CHAPTER 4
PARALLEL EVALUATION OF SPECIFICITIES

Results of this chapter have been presented in the following paper: Bharadwaj, K.K. and Varshneya, Renu, ‘Parallelization of Hierarchical Censored Production Rules’, Information and Software Technology, 37, 8, 453-460, 1995
As discussed in the previous chapters, the knowledge in HCPRs is represented in hierarchical manner, such that, all the ancestors (generalities) corresponding to a goal until the root, and all its direct descendents (specificities) have a disjoint set of rules associated with them. This characteristic of the HCPR fits the requirement of exploiting task level parallelism with minimum synchronization overheads. Further, all the specificities of a goal are mutually exclusive, implying that only one of them can be true at any instance. This allows the specificities to be evaluated asynchronously and also the preemption of tasks, as soon as any one of them evaluates to true. Also as the total number of generalities and the specificities of any goal are known in advance, it helps in estimating the number of processors that would be needed for the efficient execution of the parallel model.

4.1 The Sequential Evaluation

To understand the evaluation process, we can view the rule structure as shown in Fig 4.1. The figure shows the rules connected to each other through their generality and specificity pointers in the form of an HCPR tree. For simplicity purpose, only the generalities and the specificities of the rule have been shown; using light and the bold lines respectively. Further, for this description we have considered only three levels of specificities.

The root goal A1 has null as the generality pointer and n1 specificity pointers, pointing to A11, A12, ... , A1n(1), respectively. Each of these rules have their generality pointer pointing to their parent A1. Specificity conditions of all the rules except for A12 have been ignored. A12 has n12 specificity pointers pointing to A121, A122, ... , A12n(12); with each of their generalities
Figure 4-1: An example of rule representation in HCPRs system
pointing to $A_{12}$. This process of forward chaining continues until all rules have their specificity pointers as null ($A_{1211}$, $A_{1212}$, ..., $A_{121n(121)}$ in this example). The process of evaluation starts by looking for the rule in the rule base. After which it can take one of the following four paths depending upon the knowledge base:

1. **Rule not found.**

   If the rule is not found in the knowledge base, it gives the message that there is no information available about this rule, and the evaluation process is stopped.

2. **Rule has no generality or specificity conditions.**

   The rule is searched for in the fact base. If not present, it is searched for in the rule base, and the rules associated with its premises and censors are evaluated recursively in the same manner as before. If there are no specificities or generalities associated with the rule, the process of evaluation is terminated as soon as the complete rule has been evaluated.

3. **Rule has only specificities.**

   If the rule has no generality condition, implying that it is at the root of the HCPR tree, then the rule is evaluated as in the last case. If there is time available, its specificities are evaluated one by one, starting from left until:

   - the time permits
   - no more specificities are left to be evaluated
   - any one specificity evaluates to TRUE.
For the example of Fig 4.1, rule A1 has no generality and A1, A12, \ldots, A_{1n}(J) specificities which would be evaluated one by one starting with A1 until one of the above conditions is satisfied. Assuming A12 evaluates to true, and there is time available, specific conditions of A12 are extracted and evaluated in the same manner as before. This process of evaluating the specificities of the goal level by level, continues as long as the resources permit.

4. Rule has a valid generality pointer

If the rule has a valid generality pointer, then using this pointer, a backward (generality) link is established until the generality pointer of the rule is found to be null. For fig 4.1, if A121 was the required goal, the generality list would consist of A12 and A1. The process of evaluation starts by evaluating A1, followed by A12 and then A121. Each subsequent goal is evaluated only if the previous one holds. If any of the goal evaluates to FALSE, the process of evaluation is terminated and the previously evaluated goal gives the general answer. This is called the backward traversal of the HCPR tree. After A121 has also been fully evaluated, and is found to hold, the specificities are handled one by one as in 3.

As explained in the last chapter, the HCPR exhibits inherent task level parallelism, hence all the related goals of an HCPR can be evaluated in parallel.

