A Logical Framework for Detecting Anomalies in Drug Resistance Algorithms

L. Caroprese
Dipartimento di Elettronica, Informatica e Sistemistica
Università della Calabria
Cosenza, Italy
caroprese@deis.unical.it

B. Ó Nualláin
Computational Science
Faculty of Science
University of Amsterdam,
The Netherlands
bon@uva.nl

P.M.A. Sloot
Computational Science
Faculty of Science
University of Amsterdam,
The Netherlands
p.m.a.sloot@uva.nl

E. Zumpano
Dipartimento di Elettronica, Informatica e Sistemistica
Università della Calabria
Cosenza, Italy
zumpano@deis.unical.it

ABSTRACT

Virology research is nowadays a discipline involving a broad number of researchers gathered in different institutes and cooperating on defined issues. An example of such an endeavor is the research tackling anti-HIV treatment problems [7] conducted within the Virolab project. The main objective of the ViroLab project is to develop a Virtual Laboratory for Infectious Diseases that facilitates medical knowledge discovery and decision support for HIV drug resistance. Large, high quality in-vitro and clinical patient databases which can be used to relate genotype to drug-susceptibility phenotype have become available. The core of the ViroLab Virtual Laboratory is a rule-based ranking system. More specifically, using a Grid-based service oriented architecture, Virolab vertically integrates the biomedical information from viruses (proteins and mutations), patients and literature (drug resistance experiments), resulting in a rule-based decision support system for drug ranking. This paper is a contribution to virologists, epidemiologists and clinicians in medical knowledge discovery and decision support. The final aim is reasoning on the properties of algorithm modeling the interaction among drugs and HIV virus and detecting its anomalies such as rules that can never be satisfied and subset of rules that are in contradiction.

Categories and Subject Descriptors

H.1.m [**Information Systems**]: Models and Principles—*Miscellaneous*; H.4.m [**Information Systems**]: Information Systems Application—*Miscellaneous*; J.3 [**Computer Applications**]: Life and Medical Science—*Health*; J.3 [**Computer Applications**]: Life and Medical ScienceMedical information systems

General Terms

Theory, Algorithms

Keywords

Bio-medical databases, Anomalies in drug resistance algorithms, Logic programming, Rule-base system, Decision support system, Knowledge discovery

1. INTRODUCTION

In future years, genetic information is expected to become increasingly significant in many areas of medicine. This expectation comes from the recent and anticipated achievements in genomics, which provide an unparalleled opportunity to advance the understanding of the role of genetic factors in human health and disease, to allow more precise definition of the non-genetic factors involved, and to apply this insight rapidly to the prevention, diagnosis and treatment of diseases.

The main objective of the ViroLab project is to develop a Virtual Laboratory for Infectious Diseases that facilitates medical knowledge discovery and decision support for, e.g., (Human Immunodeficiency Virus) HIV drug resistance [9, 8, 6, 4]. In this scope the laboratory is prepared to support virologists, epidemiologists and clinicians investigating the HIV virus and the possibilities of treating HIV-positive patients. HIV is a retrovirus that can lead to Acquired Immunodeficiency Syndrome (AIDS), a condition in humans in which the immune system begins to fail, leading to life-threatening opportunistic infections. There is no cure for AIDS. However, there are new treatments that can slow down its progression.

Once HIV turns into AIDS the risk of death is much higher. Even so, the risk varies a lot from patient to patient. Some AIDS patient die shortly after being diagnosed, while others live 12 more years or longer. Fortunately, patients with AIDS are living longer as new treatments are discovered.

Large, high quality in-vitro and clinical patient databases which can be used to relate genotype to drug-susceptibility phenotype have become available. Such data comes also from a considerable portion of patients for whom drugs fail to completely suppress the virus resulting in the rapid selection of drug-resistant HIV and loss of drug efficacy. Therefore, the primary goal of the data analysis,

