Facts and Fallacies in Using Genetic Algorithms for Learning Clauses in First-Order Logic

Flaviu Adrian Mărginean

Department of Computer Science, The University of York
Heslington, York YO10 5DD, United Kingdom
flav@cs.york.ac.uk

Abstract. Over the last few years, a few approaches have been proposed aiming to combine genetic and evolutionary computation (GECCO) with inductive logic programming (ILP). The underlying rationale is that evolutionary algorithms, such as genetic algorithms, might mitigate the combinatorial explosions generated by the inductive learning of rich representations, such as those used in first-order logic. Particularly, the binary representation approach presented by Tamaddoni-Nezhad and Muggleton has attracted the attention of both the GECCO and ILP communities in recent years. Unfortunately, a series of systematic and fundamental theoretical errors renders their framework moot. This paper critically examines the fallacious claims in the mentioned approach. It is shown that, far from restoring completeness to the learner PROGOL's search of the subsumption lattice, the binary representation approach is both overwhelmingly unsound and severely incomplete.

1 Introduction

Over the last few years there has been a surge of interest in combining the expressiveness afforded by first-order logic representations in inductive logic programming with the robustness of evolutionary search algorithms [7,20,21,22]. It is hoped that such hybrid systems would retain the powerful logic programming formalism and its well-understood theoretical foundations, while bringing in to the search the versatility of evolutionary algorithms, their inherent parallelism, and their adaptive characteristics [21].

PROGOL is a first-order inductive reasoner widely regarded as state of the art. Owing to its relative importance, its soundness and completeness have been the object of numerous theoretical studies [3,8,9,10,14,24]. PROGOL has also been investigated from the point of view of its tractability [16,17]. Search in first-order logic is notoriously difficult because of the expressive power of the hypotheses that generates combinatorial explosions.

Owing to these two issues, (in)completeness and (in)tractability, the announcement by Tamaddoni-Nezhad and Muggleton [20,21,22] that genetic algorithms can solve both problems via a simple binary change of representation has attracted interest. In this paper we demonstrate that, unfortunately, such hopes are unfounded. We review Tamaddoni-Nezhad and Muggleton's aforementioned approach and show that it is provably flawed at every level. Specifically, we consider the following claims by the authors, which are central to their approach:

Fallacy 1. The proposed binary representation for clauses is novel. It encodes the subsumption lattice lower bounded by the bottom clause in a compact and complete way.
Fallacy 2. A fast evaluation mechanism for clauses has been given.
Fallacy 3. The proposed task-specific genetic operators can be viewed as a form of refinement, titled genetic refinement [20] or stochastic refinement [21].

Respectively, we show:

Fact 1. The proposed binary representations have been known for some time [1,2] and shown to be incomplete for subsumption even for function-free languages. The binary encoding of the subsumption lattice is both incomplete and unsound. Owing to unsoundness, for even a single shared variable in the bottom clause, the proposed space of binary representations is $\frac{2^{\binom{n}{2}}}{B_n}$ times bigger than the number of valid clauses that it manages to encode, where $n \overset{\text{def}}{=}$ the number of predicates that share the variable and B_n is the n-th Bell Number. Therefore the space of binary representations is not compact. An infinity of good clauses are left out (the encoding is *incomplete*) and a huge number of spurious binary strings get in (the encoding is *unsound* and *noncompact*).
Fact 2. The proposed evaluation mechanism for clauses is provably unsound.
Fact 3. The proposed task-specific genetic operators are not refinement operators because of their being provably unsound and incomplete.

The errors that we pinpoint in this paper appear to have no easy fix. They are very fundamental theoretical errors, which undermine the whole binary representation approach. We wish to emphasise that the problem of combining evolutionary computation with first-order logic learning is worth investigating, and in this respect the binary representation attempt is meritorious. However, the flaws need to be corrected. The paper is organised as follows. In Section 2 we review some preliminaries, such as inductive logic programming and inverse entailment. In Section 3 we expose the fallacies undermining the binary representation approach in correlation with our counter-arguments, while Sections 4 and 5 present the conclusions of this paper.

2 Preliminaries

The reader is assumed to be familiar with the basic formalism of first-order clausal logic. The paragraphs below are intended as brief reminders. A good general reference for inductive logic programming is [15] and Muggleton's seminal paper on PROGOL is [13].

