Adapting tableaux for classification

M.G. Jansen[1], A. Th. Schreiber[1], and B. J. Wielinga[1]

University of Amsterdam, Social Science Informatics
Roetersstraat 15, NL-1018 WB Amsterdam, The Netherlands
E-mail: {jansen,schreiber,wielinga}@swi.psy.uva.nl

Abstract. We present an approach for studying logical properties of problem-solving methods (PSMs) for knowledge-intensive tasks. It is based on semantic tableaux (a deduction-style theorem-proving technique). We show how tableaux can be manipulated in a methodical way to formalize non-deductive style PSMs.

1 Introduction

Knowledge-engineering research has delivered an abundance of problem-solving methods (PSMs) for classes of tasks. such as classification, diagnosis, and configuration (see e.g., [2]). These PSMs are used in practical knowledge systems. To get a better understanding of the different PSMs work has been done on the formalization of PSMs, e.g. [4, 10, 11]. In this paper we describe another approach, in which we adapt semantic tableaux to formalize PSMs. We show that this provides us with a technique for studying the logical properties of PSMs. In particular, we show that this approach allows us to model non-deductive style reasoning. This was a problem with previous formalizations based on first-order logic. We use the classification task as an example.

2 Characterizing Classification

We can characterize a knowledge-intensive task by defining three aspects: (1) the goal of the task, (2) the ontological commitments, and (3) the solution criteria.

Goal The goal is typically an informal description of what the task attempts to achieve. In the case of classification, the goal is to identify to which class a certain object belongs. Example classification tasks are apple classification, rock classification, and art-object classification.

Ontological commitments Ontological commitments describe our assumptions on the representation of the task domain. They provide us with a vocabulary. We can use this vocabulary to define what we mean with classification. Together, the ontological commitments form an ontology for a task. An example of such task ontology is the configuration-design ontology [6]. For classification we base the task ontology on the descriptions of classification given by [9, 8].

We define six basic ontological types, namely **attribute**, **object**, **value**, **class**, **feature**, and **observation**. A is a (finite) set of attributes, each of which is associated with a list of possible values. A feature is an admissible attribute-value pair. Objects that need to be classified are described by a finite number of attribute-value (AV) pairs. These AV-pairs are called observations. The set of observations for a particular object is called Obs.

By definition, we assume that an attribute can have only one value at the time. So, if $colour$ is an attribute and$\{red, yellow, blue\}$ is its list of possible values, then an object description can never contain the AV-pairs $colour = red$ and $colour = yellow$ simultaneously. Every attribute with more than two values can be transformed into several attributes which all have binary values. This transformation is performed as follows: For every such AV-pair we take a new attribute with possible values $true$ and $false$. The new attribute has the value $true$ if the original AV-pair holds, and $false$ otherwise.

Note that after having applied such a transformation the exclusion of multiple values for an attribute is no longer guaranteed. In order to maintain this principle every new attribute has the original attribute as it's type. Now we can say that exclusion of multiple values holds for binary AV-pairs of the same type. If one binary attribute has the value true, then all other attributes of the same type have the value false. In this way, each multi-valued attribute can be represented as a set of atomic propositions.

Classes can now be represented as follows:

$$c \rightarrow (a_1 \vee ... \vee a_n) \wedge ... \wedge (b_1 \vee ... \vee b_n)$$

The class name is represented by the proposition c and implies its features (i.e. AV-pairs). Features are here represented as atomic propositions with an index for ease of representation. a_1 represents the feature where a designates the attribute and the index $_1$ a certain value.

Domain knowledge can be represented by assigning meaningful names to such a structure. For example:

$$blackbird \rightarrow$$
$$(plumage = black \vee plumage = brown) \wedge (bill = yellow)$$

The domain theory DT consists of a conjunction of class definitions.

Solution criteria There are several alternative criteria one can formulate with respect to the goal of the classification process. We define two criteria:[1]

Weak classification In weak classification (WC) a candidate solution must be a class c which is consistent with the domain theory DT and the observations Obs made thus far. Formally, this set of candidate solutions S can be expressed by:

$$S = \{c | DT \cup \{c\} \cup Obs \nvdash \bot\}$$

[1] In Sec. 4.3 we introduce a third form of classification, namely composite-solution classification.

