

**A framework for defining  
distances between first-order  
logic objects**

*Ramon J.  
Bruynooghe M.  
Report CW 263, May 1998*



**Katholieke Universiteit Leuven  
Department of Computer Science**

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# A framework for defining distances between first-order logic objects

*Ramon J.*

*Bruynooghe M.*

*Report CW 263, May 1998*

Department of Computer Science, K.U.Leuven

## **Abstract**

Several learning systems, such as systems based on clustering and instance based learning, use a measure of distance between objects. Good measures of distance exist when objects are described by a fixed set of attributes as in attribute value learners. More recent learning systems however, use a first order logic representation. These systems represent objects as models or clauses. This paper develops a general framework for distances between such objects and reports a preliminary evaluation.

**Keywords :** Machine learning, distances, first order logic.

# 1 Introduction

In learning systems based on clustering (e.g. TIC [3], KBG [1]) and in instance based learning (e.g. [10, ch.4], RIBL [8]), a measure of the distance between objects is an essential component. Good measures exist for distances between objects in an attribute value representation (see e.g. [10, ch. 4]). Recently there is a growing interest in using more expressive first order representations of objects and in upgrading propositional learning systems into first order learning systems (e.g. TILDE [2], ICL [5] and CLAUDIEN [4]). The upgrading of clustering and instance based learning systems requires to develop a measure for the distance between first order objects, either described as clauses or as models of first order theories.

Some proposals for distance measures between atoms and clauses exists (e.g. [12] and [9]). They use Hausdorff metrics to extend distances between atoms into distances between sets of atoms (clauses or models). This has two drawbacks. Firstly, the value of the Hausdorff metric depends very much on the most extreme value in both sets. Secondly, the similarity due to occurrences of the same subterm (constant, variables, ...) in different atoms of the same clause has no influence on the value. Other authors (e.g. [8], [1]) use rather ad-hoc measures of similarity which do not comply with all axioms of a distance, in particular with the triangle axiom ( $d(x, z) \leq d(x, y) + d(y, z)$ ).

Attribute value systems also allow to compute a prototype [10, ch. 4] of a set of similar objects. A prototype is an object such that the sum of (or the sum of squares of) the distances between each object and the prototype is minimal. So far this notion has not been upgraded to first order representations.

In this paper we develop a framework for distances between clauses and distances between models. The framework can be parametrised by a measure for the distance between atoms. It is general enough to be applied both for distances between clauses and distances between models. It takes into account subterms common to distinct atoms of a set of atoms in the measurement of the distance between sets. Moreover, for a constant number of variables, the complexity of the distance computation is polynomially bounded by the size of the objects. Initial experiments show that the framework can be the basis of good clustering algorithms.

We recall some basic concepts about distances, prototypes, logic and flow networks in section 2. In section 3 we give some distance functions between sets of unlabeled elements which will be useful in further sections. In section 4 we recall some distances between atoms. We also propose some prototype functions for some of these distances. In section 5, we briefly discuss the Hausdorff distance. Next, we propose a framework that is a generalisation of three polynomial time computable similarity measures proposed by Eiter and Mannila. We also introduce another instance of this framework which is a distance, while still polynomially computable. We also define a normalised distance on sets. We instantiate the general schema with a distance between

sets of atoms (either clauses or models) in section 6. Section 7 contains some results from initial experiments. We end with a brief discussion in section 8.

## 2 Preliminaries

**Definition 1 (distance)** *Given a set of objects  $O$ , a distance function  $d$  (also called a metric) is a mapping  $O \times O \rightarrow \mathbb{R}$  such that for all  $x, y, z \in O$ :*

1.  $d(x, y) \geq 0$  and  $d(x, y) = 0 \Leftrightarrow x = y$ ,
2.  $d(x, y) = d(y, x)$  (symmetry),
3.  $d(x, z) \leq d(x, y) + d(y, z)$  (triangle inequality).

**Example 1 (Manhattan and Euclidian distance)** *Let  $x = (x_1, \dots, x_n)$ ,  $y = (y_1, \dots, y_n)$  be elements of the  $n$ -dimensional Euclidian space  $E_n$ . With  $c_1, \dots, c_n$  some positive real constants (weights), the (weighted) manhattan distance ( $d_m$ ) and the (weighted) euclidian distance ( $d_e$ ) are defined as:  $d_m(x, y) = \sum_{i=1}^n c_i |x_i - y_i|$  and  $d_e(x, y) = (\sum_{i=1}^n c_i (x_i - y_i)^2)^{1/2}$ .*

**Definition 2 (prototype)** *Let  $S = \{o_1, \dots, o_n\}$  with  $o_i \in O$ .  $p$  is an prototype (also called an euclidian prototype) of  $S$  in  $O$  for the distance  $d$  iff  $p \in O$  and  $\sum_{i=1}^n d(p, o_i)^2 = \min_{x \in O} \sum_{i=1}^n d(x, o_i)^2$ . A function  $\mathcal{P}_d : 2^O \rightarrow O$  is a general prototype function (also called an euclidian general prototype function) for  $d$  if it maps each set of objects on a prototype for it. A function  $\mathcal{P}_d : O \times O \rightarrow O$  is a binary prototype function (also called an euclidian binary prototype function) for  $d$  if it maps each set of two objects on a prototype for it.*

**Definition 3 (manhattan prototype)** *Let  $S = \{o_1, \dots, o_n\}$  with  $o_i \in O$ .  $p$  is a manhattan prototype of  $S$  in  $O$  for the distance  $d$  iff  $p \in O$  and  $\sum_{i=1}^n d(p, o_i) = \min_{x \in O} \sum_{i=1}^n d(x, o_i)$ . A function  $\mathcal{P}_d : 2^O \rightarrow O$  is a general manhattan prototype function for  $d$  if it maps each set of objects on a prototype for it. A function  $\mathcal{P}_d : O \times O \rightarrow O$  is a binary manhattan prototype function for  $d$  if it maps each set of two objects on a prototype for it.*

We also recall some preliminaries from logic. We consider terms built from an enumerable set  $\mathcal{V}$  of variables, a set  $\mathcal{C}$  of constants, and a set  $\mathcal{F}$  of functors with arity  $> 0$ . A *term* is either a variable, a constant or of the form  $f(t_1, \dots, t_n)$  with  $f/n$  a functor of arity  $n$  and  $t_1, \dots, t_n$  terms. An *atom* is of the form  $p(t_1, \dots, t_n)$  with  $p/n$  an  $n$ -ary predicate symbol and  $t_1, \dots, t_n$  terms. A *literal* is an atom or its negation. A *clause* is a set of literals. A model is a set of atoms. We denote the set of all terms (ground terms) by  $\mathcal{T}$  ( $\mathcal{T}_g$ ). The sets of all atoms (ground atoms) is denoted by  $\mathcal{A}$  ( $\mathcal{A}_g$ ).

The *lgg* (Least General Generalisation) and *lss* (Least Specific Specialisation) of two atoms  $A$  and  $B$  are defined as follows:  $lgg(A, B) = G$  iff  $G \succeq A$  and  $G \succeq B$  and  $\forall L, L \succeq A$  and  $L \succeq B : L \succeq G$ .  $lss(A, B) = S$  iff  $A \succeq S$  and  $B \succeq S$  and  $\forall L, A \succeq L$  and  $B \succeq L : S \succeq L$ .

**Definition 4 (renaming substitution)** A renaming substitution is a substitution of the form  $\{x_1 \rightarrow y_1, \dots, x_n \rightarrow y_n\}$  such that  $\{x_1, \dots, x_n\}$  is a permutation of  $\{y_1, \dots, y_n\}$ .

Next, we recall some special binary relations.

**Definition 5** A relation  $f \subseteq A \times B$  between two sets  $A$  and  $B$  is a surjection if  $\forall (a, b), (c, d) \in f : (a = c \Rightarrow b = d)$  and  $\forall b \in B, \exists a \in A : (a, b) \in f$  (Fig. 1). A surjection  $f$  between  $A$  and  $B$  is fair if  $\forall x, y \in B : ||f^{-1}\{x\}| - |f^{-1}\{y\}|| \leq 1$ , so  $f$  maps the elements of  $A$  on elements of  $B$  as evenly as possible. A linking  $f \subseteq A \times B$  is a relation such that  $\forall a \in A, \exists b \in B : (a, b) \in f$  and  $\forall b \in B, \exists a \in A : (a, b) \in f$ , so all elements of  $A$  are associated with at least one of  $B$  and vice versa. A matching  $f$  between  $A$  and  $B$  is a relation such that  $\forall (a, b), (c, d) \in f : (a = c \Leftrightarrow b = d)$ , so each element of  $A$  is associated with at most one element of  $B$  and vice versa.

