Distance Between Herbrand Interpretations: a measure for approximations to a target concept

Shan-Hwei Nienhuys-Cheng
cheng@cs.few.eur.nl

Department of Computer Science, H4-19
Erasmus University Rotterdam
P.O. Box 1738, 3000DR, Rotterdam, the Netherlands

Abstract. We can use a metric to measure the differences between elements in a domain or subsets of that domain (i.e. concepts). Which particular metric should be chosen, depends on the kind of difference we want to measure. The well known Euclidean metric on $\Re^n$ and its generalizations are often used for this purpose, but such metrics are not always suitable for concepts where elements have some structure different from real numbers. For example, in (Inductive) Logic Programming a concept is often expressed as an Herbrand interpretation of some first-order language. Every element in an Herbrand interpretation is a ground atom which has a tree structure.
We start by defining a metric d on the set of expressions (ground atoms and ground terms), motivated by the structure and complexity of the expressions and the symbols used therein. This metric induces the Hausdorff metric h on the set of all sets of ground atoms, which allows us to measure the distance between Herbrand interpretations. We then give some necessary and some sufficient conditions for an upper bound of h between two given Herbrand interpretations, by considering the elements in their symmetric difference.
Keywords: metric space, inductive logic programming, logic programming.

1 Introduction

An important problem in Inductive Logic Programming is the following *model inference problem* [18]. Suppose we have a first-order language and an interpretation T which is a subset of the set of all ground atoms $\mathcal{B}$ (*Herbrand base*). Our aim is to find a definite program H, (*target program*), whose least Herbrand model M_H equals T (*target interpretation*). If $M_H = T$, we will call H *correct w.r.t.* T. For example, given $T = \{P(a), P(f^2(a)), \ldots, P(f^{2k}(a)), \ldots\}$ (where $f^2(a)$ abbreviates $f(f(a))$, etc.), a correct program is

$P(a)$
$P(f^2(x)) \leftarrow P(x)$

However, a correct program may not always be found, and often we need to approximate it by other programs, for the following reasons:

1. A correct program w.r.t. a taregt T may not exist. As noted in [13, 14], the set of Herbrand interpretations is uncountable, while the set of programs is only countable. Hence for many interpretations T, there is no program H with $M_H = T$, and the best we can do is to find some kind of approximately correct program.
2. In general we are not given a complete interpretation $T \subseteq \mathcal{B}$ but some $E^+ \subset T$ as a set of positive examples and some $E^- \subset (\mathcal{B} \setminus T)$ as a set of negative examples. We want to find a program which implies all atoms in E^+ and none in E^-. Such a correct program H w.r.t. $E = E^+ \cup E^-$ may not be correct w.r.t. T. The idea of machine learning is to use H to predict the truth values of new examples in $\mathcal{B} \setminus E$. Since the information about T is incomplete, there is no way to know the correctness of H w.r.t. T and hence no way to guarantee correct predictions. However, there may be some way to estimate the closeness of M_H to T.
   Even if we are only interested in finding a correct program H w.r.t. the given examples E, we may still for efficiency reasons have to settle for an approximation H' which is not totally correct w.r.t. E. Thus we also want to know how close H' is to a correct program w.r.t. E.
3. The examples will sometimes not give us correct information about T because of *noise*. From noisy data it is difficult to find a correct program. Moreover, a program that is correct w.r.t. noisy examples may be inadequate, while a program that is incorrect w.r.t. noisy examples may be adequate.

For these reasons, a way to measure the 'quality' of an approximation to a correct program is needed. How can we make this notion of 'approximation' precise, so we can use it in theoretical and practical analysis? Given some examples (ground atoms) from T, we may be satisfied with a program that covers only 90% of the examples correctly. This, however, is rather crude; for instance, all different examples are then considered equally important. A more sophisticated way is used in the framework of Probably Approximately Correct learning [10]. Here the examples are drawn according to an unknown probability distribution $\mathcal{P}$, and the induced M_H would be considered approximately correct if the probability (according to $\mathcal{P}$) of getting examples from the symmetric difference $M_H \triangle T = (M_H \setminus T) \cup (T \setminus M_H)$ is less than a user-specified ϵ.

In ILP, we usually do not consider the probability distributions on the examples. Thus we will take a different viewpoint towards approximation than the one adopted in PAC-learning. We compare two ground expressions by their tree structures. For differences between trees, the nodes close to the root count the most. This measure of difference is given by a metric d defined on the set $\mathcal{E}$ of ground atoms and ground terms. For example, $P(a)$ and $Q(a)$ are considered quite different by our definition of d, and the difference between $P(a)$ and $P(f(a))$ is considered larger than the difference between $P(f(a))$ and $P(f^2(a))$. We can then use the Hausdorff metric h (see [4]) induced by d to define a distance between Herbrand interpretations. This distance can be used for measuring the difference between two programs (more precisely: between their least Herbrand models). We will see for example that if $S = \{P(a), P(f^2(a)), P(f^4(a)), \ldots\}$,

$T_1 = S \cup \{P(f^9(a))\}$ and $T_2 = S \cup \{P(f(a))\}$, then $h(S, T_1) < h(S, T_2)$.