in Virolab, is to identify patterns of mutations associated with resistance to antiviral drugs and to predict the degree of in-vitro or in-vivo sensitivity to available drugs from an HIV genetic sequence.
HIV differs from other viruses in that it has very high genetic variability. This diversity is a result of its fast replication cycle, with the generation of 109 to 1010 virions every day, coupled with a high mutation rate of approximately 3 x 10^5 per nucleotide base per cycle of replication and recombinogenic properties of reverse transcriptase. This complex scenario leads to the generation of many variants (mutations) of HIV in a single infected patient in the course of one day.
The core of the ViroLab Virtual Laboratory is a rule-based ranking system. More specifically, using a Grid-based service oriented architecture, Virolab vertically integrates the biomedical information from viruses (proteins and mutations), patients (e.g. viral load) and literature (drug resistance experiments), resulting in a rule-based decision support system for drug ranking.
The knowledge about the interactions among drugs and the HIV virus is encoded into sets of rules.
The drug rules are used to associate to a particular drug an effectiveness level over a set of mutations given in input.
A set of rules is called algorithm. *An algorithm allows to find the right combination of drugs for a given set of mutations extracted from a patient.* Therefore, in the presence of a patient affected by HIV, it is possible to use the knowledge encoded in the algorithm to establish the right medical care for that particular kind of mutation. Note that, the validity of the first care is essentially for life expectation of the patient, that is the time he/she will live strongly depends on the first cocktail of drugs, if this is not correct the virus became stronger as new mutations will occur and it will be more difficult to control.
This paper is a contribution to virologists, epidemiologists and clinicians in medical knowledge discovery and decision support. The final aim is reasoning on the properties of algorithm modeling the interaction among drugs and HIV virus and detecting its anomalies such as rules that can never be satisfied and set of rules that are in contradiction.
More specifically, the paper defines techniques that allow to deeply investigate the properties of each observation and its relationship with other observations. Virologist would benefit from knowing whether: i) his/her observation makes sense, technically if the condition in the body of the drug rule can occur; ii) the result of his/her (in-vivo or in-vitro) experiments is in contradiction with other results from colleagues, technically if there exists a set of drug rules which is contradictory.
This paper provides a theoretical investigation of the previous tasks and contributes to the virology research by providing efficient tools that, given an algorithm encoding the knowledge about the interaction among drugs and HIV drugs, perform the important issue of discovering its properties. More specifically, it will propose a technique that given an algorithm identifies unsatisfiable rules, that is rules whose body is always false and contradictory set of rules, that is set of rules that disagree of the final action to perform in the presence of the same set of mutations.

2. MODELING THE INTERACTION AMONG DRUGS AND HIV VIRUS

The Algorithm Specification Interface (ASI) [1] is a platform developed for coding genotypic interpretation algorithms. ASI makes it possible for drug resistance experts to develop and test genotypic interpretation algorithms without the assistance of a computer programmer, thereby allowing HIV drug resistance experts to focus on developing, testing, and modifying rules rather than on developing software to encode algorithms.
At the moment there exist three main algorithms using ASI technology:

- ANRS
- HIVDB [2]
- Rega Institute [3]

Drug resistance algorithms can be specified in an XML document that adheres to a particular XML Document Type Definition (DTD): the ASI DTD [5].
At the heart of an algorithm is a *drug rule*, used to associate to a particular drug an effectiveness level over a set of mutations given in input.

EXAMPLE 1. The following object is a drug rule for the drug 'TDT'.

```
<DRUG>
  <NAME>TDT</NAME>
  <RULE>
    <CONDITION>
     SELECT ATLEAST 1 FROM (65MR, 69i)
    </CONDITION>
    <ACTIONS>
     <LEVEL>3</LEVEL>
    </ACTIONS>
  </RULE>
</DRUG>
```

In this example, for the drug TDT the algorithm designer would like the drug to receive a level of 3 (the designer would have previously specified what was meant by level 3 - in Rega algorithm it means "Drug is inactive") if either the mutation 65MR is present, that is the amino acid M or R is present at position 65, or an insertion is present at position 69 . □

As shown by the previous statement the language is very much fairly self-explanatory as it reads similarly to an English sentence.
In general the *condition* is a conjunction of positive and negative assertions defining a set of properties that a a given input set of mutations could have. In the rest of this section, we first define the concept of input set, and then present the structure of these assertions. Let AS be the set containing the amino acid symbols and the symbols 'i' and 'd' reporting respectively an insertion and a deletion.

DEFINITION 1. *An* input set *is a non empty set whose elements are of the form PA where P is an integer (it represents the position of the mutation) and $A \in AS$ (it represents an amino acid an insertion or a deletion).*

An *assertion* can be either a *positive assertion* or a *negative assertion*.

DEFINITION 2. *A* positive assertion *is of one of the following forms:*

1. MUTATION*:*
   It is of the form:
   $PA_1 \dots A_n$
   where P is an integer and $A_i \in AS$, for $i \in [1..n]$ (e.g. 67G, 54i. 219EQ). This assertion is satisfied by an input set S if S contains at least an element in $\{PA_1, \dots, PA_n\}$ (i.e. $S \cap \{PA_1, \dots, PA_n\} \neq \emptyset$).

2. INFINITE INTERVAL:
It is of the form:
SELECT ATLEAST l
FROM $(M_1, \ldots, M_n)$
where n *is a positive integer,* $0 < l \leq n$ *and* M_i*, for* $i \in [1..n]$*, is a mutation.*
This assertion is satisfied by an input set S *if* S *satisfies at least* l *mutations belonging to* $\{M_1, \ldots M_n\}$.

3. FINITE INTERVAL:
It is of the form:
SELECT ATLEAST l *AND NOTMORETHAN* m
FROM $(M_1, \ldots, M_n)$
where l *and* m *are positive integers,* $0 < l \leq m \leq n$ *and* M_i*, for* $i \in [1..n]$*, is a mutation. This assertion is satisfied by an input set* S *if* S *satisfies at least* l *and not more than* n *mutations belonging to* $\{M_1, \ldots M_n\}$.

DEFINITION 3. *A* negative assertion *is of the form: NOT* α*, where* α *is a positive assertion. This assertion is satisfied by an input set* S *if* S *does not satisfy* α.