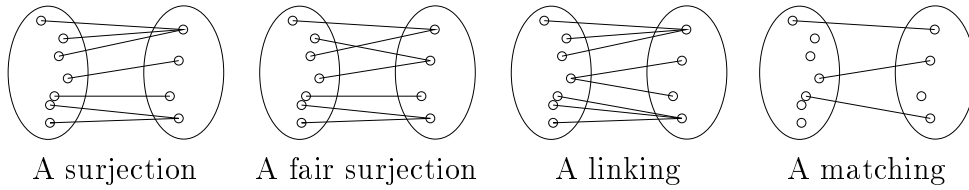


Figure 1: Examples of relations between two sets.

Finally, we recall some definitions on transport networks from [15]

**Definition 6 (indegree and outdegree)** If  $(V, E)$  is a graph and  $v \in V$ , then  $deg_{in}(v) = \#\{x \in V | (x, v) \in E\}$  and  $deg_{out}(v) = \#\{x \in V | (v, x) \in E\}$

**Definition 7 (transport network)**  $N(V, E, cap, s, t)$  is called a transport network iff  $(V, E)$  is a loop-free connected graph with  $s, t \in V$ ,  $deg_{in}(s) = 0$ ,  $deg_{out}(t) = 0$  and  $cap$  is a function  $cap : E \rightarrow \mathbb{R}^+$ .

**Definition 8 (weighted transport network)**  $N(V, E, cap, s, t, w)$  is called a weighted transport network iff  $(V, E)$  is a loop-free connected graph with  $s, t \in V$ ,  $deg_{in}(s) = 0$ ,  $deg_{out}(t) = 0$ ,  $cap$  is a function  $cap : E \rightarrow \mathbb{R}^+$  and  $w$  is a function  $w : E \rightarrow \mathbb{R}^+$ .

We say a transport network is also a weighted transport network (for  $w = 1$ )

**Definition 9 (flow)** If  $N(V, E, cap, s, t, w)$  is a weighted transport network, then a function  $f$  from  $E$  to  $\mathbb{R}$  is a flow for  $N$  iff

- $\forall e \in E : f(e) \leq cap(e)$
- $\forall v \in V \setminus \{s, t\} : \sum_{u \in V} f(v, u) = \sum_{u \in V} f(u, v)$  (if there is no edge  $(v, u) \in E$ , then  $f(v, u) = 0$ ). This is called the continuity property.

**Definition 10 (value of a flow)** If  $f$  is a flow for  $N(V, E, cap, s, t, w)$ , then  $val(f) = \sum_{v \in V} f(s, v)$  is called the value of  $f$

**Definition 11 (weight of a flow)** if  $N(V, E, cap, s, t, w)$  is a transport network, and  $f$  is a flow for  $N$ , then the weight of  $f$  is  $w(f) = \sum_{e \in E} w(e) \cdot f(e)$ .

**Definition 12 (maximal flow minimal weight flow)** If  $N(V, E, cap, s, t, w)$  is a transport network, and  $f$  is a flow for  $N$ , then  $f$  is called a maximal flow if for all flows  $f'$  for  $N$ ,  $val(f') \leq val(f)$ .  $f$  is called a maximal flow minimal weight flow iff for all maximal flows  $f'$  for  $N$ ,  $w(f') \geq w(f)$ .

In [11] the following theorem is proved:

**Theorem 1** If  $N(V, E, cap, s, t, w)$  is an integer flow network (i.e.  $\forall u \forall v : cap(u, v) \in \mathbb{N}$ ), then there is an integer maximal flow minimal weight flow for  $N$  (i.e.  $\forall u \forall v : f(u, v) \in \mathbb{N}$ )

### 3 Distances between sets of unlabeled elements

**Definition 13 (measure)** Let  $U$  be a universe. A measure is a function  $c : 2^U \rightarrow \mathbb{R}$  such that  $\forall X \in 2^U : c(X) \geq 0 \wedge (c(X) = 0 \Leftrightarrow X = \{\})$  and  $\forall X_i \subseteq 2^U : (\forall i, j : X_i \cap X_j = \phi) \Rightarrow \sum_i \#X_i = \#(\cup_i X_i)$ .

In what follows we assume that we have chosen some universe  $U$  and measure  $c$ .

**Definition 14** Given two sets  $A$  and  $B$ , their symmetric difference  $A \Delta B$  is defined by  $A \Delta B = (A \setminus B) \cup (B \setminus A)$ . We also define  $\Delta_c(A, B) = c(A \Delta B)$ .

**Lemma 1** The following property holds:

$$c(A \cup B) = [c(A) + c(B) + c(A \Delta B)]/2$$

PROOF:  
We have

$$\begin{aligned} c(A \cup B) &= c((A \setminus B) \cup B) \\ &= c(A \setminus B) + c(B) \end{aligned}$$

and

$$\begin{aligned} c(A \cup B) &= c((B \setminus A) \cup A) \\ &= c(B \setminus A) + c(A) \end{aligned}$$

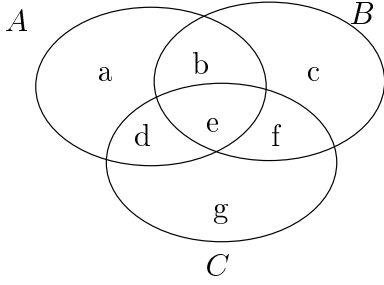


Figure 2: Figure for the proof of theorem 3

from which we derive

$$\begin{aligned}
c(A \cup B) &= [c(B \setminus A) + c(A) + c(A \setminus B) + c(B)]/2 \\
&= [c(A) + c(B) + (c(B \setminus A) + c(A \setminus B))]/2 \\
&= [c(A) + c(B) + c(A \Delta B)]/2
\end{aligned}$$

This proves the lemma.  $\square$

**Theorem 2**  $\Delta_c$  is a distance on  $2^U$ .

PROOF:

We prove that for all  $A, B$  and  $C$ ,  $\Delta_c(A, B) + \Delta_c(B, C) \leq \Delta_c(A, C)$ :

$$\begin{aligned}
\Delta_c(A, C) &= c((A \setminus C) \cup (C \setminus A)) \\
&= c(A \setminus C) + c(C \setminus A) \\
&= c((A \setminus C) \setminus B) + c((A \setminus C) \cap B) + c((C \setminus A) \setminus B) + c((C \setminus A) \cap B) \\
&= c((A \setminus C) \setminus B) + c((A \cap B) \setminus C) + c((C \setminus A) \setminus B) + c((C \cap B) \setminus A) \\
&\leq c(A \setminus B) + c(B \setminus C) + c(C \setminus B) + c(B \setminus A) \\
&= \Delta_c(A, B) + \Delta_c(B, C)
\end{aligned}$$

$\square$

**Definition 15** We define  $\Delta_{c,n}(A, B) = \frac{\Delta_c(A, B)}{c(A \cup B)}$ .

**Theorem 3**  $\Delta_{c,n}$  is a distance on  $2^U$ .

PROOF:

We assume that  $a, b, c, d, e, f$  and  $g$  are abbreviations for the measures of the parts of the sets as in figure 2, i.e.  $a = c((A \setminus B) \setminus C)$ ,  $b = c((A \cap B) \setminus C)$ , etc. We know all those numbers are positive.

$$\begin{aligned}
\Delta_{c,n}(A, B) + \Delta_{c,n}(B, C) &= \frac{c(A \setminus B) + c(B \setminus A)}{c(A \cup B)} + \frac{c(B \setminus C) + c(C \setminus B)}{c(B \cup C)} \\
&= \frac{a + d + c + f}{a + b + c + d + e + f} + \frac{b + c + d + g}{b + c + d + e + f + g}
\end{aligned}$$

$$\begin{aligned}
&\geq \frac{a+f}{a+b+e+f} + \frac{b+g}{b+e+f+g} \\
&\geq \frac{a+f}{a+b+e+f+g} + \frac{b+g}{b+e+f+g+a} \\
&= \frac{a+f+b+g}{a+b+e+f+g} \\
&\geq \frac{a+f+b+g}{a+b+d+e+f+g} \\
&= \Delta_{c,n}(A,C)
\end{aligned}$$

