The Complexity of Restricted Consequence Finding and Abduction

Alvaro del Val
E.T.S. Informática
Universidad Autónoma de Madrid
delval@ii.uam.es
http://www.ii.uam.es/∼delval.

Abstract

We analyze the complexity of propositional kernel resolution (del Val 1999), a general method for obtaining logical consequences in restricted target languages. Different choices of target are relevant to important AI tasks, e.g. prime implicates, satisfiability, abduction and non-monotonic reasoning, and polynomial-size knowledge compilation.

Based on a generalized concept of induced width, we identify new tractable classes for various targets, and show how to estimate in advance the complexity of every problem, under various atom orderings. This can be used to choose an ordering for kernel resolution.

Two applications are discussed: estimating the number of prime implicates of any theory; and identifying tractable abduction and diagnosis problems.

Introduction

Kernel resolution (del Val 1999) is a powerful method for generating consequences of a logical theory, where we can restrict in arbitrary ways which consequences we are interested in by looking only for consequences in some "target language," a subset of the full language over a given vocabulary. We analyze the complexity of propositional kernel resolution in terms of "structural" parameters closely related to induced width, a well-known parameter for characterizing the structure of logical theories and many other AI and CS problems (Dechter and Rish 1994; Dechter 1999; Bodlaender 1993). In particular, we identify new tractable classes for a variety of consequence-finding tasks, and show how to estimate space and time requirements of variously restricted kernel resolution on *every* problem instance.[1] These estimates can also be used to choose among various atom orderings prior to running kernel resolution, a choice which is crucial to its efficiency. Of special interest is our ability to estimate, for the first time, the number of prime implicates of any theory; and the identification of tractable classes for tasks such as abduction and diagnosis.

The complexity analysis in this paper is a generalization of the analysis of (del Val 2000), in these proceedings. Reading (del Val 2000) before the present paper may help the reader to understand better the results presented here. Part of the analysis of (del Val 2000) appears here as the special case for the restricted language $\mathcal{L}_\Box$.

We assume familiarity with the standard terminology of propositional reasoning and resolution. Some definitions are as follows. A clause C subsumes a clause D iff $C \subseteq D$. The empty clause is denoted $\Box$. For a theory (set of clauses) Σ, we use $\mu(\Sigma)$ to denote the result of removing all subsumed clauses from Σ. An implicate of Σ is a clause C such that $\Sigma \models C$; a prime implicate is an implicate not subsumed by any other implicate. We denote by $PI(\Sigma)$ the set of prime implicates of Σ. We are often interested only in some subset of $PI(\Sigma)$. For this purpose, we define the notion of a target language $\mathcal{L}_T$, which is simply a set of clauses. We assume $\mathcal{L}_T$ is *closed under subsumption (c.u.s.)*, i.e. for any $C \in \mathcal{L}_T$ and $D \subseteq C$, we have $D \in \mathcal{L}_T$. A target language can always be closed under subsumption by adding all subsumers of clauses in the language.

Given these definitions, the task we are interested in is finding the prime $\mathcal{L}_T$-implicates of Σ, defined as $PI_{\mathcal{L}_T}(\Sigma) = PI(\Sigma) \cap \mathcal{L}_T$. We will mainly consider the following target languages: $\mathcal{L}$ is the full language, i.e. the set of all clauses over the set $Var(\Sigma)$ of variables of Σ. $\mathcal{L}_\Box = \{\Box\}$ contains only the empty clause. Given a set of variables V, the "vocabulary-based" language $\mathcal{L}_V$ is the set of clauses over V. Finally, for a constant K, $\mathcal{L}_K$ is the set of clauses over $Var(\Sigma)$ whose length does not exceed K. Thus we have $\mathcal{L}_1$, $\mathcal{L}_2$, etc.

Each of these languages corresponds to some important AI task. At one extreme, finding the prime implicates of Σ is simply finding $PI_{\mathcal{L}}(\Sigma) = PI(\Sigma)$; at the other extreme, deciding whether Σ is satisfiable is identical to deciding whether $PI_{\mathcal{L}_\Box}(\Sigma)$ is empty. Vocabulary-based languages also have many applications, in particular in abduction and non-monotonic reasoning, both for default logic and circumscription (see e.g. (Inoue 1992; Selman and Levesque 1996; Marquis 1999) among many others). Since most of these applications rely on some form of abductive reasoning, we will analyze the abduction problem in detail. Finally, $\mathcal{L}_K$ or subsets thereof guarantee that $PI_{\mathcal{L}_K}(\Sigma)$ has polynomial size, which is relevant to knowledge compilation (surveyed in (Cadoli and Donini 1997)).

Sometimes we will be interested in theories which are logically equivalent to $PI_{\mathcal{L}_T}(\Sigma)$, but which need not in-

[1] This should be contrasted with the usual focus on hand-crafting worst-case instances whose relevance to any other problem in the class is at best doubtful.

clude all $\mathcal{L}_T$-implicates, and can thus be much more concise. We refer to any such theory as a $\mathcal{L}_T$-LUB of Σ, following (Selman and Kautz 1996), see also (del Val 1995). We'll see one particular application of $\mathcal{L}_T$-LUBs in our discussion of diagnosis later.

After reviewing kernel resolution, we analyze its complexity for one particular exhaustive search strategy. We then present several applications of our results. Finally, we briefly consider a number of refinements and extensions to other search strategies.

