Rule Induction for Knowledge Acquisition in RKF

Thomas Russ

August 2, 2003

1 Introduction

The RKF Year 2 Challenge problem highlighted the difficulty that Subject Matter Experts (SMEs) have in writing inference rules. During the evaluation period, the KRAKEN SMEs entered about five to seven rules each [Pool *et al.*, 2003]. One solution to this problem is to provide support for learning inference rules from examples rather than requiring domain experts to write the rules themselves.

We believe that this will provide another tool to accelerate the knowledge entry and knowledge base development. Learning from examples can help solve this particular problem because it is easier to describe examples than to write rules. There may also be independent sources of examples, perhaps in databases that don't support the expression of rules. Those examples could be imported and used as the input data to a rule induction algorithm.

This report describes the refinement and extension of a rule induction algorithm [Moriarty, 2000] originally implemented in PowerLoom™ [PowerLoom] as part of the High Performance Knowledge Base (HPKB) project.

2 Learning Background

Since both Cyc® and PowerLoom are logic-based representation languages, the area of learning research most closely related is Inductive Logic Programming (ILP) [Muggleton and de Raedt, 1994]. In ILP, learning algorithms make use of ground facts and background knowledge in order to discover, or *induce* inference rules.

This differs from data mining techniques such as association rule learning [Hipp *et al.*, 2000]. First off, the representation used is very different. Typical association rule learning uses feature vectors which are uniformly applied to the domain in which learning occurs. In our domain, we have a much more highly structured representation which is best described as a graph rather than a vector.

```
Start with a rule with an empty antecedent
REPEAT
  For each potential clause (predicate and variable combination)
    Evaluate current rule with new predicate using a gain metric
  If predicate with highest evaluation improves rule
    Add predicate to antecedent
UNTIL no further improvement is possible
```

Figure 1: Top-down Rule Inference

Second, there is a lot more information available about the relationships between items in the data set. Instead of trying to learn about a homogeneous class of items (for example, learning about clusters of transactions) we will be learning over a heterogeneous set of objects. The system will also know a lot more about the objects.

These two differences make ILP-style approaches more suitable. Even so, there are some ways in which our domain differs from the traditional ILP learning work.

First, both Cyc and PowerLoom use open-world semantics. That means that it is can be difficult to find true negative examples, since the presence of unknown information doesn't allow proof of negation by failure to prove the positive. Some ILP systems such as FOIL [Quinlan and Cameron-Jones, 1993] also address open world semantics.

Second, Cyc and PowerLoom knowledge bases are characterized by hierarchical types (concepts, collections). One of the new contributions of this work is the extension of the induction algorithms to recognize and exploit the type hierarchy.

3 Algorithm Development

This section provides an overview of the algorithm development process, proceeding from a general background used in traditional ILP work and then focusing on the specifics of the algorithms developed for RKF. Common to all of these learning approaches is the notion of sets of *positive examples*, that is, examples which match what we are trying to learn and *negative examples*, those which do not match.

3.1 General Approaches

ILP provides two general approaches to learning inference rules from data, top down and bottom up. In the top down approach, one starts with a very general rule and then specialize it. The initial general rule typically covers all examples, both positive and negative. Clauses are added to the antecedent in order to exclude negative examples while retaining the positive examples. An outline of the algorithm is shown in Figure 1. Quinlan's FOIL system [Quinlan and Cameron-Jones, 1993] is a well-known example of this approach.

A bottom-up approach works in the opposite manner. Instead of starting with a general rule, a very specific rule is used — one which fits a single postive instance exactly. By

```
rule = NULL
REPEAT
  DO 20 times
    if rule = NULL
      Choose two positive examples randomly: example1, example2
      lgg = least_general_generalization(example1, example2)
    else
      Choose one positive example randonly: example1
      lgg = least_general_generalization(rule, example1)
    If lgg does not cover any negatives
      If lgg covers more uncovered positives than any other lgg
        rule = lgg
   Remove covered positive examples from training examples
UNTIL lgg's provide no further improvement
RETURN rule
```

Figure 2: Bottom-up Rule Inference

comparison with other positive instances, the rule is generalized as little as possible so that additional positive instances are admitted but no negative instances. The algorithm is shown in Figure 2. GOLEM [Muggleton and Feng, 1990] and PROGOL [Muggleton, 1995] are examples of ILP systems that use the bottom-up approach.

In a very large knowledge base, one in which there are very many potential predicates that can be explored, the top-down approach is likely to exhibit very poor scalability. There are some techniques used in PowerLoom to amerliorate this, but it is a concern. Our initial testing for RKF confirmed this problem and led us to focus our efforts on the bottom-up approach.

3.2 Bottom-Up Algorithm

The bottom-up approach begins with a very specific rule that exactly covers a single example. This rule is successively expanded by pairwise generalization of the rle to cover additional positive examples while excluding negative examples. In the strict mode of inference, all negative examples must be excluded. A relaxed version of this substitutes a false-positive threshold which will allow a rule to erroneously cover a specified percentage of negative examples.

The approach we take in the bottom up algorithm is largely based on syntactic manipulation of rule clauses and ground assertions. (A ground assertion is a logical clause with no variables, only constants). The induction step involves computing generalization using a largely syntactic method of computing subsumption between pairs of clauses. The approach is characterized as largely syntactic because only object and structural identity are used in the construction of generalizations. Mismatches are generalized through the substitution of variables for the non-identical terms. There are two exceptions to the purely syntactic generalization. The two types of predicates that are treated specially with respect

```
FOREACH positive instance "i" DO
  Add the ground assertions about "i" to Facts(i).
  Augment numeric function assertions with inequality assertions.
  FOREACH new instance "j" mentioned in the ground assertions
     Add the new ground assertions about "j" to Facts(i)
     Augment numeric function assertions with inequality assertions.
     Recurse on new instances mentioned in ground assertions
     UNTIL the maximum structure depth is reached.
```

Figure 3: Instance and Fact Generation for Bottom-Up Induction

to generalization are type restrictions ("isa" predicates) and numeric inequalities. In these two cases, a semantic generalization is performed.

For isa predicates, the semantics derive from the existing type hierarchy encoded in a PowerLoom or Cyc knowledge base. This allows generalization to take advantage of the existing knowledge structure to drive the generalization reasoning. Given the facts `(isa John MalePerson)` and `(isa Jane FemalePerson)` a purely syntactic generalization would only be able to generalize to `(isa ?x1 ?x2)` using two variables. This is, of course, a nearly useless clause for a rule antecedent, since (unless there is some other constraint on `?x2`, everything in the knowledge base will satisfy it. Exploiting the existing type hierarchy our generalization algorithm can create the generalization `(isa ?x1 Person)` instead. This is a much more useful generalization, since it retains more discriminating power.