This proves the theorem  $\square$

**Theorem 4** *Given six positive real numbers  $x, y, z, d_{xy}, d_{yz}$  and  $d_{zx}$  such that  $|x-y| \leq d_{xy} \leq x+y$ ,  $|y-z| \leq d_{yz} \leq y+z$ ,  $|z-x| \leq d_{zx} \leq z+x$ ,  $d_{xy} + d_{yz} \geq d_{zx}$ ,  $d_{yz} + d_{zx} \geq d_{xy}$ ,  $d_{zx} + d_{xy} \geq d_{yz}$ , then one can construct a measure  $c$  and three sets  $A, B$  and  $C$  such that  $c(A) = x$ ,  $c(B) = y$ ,  $c(C) = z$ ,  $c(A\Delta B) = d_{xy}$ ,  $c(B\Delta C) = d_{yz}$  and  $c(C\Delta A) = d_{zx}$ .*

PROOF: Using the same letters as in the proof of theorem 3, we have:

$$\begin{cases}
a+b+d+e = x \\
b+c+e+f = y \\
d+e+f+g = z \\
a+d+c+f = d_{xy} \\
a+b+f+g = d_{zx} \\
b+c+d+g = d_{yz}
\end{cases} \quad (1)$$

And we have to find a solution such that  $a, b, c, d, e, f$  and  $g$  are positive.

Combining some equations we get

$$\begin{aligned}
2b+2e &= x+y-d_{xy} \\
2f+2e &= y+z-d_{yz} \\
2d+2e &= z+x-d_{zx}
\end{aligned}$$

We chose  $e = \frac{1}{2} \min\{x+y-d_{xy}, y+z-d_{yz}, z+x-d_{zx}\}$ . Without loss of generality we can suppose that  $e = \frac{1}{2}(x+y-d_{xy})$ ,  $2e \leq y+z-d_{yz}$  and  $2e \leq z+x-d_{zx}$ , from which we can conclude  $b=0$ ,  $f \geq 0$  and  $g \geq 0$ . Our constraints (1) simplify to

$$\begin{aligned}
2a+2d &= x-y+d_{xy} \\
2c+2f &= y-x+d_{xy} \\
2d+2f+2g &= 2z-x-y+d_{xy} \\
2f &= z-d_{yz}-x+d_{xy} \\
2d &= z-d_{zx}-y+d_{xy} \\
2e &= x+y-d_{xy} \\
2b &= 0
\end{aligned}$$

Now we have already  $f \geq 0$ ,  $d \geq 0$ ,  $e \geq 0$ ,  $b \geq 0$ . Substituting  $f$  and  $d$  in the first three equations, we get

$$\begin{aligned} 2a + z - d_{zx} &= x \\ 2c + z - d_{yz} &= y \\ -d_{zx} - d_{yz} + d_{xy} + 2g &= 0 \end{aligned}$$

from which we see that also  $a$ ,  $c$  and  $g$  are positive.

This proves the theorem. □

## 4 Distances between atoms

Nienhuys-Cheng [12] defines a distance on  $\mathcal{E}_g$ , the set of ground expressions

**Definition 16** ( $d_{nc,g}$ ) *The distance  $d_{nc,g}$  is a function  $d_{nc,g} : \mathcal{E}_g \times \mathcal{E}_g \rightarrow \mathbb{R}$ , such that*

1.  $d_{nc,g}(e, e) = 0$
2.  $p/m \neq q/n \Rightarrow d_{nc,g}(p(s_1, \dots, s_m), q(t_1, \dots, t_n)) = 1$
3.  $d_{nc}(p(s_1, \dots, s_n), p(t_1, \dots, t_n)) = \frac{1}{2n} \sum_{i=1}^n w_{p/n,i} d_{nc,g}(s_i, t_i)$  with  $\sum_{i=1}^n w_{p/n,i} < 1$ .

We extend this to a distance function which can also handle non-ground expressions:

**Definition 17** *The distance  $d_{nc}$  is a function  $d_{nc} : \mathcal{E} \times \mathcal{E} \rightarrow \mathbb{R}$ , such that*

1.  $d_{nc}(e_1, e_2) = d_{nc,g}(e_1, e_2)$  iff  $e_1, e_2 \in \mathcal{E}_g$ .
2.  $d_{nc}(p(s_1, \dots, s_n), X) = d(X, p(s_1, \dots, s_n)) = 1$  with  $X$  a variable or identifier.
3.  $d_{nc}(X, Y) = 1$  and  $d(X, X) = 0$  for all  $X \neq Y$  with  $X$  and  $Y$  variables or identifiers.

It is straightforward to adapt the proof of [12] that  $d_{nc,g}$  is a distance to  $d_{nc,g}$ . Observe that  $d_{nc}$  and  $d_{nc,g}$  are bounded:  $0 \leq d_{nc} \leq 1$ .

We also define a general manhattan prototype function  $\mathcal{P}_{d_{nc}}$  for  $d_{nc}$ . Therefor we define a function  $P_{d_{nc},h}$  which takes as input a set  $S$  of expressions and returns a prototype of  $S$  for  $d_{nc}$  and the sum of distances between the prototypes and the elements of  $S$ .

function  $P_{d_{nc},h}(S : \text{multiset of expressions})$

- $\{S_1, \dots, S_k\}$  is partition of  $S$  in multisets of expressions the same top level functor. Let  $S_i = \{t_{i,1}, \dots, t_{i,m_i}\}$
- for all  $i \in \{1, \dots, k\}$  do
  - $f/n = \text{functor}(t_{i,1})$
  - for  $j = 1..n$  do
    - \*  $A_{i,j} =$  the multiset of elements  $t/[j]$  such that  $t \in S_i$ .
    - \* Let  $(t_{i,j}, D_{i,j}) = P_{nc,h}(A_{i,j})$
  - $t_i = f(t_{i,1}, \dots, t_{i,a_i})$
  - $D_i = \#S - \#S_i + \frac{1}{2n} \sum_{j=1}^a D_{i,j}$
- Let  $I$  be such that  $D_I = \min_{i=1, \dots, k} D_i$ .
- return  $(t_I, D_I + D)$

The execution time of this function is at most linear in the sum of the description lengths of the objects. Setting  $\mathcal{P}_{d_{nc}} = t_I$  with  $P_{d_{nc},h} = (t_I, D_I)$  gives a good prototype function.

Hutchinson [9] starts with defining a size for substitutions:

**Definition 18** *A real-valued function  $S$  on substitutions is called a size iff*

- for all substitutions  $\theta$ ,  $S(\theta) \leq 0$ .
- $S(\epsilon) = 0$  with  $\epsilon$  the identity substitution.
- for any terms  $u, v$  and  $w$ : if  $\theta_{ab}$ ,  $\theta_{bc}$  and  $\theta_{ac}$  are substitutions such that  $a\theta_{ab} = b$ ,  $a\theta_{ac} = c$  and  $b\theta_{bc} = c$ , then  $S(\theta_{ac}) \leq S(\theta_{ab}) + S(\theta_{bc})$  and  $S(\theta_{bc}) \leq S(\theta_{ac})$
- if  $u$  and  $v$  are terms that can be unified,  $\theta_{us}, \theta_{vs}$ , are substitutions such that  $u\theta_{us} = s$  and  $v\theta_{vs} = s$ , and if  $g = \text{lgg}(u, v)$  and  $\theta_{gu}, \theta_{gv}$  are most general substitutions such that  $g\theta_{gu} = u$ ,  $g\theta_{gv} = v$ , then  $S(\theta_{gu}) + S(\theta_{gv}) \leq S(\theta_{us}) + S(\theta_{vs})$ .

Given positive weights  $w_{f/n}$  for all functors  $f/n$ , Set  $S(\theta) = \sum \{w_{f/n} | \exists x(x \text{ is a variable and } f/n \text{ occurs in } x\theta)\}$ . Based on this size, he defines a distance function between atoms:

**Definition 19**  $d_{ah,S}(u, v) = S(\theta_u) + S(\theta_v)$  with  $\theta_u$  and  $\theta_v$  substitutions such that  $g\theta_u = u$  and  $g\theta_v = v$  with  $g = \text{lgg}(u, v)$ .

