Average case complexity analysis of the RETE multi-pattern match algorithm

Luc Albert, Francois Fages

INRIA

UNITE DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tel (1) 39 63 55 11

Rapports de Recherche

N°773

AVERAGE CASE COMPLEXITY ANALYSIS OF THE RETE MULTI-PATTERN MATCH ALGORITHM

Luc ALBERT
François FAGES

DECEMBRE 1987

AVERAGE CASE COMPLEXITY ANALYSIS OF THE RETE MULTI-PATTERN MATCH ALGORITHM

LUC ALBERT[1] and FRANÇOIS FAGES[2]

Abstract. *The* RETE *multi-pattern match algorithm [For 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.*

ANALYSE EN MOYENNE DE L'ALGORITHME RETE DE SEMI-UNIFICATION MULTI-MOTIFS

Résumé. *L'algorithme* RETE *de semi-unification multi-motifs est un algorithme efficace pour comparer un grand nombre de motifs avec un grand nombre d'objets. Il produit toutes les combinaisons d'objets qui s'unifient avec une conjonction de motifs. Cet algorithme est très utilisé pour améliorer les performances des systèmes experts à base de règles. Dans cet article, nous employons la théorie des séries génératrices afin de déterminer un coût en moyenne précis de l'algorithme* RETE*. Nous établissons tout d'abord des résultats avec un modèle de termes aléatoires, puis nous étendons ce modèle afin de prendre en compte les différentes fréquences d'apparition des symboles de façon à modéliser fidèlement les applications réelles.*

[1] Institut National de Recherche en Informatique et Automatique, Domaine de Voluceau, Rocquencourt, BP 105, 78150 Le Chesnay Cedex France

[2] Laboratoire Central de Recherches, Thomson-CSF, Domaine de Corbeville, BP 10, 91401 Orsay Cedex France

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 [HWL 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 at 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 comparaison to the total number of objects. Therefore most of the previous pattern matching work remains valid. RETE algorithm realizes a total indexing of the data base 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) [For 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 data base 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 [For 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 first 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 :
a, $(g\ a)$, $(f\ a\ a)$, $(f\ (g\ a)\ a)$, etc ... The set of term trees is denoted by $T(F)$. The Working Memory is formalized as a tuple of such terms.

The if-part of a rule (its *left-hand side*) is a conjonction of *patterns*, represented as a tuple $(P_1, \ldots, P_n)$. A pattern is a term tree that can contain variables. Variables are denoted by $x, y, \ldots$, they are taken from an enumerable set of variables V. The set of terms formed on F and V is denoted by $T(F,V)$. Patterns are partial descriptions of facts. A fact t *matches* a pattern P if one can find a *substitution* of pattern's variables, $\sigma : V \rightarrow T(F)$, such that $\sigma t = P$. For example the substitution of x by a in pattern $(f\ (g\ x)\ x)$ matches the term $(f\ (g\ a)\ a)$.

We say that the left-hand side of a rule $(P_1, \ldots, P_n)$ is *instanciated* (or that the rule is *satisfied*) when there exists a tuple of facts $(t_1 \ldots, t_n)$ with $t_i \in WM$, called the instance, and a substitution σ such that $\sigma P_i = t_i$. We remark that since a pattern in a rule can match several facts in the working memory, a rule can be instanciated in multiple ways. The then-part of a rule (its *right-hand side*) is a sequence of *actions* that can have for effect to add or suppress a fact in (from) the working memory.

The cycle of inference consists in three steps:

- *match* the patterns with the facts in the working memory in order to determine the set of satisfied rules (called the *conflict set*);
- *select* one rule 's instance in the conflict set;
- *execute* the actions of the rule.

In RETE algorithm the pattern matching step is not separated from the execution step. Instead it occurs at each modification in the working memory, that is at initialisation time when facts are entered, and then at each assertion or retraction during execution steps. Since the RETE multi-pattern match algorithm is solely concerned with the left-hand sides of rules, from now on we shall identify rules to their left-hand sides. Rules are compiled in a discrimination network. At run-time when an assertion or a retraction is done, it is processed in the network from the root. If the fact matches a pattern it is memorized in (or removed from) the network. If other memorized facts can jointly satisfy a rule, the instance is added to (or suppressed from) the conflict set. In this way both rules and facts are represented in the network.

Two types of tests are distinguished in rule left-hand sides:

- ***mono-pattern tests*** are tests concerning solely the fact which matches the pattern. They test the equality of symbols inside the fact and the pattern, or inside the fact only. The later case arises when a variable has several occurrences in the same pattern.
- ***multi-pattern tests*** are tests involving subterms of two or more facts. They correspond to different occurrences of the same variable in different patterns.

In RETE pattern-match algorithm, mono-pattern tests are executed first, then multi-pattern tests are executed at each join with memorized facts to verify consistent binding of variables across multiple patterns in a rule 's left-hand side. Consequently the network is formed of two parts (see figure 1).

The initial part is a tree composed of mono-pattern tests, called the ***discrimination tree***. We define the ***i-pattern*** of a node i as the pattern corresponding to the tests cumulated on the path from the root, called the ***i-branch***. The root-pattern is a variable, *i.e.* a non-selective pattern (it matches any fact). The i-pattern of the leaves of the discrimination tree are all the patterns which appear in the rule left-hand sides, upto variable renaming. Several successors to a node correspond to several branches to follow (by backtracking). At run-time one fact typically reaches several leaves which correspond to different matching patterns.

The second part is a graph composed of binary joins between the leaves of the initial part. In these nodes, i-patterns are tuples of patterns $(P_1, \ldots, P_l)$, in which the sharing of variables between patterns is determined by the multi-pattern tests. Outputs are in a one-to-one correspondence with the rules. i-pattern of output nodes are the left-hand sides of the rules upto variable renaming. They are reached at run-time with the instances of the rules.

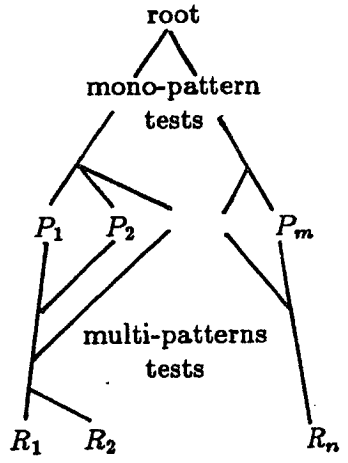
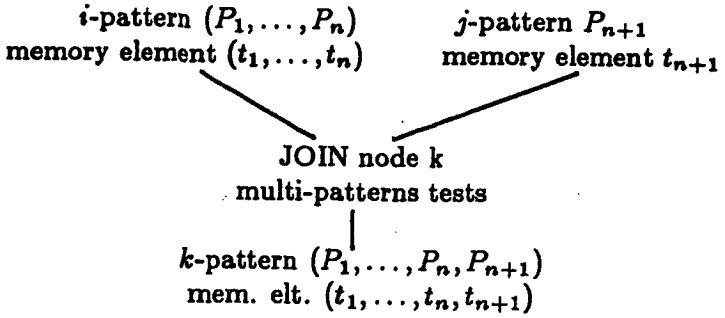


FIGURE 1 †: a RETE-graph

At run-time when a tuple reaches a join node, it is memorized in a ***local memory***, and the memory of the opposite input of the join node is searched in order to find a compatible tuple according to the multi-pattern tests (see figure 2). In many implementations the right input of a join node is restricted to be a leaf of the initial tree (not the output of a join node) as in figure 2 and 3, that eliminates a possibility of optimization consisting in grouping together the most selective patterns together.

