Bounded Metacognition

Darsana Josyula and Kenneth M'Balé
Department of Computer Science
Bowie State University
Bowie, MD USA
E-mail: darsana@cs.umd.edu and kmbale@cs.umd.edu

***Abstract*—Agents situated in dynamic environments have limited time to deliberate before performing their actions. Cautious agents that deliberate for too long may miss deadlines to accomplish tasks whereas bold agents that deliberate for too little time may behave rashly or miss opportunities. There are several approaches discussed in the literature that rely on meta-level mechanisms to monitor and control the deliberation time. These approaches seem to follow the view that the meta-level mechanism is an external component not constrained by the same resource limitations as the underlying agent's deliberation mechanism. In this paper, we present an approach to resource bounded metacognition wherein an agent monitors and controls its deliberation and metacognition within the uniform framework of Active Logic.**

Keywords-metacognition; deliberation; reasoning; active logic

I. INTRODUCTION

Metacognition [1, 2] is the ability to monitor and control one's own thinking. This ability is important to control the time agents spend deliberating to choose their actions. For, in the process of achieving goals in a dynamic environment, an agent has a finite amount of time to make a decision (deliberate) and take an action (act). The circumstances that form the starting point of the deliberative process change at various rates. The best choice given one set of circumstances is not necessarily the correct choice in another set of circumstances, even when the differences are small. The quality of a decision correlates to the amount of time an agent deliberates. In this context, quality is the degree to which the decision fits the circumstances in which the action is taken.

Agents may choose to short-circuit the deliberation and perform a default action when there is not enough time to deliberate. On the other hand, when there is more time to deliberate, agents may choose to deliberate over its current knowledge to choose the best course of action. Cautious agents that deliberate for too long may miss deadlines to accomplish tasks, whereas bold agents that deliberate for too little time may behave rashly or miss opportunities [3]. Monitoring and controlling the deliberation process can help agents behave cautiously or boldly at different times depending on what is best suited for the situation.

Different approaches to monitoring and controlling deliberation, aim at explicitly setting time limits to deliberation [4] or limiting the size of current knowledge used for deliberation [5], or both [6, 7, 8]. Setting predefined limits for deliberation time is a simple strategy; however it constrains the adaptability of the agents — e.g., from being bold to cautious, as the environment changes. If the allowable deliberation time is not preset, but updated dynamically by a metacognitive process, then the resulting agents can adapt from being bold to cautious or vice-versa. However, this requires that the metacognitive process computes and updates the allowed deliberation time dynamically. Similarly, the current knowledge used for deliberation can be constrained by simply presetting a limit to the size of the knowledge base (KB) or, by computing and updating the size dynamically by a metacognitive process. Therefore, when adaptability is important, the metacognitive process has to perform non-trivial computations in a reasonable amount of time for such computations to remain effective.

If the metacognitive process shares the same computing resources as the underlying cognitive system, then decisions need to be made on when and how much of the computing resources are allocated for metacognitive processes versus cognitive activities. If too little resources are allocated for metacognition, the resulting agent may not have the ability to self-reflect and discover new methods and solutions. On the other hand, if the agent's resources are used mainly for metacognition, the resulting agent would be reflective, but may not have enough resources to deliberate and choose the best known course of action.

When the metacognitive process runs external to the cognitive sphere of the agent, and thus not sharing the same computing resources, synchronizing the communication between the two processes becomes non-trivial. The part of the underlying cognitive system that is accessible by the metacognitive process determines how much the resulting agent can affect its own thinking. Therefore, decisions need to be made on which parts of the cognitive system is accessible by the metacognitive process and when.

In this paper, we describe how we have addressed these issues in an agent based on Active Logic. We describe how the metacognitive process of the agent shares resources with the underlying cognitive process within a uniform framework based on Active Logic. We discuss how the two processes run in an interleaved manner (similar to the fully-interleaved deliberation strategy discussed in [9]) in order to manage the sharing of resources such that the resulting agent is self-reflective as well as deliberative.

The rest of the sections are organized as follows. Section II details features of Active Logic that facilitates the fully interleaved deliberation strategy. In Section III, we give a brief overview of Alma/Carne agent based on Active Logic. In Section IV, we discuss how automatic resource sharing occurs between the metacognitive and cognitive deliberation

 ISBN: 978-1-61208-273-8

processes within the Alma/Carne agent. Section V discusses related work and Section VI gives the conclusion.