4.2 The Parallel Evaluation

The Parallel Specificity model has been designed for shared memory architecture. The overall architecture is as shown in fig 4.2. The parallel system consists of a control process, main task of which is to extract the related goals from the HCPR tree and place them in the task queue for parallel execution.
Figure 4-2: Overview of the Parallel Specificity model
We have adopted the task queue approach as it solves the load balancing problem to a large extent that could arise because of the unequal sizes of the tasks, as in static scheduling. The control process assigns these tasks to various processors as and when they are found to be idle, which evaluate them using a local copy of their inference engine. The rules and facts for the evaluation of the tasks are obtained from the knowledge base placed in the shared area. The results obtained after evaluation are placed in the status array in the shared area, which has a separate entry for each task. These results are interpreted by the control process, and used for aborting the tasks which are found to be redundant.

The control process extracts the relevant goals from the HCPR tree and places them in the task queue, using the following algorithm.

```c
form_task_queue();
{ accept goal ;
   If goal not in knowledge base
      return ( 'rule not found' );
   /* add the query to the task queue */
       add goal to the task queue ;
   /* add all the generalities until the root to
      the task queue */
       while goal → generality ≠ null
          { add goal → generality to the task queue ;
            goal = goal → generality ;
          }
   /* add all the specificities of the query to
      the task queue */
   for all the specificities of the query
      { extract specificity;
         add to the task queue ;
      }
   /* Evaluate tasks in parallel by assigning them
      to different processors */
   for all tasks in the task queue
      { assign task from the task queue to the
        idle processor ;
        evaluate_rule (task) in parallel ;
        check_for_preemption of tasks ;
      }
}
```
The task queue and the knowledge base (fact base and the rule base) are placed in the shared memory. Each processor contains a copy of the inference engine in its local memory, which it uses to evaluate the rules from the shared area, corresponding to the task assigned to it. The inference engine is implemented using the following algorithm.

```c
evaluate_rule(rule);
{ /* Search for the rule in the fact base */
  if rule in fact base
  { if ((rule is a premise with value true) or
       (rule is a censor with value false))
     return (TRUE)
   else
     return (FALSE);
  }

  /* Search for the rule in the rule base */
  else if rule in rule base
  { while ((not all premises and
           censors of the rule evaluated)
          and (result of the evaluated
               rule = TRUE))
     evaluate_rule(rule);
  }

  else /* The rule is not present in the knowledge base, thus using Close World Assumption */
   return FALSE [45]*/
   return (FALSE);
}
```

Though the censor conditions are also the set of rules, just like premises, they are interpreted slightly differently. Not all censor conditions need to be evaluated. Censors are basically the exceptions, with their chances of evaluating to true very less. However, the total absence of censors with incomplete knowledge might result in the uncertainty of the decision. Each censor condition has a certainty factor associated with it [18], which if the censor is false adds to the overall certainty of the decision and if true, negates the decision all together.

In the present work, the specificity varies while the certainty remains constant (maximum). Thus the process of evaluating the rule continues until
• all the premises and the censors have been evaluated; or
• any premise has resulted into false or any censor into true value, or
• the rule is not present in the knowledge base, in which case its value is assumed to be false.

The task queue contains the rule numbers corresponding to the goals, along with their level in the HCPR hierarchy. The levels to the various goals are assigned in the manner shown in table 4-1. The query is itself considered to be at level 0. The negative levels correspond to the generalities, while level 1 corresponds to the specificities directly connected to the query. The level numbers decrease in the upward direction of the HCPR tree.

These level numbers associated with the various tasks, are used for the preemption of the tasks in the following manner.

• Abort all the tasks at levels greater than and equal to the level i, if the task at level i evaluates to FALSE.
• Abort all the tasks at level 1 (specificities), if any one task at level 1 evaluates to true.

The final result is interpreted by the control process after all the tasks have been evaluated. The goal is said to be true only if all its generalities evaluate to true.