Strong classification In strong classification (SC) a class c is a member of the set of candidate solutions S iff the domain theory together with c explains all observations. That is, we want candidate solutions to be classes which actually possess the properties that have been observed. Formally, the criterion for SC is:

$$S = \{c | DT \cup \{c\} \models Obs\}$$

The criterion of strong classification is stronger in a logical sense than weak classification. If a class is a candidate solution according to SC it is also according to WC. This follows as SC can be formulated as an extension of WC.

3 Tableaux

Semantic tableaux were developed in the 50's (see e.g. [3]). Like resolution they form a *refutation* system. In a tableau proof a tree is constructed where nodes are labeled with formulae.[2]

In order to test whether a certain formula φ follows from a set of premises Θ a tableau tree is constructed for $\Theta \cup \{\neg\varphi\}$. Constructing such a binary tree can be seen as checking for (in)consistency of the theory. It is build using reduction rules which determine how the tree is branched. If in any branch, a formula and its negation appear the branch is said to be closed. If all branches close the theory is inconsistent.

$$\frac{\neg\neg \mathrm{Z}}{\mathrm{Z}} \quad \frac{\alpha}{\alpha^1 \\ \alpha^2} \quad \frac{\beta}{\beta^1 \mid \beta^2}$$

Table 1. Rules for the tableau trees

Table 1 shows the rules for constructing the tableau tree. The first rule indicates that double negations are redundant. All propositional formulas containing binary connectives can be divided as belonging to two types: True conjunctive formulas (α-type) and true disjunctive formulas (β-type). For example $p \rightarrow q$ can be rewritten to $\neg p \vee q$ and so is a β-formula. The rule for α-type formulas indicates that the conjuncts have to be placed on the same branch of the tree. The β-rule however indicates a branching of the tree.

To see how this works we give an example. Let Θ be $\{p \rightarrow q\}$ and $\phi = q$. We will check whether $\Theta \models \varphi$. The resulting tableau is depicted in Fig. 1. Since the tableau is open φ does not follow from Θ. Note that the open branch corresponds to a counterexample for $\Theta \models \varphi$.

[2] We limit ourselves here to the description of propositional tableaux since this seems sufficient for the description of classification. We follow [5] in this description.

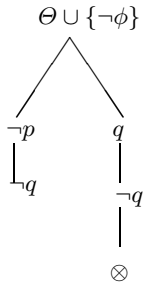

Fig. 1. Example tableau

4 Tableaux for classification

The general method to use propositional semantic tableaux as a proof procedure for classification proceeds as follows. The solution criterion for the classification task is translated into a consistency formula. Subsequently that formula is expanded into a semantic tableau. If the tableau can be closed or remains partially open, a conclusion can be derived about the solutions of the classification problem. The precise nature of the conclusion depends on the nature of the solution criterion.

The closing rule for tableaux for classification can be specialized on the basis of the ontological commitments described in Sec. 2. Since all features that occur in the domain theory are considered to be typed propositions and since two distinct propositions of the same type exclude each other, any path that contains two different propositions of the same type can be closed. This is equivalent to the addition of formulae of the following type to the domain theory for each feature:

$$a_1 \rightarrow (\neg a_2 \wedge \ldots \wedge \neg a_n)$$

Building this ontological commitment into the proof procedure retains all properties of the general procedure such as soundness and completeness, but is more efficient.

4.1 Weak classification

In weak classification we assume that the domain theory and the observations together are consistent. This assumption is implicit in the way knowledge is represented and can be viewed as an additional ontological commitment.

Because of this consistency a semantic tableau for $DT \cup Obs$ will have open branches. Fig. 2 shows a tableau for the domain theory $\{c_1 \rightarrow a_1 \wedge d_2, c_2 \rightarrow a_2 \wedge d_3\}$ and the observation d_3. The observation is added to the leaves of the tree for the domain theory.

Now, in order to check which classes are consistent with the observations made up to this point, each class must be individually added to the tableau. If the tree closes the resulting theory is inconsistent and the class is not a candidate. If a new observation is

made it must be added to the tree and a check for all remaining candidates has to be made again. Fig. 2 shows class c_1 to be inconsistent with the domain theory together with the observation d_3, as its addition to the tableau would close the tree.