II. ACTIVE LOGIC

Active Logics [10, 11] are a family of formalisms that use time sensitive inference rules to have their KB evolve with the passage of time. Technically, an Active Logic consists of a first-order language, a set of time-sensitive inference rules and an observation function that specifies aspects of the environment as first order formulae. Therefore, Active Logic can be seen either as formalism per se, or as an inference engine that implements formalism.

In Active Logic, the basic unit of time is a *step* and the passage of time is represented by a predicate *Now* that is true only for the current step in the reasoning process. The formulae at each step include those formulae that are (i) inherited from the previous step, (ii) obtained by applying the rules of inference to the formulae in the previous step and (iii) added as observations at that step. Direct contradictions at one step are not inherited to the next step; hence, they do not derive new formulae thus avoiding the traditional issue with first order logics wherein all well formed formulae are concluded from a contradiction.

In the sub-sections that follow, some of the useful general features of Active Logics are discussed.

A. Inheritance of Formulae

By default, all formulae in a step that are not directly contradicting are inherited to the next step. However, some formulae like the ones related to the current time are not inherited to the next step. The inheritance of formulae from one step to the next is controlled by inheritance rules. One simple version of such an "inheritance rule", which also illustrates the use of firing conditions, is shown in (1):

$$\begin{aligned} i : &\quad A\ [\text{condition} : \neg A \notin KB,\ A \neq \text{Now}(i)] \\ i+1 : &\quad A \end{aligned} \tag{1}$$

B. Step-wise Reasoning

In Active Logic, the formulae at step i+1 are obtained by applying the rules of inference in the logic to the formulae in step i as illustrated in (2).

$$\begin{aligned} i : &\quad A,\ A \rightarrow B,\ B \rightarrow X \\ i+1 : &\quad A,\ B,\ A \rightarrow B,\ B \rightarrow X \\ i+2 : &\quad A,\ B,\ X,\ A \rightarrow B,\ B \rightarrow X \end{aligned} \tag{2}$$

Here, at step i+1, B is derived using the formulae A and A → B, and at step i+2, C is obtained from the formulae, B and B → C. Each "step" in an Active Logic proof takes one Active Logic time-step; thus inference always moves into the future at least one step. Since the finitely-many inference rules when applied to the finite set of formulae in a step can produce only finitely-many conclusions for the next step, an Active Logic KB will have only a finite set of formulae at each time step,

C. Addition of New Formulae

The observation function can add new formulae into the logic at any step. The formulae that are added in a step are incorporated into the ongoing reasoning to derive the formulae for the next step. Step-wise reasoning coupled with this ability to add new formulae at any step, ensure that the logic will not get stuck in a lengthy proof, oblivious of the external changes that occur during the reasoning. That is; the external changes that occur while the logic is performing the lengthy proof can be added as new formulae at any step during the proof. In fact, these added formulae could change the course of reasoning, since they get included in the ongoing reasoning as soon as they are added into the knowledge base.

D. Time Sensitivity

To represent and reason about the passage of time, Active Logic employs a notion of "now" that is constantly updated by the "clock rule" shown in (3). The clock rule states that from the fact that it is step i at the current step, the step number of the next step is i + 1.

$$\begin{aligned} i : &\quad \text{Now}(i) \\ i+1 : &\quad \text{Now}(i+1) \end{aligned} \tag{3}$$

With the help of the clock rule, Active Logic keeps track of the evolving time as the reasoning progresses from one step to the next. This evolving-during-inference model of time sharply contrasts with the frozen-during-inference characterization of time that temporal logics [12, 13] have. In temporal logics, the past, present and future do not change while theorems are being derived. This time-tracking property of Active Logic is especially useful when an agent's reasoning is aimed towards meeting a deadline; see [14] for details.