Note that, a mutation $PA_1 \ldots A_n$ denotes the presence in position P of at least one among the amino acids/insertion/ deletion $\{A_1 \ldots A_n\}$. Roughly speaking, in position P more than one of these values can be present. From a biological point of view, the simultaneous presence of more values states for more mutations of the HIV virus.

EXAMPLE 2. Some examples of assertions in the ASI language are the following:

- `151M`
- `SELECT ATLEAST 3`
  `FROM (69i, 151M, 65NR, 74IV)`
- `SELECT ATLEAST 2 AND NOTMORETHAN 2`
  `FROM (69i, 151M)`
- `215FY AND NOT 184VI`
- `SELECT ATLEAST 1`
  `FROM (65NR, 74IV) AND 184IV` □

As previously discussed, the kind of positive assertions that can be present in the body of a drug rule are either mutation or finite intervals or infinite intervals. We now observe that, without loss of generality, we can consider just finite intervals, as mutations and infinite intervals can be modeled by finite intervals. More specifically,

- The mutation:
  $PA_1, \ldots, A_n$ can be modeled by the finite interval
  SELECT ATLEAST 1 AND NOTMORETHAN n
  FROM $(PA_1, \ldots, PA_n)$.
- The infinite interval:
  SELECT ATLEAST l
  FROM $(M_1, \ldots, M_n)$
  can be modeled by the finite interval:
  SELECT ATLEAST l AND NOTMORETHAN n
  FROM $(M_1, \ldots, M_n)$.

3. A NEW LANGUAGE

In the previous section we have shown that each assertion can be modeled as a (positive/negative) finite interval. Starting from this result, in this section we present a new language, , that will be used to model finite intervals and therefore can be used to model assertions in the bodies of drug rules. The language will be more suitable for our theoretical analysis.

3.1 Syntax

The syntax of the language is described in the following using BNF notation:

⟨`Interval`⟩	::=	⟨`PositiveInterval`⟩| ⟨`NegativeInterval`⟩
⟨`PositiveInterval`⟩	::=	⟨`Position`⟩⟨`AminoAcid`⟩| '['⟨`SetOfIntervals`⟩']'| '['⟨`SetOfIntervals`⟩']'$^{\langle \text{Integer} \rangle}_{\langle \text{Integer} \rangle}$
⟨`NegativeInterval`⟩	::=	$\overline{\langle \texttt{PositiveInterval} \rangle}$
⟨`SetOfIntervals`⟩	::=	⟨`Interval`⟩[';' ⟨`SetOfIntervals`⟩]
⟨`Integer`⟩	::=	⟨`Digit`⟩|⟨`Integer`⟩⟨`Digit`⟩
⟨`AminoAcid`⟩	::=	'A'|...|'i'|'d'
⟨`Position`⟩	::=	'1'|'2'|...|'500'
⟨`Digit`⟩	::=	'0'|'1'|...|'9'

A positive interval PA is also called *atom*. A *literal* is an atom PA or its negation $\overline{PA}$. Observe that the definition of the positive inteval is recursive: It is an atom or it is of the form $[I_1, \ldots, I_n]^{max}_{min}$ where I_i, for $i \in [1..n]$, is a positive or negative interval. A negative interval is of the form $\overline{I}$, where I is a positive interval.

EXAMPLE 3. The following strings belong to the language :

$[41L]^1_1$

$41L$

$[41L; 41M]^2_1$

$\overline{70R}$

$[65R; 70I]^2_1$

$[67G; 67N]^2_1$

$[[70E; 70G]^2_1; 151M]^2_1$

$\overline{[184I; 184V]^2_1}$

$[48A; 84V]^2_1$

$[48A; 84V]^1_1$ □

In the language an interval $[I_1; I_2; \ldots; I_n]^1_1$ is equivalent to $[I_1; I_2; \ldots; I_n]$ and $[PA]^1_1$ is equivalent to PA.

DEFINITION 4. *A finite interval* $I = [I_1; I_2; \ldots; I_n]^{max}_{min}$ *is* well-defined *if* $min \leq n \leq max$. □

In the rest of the paper we will always refer to well-defined finite intervals.

3.2 Semantics

The truth value of an interval w.r.t. an input set is defined recursively.

DEFINITION 5. Given an input set S,

- an atom PA is *true* w.r.t S is $PA \in S$ and *false* otherwise;
- a positive interval $I = [I_1; I_2; \ldots; I_n]^{max}_{min}$ is *true* w.r.t. S if at least *min* and not more than *max* elements belonging to $\{I_1, I_2, \ldots, I_n\}$ are *true* w.r.t. S and *false* otherwise.
- a negative interval $\overline{I}$ is *true* w.r.t. S if I is *false* w.r.t. S, and *false* otherwise.
- a conjunction of intervals $I_1, \ldots, I_n$ is *true* w.r.t. S if I_i is *true* w.r.t. S for each $i \in [1..n]$. □

EXAMPLE 4. Let's provide some example:

- $[67G; 67N]^2_1$ is true w.r.t. $S = \{67G, 67N\}$
- $[67G; 67N]^2_1$ is true w.r.t. $S = \{67G, 40B, 65R\}$
- $[67G; 67N]^2_1$ is true w.r.t. $S = \{67N\}$ □

Obviously, each finite interval in format can be translated into a correspondent assertion in the ASI language and vice-versa. The following table reports the example of some translations from into ASI.