We extend this result with a binary prototype function for  $d_{ah,S}$ .

**Definition 20** ( $\mathcal{P}_{d_{ah}}$ )  $\mathcal{P}_{d_{ah}}$  is a function  $\mathcal{P}_{d_{ah}} : \mathcal{E} \times \mathcal{E} \rightarrow \mathcal{E}$  such that  $\mathcal{P}_{d_{ah}}(u, v) = \text{lgg}(u, v)$ .

One can show that  $\mathcal{P}_{d_{ah}}(u, v)$  is a binary manhattan prototype for  $\{u, v\}$  for the distance  $d_{ah,S}$  for any size  $S$ .

In [14], a third distance is defined, which can be seen as an extension to [12].

## 5 Distances between sets of points

**The Hausdorff distance.** Well known is the Hausdorff distance (e.g. [12]). Given  $X$ , a set of points, and  $d$ , a distance function between points,  $d_h : 2^X \times 2^X \rightarrow \mathbb{R}$  is defined as:

$$d_h(A, B) = \max \left( \max_{a \in A} (\min \{d(a, b) | b \in B\}), \max_{b \in B} (\min \{d(a, b) | a \in A\}) \right)$$

While this function has all the properties of a distance function, it does not take into account much information about the points in the sets (it is determined by the distance of the most distant element of both sets to the nearest neighbour in the other set). Therefore, it is not very well suited for applications in first order logic, where it is plausible that two sets of atoms that differ only (but perhaps very strongly) in one atom are very similar. So it is very desirable to have a better distance function.

**Manhattan distances based on optimal mappings.** Eiter and Manila [7] discuss a family of Manhattan measures between sets which we can formulate as instances of the following scheme:

$$d^\beta(A, B) = \min_{r \in m^\beta(A, B)} \left\{ \left[ \sum_{(x, y) \in r} d(x, y) \right] + \frac{\#(B \setminus r(A)) + \#(A \setminus r^{-1}(B))}{2} \cdot M \right\}$$

where  $m^\beta$  is a function that maps each pair  $(A, B) \in 2^X \times 2^X$  to a relation between  $A$  and  $B$  (a subset of  $A \times B$ ),  $M$  is a constant (representing a large or the maximal distance between 2 points),  $r(A) = \{b | (a, b) \in r \wedge a \in A\}$ , and  $r^{-1} = \{(b, a) | (a, b) \in r\}$ .

This means that one sums the distances of the pairs of elements which are in  $r$  and adds a penalty  $M/2$  for each element that does not match with an element from the other set.

The authors discuss three instantiations:

- $m^\beta = m^s$  with  $m^s(A, B)$  the set of all surjections from the larger of  $A$  and  $B$  to the smaller of  $A$  and  $B$  (surjection-measure  $d^s$ ).
- $m^\beta = m^{sf}$  with  $m^{sf}(A, B)$  the set of all fair surjections from the larger of  $A$  and  $B$  to the smaller of  $A$  and  $B$  (fair surjection-measure  $d^{fs}$ ).
- $m^\beta = m^l$  with  $m^l(A, B)$  the set of all linkings between  $A$  and  $B$  (linking-measure  $d^l$ ).

They show that these measures can be evaluated in polynomial time, but are not distance functions (the triangle inequality is violated). Using matchings ( $m^\beta = m^m$  with  $m^m(A, B)$  the set of all matchings between  $A$  and  $B$ ) one obtains another instantiation for which we prove in [13] the following theorems:

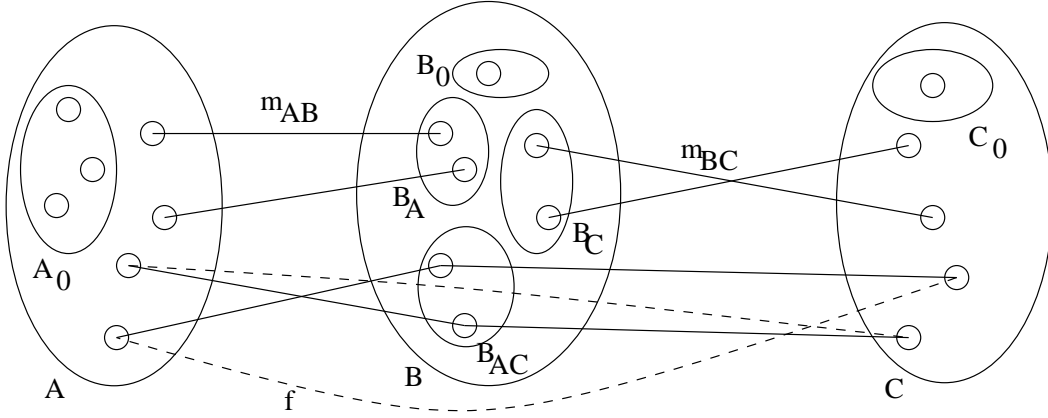


Figure 3: triangle inequality for matchings-distance

**Theorem 5**  $d^m(A, B)$  is a distance function.

PROOF:  $d^m(X, Y) = 0 \Leftrightarrow X = Y$  follows trivially from the definition and the fact that  $d$  is a good distance.

$d^m(X, Y) = d^m(Y, X)$  follows trivially from the definition.

There only remains to prove that  $d^m(A, C) \leq d^m(A, B) + d^m(B, C)$ . Given three object sets  $A, B, C \in 2^X$ . Let  $m_{AB}$  be an optimal matching<sup>1</sup> between  $A$  and  $B$  and  $m_{BC}$  be an optimal matching between  $B$  and  $C$ . So

$$d^m(A, B) = \left[ \sum_{(x,y) \in m_{AB}(B)} d(x, y) \right] + \frac{\#(A \setminus m_{AB}^{-1}(B)) + \#(B \setminus m_{AB}(A))}{2} .M \quad (2)$$

$$d^m(B, C) = \left[ \sum_{(x,y) \in m_{BC}(B)} d(x, y) \right] + \frac{\#(B \setminus m_{BC}^{-1}(C)) + \#(C \setminus m_{BC}(B))}{2} .M \quad (3)$$

Let  $A_0 = A \setminus m_{AB}^{-1}(B)$ ,  $C_0 = C \setminus m_{BC}(B)$ ,  $B_{AC} = m_{AB}(A) \cap m_{BC}^{-1}(C)$ ,  $B_A = m_{AB}(A) \setminus B_{AC}$ ,  $B_C = m_{BC}^{-1}(C) \setminus B_{AC}$ ,  $B_0 = B \setminus (B_{AC} \cup B_A \cup B_C)$  (see Fig. 3).

We have from (2) and (3)

$$d^m(A, B) = \left[ \sum_{(x,y) \in m_{AB}} d(x, y) \right] + \frac{\#B_0 + \#B_C + \#A_0}{2} .M \quad (4)$$

$$d^m(B, C) = \left[ \sum_{(x,y) \in m_{BC}} d(x, y) \right] + \frac{\#B_0 + \#B_A + \#C_0}{2} .M \quad (5)$$

Let  $m_{AC} = m_{BC} \circ m_{AB}$ . This is a matching between  $A$  and  $C$ . Then  $m_{AC}^{-1}(C) = m_{AB}^{-1}(B_{AC})$  and  $m_{AC}(A) = m_{BC}(B_{AC})$ . As the optimal matching

<sup>1</sup>the matching which minimises the formula

is at least as good, we have

$$\begin{aligned} d^m(A, C) &\leq \left[ \sum_{(x,z) \in m_{AC}} d(x, z) \right] + \frac{\#(A \setminus m_{AC}^{-1}(C)) + \#(C \setminus m_{AC}(A))}{2} \cdot M \\ &\leq \left[ \sum_{(x,z) \in m_{AC}} d(x, z) \right] + \frac{(\#B_A + \#A_0) + (\#B_C + \#C_0)}{2} \cdot M \end{aligned}$$

$d(x, z) \leq d(x, m_{AB}(x)) + d(m_{AB}(x), m_{AC}(x))$  as  $z = m_{AC}(x)$  and  $d$  is a distance. Thus

$$\begin{aligned} d^m(A, C) &\leq \sum_{x \in m_{AC}^{-1}(C)} [d(x, m_{AB}(x)) + d(m_{AB}(x), m_{AC}(x))] \\ &\quad + \frac{\#B_A + \#A_0 + \#B_C + \#C_0}{2} \cdot M \end{aligned}$$

