Project N° 002307

ASPIC

Argumentation Service Platform with Integrated Components

Instrument: Specific Targeted Research Project
Thematic Priority: Information Society Technologies, objective 2.3.1.7.

Deliverable D3.4 - Implementation of and experiments with ABML and MLBA

Due date of deliverable: Month 30

Start date of project: 01/01/2004 Duration: 36 Months

Authors:
Martin Možina, Jure Žabkar, Ivan Bratko
University of Ljubljana (ULFRI), Slovenia
Pancho Tolchinsky, Ulises Cortes
Technical University of Catalonia (UPC), Spain

Version 1.0 - 03/07/2006

Partners:

LogicDIS S.A. (Co-ordinator)	LogicDIS	Greece
Cancer Research UK	CRUK	UK
Zeus Consulting S.A.	ZEUS	Greece
NAVUS	NAVUS	Germany
University of Ljubljana	ULFRI	Slovenia
Technical University of Catalonia	UPC	Spain
Institut de Recherche en Informatique de Toulouse	IRIT	France
University of Surrey	UNiS	UK
University of Liverpool	UNiL	UK
Utrecht University	UU	Netherlands
City University of New York	CUNY University	USA

Distribution List	
ASPIC Consortium	ASPIC Consortium, EC Project Officer

Written by:	ULFRI,UPC	
Reviewed by:	ULFRI	
Approved by:		

Document Change Log

Version #	**Issue Date**	**Sections Affected**	**Relevant Information**
0.9	16/01/2006	ULFRI	Preliminary version
1.0	03/07/2006	ULFRI	Final document

1 Interactions between Machine Learning and Argumentation: Introduction

This document describes two types of interaction between machine learning and argumentation. The first is called argument-based machine learning (ABML) - a novel approach to machine learning, where learning algorithms use results of argumentation (accepted arguments) along with data to induce better hypotheses. The other described type of interaction is machine learning based argumentation (MLBA). In this approach we deal with the opposite problem - machine learning algorithms induce hypotheses and, in argumentation, these hypotheses are employed as arguments.

In argument-based machine learning the arguments are used as a special form of background knowledge. The difference between arguments and classical background knowledge is that arguments relate to specific examples. Hence, to use these arguments - either given by expert or inferred from an argumentation engine - the algorithms need to be adapted. In section 2 we explain the motivation for ABML, and show the algorithm. Experiments, given at the end of section 2, show that this approach significantly improves the quality of learned hypotheses.

In section 3 we talk about MLBA, which is more concerned with: how can machine learning aid argumentation? In argumentation arguments are usually constructed from experts' knowledge or from single examples [4, 12] - also as case-based reasoning. In our approach, machine learning based argumentation, we show how can induced theories be used as arguments and how can MLBA be used to find specific theories that are usable as arguments. This can be especially useful in argumentation applications dealing with domains with large amount of data (business applications, medicine, etc.). In section 3 we give an extended motivation to MLBA, explain algorithms used and, at the end of section, describe a medical scenario that illustrates the use of MLBA. Another application of MLBA is presented in section 4: CBR and Argument Schemes for Collaborative Decision Making. This work is similar to [4, 12], only that past cases are used to evaluate the arguments' strengths submitted in a new dialog, rather than using them for construction of arguments. This approach is applied to the transplant scenario.

At this point we should also mention the simplest interaction between argumentation and machine learning, which is not described in this document. Machine learning could simply be used to induce knowledge bases for argumentation engines. For this task, one can use any of the standard machine learning techniques.

The focus of this deliverable is to present ABML and MLBA algorithms that we developed; these are ABCN2 and CN2BA, respectively. In addition to algorithms we give an extensive description of experiments using these two algorithms and an alternative approach to MLBA using CBR. Deliverable D2.1 (chapter 5)[11] contains a more basic explanation of ABML and MLBA paradigms, while in deliverable D2.4[52], a logical formalism of these two approaches is given.

2 Argument Based Machine Learning (ABML)

2.1 Motivation and Problem Statement

A fundamental problem of machine learning is dealing with large spaces of possible hypotheses. Usually, this problem is addressed either by biasing learning towards simpler hypotheses, i.e. applying Occam's razor, or by using experts' domain knowledge for constraining search. Domingos [21] argues that the former approach is valid only if we honestly believe the target phenomenon is simple, and therefore prefers the use of domain knowledge. However, the problem of the latter approach is the difficulty that experts face when they try to articulate their background knowledge. Here, we present a novel approach for constraining search, that is based on arguments about chosen learning examples rather that on the whole domain.

Usually the problem of learning from examples is stated as:

- Given examples
- Find a theory that is consistent with the examples

We will now extend this problem statement to that of ABML. In ABML, some of the learning examples are augmented by arguments, or reasons, or (partial) justifications according to principles of argumentation [45]. Defeasible argumentation is a branch of artificial intelligence that analyzes processes of reasoning where arguments for and against a certain claim are produced and evaluated. A typical example of a such reasoning is a law dispute at court, where plaintiff and defendant give arguments for their opposing claims, and at the end of the process the party with better arguments wins the case.

In our approach to ABML, arguments are used to explain (or argue about) why a certain learning example is in the class as given (arguments for = positive arguments) or why it should not be (arguments against = negative arguments). Examples that are accompanied with arguments are called *argumented examples*. Argumented examples are a means for the domain expert to introduce his (partial) domain knowledge to the learning system prior to learning. The practical advantage of providing prior knowledge in the form of arguments about *individual* examples, in comparison with providing more general domain knowledge, is in that it is easier to explain a single example than giving general theories. As arguments are reasons for a specific example, the expert does not have to ensure that the arguments are true for the whole domain. It suffices that they are true in the context of the given argumented example.

For instance, consider the problem of predicting whether a credit application is a good one or a risky one. If we ask an expert to explain why was a certain application classified as a risky application (ask for positive argument), he or she could explain that this was due to unemployed status of the applicant. This is, naturally, not valid for the whole domain, as there are many other ways of securing

the credit, say offering residential property as a collateral. But, as the applicant did not secure his credit in any other way, the expert states unemployment as a deciding factor, while not mentioning other attributes.

With arguments, the learning problem statement changes to:

- Given examples + supporting arguments for some of the examples
- Find a theory that explains the examples using given arguments

The main motivation for using arguments in learning lies in two expected advantages:

1. Reasons (arguments) impose constraints over the space of possible hypotheses, thus reducing search complexity
2. An induced hypothesis should make more sense to an expert as it has to be consistent with given arguments.

Regarding advantage 1, by using arguments, the computational complexity associated with search in the hypothesis space can be reduced considerably, and enable faster and more efficient induction of hypotheses. Regarding advantage 2, there are many possible hypotheses that, from the perspective of a machine learning method, explain the given examples sufficiently well. But some of those hypotheses can be incomprehensible to experts. Using arguments should lead to hypotheses that explain given examples in similar terms to those used by the expert, and correspond to the actual justifications.

2.2 Implementation

We have implemented argument-based CN2 (ABCN2; see [42]). ABCN2 is an extension of the well-known CN2 rule induction algorithm of Clark and Niblett [17]. We will start this section with a description of argumented examples that are acceptable for ABCN2, and continue with a detailed description of differences between the CN2 algorithm and its argument based counterpart ABCN2. Later on, we will show problems of the original CN2 algorithm when used with arguments, and, at the end of this section, introduce three improvements to the original algorithm that solve these problems.

2.2.1 Argumented examples

In ABCN2, arguments that the domain expert may attach to selected examples have the form of the conjunction of attribute-value pairs. In this way the expert (or experts) may state those features that he or she believes are important in the context of this particular example.

A learning example E in the usual form accepted by CN2 is a pair (A, C), where A is an attribute-value vector, and C is a class value. In addition to such examples, ABCN2 also accepts argumented examples. An argumented example AE is a triple of the form:

$$AE = (A, C, Arguments)$$

As usual, A is an attribute-value vector and C is a class value. $Arguments$ is a set of arguments $Arg_1, \ldots, Arg_n$, where an argument Arg_i has one of the following forms:

$$C \text{ because } Reasons$$

or

$$C \text{ despite } Reasons$$

The former specifies a *positive* argument (speaks for the given class value), while the latter specifies a *negative* argument (speaks against the class value). $Reasons$ is a conjunction of reasons $r_1, \ldots, r_n$,

$$Reasons = r_1 \wedge r_2 \wedge \ldots \wedge r_n$$

where each of the reasons r_i can be in one of five possible forms. These forms are (explanations given to forms assume that r_i is a part of a positive argument; for negative arguments, the explanations are exactly the opposite):

- $X = x_i$ means that value x_i of attribute X is the reason why example is in the class as given. This is the only allowed form for discrete attributes.
- $X > x_i (X >= x_i)$ means that, as the value of attribute X of example is higher (or equal) than x_i, this is the reason for class value.
- $X < x_i (X <= x_i)$ the opposite to $X > x_i (X >= x_i)$.
- $X > (X >=)$ "X is high", similar to $X > x_i (X >= x_i)$, just that we do not know the threshold value and it has to be found by ABCN2. Such an argument says that the value of X of example is high enough to be in the class as given.
- $X < (X <=)$; "X is low", the opposite of $X > (X >=)$.

The set $Arguments$ may not contain rejected arguments (rejected arguments are defined in [18]), all defeated arguments must be removed prior to learning.

To illustrate the idea of argumented examples, consider a simple learning problem: learning about credit approval. Each example is a customer's credit application together with the manager's decision about credit approval. Each customer has a name and three attributes: PaysRegularly (with possible values "yes" and "no"), Rich (possible values "yes" and "no") and HairColor ("black", "blond", ...). The class is CreditApproved (with possible values "yes" and "no"). Let there be three learning examples shown in Table 1.

The expert's argument for approving credit to Mrs. Brown can be: Mrs. Brown received credit because she is rich. Similarly, Miss White received credit because she pays regularly. A negative argument can be: Mrs. Brown received credit despite she does not pay regularly. The Brown example would in our syntax be written as:

Name	PaysRegularly	Rich	HairColor	CreditApproved
Mrs. Brown	no	yes	blond	yes
Mr. Grey	no	no	grey	no
Miss White	yes	no	blond	yes

Table 1: Learning examples for credit approval.

$$((PaysRegularly = no, Rich = yes, HairColor = blond),$$
$$CreditApproved = yes, \{CreditApproved = yes \text{ because}$$
$$Rich = yes, CreditApproved = yes \text{ despite } PaysRegularly = no\}).$$

Arguments given to examples additionally constrain rules *covering* this example. Remember that in CN2, rules have the form:

$$\text{IF } Complex \text{ THEN } Class$$

where $Complex$ is the conjunction of simple conditions, called *selectors*. For the purpose of this paper, a selector simply specifies the value of an attribute, for example $HairColor = blond$ or a threshold on an attribute value, for example $Salary > 5000$. A rule for our credit approval domain can be:

$$\begin{array}{lll} \text{IF} & PaysRegularly & = no \text{ AND } HairColor = blond \text{ THEN} \\ & CreditApproved & = yes \end{array}$$

The condition part of the rule is satisfied by the attribute values of Mrs. Brown example, so we say that this rule *covers* this example.

A rule R is *consistent* with an argument: C because $Reasons$ (or C despite $Reasons$), if for all reasons r_i of $Reasons$ is true that:

1. If the reason r_i is in one of forms: "$X = x_i$" or "$X > x_i$" or "$X < x_i$", then exactly the same selector needs to be present in the complex of the rule R.

2. If the reason r_i has the form "$X >$" (or "$X <$"), then the complex of the rule R needs to contain a selector $X > x_i$ (or $X < x_i$). The threshold value x_i does not matter for consistency.

With the use of arguments in examples, the definition of a rule *covering* an example needs to be refined. In the standard definition, a rule covers an example if the condition part of the rule is true for this example. In argument based rule learning, this definition is modified to:

A rule R *AB-covers* an argumented example E if:

1. All conditions in R are true for E (same as in CN2),

2. R is consistent with at least one positive argument of E, and

3. R is not consistent with any of negative arguments of E.

As an illustration of the differences between AB-covering and the usual definition of covering, consider again the Brown example with the argument that she received credit because she is rich and despite she does not pay regularly. Now consider four rules:

Rule 1: IF $HairColor = blond$ THEN $CreditApproved = yes$.

Rule 2: IF $PaysRegularly = no$ AND $HairColor = blond$ THEN $CreditApproved = yes$

Rule 3: IF $PaysRegularly = no$ AND $Rich = yes$ THEN $CreditApproved = yes$

Rule 4: IF $Rich = yes$ THEN $CreditApproved = yes$.

All four rules cover the Brown example and have 100% accuracy on the above data set. However, Rule 1 does not AB-cover the example, because it is not consistent with the positive argument. For the same reason, rule 2 does not AB-cover the Brown example, but this rule fails also because it is consistent with the negative argument ($PaysRegularly = no$). Rule 3 fails also fails due to negative argument, although it is consistent with the positive argument. The last example AB-covers the Brown example. The following rule, Rule 5, also AB-covers the example:

Rule 5: IF $HairColor = blond$ AND $Rich = yes$ THEN $CreditApproved = yes$.

2.2.2 ABCN2 - Algorithm

The CN2 algorithm [17, 16] consists of a covering algorithm and a search procedure that finds individual rules by performing beam search. The covering algorithm induces a list of rules that cover all the examples in the learning set. Roughly, the covering algorithm starts by finding a rule, then it removes from the set of learning examples those examples that are covered by this rule, and adds the rule to the set of rules. This process is repeated until all the examples are removed.

There are two versions of CN2: one induces ordered list of rules, and the other unordered list of rules. Our algorithm in this paper is based on the second version of CN2. In this case, the covering algorithm consists of two procedures, CN2unordered and CN2ForOneClass. The first procedure iteratively calls CN2ForOneClass for all the classes in the domain, while the second induces rules only for the class given. When removing covered examples, only examples of this class are removed [16]. Essentially, CN2ForOneClass is a covering algorithm that covers the examples of the given class.

ABCN2: Covering algorithm. We augmented CN2 for learning unordered rules with the ability to learn from argumented examples. We call the resulting algorithm ABCN2.

The first requirement for ABML is that an induced hypothesis explains argumented examples using given arguments. In rule learning, this means that for each argumented example, there needs to be at least one rule in the set of induced rules that AB-covers this example. This is achieved simply by replacing covering in original CN2 with AB-covering.

Replacing the "covers" relation in CN2 with "AB-covers" in ABCN2 ensures that both argumented and non-argumented examples are AB-covered. However, in addition to simpy AB-covering all the examples, we would prefer also explaining as many as possible non-argumented examples by arguments given for the argumented examples. Therefore, we propose a change in the covering algorithm, where CN2ForOneClass is changed into ABCN2ForOneClass (see Algorithm 1). The procedure starts by creating an empty list of rules, and makes a separate set AES of argumented examples only. Then it looks for unfinished arguments - arguments that have some of the reasons unspecified ($X >$ and $X <$) and finds the best splits for these reasons. Arguments in examples AES are then evaluated [1], and examples in AES are sorted according to the "best" given argument. Then, the procedure induces a rule, using ABFind_Best_rule, to cover the first argumented example. ABFind_Best_rule is a modified beam search procedure that accepts examples, an argumented example and a target class, where the resulting rule is guaranteed to AB-cover the given argumented example. This rule is added to the rule set, and the procedure removes from AES argumented examples AB-covered by this rule. The removal of all positive examples is not necessary, as each of the argumented examples differently constrains the search and thus prevents ABCN2 from inducing the same rule again. When all argumented examples are covered, all positive examples AB-covered by rules are removed, and the remaining rules are learned using classical CN2ForOneClass.

ABCN2: Search Procedure. Algorithm 2 shows AB search procedure. The procedure takes a set of examples to learn from, an argumented example that needs to be AB-covered by the induced rule and a target class. In Algorithm 2 the underlined parts emphasize the differences between the original search procedure in CN2 and AB-search procedure:

Initial value of set STAR are positive arguments of argumented example. A rule induced from an argumented example must AB-cover this example, therefore it will have to contain the reasons of at least one of positive argument. The easiest way to ensure this is to start learning from them.

[1] Evaluation function in CN2 is Laplace's rule of succession. However, many alternatives were proposed to improve learning. In this document we assume that Laplace is always used unless is specified differently.

Algorithm 1 Covering algorithm of ABCN2 algorithm that learns rules from examples ES for given class T

Procedure ABCN2ForOneClass(Examples ES, Class T)

Let RULE_LIST be an empty list.
Let AES be the set of examples that have arguments; $AES \subseteq ES$
Find splits for arguments where threshold not specified (type $X >$, etc.
Evaluate arguments (as they were rules) of examples in AES and **sort** examples in AES according to quality of their best argument.
while AES is not empty **do**
  Let AE1 be the first example in AES.
  Let BEST_RULE be *ABFind_best_rule(ES,AE1,T)*
  Add BEST_RULE to RULELIST.
  Remove from AES examples AB-covered by BEST_RULE.
end while
for all RULE in RULE_LIST **do**
  Remove from ES examples AB-covered by RULE.
end for
Add rules obtained with *CN2ForOneClass(ES,T)* to RULE_LIST

Specialize with selectors that cover argumented example. This ensures the coverage of the seed example by the induced rule.

Remove all complexes that are consistent with any of negative arguments. Again, rules must AB-cover argumented example, therefore can not be consistent with any of negative arguments.

Statistical significance. During search for a rule, many hypotheses are taken into account, therefore it makes sense to set pre-pruning α of the likelihood ratio statistic [17] higher than this is usually done. Moreover, to ensure that specialization is significant, the statistical difference of the class distribution of a new rule needs to be compared to the class distribution corresponding to the underlying argument rather than to the distribution in the whole data set.

Let ABSTAR be best N complexes from NEWSTAR ... Rules that contain only conditions that are present in positive arguments are certainly consistent with domain knowledge. Thence, these rules can be deemed perspective, although at the moment they are not among first N best rules according to quality.

