A Hybrid Diagnosis Approach Combining Black-Box and White-Box Reasoning

Mingmin Chen[1], Shizhuo Yu[1], Nico Franz[2], Shawn Bowers[3], and Bertram Ludäscher[1]

[1] Dept. of Computer Science, University of California, Davis, CA, USA
{michen,szyu,ludaesch}@ucdavis.edu
[2] School of Life Sciences, Arizona State University, AZ, USA
nico.franz@asu.edu
[3] Dept. of Computer Science, Gonzaga University, WA, USA
bowers@gonzaga.edu

Abstract. We study model-based diagnosis and propose a new approach of hybrid diagnosis combining black-box and white-box reasoning. We implemented and compared different diagnosis approaches including the standard hitting set algorithm and new approaches using answer set programming engines (DLV, Potassco) in the application of EULER/X toolkit, a logic-based toolkit for alignment of multiple biological taxonomies. Our benchmarks show that the new hybrid diagnosis approach runs about twice fast as the black-box diagnosis approach of the hitting set algorithm.

1 Motivation and Related Work

Model-based diagnosis was studied extensively in many areas, such as type error debugging, circuit diagnosis, OWL debugging, etc. Various approaches [17,7,3,15,2,14] have been proposed to diagnose or debug errors. Most of these diagnosis approaches compute *minimal inconsistent subsets* (a.k.a. diagnoses) and/or *maximal consistent subsets*. A common element of all these approaches is that they use a routine isInconsistent as a "black-box" to determine if a set of constraints is *unsatisfiable*. The best black-box approach we know of is [14] which is a hybrid of the Logarithmic Extraction Algorithm [2] and the Hitting Set (HST) Algorithm in [15].

The downside of these black-box approaches is that they do not look into the proof itself that the reasoner may provide in the isInconsistent routine, which may potentially lose some reusable information to reduce the number of invocations of reasoners. On the other hand, various provenance approaches have been studied to provide derivations and proof trees, such as [13] which proposes an approach by adding annotations to predicates to generate a *provenance semiring* of a derivation, Datalog debugging [16] which proposes a provenance-enriched rewriting for debugging and profiling declarative rules. Inspired by these provenance approaches, we introduce our own white-box provenance approach to generate diagnosis proof trees for model-based diagnosis problem. Both approaches in [13] and [16] are not good at generating derivations of rules with negations, whereas our white-box provenance approach works for rules with negations too. White-box and black-box approaches output different products. The idea of inferring one from the other or combining both approaches is proposed in [18,4,10].

A. Bikakis et al. (Eds.): RuleML 2014, LNCS 8620, pp. 127–141, 2014.

We also propose a new hybrid approach which combines the black-box and white-box approaches to obtain diagnoses.

Our white-box provenance approach and hybrid approach can be applied to general model-based diagnosis problems which can be encoded in Datalog rules with negations and aggregates. One interesting application of the new approaches is the inconsistency analysis feature of EULER/X toolkit [6], a toolkit for logic-based taxonomy integration. In EULER/X we use answer set programming (ASP) systems (DLV and Potassco) as underlying reasoners. We implemented the existing black-box approach, our white-box approach, and hybrid approach in EULER/X toolkit and compared them in the benchmarks.

EULER/X. The problem of aligning multiple related biological taxonomies was studied and modeled in monadic first-order logic in the CLEANTAX project [20,19]. A $taxonomy$ is a containment (or *isa*) hierarchy with additional taxonomic constraints. The EULER/X toolkit [6], a toolkit for logic-based taxonomy integration, builds on this effort while utilizing additional and more time-efficient logic approaches such as answer set programming [5]. EULER/X further more provides diversified and interactive workflow features leading to the identification and visualization of consistent merged taxonomies. Under this approach an expert initiates the process of aligning two related but different taxonomies (T_1, T_2) by providing a set of articulations A and a set of taxonomic constraints TC.

Jointly these input conditions – T_1, T_2, A, and TC – can generate various and potentially inter-dependent instances of inconsistency; and thus a failure to yield consistent alignments and visualizations. When asserting the initial articulations experts will frequently make mistakes for various reasons; including (1) human error in information entry or transcription, (2) a failure to understand transitive interdependencies among input articulations, (3) incorrect accounting for low-level (child) concepts in relation to parent concepts, (4) unwarranted violations of one or more taxonomy constraints, and (5) other forms of logically inconsistent input. Each kind of error will yield a logically inconsistent alignment, where one input condition is somehow in contradiction with one or more additional conditions. Repair of such errors is needed, however the native ASP reasoner output is virtually unreadable by humans, offering little comprehension why the inputs are inconsistent and what cause the inconsistency. In order to identify and remedy these problems, it is critical to "isolate" local sources of inconsistency that are particularly relevant to facilitating the desired repair action from the global inconsistency phenomenon. To this end the EULER/X toolkit provides a novel *Inconsistency Analysis* feature which motivates the investigation of different diagnosis approaches.

Contributions. This paper proposes a new hybrid diagnosis approach combining black-box and white-box reasoning. Our white-box provenance approach records the provenance of rules with or without aggregates. We have implemented different black-box, white-box, and mixed approaches for generating diagnoses, and diagnosis proof trees in the application of EULER/X toolkit using ASP systems for constraint solving and reasoning. We also show in the benchmarks that our hybrid approach runs much faster comparing to the existing best black-box approach of hitting set algorithm for generating all diagnoses.

