Implicit Manipulation of Equivalence Classes Using Binary Decision Diagrams

Bill Lin, A. Richard Newton
University of California, Berkeley, CA 94704

Abstract

Many problems require the efficient representation and manipulation of equivalence classes. Examples of such problems range from computing communication complexity for logic decomposition to reduction of finite automata. However, the number of equivalent classes are often extremely large in practical problems. In the worst case, the number of equivalence classes for a given n dimensional space is exponential in the number of dimensions (i.e. $2^n$). In this paper, we define a new representation, called an equivalence class characterization function, that can implicitly represent equivalence classes with a compact characteristic function that will have at most n outputs. Using Binary Decision Diagrams (BDD's) and the concept of equivalence class characterization function, very large problem instances can be represented. For manipulating equivalence classes efficiently, we propose a new Boolean operator called a compatible projection operator. Conceptually, the compatible projection operator is used to uniquely select a single element from each equivalence class to “characterize” the class. In manipulating equivalence classes, the compatible projection operator is used to implicitly derive an encoding function from the equivalence relation that encodes the equivalence class information symbolically. We describe an efficient implementation based on BDD's that has been applied to very large problem instances.

1 Introduction

Recently, it has become apparent that many problems in synthesis and verification are intimately related in that they are often fundamentally dependent on the same set of basic logic manipulations. Efficient techniques developed in this area can be viewed as “core” technologies that can be applied to a wide spectrum of applications in both areas. Since many algorithms in synthesis and verification make use of the same basic set of core computations extensively, new advances in this area are extremely important. One such core technology is the Binary Decision Diagram (BDD) [3]. Although BDD’s were originally developed for symbolic simulation and verification, they have been found to be fundamental in a wide variety of applications. Another important core technology is the concept of BDD-based implicit state enumeration developed by Courter et al. [4]. The main idea is to use BDD’s to perform symbolic breadth-first execution of an implicitly defined state space. At each iteration, a large number of states is simultaneously traversed by performing symbolic image and inverse image computations. This concept can be applied to solve many problems requiring spatial or temporal analysis of some state space. These techniques are related to the concept of characteristic function.

While BDD’s, characteristic function, and implicit enumeration provide the basic machinery and concepts to manipulate functions and relations efficiently, we do not have at our disposal analogous machinery for representing and manipulating equivalence classes efficiently (other than explicit truth table enumeration). However, in many interesting applications, the ability to represent and manipulate equivalence classes is in fact the fundamental bottleneck.

Example applications where efficient representing and manipulating equivalence classes are crucial include communication complexity calculation and manipulation of sequential machines. In communication complexity calculation, the problem is to compute the amount of “real” information being transmitted between two combinational logic blocks. This calculation forms the core computation (and the most crucial bottleneck) in many problems such as functional decomposition for logic synthesis [5] and logic partitioning [1]. The problem of calculating communication complexity is equivalent to computing number of equivalence classes. Consider the case of two block partition with n signals going from one block to the other. A number of vector patterns from the first block may be considered equivalent with respect to the output behavior of the second block. These equivalent vector patterns divide the space into a number equivalent classes. Using implicit enumeration and characteristic function techniques, only pairwise equivalence relationships can be derived. However, we have no way of deriving or representing the equivalence classes efficiently. One approach is to represent each equivalence class separately with a characteristic function in BDD form. Although each characteristic function may be reasonably compact, there may be in the worst case exponential number (i.e. $2^n$) of equivalence classes. Since the number of signals is usually very large, this severely limits the analysis to only small problems. Similar problems arise in the manipulation of state machines under equivalent states.

In this paper, we present new methods based on BDD’s for representing and manipulating equivalence classes efficiently. We define the concept of equivalence class characterization function to implicitly represent equivalence classes. Informally, an equivalence class characterization function is effectively an encoding function that encodes all equivalent bit patterns into a unique bit pattern. An equivalence class characterization function has several interesting properties that makes it very useful for representing and manipulating classes. One is that the number of individual functions required to characterize all the equivalence classes implicitly is n in the worst case rather than $2^n$. This makes it possible to represent a very large number of equivalence classes implicitly since the number of functions is linear in the number of variables in the worst case. Another property of the equivalence class characterization function representation is that the number of equivalences classes can be readily determined by computing the size of the range.