It should be noted that argument based ML is not limited to CN2. Various standard ML techniques can be extended in a similar way to their argument-based variants. This document shows only one of possible implementations.

Algorithm 2 Algorithm that finds best rule that AB-covers example E from a set of argumented examples. The "quality" of a complex is evaluated by user-defined evaluation function.

Procedure ABFind_Best_Rule(Examples ES, Example E, Class T)

Let the set STAR contain positive arguments of E (written as complexes).
Evaluate complexes in STAR (using quality function).
Let $BEST\_CPX$ be the best complex from STAR.
Let $SELECTORS$ be the set of all possible selectors that are TRUE for E
Let $ARG\_REASONS$ be the set of all reasons in positive arguments of E (union of reasons).
while STAR is not empty **do**
  {Specialize all complexes in STAR as follows}
  Let $NEWSTAR$ be the set
    $\{x \wedge y \,\|x \in STAR, y \in SELECTORS\}$
  Remove from $NEWSTAR$ all complexes that are consistent with any of negative arguments of E.
  for every complex C_i in NEWSTAR **do**
    if C_i is statistically significant(ES,T) **and** quality(C_i) $>$ quality(BEST_CPX) **then** {Statistical significance function is inherited from CN2}
      replace the current value of $BEST\_CPX$ by C_i
    end if
  end for
  Let STAR be best N complexes from NEWSTAR; N is a user-defined size of STAR (usually N=5).
  Let ABNEWSTAR be such subset of NEWSTAR,
    that complexes in ABNEWSTAR contain only conditions from $ARG\_REASONS$.
  Let ABSTAR be best N complexes from ABNEWSTAR.
  Let STAR be STAR merged with ABSTAR.
end while
return rule: "**IF** $BEST\_CPX$ **THEN** T.´´

2.2.3 Problematic parts of the original CN2 algorithm

The algorithm shown in previous section is an extension of CN2 to work with argumented examples. However, there are three technical details of CN2 that render learning from argumented examples less effective as it could be. These are: examples removing strategy (after a rule is learned), evaluation function, and classification from rules. In the next three sections we outline these problems and propose their solutions.

2.2.4 Improved removing strategy

After CN2 learns a rule, it removes examples covered by this rule and recursively continues learning on remaining examples. By removing examples the approach assumes that the best possible rule for given examples was found - there exist no rule that would explain removed examples better.

This assumption might be true for classical CN2, however for ABCN2, where we first learn from argumented examples (learning is constrained by arguments of argumented example), this assumption is likely to be incorrect. A rule, learned from an argumented example, can be seen as the best possible rule covering this example, which does not assure that this rule is also the best rule for other examples covered. Therefore, removing examples after learning from argumented examples might prevent classical CN2 from learning some good rules.

The easiest solution would be to skip covering after learning from argumented examples. Note that such approach would seriously degrade the effect of arguments, since we hope that arguments will also guide ABCN2 to learn some rules that are better than rules induced by plain CN2. These rules, obviously, do not need to be learned again. Hence, we propose an alternative removing strategy, called *probabilistic removing strategy*.

In probabilistic covering strategy, the quality of rule must correspond to its classification accuracy, i.e. it is the probability of positive class if the rule triggers. Each example is tagged with the quality of the best rule (from the set of induced rules) covering this example. We will denote this probability as $example.prob$. At the beginning, when no rules have been learned yet, all examples have prior probability. When a new $rule$ is learned (let $rule.quality$ be quality of $rule$), the removing procedure updates all probabilities of covered examples as $example.prob = maximum(example.prob, rule.quality)$. We say, for covered examples of $rule$, where $rule.quality$ is higher than $example.prob$, that $rule$ *improves* the probability of this example, or shorter: $rule$ improves example.

There is still one problem that remains to be solved. Procedure $ABFind\_Best\_Rule$, as implemented at the moment, is not affected by such removing strategy and would learn the same rule all over again. Thus, the following changes are necessary:

Selection of best complex ($BEST\_CPX$) $BEST\_CPX$ can be updated only if current complex improves at least one example. This must be added to the condition in the algorithm.

Selection of best rules from STAR In the original CN2 algorithm best N rules are selected according to $rule.quality$. This heuristics fails in our case, as we wish to find a rule that has high quality and, at the same time, improves as many examples as possible. A better measure for selecting rules would be $rule.quality * P\_improve$, where $P\_improve$ is probability that final (specialized) rule will improve at least half of covered examples. With this

measure, the algorithm will select rules with high quality and with high probability of improving half of the examples.

The problem, though, is to estimate $P\_improve$. Naturally, computing it accurately is impossible, as we do not know which rules can still be specialized. We need to make a rough estimate, and we do it as follows. Let $bestrule$ be the best possible rule that can (theoretically) be induced from $rule$, and let $aveg\_prob$ be average of $example.prob$ of all covered examples by this rule. Then, if $aveg\_prob$ is between $rule.quality$ and $bestrule.quality$, $P\_improve$ is computed as: $P\_improve = \frac{bestrule.quality - aveg\_prob}{bestrule.quality - rule.quality}$. As you see, the idea behind this formula is the assumption that all qualities between current rule and the best rule are equally probable. If $aveg\_prob$ is higher than $bestrule.quality$, then $P\_improve$ is 0.0, and if $aveg\_prob$ is lower than $rule.quality$, then $P\_improve$ is 1.0.

2.2.5 Improved evaluation function

Evaluation function is used to estimate the quality of rule. Quality of rule is a user-defined measure to estimate the goodness of a rule. Generally, this measure should reflect the accuracy of the rule when classifying new examples. In original CN2-unordered implementation[16], Laplace's rule of succession is used to estimate the probability of correct classification of new (not learning) cases by the rule. Laplace's rule of succession states that if there are p successes from total n trials, the probability of success of $(n+1)$-st trial is $\frac{p+1}{n+2}$.

The problem with this formula (and also other evaluation functions) in optimization algorithms (such as rule learning) is that these algorithms choose the best hypothesis from many, what is causing optimistical evaluation estimates. This problem, also known as multiple-comparison problem in induction algorithms [30], is even worse in the case of ABCN2; rules, learned from argumented examples, were selected from less hypotheses than were rules from standard CN2 algorithm, and thus the quality of a rule learned from an argumented example is relatively under-estimated when compared to a rule learned from standard CN2.

In[39] we developed a method that accounts for multiple comparisons and believe that it should significantly improve the estimation of quality of rules learned from arguments. Technical details of the method are in Appendix A.

2.2.6 Improved classification from rules

Learned rules can also be used to predict class of a new example. In the case, where only one rule triggers for this example, classification is simply the class of the rule. In cases, where several rules trigger, we need to resolve clashes between opposing rules, and in standard CN2 these conflicts are resolved by summing the distributions of covered examples of all rules to find the most probable class. The main problem of such classification occurs when one rule has much higher coverage than

the other; imagine next two rules:

$$\text{IF } some\_condition \text{ THEN } C1[100, 50]$$

and

$$\text{IF } other\_condition \text{ THEN } C2[0, 30]$$

Sum of distributions would be $[100, 80]$ and new example would be classified into $C1$ with probability $\frac{100}{100+80}$. However, the second rule has, according to Laplace's rule, probability 0.97, while the first has only 0.66. It seems that the second rule is better and that example should be classified in the second class. There is another problem with CN2's classification. The method should also take into the account, whether these rules cover the same examples or whether their intersection is an empty set. For instance, if 30 cases covered by the second rule would be a perfect subset of 50 cases(class $C2$) covered by the first rule, then the first rule would, in argumentation terms, be defeated, because the second rule actually presents a specialization of the first rule. The importance of intersections of rules for classification was also shown by Lindgren[36].

The problem with summing distributions is even more fatal in the case of ABCN2. The patterns (rules) learned from argumented examples usually can not be learned by classical CN2, sometimes because there are not enough learning examples. Hence, rules from arguments might relatively small coverage and become relatively unimportant in classification when used together with rules learned with classical CN2. On the other hand, we showed in the previous section how to compute quality of a rule that accounts for number of tried hypotheses. Therefore, to make rules from argumented examples competitive, we need to develop a classification technique based on quality of rules.

Combining rules by quality is a major problem. The simplest approach would be to assume independence of rules given class and combine them using naive Bayesian classifier. Lindgren[36] showed that independence assumption is invalid in most of the cases, and proposed several solutions to this problem. One of them was to compute intersections between rules. We shall also pursue his idea, however his solution can not be applied here. He merely looks at the distribution of examples where all rules trigger, and computes probability of class from that distribution using Laplace formula. In ABCN2 case, the probability of class in intersection must depend on the qualities of intersecting rules.

In the remaining part of this section we will show a solution for a two-class problem. It can be also used with domain that have more classes, but we need to split one multi-class learning problem in several two-class problems. Our solution for classification makes two additional assumptions:

- The quality of a rule corresponds to its estimated classification accuracy - probability of positive class.
- If two rules predict the same class, then the probability of positive class is the maximum of qualities of both rules ($P(C|R1 \wedge R2) = max(R1.quality, R2.quality)$).

Here we assume that if two rules apply to the same example, then either (1) the intersection of these two rules is empty (or contains only a small number of examples), where it is practically impossible to make a reasonable estimation of quality, or (2) the probability of C in intersection is not higher than the maximum of qualities of both rules. In the latter case, if the probability in the intersection would be higher and there would be enough examples to make a correct estimation, the learning algorithm should have found the combined rule.

The quality of two rules together predicting the same class is therefore the same as quality of the better rule. In our classification model we will take only one rule for each class and compute the probabilities of classes in the intersection of these two rules. Two problems occur here:

1. How can we compute the probability of class in the intersection of two rules, that takes qualities of rules into consideration?
2. Which rule for each class should we select?

Regarding (1), we have the following two rules:

R1: IF $conditions(C1)$ THEN $Class = A$

R2: IF $conditions(C2)$ THEN $Class = B$

Let the distribution of $R1$ be $(a1, b1)$ and the distribution of the second rule $(a2, b2)$. We will denote the quality of rule $R1$ as $R1.quality$ and $R2.quality$ as the quality of the second rule. We will assume that qualities of rules represent their true classification accuracy on yet unseen examples.

We wish to compute probability of class A given conditions $C1$ and $C2$ $P(A|C1 \wedge C2)$. We will use m-estimate[15] of probability to estimate this probability (m will always be set to 2 ad proposed by Cestnik [15]):

$$P(A|C1 \wedge C2) = \frac{ea_i + m * prior}{ea_i + eb_i + m} \quad (1)$$

Values ea_i and eb_i are expected number of examples of class A and class B in intersection of rules $R1$ and $R2$. Let (a_i, b_i) be distribution of intersection of both rules. As $R1.quality$ usually does not equal $\frac{a_i}{a_i+b_i}$ (similar is true for $R2.quality$), (ea_i, eb_i) does not equal to (a_i, b_i), but it is computed in the following way:

$$ea_i = a_i + (R1.quality - \frac{a_i}{a_i + b_i})a_i - (R2.quality - \frac{b_i}{a_i + b_i})b_i \quad (2)$$

and $eb_i = a_i + b_i - ea_i$. Prior probability $prior$ is computed from $R1.quality$ and $R2.quality$ assuming independence of rules:

$$prior = \frac{R1.quality * (1 - R2.quality)}{R1.quality * (1 - R2.quality) + (1 - R1.quality) * R2.quality} \quad (3)$$

$A \backslash B$	R1	R2	R3
R1	0.9	0.3	0.7
R2	0.2	0.8	0.4
R3	0.4	0.6	0.9

Table 2: Probabilities for class A of intersections of imaginary three rules for A and B.

Regarding the second problem we need to decide which rule for each class should we select. This problem is illustrated in Table 7. It shows three fictitious rules for A, three rules for B, and the probabilities of their intersections for A. If we select the first rule for class A and the first rule for class B, then the probability for A would be 0.9. However, this probability would be 0.3, if the second rule for B was selected, but again 0.8, if second rules for both classes would be selected. This problem is analagous to a zero-sum game with perfect information of two opponents with simultaneous moves. An optimal solution always exists for both parties using mixed strategy, which was proven in Minimax theorem [51]. Mixed strategy means that a rule should be selected with a certain probability, where the optimal solution is the distribution of probabilities according to you should be selecting. For instance, in above problem the best strategy for A would be to call the first rule in 50% and the second rule in 50% cases, while the third rule should not be called at all. The average probability of a such strategy is 0.55, and this probability we use as the classification probability for class A. The percentage of a rule (e.g. 50%) can also be seen as the *importance* of a rule for the classification of a certain example. The same can be done for the second class. The best strategy would be to call the first rule in 41% and the second in 59% cases achieving average probability of 0.45.

2.2.7 Can improvements damage the performance of CN2?

We described several improvements to original algorithm that improve learning from argumented examples. However, these changes also affect classical CN2 learning. The question that might be raised here is: do these changes, in anyway, hinder learning of classical CN2? In this section we show that these changes do not hinder learning, they can actually improve learning without arguments.

We will compare four methods on 17 UCI [43] data sets (we selected only domains with two-class problems). The methods are:

- CN2: classical CN2-unordered [16].
- CN2-C: classical CN2-unordered with improved classification.
- CN2-CC: classical CN2-unordered with improved classification and improved covering.

	CN2	CN2-C	CN2-CC	CN2-CCE	CN2-IC	NBC	C4.5
adult	0.83	0.82	0.82	0.85	0.83	0.79	0.85
australian	0.85	0.85	0.86	0.86	0.84	0.86	0.83
breast (lju)	0.65	0.69	0.71	0.72	0.71	0.73	0.75
breast (wsc)	0.96	0.97	0.95	0.95	0.96	0.97	0.92
credit	0.83	0.83	0.85	0.86	0.82	0.86	0.86
german	0.75	0.75	0.74	0.73	0.75	0.72	0.72
hepatitis	0.82	0.83	0.82	0.81	0.80	0.85	0.79
ionosphere	0.93	0.92	0.88	0.93	0.91	0.89	0.92
monks-1	1.00	1.00	1.00	1.00	1.00	0.75	1.00
monks-2	0.75	0.67	0.74	0.66	0.66	0.63	0.66
monks-3	0.96	0.99	0.99	0.99	0.99	0.96	0.99
pima	0.73	0.73	0.75	0.76	0.72	0.75	0.75
SAHeart	0.66	0.65	0.68	0.71	0.69	0.71	0.70
shuttle	0.98	0.97	0.99	0.97	0.97	0.93	0.98
tic-tac-toe	0.99	0.98	1.00	0.99	0.98	0.70	0.85
titanic	0.78	0.79	0.79	0.79	0.79	0.77	0.79
voting	0.88	0.94	0.95	0.96	0.95	0.91	0.97
comparison		6:6	4:9	5:7			

Table 3: Classification accuracies of classical CN2 and 3 improved versions of CN2. Last three columns are standardly used methods in machine learning: CN2 with internal cross validation (CN2-IC), naive Bayesian Classifier (NBC), and C4.5. The results were obtained with 10-fold cross-validation.

- CN2-CCE: classical CN2-unordered with improved classification, covering, and evaluation.

We also give results of three other standard methods, only shown for the orientation. The first method, CN2-IC, is the classical CN2-unordered learning with m-estimate set as the evaluation function. Parameters m and α of statistical significance function are optimized with internal cross-validation. The second method is naive Bayesian Classifier (NBC), and the last method is C4.5. All methods are implemented within the Orange-toolkit [20].

We are especially interested how each improvement influences the classification performance of the method. In this sense, we will compare methods iteratively: CN2 vs. CN2-C, CN2-C vs. CN2-CC, and CN2-CC vs. CN2-CCE. Table 3 shows classification accuracies of all above-mentioned methods. The last row shows comparison of a method with its predecessor. For instance, score 4:9 in the column of CN2-CC means that CN2-CC was better than CN2-C in 9 domain and worse in 4. Note that in all comparisons, the method with improvement is either tied or has a higher number of wins than methods without this improvement. The same turns out to be true if methods are compared with AUC, therefore we conclude, that none of the improvements hinder learning in anyway. Actually, these results imply that improvement may also improve the induction of rules with classical CN2

	CN2	CN2-C	CN2-CC	CN2-CCE	CN2-IC	NBC	C4.5
adult	0.79	0.84	0.87	0.89	0.86	0.83	0.79
australian	0.89	0.93	0.92	0.93	0.93	0.93	0.84
breast (lju)	0.56	0.64	0.65	0.71	0.63	0.68	0.59
breast (wsc)	0.98	0.99	0.99	0.99	0.99	0.98	0.97
credit	0.88	0.91	0.91	0.92	0.90	0.92	0.87
german	0.70	0.73	0.75	0.76	0.73	0.79	0.66
hepatitis	0.71	0.82	0.87	0.84	0.78	0.91	0.69
ionosphere	0.94	0.96	0.95	0.95	0.95	0.94	0.91
monks-1	1.00	1.00	1.00	1.00	1.00	0.71	1.00
monks-2	0.70	0.70	0.67	0.74	0.68	0.56	0.51
monks-3	0.99	0.99	0.99	0.99	0.99	0.98	0.99
pima	0.75	0.78	0.82	0.81	0.77	0.82	0.77
SAHeart	0.64	0.68	0.72	0.74	0.69	0.76	0.67
shuttle	0.99	1.00	1.00	1.00	1.00	1.00	1.00
tic-tac-toe	1.00	1.00	1.00	1.00	1.00	0.75	0.90
titanic	0.76	0.77	0.77	0.77	0.77	0.72	0.75
voting	0.94	0.98	0.98	0.98	0.97	0.98	0.98
comparison		0:13	3:6	2:7			

Table 4: AUCs of classical CN2 and 3 improved versions of CN2. Last three columns are standardly used methods in machine learning: CN2 with internal cross validation (CN2-IC), naive Bayesian Classifier (NBC), and C4.5. The results were obtained with 10-fold cross-validation.

algorithm.