2 Background

A *system* is a pair (SD, C) where the *system description* SD is a set of fixed sentences (assumed to be true), and C is a set of *constraints* (which the user wants to be true, but which might be inconsistent with SD). $C_0 \subseteq C$ is called a *Minimal Inconsistent Subset* (MIS) of (SD, C) if (i) $SD \cup C_0$ is inconsistent and $SD \cup C'$ is consistent for any proper subset $C' \subsetneq C_0$. Conversely, $C_0 \subseteq C$ is called a *Maximal Consistent Subset* (MCS) of C if $SD \cup C_0$ is consistent and there is no other consistent C_1 with $C_0 \subsetneq C_1 \subseteq C$. We denote by $\mathcal{MIS}$ and $\mathcal{MCS}$ the set of all MIS and all MCS, respectively.[1]

Given a set C with n constraints, one can use "brute force" and find all diagnoses by checking the consistency of all 2^n subsets and prune the non-minimal ones. Often it is unnecessary to check *all* combinations. For example, if we know a combination S is inconsistent, then any superset of S is inconsistent as well, so we don't need to check those supersets. One the other hand, any subset of a consistent set is also consistent:

Fact 1. *If a set of constraints is unsatisfiable in the system, any of its superset is unsatisfiable.*

Fact 2. *If a set of constraints is satisfiable in the system, all its subsets are satisfiable.*

Actually, most of the existing black-box diagnosis approaches use these two facts. In the next section, we will recap the best black-box approach [14] we know of which is a hybrid of Logarithmic Extraction Algorithm [2] and combined with Hitting Set (HST) Algorithm in [15].

3 Black-Box Approaches

We will first look at a black-box approach for generating all diagnoses – Horridge's approach [14] which uses Logarithmic Extraction Algorithm [2] as a subroutine in the HST algorithm. Logarithmic Extraction Algorithm is to compute one single MIS which is shown in Algorithm 1.

In Algorithm 1, depending on how we split F in line 3 of Function-1R, the time complexity of this algorithm is different. In general, we split F by half and half. As the name of this algorithm suggests, it calls isInconsistent $log\ n$ times on average. In the worst case (we always go to line 8 and 9 in Function-1R), it invokes isInconsistent $O(n)$ times. We have Lemma 1 which originates from [17] and is crucial for the idea of computing all MIS.

Lemma 1. *Denote by $\mathcal{MIS}(SD, C)$ all the MIS, and assume $S = \{s_1, s_2, \ldots, s_t\}$ is a MIS, we have $\mathcal{MIS}(SD, C) = \bigcup_{1 \le i \le t} \mathcal{MIS}(SD, C \backslash \{s_i\})$.*

By this lemma, we get an algorithm to compute all MIS. However, it is inefficient if we don't remember what has been computed and what has not since we are

[1] MIS are also known as *diagnosis* [17], *justification* [14], *minimal conflict sets* [8], and *minimal unsatisfiable set* [3]; MCS are a.k.a. *maximal satisfiable set* [3].

Algorithm 1. Logarithmic Extraction

Input: System description *SD*, a set of constraints *C*
Output: One single diagnosis (MIS)

Function-1 ComputeSingleMIS(*SD, C*)
1: **return** *ComputeSingleMIS(SD, ∅, C)*

Function-1R ComputeSingleMIS(*SD, S, F*)
1: **if** $|F| = 1$ **then**
2: **return** F
3: $S_L, S_R \leftarrow$ *split(F)*
4: **if** isInconsistent($S \cup S_L$) **then**
5: **return** *ComputeSingleMIS(*$SD,\ S,\ S_L$*)*
6: **if** isInconsistent($S \cup S_R$) **then**
7: **return** *ComputeSingleMIS(*$SD,\ S,\ S_R$*)*
8: $S'_L \leftarrow$ *ComputeSingleMIS(*$SD,\ S \cup S_R,\ S_L$*)*
9: $S'_R \leftarrow$ *ComputeSingleMIS(*$SD,\ S \cup S'_L,\ S_R$*)*
10: **return** $S'_L \cup S'_R$

likely to recompute something. For instance, to get $\mathcal{MIS}(SD, C\backslash\{c_1\})$, we may compute $\mathcal{MIS}(SD, C\backslash\{c_1,\ c_2\})$ which may be already computed when getting $\mathcal{MIS}(SD, C\backslash\{c_2\})$. We definitely need some caching optimization to avoid such a case. $\{c_1, c_2\}$ is called $path$ when we compute $\mathcal{MIS}(SD, C\backslash\{c_1, c_2\})$. HST algorithm [15] which we show in Algorithm 2 records all the $path$s it has already visited (i.e. argument $allpaths$ in ComputeAllMISHST), and will not visit them again.

