

On Kernel Rules and Prime Implicants

Ron Rymon*

Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
rymon@isp.pitt.edu
In Proceedings AAAI-94

Abstract

We draw a simple correspondence between *kernel rules* and *prime implicants*. Kernel (minimal) rules play an important role in many induction techniques. Prime implicants were previously used to formally model many other problem domains, including Boolean circuit minimization and such classical AI problems as diagnosis, truth maintenance and circumscription.

This correspondence allows computing kernel rules using any of a number of prime implicant generation algorithms. It also leads us to an algorithm in which learning is boosted by an auxiliary domain theory, e.g., a set of rules provided by an expert, or a functional description of a device or system; we discuss this algorithm in the context of SE-tree-based generation of prime implicants.

Introduction

Rules have always played an important role in Artificial Intelligence (AI). In machine learning, while a variety of other representations have also been used, a great deal of research has focused on rule induction. Moreover, many of the other representations (e.g., decision trees) are directly interchangeable with a set of rules.

Prime implicants (PIs) are minimal conjunctions of Boolean literals. Always computed with respect to a given logical theory, a prime implicant has the property that it can be used alone to prove this theory. In the early days of computers, PIs were used in Boolean function minimization procedures, e.g., (Quine 52; Karnaugh 53; McCluskey 56; Tison 67; Slagle, Chang & Lee 70; Hong, Cain & Ostapko 74). In AI, PIs were used to formally model TMSs and ATMSs (Reiter 87; de Kleer 90), circumscription (Ginsberg 89; Raiman & de Kleer 92), and Model-Based Diagnosis (de Kleer, Mackworth & Reiter 90). A number of new, and improved PI generation algorithms have emerged, e.g., (Greiner, Smith & Wilkerson 89;

Jackson & Pais 90; Kean & Tsiknis 90; de Kleer 92; Ngair 92; Rymon 94).

In machine learning, it is commonly argued that simpler models are preferable because they are likely to have more predictive power when applied to new instances (a principle often referred to as *Occam's razor*). One way in which a model can be simpler is if all of its rules are simple, i.e., have fewer conditions in the antecedent. As it turns, *kernel* (minimal) rules and prime implicants are closely related. We will show a direct mapping between the two which allows *computing* kernel rules using PI generation algorithms. This will lead us to an algorithm which combines knowledge induced from examples with knowledge acquired from an expert, or which is otherwise available. This is done by combining the PIs of multiple theories. Given that prime implicants have been actively researched for a few decades now, we believe that this correspondence has the potential to benefit the machine learning community in other ways.

Kernel Rules are Prime Implicants

Consider a typical machine learning scenario: we are presented with a *training set* (TSET) of *class*-labeled examples. Each example is described by values assigned to a set of *attributes* (also called features or variables), and is labeled with its correct class. We assume all attributes and the class are Boolean. By *partial description* we refer to an instantiation of a subset of attributes; an *object* is a partial description in which all attributes are instantiated. By *universe* we refer to the collection of all possible objects.

It is common to assume that class labels were assigned based on a set of, unknown as yet, principles; for the purpose of this paper, we assume no noise. It is the role of the induction program to unearth these principles, or at least some approximation thereof. Numerous techniques were devised throughout the years for this purpose, ranging from various forms of regressions, to neural and Bayesian networks, to decision trees, graphs, rules and more. Rules are one form of representation which has also been heavily used in other branches of AI. One advantage of proving prop-

*Parts of this work were supported by NLM grant R01-LM-05217; an ARO graduate fellowship when the author was at the University of Pennsylvania; a NASA consulting contract; and self-funding.

erties for a rule-based representation is that rules are easily mapped into many of the other representations. In decision trees, for example, a rule corresponds to attribute-value assignments labeling a path from the root to a leaf.

Definition 1 *Kernel Rules*

A *rule* is a partial description such that all objects in TSET that agree with its instantiated variables are all labeled with the same class and such that there exists at least one such object. A *kernel rule* is a rule such that none of its subsets is a rule.

