Using Queueing Theory for Analytical Performance Evaluation of a Multiple Functional Unit Rete Network

José Nelson Amaral[*]
(*amaral@madona.pucrs.br*)
Departamento de Eletrônica
Pontifícia Universidade Católica do RGS
90619-900 - Porto Alegre, RS

Joydeep Ghosh
(*ghosh@pine.ece.utexas.edu*)
Dept. of Electr. and Comp. Engineering
The University of Texas at Austin
Austin, Texas 78712

Abstract

The matching phase is believed to require as much as 90 percent of the execution time of a production system. The Rete Network is the most well known and broadly used matching algorithm for production system. This paper proposes a new concurrent organization with multiple functional units for execution of Rete Networks. We address the lack of theoretical studies of matching performance in production systems by presenting an analytical model to predict the Rete Network performance. This model is based on infinite Markov chain and queueing theory and successfully estimates the average number of tokens in a system with m functional units. Little's result allows the use of this estimate to predict the performance improvement when more functional units are added to the system. Finally, we report the predictions of the model for the Rete Network used in a parallel architecture proposed in a related work [1].

1 Introduction

Production systems provide an efficient paradigm to construct knowledge base systems. However, they are haunted by the combinatorial complexity of matching of rule conditions with the set of facts stored in the knowledge base. The matching phase is the portion of production system execution whose optimization can yield the most performance improvement. Research efforts towards improving the speed of production system machines are surveyed by Kuo and Moldovan [11] and by Amaral and Ghosh [3].

In a typical production system the memory is divided into a set of productions or rules stored in a Production Memory, and a set of facts stored in a Working Memory. A production consists of

[*]Supported by a fellowship from Conselho Nacional de Desenvolvimento Cient´ıfico e Tecnológico (CNPq) and by Pontif´ıcia Universidade Católica do Rio Grande do Sul (PUCRS) - Brazil.

a set of conditions and a set of actions. A production is said to be fireable if the facts stored in the Working Memory[1] satisfy all its conditions. At each cycle during its execution the production system selects some of the fireable productions and execute the specified actions. The device responsible for verifying whether the production conditions are satisfied is called a matching engine.

The amount of work performed by the matching engine is a combinatorial function of the number of productions in the system and the number of facts in the knowledge base. However, two characteristics of production systems allow an efficient solution to this problem: distinct productions have identical condition elements and pieces of knowledge stored in the working memory change slowly over time. The slow change in the knowledge base implies that if the results of the matching in one cycle are saved for the next cycle, a substantial amount of work can be eliminated. The existence of productions with identical antecedents allows the construction of an algorithm in which the results of matching shared conditions are used by all productions that need them.

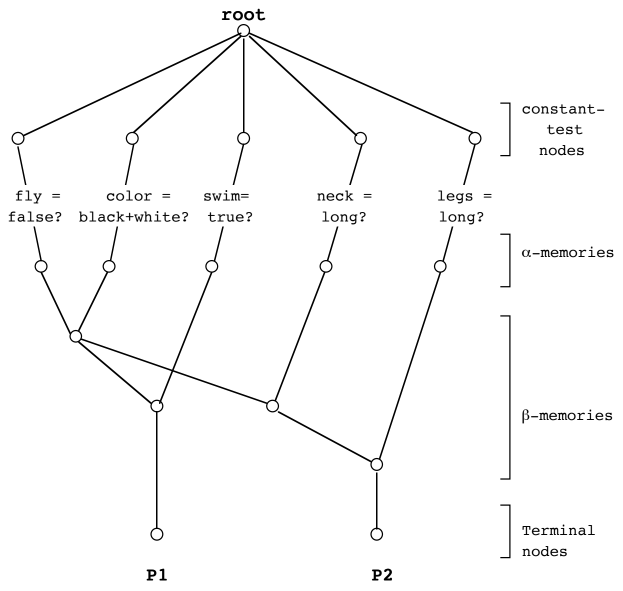
2 The Rete Network

The best-known state saving algorithm is the Rete Network created by Forgy [6]. Forgy reports that Rete is inspired by the *Pandemonium* machine of Selfridge [14]. Pandemonium was one of the earliest learning machines and consisted of multiple layers of *demons*. A demon in a given level supervised an inferior level of demons. When it observed meaningful patterns, it sent messages to a superior level. The top-level demons performed more telling actions.

The Rete Network is a data-flow graph that encodes the antecedents of productions. The inputs to the Rete Network are changes to the working memory generated in the act phase of one cycle, and the outputs of the network are changes to the conflict set used to choose a production to be fired in the next cycle. The following discussion of Rete Networks is presented here after Gupta and Forgy [7, 8], Lee and Schor [12], and Miranker [13].