This article is organized as follows. We begin with some preliminaries concerning metric spaces and logic. Then we define our distance d on the set $\mathcal{E}$ of expressions. We prove that d is indeed a metric on $\mathcal{E}$, and every subset of $\mathcal{E}$ is closed. Although this metric generates the *discrete topology*, it is still interesting for our purposes. This metric will induce a Hausdorff metric h on the set of all subsets of $\mathcal{E}$, which includes the set of all Herbrand interpretations. This can be used to measure the closeness of a program (more precisely, of its least Herbrand model) to a target interpretation. Although the Hausdorff metric may look unfamilar to some readers, its application in our situation of Herbrand interpretations is actually fairly simple and concrete. By considering elements in the symmetric difference between two Herbrand interpretations S and T, we can find some necessary and some sufficient conditions for upper bounds on the distance $h(S, T)$. In real application, we usually want to find a program which is correct w.r.t. some true examples E^+ and false examples E^-. In this situation we use $h(M_H \cap (E^+ \cup E^-), E^+)$. Notice that if $E^+ \cup E^- = \mathcal{B}$, then E^+ is the related Herbrand interpretation. Here we have the original model inference problem again.

1.1 Discussion and related work

In machine learning, using distances to compare two concepts is not new. For example, distance plays an important role in instance-based learning, case-based reasoning (see [1, 5, 6]) and neural networks (see [2]), which use concepts like similarity, nearest neighbors, etc. In related fields such as data mining, metrics is also used to group data [9]. However, defining distances based on tree structures in two levels (d and h) as given here is a novel approach. (In the field of semantics, the difference between two streams of values (input or output streams) is sometimes defined by considering only the first place where the streams differ. The way we compare trees here, may be considered as a generalization of this idea. Moreover, the Hausdorff metric has been used in [3]. It measures the difference between processes in concurrent programming. Its definition depends on metrics on some functional spaces.)

Some people may think that the tree structures of atoms are not important for application, because the applications of ILP often restrict attention to function-free languages. There are two reasons for this argument. Usually a database has the attribute-value format, and if we want to do data mining in this situation, a direct way to express a rule is as a function-free clause. Even if we choose to express some concept by means of function symbols, we will not get a deep tree structure because attributes are typed. For example, the salary of some person cannot be applied twice, so a structure such as $salary(salary(x))$ does not make sense. However, the theoretical developement of ILP should not be stopped by the fact that more complex applications are not readily apparent. In fact, finding mathematical concepts as target concepts is also a kind of application of ILP. Shapiro [18] has shown how to find a program which characterizes multiplication and addition by using examples and the successor function in the natural numbers. His paper is one of the most important papers in ILP. Moreover, the modeling of complicated scientific experiments in physics and chemistry, involving mathematical concepts like groups or matrix operations, will probably require use of function symbols. In fact, chemical reactions can be considered as functions over the set of chemicals. If we want to use ILP for data mining in these fields rather than in simpler commercial databases, we may need more than just function-free languages.

One might also think that flattening [17] will solve the problem of additional complexity brought about by functions, by transforming clauses with functions to function-free form. However, a side effect of flattening is that the number of literals in a clause will increase in proportion to the number of eliminated function symbols. Thus the complexity is not removed, but appears in another form. If we choose to keep the function symbols instead of applying flattening, then a direct measure of the difference between tree structures is necessary in order to be able to measure the degree of approximation.

One might also argue that the difference between atoms is not important; that it is the difference between clauses that we should measure. However, our aim in ILP is usually to find a program, a set of clauses rather than a single clause. The various clauses in a program usually "interact" and depend on each other to produce the right results. Since our aim is to find sets of clauses rather than individual clauses, we should look for a way to measure the distance between sets rather than individual clauses. One way to compare sets of clauses is to measure the distance between their least Herbrand models; the difference between individual clauses will not be very helpful here.

In the literature there are also distances defined for objects with complex structures. For example, in [6] a similarity measure is given for structured representation that is based on graph edit operations (the number and the cost of operations needed to transform one graph to another are used to define the distance). This is quite different from the distances given here. In the first place we have here distances defined on two levels: one for ground expressions, the other for sets of ground expressions. Furthermore, we do not use graph operations to compare two graphs. In a graph, an edge can go from any node to any other one. We, on the contrary, compare two static trees in the direction from top to bottom.

2 Preliminaries

2.1 Metric space and Hausdorff metric

Definition 1. Let $\mathcal{E}$ be a set and $\Re$ be the set of real numbers. A mapping $d : \mathcal{E} \times \mathcal{E} \rightarrow \Re$ is called a *metric* or *distance* on $\mathcal{E}$ if it has the following properties:

1. $d(x, y) \geq 0 \ \forall x, y \in E$ and $d(x, y) = 0$ iff $x = y$.
2. $d(x, y) = d(y, x) \ \forall x, y \in \mathcal{E}$.
3. $d(x, y) + d(y, z) \geq d(x, z) \ \forall x, y, z \in \mathcal{E}$ (*triangle inequality*).