To compute the equivalence class characterization function, we introduce a new Boolean operator called the compatible projection operator. Conceptually, the compatible projection operator is used to uniquely select a single element from each equivalence class to “characterize” the class. In manipulating equivalence classes, the compatible projection operator is used to implicitly derive an encoding function from the equivalence relation that encodes the equivalence class information symbolically. It can also be used as a basic mechanism for selecting a compatible map.

---

*This work was supported in part by in part by NSF and DARPA under contract number MIP-8719546.
2 Terminology and Notation

A Binary Decision Diagram (BDD) [3] is a directed acyclic graph (DAG) representation of logic where each node is associated with a variable and two fanouts. The two fanouts correspond to the cases when the variable is set to 0 and set to 1, respectively. At the leaves are two constant nodes representing the Boolean values 0 and 1. A variable ordering is imposed such that all transitive fanouts of node must have a higher index, except for constant nodes, and a variable may not be repeated in a single path. This representation is unique for a given variable ordering. Standard Boolean operators like intersection, union, and negation can be implemented directly with BDD's. Boolean quantification complexity calculation, the equivalence relation represents the set of equivalent state pairs and represents the set of equivalent vector pairs from one block to another. In the case of state machine manipulation, the equivalence relation on \( R \) induces a partition \( \pi : B^n \rightarrow B^m \) that uniquely encodes each equivalence class, we define a characterization function \( \xi : B^n \rightarrow B^m \) that satisfies the property \( \xi(y) = \xi(y') \) if and only if \( y \) and \( y' \) belong to the same equivalence class.

An equivalence class characterization function is effectively an encoding function that encodes all equivalent vertices to a unique image point in \( B^m \). This implies that two non-equivalent vertices will have different images in \( \xi \).

An equivalence class characterization function has several interesting properties that make it very useful for representing and manipulating classes.

Lemma 1 Let \( E \subseteq B^n \times B^n \) be an equivalence relation that divides the Boolean space \( B^n \) into \( p \) equivalence classes. An equivalence class characterization function \( \xi : B^n \rightarrow B^m \) for \( B^n \) can be constructed for any value \( m \geq \lceil \log p \rceil \). In the worst case, \( m \leq n \).

Proof. Because an equivalence class characterization function will map all equivalent vertices in \( B^n \) to the same image point in the output space, we only need a large enough output space to represent \( q \) points. Therefore, it can be encoded in a Boolean space with \( m \) dimensions where \( m \geq \lceil \log p \rceil \). The represents a lower bound. Since we know \( p \leq 2^n \), \( p \) equivalence classes can always be encoded with \( m \leq n \).

This result is important since we can implicitly represent \( p \) equivalence classes, \( [S_1], [S_2], \ldots, [S_p] \), with at most \( n \) functions, each representable in BDD form, rather than \( 2^n \) functions.

Lemma 2 Let \( E \subseteq B^n \times B^n \) be an equivalence relation and let \( \xi : B^n \rightarrow B^m \) be any equivalence class characterization function for \( E \). Then the size of the range, \( |\text{range}(\xi)| \), is exactly the same as the number of equivalence classes induced by \( E \).

Proof. Follows from Lemma 1.

This means that there is a one-to-one correspondence between an equivalence class of \( B^n \) and an image point in \( B^m \). Therefore, we can compute the number of equivalence classes simply by computing the size of the range. Given an equivalence relation, computing a legal equivalence class characterization function can be very difficult since the number of equivalence classes is in the worst case exponential in the number of variables. Problem sizes with over \( 10^{10} \) equivalent pairs and over \( 10^{10} \) equivalent classes are not unusual (cf. Section 6). Hence, effective mechanisms that implicitly compute this information are required. This problem is addressed next.

4 The Compatible Projection Operator

To derive an equivalence class characterization function that uniquely encodes each equivalence class, we define a new Boolean operator, called the compatible projection operator. Conceptually, the compatible projection operator, when applied to an equivalence relation \( E \), uniquely selects a single element from each equivalence class to "characterize" the class. This is performed by finding a suitable compatible function for the relation \( E \).