Now  $\#B_A + \#A_0 + \#B_C + \#C_0 \leq \#B_A + \#B_0 + \#C_0 + \#B_C + \#B_0 + \#A_0$ , so

$$\begin{aligned} d^m(A, C) &\leq \sum_{x \in m_{AC}^{-1}(C)} [d(x, m_{AB}(x)) + d(m_{AB}(x), m_{AC}(x))] \\ &\quad + \frac{\#B_A + \#B_0 + \#C_0 + \#B_C + \#B_0 + \#A_0}{2} \cdot M \\ &\leq \left[ \sum_{x \in m_{AC}^{-1}(C)} d(x, m_{AB}(x)) \right] + \frac{\#B_C + \#B_0 + \#A_0}{2} \cdot M \\ &\quad + \left[ \sum_{x \in m_{AC}^{-1}(C)} d(m_{AB}(x), m_{AC}(x)) \right] + \frac{\#B_A + \#B_0 + \#C_0}{2} \cdot M \\ &\leq \left[ \sum_{x \in m_{AB}^{-1}(B)} d(x, m_{AB}(x)) \right] + \frac{\#B_C + \#B_0 + \#A_0}{2} \cdot M \\ &\quad + \left[ \sum_{x \in m_{BC}^{-1}(C)} d(m_{BC}(x), m_{AC}(x)) \right] + \frac{\#B_A + \#B_0 + \#C_0}{2} \cdot M \\ &\leq d^m(A, B) + d^m(B, C) \end{aligned}$$

This proves the theorem.  $\square$

**Theorem 6** *If the time to compute the distance between two points is bounded by a constant, then the time to compute  $d^m(A, B)$  is bounded by a polynomial in the sum of the sizes of  $A$  and  $B$ .*

**PROOF:** Since  $d$  can be computed in unit time, we can compute all  $d(a, b)$  for  $a \in A, b \in B$  in time  $O(\#(A)\#(B))$ . Let  $A = \{a_1, \dots, a_m\}$  and  $B = \{b_1, \dots, b_n\}$ . We can construct a graph  $G = (V, E)$  such that  $V =$

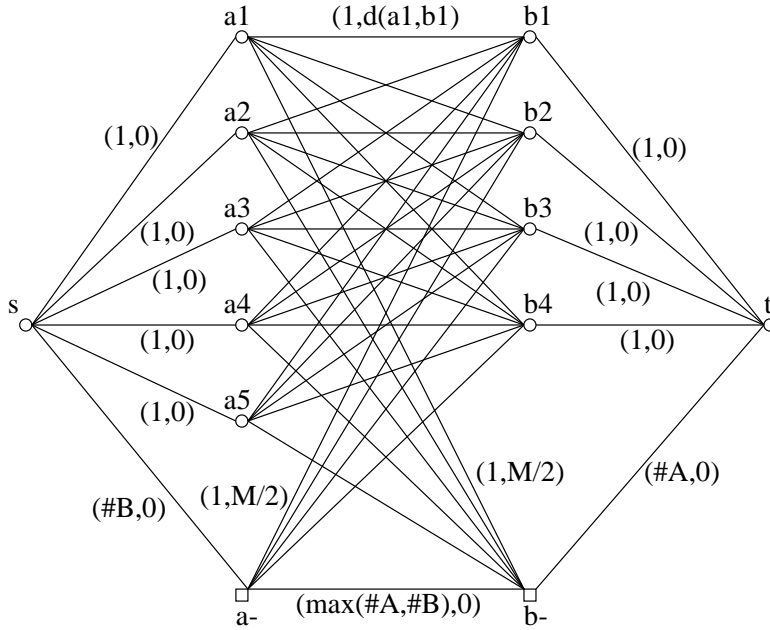


Figure 4: flow network for  $d^m(\{a_1, a_2, a_3, a_4, a_5\}, \{b_1, b_2, b_3, b_4\})$ , edges are labelled  $(cap, w)$  with  $cap$  the capacity and  $w$  the weight of the edge

$\{t, s, a_-, b_-\} \cup A \cup B$  and  $E = (\{s\} \times (A \cup \{a_-\})) \cup ((B \cup \{b_-\}) \times \{t\}) \cup ((A \cup \{a_-\}) \times (B \cup \{b_-\}))$  (see Fig. 4). We can assign a weight function  $w : E \rightarrow \mathbb{R}$  to this graph with  $\forall a \in A, \forall b \in B : w(s, a) = w(b, t) = w(s, a_-) = w(b_-, t) = w(a_-, b_-) = 0 \wedge w(a, b) = d(a, b) \wedge w(a_-, b) = w(a, b_-) = M/2$  and a capacity function  $cap : E \rightarrow \mathbb{R}$  such that  $\forall a \in A, \forall b \in B : cap(s, a) = cap(b, t) = 1 \wedge cap(s, a_-) = \#B \wedge cap(b_-, t) = \#A \wedge cap(a, b) = cap(a_-, b) = cap(a, b_-) = 1 \wedge cap(a_-, b_-) = \#A + \#B$ .

We can see that finding  $d^m(A, B)$  is equivalent to solving the minimal weight maximal flow problem from  $s$  to  $t$  in  $G$ : from theorem 1 we know that some solution of an integer network problem is an integer flow. Through each  $(s, a_i)$  and each  $(b_i, t)$  flows exactly one unit. Each flow corresponds with a matching from  $A$  to  $B$ . If a unit flows from  $a_i$  to  $b_i$  a weight  $d(a_i, b_i)$  is added to the weight of the flow. If a unit flows from  $a_i$  to  $b_-$  (or from  $a_-$  to  $b_i$ ),  $a_i$  ( $b_i$ ) is not matched with any element of  $B$  ( $A$ ) and  $M/2$  is added to the weight of the flow. Note that the maximal flow is equal to  $(\#(A) + \#(B))$ .

The minimal weight maximal flow problem is a well-known problem from graph theory (see e.g. [11] and can be solved in polynomial time. This proves the theorem.  $\square$

**Proposition 1** *If  $d(a, b)$  is invariant for renaming substitutions i.e.  $d(a, b) = d(a\theta_r, b\theta_r)$  for all atoms  $a$  and  $b$  and for all renaming substitutions  $\theta_r$  then  $d^m(A, B)$  is invariant for renaming substitutions.*

We can develop also a similar scheme for Euclidian measures (the square of the distances between elements in the relation  $m$  is used). Our result also holds for this scheme.

**Normalised matching distances.** In some applications it is desirable to work with distances in the interval  $[0, 1]$ . Having a distance between points which is in this interval, one can obtain such a distance between sets of points.  $M$  can be set to 1 when  $d(x, y)$  is bounded by 1 and the general formula for distances between sets can be simplified into:

$$d^m(A, B) = \sum_{(x,y) \in m_{AB}} d(x, y) + \frac{\#(B \setminus m_{AB}(A)) + \#(A \setminus m_{BA}(B))}{2}$$

where  $m_{BA} = m_{AB}^{-1}$  and  $m_{AB}$  is the matching which results in the minimal value.

We define

$$d^{m,n}(A, B) = \frac{2 \sum_{(a,b) \in m_{AB}} d(a, b) + \#A + \#B - 2\#m_{AB}}{\sum_{(a,b) \in m_{AB}} d(a, b) + \#A + \#B - \#m_{AB}}$$

if  $\#A + \#B > 0$  and  $d^{m,n}(\{\}, \{\}) = 0$ .

**Theorem 7**  $d^{m,n}$  is a good distance, bounded from above by 1.

PROOF:

We must prove that given three sets  $X, Y$  and  $Z$ ,  $d^{m,n}(X, Y) + d^{m,n}(Y, Z) \geq d^{m,n}(X, Z)$ .