The pair $(\mathcal{E}, d)$ is called a *metric space*. Given $\epsilon > 0$ and $x \in \mathcal{E}$, the *open ϵ-ball* $B(x, \epsilon)$ *centered at* x is the set $\{y | \, d(x, y) < \epsilon\}$. A set $T \subseteq \mathcal{E}$ is called *open* if for every $x \in T$, there is an open ball centered at x in T. A set $T \subseteq \mathcal{E}$ is *closed* if $\mathcal{E} \backslash T$ is open. The sets $\mathcal{E}$ and $\emptyset$ are both open and closed. A metric space is *bounded* if the metric is bounded, i.e. there is an m such that $d(x, y) \leq m, \forall x, y \in \mathcal{E}$.

It is well known that the union of an arbitrary collection of open sets and the intersection of a finite collection of open sets are open themselves. Given a real function f defined on a set U and $\emptyset \neq S \subseteq U$, we let $inf_{x \in S} f(x)$ denote the greatest lower bound (infimum) of $\{f(x) \,|\, x \in S\}$ if it exists. We use *sup* (supremum) for the dual concept. For example, let $d(x, y) = |x - y|$ be the usual metric defined on $\Re$. Consider the closed intervals $S = [0, 1]$ and $T = [2, 5/2]$. Then $inf_{x \in S} d(x, 5/2) = 3/2$ and $inf_{y \in T} d(0, y) = 2$.

Definition 2. Let $(\mathcal{E}, d)$ be a bounded metric space and $\mathcal{C}(\mathcal{E})$ be the set of all closed subsets of $\mathcal{E}$. For non-empty $T \in \mathcal{C}(\mathcal{E})$ and $x \in \mathcal{E}$, let $d(x, T) = inf_{y \in T} d(x, y)$. For non-empty $S, T \in \mathcal{C}(\mathcal{E})$, let $\rho(S, T) = sup_{x \in S} d(x, T)$, and let $h(S, T) = sup(\rho(S, T), \rho(T, S))$. Furthermore, define $h(\emptyset, \emptyset) = 0$ and $h(\emptyset, T) = 1$ if $T \neq \emptyset$. It can be shown that h is a metric on $\mathcal{C}(\mathcal{E})$ (see exercise 3 of 3.16 in [4]). This is called the *Hausdorff metric induced by d.*

We can also artificially define: $\rho(\emptyset, \emptyset) = 0$; if $T \neq \emptyset$, then $\rho(\emptyset, T) = 0$ and $\rho(T, \emptyset) = 1$. With this definition we also have $h(\emptyset, T) = sup(\rho(\emptyset, T), \rho(T, \emptyset)) = 1$, as above.

Example 1. Let us consider $\Re$ with $d(x, y) = |x - y|$ defined as above. Let $S = [0, 1]$ and $T = [2, 5/2]$. Then $\rho(S, T) = 2$ and $\rho(T, S) = 3/2$. Thus $h(S, T) = 2$.

Example 2. Let $\emptyset \neq S \subseteq T$ and $S \neq T$. Then $\rho(S, T) = sup_{x \in S} d(x, T) = 0$ and $\rho(T, S) = sup_{x \in T} d(x, S) = sup_{x \in T \setminus S} d(x, S)$. Now let $S = [0, 1]$, $T = [0, 2]$ in $\Re$. Then $h(S, T) = 1$.

Remarks

1. If $S \subseteq T$, then $\rho(S, T) = 0$. According to Definition 2, $\rho(S \backslash T, T) = \rho(\emptyset, T) = 0$ also. If $S \not\subseteq T$, then $S \setminus T \neq \emptyset$ and $\rho(S, T) = \rho(S \setminus T, T)$. Thus in general, $\rho(S, T) = \rho(S \setminus T, T)$.
2. Since $\emptyset$ does not contain any elements, we cannot use the definition of *inf* and *sup*. All the results in this article that also apply to $\emptyset$ are true, but we will often omit the trivial cases of proofs w.r.t. $\emptyset$, focusing only on the nonempty sets.
3. If $h(S, T) < \epsilon$, then each point in S is within an ϵ distance of a point in T and vice versa. If $h(S, T) > \epsilon$, then there is an $x \in S$ such that $d(x, y) > \epsilon \; \forall y \in T$ or there is an $y \in T$ such that $d(x, y) > \epsilon \; \forall x \in S$.