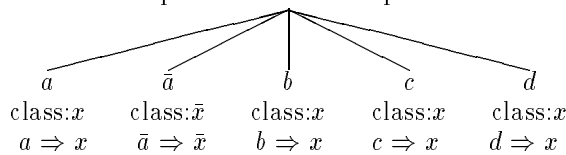
A rule is thus a *set* of instantiated variables, and a kernel rule is one which is set-wise minimal. (To save notation, we will sometimes refer to this set *with* the class variable; the distinction should be clear from the context.) Another way to view a rule is as a conjunctive set of conditions. We call it a rule because if the training data were representative of the universe, we could use it to predict the class of new instances. The more conditions are included in its conjunction, the more specific the rule is; a kernel rule is thus a most general rule.

Kernel rules are the essence of our SE-tree-based induction framework (Rymon 93). Each kernel rule corresponds to the attribute-value assignments labeling one path from the root to a leaf. We have shown that SE-trees generalize, and often outperform, decision trees as classifiers.

Example 2 Consider the following training examples consisting of various test results ($a, b, c,$ and d) of patients suspected of suffering from a disease (x):

Patient	a	b	c	d	Disease (x)
1	true	true	true	true	true
2	false	false	false	false	false
3	true	false	false	false	true

The five kernel rules inferable from these examples and their SE-tree representation are depicted next:



Definition 3 *Prime Implicants (Implicates)*

Let V be a set of Boolean variables. A *literal* is either a variable v , or its negation $\neg v$. Let Σ be a propositional theory. A conjunction of literals π is an *implicant* of Σ if $\pi \models \Sigma$ (where \models is the entailment operator). A disjunction of literals τ is an *implicate* of Σ if $\Sigma \models \tau$. Such a conjunction (disjunction) can also be thought of as a set of literals. It is a *prime implicant* (implicate) if none of its subsets is an implicant (implicate). An implicant (implicate) is *trivial* if it contains a complementary pair of literals.

Prime implicates and prime implicants are duals. In particular, any algorithm which computes prime implicates from a DNF formula can also compute prime

implicants from the corresponding CNF formula, and vice versa. Many PI generation algorithms assume the theory is given in one form or the other.

Definition 4 *Training Set Theory*

Let e be an object, and let a_i denote the attribute instantiations in e . Let x be an instantiation of the class variable. We define

$$\sigma(e, x) \stackrel{\text{def}}{=} a_1 \wedge a_2 \wedge \dots \wedge a_n \rightarrow x = \bar{a}_1 \vee \bar{a}_2 \vee \dots \vee \bar{a}_n \vee x$$

Let TSET be a set of objects $\{e_j\}_{j=1}^m$, each labeled with a class x_j . The theory given by TSET is defined:

$$\Sigma(\text{TSET}) \stackrel{\text{def}}{=} \bigwedge_{j=1}^m \sigma(e_j, x_j)$$

The purpose of this transformation is to represent logically the information contributed by a each example alone and by the collection as a whole. For the first patient in Example 2 we have:

$$a \wedge b \wedge c \wedge d \rightarrow x = \bar{a} \vee \bar{b} \vee \bar{c} \vee \bar{d} \vee x$$

As a conjunction, the training set theory can be used to constrain the classifiers considered to those who produce the same class labels for the given examples.