Kernel resolution: Review

Kernel resolution can be seen as the consequence-finding generalization of ordered resolution (Fermüller *et al.* 1993; Bachmair and Ganzinger 1999). We assume a total ordering $o = x_1, \ldots, x_n$ of the propositional variables $\mathcal{P}$, so called atom ordering or A-ordering. We speak of x_i being "earlier" or "smaller" in the ordering than x_j just in case $i < j$. We also use l_i for either x_i or $\overline{x_i}$; the ordering is extended to literals in the obvious way, i.e. $l_i < l_j$ iff $i < j$. A *kernel clause* C is a clause split into two parts, the *skip*, $s(C)$, and the *kernel*, $k(C)$. Kernel literals, those in $k(C)$, must be larger than all the skipped literals, and are the only ones which can be resolved upon. We write a kernel clause C as $A[B]$, where $C = A \cup B$, $A = s(C)$, and $B = k(C)$. Given a set of standard clauses as input, they are transformed into kernel clauses by making $k(C) = C$, $s(C) = \emptyset$ for every C in the set. We refer to a set of kernel clauses such as these with empty skip as a *standard kernel theory.*

Definition 1 *(del Val 1999) A $\mathcal{L}_T$-kernel deduction of a clause C from a set of clauses Σ is a sequence of clauses of the form $S_1[K_1]$,...,$S_n[K_n]$ such that:*

1. *$C = S_n \cup K_n$*
2. *For every k, $S_k \cup K_k$ is not a tautology.*
3. *For every k, either:*
   - *(a) $K_k \in \Sigma$ and $S_k = \emptyset$ (input clause); or*
   - *(b) $S_k \cup K_k$ is a resolvent of two clauses $S_i \cup K_i$ and $S_j \cup K_j$ $(i, j < k)$ such that:*
     - *i. the literals resolved upon to obtain $S_k \cup K_k$ are in, respectively, K_i and K_j; and*
     - *ii. K_k is the set of all literals of $S_k \cup K_k$ which are larger than the literals resolved upon, according to the given ordering, and S_k is the set of smaller literals.*
     - *iii. $S_k[K_k]$ is $\mathcal{L}_T$-acceptable, i.e. $S_k \in \mathcal{L}_T$.*

We write $\Sigma \vdash_k^{\mathcal{L}_T} C$ to indicate that there is a $\mathcal{L}_T$-kernel resolution proof of a clause which subsumes C from Σ.

The "clausal meaning" of a kernel clause $S_k[K_k]$ is simply given by $S_k \cup K_k$. The crucial aspects are that resolutions are only permitted upon kernel literals, condition 3.b.i, and that the literal resolved upon partitions the literals of the resolvent into those smaller (the skip) and those larger (the kernel) than the literal resolved upon, condition 3.b.ii. The generality of kernel resolution for consequence finding comes from condition 3.b.iii. Since the skipped literals of a clause C are never resolved upon, they appear in all descendants of C. Assuming $\mathcal{L}_T$ is c.u.s., if $s(C) \notin \mathcal{L}_T$ then no descendant of C can be in $\mathcal{L}_T$, as any descendant is subsumed by $s(C)$. Thus any such C cannot contribute to finding $\mathcal{L}_T$-implicates, is labeled as non-acceptable, and discarded.

Var	Initial	Added
s_1	$[s_1 a_1]$	
s_2	$[s_2 a_2]$	
s_3	$[s_3 a_3]$	
a_1	$[s_1 a_1]$, $[\overline{a_1 a_2 a_3}]$	
a_2	$[s_2 a_2]$, $[\overline{a_1 a_2 a_3}]$	$s_1[\overline{a_2 a_3}]$
a_3	$[s_3 a_3]$, $[\overline{a_1 a_2 a_3}]$	$s_1[\overline{a_2 a_3}]$, $s_2\overline{a_1}[\overline{a_3}]$, $s_1 s_2[\overline{a_3}]$
		$s_3\overline{a_1 a_2}$, $s_1 s_3 \overline{a_2}$, $s_2 s_3 \overline{a_1}$, $s_1 s_2 s_3$

Table 1: $\mathcal{L}$-BE(Σ_1).

Example 1 Let $\Sigma_1 = \{s_1 a_1, s_2 a_2, s_3 a_3, \overline{a_1 a_2 a_3}\}$, and $o_1 = s_1, s_2, s_3, a_1, a_2, a_3$. The "Added" column of Table 1 shows $\mathcal{L}$-kernel resolvents obtained from Σ_1 with ordering o_1. (The reason why some clauses are repeated will become clear in Example 2.) While all these resolvents are $\mathcal{L}$-acceptable, only $s_1[\overline{a_2 a_3}]$ is $\mathcal{L}_1$-acceptable, and no resolvent is $\mathcal{L}_\Box$-acceptable. Thus $\mathcal{L}_T$-acceptability greatly restricts allowable resolvents. $\Box$

The next theorem states the completeness of $\mathcal{L}_T$-kernel resolution for consequence-finding.