Figure 2 presents a Rete Network with the set of production antecedents encoded in it. The network is formed by four different kind of nodes: constant-test nodes, memory nodes, two-input nodes and terminal nodes. The constant-test nodes appear in the first layer of the network. They store attributes that have constants in the value field and perform intra-condition tests to determine if a working memory element satisfies these constant fields of the condition element. In the original Rete Network, the result of this test was stored in a local α-memory [8]. Some authors claim that

[1]Each one of these facts is called a Working Memory Element (WME).



```
(P P1 (bird <b>)
         (fly <b> false)
         (color <b> black+white)
         (swim <b> true)
         -->
         (type <b> Penguin))
```

```
(P P2 (bird <b>)
          (fly <b> false)
          (color <b> black+white
          (neck <b> long)
          (legs <b> long)
          -->
          (type <b> Ostriches))
```

Figure 1: Section of a Rete Network

the α-memories can be eliminated because the constant test takes a negligible amount of time [12]. In this case the one-input nodes are called α-nodes and are memoryless.

Two-input nodes, also called *and*-nodes, *join*-nodes, or β-nodes, perform the matching between distinct condition elements. A β-node has one memory associated with each input. A token arriving in a β-node come either from another β-node or from an α-node. Whenever a new token arrives in one of a node's inputs, it is compared with all tokens present in the other side memory. Good hashing techniques are necessary to speed up the matching in the two-input nodes. Otherwise, it might be necessary to process long lists of tokens. [7].

3 Rete Network with Multiple Functional Units

We propose a multiple functional unit architecture to implement the Rete Network algorithm in a production system machine. In the organization presented in Figure 2, a single α-unit is responsible for processing tokens in α-nodes. Each incoming token is sequentially tested against all α-nodes of the network. Whenever there is a match with an α-node, the α-unit replicates the token and attach to each copy the identity of one of the β-nodes that are successors of the α-node where the match occurred. These new tokens are then forwarded to the β-units. Therefore the processing of a single token in the α-unit may result in the generation of many tokens to be inserted in the *external* β-queue. Note that β-units also share an *internal* input queue for tokens generated by β-nodes.

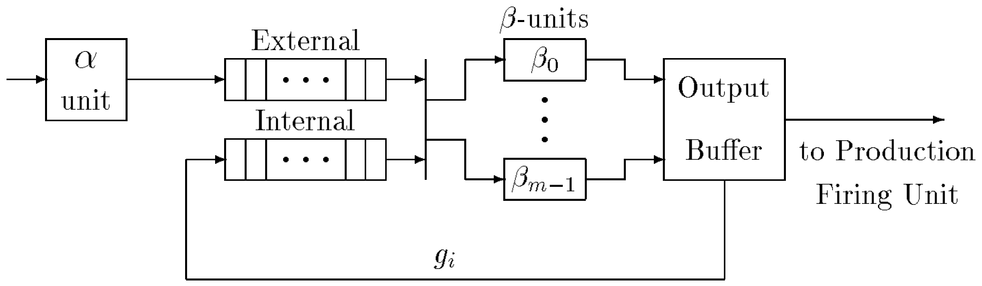


Figure 2: Rete network with m β-Units

To reduce the average time that tokens spend in the Rete network, a token placed in the external queue is processed only when the internal queue is empty. A free β-unit will take the first token from the queue, read the memory locations corresponding to the β-node to which the token is destined, execute all the β-tests and then place the newly generated tokens at the end of the internal queue. When a token destined for a P-node is produced by a β-unit it is forwarded to the production firing unit.

4 Performance Evaluation

Analytical modeling is a powerful and underused tool in the study of production system machine performance. Yukawa *et al.* [16, 10] construct an analytic model for the performance of Rete network based on basic notions of probability, utilizing a "back of the envelope" method similar to the one largely employed by Cragon [5] for the analysis of superpipelined and superscalar processors. Wang et al. [15] constructed an analytical model to measure performance of parallel-rule firing production systems based on a transaction model. In the following sections we present an analytical model to predict the performance of the concurrent Rete network presented in Figure 2.

4.1 Simplifications for Analytic Modeling

The organization considered for the construction of an analytical model for the Rete network with multiple β-units is shown in Figure 3. Whenever a token matches an α-node while being processed in the α-unit, a number of tokens are produced to be delivered to the β-FIFO. This group of tokens is called a *bulk*. The probability that a bulk has j tokens is measured by h_j. The model does not consider empty bulks, i.e., an arrival occurs when at least one token arrives in the β-FIFO. Therefore $h_0 = 0$.