The time complexity of the computation of the output of a join node is quadratic in the size of its input memories. In the worst case the size of the leaves of the initial tree is proportional to the size of the working memory $|WM|$. Therefore in the worst case the time complexity of

† m is the number of different patterns in rule left-hand sides, and n is the total number of rules

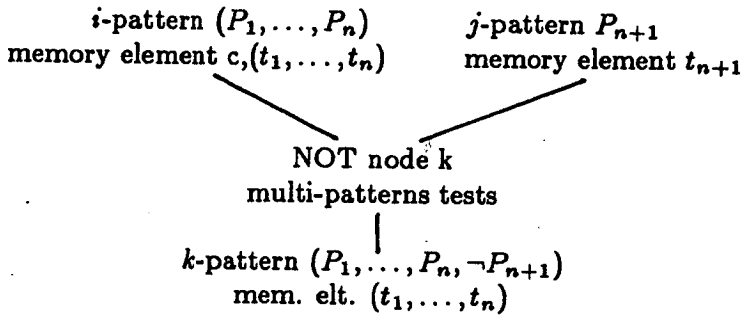


Join node for pattern P_{n+1} in a left-hand side of the form $P_1,\ldots,P_n,P_{n+1},\ldots$
FIGURE 2

the computation of the conflict set is a polynomial in function of $|WM|$, with degree being the maximum number of patterns in a rule.

Hashing techniques can be used to reduce the complexity of the search in local memories, by exploiting all equality tests in a join in order to hash the input memories. In the same way if inequality predicates are to be considered, the local memories can be organized in search trees.

Negation of patterns has not been considered yet. A negated pattern expresses the non-existence of a fact matching the pattern. Negated patterns are treated in RETE algorithm by adding a second type of join nodes. In these nodes the local memory on the left input stores with each tuple the number of elements in the right memory (the negated elements) which are compatibles according to the multi-pattern tests (see figure 3). When a counter gets to zero (resp. one) the left tuple is propagated in the network in assert (resp. retract) mode. The existential quantifier can be treated in a symmetrical way. In fact it is possible to generalize RETE algorithm in order to accept in the left-hand sides of rules any first order logic formula with an arbitrary degree of imbrication of quantifiers.



Join node for pattern $\neg P_{n+1}$ in a left-hand side of the form $P_1,\ldots,P_n,\neg P_{n+1},\ldots$
FIGURE 3 †

3. THE COST OF THE ALGORITHM

In this section, we derive the average cost of RETE algorithm and the evaluation of the average size of data in the nodes of the RETE network.

† c is the counter of facts t_{n+1} which satisfy the multi-pattern tests with $t_1,\ldots,t_n$

3.1 THE SIMPLE FAMILY MODEL

In the simple family model we consider the usual tree structure of terms. The patterns are therefore considered as trees with leaves that can be variables. In [SF 83] [Stey 84] it has been proved that the average cost to unify a pattern of size p to a term of size n is constant. In the case of RETE algorithm we have to consider the semi-unification with a conjunction of patterns.

As we said, the Working Memory is represented by any tuple of such terms, and that is called a *forest*. We will thus evaluate in this section the average number of these terms or l-tuples of terms that match at a node of the RETE network. In order to find the average time of determination of the Conflict Set, we will use as measure the number of multi-patterns tests made at a join node.

We shall now detail the representation of data. We have chosen to represent terms under the form of trees constructed from a family F of finite cardinal of symbols with fixed arities (constants are symbols with arity 0): it is the formalism of *simple family* (See [Stey 84] [Fla 85] [FS 86]).

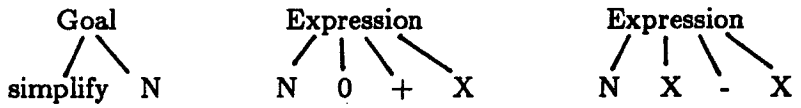
Example : Considering the following rules that express simplifications of mathematical expression:

$$Plus0x: \quad (Goal\ simplify\ N)\ \ (Expression\ N\ 0\ +\ X) \longrightarrow \ldots$$

$$Minusxx: \quad (Goal\ simplify\ N)\ \ (Expression\ N\ X\ -\ X) \longrightarrow \ldots$$

(capitalized letters stand for variables)

we can represent all the distinct condition patterns with



and it leads to the following RETE network:

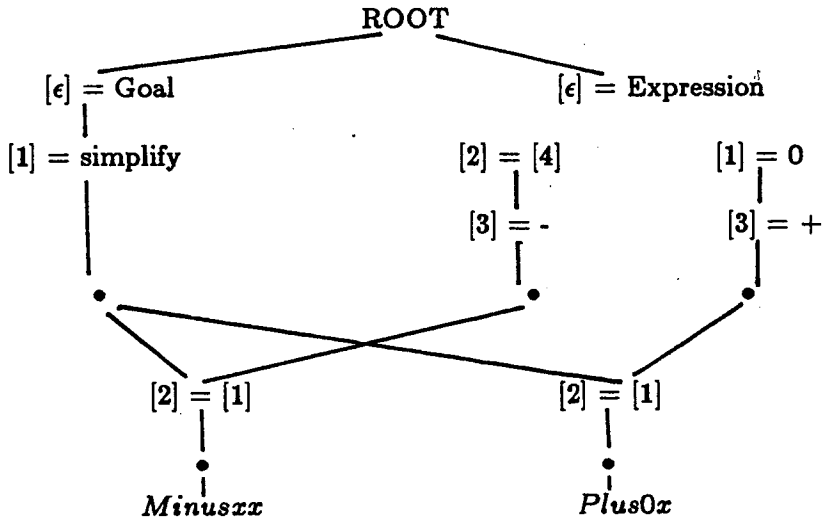


FIGURE 4 : RETE graph

We denote the nodes of the trees by a sequence of integers (for example $[m.n]$ denotes the n^{th} leftmost son of the m^{th} leftmost son of the root of the tree).

Let a_n denote the number of terms of size n belonging to a simple family (the size of a term is the number of nodes it is made of). We associate to the simple family the generating function:

$$A(z) = \sum_{n \geq 0} a_n z^n = \sum_{t \in T(F)} z^{|t|}$$

with $|t|$ the size of t and $T(F)$ the set of terms built with the symbols of F. The use of generating function is very effective, many operations on combinatorial objects can be systematically transposed to equations on generating functions (cartesian product, union ...) (See [Fla 87]).