Theorem 1 *(del Val 1999) Suppose $\mathcal{L}_T$ is c.u.s. For any clause $C \in \mathcal{L}_T$, $\Sigma \models C$ iff $\Sigma \vdash_k^{\mathcal{L}_T} C$.*

As special cases, for $\mathcal{L}$ all resolvents are $\mathcal{L}$-acceptable, and $\mathcal{L}$-kernel resolution finds all prime implicates of Σ. For $\mathcal{L}_\Box$, only resolvents whose skip is empty are acceptable; thus, we can only generate acceptable resolvents by resolving on the smallest literal of each clause. This is simply ordered resolution in the sense of (Bachmair and Ganzinger 1999), a satisfiability method which has been recently shown by (Dechter and Rish 1994) to be quite efficient on problems with "low induced width," on which it outperforms Davis-Putnam backtracking by orders of magnitude. For $\mathcal{L}_K$, only resolvents whose skip has at most K literals are acceptable.

In order to search the space of kernel resolution proofs, we associate to each variable x_i a bucket $b[x_i]$ of clauses containing x_i; these buckets are partitioned into $b[x_i]^+$ and $b[x_i]^-$ for, respectively, positive and negative occurrences of x_i. The clauses in each bucket are determined by an indexing function $I_{\mathcal{L}_T}$, so that $C \in b[x_i]$ iff $x_i \in I_{\mathcal{L}_T}(C)$. As shown in (del Val 1999), we can always use the function $I_{\mathcal{L}_T}(C) = \{$kernel variables of the largest prefix $l_1 \ldots l_k$ of C s.t. $l_1 l_2 \ldots l_{k-1} \in \mathcal{L}_T\}$, where C is assumed sorted in ascending order; resolving on any other kernel literal would yield a non-$\mathcal{L}_T$-acceptable resolvent.

For reasons of space, we will consider only one specific exhaustive strategy for kernel resolution, namely *bucket elimination*, abbreviated $\mathcal{L}_T$-BE. $\mathcal{L}_T$-BE processes buckets $b[x_1], \ldots, b[x_n]$ in order, computing in step i all resolvents that can be obtained by resolving clauses of

$b[x_i]$ upon x_i, and adding them to their corresponding buckets, using $I_{\mathcal{L}_T}$. We denote the set of clauses computed by the algorithm as $\mathcal{L}_T$-BE(Σ). We have $\mu(\mathcal{L}_T\text{-BE}(\Sigma)) \cap \mathcal{L}_T = PI_{\mathcal{L}_T}(\Sigma)$. Our analysis can be extended to other exhaustive strategies for kernel resolution, such as incremental saturation (del Val 1999); this is briefly discussed in the conclusion.

As shown in (del Val 1999), $\mathcal{L}$-BE is identical to Tison's prime implicate algorithm (Tison 1967), whereas $\mathcal{L}_\Box$-BE is identical to directional resolution, the name given by (Dechter and Rish 1994) to the original, resolution-based Davis-Putnam satisfiability algorithm (Davis and Putnam 1960).

For $\mathcal{L}_V$, we will in fact consider two BE procedures, both of which assume that *the variables of V are the last in the ordering.* $\mathcal{L}_V^1$-BE is simply $\mathcal{L}_V$-BE under this ordering assumption. $\mathcal{L}_V^0$-BE is identical, except that processing is interrupted right before the first variable of V is processed. Thus $\mu(\mathcal{L}_V^1\text{-BE}(\Sigma)) \cap \mathcal{L}_V = PI_{\mathcal{L}_V}(\Sigma)$, whereas $\mu(\mathcal{L}_V^0\text{-BE}(\Sigma)) \cap \mathcal{L}_V$ is logically equivalent but not necessarily identical to $PI_{\mathcal{L}_V}(\Sigma)$; in other words, it is a $\mathcal{L}_V$-LUB of Σ. Note that in either case, the desired set of clauses is stored in the last buckets. The advantage of this ordering is that either form of $\mathcal{L}_V$-BE behave exactly as directional resolution, which as said is an efficient satisfiability method, up to the first V-variable of the ordering. $\mathcal{L}_V^0$-BE stops right there (and is thus strictly cheaper than deciding satisfiability with DR under such orderings), while $\mathcal{L}_V^1$-BE continues, computing the prime implicates of $\mu(\mathcal{L}_V^0\text{-BE}(\Sigma)) \cap \mathcal{L}_V$ with full kernel resolution over the V-buckets.

Example 2 Table 1 illustrates $\mathcal{L}$-BE(Σ_1) along o_1, showing the initial and final contents of buckets. Note that clauses can be indexed in multiple buckets, or in none if they have empty kernels (listed in the bottom row). For $\mathcal{L}_1$-BE, initial buckets would differ only in that $[\overline{a_1}\overline{a_2}\overline{a_3}] \notin b[a_3]$; and there is a single $\mathcal{L}_1$-acceptable resolvent, $s_1[\overline{a_2}\overline{a_3}]$, which is added only to $b[a_2]$. This suffices to prove that Σ_1 has no unit implicates. Finally, $\mathcal{L}_\Box$-BE generates no resolvents along o_1.

Complexity of BE

We now introduce the main concepts used in our complexity analysis. The main idea is very simple. We capture the input theory Σ (which for BE we may assume to be a standard kernel theory) with a graph that represents cooccurrence of literals in clauses of Σ. We then "simulate" kernel resolution in polynomial time by processing the graph to generate a new "induced" graph. Finally, we recover from the induced graph information about all relevant complexity parameters for the hypothetical execution of BE for the given theory and ordering.

