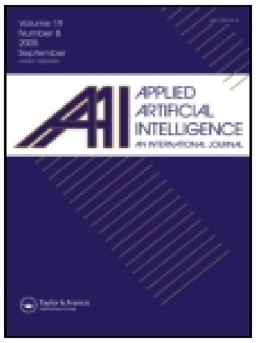

Diterpene structure elucidation from 13cnmr spectra with inductive logic programming

Saso Dzeroski , Steffen Schulze-Kremer , Karsten R. Heidtke , Karsten Siems , Dietrich Wettschereck & Hendrik Blockeel

To cite this article: Saso Dzeroski , Steffen Schulze-Kremer , Karsten R. Heidtke , Karsten Siems , Dietrich Wettschereck & Hendrik Blockeel (1998) Diterpene structure elucidation from 13cnmr spectra with inductive logic programming, Applied Artificial Intelligence, 12:5, 363-383, DOI: 10.1080/088395198117686

To link to this article: https://doi.org/10.1080/088395198117686

Published online: 26 Nov 2010.

DITERPENE STRUCTURE ELUCIDATION FROM ^{13}C NMR SPECTRA WITH INDUCTIVE LOGIC PROGRAMMING

SAŠO DŽEROSKI
Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia

STEFFEN SCHULZE-KREMER and KARSTEN R. HEIDTKE
Max-Planck Institute for Molecular Genetics, Otto-Warburg-Laboratorium, Department Lehrach, Berlin, Germany

KARSTEN SIEMS
AnalytiCon GmbH, Berlin-Wedding, Germany

DIETRICH WETTSCHERECK
GMD, FIT.KI, Schloss Birlinghoven, Sankt Augustin, Germany

HENDRIK BLOCKEEL
Department of Computer Science, Katholieke Universiteit Leuven, Heverlee, Belgium

We present a novel application of Inductive Logic Programming (ILP) to the problem of diterpene structure elucidation from ^{13}C NMR spectra. Diterpenes are organic compounds of low molecular weight with a skeleton of 20 carbon atoms. They are of significant chemical and commercial interest because of their use as lead compounds in the search for new pharmaceutical effectors. The interpretation of diterpene ^{13}C NMR spectra normally requires specialists with detailed spectroscopic knowledge and substantial experience in natural products chemistry, specifically knowledge on peak patterns and chemical structures. Given a database of peak patterns for diterpenes with known structure, we apply several ILP approaches to discover correlations between peak patterns and chemical structure. The approaches used include first-order inductive learning, relational instance based learning, induction of logical decision trees, and inductive constraint logic. Performance close to that of domain experts is achieved, which suffices for practical use.

The authors wish to thank Wim Van Laer for performing the ICL experiments. Part of this work was performed while Sašo Džeroski was a European Research Consortium for Information and Mathematics (ERCIM) fellow at the German National Research Center for Information Technology (GMD), Sankt Augustin, Germany, and Institute of Computer Science, Foundation for Research and Technology – Hellas (ICS-FORTH), Heraklion, Greece. This work was also supported in part by the ILP2 project (ESPRIT IV LTR Project 20237 Inductive Logic Programming 2).

Address correspondence to Sašo Džeroski, Department of Intelligent Systems, Jožef Stefan Institute, Jamova 39, SI-1000 Ljubljana. Email: Saso.Dzeroski@ijs.si

Applied Artificial Intelligence, 12: 363–383, 1998
Copyright © 1998 Taylor & Francis
0883-9514/98 $12.00 + .00

INTRODUCTION

NMR

Structure elucidation of compounds isolated from plants, fungi, bacteria, or other organisms is a common problem in natural product chemistry. There are many useful spectroscopic methods of getting information about chemical structures, mainly nuclear magnetic resonance (NMR) and mass spectroscopy. The interpretation of these spectra normally requires specialists with detailed spectroscopic knowledge and experience in natural products chemistry. NMR spectroscopy is the best method for complete structure elucidation (including stereochemistry) of noncrystalline samples. For structure elucidation of secondary natural products (not proteins), only ^{1}H NMR and ^{13}C NMR spectroscopy, including combined methods such as 2D NMR spectroscopy, are important because hydrogen and carbon are the most abundant atoms in natural products. In structure elucidation of peptides and proteins, ^{15}N NMR is sometimes helpful (Abraham & Loftus, 1978).

^{1}H NMR and ^{13}C NMR spectroscopy are quite different: in a ^{13}C NMR spectrum every carbon atom occurs as a separate signal in most cases, while in ^{1}H NMR spectra many signals overlap and are therefore difficult to interpret (Schulze-Kremer, 1995). ^{1}H NMR spectra are logically decomposable, and the specialist can get direct information about the structure (including stereochemistry) from the resonance frequency and shape of the signals, provided a sufficiently high resolution of resonance signals can be experimentally achieved. In ordinary ^{13}C NMR spectra, only the resonance frequency is observed, and no signal splitting (decomposition) is possible. However, the absolute value of the resonance frequency provides more information about chemical structure and (important for prediction and database maintenance purposes) is not as sensitive to varying experimental conditions (such as the magnetic strength of the NMR spectrometer or the type of solvent) as ^{1}H NMR spectroscopy is.

Additional measurements can be used to determine the number of hydrogens directly connected to a particular carbon atom. This number is the so-called multiplicity of the signal: s stands for singulet, which means there is no proton (i.e., hydrogen) connected to the carbon; d stands for a doublet with one proton connected to the carbon; t stands for a triplet with two protons and q for a quartet with three protons bound to the carbon atom. Figure 1 shows each of the four situations in the order listed above. Because of the simpler nature of ^{13}C NMR data as compared to ^{1}H NMR data, the former are easier to handle and therefore remain the preferred basis for automatic structure elucidation (Gray, 1982).

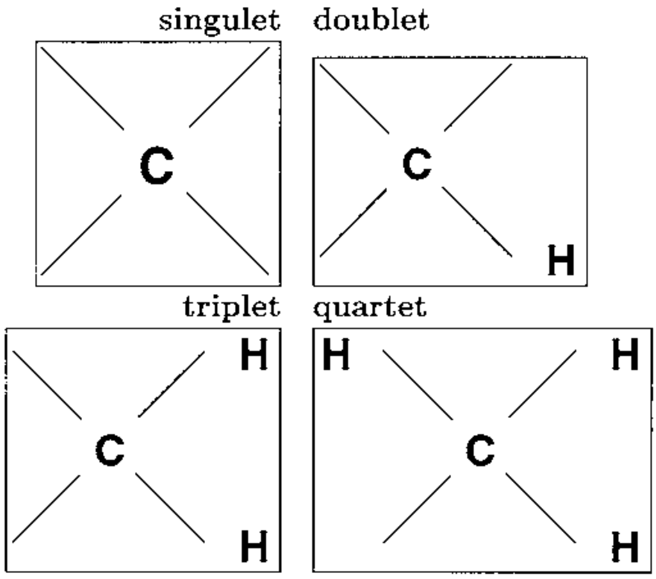

FIGURE 1. Multiplicity of carbon atoms.

