The Meta Inferences Engine : a tool to use metaknowledge

Jean-Marc NIGRO
Institut Charles Delaunay
Université de Technologie de Troyes,
12 Rue Marie Curie, BP 2060,
10010 Troyes Cedex,
nigro@utt.fr

Yann BARLOY
Institut Charles Delaunay
Université de Technologie de Troyes,
12 Rue Marie Curie, BP 2060,
10010 Troyes Cedex,
barloy@utt.fr

Abstract

The Meta Inferences Engine (MIE) is an inferences engine not only enabling the activation of rules, but also meta-rules. So it belongs to systems managing metaknowledge. This article presents the structure used, as well as the functioning of three versions of MIE. The system was tested within the framework of the CASSICE project and more particularly with the rule-based system IDRES. The results show that MIE is operational and that the last version is faster than the first two ones. Nevertheless the meta inferences engine still contains gaps and the future versions should be corrected.

Keywords: Inferences Engine, Rule and Meta-rule-based system, Metaknowledge.

1 Introduction

The development of a rule-based system is relatively easy because the designer has only to conceive the various rules and the base of facts. Then, the execution is automatic thanks to an inferences engine which has already been developed. If they are conceived cleanly, rule-based systems also have the advantage of being easy to understand and to modify.

However, this technique has several disadvantages:
1. The inferences engine has trouble with administering big bases of facts and rules.
2. The increase of the base of facts provokes an exponential time of execution.
3. Rules cannot be modified, be added or deleted during the execution.
4. It is not always easy to express knowledge of a domain with rules [6].

5. Most rules are only used during a very short time. On the other hand, they are permanently tested with the inferences engine.
6. The strategy used to choose rules (Depth, Breadth ...) is not always adapted to the developed base of rules.

Most of the time, some solutions are used in order to accelerate the execution of a rules based system:
1. To order rules and conditions
2. To add conditions or conclusions to rules to constrain an execution order
3. To use external procedures
4. To regroup rules in packages of rules

Most of these solutions can be considered as being trickies of programming, to obtain a faster running. However, they do not always correspond to a satisfying solution by their "implicit and hidden" aspects (solution 1 and 3), or by the addition of information only added to connect rules or packages of rules. Most of these solutions are brakes in developments of the techniques of Artificial Intelligence (AI) such as symbolic learning or explanatory systems. Indeed, some pieces of information are not explicit and others will be flooded in the rules description.

It is in this context that the idea to conceive a new inferences engine, called MIE (Meta-Inferences Engine) appeared. This idea was already introduced in a theoretical way by Clancey [2] or Torsun [13] who developed a logic allowing the programming of a "meta" level by using a language like PROLOG. The development of a tool allowing the manipulation of metaknowledge [10] would facilitate the implementation of AI techniques and would
directly answer to disadvantages 3, 5 and 6 described above.

So, the meta-inferences engine allows the designer to develop systems based on rules and meta-rules. Meta-rules are executed as rules. Contrary to most of systems which use meta-rules [12] in a static way, the trigger of meta-rules can modify dynamically the rules structure during the session.

First of all, the article presents the domain of application (the CASSICE project) and notably the rule-based system IDRES [9] which is used as a reference to test MIE. The functioning of MIE is then proposed, followed by the experiments of three versions (MIEv1, MIEv2 and MIEv3). Then, a critical analysis of the system is developed.

2 The domain of application

The objective aim of the CASSICE project [11] is the realization of a computerized system capable of listing real situations of real driving. This project is based on a collaboration with many French research laboratories. It uses an Experimental Vehicle (EV) equipped with a camera and a set of proprioceptive sensors (rev-counter, speed, speed of wheels rotation, lighting and road marking, sensor of wheel angle, accelerometer) which supplies different values quoted in table 1. Each lines of the table 1 will be called “data lines” in the article. The meaning of data is presented in the table 2.

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>X (m)</th>
<th>Y (m)</th>
<th>V (m/s)</th>
<th>Teta (deg)</th>
<th>Acc (m/s²)</th>
<th>Phi (deg)</th>
<th>Rg (m)</th>
<th>Rd (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>32.0</td>
<td>0.0</td>
<td>15</td>
<td>0.00</td>
<td>0.00</td>
<td>-3.50</td>
<td>0.00</td>
<td>1.50</td>
</tr>
<tr>
<td>0.02</td>
<td>31.8</td>
<td>0.0</td>
<td>15</td>
<td>0.00</td>
<td>0.00</td>
<td>-3.50</td>
<td>0.00</td>
<td>1.50</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
| 1.14    | 15.2  | 2.1   | 15      | -0.46      | 3.00       | -1.41     | 3.59  | ...