The other case where semantics is considered is in the handling of numeric inequalities with numeric constants. These inequalities are handled semantically through an expansion of the covered range.

3.3 Instance and Fact Generation

The first step of the induction algorithm is the identification of positive and negative examples. This can be done either directly, by supplying lists of the positive and negative examples, or by providing formulae that allow the generation of examples and their division into positive and negative examples.

The process of dividing the examples into positive and negative examples can be done using either open or closed world semantics.

Once the examples have been identified, a portion of the knowledge base structure is then extracted. This consists of the depth-limited transitive closure of a set of ground facts about each instance, computed by following chains of relations through all other instances mentioned in the ground facts (see Figure 3). This expansion is subject to a depth cutoff and can be further constrained to exclude certain relations. These parameters are described in Section 3.5.

```
least_general_generalization(rule, example):
new-rule = NULL
FOREACH clause in rule
  IF it matches a fact about example exactly
     THEN add the clauses to new-rule
     ELSE choose one matching fact at random from Facts(example)
          add the generalization of fact and clause to new-rule.
          IF there is matching fact, THEN drop the clause.
RETURN new-rule
```

Figure 4: Clause Generalization Algorithm—High Level

```
* Facts and rule clauses match if they have the same structure
* Facts and rule clauses have the same structure if they have the same
  arity and predicate and any functional terms have the same structure.
  Example: (P (Q i) j) matches (P (Q x) y) but not (P (F i) j).
* Clauses generalize as follows:
  If the terms are identical, they are unchanged.
  If the terms differ, unless a special case applies, substitute a
     variable. The same variable for a given term is used, if possible.
  Special cases:
     ISA predicates:  The concept term generalizes to one of the most
         specific common parents in the concept hierarchy.
         Examples: (isa ?x Man) and (isa ?x Woman) become (isa ?x Person)
                   (isa ?x Man) and (isa ?x Dog) become (isa ?x Mammal)
     Numeric Inequalities:  Literal numbers generalize so as to expand
         the range of the inequality.
         Examples:  (>= (F i) 3) and (>= (F i) 5) becomes (>= (F i) 5)
                    (>= (F a) 3) and (>= (F b) 5) becomes (>= (F ?x) 5)
```

Figure 5: Clause Matching and Generalization Rules

3.4 Clause Generalization Algorithm

The key step in the bottom up rule induction is the generalization of individual clauses in the antecedent. The rule is constructed by starting with a very specific antecedent and relaxing the conditions to cover more positive examples while continuing to exclude negative examples.

The starting point for this is a very specific rule which is composed of the facts about one randomly selected positive instance. The set of facts considered is the subset of all facts generated using the algorithm in Figure 3, with a variable substituted for all references to that particular instance. Normally, the facts are sufficiently constraining that this particular rule will match only one instance.

The coverage of the rule is therefore expanded by applying a generalization procedure shown in Figure 4. Clauses in the rule antecedent which have a potentially matching facts from another positive example are retained and generalized if necessary. The potential

match and clause generalization follows the rules shown in Figure 5.

This expansion proceeds largely syntactically, through substitution of variables for parts of predicates that don't match. The syntactic processing is augmented by limited semantic match rules that know how to generalize inequality and type clauses. Inequalities are generalized by expanding the range of numbers covered. Type clauses (`INSTANCE-OF` in PowerLoom, `isa` in Cyc) are generalized by finding the most specific common supertype. For example, `(INSTANCE-OF ?x Man)` and `(INSTANCE-OF ?x Woman)` would generalize to `(INSTANCE-OF ?x Person)`. For example, `(INSTANCE-OF ?x Man)` and `(INSTANCE-OF ?x Dog)` would generalize to `(INSTANCE-OF ?x Mammal)`.

Once a candidate generalization is generated, it is tested for exclusion of negative examples and coverage of positive examples. A number of such candidates are generated and the best choice is chosen to be the new rule antecedent.

The major innovation in the generalization rules is the introduction of hierarchy-based generalization of concept (Collection) terms. This is an area where the semantics in the knowledge base influence the learning process. It is also an area which differentiates this learning algorithm from traditional ILP work. Traditional ILP works with logic programs that do not have a rich a representational structure. Both Cyc and PowerLoom do support this, and this allows our system to learn rules eploiting the concept hierarchy, a feature not present in traditional ILP algorithms.

3.5 Algorithm Control Parameters

The learning algorithm is directed by the value of certain parameters which allow the choiced of options mentioned at various points in the algorithm descriptions. Those options are implemented as slots (fields) of the class `learning-parameters`. The options are:

closed-world-examples? (*boolean*) Determines whether the closed-world assumption is made when generating training examples for rule learning. If `false`, then explicitly negated examples must be present in the knowledge base. This only applies if the training examples are generated by the algorithm. Default is `true`.

simplify-rule-antecedents? (*boolean*) Controls whether to simplify rule antecedents so that only the minimum clauses needed to satisify the rule remain. This should be `false` for rules that will be post-editted by humans. Default is `false`.

variables-can-match-functions? (*boolean*) Controls whether variables can be used to match functional terms in formulae during generalization. If set to `true`, structural information about the functional relationship can be lost via overgeneralization. This should almost always be `false`. Default is `false`.

suppress-singleton-rules? (*boolean*) Controls whether rules are generated when there is only a single positive example. With only one example, there will be no generalization step. Note that this does not currently prevent rules that only cover one positive

example from being returned as long as there are multiple examples to learn from. Default is `true`.

max-structure-depth (*integer*) Maximum distance in relational link count for gathering facts about an instance. This controls the size of the relational universe for the rule learning. If set too low, then there may not be enough structure to find a rule. If set too high, learning will take longer. Also, either more positive examples will be needed to prune the rules or else rules may be excessively specific. Default value is 3.

max-search-iterations (*integer*) Maximum number of iterations for the generalization search. This number will not be exceeded during learning. The actual upper bound will be between 5 times the number of positive examples and this limit. Default value is 100.

allowed-false-positive-rate (*float*) Fraction of false positives a rule antecedent is allowed to have. This is zero for learning strict rules, but can be set a bit higher for default rule learning if false postitives are acceptable. Default value is 0.0, for strict rules.

taboo-operators (*(cons*) Cons list of symbols (names or relations) that will not be followed in the construction of the fact network. Default is `nil`.

verbosity (*integer*) Controls the amout of output to the console during learning. 0 gives no output. 3 produces the most. Default value is 3.