Salient features of the Parallel Specificity model are as follows:

1. Very little communication between the processors.

The processors in the parallel specificity model have two types of processes assigned to them. One is the control process which creates the tasks, allocate them to the processor when any processor is found to be idle and preempts the tasks whenever they are found to be
Table 4.1: Task queue contents as stored by the control process for general HCPR

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Task (goal)</th>
<th>Level</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A(i, i_2, \ldots, i(i-1), i_i)</td>
<td>0</td>
<td>required goal</td>
</tr>
<tr>
<td>2</td>
<td>A(i, i_2, \ldots, i(i-1))</td>
<td>-1</td>
<td>first level generality</td>
</tr>
<tr>
<td>3</td>
<td>A(i, i_2, \ldots, i(i-2))</td>
<td>-2</td>
<td>second level generality</td>
</tr>
<tr>
<td>(i)</td>
<td>A(_i)</td>
<td>-(i-1)</td>
<td>the root</td>
</tr>
<tr>
<td>(i + 1)</td>
<td>A(i, i_2, \ldots, i(i-1), i_i, 1)</td>
<td>1</td>
<td>first specificity at level 1</td>
</tr>
<tr>
<td>(i + 2)</td>
<td>A(i, i_2, \ldots, i(i-1), i_i, 2)</td>
<td>1</td>
<td>second specificity at level 1</td>
</tr>
<tr>
<td>(i + n(i_1, \ldots, i_i))</td>
<td>A(_i, i_2, \ldots, i, n(i_1, \ldots, i_i))</td>
<td>1</td>
<td>nth specificity at level 1</td>
</tr>
</tbody>
</table>
redundant. The other processors evaluate the rules corresponding to the tasks assigned to them. As each related goal in the HCPR tree has a disjoint set of facts and rules associated with it, there is little communication needed between the various processors running their own local inference engines. The control process, however, needs to communicate with all other processors, whenever it has to allocate the task to them, evaluate the final result or preempt the task assigned to them if needed.


Synchronization overheads can lead to potential serial bottlenecks, if the tasks are such that their execution can begin only at a specific instance. As most of the tasks in parallel specificity model corresponds to specificities of a goal, which are mutually exclusive to each other, there is no synchronization needed between them. Some amount of synchronization, at the time of final tabulation of result is needed between the tasks corresponding to the evaluation of the query and its generalities. The query cannot said to hold unless all its generalities have evaluated to TRUE.


The task queue approach adopted in the parallel specificity model ensures dynamic distribution of tasks amongst processors. Under this scheme each processor collects a task from the task-queue, processes the task and becomes idle again to look for more work in the queue. This solves the problem of load balancing to a large extent as the processors with smaller tasks need not wait for other tasks to be completed before being assigned a fresh task, but can immediately get more work from the task queue if available.
4. Preemption of tasks as and when found to be redundant.

All the specificities of the goal in an HCPR system are mutually exclusive to each other, implying that only one of them can be true at a time. Thus during the evaluation process as soon as any one of them evaluates to TRUE, the tasks corresponding to other specificities can be preempted thereby freeing the corresponding processors to take up other work. The tasks corresponding to levels greater than \( i \) are also preempted if the task at level \( i \) evaluates to FALSE.

4.3 Implementation Details

The model has been simulated for a shared memory architecture using a general purpose simulator MultiPascal [31]. Relevant details of MultiPascal can be obtained from Appendix A.

4.3.1 Data Structures

The main data structures used are the knowledge base and the task queue.

*The Knowledge Base*

Knowledge base contains the information about all the rules and facts available for the HCPRs system. The structure used for knowledge base is an array of records where each record represents a rule with the following structure:

\[
\text{rule_definition} = \text{record} \begin{align*}
\text{rule_name} & : \text{string} ; & /* \text{name of the rule} */ \\
\text{rule_value} & : \text{boolean} ; & /* 1 \Rightarrow \text{fact with true value} \\
& & 0 \Rightarrow \text{fact with false value} */ \\
\text{premise} & : \text{list} ; & /* \text{elements of the list are the rule number corresponding to the premise} \\
\text{censor} & : \text{list} ; & \text{or the censor, along with a pointer to} \\
& & \text{the next premise / censor} */
\end{align*}
\]
generality : integer ; /* rule number of the generality */
specificity : list ; /* pointers to the specificities */
end;

The rule is identified by the rule name for the first time when the query is being searched for in the knowledge base. For all subsequent references, corresponding to the premises, censors, generality and specificities, the rule is identified by the rule number, which is same as its index in the knowledge base. The reason behind using an array structure rather than a linear list structure is that in an array, each element can be accessed separately and individually, which makes it more suitable than a linear list for parallel implementation.

