the convention that P, Q, R represent predicates, a, b, c constants in a logical theory and x, y, z are variables.

3 Overview

3.1 Motivating Example

In this section we will describe the intuition behind our work with the help of a toy example.

Consider the well-known Pigeon-principle problem. It is typical to solve it by brute force: for example by creating a boolean formula with $O(n^2)$ variables and checking for its satisfiability. In fact, looking at Figure 1(a), it does not seem that there is any kind of structure to exploit.

We will show however, that there is a simple reformulation of this problem in FOL which leads to a readily decomposable theory. Consider Figure 1(b). We have partitioned the theory into a tree and introduced the predicates h_1^+, h_2^+ and h_3^+ where for example $h_2^+(p)$ means that pigeon p is not placed in any hole i s.t. h_i appears in the subtree rooted at h_2^+ (i.e. p is not in h_1 or h_2). Then for example we have the following axiom (which appears in the root partition):

$$\forall p[h_1^+(p) \Leftrightarrow (h_2^+(p) \wedge h_3^+(p))]$$

And to exclude each pigeon from being in more than one hole we have axioms like the following:

$$\forall p[h_2^+(p) \vee h_3^+(p)]$$

Similar axioms are placed in each interior node of the tree.To ensure that no two pigeons share a hole, in each leaf partition we assert:

$$\forall p \forall q[h_i(p) \wedge h_i(q) \Leftrightarrow (p = q)]$$

Finally, in the root partition we put the following constraint to check if there is a satisfying assignment:

$$\forall p[\neg h_1^+(p)]$$

The structure that emerges from this transformation can be used to create a propositional encoding that is more compact by a factor of $\frac{n}{\log n}$. We will show in this paper that it is only necessary to create a propositional variable for every predicate and every constant that appears in the subtree of the partition in which the predicate appears. So, in this simple example, whereas the brute-force encoding had 16 propositional variables, the new encoding has 12.

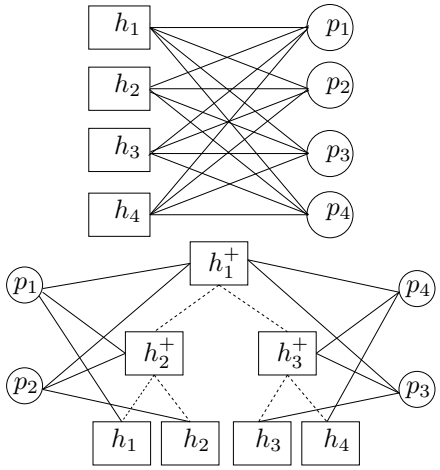

Figure 1: A naive pigeonhole encoding and its reformulation

3.2 Our Algorithm

This section presents a high-level view of the main contribution of this paper. Figure 2 presents the top-level of our algorithm, *Compact-Prop*. It takes a theory, partitions it into a tree of of sub-theories, encodes each of the parts in propositional logic separately, and returns the union of the resulting propositional theories.

Definition 1 (Tree Decomposition). *A tree decomposition is a tree of partitions $\{A_i\}_{i \leq n}$ such that if a symbol appears in A_i, A_j then it appears on the tree path between them.*

Theorem 1 (Correctness and Complexity). *Let A be a FOL theory. Procedure Compact-Prop(A,True) (Figure 2) returns propositional theory $Prop$ that is satisfiable if and only if A has a model. Compact-Prop(A,False) returns propositional theory $Prop$ that is satisfiable if A has a model. Either option takes time $O(|Prop|)$ plus the time for the chosen implementation of $treeDecomp(A)$.*

Theorem 2 (Output Size). *Let A be a FOL theory with $|P|$ predicates and $|C|$ constants. If $\langle T, \{A_i\}_{i \leq m} \rangle$ is a tree decomposition of A with degree d and at most p predicates*

Procedure Compact-Prop(FOL theory A, boolean M)