2.3 Experiments with ABCN2

In this section we present several experiments that we conducted with ABCN2. The first two, ZOO , is small and is here only for illustrative purposes. The other two are larger, the first is application of ABML to law, and the second is an application to a medical problem. All the first three experiments were done before we have developed improved evaluation function, covering strategy and classification. [At least one experiment using all three improvement will be present in the final deliverable].

Ideally, experts would give arguments to all learning examples. However, usually domains contain a large number of examples, and experts are willing to give arguments to only some of examples. Now we describe how these critical examples can be identified.

2.3.1 Identifying critical examples.

Giving arguments to all examples in the data set is obviously not feasible. Our method automatically finds "problematic" examples, that is examples where arguments would be likely to have a significant effect on learning. So the "commentator" of examples (expert who provides arguments) is asked to concentrate on these "problematic" cases. This idea is realised by the following iterative procedure:

1. *Induce a hypothesis without arguments*
2. *Find a critical example that needs to be argumented.* This step involves a search for the most "problematic" example (e.g. outlier) in the learning set. For this task we propose a $n \times k$ cross-validation (e.g. $n = 4, k = 5$), so that each example is tested n times. The example that was misclassified in most of the cases is chosen as the example that needs to be argumented. If there are several such examples, then the algorithm picks a random one.
3. *If problematic example was not found (in step 2), then stop the iteration.*
4. *An expert gives arguments to the selected example.* Two things can happen here: in the preferred case the expert finds argumenting this example easy; in the undesired case, the expert finds this to be hard. The second case may be due to different reasons (deciding attributes are not available, the example is an outlier, or argumenting may be hard in the chosen representation of arguments). Each of these cases can be mended in different ways,which still needs to be explored. In the extreme, this example can simply be discarded from the learning set.
5. *Induce rules on the learning set using ABCN2 and new argumented example.*
6. *Return to step 2.*

2.3.2 ZOO Data Set

ZOO data set contains descriptions of 101 animals (instaces) with 17 attributes: *hair, feathers, eggs, milk, predator, toothed, domestic, backbone, fins, legs, tail, catsize, airborne, aquatic, breathes, venomous, and type*, which is the class attribute. *Type* has seven possible values: *mammal, bird, reptile, fish, amphibian, insect, and other*. The main advantage of this data set is that, using encyclopedia, we can give good arguments to examples. We expect that, using arguments, comprehensibility and classification accuracy will increase.

The set was split to learning set (70%) and test set (30%), m-estimate was used with $m = 2$, and statistical significance threshold was 1.0. The values were obtained with internal-cross validation (only on learn set) to maximize the classification accuracy. Without arguments (classic CN2), the following rules were induced:

IF milk=yes THEN type=Mammal
IF fins=yes AND breathes=no THEN type=Fish
IF feathers=yes THEN type=Bird
IF legs=6 AND breathes=yes THEN type=Insect
IF legs=4 AND hair=no AND predator=yes THEN type=Amphibian
IF legs=0 AND venomous=yes AND catsize=no AND toothed=yes THEN type=Reptile
IF toothed=no AND legs=4 AND hair=no THEN type=Reptile

Considering learning data alone, the induced rules fit perfectly. However, classification accuracy on the test set is only slightly over 90%. Now, according to our method of involving argumented examples (section 2.3.1), we have to find the most problematic example using the learning set only. The internal (on learning set only) cross-validation procedure found that the most frequently misclassified example was a reptile, the tortoise. Notice that the rules covering reptiles split reptiles in two subgroups; in the first group are legless, poisonous, small, and toothed reptiles (snakes) and in the other are toothless, with four legs, and hairless reptiles (turtles). A problem with these two rules is that there also exist four-legged reptiles with teeth (crocodile, tuatara, etc. - tuatara was misclassified in the test set). A good argument for tortoise to be a reptile is that it has the backbone and it lays eggs (tortoise is reptile because backbone=yes AND eggs=yes). Using that argument for tortoise in ABCN2, the two rules for reptiles were replaced by next two rules:

IF **backbone=yes** AND **eggs=yes** AND aquatic=no AND feathers=no THEN type=Reptile
IF eggs=no AND breathes=no THEN type=Reptile

Naturally, our argument did not perfectly define reptiles, so it has been extended by ABCN2 with attributes *aquatic* and *feathers*. The first attribute separates reptiles from fishes and the second from birds, as both, fishes and birds, have backbone and lay eggs.

The next problematic example found was a sea snake (a reptile). A sea snake is an air-breathing snake that lives underwater. According to encyclopedia, a sea snake should be correctly classified with the above rule, however, after inspecting the data more thoroughly, we noticed that sea snake is, in the data, characterized as a non-breathing reptile and that it does have eggs. It is obviously a mistake and (probably) occurred, because sea snakes can live without breathing for a very long time (up to 2h) and also because, they actually do not lay eggs, but keep them in themselves until they grow inside the mother. This is also the reason for the induction of the second rule.

The next problematic example found in the learn set was a newt. The argument was provided that the newt is an amphibian because it has backbone, lays eggs and is related to water. This resulted in the rule:

IF **backbone=yes** AND **aquatic=yes** AND **eggs=yes** AND legs=4 THEN type=Amphibian

---

After adding these arguments to the two examples, ABCN2 induced a perfect set of rules for the ZOO data set. There were no misclassified examples in the test set.

This example clearly illustrates the effectiveness of our method for choosing critical examples (both identified examples were really critical) and shows how arguments can be used to learn concepts that are otherwise very hard to learn. The example also nicely illustrates that it is much easier to give arguments only to an individual example (e.g why tortoise is a reptile), than it would be to articulate general rules that correctly classify the animals.

2.3.3 A business application - learning about credit status

Learning about credit status is a sub-problem of a larger B2B scenario[50] used in ASPIC as the main large-scale demonstration application for all argument-based methods. In this experiment we show how ABCN2 can be also used to improve existing knowledge bases in argumentation based expert systems.

The main problem of the global scenario is to determine whether client should be granted credit for a purchase. This is determined by the system through a set of if-then business rules. A part of them is used to determine whether the applicant credit status is “good”, “bad”, “average”, or “unavailable”. In the initial knowledge base (given by experts), the following three rules were used for classification:

Credit History = Good If all past debts were paid before or on the date due.

Credit History = Average If equal or less number of past debts were paid in less than a week late, to the ones paid on time.

Credit History = Bad If more debts paid after the deadline than the ones paid on time.

Along to classification rules, ZEUS also provided a data set of 5000 companies described with 18 attributes (three of them are actually relevant). We began the experiment with the induction of rules from 2500 examples (learning set) without considering given classification rules. The system learned a set of 40 rules. As there was a lot of data available, the method was able to correctly classify 95% of examples in test set - the remaining 2500 examples - which is a quite good result with respect to classification accuracy.

In the next step we ran the algorithm for finding problematic example. It was a company that had "bad" credit status as class value. We looked up the values of this example and noticed that third rule from experts apply. This company did historically pay more often late than on time (attributes $past\_debts\_after\_date$, $past\_debts\_on\_time$, $past\_debts\_before\_time$). The argument for bad is thus:

$$\begin{aligned} Credit\_History \quad = \quad & Bad \text{ because } past\_debts\_after\_date > \\ & past\_debts\_on\_time + past\_debts\_before\_time \end{aligned} \quad (4)$$

The problem with given argument that it is not consistent with the format of if-then rules learned by CN2. The condition part of a rule is a conjunction of several attribute-value pairs, whereas our condition contains comparison of three attribute values and also a sum of two of them. We solved this problem by constructing another boolean attribute that has value 1 when the above condition is true and 0 when it is not. We named it "latePayer" and the argument thus changed to

$$Credit\_History = Bad \text{ because } latePayer = 1$$

Note that this example implies to another useful applicability of ABML approach - inclusion of knowledge in the domain, which would otherwise be impossible to induce given hypotheses space.

With the additional argument and the same 2500 learning examples, ABCN2 learned only 27 rules. Due to argument and new attribute the number of rules for classes "bad" and "average" dropped significantly. At the same time classification accuracy measured on test data was slightly improved to 97%.

In the next pass we repeated the search for problematic example, this time it was from class "Credit_history"="good". The argument from initial knowledge base was therefore:

$$Credit\_History = Good \text{ because } past\_debts\_after\_date < 1$$

We relearned rules from learn set and two given arguments and ABCN2 induced 6 rules. Classification accuracy of these rules on test set was 99.9%.

This experiment showed how existing argumentation engine with imperfect knowledge base can be used to deduce arguments needed by ABCN2. It also showed that the accuracy of the final model is better than knowledge from argumentation engine itself and is better from accuracy of model learned from data only.

2.3.4 Argument Based Machine Learning Applied to Law

This experiment is published in [40] and [41]. This section presents these two papers.

The data set. The data set used in these experiments was first used in [7]. The data concerns a fictional welfare benefit. The benefit is payable if six conditions are satisfied. These conditions were chosen to represent different kinds of condition that are found in the legal domain, so that we can see whether the different form of conditions affects their discoverability.

The notional benefit was a fictional welfare benefit paid to pensioners to defray expenses for visiting a spouse in hospital. The conditions were:

1. The person should be of pensionable age (60 for a woman, 65 for a man);
2. The person should have paid contributions in four out of the last five relevant contribution years;
3. The person should be a spouse of the patient;
4. The person should not be absent from the UK;
5. The person should have capital resources not amounting to more than 3,000;
6. If the relative is an in-patient the hospital should be within a certain distance: if an out-patient, beyond that distance.

These conditions represent a range of typical condition types: 3 and 4 are Boolean necessary conditions, one which should be true and one false; 5 is a threshold on a continuous variable representing a necessary condition, and 2 relates five Boolean variables, only four of which need be true. 1 and 6 relate the relevance of one variable to the value of another: in 1 sex is relevant only for ages between 60 and 65, and in 6 the effect of the distance variable depends on the Boolean saying whether the patient is an in-patient or an out-patient. We can see these six conditions either as explicit conditions or as ways of making operational concepts such as elderly, sufficient contribution record, close relative, presence in the UK, insufficient capital resources, and attributable expenses respectively.

The data was generated using a program written in Common LISP. For this experiment a data set of 2400 records was used: 1200 satisfying all of the conditions, and equal numbers of the remainder being designed to fail. For records designed to

fail one of the conditions, satisfaction or otherwise of the remaining conditions was decided randomly for each condition separately. There are thus twelve attributes relevant to the decision: age, sex, the five contribution conditions (called cont5, cont4, cont3, cont2 and cont1), spouse, absent, capital, distance and inpatient. In addition to these attributes each record contains fifty two irrelevant attributes, half of which are continuous and half Boolean. An ideal set of rules would be:

1. IF age $<$ 60 THEN qualified = no;
2. IF age $<$ 65 and sex = m THEN qualified = no;
3. IF any two of cont5, cont4, cont3, cont2 and cont1 = n
   THEN qualified = no;
4. IF spouse=no THEN qualified=no;
5. IF absent=yes THEN qualified=no;
6. IF capital$>$3000 THEN qualified=no;
7. IF inpatient=yes AND distance$>$ 750 THEN qualified=no;
8. IF inpatient=no AND distance$\leq$ 750 THEN qualified=no;

Probably we should expect (3) to be expressed as ten separate rules containing each pair of the contribution factors, which would be a total of sixteen rules to describe the problem fully.

The experiment. The original data (2400 records) was randomly split into a learning set containing 70% of the cases, and a test set (the remaining 30%) used to assess the accuracy of the generated rules on new cases. As already mentioned we used the original algorithm without any improvements. As evaluation we used m-estimate (m=2), statistical significance was set to $\alpha = 0.001$ and minimal coverage of a rule was set to 2 (rule needs to cover at least two examples).

The first set of rules was generated from examples without any arguments. So the resulting rules were as if generated with CN2. These rules were:

1. IF capital $>$ 2900 THEN qualified = no;
2. IF age $\leq$ 59 THEN qualified = no;
3. IF absent $=$ yes THEN qualified = no;
4. IF spouse $=$ no THEN qualified = no;
5. IF cont4 $=$no AND cont2 $=$ no THEN qualified=no;
6. IF age $>$ 89 THEN qualified $=$ no;

Of these (1) - (5) are correct (or very close). Nine contributions rules and the two distance rules are missing and rule (6) is wrong. There is thus considerable scope for improvement. None the less a high degree of accuracy is achieved by these six rules: 99% for both the learning and the test sets. Accuracy of classification is not, however, of prime interest: it is the interpretability of the rules discovered that is of primary interest.

In the case of our legal data, the most problematic example failed on the contributions condition. The argument given for this example was that cont5, cont4 and cont1 are all false. When this argumented example, together with all the other (non-argumented) examples was given to ABCN2, two additional contribution rules were induced:

IF $cont5 = no$ AND $cont4 = no$ AND $cont1 = no$
THEN $qualified = no$;

IF $cont2 = no$ AND $cont3 = no$ THEN $qualified = no$;

and the accuracy increased slightly.

In a third learning iteration, an argument was added to an additional misclassified case in which distance was too great for an inpatient. This time the erroneous rule (6) disappeared and was replaced by a rule relating to inpatiency and distance (although with an approximate threshold) and another contributions rule:

IF $inpatient = yes$ AND $distance > 735$ THEN $qualified = no$;

IF $cont1 = no$ AND $cont5 = no$ THEN $qualified = no$;

Iterations four, five and six added three more arguments based on failure of the contribution conditions, which resulted in a different contributions rule. In iteration seven, the argument was given that distance was too small for outpatient. This produced the rule

IF $inpatient = no$ AND $distance \leq 735$ THEN $qualified = no$;

and rearranged the contribution rules somewhat.

At this point there were no misclassified examples in the internal validation tests any more, and so no further argumentation with our iterative procedure was possible. The final accuracy on the test set was 99.8%. This means that one out of 720 test cases was misclassified, and all the rest were classified correctly. The final set of rules were:

1. IF $capital > 2900$ THEN $qualified = no$;
2. IF $age \leq 59$ THEN $qualified = no$;
3. IF $absent = yes$ THEN $qualified = no$;
4. IF $spouse = no$ THEN $qualified = no$;

5. IF $cont4 = no$ AND $cont2 = no$ THEN $qualified = no$;

6. IF $inpatient = yes$ AND $distance > 735.0$
   THEN $qualified = no$;

7. IF $inpatient = no$ AND $distance \leq 735$ THEN $qualified = no$;

8. IF $cont3 = no$ AND $cont2 = no$ THEN $qualified = no$;

9. IF $cont5 = no$ AND $cont3 = no$ AND $cont1 = no$
   THEN $qualified = no$;

10. IF $cont4 = no$ AND $cont3 = no$ AND $cont1 = no$
    THEN $qualified = no$;

11. IF $cont5 = no$ AND $cont4 = no$ AND $cont1 = no$
    THEN $qualified = no$;

(1) - (5) are all good rules and remain from the first pass. (6) and (7) have the right format, but the threshold is slightly inaccurate. The remaining four rules approximate the ten ideal contribution rules. The total number of argumented examples after the seven iterations was seven. The one misclassified example was (omitting the irrelevant attribute values): (($age = 84$, $sex = male$, $cont1 = no$, $cont2 = yes$, $cont3 = no$, $cont4 = yes$, $cont5 = yes$, $spouse = yes$, $absent = no$, $capital = 130$, $distance = 1320$, $inpatient = no$), $qualified = no$). This example is misclassified because the particular combination of contribution conditions is not covered by the approximate contribution rules induced.

Experiments with noisy data. Learning from artificial data sets is usually considered easier than learning from real world data sets. One reason for this is that artificial data are typically noise-free whereas real world data typically contain noise. So to cope with real world data, a learning method has to be able to deal with noise. In this section we investigate the question how robust our learning algorithm ABCN2 is with respect to noise in data. To this end we artificially introduced random noise of varying severity in our learning data as described below. Intuitively we expected that background knowledge in the form of arguments should improve the method's resistance to noise in comparison with CN2. This expectation was confirmed by the experiments described in this section.

The experimental procedure for this experiment was as follows. First, we split the data set to learning set (70%) and test set (30%). Then we added random noise into the learning set, induced rules with both CN2 and ABCN2, and measured the accuracy of both sets of rules on the (noise-free) test set. To study how the severity of noise affects the success of learning, we repeated the experiment for various rates of noise. The chosen rates were: 0%, 2%, 5%, 10%, 20%, and 40%. A noise rate p means that with probability p the class value of each learning example is replaced by a random value drawn from $\{$yes, no$\}$ with distribution (0.5, 0.5). Each

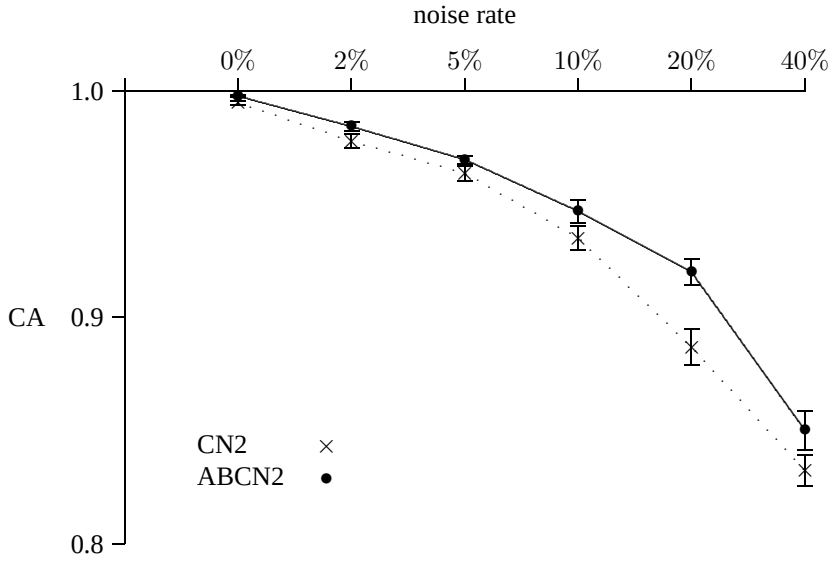


Figure 1: CN2 and ABCN2 compared on learning data sets with different proportions of noise in class variable.

time CN2 and ABCN2 were run with correspondingly "noised" examples plus the same noise-free argumented examples as in section 3.2. This whole procedure was repeated 10 times in order to obtain confidence intervals of the estimated average classification accuracy for both CN2 and ABCN2 for each noise rate.