Table 1: Acquired data with EV

The first level of the IDRES system is constituted with a base of rules. The objective is to recognize the possible states of the vehicle at each time of a session. Several states can be candidates simultaneously. For example, when EV is at the point of filtering on the left lane to overtake TV, then two states can be recognized: Beginning left lane change and Overtaking Intent.

The second level tries to refine the selection of the first level to recognize maneuvers (here, an overtaking): when several states were confirmed simultaneously, it will keep only the one which was dealt with. The following principles were adopted:

- a maneuver is decomposed in a sequence of states of which it is important to respect the order of realization. So, the overtaking maneuver was decomposed into ten states: Waiting for overtaking, Overtaking intent, Beginning left lane change, Crossing left discontinuous line, End left lane change, Passing, End of passing, Beginning right lane change, crossing right discontinuous line, End right lane change.
- during a maneuver, some state cannot be recognized. For example, during an overtaking, the Overtaking intent, corresponding to the release of the flashing light, is optional.
- when the system is not capable to determine a state at T time, it uses the concept of persistence of the previous state. This persistence is validated only during a short lapse of time.

Table 2: Meaning of data.

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Clock (s)</th>
<th>Acc</th>
<th>EV's relative acceleration with regard to TV1 (m/s²)</th>
<th>Phi</th>
<th>Angle of EV's front wheels (degree)</th>
<th>Rd</th>
<th>EV's position with regard to right side of the road (m)</th>
<th>Rg</th>
<th>EV's position with regard to left side of the road (m)</th>
<th>Teta</th>
<th>Angle of the target VC (degree)</th>
<th>V</th>
<th>EV's relative speed with regard to TV (m/s)</th>
<th>X</th>
<th>Relative axial position of TV in reference to EV (m)</th>
<th>Y</th>
<th>Relative lateral position in reference to EV (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The first level of the IDRES system is constituted with a base of rules. The objective is to recognize the possible states of the vehicle at each time of a session. Several states can be candidates simultaneously. For example, when EV is at the point of filtering on the left lane to overtake TV, then two states can be recognized: Beginning left lane change and Overtaking Intent.

The second level tries to refine the selection of the first level to recognize maneuvers (here, an overtaking): when several states were confirmed simultaneously, it will keep only the one which

1 TV: Target Vehicle. TV is the vehicle targeted by EV to make a maneuver.
is more "forward" in the maneuver. The figure 1 presents the articulation of the both levels.

Figure 1: The two levels of decision.

IDRES is operational and gives good results. Nevertheless, its execution presents some difficulties: the simultaneous treatment of more than 40 lines of acquisitions causes a very slow time of answer. The system is so compelled to cut the sequence in intervals of 30 lines of acquisitions and to make the recognition of the maneuver on every interval.

Figure 2: Recognition of overtaking maneuver.

By looking at the release of the rules of the first IDRES's level, we notice that most of them are useful only during a relatively short lapse of time. In addition, they require many resources (memory and time) during the running of inferences engine. The rectangular light-grey zone in figure 2 shows that the matchings of the conditions of all rules with all the facts are tested from the beginning to the end of the session.

To resolve this problem, a solution would be to remove these rules when they cannot be executed any more. The use of MIE would allow creating meta-actions which would remove some rules in a convenient moment (cf. the meta-rule Beginning_left_lane_change).

Figure 3: Validity of rules by the use of meta-actions.

The use of these meta-actions in each rule of the first level gives a more effective system. The difference between the light gray zone of figures 2 and 3 show a gain of more than 55 % in term of resources used for the matching of the rules.

Rule Beginning_left_lane_change
If  EV and TV on the same lane
Wheel are turning to the left during 0.2 s
EV behind TV
Movement towards the left
Then State = "Beginning left lane change"

Meta-rule Beginning_left_lane_change
If  EV and TV on the same lane
Wheel are turning to the left during 0.2 s
EV behind TC
Movement towards the left
Then State = "Beginning left lane change"
Delete the rule " Overtaking intent"

This example shows that the use of metaknowledge can be useful and effective. On the other hand, it is necessary to verify that this advantage is not cancelled by a high cost of the management. In our case, this management is made with MIE to match metaconditions or to execute meta-actions. In the future, it will be better to implement this metaknowledge in a specific module independent of domain rules.