Theorem 5 *Kernel Rules are Prime Implicants*

Let TSET be a training set, x the class variable. Let TSET^+ be the set of positive examples, and TSET^- the set of negative examples. Let Σ^+ denote $\Sigma(\text{TSET}^+)$ and similarly let Σ^- denote $\Sigma(\text{TSET}^-)$. Let KR^+ be the set of positive kernel rules, i.e., with x in their consequent, and KR^- the set of negative kernel rules, i.e., with \bar{x} in their consequent. Let $\text{PI}(T)$ denote the collection of non-trivial PIs for a theory T . Then

$$\text{KR}^- = \text{PI}(\Sigma^+) - \{x\} \text{ modulo subsumption}^1 \text{ with } \text{PI}(\Sigma^-);$$

$$\text{KR}^+ = \text{PI}(\Sigma^-) - \{\bar{x}\} \text{ modulo subsumption with } \text{PI}(\Sigma^+).$$

Proof: Let r be a partial description. First, it is clear that x (respectively \bar{x}) is a PI for Σ^+ (respectively Σ^-)². We will prove that (1) if $r \in \text{PI}(\Sigma^+)$ and $r \neq x$ then either $r \in \text{KR}^-$ or r is subsumed by some $r' \in \text{PI}(\Sigma^-)$, and (2) vice versa, i.e., if $r \in \text{KR}^-$ then $r \in \text{PI}(\Sigma^+)$. The proof for the other part of the theorem is analogous.

- (1) Suppose $r \in \text{PI}(\Sigma^+)$ and that r is not subsumed by any PI of Σ^- . As a PI, $r \models \Sigma^+$ and thus contradicts at least one variable assignment in every positive example, and so covers none of these. We still have to show that there is a negative example that is covered by r and that r is minimal.

¹One rule subsumes another if it is a subset of the other. This operation removes from one set all rules subsumed by any rule from the other set. Note that if a PI appears in both sets, it is removed from both.

²Also note that x does not appear in any other PI for TSET^+ . In fact, to make things computationally easier, x and \bar{x} can be omitted from the respective theories; they were only included for pedagogical reasons to emphasize the correspondence between clauses and examples.

Suppose that none of the negative examples is covered by r . Since every example assigns a value to each variable, it must be the case that r contradicts every negative example by at least one variable-assignment. Thus, r is an implicant of Σ^- and therefore there is a prime implicant in $\text{PI}(\Sigma^-)$ which subsumes r . In contradiction to the assumption.

As a prime implicant, r must be minimal and therefore it is a kernel rule.

- (2) Suppose $r \in \text{KR}^-$. Then r does not cover any of the positive examples, and therefore it must contradict at least one variable assignment in each and every positive example. Thus, by definition, r is an implicant of Σ^+ . As a kernel rule, r is minimal and therefore it is a prime implicant. Q.E.D.

Consider again Example 2:

$\Sigma^+ \stackrel{\text{def}}{=} (\bar{a} \vee \bar{b} \vee \bar{c} \vee \bar{d} \vee x) \wedge (\bar{a} \vee b \vee c \vee d \vee x)$, and so $\text{PI}(\Sigma^+) = \{x, \bar{a}, \bar{b}\bar{c}, \bar{b}\bar{d}, \bar{b}\bar{c}\bar{d}, \bar{c}\bar{d}, \bar{c}\bar{d}x\}$. Computed similarly $\text{PI}(\Sigma^-) = \{\bar{x}, a, b, c, d\}$. Six of the former PIs are subsumed by some of the latter, leaving as a negative kernel rule only \bar{a} . All the PIs for Σ^- , except for \bar{x} which is removed, are positive kernel rules.

The first immediate application of Theorem 5 is that kernel rules can be *computed* using any of a number of PI generation algorithms. We briefly explore this possibility next. This theorem also leads to an opportunity to combine kernel rules with other available knowledge. As PIs, kernel rules can be combined with PIs of another theory, e.g., an auxiliary domain theory, to obtain a more refined classifier. We discuss this possibility in the subsequent sections of this paper. Besides these two immediate applications, we believe this correspondence may lead to new insights drawn from one area of research to the other.

Computing Rules as Prime Implicants

Assuming the availability of a PI generation algorithm, Theorem 5 suggests a very simple way to compute kernel rules: transform the training set into positive and negative theories; then compute the PIs for each of the theories; then, after removing the trivial x and \bar{x} , take the union of the two sets while removing subsumed conjuncts. The consequent of each rule is determined by the set from which it came: x in rules originating from $\text{PI}(\Sigma^-)$ and \bar{x} in those from $\text{PI}(\Sigma^+)$.