The structure depth is the key parameter that controls the search space of the generalization routine, since it indirectly determines how many facts each positive instance has associated with it. Using the development cases, the roadways example can be learned at depth 2 whereas the airfields requires depth 3. A depth of three seems to be a reasonable limit, since it allows consideration of one level of facts associated with instances connected to the example itself.

In fact, because of the increase in search times, a reasonable strategy would be to start at depth 3 and increase the depth if no rules can be found. This is similar to the iterative deepening method of doing a breadth-first search. If an answer is found at a higher depth setting, it will usually not take very much more time to find it by this iterative process than by choosing the higher depth the first time.

4 Measuring Rule Quality

We developed a rule scoring method based on ideas from association rule learning [Hipp *et al.*, 2000]. In most data-mining type operations 100,000 rules are reported as not uncommon. To limit the consideration of this large space, a number of metrics are used. One of the filtering metrics is “frequency” of an association rule. In effect this is the number of

bindings for the antecedent of the rule. Each rule also has a "confidence" score, which is the percentage of times the consequent follows from the antecedent.

In our application, we do not anticipate having such a large number of potential rules, especially since the rule generation will be guided by domain experts using the knowledge acquisition system. Nevertheless, the ideas of measuring the resulting rules by comparison with the data have merit, but they need to be extended slightly.

We should also note that other ILP systems (like CHILLIN [Zelle *et al.*, 1994]) use rule size as a metric, preferring more compact rules to more verbose ones. This is clearly a desirable characteristic as well, but we chose to focus on a data-based metric.

The following data-based measures were ones that seem to be both reasonable and useful for describing rules. This is for a rule of the form $A \Rightarrow B$:

F—Frequency The number of tuples satisfying A.

P—True Positives The fraction of tuples satisfying A that also satisfy $A \wedge B$.

N–False Positives The fraction of tuples satisfying A that also satisfy $A \wedge \neg B$

U–Usefulness The fraction of tuples where B is not known. $U = (A - N - P)/A$.

The addition of separate false negative and usefulness ratings is because of the use of open-world semantics. With closed world semantics, there are would be no unknown items to cover.

In association rule learning, it is easy to determine the size of the set, since the learning algorithm operates over a finite domain. In learning rules in first order logic, there is a potentially infinite number of antecedents, and since the same object can appear in more than one place in the resulting tuple, an effectively infinite number of tuples. Therefore, frequency will need to be treated as a simple count rather than a relative frequency.

U is termed usefulness because it tells us in how many cases using the rule would give information that is currently not present in the knowledge base. This measure of usefulness is based on the information gain of the rule.

In Section sec:score, an example of how these metrics can be used is presented.

5 Data and Use Case

For software development, we have chosen problems related to terrain analysis, as defined in the Revised Student IPB Questions document prepared by Dutch Sley of SAIC. This document is attached in an appendix. For development we chose questions 1 and 4 (see Figure 6). Question 1 (Roadways) was chosen because it was a simple test case, since the key information needed to classify the roadways is a direct feature of the Roadway itself. Question 4 (Airfields) was chosen because it allowed us to use a more involved structure. As we conceived the situation, the features needed for inference were not direct features of airfields, but rather features of runways, which were in turn features of airfields.

```
1. All hard surface roads 4 meters wide or greater support the
   movement of maneuver forces.
     a. Does road xxx support the movement of maneuver forces?

4. Short unimproved airfields are suitable for military use.
     a. Is xxx airfield 3,500 ft long and 60 ft wide?
        (KC-130 transport aircraft reqirement)
```

Figure 6: Questions for Rule Induction

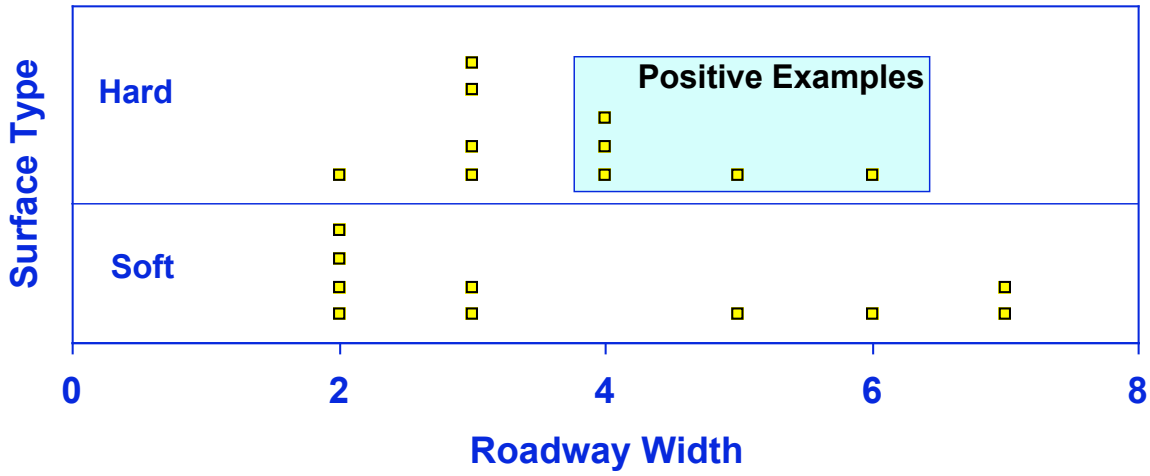

Figure 7: Randomly Generated Roadway Examples

Using these as the basis of our development, we created an ontology describing these domains and generated instances with randomly distributed charteristics as shown in Figures 7 and 8.

The result of running the induction algorithm for the roadways example is shown in Figure 9. The rule antecedent has 7 clauses in it, and finds rules that satisfy the requirements, although there are more clauses than might be desired. Two of the clauses are irrelevant ([3],[4]) and one of them ([5]) is redundant—it only requires that the width be present, but that requirement is implicit in clauses [6] and [7]. Unfortunately, it is difficult in general to detect that the conditions on variable ?X1 are subsumed by the conditions on variable ?Y (since there is no constraint saying that ?X1 and ?Y must be different) using only pairwise comparisons. One empirical solution is to use the training set to simplify the rule antecedents. This is discussed below.