Figure 1 shows the results of both methods CN2 and ABCN2. Both methods were run with the default settings of their parameters. These standard values are: m=2 in m-estimate, α=0.001(likelihood ratio statistics threshold), and minimal coverage of a rule is set to 2 (that is, a rule to be acceptable has to cover at least two examples).

This figure shows that, as expected, ABCN2 clearly outperforms CN2 when rate of noise in data increases. The difference in average accuracies between ABCN2 and CN2 jumped from 0.3% at 0% noise to 3.3% at 20% and to 1.7% at 40% noise. ABCN2 outperformed CN2 on 0%, 20%, and 40% noise with high statistical significance (t-test,p$<$0.001; see Table 5). ABCN2 was also the better method for other noise rates but with lower significances.

The ability to handle large amount of noise is very important in the kind of routine legal decision making we are addressing. Errors rates are very high: Groothuis and Svenson [27] report experiments which suggest that 20% may be a low estimate of incorrectly decided cases. The problem is internationally widespread. The US National Bureau of Economic Research reports of a particular benefit[1]:

"The multistage process for determining eligibility for Social Se-

[1]From Web Page: *http://www.nber.org/aginghealth/winter04/w10219.html*

noise	CN2	ABCN2	Sig.
0%	0.9947±0.0010	0.9976±0.0005	<0.001
2%	0.9778±0.0030	0.9842±0.0020	0.002
5%	0.9636±0.0034	0.9696±0.0017	0.005
10%	0.9351±0.0053	0.9469±0.0051	0.037
20%	0.8869±0.0079	0.9200±0.0056	<0.001
40%	0.8326±0.0068	0.8503±0.0088	0.001

Table 5: Results of CN2 and ABCN2 on noisy data. The first column shows noise rates, second and third contain average accuracy and standard error of accuracy estimate for CN2 and ABCN2 respectively, and the last column gives significances of differences between averages computed with pairwise t-test.

> curity Disability Insurance (DI) benefits has come under scrutiny for the length of time the process can take - 1153 days to move through the entire appeals process, according to a recent Social Security Administration (SSA) analysis - and for inconsistencies that suggest a potentially high rate of errors. One inconsistency is the high reversal rate during the appeals process - for example, administrative law judges, who represent the second level of appeal, award benefits in 59% of cases. Another inconsistency is the variation in the award rates across states - from a high of 65% in New Hampshire to a low of 31% in Texas in 2000 - and over time - from a high of 52% in 1998 to a low of 29% in 1982."

Again an official UK Publication produced by the Committee of Public Accounts[2]:

> "Finds that the complexity of the benefits system remains a major problem and is a key factor affecting performance. Skills of decision-makers need to be enhanced through better training and wider experience. Too few decisions are right first time, with a error rate of 50% for Disability Living Allowance. There are also regional differences in decision making practices that may lead to payments to people who are not eligible for benefits."

This makes it clear that robustness in the face of large amounts of noise is essential if a learning techniques is to be applied to data from this domain

Comparison with other learning methods. In this section we will compare this experiment with previous learning experiments with the same data set. We will compare three learning techniques here:

[2]Getting it right: Improving Decision-Making and Appeals in Social Security Benefits. Committee of Public Accounts. London: TSO, 2004 (House of Commons papers, session 2003/04; HC406)

- Argument Based Machine Learning (ABML),
- Neural Nets (NN), in [7]
- Defeasible Logic (DL), in [31].

DL uses an algorithm called HeRo to derive a minimal theory in Defeasible Logic. Bench-Capon [7] describes experiments with NN using a number of net topologies, the most successful of which used a single hidden layer of 12 nodes in a fully connected network. There has also been an experiment in learning Association Rules (AR) [9]. We will not include AR in our comparison because of significant differences in their experimental setup. Irrelevant attributes were omitted from the learning data in that experiment making the learning easier, and the format of the rules produced was somewhat different. DL does not handle continuous variables and so such variables are mapped into Booleans using user-supplied thresholds.

The rules discovered in DL were:

IF $distance > 700$ AND $inpatient = no$ THEN $qualified = no$;

IF $spouse = no$ THEN $qualified = no$;

IF $absent = yes$ THEN $qualified = no$;

IF $age < 60$ THEN $qualified = no$;

IF $capital > 300$ THEN $qualified = no$;

These rules are a subset of the rules discovered by ABML: they fail to recognise the contributions conditions, and also omit the rule governing inpatients.

The neural network did not produce rules, but the contribution of the various inputs was estimated using the technique described in [7]. The most important contributing attributes, in descending order of importance, are shown in Table 6.

This experiment did recognise the importance of the contribution conditions, but the inpatient condition and distance to hospital were not seen as of any significance. Also two irrelevant attributes, registered and score, were seen as more important than sex. Note also that the degree of influence drops quite steeply before sex is found.

Both DL[31] and NN[7] report a very high level of accuracy (although details of accuracy evaluation are not given). This was also obtained by the traditional machine learning method of CN2. That it is possible to train a system to classify with an excellent degree of success is also the experience of other neural nets work such as Split-Up [54] and the experiments of Borges et al. [10]. Such success, however, should not blind us to the problems: none of the techniques produced rules which faithfully reflected the conditions for the benefit. This should, perhaps, be unsurprising given that all machine learning methods respect some sort of minimum description length principle. However, this suggests that their use to classify actual cases would be of dubious acceptability, since they would make some systematic errors, and so discriminate against particular classes of individuals.

Table 6: Important inputs in [7].

Factor	Influence	Degree
Spouse	positive	0.995
Absent	negative	0.984
Cont5	positive	0.920
Capital	negative	0.882
Cont1	positive	0.875
Cont4	positive	0.819
Cont2	positive	0.809
Cont3	positive	0.797
Age	positive	0.779
Registered	positive	0.776
Score	positive	0.720
Sex	negative	0.646

2.3.5 Argument Based Machine Learning Applied to Medical Domain

Infections in the aging population present an increasing problem in the developed countries. The population of people over 65 years of age is rapidly growing and so are the costs of treating these patients. Our goal is to build a model which would help the physician, at the first examination of the patient, decide how severe the infection is and consequently, whether the patient should be taken into the hospital or could be treated as an outpatient. More, we would like to have an understandable model, not a black box, to see which parameters play a decisive role.

2.4 Data

The data for our study was gathered at the Clinic for Infectious Diseases in Ljubljana, from June 1st, 2004 to June 1st, 2005. The physicians included only the patients over 65 years of age with CRP value over 60 mg/l, which indicated a bacterial etiology of the infection. The patients were observed 30 days from the first examination or until death caused by the infection. The data includes 40 clinical and laboratorial parameters (attributes) acquired at the first examination for each of 298 patients (examples). The infections are distinguished with respect to the site where bacteria is found or on the clinical basis (respiratory, urinary tract, soft tissues, other). The continuous attributes were categorized by the physician. The distribution of the class values is the following:

- 34 examples (11,4%) for 'death = yes'
- 263 examples (88,6%) for 'death = no'

2.5 Arguments

The argumentation was done by the physician who was treating the patients and could by her expert knowledge state several positive and negative arguments to 32 examples, where all argumented examples were from class *death = yes*, namely she gave the reasons she believed caused death for each selected patient. A sample argumented example is shown in Table 7.

Attribute	**Value**	
GENDER	Z	**Positive arguments**
AGE_YEARS	92	
AGE	C	DEATH=YES because RESPIRATORY_RATE_D="$>= 16$"
NURSING_HOME_RESIDENT	NO	DEATH=YES because SATURATION_D="$<= 90$"
COMMORBIDITY	0	DEATH=YES because BLOOD_PRESSURE_D="$<= 100$"
DIABETES	NO	DEATH=YES because TEMPERATURE_D="> 37.9"
HEART	NO	DEATH=YES because LEUKOCYTES_D="$>= 12$"
KIDNEY	NO	DEATH=YES because CREATININE_D="$>= 100$"
LIVER	NO	DEATH=YES because BLOOD_UREA_D="$>= 13$"
LUNG	NO	DEATH=YES because NA_D="> 147"
IMMUNITY	NO	DEATH=YES because AGE_YEARS is high
CENTRAL_NERVE_SYSTEM	NO	DEATH=YES because WEAKNESS=YES
MOBILITY	YES	DEATH=YES because CONSCIOUSNESS=DISSORIENTED
CONTINENCE	YES	
BEDSORE	NO	**Negative arguments**
CATHETER	NO	
IMPLANT	NO	DEATH=YES despite MOBILITY=YES
VOMITING	NO	DEATH=YES despite CONTINENCE=YES
DIABLOODPRESSUREHEA	NO	DEATH=YES despite TROMBOCYTES_D="$>= 100$"
WEAKNESS	YES	DEATH=YES despite HEART_RATE_D="< 100"
CONSCIOUSNESS	DISSORIENTED	DEATH=YES despite RODS_D="< 10"
TROMBOCYTES_D	$>= 100$	DEATH=YES despite CRP_D="< 150"
TEMPERATURE_D	> 37.9	DEATH=YES despite COMMORBIDITY=0
RESPIRATORY_RATE_D	$>= 16$	
SATURATION_D	$<= 90$	
HEART_RATE_D	< 100	
BLOOD_PRESSURE_D	$<= 100$	
LEUKOCYTES_D	$>= 12$	
RODS_D	< 10	
CRP_D	< 150	
CREATININE_D	$>= 100$	
BLOOD_UREA_D	$>= 13$	
GLU_D	< 15	
NA_D	> 147	
INFECTION_TYPE	RESPIRATORY	
DEATH (class value)	YES	

Table 7: A sample argumented example from the infections database.

One could, at this point, ask an interesting question about these arguments: whether they would, if used as rules, describe the domain sufficiently well. We built a simple classifier from the given arguments and tested it on the same data set; for each case, we counted the number of applicable arguments for class *death = yes* and compared this number to the number of arguments for class *death = no*. The accuracy of a such classifier is only slightly above 40%, therefore there is still a large space available for machine learning to improve. Since the default accuracy in this domain is 88.6% it indicates that the knowledge which is hidden in arguments is far from perfect. However, please note that this experiment is not used to validate the expert knowledge. To do that, at least the arguments to examples from the opposite class should be given as well. Our intention is merely to show that the knowledge given by the arguments is neither perfect nor complete though it can still help to improve learning.

2.6 Results

Learning and testing was performed by 10-fold cross validation which was carried out 10 times with different random splits of examples into folds. We compared the algorithms ABCN2 and CN2, where both methods used improvements shown in the previous section, so that their comparison directly represents the influence of the arguments added to the learning examples. Both algorithms are then compared to Naïve Bayes (NB), decision trees (C4.5) and logistic regression (LogR). Algorithms are compared with regard to classification accuracy, area under ROC (AUC) and Brier score. All the methods and tests were implemented within Orange toolkit [20]. The results are shown in Fig. 2- 4.

Observing classification accuracy, that is the percentage of correct classifications, we can see that CN2, ABCN2 and C4.5 achieve similar results while NB and LogR perform significantly worse (Fig. 2). Although classification accuracy is important it should be accompanied by other estimates especially because the majority classifier itself is quite accurate in this domain due to imbalance between the two classes. Therefore we also measure AUC and Brier score, which are applicable as all the methods also give the probability of predicted class. AUC measures how well the method ranks examples; it is the probability that for two randomly chosen examples with different classes, method will correctly decide classes of these examples (it is not allowed to classify both in the same class, as it knows that they are from different classes). This measure is often used to evaluate hypotheses in medical domains, where we wish to have methods that separate positive from negative examples as good as possible. Figure 3 shows that, according to AUC, ABCN2 significantly outperforms all other methods. The same effect also comes out in Brier scores (Fig. 4), which measures the average quadratic error of predicted probability. It is important to note that for imbalanced domains, as our domain, AUC and Brier score are more relevant measures of success than accuracy.

2.7 Discussion

ABCN2 achieved better results than CN2 according to all three measures by using arguments given by expert. The question is how the induced hypotheses from both measures differ and why ABCN2 is the better method. To examine the hypotheses, we induced a set of rules from the whole data set with ABCN2 and CN2. As the arguments were given only to examples with class value *death=yes*, the induced rules for *death=no* were the same for both methods. Both methods induced 14 rules for class *death=yes*, however there were two important differences between these two sets of rules. First, due to the restriction of hypotheses space with arguments, about half of the rules were different. While inspecting the rules that were the same in CN2's and ABCN2's set, we noticed that the quality estimates of these rules were different. For example, the rule:

IF trombocites<100 AND mobility=no THEN death=yes

was present in both rule sets. It covers 6 examples with class value *death=yes* and 1 with *death=no*, which means that the relative frequency of *death=yes* is 6/7 = 0.86. However, the evaluation function based on extreme value distributions [39] used in CN2 estimated the probability of this class (given that the conditions are true) as 0.47, which is much less than 0.86. This happens because there is a high probability that such a rule would be found by chance. On the other hand, when learning with ABCN2, the evaluation of the same rule is 0.67. In CN2, this rule was obtained by searching the whole space unguided by expert knowledge while in ABCN2 the rule was built from the argument *'death=yes BECAUSE trombocites<100'*. The search space in ABCN2 is smaller, which means that the probability of finding such a rule by chance is lower. So, the expected quality of the rule is higher.

In the above paragraph we have shown the importance of the first expected advantage of ABML: "Arguments impose constraints over the space of possible hypotheses, thus reducing search complexity". Regarding the second advantage, that induced rule should make more sense to an expert, we asked our expert (Jerneja Videčnik) to examine the rules and compare them. Unfortunately, she could not decide which rules are more understandable to her. We believe that this occurs due to the large number of arguments with only one reason given to each example, while our restriction is that the rule must be consistent with at least one positive argument. The rule must, therefore, contain only one of the given reasons and can neglect the others.

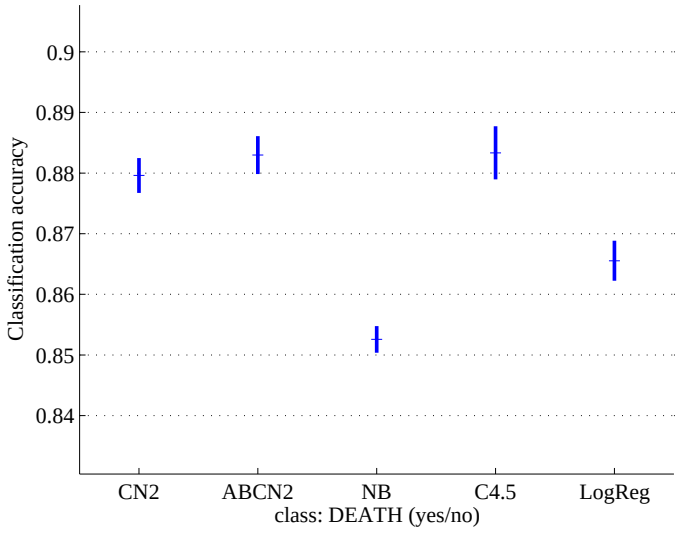


Figure 2: Mean values and standard errors of classification accuracy across tested methods.

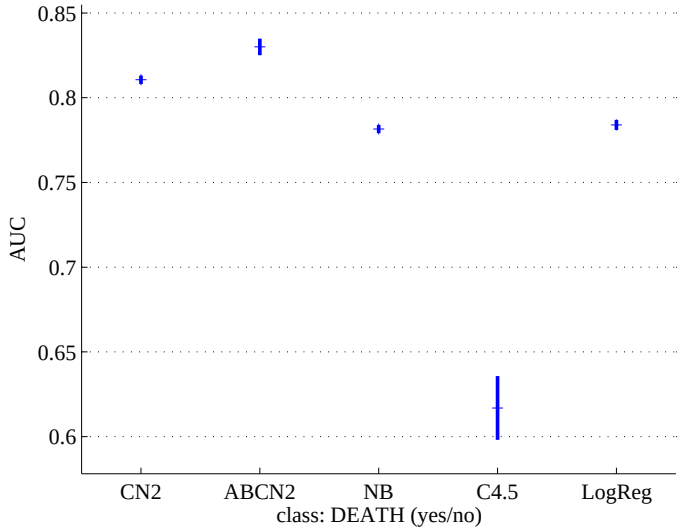


Figure 3: Mean values and standard errors of AUC across tested methods.

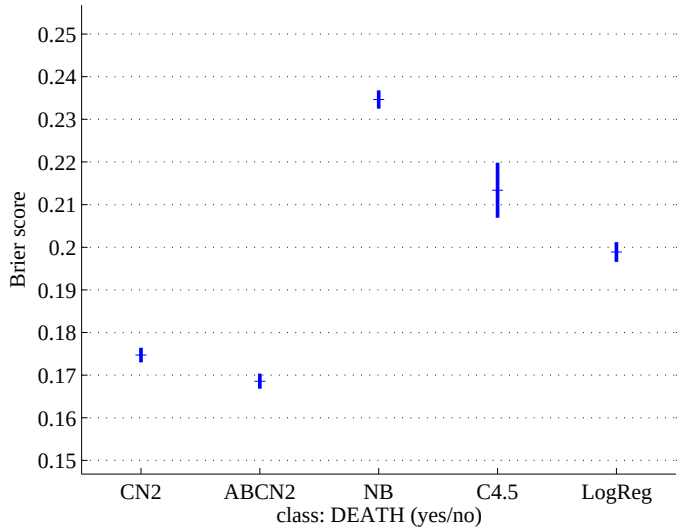


Figure 4: Mean values and standard errors of Brier score across tested methods.