Diterpenes

Diterpenes are one of a few fundamental classes of natural products with about 5000 known members (Natural, 1995). The skeleton of every diterpene contains 20 carbon atoms. Sometimes there are additional groups linked to the diterpene skeleton by an oxygen atom, with the possible effect of increasing the carbon atom count to more than 20 per diterpene. About 200 different diterpene skeletons are known so far, but some are only represented by one member compound. Most of the diterpenes belong to 1 of 20 common skeleton types.

The problem of structure elucidation of diterpenes requires knowledge about biosynthesis, the way in which biological organisms synthesize natural products. Not every combination of carbons, protons, and oxygens that is feasible from a chemical point of view actually occurs in nature, as biosynthesis in biological organisms is limited to a characteristic subset of chemical reactions that constrain the structure space. Structure elucidation of diterpenes from ^{13}C NMR spectra can be separated into three main stages: (1) identification of residues (ester and/or glycosides), (2) identification of the diterpene skeleton, and (3) arrangement of the residues on the skeleton.

This work deals with the second stage, the identification of the skeleton. A skeleton is a unique connection of carbon atoms, each with a specific atom number and, normalized to a pure skeleton molecule without residues,

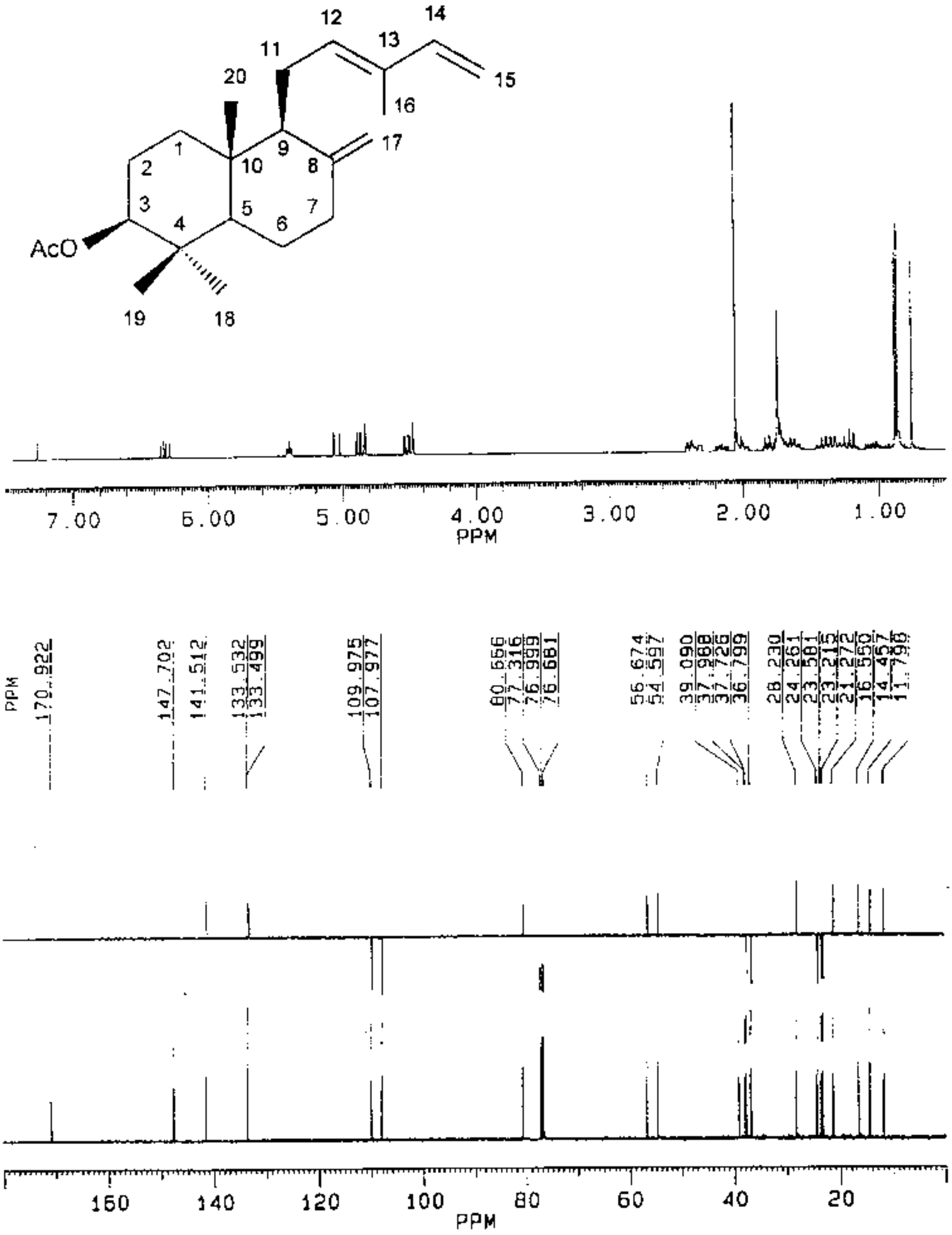

FIGURE 2. ^{1}H NMR (top) and ^{13}C NMR (bottom) spectra of a diterpene belonging to the skeleton type Labdan, with acetyl as a residue. At positions 76.681, 76.999, and 77.316 is the signal of the solvent. The other signals correspond to carbon atoms. The additional measurements (above the ^{13}C NMR spectrum) show the multiplicity of carbon atoms.

a certain multiplicity (s, d, t, or q). Figure 2 shows the ^{1}H NMR and ^{13}C NMR spectra of a diterpene belonging to the skeleton type Labdan. The structure of the diterpene is also depicted, giving the arrangement and numbering of the 20 skeletal carbon atoms.

The task we address is to identify the skeleton (type) of diterpenoid compounds, given their ^{13}C NMR spectra that include the multiplicities and the frequencies of the skeleton atoms. This task is usually done manually by human experts with specialized background knowledge on peak patterns and chemical structures. In the process, each of the 20 skeletal atoms is assigned an atom number that corresponds to its proper place in the skeleton, and the diterpene is classified into one of the possible skeleton types. We address the problem of learning classification rules from a database of peak patterns for diterpenes with known structure. Recently, propositional methods were successfully applied to classifying spectra with assigned atom numbers (Džeroski et al., 1996). As the assignment of atom numbers is a difficult process in itself (and possibly indistinguishable from the classification process), we apply Inductive Logic Programming (ILP) (Lavrač & Džeroski, 1994), i.e., relational learning, to the problem of classifying spectra without assigned atom numbers.

