# The AQ21 Natural Induction Program for Pattern Discovery:
## Initial Version and its Novel Features

J. Wojtusiak, R.S. Michalski*, K.A. Kaufman, and J. Pietrzykowski
George Mason University
*{jwojt, michalski, kaufman, jarek}@mli.gmu.edu*

*Also with the Institute of Computer Science, Polish Academy of Sciences

## Abstract

*The AQ21 program aims to perform natural induction, a process of generating inductive hypotheses in human-oriented forms that are easy to interpret and understand. This is achieved by employing a highly expressive representation language, Attributional Calculus, whose statements resemble natural language descriptions. This paper focuses on the Pattern Discovery mode of AQ21, which produces attributional rules that capture strong regularities in the data, but may not be fully consistent or complete with regard to the training data. AQ21 integrates several novel features, such as optimizing patterns according to multiple criteria, learning attributional rules with exceptions, generating optimized sets of alternative hypotheses, and handling data with unknown, irrelevant and/or non-applicable meta-values.*

## 1. Introduction

One of the limitations of current machine learning and data mining programs is that they employ relatively simple pattern representation languages, e.g., decision trees, Bayesian nets, neural nets, etc., which limit the range of patterns they can discover. As a result, such programs may not be able to discover patterns that are cognitively simple but not easily representable by the program. This paper briefly describes a methodology for *natural induction* that seeks patterns represented as rules in Attributional Calculus ($\mathcal{AC}$) that are more expressive than rules typically used in machine learning or data mining programs (e.g.[1],[2], [7]).

This methodology is being implemented in the AQ21 program. Due to space limitations, we focus primarily on the AQ21's *Pattern Discovery mode* that seeks approximate regularities in datasets likely infested by errors. AQ21 has also two other modes of operation: *Theory Formation* (TF) that generates consistent and complete (CC) data generalizations, and *Approximate Theory Formation* (ATF) that optimizes the CC description to maximize a criterion of *description utility* that pits errors on the training dataset against description simplicity. While TF is oriented toward error-free data, ATF is most useful when a small amount of errors may be present or the user seeks only approximate, but a simple data generalization.

Other important AQ21 features are that it can discover different types of regularities in data, such as conjunctive patterns, general rules with exceptions, gives a choice to the user to induce rulesets for *parallel* or *sequential* execution, and can generate an optimized collection of alternative hypotheses from the same data. AQ21 can also handle various types of meta-values in data, such as unknown, irrelevant and/or non-applicable. Because it is not possible to describe precisely algorithms for implementing these features in one short paper, we provide here only the central idea and brief description of them. For more details, see e.g. [8], and relevant papers downloadable from http://www.mli.gmu.edu/mpubs.html.

## 2. AQ Learning

The general learning problem considered here is to generate general hypotheses H1, .., Hk about classes C1,…,Ck, respectively, on the basis of a set of training examples, e1, .., en, drawn from these classes. The AQ learning methodology generates hypotheses in the form of *attributional rulesets* that optimize a given multi-criterion measure of *hypothesis utility*. An attributional ruleset is a set of attributional rules describing the same class [4].

The basic form of an attributional rule is:

CONSEQUENT <= PREMISE

where both CONSEQUENT and PREMISE are conjunctions of *attributional conditions* in the form:

[L rel R: A]

where L is an attribute, an internal conjunction or disjunction of attributes, a *compound attribute*, or a *counting attribute*; rel is one of =, :, >, <, ≤, ≥, or ≠, and R is an attribute value, an internal disjunction of attribute values, an attribute, or an internal conjunction of values of attributes that are constituents of a compound attribute, and A is an optional annotation that lists statistical information about the condition (e.g., $p_c$ and $n_c$ condition coverages, defined as the numbers of positive and negative examples, respectively, that satisfy the condition). Here is an example of a simple attributional rule (without annotations):

[activity=running_experiments]
<= [day = weekend] & [clock_speed >= 2GHz] &
[location = lab1 v lab3] & [weather: quiet & warm]

which can be paraphrased: the activity is "running experiments" if it is weekend (a higher-level value of the *structured* attribute "day"), the computer clock_speed is at least 2 GHz, the experiment takes place in lab1 or lab3, and the weather is quiet & warm. The attribute "weather" is an

example of a *compound attribute*, a new type of attribute introduced in AQ learning that takes a conjunction of values [4]. Note that the rule closely corresponds to its equivalent natural language interpretation. A counting attribute is not illustrated here, because it is a more elaborate concept, but a detailed description of such attributes is in [4].

As shown here, an attributional rule in AQ learning uses a richer representation language than in typical rule learning programs, in which conditions are usually limited to a simple form [<attribute> <relation> <attribute_value>].