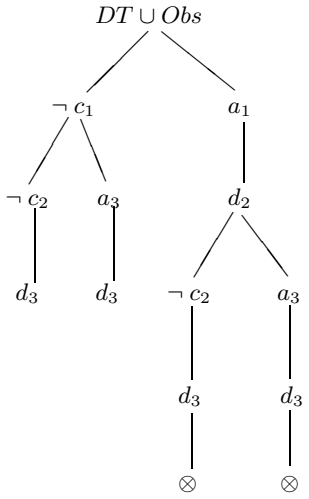

Fig. 2. Initial tableau for weak classification

The trouble with this representation is that it leads to a rapidly expanding tree. Notice that since class definitions are $\beta-$formulae each class definition leads to branching of the tree. In classification, observations are typically added incrementally. This means that classes that at some point are proven to be inconsistent will remain part of the tree. This can be be remedied by putting the observations at the root, instead of adding them to the leaves of the tableau for the domain theory. The result is a much smaller tree. Subsequent observations can also be added to the root instead of the leaves. This proof strategy is depicted in Fig. 3. The example shows a general property of tableau proofs: the length of a proof typically depends on the order of application of the expansion rules.

This procedure for weak classification with semantic tableaux can be summarized as follows:

```
Procedure WC-1:
  1. Construct a tableau for the observations
     and the domain theory (in this order)
  2. FOR each possible candidate class c DO
       IF c (and c alone) is added to the tableau
          AND the tableau closes
       THEN c is not a possible candidate
       ELSE c remains a possible candidate
  3. When new observations are made:
```

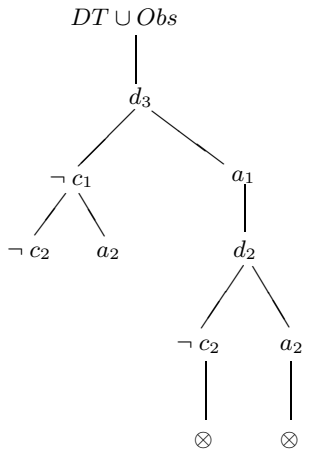

Fig. 3. Alternative tableau for weak classification

```
a. add observation(s) to top of the tableau
b. redo step 2
```

It is interesting to view weak classification in terms of what is actually deduced during the process. In order to prove that a formula follows from a theory with the help of semantic tableaux one has to prove the inconsistency of the negation of the formula together with the theory. In the case of weak classification the aim is to prove the inconsistency of a class together with the domain theory and observations. In terms of consequence this means proving the negation of a class from the domain theory and observations. More formally:

$$DT \cup Obs \cup \{c\} \vdash \bot \Leftrightarrow DT \cup Obs \models \neg c$$

In other words: weak classification only provides negative information about classes. The method is actually more about ruling out candidates than looking for candidates which can explain the observations. This is left to the stronger criterion of strong classification.

The procedure WC-1 is still inefficient since it generates the tableau for the entire domain theory. The ontological commitments about the structure of the knowledge base, i.e. that class definitions are conjunctions of disjunctive feature sets, allow us to specialize the procedure even more. A branch in the tableau tree that contains no literals involving classes, can never be closed by adding other parts of the domain theory that do not concern the class under investigation. This leads to the following procedure:

```
Procedure WC-2:
  1. Construct a tableau for the observations
  2. FOR each possible candidate class c DO
      2a. add the formulae that concern c from
          DT to the tableaux
```

```
    2b. IF c (and c alone) is added to tableau
           AND the tableau closes
        THEN c is not a possible candidate
        ELSE c remains a possible candidate
3. When new observations are made:
   a. add observation(s) to top of the tableau
   b. redo step 2 for those classes that were
      possible candidates
```

This procedure is much more efficient than WC-1, but given the ontological commitments still sound and complete. In fact, this proof procedure explains why the "test" part in implemented problem solving methods can remain so simple. It is also clear from this procedure that generate-and-test methods that do not test for all classes are sound, but not complete. In set 3b we make use of the fact that addition of new observations will not make classes that have been ruled out on the basis of earlier observations, candidates again.

4.2 Strong classification

We now proceed to describe a procedure for testing whether a candidate class fulfills the SC requirement. If it can be shown for a certain class c that $DT \cup \{c\} \cup \{\neg o_1 \vee \ldots \neg o_n\}$ is inconsistent, the criterion of SC is met with respect to c. In order to show this, first a tableau for DT and the disjunction of negated observations is build. Since this theory is consistent the tree will have open branches. If c is added to the tree and closes the tableau, c is a SC candidate, otherwise it is not. Here we make use of the ontological assumption that $DT \cup \{c\}$ is consistent. An example is given in Fig. 4.