One should also not that the induced rule has an additional width constraint [7] that is not required by the task. This upper limit on the road width is an interaction between the training examples (which had no roads wider than 6m) and the desire to use a minimal generalization. As such, the basic bottom-up algorithm will not generalize beyond the

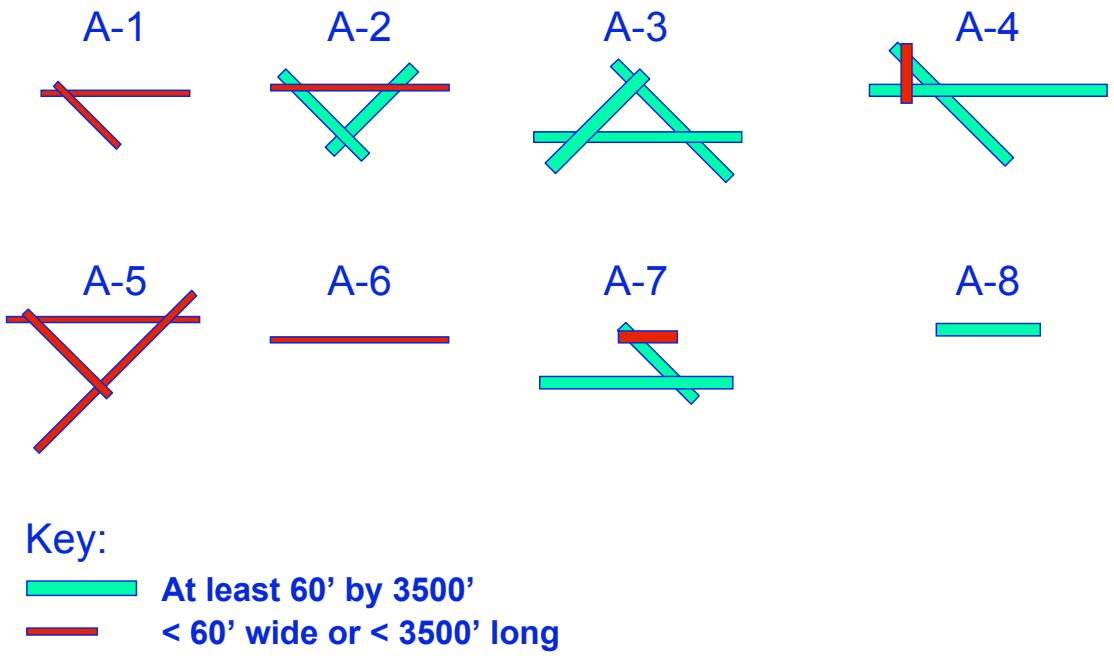

Figure 8: Randomly Generated Airfield Examples

```
(=> (AND (surfaceType ?Y hardSurfaceType)                    [1]
         (INSTANCE-OF ?Y Roadway)                            [2]
         (INSTANCE-OF ?X1 Roadway)                           [3]
         (surfaceType ?X1 hardSurfaceType)                   [4]
         (widthOfObject-m ?Y ?X2)                            [5]
         (>= (widthOfObject-m ?Y) 4)                         [6]
         (=< (widthOfObject-m ?Y) 6))                        [7]
   (RoadwaySupportsMovementOfMilitaryForces ?Y))
```

Figure 9: Learned Rule for Roadways, Full

```
(=> (AND (>= (widthOfObject-m ?Y) 4)                          [1]
         (surfaceType ?Y hardSurfaceType))                    [2]
    (RoadwaySupportsMovementOfMilitaryForces ?Y))
```

Figure 10: Learned Rule for Roadways, Simplified

```
(=> (AND (INSTANCE-OF ?Y Airfield)                            [1]
         (hasRunway ?Y ?X1)                                   [2]
         (INSTANCE-OF ?X1 Runway)                             [3]
         (lengthOfObject-ft ?X1 ?X2)                          [4]
         (>= (lengthOfObject-ft ?X1) 3500)                    [5]
         (=< (lengthOfObject-ft ?X1) 5500)                    [6]
         (widthOfObject-ft ?X1 ?X3)                           [7]
         (>= (widthOfObject-ft ?X1) 75)                       [8]
         (=< (widthOfObject-ft ?X1) 85))                      [9]
    (AirfieldSuitableForMilitaryUse ?Y))
```

Figure 11: Learned Rule for Airfields, Full

boundaries of the training instances. This rule, however, will serve as a good starting point for refinement by a domain expert.

Empirical simplification is a process whereby individual clauses of a rule antecedent are tested for relevance. Relevance is determined by removing the clause from the antecedent and seeing if any (more) negative examples are allowed. If a clause in the antecedent does not exclude any negative examples in the training set, then that clause has no discrimintating power with respect to the training set. An example of the roadway rule that has been simplified is shown in Figure 10. The simplification has reduced the criteria in the antecedent to the absolute minimum, namely a hard road surface and a width greater than 4m. This minimum is a much broader generalization of the training data than the basic algorithm allows.

In fact, the rule as derived doesn't even require that ?Y be a Roadway! This latter omission is an result of testing relevance only with respect to the training examples. The use of limited examples for rule induction contributes to the speed of learning, especially in a situation where there are many examples in the knowledge base and the rule induction is only performed over a small subset. This means that if simplification of rules is used in a fully automated system, then the rules will need to be augmented by the addition of clauses that where used to restrict the domain of the training examples.

The more complicated example of the airfields will be discussed next. Figure 11 shows the rule induced for the airfield example without any simplification. This rule happens to come very close to the original criteria set in the questionnaire. The upper bounds on width and length of runways ([6], [9]) are again consequences of the conservative generalization strategy used in the rule induction algorithm. The larger minimum width of 75ft in clause [8] versus the 60ft in the problem description is an artifact of the examples used.

```
(=> (AND (hasRunway ?Y ?X1)                                    [1]
         (>= (widthOfObject-ft ?X1) 80))                       [2]
    (AirfieldSuitableForMilitaryUse ?Y))
```

Figure 12: Learned Rule for Airfields, Simplified

```
(=> (AND (=< (widthOfObject-m ?R) 6)                           [1.1]
         (>= (widthOfObject-m ?R) 5)                           [1.2]
         (=< (orientationOfObject ?R) 44))                     [1.3]
    (RoadwaySupportsMovementOfMilitaryForces ?R))
(=> (AND (=< (widthOfObject-m ?R) 6)                           [2.1]
         (>= (widthOfObject-m ?R) 4)                           [2.2]
         (>= (lengthOfObject-ft ?R) 14060))                    [2.3]
    (RoadwaySupportsMovementOfMilitaryForces ?R))
```

Figure 13: Learned Rule for Airfields with Irrelevant Features, Simplified

The simplification of the airfield rule shown in Figure 12 is a much more radical simplification than for the roadway case. It turns out that all of the airfields in the test set which satisfied the criteria had at least one runway of width 80ft or more and none of the negative training examples had runways that wide. This is exacerbated by the much smaller number of examples used in the training set, and illustrates the risk of overgeneralization when automatically simplifying rules that have very few examples (positive or especially negative). The airfield problem was run with only 5 positive and 3 negative examples.