The rest of the article is organized as follows. The next section describes the database and the preprocessing of the data. This is followed by a summary of recent work on applying propositional learning in this domain that uses data with assigned atom numbers. When only multiplicities and frequencies, but no atom numbers are available, we have an ILP problem. Several formulations of this problem and the results of applying FOIL (Quinlan, 1990) to these formulations are presented next. Also given is a comparison to C4.5 (Quinlan, 1993), nearest neighbor classification (Wettschereck, 1994), and neural networks (Zell et al., 1995) applied to different propositionalizations of the problem. Subsequent sections present the results of applying RIBL (Emde & Wettschereck, 1996), TILDE (Blockeel & De Raedt, 1997a), and ICL (De Raedt & Van Laer, 1995) to the ILP problem formulations. The final section concludes with a discussion and an outline of directions for further work.

DATA

Database

AnalytiCon GmbH maintains a database of diterpenoid compounds. This is done using the Integrated Scientific Information System (ISIS), a product of MDL Information Systems, Inc., on IBM PCs under MS Windows. The relational database contains information on 1503 diterpenes with known structure, stored in three relations—`atom`, `bond`, and `nmr`.

The first relation specifies to which element an atom in a given compound belongs. The second relation specifies which atoms are bound and in what way in a given compound. The `nmr` relation stores the measured ^{13}C NMR spectra. For each of the 20 carbon atoms in the diterpene skeleton, it contains the atom number, its multiplicity, and frequency. For each compound, the skeleton type that it represents is also specified within the database. Table 1 gives an excerpt from the `nmr` relation describing molecule `v1`. The relation schema is `nmr(MoleculeID, SkeletonType, AtomID, AtomNumber, Multiplicity, Frequency)`.

Preprocessing

Every substituent or residue connected to a carbon atom exerts a characteristic shift on the resonance frequency signal of the carbon atom and sometimes also changes its multiplicity. Simple rules based on expert knowledge can be used to take this effect into account.

They transform the raw, measured, NMR multiplicities into the so-called reduced multiplicities, which carry more information about the skeleton types, as shown below. These rules are given in Table 2. They leave the measured frequencies unchanged. The reduced multiplicities that correspond

TABLE 1 Facts Describing the ^{13}C NMR Spectrum of Molecule `v1` That Belongs to Skeleton Type Labdan (52)

```
nmr(v1,52,v1_1,1,t,39.00).
nmr(v1,52,v1_2,2,t,19.30).
nmr(v1,52,v1_3,3,t,42.20).
nmr(v1,52,v1_4,4,s,33.30).
nmr(v1,52,v1_5,5,d,55.60).
nmr(v1,52,v1_6,6,t,24.30).
nmr(v1,52,v1_7,7,t,38.30).
nmr(v1,52,v1_8,8,s,148.60).
nmr(v1,52,v1_9,9,d,57.20).
nmr(v1,52,v1_10,10,s,39.70).
nmr(v1,52,v1_11,11,t,17.60).
nmr(v1,52,v1_12,12,t,41.40).
nmr(v1,52,v1_13,13,s,73.50).
nmr(v1,52,v1 14,14,d,145.10).
nmr(v1,52,v1_15,15,t,111.40).
nmr(v1,52,v1 16,16,q,27.90).
nmr(v1,52,v1_17,17,t,106.30).
nmr(v1,52,v1_18,18,q,33.60).
nmr(v1,52,v1_19,19,q,21.60).
nmr(v1,52,v1_20,20,q,14.40).
```

TABLE 2 Rules for Generating Reduced Multiplicities from Observed Multiplicities

```
IF ObservedM = s AND Frequency in [ 64.5, 95 ]  THEN ReducedM = d
IF ObservedM = s AND Frequency in [ 96, 114 ]   THEN ReducedM = t
IF ObservedM = s AND Frequency in [ 115, 165 ]  THEN ReducedM = d
IF ObservedM = s AND Frequency in [ 165, 188 ]  THEN ReducedM = q
IF ObservedM = s AND Frequency in [ 188, inf ]  THEN ReducedM = t
IF ObservedM = d AND Frequency in [ 64.5, 95 ]  THEN ReducedM = t
IF ObservedM = d AND Frequency in [ 105, 180 ]  THEN ReducedM = t
IF ObservedM = d AND Frequency in [ 96, 104 ]   THEN ReducedM = q
IF ObservedM = d AND Frequency in [ 180, inf ]  THEN ReducedM = q
IF ObservedM = t AND Frequency in [ 59, 90 ]    THEN ReducedM = q
IF ObservedM = t AND Frequency in [ 90, inf]    THEN ReducedM = q
```

to those in Table 1 are given in Table 3. Note that the multiplicities of atoms 8, 13, 14, 15, and 17 are changed.

PROPOSITIONAL LEARNING

Given the above data, Džeroski et al. (1996) formulate several propositional learning problems. Twenty-three different skeleton types are represented in the whole set of 1503 compounds: there are thus 23 possible class values. The attributes are the multiplicities and frequencies of each of the 20 skeleton atoms. For instance, attribute `A1M` is the multiplicity of atom

TABLE 3 Preprocessed Facts Describing the ^{13}C NMR Spectrum of Molecule `v1`

```
red(v1, 52, v1_1, 1, t, 39.00).
red(v1, 52, v1_2, 2, t, 19.30).
red(v1, 52, v1_3, 3, t, 42.20).
red(v1, 52, v1_4, 4, s, 33.30).
red(v1, 52, v1_5, 5, d, 55.60).
red(v1, 52, v1_6, 6, t, 24.30).
red(v1, 52, v1_7, 7, t, 38.30).
red(v1, 52, v1_8, 8, d, 148.60).
red(v1, 52, v1_9, 9, d, 57.20).
red(v1, 52, v1_10, 10, s, 39.70).
red(v1, 52, v1_11, 11, t, 17.60).
red(v1, 52, v1_12, 12, t, 41.40).
red(v1, 52, v1_13, 13, d, 73.50).
red(v1, 52, v1_14, 14, t, 145.10).
red(v1, 52, v1_15, 15, q, 111.40).
red(v1, 52, v1_16, 16, q, 27.90).
red(v1, 52, v1_17, 17, q, 106.30).
red(v1, 52, v1_18, 18, q, 33.60).
red(v1, 52, v1_19, 19, q, 21.60).
red(v1, 52, v1_20, 20, q, 14.40).
```

number 1, and A7F is the frequency of atom number 7. There are thus 40 attributes, 20 discrete (multiplicities) and 20 continuous (frequencies). At a suggestion from the domain experts, a version of the problem is also considered where only the multiplicities are used as attributes. Four series of experiments are performed, with the following sets of attributes: observed multiplicities and frequencies, reduced multiplicities and frequencies, observed multiplicities only, and reduced multiplicities only.