## 2.1 Pattern Discovery Mode

In PD mode, the program searches for strong patterns that maximize an assumed pattern quality measure. The method takes as input a set of positive examples (Pos), a set of negative examples (Neg), and a multi-criterion pattern quality measure, defined by the user using LEF ("Lexicographic Evaluation Functional", see, e.g. [3]). It follows the general algorithm presented in Figure 1.

```
Hypothesis = null
While Pos is not empty
  Select a seed s from Pos
    Generate approximate star G(s, Neg)
    Select from G the best k rules according
      to LEF, and include them in Hypothesis
    Remove from Pos all positive examples
      covered by the selected rules
Optimize final rules according to LEF
Select the final hypothesis
```

**Figure 1.** The simplest form of PD mode in AQ21.

The algorithm starts by focusing attention on one positive example of a concept, called the *seed*, and then generates a *star*, defined as set of alternative patterns (generalizations) of the seed that maximize LEF. In PD mode, the default LEF is to maximize the *pattern quality*, then its coverage, and then to minimize the pattern length (number of conditions). The pattern quality is defined as:

$$Q(w) = cov^{w} * config^{1-w}$$

where cov = p/P is the *relative coverage of the pattern* (a.k.a *support)*, P and N are numbers of the positive and negative examples in Pos and Neg, config, *confidence gain*, is ((p / (p + n)) – (P /(P + N))) * (P +N) /N, and w is a parameter controlling the relative importance of relative coverage and confidence gain [5].

Config is 0 if the *pattern confidence*, p/(p+n), is equal the confidence of random guessing, and reaches 1 if the pattern is fully consistent with the data (n=0). The highest scoring patterns are those that represent a desired tradeoff between the coverage and confidence gain, as defined by w.

A method for generating a star has been described in various past publications, e.g., [3]. It uses a beam search to protect the process from a combinatorial explosion.

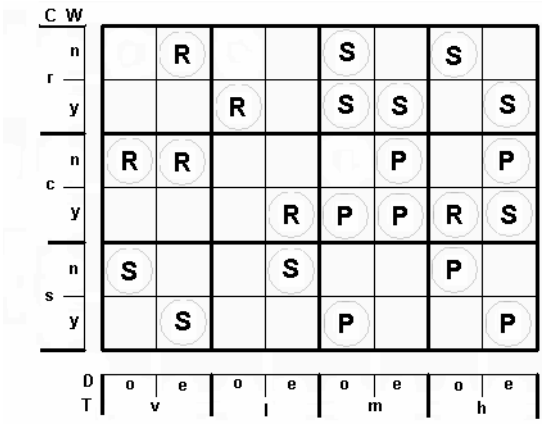
Several new features of AQ21 are designed to improve its efficiency and pattern quality: the use of several seeds for parallel star generation, selection of negative examples based on their distributions, selection of the most relevant attributes prior to star generation, and optimized discretization of continuous attributes. A detailed description of the features is beyond the scope of this paper, and will be described in a separate paper.

# 3. A Simple Problem to Illustrate AQ21

To illustrate AQ21's basic capabilities, we use a very simple designed problem in which 22 training examples are defined in terms of one output (*activity*) and 4 input attributes (Figure 2). The space spanned over the input attributes is illustrated by *Generalized Logic Diagram* (GLD) in Figure 3. Letters P (for "play"), R (for "read"), and S (for "shop) mark cells that correspond to the examples for these activities. The task is to determine patterns in examples of "play".

```
condition   linear  {rainy, cloudy, sunny}
wind        nominal {no, yes}
temperature linear  {very_low, low, medium, high}
daytype     nominal {workday, weekend}
activity    nominal {play, shop, read}
```

**Figure 2.** Names, types and domain of the attributes.




r – rainy, c – cloudy, s – sunny, n – no, y – yes, v – very low, l – low m – medium, h – high, o – workday, e – weekend, P – play, R – read, S - shop

**Figure 3. A** GLD with the example problem.

AQ21 applied with default parameters to the training data discovered a pattern (single rule) shown in Figure 4.

```
[activity=play]
   <= [condition=cloudy v sunny: 7,8] &
      [temperature=medium v high: 7,7]:
       p=7,n=2,q=0.67
```

**Figure 4.** A pattern discovered by AQ21.

The rule states that the activity is play when the weather condition is cloudy or sunny and the temperature is medium or high. The numbers in the conditions are parameters $p_c$ and $n_c$.