As previously mentioned, research over the years has produced an abundance of PI generation algorithms. Since there may sometimes be an exponential number of PIs, there are also many algorithms which compute *subsets* of these, or which compute them according to some prioritization scheme. In machine learning, (Hong 93) used a logic minimization algorithm (Hong, Cain & Ostapko 74) to induce a minimally covering set of minimal rules. Each iteration in the STAR algorithm (Michalski 83) essentially computes the PIs of all negative examples and one positive example. A version space's most general rules (Mitchell 82) correspond to

the positive kernel rules, or the PIs of the negative theory. (Ngair 92) shows that both a version space and PIs are modelable as general order-theoretic structures and are thus computable using simple lattice operations. The SE-tree-based learning framework (Rymon 93) and PI generation algorithm (Rymon 94) both support partial exploration of rules, e.g., minimal covers or maximizers of some user-specified priority.

Most of the PI generation algorithms assume that the input theory is given in either CNF or DNF. For the purpose of computing the PIs of a training set theory, a PI generation algorithm should be able to receive its input in CNF. However, as will soon be discussed, one may wish to combine these with the PIs of another theory which may be given in a different form; hence the flexibility offered by the variety of algorithms. Furthermore, certain algorithms may outperform or underperform others, depending on certain features of the underlying theory and of its PIs.

The flexibility offered by the fact that positive and negative kernel rules can be computed *separately* and then combined using a simple union-with-subsumption operator may be of practical importance when dealing with large problems. The disadvantage of this is that many PIs may later be subsumed; a similar consideration applies when combining PIs of the training set theory with those of an auxiliary theory. Some of this duplicity can be avoided in an SE-tree-based framework, as will be discussed later.

Boosting Learning with an Auxiliary Domain Theory

One major problem in *applying* machine learning is that examples are often scarce. Even where examples are seemingly abundant, their number is often minuscule relative to the syntactic size of the domain. Learning programs thus face a hard *bias selection* problem, having to decide between a large number of distinct classifiers that equally fit the training set. We propose that the PI-based approach lends itself to use of auxiliary domain knowledge, in the form of a logical theory, to leverage learning by restricting the set of hypotheses considered. Computationally, at least if an SE-tree-based algorithm is used, significant parts of the search space may be discarded without even being searched.

Consider Example 2 again. Since the universe size is $16 (2^4)$, and since only three examples were given, there are $2^{16-3} = 2^{12}$ different classifiers *consistent* with the training examples. Prime implicants belong to a somewhat stricter class, namely conjunctions which *entail* the training set theory. While each of the kernel rules is consistent with the examples, they may contradict on other objects. Indeed, in the SE-tree-based classification framework, the number of classifiers potentially embodied in a collection of kernel rules depends on the number of objects on which two or more rules contradict. In Example 2, there are 7 such objects (Figure 1a) and thus 2^7 classifiers.

Now suppose that in addition to the training examples, we are also given an auxiliary domain theory (ADT) which we will assume holds in the domain and thus consistent with the examples. It is reasonable to demand that labels assigned by a candidate classifier be consistent with this theory. Furthermore, we will insist that the classifier *entails* ADT. To achieve this, we will compute rules as PIs of the conjunction of the respective training set theory and ADT.