Premise and the censor lists are used to store the rule numbers corresponding to all the premises and the censors of the rule respectively. Rule value, for such type of entries, has no meaning, till all its premises and censors have been evaluated. Facts, obviously, will have no premises and censors associated with them. Rule value field in this case indicates the true / false value of the fact. Generality field of the rule contains the rule number corresponding to its parent node. Specificity list contains a list of rule numbers corresponding to the specificities of the rule. Both the generality and the specificity fields will be null for facts and for rules which do not have any generalities or specificities associated with them.

As an example consider a set of hierarchical censored production rule.

```
IF P11 AND P12 AND P13 AND P14 /* Rule */
THEN A1
UNLESS C11 OR C12
GENERALITY A
SPECIFICITY A11 XOR A12 XOR A13 XOR A14 XOR A15

IF / ] /* Fact in the form of an
THEN P11
```

HCPR*/

63
Figures 4.3(a) and 4.3(b) show the internal representation of the rules and facts in the knowledge base. The names of the rules are stored in the form of a string. In the above example they are AI and PII respectively. The second field determines whether the entry corresponds to a rule (AI) or a fact (PII). In case of a fact, the entry corresponds to its value ( 1 means TRUE ). For the rule entries, the remaining fields contain a list of the rule numbers corresponding to its premises (PII, PII, PII, PII), the censors (CII, CII), the generality (A) and the specificities (AII, AII, AII, AII, AII) respectively. Assuming rule AI is the first entry in the rule base, the rule number for AI will be 1 and similarly for PII it can be assumed to be 4. For the fact entry (PII) in the knowledge base, the field entries for the premises, censors, generality and the specificities are null. All other rule number entries are also simple assumptions.

The Task Queue

The task queue contains the rule numbers for all the tasks that can be evaluated in parallel, along with their level numbers. It is defined as an array of records with each record having the following structure.

\[
\text{task\_queue\_entry = record}
\begin{align*}
\text{rule\_number : integer ;} \\
\text{level : integer ;} & \quad /* \text{level of the rule} */
\end{align*}
\]

An array of integers called abort array is defined for preemption of tasks. Each entry in the abort array defines whether the corresponding task from the task
Fig 4-3: (a) Rule and (b) Fact representation in Parallel Specificity Model
queue needs to be aborted. Each task is assigned an index in this array which it uses to flag to the tasks that need to be aborted.

For our present simulation we have assumed that

- The rules are numbered from 1 to n.
- The processors are numbered from 1 to m.
- All censors have to be evaluated.
- There are no specificities and generalities associated with the rules and facts corresponding to the premises and the censors.

### 4.3.2 Sample Run

To understand the algorithms and the implementation of the parallel specificity model better, let us consider a small example of an HCPR system.

R1: IF P11 AND P12
    THEN A1
    UNLESS C11
    GENERALITY [ ]
    SPECIFICITY A11 XOR A12 XOR A13

R2: IF P111 AND P112
    THEN A11
    UNLESS C111 OR C112
    GENERALITY A1
    SPECIFICITY [ ]

R3: IF P121 AND P122
    THEN A12
    UNLESS [ ]
    GENERALITY A1
    SPECIFICITY A121 XOR A122

R4: IF P131
    THEN A13
    UNLESS [ ]
    GENERALITY A1
    SPECIFICITY [ ]
R5: IF $P1211$ AND $P1212$ AND $P1213$
THEN $A121$
UNLESS {}
GENERALITY $A12$
SPECIFICITY {}

R6: IF $P1221$
THEN $A122$
UNLESS {}
GENERALITY $A12$
SPECIFICITY {}

For simplicity purpose we can assume all premises and all censors to be facts and stored in the knowledge base continuously in the order of their appearance. The rule numbers for them would thus start from 7. The above HCPRs can be represented in the form of a tree as shown in fig 4.4. The knowledge base is as shown in fig 4.5.