E. Contradiction Tolerance

The ability of Active Logic to explicitly track the individual steps of a deduction makes it a natural mechanism for reasoning about contradictions and their causes. If directly contradictory wffs, P and ¬P, occur in the KB at time i, Active Logic notes the contradiction at i+1 using a 'conflict-recognition" inference rule like (4), so that further reasoning can be initiated to repair the contradiction, or at least to adopt a strategy to deal with it, such as preventing the contradictions from deriving any new formulae in the later steps.

$$\begin{aligned} i : &\quad P, \neg P \\ i+1 : &\quad \text{Contra}(i, P, \neg P) \end{aligned} \tag{4}$$

 ISBN: 978-1-61208-273-8

Disinheriting contradicting predicates is a reasonable immediate response to deal with a contradiction; however, it is not enough to "defuse" the contradiction for long. The formulae that derived P and ¬P may re-derive the contradicting predicates, or other conflicts may occur. Thus, [15, 16, 17] investigate ways to allow an Active Logic-based reasoner to retrace its history of inferences, examine what led to the contradiction, and perform meta-reasoning concerning which of these warrants continued belief.

The *Contra* predicate in (4) is a meta-predicate: it is about the course of reasoning itself (and yet is also part of that same evolving history). Thus, unlike in truth maintenance systems [18] where a separate process resolves contradictions using justification information, in Active Logic the contradiction detection and handling occur in the same reasoning process.

F. Representation of Defaults

If no evidence is already known that would prevent a default conclusion, then Active Logic derives that default conclusion. In Active Logic, defaults can be represented using default rules like (5), which states that if ¬P is not known at the current time, and if Q is known, then P is inferred by default at the next time step.

$$\begin{aligned} i: &\quad Q, \neg Know(\neg P, i), Now(i) \\ i+1: &\quad P \end{aligned} \tag{5}$$

Since, only a linear lookup in the belief set for time i is needed to tell that ¬P is not there (and that Q is there), the decidability issues of traditional default mechanisms do not arise in Active Logic. The default rule in itself does not deal with problems arising from interacting defaults. However, such cases tend to involve contradictory conclusions, as when, evidence for ¬P becomes known. Therefore, they can be treated as any other contradictions. One simple expedient in such cases is to disinherit the default conclusion and accept the non-default evidence.

G. Introspection

In Active Logic, negative introspection—the ability to determine that one does not know something—is often encoded as the following inference rule (where the notation [B] means that B is not present):

$$\begin{aligned} i: &\quad ...[B] \\ i+1: &\quad \neg Know(i,B) \end{aligned} \tag{6}$$

This mandates the conclusion at time i + 1 that statement B was not known to the logic at time i (that is, B does not appear among the beliefs at time i).

H. History Tracking

Active Logic maintains a temporal history of its reasoning process that can be used by the logic for further reasoning. The history enables the logic to determine when each formula was added or deleted in its past and thus provides a mechanism to reason about the past reasoning.

I. Quotation

Quotation mechanism names the different formulae in Active Logic. This allows an individual formula to be referenced using its name. The quotation and the history mechanism together provide a mechanism for meta-reasoning within the reasoning process itself.

J. Integration with non-logical processing

Finally, Active Logic can initiate, observe and respond to external events and non-logical processes by proving specialized predicates. For example, the proposition call initiates an external action.

III. ALMA/CARNE AGENT

Alma/Carne [19] is a general purpose implementation of Active Logic. It has a dual role: (i) as the language to specify Active Logic based applications and (ii) as the core reasoning engine for these applications.

In its role as a language, Alma/Carne allows applications to be specified as a set of logical sentences and procedures. When the sentences are loaded into Alma and the procedures into Carne, Alma/Carne takes the role of a reasoning engine. In this role, Alma generates Active Logic inferences, some of which trigger procedures in Carne. These procedures can perform computations or cause effects in the world, and can include non-logical reasoning procedures like probabilistic reasoners and parsers (thus, allowing close interaction between different kinds of reasoning). Alma's KB is updated with the status of the procedures (e.g., done, doing) which enables reasoning about the processes Alma triggered. Failure of a procedure, for instance, can lead to reasoning that causes retraction of earlier assumptions.

Carne can also monitor the world and assert formulae about the state of the world into Alma, implementing the observation functionality of Active Logic. This enables Alma to react to changes in the world. Thus Alma/Carne can initiate, observe and respond to external events and non-logical processes. Each step in Alma/Carne is bounded by a maximum time. Therefore, Alma/Carne remains responsive to incoming observations even when the size of the knowledge base increases.

Alma/Carne allows priorities to be set to formulae in its KB. Formulae are examined in the order of their priorities while deriving formulae for the next step. This provides a mechanism for limiting the formulae that are examined to derive the formulae at the next step.