Now let  $x = \#X$ ,  $y = \#Y$ ,  $z = \#Z$ ,  $d_{xy} = 2.d^m(X, Y)$ ,  $d_{yz} = 2.d^m(Y, Z)$  and  $d_{zx} = 2.d^m(Z, X)$ . One can verify that  $x, y, z, d_{xy}, d_{yz}$  and  $d_{zx}$  satisfy the conditions of theorem 4, so we can find sets  $A, B$  and  $C$  such that  $c(A) = \#X$ ,  $c(B) = \#Y$ ,  $c(C) = \#Z$ ,  $c(A\Delta B) = d_{xy} = 2.d^m(X, Y)$ ,  $c(B\Delta C) = 2.d^m(Y, Z)$  and  $c(C\Delta A) = 2.d^m(Z, X)$ .

We have

$$\Delta_{c,n}(A, B) = \frac{c(A\Delta B)}{c(A \cup B)}$$

applying lemma 1 we get

$$\Delta_{c,n}(A, B) = \frac{2.d^m(X, Y)}{[c(A) + c(B) + c(A\Delta B)]/2}$$

Replacing  $c(A)$ ,  $c(B)$  and  $c(A\Delta B)$  by  $\#X$ ,  $\#Y$  and  $2.d^m(X, Y)$ , we get

$$\Delta_{c,n}(A, B) = \frac{2.d^m(X, Y)}{[\#X + \#Y + 2.d^m(X, Y)]/2}$$

We fill in the definition of  $d^m$ :

$$\Delta_{c,n}(A, B) = \frac{2 \sum_{(a,b) \in m_{XY}} d(a, b) + \#X + \#Y - 2\#m_{XY}}{[\#X + \#Y + 2 \sum_{(a,b) \in m_{XY}} d(a, b) + \#X + \#Y - 2\#m_{XY}]/2}$$

from which we see

$$\Delta_{c,n}(A, B) = d^{m,n}(X, Y)$$

Similarly, we have  $\Delta_{c,n}(B, C) = d^{m,n}(Y, Z)$  and  $\Delta_{c,n}(A, C) = d^{m,n}(X, Z)$ .

We know from theorem 3 that  $\Delta_{c,n}$  is a distance, so

$$\Delta_{c,n}(A, B) + \Delta_{c,n}(B, C) \geq \Delta_{c,n}(A, C)$$

from which we can conclude the needed triangle inequality. This proves the theorem.  $\square$

**Proposition 2** *If  $d(a, b)$  is invariant for renaming substitutions then  $d^{m,n}(A, B)$  is invariant for renaming substitutions.*

**Generalisation** We now generalise the notion of matching distance using flow networks. This also solves some problems that might arise when comparing sets of different sizes. This is achieved by making it possible to normalise the flow through the objects.

**Definition 21 (Weighting function)** *A weighting function  $W$  for  $X$  is a function that maps each object  $A \in 2^X$  to a function  $W[A] : A \rightarrow \mathbb{R}$ .*

**Definition 22 (Size under weighting function)** *The size of an object  $A \in 2^X$  under a weighting function  $W$  is  $\sum_{a \in A} W[A](a)$*

**Example 2** *The function  $W_1$  with  $W_1[A] = \{(a, 1/\#A) | a \in A\}$  is a weighting function such that the size of all objects  $A \in X$  under this weighting function is 1.*

**Definition 23 (distance network)** *Given a set  $X$ , a distance  $d$  on  $X$ , a constant  $M$ , and a weighting function  $W$  for  $X$ . Then for all  $A, B \in 2^X$  with  $A = \{a_1, \dots, a_m\}$  and  $B = \{b_1, \dots, b_n\}$ , if  $Q \geq \sum_{i=1}^n W[A](b_i) + \sum_{i=1}^m W[B](a_i)$ , we define a distance network between  $A$  and  $B$  for  $d$ ,  $M$  and  $W$  in  $X$  to be  $N[X, d, M, W, A, B] = N(V, E, cap, s, t, w)$  with  $V = A \cup B \cup \{s, t, a_-, b_-\}$ ,  $E = (\{s\} \times (A \cup \{a_-\})) \cup ((B \cup \{b_-\}) \times \{t\}) \cup ((A \cup \{a_-\}) \times (B \cup \{b_-\}))$ ,  $\forall a \in A, \forall b \in B : w(s, a) = w(b, t) = w(s, a_-) = w(b_-, t) = w(a_-, b_-) = 0 \wedge w(a, b) = d(a, b) \wedge w(a_-, b) = w(a, b_-) = M/2$  and  $\forall a \in A, \forall b \in B : cap(s, a) = W[A](a) \wedge cap(b, t) = W[B](b) \wedge cap(s, a_-) = Q - \sum_{i=1}^m W[A](a_i) \wedge cap(b_-, t) = Q - \sum_{j=1}^n W[B](b_j) \wedge cap(a, b) = cap(a_-, b) = cap(a, b_-) = 1 \wedge cap(a_-, b_-) = \infty$ .*

**Proposition 3** *The weight of a maximal flow minimal weight flow of a distance network does not depend on  $Q$ .*

**Definition 24 (netflow distance)** *Given a set  $X$ , a distance  $d$  on  $X$ , a constant  $M$ , and a weighting function  $W$  for  $X$ . Then for all  $A, B \in 2^X$ , we define the netflow distance from  $A$  to  $B$  under  $d$ ,  $M$  and  $W$  in  $X$ , denoted  $d_{X,d,M,W}^N(A, B)$ , to be the weight of the minimal weight maximal flow problem from  $s$  to  $t$  in  $N[X, d, M, W, A, B]$ .*

**Notation 1** We use  $\sum_{i \in \{1..m, -\}} \text{expr}_i$  to denote  $\sum_{i=1}^m \text{expr}_i + \text{expr}_-$

**Theorem 8** *The netflow distance satisfies all properties of a distance.*

**PROOF:** We denote  $d_{X,d,M,W}^N$  by  $d^N$ . We use the notations used in definition 23 We prove the three needed properties:

- $d^N(A, A) = 0$ .  
If  $B = A$  then  $a_i = b_i$  and  $d(a_i, b_i) = 0$ . So if we let the flow consist of flows from  $s$  to  $a_i$  (or  $a_-$ ) to  $b_i$  (or  $b_-$ ) to  $t$ , then this flow has weight 0.
- $d^N(A, B) = d^N(B, A)$ .  
This follows from the fact that the solution of a minimal weight maximal flow problem has the same weight as the solution obtained with the source and sink are reversed.
- $d^N(A, B) + d^N(B, C) \geq d(A, C)$   
We can see that  $d^N(A, B) + d^N(B, C)$  is the weight of the solution of the minimal weight maximal flow problem in figure 5. Also,  $d(A, C)$  is the weight of the solution of the minimal weight maximal flow problem in figure 6 We now prove that for each flow  $f_1$  for figure 5, there exists a flow  $f_2$  for figure 6 that has the same total flow and weight which is smaller or equal.

Suppose we have a flow for figure 5. Then, let

- $f_2(s^*, a_i^*) = f_1(s, a_i)$ ,  $f_2(s^*, a_-^*) = f_1(s, a_-)$ ,  $f_2(b_i^*, t^*) = f_1(b_i, t)$  and  $f_2(b_-^*, t^*) = f_1(b_-, t)$ .
- for  $j = 1..n, -$ ,  $TB_j = \sum_{i \in \{1..m, -\}} f_1(a_i, b_j) = \sum_{k \in \{1..r, -\}} f_1(b_j, c_k)$   
The equality  $\sum_{i \in \{1..m, -\}} f_1(a_i, b_j) = \sum_{k \in \{1..r, -\}} f_1(b_j, c_k)$  holds because for  $j = 1..n$ :  $\sum_{i \in \{1..m, -\}} f_1(a_i, b_j) = f_1(b_j, r) = \text{cap}(b_j, r) = W[B](b_j) = \text{cap}(r, b'_j) = f_1(r, b'_j) = \sum_{k \in \{1..r, -\}} f_1(b'_j, c_k)$  and because  $\sum_{i \in \{1..m, -\}} f_1(a_i, b_j) = f_1(b_j, r) = \text{cap}(b_j, r) = Q - \sum_{j=1..n} W[B](b_j) = \text{cap}(r, b'_j) = f_1(r, b'_j) = \sum_{k \in \{1..r, -\}} f_1(b'_j, c_k)$  (one can prove that the edges  $(b_j, r)$  and  $(r, b'_j)$  must be saturated ( $f_1(b_j, r) = \text{cap}(b_j, r)$ ) in this graph for a manimal flow.
- $f_2(a_i^*, c_k^*) = \sum_{j \in \{1..n, -\}} \frac{f_1(a_i, b_j) \cdot f_1(b_j, c_k)}{TB_j}$

We can verify that  $f_2$  is a flow for the network in figure 6. The least trivial part of this is the continuity in  $a_i^*$  and  $c_k^*$ . For  $a_i^*$  is:

$$\begin{aligned} \sum_u f_2(u, a_i^*) - \sum_u f_2(a_i^*, u) &= f_2(s^*, a_i^*) - \sum_{k \in \{1..r, -\}} f_2(a_i^*, c_k^*) \\ &= f_1(s, a_i) - \sum_{k \in \{1..r, -\}} \sum_{j \in \{1..n, -\}} \frac{f_1(a_i, b_j) \cdot f_1(b_j, c_k)}{TB_j} \end{aligned}$$

$$\begin{aligned}
&= f_1(s, a_i) - \sum_{j \in \{1..n, -\}} \frac{f_1(a_i, b_j) \cdot \sum_{k \in \{1..r, -\}} f_1(b_j, c_k)}{TB_j} \\
&= f_1(s, a_i) - \sum_{j \in \{1..n, -\}} f_1(a_i, b_j) \\
&= 0
\end{aligned}$$

The continuity in  $c_k^*$  can be verified similarly.