In the worst case, *ComputeAllMIS* calls *ComputeSingleMIS* for $\Theta(2^n)$ times. Horridge et al. [14] proposed a mixed algorithm of HST algorithm [17] and Logrithmic extractraction algorithm [2] to generate $\mathcal{MIS}$, i.e., Algorithm 1 is a subroutine used in the Algorithm 2 to compute one single MIS. This gives us the worst case time complexity of $O(2^n) * O(n) * R(n) = O(n * 2^n) * R(n)$ where $R(n)$ is the time complexity of isInconsistent. We implement isInconsistent using Answer Set Programming, so $R(n)$ is as hard as Σ_2^P by [9]. Eiter and Gottlob [8] have pointed out the time complexity of computing all diagnoses (i.e. $\mathcal{MIS}$) is *TRANS-ENUM*-complete, which means there is no efficient (polynomial time) algorithm to get $\mathcal{MIS}$ unless *TRANS-ENUM* had (but is believed not) a polynomial time algorithm.

4 White-Box Provenance Approach

As mentioned in the last section, the best black-box approach calls isInconsistent a (large) number of times to generate $\mathcal{MIS}$, which seems not quite efficient. isInconsistent routine is usually implemented using the underlying reasoner. Can we get from the reasoner not only the yes/no answer? Can we call the reasoner once to obtain all desired diagnoses?

We consider the diagnosis problem whose isInconsistent is implemented using answer set programming system, and either a system description sentence or a constraint is encoded as Datalog rules (with/without negation/aggregate). Inspired by the ideas of

Algorithm 2. ComputeAllMIS (HST Algorithm)

Input: System description *SD*, a set of constraints *C*
Output: All diagnoses ($\mathcal{MIS}$)

Function ComputeAllMIS(SD, C)
1: S, curpath, allpaths $\leftarrow \emptyset$
2: $S \leftarrow$ *ComputeAllMISHST(SD, C, S, curpath, allpaths)*
3: **return** S

Function ComputeAllMISHST(*SD,C,S,curpath,allpaths*)
1: **for** $path \in allpaths$ **do**
2: **if** $curpath \supseteq path$ then
3: // Path termination without consistency check
4: **return** *S*
5: **if** not isInconsistent*(SD, C)* **then**
6: allpaths $\leftarrow$ allpaths $\cup$ {curpath}
7: **return** *S*
8: $J \leftarrow \emptyset$
9: **for** $s \in S$ **do**
10: **if** $s \cap$ curpath = $\emptyset$ **then**
11: // MIS reuse (saves recomputing a MIS)
12: $J \leftarrow s$
13: **if** $J = \emptyset$ **then**
14: $J \leftarrow$ *ComputeSingleMIS(SD, S)*
15: $S \leftarrow S \cup \{J\}$
16: **for** $ax \in J$ **do**
17: curpath $\leftarrow$ curpath $\cup \{ax\}$
18: **return** *ComputeAllMISHST*$(SD, C \backslash \{ax\}$*, S, curpath, allpaths)*

provenance semiring [13] and Datalog debugging [16], we propose a white-box provenance approach which rewrites all the Datalog rules, records the derivation of inconsistency which is a boolean expression, and generates an inconsistency proof tree using the boolean expression. The basic idea of recording the derivation of inconsistency is to first rewrite all the rules by adding annotations. For a rule without head (or false is the head), NOK is added as the head which stands for "Not OK", i.e., inconsistency. We show the detailed Datalog rule rewritings as follows.

4.1 Non-aggregate Rule Rewriting

A rule r is *safe* if every variable in r must also occur positively in the body.

1. For any constraint rule with head predicate:

$$r_1 : H_1(\bar{Y}) :- B_1(\bar{X_1}), B_2(\bar{X_2}), \ldots, B_n(\bar{X_n}).$$

We rewrite it by adding annotations for each predicate (including head and body predicates) where P_i is the provenance of $B_i(\bar{X}_i)$ for $1 \leq i \leq n$, and we use $\otimes$ to represent logical and[2]:

$$H_1(\bar{Y}, r_1 \otimes (P_1 \otimes \ldots \otimes P_n)) :- B_1(\bar{X}_1, P_1), B_2(\bar{X}_2, P_2), \ldots, B_n(\bar{X}_n, P_n).$$

2. For any constraint rule without head predicate (i.e., false is the head):

$$r_2 : \mathsf{false} :- B_1(\bar{X}_1), B_2(\bar{X}_2), \ldots, B_n(\bar{X}_n).$$

We rewrite it to a constraint with head predicate NOK where P_i is the provenance of $B_i(\bar{X}_i)$ for $1 \leq i \leq n$ and NOK stands for "Not OK", i.e. inconsistency:

$$\mathsf{NOK}(r_2 \otimes (P_1 \otimes \ldots \otimes P_n)) :- B_1(\bar{X}_1, P_1), B_2(\bar{X}_2, P_2), \ldots, B_n(\bar{X}_n, P_n).$$

We use a trick to get rid of non-safe rules: For any predicate V that has negation in some rules, we add a complement predicate $\tilde{V}$ for V, and add choice rules of "$V(\bar{X}) :-$ not $\tilde{V}(\bar{X}), domain(\bar{X})$" and "$\tilde{V}(\bar{X}) :-$ not $V(\bar{X}), domain(\bar{X})$".

4.2 Aggregate Rule Rewriting

Datalog rules in answer set programming could also have aggregates. For example, in DLV, we may have aggregates such as #count. We show the rewriting for constraints with #count.