Ł	ASI
$41L$	41L
$\overline{70R}$	NOT 70R
$[65R; 70I]^2_1$	SELECT ATLEAST 1 AND NOTMORETHAN 2 FROM (65R,70I)
$[67G; 67N]^2_1$	67GN
$[[70E; 70G]^2_1; 151M]^2_1$	SELECT ATLEAST 1 AND NOTMORETHAN 2 FROM (70EG,151M)
$\overline{[184I; 184V]^2_1}$	NOT 184IV
$[48A; 84V]^2_1$	SELECT ATLEAST 1 FROM (48A,84V)

Let us now provide further details on the above translations.
The first interval - $41L$ - is a shorthand for $[41L]^1_1$ and corresponds to the assertion 41L in ASI format. The second interval - $\overline{70R}$ - requires the absence of the amino acid R in the position 70 and corresponds to NOT 70R. The third interval - $[65R; 70I]^2_1$ - requires the presence of at least one between {65R,70I} and corresponds to the ASI assertion SELECT ATLEAST 1 AND NOTMORETHAN 2 FROM (65R,70I). The fourth interval - $[67G; 67N]^2_1$ - corresponds to the asssertion 67GN in the ASI format. In general, the ASI assertion $PA_1 \ldots A_n$ is equivalent to the interval $[PA_1; \ldots; PA_n]^n_1$. The fifth interval - $[[70E; 70G]^2_1; 151M]^2_1$ - is a *nested* finite interval, that is a finite interval that contains a further finite interval, and corresponds to the ASI assertion - SELECT ATLEAST 1 AND NOTMORETHAN 2 FROM (70EG, 151M). The sixth negative interval - $\overline{[184I; 184V]^2_1}$- requires the absence of both the amino acid I and V in position 184 and is translated into NOT 184IV in the ASI format. Finally, the last interval - $[48A; 84V]^2_1$ - is equivalent to the assertion

SELECT ATLEAST 1 AND NOTMORETHAN 2
FROM (48A,84V)

or more synthetically

SELECT ATLEAST 1 FROM (48A,84V).

We observe that, given an interval $I = [I_1; \ldots; I_n]^{max}_{min}$, the order of its sub-intervals $I_1, \ldots, I_n$ is immaterial. Therefore, instead of I we can use any other interval $I_{\pi(\cdot)} = [I_{\pi(1)}; \ldots; I_{\pi(n)}]^{max}_{min}$ where $\pi(\cdot)$ is a permutation of the sub-intervals $I_1, \ldots, I_n$.

DEFINITION 6. *An* rule *is of the form:*

$$\texttt{D : ACTION} \leftarrow \texttt{CONDITION} \quad (1)$$

where `CONDITION` *is a conjunction of intervals, defining a set of properties of an input set S, and* `ACTION` *is an integer associating to the drug D an effectiveness level over S.* □

Obviously, each rule the form (1) is equivalent to a an drug rule in ASI format and vice-versa.

EXAMPLE 5. The drug rule in ASI format, reported in Example 1, corresponds to the rule:
$TDT : 6 \leftarrow [[65M; 65R]^2_1; 69i]^2_1$. □

3.3 Unpacking Process

Why the need of a new language? As we will explain, each rule r of the form (1) is equivalent to a set $R = \{r_1, \ldots, r_n\}$ of rules whose bodies are conjunctions of literals. A rule is unsatisfiable when its body is always false (i.e. it is false w.r.t. each possible input set S). This happens when it requires the presence and the absence of an amino acid in a certain position at the same time. Clearly, the rule R is unsatisfiable if each rule in R is unsatisfiable and it is easy to check whether a rule whose body is a conjunction of literals is unsatisfiable: It is sufficient to verify if the body contains two literals of the form A and $\overline{A}$.
We introduce a shorthand for a set of rules having the same head, allowing the disjunction in the body. In particular a set of rules

$$R = \{D : A \leftarrow C_1, \ldots, D : A \leftarrow C_n\}$$

will be denoted a *generalized rule* of the form:

$$D : A \leftarrow C_1 \vee \cdots \vee C_n.$$

We call the procedure that allows to obtain from an rule $D : A \leftarrow C$ the equivalent generalized rule $D : A \leftarrow C_1 \vee \cdots \vee C_n$ – where C_i, for $i \in [1..n]$, is a conjunction of literals – the *unpacking process*. Using the language we can easily derive and analyze this process.