## 4. Learning Rules with Exceptions

The concept of an exception is commonly used by people when describing rarely occurring anomalies in described phenomena. It is not unusual that a simple theory may work well for most cases, but turning it into a fully consistent and complete theory would require making it significantly more complex. In such cases, it is desirable to learn rules with exceptions [4]. AQ21 can be set to learn rules with exceptions (a.k.a censored rules) in the form:

CONSEQUENT <= PREMISE |_ EXCEPTION

where EXCEPTION is either an attributional conjunctive description or a list of examples constituting exceptions to the rule. Note that exceptions in such rules are always negative examples. Learning of censored rules in PD mode involves creating patterns as before, and then determining descriptions of the exceptions from the patterns (covered negative examples). The descriptions are expressed as conjunctions of attributional conditions, determined by applying AQ learning again, but only to examples covered by the pattern. If all of the exceptions can be characterized by one conjunctive description, this description is used as the EXCEPTION clause in the rule; otherwise, an explicit list of exceptions is generated.

Applying AQ21 to the same problem as before produced the censored rule presented in Figure 5

```
[activity=play]
<= [condition=cloudy v sunny: 7,8] &
   [temp= medium v high: 7,7]
  |_ [condition=cloudy]&[wind=yes]&[temp= high]
   : p=7,n=0,q=1
```

**Figure 5.** A strong pattern with exception found by AQ21.

The rule states that activity is play if weather is cloudy or sunny and temperature is medium or high, unless weather is cloudy, there is wind, and temperature is high.

In creating this rule, AQ21 generalized the two negative examples covered by the previous rule (Figure 4) into one conjunctive description, and presented it as the EXCEPTION clause. If the two examples could not be generalized into one conjunctive description, the program would have listed them explicitly.

## 5. Handling Meta-values in Data

Another important new feature of AQ21 is its ability to learn from examples that may have attribute meta-values, which can be of three types: *unknown*, *irrelevant*, and *not-applicable*. These meta-values correspond to three possible answers to a question requesting an attribute value in situations in which a regular value cannot be provided [6].

*Unknown* (a.k.a. "Don't know"), denoted by a "?" in the training dataset, is given to an attribute whose value for a given entity exists, but is not available in the given data base for some reason. For example, the attribute may not have been measured for this entity, or may have been measured, but not recorded in the database.

Two internal methods for handling Unknown values are implemented in AQ21: (L1), which ignores the extension-against operation (a basic generalization operation in AQ [4]) for attributes with missing values, and (L2), which treats "?" as a regular value in the examples, but avoids examples with a "?" when selecting seeds. When extending a seed against a missing value, it creates a condition:

[xi $\neq$ ?], regardless of the value of attribute $x_i$ in the seed.

*Not-applicable*, denoted by an "NA," is given to an attribute that does not apply to a given entity, because its value does not exist. For example, in a library inventory database, the "number of pages" attribute does not apply to a chair in the library, but it applies to the books. If a dataset has "not-applicable" values for some attributes, AQ21 removes all such attributes from all examples with that value when executing the extension-against operator, but does not remove them from examples that do not have that value regardless of whether they are positive or negative events.

*Irrelevant* values, denoted by an "*", indicate that values exist, but an attribute is considered irrelevant to the learning problem, to the concept (class) to be learned, or to the particular event. An attribute is *task-irrelevant* if it is irrelevant for the entire learning problem. For example, a student's hair color can be declared as irrelevant for learning rules for classifying students into groups representing their academic performance. An attribute is *class-irrelevant* if it is irrelevant for a given class (given value of the output attribute), but relevant for other classes. For example, the patient's Prostate Specific Antigen (PSA) level is relevant for diagnosing prostate diseases, but is irrelevant for diagnosing eye diseases. These two types of irrelevance are handled during preprocessing. An attribute is *event-irrelevant* if it is irrelevant only for a particular example in the class to be learned.

A detailed description of the methods for reasoning with meta-values in the learning and testing phases and results from their experimental investigation are presented in [6].

## 6. Alternative Hypotheses

From non-trivial sets of concept examples, it is usually possible to generate many alternative generalizations of these examples. Having alternative hypotheses can be useful for a variety of practical applications. For example, in medical decision making, when some tests required by a given diagnostic procedure cannot be performed (e.g. because the equipment is unavailable), one would like to know an alternative procedure that would not require

measuring these test. Alternative hypotheses can also be used to increase the accuracy of classification decisions by simple voting on decisions assigned by different hypotheses, or by weighted voting, as is done in boosting. AQ21 can not only determine alternative hypotheses, but seeks those that as a collection optimize a user-defined multi-criterion measure of the collection quality. For illustration, Figure 6 presents alternative rulesets obtained by AQ21 applied to the dataset in Figure 3 (running in Theory Formation mode).

```
[activity=play]
<= [condition=cloudy v sunny: 7,8] &
   [temperature=medium: 4,3] : p=4,n=0
<= [condition=sunny: 3,3] &
   [temperature=medium v high: 7,7] : p=3,n=0
<= [condition=cloudy v sunny: 7,8]&[wind=no:3,7]&
   [temperature=medium v high: 7,7] : p=3,n=0

[activity=play]
<= [condition=cloudy v sunny: 7,8] &
   [temperature=medium: 4,3] : p=4,n=0
<= [condition=sunny: 3,3] &
   [temperature=medium v high: 7,7] : p=3,n=0
<= [condition=cloudy v sunny: 7,8]&[wind=no: 3,7]
   [temperature=medium v high: 7,7]  : p=3,n=0
```

**Figure 6.** Two alternative rulesets learned by AQ21.

Both rulesets are complete and consistent with regard to the training data. They differ in the way the last example is covered by the third rule.