Example : Consider the simple family of binary trees. It has two symbols: a constant $-c-$ and a symbol with arity 2: $- \bullet -$ (think of "cons" in Lisp). We can perform the computation of $A(z)$ by case analysis on terms:

$$A(z) = \sum_{t=c} z^{|t|} + \sum_{t=t_1.t_2} z^{|t_1.t_2|}$$

Since $|t_1.t_2| = 1 + |t_1| + |t_2|$:

$$A(z) = z + z \sum_{t_1 \in T(F)} \sum_{t_2 \in T(F)} z^{|t_1|} z^{|t_2|} = z(1 + A(z)^2)$$

In the general case, if c_k is the number of symbols with arity k in F and $\Phi(X)$ the polynomial $\Phi(X) = \sum_{k=0}^{p} c_k X^k$ (with p the greatest arity among the symbols) then the generating function $A(z)$ verify the functional equation:

$$A(z) = z\Phi(A(z))$$

This classical combinatorial result expresses a systematic transposition of the structure of terms to the functional equation of the generating function. One can find a proof of this result in [SF 83] and in [MM 78]. We shall assume about Φ that:

- $\exists\ c_k > 0$ with $k \geq 2$, *i.e.* there exists a function symbol of arity ≥ 2.
- and there exists no $\Psi(X)$ polynomial and d integer $d \geq 2$ such that $\Phi(X) = \Psi(X^d)$ (H.C.F condition). This condition implies the existence of one unique solution τ to the fundamental equation: $\Phi(\tau) = \tau\Phi'(\tau)$, τ real and positive. This hypothesis is verified as soon as there is in F an unary symbol. Nevertheless, the existence of such a d is not embarrassing, the following calculations are identical. d appears as a multiplicative factor in some results but disappears as soon as it is an average calculation (See [Stey 84]), therefore the main results are unchanged.

3.2 EXACT COMBINATORIAL FORMULAE

We consider that the RETE algorithm takes as input any given Working Memory and we shall determine the average quantity of terms or of l-tuple of terms (*i.e.* a list of l terms) that match at a node i of the RETE network. We are doing an average calculation over all the possible input-Working Memory.

In this subsection, we shall obtain exact combinatorial expressions of which we shall give in subsection 3.3 a simplified expression using asymptotic analysis over the size of the input-Working Memory. We shall consider a forest of k terms of total size n (*i.e.* n is the sum of the sizes of terms composing the Working Memory). The generating function of a forest of k terms of a simple family determineed by $A(z)$ is given by:

$$F(z) = A(z)^k$$

Thus $[z^n]A(z)^k$ represents the number of forests of size n with k elements.

For a node i of the discrimination tree, we define: $B^i(z) = \sum_{m \geq 0} b_m^i z^m$ with b_m^i the number of terms of size m that match at the node i of the discrimination tree.

For a node i of the join network that outputs a list of l-tuples that partially instanciates a left-hand side of a rule, we define: $B^i(z) = \sum_{m \geq 0} b_m^i z^m$ with b_m^i the number of l-tuples of size m that match this i join node.

For a node i of the discrimination tree, we call i-branch of the discrimination tree, the test branch of the tree that goes from the root to i included. It determines thus the i-pattern and we denote:

- h_i the height of the node i in the discrimination tree ($h(root) = 0$)
- ω_i the arity of the symbol of the family F which is tested at the node i
- $\Omega_i = \sum_{j=root}^{i} \omega_i$
- x_j^i = number of j-tuple ($j \geq 2$) of identical variables tested from the root to i included
- $X_i = \sum_j x_j^i$
- $Y_i = \sum_j (j-1)x_j^i$ = number of equality tests between two variables that have to be done for the i-pattern

We thus find on the i-branch Y_i test nodes testing the equality of two variables of the i-pattern and $P_i = h_i - Y_i$ nodes testing a non variable-symbol of the i-pattern. We can pose $P_i = |i\text{-pattern}|$ and we have $\Omega_i - h_i + 1$ variable leaves in the i-pattern.

We can now determine $B^i(z)$:

THEOREM 1 : *The generating function of the terms which match at a node i of the discrimination tree is:*

$$B^i(z) = z^{P_i} \prod_{j=1}^{\Omega_i - h_i + 1} A^{x_j^i}(z^j)$$

with $x_1^i = \Omega_i - h_i + 1 - X_i$ that represents the number of distinct variables in the i-pattern.

Example : Let us consider the following condition-pattern:

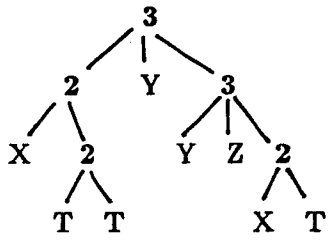


FIGURE 5

in which we have represented variables with capital letters and indicated the symbols of F only with their arity (bold-faced numbers).

In this example the tests of the i-branch are:

$$\begin{cases} [](i.e.\ root) = 3 \\ [1] = 2 \\ [3] = 3 \\ [1.2] = 2 \\ [3.1] = [2] \\ [3.3] = 2 \\ [1.2.1] = [1.2.2] \\ [3.3.1] = [1.1] \\ [3.3.2] = [1.2.1] \end{cases}$$

We have indicated the nodes of the pattern with the previous numbering . Thus we get:

$$h_i = 9; x_1^i = 1; x_2^i = 2; x_3^i = 1 \quad \text{next } x_j^i \text{ are null}$$

$$X_i = 3; Y_i = 4; \Omega_i = 12 \quad \text{and} \quad P_i = 5$$

Therefore the generating function of the terms that match with this pattern is:

$$B^i(z) = z^5 A(z)\ A(z^2)^2 A(z^3)$$

PROOF OF THE THEOREM: We have $B^i(z) = \sum_{t \in T_i} z^{|t|}$ with T_i the set of terms matching at i. Let $t \in T_i$. We have

$$|t| = P_i + \sum_{j=1}^{\Omega_i - h_i + 1} j \left(\sum_{k=1}^{x_j^i} |t_{j,k}| \right)$$

$t_{i,j}$ stands for one of the subtrees which are j-times duplicated in t.
Thus

$$B^i(z) = \sum_{t \in T_i} z^{|t|} = z^{P_I} \prod_{j=1}^{\Omega_i - h_i + 1} \left(\prod_{k=1}^{x_j^i} \sum_{t_{j,k}} z^{j|t_{j,k}|} \right)$$

Since t is arbitrary, all the $t_{j,k}$ are arbitrary and therefore we have:

$$B^i(z) = z^{P_I} \prod_{j=1}^{\Omega_i - h_i + 1} \left(\prod_{k=1}^{x_j^i} (\sum_{t \in T} z^{j|t|}) \right) = z^{P_I} \prod_{j=1}^{\Omega_i - h_i + 1} A^{x_j^i}(z^j) \quad \blacksquare$$

Remark : Hence $B^{root}(z) = A(z)$.

Let us consider now a node i of the join network. i outputs lists of l-tuples. For a given i we thus know the l patterns which match each of the components of the output-l-tuples (without considering the tests on similar variables between the different patterns).L_i stands for this set of l-patterns. Then, let us denote:

- $P_i = \sum_{p \in L_i} P_p$
- $\Omega_i = \sum_{p \in L_i} \Omega_p$
- $h_i = \sum_{p \in L_i} h_p$
- for $j \geq 1$, $x_j^i = \left(\sum_{p \in L_i} x_p^i \right) + c_j^i$

where c_j^i represents a correcting term from the appearance of identical variables between the l-patterns. *If we consider the global set of all the variables of the different l-patterns, x_j^i represents still the number of j-tuples of identical variables in this set.* We determine thus easily the c_j^i in function of the similar variables tested at the node i and we have: $c_1^i \leq 0$ and $\forall\ j \geq 2,\ c_j^i \geq 0$ because the join increase *a priori* the number of identical variables.