EXAMPLE 6. Consider the rule

$$r = TDT : 6 \leftarrow [67G; 67N]^2_1$$

Its body is true if at least one mutation in $\{67G, 67N\}$ is present in the input set.Therefore, it is true if either 67G is true and 67N is false - i.e. $67G, \overline{67N}$ is true - or 67N is true and 67G is false - i.e. $\overline{67G}, 67N$ is true - or both of them are true - $67G, 67N$ is true. It follows that, r is equivalent to

$$TDT : 6 \leftarrow 67G, \overline{67N} \vee \overline{67G}, 67N \vee 67G, 67N. \quad \square$$

For the sake of simplicity, in the rest of this section, we will consider just the body of the rules (the only part that is unpacked).

DEFINITION 7. [ONE STEP UNPACKING OPERATOR] *Let I be the interval $[I_1;\ldots;I_n]^{max}_{min}$. We define the* one step unpacking operator *$U(\cdot)$:*

$$U(I) = \bigvee_{i=1}^{h} [I_{\pi(1)};\ldots;I_{\pi(m)};\overline{I_{\pi(m+1)}};\ldots;\overline{I_{\pi(n)}}]$$

where $min \leq m \leq max$ and $\pi(\cdot)$ is a permutation of the intervals $I_1,\ldots,I_n$.

$$U(\overline{I}) = [I_1;\ldots;I_n]_0^{min-1} \vee [I_1;\ldots;I_n]_{max+1}^{n}$$

Given a conjunction of intervals $C = I^1,\cdots,I^t$ we define $U(C)$ as the expansion of (i.e. the disjunction of conjunctions obtained from) $U(I^1),\ldots,U(I^t)$.
Given a disjunction of intervals $D = I^1 \vee \cdots \vee I^t$ we define $U(C) = U(I^1) \vee \cdots \vee U(I^t)$. □

PROPOSITION 1. *Let $I = [I_1;\ldots;I_n]^{max}_{min}$ and $U(I) = \bigvee_{i=1}^{h} I_i$ its unpacked version. Then $h = \sum_{k=min}^{max} \binom{n}{k}$.*

Proof: The number of subsets of cardinality k that can be extracted from a set of n elements is equal to $\binom{n}{k}$. Given an interval $I = [I_1;\ldots;I_n]^{max}_{min}$, the value of k ranges from min to max. Therefore the number of valid-operands are $\sum_{k=min}^{max} \binom{n}{k}$. □

PROPOSITION 2. *Let $C = I^1,\cdots,I^t$, where $I^i = [I^i_1,\ldots,I^i_{n_i}]^{max_i}_{min_i}$ and $i \in [i..n]$, and let $U(C) = \bigvee_{i=1}^{h} J_i$, where J_i are the conjunctions obtained by applying the $U(\cdot)$ operator, $i \in [1..h]$. Then $h = \Pi_{i=1}^{t} \sum_{k=min_i}^{max_i} \binom{n_i}{k}$.*

Proof: Straightforward from Proposition 1. □

THEOREM 1. *Let I be an interval and S an input set. Then, I and $U(I)$ are equivalent that is I is true w.r.t. S iff $U(I)$ is true w.r.t. S.* □

EXAMPLE 7. Given $I = [46I;10F;71V]^3_2$ we have that:

$$U(I) = 46I, 10F, \overline{71V} \vee 46I, \overline{10F}, 71V$$

$$\vee \overline{46I}, 10F, 71V \vee 46I, 10F, 71V.$$

Observe that, the number of conjunctions of $U(I)$ is $\sum_{k=2}^{3} \binom{3}{k} = 4$. □

An important property of the one step unpacking operator is that if $I = [I_1;\ldots;I_n]$ then $U(I) = I$. It follows that given a disjunction $D = I^1 \vee \cdots \vee I^t$ s.t. $I^i = [I^i_1;\ldots;I^i_{n_i}]$ for $i \in [1..n]$ we have $U(D) = D$.

DEFINITION 8. [UNPACKING OPERATOR] *Given a conjunction of intervals $C = I_1,\ldots,I_n$ we define:*

- $U^0(C) = U(C)$
- $U^{i+1}(C) = U(U^i(C))$. □

PROPOSITION 3. [FIX POINT] *For each conjunction of intervals $C = I_1,\ldots,I_n$ there exists a least index i s.t. $U^i(C) = U^{i+1}(C)$. We set $U^\infty(C) = U^i(C)$ (*unpacking operator*).* □

Therefore, the operator $U^\infty(\cdot)$ unpack a conjunction of intervals into a disjunction of conjunctions of literals.

EXAMPLE 8. We report some examples of the unpacking process:

$$U^\infty([65R;70I]^2_1) = 65R, \overline{70I} \vee 70I, \overline{65R} \vee 65R, 70I$$

$$U^\infty([[46I;10F;71V]^3_2;70K]) =$$
$$\overline{46I}, 10F, \overline{71V}, 70K \vee 46I, \overline{10F}, 71V, 70K \vee$$
$$\overline{46I}, 10F, 71V, 70K \vee 46I, 10F, 71V, 70K$$

$$U^\infty([[70E;70G]^2_1;151M]^2_1) =$$
$$\overline{70E}, 70G, \overline{151M} \vee 70E, 151M \vee$$
$$\overline{70G}, 151M \vee \overline{70E}, \overline{70G}, 151M \vee 70E, \overline{70G}, \overline{151M} \vee$$
$$\overline{70E}, \overline{151M} \vee 70E, \overline{70G}, 151M \vee 70E, 70G, \overline{151M}$$
$$\vee 70E, 70G, 151M.$$ □

Tree Representation

The unpacking process performed by the operator $U^\infty(\cdot)$ can be represented by a tree.