Three different machine learning approaches were used: backpropagation networks (Gallant, 1993; Zell et al., 1995), nearest neighbor classification (Cover & Hart, 1968; Wettschereck, 1994), and decision tree induction (Quinlan, 1986, 1993). Below, we give some details on the experimental setup.

The neural networks we experimented with had 23 output nodes, representing the 23 types of diterpene skeletons. Input and output nodes were fully interconnected. There were 80 input nodes, 4 per skeleton atom, each corresponding to the possible multiplicity values. The neuron corresponding to the actual value of AiM is fed the frequency AiF as a floating point number. When no frequencies are used, the neuron is fed a 1 instead of AiF.

The above architecture was selected after trial-and-error preliminary experiments with different architectures, which also included hidden layers. The standard activation function $1/(1 + e^{-x})$ was employed. Weights were initialized at random with values between -400 and 400. A thousand iterations of weights adjustment were performed using the delta-rule. The learning rate was set to 0.5, and the momentum parameter was set to 0.9.

The nearest neighbor (NN) algorithm treats attributes as dimensions of a Euclidean space and examples as points in this space. The distance between two examples, q and u, is defined as

$$D(q, u) = \left(\sum d(q_i, u_i)^2 \right)^{1/2}$$

where

$$d(q_i, u_i) = \begin{cases} q_i - u_i & \text{for numeric attribute } i \\ 1 & \text{if } q_i = u_i\text{, for symbolic attribute } i \\ 0 & \text{otherwise} \end{cases}$$

The kNN method takes the k nearest training examples and determines the class of a new instance by majority vote, where the votes of each of these neighbors are weighted by their respective proximity to the new instance.

Advanced implementations of the kNN algorithm determine the optimal value of k automatically from the training set by using leave-one-out cross validation. They can also weight the contribution of each attribute to the distance by the mutual information between that attribute and the class, where these so-called feature weights are determined automatically on the training set. For our experiments, we used the kNN algorithm as implemented by Wettschereck (1994), with $k = 1$ and no feature weighting. These settings performed best in preliminary experiments.

For our decision tree experiments, we used the C4.5 algorithm (Quinlan, 1993). In all experiments, we used the default parameter settings for C4.5, with one exception: groups of discrete attributes' values are also considered along branches, instead of single values only (as done by default). This produces smaller and thus more understandable trees at no cost in terms of accuracy.

Table 4 gives a summary of the accuracies on unseen cases for the three approaches and the four different problems as estimated by 10-fold cross validation on the same folds. All three approaches achieved high accuracy on unseen cases, especially when provided with reduced multiplicities. The nearest neighbor classifier performed slightly better than the decision trees, while these in turn, performed slightly better than the backpropagation networks. The decision trees induced were inspected by a domain expert and were found to be understandable and to contain valuable domain knowledge.

While the accuracies achieved are very high and the problem may be considered solved at first sight, one should bear in mind that the above propositional formulation crucially depends on the atom number information. Atom numbers are used to link the frequency and multiplicity to an individual atom in a molecule. Without atom numbers, one only knows that a certain frequency/multiplicity pair occurs for a molecule but not which atom gives rise to that frequency/multiplicity pair. The atom-frequency/multiplicity pair correspondence is of great importance because (1) neighboring atoms influence each others' frequency/multiplicity pairs and (2) a proper assignment of atoms to frequency/multiplicity pairs facilitates structure prediction.

The atom numbering is unique but is only known when the structure of the diterpene is known. For new, unknown compounds, atom numbers are not available because the NMR detector only tells us which frequency/multiplicity pairs exist but not which atom is responsible for which pair or which atoms are directly connected. If we are to fully automate the task of

TABLE 4 Classification Accuracy on Unseen Cases When Learning from Classified ^{13}C NMR Spectra with Assigned Atom Numbers

Problem/Approach	Backpropagation networks	Nearest neighbor	C4.5
Observed	89.6%	96.7%	96.4%
Reduced	95.5%	99.3%	97.6%
Observed—No Frequencies	83.9%	94.4%	93.1%
Reduced—No Frequencies	97.1%	98.8%	98.4%

classifying NMR spectra, we need to be able to perform it without atom numbers being provided. Without atom number information, however, there is no obvious propositional representation of the problem, and more careful representation engineering (feature construction) or the use of more powerful techniques, such as ILP seems necessary. In the remainder of the article, we describe experiments that apply each of these approaches separately, as well as in combination.

APPLYING FOIL

As we are dealing with a learning problem where 23 different classes exist, there are 23 target relations `classC(MoleculeID)`, where `C` ranges over the possible classes. For example, the target relation `class52(MoleculeID)` corresponds to the skeleton type Labdan, most common among the 1503 diterpenes in the database (448 instances). The correspondence between the chemical names and the codes in the data is given in Table 5 together with the number of instances of each class. Only classes with more than one instance are listed.

Using Multiplicities and Frequencies

Given that much better results were obtained using the reduced multiplicities in earlier experiments, we decided to use only these (and not the observed multiplicities) for our experiments with ILP. After one takes the atom number information away, the background relation red can be simplified to a three-argument relation `red(MoleculeID, Multiplicity, Frequency)`. A particular fact of this relation states that the ^{13}C NMR

TABLE 5 The Classes, Their Chemical Names, and Number of Instances in the Database

Code	Chemical name	Number of instances
c2	Trachyloban	9
c3	Kauran	353
c4	Beyeran	72
c5	Atisiran	33
c15	Ericacan	2
c18	Gibban	13
c22	Pimaran	155
c28	6,7-seco-Kauran	9
c33	Erythoxilan	9
c36	Spongian	10
c47	Cassan	12
c52	Labdan	448
c54	Clerodan	356
c71	Portulan	5
c79	5,10-seco-Clerodan	4
c80	8,9-seco-Labdan	6

spectrum of a given molecule contains a peak of a given multiplicity and frequency. For example, the fact `red(v1,t,39.00)` states that the ^{13}C NMR spectrum of molecule `v1` has a peak at frequency 39.00 ppm with multiplicity `t` (a triplet).

The background relation `red` is nondeterminate, as there are 20 tuples for each molecule. This prevents the use of ILP systems like GOLEM (Muggleton & Feng, 1990) or DINUS (Lavrač & Džeroski, 1994). While PROGOL (Muggleton, 1995) would be applicable, preliminary experiments showed it has prohibitive time complexity if longer rules/clauses are needed. Therefore we used FOIL (Quinlan, 1990), in particular, FOIL6.4. Except for a variable depth limit of 1, all other settings were left in their default state.

We first used FOIL to induce rules on the entire data set and then performed 10-fold cross validation on the same partitions of training examples as used for the propositional case (see the previous section) by Džeroski et al. (1996). For each experiment, FOIL was run 23 times, i.e., once for each target relation. The rules from the 23 runs were then taken together to produce a rule set. Before classification, the rules from the rule set were checked against the training set to record the number of examples of each class that they cover, as well as to note the majority class in the training set. This information is then used when classifying new examples, e.g., when testing the accuracy of the rule set.