From this, we prove in the same way as previously:

THEOREM 2 : *The generating function of the l-tuples of terms which match at node i in the join network is:*

$$B^i(z) = z^{P_i} \prod_{j=1}^{\Omega_i - h_i + l} A^{x_j^i}(z^j)$$

From this, let us deduce now $\overline{b_{n,k}^i}$ which stands for the average number of terms or of l-tuples of terms which match at i considering as input of the RETE algorithm any forest of k terms with total size n.

THEOREM 3 : *For a node i of the discrimination tree, the average number of terms which match at i is:*

$$\overline{b_{n,k}^i} = \frac{k[z^{n-P_i}]\prod_{j=1}^{\Omega_i - h_i + 1} A^{\nu_j^i}(z^j)}{[z^n]A(z)^k}$$

with $\nu_1^i = k + \Omega_i - h_i - X_i$ and for $j \geq 2$, $\nu_j^i = x_j^i$.

PROOF : The basic idea is to split the generating function $A(z)$ in: $A(z) = B^i(z) + C^i(z)$ with $C^i(z)$ the "rest function" *i.e.* the generating function of all the terms which don't match with the i-pattern. Then we shall mark the matching at i elements with the variable u and note:

$$A(z,u) = uB^i(z) + C^i(z) = A(z) + (u-1)B^i(z)$$

Thus $f_{n,p,k}^i = [u^p z^n](A(z,u))^k$ represents the number of forests of size n with k terms among which there is exactly p terms matching at node i.
Therefore $\overline{b_{n,k}^i}$ is the quotient of $\sum_{p=0}^{+\infty} p f_{n,p,k}^i \left(= \sum_{p=0}^{k} p f_{n,p,k}^i\right)$ by the total number of forests of size n with k terms, thus:

$$\overline{b_{n,k}^i} = \frac{\sum_p p f_{n,p,k}^i}{[z^n]A(z)^k}$$

We have $(A(z,u))^k = \sum_{p,n} f_{n,p,k}^i u^p z^n$ hence

$$\frac{\partial}{\partial u}\left(A(z,u)^k\right) = \sum_{p,n} f_{n,p,k}^i p u^{p-1} z^n$$

hence

$$\frac{\partial}{\partial u}\left(A(z,u)^k\right)\Big|_{u=1} = \sum_n \left(\sum_p p f_{n,p,k}^i\right) z^n$$

Therefore

$$\overline{b_{n,k}^i} = \frac{[z^n]\frac{\partial}{\partial u}\left(A(z,u)^k\right)|_{u=1}}{[z^n]A(z)^k}$$

And

$$\frac{\partial}{\partial u}\left(A(z,u)^k\right)\Big|_{u=1} = kA(z,u)^{k-1}\frac{\partial}{\partial u}\left(A(z,u)\right)\Big|_{u=1}$$
$$= kA(z)^{k-1}B^i(z)$$

And the Theorem is established. ∎

THEOREM 4 : *With the previous notations for a node i of the join network, the average number of l-tuples of terms which match at i is:*

$$\overline{b^i_{n,k}} = \frac{k^l[z^{lk^{l-1}n-P_i}]\prod_{j=1}^{\Omega_i-h_i+l}A^{\nu^i_j}(z^j)}{[z^{lk^{l-1}n}]A(z)^{lk^l}}$$

with $\nu^i_1 = l(k^l-1)+x^i_1$ and for $j \geq 2$, $\nu^i_j = x^i_j$.

PROOF : A forest of size n with k terms f yields to k^l l-tuples of elements of f. Among these l-tuples we have got to determine the ones which match at i. If these are uniformly distributed in the k^l-lists of any l-tuples, considering that f yields to an k^l-list of l-tuples of arbitrary terms of total size $lk^{l-1}n$, we obtain the Theorem with the same reasoning as before. ∎

Notice that the results we obtain are expressed as a function of the characteristics of the RETE network that is considered as a fixed data. These results will thus enable an optimization of the creation of the RETE network.

Those exact results can be studied using the Lagrange inversion theorem but they lead to expressions that are hard to use (multinomial coefficient ...). In order to express them simply we shall derive an asymptotic expansion with respect to k and n.

3.3 ASYMPTOTIC EVALUATION

To derive simple expressions, we make use of singularity analysis methods. We estimate the value of integrals with complex analysis methods (saddle-point method). We consider a Working Memory of size n with k terms as a given data of the algorithm. Thus we consider that k and n are both increasing.

According to previous section, we have to evaluate coefficients such as: $[z^n]\prod_{j=1}A^{\nu^i_j}(z^j)$. We shall first evaluate $[z^n]A^k(z)$, then we will prove how one can deduce the expression of $\overline{b^i_{n,k}}$.
Let $a^k_n = [z^n]A^k(z)$. By Cauchy's theorem this quantity can be expressed as the following integral

$$a^k_n = \frac{1}{2i\pi}\int_\Gamma \left(A(z)^k\right)\frac{dz}{z^{n+1}}$$

where the contour Γ lies inside the domain of analycity of A and simply encircles the origin. In order to get an equivalent of this integral when k and n tend both to infinity, we will use the saddle point method (See [dB 58] and [Henr 74]). Before that, we shall find a relationship between n and k in simple families. With classical combinatorial methods, we obtain the following result:

THEOREM 5 : *Given n, the average number of trees $\bar{k}_n$ in a forest of size n is:*

$$\bar{k}_n = \frac{[z^n]\left(\frac{A(z)}{(1-A(z))^2}\right)}{[z^n]\left(\frac{1}{1-A(z)}\right)}$$

Let us repeat the notations which will be used in the following:

PROPOSITION: *Let us consider a simple family. We denote $A(z)$ its generating function, as we have seen before it verifies*

$$A(z) = z\Phi(A(z))$$

We note τ the smallest positive root of the equation

$$\Phi(X) = X\Phi'(X)$$

(uniqueness being guaranteed by HCD condition) and $\rho = \tau/\Phi(\tau)$ is the radius of convergence of $A(z)$. We have $0 < \rho < 1$ and $\tau = A(\rho)$.