2.1 Inductive Logic Programming (ILP)

Under ILP's normal setting (also called the *monotonic* setting, or the *learning-from-entailment* setting) one considers background knowledge B, positive and negative examples E^+ and E^-, and a hypothesis H. B, E^+, E^- and H are all finite sets of clauses (*theories*). Further restrictions can be applied, for instance by requiring that B, H, E^+, E^- are logic programs, rather than general theories, or imposing that positive examples are ground unit definite clauses and negative examples are ground unit headless Horn clauses. The central goal is to induce (*learn*) H such that the following two conditions are satisfied:

$$\begin{cases} B \wedge H \models E^+ \\ B \wedge H \wedge E^- \not\models \Box \end{cases}$$

This looks rather much like solving a system of inequations (logical entailment $\models$ is a quasi-order, i.e. reflexive and transitive), except that it is a rather complicated one. However, if one only considers the positive examples, in the first instance, the following simpler system is obtained:

$$\begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \Box \end{cases}$$

Progress has been made towards characterising the solutions to this system as follows:

2.2 Inverse Entailment

Definition 1 (Subsumption). *Let C and D be clauses. Then C θ-subsumes D, denoted by $C \succeq D$, if there exists a substitution θ such that $C\theta \subseteq D$ (i.e. every literal in $C\theta$ is also a literal in D).*

Definition 2 (Inverse Entailment). *Inverse Entailment is a generic name for any computational procedure that, given B, E^+ as input, will return a bottom clausal theory $\bot(E^+, B)$ as output, such that the following condition is satisfied:*

$$H \models \bot(E^+, B) \iff \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \Box \end{cases}, \ \forall H$$

Inoue (2001) has provided the only known example of Inverse Entailment in the general case (under the name of Consequence-Finding). It was hoped that entailment on the left-hand side might be replaced with subsumption or another decidable quasi-order, as entailment is only semidecidable in the general case. However, this hope was largely unfulfilled. In more restricted settings, the following results have been obtained:

For H, E restricted to be clauses rather than theories, Yamamoto (1997) gives a computational procedure that computes a bottom clause $\bot(E^+, B)$ such that the following condition is satisfied:

$$H \succeq \bot(E^+, B) \iff \begin{cases} H \succeq_B E^+ \\ H \not\succeq_B \Box \end{cases}, \ \forall H$$

Note that entailment has been replaced with the more restricted subsumption on the left-hand side and Plotkin's relative subsumption on the right-hand side.

For H, E restricted to be function-free clauses and B a function-free Horn theory, Muggleton (1998) gives a computational procedure that computes a bottom clause $\bot(E^+, B)$ such that the following condition is satisfied:

$$H \succeq \bot(E^+, B) \overset{\not\Longrightarrow}{\Longleftarrow} \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \Box \end{cases}, \; \forall H$$

Note that entailment has been replaced with the more restricted subsumption on the left-hand side but the soundness of Inverse Entailment ($\Longrightarrow$) has been lost.

For H, E restricted to be function-free Horn clauses and B a function-free Horn theory, Muggleton (1995) gives a computational procedure that computes a bottom Horn clause $\bot(E^+, B)$ such that the following condition is satisfied:

$$H \succeq \bot(E^+, B) \overset{\not\Longleftarrow}{\Longrightarrow} \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \Box \end{cases}, \; \forall H$$

Note that entailment has been replaced with the more restricted subsumption on the left-hand side but the completeness of Inverse Entailment ($\Longleftarrow$) has this time been lost. Completeness can be restored to this version if either entailment is restored on the left-hand side or the unique bottom clause is replaced with a family $\{\bot_i(E^+, B)\}_i$ of bottom clauses (subsaturants of the unique bottom clause computed by Muggleton's procedure):

$$\bigvee_i \left[H \succeq \bot_i(E^+, B)\right] \Longleftrightarrow \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \Box \end{cases}, \; \forall H$$

Subsumption is, in general, preferred to entailment on the left-hand side since it is decidable. However, as apparent from before, it only guarantees completeness and soundness of Inverse Entailment when the general ILP setting is restricted and multiple bottom clauses are generated.