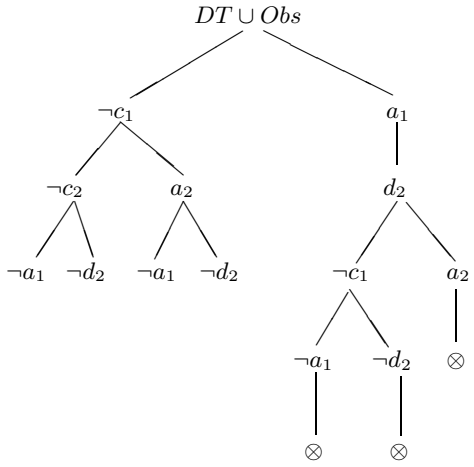

Fig. 4. Tableau for strong classification

This procedure for SC can be summarized as follows:

```
Procedure SC-1:
  1. Construct a tableau for the domain theory
  2. Add the disjunction of the negation of each
     element of Obs to the leaves of the tableau
     for the domain theory
  3. FOR each possible candidate class c DO
       IF c (and c alone) is added to the tableau
          AND the tableau closes
       THEN c remains a possible candidate
       ELSE c is not a possible candidate
  4. IF new observations become available
     THEN redo step 2 and 3
```

SC is in general more complex to compute than WC. To infer the observations from the domain theory and a candidate class , the disjunction of all negated observations should be added to the tableau. In contrast to WC, each observation will give rise to a branching of the tree.

The same specializations of the proof procedure that we have described for WC apply to SC.

4.3 Classification and abduction

The procedure SC-1 is identical to the way abduction is performed with tableaux as described in [1]. This is intentional. As defined above SC comes down to an abductive method. Abduction is often linked to a style of reasoning which produces causal explanations for observations. Classification is rarely, if ever, considered to be an abductive task. Still the criterion of SC is formally in line with those of abduction, as for example put forward by [7].

In [1] five different styles of abductive reasoning are defined. Given Θ (a theory) and φ (a sentence), α is an abductive explanation (abducible) if:

- Plain: $\Theta, \alpha \models \varphi$
- Consistent: $\Theta, \alpha \models \varphi$ and Θ, α is consistent.
- Explanatory: $\Theta, \alpha \models \varphi$ and $\Theta \not\models \varphi$ and $\alpha \not\models \varphi$
- Minimal: $\Theta, \alpha \models \varphi$ and α is the weakest such explanation.
- Preferential: $\Theta, \alpha \models \varphi$ and α is the best explanation according to some given preferential ordering.

Interestingly enough, SC displays four of these properties of abduction. The "'plain" property follows directly from the solution criterium. Classification is a restricted form of abduction: the only abducibles allowed are atomic class propositions. Since it is an ontological assumption that class definitions are individually consistent with the domain theory (i.e., $DT \cup c$ is consistent), it follows that solutions found by the SC-1 method are consistent. SC is minimal since we restrict the form of abducibles to single classes (atomic propositions) only.

Weak classification does not exhibit any of the properties of abduction, since no formula (φ) is assumed to be entailed by the theory (Θ) and the abducibles (α). Intuitively, WC generates a formula and tests if it is consistent with the current domain theory and observables, but it does not try to explain anything. This makes WC a very different task from SC. This is in line with our earlier observation that WC is a ruling out task rather than an explanation task. From a logical point of view, one could argue that the two forms of classification are rather different ways of reasoning, even though procedurally they are very similar. One could even go as far as considering WC not as a classification task, but as a refutation task.

For the property of minimality, there is a difference between abduction and strong classification. If we decide to allow not only single classes as abducibles but conjunctions of classes as well we end up with a different style of classification. In that case we would get a form of classification in which more than one class explains the observations and counts as a solution. Thus, a conjunction of classes can act as a solution candidate. This is known as composite-solution classification (CSC) [9].

The solution predicate can be formulated as follows:

$$S = \{c_1 \wedge ... \wedge c_n | DT \cup \{c_1 \wedge ... \wedge c_n\} \models Obs\}$$

Note however that we can no longer assume the property of consistency ($DT \cup \{c_1 \wedge ... \wedge c_n\}$), as the simultaneous addition of two or more classes (e.g., a black bird and a white bird) to the tableau of the domain theory may lead to inconsistency. Therefore, the procedure will have to test explicitly for consistency. For example, if $DT = \{c_1 \rightarrow a_1, c_2 \rightarrow a_2\}$, adding the composite solution $\{c_1 \wedge c_2\}$ makes the theory inconsistent, since a_1 and a_2 are regarded as exclusive.