3 Machine Learning Based Argumentation (MLBA)

3.1 Motivation

We can use induced theories as support (or reasons) for arguments. In classical machine learning the algorithms try to find hypotheses that score best on data ac-

cording to some user-specified criteria, e.g. classification accuracy. However, in MLBA we try to generate reasons for arguments that have a desired claim for a specific case.

Therefore, the induced hypotheses must, beside scoring high, also support the user-specified claim for the chosen example. We will illustrate this idea using our small data set for credit approval, presented in Table 8.

Name	PaysRegularly	Rich	HairColor	CreditApproved
Mrs. Brown	no	yes	blond	yes
Mr. Grey	no	no	grey	no
Miss White	yes	no	blond	yes

Table 8: Learning examples for credit approval.

Let us consider now a new applicant. He does not pay regularly, is poor, and has blond hair. First, the bank would ask MLBA to induce rules that disapprove giving him credit. MLBA would find the following rule:

IF $PaysRegularly = no$ AND $Rich = no$ THEN $CreditApproved = no$

On the other hand, the applicant could also use this system and ask for hypotheses proving his validity for credit. The system would return:

IF $HairColor = blond$ THEN $CreditApproved = yes$

As you can see, both hypotheses score perfectly on learning data, however support opposing arguments for the chosen example. Arguments rebut each other. In the remaining of this section we will show our implementation of MLBA and a specific scenario that will be used for experimenting with MLBA.

3.2 Implementation

We have implemented CN2BA - an extension of CN2 algorithm - that can induce rules biased to specific claim and given example. The control functions of CN2 and CN2BA are the same, whereas the search function is different, see Alg. 3. The difference between CN2's search function and CN2BA's search function (underlined part in Alg. 3) is: **Let** $SELECTORS$ **be the set of all possible selectors that are TRUE for E**. The function, thence, finds the best rule for class T that covers the seed example.

3.2.1 Undercutting

Rule learning algorithm can be also used to induce undercutters of arguments. Let A (R_A) be an argument (supporting rule) induced by the above algorithm. To find an undercutter B of A, the algorithm should seek for a specialized rule of R_A that predicts the opposing class as R_A does.

Algorithm 3 Algorithm that finds the best rule for the selected class and the selected example.

Procedure Find_Best_Rule(Examples ES, Example E, Class T)

Let the set STAR contain empty complex
Let $BEST\_CPX$ be nil.
Let $SELECTORS$ be the set of all possible selectors that are TRUE for E
while STAR is not empty **do**
  {Specialize all complexes in STAR as follows}
  Let $NEWSTAR$ be the set $\{x \wedge y \parallel x \in STAR, y \in SELECTORS\}$
  for every complex C_i in NEWSTAR **do**
    if C_i is statistically significant(ES,T) **and** quality(C_i) $>$ quality(BEST_CPX) **then**
      replace the current value of $BEST\_CPX$ by C_i
    end if
  end for
  Let STAR be best N complexes from NEWSTAR.
end while
return rule: "**IF** $BEST\_CPX$ **THEN** T.´´

3.3 Guardian Agent Scenario with MLBA

The main experiment of MLBA is guardian agent scenario. This scenario was already described and illustrated in D1.4 (machine learning section)[5]. Its description here is merely to complete the machine learning based argumentation section.

The purpose of this scenario is two-fold:

- Show how arguments can be induced from data at a request. An agent knowledge base rarely contains knowledge about all possible cases. In such cases MLBA can be used to obtain arguments from available data.
- Show an example how an agent can use and interpret new evidence about new drugs. This evidence is then used together with well-established facts about known drugs for the same disease.

As you can see, the first purpose is more general, and presents the usability of MLBA in any argumentation process. The second purpose is specific for medicine.

In this scenario we have three agents: A cardiac agent (CA), a patient information agent (PIA), and a guardian agent (GA). The CA is an "expert" agent, who can prescribe drugs. Guardian agent has remit to construct safety cases against proposals made by expert agents. It has limited medical knowledge, relating only to safety of medical procedures. However, it has extensive knowledge about how to reason effectively about safety. CA must check safety of all its recommendations with GA. If safety arguments of GA are stronger than arguments given by

CA for the specific drug, CA must prescribe an alternative. All three agents can use MLBA and internal argumentation engines for constructing arguments.

To understand the general concept we continue this section giving a specific scenario for a specific patient. We will use Prolog's notation for describing patients and rules, because the scenario is implemented in Cogent (CRUK):

1. Cardiac agent is dealing with a patient with suspected myocardial infarct (MI). Let us name him John.
2. CA wishes to prescribe a drug to prevent blood clotting.
3. CA has information that drugs which reduce platelet adhesion prevent blood clotting.
4. CA has information that aspirin and chlopridogrel reduce platelet adhesion.
5. CA proposes aspirin and chlopridogrel as candidate drugs for prescription.
6. CA has information that aspirin has low cost and high availability.
7. CA constructs one argument for chlopridogrel (prevents blood clotting) and three for aspirin (prevents blood clotting, low cost, high availability). These arguments were constructed from CA's knowledge base.

   A1 (chlopidogrel): $administer(chlopidogrel) -> reduces(\_platelet\_adhesion), reduces(\_platelet\_adhesion) -> prevents(\_blood\_clotting)$

   A2 (aspirin): $administer(aspirin) -> reduces(\_platelet\_adhesion), reduces(\_platelet\_adhesion) -> prevents(\_blood\_clotting).$

   A3 (aspirin): $administer(aspirin) -> cost(aspirin, low)$

   A4 (aspirin): $administer(aspirin) -> availability(aspirin, high)$
8. CA provisionally commits to prescribing aspirin (as there are more arguments in favor of this decision option) and requests confirmation from guardian agent.
9. Guardian agent attempts to construct safety arguments against aspirin.
10. GA has information that use of aspirin in an individual susceptible to gastritis represents a hazard.

    $susceptible\_to(Patient, gastritis)$ AND $administer(Patient, aspirin) -> hazard(Patient, gastric\_bleeding)$
11. GA attempts to determine if patient is susceptible to gastritis. GA asks patient information agent to confirm $john$'s susceptibility to gastritis:

$$susceptible\_to(john, gastritis)$$

12. Patient information agent first checks for explicit statement in John's record that he is susceptible to gastritis, and finds none. The PIA can also make use of rules (linked to encoded online repositories of drug information such as the British National Formulary(www.bnf.org)) of the form:

    $currently\_taking(Patient, Drug) AND side\_effect(Drug, Condition) -> susceptible(Patient, Condition)$

    to infer $susceptible(Patient, Condition)$. However, there is a new drug $aspican$ that $patient$ is using as a part of clinical trial, about which information such as side-effects is not as yet known. The PIA is under obligation to inform the GA that the query $susceptible\_to(john, gastritis)$ fails, together with the clause that it fails because of unknown information, i.e.,

    $$side\_effect(aspican, gastritis)$$

13. GA is obligated use MLBA to induce arguments for claim.

14. GA composes a query and sends it to MLBA. The target class is $side\_effect(aspican, gastritis)$ and John's data consisting of patient attributes and their values (gleaned from the PIA). The GA must have access to currently collected data for a number of patients who have been enrolled on the trial of the drug $aspican$. The MLBA algorithm can thus induce arguments for the claim $side\_effect(aspican, gastritis)$ where the grounds of the argument refer to the attribute values of John that are shared by the patients in the trial who suffered from the side-effect.

15. If any arguments that John is susceptible to gastritis are acceptable, GA uses them to constructs an argument against aspirin on safety grounds and returns it to CA. (if none are acceptable it confirms the CA's decision to use aspirin).

16. Assuming GA has passed back a safety argument, CA adds this argument to its database and re-runs its argumentation procedure. We assume that the safety argument has a greater strength than efficacy, cost and availability arguments A1, A3 and A4 for aspirin and so argument A2 for chlopridogrel is now the CA's acceptable argument.

17. CA must test the safety of chlopridogrel with GA.

3.4 Discussion and Related Work

In this section we showed how rule learning algorithms can be extended to induce arguments for selected examples. The general problem of machine learning (and rule learning) is overfitting hypotheses to given data, and the same problem is inherited by MLBA. Therefore, we believe that MLBA can be a very valuable tool to help experts automatically find arguments when they can not remember reasons for

their claims, and where they will revise arguments - whether they make sense or not. On the other hand, using MLBA automatically should be done with caution.

Rule learning was used here only as an example of a MLBA algorithm, and we believe that a similar extension could be applied to any machine learning algorithm. There were already some similar approaches to combining machine learning and argumentation:

- The HYPO family [4] of case-based reasoning (CBR) models offers a lazy learning approach to modelling legal reasoning.
- Stefanie Brueninghaus and Kevin D. Ashley[12] proposed an algorithm (IBP) that combines reasoning with abstract domain model and case-based reasoning techniques to predict the outcome of case-based legal arguments.
- Gomez and Chesñevar suggested in their report [26] several ways of combining machine learning methods and argumentation.
- CBR and argument schemes for collaborative decision making.[2]

[2]Described in the following section.

4 CBR and Argument Schemes for Collaborative Decision Making

In this section we propose an approach that combines Case-Based Reasoning (CBR) together with Argumentation. This approach involves *1)* the use of CBR [33] for evaluating the arguments submitted by agents in collaborative decision making dialogs, and *2)* the use of Argument Schemes and Critical Questions [53] to organize the CBR memory space. The former involves use of past cases to resolve conflicts among newly submitted arguments by assigning them a strength, and possibly submitting additional arguments deemed relevant in similar past deliberations. The latter enables use of agents' submitted arguments instantiating Argument Schemes and Critical Questions, to assess the similarity among cases. Hence, a case is simply defined as a placeholder for the available data related to an experience, and it is the submitted arguments associated with each experience that provide means for comparing cases. This use of CBR and argumentation is formulated with the *ProCLAIM* model, which features a Mediator Agent that directs proponent agents in their deliberation and subsequently evaluates their submitted arguments so as to conclude whether a proposed decision is valid. To motivate and substantiate the practical value of this approach, we illustrate its application in the human organ transplantation field.

In the following section we describe the *ProCLAIM* model. In §4.2 we introduce the transplant scenario, and in §4.3 we show how the CBRe makes use of arguments to compare cases and how cases can be used to resolve conflicts among arguments. And in §4.4 We finalize with a brief discussion on the related works, the current state of development of the CBRe and the plans for future research.

4.1 The *ProCLAIM* Model

Broadly construed, the *ProCLAIM* model consist of a mediator agent, MA, directing *proponent* agents in an argument based collaborative decision making dialog, in which the final decision must comply with certain domain dependent guidelines. The arguments submitted by the proponent agents may also persuade the MA to accept decisions that deviate from the guidelines. For example, the MA may be able to reason that the submitted arguments supporting an alternative decision have proven to be correct in previous similar deliberations.

ProCLAIM is of particular value in safety-critical domains (although the scope of domain may well be wider) where the consequences ensuing from a *wrong* decision may be catastrophic. Guidelines in such sensitive environments usually exist and are created in an attempt to minimize hazardous decisions. Nonetheless, there are circumstances in which a decision is appropriate despite violating established guidelines. Moreover, in such environments, arguments supported by empirical evidence are somewhat more persuasive.

ProCLAIM defines three main tasks for the MA: *1)* Inform the proponent agents as to what are their dialectical possible moves at each stage of the delib-

eration; *2)* Ensure that the submitted arguments are relevant (*e.g.*, comply with the guidelines), and *3)* Evaluate the submitted arguments in order to identify the *winning* arguments and thus determine whether a proposed decision is valid. This last task may require the assignment of strengths to the given arguments and possibly submission of additional arguments. In order to undertake these tasks, MA makes use of four knowledge resources (see fig. 5):

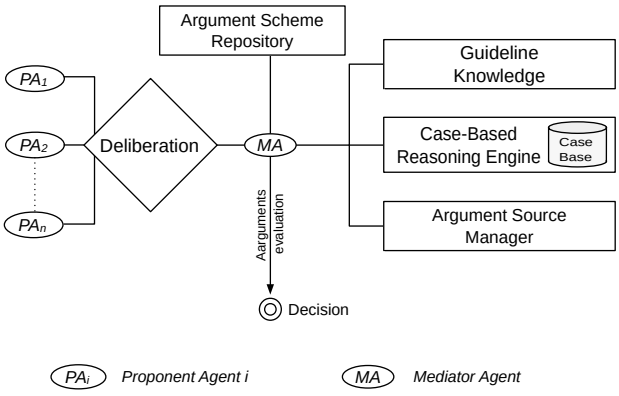


Figure 5: *ProCLAIM*'s Architecture

Argument Scheme Repository (ASR): In order to direct the proponent agents in their deliberation the MA makes use of a repository of argument schemes and their associated critical questions formalized in a way that defines a protocol based exchange of arguments, the schemes are instantiated by agents in order to construct arguments. Thus, for example, given a submitted argument A instantiating a scheme of ASR, the MA can reference the ASR in order to identify the schemes that, if effectively instantiated, constitute an attack on A. As we will see in §4.3, the ASR also structures the CBRe's memory space.

Guideline Knowledge (GK): This component enables the MA to check whether the arguments comply with the established knowledge, by checking what are the valid instantiations of the schemes in ASR (the ASR can thus be regarded as an abstraction of the GK).

Case-Based Reasoning Engine (CBRe): This component enables MA to assign strengths to the submitted arguments on the basis of their associated evidence gathered from past deliberations, as well as provide additional arguments deemed relevant in previous similar situations.

Argument Source Manager ASM: Depending on the source from whom, or where, the arguments are submitted, the strengths of these arguments may be readjusted by the MA. Thus, this component manages the knowledge related to

the agents' roles and/or reputations, and/or the types of certificates, or references, that may empower agents to undertake some exceptional decision.

The agents' argument construction is based on a first order logic programming language described in [38]. This work also defines the conflict based interaction between arguments. Given the constructed arguments and their interactions we apply Dung's seminal *calculus of opposition* [22] to determine the justified or *winning* arguments. However, determining the winning arguments may require the MA to assign strengths to the submitted arguments and possibly the submission of additional arguments. This is further discussed and illustrated in sections 4.2 and 4.3. The agents' dialog and in particular, the role of the MA in directing the deliberation by referencing the ASR is defined in [49]. Agents construct and submit arguments by instantiating the schemes and critical questions in the ASR. The MA's task is then to determine which are the winning arguments in order to conclude whether the proposed decision complies with the GK. This may involve referencing the CBRe to access similar past experiences and arguments given to support an undertaken decision not compliant with the GK, but which proved to have a successful outcome. This may also involve referencing past experiences in order to resolve mutually attacking arguments by assigning relative strengths to these arguments. These roles of the CBRe are further developed in §4.3. We now introduce the transplant scenario in order to illustrate the practical value of *ProCLAIM*, and in particular the value of Case-Based Reasoning in resolving conflicting arguments and the use of arguments for comparing cases.

4.2 The Transplant Scenario

Human organ transplantation constitutes the only effective therapy for many life-threatening diseases. However, while the increasing success of transplants has led to increase in demand, the lack of a concomitant increase in donor organ availability has led to a growing disparity between supply and demand [37]. In spite of this, an important percentage of human organs available for transplantation are discarded as being deemed non-viable for that purpose.

The human organ selection process illustrates the ubiquity of disagreement and conflict of opinion in the medical domain. What may be a sufficient reason for discarding an organ for some qualified professionals may not be for others. Different policies in different hospitals and regions exist, and a consensus among medical professionals is not always feasible. Hence, contradictory conclusions may be derived from the same set of facts. For example, suppose a donor with a smoking history of more than 20-30 packs a year and no history of *chronic obstructive pulmonary disease* (COPD). The medical guidelines indicate that a donor's smoking history is a sufficient reason for deeming a donor's lung as non-viable [44]. However, there are qualified physicians that reason that the donor's lung is viable given that there is no history of COPD [37]. Similarly, the guidelines suggest discarding the kidney of a donor whose cause of death was *streptococcus viridans endocarditis*

(*sve*)[44]. However, some reason that by administrating *penicillin* to the recipient the kidney can safely be transplanted [14].

The transplant scenario begins when a potential donor becomes available. The donor's organs deemed non-viable by the Transplant Coordinator (which we name the Donor Agent, DA) are discarded, whereas the organs deemed viable are offered via a third-party (Transplant Organization) in a queue of Transplant Units, (which we name Recipient Agents) that may be located in different hospitals. These Recipient Agents, RA_1,...,RA_n, to which the organ may eventually be offered may accept it, in which case they may attempt to implant it to a potential recipient they are responsible for. Or, if every RA_j fails to accept the organ, it is discarded, *i.e.* not extracted from the donor.

A DA's decision to not offer an organ which he believes to be non-viable prevents other RA_j's from having the opportunity to make use of that organ. The human organ selection process is described in more detail in [48] where an alternative selection process is proposed to be managed by *CARREL*, an agent-based organization designed to improve the overall transplant process. In this alternative process a DA_i that detects a potential donor offers all the potentially transplantable organs irrespective of whether he believes the organs to be viable or non-viable. *CARREL* then distributes the offer to the appropriate RAs. Together with an organ offer, the DA_i has to provide the arguments that support his assessment over the organ's viability. In that way, a RA_j will be able to counter-argue DA_i's assessment when there is disagreement. The DA_i, in turn, will have the chance to counter-argue, and so on. Thus an argument-based dialog may take place between DA_i and RA_j. In particular, a DA_i's arguments for the non-viability of an organ may now be defeated by the RA_j's arguments for viability, and thus, RA_j may have the opportunity to make use of that organ. In the same way, DA_i's arguments for the viability of the offered organ may be stronger than those of a RA_j for non-viability. This will result in committing RA_j to transplant the offered organ as his decision for not transplanting it would be deemed unjustified.