Formulae in Alma/Carne have an associated name. This characteristic allows easy reference to a formula for deleting it from the KB or distrusting it at any step. Formula names also help to locate an existing formula in order to change its priority.

Alma/Carne agent is implemented by loading into the Alma KB a set of formulae that correspond to the agent's world knowledge, goals, plans, expectations, action rules, contradiction handlers, expectation handlers and expectation violations. Formulae corresponding to the agent's world

 ISBN: 978-1-61208-273-8

knowledge include predicates that specify the objects in its domain, their properties and relationships. Plans are implication formulae with goals (or sub-goals) as consequents and preconditions as antecedents. Preconditions can be any predicates including goals or sub-goals. Expectation formulae are predicates or implications that specify what the agent expects or what the agent can expect when certain conditions are true. Action rules are implication formulae that specify the conditions under which each action can be performed. Goals appear as antecedents and actions (*call* predicate) appear as consequents in action rules. Contradiction handlers are implication formulae that specify how to deal with direct contradictions (*contra* predicate) in the Alma KB. Expectation handlers are implication formulae that detect and note expectation violations.

As step-wise reasoning proceeds with the initial KB, new formulae get derived. When a *call* predicate gets derived in the course of reasoning, Alma requests Carne to perform the corresponding action at the next time step. At the same time, the expectation formula associated with the action generates an expectation in the Alma KB. If the expectation is violated, then the agent notes the violation and reasons about it. Expectation violations can appear as preconditions in plans; therefore, the agent can adopt goals to deal with expectation violations. These goals can cause Alma to derive other actions based on the current set of action rules in the KB.

IV. COGNITION AND METACOGNITION IN ALMA/CARNE AGENT

A. Cognitive Deliberation

The cognitive deliberations of the agent are achieved by the set of formulae that specify the agent's world knowledge, goals, plans and action rules. The initial belief set of the agent includes its initial model of the world, initial set of goals, plans to achieve various goals, and action rules that specify when to perform different actions. As reasoning proceeds in Alma/Carne, formulae get added into the KB or retracted from the KB at each step, based on the formulae in the previous step. Observations of the agent also get asserted in the KB as beliefs of the agent in the next time step.

When a goal is derived in the agent's KB, the plans with the goal as one of the antecedents will cause new formulae (consequents) to get asserted in the Alma KB. These formulae may correspond to new goals (or sub-goals); in which case the new (sub-) goals get derived at the next time step. Some goals appear as preconditions in action rules. When all the preconditions for executing an action is met at a time step, then the *call* predicate for that action gets derived in the next time step. Thus, the agent's current set of beliefs about the world, goals, plans and action rules determine which action the agent chooses to execute.

When a *call(α)* predicate gets derived in a time step, Alma sends the action predicate α to Carne for execution. As Carne begins executing the action α, it asserts a *doing(α)* predicate in Alma. When the action is done, Carne asserts *done(α)* in the Alma KB.

B. Metacognition

Metacognition in Alma/Carne agent is achieved by the set of formulae that correspond to expectations, contradiction handlers and expectation handlers. The agent maintains expectations about how its KB evolves in order to perform metacognitive alterations to its deliberative process. The types of expectations include ones related to time, content and feedback. *Time related expectations* specify when a certain predicate is expected. *Content related expectations* specify the expected values of parameters in a predicate. *Feedback related expectations* specify the predicates that are expected.

Expectations are represented as *expectation(ρ, t_1, t_2)*, where t_1 and t_2 are values for time steps, and ρ is a predicate. That is, ρ is expected between time steps t_1 and t_2, or if t_2 is zero, sometime after t_1. If t_1 is zero, ρ is expected to be true from the current time step, Now, until t_2. If both t_1 and t_2 are zero, ρ is expected to always be true. ρ is of the form *predicate(paramlist)* where *paramlist* can contain values, variables and predicates. Any parameter in the *paramlist* can be set to "_" to indicate that the specific parameter can take any value, that is, the value does not matter as long as the parameter has some value (existential quantification). If a parameter is set to a specific value, then that value is expected for that parameter. On the other hand, if the parameter is set to a variable, then that variable is assumed to be universally quantified.