3 The meta inferences engine

3.1 The first structure

Two types of inferences engines are known: those based on filters algorithm and those using RETE network [4]. Various models were developed according to the principle of a RETE network. One of the differences between all these models comes from the memorization of facts in the nodes of the network. SNARK [7] and TREAT [14] make few memorization whereas OPS [1] and TANGO [3] memorise all facts and sets of facts (instances).
The use of a RETE network is much faster than the technique of filtering. On the other hand, it has a physical structure which is more complex: various types of nodes and joints. If this implementation had been chosen, every execution of a meta-action (for example to modify or to delete a rule) would have entailed, every time, an expensive reorganization of the RETE network. It is for that reason that the technique of filtering was chosen in the first time. Even if it is slower, it has the advantage of being more useful for the application of metaknowledge.

More concretely, the idea of the two first versions is to split the rules-based system BR1 of the domain under the shape of facts which are added to the base of facts BF1 to form BF2 (cf. figure 4).

Once the base of facts BF2 has been built, the meta inferences engine allows creation, modification, deleting of facts as a classic inferences engine would make (cf. figure 5). Difference comes from the fact that MIE is able to manipulate indirectly rules of the domain BR1 because they were split with facts: how an action of a BR1 rule can add, modify or kill BF2 facts, and that some facts represent a part (a condition or an action by example) of a rule, it allows the user to create meta-rules containing meta-actions or meta-conditions which will modify certain parts of rules during the execution.

But the counterpart of this capacity to conceive and to execute meta-rules is that the manipulation of this new structure requires a more complicated management of rules. Indeed, the meta inferences engine has to make not only all the tasks of a standard inferences engine (matchings, follow-up of instances, execution of actions, etc.) but it is also necessary that for each of these tasks, it "rebuids" the various elements of rules. For example, it is necessary that conditions (which were split in a set of facts in BF2) are reconstituted before trying to match them with BF1 facts.

### 3.2 The fragmentation of rules

This section presents the principle of the fragmentation of rules by an example: the `Beginning_left_lane_change_initial` rule is rewritten in several facts.

```
(Rule Beginning_left_lane_change_initial
  (?Data (= Y 0)) ; EV and TV same lane
  (?Data (< X 0)) ; EV behind TV
  (?Data (> Phi 0)) ; Wheel of EV to the left
  (?Data (= time ?t)) then (Assert (Step Name "Beginning left lane change"))
)
```

Every fact has an elementary syntax like `(Object Attribute Value)`. In the article, the term "object" will be used for to indicate a set of facts (a set of triplets) having the same name of object.

```
(Rule1 Class Rule)
(Rule1 Rule Beginning_left_lane_change_initial)
(Rule1 Nbr_conditions 4)
(Rule1 List_condition (Cond1 Cond2 Cond3 Cond4))
(Rule1 Nbr_actions 2)
(Rule1 List_actions (Action1 Action2))
```

At first, a global object presenting the rule is built for each rule of the domain. For example, the object Rule1, corresponding to the `Beginning_left_lane_change_initial` rule, is composed of six facts presented above.

Secondly, each condition of each rule is change into an object containing seven facts. The following example presents two objects (`Cond1` and `Cond2`) which gather all information concerning the first two conditions of the `Beginning_left_lane_change_initial` rule.

```
(Rule1 Class Rule)
(Rule1 Rule Beginning_left_lane_change_initial)
(Rule1 Nbr_conditions 4)
(Rule1 List_condition (Cond1 Cond2 Cond3 Cond4))
(Rule1 Nbr_actions 2)
(Rule1 List_actions (Action1 Action2))
```
Thirdly, each action of rules is also coded with an object composed with seven facts. Two versions of meta inferences engine using this principle of decomposition were developed (MIEv1 and MIEv2) and are presented in the sections below.

### 3.3 The meta inferences engine - version 1

This section describes the first version of the meta-inferences engine (called MIEv1). It consists in developing an inferences engine coded with a rules-based system BR2 which is executed with CLIPS (cf. figure 6). Then BR2 rules can easily manipulate the facts of domain BF1 as well as rules of domain BR1 because there are split with a set of facts.

![Figure 6: The first version of MIE.](image)

Four rules of BR2 base of rules symbolizing the various functions of MIEv1 are presented below: the matching of facts with conditions, the addition of a rule in the set of candidate rules, the choice of the rule to be activated, then its execution.

At first, a basic matching rule is presented above. The conditions zone uses three classes of objects: a fact (from BF1), a rule of the domain of application (from BR1) and a condition of this rule. The rule Matching1 is triggered only if the rule of BR1 has no created instance, and only if a fact of BF1 matches with one of the conditions. We will note that the variable ?num contains the fact number of the fact matched with the condition (?fact (= ?attribute ?value)).