Definition 2 (split interaction graph) *Let Σ be a set of kernel clauses. The* split interaction graph of Σ *is $GS(\Sigma) = \langle V_S, E_S\rangle$, where:*

- *The set of vertices V_S is the set of all literals of Σ.*
- *E_S is a set of labeled undirected edges (l_j, l_k), where the label $L(l_j, l_k)$ is either kernel-kernel (kk), skip-skip (ss), or skip-kernel (sk).*

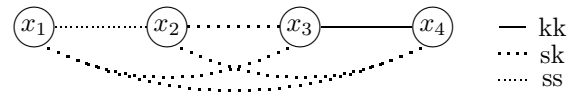

Figure 1: The split interaction graph for the single clause $x_1x_2[x_3x_4]$. Unconnected vertices not shown.

- *There is an edge $(l_j, l_k) \in E_S$ whenever there exists $C \in \Sigma$ such that $l_j, l_k \in C$. In this case, $L(l_j, l_k)$ is:*
  - *kk iff both $l_j, l_k \in k(C)$;*
  - *sk iff $l_j \in k(C), l_k \in s(C)$, or viceversa;*
  - *ss iff $l_j, l_k \in s(C)$.*

If an edge (l_j, l_k) is determined by these rules to have more than one label, then $L(l_j, l_k)$ is the largest possible label according to the order $kk > sk > ss$.

Edges represent cooccurrence of literals, where we distinguish three types of cooccurrence: in the kernel, in the skip, or mixed. In the mixed case, we speak of the $l_k \in k(C)$ as the "kernel end" of the edge, the "skip end" being the other literal $l_j \in s(C)$. In the kk case, both literals are kernel ends, in the ss case, both are skip ends. Figure 1 illustrates the interaction graph for the single kernel clause $x_1x_2[x_3x_4]$, and introduces our graphical conventions for each type of edge.

Every clause is represented in an interaction graph by a clique (fully connected graph) consisting of all its literals, with edge labels depending on where in the clause each literal occurs. All our complexity estimates will be based on approximately counting such cliques.

The split interaction graph is a generalization of the interaction graph of (Dechter and Rish 1994), in two ways: (a) our nodes are literals rather than variables, which yields a more fine-grained "simulation" of resolution; (b) we use various kinds of edges to deal with various forms of cooccurrence. Note however that (b) is irrelevant to the analysis of $\mathcal{L}_\Box$-BE, which only uses kk edges (see below), and is the only procedure analyzed in (Dechter and Rish 1994). See also (del Val 2000). We next define an "induced graph" analogous to (Dechter and Rish 1994).

Definition 3 ($\mathcal{L}$-induced split interaction graph) *Let Σ be a standard kernel theory, and $o = x_1, \ldots, x_n$ an ordering.*

The $\mathcal{L}$-induced split interaction graph of Σ along ordering *o is the graph $I_o(GS(\Sigma), \mathcal{L}) = \langle V_S, EL_S^o\rangle$ obtained by augmenting $GS(\Sigma)$ as follows:*

1. *initially, $EL_S^o = E_S$;*
2. *for $i = 1$ to n do: if $(x_i, l_j) \in E_S^o$, $(\overline{x_i}, l_k) \in E_S^o$, $j \neq k$, and $(l_j, l_k) \notin E_S^o$, then add the edge (l_j, l_k) to E_S^o, where the label $L(l_j, l_k)$ is:*
   - *(a) kk, if the edge is added as above with $i < j$, $i < k$;*
   - *(b) sk, if the edge is added with $k < i < j$ or $j < i < k$;*
   - *(c) ss, if it was added with $k < i$ and $j < i$.*

The $\mathcal{L}$-induced graph can be generated in $O(n^3)$. We adopt in what follows the convention of drawing graphs

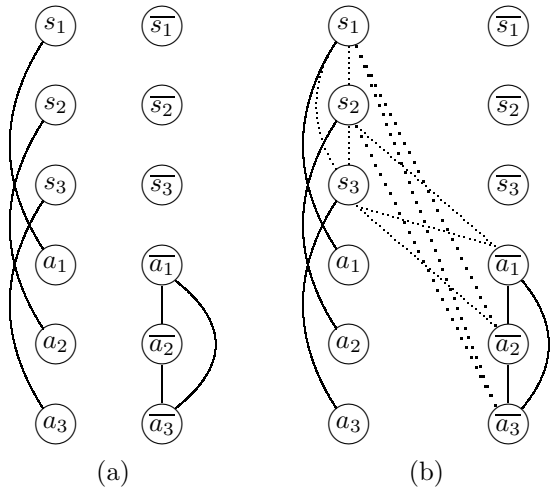

Figure 2: Interaction and $\mathcal{L}$-induced graphs for Σ_2.

with literal nodes ordered downwards along the given ordering, that is, with earlier literals above later literals, which motivates the following terminology:

Definition 4 *The* downward set $D(l_i)$ of a literal l_i along ordering o *is the set of literals* l_j *such that there is an edge* (l_i, l_j) *with* $i < j$*; the* upward set $U(v_i)$ of v_i along o *is the set of literals* l_j *such that there is an edge* (l_i, l_j) *with* $i > j$.