When a formula, *expectation(ρ, t_1, t_2)*, gets derived at a time step, the expectation handler rules in the KB monitors the KB for the presence of ρ during the time period specified by t_1, and t_2. If a violation occurs, an expectation handler notes the violation. Example of an expectation handler that notes a violation for expectations with values of t_1 and t_2 as 0, is given in (7).

$$\text{expectation}(\rho, 0, 0) \wedge \rho \rightarrow \text{violated}(\rho) \quad (7)$$

When violated(ρ) appears in the KB, plans associated with the violation will create goals to deal with the specific violation in the next step. These goals in turn may cause new (sub-) goals to be derived; eventually, all the preconditions of some action rules become true and as a result the corresponding action gets executed.

The other type of metacognitive monitoring and control in Alma/Carne agent is achieved by the contradiction noting and handling mechanism. If *not*(ρ) gets proven in the KB during any time step when ρ is in the KB, a contradiction is asserted. This causes the contradiction handling rules to fire at the next time step. The contradiction handling rules evaluate the situation and assert new formulae or retract existing formulae from the KB. This would cause changes in the agent's normal deliberative process since the agent's reasoning at any step is solely dependent on the formulae at the previous step.

C. Discussion

Both the metacognitive process and the cognitive process have access to the formulae in each step; therefore,

 ISBN: 978-1-61208-273-8

information sharing between the two processes is trivial. As a consequence, the metacognitive process can alter any aspect of the cognitive processing of the agent.

Since no distinction is made between the processing of formulae that contribute to metacognitive deliberations and those that contribute to cognitive deliberations, both processes happen simultaneously intertwined with each other in a step-by-step manner. Since each step is limited in time, neither the metacognitive process nor the cognitive process can monopolize the agent. Both processes are constrained by the same time and resource limitations.

The actions that the agent executes at any step depend on the facts that exist in its KB at the previous step; and not on what is derivable from the facts in the KB at a later step. As the agent deliberates using step-wise reasoning from its goals to actions, metacognitive suggestions and new observations are incorporated into the reasoning. Therefore, the agent may appear cautious when observations cause new formulae to be derived or deleted from the KB, and bold when new observations and suggestions are not added to the KB.

The likelihood of a new formula getting derived depends on the priority of the rules that can produce it; the higher the priority, the more the likelihood. This means that the formulae (low priority ones) that contribute to the creativity of the agent may not get its chance to fire during the time steps when the agent is busy with its high priority deliberations. During those time steps when fewer rules are fired by the deliberative process, the lower priority rules have the opportunity to manifest itself in the stepwise reasoning of the agent and alter the KB.

V. RELATED WORK

Research on metacognitive monitoring and controlling of deliberation can be broken into four broad classes. Each class — Heuristics, Focusing, Intermodal, and Short-Circuit, is described below using a representative paper.

The Heuristics class of approaches is represented by [6]. The paper proposes an approach to focusing deliberation by using heuristics to inform the unrolling of an agent's Markov decision process. A Markov decision process is a probabilistic model that, departing from a well-known current state, can predict the probability of future states based on a set of actions. A rule for choosing an action is called a policy. The process of projecting future states is called unrolling. New states are placed on an open list that is sorted based on relevance to the current context and environment of the agent. The agent balances unrolling with selecting the highest priority state on the open list to begin deriving new policies. The heuristics inform the agent in its decision to unroll or derive.

The Focusing class of approaches is represented by Fox and Leake [20]. The paper proposes to focus the deliberation process by narrowing the KB the agent uses to the subset relevant to the circumstances. The reduced focus set enables the agent to spend the available deliberation time to produce the optimal action by adapting to the current circumstances.

The Intermodal class of approaches is represented by Dylla et al. [21]. The paper proposes a hybrid approach where the deliberative process simultaneously produces an immediate action and a strategy. Since the action is immediately available, the component of the agent responsible for taking action does not need to wait for the strategy in circumstances that do not permit doing so. This approach implies that the action component is intelligent enough to determine when to wait for a strategy.

The Heuristics and Focusing classes above are representative of the current body of work where the function of the deliberative process is not affected by the deadline. Instead, the deadline affects the scope of the input into the process. These approaches suffer from an inability to adapt to circumstances that require an action sooner than the minimum deliberation time. The Intermodal class addresses the weakness by providing an immediate action while deliberating. This approach subordinates deliberation to action. The deliberation process is no longer responsible to action. Instead, an action process is able to ignore the results of deliberation. This approach does not actually solve the problem because the action process has to deliberate about when to act and when to wait for the strategy. This deliberation itself has a deadline.