2.2 Definite programs and least Herbrand models

In the sequel we will consider some fixed first-order language with a finite number of function symbols (including constants) and a finite number of predicate symbols. For most definitions from logic, we refer to [12, 14]. Let us just recall the following:

- *Terms* are defined inductively as follows: constants and variables are terms, and if $t_1, \ldots, t_n$ are terms and f is a function symbol of *arity* n, then $f(t_1, \ldots, t_n)$ is a term.
- An *atom* has the form $P(t_1, \ldots, t_n)$ where P is an n-ary predicate symbol and each t_i is a term.
- A *definite program clause* (often abbreviated to *clause* in this paper) is an implication of the following form: $A \leftarrow B_1, \ldots, B_n$, where A and B_i are atoms. It should be read as "B_1 and ... and B_n together imply A." A is called the *head* of the clause, $B_1, \ldots, B_n$ is the *body*. Each variable in a clause is universally quantified, but we will not write out the quantifiers explicitly. A term or clause is *ground* if it does not contain any variables.
- A *definite program* (or just *program*) is a finite set of clauses.
- An *Herbrand interpretation* T assigns to each ground atom in the language a truth-value (*true* or *false*). T may be identified with the set of ground atoms to which it assigns *true*. A ground clause is true under T if its head is true and/or at least one of the atoms in its body is false under T. A non-ground clause is true under T if all its ground instances are true under T. An Herbrand interpretation T is called an *Herbrand model* of a definite program H, if all clauses in H are true under T.
- It can be shown that the intersection of all Herbrand models of a definite program H is itself an Herbrand model of H. This is called the *least Herbrand model* of H, denoted by M_H. It equals the set of all ground atoms that are logically implied by H.

2.3 Positions of symbols in an expression

Definition 3. An *expression* is a ground term or a ground atom. (In this article, we use p, q, etc. in general to denote arbitrary function or predicate symbols occuring in an expression.) An expression e has a tree structure. We can express this structure by coding the *positions* of symbols in e in the following way.

1. The leftmost symbol in e has $\langle 1 \rangle$ as its position.
2. If $\langle a_1, a_2, \ldots, a_n \rangle$ is the position of a predicate or function symbol p occurring in e, and p has $k > 0$ arguments, then the leftmost symbol of the i-th argument has $\langle a_1, \ldots, a_n, i \rangle$ as its position in e.

For an expression e and position *pos*, we use $e(pos)$ to denote the symbol at position *pos* in e.

The *depth* of a symbol occurrence is the length of its position. The *depth* of an expression e, denoted by $depth(e)$, is the depth of the symbol in e with the greatest depth.

Given expressions e and e', we say they are *the same to (at least) depth* n if for every $k \leq n$, $\langle a_1, a_2, \ldots, a_k \rangle$ is a position of a symbol in e iff it is also a position of the same symbol in e'.

Example 3. Consider $e = P(g(b), f(a, g(b)))$ and $e' = P(g(b), f(a, b))$. Then $e(\langle 1 \rangle) = P$, $e(\langle 1,1,1 \rangle) = b$ and $e(\langle 1,2,2,1 \rangle) = b$. The depth of e is 4 and the depth of e' is 3. Expressions e and e' are the same to depth 2. They are not the same to depth 3, because e has g at position $\langle 1,2,2 \rangle$, while e' has b at that position. Consider also $e = P(a, g^2(a))$ and $e' = P(a, g^3(a))$. Then e and e' are the same to depth 3.

Notice that if $e = e'$, then e and e' are the same to any depth. If $depth(e) = n$ and $e \neq e'$, then e and e' may be the same to depth $n - 1$, but not more.

3 A metric on expressions

From now on we use $\mathcal{E}$ to denote the set of all expressions in a first-order language. We would like to define a metric d, bounded by 1, on $\mathcal{E}$ such that $d(s,t)$ can be used to measure the difference between s and t.

Definition 4. We define $d : \mathcal{E} \times \mathcal{E} \to \Re$ as follows.

1. $d(e,e) = 0, \forall e \in \mathcal{E}$
2. If $p \neq q$, then $d(p(s_1, \ldots, s_n), q(t_1, \ldots, t_m)) = 1$
3. $d(p(s_1, \ldots, s_n), p(t_1, \ldots, t_n)) = \frac{1}{2n} \sum_{i=1}^{n} d(s_i, t_i)$

Example 4. The following examples illustrate how d works.
$d(f(a), P(a,b)) = 1$,
$d(f(a), f(b)) = \frac{1}{2} d(a,b) = \frac{1}{2}$,
$d(g(f(a), g(a,b)), g(f(b), b)) = \frac{1}{2 \cdot 2}(d(f(a), f(b)) + d(g(a,b), b))$
$= \frac{1}{4}(\frac{1}{2} + 1) = \frac{3}{8}$.
Let $T = \{P(f(a)), P(f^3(a)), P(f^5(a)), \ldots\}$. Suppose d is a metric (as we will prove below). Since $d(P(f^2(a), P(f^3(a)) = \frac{1}{8}$, we have $d(P(f^2(a)), T) = \frac{1}{8}$.

Notice that differences between terms that occur at low depth are given more "weight" than differences at higher depth.

Theorem 5. $(\mathcal{E}, d)$ is a metric space with d bounded by 1.