For a constraint:

$$r_3 : \mathsf{false} :- \#\mathsf{count}\{X : V(X), B_1(X, \bar{Y}_2), \ldots, B_n(X, \bar{Y}_n)\} = 0.$$

First we have the complement rule $\tilde{V}$ for V, and add choice rules of "$V(X) :-$ not $\tilde{V}(X), domain(X)$" and "$\tilde{V}(X) :-$ not $V(X), domain(X)$". Then we rewrite the constraint to a soft constraint where P_i is the provenance of $B_i(X, \overline{Y_i})$ for $1 \leq i \leq n$ and add two predicates POK and OK which stand for Possibly OK and OK (i.e. consistency), respectively:

$$\begin{aligned}
\mathsf{POK}(r_3, P_{\tilde{V}} \otimes P_1 \otimes \ldots \otimes P_n) :- & \tilde{V}(X, P_{\tilde{V}}), \\
& B_1(X, \bar{Y}_1, P_1), B_2(X, \bar{Y}_2, P_2), \ldots, B_n(X, \bar{Y}_n, P_n). \\
\mathsf{OK}(r_3) :- & V(X), \\
& B_1(X, \bar{Y}_1), B_2(X, \bar{Y}_2), \ldots, B_n(X, \bar{Y}_n). \\
\mathsf{NOK}(r_3 \otimes P) :- & \mathsf{POK}(r_3, P), \mathsf{not}\ \mathsf{OK}(r_3).
\end{aligned}$$

We only show the rewriting for rules with aggregates of such a format because in this is the only format with aggregate we encounter in our real world application of EULER/X toolkit. Rules of other format can also be rewritten similarly.

[2] It is the same as times operator as in provenance semiring [13].

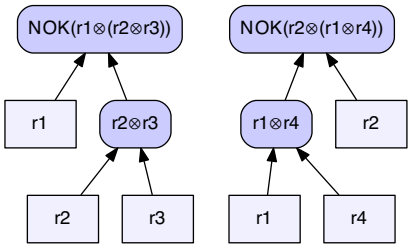

Fig. 1. Example diagnosis proof trees

4.3 Generation of Diagnosis Proof Tree

With the rewritten rules enriched with annotations, one can query answer set programming system all the possible answers for NOK and obtain a boolean expression for each possible answer. For instance, suppose the possible answers for NOK is

$$\{\mathsf{NOK}(r_1 \otimes (r_2 \otimes r_3)), \mathsf{NOK}(r_2 \otimes (r_1 \otimes r_4))\}$$

The boolean expressions for NOK are $r_1 \otimes (r_2 \otimes r_3)$ and $r_2 \otimes (r_1 \otimes r_4))$. For each boolean expression, we construct its boolean expression tree [1] which is a diagnosis proof tree for the inconsistency. We have the two diagnosis proof trees shown in Fig. 1.

A diagnosis proof tree shows how different constraints together lead to the inconsistency. The rule-rewritting based white-box approach is shown in Algorithm 3. We will show an example of diagnosis proof tree in the Application section (Section 6).

Algorithm 3. White-Box Approach

Input: System description *SD*, a set of constraints *C*
Output: All diagnosis proof trees

ComputeAllProofTrees(SD, C):
1: Encode *SD* and *C* in Datalog rules
2: Rewrite Datalog rules to ones with provenance
3: Run ASP reasoner to get boolean expressions for NOK
4: Construct diagnosis proof trees using the boolean expressions

5 Mixed Black-Box / White-Box Approach

It is hard to compare black-box approaches and white-box provenance approach in the sense that they generate different outputs for model-based diagnosis problem. Black-box approaches generate $\mathcal{MIS}$ whereas white-box approach generates proof trees. Note that the leaf nodes of a proof tree together forms a set of constraints which is either a MIS or a superset of a MIS. Starting from these constraints, a MIS may be obtained by running HST algorithm.

We propose the hybrid approach, in which white-box approach serves as a filter to shrink the universe (constraint candidates for $\mathcal{MIS}$) when generating $\mathcal{MIS}$. This hybrid approach is shown in Algorithm 4.

Algorithm 4. Hybrid Approach

Input: System description *SD*, a set of constraints *C*
Output: All diagnoses ($\mathcal{MIS}$)

ComputeAllMISHybrid(SD, C):
1: *Ts* ← *ComputeAllProofTrees(SD, C)*
2: *C'* ← set of leaf nodes of the proof trees Ts
3: **return** *ComputeAllMIS(SD,C')*

If the size of the input constraint set C is large, white-box approach could potentially shrink the constraint set to a much smaller one C' for HST Algorithm and thus reduce the running time of the HST algorithm. In the following sections, we will show the application of different diagnosis algorithms in EULER/Xtoolkit and compare the performance between black-box approach and hybrid approach for generating $\mathcal{MIS}$ in the benchmarks.

Relation Between All These Approaches. We show the relation between different approaches in Fig. 2. A system (SD, C) is the initial input for different diagnosis approaches. We will also show how to compute $\mathcal{MIS}$ / $\mathcal{MCS}$ from each other in Section 6.2.