Fig 4-4: HCPR tree for sample run example
<table>
<thead>
<tr>
<th>A1</th>
<th>2</th>
<th>P11</th>
<th>P12</th>
<th>φ</th>
<th>C11</th>
<th>φ</th>
<th>φ</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>φ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A11</td>
<td>2</td>
<td>P111</td>
<td>P112</td>
<td>φ</td>
<td>C111</td>
<td>φ</td>
<td>1</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A12</td>
<td>2</td>
<td>P121</td>
<td>P122</td>
<td>φ</td>
<td>φ</td>
<td>5</td>
<td>6</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A13</td>
<td>2</td>
<td>P131</td>
<td>φ</td>
<td>φ</td>
<td>1</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A121</td>
<td>2</td>
<td>P1211</td>
<td>P1212</td>
<td>φ</td>
<td>P1213</td>
<td>φ</td>
<td>3</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A122</td>
<td>2</td>
<td>P1221</td>
<td>φ</td>
<td>φ</td>
<td>3</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P11</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P12</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C11</td>
<td>0</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P111</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P112</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C111</td>
<td>0</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C112</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P121</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P122</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P131</td>
<td>0</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1211</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1212</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1213</td>
<td>0</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1221</td>
<td>1</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Fig 4-5: Knowledge base for sample run example*
Let us assume the query to be $A_{12}$. The query is first searched in the knowledge base. It exists there at the index 3 (which is also called as its rule number). From the knowledge base the generalities and the specificities of the query are extracted and are placed in the task queue. There will be four entries in the task queue as shown in table 4-2.

The first entry is the rule number corresponding to the task which has to be evaluated in parallel. Second entry corresponds to the level number, indicating the level of hierarchy in the HCPR tree with respect to the query.

<table>
<thead>
<tr>
<th>Task number</th>
<th>Rule Number</th>
<th>Level</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>3</td>
<td>0</td>
<td>queried goal</td>
</tr>
<tr>
<td>T2</td>
<td>1</td>
<td>-1</td>
<td>generality</td>
</tr>
<tr>
<td>T3</td>
<td>5</td>
<td>1</td>
<td>specificity 1</td>
</tr>
<tr>
<td>T4</td>
<td>6</td>
<td>1</td>
<td>specificity 2</td>
</tr>
</tbody>
</table>

Level number assigned to the query is 0, it has one ancestor (rule number 1), the level of which is assigned -1, indicating it is one up the query in the hierarchy. There are two specificities (rule numbers 5 and 6), the level number assigned to each of them is 1.

From the knowledge base shown in fig 4.5, it can be seen that the facts are such that rules $A_{1}, A_{12}$ and $A_{122}$ evaluate to TRUE and all others evaluate to FALSE. Assuming there are five processors, viz., $P_{r1}, P_{r2}, P_{r3}, P_{r4}, P_{r5}$, available in the system, the first free processor, say $P_{r1}$ is assigned the control.
process which will form the task queue as shown in table 4-2, and assign the tasks to the idle processors. Let us assume the tasks are assigned in the manner shown below.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Task Number</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr2</td>
<td>T1</td>
<td>Corresponding to the query A12</td>
</tr>
<tr>
<td>Pr3</td>
<td>T2</td>
<td>Corresponding to the generality A1</td>
</tr>
<tr>
<td>Pr4</td>
<td>T3</td>
<td>Corresponding to the first specificity A121</td>
</tr>
<tr>
<td>Pr5</td>
<td>T4</td>
<td>Corresponding to the second specificity A122</td>
</tr>
</tbody>
</table>

All the four tasks begin the evaluation of corresponding goals together. Task T1 has 2 premises to evaluate, task T2 has 2 premises and 1 censor, T3 has 3 premises while T4 has only one premise to evaluate. Assuming the evaluation of each premise / censor takes equal time to evaluate, Pr5 completes task T4 first and finds the value to be TRUE. As this corresponds to one of the specificities of the goal A12, it will send the signal to the control process, which sets the abort flag for the task T3, being executed by the processor Pr4, corresponding to the specificity of the same parent as task T4, viz., A12, as equal to one. Pr4 on receiving the signal aborts the task T3. Execution of tasks T1 and T2 continue without obstruction. The process is completed when all the tasks (T1 and T2) have also finished their evaluation.