Theorem 6 *Rules for Examples + ADT*

Let TSET be a training set, x a class variable, and Σ^+ and Σ^- as before. Let ADT be an auxiliary domain theory such that $\text{ADT} \stackrel{\text{def}}{=} \text{ADT}^0 \cup \text{ADT}^- \cup \text{ADT}^+$ where ADT^0 does not mention x nor \bar{x} ; ADT^- is in CNF and does not mention x ; and ADT^+ is in CNF and does not mention \bar{x} . Let $\text{PI}^+ \stackrel{\text{def}}{=} \text{PI}(\Sigma^+ \cup \text{ADT}^0 \cup \text{ADT}^+)$ and $\text{PI}^- \stackrel{\text{def}}{=} \text{PI}(\Sigma^- \cup \text{ADT}^0 \cup \text{ADT}^-)$. If r is a partial description then

- (1) if $r \in (\text{PI}^- \text{ modulo subsumption with } \text{PI}^+)$ then r does not cover any negative example and does cover at least one positive example; r is minimal as such.
- (2) if $r \in (\text{PI}^+ \text{ modulo subsumption with } \text{PI}^-)$ then r does not cover any positive example and does cover at least one negative example; r is minimal as such.

Proof: We will only prove (1); the proof for (2) is analogous. If $r \in \text{PI}^-$ then r contradicts at least one assignment in each of the negative examples; thus it does not cover any negative example. If r did not cover any positive example, then $r \models \Sigma^+$ and therefore there exists $r' \in \text{PI}^+$ such that $r' \subseteq r$, in contradiction to the assumption. Q.E.D.

Note that the decomposition of ADT was not used in the proof. The theorem still holds if ADT is taken as a whole and PIs for $\Sigma^+ \cup \text{ADT}$, modulo subsumption, are taken as negative rules and vice versa for positive rules. The problem is that important rules may be lost that way. In particular, consider a situation in which Σ^- was included as part of ADT. Then, $\text{PI}(\Sigma^+ \cup \text{ADT})$ is subsumed by $\text{PI}(\Sigma^- \cup \text{ADT})$ and we lose all negative rules.

The new ADT-boosted induction algorithm will thus partition ADT as above, and then use the respective components to compute positive and negative rules. Compared to its predecessor, the new algorithm will typically result in rules with a more restricted scope. Note that some new PIs may appear which are independent of the class labeling decision, e.g., a domain rule such as “males can never be pregnant”. However, these will appear in both the positive and negative PIs and will thus be removed by subsumption.

Thanks to the diversity of PI generation algorithms, ADT^0 can be given in a variety of syntactic forms; if it is in DNF, its PIs can be computed separately using an algorithm which accepts DNF input. The PIs of the combined theories can then be computed as the PIs of the combination of the PIs of each of the respective

theories, by invoking same program again. If ADT^0 is also in a CNF then the PIs of the combined theories can be computed in a single shot. Most notably, a set of rules such as the ones typically gathered from domain experts can easily be transformed into a CNF.

Consider Example 2 once again. Suppose that in our domain it is impossible for test (attribute) b to be positive if test a is negative, i.e., $\bar{a} \rightarrow \bar{b}$. We first transform this statement to a domain theory in CNF: $\text{ADT} \stackrel{\text{def}}{=} \text{ADT}^0 \stackrel{\text{def}}{=} (a \vee \bar{b})$. Then, we compute PIs for $\Sigma^+ \cup \text{ADT}$, and similarly for $\Sigma^- \cup \text{ADT}$. After removing subsumed PIs, only two rules are left: $a \Rightarrow x$, and $\bar{a}\bar{b} \Rightarrow \bar{x}$. Figure 1b depicts a class-labeled universe according to these two rules. Notably, there are no contradictions left (although this does not hold in general). Also note that part of the syntactic universe that was covered by the previous set of rules is not covered by the new rules; according to the ADT, these object are not part of the *real* universe as it is impossible for a to be negative without b being negative as well.

ab	$a\bar{b}$	$\bar{a}b$	$\bar{a}\bar{b}$		ab	$a\bar{b}$	$\bar{a}b$	$\bar{a}\bar{b}$	
x	x	$x \bar{x}$	$x \bar{x}$	cd	x	x		\bar{x}	cd
x	x	$x \bar{x}$	$x \bar{x}$	$c\bar{d}$	x	x		\bar{x}	$c\bar{d}$
x	x	$x \bar{x}$	$x \bar{x}$	$\bar{c}d$	x	x		\bar{x}	$\bar{c}d$
x	x	$x \bar{x}$	\bar{x}	$\bar{c}\bar{d}$	x	x		\bar{x}	$\bar{c}\bar{d}$