For classification, we implemented a procedure analogous to that of CN2 (Clark & Boswell, 1991): when an example is classified, all clauses that cover it are taken into account. The distributions of examples of each class are summed for all rules that match, and the majority class is assigned to the example. If no rule matches, the majority class (from the training set) is assigned.

Given the background relation `red(MoleculeID, Multiplicity, Frequency)`, FOIL induced 90 rules from the entire data set, comprising 607 literals in their bodies. At first sight, the rules are quite specific, covering relatively few examples. Also, many examples are left uncovered. Thus, even on the training set, these rules only achieve 51.6% accuracy. The 10-fold cross validation yields 46.5% accuracy on unseen cases as the average over the 10 runs. It seems that the background knowledge is sufficient to distinguish among the different classes (the rules typically cover examples of one class only), but FOIL induces too specific rules. The most general rule `class52(A) :- red(A,d,B), B=<73.7, red(A,C,D), D>38.5, D=<44.6, B>73.2` covers 43 examples of class 52.

Using Multiplicities Only

According to the domain expert (Karsten Siems), the multiplicities should suffice for correct classification, at least when atom numbers are available (Džeroski et al., 1996). If we remove the `Frequency` argument from the relation `red`, all the information left about a particular molecule is captured by the number of atoms that have multiplicity `s`, `d`, `t`, and `q`, respectively. The relation `prop(MoleculeID, SAtoms, DAtoms, TAtoms, QAtoms)` stores this information. For our molecule `v1`, we have the fact `prop(v1,2,4,8,6)`.

Given the entire data set, the 23 target relations and the background relation prop, FOIL induced 17 rules with 52 literals in the bodies. The same settings were applied in FOIL as for the experiments with `red`. The rules induced in this case are much more general than those obtained with `red`. The most general rule `class52(A) :- prop(A,B,C,D,E), E>5, C>3, D>7, B>1` covers 429 examples (355 are of class 52, 72 of class 54). Many rules cover examples of several classes. It seems that the background knowledge is insufficient to completely distinguish among the different classes, and FOIL is forced to induce overly general rules. This, however, has a positive effect on accuracy as compared to the experiments with `red`. Using `prop`, FOIL achieves 69.0% accuracy on the entire data sets, and 70.1% accuracy on unseen cases (10-fold cross validation).

Note at this point that we have, in fact, applied FOIL to a propositional problem. Namely, from the nondeterminate representation with the `red` relation, we have constructed a four-feature representation of each molecule. A comparison to propositional learners is given in the section below, Comparing FOIL to Propositional Approaches.

Combining Engineered Features and a Relational Representation

Having introduced the four new features with the `prop` relation, which seem to capture some general properties of the molecules, we repeated the experiment with the nondeterminate relational representation. This time FOIL was given both the relation `red` and the relation prop. The same settings of FOIL were applied as in the previous two subsections.

Given the entire data set, the 23 target relations and the background relations `red` and `prop`, FOIL induced 68 rules with 401 literals in the bodies. The rules were more general than rules induced using `red` only but were more specific than rules induced using prop only. In most cases, each rule covers examples of one class only. The most general rule `class52(A) :- prop(A,B,C,D,E), E>5, C>3, D>7, B>1, red(A, d,F), F>54.8, F=<72.3` covers 227 examples of the correct class. The induced rules achieve 83.2% accuracy on the entire data set and 78.3% accuracy on unseen cases (10-fold cross validation). Combining the engineered features with the relational representation thus has a positive effect.

Comparing FOIL to Propositional Approaches

C4.5 Using Multiplicities Only

As mentioned above, the relation `prop` defines a propositional version of our classification problem. We therefore applied the propositional learner C4.5 (Quinlan, 1993) to this problem. The same experimental setting (tree induced on whole data set first, then 10-fold cross validation) and the default settings of C4.5 were used.

The induced tree achieves 80.4% accuracy on the entire data set. The leaves of the tree typically contain examples of several different classes, confirming the suspicion that the four features do not suffice for completely correct classification. The classification accuracy on unseen cases as measured by 10-fold cross validation (same folds as above) is 78.5%, which is almost the same as the accuracy achieved by FOIL using both `red` and `prop`.

Nearest Neighbor Using Multiplicities Only

We also applied the *k*NN nearest neighbor classifier (Cover & Hart, 1968; Wettschereck, 1994) to the propositional problem defined by the relation `prop`. The experimental setting from the section above, Propositional Learning (cross-validation folds) was used.

Cross validation on the number of neighbors used in classification was tried out, as well as two different forms of feature weighting, but the basic nearest neighbor method ($k = 1$, no feature weighting) performed best. The classification accuracy on unseen cases (average over the 10 folds) is 79.0%, which is almost the same as the accuracy of C4.5 and the accuracy achieved by FOIL using both `red` and `prop`.

Backpropagation Networks

Various network architectures were explored to see how, in comparison, backpropagation networks would perform at the classification of skeletons based on unassigned ^{13}C NMR data. This was done independently of the above experimental setup but using the same partitions for cross validation.

A standard backpropagation network (Gallant, 1993; Zell et al., 1995) with 960 input neurons, no hidden units, and 23 output units was trained with the same input data as for the ILP experiments. We divided the 960 input neurons into four equally large sets of 240 nodes, one each for singulets, doublets, triplets, and quadruplets. The best representation was to have for each multiplicity 24 frequency intervals (0–10, 11–20, ..., 231–240 ppm) with 10 nodes each and to feed the value of 1 for each frequency in the spectrum into the next unoccupied input neuron of the appropriate interval. All remaining input nodes are filled with zeros. Thus the artificial neural net sees a discretized distribution of multiplicity signals.

During cross validation, accuracy on the training set (average for the 10 runs) reached 97.9%, while accuracy on unseen cases reached 79.9%. Other network variations, e.g., feeding the actual frequency value into the appropriate neuron instead of 1 or using a 4×15 architecture that receives the sorted frequencies for each multilicity as input did not produce better results.

APPLYING RIBL

Relational instance-based learning, or RIBL (Emde & Wettschereck, 1996), generalizes the nearest neighbor method to a relational representation. RIBL first constructs cases by putting together all facts that relate to (in this case) a single molecule. Training cases are stored for further reference. When a new molecule has to be classified, RIBL calculates the similarities between it and each of the training cases, then assigns it the class of the nearest training case.

The similarity measure used in RIBL is a generalization of similarity measures used in propositional instance-based learners. In fact, given a propositional problem, RIBL becomes the classical nearest neighbor method and has the same performance as the latter. This is a very desirable property for a relational learner.