The reason the width criteria are different in figures 11 and 12 is because they were produced by different runs of the rule induction algorithm. Since the matching clause in the clause generalization routine is randomly chosen from among the candidates, there will be some variability in the results depending on precisely which clauses happen to be chosen.

5.1 Adding Irrelevant Characteristics

The initial set of rules were induced under ideal conditions, since all of the attributes of the examples were relevant attributes. This simplification was done for the convenience of developing the basic algorithm. After the basic algorithm was developed, we conducted more experiments to assess the effect of adding assertions using additional irrelevant attributes. We expected that the effects would be two fold. First, the algorithm would produce rules more slowly and second that one would need more training examples to get good rules.

The roadway examples were augmented with randomly chosen length and (compass) orientation attributes. For simplicity, we will examine an automatically simplified rule first.

Figure 13 shows the results. In this case, two rules were induced. That is because the first rule produced by the induction algorithm only covered two positive examples. The second rule happens to cover all five positive examples in the training set, including the two

also covered by the first rule. This was a serendipitous result since the algorithm removed the two already covered examples from the training set before trying to induce the second rule.

One of the striking results is that the roadway surface type is not present in the simplified rule. Instead, length, width and orientation appear. It happens that those features in the training set correlated with road surface type. This illustrates why more training examples are needed when there are irrelevant attributes. With a small sample size like we used, there is too high a chance for spurious correlations to occur and be enshrined in the learned rules, especially when all of the processing is done automatically.

Figure 14 shows the full rules learned in a separate run of the program. The full rules have the kernel of what we would have wanted (clauses [2.1], [2.2], [2.4]), and so could be reduced to good, concise rules by human post-editting.

When operating with real-world datasets, it is apparent that there will need to be more examples used than in the demonstration described here. As this last example shows, the learning algorithm will sometimes produce overlapping rules, when the generalization makes incorrect choices early in the process. To support more efficient navigation and evaluation of the prototype rules, it would be useful to have some measure of rule quality, using the metrics described in Section 4. We will explore this issue using the alternate use case described in the next section.

5.2 Alternate Use Case and Examples

To try the work on a slightly different type of input, We also created a small example of a military organization with various hierarchically arranged military units, their commanders and the commanders' ranks. A graphical view of the domain can be seen in Figure 15. Using this domain, we can see some of the potential that can be learned, and how the application of the scoring mechanisms that can be used.

For example, one can learn that platoons are subordinate units of companies. One can also learn that a military unit commanded by someone with the rank of sergeant is a squad and that a military unit with a captain as commander is a company. The example domain is setup with a few ranks unknown (as indicated by "??" in the organization chart), so that there would be some cases where the data needed for the induction are missing.

A very verbose example rule produced from the unit organization dataset is shown in figure 16. This is a learned rule for figuring out if some military unit is a platoon.

Examining the output, there are actually a couple of good candidate rules for our domain, but there is a lot of additional "chaff" that one would want to separate as well. The reasonable rules are one using clause [1] and [3] to imply the conclusion. This gives a rule that can be paraphrased as "Units whose superior unit is a company are platoons". This is actually a good and useful rule to have.

The induction actually explores this path a bit too long and also comes up with a combination [1] and [4] which paraphrases as "A unit whose superior unit's superior unit is the 1-1st-Infantry battalion is a platoon." There are two problems with this rule. One is

```
(=> (AND (INSTANCE-OF ?R Roadway)                          [1.1]
         (surfaceType ?R hardSurfaceType)                  [1.2]
         (widthOfObject-m ?R 4)                            [1.3]
         (=< (widthOfObject-m ?R) 4)                       [1.4]
         (>= (widthOfObject-m ?R) 4)                       [1.5]
         (lengthOfObject-ft ?R ?X2)                        [1.6]
         (>= (lengthOfObject-ft ?R) 16750)                 [1.7]
         (=< (lengthOfObject-ft ?R) 20130)                 [1.8]
         (orientationOfObject ?R ?X3)                      [1.9]
         (>= (orientationOfObject ?R) 98)                  [1.10]
         (=< (orientationOfObject ?R) 143)                 [1.11]

         (INSTANCE-OF ?X1 Roadway)                         [1.12]
         (surfaceType ?X1 hardSurfaceType))                [1.13]
      (RoadwaySupportsMovementOfMilitaryForces ?R))
(=> (AND (INSTANCE-OF ?R Roadway)                          [2.1]
         (surfaceType ?R hardSurfaceType)                  [2.2]
         (widthOfObject-m ?R ?X5)                          [2.3]
         (>= (widthOfObject-m ?R) 4)                       [2.4]
         (=< (widthOfObject-m ?R) 6)                       [2.5]
         (lengthOfObject-ft ?R ?X3)                        [2.6]
         (>= (lengthOfObject-ft ?R) 14060)                 [2.7]
         (=< (lengthOfObject-ft ?R) 24810)                 [2.8]
         (orientationOfObject ?R ?X4)                      [2.9]
         (>= (orientationOfObject ?R) 41)                  [2.10]
         (=< (orientationOfObject ?R) 143)                 [2.11]

         (INSTANCE-OF ?X1 Roadway)                         [2.12]
         (surfaceType ?X1 hardSurfaceType)                 [2.13]

         (INSTANCE-OF ?X2 Roadway)                         [2.14]
         (surfaceType ?X2 hardSurfaceType))                [2.15]
     (RoadwaySupportsMovementOfMilitaryForces ?R))
```

Figure 14: Learned Rule for Roadways with Irrelevant Features

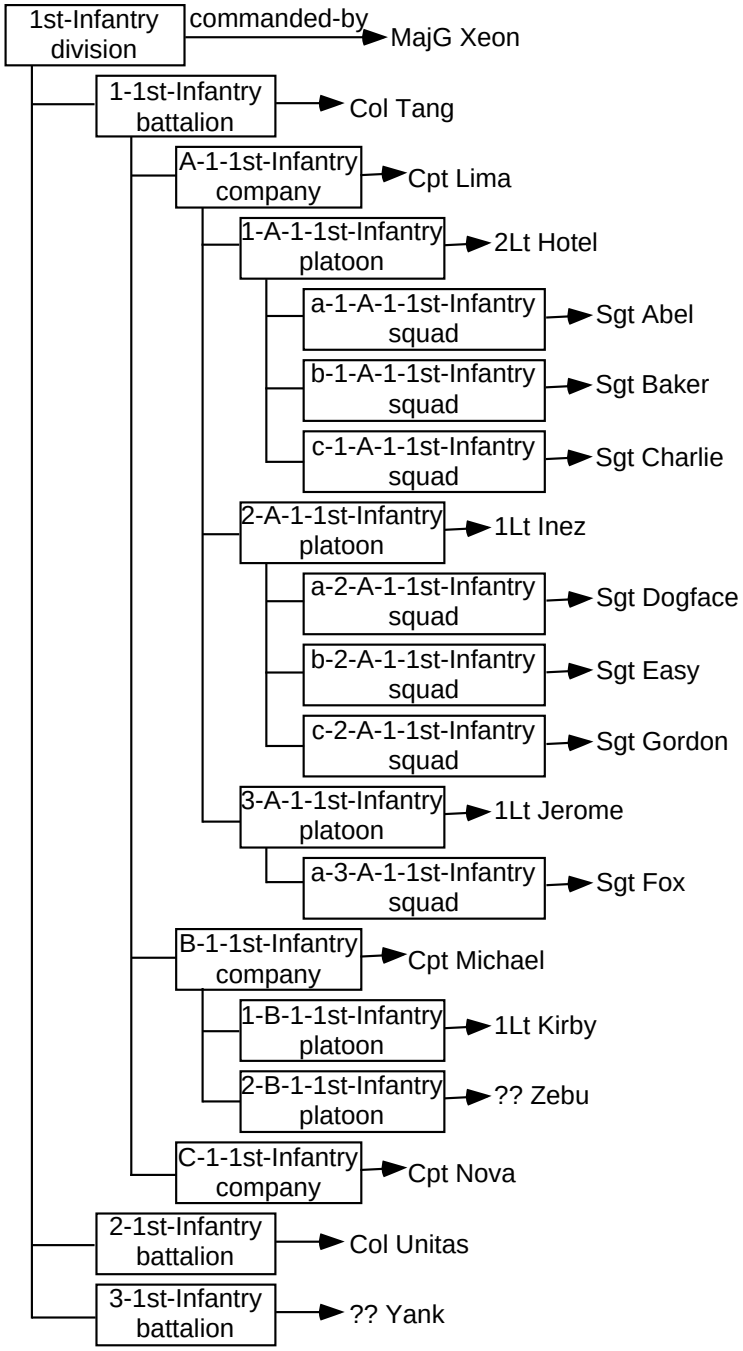

Figure 15: Rule Learning Data: Organization Chart

```
(=> (AND (SUPERIOR-UNIT ?Y ?X1)                          [1]
         (COMMANDER-OF ?Y ?X2)                           [2]

         (INSTANCE-OF ?X1 COMPANY)                       [3]
         (SUPERIOR-UNIT ?X1 1-1ST-INFANTRY)              [4]
         (INSTANCE-OF ?X2 SOLDIER)                       [5]
         (HOLDS-MILITARY-RANK ?X2 ?X3)                   [6]
         (INSTANCE-OF ?X3 MILITARY-RANK)                 [7]

         (INSTANCE-OF ?X4 PLATOON)                       [8]
         (INSTANCE-OF ?X4 MILITARY-UNIT)                 [9]
         (SUPERIOR-UNIT ?X4 ?X1)                        [10]
         (COMMANDER-OF ?X4 ?X2))                        [11]

    (INSTANCE-OF ?Y PLATOON))
```

Figure 16: A bottom-up learned rule for identifying a platoon from military-unit.

that it is overly specific, but that is because of the limited data set used for the example, although if one replaced 1-1st-Infantry with a variable in [3] and added a type constraint this would produce a valid rule similar to the previous rule. The reason there is no type constraint is that the structure cutoff depth was reached before that fact was added to the platoon facts. The problem is that the chaining along the superior-unit link, although not incorrect, doesn't add any discrimination power.

There is also another interesting rule included in the induction output. It is the rule that combines [1], [8] and [10]. This says that "if a unit and a known platoon have the same superior unit, then the unit is a platoon." In other words, if a unit's sibling unit in the command hiearchy is a platoon, so is that unit.

Another correct, but excessively general subset is the rule comprised of clauses [2] and [5]. It says that if your unit is commanded by a soldier, then you are a platoon. It is, of course, slightly unfair to lift clauses out of the context of more restrictive clauses and complain about them being over general, but I will use it to illustrate the benefits of scoring in Table 1. For completeness, I would note that such a rule would not actually be produced by the learning algorithm since it would also match all of the negative examples.

5.3 Example of Rule Scoring

Applying the rule scoring predicate to the simple rules, I get the results shown in Table 1

Note that rule R3 gets an increase in frequency because of the combinatorics of siblings. By just evaluating the matches for the antecedent, there is some multiple counting.

Rule R4 has a large number of results (one for each unit) and a large number of unknown

Table 1: Scores for some posited learned rules

ID	Rule	F	P	N	U
R1	Superior is a company	5	1.0	0.0	0.0
R2	Superior's superior is 1-1st-Infantry	5	1.0	0.0	0.0
R3	Siblings are platoons	13	1.0	0.0	0.0
R4	Commander is a soldier	19	0.26	0.0	0.74
R4′	Commander is a soldier	19	0.26	0.74	0.0

results. The lack of false positives and high perceived usefulness make this seem to be a good rule. The problem is that proving true negation is difficult, and the knowledge base for the test did not have a key disjointness axiom stating that platoons are disjoint from companies, etc. Adding the appropriate disjointness information gives the results shown for R4′, which is clearly a bad rule. This does suggest that, heuristically, one should be suspicious about rules with very high usefulness ratings and low true positive ratings.

6 Conclusion

We have been able to develop a set of algorithms for learning inference rules in a rich knowledge environment using few examples. In this section, we discuss the coverage of the IPB questions and the limitations of the currently learned rules. We then discuss the performance characteristics of the algorithm. Finally we present a scenario for how this function can be added to the KRAKEN interface.

6.1 Limitations of Current Learned Rules

The bottom-up approach considers only the facts asserted about positive examples in constructing rules. That means that the algorithm will not add negated clauses to the rules that do not appear as explicit assertions. In other words, a rule which says an all-weather travel route is one which does not cross a ford will likely not be discovered by this algorithm. That is based on the likely case that ford crossings will be asserted only in the cases where they exist and a closed-world assumption will be used for the much more common case of no fords. But since the closed-world assumption is implicit rather than explicit, there is no syntactic trace of the lack of fords. Future work may be able to consider adding negated clauses from negative examples, but the current algorithm does not look at the structure of negative examples at all.

Since the learning algorithm existentially quantifies variables, the induced rules will not work in cases where counting of relation fillers is required. That is because two distinct fillers of a single relation `(R i1 f1)` and `(R i1 f2)` can be subsumed in the generalization step by a single clause: `(R i1 ?x1)`. Once this happens, the number of clauses

```
23. Avenues of Approach include at least 2 viable mobility corridors.
       a. Does the area include 2 mobility corridors?

24. Avenues of Approach contain at least 2 mobility corridors close
    enough to be mutually supporting.
       a. Does the Regimental AA's have 2 Battalion mobility corridors
          no more than 6 km apart?
       b. Does the Battalion AA's have 2 Company mobility corridors no
          more than 2 km apart?
```

Figure 17: IPB Questions with Number Restrictions

in the antecedent is reduced and the number of fillers is lost. Some examples from the IPB Questions are shown in Figure 17.

This can be partially offset by the inclusion of specialized number restrictions on the fillers. This would allow induction of rules that took advantage of that information. With the present configuration, the number restrictions would have to match exactly, although an extension of the generalization code to introduce inequalities for the numbers would be possible. This extension could be implemented in a way similar to the current system's treatment of functional terms with numeric values during fact generation, or through a specialized generalization mechanism similar to what is currently done for inequality predicates.

6.2 Coverage of IPB Questions

Of the 27 questions in the IPB questions, the current implementation of the learning algorithm can easily handle 10 of them (1–5, 9, 11–13, 16). A further 9 (6–8, 14, 17–21) may be handled, depending on details of how the representation is chosen. If there is an explicit representation in the ontology of features such as "elevation variance of X meters per square kilometer", then the rules would be amenable to learning using this algorithm. Finally, 8 of the questions are too hard. They either require explicit counting (15, 23–25), implicit counting (10 "several areas"), or require geometric reasoning that may not be encoded in the logical descrition of the domain (22, 26, 27). These latter cases include predicates such as "bypass", "parallel" and "intersect".

6.3 Algorithm Performance

When we began development using the Roadways and Airfields examples, the runtime of the learning algorithm was quite large. Typically this was about 10 to 30 minutes respectively. The rules produced were also quite large, with many redundant and irrelevant clauses. Improvements to the generalization algorithm eliminated one source of combinatorial explosion and produced both better runtimes, but also more compact rules. Table 2

Table 2: Mean Runtimes for Rule Induction Examples

	Roadway		Airfields	
Rule Simplified?	No	Yes	No	Yes
Basic Facts	1.4s	1.1s	1.9s	2.1s
Additional Facts	9.6s	10.8s	6.3s	7.9s

"Rule Simplified?" is a post-processing step that uses additional knowledge base queries to reduce a rule antecedent to the simplest form supported by the examples.
"Basic Facts" reports learning with only relevant facts asserted about the instances.
"Additional Facts" means that additional irrelevant facts were added to the knowledge base.

Timing results are the mean times of five repetions of the rule induction. Results produced using the Java version of PowerLoom and the rule induction algorithm on an 800Mhz G4 processor with 1GB RAM, Java Hostspot^{TM} Client VM 1.4.1

presents the runtime results for our algorithm using the Roadway and Airfield data, showing improved run times of a few seconds.

As might be expected, the process of learning rules when there are more features to be sorted through takes longer. It is also the case that simplifying the rules should add to the run time, although the actual impact on performance is small. This is because the simplification represents a one-time pass through the rule after learning. This is a negligible part of the interaction with the knowledge base compared to the multiple evaluations of rule antecedents during the learning phase. This means that the decision to simplify rules or not should depend on other considerations.

6.4 Integration with KRAKEN

Currently when the user enters new knowledge in KRAKEN, the Cyc interface leads the user through a dialog to see if some of the information (rules) can be generalized. In a situation where there are enough facts in the knowledge base, a rule evaluation predicate based on the rule quality metrics could be applied to determine the true and false positive rate and usefulness of potential generalizations. If it is already known from the data that the generalization either leads to contradictions, or is not highly supported by the data, then asking about generalizing the rule would be pointless.

It is also the case, that given the large number of potential rules that could be learned, that we would need to have some method of user-invoked control over the learning algorithm. This could be done by having a module invoked when a query doesn't return expected results. This would be similar to the current method of invoking the WhyNot

module [Chalupsky and Russ, 2002]. In this way, the system would only look for rules that conclude things that the person entering the knowledge thinks the system should know or be able to figure out.

Adding rule learning to KRAKEN will allow more rapid entry of rule-knowledge, one of the tasks that is known to be difficult for subject-matter experts. Having a learning component also means that revisions of the rules when more knowledge has been entered can be done (semi-)automatically, at least for those rules which had been created by the induction algorithm.

References

[Chalupsky and Russ, 2002] Hans Chalupsky and Thomas A. Russ. WhyNot: Debugging failed queries in large knowledge bases. In *Proceedings of the Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pages 870–877, 2002.

[Hipp *et al.*, 2000] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[Moriarty, 2000] David E. Moriarty. Determining effective military decisive points through knowledge-rich case-based reasoning. In *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 453–462, 2000.

[Muggleton and de Raedt, 1994] Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, (19,20):629–679, 1994.

[Muggleton and Feng, 1990] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 368–381. Ohmsma, Tokyo, Japan, 1990.

[Muggleton, 1995] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[Pool *et al.*, 2003] Mike Pool, Ken Murray, Julie Fitzgerald, Mala Mehrotra, Robert Schrag, Jim Blythe, Jihie Kim, Hans Chalupsky, Pierluigi Miraglia, Thomas Russ, and Dave Schneider. Evaluating SME-authored COA critiquing knowledge. In *Proceedings of the Second International Conference on Knowledge Capture*, 2003. to appear.

[PowerLoom] PowerLoom knowledege representation language. University of Southern California, Information Sciences Institute Website http://www.isi.edu/isd/LOOM/PowerLoom/.

[Quinlan and Cameron-Jones, 1993] J. Ross Quinlan and R. Mike Cameron-Jones. FOIL: A midterm report. In *Machine Learning: ECML-93, European Conference on Machine Learning, Proceedings*, volume 667, pages 3–20. Springer-Verlag, 1993.

[Zelle *et al.*, 1994] John M. Zelle, Raymond J. Mooney, and Joshua B. Konvisser. Combing top-down and bottom-up techniques in inductive logic programming. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 343–351. Morgan Kaufmann, 1994.

Appendix: Student IPB Questions

The following Student IPB (Intelligence Preparation of the Battlefield) Questions were produce by Dutch Sley of SAIC for training military personnel in terrain analysis of the battlefield.