Whenever free, a β-unit will take the first token from the queue, read the memory locations corresponding to the β-node to which the token is destined, execute all the β-tests and then place the newly generated tokens at the end of the β-FIFO. If the processing of a token in a given β-node produces a new tokens, and that β-node has b β-node successors, we consider that ab new tokens were generated and placed at the end of the queue. The probability that i new tokens are generated when a token is processed in a β-node is given by g_i. Also, any new instantiation at a terminal node is sent to the production firing unit that will select one (or more) production to be fired.

For the construction of the analytical model, we consider that there is a "waiting station" at the β-unit output that allows it to hold all the tokens generated until the end of the execution of the current token, and then adds all of them at once to the queue. Although this assumption may not reflect the actual behavior of a physical machine[2], it simplifies the analytical model.

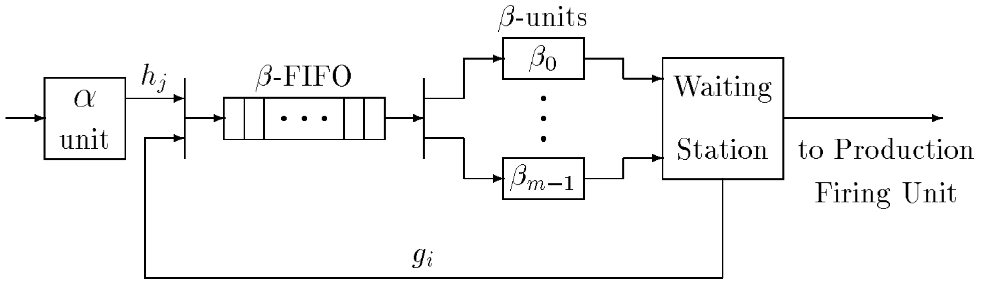


Figure 3: System with m β-Units

In this model, the arrival of a bulk of tokens in the β-queue from an α-node is called an *external arrival*. The arrival of a group of tokens from the *waiting station* into the β-FIFO is an *internal arrival*. The moment at which a β-unit finishes processing a token and places all the newly generated tokens (if any) in the β-FIFO is called *departure*. To develop the queueing theory model, we assume

[2]In an actual machine the β-unit would place tokens in the queue as they are produced.

that the β-FIFO has infinite capacity; the overhead of placing tokens in the β-FIFO is zero; the external arrival of tokens is a Poisson process; and the processing time for tokens in the β-nodes follows a Poisson process. In fact events in the machine are depend on the specific production system being executed and change as the execution of the system proceeds. However, the simplified model obtained still produces a good estimate for the amount of time that a token spends in the system, and thus for the reduction in the average time spent by a token in an m β-unit system compared with a single β-unit system.

4.2 Multiple β-Unit Model

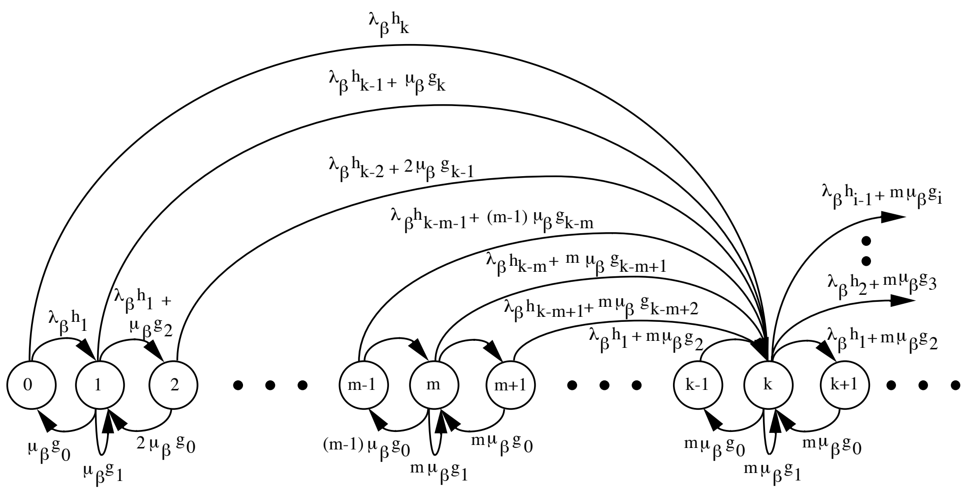
The infinite Markov chain that represents the multiple β-unit system is presented as a state-transition-rate diagram in Figure 4. The circles represent states and the arcs represent state transitions. The value associated with each arc indicates the transition rate. In this representation, the system is in state k if there are k tokens to be processed in the system, including the tokens currently being processed. Let g_i be the probability that the processing of a token in a β-unit produces i new tokens, h_j be the probability that an external arrival brings into the system a bulk with j tokens, λ_β be the external arrival rate into β-nodes, and μ_β be the processing rate in β-nodes.