3 The Binary Representation Approach: Facts and Fallacies

In the context of the Inverse Entailment procedure discussed in the preceding section, Tamaddoni-Nezhad and Muggleton consider the case where one has a function-free bottom Horn clause $\bot(E^+, B)$ and claim that the space of solutions $\{H \mid H \succeq \bot(E^+, B),\ H$ is a function–free Horn clause$\}$ can be described as a boolean lattice obtained from the variable sharing in the bottom clause according to a simple procedure (Fig. 1).

3.1 Fallacy 1 — Fact 1

We first give a description of the proposed binary representation. In [20,21] the following definition is given:

$$\begin{tikzcd}
& \begin{array}{c} p(U,V) \longleftarrow q(W,X), r(Y,Z) \\ \text{(Binary Encoding: 000)} \end{array} \arrow[dl, "\{W/U\}"'] \arrow[d, "\{Z/V\}"] \arrow[dr, "\{Y/X\}"] & \\
\begin{array}{c} p(U,V) \longleftarrow q(U,X), r(Y,Z) \\ \text{(Binary Encoding: 100)} \end{array} \arrow[d, "\{Z/V\}"'] \arrow[dr, "\{Y/X\}"] & \begin{array}{c} p(U,V) \longleftarrow q(W,X), r(Y,V) \\ \text{(Binary Encoding: 010)} \end{array} \arrow[dl, "\{W/U\}"] \arrow[dr, "\{Y/X\}"] & \begin{array}{c} p(U,V) \longleftarrow q(W,X), r(X,Z) \\ \text{(Binary Encoding: 001)} \end{array} \arrow[dl, "\{W/U\}"'] \arrow[d, "\{Z/V\}"] \\
\begin{array}{c} p(U,V) \longleftarrow q(U,X), r(Y,V) \\ \text{(Binary Encoding: 110)} \end{array} \arrow[dr, "\{Y/X\}"] & \begin{array}{c} p(U,V) \longleftarrow q(U,X), r(X,Z) \\ \text{(Binary Encoding: 101)} \end{array} \arrow[d, "\{Z/V\}"] & \begin{array}{c} p(U,V) \longleftarrow q(W,X), r(X,V) \\ \text{(Binary Encoding: 011)} \end{array} \arrow[dl, "\{W/U\}"'] \\
& \begin{array}{c} p(U,V) \longleftarrow q(U,X), r(X,V) \\ \text{(Binary Encoding: 111)} \end{array} &
\end{tikzcd}$$

Fig. 1. Tamaddoni-Nezhad and Muggleton's "subsumption lattice" bounded below by the bottom clause $p(U,V) \longleftarrow q(U,X), r(X,V)$

Definition 3 (Binding Matrix). *Suppose B and C are both clauses and there exists a variable substitution θ such that $C\theta = B$. Let C have n variable occurrences representing variables $\langle v_1, v_2, \ldots, v_n \rangle$. The binding matrix of C is an $n \times n$ matrix M in which m_{ij} is 1 if there exist variables v_i, v_j and u such that v_i/u and v_j/u are in θ and m_{ij} is 0 otherwise. We write $M(v_i, v_j) = 1$ if $m_{ij} = 1$ and $M(v_i, v_j) = 0$ if $m_{ij} = 0$.*

This definition is unsound because the binding matrix of C is defined with respect to an arbitrary clause B. It is obvious that such a binding matrix may not be unique. We therefore assume that the authors meant B to be a fixed bottom clause and the binding matrix of C was defined with respect to this fixed bottom clause. Let us consider the bottom clause $p(U,V) \longleftarrow q(U,X), r(X,V)$ in Fig. 1. Using the equality predicate we can re-write the clause as follows:

$$p(X_1, X_2) \longleftarrow q(X_3, X_4), q(X_5, X_6), X_1 = X_3, X_2 = X_6, X_4 = X_5$$