Revised Student IPB Questions

1. All hard surface roads 4 meters wide or greater support the movement of maneuver forces.
    a. Does road xxx support the movement of maneuver forces?

2. Undamaged commercial railways support movement of military forces and equipment.
    a. Does railroad xxx support movement of military forces and equipment?

3. Pipelines transport petroleum, oils, and lubricants.
    a. Does xxx pipeline transport petroleum, oils, or lubricants?
    b. Is xxx pipeline above ground?
    c. Is xxx pipeline an obstacle to maneuver forces?

4. Short unimproved airfields are suitable for military use.
    a. Is xxx airfield 3,500 ft long and 60 ft wide? (KC-130 transport aircraft reqirement)

5. Hydrology:
    a. Rivers, streams, lakes, swamps, and bogs wider than 60 feet are NO GO terrain for vehicles.
        i. Is the xxx wider than 60 feet?
    b. Rivers, streams, lakes, swamps, and bogs deeper than 96 inches are NO GO terrain for Armor.
        i. Is the xxx deeper than 96 inches?
    c. Rivers and streams with a velocity greater than 5 feet per second are NO GO terrain.
        i. Is the xxx velocity greater than 5 feet per second?
    d. Vertical riverbanks higher than 4 feet are NO GO terrain.
        i. Is the riverbank higher than 4 feet?
    e. Slopes greater than 60% are NO GO terrain for Armor.
        i. Is the hill slope greater than 60 %?

6. Built-up areas wider than 500 m or areas that cannot be easily bypassed on both sides are NO GO terrain (armored forces only).
    a. Is the built-up area wider than 500 m?

7. Terrain with elevation variations of 200-400 meters per kilometer is NO GO terrain.
    a. Does the elevation vary between 200- 400 meters in one square kilometer?

8. Areas that have only one trail per km and no hard-surface roads are NO GO terrain.
    a. Does an area have only one trail per km and no hard-surface roads?

9. Manmade obstacles (minefields, tank ditches, abate) are NO GO terrain.
    a. Does an area have any minefield, tank ditch, abatis?

10. Rivers, streams, lakes and flooded areas that can be forded or spanned in several areas are SLOW GO terrain.
    a. Can the xxx be forded or spanned in several areas?

11. Frozen rivers, streams, or lakes with ice more than 31.5 inches thick that would support tanks are SLOW GO terrain.
    a. Is the xxx frozen to form ice 31.5 inches thick?

12. Slopes of 30-45% uphill (directional) are SLOW GO Terrain.
    a. Is the slope 30-45%?

13. Wooded areas where the trees are 2" thick with less than 20 ft interval (armored forces only) is SLOW GO Terrain.
    a. Does the area contain trees that are 2" thick and less than 20 ft intervals?

14. Areas with elevation changes of no more than 100-200 meters per km are SLOW GO Terrain.
    a. Does the area contain elevation changes of no more than 100-200 meters?

15. Dense woods with 1 hard surface road or 2 trails per km are SLOW GO Terrain.
    a. Does the area contain 1 hard surface road or 2 trails per km?

16. All man-made obstacles: minefields, tank ditches, abatis, barriers are OBSTACLES.
    a. Does the area contain a minefields, tank ditches, abates, or barriers?

17. GO Terrain. Go terrain is fairly open terrain which presents no hindrance to ground movement. Nothing needs to be done to enhance movement. Examples are:
    a. Can the rivers and streams be forded anywhere along their length or are 5 ft or less wide?
    b. Are the rivers and streams that are frozen 4 inches thick? (for dismounted infantry)
    c. Are the slopes of less than 30% in an area?
    d. Are the trees less than 2 inches thick or with intervals greater than 20 ft in an area?
    e. Does the elevation vary from 0-100 meters per kilometer in an area?
    f. Are there 2 or more, hard-surface roads per km in an area?

18. Company size mobility corridors have a minimum width of 500m.
    a. Does the area contain Company size mobility corridors?

19. Battalion size mobility corridors have a minimum width of 1.5 km.
    a. Does the area contain Battalion size mobility corridors?

20. BDE/Regt size mobility corridors have a minimum width of 3 km.
    a. Does the area contain BDE/Regt size mobility corridors?

21. Division size mobility corridors have a minimum width of 6 km.
    a. Does the area contain Division size mobility corridors?

22. Mobility corridors bypass all NO GO terrain and only occasionally pass through or over SLOW GO terrain.
    a. Does the area contain no NO GO terrain and less than 1 km of SLOW GO terrain?

23. Avenues of Approach include at least 2 viable mobility corridors.
    a. Does the area include 2 mobility corridors?

24. Avenues of Approach contain at least 2 mobility corridors close enough to be mutually supporting.
    a. Does the Regimental AA's have 2 Battalion mobility corridors no more than 6 km apart?
    b. Does the Battalion AA's have 2 Company mobility corridors no more than 2 km apart?

25. Avenues of Approach are identified in the Area of Operations and the Area of Interest.
    a. Are there Avenues of Approach in the Area of Operations and the Area of Interest?

26. The most likely Avenue of Approach parallels key terrain and leads to an objective.
    a. Does parallel key terrain and intersect an objective?

27. The elements of OCOKA are used to analyze Avenues of Approach (AA).
    a. Observation and Fires.
        i. Observation corridors from defensive positions intersect with Avenues of Approach.
            1. Where are the observation corridors that intersect with the Avenue of Approach?
        ii. Direct and indirect fire weapon range fans intersect with Avenues of Approach.
            1. Where do the direct and indirect weapon range fans intersect with the Avenue of Approach?

    b. Cover and Concealment.
        i. Natural cover protects the defender.

1. Does the area contain natural cover?

ii. Terrain and vegetation can conceal the attackers movement.
    1. Where does the terrain or vegetation conceal movement?

c. Obstacles, existing and potential.
    i. Obstacles impede the attackers advance.
        1. Does the Avenue of Approach contain obstacles?
    ii. Natural obstacles are reinforced by man made obstacles.
        1. Does the Avenue of Approach contain man made obstacles?
    iii. Obstacles parallel to the AA and afford flank protection and limit lateral movement.
        1. Are there obstacles parallel to the Avenue of Approach?

d. Key and Decisive Terrain.
    i. Key terrain features that in the control of a combatant will affect the conduct of an operation.
        1. Does the area contain key terrain features?

e. Avenues of Approach should provide adequate maneuver space, ease of movement, maintain trafficability, and provide a direction of approach to the objective.
    i. Does the Avenue of Approach support maneuver?