6 Real World Application – EULER/X

EULER/X [6] is a logic-based toolkit for aligning multiple biological taxonomies. A $taxonomy$ is an isa hierarchy made up of taxonomic concepts. An $articulation$ defines a relation between taxonomic concepts using union ($\cup$) and Region Connection Calculus (RCC-5) relations. RCC-5 includes five basic relationships that compare the extensions of taxonomic concepts: viz. (1) congruence ($==$), $\mathsf{E}_1 == \mathsf{E}_2$ meaning that two taxonomic concepts E_1 and E_2 are equivalent; (2) proper inclusion ($>$), $\mathsf{E}_1 > \mathsf{E}_2$ meaning that E_1 properly includes E_2; (3) inverse proper inclusion ($<$), $\mathsf{E}_1 < \mathsf{E}_2$ meaning that E_1 is properly included in E_2; (4) overlap ($><$), $\mathsf{E}_1 >< \mathsf{E}_2$ meaning that E_1 is overlapping with E_2; (5) exclusion (!), E_1 ! E_2 meaning that E_1 and E_2 have an empty intersection. Ambiguity can be asserted using the disjunction 'or'. *isa* in the taxonomy can be treated as $<$ or $==$. The toolkit ingests the taxonomies (T_1, T_2), a set of articulations [12,11] (A), and takes into account three additional constraints (TC): (1) nonemptiness - a given concept has minimally one representing instance; (2) sibling disjointness - two given child concepts of a parent concept are exclusive of each other; and (3) coverage - a given parent concept is completely circumscribed by the union of its children concepts. The toolkit then generates merged taxonomies.

A taxonomy alignment can be treated as a system (SD, C) where two input taxonomies and taxonomic constraints together are the system description, i.e., SD =

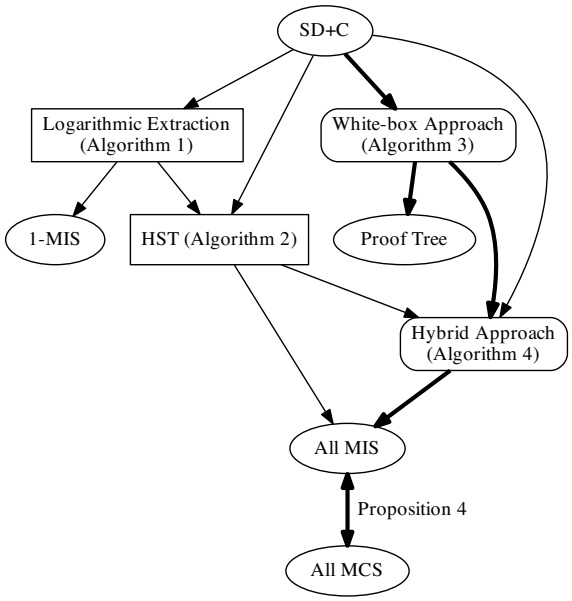

Fig. 2. Relations between different approaches (boxes: existing approaches; rounded boxes: new approaches; bold edges: how our hybrid approach works from end to end)

$T_1 \cup T_2 \cup TC$, and input articulations are the constraints C. As mentioned in Section 1, a taxonomy alignment may yield inconsistent results because articulations may be wrongly asserted by domain experts due to various reasons. To analyze the inconsistency of a taxonomy alignment, EULER/X toolkit applies different diagnosis approaches including black-box approach, white-box approach, and hybrid approach.

6.1 Example

Example 1. Suppose we have two minimal taxonomies as shown on the left in Fig. 3. Taxonomy 1 has three concepts 1.A, 1.B, and 1.C. The concept hierarchy is modeled by the two "is-a" constraints: c_0: "1.B isa 1.A", and c_1: "1.C isa 1.A". We also have a *coverage* constraint c_2: "1.A $=$ 1.B $+$ 1.C"; and a *sibling disjointness* constraint c_3: "1.B disjoint 1.C". Taxonomy 2 has a single concept 2.D. There are three articulations between the taxonomies: c_4: "1.A $>$ 2.D", c_5: "1.B ! 2.D", and c_6: "1.C ! 2.D". By running EULER/X, we find that this alignment is inconsistent.

With HST algorithm, we find that there is only one diagnosis (MIS) which is "c_4 : 1.A $>$ 2.D, c_5 : 1.B ! 2.D, c_6 : 1.C ! 2.D". White-box provenance approach generates one proof tree, which is shown as Fig. 3. Domain expert may interpret the proof tree though not that obvious that $c_2 \otimes c_5 \otimes c_6$ means "1.A ! 2.D", and $\mathsf{NOK}(c_4)$ means that c_2, c_5, c_6 and c_4 together introduce the inconsistency, i.e., "1.A ! 2.D" and "1.A $>$ 2.D" cannot both hold. The set of leaf nodes of the proof tree is $\{c_2, c_4, c_5, c_6\}$. Since c_2 is a

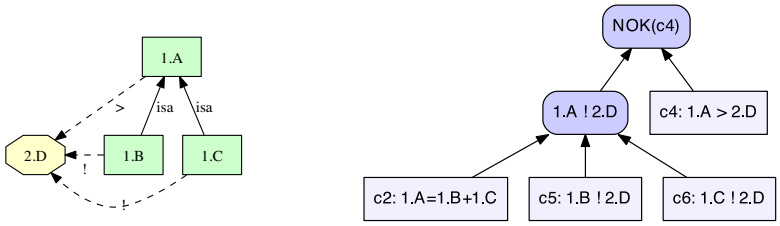

Fig. 3. Input alignment (left) and proof tree (right) for Example 1. From constraints c_2, c_5, and c_6 it follows that "1.A ! 2.D", which is inconsistent with the articulation c4: "1.A $>$ 2.D"

taxonomic constraint which is part of system description, we get a set of $\{c_4, c_5, c_6\}$ as constraint candidates for MIS.