## 7. Selected Experimental Results

This section presents selected results of application of the AQ21, C4.5 [7], CN2 [1], and RIPPER [2] programs to two real-world datasets (Volcanoes and World Factbook) and one designed dataset. Results are presented in Table 1.

**Table 1.** Accuracies obtained by five methods.

| Dataset | Accuracy % / Number of Rules or Nodes | | | | |
|---|---|---|---|---|---|
| | AQ21 | C4.5 | C4.5R | CN2 | RIPPER |
| Volcanoes | 99.45/ 12 | 98.8 / 120 | 99 / 65 | 95.5 / 190 | 98.96 / 33 |
| WorldFact | 94.29 / 2 | 91.9 / 6 | 91.9 / 5 | 93.5 / 7 | 93.55 / 2 |
| Designed | 100 / 2 | 92.1 / 83 | 92.7 / 46 | 91.7 / 164 | 92.8 / 8 |

The *Volcanoes* dataset, provided by the Smithsonian institution, contains information about 20,000 volcanoes and their eruptions. The learning goal was to distinguish eruptions that caused fatalities from those that did not. The *World Factbook* dataset, downloaded from the CIA website, contains information about 266 countries around the world (2 classes of countries -- those with birth rates above and below 20 per 1000 population). The designed dataset consists of 12 binary, 1 structured, and 1 nominal (6 values) input attributes describing 1000 training and 1000 testing examples (500 positive and 500 negative examples in each set) of the concept "the number of present Features is 2 or 3 and Shape is oval or Color is red."

Table 1 shows that AQ21 performed on all 3 problems better than other methods in terms of both predictive accuracy and simplicity (number of rules), except for one case: predictive accuracy on Volcanoes data, 99.5% by CN2 vs. 99.45% by AQ21. In this case, however, CN2 required 190 rules, while AQ21 required only 12.

## 8. Conclusion

AQ21 was presented as a natural induction program that equally stresses the accuracy and the simplicity of discovered knowledge. The paper focused on AQ21's pattern discovery mode, and briefly described its several novel features, such as discovering attributional patterns with or without exceptions, discovering optimized collections of alternative hypotheses for the same data, and handling meta-values. AQ21 has also several new features, not described here, that improve its efficiency. To the authors' best knowledge, AQ21 is the only learning program at present that integrates so many features.

Although presented results were obtained from applying AQ21 to a small sample of problems, the program has been successfully applied to many other problems, some of them of much greater complexity (involving hundreds of attributes and/or hundreds of thousands of examples).

## 9. Acknowledgements

This research has been supported in part by the National Science Foundation Grants No. IIS 9906858 and IIS 0097476. The findings and opinions expressed here are those of the authors, and do not necessarily reflect those of the above sponsoring organizations.

## 10. References

[1] Clark, P. and Niblett, T., "The CN2 Induction Algorithm", *Machine Learning*, 3, 1989, pp. 261-289.

[2] Cohen, W., "Fast Effective Rule Induction", *Proceedings of the 12th International Conference on Machine Learning*, 1995.

[3] Michalski, R.S., "Theory and Methodology of Inductive Learning", In R.S. Michalski, J. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Palo Alto: Tioga Publishing Co., 1983.

[4] Michalski, R.S. "ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction", *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, 2004.

[5] Michalski, R.S. and Kaufman, K., "Learning Patterns in Noisy Data: The AQ Approach", In G. Paliouras, V. Karkaletsis and C. Spyropoulos, (Eds.) *Machine Learning and its Applications*, Springer-Verlag, 2001, pp. 22-38.

[6] Michalski, R.S. and Wojtusiak, J., "Reasoning with Meta-values in AQ Learning", *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, 2004.

[7] Quinlan, J.R., *C4.5 Systems for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[8] Wojtusiak, J., Michalski R.S., Kaufman K.A. and Pietrzykowski J., "Multitype Pattern Discovery Via AQ21: A Brief Description of the Method and Its Novel Features", *Reports of the Machine Learning and Inference Laboratory*, MLI 06-2, George Mason University, June, 2006.