Therefore, the *ProCLAIM* model is instantiated in order to extend the *CARREL* System so as to support the new selection process which we believe has the potential to increase the number of organs current selection processes make available. In particular, the proponent agents are the DA_i and RA_j, the GK is instantiated by the Acceptability Criteria Knowledge Base (ACKB) that encodes the criteria the medical doctors should refer to when deciding the organs' viability. The Argument Source Manager relates to the agents' reputation. Namely, the MA may deem as stronger the arguments submitted by agents with good reputation (*e.g.* a RA_j that have in the past successfully transplanted those organs which he claimed to be viable). Finally, the CBRe allows the MA to evaluate the submitted arguments on the basis of past transplantation experiences. For example, if an agent argues that the lung of a donor with a smoking history can safely be transplanted because he did not have COPD, the MA references the CBRe in order to evaluate this argument's evidential support. Note that at the same time, the submitted arguments highlight what are the relevant factors for deciding a case. Namely, the

argument graphs highlight the relevant attributes for assessing the similarity among cases.

The stage in the transplant experience in which arguments are submitted have associated different evidential weight. Arguments submitted before an organ is extracted are referred to as ***phase 1*** arguments and have associated weaker evidential wight. If an organ is deemed viable for a RA_j, the organ is extracted. At this time, new evidence may indicate that the organ is in fact non-viable, and so it is discarded. The RA_j is then obliged to provide *CARREL* with the new arguments (capturing the new evidence) as to *why* the organ is non-viable. These are referred to as ***phase 2*** (post-extraction/pre-transplantation) arguments. If complications arise after transplantation, then RA_j provides *CARREL* with arguments justifying (explaining) *how* the complications resulted in failure (eventually making the organ non-viable), or, conversely, arguments explaining *how* the complications were overcome so as to result in a successful transplant (eventually making the organ viable). These are referred to as ***phase 3*** (post-transplant) arguments and are deemed as providing stronger evidence. Hence, *phase 1* arguments are *presumptive*, submitted prior to undertaking any decision, whereas, *phase 2* and *3* arguments are submitted once the consequences of the decision is known, and so they are *conclusive* or *explanatory* arguments. We now give a short example of *phase 1* arguments.

a)

Scheme ID	Scheme Name	Scheme Description
VS	Viability Scheme	Donor *D* of organ *O* And no contraindications are known for donating organ *O* to recipient *R* Therefore, organ *O* is viable
NVS1	Non-Viability Scheme1	Donor *D* of organ *O* has contraindications *C* for donating organ *O* to recipient Therefore, organ *O* is non-viable
DDTS	Donor Disease Transfer Contraindicatiom Scheme	When transplanting organ *O* from donor *D* with condition *C1* to a recipient *R*, *R* may result in having condition *C2* and *C2* is harmful Hence, *C1* is a contraindication for donating *O*
DCCAPS	Donor Condition Course of Action Prevention Scheme	Following course of action *A* on recipient *R* prevents donor *D*'s condition *C1* resulting in condition *C2* on *R* And *A* is intended Therefore *R* will not result in having *C2* as a consequence of *D* having *C1*

b)

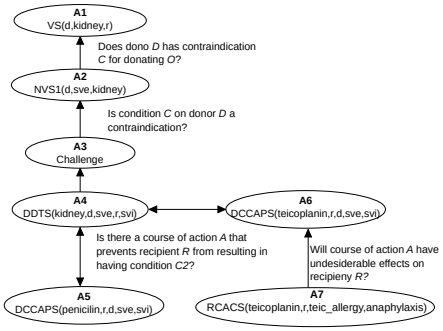


Figure 6: a) Fragment of the schemes in the ASR. b) Argument graph that results from arguing over the viability of a kidney of a donor with sve. The arrows represent the attack relation ($a \rightarrow b$, a attacks b) and the texts of the arrows are critical questions associated to the schemes. We denote $Scheme_i(x_1, .., x_n)$ as an instantiation of the scheme $Sheme_i$, with $x_1, .., x_n$ grounded and preserving the order in which the variables appear in the scheme definition. The graph also depicts the proposal for treating svi with $teicoplanin$, $A6$, but it is defeated by $A7$, the recipient r is $allergic$ to this antibiotic.

Figure 6a. captures the schemes used by the agents in order to argue over the viability of an offered kidney of a donor d whose cause of death was sve. The argument graph that may result from such deliberation is illustrated in figure 6b.

A deliberation must begin with the instantiation of the scheme that captures the decision under debate, the topic of the deliberation. In this case, the instantiation of the Viability Scheme ($A1$ in fig. 6b.). The later submitted arguments will attack $A1$ or reinstate it (see [49] for a more detailed description of the dialog process). Note that in fig. 6b. arguments $A5$ and $A4$ mutually attack each other. This is because the claim of $A5$ *–Recipient* r *will not result in having* svi *as a consequence of donor* d *having* sve*–* is in contradiction with the statement in $A4$ *–*r *may result in having* svi*–*. Intuitively, it remains a moot point as to whether administering *penicillin* is a sufficiently efficacious action for preventing svi in r ($A5$ wins out over $A4$) or not ($A4$ wins out over $A5$). Therefore, it cannot be concluded whether the kidney is viable or not. Applying Dung's calculus of opposition to the fig. 6b. graph only $A7$ is evaluated as winning. However, if we take $A5$ to asymmetrically defeat $A4$ (succeeds in its attack at the expense of $A4$'s attack on $A5$) then Dung's winning arguments are $A5$, $A3$ and $A1$. Thus, the organ would be deemed *viable*. But if $A4$ defeats $A5$, then $A4$ and $A2$ win and the organ would be deemed *non-viable*. In order to resolve this impasse in the argument evaluation, the MA makes use of the three knowledge resources: ACKB, the agents' reputation and the CBRe. Supposing *penicillin* is a novel treatment for preventing svi, the ACKB would not value argument $A5$ as reliable, and so the MA would derive that $A4$ defeats $A5$. However, supposing the agent submitting argument $A5$ has good reputation, $A5$ may be deemed stronger than $A4$, hence the MA would conclude that the kidney is viable[3]. We now describe the CBRe role in resolving conflicting arguments and the ASR structure the case base.

4.3 The Case-Based Reasoning Engine

Case Based Reasoning (CBR) has proven to be an appropriate reasoning and learning approach for ill-structured domains, where capturing experts' knowledge is difficult and/or the domain theory is weak or incomplete. However, CBR developers still have to face problems such as having to decide *how* to represent a case, *what* are the relevant factors for comparing them and *how* to retain new cases that encode, in a useful way, both the success and failure of the cases' proposed solutions. On the other hand, argumentation has proven to be a suitable approach for reasoning under uncertainty, with inconsistent knowledge sources, and in dialog based communication. However, one unresolved issue in argumentation is *how* to reuse the knowledge encoded in the arguments used in previous dialogs. A few approaches (see [13] and [32]) address this issue by providing support to end users for accessing or retrieving previous stored dialogs. On the other hand, [8] formalizes the way in which arguments used in previous legal cases can be integrated into the current dialog, also represented as a Dung's argument graph.

In this section we propose the use of CBR together with argumentation to: *1)* make use of previous resolved deliberations for evaluating the argument graph

[3]It is beyond the scope of this report to discuss the conflict resolution based on the other knowledge resources.

resulting from a new deliberation. This amounts to assigning a strength to the submitted arguments and possibly submitting additional arguments deemed relevant in previous similar deliberations; and *2)* organize the case memory by making use of the structure of argument schemes and critical questions encoded in the ASR. We show that in this way, a case can simply be defined as a placeholder for the available data in an experience (*e.g.* a transplant experience) together with the agents' submitted arguments; and that it is these argument graphs associated with each case that provide the means for case comparison. Moreover, these argument graphs represent two aspects of an experience. In the first case they capture the arguments exchanged by the proponent agents in arriving at a decision; thus *presumptive* arguments (*phase 1* arguments). In the second case they capture the *downstream* outcome of actions taken as a result of the decision arrived at in the first case; thus *conclusive* or *explanatory* arguments (*phase 2* and *3* arguments). In this way, the appropriateness of the decision is fed back into the argument graph associated with the case. Hence, the success and failure of a case's proposed solution is given by the dialectical status of the argument representing the decision.

4.3.1 Cases and Argument-Graphs Representation

Each (transplant) experience constitutes a case. The textual (medical) information describing an experience - the ***case description*** - along with the graph of (presumptive and explanatory) arguments submitted by the agents capture the case's features. In different experiences the arguments given by the agents may be the same, *i.e.* different *cases* may share the same graph (see fig.7). Each argument graph has an associated ***evidential support*** represented by a tuple of natural numbers *(F,K)*. F indicates the degree of certainty in the decision's correctness and K is the number of *cases* that share the argument graph. Thus, graphs with bigger F and K provide stronger evidence. Note that graphs representing cases with no feedback on the decisions' correctness have a more presumptive nature (smaller F) than those whose decision is supported or attacked by factual evidence (bigger F) which are more conclusive, or explanatory in nature. In the transplant scenario this accounts for F being 1,2 or 3 according to the phase in which the transplant experience was resolved. An argument graph may be deemed as having ***sufficient evidential support***, when the evidential support is bigger than a given threshold (*e.g.* $K > 5$).

As described in §4.2, it may be that the argument graph G of a new case may have nodes connected by bi-directional links, *i.e.*, arguments A and A' mutually attack. One of the CBRe tasks is to decide, on the basis of argument graphs associated with past experiences, whether A defeats A' or *vice versa*, and thus help establish whether a decision should be accepted (*e.g.* whether the organ being decided over in the new case is viable or not). Referring to the example in §4.2, this would involve determining whether the evidence represented by past cases indicates that *penicillin* is ($A5$ defeats $A4$) or is not ($A4$ defeats $A5$) effective in preventing the recipient from contracting *svi*. Another example would be the use

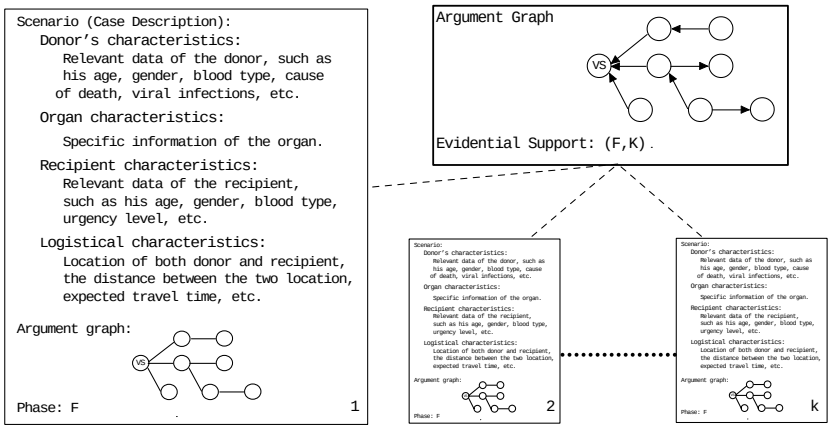


Figure 7: Case and Argument Graph Representation. Although, for readability purposes, we depict graphs as trees, with the argument for viability as the root, a child node may have more than one parent.

of evidence to determine whether or not lung transplants are successful where the donor had smoking history but no COPD. In the next subsection we describe the CBRe's reasoning cycle [1]. That is, the four processes: retrieve, reuse, revise and retain, that enable CBRe to carry out its task. As we will also see, these processes assume an organization in the case-base memory space given by the ASR.

4.3.2 The CBRe Reasoning Cycle

Retrieval: We describe in some detail the first reasoning process in which, given a target problem, the relevant cases for solving it are retrieved from the memory. The relevant cases will be represented by the argument graphs associated to the cases. The relevant graphs to retrieve are those whose arguments apply to the new situation and such that they have sufficient evidential support. The memory from which the *relevant* argument graphs are retrieved is a set M of directed graphs whose nodes are instantiated argument schemes or critical questions of the ASR, and whose edges represent attacks or defeats between arguments allowed by the ASR. Also, every graph $G_i \in M$ contains a single node that captures the topic under debate. In the transplant scenario this account only for the Viability Scheme. In order to facilitate the retrieval process, the memory space is organized on the basis of three partial orderings:

Definition 1 *Let S be defined as the memory space M and let S' be equal to S (containing the 'same' graphs) except that in S' the edges are not directional and nodes are the identifiers of the schemes or critical questions of ASR. Thus, if for example,* VS(donor,organ,recipient) *is a node of a graph in S its correspondent*

node in S' is VS[4]. *Let p_S be the canonical projection function from S into S'. Given $G1, G2 \in S$, we say that $G2$ structurally contains $G1$, $G1 \preceq_S G2$, if and only if the graph $p_S(G1)$ is a subgraph of $p_S(G2)$.*

Given a new target problem with an associated graph G, the CBRe first identifies those graphs in its memory M that structurally contain G[5], *i.e.* the set $\{G_1, \ldots, G_n\} \subseteq M$ such that for $i = 1 \ldots n$, $G \preceq_S G_i$ (where the set S of Definition 1 is $M \cup \{G\}$). The instantiation of schemes in G_i may differ from the instantiations in G. We wish to retrieve only those G_i whose instantiations are related to that of G as determined by an ontological hierarchy of instantiating terms.

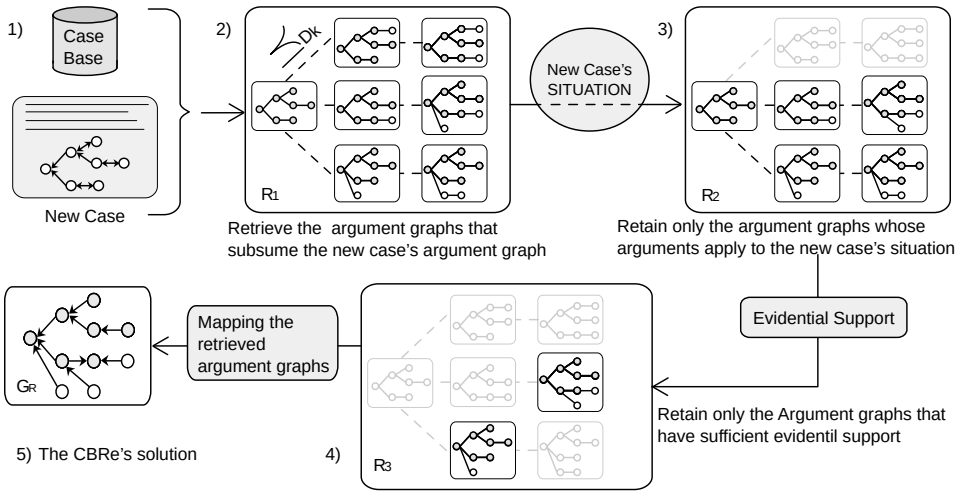


Figure 8: The Retrieval Process: steps 1 until 4. The Reuse Process, step 5

Definition 2 *Let O be the ontology whose terms instantiate the argument schemes of ASR, where O is expressed as an ordering $\prec_T$ on terms, and $\prec_T$ is interpreted as 'more specific than' (*e.g. $svi \prec_T bacterial\_infection \prec_T infection$*)*

We are only interested in those G_i related to G, where the degree of similarity, or the *distance* between them, falls below a given threshold. To evaluate this, we use the distance between terms in O, denoted as δ_O (*e.g.*, $\delta_O(infection, svi)$ = 2), to determine a distance δ_{sch} between scheme instantiations, and so a distance between graphs that share the same structure.

Definition 3 *Let $G1, G2 \in M$ such that $G1 =_S G2$ (*i.e. $p_S(G1) = p_S(G2)$*) and $sch_1 \ldots sch_m$ be the nodes of both $p_S(G1)$ and $p_S(G2)$. That is, for $j = 1 \ldots m$,*

[4]Graphs in S' ignore both, edges' direction and schemes' instantiation.

[5]Note that these are labelled graphs, moreover they all have a single node representing the topic under debate (*e.g.* Viability Scheme), hence, the graph comparison does not result in a computational overhead.

$sch_j(x_1, .., x_n)$ is a node in $G1$ iff $sch_j(y_1, .., y_n)$ is a node in $G2$. Then the ***distance*** *between $G1$ and $G2$ is given by: $\delta(G1, G2) = max_{j=1}^{m}(\delta_{sch}(sch_j(x_1, .., x_n)$ $, sch_j(y_1, .., y_n)))$, where $\delta_{sch}(sch_j(x_1, .., x_n), sch_j(y_1, .., y_n)) = max_{i=1}^{n}(\delta_O(x_i, y_i))$.*

We then state a threshold k such that the CBRe retains only those G_i such that $\delta(G, G_i) < k$. [6] To summarize, given a target graph G, CBRe retrieves the set $R_1 = \{G_1...G_n\}$ such that for $i = 1...n$, $G \preceq_{D_k} G_i$ (step 2 of fig. 8), where $\preceq_{D_k}$ is defined as follows:

Definition 4 *Let $G1, G2 \in M$ such that for some sub-graph $G3$ of $G2$, $G1 =_S G3$ (hence $G1 \preceq_S G2$). Then, $G1 \preceq_{D_k} G2$ if $\delta(G1, G3) < k$.*

From R_1, the CBRe excludes the graphs that have arguments that are not applicable in the target case, resulting in the set R_2 (step 3 of fig. 8). For example, a graph G_x in R_1 will not be retained in R_2 if G_x has an argument A_x that assumes the donor has property X which is not true in the target case. This process implies searching for property X on the donor in the target case's description. Note that if this property is actually found in the case's description G_x will remain in R_2 and thus argument A_x. Although no belonging to the target graph it may be deemed relevant for resolving the target case. From the resulting set R_2, the CBRe selects

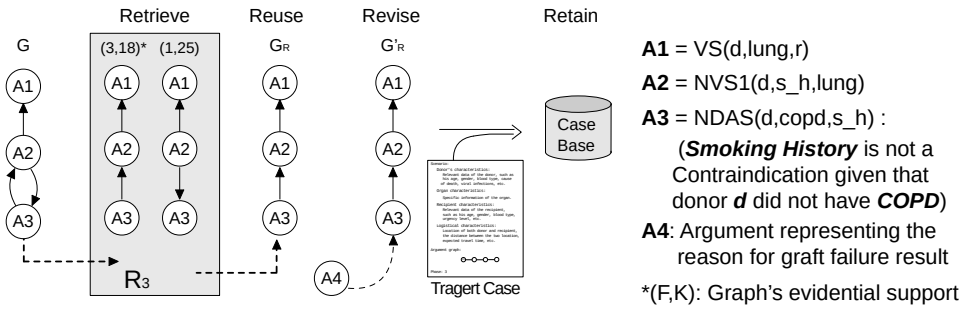


Figure 9: Smoking history example illustrating the CBRe reasoning cycle

the graphs with *sufficient evidential support* (see sub§4.3.1), resulting in R_3. At this stage (step 4 of fig. 8), each $G_i \in R_3$ is an argument graph that is applicable to the new case's situation, taking into account all the submitted arguments, and such that it has sufficient evidential support. Therefore, each argument in G_i is *relevant*.