With these notations, we have in the neighbourhood of $z = \rho$ (See [MM 78])

$$A(z) = \tau - \sqrt{\frac{2\Phi(\tau)}{\Phi''(\tau)}}\left(1-\frac{z}{\rho}\right)^{\frac{1}{2}} + \gamma\left(1-\frac{z}{\rho}\right) + O\left(\left(1-\frac{z}{\rho}\right)^{\frac{3}{2}}\right)$$

From which it follows

THEOREM 6 : *Considering a forest of terms of this simple family. The average number $\bar{k}_n$ of terms in a forest of fixed size n is:*

1) If $\tau < 1$

$$\bar{k}_n = \frac{1+\tau}{1-\tau}\left(1+O\left(\frac{1}{n}\right)\right)$$

2) If $\tau = 1$

$$\bar{k}_n = \sqrt{\frac{\pi\Phi''(1)}{\Phi(1)}}n^{\frac{1}{2}}\left(1+O\left(\frac{1}{\sqrt{n}}\right)\right)$$

3) If $\tau > 1$ $\exists\, \omega < \rho$ / $A(\omega) = 1$ $(\omega = \Phi(1)^{-1})$ et

$$\bar{k}_n = \underbrace{\frac{1}{2}\left(1-\frac{\Phi'(1)}{\Phi(1)}\right)}_{<1} n\left(1+O\left(\frac{1}{n^2}\right)\right)$$

The size of terms is approximatively constant in implementations *i.e.* $\bar{k}_n$ is proportional to n. Thus the third case fits well on reality. With regard to theory it is the case where $\Phi(0)$ the number of constants is large (it is thus easy to verify that the constant of proportionality between k and n in this case is < 1).

We shall now evaluate a_n^k in the three following cases:

1) $k = \alpha + \mu/n$ and $\tau < 1$
2) $k = \lambda\sqrt{n} + \mu$ and $\tau = 1$
3) $k = An + C + \mu/n$ and $\tau > 1$

(For more details on the following calculations, see [Alb 87], or [SF 83] where a similar evaluation is developed in the case where $k \leq \sqrt{n}/\log^3 n$).

Let us succinctly explain the principles of the calculations. We have

$$a_n^k = \frac{1}{2i\pi}\int_\Gamma (A(z)^k)\frac{dz}{z^{n+1}}$$

We shall perform a change of variable making $y = A(z)$ the independent variable. With the functional equation, a_n^k becomes:

$$a_n^k = \frac{1}{2i\pi}\int_C \Phi^n(y)\left(1 - y\frac{\Phi'(y)}{\Phi(y)}\right)\frac{dy}{y^{n-k+1}}$$

with C the image of Γ under z. C can be taken as an arbitrary contour around 0 inside the domain of analicity of Φ. We shall use the *saddle point* method. In the first and the second case, we shall choose as contour of integration the following circle:

$$C = \{z/|z| = r\}$$

This choice is motivated by the fact that in these cases r is a saddle-point of the "main" part of the integrand as n gets large, namely, of $(\Phi(y)/y)^{n-1}$.

In the third case, we will choose as saddle-point σ the smallest real positive root of the fundamental equation:

$$y\Phi'(y) = (1 - A)\Phi(y)$$

Then, the basic idea of the calculation is to split this integral into two parts:

$$a_n^k = \int_C = \int_{C1} + \int_{C2} = I_1 + I_2$$

where $C1 = \{z/|z| = r \text{ and } -\epsilon \leq Arg(z) \leq \epsilon\}$ and $C2 = C/C1$ taking $\epsilon = \log n^2/\sqrt{n}$. This choice is made so that on the one hand the integral along $C2$ is negligible being exponentially small, and on the other hand ϵ is small enough so that local expansions on the integrands can be performed to estimate the integral along $C1$.

From these calculations we obtain the following Theorems:

THEOREM 7 : *With* $\bar{k}_n = \alpha\left(1 + O\left(\frac{1}{n}\right)\right)$ *(case where* $r < 1$*):*

$$\begin{aligned} a_n^k &= [z^n](A(z)^k) \\ &= \sqrt{\frac{\Phi(r)}{2\pi\Phi''(r)}}\rho^{-n}r^{\alpha-1}\alpha n^{-\frac{3}{2}}\left(1 + O\left(\frac{1}{n}\right)\right) \end{aligned}$$

THEOREM 8 : *With* $\bar{k}_n = \lambda\sqrt{n} + O(1)$, *and* $r = 1$, *we have*

$$a_n^k = \frac{\Phi(1)^n}{2}e^{-\pi}n^{-1}\left(1 + O\left(\frac{1}{\sqrt{n}}\right)\right)$$

THEOREM 9 : *With* $\bar{k}_n = An + C + O\left(\frac{1}{n}\right)$, $r > 1$, *we have*

$$a_n^k = \frac{A}{4\sqrt{\pi}}\left(\frac{\sigma^2\Phi''(\sigma)}{2\Phi(\sigma)} + \frac{A(1-A)}{2}\right)^{-\frac{1}{2}} \sigma^{-n(1-A)}\Phi(\sigma)^n n^{-\frac{1}{2}}\left(1 + O\left(\frac{1}{n}\right)\right)$$

with σ *the smallest positive root of the equation*

$$y\Phi'(y) = (1-A)\Phi(y)$$

With these results, choosing the case closest to reality, we are now able to perform the evaluation of the denominator of previous expression of $\overline{b_{n,k}^i}$. For the numerator we will use the following lemma that comes from the previous calculations:

LEMMA : *Let* ρ *be the radius of convergence of* $A(z)$. *If* $g(z)$ *is analytic in the neighbourhood of* ρ *then, in the three previous cases, we have:*

$$[z^n]\{A^k(z)g(z)\} = g(\rho)[z^n]A^k(z)$$

The radius of convergence of $A(z^j)$ is $\rho^{\frac{1}{j}}$. As $\rho < 1$, we get $\rho = min\left\{\rho^{\frac{1}{j}}\ ,\ j \geq 1\right\}$, hence

$$[z^n]\prod_{j=1} A^{\nu_j^i}(z^j) = \prod_{j\geq 2} A^{\nu_j^i}(\rho^j)\left([z^n]A^{\nu_1^i}(z)\right)$$

Then, the calculations are almost immediate and we obtain the Theorem:

THEOREM 10 : *Let us consider a forest of size* n *with* k *terms with* $\bar{k}_n = \frac{1}{2}\left(1 - \frac{\Phi'(1)}{\Phi(1)}\right)n\left(1 + O\left(\frac{1}{n^2}\right)\right)$ *(case where* $r > 1$*), we have:*

- *for a node* i *of the discrimination tree:*

$$\overline{b_{n,k}^i} = \Pi_i n\left(1 + O\left(\frac{1}{n}\right)\right)$$

$$\text{with} \qquad \Pi_i = \frac{1}{2}\left(1 - \frac{\Phi'(1)}{\Phi(1)}\right)\left(\frac{\sigma^{\frac{1}{2}\left(1+\frac{\Phi'(1)}{\Phi(1)}\right)}}{\Phi(\sigma)}\right)^{h_i - Y_i}\left(\prod_{j=2}^{\Omega_i - h_i + 1} A^{\nu_j^i}(\rho^j)\right)$$