Proof:
d bounded by 1: It is clear that $d(e, e') \leq 1$ if it is defined by item 1 or 2 above. Now suppose item 3 of the definition is used: $d(e,e') = d(p(s_1, \ldots, s_n), p(t_1, \ldots, t_n)) = \frac{1}{2n} \sum_{i=1}^{n} d(s_i, t_i)$ and suppose $d(s_i, t_i) \leq 1, \forall i$ (induction hypothesis). Then $d(e, e') \leq \frac{1}{2n}(1 + \ldots + 1) = \frac{n}{2n} = \frac{1}{2}$.

d is a metric: We prove only $d(r,s) + d(s,t) \geq d(r,t)$, by induction on the depth of these expressions. Suppose $depth(r) = depth(s) = depth(t) = 1$, so r, s, and t are function or predicate symbols of arity 0. If $r = s = t$, then $d(r,s) = d(s,t) = d(r,t) = 0$. Otherwise, $r \neq s$ and/or $s \neq t$, then $d(r,s) + d(s,t) \geq 1 \geq d(r,t)$.

Suppose this inequality is true if the depth of r, s, t is at most k, and assume the expression among r, s, t that has greatest depth, has depth $k + 1$.

- $r(\langle 1\rangle) \neq t(\langle 1\rangle)$: In this situation $d(r,t)=1$. However, if $r(\langle 1\rangle)=s(\langle 1\rangle)$, then $d(s,t)=1$. Thus the inequality holds.
- $r(\langle 1\rangle) = t(\langle 1\rangle) \neq s(\langle 1\rangle)$: In this situation $d(r,s)+d(s,t)=2 \geq d(r,t)$.
- $r(\langle 1\rangle) = t(\langle 1\rangle) = s(\langle 1\rangle)$:
  $d(p(r_1,\ldots,r_n),p(s_1,\ldots,s_n))+$
  $d(p(s_1,\ldots,s_n),p(t_1,\ldots,t_n))$
  $=\frac{1}{2n}\sum_{i=1}^{n} d(r_i,s_i)+\frac{1}{2n}\sum_{i=1}^{n} d(s_i,t_i)$
  $=\frac{1}{2n}\sum_{i=1}^{n}(d(r_i,s_i)+d(s_i,t_i))$
  $\geq \frac{1}{2n}\sum_{i=1}^{n} d(r_i,t_i)=d(r,t)$.

Example 5. Let $e=P(a,b), e'=P(a,c), e''=P(c,d)$. Then $d(e,e')=\frac{1}{4}$, and $d(e,e'')=\frac{1}{2}$.

In general, if $e \neq e'$, then there is $n \geq 0$ such that they are the same to depth n but not the same to depth $n+1$. Given e and e', the upper and lower bounds of $d(e,e')$ are given by the following two lemmas.

Lemma 6. If e and e' are the same to at least depth n, then $d(e,e') \leq \frac{1}{2^n}$.

Proof: $n=0$: This means $d(e,e') \leq 1$.
$n=k+1$: Let $e=p(s_1,\ldots,s_m)$ and $e'=p(t_1,\ldots t_m)$. Then $d(e,e')=\frac{1}{2m}\sum_{i=1}^{m} d(s_i,t_i) \leq \frac{1}{2m}\cdot\frac{m}{2^k}=\frac{1}{2^{k+1}}$.

Lemma 7. Let M be an integer which is greater than the maximal arity of all function and predicate symbols in the logic language. Then if e and e' are not the same to depth n, we have $d(e,e')>\frac{1}{(2M)^n}$.

Proof: $n=1$: Then $d(e,e')=1>\frac{1}{2M}$.
$n=k+1$: Let $e=p(s_1,\ldots,s_m)$ and $e'=p(t_1,\ldots t_m)$. There is an i such that s_i and t_i are not the same to depth k. That means $d(s_i,t_i)>\frac{1}{(2M)^k}$. Now $d(e,e')=\frac{1}{2m}\sum_{j=1}^{m} d(s_j,t_j) \geq \frac{1}{2m}d(s_i,t_i)>\frac{1}{2M}d(s_i,t_i)>\frac{1}{(2M)^{k+1}}$.

Lemma 8. Every element $e \in \mathcal{E}$ is an open set.

Proof: Let M be defined as in Lemma 7. Given e with $depth(e)=n$, we will prove that the open ball $B=\{t \in E \mid d(t,e)<\frac{1}{(2M)^n}\}$ contains only e. Consider an arbitrary $e' \neq e$ with $depth(e')=m$.

- If $n \geq m$, then e and e' can be the same only to some $k<m$ (see remark after Example 3). Thus $d(e,e')>\frac{1}{(2M)^n}$ according to lemma 7.
- If $n<m$, then e and e' can only be the same to depth $k<n$. Then $d(e,e')>\frac{1}{(2M)^n}$ according to lemma 7.

Thus $d(e,e')>\frac{1}{(2M)^n}$ for all $e' \neq e$. Since every open ball is open and the open ball B contains only one point e, we have that $\{e\}$ is open.

Since the union of open sets is open, every subset of $\mathcal{E}$ is open and hence every subset of $\mathcal{E}$ is closed as well. Thus we have the following theorem:

Theorem 9. Every subset of $\mathcal{E}$ is both open and closed.

Remark A metric space is a special kind of topological space. A topological space in which every single point constitutes an open set is called a *discrete topological space*.