(a) TSET only

(b) with ADT

Figure 1: Class Labelings with and without ADT

Kernel rules can be computed in various orders:

1. Compute PIs separately for each of $\Sigma^+ \cup \text{ADT}^+$, $\Sigma^- \cup \text{ADT}^-$, and ADT^0 ; then merge while subsuming supersets. In this case, $\text{PI}(\text{ADT}^0)$ is only computed once. Using the SE-tree data structure, merging is linear in the size of the trees. This may be wasteful, however, if many PIs for one theory are subsumed by another.
2. Compute PIs for the two *combined* theories directly. This may save time and space if many PIs of ADT^0 are later subsumed. However, in essence, many of the PIs of ADT^0 are computed twice.
3. If the SE-tree method is used, compute $\text{PI}(\text{ADT}^0)$, and then use the resulting SE-tree as the basis for search for the PIs of the combined theories. In expanding this tree, nodes previously pruned shall remain pruned. However, unexplored branches may have to be “re-opened”. Some of the PIs of ADT^0 may have to be further expanded.

An SE-tree-based Implementation

Set-Enumeration (SE) trees were proposed in (Rymon 92) as a simple way to systematically search a space of sets. It was suggested they can serve as a uniform model for many problems in which solutions are modeled as unordered sets.

Given a set of attributes, a *complete* SE-tree is a tree representation of all sets of attribute-value pairs. It uses an indexing on the set of attributes to do so systematically, i.e., to *uniquely* represent *all* such sets. The SE-tree’s root is always labeled with the empty set. Then, a node’s descendants are each labeled with an expansion of the parent’s set with a single attribute-value assignment. The key to systematicity is that a node is only expanded with attributes ranked higher in the appropriate indexing scheme than attributes appearing in its own label. For example, assuming alphabetic indexing, a node labeled $a\bar{b}d$ will not be expanded with c nor with \bar{c} , but only with e, f , etc. Allowed attributes are referred to as that node’s *View*. Of course, the complete SE-tree is too large to be completely explored and so an algorithm’s search will typically be restricted to its most relevant parts. A simple PI generation algorithm is outlined in (Rymon 92) as an example application of SE-trees.

In (Rymon 93), we presented an SE-tree-based *induction framework* and have argued that it generalizes decision trees in several ways. Like decision trees, an SE-tree is induced via recursive partitioning of the training data. Also like decision trees, classification requires traversing matching paths in the tree. However, an SE-tree embodies many decision trees and thus allows for explicit mediation of conflicts. While here we assume a fixed indexing, attributes in a node’s *View* can be dynamically re-ordered, e.g. by information-gain, without infringing on completeness.

An improved version of the SE-tree-based PI generation algorithm is detailed in (Rymon 94). This algorithm accepts input in CNF and works by computing minimal *hitting sets* for the collection of clauses. It is briefly presented next:

First, given a collection of sets, a hitting set is a set which “hits” (shares at least one element with) each set in the collection. Non-trivial PIs correspond to minimal hitting sets (excluding those which include both a variable and its negation.)

The algorithm works by exploring an imaginary SE-tree in a best-first fashion, where PIs are explored in an order conforming to some user-specified prioritization; thus, if time constraints are imposed, the most important PIs will be discovered. Exploration starts with the empty set. Then, iteratively, an open nodes with the highest priority is expanded with *all* possible one-attribute expansions which (a) are in that node’s *View*, and (b) hit a set not previously hit by that node. Expanded nodes which hit all sets are marked as hitting sets and the rest remain open for further expansion.

The algorithm uses two pruning rules. First, nodes subsumed by previously discovered hitting sets can be pruned; they cannot lead to *minimal* hitting sets. Second, a node is pruned if any of the sets it does not hit is completely outside its *View*; given the SE-tree structure, such a node cannot lead to a hitting set.