If one allows composite solutions, one could still prefer single solutions. In this case CSC is defined as preferential abduction in the above sense.

5 Discussion

Even though classification is one of the simplest knowledge-intensive tasks in the knowledge-engineering domain, it has been quite hard to prove that certain computational methods satisfy logical competence theories. Similarly, it has been difficult to transform logical competence requirements into an operational method. Problems encountered include: the abductive nature of classificatory reasoning, the incremental nature of observation gathering and the mapping of logical theories onto different computational strategies in classification methods [11]. In this paper we have presented some steps forward towards solving these problems.

A first insight is that strong classification is a special case of abduction. The solution of a classification problem is considered as an abducible of some domain theory and the observed facts. Theories of abduction provide several types of abduction that can be mapped onto different variants of the classification task. However, classification is more specific than abduction in the sense that it restricts the vocabulary of abducibles to a predefined set of classes and that it assumes a particular structure of the domain theory.

Second, it appears that the semantic tableau proof method has some features that makes it suitable to model various forms of classificatory reasoning. Tableaux provide a natural way of handling incrementally growing theories as they often occur in knowledge-based systems, where new facts are incrementally obtained from a user. In classical logical approaches that attempt to formalize classification reasoning, this problem is not easily solved [11]. Tableaux also provide a way of thinking about the search space of possible inferences in a formal context. As we have shown, the ontological commitments of the task restrict the possible expansions and closures of the tableaux. These restrictions can be translated into the proof procedure itself, thus reducing the space of formulae to be processed. This is precisely where knowledge-based systems derive their power from, when compared to general theorem-proving approaches. It can be proven that the specialized proof procedures are equivalent to the normal proof procedure when the ontological commitments are added as axioms to the domain theory.

The third result of our investigation is that PSMs for classification that have been published [9, 11] can be mapped onto proof procedures. For example, procedure WC-1 and WC-2 formalize the "pruning" method, which can therefore be characterized as both sound and complete. WC-1 is a "pure" logical method, but computationally not very efficient. In tableau terminology it generates a much larger tableau than WC-2. WC-2 is an optimized method which is in fact close to operational methods used in classification systems. Procedure SC formalizes a generate-and-test method for classification. Here, we can see from the formalization that if the method would terminate after having found one solution (which is often the case in operational methods), the method is sound, but not complete. In short, this type of mapping of PSMs used in knowledge-based systems onto specific tableau proof procedures provides a powerful way of establishing the competence of these methods in logical terms.

References

[1] A. Aliseda-LLera. *Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence*. PhD thesis, ILLC, University of Amsterdam, Amsterdam, The Netherlands, 1997.

[2] V. R. Benjamins and D. Fensel. Problem-solving methods. *Int. J. Human-Computer Studies*, 49(4):305–313, 1998. Editorial special issue.

[3] E.W. Beth. Semantic entailment and formal derivability. In J. Hintikka, editor, *The Philisophy of Mathematics*, pages 9–41. Oxford University Press, 1969.

[4] D. Fensel and R. Groenboom. Specifying knowledge-based systems with reusable components. In *Proceedings of SEKE-97*, 1997.

[5] M. C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, New York, 1990.

[6] T. R. Gruber, G. R. Olsen, and J. Runkel. The configuration-design ontologies and the VT elevator domain theory. *Int. J. Human-Computer Studies*, 44(3/4):569–598, 1996.

[7] A. C. Kakas, R. A. Kowalski, and M. Toni. Abductive logic programing. *Journal of Logic and Computation*, 2(6):719–770, 1993.

[8] A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, N. R. Shadbolt, W. Van de Velde, and B. J. Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, Cambrdige, MA, 1999.

[9] M. Stefik. *Introduction to Knowledge Systems*. Los Altos, CA. Morgan Kaufmann, 1993.

[10] F. van Harmelen and A. ten Teije. Characterising problem solving methods by gradual requirements. In *Proceedings of the Eleventh Workshop on Knowledge Acquisition for Knowledge-Based Systems (KAW'98)*, Banff, Alberta, 1998.

[11] B. J. Wielinga, J. M. Akkermans, and A. Th. Schreiber. A competence theory approach to problem-solving method construction. *Int. J. Human-Computer Studies*, 49:315–338, 1998.