4 Hausdorff metric on $\mathcal{C}(\mathcal{E})$

Since every subset of $\mathcal{E}$ is closed, we have that $\mathcal{C}(\mathcal{E})$ equals the set of all subsets of $\mathcal{E}$. We can now define the distance between two subsets of $\mathcal{E}$ by means of the Hausdorff metric h on $\mathcal{C}(\mathcal{E})$.

Example 6. Let $S = \{P(a), P(f^2(a)), P(f^4(a)), \ldots\}$ and $T = \{P(f(a)), P(f^3(a)), \ldots\}$. Then $\rho(S,T) = sup_{x \in S} d(x,T) = sup\{1/2, 1/8, 1/32, \ldots\} = 1/2$ and $\rho(T,S) = sup_{x \in T} d(x,S) = sup\{1/4, 1/16, 1/64, \ldots\} = 1/4$. Thus $h(S,T) = 1/2$.

Example 7. Let $S = \{P(a), P(f^2(a)), P(f^4(a)), \ldots\}$ and $T = S \cup \{P(f^3(a))\}$. Then $\rho(S,T) = 0$ and $\rho(T,S) = sup_{x \in T \setminus S} d(x,S) = d(P(f^3(a)), S) = 1/16$. Thus $h(S,T) = 1/16$.

Theorem 10. Consider $S, T \subseteq E$. Suppose their symmetric difference $S \triangle T = (S \setminus T) \cup (T \setminus S)$ has the following property: for every $e \in S \setminus T$, there is an e' in T such that e and e' are the same to depth n. For every $e \in T \setminus S$, there is $e' \in S$ with the same property. Then $h(S,T) \leq \frac{1}{2^n}$.

Proof: We know $S = (S \cap T) \cup (S \setminus T)$. If $e \in S \cap T$, then $d(e,T) = 0$. If $e \in S \setminus T$, there is an $e' \in T$ such that e and e' are the same to depth n. By lemma 6 we have $d(e,e') \leq \frac{1}{2^n}$. Thus $d(e,T) \leq \frac{1}{2^n}$. Hence we have $\rho(S,T) \leq \frac{1}{2^n}$. Similarly, we have $\rho(T,S) \leq \frac{1}{2^n}$.

Theorem 11. Let M be defined as in Lemma 7. If $h(S,T) \leq \frac{1}{(2M)^n}$, then

- For every $e \in S \setminus T$, there is an $e' \in T$ such that e and e' are the same to at least depth $n-1$.
- For every $e \in T \setminus S$, there is an $e' \in S$ such that e and e' are the same to at least depth $n-1$.

Proof: We prove this by contradiction. Suppose for some $e \in S \setminus T$ we have for every $e' \in T$ that e and e' are not the same to depth $n-1$. By lemma 7, we have $d(e,e') > \frac{1}{(2M)^{n-1}}$. That means $d(e,T) \geq \frac{1}{(2M)^{n-1}}$, so $\rho(S,T) \geq \frac{1}{(2M)^{n-1}}$. Thus $h(S,T) = sup(\rho(S,T), \rho(T,S)) \geq \rho(S,T) \geq \frac{1}{(2M)^{n-1}} > \frac{1}{(2M)^n}$. This is a contradiction. Similarly for the analogous situation of $\rho(T,S)$.

In the following examples, $\mathcal{B}(\subseteq \mathcal{E})$ is the set of all ground atoms (Herbrand base).

Example 8. Let $\mathcal{B} = \{P(a), P(f(a)), P(f^2(a)), P(f^3(a)), \ldots\}$. Let $S = \{P(a), P(f^2(a)), P(f^4(a)), \ldots\}$ and $T = (S \cup \{P(f^{99}(a))\}) \setminus \{P(f^{100}(a))\}$. Consider $e = P(f^{99}(a)) \in T \setminus S$. Then $d(e, S) = \frac{1}{2^{100}}$. Now let $e = P(f^{100}(a)) \in S \setminus T$. We have $d(P(f^{100}(a)), T) = \frac{1}{2^{101}}$. Thus $h(S, T) = \frac{1}{2^{100}}$. Now consider a program H, with the following clauses:

$P(a)$
$P(f^2(x)) \leftarrow P(x)$

Then $M_H = S$. Suppose the target interpretation is T. Then the model inference problem is approximately solved using H, where the "quality" of the approximation can be measured by $h(M_H, T) \leq \frac{1}{2^{100}}$.

Example 9. Let $\mathcal{B} = \{Q(a), P(a), P(f(a)), P(f^2(a)), \ldots\}$, $S = \{P(a), P(f^2(a)), P(f^4(a)), \ldots\}$ and $T = S \cup \{Q(a), P(f^{99}(a))\}$. Then $h(S, T) = 1$. This means that S and T are far from each other, according to our metric.

Now suppose that the target is T. Consider the same program H as in the above example. Then H does not give a good approximation to the target concept. However, suppose we have another program H'

$P(a)$
$Q(a)$
$P(f^2(x)) \leftarrow P(x)$

Then $M_{H'} = S \cup \{Q(a)\}$ and $h(M_{H'}, T) \leq \frac{1}{2^{100}}$.