Both sets can be partitioned according to the type of edge. $D_{ss}(l_i)$, $D_{sk}(l_i)$ *and* $D_{kk}(l_i)$ *are respectively the subsets of* $D(l_i)$ *determined by, respectively, ss-, sk- and kk-edges. Similarly for* U_{ss}, U_{sk} *and* U_{kk}.[2]

Example 3 Figure 2 illustrates the interaction graph and $\mathcal{L}$-induced graph for the theory Σ_1 and ordering o_1 of Example 1. Note that the generation of the induced graph closely matches the generation of resolvents by $\mathcal{L}$-BE. Processing a_1 by $\mathcal{L}$-BE(Σ_1) yields the resolvent $s_1[a_2a_3]$ from $[s_1a_1]$ and $[\overline{a_1}\overline{a_2}\overline{a_3}]$ (see Table 1). In the generation of the induced graph, we find when processing a_1 that s_1 cooccurs with a_1, and $\overline{a_2}$ and $\overline{a_3}$ coocur with $\overline{a_1}$. Thus we can "predict" from the graph that resolving on a_1 will make s_1 cooccur with both $\overline{a_2}$ and $\overline{a_3}$. Further, since $s_1 < a_1 < a_2, a_3$, these cooccurrences must be of type sk. Similarly for all other added edges.

Other forms of BE can be similarly simulated:

Definition 5 *Define the following graphs:*

1. The $\mathcal{L}_\Box$*-induced graph* $I_o(GS(\Sigma), \mathcal{L}_\Box)$ *is generated as* $I_o(., \mathcal{L})$ *, except that we require* $l_j \in D(x_i)$ *and* $l_k \in D(\overline{x_i})$*. In other words, only kk-edges are added.*

2. The $\mathcal{L}_1$*-induced graph* $I_o(GS(\Sigma), \mathcal{L}_1)$ *is generated as* $I_o(., \mathcal{L})$*, except that we require that either* $l_j \notin U(x_i)$ *or* $l_k \notin U(\overline{x_i})$*. That is, only sk and kk edges are added.*

3. The $\mathcal{L}_V^0$*-induced graph* $I_o(\Sigma, \mathcal{L}_V^0)$ *is generated as* $I_o(., \mathcal{L}_\Box)$ *up to the first variable of* V *in the ordering (recall* V*-variables are assumed last in* o*), and then stops. The* $\mathcal{L}_V^1$*-induced graph* $I_o(GS(\Sigma), \mathcal{L}_V^1)$ *is generated in the same way up to the first* V *variable, then as* $I_o(., \mathcal{L})$.

[2]Note that l_i must be the skip end of sk-edges linking it to literals in $D_{sk}(l_i)$, and the kernel end of sk-edges linking it to literals in $U_{sk}(l_i)$.

Example 4 $I_{o_1}(\Sigma_1, \mathcal{L}_\Box)$ is identical to $GS(\Sigma_1)$ (Figure 2.a). $I_{o_1}(\Sigma_1, \mathcal{L}_1)$ can be obtained by eliminating all ss-edges from $I_{o_1}(\Sigma_1, \mathcal{L})$.

Our first lemma shows that the induced graphs simulate $\mathcal{L}_T$-BE as far as the interaction graph is concerned.

Lemma 2 *Let* $\mathcal{L}_T$ *be one of* $\mathcal{L}$, $\mathcal{L}_\Box$, $\mathcal{L}_V^0$, $\mathcal{L}_V^1$, *or* $\mathcal{L}_1$. $I_o(GS(\Sigma), \mathcal{L}_T)$ *is a subgraph of* $GS(\mathcal{L}_T\text{-}BE(\Sigma))$.

Proof: Similar to the proof of (del Val 2000, Lemma 2). □

We can now characterize problem complexity.

Definition 6 *Given* Σ, o, *and* $\mathcal{L}_T$ *as above, let* $D(l_i)$ *and* $U(l_i)$ *be, respectively, the downward and upward sets of* l_i *in* $I_o(GS(\Sigma), \mathcal{L}_T)$*. Define:*

$$
\begin{array}{ll}
w_i^+ = |D(x_i)| & w_i^- = |D(\overline{x_i})| \\
wk_i^+ = |D_{kk}(x_i)| & wk_i^- = |D_{kk}(\overline{x_i})| \\
ws_i^+ = |D_{ss}(x_i)| & ws_i^- = |D_{ss}(\overline{x_i})| \\
wm_i^+ = |D_{kk}(x_i) \cup D_{sk}(x_i)| & wm_i^- = |D_{kk}(\overline{x_i}) \cup D_{sk}(\overline{x_i})| \\
um_i^+ = |U_{kk}(x_i) \cup U_{sk}(x_i)| & um_i^- = |U_{kk}(\overline{x_i}) \cup U_{sk}(\overline{x_i})|
\end{array}
$$

In addition, let $w_i = w_i^+ + w_i^-$*, and similarly for* wk_i*, etc.*

These can be seen as various forms of "induced width." Note that they are relative to the $\mathcal{L}_T$-induced graph, hence to both $\mathcal{L}_T$ and o. The mnemonics are: the initial letters w and u refer respectively to downward and upward sets, and the qualifiers are k for kk, s for ss, and m for mixed, meaning sk plus kk. The closest to the standard induced width of (Dechter and Rish 1994) would be given, if we abstract from the fact that we use literals rather than variables as nodes, by the wk_i's, which for $\mathcal{L}_\Box$ are identical to the w_i's, since there are only kk edges in $I_o(GS(\Sigma), \mathcal{L}_\Box)$. Again, see (del Val 2000).

