Compact Propositionalizations 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. We present new insights and algorithms for converting reasoning problems in monadic First-Order Logic (includes only 1-place predicates) into equivalent problems in propositional logic. Our algorithms improve over earlier approaches in two ways. First, they are applicable even without the unique-names and domain-closure assumptions, and for possibly infinite domains. Therefore, they apply for many problems that are outside the scope of previous techniques. Secondly, our algorithms produce propositional representations that are significantly more compact than earlier approaches, provided that some structure is available in the problem. We examined our approach on an example application and discovered that the number of propositional symbols that we produced is smaller by a factor of $f \approxeq 40$ than traditional techniques, when those techniques can be applied. This translates to a factor of about 2^f increase in the speed of reasoning for such structured problems.

1 Introduction

It is often advantageous to perform reasoning with a First-Order Logic (FOL) theory by first transforming it into an equivalent propositional theory (*propositionalization*) and then using propositional inference methods on it [7]. There are a number of good reasons for this: Propositional Inference is decidable and the last 20 years of research have resulted in relatively efficient and successful algorithms (e.g. [14, 13]).

The simplest propositionalization of a First-Order theory is obtained by creating a proposition for every ground atom of the theory. Quantified variables are systematically replaced with constants from the language in the theory. This leads to $O(|P||C|^k)$ propositions, where $|P|$ and $|C|$ are the number of predicates and constants in the FOL theory and k is the maximum arity of any predicate. This propositionalization relies on the *Domain Closure Assumption*(DCA[1]; every object in the universe is referenced by a constant symbol), and the *Unique Names Assumption* (UNA; every constant symbol refers to a unique object).

Propositionalization is used in a number of applications involving First-Order representations, such as planning [8] and Relational Data Mining [10]. Many specialized propositionalization algorithms exist for such domains that use prior knowledge to construct efficient (small) propositionalizations. ILP systems such as LINUS [6] use the training data to guide construction of the propositionalizations. Bottom-Up Propositionalization [9] is tailored for biochemical databases and chooses propositions by constructing frequently occuring fragments of linearly connected atoms.

[1] This is also called the *Closed-World Assumption* (CWA)

While the DCA holds for a number of important domains (e.g., Planning) there are a number of applications where it is not reasonable, notably in problems with potentially unbounded or infinite number of objects. Also, the number of propositions that are generated for many problems (e.g., planning problems) is still prohibitively large. We address these problems in what follows.

In this paper we present algorithms that propositionalize a general monadic function-free First-Order Theory. Our algorithms do not make the DCA or the UNA, and they yield a set of propositional symbols that is significantly smaller than previous algorithms, if some structural assumptions hold. We detail these results below.

First, we show that every monadic FOL theory T (with or without DCA,UNA) can be reformulated into a propositional theory, T_p, with at most $3^P + PC$ propositional symbols, if P, C are the number of predicates and constant symbols in T, respectively. Consequently, we achieve a better complexity bound for the decidable class of monadic FOL than was known before: To the best of our knowledge, the best earlier result about inference (e.g., satisfiability) in monadic FOL is $O(2^{E^{2 \cdot U^{2^P}}})$ for a given FOL formula φ with E existentially quantified variables or constant symbols, U universally quantified variables, and P predicates [5] (this follows from the need to check all Herbrand structures of size at most $E^{2 \cdot U^{2^P}}$). In comparison, our result is independent of the number of quantifiers in φ and is at most $O(2^{3^P + PC})$ (significantly smaller in practice, using current propositional SAT solvers).

Secondly, we use structure (in the manner of *Partition-based Reasoning* [3, 2]) to obtain a propositionalization that has an exponential-factor fewer propositions than the one above. Our method is quite general and does not require special knowledge of the underlying domain or any semantic restrictions. Precisely, in many real-world cases we can partition a given theory T into n loosely dependent partitions that are arranged in a tree structure. Each partition includes axioms that are restricted to a fraction of the vocabulary of T. If each partition has $2P/n$ predicates and C/n constant symbols, then the number of propositions that we need is at most $\frac{2PC}{n} + 3^{2P/n}n$. For example, if $P = 200$, $C = 2500$, $n = 200$, then we need at most $26,200$ propositional symbols (compared with $500,000 + 3^{200}$ when no structure is used). For further comparison, consider the case when we can make the DCA,UNA. Current techniques yield $1,000,000$ propositions for the same problem. This is a factor of about 40 times more propositions (translating to computation that is slower by a factor of about 2^{40}).

The rest of this paper is organized as follows[2]. Section 2.1 gives some preliminary definitions and Section 2.2 gives a motivating example. Section 3 introduces the problem of Propositionalization, and presents our results and algorithm for propositionalizing without DCA,UNA. Section 4 presents our algorithm for compact propositionalization using structure and an analysis of the number of propositions it creates.