For clarity, not all transitions are shown in Figure 4. Consider the transitions from and to state k. Any state $i < k$ can reach state k in a single transition. The transition from state k to state $k-1$ indicates that the processing of a token produced no new tokens.

A transition out of state k might occur due to an external arrival or due to the completion of the processing of a token in the β-unit. Whenever $i > 1$ tokens are generated by the β-unit, the system changes from state k to state $k+i-1$ upon the completion of β-unit processing. If a bulk of size j arrives from the α-unit while the system is at state k, the system moves to state $k+j$.

The only possible transitions out of state 0 are due to the external arrival of a bulk of tokens. The processing rate at state 0 is equal to zero. This reflects the fact that no token can be processed when there are no tokens in the system.

The transition rate out of states 1 through $m-1$ due to the processing of tokens in β-units is proportional to the number of tokens in the system. This occurs because if $i < m$ tokens are present, all tokens are being processed, and $m-i$ β-units are idle. Therefore, the processing rate is proportional to the number of tokens in the system. If, however, $j \geq m$ tokens are present in the system, m tokens are being processed and $j-m$ tokens are waiting in the queue; therefore, the processing rate is proportional to the number of β-units. Every state (except for state 0) has a self transition. If $i < m$, the self transition rate for state i is $i\,\mu_\beta\, g_1$. Otherwise the self transition rate is $m\,\mu_\beta\, g_1$.



$\lambda_\beta h_k$
$\lambda_\beta h_{k-1} + \mu_\beta g_k$
$\lambda_\beta h_{k-2} + 2\mu_\beta g_{k-1}$
$\lambda_\beta h_{k-m-1} + (m-1)\mu_\beta g_{k-m}$
$\lambda_\beta h_{k-m} + m\mu_\beta g_{k-m+1}$
$\lambda_\beta h_{k-m+1} + m\mu_\beta g_{k-m+2}$
$\lambda_\beta h_1 + m\mu_\beta g_2$
$\lambda_\beta h_{i-1} + m\mu_\beta g_i$
$\lambda_\beta h_2 + m\mu_\beta g_3$
$\lambda_\beta h_1 + m\mu_\beta g_2$
$\lambda_\beta h_1$
$\lambda_\beta h_1 +$
$\mu_\beta g_2$
0
1
2
m-1
m
m+1
k-1
k
k+1
$\mu_\beta g_0$
$\mu_\beta g_1$
$2\mu_\beta g_0$
$(m-1)\mu_\beta g_0$
$m\mu_\beta g_1$
$m\mu_\beta g_0$
$m\mu_\beta g_0$
$m\mu_\beta g_1$
$m\mu_\beta g_0$

Figure 4: State Diagram for System with m β-Units

We are interested in the steady state behavior of the system. At steady state, the input flow must equal the output flow in each state. Therefore, we can write difference equations for the system. The $m-1$ boundary conditions of this system are expressed by equation 1. The flow conservation equation for state $k \geq m$ is given by equation 2.

For $k < m$:

$$(\lambda_\beta + k\,\mu_\beta)\, p_k = (k+1)\,\mu_\beta\, g_0\, p_{k+1} + \lambda_\beta \sum_{j=0}^{k-1} p_j\, h_j + \mu_\beta \sum_{i=1}^{k} i\, p_i\, g_{k-i+1}. \tag{1}$$

For $k \geq m$:

$$\begin{aligned}(\lambda_\beta + m\,\mu_\beta)\, p_k &= m\,\mu_\beta\, g_0\, p_{k+1} + \lambda_\beta \sum_{l=0}^{k-1} p_l\, h_l + \\ &\quad \mu_\beta \sum_{i=1}^{m-1} i\, p_i\, g_{k-i+1} + \mu_\beta \sum_{j=m}^{k} m\, p_j\, g_{k-j+1}. \end{aligned} \tag{2}$$

where p_k is the probability of k tokens being in the system, including the token being processed.