Lemma 2, together with the restrictions on the contents of buckets imposed by $I_{\mathcal{L}_T}$ allow us to show:

Theorem 3 *For* $\mathcal{L}_T \in \{\mathcal{L}, \mathcal{L}_\Box, \mathcal{L}_V^0, \mathcal{L}_V^1\}$*, the size of* $\mathcal{L}_T$*-BE*(Σ) *along ordering* o *is bounded by* $\sum_{1 \le i \le n}(2^{w_i^+} + 2^{w_i^-})$*. The size of* $\mathcal{L}_K$*-BE*(Σ) *is bounded by* $\sum_{1 \le i \le n}((1 + ws_i^+)^{K-1} 2^{wm_i^+} + (1 + ws_i^-)^{K-1} 2^{wm_i^-})$.

The number of resolutions steps performed by $\mathcal{L}_T$*-BE*(Σ) *is bounded by:*

1. $\mathcal{L}$*-BE:* $\sum_{1 \le i \le n} 2^{wk_i + um_i}$.

2. $\mathcal{L}_\Box$*-BE:* $\sum_{1 \le i \le n} 2^{wk_i}$.

3. $\mathcal{L}_V^0$*-BE:* $\sum_{x_i \notin V} 2^{wk_i}$.

4. $\mathcal{L}_V^1$*-BE:* $\sum_{x_i \in V} 2^{wk_i + um_i} + \sum_{x_i \notin V} 2^{wk_i}$.

5. $\mathcal{L}_K$*-BE:* $\sum_{1 \le i \le n}((1 + um_i^+)^K (1 + um_i^-)^K 2^{wk_i})$.

Proof: The size estimate is obtained as in (del Val 2000, Theorem 4). For time, the number of resolution steps when processing x_i is bounded by $|b[x_i]^+| \times |b[x_i]^-|$. Consider the $\mathcal{L}$ case. Let $C \in b[x_i]$, $l \in C$. Then (l, x_i) must be an edge of $GS(\mathcal{L}\text{-BE}(\Sigma))$, with x_i as a kernel end (as $C \in b[x_i]$ implies $x_i \in k(C)$, given $I_\mathcal{L}$). By Lemma 2, (l, x_i) is also in $I_o(GS(\Sigma), \mathcal{L})$. Thus, $l \in U_{sk}(x_i) \cup U_{kk}(x_i) \cup D_{kk}(x_i)$. The cardinality of this set is $um_i^+ + wk^+i$, which gives us a bound on the number of literals which cooccur with x_i in a clause which has x_i in the kernel. This immediately gives us

$|b[x_i]^+| \leq 2^{wk_i^+ + um_i^+}$. Bounding in a similar way $|b[x_i]^-|$ and summing up, we obtain the number of resolution steps given in the theorem for $\mathcal{L}$-BE. $\Box$

We emphasize that the values of w_i's in the size estimates depend on the induced graphs, which in turn depend on $\mathcal{L}_T$; in particular, $w_i = wk_i$ for $\mathcal{L}_\Box$ and $\mathcal{L}_V^0$, as well as, for $x_i \notin V$, for $\mathcal{L}_V^1$. This is made explicit in the time estimates. Similarly, um_i may be smaller in $\mathcal{L}_V^1$ than in $\mathcal{L}$.[3] As expected, $\mathcal{L}_\Box$-BE is cheaper in space and time than $\mathcal{L}$-BE, with $\mathcal{L}_K$-BE smoothly spanning the complexity gap between both;[4] and, if the V-variables are last, then $\mathcal{L}_V^1 - BE$ is cheaper than $\mathcal{L}_\Box$-BE, which is cheaper than $\mathcal{L}_V^0$-BE, which is cheaper than $\mathcal{L}$-BE.

We next rewrite these estimates more compactly.

Definition 7 *The* $\mathcal{L}_T$-induced kernel set $K(x_i)$ *of a variable* x_i *along an ordering* o *is defined by reference to* $I_o(GS(\Sigma), \mathcal{L}_T)$ *as follows:*

1. $\mathcal{L} : D_{kk}(x_i) \cup D_{kk}(\overline{x_i}) \cup U_{kk}(x_i) \cup U_{kk}(\overline{x_i}) \cup U_{sk}(x_i) \cup U_{sk}(\overline{x_i})$.

2. $\mathcal{L}_\Box : D_{kk}(x_i) \cup D_{kk}(\overline{x_i})$.

3. $\mathcal{L}_V^0 : D_{kk}(x_i) \cup D_{kk}(\overline{x_i})$ *for* $x_i \notin V$, $\emptyset$ *otherwise.*

4. $\mathcal{L}_V^1 : D_{kk}(x_i) \cup D_{kk}(\overline{x_i})$ *for* $x_i \notin V$, *and otherwise as the* $\mathcal{L}$*-kernel set of* x_i.

Definition 8 *The* $\mathcal{L}_T$*-induced s-width of an ordering* o *is* $sw(o) = max(w_i^+, w_i^- \mid 1 \leq i \leq n)$. *The* $\mathcal{L}_T$*-induced kernel width of* o *is* $kw(o) = max(|K(x_i)| \mid 1 \leq i \leq n)$.

Corollary 4 *Let* $\mathcal{L}_T \in \{\mathcal{L}, \mathcal{L}_\Box, \mathcal{L}_V^0, \mathcal{L}_V^1\}$. *Then* $|\mathcal{L}_T\text{-}BE(\Sigma)| = O(n \cdot 2^{sw(o)+1})$. *The number of resolution steps is* $O(n \cdot 2^{kw(o)})$, *hence the total time complexity is* $O(n^2 \cdot 2^{kw(o)})$.