We used the same formulations and experimental methodology as for FOIL (previous section). As it stores all training cases, RIBL achieves 100% accuracy given the entire data set for both training and testing. To estimate accuracy on unseen cases, 10-fold cross validations on the same folds as above were performed.

Given only the `red` relation, RIBL achieved 86.5% classification accuracy (average over the 10 partitions) on unseen cases. This is an increase of 40% over the accuracy achieved by FOIL. Note that propositional approaches (e.g., C4.5) are not applicable to this formulation of the problem.

Given only the `prop` relation, RIBL behaves identically to the nearest neighbor method, thus yielding 79.0% accuracy on unseen cases, a performance equivalent to that of C4.5. When provided with both the `red` and the `prop` relations, RIBL achieves an accuracy of 91.2% on unseen cases. Using the engineered features improves RIBL's performance by roughly 5%, pushing further the best result achieved at classifying diterpene NMR spectra without assigned atom numbers. Again, propositional approaches (e.g., C4.5) are not applicable to this formulation of the problem.

APPLYING TILDE

The TILDE system (Blockeel & De Raedt, 1997*a*) is a first-order logic upgrade of the C4.5 system (Quinlan, 1993) that induces logical decision trees. It learns from interpretations (De Raedt & Džeroski, 1994), which means that the `MoleculeID` argument does not occur in the predicates;

apart from that, the same representation is used as for the FOIL experiment. We have also used the same experimental methodology: using multiplicities and frequencies, using multiplicities only by means of the engineered predicate `prop`, and combining both original and engineered predicates.

With purely propositional data, TILDE achieves exactly the same accuracies (80.4% for training on the entire data set, 78.5% for 10-fold cross validation) as C4.5, which is not unexpected, as it is an upgrade of the latter. Table 6 lists the accuracies achieved by all ILP systems applied to all problem formulations (average accuracies of 10-fold cross validation).

For the nonpropositional experiments, discretization was performed on the frequencies. The discretization algorithm used by TILDE (and ICL) is based on the algorithm of Fayyad and Irani (1993) but adapted to the ILP context; see Van Laer et al. (1997) and (Blockeel & De Raedt, 1997*b*) for details of the discretization procedure. The number of thresholds is a parameter of TILDE and was varied. Best results were obtained with five discretization tresholds, and this value was used for all subsequent experiments. Except for the number of discretization thresholds, default values were used for all parameters.

The lookahead facility (Blockeel & De Raedt, 1997*b*) of TILDE enables us to consider literals of the form `red(M,F)` together with literals of the form `between(L,U)`: this alleviates problems that would arise when the first literal is not informative. At least one of `L` and `U` should be a real number; `L` can also be `-inf`, while `U` can also be `+inf`. `M` can be either uninstantiated or take one of the values `s, d, t, q`: discretization is performed separately for each of the five cases with the same number of discretization thresholds.

An accuracy of 81.6% is achieved on the `red` formulation, and an accuracy of 90.4% on the `red+prop` formulation. The latter is essentially the same as that of RIBL (91.2%), while the former is five percentage points lower than RIBL's respective performance (86.5%).

Despite using a post pruning mechanism similar to that of C4.5, TILDE with default parameters produces trees that are very large and, consequently, difficult to interpret. We thus varied the minimal number of examples that is to be covered by each leaf (by default, this is 2) in order to obtain smaller trees. Increasing this parameter lowers predictive accuracy but also makes

TABLE 6 Accuracies of Different ILP Approaches When Classifying NMR Spectra Without Assigned Atom Numbers

Problem/System	FOIL	RIBL	TILDE	ICL
red	46.5%	86.5%	81.6%	65.3%
prop	70.1%	79.0%	78.5%	79.1%
red + prop	78.3%	91.2%	90.4%	86.0%

TABLE 7 Predictive Accuracy and Average Complexity of TILDE Trees Induced for Different Values of the Minimal Coverage Parameter

Minimal coverage	Accuracy	Complexity	Most general leaf
2	90.4%	74.8	c52 (328/330)
5	88.9%	45.7	c52 (328/330)
10	86.4%	26.8	c52 (347/357)
20	83.0%	14.9	c52 (347/357)

the trees simpler and more interpretable. Table 7 shows the average accuracy and tree complexity (number of internal nodes) of 10-fold cross validation. The last column lists the class and coverage (correct/total) of the most general leaf. Figure 3 shows the tree obtained with minimal coverage set to 20 when learning on the whole data set. For comparison, Figure 4 shows the tree obtained with minimal coverage set to 10.

```
prop(A,B,C,D)
A = 4 ?
+--yes: c4 [72 / 95]
+--no:  A = 3 ?
        +--yes: D =< 4 ?
        |       +--yes: C =< 10 ?
        |       |       +--yes: red(t,E),between(E,[99.7,148.15]) ?
        |       |       |       +--yes: red(s,F),between(F,[ - inf,38.9]) ?
        |       |       |       |       +--yes: c3 [24 / 35]
        |       |       |       |       +--no:  c54 [17 / 26]
        |       |       |       +--no:  B = 3 ?
        |       |       |               +--yes: c3 [10 / 20]
        |       |       |               +--no:  red(q,G),between(G,[ - inf,15.2]) ?
        |       |       |                       +--yes: red(s,H),between(H,[38.9,47.11]) ?
        |       |       |                       |       +--yes: c3 [50 / 62]
        |       |       |                       |       +--no:  c5 [18 / 20]
        |       |       |                       +--no:  c3 [246 / 248]
        |       |       +--no:  c54 [30 / 30]
        |       +--no:  B =< 3 ?
        |               +--yes: c22 [139 / 161]
        |               +--no:  c18 [11 / 34]
        +--no:  red(q,I),between(I,[20.97,59.5]) ?
                +--yes: B =< 4 ?
                |       +--yes: red(d,J),between(J,[49.12,93.8]) ?
                |       |       +--yes: D = 6 ?
***             |       |       |       +--yes: c52 [347 / 357]
                |       |       |       +--no:  red(d,K),between(K,[ - inf,49.12]) ?
                |       |       |               +--yes: red(L,M),between(M,[167.4,176.6])?
                |       |       |               |       +--yes: c54 [28 / 32]
                |       |       |               |       +--no:  c52 [14 / 23]
                |       |       |               +--no:  c52 [40 / 43]
                |       |       +--no:  c54 [46 / 64]
                |       +--no:  c54 [20 / 45]
                +--no:  c54 [188 / 208]
```

FIGURE 3. Tree induced by TILDE, where each leaf has to cover at least 20 examples. For each leaf, the class it predicts, the number of correctly classified instances, and the total number of instances covered are listed in the following format: class (correct/total).

```
prop(A,B,C,D)
A = 4 ?
+--yes: red(d,E),between(E,[62.15,inf]) ?
|       +--yes: c3 [4 / 11]
|       +--no:  B = 2 ?
|               +--yes: c4 [63 / 63]
|               +--no:  red(q,F),between(F,[ - inf,15.2]) ?
|                       +--yes: c2 [9 / 11]
|                       +--no:  c4 [7 / 10]
+--no:  A = 3 ?
        +--yes: D =< 4 ?
        |       +--yes: C =< 10 ?
        |       |       +--yes: red(d,G),between(G,[93.8,126]) ?
        |       |       |       +--yes: c36 [7 / 14]
        |       |       |       +--no:  red(d,H),between(H,[49.12,inf]) ?
        |       |       |               +--yes: red(d,I),between(I,[62.15,inf]) ?
        |       |       |               |       +--yes: red(q,J),between(J,[ - inf,15.2]) ?
        |       |       |               |       |       +--yes: red(s,K),between(K,[47.11,47.9]) ?
        |       |       |               |       |       |       +--yes: c5 [18 / 23]
        |       |       |               |       |       |       +--no:  c3 [50 / 62]
        |       |       |               |       |       +--no:  c3 [278 / 283]
        |       |       |               |       +--no:  c3 [9 / 19]
        |       |       |               +--no:  c54 [10 / 10]
        |       |       +--no:  c54 [30 / 30]
        |       +--no:  B = 5 ?
        |               +--yes: c18 [11 / 13]
        |               +--no:  C = 7 ?
        |                       +--yes: c28 [6 / 14]
        |                       +--no:  red(L,M),between(M,[167.4,176.6]) ?
        |                               +--yes: c54 [7 / 14]
        |                               +--no:  c22 [139 / 154]
        +--no:  red(q,N),between(N,[20.97,59.5]) ?
                +--yes: B =< 4 ?
                |       +--yes: red(d,O),between(O,[49.12,93.8]) ?
                |       |       +--yes: D = 6 ?
***             |       |       |       +--yes: c52 [347 / 357]
                |       |       |       +--no:  red(d,P),between(P,[ - inf,49.12]) ?
                |       |       |               +--yes: red(Q,R),between(R,[167.4,176.6]) ?
                |       |       |               |       +--yes: c64 [28 / 32]
                |       |       |               |       +--no:  c52 [14 / 23]
                |       |       |               +--no:  c52 [40 / 43]
                |       |       +--no:  c54 [46 / 64]
                |       +--no:  red(d,S),between(S,[148.7,inf]) ?
                |               +--yes: c47 [9 / 12]
                |               +--no:  red(t,T),between(T,[99.7,210.75]) ?
                |                       +--yes: c54 [17 / 20]
                |                       +--no:  c52 [5 / 13]
                +--no:  c54 [188 / 208]
```

FIGURE 4. Tree induced by TILDE with minimal coverage set to 10 examples.

While the trees become simpler, on the higher levels they are still very similar. We have taken a look at the most general leaf occurring in the tree and observed its evolution as the minimal coverage changes. The number of examples covered by this leaf does not change much, as can be seen in Table 7. The leaf is indicated by asterisks in Figures 3 and 4. The rule corresponding to this leaf is `c52 :- prop(A,B,C,D), A/=4, A/=3, red(q,I), 20.97<=I<=59.5, B<=4, red(d,J), 49.12<=J<=93.8, D=6`. It correctly classifies 347 of 357 examples of class 52.

Since the trees are quite similar, one could say that the simpler trees give an understandable approximate theory, while the complex trees refine this theory, making it much more detailed and less understandable but more accurate.

APPLYING ICL

The ICL system (De Raedt & Van Laer, 1995) uses exactly the same representation of training examples as TILDE but induces rule sets instead of trees. It is a first-order logic upgrade of the CN2 algorithm (Clark & Boswell, 1991).

For the ICL experiments, both frequencies and multiplicity counts (arguments of the `prop` predicate) were discretized. This was done mainly for efficiency reasons. Although multiplicity counts are discrete already, the discretization algorithm reduces the number of thresholds. The number of thresholds was (arbitrarily) chosen to be 10, both for the multiplicity counts and for frequencies.

On the propositional data, ICL achieves 79.1% accuracy with an average theory complexity of 22 rules. When learning from the whole data set, a theory of 22 rules is found, of which the most general rule is `c52 :- prop(S,D,T,Q), S=2, D=4, Q=6`. This rule covers 429 examples, 355 of them of class c52 and 72 of class c54: the examples covered are practically the same as those covered by FOIL's rule `c52 :- prop(S,D,T,Q), S>1, D>3, T>7, Q>5`.

Using frequencies only, ICL achieves 65.3% accuracy on unseen cases (with an average complexity of 57 rules). Learning on the whole data set yields a theory of 54 rules. Compared to FOIL, these rules are much more general (the most general rule covering 317 examples, of which 190 correct).

Using both frequencies and multiplicity counts, an average accuracy of 86.0% and average theory size of 64 rules were achieved. When learning a theory from all examples, 79 rules were found, the most general being `c3 :- prop(S,4,9,Q), Q<=4, red(s,B), B>=42.5, red(s,B2), 38.9<=B2<=44.2`, which covers 262 examples (242 of class `c3`).

Table 8 gives an idea of the generalization power of the different systems by showing the coverage of the most general rule (i.e., the rule covering the largest number of examples). The rules induced by FOIL are obviously more

TABLE 8 Comparison of the Generalization Achieved with Different ILP Systems: The Class Predicted by the Most General Rule or Leaf and the Number of Examples Covered by IT (Classified Correctly/Total)

Problem/System	FOIL	TILDE	ICL
red	c52 (43/43)	c3 (233/238)	c3 (190/317)
prop	c52 (355/429)	c52 (365/444)	c52 (355/429)
red + prop	c52 (227/227)	c52 (328/330)	c3 (242/262)

TABLE 9 Comparison of the Complexity of Rules Induced by ICL and FOIL

Problem/System	ICL rules	Literals	Literals/rule	FOIL Rules	Literals	Literals/rule
red	54	431	7.98	90	607	6.74
prop	22	83	3.77	17	52	3.06
red + prop	79	686	8.68	68	401	5.90

specific in terms of coverage. They also cover fewer negative examples from the training set than rules induced by TILDE and ICL. It is thus clear that FOIL overfits the training data more than TILDE and ICL, which is the cause for its lower performance.

Note that the complexity of a rule (e.g., the number of literals in its body) is a different, although related notion. A rule may cover many examples (i.e., be general) and contain many literals in its body (i.e., be complex) at the same time. Table 9 gives a comparison of rules induced by ICL and FOIL in terms of the total number of rules, total number of literals (in the bodies), and average number of body literals per rule. FOIL has a tendency to induce shorter rules, which (as we have seen above) cover fewer examples. Overall, the information gain and minimum description length (MDL) heuristics used by FOIL favor shorter clauses that are more accurate on the training set, while the Laplace heuristic used by ICL favors rules with higher total coverage of training examples.

DISCUSSION

For practical purposes, ^{13}C NMR spectra of diterpenes without assigned atom numbers have to be classified. This is a problem that is not directly transformable to propositional form and calls for either representation engineering or the use of ILP. We explored both approaches separately and in combination. Adding the engineered features to the natural relational representation improved the performance of all relational learning systems used.

Using the engineered features only, propositional approaches (in particular C4.5, nearest neighbor, and neural networks) achieve around 79% accuracy on unseen cases. This is roughly 20% less than the accuracies achieved when classifying ^{13}C NMR spectra of diterpenes with assigned atom numbers.

Using FOIL on the natural relational representation yields unsatisfactory results. Combining the relational representation with the engineered features greatly improves FOIL's performance. However, given the engineered features only, FOIL performs much worse than C4.5, so that the best performance of FOIL (combining the relational representation and the engineered features) is comparable to that of C4.5. The reason for FOIL's poor performance is that the rules induced are overly specific, as indicated by their coverage and confirmed by expert comments.

From the above, it is clear that a desirable property of relational learning systems is the following: given a propositional problem, a relational learning system should perform comparably to propositional systems. RIBL, TILDE, and ICL, which extend propositional learning paradigms to a relational framework, all have this property. Given the engineered features only, they achieve 79%, 78.5%, and 79.1% accuracy on unseen cases, respectively. Given the relational representation only, RIBL performs best, followed by TILDE and ICL. All three perform much better than FOIL. The situation is similar for the problem formulation combining the relational representation and the engineered features: RIBL is followed closely by TILDE, which performs at the same level; ICL follows, and FOIL is far behind. The best (91%) accuracy on unseen cases is 11% better than the best propositional result of 80% (backpropagation networks with 960 features).

The accuracies achieved by RIBL and TILDE are in the range of the accuracy with which experts classify diterpenes into skeleton types given ^{13}C NMR spectra only. That number can actually only be estimated, since it is expensive to have an expert carry out a statistically significant number of structure predictions without using other additional information that often becomes available from heterogeneous sources (such as the literature and ^{1}H NMR spectra). This basically means that ^{13}C NMR is not completely sufficient for classifying diterpenes and that great improvements of classification accuracy are not to be expected.

While RIBL does not yield understandable theories, FOIL, TILDE, and ICL are able to do so, at least in principle. In practice, understandability of the theories is diminished when only complex theories are accurate. Results with TILDE show, however, that it may be advantageous to find a slightly less accurate theory that is more understandable. The latter theory can be considered a cruder version of the more accurate theory: indeed, in the case of TILDE, large similarities are found between the complex, accurate tree and the simpler, less accurate tree. This offers an interesting compromise between understandability and correctness.

It would be interesting to see if similar effects take place when using newer versions of RIBL based on IB3 (Aha et al., 1991), which store only a fraction of all training cases as prototypes. These prototypes may be considered as explanations/rules, and a smaller number of prototypes would correspond to a shorter theory.

Another direction for further work is to carefully evaluate the rules produced by FOIL, TILDE, and ICL from the understandability point of view (domain expert). In this way, the trade-off between accuracy and understandability in this domain would be understood better. The explanatory power of these rules should be compared to that of prototypes offered as explanations by the newer versions of RIBL.

Finally, the use of additional information sources in the learning process, such as ^{1}H NMR spectra (where available), should be considered.

REFERENCES

Abraham, R. J., and P. Loftus. 1978. *Proton and carbon 13 NMR spectroscopy: An integrated approach.* London: Heyden.

Aha, D., D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.

Blockeel, H., and L. De Raedt. 1997*a*. Experiments with top-down induction of logical decision trees. Technical Report CW 247, Department of Computer Science, Katholieke Universiteit Leuven, Heverlee, Belgium.

Blockeel, H., and L. De Raedt. 1997*b*. Lookahead and discretization in ILP. In *Proc. 7th International Workshop on Inductive Logic Programming*, pp. 77–84. Berlin: Springer.

Clark, P., and R. Boswell. 1991. Rule induction with CN2: Some recent improvements. In *Proc. 5th European Working Session on Learning*, pp. 151–163. Berlin: Springer.

Cover, T. M., and P. E. Hart. 1968. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13:21–27.

De Raedt, L., and S. Džeroski. 1994. First order *jk*-clausal theories are PAC-learnable. *Artificial Intelligence* 70:375–392.

De Raedt, L., and V. Van Laer. 1995. Inductive constraint logic. In *Proc. 6th International Workshop on Algorithmic Learning Theory*, pp. 80–94. Berlin: Springer.

Džeroski, S., S. Schulze-Kremer, K. Heidtke, K. Siems, and D. Wettschereck. 1996. Diterpene structure elucidation from ^{13}C NMR spectra with machine learning. In *Proc. ECAI'96 Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pp. 21–26. Budapest: University of Economics.

Emde, W., and D. Wettschereck. 1996. Relational instance-based learning. In *Proc. 13th International Conference on Machine Learning*, pp. 122–130. San Mateo, California: Morgan Kaufmann.

Fayyad, U., and K. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc 13th International Joint Conference on Artificial Intelligence* pp. 1022–1027. San Mateo, Calif.: Morgan Kaufmann.

Gallant, S. I. 1993. *Neural network learning and expert systems*. Cambridge, Massachusetts: MIT Press.

Gray, N. A. B. 1982. *Progress in NMR spectroscopy*, vol. 15, pp. 201–248.

Lavrač, N., and S. Džeroski. 1994. *Inductive logic programming: Techniques and applications*. Chichester: Ellis Horwood.

Muggleton, S. 1995. Inverse entailment and PROGOL. *New Generation Computing* 13:245–286.

Muggleton, S., and C. Feng. 1990. Efficient induction of logic programs. In *Proc. First Conference on Algorthmic Learning Theory*, pp. 368–381. Tokyo: Ohmsha.

Natural. 1995. *Natural products on CD-ROM*. London: Chapman and Hall.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239–266.

Quinlan, J. R. 1993. *C4.5: Programs for machine learning*. San Mateo, Calif.: Morgan Kaufmann.

Schulze-Kremer, S. 1995. *Molecular bioinformatics – Algorithms and Applications*. Berlin: De Gruyter.

Van Laer, W., L. De Raedt, and S. Džeroski. 1997. On multi-class problems and discretization in inductive logic programming. In *Proc. 8th International Symposium on Methodologies for Intelligent Systems*. Berlin: Springer, pp. 277–286.

Wettschereck, D. 1994. A study of distance-based machine learning algorithms. Ph.D. thesis, Department of Computer Science, Oregon State University, Corvallis, Oregon.

Zell, A., et al. 1995. SNNS: Stuttgart neural network simulator. Report 6/95. University of Stuttgart, Institute for Parallel and Distributed High Performance Systems, Stuttgart, Germany.