(Rymon 94) also suggests a recursive problem de-

composition heuristic in which the collection of sets not hit by a node is partitioned into variable-disjoint sub-collections. If such partitioning exists, the minimal hitting sets for the union are given by the product of the minimal hitting sets for the sub-collections. These are computed via recursive application of the algorithm to each of the sub-collections.

Consider Example 2 again: Σ^+ consists of the sets $\{\bar{a}, \bar{b}, \bar{c}, \bar{d}, x\}$, $\{\bar{a}, b, c, d, x\}$. Figures 2a,b depicts the SE-trees explored for computing $\text{PI}(\Sigma^+)$ and $\text{PI}(\Sigma^-)$, ignoring PIs with x or \bar{x} . Note that, in the former, a branch labeled a was never considered because a does not appear in Σ^+ and that a branch labeled d was pruned because it cannot lead to a solution. However, except for nodes labeled with d or \bar{d} , other nodes cannot be pruned for having too narrow a *View*; this is because examples assign values to *all* variables. For the same reason, decomposition is also impossible.

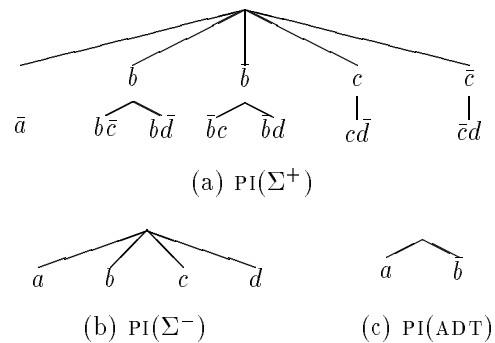


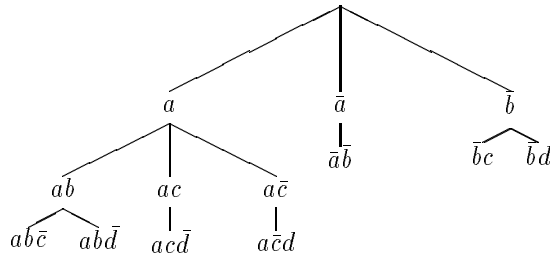
Figure 2: Original SE-trees

Consider now the ADT $\stackrel{\text{def}}{=} (a \vee \bar{b})$, as discussed before. Figure 2c shows the SE-tree for $\text{PI}(\text{ADT})$. Figures 3a,b shows SE-trees for the combined theories. Note that now, branches from the root labeled with c or \bar{c} can be pruned because they cannot lead to hitting sets for ADT. Also, once all sets in the training set theories are hit, one can take advantage of the decomposition heuristic. Notably, PIs for the combined theories are more complex; they have to hit more sets. This makes the resulting rules *less* conflicting. Figure 3c shows the kernel rules obtained by merging-with-subsumption the trees in 3a,b.

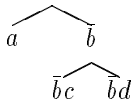
Summary

We have shown a simple correspondence between kernel rules and prime implicants which

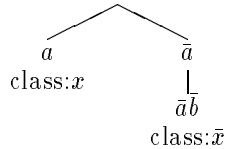
- a. Allows *computing* kernel rules using any of a number of prime implicants generation algorithms; and
- b. Leads to a PI-based learning algorithm which can be boosted with an auxiliary domain theory, e.g., a set of rules provided by a domain expert, or a functional description of a device.



(a) $PI(\Sigma^+UADT)$



(b) $PI(\Sigma^-UADT)$



(c) kernel rules

Figure 3: SE-trees for combined theories

We outline an SE-tree-based algorithm which allows exploring rules according to some user-defined preference criterion. We hope the domain theory enhancement will eventually contribute to the applicability of this machine learning approach to real-world domains. In addition, given the significant research involving prime implicants, we believe the correspondence presented here may lead to new insights as researchers reinterpret these results in the realm of machine learning.