#### Rule Matching1

```
(?fact (= Class Fact)) ; A fact of application domain (BF1)
?num <- (?fact (= ?attribute ?value)); with the number ?num
(?Condition (= Class Condition)) ; matches with a condition
(?Condition (= Operator =))
(?Condition (= Attribute ?attribute))
(?Condition (= Value ?value))
(?Condition (= Rule ?Rule))
(?Rule (= Class Rule)); belonging to a rule which
(?Rule (= Nbr_actions ?nbr_actions))
(?Rule (= List_actions ?l_actions))
(not (and (?Instance (= Class Instance)) ; has no instance
(?Instance (= Rule ?Rule)))

Then (Create_New_Instance ?Rule ?num ?Condition);
; Partial creation of instance for the rule ?Rule with the fact number ?num which matches with the condition ?Condition
```

The Candidate1 rule indicates that a rule of BR1 has all its conditions satisfied and so it is candidate to be triggered.

#### Rule Candidate1

```
(?Rule (= Class Rule))
(?Rule (= Nbr_conditions ?nbr_conditions))
(?Instance (= Class Instance))
(?Instance (= Rule ?Rule))
(?Instance (= Nbr_Facts ?nbr_conditions))
(?Instance (= State Building))

Then (modify ?instance State Candidate)
```

The rule Basic_strategy indicates that a rule of BR1 with the state Candidate is ready to be triggered. This stage is important because it allows the designer to replace the rule Basic_strategy by one set of strategic rules (a strategic module) which will select the rule to be triggered according to criteria to be defined.

#### Rule Basic_strategy

```
(?Instance (= Class Instance))
(?Instance (= State Candidate))

Then (modify ?instance State Ready_to_be_triggered)
```

The rule Execution1 allows to execute an action ?Action of a rule ready to be triggered: it removes one from the number of actions to execute, and removes ?Action of the list of actions to execute, then executes ?Action.

#### Rule Execution1

```
(?instance (= Class Instance))
(?instance (= State Ready_to_be_triggered))
(?instance (= Nbr_Actions_to_execute 0))
(?instance (= Nbr_Actions_to_execute ?nbr_actions))
(?instance (INCLUDE L_Actions_to_execute ?l_actions))
(?instance (= L_Actions_to_execute ?l_actions))
(\action (= Class Action))
(\action (= Operator \operator))
(\action (= Object \object))
(\action (= Attribute \attribute))
(\action (= Value \value))

Then
(modify ?instance Nbr_Actions_to_execute (- ?nbr_actions 1))
(modify ?instance L_Actions_to_execute (- ?l_actions ?action))
(eval \operator \object \attribute \value)
```

These rules simulating the functioning of an inferences engine (coded with rules) were
implemented and tested. The results are presented in the section Experiments.

### 3.4 The meta inferences engine - version 2

The second version of the meta inferences engine MIEv2 was conceived to attenuate the very slowness of the first version (cf sections experiments). It is always based on the principle of the fragmentation of rules BR1 with a set of facts. Nevertheless, the code is different: the meta inferences engine (cf figure 7) was directly coded in Pascal language. So, it directly administers rules of BR1 which were split.

![Figure 7: The second version of MIE.](image)

This technique allows gaining a lot of time by avoiding a double interpretation of the first version: CLIPS interprets BR2 rules which interpret BR1 rules with facts BF1. In this version, MIEv2 directly interprets BR1 rules (were split in a set of facts) with facts BF1.

### 3.5 The meta inferences engine - version 3

Although MIEv2 is more effective than MIEv1, it is not easily usable because its performance is still too slow. The problem mainly comes from the large quantity of facts to be administered because the rules of BR1 are transformed into several facts. The technique used with the first two versions being mainly filtering, the time of answer increases exponentially according to the number of facts and rules. MIEv3 tries to correct this problem by treating the rules of BR1 without splitting them up in order for not to have a too voluminous base of facts.

The solution consists in building a tree where various information describing rules are coded (see figure 8). So, the rules are not split; the base of facts is smaller and the time of execution is better.

![Figure 8: The structure used by MIEv3.](image)

### 4 Experiments

MIE was tested with a 166 MHz Pentium by executing the first IDRES's level. At first, no metaknowledge has been implemented. This first experiment consisted in recreating the rule-based IDRES (see the section The fragmentation of rules) and the base of facts corresponding to the lines of data (see table 1) with a frame useful by MIE.