If there more specificities of the goal A122 available, and more time available with the user, then if he so desires, the goal A122 becomes the target goal, and its specificities are evaluated as earlier. The task queue now contains only the specificities of A122, the generalities are not evaluated, and their result is simply carried forward from the previous evaluations. The result in this case is said to be even more specific as it corresponds to the second level of specificities of the query A12.
4.4 Measurement Techniques and Results

The control process previously described is used to monitor and time the processing. Measurements begin at the point after the query has been searched for in the knowledge base. The time taken to build the task queue, allocate the tasks to the processors and preempt the tasks, if required, are all added as the overheads in the parallel execution. For the experimentation, a rule set with 30 HCPRs involving about 1500 rules with the number of specificities ranging from 0 to 7 are considered (Fig 4.6). The rule naming scheme followed is same as described in section 2.2. A17772 is the second specificity of the goal A1777, while P113 is the third premise of the goal A11. The experimentation includes the following cases:

Case I: has the query as A1777 appearing at the third level of specificity in the HCPR hierarchy (Fig 4.7(a)). It has 3 generalities and 7 specificities, ranging from A17771 through A17777 associated with it. The knowledge base is such that the premises P11, P177711, P177721, P177731, P177741, P177751 and P177761 are FALSE rest all other premises are TRUE and censors FALSE. Thus the HCPRs A1, A17771, A17772, A17773, A17774, A17775 and A17776 will evaluate to FALSE and rest all other HCPRs to TRUE. Total number of goals that can be evaluated concurrently is 11. Thus once A1 has evaluated to FALSE, all the other tasks in progress would be preempted.

Case II: has the same goal as in Case I, except that premise P11 in this case is TRUE, implying that all the generalities of the goal evaluate to TRUE and all the specificities, except for the last one, viz., A17777 evaluate to FALSE. The tasks corresponding to specificities (A17771, A17772, A17773, A17774, A17775, A17776) still in progress will be preempted as soon as the task corresponding to A17777 is completed.
Fig 4-6: HCPR tree considered for experimentation
Case III: has All as the goal with no specificity and 1 generality condition (Fig 4.7(b)). Thus total number of tasks to be evaluated concurrently is equal to 2. The knowledge base is such that both the goals evaluate to TRUE. No preemption of tasks is done.

Case IV: (Fig 4.7(c)) has AI as the goal, no generalities and seven specificities ranging from All through A17. The knowledge base is such that the goal AI and the first specificity All evaluate to TRUE and rest all specificities to FALSE. Total number of tasks to be evaluated concurrently is 8. Tasks corresponding to the remaining specificities (A12, A13, A14, A15, A16, A17) still in progress will be preempted, as soon as the specificity All evaluation is completed.

Case V: (Fig 4.7(d)) has All as the goal appearing at the fourth level of specificity in the HCPR hierarchy, with no specificities attached to it. The knowledge base is such that no tasks need to be preempted.

Table 4-3 summarizes all the above discussed cases. Each of these cases is simulated for shared memory architectures. The number of processors in each case is varied from 1 to 11.

The objective of the experiment is to:

- show the relationship between the speedup and the number of processors with respect to the number of goals present in the HCPR that need to be evaluated
- the effect of the knowledge base on the speedup for same number of goals and processors
- the effect on processor utilization with different number of processors for various cases

Speedup is defined as:

\[ S_p = \frac{T(1)}{T(p)} \]
Figure 4-7: Graphical representation for (a) CASEs I and II (b) CASE III (c) CASE IV (d) CASE V
Table 4-3: Cases for the experimentation of Parallel Specificity model

<table>
<thead>
<tr>
<th>Cases</th>
<th>Query</th>
<th>Number Of Tasks</th>
<th>Number Of Generalities</th>
<th>Number Of Specificities</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| I     | A1777 | 11              | 3                      | 7                       | P11 - FALSE  
P177711 - FALSE  
P177721 - FALSE  
P177731 - FALSE  
P177741 - FALSE  
P177751 - FALSE  
P177761 - FALSE  
rest all premises TRUE  
all censors FALSE |
| II    | A1777 | 11              | 3                      | 7                       | P177711 - FALSE  
P177721 - FALSE  
P177731 - FALSE  
P177741 - FALSE  
P177751 - FALSE  
P177761 - FALSE  
rest all premises TRUE  
all censors FALSE |
| III   | A11   | 2               | 1                      | 0                       | all premises TRUE  
all censors FALSE |
| IV    | A1    | 8               | 0                      | 7                       | P121 - FALSE  
P131 - FALSE  
P141 - FALSE  
P151 - FALSE  
P161 - FALSE  
P171 - FALSE  
rest all premises TRUE  
all censors FALSE |
| V     | A17777 | 5                | 4                      | 0                       | all premises TRUE  
all censors FALSE |
where $T(l)$ is the sequential time and $T(p)$ is the time taken by $p$ processors to complete the same task.