DEFINITION 9. *Given a conjunction of intervals $C = I_1,\ldots,I_n$ we define the* unpacking tree *$G(C) = \langle N, E\rangle$ as:*

- $C \in N$
- *Let $X \in N$ and $U(X) = \bigvee_{i=1}^{n} X_i$ the unpacked version of X. Then $X_i \in N$ and $(X, X_i) \in E$ for each $i \in [1..n]$.* □

EXAMPLE 9. The unpacking process of the body of the rule reported in Example 6 ($[67G, 67N]^2_1$) is represented by the tree in Fig. 1.

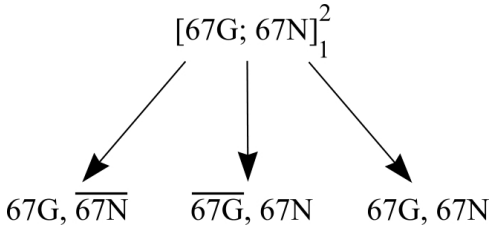

Figure 1: $[67G, 67N]^2_1$

□

The following example shows a more complex unpacking tree.

EXAMPLE 10. The unpacking process of the interval:

$$[[70E;70G]^2_1;151M]^2_1$$

corresponding to the ASI assertion

SELECT AT LEAST 1 AND NOTMORETHAN 2
FROM (70EG, 151M)

is represented by the unpacking tree reported in Fig. 2.

□

Thus, the body C of each drug rule is equivalent to the disjunction of the leaves of the unpacking tree $G(C)$.

4. ANOMALIES IN DRUG RESISTANCE ALGORITHMS

This section provides a theoretical investigation of the problem related to the detection of anomalies in drug resistance algorithm. In particular:

- *given a drug rule we want to check whether it is unsatisfiable* that is its body cannot be satisfied by any input set. This happens when it requires the presence and the absence of an amino acid in a certain position at the same time. An unsatisfiable rule could be introduced in an algorithm by mistake or because of some anomaly in the experiment which it is derived from. From a logical point of view this rule can be deleted, whereas form a medical point of view this analysis detects anomalies in medical knowledge base.

- *we are interested in checking whether an algorithm contains contradictory sets of rules*. Two rules are contradictory if they refer to the same drug D and for some input set they return two different effectiveness levels for D.

The following two subsections will face these two problems and provide a solution that allows to identify the two types of anomalies and thus can be used to optimize the code of a given algorithm by removing inconsistencies.

4.1 Unsatisfiable rules

In this section we analyze the problem of checking whether the body of a drug rule is unsatisfiable. If an interval is a conjunction of literals it is trivial to check whether it is satisfiable or not.

DEFINITION 10. *A conjunction $C = L_1, \ldots, L_n$ of literals is* unsatisfiable *if there exist $i, j \in [1..n]$ such that $L_i = \overline{L_j}$.* □

Now we are ready to define an unsatisfiable body (conjunction of intervals) and an unsatisfiable rule.

DEFINITION 11. [UNSATISFIABLE RULE]. *Let $C = I_1, \ldots, I_n$ a conjunction of intervals and $U^\infty(C) = \bigvee_{i=1}^n C_i$ its unpacked version. C is* unsatisfiable *if C_i is* unsatisfiable *for each $i \in [1..n]$. An rule $D : A \leftarrow C$ is unsatisfiable if C is unsatisfiable.* □

An equivalent way to check whether a conjunction C is unsatisfiable is by using the tree representation of its unpacking process. C is unsatisfiable if each leaf of the tree $G(C)$ is unsatisfiable.
Although the operator $U^\infty(\cdot)$ builds an unpacking tree G using a breadth-first technique (level by level), G can be built using a depth-first technique that allows to reach the leaves of the tree as soon as possible. Although the number of leaves is exponential in the number of original mutations occurring in the body of a drug rule, this technique makes the checking process pretty fast, due to two different reasons:

- in the case the body is satisfiable it is sufficient to generate just one path from the root to a satisfiable leaf;

- in the case the body is unsatisfiable we do not need to complete the unpacking process by generating the whole search space as the unpacking process of a path can be stopped as soon as a node is recognized as unsatisfiable (the conjunction even not completely unpacked contains two literals of the form L and $\overline{L}$).

EXAMPLE 11. Fig. 3 shows how the unpacking tree is used to verify that the interval $[46I; 10F; 71V]_2^3, \overline{71V}, [46I; 10F]_1^1$, corresponding to the ASI assertion *SELECT AT LEAST 2 AND NOTMORETHAN 3 FROM (46I, 10F, 71V) AND NOT 71V AND SELECT AT LEAST 1 AND NOTMORETHAN 1 FROM (46I, 10F)* is unsatisfiable.

