Rule-based System for Emotional Decision-Making Agents

Valentin LUNGU

***Abstract*—This paper proposes a rule-based system that uses a forward chaining and a backward chaining inference engine, a truth maintenance system and emotion simulation to achieve reasoning, fast decision-making intelligent agents.**

An agent needs to be able to accomplish its goals. Hierarchical goal decomposition is a powerful tool, allowing the agent to represent and solve complex problems. A backward chaining inference engine is best at breaking down goals into sub goals.

Agents in a dynamic environment where multiple aspects of the world are currently changing must be able to infer new knowledge about the world. Also, said agents should also be able to act in uncertain conditions (conditions of uncertain knowledge). A forward chaining inference engine is used to infer knowledge about the world that is not strictly goal-related, and a truth maintenance system is used to handle conflicting knowledge and maintain a consistent set of beliefs about the world.

Emotion integration is necessary in generating complex believable behavior, making the agent decision-making process less predictable and more realistic as well as generating actions in time comparable to human reaction time.

***Index Terms*—multi-agent system, rule-based system, inference engine, forward chaining, backward chaining, truth maintenance system**

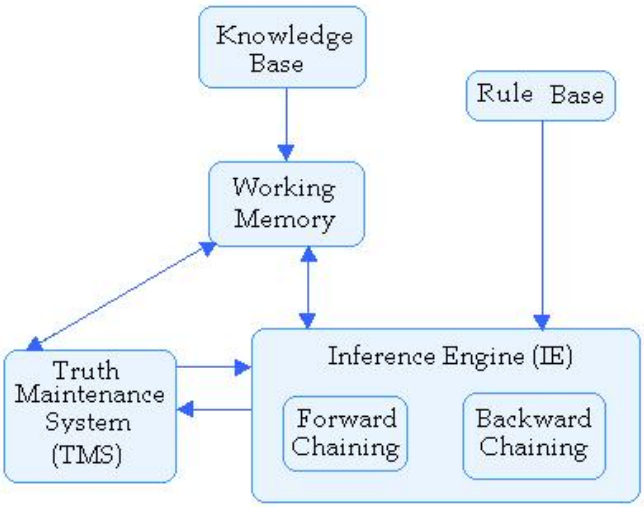
I. Introduction

AUTONOMOUS artificial intelligent entities are required to be able to exist in a complex environment and respond to relevant changes in the environment, deliberate about the selection and application of operators and be able to pursue and accomplish goals. This paper proposes a rule-based system that involves both forward and backward chaining and a rule maintenance system in order to provide an architecture for the development of fast decision-making agents. In the following sections, the features of this framework, the reasons for which they are needed and the way they interface with each other will be elaborated. Last, but not least, we discuss the value of integrating emotions with our agents, and a possible way to do so.

Manuscript received December 10, 2009.

Valentin LUNGU is a Ph. D student at the Faculty of Computer Science and Engineering and Automatics, Politehnica University, Bucharest. (phone:0723159898; e-mail: valentine.lungu1403@cti.pub.ro).

II. Inference Engine



A. Backward Chaining

Backward chaining is a lazy inference method. It only does as much work as it has to. Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent that matches a desired goal. If the antecedent of that rule is not known to be true, then it is added to the list of goals (in order for one's goal to be confirmed one must also provide data that confirms this new rule), making it ideal at building an at least partially (we may have to accomplish disjoint goals, as well as having to accomplish several goals concurrently with no specified order in order to confirm a rule) ordered plan to follow in order to achieve the agent's goals, dynamically decomposing goals into subgoals recursively until the agent selects among primitive operators that perform actions in the world. The backward chaining part of our inference engine accomplishes our agent's goal-oriented planning, resulting in the composition of a plan, a sequence of partially-ordered goals to pursue in the future.

B. Forward Chaining

Forward chaining is a data-driven inference method. The engine hungrily awaits new knowledge in order to apply rules that match existing data.

Forward chaining starts with the available data and uses inference rules to extract more data (from an end user for example) until a goal is reached. An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (**If** clause) is known to be true. When found it can conclude, or infer, the consequent, resulting in the addition of new information to its data.

Forward chaining inference engines will iterate through this process until a goal is reached. This seems as an ineffective way to go about solving a problem, especially when we have a large number of rules and data and only a few paths that lead to goal states. Backward chaining seems to be the better problem solver, and indeed it is; however, backward chaining does have a weakness, it never infers data that is not explicitly goal-related, even if somewhere down the inference path it may lead to goal-related data, or worse, conflicting data. In essence, we are using the forward chaining part of our inference engine to gather knowledge about the complex, dynamic environment, where multiple aspects of the world may be changing at any given moment. When run, a forward-chaining algorithm presents a conflict set of rules to apply (all rules that match current knowledge). Ways of deciding on application order will be discussed in a further section. At times, our forward-chaining inference engine may infer contradictory data; we need a way of detecting and dealing with such situations. The best approach is a truth maintenance system. Also, given that rule-based systems may have a large number of rules working with a large knowledge base, we need a fast pattern-matching algorithm (pattern-matching may be as much as 80% to 90% of the effort of a forward chaining rule based-system). A good approach to this is using the RETE pattern matching algorithm, as it interfaces well with a truth maintenance system, as well as with our backward-chaining generated plan. The forward chaining inference engine's purpose is to reason about the world, gather new information and make necessary assumptions in order to keep belief (knowledge) base consistency; rules that contain primitive operators that take actions in the world should not be here, although we will keep the architecture flexible and allow for them; a separation mechanism will be required between the two. The end-product of the forward chaining inference cycle will result in the addition and retraction of rule instances to and from an agenda of rules to be applied.

C. *Pattern Matching (RETE)*

The RETE algorithm is a fast pattern-matching algorithm that compiles the left hand side of the production rules into a discrimination network (in the form of an augmented dataflow network). Changes to working memory are input to and propagated down the network and the network reports changes to the conflict set of rules to be executed (adding new rule instances that have activated and retracting old ones that are no longer valid from the program's agenda). Our backward-chaining generated plan may benefit from this compilation of the network as well, since we may identify goals in the plan that may have been instanced in the agenda.

Our next problem is conflict resolution. In which order should the instanced rules in the conflict set be scheduled in the agenda, in order to be applied? The following approach is proposed: rules are ordered by user-specified priority (as *salience* in CLIPS, or PR in GPSS/H), then, these subgroups are sorted by specificity (from most specific to least specific), and these subgroups are further sorted by how recently the facts used to instantiate the rule were added to the knowledge base. A metarule approach is also possible.

D. *Truth Maintenance System*

A truth maintenance system is used to maintain a consistent set of beliefs about the world. It is based on the following principles:

- each action in the problem-solving process has an associated justification
- when a contradiction is obtained, find the minimal set of assumptions that generated the contradiction. Select an element from this set and defeat it. The justification for the contradiction is no longer valid and the contradiction is removed.
- propagate the effects of adding a justification and of eliminating a belief (keep consistency)

A truth maintenance system keeps beliefs as a network of two types of nodes: beliefs and justifications. A belief is an expression which can be true or false. A belief node contains a label (IN if the belief is considered true, OUT otherwise), a list of justifications for the node, and a list of the justifications of which the node is part of (and the label it must have in order for that justification to be valid). Justification nodes contain the following information: the inference type of a justification (in our case premise (always IN) or modus ponens (inferred)), a list of nodes the justification justifies and a list of nodes (and necessary associated values) that participated in the inference. This representation allows the propagation of consequences through the belief network of an agent. A truth maintenance system is easily integrated with the RETE pattern-matching algorithm (nodes that are IN will be considered true, and present in the RETE network, and nodes that are OUT will be considered false and will not be present in the RETE network; nodes that change state from OUT to IN will be added and nodes that change state from IN to OUT will be retracted from the RETE network).

As can be seen, the RETE network and a TMS are easy to interface with each other, since they operate on different areas of a rule system (working memory vs. rule base).

Truth maintenance systems are not usually used alongside backward chaining inference engines because the inference engine rarely changes a node's state; however, should this occur, the change is fed into the TMS by the inference engine and the changes propagated through the network, just as in the forward chaining case.

E. *Operating Principles*

In this section we will present the operating principles of the proposed architecture. The architecture is meant to

facilitate the design of fast reasoning decision-making agents. We have discussed the mechanisms that will allow an agent to decide which rules to execute in order to achieve its goals in previous sections. We will present the way in which the rule based system's components are meant to operate.

We have affirmed that a forward chaining inference engine is a slow and ineffective way of handling a given problem, however, it is needed in order to gather knowledge about the dynamic environment in which the agent exists. This means that we do not need to run the forward chaining engine every cycle. Our agent can keep working towards its goals, and only gather extra information when it is deemed necessary, therefore making decisions a lot faster, and more goal oriented, while still reasoning about the environment.

```
    // basic operating algorithm
  algorithm (KB, RB, goals) {   // knowledge
base,
                                // rule base,
                                goals
    beliefs = TMS (KB)   // assert all
knowledge as
                         // premises in belief
base
    rete_net = RETE (RB) // initialize RETE
network
    plan = {goals}
    while (not (null plan)) {
      update (plan)
      action = extract (plan) // choose
possible action
                           // from plan
      execute (action)
      update (beliefs)
      update (rete_net)
      if (should infer new knowledge){
        while (not (null agenda)){
          rule = choose (agenda)
          apply (rule)
          update (beliefs)
          update (rete_net)
          - remove rules which are no
            longer valid from agenda
          + add new valid rules to agenda_tmp
        }
        agenda = merge (agenda, agenda_tmp)
      }
    }
  }
```

III. Future Work

A. Satisficing

In some situations, an agent may not have time to consider all options and possibilities. An agent should be able to take reasonable action with the information at hand, even if incomplete.

B. Reasoning Under Uncertainty

When presented with a conflict in the truth maintenance system, an agent needs to make a decision which assumption to defeat from the minimal set of assumptions justifying the contradiction. Also, when presented with an impasse, the agent needs to determine the best assumption to make. The assumption that brings it closest to its goal state is not necessarily the best.

C. Emotions and Decision-Making

Although it is clear that emotions sometimes impede deliberative decision-making, one school of thought affirms that emotions provide a way of coding and compacting experience to enhance fast response selection. In evolutionary terms, it is better to respond immediately to a threat than take the time to rationally consider the best course of action.

Another use for emotion simulation in multi-agent systems is generating complex believable behavior, important in simulation environments and human – computer interaction.

Future work involves exploring and integrating both schools of thought into the architecture.

D. Learning

An agent needs to be able to analyze past actions and determine the cause of its successes and / or failures, as well as be able to shortcircuit rule application sequences. Future research will attempt the integration of a history function and tools that will allow the agent to analyze and synthesize its history, possibly adding new rules to its long-term memory.

References

[1] Langley, P. et al., Cognitive architectures: Research issues and challenges, Cognitive Systems Research (2008), doi:10.1016/j.cogsys.2006.07.004.

[2] E. Chown, R. M. Jones, A.E. Henniger, An Architecture for Emotional Decision-Making Agents, Interservice/Industry Training Systems

[3] and Education Conference(I/ITSEC) 2002

[4] R. M. Jones, An Introduction to Cognitive Architectures for Modeling and Simulation, Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2004

[5] Soar: Along the Frontiers, Soar technology 2002, www.soartech.com

[6] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. Artificial Intelligence, 33(1): 1-64.