We can also see that  $f_2$  is a maximal flow.

The only thing that remains to be proved is that the weight of  $f_2$  is smaller than the weight of  $f_1$ .

We have

$$\begin{aligned}
w(f_2) &= \sum_{i \in \{1..m, -\}} \sum_{k \in \{1..r, -\}} f_2(a_i^*, c_k^*) w(a_i^*, c_k^*) \\
&= \sum_{i \in \{1..m, -\}} \sum_{k \in \{1..r, -\}} \sum_{j \in \{1..n, -\}} \frac{f_1(a_i, b_j) \cdot f_1(b_j, c_k)}{TB_j} w(a_i^*, c_k^*) \\
&\quad (\text{since } w(a_i^*, c_k^*) = d(a_i^*, c_k^*) \leq w(a_i, b_j) + w(b_j, c_k) = w(a_i, b_j) + w(b_j, c_k)) \\
&\leq \sum_{i \in \{1..m, -\}} \sum_{k \in \{1..r, -\}} \sum_{j \in \{1..n, -\}} \frac{f_1(a_i, b_j) \cdot f_1(b_j, c_k)}{TB_j} [w(a_i, b_j) + w(b_j, c_k)] \\
&= \sum_{i \in \{1..m, -\}} \sum_{k \in \{1..r, -\}} \sum_{j \in \{1..n, -\}} \frac{f_1(a_i, b_j) \cdot f_1(b_j, c_k)}{TB_j} w(a_i, b_j) \\
&\quad + \sum_{i \in \{1..m, -\}} \sum_{k \in \{1..r, -\}} \sum_{j \in \{1..n, -\}} \frac{f_1(a_i, b_j) \cdot f_1(b_j, c_k)}{TB_j} w(b_j, c_k) \\
&= \sum_{i \in \{1..m, -\}} \sum_{j \in \{1..n, -\}} f_1(a_i, b_j) w(a_i, b_j) + \sum_{k \in \{1..r, -\}} \sum_{j \in \{1..n, -\}} f_1(b_j, c_k) w(b_j, c_k) \\
&= \sum_{i \in \{1..m, -\}} \sum_{j \in \{1..n, -\}} f_1(a_i, b_j) w(a_i, b_j) + \sum_{k \in \{1..r, -\}} \sum_{j \in \{1..n, -\}} f_1(b_j, c_k) w(b_j, c_k) \\
&= w(f_1)
\end{aligned}$$

This proves the theorem. □

## 6 Distances between sets of atoms

In the previous section, we developed functions for measuring the distance between two sets of points which, contrary to the Hausdorff distance, return a value which depends on all points in the two sets. In our application of interests, our “points” are atoms and the sets are either clauses or models. Simply taking an existing distance function between atoms and applying the distance functions of the previous section is not what we aim at as it ignores the similarity between different atoms in a set.

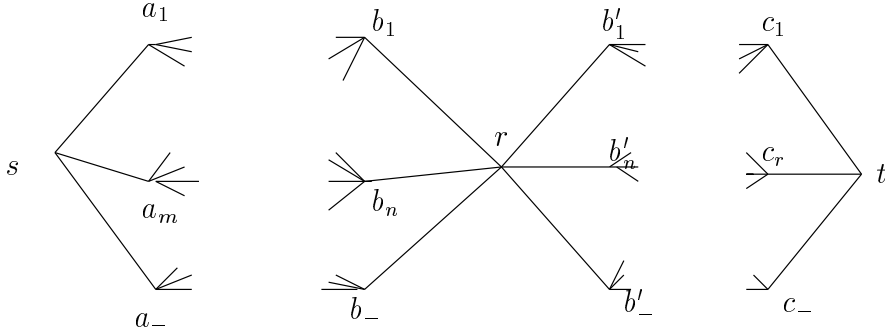


Figure 5: Network for  $d^N(A, B) + d^N(B, C)$



Figure 6: Network for  $d^N(A, C)$

**Example 3** Let  $A = \{r(xt1, xc1), p(xt1, x), q(xc1, x)\}$ ,  $B = \{r(xt2, xc2), p(xt2, y), q(xc2, y)\}$  and  $C = \{r(xt4, xc3), p(xt3, y), q(xc1, v)\}$  be sets of atoms. Applying any of the distance functions of the previous section, one obtains  $d(A, B) = d(A, C)$ . However,  $A$  is, up to renaming, equivalent to  $B$  while different from  $C$ . Note that  $A$  and  $B$  are renamings of  $\text{lgg}(A, B)$ , while  $A$  is not a renaming of  $\text{lgg}(A, C)$ .

We want to adjust the distance functions of the previous sections with a factor accounting for the recurrence of terms in different atoms of the same set. Our approach is to generalise the two sets so that their distance becomes smaller, to add factors accounting for the complexity of the substitutions needed to return to the original set and to set the cost of a renaming to 0.

In what follows,  $d^m$  is the distance function between sets of atoms based on a distance function  $d$  between atoms, e.g. as defined in the previous section. Let  $\Theta$  be a set of substitutions, and let  $\text{cost} : \mathcal{S} \rightarrow \mathbb{R}$  be a function on  $\Theta$ . A family of potential distances between sets of atoms  $A$  and  $B$  is:

$$d_{\text{cost}}^\Theta(A, B) = \min_{\theta_a, \theta_b \in \Theta; X, Y \in \mathcal{A}} d^m(X, Y) + \text{cost}(\theta_a) + \text{cost}(\theta_b)$$

where  $A = X\theta_a$ ,  $B = Y\theta_b$ .

With appropriate choices for  $\Theta$  and *cost* one can obtain that  $d_{cost}^\Theta$  is a distance function<sup>2</sup>. However we have to be careful to obtain computationally feasible solutions (we want to measure the distance between models consisting of hundreds of atoms). We should somehow limit the number of generalisations and thus the class of candidate substitutions. In the following we give an example of how this can be done

One possible choice which is in agreement with the intuitions sketched in example 3 is to allow only renaming substitutions.

Restricting the substitutions to  $\Theta_r$ , the set of all renaming substitutions, and setting the cost of substitutions to 0, we obtain a function which, as desired, returns 0 for sets of atoms which are equivalent up to a renaming of the variables.

**Theorem 9** *If  $d^m$  is invariant under renaming substitutions, then*

$$d_0^{\Theta_r}(A, B) = \min_{\theta_a, \theta_b \in \Theta_r; X, Y \in \mathcal{A}} d^m(X, Y) \text{ where } A = X\theta_a, B = Y\theta_b$$

*is a distance function on sets of atoms.*

**PROOF:** Note that the inverse of a renaming substitution is a renaming substitution and that the composition of two renaming substitutions is also a renaming substitution. To prove the triangle inequality, we have to prove:  $d_0^{\Theta_r}(A, C) \leq d_0^{\Theta_r}(A, B) + d_0^{\Theta_r}(B, C)$ .