We note that the variable sharing in the bottom clause is now completely described by the three equalities. Any other clause in Fig. 1 can be re-written as a combination of the common factor $p(X_1, X_2) \longleftarrow q(X_3, X_4), q(X_5, X_6)$ and a subset of the three equalities $\{X_1 = X_3, X_2 = X_6, X_4 = X_5\}$ that describe the variable sharing in the bottom clause. For instance, clause $p(U,V) \longleftarrow q(U,X), r(Y,Z)$ will become:

$$p(X_1, X_2) \longleftarrow q(X_3, X_4), q(X_5, X_6), X_1 = X_3$$

It is now clear that we do not need the common factor, every clause in Fig. 1 being describable by simply noting which of the three equalities in the bottom clause it sets off. If we use the binary string $(1, 1, 1)$ to indicate that the bottom clause satisfies all three equalities, then the second clause above may be encoded as $(1, 0, 0)$. This is the binary representation approach, an instantiation of a technique more commonly known as propositionalisation. This approach was first investigated rigorously within the European ILP2 project supported by ESPRIT Framework IV, which ended in 1999. The deliverables of the ILP2 project [1,2] showed clearly that the approach could not yield completeness for subsumption, not even in the simple case of function-free (Datalog) languages [1]. We now show that this is indeed the case for subsumption lattices bounded below by bottom clauses.

Binary Representation Space is Incomplete. The following clauses are missing from Tamaddoni-Nezhad and Muggleton's subsumption lattice, as pictured in Fig. 1:

$\longleftarrow$, the empty clause, subset of the bottom clause
$p(U, V) \longleftarrow$, subset of the bottom clause
$q(U, X) \longleftarrow$, subset of the bottom clause
$r(X, V) \longleftarrow$, subset of the bottom clause
$p(U, V) \longleftarrow q(W, X)$, maps into the bottom clause by substitution $\theta = \{W/U\}$
$p(U, V) \longleftarrow r(X, Z)$, maps into the bottom clause by substitution $\theta = \{Z/V\}$
$\longleftarrow q(U, X), r(Y, V)$, maps into the bottom clause by substitution $\theta = \{Y/X\}$

These clauses, together with the ones in Fig. 1, are the ones that *weakly* subsume the bottom clause[1], i.e. those that map literals injectively to the bottom clause. In addition, an infinity of other clauses that subsume $p(U, V) \longleftarrow q(U, X), r(X, V)$ are also missing, for instance: $p(U, V) \longleftarrow \{q(U, X_i), r(X_j, V) \mid i \neq j,\ 1 \leq i, j \leq n\}$ for $n \geq 2$, which maps onto the bottom clause by substitution $\{X_i/X\}_{1 \leq i \leq n}$. We may wonder whether completeness has instead been achieved under subsumption equivalence, i.e. one clause from each equivalence class under subsumption is present in the search space. However, this is not the case: neither of the exemplified missing clauses are subsume-equivalent with any of the clauses already in the search space, nor are they subsume-equivalent between themselves.

The quasi-order used in Fig. 1 is therefore not subsumption but the much weaker atomic subsumption $\succeq_a$ as defined in [15, p. 244]. If, as before, we denote entailment by $\models$, subsumption by $\succeq$, weak subsumption by $\succeq_w$ and the atomic subsumption by $\succeq_a$, we have the following relationship between the four orders:

$$\succeq_a \overset{\not\Leftarrow}{\Longrightarrow} \succeq_w \overset{\not\Leftarrow}{\Longrightarrow} \succeq \overset{\not\Leftarrow}{\Longrightarrow} \models$$

Consequently, we have:

[1] See [3] for a complete and sound encoding of the weak subsumption lattice with a refinement operator.

$$H_{\succeq_a} \subsetneq H_{\succeq_w} \subsetneq H_{\succeq} \subsetneq H_{\models}$$

where

$$H_{\succeq_i} \stackrel{\text{def}}{=} \{H \mid H \succeq_i \bot(E^+, B), H \text{ is a function–free Horn clause}\}$$

for $\succeq_i \in \{\succeq_a, \succeq_w, \succeq, \models\}$. PROGOL's existing refinement encodes $H_{\succeq_w}$, which is incomplete with respect to subsumption: $H_{\succeq_w} \subsetneq H_{\succeq}$. However, the binary representation approach only captures $H_{\succeq_a}$, which is even more incomplete: $H_{\succeq_a} \subsetneq H_{\succeq_w}$.