Reusing: The aim of this process is to map R_3 to a solution for the target graph G. All the argument graphs in R_3 are merged into a single graph G_R such that it contains all the arguments in all graphs in R_3, and therefore in G (step 5 of fig.8), *i.e.* G_R is the minimal graph such that $G_i \preceq_{D_k} G_R$, $G_i \in R_3$. Note that in merging the graphs it may be that there are $G_1, G_2 \in R_3$ such that an argument A

[6] Note that the donors and recipients are not relevant for the graph comparison, thus $\delta_O(d, d') = 0$ and $\delta_O(r, r') = 0$ for every two donors d and d' and recipients r and r'.

asymmetrically defeats A' in G_1 but A' asymmetrically defeats A in G_2. We thus must decide the direction of the defeat (the edge direction) in G_R. Recalling the mutually attacking arguments $A4$ and $A5$ in the target graph shown in fig.6b), this amounts to deciding which argument asymmetrically defeats the other given the previous graphs G_1 and G_2, (where $A4$ = A and $A5$ = A').

Suppose that for each edge connecting arguments A and A' of $G_i \in R_3$ such that A asymmetrically defeats A' we associate the evidential support of G_i, writing $ES(G_i, A, A')$ = (F, K). Whereas if A does not asymmetrically defeat A' then $ES(G_i, A, A')$ = $(0, 0)$. Now, for every two connected arguments A, A' in G_R, if $max_{G_i \in R_3}(ES(G_i, A, A'))$ is **sufficiently greater**[7] than $max_{G_i \in R_3}(ES(G_i, A', A))$, then the edge in G_R will go from A to A' indicating that A defeats A'. Otherwise, A and A' will remain connected by a bi-directional edge in G_R indicating a mutual attack, which means there is no sufficient evidence to resolve the conflicting arguments.

Thus, G_R is the CBRe proposed solution, where, as described above, evidential supports are used to determine defeats and so a winning argument for viable or non-viable as described in §4.2. However, G_R can also determine the decision's validity given additional arguments in G_R that are not in G. G_R may identify additional arguments, not in G, that are applicable to the target case and belonging to a graph with sufficient evidential support. Thus, these additional arguments may identify new relevant factors for deciding the target problem which were not taken into account initially in G. Recall also that G will be a graph constructed from presumptive arguments (*phase 1* arguments), whereas G_R may also contain conclusive arguments (*phases 2* and *3* arguments). As described at the end of §4.2, *phase 2* arguments in G_R may provide conclusive evidence supporting a final decision for non-viability. *Phase 3* arguments may provide conclusive evidence supporting a final decision for non-viability or viability (*e.g.*, arguments describing post-transplant procedures that unsuccessfully, respectively successfully, dealt with post-transplant complications). To summarize, G_R provides: 1) evidential support to determine defeats amongst arguments in G and so determine the decision's validity; 2) new arguments for determining the decision's viability; 3) additional arguments that may serve, for example, as guidance to the RA_j for post-transplant management of patients.

Revising: The solution G_R must be tested in the real world, and if necessary, revised. This is achieved by requiring the agents to continue submitting arguments to G_R until the (transplant) experience ends. For example, if in the smoking history example the lung is deemed viable in G_R (see fig.9) but there is a graft failure the reasons for the failure will be submitted as new argument $A4$, that will reinstate the argument for non-viability. The resulting updated argument graph G'_R will then be stored in the case base.

[7]The definition of 'sufficiently greater' is domain dependent. We can say that $(F1, K1)$ is sufficiently greater than $(F2, K2)$ if : a)$F1 > F2$, or b) $F1 = F2$ and $F1 > 1$, $F2 > 1$, and $K1 * 0.6 > K2$.

Retain: The aim is to store the possibly updated G'_R as a new graph in the memory. Hence, when a (transplant) experience finishes, the *case* describing this experience is retained by the CBRe. If there already exists an argument graph G_M in the memory such that $G'_R =_{D_k} G_M$ and the edges directions coincide, then the *case* is associated with G_M increasing G_M's evidential support. Otherwise, the target case is retained as associated to G'_R which is added as a new argument graph to M.

4.4 Discussion

Previous works that have combined argumentation with cases have focussed in providing support to end users in the construction of arguments using previous cases, rather than using previous dialogs to evaluate the arguments' strengths submitted in a new dialog. Such works proliferate in the legal domain, particularly in the context of the *Common law system*, a legal system based on unwritten laws developed through judicial decisions that create binding precedent. The inherent argumentative nature of the legal domain and the particular features of the Common law system provide a scenario for developing models and systems for reasoning and arguing with precedents, *i.e.* past cases. Exponents of these works are systems such as HYPO [3], CATO [2] and CABARET [46]. However, it could be argued that these works are best described as systems or models for arguing with cases, rather than CBR systems in the sense of [1]. Moreover, systems such as HYPO, do not define any kind of automated procedure for retaining new cases.

Also intended for assisting end users in the construction of arguments is the extension to the HERMES System proposed in [32] that aims to support human agents involved in group decision making processes to retrieve, adapt and re-use past cases.

Currently the CBRe is being prototyped so as to extend an existing prototype of the logical argumentation model described in [38]. Future work will focus on extending the retrieval process so as to address adaptation of previous cases in order to increase the scope of the relevant cases. Another future line of work is use of the case base for searching patterns in order to propose new arguments, *i.e.* to propose new instantiations of argument schemes (*e.g.* relating a donor condition x with unsuccessful transplants: NVS1(d,x,$organ$)). Furthermore, we intend to formalize the use of the CBRe in triggering revisions on the ACKB (and in general, on the GK). Namely, the use of the CBRe for detecting *when* there is sufficient evidence to change the current established guidelines encoded in the GK. For example, determining that there is sufficient evidence to change the current medical criteria so as to encode that sve on a donor can no longer be considered as a contraindication for transplanting a $kidney$ when the recipient is not allergic to $penicillin$ or $teicoplanine$.

The transplant scenario serves to illustrate *ProCLAIM*'s practical value. We believe *ProCLAIM*, and in particular the CBRe, may also prove to be useful in other safety-related environments. We are currently investigating the application of *ProCLAIM* as an extension to *DAI-DEPUR* [19], a decision support systems

for Wastewater Treatment Plants (wwtp). In this scenario, the proponent agents would represent the wwtp operators, the Argument Source Manager would relate to the operators' hierarchy within the plant and the GK would be instantiated by the guidelines encoding compliance with the environmental legislations. The CBRe will help to establish on an evidential basis, indicating whether the operators' decisions are appropriate and thus environmentally safe, in light of their given arguments.

A Correction of rule evaluation functions

The goal of a single step in rule learning algorithm is to return a rule with accurate class predictions, or, in other words, have a high probability of the positive class among all examples (not only learning examples) covered by rule. Thus, it would make sense to use relative frequency, an unbiased estimator of probability:

$$Q(r) = \frac{s}{n} \tag{5}$$

where n is the number of learning examples covered by the rule r and s is the number of positive examples among them.

However, the assumption that the relative frequency indeed estimates the probability of positive class is wrong. Fig. 10(a) shows how searching through a large space of rules, which tries to maximize relative frequencies, can always find rules with 100% positive subsets, though these are usually purely random patterns in the data and their true positive class probabilities are much lower.[8] Class proportions for the rules found by the search process are thus completely uncorrelated with the true class probabilities.

A more general version of this problem has been extensively explored by Jensen and Cohen [30] who blame multiple comparisons during the search to be responsible for plethora of pathologies in induction algorithms. Our paper proposes a method which can fix the relative frequency estimate and other rule evaluation measures by taking multiple comparisons into account through the use of extreme value distributions [24].

A.1 Experimental Study of Rule Estimators

The m-estimate [15] computes the class probability (or, in our case, the rule quality) as

$$Q_m(r) = \frac{s + m \times p_a}{n + m} \tag{6}$$

where p_a is the prior probability and m is a parameter of the method. Fuernkranz and Flach [25] showed that the m-estimate presents a trade off between precision (relative frequency) and linear cost metrics (for instance, weighted relative accuracy [34, 47]). Different values of parameter m can be used to approximate many currently used evaluation functions. For instance, when $m = 0$, m-estimate equals the relative frequency. Instead of citing various proposals from the extensive related work, we shall thus concentrate on the more general m-estimate.

To observe the correlation between the true and the estimated class probabilities, we constructed a set of artificial data sets with controlled class probabilities for each possible rule. We have prepared 300 data sets with ten binary attributes. Five attributes in each data set were unrelated with the class. For the other five, we prescribed a (random) class probability for each combination of their values.

[8] Experimental details are provided in the next section.

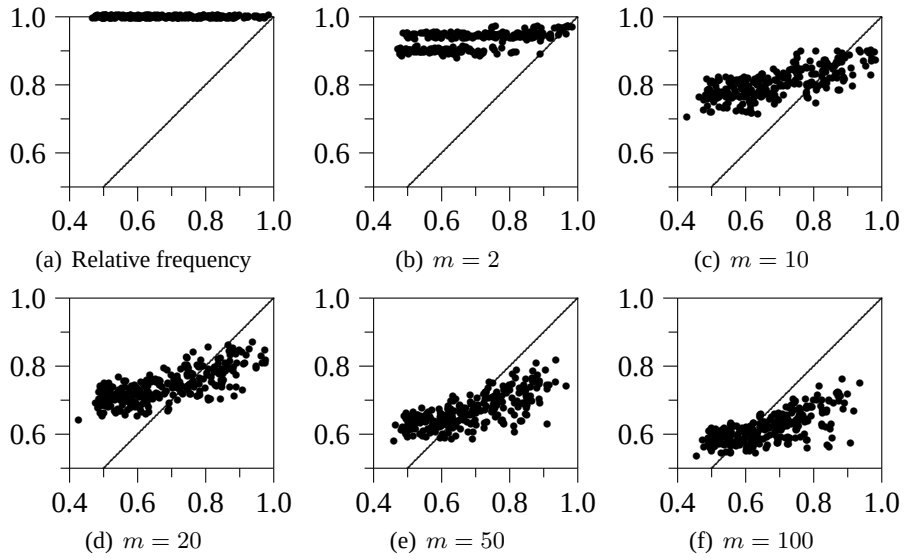


Figure 10: Relation between the estimated (y-axis) and true (x-axis) class probabilities for rules from artificial data sets.

We then generated 2^{10} examples for each data set, one for each combination of attribute values, and assigned the classes randomly according to the prescribed probabilities for the combination of informative attributes. Note that the actual class proportions in the data set do not necessarily match the defined probabilities for a particular combination of attribute values.[9]

For each of 300 data sets we learned a single rule using different values for m (0, 2, 10, 20, 50, 100). Fig. 10 shows the relation between the rule's estimated class probability $Q_m(r)$ and the known true probability, which we shall denote by $\widetilde{Q}(r)$. As we already mentioned in the introduction, at $m = 0$ (relative frequency), the method is extremely optimistic. With increasing values of m, the method is still optimistic for rules with lower true probability, but pessimistic for rules with higher true probability. It seems that m-estimate lowers the estimated quality by the same amount for all rules, which can not adjust the estimates to lie on (or at least close to) the ideal diagonal line representing the perfect correlation.

Table 9 compares the measured evaluation functions by

- the average true prediction accuracy of the induced rules, which reveals the quality of the evaluation function as search heuristics;
- the Spearman correlation coefficient between $Q_m(r)$ and $\widetilde{Q}(r)$, that shows the quality of the rule ordering, which is crucial when rules are used for classification, where we need to distinguish between "stronger" and "weaker" rules;

[9]We obtained similar results in experiments with other ways of constructing artificial data sets.

Table 9: Comparison of rules obtained from artificial data sets with different values for m: the average true class probability, Spearman correlation between the true probability and the estimate, and the mean square error of the estimate.

m	avg. accuracy	Spearman	mean error
0	0.68	0.00	0.119
2	0.68	0.54	0.074
10	0.68	0.68	0.027
20	0.68	0.72	0.015
50	0.67	0.70	0.009
100	0.66	0.65	0.010

- the mean square difference between $Q_m(r)$ and $\widetilde{Q}(r)$ which indicates the rule's accuracy when used as probabilistic predictor.

The first column of Table 9 suggests that lower values of m give (marginally) better rules than higher values. However, higher m's score better in terms of the Spearman correlation and give better probability estimates.

In conclusion, using the m-estimate with a suitably tuned m can considerably decrease the error of the estimated probabilities, yet, as seen from the graphs, the major effect comes from reducing the optimism, while the correlation between the true and the estimated probability remains rather poor. m-estimate and the many other similar techniques are thus not a satisfactory solution to the problem of overfitting, wrong rule quality estimates and optimistic probability predictions.

A.2 Algorithm for Improved Probability Estimate

Relative frequencies, m-estimates and other potential measures of rule quality are computed from the number of examples covered by the rule (n) and the number positive examples among them (s). We have seen that relative frequencies overestimate the true probability because the algorithm searches for the rule with the highest $s : n$ ratio. Since the training data presents only a limited sample from the population, the observed ratio for each rule is subject to random distribution, so the found rule is therefore not necessarily the optimal one, and it almost certainly has an optimistic $s : n$ ratio. One way of preventing these unwanted effects and improving the probability estimate s/n is to try to find the expected value of s, which we shall denote by $\tilde{s}$.

The outline of the proposed procedure is illustrated in Fig. 11. For reasons that will become clear later, we start by computing the log-likelihood ratio statistics (LRS) for 2×2 tables derived by Dunning [23]. It is usually assumed that LRS is distributed according to $\chi^2(1)$. This is, however, true only for randomly chosen rules, or, in our case, for LRS computed from the expected value of s, $\tilde{s}$.

The highest observed LRS (computed from s, not $\tilde{s}$) is distributed according

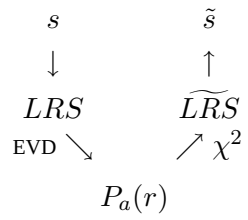


Figure 11: An outline of the proposed procedure

to the Fisher-Tippet extreme value distribution (EVD) [24].[10] Since EVD depends only the number of rules considered in the search (it is more likely to get higher LRS if number of rules considered is higher), which is determined by the rule length, the chosen search algorithm and the properties of the data set (number of examples, number and type of attributes), we will be able to compute corresponding EVDs – for rules of different lengths for the selected algorithm and a particular data set – in advance.

Now consider a specific rule. From n and the observed s, we can compute the LRS and then, knowing its distribution (EVD), find the probability that a rule this good (or better) is found under the null-hypothesis of no relation between the attribute and class values. We shall denote this probability, the "significance of the rule", by $P_a(r)$. Note that $P_a(r)$ takes the multiple comparisons into account, so this estimate is unbiased.

On the other hand, imagine that we knew the expected $\tilde{s}$ of that rule. We could compute its true log-likelihood ratio $\widetilde{\text{LRS}}$ and, through $\chi^2(1)$ distribution, arrive at the same significance $P_a(r)$ as above.

The trick that we use in this paper is to reverse the second path. So, from the unbiased $P_a(r)$ which we get from the known (but optimistic) s through LRS and EVD, we shall compute the unbiased $\widetilde{\text{LRS}}$ and the corresponding $\tilde{s}$.

The reason for which we need to compute LRS instead of computing the extreme value distribution for $Q(r) = s/n$ directly, and estimate the unbiased $\widetilde{\text{Q}}$ through $P_a(r)$, is that the extreme value of a sampled random variable (such as s, $Q(r)$ or LRS) is distributed by the Fisher-Tippet (or some other) extreme value distribution only if the variable's values are taken from a fixed distribution (independent from s and n). LRS, as we just noted, fulfils this criterion, while s/n is distributed according to $\beta(s, n-s)$ and is thus not the same for all rules.

As a side note, our approach to correcting quality estimators can be generalized to other criteria beside $Q(r) = s/n$. If the density distribution of a criterion depends upon the rule (like is the case with s/n), we need to find a measure which is well-correlated with the criterion (additional explanation is given later), yet drawn from a fixed distribution (like LRS, which is drawn from $\chi^2(1)$ and still reason-

[10] For illustration, although the daily levels of a river are usually distributed normally, the distribution of maximal annual levels is not normal but Fisher-Tippet's.

ably correlated with s/n).[11] If the observed criterion already comes from a fixed distribution (if, for example, LRS would be used as the main evaluation function), finding a correlated measure is not needed and we can immediately proceed to the computation of EVD.

This section will present the details of the algorithm, along with a running example for illustration.

A.2.1 Step 1: From s to LRS.