<table>
<thead>
<tr>
<th>Number of data lines</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of initial facts</td>
<td>872</td>
<td>883</td>
<td>894</td>
<td>905</td>
<td>916</td>
</tr>
</tbody>
</table>

So, the 10 rules (102 conditions and 10 actions) of the IDRES's first level were transformed into facts to test MIEv1 and MIEv2. There are 17 facts which are parameters setting. Every data line creates 11 facts (Time, X, Y, V, Teta, Acc, Phi, Rg, Rd, Number of line, class of fact). Finally, MIE will have to administer \((n_{rules} * 6) + (n_{conditions} * 7) + (n_{actions} * 7) + 17 + (n_{data_lines} * 11)\) initial facts (cf. table 3) and some facts corresponding to the management of instances. Experiment was made from 1 to 4 data lines for MIEv1 and from 1 to 5 data lines for MIEv2.
Figure 9: Performances of MIEv1 and MIEv2.

The figure 9 shows that the times of execution were effectively considerably improved with MIEv2. So, the search for the states of the vehicle maneuver with 4 data lines only costs 4.45 seconds for MIEv2 against 266 seconds for MIEv1.

Figure 10: Performance of MIEv2 and MIEv3.

The lastest version of the meta inferences engine is quicker: 4 data lines are executed in 0.6 seconds. We will also notice that the curve is less exponential than the two other versions.

Another experiment consists in testing the execution of metaknowledge by MIEv3 and to verify that its use allows improving the performances of a rule-based system. The section The IDRES system proposed meta-actions that allow deleting rules which became useless. This idea was put into practice by adding to all rules of the IDRES's first level a meta-action able to kill useless rules. The figure 11 compares times of execution of MIEv3 without meta-actions and MIEv3 with meta-actions.

Figure 11: Use of meta-actions.

The figure shows that the use of meta-actions is effective in the context of our experiment. Furthermore, the gain of time will also increase with the number of data lines. Indeed, there are more chances that rules will be deleted by meta-actions. It avoids useless tests of matching.

5 Synthesis

The structure used to develop and to use MIE has the advantage of being relatively simple and fast to be developed. It allows the conception of meta-rules, meta-conditions or meta-actions as easily as one rule. It also authorizes the creation of reflexive meta-rules (which apply to themselves) without having to duplicate knowledge [6]. Another advantage is that the meta inferences engine is able to execute these meta-rules during the execution. It authorizes the development of learning techniques in real time or in very dynamic domains environment (like assistance in driving cars).

The three versions are operational and able to administer metaknowledge. MIEv1 and MIEv2 are both based on the use of a base of facts which contains, besides the facts of the domain BF1, facts resulting from the splitting of rules BR1. First experiments of MIEv1 showed a very important slowness because of the management of a number of facts too big because of a double interpretation of the rules. Indeed, CLIPS executes a BR2 rule which makes the matching between a fact of the domain and the facts describing one condition of a rule BR1.

This handicap accelerated the revision of the initial approach by developing a meta inferences engine MIEv2 able to making directly this double interpretation. MIEv2 does not contain a rule-based system (MIEV1 used BR2 which simulates the functioning of an inferences engine). This solution corrects a part of the first system disadvantages. Then the gain of time is
very considerable. On the other hand, the structure of MIEv1 gives the possibility of learning from the functioning of BR2 and also on the functioning of the meta inferences engine itself, which is not possible or very difficult with MIEv2 because it is compiled. Furthermore, even MIEv2 is 60 times faster than MIEv1, its performances are still not sufficient.

In order to try to improve the times of execution, the last version MIEv3 does not split up any more the rules of the domain with facts. It uses a tree structure describing the base of rules. This option allowed the meta inferences engine to be 7.5 times faster than MIEv2. Thanks to this performance, meta-actions were tested successfully in IDRES's context. Results show that the use of metarules is advantageous in some cases.

6 Conclusion

The meta inferences engine was implemented and is operational. It is useful for the development of metaknowledge. Even if CLIPS is quicker, MIE proposes easinesses to develop IA capacities. In the futur, it will be interesting to compare it with metaknowledge created with PROLOG.

Several prototypes of MIE have ended in a system enough fast to be able to test metaknowledge with a rule-based system. Nevertheless, the performances should again be improved for a use in the education context or with indepent Artificial Intelligence modules (strategy, learning, explanation, ...).

Acknowledgements

Research supported in part by Champagne-Ardenne Regional Council (district grant) and the European Social Fund.

References