We thus obtain tractable classes:

Corollary 5 *If the* $\mathcal{L}_T$*-induced s-width along* o *is bounded by a constant then* $\mathcal{L}_T$*-BE*(Σ) *requires only polynomial space; if the* $\mathcal{L}_T$*-induced kernel width is bounded by a constant then* $\mathcal{L}_T$*-BE along* o *takes polynomial time.*

There exist methods to recognize in $exp(k)$ theories whose *standard* induced width is bounded by the constant k (Dechter and Rish 1994); we conjecture that these methods can be adapted to recognize bounded induced s-width at least for $\mathcal{L}_\Box$, possibly for the other $\mathcal{L}_T$'s as well. Even if not, we can use the induced graphs to choose good orderings.

Applications

We next discuss very briefly some applications.

Prime implicates. Since $PI(\Sigma) = \mu(\mathcal{L}\text{-BE}(\Sigma))$, theorem 3 bounds the number of prime implicates of *any* theory. To our knowledge this is the first result of this kind in the literature (see the survey (Marquis 1999)). Much tighter estimates can be obtained along the lines discussed in the conclusion.

Diagnosis. In diagnosis (de Kleer *et al.* 1992), we are given a theory Σ describing the normal behavior of a set of components; for each component, there is an "abnormality" predicate ab_i. Let V be the set of ab_i's. Given a set of observations O (typically given as unit clauses), the diagnosis are given by the prime $\mathcal{L}_V$*-implicants*[5] of $PI_{\mathcal{L}_V}(\Sigma \cup O)$. These implicants can also be obtained from the smaller, equivalent $\mathcal{L}_V$-LUB; in other words, we can use either $\mathcal{L}_V^1$-BE or $\mathcal{L}_V^0$-BE to obtain a set of $\mathcal{L}_V$ clauses whose implicants yield the diagnosis. Thus, unlike in (de Kleer *et al.* 1992), we do not need to compute $PI(\Sigma \cup O)$; not even $PI_{\mathcal{L}_V}(\Sigma \cup O)$.

Corollary 5 yields classes of devices for which this intermediate, but crucial step of diagnosis is tractable. In addition, we can bound the size of the resulting $\mathcal{L}_V$ theory. This is simply the sum over $x_i \in V$ of $2^{w_i^+} + 2^{w_i^-}$ (which, again, may be smaller for $\mathcal{L}_V^0$).[6]

Abduction. Given a theory Σ, and a set A of variables, variously called assumptions, hypothesis, or abducibles, an *abductive explanation* of a literal l wrt. Σ is a conjunction L of literals over A such that $\Sigma \cup L$ is consistent and entails l; L is a minimal explanation iff no subset of it is also an explanation. Letting $Lit(A)$ be the set of literals over A, it is easy to show that $L \subseteq Lit(A)$ is a minimal explanation of l iff the clause $l \vee \neg L$ is in $PI(\Sigma)$ (see e.g. (Reiter and de Kleer 1987)).

Let $\mathcal{L}_A = \{l \vee C \mid l \text{ is a literal, clause } C \subseteq Lit(A)\}$. It follows from the above that we can obtain all minimal explanations of all literals from $PI_{\mathcal{L}_A}(\Sigma)$. An efficient form of $\mathcal{L}_A$-kernel resolution can be obtained by putting all assumptions A last in the ordering. The resulting BE procedure behaves as $\mathcal{L}_1$-BE up to the last variable not in A, and thereafter as $\mathcal{L}$-BE. The indexing function $I_{\mathcal{L}_A}(C)$ should index C by the kernel variables of C which are preceded by at most one non-A variable in C.

Example 5 Consider again Σ_1, and let $A = \{a_1, a_2, a_3\}$. The initial buckets would be identical to those of Table 1. $\mathcal{L}_A$-BE does not generate the resolvent $s_1 s_2 s_3$, as its parent $s_1 s_2 [\overline{a_3}]$ is rejected as not $\mathcal{L}_A$-acceptable; other resolvents are generated but rejected unless they contain exactly one s_i.

The $\mathcal{L}_A$-induced graph is identical to the $\mathcal{L}$-induced graph, except that no ss-edges linking two non-A-variables are allowed. We can use it to show:

Theorem 6 $|\mathcal{L}_A\text{-}BE(\Sigma)| \leq \sum_{1 \leq i \leq n} (2^{w_i^+} + 2^{w_i^-})$.

The number of resolution steps is bounded by $\sum_{x_i \notin A} (1 + um_i^+)(1 + um_i^-) 2^{wk_i} + \sum_{x_i \in A} 2^{wk_i + um_i}$.

[3] E.g. say $x_3 \in V$. With $\mathcal{L}$, if $x_1 \in U_{sk}(x_2)$ and $x_3 \in D_{kk}(\overline{x_2})$ we must add x_1 to $U_{sk}(x_3)$. With $\mathcal{L}_V^1$, if $x_2 \notin V$ then $U_{sk}(x_2) = \emptyset$, hence x_1 is not added to $U_{sk}(x_3)$.

[4] This is made clearer by noting that $(1 + um^+ i)^K$ can be replaced by $min((1 + um_i^+)^K, 2^{um_i^+})$ in the time estimate for $\mathcal{L}_K$, and similarly with um_i^-. This yields a maximum of $2^{um_i^+} \cdot 2^{um_i^-} \cdot 2^{wk_i}$, which equals the time estimate for $\mathcal{L}$.