- *for a node* i *of the join network:*

$$\overline{b_{n,k}^i} = \Pi_i n^l\left(1 + O\left(\frac{1}{n^2}\right)\right)$$

$$\text{with} \qquad \Pi_i = \frac{1}{2}\left(1 - \frac{\Phi'(1)}{\Phi(1)}\right)^l\left(\frac{\sigma^{\frac{1}{2}\left(1+\frac{\Phi'(1)}{\Phi(1)}\right)}}{\Phi(\sigma)}\right)^{h_i - Y_i}\left(\prod_{j=2}^{\Omega_i - h_i + l} A^{\nu_j^i}(\rho^j)\right)$$

with σ *the smallest positive root of the equation*

$$y\Phi'(y) = \frac{1}{2}\left(1 - \frac{\Phi'(1)}{\Phi(1)}\right)\Phi(y)$$

We can now define a matching-rate $\overline{\alpha_n^i}$ at a node i of the RETE network which expresses the probability for a term or a l-tuple of terms to match at a node i of the RETE network. We get easily the following result:

THEOREM 11 : *For a node i of the* RETE *network, the matching-rate is*

$$\overline{\alpha_n^i} \simeq \left(\frac{\sigma^{\frac{1}{2}\left(1+\frac{\Phi'(1)}{\Phi(1)}\right)}}{\Phi(\sigma)} \right)^{h_i - Y_i} \left(\prod_{j=2}^{\Omega_i - h_i + l} A^{\nu_j^i}(\rho^j) \right)$$

l represents for a node i of the join network the number of components of the output-tuples and for a node i of the discrimination tree we pose $l = 1$.
(For the sake of the presentation, we shall keep this notation for the following results).

Remark : Notice that:

$$\frac{\sigma^{\frac{1}{2}\left(1+\frac{\Phi'(1)}{\Phi(1)}\right)}}{\Phi(\sigma)} \leq 1$$

and therefore Π_i dicreases rapidly with h_i.

3.4 RESULTS OF THE SIMPLE FAMILY MODEL

We can now evaluate the average cost of the RETE algorithm, given an arbitrary Working Memory and a fixed RETE network. We shall estimate the average time needed by the RETE algorithm to produce the Conflict Set from a given input-Working Memory. We determine the average number of multi-patterns tests realized at the join nodes (this is as we previously said the main part of the total time cost). Let us denote l_i the number of components of the output-tuples at a join node i and β_i the number of multi-patterns tests performed at i considering only one l_i-input-tuple. The average number of tests performed at i is the average number of l_i-tuples tested at i multiplied by β_i. To obtain the average number of l_i-tuples that inputs at i we just have to consider that at i no test is performed (*i.e.* $\forall j$, $c_j^i = 0$) and evaluate the so modified $\overline{b_{n,k}^{\prime i}}$. Let us denote $\Pi_i' n^{l_i}$ this quantity (notice that this quantity is precisely equal to $\overline{b_{n,k}^{i1}} \times \overline{b_{n,k}^{i2}}$ where $i1$ and $i2$ design the two nodes inputs of i).

Then, we can propose

$$\overline{cost_n} \simeq K \sum_{i \ join-nd} \beta_i \Pi_i' n^{l_i} = K \sum_{i \ join-nd} \beta_i (\Pi_{i1} \Pi_{i2}) n^{l_i} \qquad (1)$$

with K an implementation constant.

Of course asymptotically with respect to n, we get the main part of this expression keeping only the nodes with the larger l_i.

Remark : By hashing local memories according to equality tests at the join nodes, the average number of semi-unifications performed at a join node i is only the number of output-tuples *i.e.* $\overline{b_{n,k}^i}$. Thus in the above formula we just have to replace Π_i' by Π_i.

From this result, *that can be made fully precise on any case*, we can *a posteriori* assume Forgy's hypotheses (See [Forg 79]). Thus if we assume that the number of condition-patterns per left-hand side is in the Rule Base constant equal to c, the previous result becomes:

$$\overline{cost_{|WM|}} \simeq K A^c |WM|^c \left(\sum_{j=1}^{|RB|} \beta_j \Pi_j' \right) = K A^c |WM|^c \left(\sum_{j=1}^{|RB|} \beta_j \Pi_{j1} \Pi_{j2} \right)$$

with $j1$ and $j2$ denoting the two inputs of the last join node j of the Rj rule.
And we find the same type of relation proposed by Forgy: a linear cost as a function of the number of rules and a polynomial cost as a function of the size of the Working Memory $|WM|^c$ (we have determined the time needed by the RETE algorithm with an input of $|WM|$ modification terms, Forgy found a cost in $|WM|^{c-1}$ because he considered only one modification term as input).

4. WEIGHTED MODEL

4.1 INTRODUCTION

The model studied in the last section considers a uniform distribution over all trees of the same size n. In order to get an even more likely modelling, we shall consider a model taking into account the fact that for many applications certain symbols are more frequent than others. Thus we present here a probabilistic model called the *branching model* (See [FSS]).
Let F be a set of functional symbols, each with an *arity*. The corresponding set of terms T is defined in the usual way. Let w be a *weight function* that assigns to each symbol $f \in F$ a non negative real number $w[f]$. Then that weight is extended multiplicatively to trees. If t is a tree, its weight $w[t]$ is defined as

$$w[t] = \prod_{f \in t} w[f]$$

where the product is extended to all node symbols of t. We wish to use a probabilistic model for the symbols, then

$$\sum_{f \in F} w[f] = 1$$

Those weighted model are of interest for several reasons. By construction, they are likely to represent real-life situations better than the uniform models. Those models are also natural from a mathematical point of view since they are equivalent to assuming that the trees are generated by a *branching process* with a conditioning by the size n of the result. We still can associate with (F, w) a structural polynom: $\Phi_1(y) = \sum_{f \in F} w[f] y^{ari[f]}$. Then the generating function $A_1(z) = \sum_{t \in T} w[t] z^{|t|}$ still similarly verify: $A_1(z) = z\Phi_1(A_1(z))$. With this model $B^i(z)$ becomes:

THEOREM 12 : *For a node i of the* RETE *network, we have:*

$$B^i(z) = W_i z^{P_i} \prod_{j=1}^{\Omega_i - h_i + l} A_j^{\nu_j^i}(z^j)$$

with W_i the weight of the i-pattern i.e. for a node i of the discrimination tree $W_i = \prod_{f \in i-pat} w[f]$ and for a node i of the join network, since the i-pattern is a l_i-tuple of patterns, we pose $W_i = \prod_{j_i=1}^{l_i} W_{j_i}$ with W_{j_i} the weight of the j^{th} component of the i-pattern,
and with $A_j(z)$ that verify

$$A_j(z) = z\Phi_j(A_j(z)) \qquad with \qquad \Phi_j(y) = \sum_F w^j[f] y^{ari[f]}$$

We prove this Theorem with the same reasoning as before. We introduce the $A_j(z)$ generating functions from the appearance in the calculations of such series: $\sum_{t \in T} w^j[t] z^{j|t|} \quad (= A_j(z^j))$.