Informally, the compatible projection operator is defined as follows. Given a binary relation \( R \subseteq B^n \times B^n \),
the compatible projection of \( R \) is the compatible function, written in characteristic function form,
\[
F = \{(x,y) | (x,y) \in R, y = SEL(x)\}
\]
where \( SEL(x) \) is any selection function that uniquely selects a member from the equivalence class of \( x \).

We choose a criterion that uniquely orders the elements in the co-domain (output space) such that the output choice lowest in the order is always selected for each \( x \in B^n \). This can be performed by using a distance metric to determine a total ordering on the choices.

**Definition 2** Let \( B^n \) be a \( n \)-dimensional Boolean space and \( y_1 < y_2 < \ldots < y_m \) be an ordering of its variables. The distance between two vertices \( x \in B^n \) and \( y \in B^n \) is defined as follows:
\[
d(a, b) = \sum_{i=1}^{n} |a_i - b_i| \cdot 2^{-a_i}
\]

Using the distance operator, we can define an ordering on the vertices of a Boolean space relative to some reference vertex. In fact, the relative ordering between subsets is also well defined. Using the distance metric, we can define a selection operator that can be used to select a unique member from a set.

**Definition 3** Given \( a \in B^n \), \( \Sigma \subseteq B^n \), and a variable ordering \( y_1 < y_2 < \ldots < y_m \) on \( B^n \), the closest interpretation of \( a \) in \( \Sigma \) is defined as follows:
\[
\exists (a, \Sigma) = \arg \min_{\sigma \in \Sigma} d(a, \sigma)
\]
The definition of closest interpretation is unique for a given variable ordering. We can now formally define the compatible projection operator.

**Definition 4** Let \( R \subseteq B^n \times B^n \) be a binary relation and \( a \in B^n \) be a reference vertex. The compatible projection, or simply projection, of \( R \) relative to \( a \), denoted as \( cprojection(R, a) \), is the compatible function \( F \) defined as follows:
\[
F = \{(x,y) | (x,y) \in R, y = \exists (a, R(x))\}
\]

**Lemma 3** Given \( R \subseteq B^n \times B^n \), \( a \in B^n \), and a variable ordering \( y_1 < y_2 < \ldots < y_m \) on \( B^n \), the compatible mapping \( F = cprojection(R, a) \) is canonical.

**Proof.** Given a reference vertex \( a \) and a variable ordering on the \( y_i \) variables, the \( \exists (a, R(x)) \) operator uniquely selects a \( y \) mapping from \( R(x) \) for every \( x \).

This result says that a unique mapping can always be derived. If we consider the special case of an equivalence relation \( E \), then we can show that the new function \( F \) produced by the compatible projection operator is indeed a legal equivalence class characterization function for \( E \).

**Theorem 4** Let \( E \subseteq B^n \times B^n \) be an equivalence relation on \( B^n \). Then \( F = cprojection(E, a) \) with respect to any reference vertex \( a \) is a legal equivalence class characterization function for \( E \).

**Proof.** Let \( \pi(B^n) = \{S_1, S_2, \ldots, S_s\} \) be the partition of equivalence classes induced by the equivalence relation \( E \), and let \( F = cprojection(E, a) \) be a compatible projection of \( E \). Then we must show that for every \( \forall S_i \in \pi(B^n), \forall x, x' \in S_i, F(x) = F(x') \). The \( \exists \) operator guarantees that the mapping with the lowest cost, according to the distance metric relative to \( a \), will be chosen. This implies that if \( x \) and \( x' \) belong to the same equivalence class, then the lowest cost choice in both cases will be the same. Since the lowest cost mapping will be selected in both cases, they will have the encoding in \( F \).

We now give an efficient recursive algorithm for computing the compatible projection operation that directly exploits recently developed techniques for efficient BDD implementation such as the strong canonical form, caching, and the ITE operator \([2]\). The proposed recursive algorithm traverses directly on the BDD graph structure and uses caching techniques to remember intermediate results, which enables the algorithm to be performed in a bottom-up fashion on the BDD. This is based on the following theorem.