Remark From the examples above we may get the impression that if two interpretations S and T differ only in ground atoms with large depth, then they are close to each other in h. This is not completely correct, as we can see from the following example. Let $S = \{P(a)\}$ and $T = \{P(a), P(f^{100}(a))\}$. Then $\{e \in S \mid depth(e) \leq 100\} = \{e \in T \mid depth(e) \leq 100\}$, but $h(S, T) = \frac{1}{2}$. In fact, $h(S, T)$ is small if for every element e in S we can find an e' in T which has the same structure as e to a large depth and vice versa. Now $P(f^{100}(a))$ is already different from any element in S from depth 2. If we consider $S' = S \cup \{P(f^{99}(a))\}$, then $h(S', T) = \frac{1}{2^{100}}$, which is rather small.

5 Metric on examples

The model inference problem concerns an interpretation T, which can be expressed as a subset of the set of all ground atoms $\mathcal{B}$ (namely the subset of *true* ground atoms). In this case, the distance between a target and the found program H is given by $h(M_H, T)$, which will be 0 iff M_H is correct w.r.t. T.

In applications, however, we are not given the complete T, but only a sample from T: a set of ground atoms $E = E^+ \cup E^-$, where E^+ is called the set of *positive examples* and E^- the set of *negative examples*. We would like to find a program H which implies all the positive examples and none of the negative ones. In this situation we may also only find a program which is not completely correct w.r.t. E, and we would like to use our distance to measure the degree of approximation. Unfortunately, we cannot straightforwardly use $h(M_H, E^+)$ instead of $h(M_H, T)$. For example, let $E^+ = \{P(0), P(f^2(0))\}$ and $E^- = \{P(f(0)), P(f^3(0))\}$. Then

$H = \{P(0), (P(f^2(x)) \leftarrow P(x))\}$ is a correct program w.r.t. to these examples. Since H implies much more than just the atoms in E^+, we have for instance $d(P(f^4(0)), P(f^2(0))) = \frac{1}{2^4}$. Thus we have $h(M_H, E^+) \geq \frac{1}{2^4}$, which gives the impression that a correct program may not even be a good approximation. To solve this problem, we restrict attention from the set of all ground atoms $\mathcal{B}$ to the set of given example E, because this is all we know about T. (If $E = \mathcal{B}$, we have the original situation given in the last sections.) Given a program H, we consider $S = M_H \cap E$ and $h(S, E^+)$. Notice that $S \setminus E^+$ contains negative examples which are implied by H and $E^+ \setminus S$ contains positive examples which are not impled by H. Thus both negative and positive examples play a role for the distance $h(S, E^+)$. If H is correct w.r.t. E, then $h(S, E^+) = 0$.

6 Summary and future work

This article has been motivated by the problem in ILP of measuring differences between the least Herbrand models of a definite program and a target interpretation. Our aim was to define a distance between Herbrand interpretations in a natural way. We started with a definition of a metric d on the set of expressions $\mathcal{E}$, which allowed us to define the Hausdorff metric h on $\mathcal{C}(\mathcal{E})$, the set of all closed subsets of $\mathcal{E}$. Since every subset of $\mathcal{E}$ is closed, every Herbrand interpretation is such a closed subset. Thus h can be used to measure the distance between two Herbrand interpretations S and T. Moreover, $h(S,T)$ can be characterized by elements in the symmetric difference $S \bigtriangleup T$. This implies that we have a concrete idea of an 'approximation' to a target interpretation (or to E^+ in practical applications where T is unknown), which may turn out to be very useful in ILP.

It should be noted that it is in general undecidable whether some ground atom is a member of M_H. In order to deal with this, we may use the idea of *h-easyness*, which is also used in [18]. This amounts to bounding the maximal number of resolution steps that can be taken when trying to deduce an atom from H.

There are the following possible lines for future work.

First of all, there should be more applications of the Hausdorff metric in Inductive Logic Programming, especially on topics related to mathematics, chemistry and physics. I think one can show by applications that function-free languages are not satisfactory for these fields.

Secondly, given examples $E = E^+ \cup E^-$ and abbreviating $S = M_H \cap E$, we can use $h(S, E^+)$ as a heuristic in the search for an H that is correct w.r.t. given E^+ and E^-. Initially we start with an empty (or otherwise inadequate) H, so $h(S, E^+)$ will be high. Then we stepwisely adjust H in such a way that $h(S, E^+)$ decreases. Eventually we hope to end up with an H for which $h(S, E^+)$ is small (even 0, in case the final H is fully correct w.r.t. E^+ and E^-).

Thirdly, one motivation of this article is to find a good hypothesis even when there is noise in data. Suppose E is a set of examples which contains noise. Suppose H is a program which is correct with respect to examples if the examples are correctly given. Let $S = M_H \cap E$. If a given wrong positive example

e is close to some $e' \in S$, i.e. $d(e, e')$ is small, then $inf_{x \in S} d(e, x)$ is also small and $h(S, T)$ will not become large because of e, and we may well be able to find a good approximation in this case. If the wrong e is distant from all elements in S, then S will not be a good approximation of E^+. More research about different types of noise and their influences on h should be done in the future.