6.2 Diagnostic Lattice

With all diagnoses $\mathcal{MIS}$ (and $\mathcal{MCS}$), it may be helpful to visualize all diagnoses. EULER/X toolkit visualizes diagnoses as in a lattice. Consider the 2^n combinations of n articulations, we can build a lattice where an edge means there is a direct subset relation between the two sets, i.e., there is an edge (A, B) iff $A \subsetneq B$ and $|B - A| = 1$. We call it *Diagnostic Lattice*. For example, the lattices with articulation set size of 2, 3, 4 are shown in Fig. 4.

In the diagnostic lattice, we color a node red if the set of articulations it represents is inconsistent; otherwise, color it green. Recall Fact 1 and 2, which we could call *Inconsistency Propagation* (red edges) and *Consistency Propagation* (green edges), respectively: Any ancestor (superset) of an inconsistent (red) node is also inconsistent; similarly, any descendent (subset) of a consistent (green) node is also consistent. MIS is essentially a red node whose parents are green; MCS is a green node whose children are red. We color MIS, MCS solid red, solid green, respectively. We color an edge as dashed red if it applies Red Propagation Rule; color it as dashed green if it applies Green Propagation Rule; color it solid blue if it applies neither of the two rules. For example, we have four articulations $\{a, b, c, d\}$, among which $\{a, b\}$, $\{a, c\}$, and $\{d\}$ are MIS, we have the colored lattice in Fig. 5.

Actually we can represent both MIS and MCS with boolean functions. Using the solid red nodes, we get $\mathsf{NOK}(\{a, b, c, d\}) = (a \wedge b) \vee (a \wedge c) \vee d$. Using the solid green nodes, we get $\mathsf{OK}(\{a, b, c, d\}) = a \vee (b \wedge c)$. We found that $\mathsf{NOK}(\{a, b, c, d\}) = \neg\mathsf{OK}(\{a, b, c, d\})$.

Example 2. Fig. 6 shows a more complex example with 12 articulations, so the number of combinations of articulations (the number of the nodes in lattice) is 2^{12}, which is 4096. By using our lattice approach, we get 5 MIS and 7 MCS among all 4096 combinations, together with the clusters of other inconsistent or consistent nodes, shown in Fig. 7.

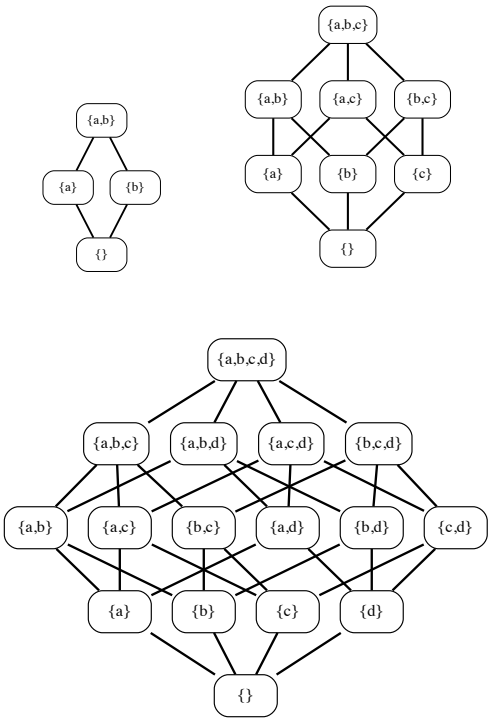

Fig. 4. Lattices with articulation set size of 2, 3, and 4

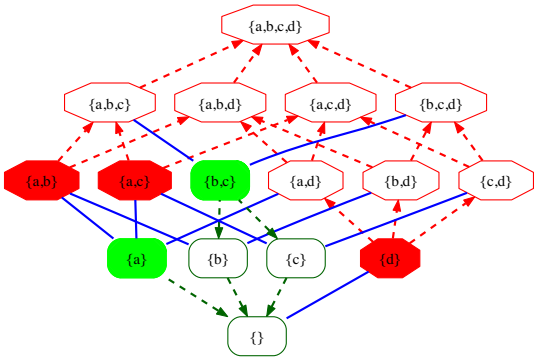

Fig. 5. Diagnostic lattice. (solid red octagon: MIS, solid green rounded box: MCS)

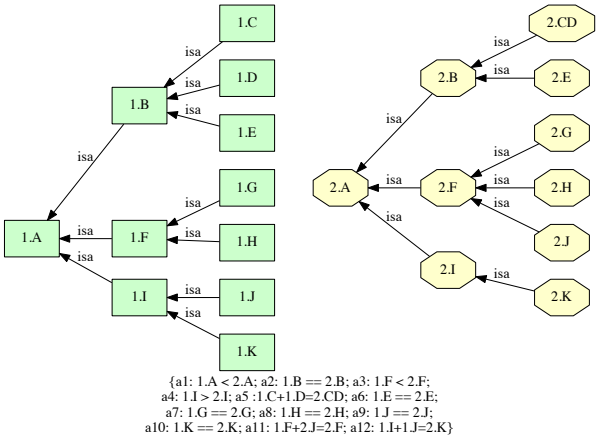

Fig. 6. Input for Example 2 (Green: Taxonomy1, Yellow: Taxonomy2, a1-a12: Articulations)

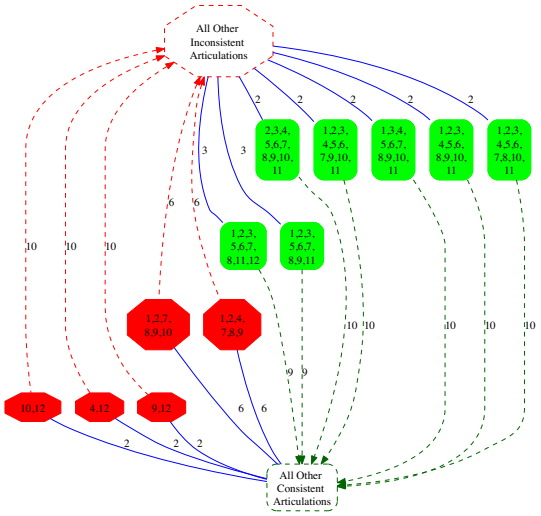

Fig. 7. $\mathcal{MIS}$ (Octagon) and $\mathcal{MCS}$ (Rounded Box) for Example 2. All other non-minimal inconsistent subsets and non-maximal consistent subsets are collapsed as "clouds", the labels of edges show the path length from MIS/MCS to the top/bottom of the lattice.

7 Implementation and Benchmarks

We implemented black-box, white-box, and hybrid approach combining black-box / white-box approaches in EULER/X toolkit[3]. We use different answer set programming engine in our implementation, such as DLV, Potassco. We do benchmarks using both

[3] It is an open-source toolkit which can be downloaded in `http://bitbucket.org/eulerx/euler-project`

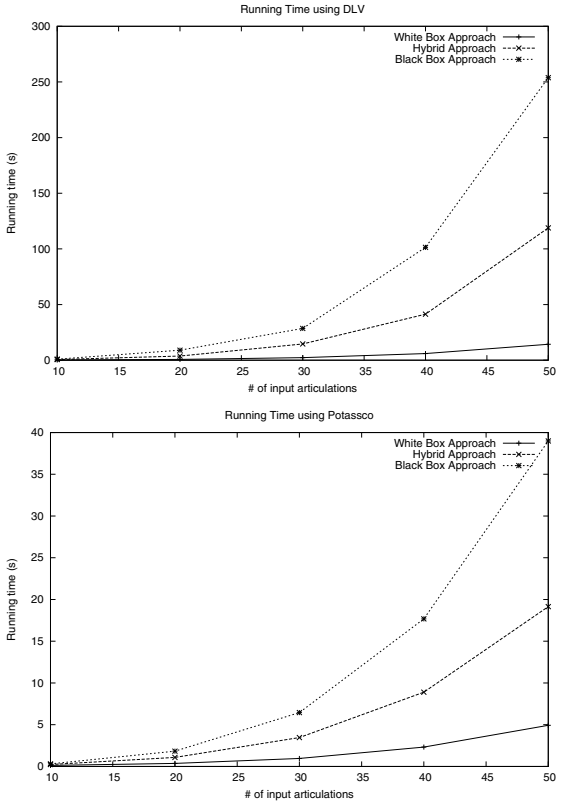

Fig. 8. Average running time using DLV (top) and Potassco (down) as underlying reasoners

real-world and artificial examples on all three approaches, of which white-box approach generates proof trees, the other two generate all diagnoses ($\mathcal{MIS}$). The two approaches we benchmarked to generate $\mathcal{MIS}$ are the Horridge's black-box approach [14] (HST Algorithm with Logarithmic Extraction Algorithm as its subroutine) and the hybrid approach.

We measure in our benchmarks average running time of different approaches using increasingly larger input datasets generated by an artificial inconsistent dataset generator. An artificial dataset includes two isomorphic taxonomies T_1 and T_2 each with n taxonomic concepts and a set of n articulations. Assuming φ is the isomorphism mapping[4] from T_1 to T_2, such that for two taxonomic concepts $T_1.A$ and $T_1.B$ such that $T_1.A\ isa\ T_1.B$, we have $\varphi(T_1.A)\ isa\ \varphi(T_1.B)$. We say that $T_1.C$ and $\varphi(T_1.C)$ is a *pair*, and these n articulations are between the n pairs. We keep the ratio of problematic articulations to be 10% in the artificial examples. All tests run on an 8-core, 32GB-memory Linux server. The average running time is shown as in Fig. 8.

[4] There could be many of such isomorphism mappings, but we only consider one of them.