Let  $A_B = A\theta_{ab}^{-1}$  and  $B_A = B\theta_{ba}^{-1}$  be the sets of atoms used to compute  $d_0^{\Theta_r}(A, B)$  and  $B_C = B\theta_{bc}^{-1}$  and  $C_B = C\theta_{cb}^{-1}$  be the sets of atoms used to compute  $d_0^{\Theta_r}(B, C)$  i.e.  $d_0^{\Theta_r}(A, B) = d^m(A_B, B_A)$  and  $d_0^{\Theta_r}(B, C) = d^m(B_C, C_B)$ .

Let  $A_C = A_B\theta_{ba}$  and  $C_A = C_B\theta_{bc}$ . We have:

$$d_0^{\Theta_r}(A, C) \leq d^m(A_C, C_A) = d^m(A_B\theta_{ba}, C_B\theta_{bc}) \text{ by definition of } d_0^{\Theta_r} \quad (1)$$

$$d^m(A_B\theta_{ba}, C_B\theta_{bc}) \leq d^m(A_B\theta_{ba}, B) + d^m(B, C_B\theta_{bc}) \text{ because } d^m \text{ is distance} \quad (2)$$

$$\begin{aligned} \text{Now } d^m(A_B\theta_{ba}, B) &= d^m(A_B\theta_{ba}, B\theta_{ba}^{-1}\theta_{ba}) \text{ because } \theta_{ba}^{-1}\theta_{ba} = \epsilon \\ &= d^m(A_B\theta_{ba}, B_A\theta_{ba}) = d^m(A_B, B_A) \text{ because } d^m \text{ is invariant under renaming} \\ &= d_0^{\Theta_r}(A, B) \end{aligned}$$

$$\text{Similarly: } d^m(B, C_B\theta_{bc}) = d_0^{\Theta_r}(B, C).$$

Combining this with (1) and (2) gives:

$$d_0^{\Theta_r}(A, C) \leq d_0^{\Theta_r}(A, B) + d_0^{\Theta_r}(B, C). \quad \square$$

**Example 4** *Let  $A = \{r(xt1, xc1), p(xt1, x), q(xc1, x)\}$  and  $C = \{r(xt4, xc3), p(xt3, y), q(xc1, v)\}$ . The distance is minimal with  $X = \{r(xt, xc), p(xt, x), q(xc, x)\}$  and  $Y = \{r(xt, xc), p(xt3, x), q(xc, y)\}$ .  $X$  and  $Y$  differ on one position in the  $p$  atom and one position in the  $q$  atom, so the distance will be non-zero.*

## 7 Results in practice

We present some preliminary empirical results for relational instance based learning using  $d_0^{\Theta_r}$ .

---

<sup>2</sup>With only the empty substitution in  $\Theta$  and with zero cost function we obtain  $d^m$ .

## 7.1 Mutagenesis

As a first experiment, we did simple instance based learning on the mutagenesis database [16]. Using a 9-nearest-neighbours method, we tried to predict the classes of all examples. In a first (propositional) experiment using only the numerical attributes lumo and logp in an euclidian distance, 77% of the examples was predicted correctly. Next, we performed the same experiment with only structural data (so we didn't make use of the numerical attributes lumo and logp). As instance of our scheme we used the manhattan, unnormalised, matching distance with renaming substitutions based on the distance  $d_{nc}$  from [12] between atoms. We obtained 83%. This is significantly better than what we obtained using the propositional data (the logp and lumo attributes) that correlates very well with mutagenicity. This shows that our distance performs well on pure structural data.

## 7.2 Diterpenes

The diterpene database is described in [6], where also some experimental results with FOIL, RIBL, TILDE and ICL are reported. This are all classification systems (which make use of information that assigns a class to each example in the training set during the building of the decision tree).

TIC is a clustering system based on TILDE. Clustering does not make use of class information during the building of the decision tree (See also [3] for more information on evaluating clustering trees via prediction). TIC uses a distance measure for choosing the best tests. Unfortunately, until now only euclidian distances (on the propositional part of the data) could be used. Using a first order measure much better results can be reached. We extended TIC such that our new distances can be used. We used for the atom-level a manhattan distance on the components of the atoms. For the second level, we used the distance based on matchings ( $d^m$ ) as well as the similarity measures  $d^l$  and  $d^s$  from [7]. The diterpenes database does not contain variables, so we do not need a third level for our distance.

The following table summarises the results we obtained: The first column gives the system used, the second gives the results using only the attribute-value part of the data and the third gives the results using all data.

We see that, for the second level the distance  $d^m$  performs better than the similarity measures  $d^s$  and  $d^l$ .

This result obtained by TIC which doesn't use class information is comparable to that of good classification systems which do use class information during the induction of the tree. It also shows that the use of a first order distance measure is superior to attribute-value measures, and that it is possible to construct distance measures for first order models which perform well in practice.

System	Propositional	First order
FOIL	70.1%	78.3%
RIBL	79.0%	91.2%
TILDE	78.5%	90.4%
ICL	79.1%	86.0%
TIC	78.2%	
TIC - matchings ( $d^m$ )		84.8%
TIC - linkings ( $d^l$ )		77.6%
TIC - surjections ( $d^s$ )		79.3%

## 8 Discussion

We developed a scheme for distances between clauses in three levels. On the first level one chooses a distance between atoms, e.g. [9], [12] or [14]. The second level upgrades this distance to a distance between sets of atoms. We developed a scheme for similarity between sets of points and an instance which is a real distance function, computable in polynomial time. Multiple variants are possible. We developed also a prototype function for sets of points.

Next, in the third level, we developed a variant which takes into account the “similarity” of the matched “points”, i.e. of the matched atoms in sets of atoms and their common subterms.

We did clustering experiments using instances of this scheme, i.e. distance functions between models. We obtained promising results, much better than when data are represented in an attribute value setting, and comparable to other first-order learners.

In further work, more experiments could be done to investigate the applications of the given framework. Also, we can investigate the use of the distance in [14] in this framework.

## Acknowledgements

Maurice Bruynooghe is supported by the Fund of Scientific Research, Flanders. This work is supported by the European community Esprit project no. 20237, Inductive Logic Programming 2.

## References

- [1] G. Bisson. Conceptual clustering in a first order logic representation. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 458–462. John Wiley & Sons, 1992.
- [2] H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.

- [3] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, 1998. <http://www.cs.kuleuven.ac.be/~ml/PS/ML98-56.ps>.
- [4] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.
- [5] L. De Raedt and W. Van Laer. Inductive constraint logic. In Klaus P. Jantke, Takeshi Shinohara, and Thomas Zeugmann, editors, *Proceedings of the 6th International Workshop on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*, pages 80–94. Springer-Verlag, 1995.
- [6] S. Džeroski, S. Schulze-Kremer, K. R. Heidtke, K. Siems, D. Wettschereck, and H. Blockeel. Diterpene structure elucidation from <sup>13</sup>C NMR spectra with inductive logic programming. *Applied Artificial Intelligence: Special Issue on First-Order Knowledge Discovery in Databases*, 12(5):363–384, July-August 1998.
- [7] T. Eiter and Mannila H. Distance measures for point sets and their computation. *Acta Informatica*, 34, 1997.
- [8] W. Emde and D. Wettschereck. Relational instance based learning. In *Proceedings of the 1995 Workshop of the GI Special Interest Group on Machine Learning*, 1995.
- [9] A. Hutchinson. Metrics on terms and clauses. In *Proceedings of the 9th European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, pages 138–145. Springer-Verlag, 1997.
- [10] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996.
- [11] K. Mehlhorn. *Graph algorithms and NP-completeness*, volume 2 of *Data structures and algorithms*. Springer, 1984.
- [12] Shan-Hwei Nienhuys-Cheng. Distance between herbrand interpretations: A measure for approximations to a target concept. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1997.
- [13] J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. Technical Report CW 263, Department of Computer Science, Katholieke Universiteit Leuven, 1998. <http://www.cs.kuleuven.ac.be/publicaties/rapporten/-CW1998.html>.

- [14] J. Ramon, M. Bruynooghe, and W. Van Laer. Distance measures between atoms. Technical Report CW 264, Department of Computer Science, Katholieke Universiteit Leuven, 1998. <http://www.cs.kuleuven.ac.be/publicaties/rapporten/CW1998.html>.
- [15] Grimaldi R.P. *Discrete and combinatorial mathematics*. Addison-Wesley, 1989.
- [16] A. Srinivasan, S.H. Muggleton, R.D. King, and M.J.E. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, pages 217–232. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.