2 Preliminaries

2.1 Definitions

We make some definitions here that we will use later. We assume familiarity with the standard definitions of FOL. Recall that an *atomic formula* of a language is of the form $P(t_1 \dots t_k)$, where P is a k-place predicate and $t_1 \dots t_k$ are terms. Atomic formulas not containing variables are called *atoms*. A *signed atomic formula* or *literal* is either an atomic formula or a negated atomic formula. The *Matrix* of a formula F, denoted $Matrix(F)$, is the formula obtained by deleting each occurrence of a quantifier as well as the occurrence of the variable immediately to its right. In this paper we will mainly be concerned with monadic predicates, i.e., predicates of one variable. A *factor* is a monadic First-Order formula that is a monadic atom or is of the form $[\neg]\exists x(F_1 \wedge F_2 \dots F_n)$ where each F_i is a monadic literal with argument x and each predicate occurs at most once in some F_i.

For a logical theory τ, $L(\tau)$ is its signature (the set of non-logical symbols) and $\mathcal{L}(\tau)$ is its language (the set of formulas built with $L(\tau)$). $L_{pred}(\tau)$ and $L_{const}(\tau)$ are the set of predicate symbols and constant symbols respectively of τ. For the rest of the paper, we assume that all logical theories are function-free. We will use the convention that A, B, C stand for constants in a logical theory, x, y, z are variables and a, b, c are objects in a universe.

2.2 Motivating Example

Consider the following example. A factory has two machines Q^1 and Q^2 that can process items in incremental time steps. Every item is in one of n states at any point in time. We use $P_i(j)$ to denote that our item is in state $j \leq n$ at time i. The machine is either available or not at any point in time to treat items in state j (it may treat more than one item at a time, though, if they are in different states (e.g., consider machines that works with a pipeline)). We write $Q_i^r(j)$ to say that the machine r is available at time i to treat item j.

The function of the machine is to preserve the state of the item. If at a given time, neither machine is available to process the item, then it is lost. Thus the following relations hold for every time step i.

$$\forall j\ Q_i^1(j) \Rightarrow (P_i(j) \Rightarrow P_{i+1}(j)) \tag{1}$$
$$\forall j\ Q_i^2(j) \Rightarrow (P_i(j) \Rightarrow P_{i+1}(j)) \tag{2}$$

At certain predefined time steps, if an item in a certain state can be moved to another state. For example in time step 2314, an item in state 42 can be moved to state 29. We represent this information as follows.

$$\neg Q_i^1(42) \wedge \neg Q_i^2(42) \Rightarrow (P_{2314}(42) \Rightarrow P_{2315}(29)) \tag{3}$$

[2] The authors wish to apologize for the rough nature of this manuscript. It constitutes work in progress. The final version will be significantly more polished.

In addition, there are a number of constraints on the scheduling of the machines. For example,

$$\forall i\ Q_{12}^1(i) \Rightarrow (\neg Q_{13}^1(i) \vee \neg Q_{14}^1(i))$$
$$\neg Q_{313}^1(56)\ \vee\ \neg Q_{314}^2(56)$$
$$\vdots$$

Assume that we know such relationships among the first 125 time steps, but have no knowledge about steps beyond 125 (e.g., because our scheduling personnel do not look beyond 125 steps).

Finally, suppose that we know that an item is in state 1 initially (time 1), and we wish to know if it is possible for the system to reach a state n after say, 125 steps. Call the axioms above T. Then our task is to determine if $T \wedge P_1(1) \models P_{125}(n)$.

Assume that we try to solve this problem by making the DCA and a naive propositionalization in the style of [8],, then the number of propositional symbols is the number of constants times the number of predicates, i.e., $500 \cdot 3n$, which is impractical even for a small number of timesteps n. For example, for 2500 states we get $9, 37, 500$ propositional variables, a number that is way beyond the capabilities of current SAT solvers. In the next two sections we will see an approach that leads to a solution for this problem without DCA and with at most $20, 000$ variables (a reduction by a factor of 50), which is a solvable size with state-of-the-art SAT technology.