Four, we may systematically analyze distance measures between clauses, even though this seems not very useful at first sight (see subsection 1.1). That means we first have to generalize d to general clauses (the distances in the literature usually only apply to function-free clauses). Suppose we have two programs with clauses containing functions, then we may compare three kinds of distances between the programs. One is the h of this article. One is a generalization of d to clauses (if we can find a satisfactory one). The third approach is to flatten clauses and apply some other distance defined for function-free clauses.

Fifth, we can compare h with the difference measure of PAC-learning, that uses a probability distribution on the set of possible examples. For instance, the more complex examples may be assigned a lower probability, in analogy with the *universal* probability distribution [11].

Finally, since all results in this paper could be proved by considering only the tree structure of expressions, and do not depend on the details of first-order logic, metrics like the ones defined here may be applicable in fields of computer science other than ILP.

Acknowledgements My thanks go to Ronald de Wolf and Jan Willem Nienhuys for their help.

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. R. Beale and T. Jackson *Neural Computing, an Introduction* Adam Hilger.
3. J. W. de Bakker and J. I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 1/2, 1984.
4. J. Dieudonné. *Foundations of Modern Analysis*. Academic Press, 1969.
5. W. Emde and D. Wettschereck. Relational instance-based learning. In: L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning (ICML-96)*, pages 122–130. Morgan Kaufmann, 1996.
6. H. Bunke and B.T. Messmer. Similarity measure for strutured representations In: S. Wess, K.D. Althoff and M. Richter, editors, *Topics in Case-Based Reasoning, First European workshop, EWCBR-93*, pages 106-118, 1993. Springer-Verlag.
7. E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
8. A. Hutchinson. Metrics on Terms and Clauses. In: M. Someren, G. Widmer, editors. Proceedings of the 9th European Conference on Machine Learning (ECML-97), pages 138-145, 1997. Springer-Verlag.
9. D. T. Kao, R. D. Bergeron, M. J. Cullinane, and T. M. Sparr. Semantics and mathematics of science datasampling. Technical Report 95–14, Department of Computer Science, University of New Hampshire, 1995.

10. M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge (MA), 1994.
11. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, Berlin, second edition, 1997.
12. J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, second edition, 1987.
13. S. H. Nienhuys-Cheng and R. de Wolf. A complete method for program specialization based on unfolding. In: W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, pages 438–442. Wiley, 1996.
14. S. H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, LNAI Tutorial 1228, Springer-Verlag, May 1997.
15. G.D. Plotkin. A Note on Inductive Generalization. *Machine Intelligence*, 5:153–163, 1970.
16. J.C. Reynolds. Transformational Systems and the Algebraic Structure of Atomic Formulas. *Machine Intelligence*, 5:135–153, 1970.
17. C. Rouveirol. Flattening and saturation: Two representation changes for generalization. *Machine Learning*, 14:219–232, 1994.
18. E. Y. Shapiro. Inductive inference of theories from facts. Research Report 192, Yale University, 1981.
19. D. Wettschereck. *A Study of Distance-Based Machine Learning Algorithms.* PhD thesis, Oregon State University, 1994.

Appendix

A recently published article [8] defines a size function as a certain type of real valued function on the set of substitutions. A size function S can induce a pseudo-metric d_S on the set of terms and literals. This pseudo-metric induces a Hausdorff pseudo-metric on clauses. For this a clause should be interpreted as a set of literals. Some differences between the approach of [8] and the method outlined above are:

1. [8] uses pesudo-metrics rather than metrics, i.e. different elements may have 0 distance. The Hausdrff metric induced by a metric is defined on closed sets otherwise it would be a pseudo-metric. In [8] not even all finite subsets of terms are closed whereas in our metric space of expressions every subset is closed.
2. In [8] a concrete size function S is given. With this S and the weights W assigned to function symbols in the manner of [8] one obtains for any n, if $k > 0$
   $d_S(P(f^n(a)), P(f^{n+k}(a))) = W(a) + W(f)$.
   Using our definition of d, we have
   $d(P(f^n(a)), P(f^{n+k}(a))) = \frac{1}{2^{n+1}}$.
3. Since on the one hand $d_S(P(x,x), P(x,y)) = 0$ and on the other hand two alphabetical variants can have a positive distance, [8] tries to adapt the distance definition. In the new definition variants would have distance 0. However, two clauses may be equivalent even they are not variants. For example, let $C = P(a) \leftarrow Q(f(a))$ and $D = P(a) \leftarrow Q(f(a)), Q(x)$. Then C and D are subsume equivalent and hence logically equivalent. Since $d_S(Q(x), Q(f(a))) =$

$W(f) + W(a) > 0$, the proposed new definition will induce a positive distance between C and D. In our method the difference between programs is indicated by the distance h between their least Herbrand models, and two logically equivalent programs have zero distance.