Acknowledgement

Discussions with Dr. John Clarke have motivated this work. I also thank Ron Kohavi, Foster Provost, Bob Schrag and anonymous reviewers for important discussions and comments.

References

- de Kleer, J., Exploiting Locality in a TMS. In Proceedings *8th National Conf. on Artificial Intelligence*, pp. 254-271, Boston MA, 1990.
- de Kleer, J., Mackworth, A. K., and Reiter, R., Characterizing Diagnoses. In Proceedings *8th National Conf. on Artificial Intelligence*, pp. 324-330, Boston MA, 1990.
- de Kleer, J., An Improved Incremental Algorithm for Generating Prime Implicates. In Proceedings *10th National Conf. on Artificial Intelligence*, pp. 780-785, San Jose CA, 1992.
- Ginsberg, M., A Circumscriptive Theorem Prover, *Artificial Intelligence*, 39, pp. 209-230, 1989.
- Greiner, R., Smith, B. A., and Wilkerson R. W., A Correction to the Algorithm in Reiter's Theory of Diagnosis. *Artificial Intelligence*, 41, pp. 79-88, 1989.

Hong, S. J., Cain, R. G., and Ostapko, D. L., MINI: A Heuristic Approach for Logic Minimization. *IBM Journal of Research and Development*, pp. 443-458, 1974.

Hong, S. J., R-MINI: A Heuristic Algorithm for Generating Minimal Rules from Examples. IBM Research Reporting RC 19145, 1993.

Jackson, P., and Pais, J., Computing Prime Implicants. In Proceedings *Conf. on Automated Deduction*, pp. 543-557, 1990.

Karnaugh, G., The Map Method for Synthesis of Combinational Logic Circuits. *AIEE Trans. Communications and Electronics*, vol. 72, pp. 593-599, 1953.

Kean, A., and Tsiknis, G., An Incremental Method for Generating Prime implicants/Implicates. *Journal of Symbolic Computation*, 9:185-206, 1990.

McCluskey, E., Minimization of Boolean Functions. *Bell System Technical Journal*, 35:1417-1444, 1956.

Michalski, R., A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20, 1983, pp. 111-116

Mitchell, T. M., Generalization as Search. *Artificial Intelligence*, 18, 1982, pp. 203-226.

Ngair, T., *Convex Spaces as an Order-Theoretic Basis for Problem Solving*, Ph. D. Thesis, Computer and Information Science, Univ. of Pennsylvania, 1992.

Quine, J. O. W., The Problem of Simplifying Truth Functions. *American Math. Monthly*, 59:521-531, 1952.

Raiman, O., and de Kleer, J., A Minimality Maintenance System. In Proceedings *3rd Int'l Conf. on Principles of Knowledge Representation and Reasoning*, Cambridge MA, pp. 532-538, 1992.

Reiter, R., A Theory of Diagnosis From First Principles. *Artificial Intelligence*, 32, pp. 57-95, 1987.

Rymon, R., Search through Systematic Set Enumeration. In Proceedings *3rd Int'l Conf. on Principles of Knowledge Representation and Reasoning*, Cambridge MA, pp. 539-550, 1992.

Rymon, R., An SE-tree-based Characterization of the Induction Problem. In Proceedings *10th Int'l Conf. on Machine Learning*, pp. 268-275, Amherst MA, 1993.

Rymon, R., An SE-tree-based Prime Implicant Generation Algorithm. To appear in *Annals of Math. and A.I.*, special issue on Model-Based Diagnosis, Console & Friedrich eds., Vol. 11, 1994.

Slagle, J. R., Chang, C., and Lee R. C., A New Algorithm for Generating Prime Implicants. *IEEE Trans. on Computers*, 19(4), 1970.

Tison, P., Generalized Consensus Theory and Application to the Minimization of Boolean Functions. *IEEE Trans. on Computers*, 16(4):446-456, 1967.