Let s again be the number of positive examples covered by rule and let s^c be the number of positive examples not covered by the rule. Similarly let n be the number of covered examples by the rule and n^c be the number of examples not covered by the rule. LRS is then defined as:

$$\text{LRS} = 2\left[s\log\frac{s}{e_s} + (n-s)\log\frac{n-s}{e_{n-s}} + s^c\log\frac{s^c}{e_{s^c}} + (n^c-s^c)\log\frac{n^c-s^c}{e_{n^c-s^c}}\right] \tag{7}$$

where e_x is the expected value of x. For instance, e_s is computed as $n\frac{s+s^c}{n+n^c}$. When computed on a randomly chosen rule, LRS is distributed according to $\chi^2(1)$ distribution, disregarding properties of the rule (length, s, $n\ldots$) and the data. Note that a similar formula for LRS, without the last two terms, was used in [17, 16] for computing significance of rules. However, as that formula is approximately correct only if n is small enough when compared to n^c, we prefer to use the formula 7 derived by Dunning [23].

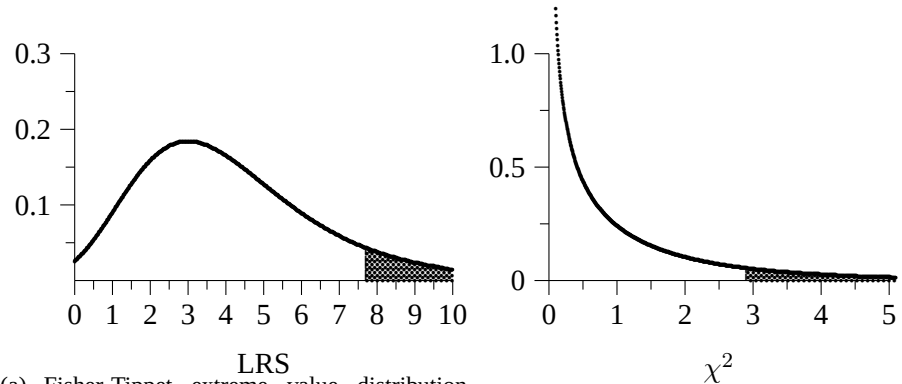
Example. We have a data set with 20 examples where the prior probability of the positive class is 0.5. Learning from that data, the rule search algorithm found a rule r with two conditions which covers 10 examples with 8 of them belonging to the positive class. Its LRS is, according to (7), 7.7.

A.2.2 Step 2: From LRS to $P_a(r)$.

$P_a(r)$ measures the probability that, given a random data with no relation between the attribute and class values, the rule found by the chosen search procedure will have the quality of at least LRS(r) (or another suitable measure of quality). This definition suggests a way of computing $P_a(r)$: like Jensen and Cohen [30], we permute the class values in the data set so that all rules are purely random and their true probability for positive class equals the prior probability. We then induce a rule on the randomized data set and compute its LRS. Repeating it for many times we get a distribution for LRSs.

Gumbel and Lieblein [28, 29] (cited in [35]) have shown that the limiting distribution of all χ^2 distributions is the Fisher-Tippet distribution (Fig. 12(a)). Fisher-Tippet distribution is characterized by two parameters, location (μ) and scale (β).

[11] The correlation would be perfect if every rule covered the same number of examples.



(a) Fisher-Tippet extreme value distribution ($\mu = 3$, $\beta = 2$)

(b) χ^2 with 1 degree of freedom.

Figure 12: Probability density functions

For LRS, it can be shown that β always equals 2, and μ equals the median of the above sample to which we add $2 \ln \ln 2$. In general, values of μ and β depend upon the number of rules covered by the search (which does not necessarily equal the number of *explicitly* evaluated rules), which in turn depends upon the rule length and the data set (and, of course, the search algorithm). Due to their independence of the actual rule, we can compute values $\mu(L)$ and $\beta(L)$ for different rule lengths before we begin learning, using the algorithm shown in Fig. 13. The algorithm runs until $\mu(L)$ is smaller than $\mu(L-1)$, which signifies that rules of length $L-1$ can not be improved because they are perfect or they do not cover enough examples.

During learning we use the cumulative Fisher-Tippet distribution function with the pre-computed parameters to estimate $P_a(r)$.

Example (continued). Say that algorithm from Fig. 13 found $\mu(2) = 3$ and $\beta(2) = 2$ (remember that rule r has two conditions). The curve with such parameters is depicted in Fig. 12(a), so the probability $P_a(r)$ for the rule from our example corresponds to the shaded area right of LRS=7.7. $P_a(r)$ equals approximately 0.09.

A.2.3 Step 3: From $P_a(r)$ to $\widetilde{\text{LRS}}(r)$.

To compute $\widetilde{\text{LRS}}(r)$ we need to do the opposite from the last step. Looking at the $\chi^2(1)$ distribution (Fig. 12(b)), we need to find such a value $\widetilde{\text{LRS}}(r)$ that the area under the curve to the right of it will equal $P_a(r)$. In other words, the shaded areas under the curves in Fig. 12 should be the same.

[12]Note that using LRS at a given rule length will always order rules the same as would $\widetilde{\text{LRS}}$. However, as we will be using $(s)/n$ in the actual learning phase, in order to correctly estimate parameters of Fisher-Tippet distribution, measures $\tilde{s}/n$ and $\widetilde{\text{LRS}}$ should be well correlated.

1. Let $L = 1$ (L is the maximum rule length).
2. Permute values of class in the data.
3. Learn a rule on this data (using LRS as evaluation measure), where the maximum length of rule is L.[12]
4. Record the LRS of the rule learned.
5. Repeat steps 2-4 to collect a large enough (say 100) sample of LRSs
6. Estimate parameters $\mu(L)$ and $\beta(L)$ of the Fisher-Tippet distribution.
7. If $\mu(L) > \mu(L-1)$, then $L = L + 1$ and return to step 2.

Figure 13: The algorithm for computing parameters of the Fisher-Tippet distributions

Example (continued). The corresponding $\widetilde{\text{LRS}}$ for our examples as read from Fig. 12(b) is 2.9. Note that this is much less than LRS $= 7.7$, which we computed directly from the data and which would essentially be used by an unmodified rule induction algorithm.

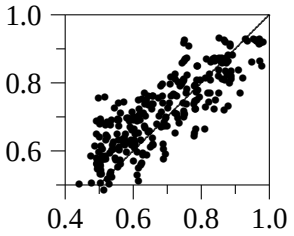
The remaining task is trivial: compute $\tilde{s}$ from the formula for $\widetilde{\text{LRS}}$ using an arbitrary root finding algorithm. Similar would be done for statistics other than $\widetilde{\text{LRS}}$ and $\tilde{s}$. In our task we are correcting probability estimates based on relative frequencies, so we shall compute them by dividing the corrected $\tilde{s}$ by n.

Example (conclusion). We used Brent's method [6] to find that $\widetilde{\text{LRS}} = 2.9$ corresponds to $\tilde{s} = 6.95$. The rule covers ten examples, so the corresponding class probability is $6.95/10 = 0.695$. Note that this estimate is quite smaller than the uncorrected 0.8.

A.3 Experiments

We have tested the algorithm on artificial data described in Section A.1. We used beam search [16, 17] with a beam width set to 5. The algorithm was implemented as a component for the rule based learner in machine learning system Orange [20].

The results of using the corrected measure on the artificial data are shown in Fig. 14. The estimated class probabilities are nicely strewn close to the diagonal axis, which is a clear improvement in comparison with the results from Fig. 10. This is also confirmed by the quantitative measure of fit: the average true probability is the same as the highest values in Table 9, the mean quadratic error is a little better than that of m-estimates, while the Spearman coefficient is clearly superior.



Average quality: 0.68
Spearman correlation: 0.83
Mean squared error: 0.007

Figure 14: Relation between the corrected (y-axis) and the true (x-axis) class probability.

A.4 Correction of evaluation of rules learned from arguments

In the search for the best rule with the help of arguments the algorithm makes less comparisons than classical search due to the following two reasons:

- a part of the rule is already given as an argument and
- selected conditions must be true for argumented example.

We need to make two changes to the algorithm from the previous section so that it can consider less comparisons and still correctly compute class probability.

A.4.1 Computing parameters of the Fisher-Tippet distributions

In the algorithm for computing Fisher-Tippet (Fig. 13) we need to add another step after step 2:

- Select a random example from data.

This example is then used in step 3 as the constraint for rule induced by CN2 - rule must cover this example. This changes to the algorithm have the expected output - estimated mode values of LRS (μ) are lower.

A.4.2 Removing optimism from rule's LRS

Let us briefly recall the algorithm for removing optimism of rule's LRS. We start by computing LRS, then estimate the probability that such rule would be found by luck P_a, and, in the last step, find such corrected LRS that gives the same probability. In the case of learning from argumented examples, there were less comparisons made as some of the rule's conditions were already determined by the argument. Therefore, we need to change the step where we estimate the probability that such rule was found by luck.

Let L be the number of conditions of rule and L_a the number of conditions of rule given from argument. Then, we first need to split LRS to LRS_a (LRS obtained due to argument) and LRS_s. LRS_a is likelihood ratio statistics of argument

if used as rule and $LRS_s = LRS - LRS_a$. Obviously, the optimistic part of LRS is only LRS_s, as argument was given by expert - without searching comparing several rules. We correct LRS_s with a method similar to the one described and obtain $\widetilde{LRS}_s$. The special thing of this method is the computation of P_a. In the standard algorithm P_a is computed as:

$$P_a = e^{-e^{-\frac{LRS-\mu[L]}{\beta[L]}}} \tag{8}$$

while in our case this formula changes to

$$P_a = e^{-e^{-\frac{LRS_s-\mu[L]+\mu[L_a]}{\beta[L]}}} \tag{9}$$

The corrected $\widetilde{LRS}$ of the whole rule is thus $LRS_a + \widetilde{LRS}_s$.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.

[2] V. Aleven. *Teaching case-based argumentation through a model and examples*. PhD thesis, University of Pittsburgh, Pittsburgh, PA., 1997.

[3] K. D. Ashley. Reasoning with cases and hypotheticals in hypo. *International Journal of Man-Machine Studies*, 34(6):753–796, 1991.

[4] K. D. Ashley and E. L. Rissland. Law, learning and representation. *Artificial Intelligence*, 150:17–58, 2003.

[5] ASPIC. Final consensus framework of argumentation concepts. Deliverable 1.4 for the ASPIC project, 2005.

[6] K. E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley and Sons, New York, 1989.

[7] T. Bench-Capon. Neural nets and open texture. In *Fourth International Conference on AI and Law*, pages 292–297, Amsterdam, 1993. ACM Press.

[8] T. Bench-Capon. Representation of case law as an argumentation framework. In *Legal Knowledge and Information Systems. JURIX 2002*, pages 103–112, 2002.

[9] T. Bench-Capon and F. Coenen. An experiment in discovering association rules in the legal domain. In *11th International Workshop on Database and Expert Systems Applications*, pages 1056–1060, Los Alamitos, 2000. IEEE Computer Society.

[10] F. Borges, P. Borges, and B. D. A connectionist model to justify the reasoning of a judge. In *Proceedings of Jurix*, pages 113–122, IOS Press, 2002. IOS Press.

[11] I. Bratko and M. Možina. Argumentation and machine learning. In: Deliverable 2.1 for the ASPIC project, 2004.

[12] S. Brüninghaus and K. D. Ashley. Predicting the outcome of case-based legal arguments. In G. Sartor, editor, *Proceedings of the 9th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 233–242, Edinburgh, United Kingdom, June 2003.

[13] S. Buckingham Shum. Design argumentation as design rationale. *The Encyclopedia of Computer Science and Technology (Marcel Dekker Inc: NY)*, 35:95–128, 1996.

[14] F. Caballero, A. López-Navidad, M. Perea, C. Cabrer, L. Guirado, and R. Solà. Successful liver and kidney transplantation from cadaveric donors with left-sided bacterial endocarditis. *American Journal of Transplantation*, 5(4):781–787, 2005.

[15] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–149, 1990.

[16] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Machine Learning - Proceeding of the Fifth Europen Conference (EWSL-91)*, pages 151–163, Berlin, 1991.

[17] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning Journal*, 4(3):261–283, 1989.

[18] A. consortium. Draft formal semantics for aspic system. In: Deliverable 2.5 for the ASPIC project, 2005.

[19] U. Cortés, I. R.-Roda, M. Sànchez-Marrè, J. Comas, C. Cortes, and M. Poch. Dai-depur: An environmental decision support system for the control and supervison of municipal wastewater treatment plants. In *ECAI*, pages 603–607, 2002.

[20] J. Demšar and B. Zupan. Orange: From experimental machine learning to interactive data mining. White Paper [http://www.ailab.si/orange], Faculty of Computer and Information Science, University of Ljubljana, 2004.

[21] P. Domingos. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[22] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[23] T. E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[24] R. Fisher and L. Tippett. Limiting forms of the frequency distribution of the largest and smallest member of a sample. *Proc. Camb. Phil. Soc.*, 24:180–190, 1928.

[25] J. Fuernkranz and P. A. Flach. Roc 'n' rule learning – towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, January 2005.

[26] S. A. Gomez and C. I. Chesnevar. Integrating defeasible argumentation and machine learning techniques. Technical report, Universidad Nacional del Sur, 2004.

[27] M. Groothius and J. Svensson. Expert system support and juridical quality. In *Jurix*, pages 1–10, Amsterdam, 2000. IOS Press.

[28] E. J. Gumbel. Statistical theory of extreme values and some practical applications. *National Bureau of Standards Applied Mathematics Series (US Government Printing Office)*, 33, 1954.

[29] E. J. Gumbel and J. Lieblein. Some applications of extreme-value models. *American Statistician*, 8(5):14–17, 1954.

[30] D. D. Jensen and P. R. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, March 2000.

[31] B. Johnston and G. Govenatori. Induction of defeasible logic theories in the legal domain. In *Ninth International Conference on AI and Law*, pages 204–213, Edinburgh, 2003. ACM Press.

[32] N. Karacapilidis, B. Trousse, and D. Papadias. Using case-based reasoning for argumentation with multiple viewpoints. In *ICCBR 1997*, 1997.

[33] J. Kolodner. *Case-based reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[34] N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 174–185, Bled, Slovenia, 1999.

[35] W. Li, F. Sun, and I. Grosse. Extreme value distribution based gene selection criteria for discriminant microarray data analysis using logistic regression. *Journal of Computational Biology*, 11(2/3):215–226, 2004.

[36] T. Lindgren. Methods for rule conflict resolution. In *Proc. of the 15th European Conference on Machine Learning (2004)*, pages 262–273, Pisa, Italy, Sept. 2004.

[37] A. López-Navidad and F. Caballero. Extended criteria for organ acceptance: Strategies for achieving organ safety and for increasing organ pool. *Clin Transplant, Blackwell Munksgaard*, 17:308–324, 2003.

[38] S. Modgil, P. Tolchinsky, and U. Cortés. Towards formalising agent argumentation over the viability of human organs for transplantation. In *MICAI*, pages 928–938, 2005.

[39] M. Možina, J. Demšar, J. Žabkar, and I. Bratko. Why is rule learning optimistic and how to correct it. In *ECML proceedings, to be published*, 2006.

[40] M. Možina, J. Žabkar, T. Bench-Capon, , and I. Bratko. Argument based machine learning applied to law. In *Argumentation in Artificial Intelligence and Law, Workshop at Tenth International Conference on Artificial Intelligence and Law (ICAIL 2005)*, pages 87–94, Bologna, 2005. Wolf Legal Publishers.

[41] M. Možina, J. Žabkar, T. Bench-Capon, , and I. Bratko. Argument based machine learning applied to law. *Artificial Intelligence and Law*, to appear in 2006.

[42] M. Možina, J. Žabkar, and I. Bratko. Argument based rule learning. In *ECAI proceedings, to be published*, 2006.

[43] P. M. Murphy and D. W. Aha. UCI Repository of machine learning databases [`http://www.ics.uci.edu/~mlearn/mlrepository.html`]. Irvine, CA: University of California, Department of Information and Computer Science, 1994.

[44] ONT. Organización Nacional de Transplantes. http://www.ont.es.

[45] H. Prakken and G. Vreeswijk. *Handbook of Philosophical Logic, second edition*, volume 4, chapter Logics for Defeasible Argumentation, pages 218–319. Kluwer Academic Publishers, Dordrecht etc, 2002.

[46] D. B. Skalak and E. L. Rissland. Arguments and cases: An inevitable intertwining. *Artificial Intelligence and Law*, 1(1):3–44, 1992.

[47] L. Todorovski, P. Flach, and N. Lavrač. Predictive performance of weighted relative accuracy. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the 4th European Conference of Principles of Data Mining and Knowledge Discovery (PKDD-00)*, pages 255–264, Lyon, France, 2000.

[48] P. Tolchinsky, U. Cortés, J. C. Nieves, F. Caballero, and A. López-Navidad. Using arguing agents to increase the human organ pool for transplantation. In *3rd Workshop on Agents Applied in Health Care (IJCAI-05)*, 2005.

[49] P. Tolchinsky, S. Modgil, and U. Cortés. Argument schemes and critical questions for heterogeneous agents to argue over the viability of a human organ. In *AAAI 2006 Spring Symposium Series; Argumentation for Consumers of Healthcare*, 2006.

[50] S. Toptchiyski. Large scale demonstration scenario. In: Deliverable 5.1 for the ASPIC project, 2005.

[51] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928. English Translation Fin Tucker, A.W. and R.D.Luce, Contributions to the Theory of Games IV, Annals of Mathematics Studies 40, 1959.

[52] J. Žabkar, M. Možina, and I. Bratko. Draft formal semantics for argumentation-based data-mining and machine learning. Deliverable 2.4 for the ASPIC project, 2005.

[53] D. N. Walton. *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1996.

[54] J. Zeleznikow and A. Stranieri. Knowledge discovery in the split up project. In *Proceedings of the Sixth International Conference on AI and Law*, pages 89–97, New York, 1997. ACM Press.