[5] An implicant of a theory Γ is a conjuntion of literals which entails Γ.

[6] Use of $\mathcal{L}_V^0$ is beneficial if the ab_i's occur negatively in the theory, as it is the case when there are fault models; otherwise, both procedures yield the same result.

Note again that the w_i's can be significantly smaller than for $\mathcal{L}$, because of the restriction on ss edges. Again, if the exponents are bounded by a constant we obtain polynomial time and/or space. In this case we obtain tractable abduction in a rather strong sense, since $\mathcal{L}_A$ yields explanations for all literals. We can also bound the $\mathcal{L}_A$ clauses obtained, the number of explanations per literal, etc.

In addition to linking induced width with abduction, this tractability result is significant also because of the difficulty of abduction. Basically, the only known tractable classes are binary and definite theories. Finding *one* explanation for *one* literal is NP-complete even for acyclic Horn theories (Selman and Levesque 1996).

Polynomial size compilation The goal of knowledge compilation (Cadoli and Donini 1997) is to ensure tractable answers to all or some queries, by replacing a theory by a compiled one with better complexity. However, knowledge compilation faces fundamental limits in the sense that for many query languages it is extremely unlikely that one can guarantee tractable compiled representations of polynomial size (Selman and Kautz 1996; Cadoli and Donini 1997). It seems the only way out of this hurdle is to ensure that the query language has polynomial size. Our analysis of $\mathcal{L}_K$-consequence finding is a contribution to this goal.

Extensions and refinements

All of the above should be seen as a quite condensed summary of our complexity analysis of kernel resolution. In the long version of this paper, we refine and extend the analysis in a number of directions.

First, the estimates given can be made significantly tighter. As mentioned above, they are based on counting cliques in the $\mathcal{L}_T$-induced graphs. They do so very loosely, by simply assuming that the relevant subsets of $D(l) \cup U(l)$ are themselves cliques in the induced graph; and that these cliques have the appropriate structure (see Figure 1). While counting cliques is NP-hard, it is not difficult to devise cheap methods which provide much tighter estimates, and which can yield *new tractable classes* even with unbounded induced width. The refinements apply to the whole range of applications discussed here. It is also possible to take into account the effect of subsumption to further tigthen the estimates. All this is discussed in detail for the $\mathcal{L}_\Box$ case in (del Val 2000).

Second, we have focused only on BE, though we have mentioned that kernel resolution is compatible with other exhaustive search strategies, such as incremental saturation (IS) (del Val 1999), where clauses can be added/processed incrementally. In the extended paper, we show that we can simulate the behavior of IS as well, with suitable defined induced graphs. In a nutshell, the idea is to distinguish between passive and active edges, where the latter correspond to the newly added clause and the resolvents obtained from it and its descendants. We can thereby bound the space and time complexity of IS in a similar manner as we did with BE.

An interesting property of IS is that it allows us to obtain the *new* $\mathcal{L}_T$-implicates derivable from $\Sigma \cup C$ but not from Σ alone. Many tasks in common-sense reasoning, such as abduction and non-monotonic reasoning (both default logic and circumscription) can be cast in these terms (Inoue 1992; Marquis 1999). Thus the complexity analysis of IS helps us understand these tasks, and, again, obtain tractable classes.

References

L. Bachmair and H. Ganzinger. A theory of resolution. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier, 1999.

Bodlaender, H. L. 1993. A tourist guide through treewidth. *Acta Cybernetica* 11:1–21.

M. Cadoli and F. M. Donini. A survey on knowledge compilation. *AI Communications*, 10:137–150, 1997.

M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnosis and systems. *Artificial Intelligence*, 56:197–222, 1992.

R. Dechter and I. Rish. Directional resolution: The Davis-Putnam procedure, revisited. In *KR'94, Proc. 4th Int. Conf. on Knowledge Representation and Reasoning*, pages 134–145. Morgan Kaufmann, 1994.

R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

A. del Val. An analysis of approximate knowledge compilation. In *IJCAI'95, Proc. 14th Int. Joint Conf. on Artificial Intelligence*, pages 830–836, 1995.

A. del Val. A new method for consequence finding and compilation in restricted languages. In *AAAI'99, Proc. 16th (U.S.) Nat. Conf. on Artificial Intelligence*, pages 259–264, 1999.

del Val, A. 2000. Tractable classes for directional resolution. In *AAAI'2000, Proc. 17th (U.S.) National Conference on Artificial Intelligence*. AAAI Press/MIT Press.

C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Methods for the Decision Problem*. Springer-Verlag, 1993.

K. Inoue. Linear resolution for consequence-finding. *Artificial Intelligence*, 56:301–353, 1992.

P. Marquis. Consequence-finding algorithms. In D. Gabbay and Ph. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*. Kluwer Academic Publishers, 1999.

R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems. In *AAAI'87, Proc. 6th (U.S.) Nat. Conf. on Artificial Intelligence*, 1987.

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

B. Selman and H. J. Levesque. Support set selection for abductive and default reasoning. *Artificial Intelligence*, 82:259–272, 1996.

P. Tison. Generalized consensus theory and application to the minimization of boolean circuits. *IEEE Transactions on Computers*, EC-16:446–456, 1967.