Binary Representation Space is Unsound. As mentioned before, the incompleteness of propositionalisation for subsumption is known. Now we show that the binary representation is also unsound, i.e. a counter-example may be given wherein the binary representation codifies binary strings that do not correspond to any clause in the subsumption lattice. The bottom clause chosen in Fig. 1 is well-behaved, in the sense that all three equalities involve distinct variables. This was the original example given in [21]. Let us consider, however, the following bottom clause:

$$p(U,U) \longleftarrow q(U,X), r(Y,Z)$$

Using equality to re-write the clause we arrive at:

$$p(X_1,X_2) \longleftarrow q(X_3,X_4), q(X_5,X_6), X_1 = X_2, X_1 = X_3, X_2 = X_3$$

If we describe this bottom clause by $(1,1,1)$ as the binary representation approach prescribes, then $(1,1,0)$, $(1,0,1)$ and $(0,1,1)$ will correspond to no clause. Because of the transitivity of equality, when one has $X_1 = X_2, X_1 = X_3$ for instance, one will also have $X_2 = X_3$. In other words, a certain number of binary strings will be spurious. Interestingly, in [21] definitions 3, 4 and 5 capture precisely the space of binary strings that are mapped to valid clauses, i.e. those corresponding to normalised binding matrices or, in the language of this paper, those subsets of binary equalities that are closed under the transitivity of equality. However, it appears from [20,21] that the authors' encoding of the search space used in implementations is not the set of normalised binding matrices or corresponding normalised strings but the unsound space of all boolean binary strings. Now we show that the number of such spurious strings can grow wildly compared to the number of valid clauses encodable by this approach.

Binary Representation Space is Not Compact. Let us consider a bottom clause that has n predicates, all sharing the first variable. For simplicity we assume that all the other variables are distinct:

$$p_0(X,\dots) \longleftarrow p_1(X,\dots), p_2(X,\dots), \dots, p_{n-1}(X,\dots)$$

Using equality to re-write the clause we arrive at:

$$p_0(X_0,\dots) \longleftarrow p_1(X_1,\dots), \dots, p_{n-1}(X_{n-1},\dots), X_0 = X_1 \cdots = X_{n-1}$$

Note that we have slightly abused the notation in order to write the clause more compactly. The number of binary equalities $X_0 = X_1, X_0 = X_2, \dots$ etc. is now $\binom{n}{2}$, which will yield $2^{\frac{n(n-1)}{2}}$ binary strings upon encoding. On the other hand, the number of clauses that can be obtained by anti-unification of variables (as we have seen, Tamaddoni-Nezhad and Muggleton do not consider adding/dropping literals, thereby generating incompleteness) is given by the n-th Bell number, i.e. the number of all partitions of the set of variables $\{X_0, \dots, X_{n-1}\}$. The space of encodings will therefore be $\frac{2^{\binom{n}{2}}}{B_n}$ times bigger than the number of valid clauses that it encodes. To get a feeling for this difference, for $n = 12$ the space of encodings will contain

$$2^{66} = 73786976294838206464$$

binary strings, while the space of clauses will contain $B_{12} = 4213597$ valid clauses, i.e. about 1 good encoding for every 17.5 trillion spurious ones. In ILP it is not uncommon for bottom clauses to contain tens or hundreds of predicates with complex variable sharing. It can be shown that, as n grows, this gets worse and worse, i.e. the asymptotic behaviour confirms this tendency: $\frac{2^{\binom{n}{2}}}{B_n} \overset{n\to\infty}{\longrightarrow} \infty$.