**Theorem 5** Let \( y_1 = \text{top}_{*}(Y) \), \( a_i = \text{literal}_{of}(y_i) \), \( \gamma = (3Y)\alpha_{i} \). Then
\[
cprojection(r, Y) = a_i \cdot cprojection(r_{\alpha_{i} Y_{next}}) + (1 - \gamma \cdot cprojection(r_{\gamma Y_{next}}))
\]
Here, \( Y \) is the set of projection variables, and \( Y_{next} \) is the set of remaining projection variables excluding \( y_i \). The pseudo code for the recursive procedure is shown in Figure 1. In the actual implementation, the procedure begins at the root of the BDD. It then recursively computes the projection of its left branch and its right branch. A cache is used to remember previously computed results for \( \gamma = (3Y)\alpha_{i} \) at each BDD node. Caching is also used to remember intermediate cprojection results so that previous computation is not repeated.

**5 Example Applications**

**5.1 Communication Complexity**

In this section, we consider the problem calculating communication complexity using the concept of equivalence class characterization function and the compatible projection operator. Consider the example shown in Figure 2. Here, we have a hierarchically defined combinational network implemented in two separate combinational blocks \( N_1 \) and \( N_2 \). Consider the set of signals, \( Y = \{y_1, y_2, \ldots, y_n\} \), going from \( N_1 \) to \( N_2 \). From the point of view of the block \( N_2 \), two vertices (bit patterns) \( y \in B^n \) and \( y' \in B^n \) over the variables \( y_1 y_2 \ldots y_m \) are deemed equivalent at the primary outputs \( Z = \{z_1, z_2, \ldots, z_k\} \) if \( Z(y) = Z(y') \). This is denoted as \( y \sim y' \). This equivalence relation divides the Boolean space \( B^n \) into \( p \leq 2^n \) equivalence classes \( \{S_1, S_2, \ldots, S_p\} \). In the worst case, \( p \) is exponential in \( n \) (i.e. \( p = 2^n \)). Using BDD’s and the concept of characteristic function, we can very efficiently compute an equivalence
in the previous sections, an exact algorithm can be devised when a state transition graph model can be extracted, but these algorithms are only feasible for machines with at most a few hundred states. Using the machinery developed applying the techniques to the problems of communication and state minimization of large sequential circuits. The capabilities are demonstrated by the experimental results presented were measured on a DEC 5000 workstation and the CPU times presented are quoted in seconds.

For communication complexity analysis, several large MCNC logic benchmarks were selected to expose the need for representing and manipulating equivalence classes efficiently. For each benchmark, a cut through the network was found starting at the gate-level description. The set of all equivalent pairs of vector patterns along the cut was computed using Equation 2. This equivalence relation is

\[
E \subseteq B^n \times B^n \text{ that represents the set of all possible pairs of equivalent states at } n \text{ levels that cannot be distinguished by the functions } E = \{ x_1, x_2, \ldots, x_n \}. \text{ That is, each pair of vertices (} y, y' \text{) } \in E \text{ is interpreted to mean } y \text{ and } y' \text{ are equivalent. The expression for computing the equivalence relation is}
\]

\[
E(Y, Y') = \bigvee_I \bigwedge_{i=1}^n (y_i(I, Y) \oplus z_i(I, X')).
\]

Here, \( Y' = \{ y_1', y_2', \ldots, y_n' \} \) is a set of duplicated variables corresponding to the variables in \( Y \). Using the compatible projection operator, we can readily compute an equivalence class characterization function \( \xi \) using any reference vertex \( \alpha \in B^n \) as follows:

\[
\xi = cprojection(E, \alpha)
\]

The communication complexity from state 1 to state 2 is simply the size of the range of \( \xi \), which can be computed as follows:

\[
| Q = \{ y_1, \ldots, y_n \} | \xi \} \}
\]

where \( y_1, \ldots, y_n \) denotes the set of variables corresponding to the first component of \( \xi \).