From this we deduce:

THEOREM 13 : *For a node i of the* RETE *network, with the weighted model the average number of terms which match at i is:*

$$\overline{b^i_{n,k}} = \frac{k^l W_i [z^{lk^{l-1}n - P_i}] \prod_{j=1}^{\Omega_i - h_i + l} A^{\nu^i_j}_j(z^j)}{[z^{lk^{l-1}n}] A(z)^{lk^l}}$$

There will be two cases for the following asymptotic evaluation:
all the $A_j(z)$ have distinct radius of convergence $\rho_j < 1$ (and τ_j) that can be determined by the same way as before. Either $\rho_1 = min\{\rho_j^{\frac{1}{j}} \ , \ j \geq 1\}$ and the results will be similar to the uniform model, or $\exists\, m \neq 1 \ / \ \rho_m^{\frac{1}{m}} = min\{\rho_j^{\frac{1}{j}} \ , \ j \geq 1\}$. In this case the evaluation of the denominator of $\overline{b^i_{n,k}}$ is of course unchanged but for the numerator the series with the minimal radius of convergence is $A^{\nu^i_m}_m(z^m)$ and the asymptotic evaluation will thus use the Theorem 7.

4.2 RESULTS WITH THE WEIGHTED MODEL

We obtain:

THEOREM 14 : *With the weighted model, let us consider a forest of size n with k terms with*

$$\bar{k}_n = \frac{1}{2}\left(1 - \frac{\Phi_1'(1)}{\Phi_1(1)}\right) n \left(1 + O\left(\frac{1}{n^2}\right)\right) = An\left(1 + O\left(\frac{1}{n^2}\right)\right)$$

we have:

- *if $\rho_1 = min\{\rho_j^{\frac{1}{j}} \ , \ j \geq 1\}$*
  - *for a node i of the discrimination tree:*

$$\overline{b^i_{n,k}} = \Pi_i n \left(1 + O\left(\frac{1}{n}\right)\right)$$

  - *for a node i of the join network:*

$$\overline{b^i_{n,k}} = \Pi_i n^l \left(1 + O\left(\frac{1}{n^2}\right)\right)$$

with in the two cases $\quad \Pi_i = W_i A^l \left(\frac{\sigma^{1-A}}{\Phi_1(\sigma)}\right)^{P_i} \left(\prod_{j=2}^{\Omega_i - h_i + l} A^{\nu^i_j}_j(\rho_1^j)\right)$

and the matching-rate $\quad \overline{\alpha^i_n} \simeq W_i \left(\frac{\sigma^{1-A}}{\Phi_1(\sigma)}\right)^{P_i} \left(\prod_{j=2}^{\Omega_i - h_i + l} A^{\nu^i_j}_j(\rho_1^j)\right)$

- *If $\exists\, m \neq 1 \ / \ \rho_m^{\frac{1}{m}} = min\{\rho_j^{\frac{1}{j}} \ , \ j \geq 1\}$*
  - *for a node i of the discrimination tree:*

$$\overline{b^i_{n,k}} = M_{i,n} \left(1 + O\left(\frac{1}{n}\right)\right)$$

- *for a node i of the join network:*

$$\overline{b^i_{n,k}} = M_{i,n}\left(1+O\left(\frac{1}{n^2}\right)\right)$$

with in the two cases

$$M_{i,n} = \frac{W_i}{l}\left(\prod_{\substack{j=1\\ \text{except } m}}^{\Omega_i-h_i+1} A_j^{\nu^i_j}(\rho^{\frac{j}{m}}_m)\right)\sqrt{\frac{m\Phi_m(\tau_m)}{2\pi\Phi''_m(\tau_m)}}\rho_m^{-\frac{lA^{l-1}n^l-P_i}{m}}\tau_m^{\nu^i_m-1}4\nu^i_m\sqrt{\pi}$$

$$\times\left(\frac{\sigma^2\Phi''_1(\sigma)}{2\Phi_1(\sigma)}+\frac{A(1-A)}{2}\right)^{\frac{1}{2}}\left(\frac{\sigma^{1-A}}{\Phi_1(\sigma)}\right)^{lA^{l-1}n^l}$$

All the constants in these expressions can be fully precised on an example. We know all the constants depending on the characteristics of the RETE network and on the characteristics of the weight model $(w, \Phi_j \ldots)$. And we have: τ_j the smallest positive root of $\Phi_j(y) = y\Phi'_j(y)$, that yields to the ρ_j $(\rho_j = \frac{\tau_j}{\Phi_j(\tau_j)} = \frac{1}{\Phi'_j(\tau_j)})$, and σ the smallest positive root of the equation $y\Phi'(y) = (1-A)\Phi(y)$, and $A_j(\rho^{\frac{j}{m}}_m)$ the limit of the sequence $u_0 = A(0) = 0$ and $u_{h+1} = \rho^{\frac{j}{m}}_m\Phi_j(u_h)$ (which converges geometrically). All these values are easy to determine with assistance of a symbolic manipulation program such as MAPLE program (See [MAPLE 85]). Therefore the user of these results does not need to actually manipulate formal series, our results are given under the form of ***ready-to-use formulae.***

For the average case analysis of the RETE algorithm, in the first case, the weighted model only introduce a multiplication factor of W_i in the Π_i. Thus the result with this modification is the same as in the uniform model. In the second case, which to our mind is only a theoretical case, the total semi-unification cost is the sum for all the node i of the join network of the $\beta_i\overline{b^{l_i}_{n,k}}$.

5. THE NEGATION

We can easily extend the previous calculation to a model taking into account the ***negation.*** More precisely, we can consider the negation of condition-patterns in the left-hand side of the rules and non-equality tests between variables too.

As a matter of fact, the average number of terms matching with the following pattern (in which the X_k's stand for the variable leaves of the pattern and $HEAD$ denotes the non variable part of the pattern)

$$HEAD[X_1, \ldots, \neg X_i, \ldots, \neg X_j, \ldots, X_p]$$

is equal to the average number of terms matching with $HEAD[X_1, \ldots, \star, \ldots, \star, \ldots, X_p]$ minus the average number of terms matching with $HEAD[X_1, \ldots, X_i, \ldots, X_j, \ldots, X_p]$ (with $\star$ a don't care symbol *i.e.* each time a new variable distinct from all the X_k and previous $\star$), and we know these two quantities. Let us consider now the negation of a condition-pattern or of a tuple of condition-patterns.

For a node i (see figure 6), we know the matching-rate of the node k $\overline{\alpha^k_n}$ and the matching-rate of the node j $\overline{\alpha^j_n}$. The matching rate of the pattern $\neg(P_1, \ldots, P_n)$ is $1 - \overline{\alpha^k_n}$ and thus the matching rate of the node i corresponding to the i-pattern $(Q_1, \ldots, Q_m, \neg(P_1, \ldots, P_n))$ is $\overline{\alpha^i_n} = \overline{\alpha^j_n}(1 - \overline{\alpha^k_n})$. And the matching rates of the join nodes that are under this node i (*i.e.* that can be reached from i) are modified in the same way: we determine their usual matching rate forgetting the existence of the tuple of patterns $\neg(P_1, \ldots, P_n)$ and we multiply this result to obtain the real matching rate by the coefficient $(1 - \overline{\alpha^k_n})$.