3 Propositionalizing First-Order Theories

First-Order Theory	Propositionalization
$\neg(P(A) \wedge Q(B))$	$\neg(P_A \wedge Q_B)$
$\forall x P(x) \Rightarrow \forall y \neg Q(y)$	$(P_A \wedge P_B \wedge P_C \Rightarrow (\neg Q_A \wedge \neg Q_B \wedge \neg Q_C))$
$\exists z(R(z, C) \wedge \neg Q(x))$	$(R(A, C) \wedge \neg Q(A)) \vee (R(B, C) \wedge \neg Q(B)) \vee (R(C, C) \wedge \neg Q(C))$

Table 1. Propositionalization with the DCA

Table 1 gives examples of propositionalizations of First-Order theories, created by replacing ground atoms with the corresponding subscripted proposition (e.g $P(A)$ with P_A), Universally quantified formulas with the conjunction of their instantiations ($\forall x(P(x) \vee Q(x))$ with $(P_A \vee Q_A) \wedge (P_B \vee Q_B) \dots$) and existentially quantified formulas with the disjunction of their instantiations (e.g $\exists x P(x, C)$ with $P_{\langle A,C \rangle} \vee P_{\langle B,C \rangle} \dots$).

Converting a First-Order theory into a propositional satisfiability problem as shown, is neither sound nor complete unless the DCA is made. The intuition is that there may be a model M with some object in its universe a such that $P^M(a)$ is true. There, $M \models \exists P(x)$, but there need not exist some constant A in the theory such that $P(A)$ is true, unless the DCA holds.

The DCA is reasonable in a planning scenario, since one expects the world to be more or less completely specified by the initial assumptions and operator definitions. Several techniques have been employed in the planning literature to obtain optimized propositional encodings of planning problems stated in a Situation Calculus [11] formalism. Some use lifted causal encodings, an idea borrowed from the Theorem Proving community and others reduce the number of variables by compiling away state variables and fluents. A good introduction is [8].

3.1 Propositionalization Without the DCA

We now demonstrate a technique to construct a propositionalization of a monadic function-free FOL theory with open domain semantics. Although this propositionalization is sound and complete, it creates an excessively large number of propositions. In Section 4 we describe how to make a more compact propositionalization by using a Partitioned Reasoning algorithm.

It is common to try to propositionalize a theory by creating a new constant for every existentially quantified variable. This will not work however for universally quantified formulas. For example, the formula $\forall x P(x) (\equiv \neg \exists x \neg P(x))$ would become $P(c)$ for some new constant c, but this does not mean the predicate P is true for *every* argument. In general, it is impossible to describe an algorithm to convert an arbitrary FOL formula into a propositionalization since that would imply the existence of a decision procedure for FOL. Hence, we choose to concentrate on *decidable fragments* [5] of FOL, specifically monadic logic. Essentially our idea is to introduce a proposition for every formula of the form $\exists x F$, where F is a conjunction of literals. The propositionalization is constructed in such a way that the semantic meaning of the existential operator is preserved. We show that the resulting propositionalized theory is implicationally equivalent to the original First-Order theory. Reasoning with it is therefore sound and complete.

Definition 1. *A monadic First-Order formula τ in prenex form is in* proposition-ready form *if $Matrix(\tau)$ is a conjunction of disjunctions of factors.*

Theorem 1. *Algorithm Make-SPR (Figure 1) converts every monadic First-Order formula τ to a logically equivalent formula τ'' in proposition-ready form.*

For any monadic FOL formula τ, let the result of Make-SPR be the *standard proposition-ready form* of τ, $SPR(\tau)$. We now define what it means to propositionalize τ in syntactic terms. First we define the set of propositions generated from a given formula. If L is the language of a monadic first order Formula τ then,

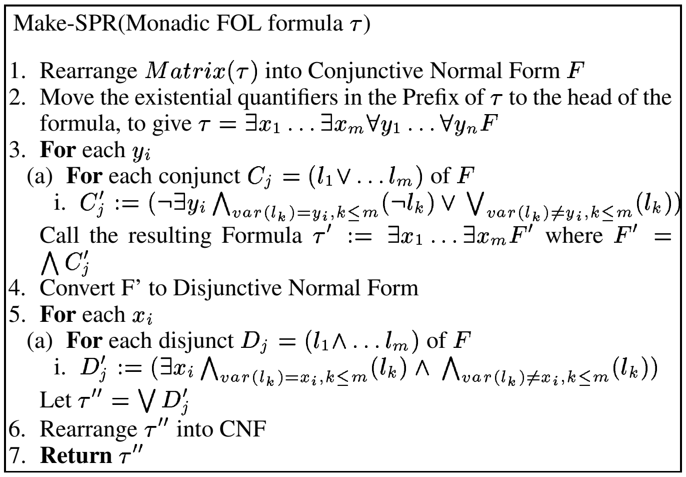
$$Prop(L) \triangleq \{P_c | P \in L_{pred}(L), c \in L_{const}(L)\} \cup \{E_{\langle [\neg]P_1, [\neg]P_2, \ldots [\neg]P_n \rangle} | P_1 \ldots P_n \in L_{pred}(L)\}$$