Fig. 8 shows that using DLV and Potassco as underlying reasoners:

1. White-box approach runs much faster than either hybrid approach or black-box approach;
2. Hybrid approach runs around twice fast compared to black-box approach;

The reason is simple, white-box approach invokes the reasoner only once whereas the other two invokes reasoner multiple times. Also, hybrid approach runs faster than black-box approach because white-box approach significantly shrinks the candidate constraints for inconsistency, which results in the number invocations to reasoner decreases. However, notice that white-box approach generates proof trees and does not generate all diagnoses. The new hybrid approach improves the diagnosis generation significantly compared to the existing black-box approach.

8 Conclusion and Future Work

We discuss different approaches for general model-based diagnosis, including existing black-box approach [14], and new approaches proposed in this paper, white-box provenance approach, and hybrid approach combining black-box and white-box provenance. white-box provenance is a new approach which rewrite answer set programming rules (including safe, non-safe and with aggregates) to generate diagnosis proof trees. Hybrid approach combines both black-box and white-box provenance and generates all diagnoses. We implemented all these approaches in the application of EULER/X toolkit for taxonomy alignment. Benchmarks show that our hybrid approach runs twice as fast as the existing black-box approach of HST algorithm. Future work includes understanding the relation between the white-box diagnosis proof trees and $\mathcal{MIS}$ and optimizing the generation of $\mathcal{MIS}$.

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments on this paper. This work was supported in part by NSF awards IIS-1118088 and DBI-1147273.

References

1. Andersen, H.R., Hulgaard, H.: Boolean expression diagrams. In: Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, LICS 1997, pp. 88–98. IEEE (1997)
2. Baader, F., Suntisrivaraporn, B.: Debugging snomed ct using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: Proceedings of the International Conference on Representing and Sharing Knowledge Using SNOMED (KR-MED 2008). Citeseer, Phoenix (2008)
3. Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Hermenegildo, M.V., Cabeza, D. (eds.) PADL 2004. LNCS, vol. 3350, pp. 174–186. Springer, Heidelberg (2005)
4. Beckert, B., Gladisch, C.: White-box testing by combining deduction-based specification extraction and black-box testing. In: Gurevich, Y., Meyer, B. (eds.) TAP 2007. LNCS, vol. 4454, pp. 207–216. Springer, Heidelberg (2007)

5. Bonatti, P., Calimeri, F., Leone, N., Ricca, F.: Answer set programming. In: Dovier, A., Pontelli, E. (eds.) 25 Years of Logic Programming. LNCS, vol. 6125, pp. 159–182. Springer, Heidelberg (2010)
6. Chen, M., Yu, S., Franz, N., Bowers, S., Ludäscher, B.: Euler/x: A toolkit for logic-based taxonomy integration. In: 22nd Intl. Workshop on Functional and (Constraint) Logic Programming (WFLP), Kiel, Germany (2013)
7. de la Banda, M.G., Stuckey, P.J., Wazny, J.: Finding all minimal unsatisfiable subsets. In: Proceedings of the 5th ACM SIGPLAN International Conference on Principles and Practice of Declaritive Programming, pp. 32–43. ACM (2003)
8. Eiter, T., Gottlob, G.: Hypergraph transversal computation and related problems in logic and AI. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 549–564. Springer, Heidelberg (2002)
9. Eiter, T., Ianni, G., Krennwallner, T.: Answer set programming: A primer. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web. LNCS, vol. 5689, pp. 40–110. Springer, Heidelberg (2009)
10. Engel, C., Hähnle, R.: Generating unit tests from formal proofs. In: Gurevich, Y., Meyer, B. (eds.) TAP 2007. LNCS, vol. 4454, pp. 169–188. Springer, Heidelberg (2007)
11. Franz, N., Chen, M., Yu, S., Bowers, S., Ludäscher, B.: Names are not good enough: reasoning over taxonomic change in the andropogon complex. submitted for publication (2014)
12. Franz, N., Peet, R.: Perspectives: Towards a language for mapping relationships among taxonomic concepts. Systematics and Biodiversity 7(1), 5–20 (2009)
13. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 31–40. ACM (2007)
14. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in owl ontologies. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS (LNAI), vol. 5785, pp. 124–137. Springer, Heidelberg (2009)
15. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of owl dl entailments. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
16. Köhler, S., Ludäscher, B., Smaragdakis, Y.: Declarative datalog debugging for mere mortals. In: Barceló, P., Pichler, R. (eds.) Datalog 2.0 2012. LNCS, vol. 7494, pp. 111–122. Springer, Heidelberg (2012)
17. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32(1), 57–95 (1987)
18. Tan, J., Narasimhan, P.: Rams and blacksheep: Inferring white-box application behavior using black-box techniques. Technical report, Technical Report CMU-PDL-08-103. Carnegie Mellon University Parallel Data Laboratory (2008)
19. Thau, D., Bowers, S., Ludäscher, B.: Merging taxonomies under rcc-5 algebraic articulations. In: 2nd International Workshop on Ontologies and Information Systems for the Semantic Web, pp. 47–54. ACM (2008)
20. Thau, D., Ludäscher, B.: Reasoning about taxonomies in first-order logic. Ecological Informatics 2(3), 195–209 (2007)