1. Let $A' \leftarrow$ *Reduce-FOL*$(Prenex(A), Const(A), M)$.
2. Partition A' into a tree T of partitions: Let $\langle T, \{A_i\}_{i \leq m} \rangle$ be the returned value from *treeDecomp*(A').
3. For every $i \leq m$, set $Prop_i \leftarrow$ *Part-Prop*(A_i, T, M)
4. Return $Prop = \bigcup_{i \leq m} Prop_i$

Procedure Part-Prop(FOL partition A_i, Tree T, boolean M)

1. If M, then return *Part-Prop-MFOL*(A_i) (see Section 5.1).
2. Otherwise, return *Part-Prop-EAFOL*(A_i, T) (see Section 5.2).

Figure 2: Our translation procedure to propositional logic.

and c constants in every partition, then Compact-Prop(A, M) returns

1. $O((|P| + |C|) \cdot (3^p + pc))$, *if* $M = TRUE$.
2. $O(|P|^k \cdot c^k \cdot p \cdot \log_d |P|)$, *otherwise, for arity-$k$ predicates.*

The intuition behind Compact-Prop is that under some conditions one may limit predicate instantiations to only some constants. Such conditions hold for theories in MFOL and EAFOL that are partitioned in a tree decomposition. In addition, when we are given an arbitrary FOL theory, we make the DCA (if possible), convert the theory to EAFOL or MFOL, and apply the propositional conversion to this new theory. Sections 4.1 and 4.2 describe these cases in detail. Section 5.3 describes the partitioning process and the general conversion of FOL to EAFOL and MFOL (assuming DCA).

A sketch for the proof of the number of propositions (Theorem 2) is as follows. If the tree has degree d and we have an EAFOL theory, then level h in the tree has d^h nodes (the root is at level 0). At every node of level h we have $d^{H-h}c$ constant symbols (there are c constants in each leaf of the tree) that may be combined with every predicate (of arity k) at that node. Thus, the total number of propositions for level h is $d^h \cdot p \cdot d^{(H-h)\cdot k} \cdot c^k = d^H \cdot d^{(H-h)\cdot(k-1)} \cdot c^k \cdot p \leq |P| \cdot p \cdot d^{(H-h)\cdot(k-1)} \cdot c^k$.

Thus, the total number of propositions overall is $O(\sum_{h=0}^{\log_d m} \cdot |P| \cdot p \cdot d^{(H-h)\cdot(k-1)} \cdot c^k)$ which is bounded from above by to $O(\log_d m \cdot p \cdot |P|^k \cdot c^k)$ for $k \geq 1$. That bound becomes $O(|P|^k \cdot c^k \cdot p \log_d |P|)$ because $m \leq |P|$. The proof for MFOL is similar.

4 Propositionalizing Partitions

The *naive prop.* of an FOL theory is created by replacing ground atoms with the corresponding subscripted proposition(e.g $P(A)$ with P_A), universally quantified formulae with the conjunction of their instantiations ($\forall x(P(x) \vee Q(x))$ with $(P_A \vee Q_A) \wedge (P_B \vee Q_B) \ldots$) and existentially quantified formulae with the disjunction of their instantiations (e.g $\exists x P(x, C)$ with $P_{\langle A,C\rangle} \vee P_{\langle B,C\rangle} \ldots$).

This approach requires the Domain Closure Assumption (DCA; every element in the universe is referenced by some constant symbol) and (if the theory has equality) the Unique Names Assumption (UNA; each constant symbol refers to a unique element). It is neither sound nor complete without them. The intuition is that there may be a model M with some element in its universe A such that $P^M(A)$ is true and thus $M \models \exists x P(x)$, but there need not exist some constant a in the theory such that $P(a)$ is true, unless the DCA holds.

Here we present principled approaches to prop. for MFOL and EAFOL without making the DCA or the UNA. These techniques will be used in the construction of the efficient partitioned propositional encodings of section 5.

4.1 Monadic Theories

We define a *factor* as a monadic FOL formula that is either an atom or is of the form $\exists x(L_1 \wedge \ldots \wedge L_n)$ where each L_i is a monadic literal with argument x and each predicate occurs at most once in some L_i.

Definition 2. *A monadic FOL formula τ in prenex form is in* proposition-ready (PR) form *if $Matrix(\tau)$ is a conjunction of disjunctions of factors.*

Theorem 3. *Every monadic FOL formula can be represented in a logically equivalent proposition-ready form.*

Example 1. Let $F = \forall y \exists x(P(x) \wedge Q(y))$. Then, the step-by-step conversion of F into PR form is shown below:

$$\begin{aligned} \forall x \exists y(P(x) \wedge Q(y)) &\Leftrightarrow \exists y \forall x(P(x) \wedge Q(y)) \\ &\Leftrightarrow \exists y(\forall x P(x) \wedge Q(y)) \\ &\Leftrightarrow \exists y(\neg \exists x \neg P(x) \wedge Q(y)) \\ &\Leftrightarrow \neg \exists x \neg P(x) \wedge \exists y Q(y) \end{aligned}$$

Note that in the first step, we have used the fact that the relative order of existential and universal quantifiers is irrelevant when all the predicates are monadic (Börger *et al.* 1996).□

We now define the alphabet of our propositional encoding. If L is the language of a monadic first order formula τ then,

$$\begin{aligned} Prop(L) \triangleq \{P_c \mid P \in L_{pred}(L), c \in L_{const}(L)\} \cup \\ \{E_{\langle [\neg]P_1, [\neg]P_2, \ldots [\neg]P_n\rangle} \mid P_1 \ldots P_n \in L_{pred}(L)\} \end{aligned}$$

Given a formula τ in PR form, each factor appearing in it can be replaced by the corresponding propositional symbol in the above alphabet. The result of these substitutions is a propositional formula we call $\mathcal{P}(\tau)$.

Definition 3. *$\mathcal{P} : \mathcal{L}(L) \rightarrow \mathcal{L}(Prop(L))$ is defined as follows: If τ is in PR form,*

1. *If $\tau = P(a)$ then $\mathcal{P}(\tau) \triangleq P_a$*
2. *If $\tau = \exists x([\neg]P_1(x) \wedge \ldots \wedge [\neg]P_n(x))$ then $\mathcal{P}(\tau) \triangleq E_{\langle [\neg]P_1, \ldots, [\neg]P_n\rangle}$*
3. *If $\tau = \neg\tau'$ then $\mathcal{P}(\tau) \triangleq \neg\mathcal{P}(\tau')$*
4. *If $\tau = \tau_1 \wedge \ldots \wedge \tau_n$, then $\mathcal{P}(\tau) \triangleq \mathcal{P}(\tau_1) \wedge \ldots \wedge \mathcal{P}(\tau_n)$*
5. *If $\tau = \tau_1 \vee \ldots \vee \tau_n$, then $\mathcal{P}(\tau) \triangleq \mathcal{P}(\tau_1) \vee \ldots \vee \mathcal{P}(\tau_n)$*

Otherwise $Prop(\tau)$ is undefined.

Now we define a set of axioms $\mathcal{E}(P, C)$ that relates propositions of the form $E_{\langle P_1 \ldots\rangle}$ with each other and those of the form P_a. Let $P = \{P_1, P_2, \ldots P_n\}$ be a set of monadic FOL predicates and C be a set of constants. Then

$$\mathcal{E}(P, C) \triangleq \bigwedge_{\substack{c \in C, k \leq n, \\ i_1 \ldots i_k \in P}} ([\neg]P_{i_1 c} \wedge \ldots [\neg]P_{i_k c} \Rightarrow E_{\langle [\neg]P_{i_1}, \ldots [\neg]P_{i_k}\rangle}) \tag{1}$$

$$\wedge \bigwedge_{\substack{k \leq n, l \leq k \\ i_1 \ldots i_k \in P}} E_{\langle [\neg]P_{i_1} \ldots [\neg]P_{i_k}\rangle} \Rightarrow E_{\langle [\neg]P_{i_1} \ldots [\neg]P_{i_l}\rangle} \wedge E_{\langle [\neg]P_{i_{l+1}} \ldots [\neg]P_{i_k}\rangle}$$

$$\wedge \bigwedge_{i_1 \ldots i_k \in P} E_{\langle [\neg]P_{i_1} \ldots [\neg]P_{i_{k-1}}\rangle} \Rightarrow E_{\langle [\neg]P_{i_1} \ldots P_{i_k}\rangle} \vee E_{\langle [\neg]P_{i_1} \ldots \neg P_{i_k}\rangle}$$

For example, if $P_c (\equiv P(c))$ is true then the axiom $P_c \Rightarrow E_{\langle P\rangle}$ ensures that $E_{\langle P\rangle} (\equiv \exists x P(x))$ is also true. Similarly, if $E_{\langle P,Q\rangle}$ ($\equiv \exists x[P(x) \wedge Q(x)]$) is true then the axiom $E_{\langle P,Q\rangle} \Rightarrow E_{\langle P\rangle} \wedge E_{\langle Q\rangle}$ ensures that $E_{\langle P\rangle}$ and $E_{\langle Q\rangle}$

are true; and if $E_{\langle P\rangle}$ is true, then by the axiom $E_{\langle P\rangle} \Rightarrow E_{\langle P,Q\rangle} \vee E_{\langle P,\neg Q\rangle}$, $E_{\langle P,Q\rangle} \vee E_{\langle P,\neg Q\rangle}$ is true.

We will use $\mathcal{E}(\tau)$ to mean $\mathcal{E}(L_{pred}(\tau), L_{const}(\tau))$.

Example 2. Let $\tau = (\neg\exists x(P(x) \wedge Q(x)) \vee \exists y(R(y) \wedge \neg S(y))) \wedge (\exists y(\neg S(y)) \vee \exists y(R(y) \wedge \neg S(y)))$. The propositionalization of τ is

$$\mathcal{P}(\tau) = (\neg E_{\langle\neg P,\neg Q\rangle} \vee E_{\langle R,\neg S\rangle}) \wedge (E_{\langle\neg S\rangle} \vee E_{\langle R,\neg S\rangle})$$

Assuming that $L_{const}(\tau) = \{a\}$

$$\mathcal{E}(\tau) = P_a \Rightarrow E_{\langle P\rangle} \quad \wedge \quad P_a \wedge \neg Q_a \Rightarrow E_{\langle P,\neg Q\rangle}$$
$$\wedge \quad E_{\langle P,\neg Q\rangle} \Rightarrow E_{\langle P\rangle} \wedge E_{\langle\neg Q\rangle} \ldots \Box$$

Theorem 4 (Correctness). *If τ is a monadic FOL theory, τ is satisfiable iff $\mathcal{P}(\tau) \wedge \mathcal{E}(\tau)$ is satisfiable. The number of propositional symbols in $\mathcal{P}(\tau)$ is at most $|P| \cdot |C| + 3^{|P|}$.*

This method creates an exponential number of propositional variables. Section 5.1 discusses how to reduce this number by a significant amount to make it usable.

4.2 The EAFOL Class

Let T be an FOL theory of the form $\forall x_1 \ldots \forall x_n M$ where M is quantifier-free, and let C be a set of constants. Then we define the operator $\otimes$ as follows:

$$T \otimes C = \bigwedge_{c_1,\ldots c_n \in C} M[c_1/x_1, \ldots, c_n/x_n] \qquad (2)$$

Theorem 5. *Let $\tau = \exists y_1 \ldots \exists y_m \forall x_1 \ldots \forall x_n M$. Then τ is satisfiable iff $\tau' = M[c_{y_1}/y_1, \ldots c_{y_m}/y_m] \otimes L_{const}(\tau) \bigcup \{c_{y_i} | 1 \leq i \leq m\}$ is satisfiable. The number of atoms in τ' is bounded by $|P|(|C| + m)^k$.*

Proof(sketch): If $m = 0$, the theorem follows directly from the closure of universal sentences under substructures (Börger *et al.* 1996): If $\mathfrak{U}$ is a substructure of $\mathfrak{B}$ and $\mathfrak{B}$ is a model of a universal theory τ, then also $\mathfrak{U} \models \tau$. Therefore it suffices to check satisfiability for structures with $|C|$ elements.

Otherwise, since the existential quantifiers occurs before the universal quantifiers, they can be replaced by skolem constants to get a universal theory for which the above case holds. $\Box$

Example 3. Let $\tau = \exists y \forall x[P(a,x) \vee Q(b,y)]$. Since τ is in the EA class, we can propositionalize τ as

$$[P(a,x) \vee Q(b,c_y)] \otimes \{a, b, c_y\}$$
$$= (P(a,a) \vee Q(b,c_y)) \wedge (P(a,b) \vee Q(b,c_y))$$
$$\wedge (P(a,c_y) \vee Q(b,c_y))$$

where c_y is the skolem constant for y and each atom is treated as a propositional variable. $\Box$

5 Creating Compact Propositional Encoding

In this section we present the subroutines of our propositional encoding algorithm Compact-Prop (Figure 2). Our method is informed by the principles of *Partition-based reasoning* (Amir and McIlraith 2005), an efficient reasoning framework for FOL where theories are decomposed into domains connected in a tree structure and inference is performed by consequence finding within the domains and *message-passing* between domains.

PART-PROP-MFOL($\mathcal{A}$)
$\mathcal{A}$ is a partition containing only monadic predicates.
1. **Return** $\mathcal{P}(\mathcal{A}) \cup \mathcal{E}(\mathcal{A})$ (see Definition 3 and Equation 1).

Figure 3: Compact Propositional encoding for Monadic theories

It relies on a subroutine *treeDecomp* that finds tree decompositions (e.g., (Amir 2001; Amir and McIlraith 2005)), for which we omit the details. Implementations can be found on the website of the authors. The success of our algorithm relies on the existence of *graphical structure* in the problem instance. However, we do not require that the problem be completely decomposable into sub-theories. Any degree of decomposition in the FOL representation can be exploited to yield a more compact SAT encoding.

The completeness and soundness of Partition-based reasoning crucially depends upon the following theorem from (Craig 1957):

Theorem 6 (Craig's Interpolation Theorem). *If α and β are First-order formula s.t. $\alpha \vdash \beta$, then there is a formula $\gamma \in \mathcal{L}(L(\alpha) \cap L(\beta))$ such that $\alpha \vdash \gamma$ and $\gamma \vdash \beta$.*

This result ensures that if all messages from one partition that are relevant to another are sent to it, then the inference procedure is complete.

5.1 Partitioned Encoding for the Monadic Class

For the MFOL class, the propositionalization algorithm (PART-PROP-MFOL in Figure 3) needs to propositionalize each predicate with only the constants in the partition it occurs in. The reason is Craig's Interpolation Theorem. Consider a simple situation with two partitions. We can prove UNSAT iff there is an interpolant (γ) in the intersection of the languages of the partitions such that $\mathcal{A}_1 \models \gamma$ and $\mathcal{A}_2 \cup \gamma \models FALSE$. If indeed our theory is UNSAT, then we have such a γ. Then, the propositional-ready form of γ corresponds to a propositional sentence in the MFOL propositionalization of the previous section. Theorem 4 guarantees that the propositionalization of $\mathcal{A}_1$ indeed entails the propositionalization of γ, and that the propositionalization of $\mathcal{A}_2$ entails the negation of that γ. Thus, we can conclude that the union of the propositionalizations will be UNSAT. A similar argument gives the other direction (SAT).

Theorem 7. *Let $\mathcal{A} = \bigcup_{i \leq n} \mathcal{A}_i$ be a partitioned monadic theory with a tree decomposition G. Then $\mathcal{A}$ is satisfiable iff the output of Algorithm COMPACT-PROP(which calls PART-PROP-MFOL) is satisfiable.*

Depending on the treewidth of the partitioning (a measure of how well it can be decomposed (Robertson and Seymour 1986)), the size of the propositional encoding is less than that of Section 4.1 by an exponential factor. This is because: (a) the number of predicates that occur in any given partition

PART-PROP-EAFOL (FOL partition $\mathcal{A}_i$, Tree T)

1. Set $C_{subtree(i)}$ the set of constants that appear in the subtree of i in T (with root at 1).
2. return $A_i \otimes C_{subtree(i)}$ (see Equation (2)).

Figure 4: Compact propositional encoding of an EAFOL partition.

Procedure Reduce-FOL(FOL theory A, constants C, boolean M)

1. If M and A in MFOL, return A.
2. If $\neg M$ and A in EAFOL, return A.
3. If A is $\forall x \psi(x)$ (some ψ in FOL with only free variable x), then return $\bigwedge_{i=1}^{|C|}$ *Reduce-FOL*$(\psi(c_i), C, M)$.
4. Else, A is $\exists x \psi(x)$ (some ψ in FOL with only free variable x), so return $\bigwedge_{i=1}^{|C|-1}(P^x(c_i) \Rightarrow$ (*Reduce-FOL*$(\psi(c_i), C, M) \vee P^x(c_{i+1}))) \wedge P^x(c_1) \wedge (P^x(c_{|C|}) \Rightarrow$ *Reduce-FOL*$(\psi(c_{|C|}), C, M))$. for a new predicate P^x.

Figure 5: Converting a FOL theory to EAFOL.

is a fraction of $|P|$, and thus the number of propositions of the form $E_{\langle P_1,\ldots,P_k\rangle}$ is exponentially smaller. (b) Each predicate does not have to be propositionalized with every constant in the theory. Assuming that the number of constants in the theory is much more than the number of predicates (usually the case in a real-world domain), the number of propositions is significantly less than even the naive prop.(theorem 2).

5.2 Partitioned Encoding for the EAFOL Class

Unlike the previous section, with EAFOL we cannot restrict the propositionalization to within each partition. For example, suppose $P(c)$ and $\forall x[P(x) \Rightarrow Q(a)]$ are in different partitions. Because we propositionalize these separately, $P(c) \Rightarrow Q(a)$ will not be in the propositional theory and thus $Q(a)$ cannot be deduced.

Unfortunately, this seems to preclude a completeness theorem (similar to Theorem 7) for EAFOL. We could not find an encoding (or proof of the current method) that allows such a theorem, at the time of this submission. Thus, this method (Figure 4) is only approximate, as far as we know. Still, our experimental results (see Section 6) for the current partitioned encoding of EAFOL show promise because all satisfiability tests with the new encodings (see Table 1) give correct answers (verified by comparing with the naive encoding).

5.3 General FOL Theories with the DCA

The subroutine *Reduce-FOL*(Figure 5) is important because it supplies the necessary conversion that makes our method applicable to all of FOL (assuming the DCA).

Traditionally, one converts a formula $Qx_1...Qx_n\varphi(x_1,...,x_n)$ to propositional logic using DCA by removing quantifiers until we get a fully propositionalized formula. One removes quantifiers recursively as follows:

- Replace $\exists x_1 Qx_2...Qx_n\varphi(x_1,...,x_n)$ with $\bigvee_{c_1 \in C} Qx_2...Qx_n\varphi(c_1,...,x_n)$.
- Replace $\forall x_1 Qx_2...Qx_n\varphi(x_1,...,x_n)$ with $\bigwedge_{c_1 \in C} Qx_2...Qx_n\varphi(c_1,...,x_n)$.

The traditional method needs to remove n quantifiers, and gets a formula of size $O(n^n * |\varphi|)$ with $O(|P| \cdot |C|^k)$ for predicates of arity k

Our approach differs from the traditional one in two ways. First, we stop the process when we reach an EAFOL or MFOL formula (avoiding some of the exponential blowup in size). Secondly, we break disjunctions into smaller clauses that are easier to decompose later (each new disjunction has at most 2 constant symbols, and the order of the constant symbols is that always c_i appears either with c_{i+1} or c_{i-1}). We get a total number of at most q new predicates (we removed q quantifiers using our replacement above).

6 Experimental Results

Hardware emulators for verifying VLSI circuits are often designed by using a number of interconnected field programmable gate arrays(FPGA's). Signals between the chips must be routed using a limited number of crossbar switches and pins on each chips. In the *Board-Level Routing Problem* (Song *et al.* 2002), given P chips, each having K groups of I/O ports, and each group having N ports, we must determine if for a set of routing constraints $S = \{n_1, \ldots, n_N\}$, where $n_t = (i, j), i \neq j, i, j \in \{1 \ldots P\}$, there exists an assignment from S to the I/O ports of the P chips s.t. for each $n_t = (i, j), type(i) = type(j)$.

(Song *et al.* 2002) solve the BLRP by encoding it directly into a SAT formula. Using our EAFOL propositionalization algorithm we obtained much more compact encodings. These translate to a uniform improvement in the running time of the SAT solvers.

Our algorithm preformed better on all the problem instances reported in (Song *et al.* 2002). The results are shown in Figure 1, where both the number of propositional variables obtained and the running time on the SAT solver are plotted against the parameters of the problem instance. On one particular instance(not plotted in the figure), where (Song *et al.* 2002) needed 1687 variables and took 1687 sec. to solve, our algorithm used 695 variables and took 413 sec.

For the second example, we took a machine scheduling problem described in (Ramchandran and Amir 2004). Briefly, there are items in an assembly line operated upon by a number of machines in sequence. Each item possesses a state manipulated by the machines. Items must be scheduled for work on particular machines at particular time steps so that they go through the correct sequence of states needed for assembly. The problem is given a set of constraints(randomly generated in our experiments) on the items and the machines, to determine whether there exists a schedule such that all items can be assembled by the deadline. This problem is in the MFOL class.

Table 2 shows the results. The MFOL algorithm outperformed the naive encoding by a large margin. [1]

[1] It would have been useful to compare the results of our methods with those of (Dixon *et al.* 2004). Unfortunately we were not

Instance (P,K,M,N)	#Vars (naive)	#Vars (part.)	ZChaff (naive)	Zchaff (part.)	Satzoo (naive)	Satzoo (part.)	Berkmin (naive)	Berkmin (part.)	Relsat (naive)	Relsat (part.)
20-5-2-07	245	224	0.68	1.93	1.8	1.6	0.10	0.46	0.7	0.5
20-5-3-14	275	184	85.88	64.82	126.3	79.4	64.12	38.4	115.3	76.6
20-5-3-11	300	232	100.71	73.23	142.7	96.1	83.56	47.89	123.5	81.4
20-7-3-8	735	420	207.79	113.57	289.8	147.8	162.33	77.67	266.8	141.8
50-5-3-8	855	287	109.56	39.92	137.0	51.2	67.92	27.1	152.3	53.7
50-7-3-8	1687	695	1453.00	413.09	1623.9	472.4	1066.02	314.12	1771.1	489.9
100-5-3-8	1720	633	112.62	34.64	166.6	49.5	69.75	22.17	168.4	59.3
200-5-3-8	3585	1014	91.11	29.37	128.9	45.7	84.11	25.99	117.5	34.2

Table 1: Board Level Routing Problem

Instance (#states,#steps)	#Vars (naive)	#Vars (part.)	ZChaff (naive)	Zchaff (part.)	Satzoo (naive)	Satzoo (part.)	Berkmin (naive)	Berkmin (part.)	Relsat (naive)	Relsat (part.)
(150,100)	45000	41776	15.1	12.3	5	7	13	18	24	18
(250,100)	90000	76524	79.4	65.2	129	75	87	68	134	88
(350,100)	135000	58419	211.7	78.4	287	91	247	85	331	109
(150,200)	75000	38697	74.2	56.3	89	60	63	42	122	78
(250,200)	150000	93442	696.8	244.6	722	266	744	277	872	287
(350,200)	225000	137868	1562.1	359.6	1487	348	1612	381	2216	511
(150,300)	105000	68645	402.5	111.2	462	126	376	99	655	194
(250,300)	210000	115581	1997.8	344.0	1844	330	1833	313	-	577
(350,300)	315000	162517	-	583.2	-	537	-	560	-	829

Table 2: The Machine Scheduling Problem

7 Conclusions and Future Work

In this paper we presented a principled approach to the problem of SAT-reducing an FOL theory. Our algorithms work in polynomial time, and produce encodings that lead to significant improvement in speed of inference.

The motivation for our work is the evidence that First Order representations hold structure that is lost in the translation to propositional logic. Rediscovering this structure in the propositional level seems equivalent to checking graph isomorphism (an NP-hard problem), so it is unlikely that SAT solvers can rediscover and use this structure at the propositional level.

We view our work in the context of a body of literature that recognizes that real world problems are structured. In particular, our techniques takes advantage of the local structure of theories in order to do efficient reasoning. For a complementary approach that exploits the symmetry of problem instances rather than their locality see (Dixon *et al.* 2004).

The success of these methods suggests that the scope of propositional encoding techniques can be expanded beyond its current usage. We believe that the first applications could be in Planning, where there is already an extensive literature on the use of propositionalization e.g. (Kautz and Selman 1996) and Probabilistic Relational inference (Poole 2003).

Acknowledgements

We would like to thank N.N. Hung for his help with our experiments.This research was supported by the DAF Air Force Research Laboratory grant FA8750-04-2-0222(DARPA REAL program).

able to obtain an implementation of their algorithms at the time of submission.

References

E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence*, 162(1-2):49–88, 2005.

E. Amir. Efficient approximation for triangulation of minimum treewidth. In *UAI'01*, pages 7–15. Morgan Kaufmann, 2001.

E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1996.

W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic*, 22:250–268, 1957.

H. E. Dixon, M. L. Ginsberg, D. K. Hofer, E. M. Luks, and A. Parkes. Implementing a generalized version of resolution. In *AAAI'04*, 2004.

P.C. Gilmore. A proof method for quantification theory: It's justification and realization. *IBM Journal of Research and Development*, 4(1):28–35, January 1960.

H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *AAAI'96*, 1996.

M. Krogel, S. Rawles, F. Zelzny, P.A. Flach, N. Lavrac, and S. Wrobel. Comparative evaluation of approaches to propositionalization. In *ILP*, pages 197–214, 2003.

T. Kropf, editor. *Introduction to Formal Hardware Verification*. Springer, 1999.

D. Poole. First-order probabilistic inference. In *IJCAI'03*, 2003.

Deepak Ramchandran and Eyal Amir. Compact propositionalization of first order theories. In *5th International Workshop on Strategies in Automated Deduction*, 2004.

N. Robertson and P. D. Seymour. Graph minors. II: algorithmic aspects of treewidth. *Journal of Algorithms*, 7:309–322, 1986.

Xiaoyu Song, Willian N. N. Hung, Alan Mischenko, Malgorzata Chrzanowska-Jaeske, Alan Coppola, and Andrew Kennings. Board level multiterminal net assignment. In *Great Lakes Symposium on VLSI*. ACM, April 2002.