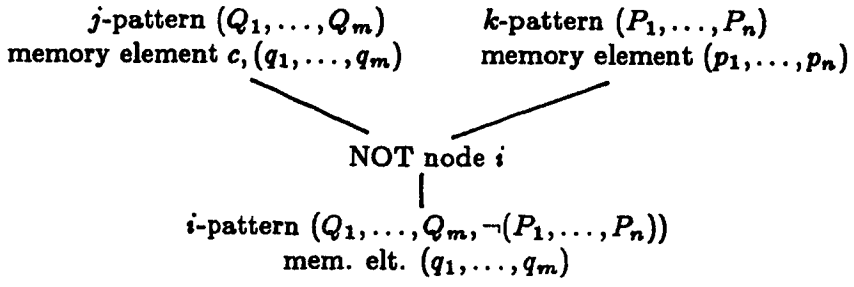


FIGURE 6

6. CONCLUSION

Considering an uniform distribution law of symbols inside facts and patterns the average case complexity of Rete algorithm is given by formula (1) in section 3.4. Formula (1) gives the average cost for computing with Rete algorithm the set of satisfied rules from the set of initial facts and a fixed set of rules. This result is given in function of the average sizes of local memories in the RETE network. These quantities are formulated in theorem 10 and can be computed from the parameters of any particular Rule Base and Working Memory (especially by a symbolic manipulation program such as Maple or Macsyma).

Considering a weight function that assigns to each symbol its frequency of occurrence inside a fact or a pattern, theorem 14 gives an estimation of the size of the local memories in join nodes, and of the probability for a term to be memorized in a given node. From these values we derive the average cost of the RETE algorithm. Here again these quantities can be computed given the parameters of any particular Rule Base and Working Memory. These theoretical results will be confronted to experimentation (similar experimentations in the case of the Lisp language have been done by Clark, see [Clark 79]).

Negation has been studied in section 5. More work is needed to take into account other predicates (such as arithmetic inequality), or arbitrary tests. As it has been mentioned in the text in the case of the hashing of local memories, the formulae we obtained can be adapted to many variants of RETE algorithm. The point is that generating functions theory is a very general framework to conduct the complexity analysis of algorithms, and the model we developed in this paper is interesting as a piece of a more general modelling of symbolic computation, and by itself for the combinatorial theorems we proved in order to make calculations.

7. ACKNOWLEDGEMENTS

We would like to express our gratitude to Ph. Flajolet, J.M. Steyaert and M. Soria for the numerous advices they gave to us. And to all the members of the Algo project at INRIA for their help.

8. REFERENCES

[Alb 87] L. Albert "Présentation et évaluation de la complexité de l'algorithme RETE de multi-pattern matching dans les systèmes de règles de production", rapport de DEA Universite de PARIS XI, ENS, rapport de recherche 87-8 LCR Thomson-CSF, 1987.

[dB 58] NG de Bruijn *Asymptotic Methods in Analysis* Dover 1958.

[CKS] C.Choppy, S.Kaplan,M.Soria "Algorithmic complexity of term rewriting systems", proceedings of the first Conference on Rewriting Techniques and Applications, Dijon, France. 1986.

[Clark 79] D.W. Clark "Measurements of Dynamic List Structures Use in Lisp". IEEE Transactions on Software Engineering SE-5(1), pp.51-59. (Jan. 1979).

[Dieu 68] J. Dieudonné *Calcul infinitésimal* Hermann 1968.

[Duf 84] P. Dufresne "Contribution algorithmique à l'inférence par règles de production", Thèse Université Paul Sabatier Toulouse 1984.

[Fag 86] F. Fages "On the proceduralization of rules in expert systems", First France-Japan Symposium on Artificial Intelligence, Tokyo, Nov. 86. In Programming of Future Generation Computers, Addison-Wesley, Eds. M. Nivat and K. Fuchi.

[Fla 85] P. Flajolet "Mathematical methods in the analysis of algorithms and data structures", INRIA, Research Report 400,1985. To appear in the Mathematical Handbook of Theoretical Computer Science, North-Holland.

[Fla 87] P. Flajolet "The symbolic operator method", *in Mathematical methods in the analysis of algorithms and data structure*, L.N.C.S., Springer Verlag, to appear 1987.

[Forg 79] C. Forgy "On the efficient implementation of production systems", PhD Thesis Carnegie Mellon University 1979.

[Forg 81] C. Forgy "OPS-V user's manual", Computer Science Department, Carnegie Mellon University, Pittsburgh, MA, 1981.

[Forg 82] C. Forgy "RETE, a fast algorithm for the many patterns many objects Match problem", *Artificial Intelligence* 19, 1982,pp 17-37.

[FS 86] P. Flajolet, R. Sedgewick "Mathematical analysis of algorithms", Computer Science 504, lecture Notes for Princeton University 1986.

[FSS] P. Flajolet, P. Sipula et J.M. Steyaert "The analysis of tree compaction in symbolic manipulations", preprint.

[GD 87] M. Ghallab, P. Dufresne "Moteurs d'inférences pour systèmes de règles de production : techniques de compilation et d'interprétation", LAAS Toulouse 1987,pp 89-103.

[GF 83] A. Gupta et C.L. Forgy "Measurements on production systems", Carnegie Mellon University Technical Report CMU-CS-83-167,1983.

[Gupt 84] A. Gupta "Parallelism in production Systems : the sources and the expected Speed-up", Carnegie Mellon University Technical Report CMU-CS-84-169,1984.

[Henr 74] P. Henrici *Applied and Computational complex Analysis* Volumes 1-3 Wiley New-York.

[HWL 83] F. Hayes-Roth, D.A.Waterman and D.A. Lenat. *Building Expert Systems.* Addison-Wesley, Reading, M.A. (1983).

[LNR 87] J.E. Laird, A. Newell and P.S. Rosenbloom. "SOAR: An Architecture for General Intelligence", *Artificial Intelligence* 33, pp 1-64. (1987).

[MAPLE 85] B.W. Char, K.O. Geddes, G.H. Gonnet, S.M. Watt, *MAPLE: Reference Manual*, University of Waterloo, 1985.

[Mir 87] D.P. Miranker "TREAT: A Better Match Algorithm for AI Production Systems" Proceedings of the 1987 National Conference on Artificial Intelligence. Seattle, Washington. (1987).

[MM 78] A. Meier, J.W. Moon "On the altitude of nodes in random trees",*Canadian Journal of Math* 30,1978,pp 997-1015.

[SBM 83] Stefik M., Bobrow D., Mittal S. and Conway L "Knowledge Programming in LOOPS: Report on an Experimental Course" The AI magazine Fall 83, pp 3-13. (1983).

[SF 88] Schang T. and Fages F "A Real-Time Expert System for On-Board Radar Identification" 55th Symposium AVP-AGARD on Software Engineering and its Applications to Avionics.(1988).

[SF 83] J.M. Steyaert, P. Flajolet "Patterns and Pattern Matching in trees : an analysis", *Information and Control* 58,1983,pp 19-58.

[Stey 84] J.M. Steyaert "Complexité et structures des algorithmes", Thèse d'Etat Université de Paris 7 1984.

[Vien 86] G. Viennot "La combinatoire bijective par l'example", Université de Bordeaux 1, 1986.

[War 86] Warren D.H.D. An abstract Prolog Instruction Set. Technical Note 309, SRI International. 1985.

[WGF 86] Wright, M.L., Green M.W., Fiegl G., Cross P.F., "An Expert System for Real-Time Control" SRI International, in IEEE Software. (March 1986).

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique