CHAPTER 3

PARALLELISM IN HIERARCHICAL CENSORED PRODUCTION RULES

Results of this chapter have been presented in the following paper: Varshneya, R. and Bharadwaj, K.K., 'Hierarchical Censored Production Rules: Some Avenues for Parallelization', Proceedings International Workshop on Parallel Processing, Bangalore, India, 717-722, 1994.
Hierarchical Censored Production Rules (HCPRs) are standard production rules augmented with the UNLESS, SPECIFICITY and the GENERALITY operators represented as

- **IF** <premises> <br>  **THEN** <decision> <br>  **UNLESS** <censor> <br>  **SPECIFICITY** <specificities> <br>  **GENERALITY** <generality> 

\[
\text{IF } \langle \text{premises} \rangle \quad \langle P_1 \land P_2 \land \ldots \land P_n \rangle \\
\text{THEN } \langle \text{decision} \rangle \quad \langle D \rangle \\
\text{UNLESS } \langle \text{censor} \rangle \quad \langle C_1 \lor C_2 \lor \ldots \lor C_i \rangle \\
\text{SPECIFICITY } \langle \text{specificities} \rangle \quad \langle S_1 \oplus S_2 \oplus \ldots \oplus S_j \rangle \\
\text{GENERALITY } \langle \text{generality} \rangle \quad \langle G \rangle 
\]

Thus parallelizing HCPRs would mean,

1. parallelizing the evaluation of premises \((P_1, P_2, \ldots, P_n)\)
2. parallelizing the evaluation of censors \((C_1, C_2, \ldots, C_i)\)
3. parallelizing the evaluation of specificities and the generalities of the rule \((S_1, S_2, \ldots, S_j, G)\)

Over and above these, the match phase of the rule can also be parallelized. Each of these operators are modular in nature, implying that each of them contributes independently to the overall speedup of the rule.

As shown in the last chapter, a lot of research has already been carried out towards the parallelization of the match phase and the premises. Because of the modular nature of the HCPRs, the results of the standard production rules can be incorporated into restricted form of HCPRs (without UNLESS, SPECIFICITY and GENERALITY operators) also. Parallelization of censors and specificities and generalities is still unexplored and that is what we have concentrated upon in this thesis.
3.1 Parallel Evaluation of Censors

We have already seen that censors are basically exceptions to the rule, with their chances of evaluating to true very low. Further, each of the censors can again be rules, with their own set of premises and censors. As an example consider the set of following set of Censored Production Rules:

\[
\begin{align*}
\text{IF } P_1 & \text{ THEN } D \text{ UNLESS } C_1 \\
\text{IF } P_2 & \text{ THEN } P_1 \text{ UNLESS } C_2 \\
\text{IF } P_3 & \text{ THEN } C_1 \text{ UNLESS } C_3 \\
\text{IF } P_4 & \text{ THEN } P_2 \text{ UNLESS } C_4 \\
\text{IF } P_5 & \text{ THEN } C_2 \text{ UNLESS } C_5 \\
\text{IF } P_6 & \text{ THEN } P_3 \text{ UNLESS } C_6 \\
\text{IF } P_7 & \text{ THEN } C_3 \text{ UNLESS } C_7
\end{align*}
\]

\(P_4, P_5, P_6, P_7, C_4, C_5, C_6\) and \(C_7\) can be assumed to be all facts. This when represented graphically gives rise to the rule tree of the kind represented in Fig 3.1. The amount of search that needs to be performed to arrive at the decision with a required certainty factor is called its censor chaining depth and is defined as the length of the deepest subpath. The rule tree of fig 3.1 has 4 levels of censor chaining depth, 0 through 3, with each depth consisting of the following premises, \(P_i\), and censors, \(C_i\):

- **depth 0**: \(P_1, P_2, P_4\)
- **depth 1**: \(C_1 (P_3, P_6), C_2 (P_5), C_4\)
- **depth 2**: \(C_3 (P_7), C_5, C_6\)
- **depth 3**: \(C_7\)

In fig 3.1, the depth at which the corresponding rule will be evaluated is shown in brackets along with that rule. The certainty in the decision increases as more and more exception conditions are evaluated. However, as the chances of exception holding TRUE are very low, as compared to the premises, depending on the time or other resource constraint, the censors can be partially or totally ignored.
Figure 3-1: Rule tree representing various censor chaining depths
Michalski and Winston [33] suggested a number of control schemes for handling censors. The two extreme possibilities are

- **The show-me method**: Treat the UNLESS operator as if they were exclusive-or operators, thereby ignoring expectation informations. It is good in situations in which expectations are unreliable and for which nothing should be assumed.

- **The ask-questions-later method**: Ignore all censors. Good in situations for which rapid response is critical.

In addition to these, several intermediate possibilities are also suggested, of which the following are representative:

- **The trusting-skeptic method**: Once a premise is established, try to show that censor conditions are true, but fix the depth of rule chaining to some prescribed number of levels or to some specified consumption of resources. This method is called trusting-skeptic method because it reflects an assumption that the indicated expectations are solid and that there is little point in putting more than a little effort into overturning these expectations.

- **The stubborn-donkey method**: Do not allow situations in which censors are triggered by rules that themselves have censors that are triggered by other rules and nauseam. Fix the depth of censor chaining to some prescribed consumption of resources. This method is so called because the number of times the decision is reversed is limited.

- **The tapered-search method**: Allow any number of levels of censor chaining, but reduce the resources allocated to showing that censors are true in proportion to the depth of chaining.

Extending on the lines of the schemes suggested by Winston, we propose two reasoning schemes for our parallel models [54]:

1. **Evaluate all censors partially.**

   In this method the rule tree is evaluated in breadth first manner, where a processor is allocated to each censor. Assuming $C_1$ in Fig 3.1 to be a collection of censors, viz., $C_{11}, C_{12}, C_{13}, \ldots, C_{1m}$ connected to each other through OR operator, then each of these censors will form $m$ tasks to be evaluated in parallel. The processors start the evaluation concurrently picking the censors one by one starting from left. Each censor is assigned one processor which evaluates it sequentially. The censors are evaluated level by level, i.e., all the censors will be first evaluated till level one, then if there is more time available the next level of censors are picked up for parallel evaluation. Thus under this scheme the censors $C_{11}, C_{12}, C_{13}, \ldots, C_{1m}$ will be first evaluated till first level of censor chaining, then if the resources permit to the second level of censor chaining, and so on. The process of evaluation continues until:

   - all the censors have been fully evaluated
   - any one censor at level one has evaluated to TRUE
   - time limit has elapsed

2. **Evaluate some censors fully.**

   Under this method the rule tree is traversed in depth first manner. The censors are handled one at a time. The evaluation of the censors start from left in the censor list. Each of them is considered as the main task, and evaluated like a separate rule. All the available processors in the system are allocated one censor from the first level. The rule is divided into various censor chaining depths and the censors at each depth are evaluated concurrently till

   - that censor has been fully evaluated
• the net result of the rule is TRUE, \( i.e., \) the current censor evaluates to TRUE

The processors are allocated the next censor in the queue, after they have finished with the evaluation of the current censor. Thus the control will not be transferred to censor \( C_{12} \) till the censor \( C_{11} \) has been fully evaluated.

Haddawy [18] carried out experimentations on various knowledge bases to show that significant savings in inference time can be realized at little expense in terms of precision. This is due to the fact that many knowledge bases produce inference behavior which conforms to the law of diminishing returns in terms of the precision of inference. This implies that the censors at upper levels of censor chaining contribute relatively more to the decision than the ones confined to lower levels of censor chaining.

Based on Haddawy's observation, one can say, that in case of time limit, it would be better to follow the breadth first evaluation scheme, in which all the censors directly connected to the rule are evaluated level by level, going down the censor chaining depth only if the time and other resources permit. Thus even though we have suggested two reasoning schemes to evaluate the rule under time constraint, we have concentrated upon only the first one, in which the attempt has been to look at maximum number of censors, even though not very deeply.

Each of the properties of the censors discussed in the last chapter offers a lot of scope for parallelization in the following manner:

1. The censors have a disjoint set of rules and facts associated with them [30]:

}\]
This implies that all the censors are independent of each other and hence can be evaluated in parallel with minimum synchronization overheads.

2. \( cf\textunderscore decision = cf\textunderscore of\textunderscore premises + \sum_{i=1}^{n} (cf\textunderscore of\textunderscore censor_i) \):

Assuming each censor contributes equally to the certainty of the decision, and the certainty factor obtained after the evaluation of all the premises (\( cf\textunderscore of\textunderscore premises \)) is known, the total number of censors required to be evaluated (\( n \)) to arrive at the decision with a desired certainty (\( cf\textunderscore decision \)) can be calculated in advance. Correspondingly total number of processors that are needed for the efficient execution can also be estimated.

3. The censors are all related to each other through disjunction operators [33]:

While evaluating the censors, as soon as any of them evaluates to TRUE, the processing of all other censors can be terminated and the decision reversed. This gives the advantage of the preemption of tasks as and when found to be redundant. This can also lead to high speedups over sequential evaluation if the censor that has evaluated to TRUE is towards the end of the censor list.

4. The evaluation of rule proceeds in steps.

As only one level of censor chaining is handled at a time, the processors engaged in evaluation of depth \( n \) are freed at the completion of evaluation to be used later for evaluation of depth \( n+1 \). This leads to the efficient utilization of processors.
As an example, consider the rule tree of Fig 3.1. Assuming only the censors are to be evaluated concurrently and there are enough processors available to handle individual censors, the distribution of processors can be carried out in the as shown below:

<table>
<thead>
<tr>
<th>Level</th>
<th>No. Of Censors to be evaluated</th>
<th>No. Of Processors needed (Pri)</th>
<th>Task distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>none</td>
<td>1 (Pr1)</td>
<td>Pr1 - P1, P2, P4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3 (Pr1, Pr2, Pr3)</td>
<td>Pr1 - C1 → P3, P6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pr2 - C2 → P5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pr3 - C4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3 (Pr1, Pr2, Pr3)</td>
<td>Pr1 - C3 → P7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pr2 - C5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pr3 - C6</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1 (Pr1)</td>
<td>Pr1 - C7</td>
</tr>
</tbody>
</table>

The processors used for the evaluation of censors at level 1 are freed after the evaluation of corresponding tasks, to be used by the next level of censors. The process gains importance when the number of censors to be evaluated is much larger than the number of processors available, as this leads to the efficient utilization of processors.

Thus for the most general HCPR, it should be possible to evaluate all the censors, viz.,

\[ C(j_1, j_2, ..., j_{(i-1)}, j_i, 1), \ C(j_1, j_2, ..., j_{(i-1)}, j_i, 2), \ C(j_1, j_2, ..., j_{(i-1)}, j_i, m(j_1, j_2, ..., j_{(i-1)}, j_i)) \]

corresponding to the goal \(A(j_1, j_2, ..., j_{(i-1)})\) concurrently.
The above discussion summarizes, that the censors form a very good source of parallelization, as:

- they can be parallelized with minimum synchronization overheads
- the censors of the same parent are independent of each other, and therefore can be evaluated asynchronously
- they can be sliced into various censor chaining depth, and each level can be handled in a modular manner, thereby leading to efficient utilization of processors.

The process of parallelization of censors is expected to give large speedups over sequential evaluation, when the number of censors associated with any rule are large, and most of the censors are rules with almost equal censor chaining depth.

### 3.2 Evaluating Specificities Concurrently

The specificities and the generalities corresponding to a goal are again HCPRs. Consider the example of following related HCPRs. Only the relevant rules are shown, thereby avoiding unnecessary details:

/*level 0 */
IF $P_1$
THEN $A_1$
SPECIFICITY $A_{11} \oplus A_{12} \oplus A_{13} \oplus \ldots \oplus A_{1n(1)}$

/*level 1 */
IF $P_{13}$
THEN $A_{13}$
SPECIFICITY $A_{131} \oplus A_{132} \oplus A_{133} \oplus \ldots \oplus A_{13n(13)}$
GENERALITY $A_1$

/*level 2 */
IF $P_{133}$
THEN $A_{133}$
SPECIFICITY $A_{1331} \oplus A_{1332} \oplus A_{1333} \oplus \ldots \oplus$
GENERALITY $A_1$

/\* level 3 /\*
IF $P_{133}$
THEN $A_{133}$
SPECIFICITY $A_{1331} \oplus A_{1332} \oplus A_{1333} \oplus \ldots \oplus A_{133n(133)}$
GENERALITY $A_{133}$

/\* level i /\*
IF $P_{(j_1, j_2, \ldots, j(i-1), j_i)}$
THEN $A_{(j_1, j_2, \ldots, j(i-1), j_i)}$
SPECIFICITY $A_{(j_1, j_2, \ldots, j(i-1), j_i, 1)} \oplus A_{(j_1, j_2, \ldots, j(i-1), j_i, 2)} \oplus \ldots \oplus A_{(j_1, j_2, \ldots, j(i-1), j_i, n(j_1, j_2, \ldots, j(i-1), j_i))}$
GENERALITY $A_{(j_1, j_2, \ldots, (i-1))}$

All the decision nodes of the HCPRs can be collectively and systematically represented in the form of a tree, called HCPRs-tree. An example of such a tree is shown in Fig 3.2. The tree indicates only the specificities of the goal $A_1$ going down to the $(i+1)$th specificity level. The goal to be evaluated can be at any level of the tree. If it is not the root of the tree, then it will not hold unless all its generalities conditions are also satisfied.

As an example, consider the case, from Fig 3.2, where the goal that needs to be evaluated is $A_{1333}$. As it appears at the third level of the specificity, it will not hold unless all its direct ancestors up to the root have evaluated to true. The evaluation process starts by traversing the tree in the backward direction from the goal up to the root. This implies evaluating goals, $A_{133}$, $A_{13}$ and $A_1$ in addition to $A_{1333}$. Once all the generality goals including $A_{1333}$ have evaluated to true, the HCPR-tree can be traversed in the forward direction level by level looking for more specific cases of the required goal. To start with the first level of specificities of $A_{1333}$ are evaluated, viz. $A_{13331}$, $A_{13332}$, ..., $A_{1333n(133)}$. Under sequential
Figure 3-2: HCPR tree representing various specificity levels
evaluation, the specificities are evaluated one by one starting from left. As all of them are mutually exclusive to each other, only one of them will evaluate to TRUE. This would also mean terminating the evaluation process of all other specificities at that level. Assuming out of all the specific cases \( A_{13332} \) evaluates to TRUE, then as the next level, the specific cases of \( A_{13332} \) are evaluated.

HCPRs systems facilitate the handling of trade-off between the certainty of conclusion and its specificity. The certainty in the decision decreases as the probe becomes narrower, with the increased level of specificity. Bharadwaj and Jain [3] have suggested a General Control Scheme (GCS) using which the certainty factor of the decision at the \( i \)th level of specificity and maximum level of specificity that can be reached, depending upon desired certainty and various resource constraint, can be calculated. The maximum specificity that can be achieved for a decision depends upon

1. the threshold certainty factor which is set for the decision to hold.
2. various resource constraints like memory or time or both
3. user requirement on precision.

The goals in an HCPR are evaluated using backward chaining technique, looking for facts, to see if the goal is satisfied or not. The reasoning process is HCPRs system is very flexible as it allows both backward (evaluation of generalities of the goal) and forward (evaluation of specificities of the goal) traversal of the HCPRs-tree.

Any two nodes in an HCPR tree are said to be related to each other, if one is a direct descendant or an ancestor of the other. Thus, assuming \( A_{1333} \) to be the starting goal, \( A_{133}, A_{13}, A_1 \) would be called direct ancestors and \( A_{13331}, A_{13332}, \ldots, A_{1333n(1333)} \) as direct descendants, they are all said to be related to each other.
As stated and later explained in chapter 2, each pair of related nodes have a disjoint set of facts and rules associated with them. This property of HCPRs is maintained all through the learning process by applying fission techniques (refer section 2.2).

The above discussion implies that the characteristics of HCPRs system should exploit inherent parallelism in the following manner.

1. **The knowledge in HCPRs is represented in an hierarchical manner**, such that, all the ancestors (generalities) of the goal until the root and all the descendants (specificities) directly connected to it contains a disjoint set of facts and rules. The characteristics of HCPR fits the requirement of exploiting task level parallelism, where the goals can be evaluated with minimum synchronization overheads.

2. **All the specificities of the goal and all its generalities are independent of each other**, implying that they can be evaluated asynchronously. However, for the final tabulation of the result, the goal at level $i$ will said to be TRUE only after all its ancestors from level $i-1$ upto root have also evaluated to TRUE.

3. **From the tree structure, the total number of nodes directly connected to the goal and are related to it can be computed** before the actual evaluation begins. It helps in estimating the number of processors that would be needed for the efficient execution of the parallel model. For the previous example, where the required goal is $A_{1333}$, with 3 levels of generalities and $n(1333)$ specificities directly connected to it, total number of tasks that can be evaluated concurrently, and hence the total number of processors needed for efficient execution would be $n(1333)+4$.

4. **All the specificities of the goal are mutually exclusive to each other**, implying that only one of them can be TRUE at a time. This
allows that the processors associated with the task to be preempted as soon as any one of them evaluates to TRUE. Further as for any goal at level $i$ to hold, all its ancestor up to the root should evaluate to TRUE. If any of the nodes at an intermediate level $j$, such that $j \leq i$, is found to evaluate to FALSE, all the processors attached to tasks at levels greater than or equal to $j$ and less than or equal to $i+1$ can be preempted.

As an example consider Fig 3.2 again with required goal $A_{1333}$. Let us assume the knowledge base to be such that the goal $A_{133}$ evaluates to FALSE. As $A_{1333}$ and goals thereafter are all specific cases of $A_{133}$, there is no need to evaluate them and thus the processors corresponding to these tasks can be preempted. The process of evaluation of tasks $A_{13}$ and $A_{1}$ continues. The final result is obtained after both the generalities, viz., $A_{13}$ and $A_{1}$ have also been fully evaluated. The decision will then said to generally hold.

The process of preemption gains importance, especially if the number of processors available is less than the number of tasks to be evaluated; as now the preempted processors can be handed over to the tasks in waiting.

Parallelization of specificities can be expected to give high speedups over sequential execution under following circumstances:

- **the knowledge is represented in the form of large hierarchy of specificities.**

Assume a general HCPR tree with the goal appearing at the $i$th level of specificity and having $n$ specificities directly connected to it. Further assume the knowledge base to be such that all goals take equal time $t$ to be evaluated, and all the related goals need to be evaluated. Then to evaluate the tree up to the first level of specificity

$$\text{sequential time needed} = (i + n) \times t$$
parallel time needed \( = t + \delta t \)

where \( \delta t \) is the synchronization overheads needed for parallel evaluation. Thus

\[
speedup = \frac{((i + n) \times t)}{(t + \delta t)}
\]

However, due to the hierarchical nature of HCPRs, \( \delta t \) can be considered to be very small. Thus larger the value of \( i + n \), or the level of specificity, more is the expected speedup.

- the facts are such that all the generalities of the goal evaluate to TRUE and all the specificities except for the one towards the end of the specificity list evaluates to FALSE.

Continuing from the last case, if the root of the HCPR tree itself evaluates to FALSE, the total number of goals that will be evaluated sequentially will be equal to 1. Thus

\[
total \text{ sequential time} = t
\]

\[
total \text{ parallel time} = t + \delta t
\]

\[
speedup = \frac{t}{(t + \delta t)}
\]

Parallelization in such type of a case can actually deteriorate the performance of the system. Thus the maximum speedups can be expected in cases where the knowledge base is such that all the generalities of the goal evaluate to TRUE and all the specificities except for the one towards the end of the specificity list evaluates to FALSE.

In the subsequent chapters we have exploited these sources of parallelism in HCPRs to develop two parallel models, viz., Parallel Specificity model and Parallel Censored Production Rules model over shared memory architecture.