Efficiency or the processor utilization is defined as:

$$E_p = \frac{S_p}{p}$$

where $p$ specifies the number of processors.

Table 4-4 shows the speedups and the processor efficiency obtained for different cases for various number of processors. The speedups for different number of processors for various cases are represented graphically in Fig 4.8, and the processor efficiency in Fig 4.9 for all the cases discussed above.

Case I (Table 4-3, Fig 4.7(a)) consists of 11 tasks in the task queue. In the sequential execution, the goals $A1777, A177, A17$ are evaluated successfully. However, the evaluation of the goal $A1$ results into FALSE value thereby reversing the decision. Subsequently, the specificities are not evaluated. In parallel execution, the specificities are also included in the task queue for concurrent execution. The performance of the system increases linearly until four processors, after which, the speedup gradually starts becoming stagnant. Processors 5 and above are actually the overheads above the sequential execution (Fig 4.8).

The second case (Table 4-3, Fig 4.7(a)), though consists of the same number of tasks as in case I, but because the facts are such that all the 11 tasks are fully evaluated, the speedups obtained are better than the first case for the same number of processors. The speedup for 4 and 5 processors, then from 6 to 10 processors is constant (Fig 4.10). For 11 processors, the speedup suddenly increases to 8.85. This behavior is attributed because of the load balancing problem that occurs because of the large size of the tasks associated with each processor. When the number of processors are six, the tasks to various processors will be allocated in the manner as shown in Table 4-5(a). Impling
Table 4.4: Speedups and processor efficiency for various number of processors

<table>
<thead>
<tr>
<th>No of Procs</th>
<th>CASE I</th>
<th></th>
<th>CASE II</th>
<th></th>
<th>CASE III</th>
<th></th>
<th>CASE IV</th>
<th></th>
<th>CASE V</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.58</td>
<td>0.79</td>
<td>1.78</td>
<td>0.89</td>
<td>2.0</td>
<td>1.0</td>
<td>1.92</td>
<td>0.96</td>
<td>1.94</td>
<td>0.97</td>
</tr>
<tr>
<td>3</td>
<td>2.08</td>
<td>0.69</td>
<td>2.58</td>
<td>0.86</td>
<td>2.0</td>
<td>0.67</td>
<td>2.49</td>
<td>0.83</td>
<td>2.99</td>
<td>0.97</td>
</tr>
<tr>
<td>4</td>
<td>2.61</td>
<td>0.65</td>
<td>3.35</td>
<td>0.83</td>
<td>2.0</td>
<td>0.50</td>
<td>3.54</td>
<td>0.89</td>
<td>3.03</td>
<td>0.76</td>
</tr>
<tr>
<td>5</td>
<td>2.86</td>
<td>0.57</td>
<td>3.35</td>
<td>0.67</td>
<td>2.0</td>
<td>0.40</td>
<td>3.55</td>
<td>0.71</td>
<td>3.96</td>
<td>0.79</td>
</tr>
<tr>
<td>6</td>
<td>2.88</td>
<td>0.48</td>
<td>4.67</td>
<td>0.78</td>
<td>2.0</td>
<td>0.33</td>
<td>3.55</td>
<td>0.59</td>
<td>3.96</td>
<td>0.66</td>
</tr>
<tr>
<td>7</td>
<td>2.90</td>
<td>0.41</td>
<td>4.69</td>
<td>0.67</td>
<td>2.0</td>
<td>0.29</td>
<td>3.56</td>
<td>0.51</td>
<td>3.96</td>
<td>0.57</td>
</tr>
<tr>
<td>8</td>
<td>3.49</td>
<td>0.43</td>
<td>4.69</td>
<td>0.59</td>
<td>2.0</td>
<td>0.25</td>
<td>6.76</td>
<td>0.85</td>
<td>3.96</td>
<td>0.50</td>
</tr>
<tr>
<td>9</td>
<td>3.49</td>
<td>0.39</td>
<td>4.69</td>
<td>0.52</td>
<td>2.0</td>
<td>0.22</td>
<td>6.76</td>
<td>0.75</td>
<td>3.96</td>
<td>0.44</td>
</tr>
<tr>
<td>10</td>
<td>3.52</td>
<td>0.35</td>
<td>4.69</td>
<td>0.47</td>
<td>2.0</td>
<td>0.20</td>
<td>6.76</td>
<td>0.68</td>
<td>3.96</td>
<td>0.40</td>
</tr>
<tr>
<td>11</td>
<td>3.77</td>
<td>0.34</td>
<td>8.85</td>
<td>0.80</td>
<td>2.0</td>
<td>0.18</td>
<td>6.76</td>
<td>0.61</td>
<td>3.96</td>
<td>0.36</td>
</tr>
</tbody>
</table>
Figure 4-8: Speedups for different number of processors for various knowledge bases.
Figure 4-9: Processor efficiency of different processors for various knowledge bases.
Figure 4-10: Speedups and processor efficiency for CASE II
Table 4-5: Processor allocation scheme followed by Parallel Specificity model (a) for 6 processors (b) for 10 processors