□

Body simplification.
The unpacking process can be used to simplify the body of a rule by deleting its unsatisfiable portion. Let us consider a body $C = I_1, \ldots, I_n$. We know that it is equivalent to the disjunction $U^\infty(C) = \bigvee_{i=1}^n C_i$. Now, let us suppose, without loss of generality, that the first k conjunctions are satisfiable and the other $n - k$ are unsatisfiable. Then, C is equivalent to $U^\infty(C) = \bigvee_{i=1}^k C_i$ as the unsatisfiable conjunctions can be dropped.

EXAMPLE 12. Let us consider the conjunction $C = [67G, 68N]_1^1, \overline{68N}$. Its unpacking version is $U(C) = 67G, \overline{68N} \vee 67G, 68N, \overline{68N}$. As the $67G, 68N, \overline{68N}$ is unsatisfiable, C is equivalent to $67G, \overline{68N}$. □

4.2 Contradictory set of rules

The problem of finding sets of contradictory rules is the following: *does an algorithm lead to two or more different levels of effectiveness of a drug for some input set of mutations?* The solution to this problem can be easily derived from the solution of the previous problem, related to the management of unsatisfiable rules.
Before formally face the problem, we provide the intuition at the basis of the proposal using a simple example.

EXAMPLE 13. Consider the following two rules of the language both referring to the drug AZT:

$AZT : 6 \leftarrow [151M; 69i]_1^2$
$AZT : 4 \leftarrow [215F; 215Y]_1^2, \overline{[184I; 184V]_1^2}$

If the input set contains the mutations 151M and 215F and does not contain the mutations 184I and 184V the bodies of both rules are true. In this case, the algorithm assigns to the drug rules both effectiveness levels 6 and 4 at the same time. Therefore, the two rules are contradictory. □

Therefore, in the presence of two different rules referring to the same drug, an anomaly has to be detected if both rules are satisfied. Let's firstly define the concept of coherent and contradictory couples of rules.

DEFINITION 12. *Given two rules:*
$r_1 : DRUG : Level_1 \leftarrow Body_1$
$r_2 : DRUG : Level_2 \leftarrow Body_2$
the couple (r_1, r_2) is coherent *if either*

- $Level_1 = Level_2$ *or*
- *the conjunction $C = Body_1, Body_2$ is unsatisfiable.*

(r_1, r_2) is contradictory *if it is not coherent.* □

Two rules that refers to the same drug are coherent if they report the same effectiveness level or - if the levels are different - their bodies cannot be satisfied at the same time. This test can be performed by building the conjunction C of the two bodies and verifying whether C is unsatisfiable.
Therefore, given an algorithm modeling the interaction among drugs and HIV virus (e.g. Rega) we can generalize the technique so that

we can look for a contradictory set of rules. In this case we have to cluster the rules w.r.t. the drugs and check whether each cluster is coherent/contradictory.

DEFINITION 13. [CONTRADICTORY SET OF RULES] Given a cluster CR of rules:
s
$r_1 : DRUG : Level_1 \leftarrow Body_1$
$r_2 : DRUG : Level_2 \leftarrow Body_2$
...
$r_n : DRUG : Level_n \leftarrow Body_n$

CR is *coherent* if each (r_i, r_j), with $i \neq j, i, j \in [1..n]$, is coherent. CR is *contradictory* if it is not coherent. □

An algorithm is satisfiable if each cluster is satisfiable. Let us present a complete example directly derived from the Rega algorithm.

EXAMPLE 14. Let us consider the cluster of Example 13. It is contradictory as the conjunction

$$[151M; 69i]_1^2, [215F; 215Y]_1^2, \overline{[184I; 184V]_1^2}$$

obtained from the bodies of the rules is satisfiable. □

5. DISCUSSION AND FUTURE WORK

This paper is a contribution to virologists, epidemiologists and clinicians in medical knowledge discovery and decision support. The final aim is supporting them in the important task of modeling 'correct' information derived from in-vivo and in-vitro experiments.
More specifically, we have conducted a theoretical investigation of drug resistance rules and proposed a technique whose aim is retrieving unsatisfiable rules and contradictory set of rules. We have faced the above mentioned anomalies from a logical perspective. A drug rule is unsatisfiable if a logical inconsistency occurs: its body requires, at the same time, the presence and the absence of a value in a particular position (Ex. $68R, \overline{68R}$). Anyhow, besides logical anomalies also biological inconsistencies can lead to unsatisfiable rule. A biological inconsistency is a combination of assertions which are incompatible from a biological perspective.
Our framework can be easily extended to manage biological anomalies. We assume that the biological inconsistencies are modeled by a set $BI = \{B_1, \ldots, B_n\}$ provided 'off-line' by the experts. Each $B_i - i \in [1..n]$ – is a conjunction of literals that cannot be true for biological reasons.
Now we can *update* our definition of *unsatisfiability*. A conjunction C of literals is unsatisfiable if:

- it is *logically unsatisfiable* (it contains two literals A and $\overline{A}$) or
- it is *biologically unsatisfiable* (it contains at least a conjunction $B_i \in BR$.