Because of the sheer number of spurious binary encodings that parasitise the representation space even for low values of n, the search domain may become unsamplable. Furthermore, the authors observe in footnote 4 of [21, p. 642] and paragraph following Example 2 in [20, p. 249] that certain operations on binding matrices may result in matrices that are not normalised, this leading to unsoundness ('inconsistency' in their language). They say that this does not affect the genetic search in practice and could be avoided by normalisation closure. However, this is not the case. We have shown above that unsoundness can cause serious computational problems. Normalisation closure, on the other hand entails itself computational difficulties: in order to be able to compute normalisation closures, one has to revert to variables or keep a separate list of normalisation rules, valid for the bottom clause at hand. For instance, to infer $X_2 = X_3$ from $X_1 = X_2$ and $X_1 = X_3$ one has to encode the variables. Otherwise, if $X_1 = X_2, X_1 = X_3, X_2 = X_3$ are encoded as 3-bit binary strings (Y_1, Y_2, Y_3), then one needs to know that $Y_1 = 1 \ \wedge \ Y_2 = 1 \implies Y_3 = 1$ etc. However, the real problem is that, with the proposed encoding, normalisation closures are syntactically computable but meaningless from a semantic point of view. Suppose that, as in our example before, we get by random sampling (seeding) the string $(1, 1, 0)$. The 0 bit in the string does not mean that we do not know that $X_2 = X_3$. It means that we know that $X_2 \neq X_3$. This is because the encoding is made under a Closed World Assumption: it is assumed that a binary string encodes the variable sharing information completely. If this were not so, encodings would not represent clauses but subsets of clauses, i.e. all clauses in the search domain that are compatible with the variable sharing described by the 1-bits in the binary string. Since a 0 does not reflect incomplete information in the binary string but encodes negative information, normalisation by closure does not fulfill its intended meaning, which is to fill in inferred information.

Normalisation by closure remains to be but one of many alternative ways of resolving inconsistency. It proceeds by switching the minimum number of 0-bits to 1-bits that allows the consistency rules to be satisfied. To resolve inconsistency we can proceed with any other algorithm that, switching a number of bits, would achieve consistency under the normalisation rules. For instance, we can decide that, instead of switching 0-bits to 1-bits, we switch 1-bits to 0-bits. While we can choose, for instance, that the number of bits so changed be minimal (i.e. we determine all valid clauses at minimal Hamming distance), or some other criterion, we do not have any rational basis for preferring one criterion over another. The normalisation by closure is therefore syntactically computable but semantically undefined.

3.2 Fallacy 2 — Fact 2

Let us consider Example 2 in [21]. It is claimed that the clause $p(U,V) \longleftarrow q(U,X), r(X,Z)$ is the *mgi*[2] of the clauses $p(U,V) \longleftarrow q(U,X), r(Y,Z)$ and $p(U,V) \longleftarrow q(W,X), r(X,Z)$ under subsumption $\succeq$. However, this is not true. The *mgi* of the two clauses under $\succeq$, according to [15, p. 251], is $p(U,V) \longleftarrow q(U,X), q(W,X'), r(Y,Z), r(X',Z')$. We may wonder if this is not subsume-equivalent with $p(U,V) \longleftarrow q(U,X), r(X,Z)$. Suppose, by *reductio ad absurdum*, that the latter clause subsumes the former. Then there should be a substitution mapping it onto a subset of the former. Necessarily, $\{U/U, V/V\}$ because of the common head. Then $q(U,X)$ is mapped to some $q(U,\_)$ and the only such literal available is $q(U,X)$, therefore $\{X/X\}$. Then $r(X,Z)$ is mapped to some $r(X,\_)$ but there are no such literals in the former clause, contradiction. In fact the *mgi* given in [21] is computed under atomic subsumption $\succeq_a$. Crucially, for *mgi*'s under $\succeq_a$ Theorem 4 in [21] does not hold, i.e. one may not compute coverages of such *mgi*'s by intersecting the coverages of the parent clauses as described in [21]. This is because coverages of clauses are computed under entailment $\models$ (or, in some circumstances, subsumption $\succeq$) but not under the much weaker atomic subsumption $\succeq_a$. For instance, consider the clause $p(U,V) \longleftarrow q(U,X), q(W,X'), r(Y,Z), r(X',Z')$ referred to above. Under $\succeq$ it is covered by both $p(U,V) \longleftarrow q(U,X), r(Y,Z)$ and $p(U,V) \longleftarrow q(W,X), r(X,Z)$, therefore it belongs to the intersection of their coverages. However, as shown, it is not covered by the clause $p(U,V) \longleftarrow q(U,X), r(X,Z)$, the alleged *mgi*. This renders the proposed fast evaluation mechanism unsound and therefore moot.