To compute the generating function $P(z)$, we have to sum equation 1 from 0 to $m-1$, and sum equation 2 from m to infinity, and then combine both results and simplify [1]. This results in

$$P(z) = \frac{\mu_\beta\,[z - G(z)]\,\sum_{k=0}^{m-1}(m-k)\,p_k\,z^k}{\lambda_\beta\,z\,[1 - H(z)] \,+\, m\,\mu_\beta\,[z - G(z)]}. \tag{3}$$

where $P(z)$, $G(z)$, and $H(z)$ are defined as

$$P(z) \,=\, \sum_{k=0}^{\infty} p_k\,z^k, \qquad G(z) \,=\, \sum_{k=0}^{\infty} g_k\,z^k, \qquad H(z) \,=\, \sum_{k=0}^{\infty} h_k\,z^k. \tag{4}$$

Computing the limit of $P(z)$ as z goes to 1, and making the result equal to 1 (series sum property), we obtain relation 5. Equation 1 provides m equations and $m+1$ unknowns. The relation expressed in 5 is the last equation we need to solve this linear system and obtain the values for the boundary probabilities p_0, p_1, ..., p_m.

$$\sum_{k=0}^{m-1}(m-k)\,p_k \;=\; m\,(1 - \rho_m) \tag{5}$$

$$\rho_m \;=\; \frac{\lambda_\beta\,\overline{H}}{m\,\mu_\beta\,(1 - \overline{G})} \tag{6}$$

To guarantee stability, the utilization factor ρ_m for the system with m β-units must be less than unity. Therefore the distributions of g_i and h_i must have the following property:

$$\frac{\overline{H}}{1 - \overline{G}} < \frac{\lambda_\beta}{m\,\mu_\beta}. \tag{7}$$

$$\overline{G} \,=\, \lim_{z\to 1} G^{(1)}(z) \,=\, \sum_{i=0}^{\infty} i\,g_i \tag{8}$$

$$\overline{H} \,=\, \lim_{z\to 1} H^{(1)}(z) \,=\, \sum_{i=0}^{\infty} i\,h_i \tag{9}$$

From the definition of $P(z)$ it follows that

$$\lim_{z\to 1} P^{(1)}(z) \,=\, \lim_{z\to 1} \sum_{k=1}^{\infty} k\,p_k\,z^{k-1} \,=\, \sum_{k=1}^{\infty} k\,p_k \,=\, \overline{N}(m), \tag{10}$$

where $P^{(1)}(z)$ represents the first derivative of $P(z)$ with respect to z, and $\overline{N}(m)$ is the average number of tokens in the m β-unit system including the tokens currently being processed.

For g_i we verify that

$$\lim_{z\to 1} G^{(2)}(z) \,=\, \lim_{z\to 1} \sum_{i=0}^{\infty} i\,(i-1)\,g_i\,z^{i-2} \,=\, \sum_{i=1}^{\infty} i\,(i-1)\,g_i \,=\, \sum_{i=1}^{\infty} i^2\,g_i \,-\, \sum_{i=1}^{\infty} i\,g_i \,=\, \overline{G^2} \,-\, \overline{G}, \tag{11}$$

where $G^{(2)}(z)$ is the second derivative of $G(z)$ with respect to z, $\overline{G}$ is the *first moment* of $G(z)$, and $\overline{G^2}$ is the *second moment* of $G(z)$. In a similar fashion, we can establish that

$$\lim_{z \to 1} H^{(2)}(z) \;=\; \overline{H^2} \;-\; \overline{H}. \tag{12}$$

According to equation 10, the average number of tokens in the system with m β-units, including the tokens that are currently being processed, is obtained by computing the limit of the first derivative of $P(z)$ as z goes to 1 [1].

$$\overline{N}(m) \;=\; \frac{\rho_m}{2\,(1-\rho_m)} \left[\frac{(\overline{H^2}+\overline{H})}{\overline{H}} \;+\; \frac{(\overline{G^2}-\overline{G})}{1-\overline{G}} \right] \;+\; \frac{\sum_{k=0}^{m-1} k\,(m-k)\,p_k}{m\,(1-\rho_m)}. \tag{13}$$

The first and second moments of $G(z)$ and $H(z)$ can be obtained from measures of g_i and h_j from a simulator[1]. The boundary probabilities are obtained by solving the system of linear equations mentioned previously. Little's result $\overline{N}(m)$ can be used to estimate the average time that a token spends in the system.

$$\overline{T}(m) \;=\; \frac{\overline{N}(m)(1-\overline{G})}{\lambda_\beta \overline{H}}, \tag{14}$$

where $\overline{T}(m)$ represents the average total time that a token spends in a m β-unit system, including the time the token is waiting in the queue and the processing time. Observe that we have to consider the total arrival rate in the β-FIFO, i.e., external tokens originated in the α-nodes as well as tokens generated in the β-nodes. In equation 14, the external arrival rate is $\lambda_\beta \overline{H}$ and the arrival rate due to generation of tokens in β-units is $(1-\overline{G})^{-1}$.

4.3 Rete Processing Improvement

Using Little's result, the ratio between the average time spent in the single β-unit system and the average time spent in the system with m β-units can be obtained from the average number of tokens in each one of these systems. We define the improvement in system time according to equation 15.

$$I_s(m) = \frac{\overline{T}(1)}{\overline{T}(m)} = \frac{\overline{N}(1)}{\overline{N}(m)}, \tag{15}$$

$I_s(m)$ indicates how much faster a token goes through the β-node portion of the Rete network when m β-units are used instead of one. Notice that the definition of $I_s(m)$ assumes that the effective arrival rate of the system does not change with the number of β-units.

We purposely avoid calling this performance improvement *speedup* because of the loaded meaning of that word. Speedup is a system level concept that is applied to different settings, e.g. fixed-time speedup, fixed-load speedup, memory-bound speedup, etc [9]. Also, except for very rare cases, the speedup of a computer system is always a sublinear function of the number of processors in the system. In our system, because of the nature of the incoming flow of tokens, the amount of improvement in the time spent in the β-network can be superlinear. This happens because the existence of occasional bursts of traffic in the incoming flow of tokens can cause tokens to spend considerable amount of time waiting in the β-unit input queue. Of course, if the incoming flow of token were to be steady, the improvement in the time spent in the system would be at most linear.

4.4 Analytical Model Predictions

The analytical model presented in this paper assumes a steady state in the flow of tokens through the Rete network. An approximation of such a situation is only encountered in fairly large production system programs. The multiple functional unit Rete Network presented in this paper was used as the matching engine for a parallel production system architecture that we proposed [1, 2, 4]. Using a comprehensive system level simulator we obtained parameter measurements for a fairly large benchmark program (`moun2`. Table 1 present measurements for the first and second moment of g_i and h_j, for the service rate μ_β and the average number of tokens in the single β-node system $\overline{N}(1)$.

# Proc	$\overline{G}$	$\overline{G^2}$	$\overline{H}$	$\overline{H^2}$	μ_β	$\overline{N}(1)$
1	0.78	3.35	5.45	40.22	0.00096	644.8
2	0.80	3.10	2.89	11.58	0.00093	244.4
4	0.79	4.17	2.15	6.15	0.00096	160.2
6	0.80	4.20	1.59	3.51	0.00094	89.6
10	0.80	5.06	1.41	2.48	0.00102	47.8

Table 1: Parameter Measurements for `moun2`.

In the architecture simulated in [1], a production set is divided among the processors in the machine. Therefore, a machine with a larger number of processors has smaller Rete networks in each processor. Moreover, the amount of tokens processed in each Rete network is smaller. Note that the first and second moments of g_i do not change significantly when the Rete network is divided into a larger number of networks, but rather seems to be a characteristic of the benchmark program. On the other hand, the moments of h_j do change substantially when each processor has smaller

Rete networks because in such networks each α-node has fewer successors. The value of μ_β is also independent of the size of the Rete network. The measures presented in Table 1 were obtained by forcing the simulator to implement a single β-FIFO to replicate the simplification used in the analytical model.

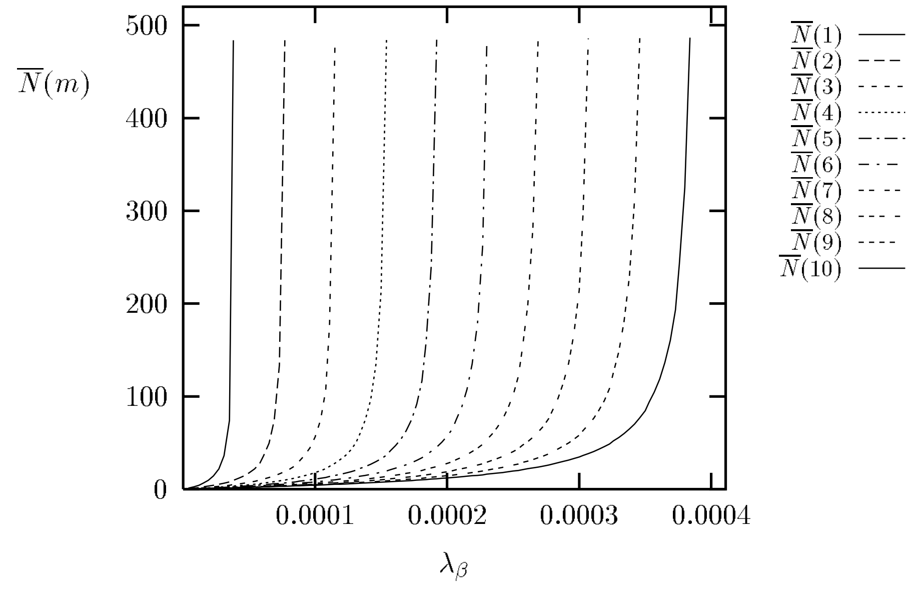


Figure 5: Average Number of Tokens in the System versus λ_β for `moun2` in a Single Processor Architecture

Figure 5 shows the variation in the average number of tokens in the system with the value of the arrival rate λ_β for Rete network of *moun2*, in a single processor architecture with one up to ten β-units. Notice that there is a dramatic increase in the number of tokens in the system when the utilization rate tends to one. For systems operating close to such an asymptote, the addition of a single β-unit can improve the performance significantly. This improvement is due to the reduction in the amount of time spent waiting in the queues.

A difficulty in the utilization of this analytical model to estimate performance improvements is the correct estimation of arrival rate λ_β. This rate does not remain constant when the number of β-units is increased. The outputs of the Rete network described in this paper are used to select productions that are fired and generate more changes to the Working Memory. A Rete network with a larger number of β-units produces changes to the conflict set at a faster pace resulting in a higher arrival rate. The predictions of speed improvement in Figure 6 were obtained using a value

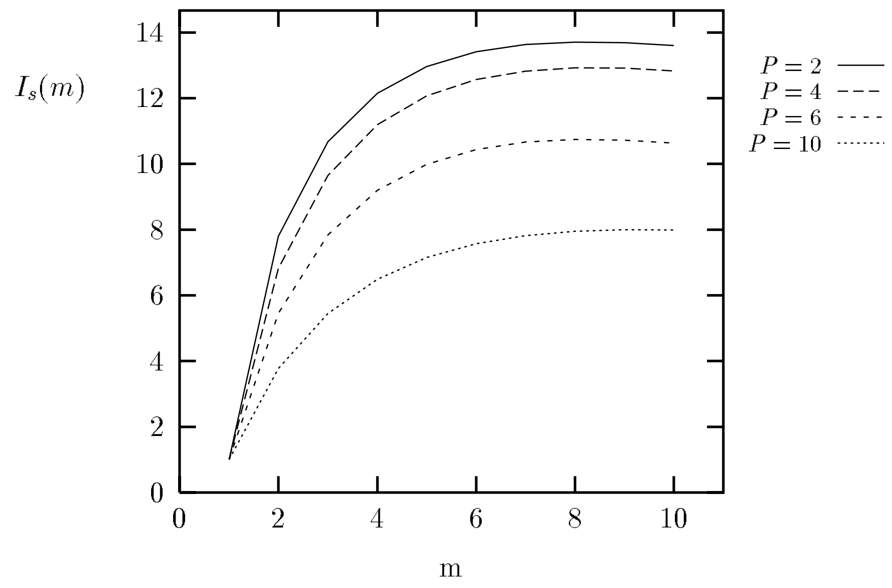


Figure 6: Prediction of the Rete network speed improvement for `moun2` considering that λ_β changes when more β-units are used.

measured in the simulator for the arrival rate when the number of beta units is increased. Future research with experimental measurements of a larger set of benchmarks might determine a general rule to estimate the variation of the arrival rate with the number of β-units.

Figure 7 presents the actual speed improvement obtained from measures in the simulator when multiple β-units are used in the architecture. To measure only the speed improvement due to the addition of β-units, each curve in Figure 7 is has an architecture with the same number of processors and a single β-unit as base of comparison. Because Figure 6 plots the predicted speed improvement only in the Rete network while Figure 7 plots the actual speed improvement for the whole machine, the absolute values are quite different. However, comparing the plots of Figures 6 and 7 we verify that the analytical model was successful in anticipating the trend of these curves. A computer architect could estimate the number of β-units to use in a design only from the analytical model predictions.