We are currently developing a Knowledge Optimizer (KR) of the medical knowledge about interactions among drugs and virus mutations, that implements the approach for the management of logical anomalies here proposed.

6. ACKNOWLEDGMENTS

The authors would like to thank the ViroLab consortium. This research was supported by the European ViroLab grant INFSO-IST-027446. We thank Sergio Flesca for his valuable insight and contributions to this paper.

7. REFERENCES

[1] Algorithm specification interface (asi). *http://hivdb.stanford.edu/pages/asi/*.
[2] Hiv drug resistance database - stanford university. *http://hivdb.stanford.edu/*.
[3] Retrogram. *http://www.openclinical.org/aisp_retrogram.html*.
[4] The virolab project. *http://www.virolab.org/*.
[5] Xml. *http://www.w3.org/XML/*.
[6] P.M.A. Sloot. Virolab: from the molecule to the man. *eStrategies Projects*, (4):53–55, 2008.
[7] P.M.A. Sloot, P. Coveney, G. Ertaylan, V. Muller, C. Boucher, and M.T. Bubak. Hiv decision support: From molecule to man. *Phil. Trans. R. Soc. A*, 367(1898), 2009.
[8] P.M.A. Sloot, P.V. Coveney, M.T. Bubak, A.M. Vandamme, B. Ó Nualláin, D. van de Vijver, and C.A.B. Boucher. Virolab: A collaborative decision support system in viral disease treatment. *Reviews in Antiviral Therapy, Virology Education*, 3:4–7, 2008.
[9] P.M.A. Sloot, A. Tirado-Ramos, I. Altintas, M.T. Bubak, and C.A.B. Boucher. From molecule to man: Decision support in individualized e-health. *IEEE Computer, (Cover feature)*, 39(11):40–46, 2006.

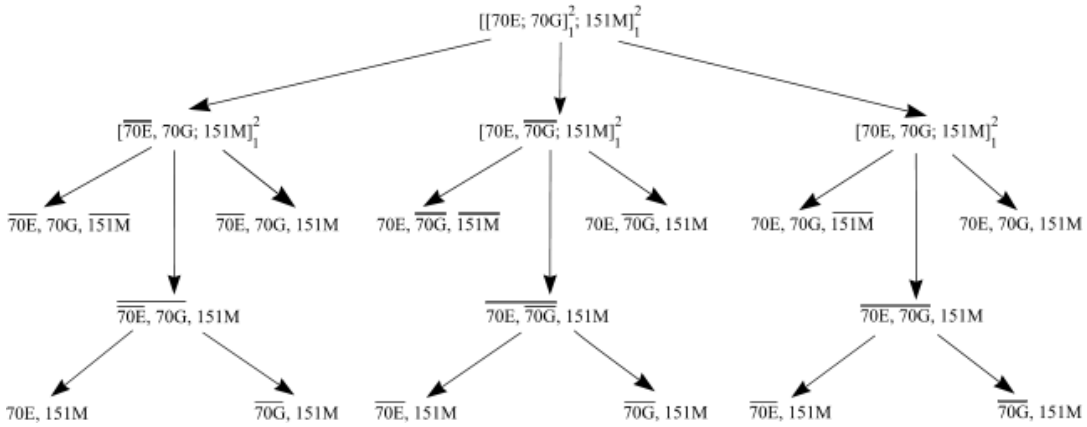

Figure 2: $[70EG; 151M]_1^2 \equiv$ **SELECT AT LEAST 1 AND NOTMORETHAN 2 FROM (70EG, 151M)**

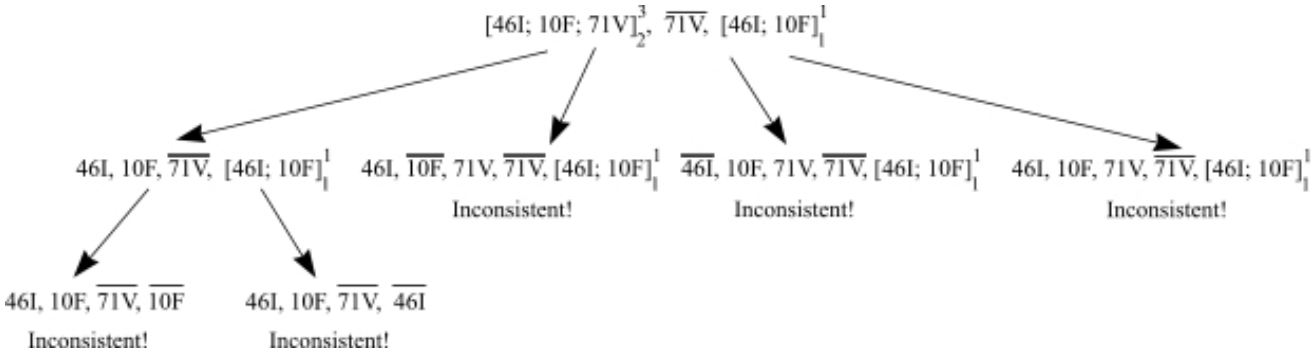

Figure 3: **An unsatisfiable body**