<table>
<thead>
<tr>
<th>Processor</th>
<th>Pr1</th>
<th>Pr2</th>
<th>Pr3</th>
<th>Pr4</th>
<th>Pr5</th>
<th>Pr6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>A1777</td>
<td>A177</td>
<td>A17</td>
<td>A1</td>
<td>A17771</td>
<td>A17772</td>
</tr>
<tr>
<td>Allocated</td>
<td>A17773</td>
<td>A17774</td>
<td>A17775</td>
<td>A17776</td>
<td>A17777</td>
<td></td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Processor</th>
<th>Task Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr1</td>
<td>A1777, A17777</td>
</tr>
<tr>
<td>Pr2</td>
<td>A177</td>
</tr>
<tr>
<td>Pr3</td>
<td>A17</td>
</tr>
<tr>
<td>Pr4</td>
<td>A1</td>
</tr>
<tr>
<td>Pr5</td>
<td>A17771</td>
</tr>
<tr>
<td>Pr6</td>
<td>A17772</td>
</tr>
<tr>
<td>Pr7</td>
<td>A17773</td>
</tr>
<tr>
<td>Pr8</td>
<td>A17774</td>
</tr>
<tr>
<td>Pr9</td>
<td>A17775</td>
</tr>
<tr>
<td>Pr10</td>
<td>A17776</td>
</tr>
</tbody>
</table>

(b)
that most of the processors will be busy, most of the time during the parallel execution. If we assume, the time taken to execute each task is same (say $T$), then the total time taken to complete the parallel execution is equal to $2T$. Now if we increase the number of processors to say 10 (portion © of figure 4.10), the allocation of tasks will be as shown in table 4-5(b). That is, if we again assume $T$ to be the time taken to complete the execution of one task, the total time to complete the parallel execution will be $2T$. The efficiency of processors gradually decreases when the number of processors is varied from 6 to 10. This is because most of the processors are idle most of the time. The above discussion implies that as long as the number of tasks is an integral multiple of the number of processors available, the speedup increases almost linearly, after which it becomes stagnant even if the number of processors are increased thereby reducing the processor efficiency (portions (a) and © of figure 4.10). Further, cases II, III (Fig 4.7(b)), IV (Fig 4.7©) and V (Fig 4.7(d)), also indicate that the speedup will linearly increase till the number of processors is equal to the number of goals after which it becomes constant.

The following observations can be made about the results. The speedups have been found to be better for the cases where:

1. the total number of goals to be evaluated in parallel is large (Cases II, III, IV and V).
2. the knowledge base is such that the specificity that turns out to be TRUE is towards the end of the specificities list, and all the generalities of the goal are TRUE (Cases I and II).
3. the number of tasks is an integral multiple of the number of processors available (Cases II and IV).

Next chapter deals with the design and implementation of Parallel Censored Production Rules model. In this model all the premises and the censors of a rule in an HCPR are evaluated concurrently.