3.3 Fallacy 3 — Fact 3

According to the theory of refinement in ILP [11,15], refinement operators maintain the search space implicitly. Various qualities may be required of any well-defined refinement operator, such as soundness, completeness (weak or strong), local finiteness, properness, minimality etc. Soundness and completeness are minimal requirements. However, we have seen in the preceding sections that the

[2] most general instantiation

search space explored by the task-specific genetic operators in [20,21] is both unsound and incomplete. The most we could hope for is that such operators, once applied to valid encodings, will produce a valid encoding. However, this is not the case. Both *mgi* and mutation may produce spurious outcomes starting from valid parent strings, for instance taking *mgi* of $(1, 0, 0)$ and $(0, 1, 0)$ in our running counter-example will give the inconsistent $(1, 1, 0)$, while 1-bit mutation applied to $(1, 0, 0)$ will yield two inconsistent strings and a consistent one.

4 Potential Solutions

This paper was initially written and reviewed in a two-part format, containing both our criticism of the binary representation approach and our alternative thereto. Unfortunately, for reasons strictly connected with the 12-page limit allowed for full papers in the conference proceedings, the second part had to be dropped. The author plans to submit an extended version of this paper to a journal in the Artificial Intelligence field, including our proposed solution to the binary representation flaws described in this paper. The interested reader is invited to contact the author for details.

5 Conclusions

Although combining genetic algorithms with inductive logic programming is potentially a valuable approach, it is not straightforward. Inductive logic programming owes its existence both to the first-order representations that it uses, but also to the recognition of the fact that it employs search mechanisms that are not easily translatable in the propositional domain. Even when such transformations can be made, it is usually under heavy restrictions or at the expense of exponential blow-ups in complexity [6]. It is why the inductive logic programming community has painstakingly developed inductive mechanisms that work with first-order representations directly. Particularly, refinement operators defined on clauses lie at the core of many inductive logic programming systems.

Tamaddoni-Nezhad and Muggleton have proposed [20,21] an approach that recycles some well-known propositionalisation techniques under the name of binary representations. Although it is claimed that this approach achieves completeness in respect of PROGOL's search, it is in fact even more incomplete than PROGOL's existing search. Although they use the phrases "genetic refinement" in [20] and "stochastic refinement" in [21] to name this type of search, no definition of these terms is given and no explanation on how they relate to the very consistent body of research on refinement [11,12,15,23]. At present it is not possible to determine with enough precision how the flaws in their theoretical framework affect their implementation since they do not give the algorithm on which the implementation is based, nor enough detail regarding their experimental settings. It is therefore not possible to either replicate or criticise their experimental evidence. However, in [20] the authors state:

"In our first attempt, we employed the proposed representation to combine Inverse Entailment in C-PROGOL4.4 *with a genetic algorithm. In this implementation genetic search is used for searching the subsumption lattice bounded below by the bottom-clause ($\perp$).* ***According to Theorem 3 the search space bounded by the bottom clause can be represented by $S(\perp)$."***

However, a careful consideration of Theorem 3 given in their paper will show that the theorem's statement does not entail the identity between $H_{\succeq}$ and $S(\perp)$. Furthermore, the argument can not be repaired: $S(\perp)$ is the space of binary representations shown in this paper to be incomplete, unsound, and noncompact with respect to $H_{\succeq}$. Since, by the results of this paper, the statement quoted above does not hold, we should conjecture that caution is needed in accepting the authors' alleged experimental results until such time that the flaws unearthed in this paper are given a proper solution.

Acknowledgements. Thanks to James Cussens for proofreading a preliminary version of this paper. The author thanks the four anonymous reviewers for their interest, careful evaluation and commentary, part of which is going to be used in future extensions of this work. This paper is dedicated to my maternal grandparents: Maria Iclănzan (Buni) and Andrei Iclănzan (Bunicu'), who taught me the value of imagination and rational endeavour, respectively; and to the memory of my departed paternal grandparents, Anica Mărginean and Ioan Mărginean, who gave me my name and taught me the meaning of faith and dignity in face of adversity.

References

1. É. Alphonse and C. Rouveirol. Object Identity for Relational Learning. Technical report, LRI, Université Paris-Sud, 1999. Supported by ESPRIT Framework IV through LTR ILP2.
2. É. Alphonse and C. Rouveirol. Test Incorporation for Propositionalization Methods in ILP. Technical report, LRI, Université Paris-Sud, 1999. Supported by ESPRIT Framework IV through LTR ILP2.
3. L. Badea and M. Stanciu. Refinement Operators Can Be (Weakly) Perfect. In S. Džeroski and P. Flach, editors, *Inductive Logic Programming, 9th International Workshop, ILP-99*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 21–32. Springer, 1999.
4. J. Cussens and A. Frisch, editors. *Inductive Logic Programming—ILP 2000, Proceedings of the 10th International Conference on Inductive Logic Programming*, Work-in-Progress Reports, Imperial College, UK, July 2000. Work-in-Progress Reports.
5. J. Cussens and A. Frisch, editors. *Inductive Logic Programming—ILP 2000, Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, Imperial College, UK, July 2000. Springer.

6. L. De Raedt. Attribute-Value Learning versus Inductive Logic Programming: The Missing Links. In Page [18], pages 1–8.
7. F. Divina and E. Marchiori. Knowledge Based Evolutionary Programming for Inductive Learning in First-Order Logic. In Spector et al. [19], pages 173–181.
8. K. Furukawa and T. Ozaki. On the Completion of Inverse Entailment for Mutual Recursion and its Application to Self Recursion. In Cussens and Frisch [4], pages 107–119.
9. K. Inoue. Induction, Abduction and Consequence-Finding. In C. Rouveirol and M. Sebag, editors, *Inductive Logic Programming—ILP 2001*, volume 2157 of *Lecture Notes in Artificial Intelligence*, pages 65–79. Springer, 2001.
10. K. Ito and A. Yamamoto. Finding Hypotheses from Examples by Computing the Least Generalization of Bottom Clauses. In S. Arikawa, editor, *Proceedings of Discovery Science*, volume 1532 of *Lecture Notes in Artificial Intelligence*, pages 303–314. Springer-Verlag, 1998.
11. P.R. Laag. An Analysis of Refinement Operators in Inductive Logic Programming. Technical Report 102, Tinbergen Institute Research Series, 1995.
12. Philip D. Laird. *Learning from Good Data and Bad.* PhD thesis, Yale University, 1987.
13. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
14. S. Muggleton. Completing Inverse Entailment. In Page [18], pages 245–249.
15. S-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming.* Springer-Verlag, Berlin, 1997. LNAI 1228.
16. K. Ohara, N. Babaguchi, and T. Kitahashi. An Efficient Hypothesis Search Algorithm Based on Best-Bound Strategy. In Cussens and Frisch [4], pages 212–225.
17. H. Ohwada, H. Nishiyama, and F. Mizoguchi. Concurrent Execution of Optimal Hypothesis Search for Inverse Entailment. In Cussens and Frisch [5], pages 165–173.
18. D. Page, editor. *Inductive Logic Programming, Proceedings of the* 8^{th} *International Conference, ILP-98*, volume 1446 of *Lecture Notes in Artificial Intelligence.* Springer, July 1998.
19. L. Spector, E.D. Goodman, A. Wu, W.B. Langdon, H-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M.H. Garzon, and E. Burke, editors. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, San Francisco, CA, July 7–11 2001. AAAI, Morgan Kaufmann.
20. A. Tamaddoni-Nezhad and S.H. Muggleton. Searching the Subsumption Lattice by a Genetic Algorithm. In Cussens and Frisch [5], pages 243–252.
21. A. Tamaddoni-Nezhad and S.H. Muggleton. Using Genetic Algorithms for Learning Clauses in First-Order Logic. In Spector et al. [19], pages 639–646.
22. A. Tamaddoni-Nezhad and S.H. Muggleton. A Genetic Algorithms Approach to ILP. Proceedings of the Twelfth International Conference on Inductive Logic Programming, 2002. In Press.
23. F. Torre and C. Rouveirol. Private Properties and Natural Relations in Inductive Logic Programming. Technical report, LRI, Université Paris-Sud, July 1997.
24. A. Yamamoto. Which Hypotheses Can Be Found with Inverse Entailment? In N. Lavrač and S. Džeroski, editors, *Inductive Logic Programming,* 7^{th} *International Workshop, ILP-97*, volume 1297 of *Lecture Notes in Artificial Intelligence*, pages 296–308. Springer, 1997.