5.2 Reduction of Finite Automata

In this section, we examine the use of symbolic class manipulation on the problem of minimizing state machines. In practice, large sequential circuits may not be state minimal. This is especially true when the sequential circuits are automatically compiled from high-level descriptions. There are well known algorithms for state minimization when a state transition graph model can be extracted, but these algorithms are only feasible for circuits with at most a few hundred states. Using the machinery developed in the previous sections, an exact algorithm can be devised that can feasibly state minimize finite state machines with over \( 10^{10} \) states. The outline of the minimization algorithm is as follows:

1. Given a finite state machine \( M \), first compute the equivalence relation \( E \subseteq B^n \times B^n \) corresponding to the set of all equivalent state pairs. Two states \( q_i, q_j \) are equivalent if and only if \((q_i, q_j) \in E \). The relation \( E \) is computed by first building a product machine \( M^* = M_1 \otimes M_2 \). The set of equivalent state pairs \( E \subseteq B^n \times B^n \) can then be efficiently computed using the BDD-based symbolic traversal algorithms described in [6].

2. The relation \( E \) is a one-to-many mapping that maps each state \( q \in B^n \) to any one of its equivalent state \( \{ q \} \). We use the compatible projection operator to find a characterization function \( \xi \subseteq B^n \rightarrow B^n \), in relation form, that maps all equivalent states to the same state. Modify the transition relation using the characterization function \( \xi \):

\[
\xi = cprojection(E, q_0)
\]

By default, the reset state \( q_0 \) is used as the reference vertex. This guarantees the reset state code is retained (i.e. all states equivalent to the reset state will be re-assigned the reset state code). However, this is not a necessary condition. The reduced set of states can be computed as follows:

\[
Q = \{ w_1, \ldots, w_n \} \xi
\]

where \( w_1, \ldots, w_n \) denotes the set of variables corresponding to the first component of \( \xi \). This corresponds effectively to the range of \( \xi \). The number of elements in the set \( Q \), \( | Q | \), corresponds to the number of reduced states. Using \( \xi \), the transition relation of the finite state machine can be simplified as follows:

**Definition 5** The next state transition relation for a finite state machine \( M \) is a \( T \subseteq B^n \times B^n \) such that \((i, x) \in T \) if and only if the state \( y \) can be reached in exactly one state transition from state \( x \) when input \( i \) is applied.

The transition relation implicitly defines the finite state machine. Here, \( i \) denotes the primary input variables, \( x \) the present state variables, and \( y \) the next state variables. We can then compute the state minimized transition relation as follows:

\[
T_{\min}(i, x, y) = (3z'x')[(T(i, x', y') \cdot \xi(x', x) + \xi(y', y)]
\]

In the above equation, we explicitly stated the variable names in the parentheses for each relation to illustrate the correspondence. New invalid states can be used to further simplify the minimized transition relation.

6 Experimental Results

We present experimental results to show the viability of the proposed techniques for implicit manipulation of equivalence classes. The capabilities are demonstrated by the applying the techniques to the problems of communication complexity analysis for large combinational circuits and exact state minimization of large sequential circuits. The algorithms make use of the concept of equivalence class characterization function and the compatible projection operator described in this paper, which we have efficiently implemented using BDD's. All experimental results presented were measured on a DEC 5000 workstation and the CPU times presented are quoted in seconds.
of signals along the cut, is the amount of time required to compute the equivalence quite large (over then derived implicitly using the compatible projection operation. The transition functions of the reduced machine were accordingly constructed. Tables 3 and 4 show the results of the above experiment. The total number of states for each example is indicated under the column labeled states. We quote here the sum of reachable and unreachable states because both are considered when deciding state equivalence. This is more general, but equivalent states analysis for the reachable subset is a trivial extension. The number of equivalent state pairs computed for each example is indicated under the column labeled pairs. For the example tkey, there were over \(10^{10}\) equivalent state pairs. The times for computing the equivalence relation using the algorithms described in [6,7] are indicated in the column labeled time.

Table 1: Computing and representing equivalent pairs.

