Abstract. The Rete multi-pattern match algorithm [Forg 82] is an efficient method for comparing a large collection of patterns to a large collection of objects. It finds all the combinations of objects which match with a conjunction of patterns. This algorithm is widely used to improve the run-time performance of rule-based expert systems. In this paper we employ generating functions theory in order to perform a precise average case complexity analysis of Rete algorithm. Our results are first established under a “simple” random term model, and later extended to take into account different frequency coefficients for symbols in a way that can closely model real-life applications.

1. INTRODUCTION

The Rete pattern match algorithm [Forg 79] [Forg 82] has been introduced by C. Forgy in the line of his work on production systems [Forg 81]. Production systems, or more generally rule-based systems, are widely used in Artificial Intelligence for modelling intelligent behaviour [LNR 87] and building expert systems [HVL 83].

In an expert system the knowledge needed to solve the problem is represented in a declarative form and is separated from the inference engine that exploits that knowledge. Descriptive knowledge on the objects that have to be modelled can be encoded using object-oriented programming paradigms [SBM 83], while deductive knowledge involved in problem solving is most likely represented using rule-oriented programming paradigms.

In a rule-based system each rule represents an independent piece of knowledge. Rules can express actions to take in a given situation or represent logical implications allowing to build deductions. Unlike statements in a traditional program, rule actions do not explicitly invoke other rules. Instead they modify the set of objects and logical assertions, having as effect to modify the set of satisfied rules. In this way rule-based computation is data-driven. The most time consuming process in a rule-based system is the pattern match phase that consists in maintaining the set of satisfied rules among changes in the data base. This computation can represent more than 90% of the overall computation time in an application.
Rete algorithm is an efficient method for computing the set of satisfied rules incrementally after each rule execution. The incremental computation is justified in expert systems applications by the fact that the execution of a rule affects a relatively small number of objects in comparison to the total number of objects. Therefore most of the previous pattern matching work remains valid. Rete algorithm realizes a total indexing of the database according to rule conditions. Conditions common to several rules are shared in such a way that several rules can be found to be satisfied by testing only once some patterns. This feature marks a significant difference with compilation techniques used in other rule-based languages, such as Prolog for instance [War 85].

With Rete algorithm the worst case complexity for computing the set of satisfied rules is linear as a function of the number of rules, and polynomial in the number of objects (with degree being the maximum number of conditions in a rule) [Forg 79]. In the best case the time complexity is a constant. Between these extremes the sensitivity of pattern matching time to the size of the database is highly dependent on rule characteristics.

There are many motivations in searching for a finer model of computation and an average case complexity analysis of Rete algorithm. First, Rete algorithm admits many variants and optimizations, concerning the representation of local memories [Forg 79], the sharing of conditions [Gha 87], the computation of joins [Mir 87], the total compilation principle [Fag 86], the parallelization of the algorithm [Gupt 84], etc... An average case complexity analysis can be used to evaluate these optimizations and propose new ones based on the statistical model. Second, run-time performance prediction is a necessity for the development of real-time expert systems [WGF 86], [SF 88]. A mathematical model of run-time requirements can be used to extrapolate from the run-time performance of a prototype the performances of the expert system in real size, and define the range of its applicability in terms of number of objects that can be treated at a given time. Third, a mathematical model can be used also to conceive significant benchmarks in order to compare several implementations according to the relevant characteristic parameters of a knowledge base [GF 83].

In this paper we present an average case complexity analysis of Rete algorithm. To our knowledge, it is the first time that is analysed in a rigorous and complete way this kind of algorithm in use in AI. Moreover, the mathematical methods developed in this article are not only interesting for the results they provide. The average case complexity analysis under the model of simple families and the weighted model introduce mathematical methods of independent interest.

We restrict ourselves to the model of term trees (or expression trees) for representing objects and assertions, and terms with variables for representing patterns. In this way only equality tests are considered. In section 2, we present the Rete algorithm in this framework. Then in section 3, we introduce generating functions theory that is employed to analyse the average case complexity of algorithms. We obtain a result under the simple family model (3.4). In section 4, we present results obtained by considering different frequency coefficients for symbols. Lastly in section 5 we extend the results by taking into account negation.

2. THE RETE MULTI-PATTERN MATCH ALGORITHM

In the sequel the production systems we shall consider are composed of a fixed set, denoted by RB (for Rule Base), of if-then rules called productions, and a changing set of facts, called the Working Memory and denoted by WM. Facts are formalized as term trees formed on a finite alphabet F of function symbols given with their arity. Constants are symbols of arity 0. For instance, given symbols f of arity 2, g of arity 1 and a constant a, one can form the following terms :