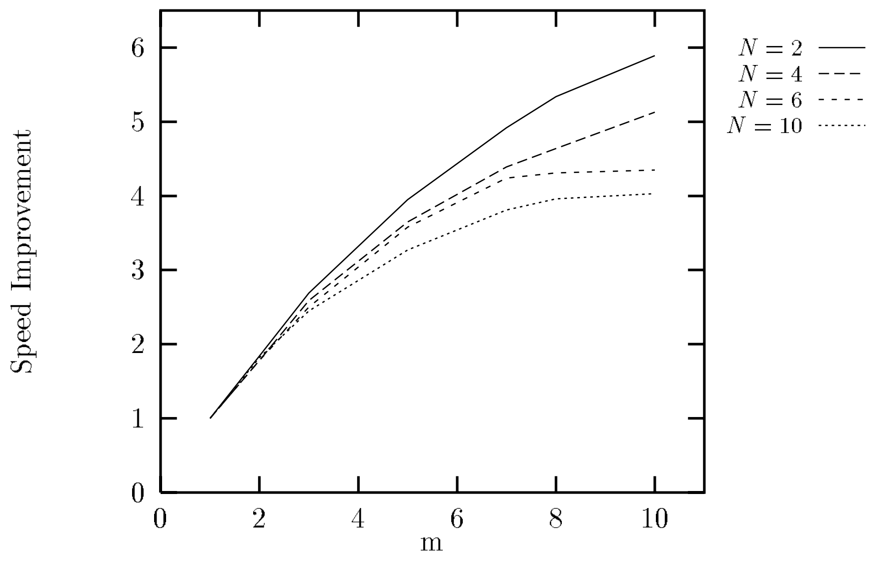


Figure 7: Actual speed improvement due to increased number of β-units for `moun2`.

5 Conclusion

This paper proposed a new concurrent organization for execution of the classic Rete network algorithm. The performance improvement predicted by the analytical model developed indicates that a modest number of extra β-units might eliminate wasted waiting time, producing significant improvement in the average time taken for a token to be processed in the Rete network. Further studies along this line of research must include the construction of an analytical model for the priority system with an external and an internal β-queue and further experimental studies with this model to determine how the arrival rate changes when the number of β-units is increased.

References

[1] J. N. Amaral. *A Parallel Architecture for Serializable Production Systems.* PhD thesis, The University of Texas at Austin, Austin, TX, December 1994. Electrical and Computer Engineering.

[2] J. N. Amaral and J. Ghosh. An associative memory architecture for concurrent production systems. In *Proc. 1994 IEEE International Conference on Systems, Man and Cybernetics*,

pages 2219–2224, San Antonio, TX, October 1994.

[3] J. N. Amaral and J. Ghosh. Speeding up production systems: From concurrent matching to parallel rule firing. In L. N. Kanal, V. Kumar, H. Kitani, and C. Suttner, editors, *Parallel Processing for AI*, chapter 7, pages 139–160. Elsevier Science Publishers B.V., 1994.

[4] J. N. Amaral and J. Ghosh. Performance measurements of a concurrent production system architecture without global synchronization. In *Proc. 9th International Parallel Processing Symposium*, pages 790–797, Santa Barbara, CA, April 1995.

[5] H. G. Cragon. *Memory Systems and Pipelined Processors.* Jones and Bartlet Publishers, Sudbury, MA, 1996.

[6] C. L. Forgy. *On the Efficient Implementations of Production Systems.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1979.

[7] A. Gupta. Implementing OPS5 production systems on DADO. In *Proceedings of International Conference on Parallel Processing*, pages 83–91, 1984.

[8] A. Gupta, C. Forgy, and A. Newell. High-speed implementations of rule-based systems. *ACM Transactions on Computer Systems*, 7:119–146, May 1989.

[9] K. Hwang. *Advanced Computer Architecture: Parallelism, Scalability and Programmability.* McGraw-Hill, New York, 1993.

[10] H. Kikuchi, T. Yukawa, K. Matsuzawa, and T. Ishikawa. Presto: A bus-connected multiprocessor for a Rete-based production system. In *Joint International Conference on Vector and Parallel Processing - CONPAR 90*, pages 63–74, February 1990.

[11] S. Kuo and D. Moldovan. The state of the art in parallel production systems. *Journal of Parallel and Distributed Computing*, 15:1–26, June 1992.

[12] H. S. Lee and M. I. Schor. Match algorithms for generalized Rete Networks. *Artificial Intelligence*, 54:249–274, April 1992.

[13] D. P. Miranker. *TREAT: A New and Efficient Match Algorithm for AI Production Systems.* Pittman/Morgan-Kaufman, 1990.

[14] O. G. Selfridge. Pandemonium: A paradigm for learning. In *Proceeding of a Symposium Held at the National Physical Laboratory*, pages 513–526, November 1958. Reprinted in *Neurocomputing - Foundations of Research*, edited by J. A. Anderson and R. Rosenfeld, The MIT Press, 1988.

[15] J.-H. Wang, J. Srivastava, and W. T. Tsai. A transaction model for parallel production systems: Part ii. model and evaluation. *International Journal on Artificial Intelligence Tools*, 2(3):431–457, 1993.

[16] T. Yukawa, T. Ishikawa, H. Kikuchi, and K. Matsuzawa. TWIN: A parallel scheme for a production system featuring both control and data parallelism. In *Proceedings of the 7th Conference on Artificial Intelligence Applications*, pages 64–70, February 1991.