Definition 2. *$\mathcal{P} : \mathcal{L}(L) \rightarrow \mathcal{L}(Prop(L))$ is defined as follows.*
If τ is in proposition-ready form,
1. *If $\tau = P(a)$ then $\mathcal{P}(\tau) \triangleq P_a$*
2. *If $\tau = \exists x([\neg]P_1(x) \wedge [\neg]P_2(x), \ldots [\neg]P_n(x))$ then $\mathcal{P}(\tau) \triangleq E_{\langle [\neg]P_1, [\neg]P_2 \ldots [\neg]P_n \rangle}$*
3. *If $\tau = \neg\tau'$ then $\mathcal{P}(\tau) \triangleq \neg\mathcal{P}(\tau')$*
4. *If $\tau = \tau_1 \wedge \tau_2 \ldots \tau_n$, then $\mathcal{P}(\tau) \triangleq \mathcal{P}(\tau_1) \wedge \mathcal{P}(\tau_2) \ldots \mathcal{P}(\tau_n)$*
5. *If $\tau = \tau_1 \vee \tau_2 \ldots \tau_n$, then $\mathcal{P}(\tau) \triangleq \mathcal{P}(\tau_1) \vee \mathcal{P}(\tau_2) \ldots \mathcal{P}(\tau_n)$*

Finally, if τ is not in proposition-ready form, then $Prop(\tau) = Prop(SPR(\tau))$.

Now, we show the relation between a monadic FOL Theory and its propositionalization.

We define a set of axioms $\mathcal{E}(P, C)$ that relate values of our propositions. Let $P = \{P_1, P_2, \ldots P_n\}$ be a set of monadic FOL predicates and C be a set of constants. Then,



```
Make-SPR(Monadic FOL formula $\tau$)

1. Rearrange $Matrix(\tau)$ into Conjunctive Normal Form $F$
2. Move the existential quantifiers in the Prefix of $\tau$ to the head of the
   formula, to give $\tau = \exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n F$
3. For each $y_i$
  (a) For each conjunct $C_j = (l_1 \vee \ldots l_m)$ of $F$
      i. $C'_j := (\neg\exists y_i \bigwedge_{var(l_k)=y_i,k\leq m}(\neg l_k) \vee \bigvee_{var(l_k)\neq y_i,k\leq m}(l_k))$
   Call the resulting Formula $\tau' := \exists x_1 \ldots \exists x_m F'$ where $F' = \bigwedge C'_j$
4. Convert F' to Disjunctive Normal Form
5. For each $x_i$
  (a) For each disjunct $D_j = (l_1 \wedge \ldots l_m)$ of $F$
      i. $D'_j := (\exists x_i \bigwedge_{var(l_k)=x_i,k\leq m}(l_k) \wedge \bigwedge_{var(l_k)\neq x_i,k\leq m}(l_k))$
   Let $\tau'' = \bigvee D'_j$
6. Rearrange $\tau''$ into CNF
7. Return $\tau''$
```

Figure 1. Conversion to Standard Proposition Ready Form

$$\begin{aligned}\mathcal{E}(P,C) \triangleq & \bigwedge_{c\in C, k\leq n, i_1 \ldots i_k \in P} ([\neg]P_{i_1 c} \wedge \ldots [\neg]P_{i_k c} \Rightarrow \\ & E_{\langle [\neg]P_1, \ldots [\neg]P_k \rangle}) \bigwedge \\ & \bigwedge_{k\leq n, l\leq k, i_1 \ldots i_k \in P} (E_{\langle [\neg]P_{i_1}, \ldots [\neg]P_{i_k} \rangle} \Rightarrow \\ & E_{\langle [\neg]P_{i_1}, \ldots [\neg]P_{i_l} \rangle} \wedge E_{\langle [\neg]P_{i_{l+1}}, \ldots [\neg]P_{i_k} \rangle})\end{aligned}$$

We sometimes use $\mathcal{E}(\tau)$ to mean $\mathcal{E}(L_{pred}(\tau), L_{const}(\tau))$. For example, consider the formula τ in Table 2. The development of its propositionalization is shown.

τ	$\forall x \exists y[(P(x) \vee Q(x) \vee R(y)) \wedge \neg S(y)]$
SPR(τ)	$(\neg\exists x(P(x) \wedge Q(x)) \vee \exists y(R(y) \wedge \neg S(y))) \wedge (\exists y(\neg S(y)) \vee \exists y(R(y) \wedge \neg S(y)))$
$\mathcal{P}(\tau)$	$(\neg E_{\langle \neg P, \neg Q \rangle} \vee E_{\langle R, \neg S \rangle}) \wedge (E_{\langle \neg S \rangle} \vee E_{\langle R, \neg S \rangle})$
$\mathcal{E}(L_{pred}(\tau), \{A\})$	$P_A \Rightarrow E_{\langle P \rangle} \wedge P_A \wedge \neg Q_A \Rightarrow E_{\langle P, \neg Q \rangle} \wedge E_{\langle P, \neg Q \rangle} \Rightarrow E_{\langle P \rangle} \wedge E_{\langle \neg Q \rangle} \cdots$

Table 2. Propositionalization

Intuitively, the $\mathcal{E}$-set of a theory is the set of propositional axioms required for consistency of the propositionalization. Each axiom states that the existence of a constant c for which a conjunction of literals instantiated with c can be deduced, implies that the corresponding existential proposition is true. They ensure that the interpretations of the propositions are consistent with the underlying First-Order theory. Our main result follows:

Theorem 2 (Consistency and Completeness). *If α and β are monadic FOL theories, $\alpha \models \beta$ iff $\mathcal{P}(\alpha) \wedge \mathcal{E}(L_{pred}(\alpha), L_{const}(\alpha)) \models \mathcal{P}(\beta) \wedge \mathcal{E}(L_{pred}(\beta), L_{const}(\beta))$*

Theorem 2 formalizes the idea that reasoning in $\mathcal{P}(\tau) \wedge \mathcal{E}(\tau)$ is equivalent to reasoning in τ. Thus $\mathcal{P}(\tau) \wedge \mathcal{E}(\tau)$ is the *propositionalization* of τ.

```
PART-PROP($\{\mathcal{A}_i\}_{i\leq n}, G$)

1. $\{\mathcal{A}_i\}_{i\leq n}$ a partitioning of the theory $\mathcal{A}$, $G = (V, E, l)$ a graph
   describing the connections between the partitions.
2. For $i := 1 \to n$ do
   (a) $\mathcal{A}_i' := \mathcal{P}(\mathcal{A}) \cup \mathcal{E}(\mathcal{A})$
   (b) For $j := 1 \to n$ do
       i. $l'(i,j) := Prop(l(i,j))$
3. $G' := (V, E, l')$
4. $Q' := \mathcal{P}(Q)$
5. Return $(\{\mathcal{A}_i'\}_{i\leq n}, G', Q')$
```

Figure 2. Compact Propositionalizing algorithm

```
FORWARD-MP($\{\mathcal{A}_i\}_{i\leq n}, G, Q$)

1. $\{\mathcal{A}_i\}_{i\leq n}$ a partitioning of the theory $\mathcal{A}$, $G = (V, E, l)$ a graph
   describing the connections between the partitions, $Q$ a query in
   $\mathcal{L}(A_k) (k \leq n)$
2. Determine $\prec$ as in Definition 3
3. Concurrently
   (a) Perform consequence finding for each partition $\mathcal{A}_i, i \leq n$.
   (b) For every $(i,j) \in E$ such that $i \prec j$ for every consequence $\varphi$
       of $\mathcal{A}_j$ found (or $\varphi$ in $\mathcal{A}_j$), if $\varphi \in \mathcal{L}(l(i,j))$, then add $\varphi$ to the
       set of axioms of $\mathcal{A}_i$.
   (c) If $Q$ is proved in $\mathcal{A}_k$ return YES
```

Figure 3. Message Passing

This approach creates $|P| \cdot |C| + 3^{|P|}$ propositions. In the next section we describe a method to reduce this number by a significant amount.

4 Structure and Compact Propositionalization

Section 3.1 describes a propositionalization which can require an excessively large number of propositions. An analysis of most domains shows that many of these propositions are unnecessary. For the purpose of soundness and completeness it is clearly not required to instantiate a literal with every possible constant as an argument, but only those from which useful inferences can be made. Deciding which propositions to retain should therefore be an important aspect of an efficient propositionalization algorithm. One popular strategy has been to use typed predicates or *Many-sorted Logics* [12] to restrict the set of objects that can substitute for an argument in a predicate.

We are interested in a more general setting where an efficient propositionalization can be derived purely from the syntactic features of the theory independent of its intended semantics. Specifically, our intention is to determine which predicates need to be instantiated with which constants by analyzing the global properties of the theory. Our idea is to use the principles of *Partition-based Reasoning* [3, 2] to do so.

This describes an algorithm that finds more compact propositionalizations using partitioning. We present this algorithm, its analysis and an example application in the following.

4.1 Factored Propositionalization

We will now describe our factored propositionalized algorithm. Briefly, the theory is divided into domains such that the predicates and the constants are divided among the domains as evenly as possible (Figure 4 shows this applied to our example from Section 2.2). Then, each subdomain is individually propositionalized. After this we may choose to do either of two things. The partitioning of the theory can be retained and reasoning can be done using the (sound and complete) Message-Passing algorithms described in [3, 2] (Henceforth, we call this *Method 1*). Alternatively, the domains can be merged together, creating a single propositionalized theory , to which any propositional consequence-finder can be applied (Henceforth, we call this *Method 2*).

In this view, partition-based reasoning is just a means to an end - It is a strategy to determine which propositions can be safely left out, while maintaining completeness, by exploiting Craig's Interpolation Theorem [4]. We expect that Method 1 will yield the faster reasoning algorithm, but Method 2 is presented for comparison with other propositionalization techniques. Also, for certain applications, one may need to return a single propositionalized theory, rather than a partitioned domain with a specialized message-passing algorithm.

Formally, $\{\mathcal{A}_i\}_{i\leq n}$ is a *partitioning* of a logical theory $\mathcal{A}$ if $A = \cup_i \mathcal{A}_i$. Each individual $\mathcal{A}_i$ is called a *partition*. Each partitioning defines a labeled graph $G = (V, E, l)$, which we call the *intersection graph*. In the intersection graph, each node i represents an individual partition $\mathcal{A}_i$, $(V = \{1, \ldots, n\})$, $E = \{(i,j) | L(\mathcal{A}_i) \cap L(\mathcal{A}_j) \neq \emptyset\}$, and $l(i,j) = L(\mathcal{A}_i) \cap L(\mathcal{A}_j)$.

Figure 2 presents algorithm PART-PROP. The input to PART-PROP is a partitioning of the monadic FOL theory τ and its intersection graph G. PART-PROP propositionalizes each sub-theory $\mathcal{A}_i$ and the link languages $l(i,j)$ in the manner of Section 3.1, and returns the Partitioning and intersection graph for the propositionalized theory.

Recall our example from Section 2.2. Procedure PART-PROP (Figure 2) applies to it by examining a partitioning of the set of axioms as presented diagrammatically in Figure 4. Here, every partition includes the set of axioms that describe the effects of the machine being ready and not ready, as well as knowledge about the availability of the machine in different times for different states of our item. The edges between the partitions are labeled with the set of nonlogical (predicate and constant) symbols that are shared between partitions.

Figure 3 reproduces the Message Passing algorithm FORWARD-MP from [3, 2]. Given a partitioned theory, its intersection graph and query Q in the language of one of the partitions $\mathcal{A}_k$, FORWARD-MP will try to prove Q. The idea is to reason within each partition independently and transmit *messages* that are in the intersection languages of partitions $\mathcal{A}_i, \mathcal{A}_j$ between them. The algorithm uses an ordering relation between partitions defined below:

Definition 3 ($\prec$)**.** *Given partitioned theory $\mathcal{A} = \bigcup_{i\leq n} \mathcal{A}_i$, intersection graph $G = (V, E, l)$ and query $Q \in \mathcal{L}(\mathcal{A}_k)$, let $dist(i,j) (i,j \in V)$ be the length of the shortest path between nodes i, j in G. Then $i \prec j$ iff $dist(i,k) < dist(j,k)$.*

Theorem 3. *Let $\mathcal{A} = \bigcup_{i\leq n} \mathcal{A}_i$ be a partitioned theory with the the intersection graph G being a tree. Let $k \leq n$ and φ a sentence in $\mathcal{L}(\mathcal{A}_k)$. If the reasoning procedure in each partition is sound and complete for consequence finding in FOL, then $\mathcal{A} \models \varphi$ iff*
1. FORWARD-MP(PART-PROP($\mathcal{A}$)) returns TRUE.
2. Consequence finding on PART-PROP($\mathcal{A}$) returns $\mathcal{P}(Q)$.

These correspond respectively to the Method 1 and Method 2 discussed above. The proof of Theorem 3 uses Craig's Interpolation theorem [4] stated below to show that FORWARD-MP transmits between partitions exactly those messages that are necessary for completeness.

Theorem 4 (Craig's Interpolation Theorem). *If $\alpha \vdash \beta$, then there is a formula $\gamma \in \mathcal{L}(L(\alpha) \cap L(\beta))$ such that $\alpha \vdash \gamma$ and $\gamma \vdash \beta$.*

FORWARD-MP ,as described, will work only when the intersection graph is a tree and the query is contained in the language of one of the partitions. It is possible to modify FORWARD-PROP such that it works when neither of these conditions are true. The details will be given in the final version, and are along the lines given in [3].

The quality of the propositionalization obtained depends on how balanced the partitions are, that is how evenly the predicates and constants are divided in the partitions. Finding a balanced partitioning can be done with human guidance or automatically. The problem is reducible to finding graph decompositions of minimum treewidth. A good reference is [1].

We conclude this section with a note. The algorithms above give sound and complete propositionalizations without the DCA by using the $\mathcal{E}$-sets. Even if we are allowed to make the DCA for the entire problem, a partitioned propositionalization would still need the $\mathcal{E}$-sets for completeness. The reason is that, even though the theory as a whole is closed, each individual partition is not, as the constant representing an object could be in a different partition.

4.2 Analysis

Consider a theory τ with $L_{pred}(\tau) = P, L_{const}(\tau) = C$. Let $\mathcal{A}$ be a partitioning of τ into $\mathcal{A}_1, \mathcal{A}_3 \ldots \mathcal{A}_n$. Assume that $L_{pred}(\mathcal{A}_i) = O(|P|/n)$ and $L_{const}(\mathcal{A}_i) = O(|C|/n)$. The number of propositions created by a naive propositionalization of τ is $N(\tau) = |P| \cdot |C| + 3^P$. The number of propositions needed to propositionalize $\mathcal{A}$ is

$$\begin{aligned}
N(\mathcal{A}) &= \sum_{1 \le i \le n} (L_{pred}(\mathcal{A}_i) \cdot L_{const}(\mathcal{A}_i) + 3^{L_{pred}(\mathcal{A}_i)}) \\
&= O(n \cdot (\frac{|P| \cdot |C|}{n^2} + 3^{\frac{|P|}{n}})) \\
\frac{N(\tau)}{N(\mathcal{A})} &= \Omega(\frac{|P| \cdot |C| + 3^P}{n \cdot (\frac{|P| \cdot |C|}{n^2} + 3^{\frac{|P|}{n}})}) = \Omega(\frac{3^{|P|}}{n 3^{\frac{|P|}{n}}}) \\
&= \Omega(1/n \cdot 3^{(1-1/n)|P|})
\end{aligned}$$

Suppose that the DCA is made in τ. Then the number of propositions required is $|P||C|$.

$$\frac{N(\tau)}{N(\mathcal{A})} = \Omega(\frac{|P| \cdot |C|}{n \cdot (\frac{|P| \cdot |C|}{n^2} + 3^{\frac{|P|}{n}})})$$

If the number of constants in the theory is fairly large, i.e. $|C| \geq \frac{n^2 3^{|P|/n}}{|P|}$ we still get an improvement in the number of propositions that is proportional to the number of partitions.

$$\frac{N(\tau)}{N(\mathcal{A})} = \Omega(n)$$

Finally, a computation examination of the propositionalization algorithm PART-PROP shows that it takes time that is linear in the size of the output.

4.3 The Factory Example Revisited

Recall the machine scheduling problem outlined in Section 2.2. We now outline our Partition-Based approach to solve this problem problem.

We create a partition $\mathcal{A}_i$ for every time step i. and place the following (propositional) axioms in it.(This is the propositionalalization of Equations 1 and 2)

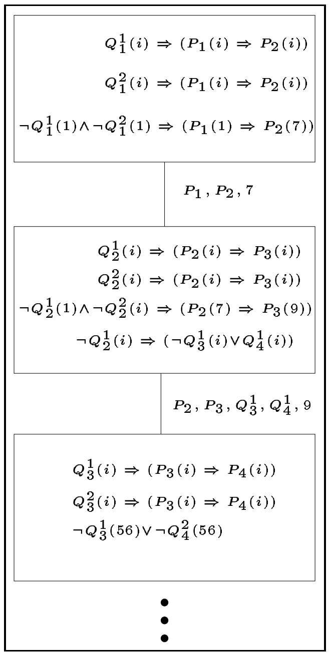
$$\neg E_{\langle \neg Q_i^1 \rangle} \Rightarrow (\neg E_{\langle \neg P_i \rangle} \Rightarrow \neg E_{\langle \neg P_{i+1} \rangle})$$

Note that we have used the fact that $\forall x P(x) \equiv \neg \exists \neg P(x) \equiv \neg E_{\langle \neg P \rangle}$. Also, if the system allows the item to move from state j to k at time step i, then we also add

$$P_{ij} \Rightarrow P_{i+1\,k}$$

the propositionalization of Equation 3.

Similarily we propositionalize any constraints on the schedules of Q^1 and Q^2 that may exist and add them to the appropriate partition. Figure 4 gives an example of the final partitioned theory(before propositionalization).



$Q_1^1(i) \Rightarrow (P_1(i) \Rightarrow P_2(i))$
$Q_1^2(i) \Rightarrow (P_1(i) \Rightarrow P_2(i))$
$\neg Q_1^1(1) \wedge \neg Q_1^2(1) \Rightarrow (P_1(1) \Rightarrow P_2(7))$
$P_1, P_2, 7$
$Q_2^1(i) \Rightarrow (P_2(i) \Rightarrow P_3(i))$
$Q_2^2(i) \Rightarrow (P_2(i) \Rightarrow P_3(i))$
$\neg Q_2^1(1) \wedge \neg Q_2^2(i) \Rightarrow (P_2(7) \Rightarrow P_3(9))$
$\neg Q_2^1(i) \Rightarrow (\neg Q_3^1(i) \vee Q_4^1(i))$
$P_2, P_3, Q_3^1, Q_4^1, 9$
$Q_3^1(i) \Rightarrow (P_3(i) \Rightarrow P_4(i))$
$Q_3^2(i) \Rightarrow (P_3(i) \Rightarrow P_4(i))$
$\neg Q_3^1(56) \vee \neg Q_4^2(56)$

Figure 4. A partitioning of theory T (Section 2.2), which describes machines and item state in a factory.

Each partition has at least 4 predicates $P_i, P_{i+1}, Q_i^1, Q_i^2$ (it may have more due to additional constraints on the Q's but we will assume they are not significant). The structural assumption we make with this partitioning is that only a limited set of constants (ie. states), say at most 20, appears in each partition.

Using Method 2 for this propositionalization, we get a propositionalization with $n(3^4 + 4 \cdot 20)$ propositional symbols. Substituting the number of time steps to be ,say, 125 (from the example in Section 2.2) gives us 20,000 propositions, which is solvable using state-of-the-art SAT solvers. For comparison, we showed in Section 2.2, that a conventional propositionalization creates nearly 1 million propositional symbols, which is beyond the capabilities of any current SAT solver.

5 Conclusions and Future Work

We have described algorithms that construct compact propositionalizations of function-free monadic First Order Logic Theories by exploiting structure in them. Our methods are quite general and result in significant savings in the number of propositional symbols required. They have applications to a number of domains that use logical reasoning such as Program Verification, Deductive Databases, Planning and Commonsense Reasoning.

In the near future we expect to extend this approach to reasoning with equality and binary predicates. We eventually intend to explore applications of our methods to Planning and Probabilistic Relational Models.

REFERENCES

[1] Eyal Amir, 'Efficient approximation for triangulation of minimum treewidth', in *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI '01)*, pp. 7–15. Morgan Kaufmann, (2001).

[2] Eyal Amir and Sheila McIlraith, 'Partition-based logical reasoning', in *Principles of Knowledge Representation and Reasoning: Proc. Seventh Int'l Conference (KR '2000)*, pp. 389–400. Morgan Kaufmann, (2000).

[3] Eyal Amir and Sheila McIlraith, 'Partition-based logical reasoning for first-order and propositional theories', *Artificial Intelligence*, (2004). Accepted for publication.

[4] William Craig, 'Linear reasoning. A new form of the Herbrand-Gentzen theorem', *Journal of Symbolic Logic*, **22**, 250–268, (1957).

[5] Burton Dreben and Warren D. Goldfarb, *The decision problem; solvable classes of quantificational formulas*, Addison-Wesley, 1979.

[6] S. Dzeroski and N. Lavrac, 'Learning relations from noisy examples: An empirical comparison of linus and foil', in *Proceedings of the 8th International Workshop on Machine Learning*, eds., L. Birnbaum and G. Collins, pp. 399–402. Morgan Kaufmann, (1991).

[7] 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).

[8] Henry Kautz and Bart Selman, 'Pushing the envelope: Planning, propositional logic, and stochastic search', in *Proc. National Conference on Artificial Intelligence (AAAI '96)*, (1996).

[9] Stefan Kramer and Eibe Frank, 'Bottom-up propositionalization', in *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, eds., J. Cussens and A. Frisch, pp. 156–162, (2000).

[10] Nada Lavrac Mark-A. Krogel and Stefan Wrobel, 'Comparative evaluation of approaches to propositionalization', in *ILP*, pp. 197–214, (2003).

[11] John McCarthy and Patrick J. Hayes, 'Some philosophical problems from the standpoint of artificial intelligence', in *Machine Intelligence 4*, eds., B. Meltzer and D. Michie, 463–502, Edinburgh University Press, (1969).

[12] Karl Meinke and John V. Tucker, *Many Sorted Logic and its Applications*, Wiley Proffesional Computing, John Wiley & Sons, Chichester, New York, 1993.

[13] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik, 'Chaff: Engineering an Efficient SAT Solver', in *Proceedings of the 38th Design Automation Conference (DAC'01)*, (2001).

[14] Bart Selman, Henry A. Kautz, and Bram Cohen, 'Noise strategies for local search', in *Proc. 12th National Conference on Artificial Intelligence, AAAI'94, Seattle/WA, USA*, pp. 337–343, (1994).