<table>
<thead>
<tr>
<th>circuit</th>
<th>vars</th>
<th>elem.</th>
<th>pairs</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ampxd1</td>
<td>31</td>
<td>2.15e+9</td>
<td>5.52e+13</td>
<td>52</td>
<td>1.84</td>
</tr>
<tr>
<td>apx6</td>
<td>68</td>
<td>2.95e+20</td>
<td>7.76e+32</td>
<td>439</td>
<td>7.25</td>
</tr>
<tr>
<td>dfigrclb</td>
<td>54</td>
<td>1.80e+16</td>
<td>1.76e+29</td>
<td>3150</td>
<td>5.76</td>
</tr>
<tr>
<td>fconclub</td>
<td>31</td>
<td>2.15e+9</td>
<td>1.65e+17</td>
<td>395</td>
<td>1.79</td>
</tr>
<tr>
<td>k2</td>
<td>23</td>
<td>8.30e+6</td>
<td>1.57e+11</td>
<td>158</td>
<td>31.61</td>
</tr>
<tr>
<td>kctlcb</td>
<td>41</td>
<td>2.20e+12</td>
<td>1.44e+20</td>
<td>230</td>
<td>3.22</td>
</tr>
</tbody>
</table>

Table 2: Computing the communication complexity.

<table>
<thead>
<tr>
<th>circuit</th>
<th>vars</th>
<th>elem.</th>
<th>pairs</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>s208</td>
<td>8</td>
<td>250</td>
<td>3310</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>s298</td>
<td>14</td>
<td>16384</td>
<td>510000</td>
<td>35.88</td>
<td></td>
</tr>
<tr>
<td>tlc</td>
<td>10</td>
<td>1024</td>
<td>25400</td>
<td>5.41</td>
<td></td>
</tr>
<tr>
<td>vit3</td>
<td>9</td>
<td>512</td>
<td>18400</td>
<td>5.08</td>
<td></td>
</tr>
<tr>
<td>viterbi</td>
<td>12</td>
<td>4100</td>
<td>10700</td>
<td>17.50</td>
<td></td>
</tr>
<tr>
<td>tkey</td>
<td>228</td>
<td>4.31e+68</td>
<td>1.06e+124</td>
<td>177.83</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>228</td>
<td>4.31e+68</td>
<td>4.31e+68</td>
<td>517.85</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Computation of equivalent state pairs.

<table>
<thead>
<tr>
<th>circuit</th>
<th>vars</th>
<th>elem.</th>
<th>pairs</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>s208</td>
<td>255</td>
<td>40</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s298</td>
<td>16384</td>
<td>8060</td>
<td>1.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tlc</td>
<td>1024</td>
<td>254</td>
<td>0.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vit3</td>
<td>512</td>
<td>15</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>viterbi</td>
<td>4100</td>
<td>3120</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tkey</td>
<td>4.31e+68</td>
<td>1.76e+13</td>
<td>2.20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>4.31e+68</td>
<td>4.31e+68</td>
<td>2.48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: State minimization results.

represented using a BDD. In Table 1, vars is the number of signals along the cut, elem. is the number of possible vector patterns, and pairs is the number of equivalent pairs. In some examples, the number of equivalent pairs is quite large (over \(10^{10}\)). size is the number of nodes in the BDD representation for the equivalence relation, and time is the amount of time required to compute the equivalence relation. An equivalence class characterization function is then derived implicitly using the compatible projection operator. The number of states for each example is indicated under the column labeled states. This is also the communication complexity along the cut. In Table 2, size is the number of nodes in the BDD representation for the characterization function and time is the amount of time required to perform the compatible projection. As can be seen, the compatible projection operation is extremely fast. The transition functions of the reduced machine were accordingly constructed. Tables 3 and 4 show the results of the above experiment. The total number of states for each example is indicated under the column labeled states. We quote here the sum of reachable and unreachable states because both are considered when deciding state equivalence. This is more general, but equivalent states analysis for the reachable subset is a trivial extension. The number of equivalent state pairs computed for each example is indicated under the column labeled pairs. For the example tkey, there were over \(10^{10}\) equivalent state pairs. The times for computing the equivalence relation using the algorithms described in [6,7] are indicated in the column labeled time.

7 Conclusions

We have provided new methods for representing and manipulating equivalence classes efficiently, namely a new representation called an equivalence class characterization function and a new operator called compatible projection. We have shown the power of these methods on several crucial problems. There are clearly many other applications of implicit class manipulation, and we believe that the capabilities have just begun to be investigated.

References