The Short-Circuit class of approaches is represented by Josyula et al. [22]. The paper proposes to inform the deliberation process with the emotional state of the agent. The emotional state is defined over continuums of stress (y-axis) and pleasure (x-axis). Pleasure represents the level of discordance between expectations and observations, both in terms of the number of expectation failures and the magnitude of discrete failures. Stress represents the number of perturbations or observable environmental changes the agent has to deal with. The level of stress modulates deliberation by affecting the probability of the agent developing new strategies in response to environmental stimuli. This Short-Circuit class is representative of biologically inspired approaches. An emotional component informs the metacognitive process, based on the magnitude of expectation failures. Emotion can then affect how the agent uses its failure and response ontologies to decide on an action. The imminence of a deadline can directly affect the emotional state of the agent and short-circuit the deliberation similar to a biological emotional reaction.

VI. CONCLUSION AND FUTURE WORK

This paper described the implementation of a metacognitive process that shares resources with the underlying cognitive process, in a manner that allows the resulting agent to be both self-reflective and deliberative. In our implementation of the Alma/Carne agent, the cognitive component adds or prunes formulae from the agent's KB based on the agent's current set of goals and beliefs. At the same time, the metacognitive component monitors and adjusts the KB by also adding and removing formulae from the same KB. As a result, the KB contains formulae derived from cognitive and metacognitive reasoning.

Since both processes proceed simultaneously in a step-by-step manner, they can influence each other immediately. Therefore, the agent is seamlessly self-reflective and deliberative as it processes the two types of wffs in each step. When the agent has to deal with too many formulae in a step,

 ISBN: 978-1-61208-273-8

then the default priorities help ensure that higher priority rules fire sooner than later. Also, by limiting the length of each step, the number of formulae processed in a step is kept within bounds and thus ensures that the agent remains responsive to new formulae that get added into the KB because of external changes or internal reasoning.

In the future, we plan to develop mechanisms to adapt metacognition to deal with changes in the intensity and frequency of violations. To this direction, we are currently exploring methods to change priorities of existing formulae.

ACKNOWLEDGEMENTS

This research is supported by a grant from the Office of Naval Research N00014-12-1-0430.

VII. REFERENCES

[1] T. O. Nelson and L. Narens, "Metamemory: A theoretical framework and new findings," *The Psychology of Learning and Motivation*, vol. 26, pp. 125-173, 1990.

[2] M. R. Cox, "Metacognition in Computation: A selected research review," *Artificial Intelligence*, vol. 169, no. 2, pp. 104-141, 2005.

[3] M. P. Georgeff and D. N. Kinny, "Commitment and effectiveness of situated agents," in *IJCAI'91 Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sydney, New South Wales, Australia, 1991.

[4] E. A. Hansen and S. Zilberstein, "Monitoring and control of anytime algorithms:A dynamic programming approach," *Artificial Intelligence*, vol. 126, no. 1-2, pp. 139-157, 2001.

[5] U. Ramamurthy and S. Franklin, "Memory Systems for Cognitive Agents," in *Proceedings of the Symposium on Human Memory for Artificial Agents, AISB'11 Convention*, York, United Kingdom, 2011.

[6] G. Alexander, A. Raja, and D. Musliner, "Controlling Deliberation in a Markov Decision Process-Based Agent," in *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portugal, 2008.

[7] A. Raja and V. Lesser, "A Framework for Meta-level Control in Multi-Agent Systems," *Autonomous Agents and Multi-Agent Systems*, vol. 15, no. 2, pp. 147-196, 2007.

[8] N. Alechina, B. Logan, H. N. Nguyen, and A. Rakib, "Logic for coalitions with bounded resources," *Journal of Logic and Computation*, vol. 21, no. 6, pp. 907-937, 2009.

[9] N. Alechina, M. Dastani, B. S. Logan, and J.-J. C. Meyer, "Reasoning About Agent Deliberation," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 22, no. 2, pp. 356-381, 2011.

[10] J. Elgot-Drapkin and D. Perlis, "Reasoning Situated in Time I: Basic Concepts," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 2, no. 1, pp. 75-98, 1990.

[11] K. Purang, D. Josyula, D. Traum, C. Andersen, and D. Perlis, "Practical Reasoning and Plan Execution with Active Logic," in *Proceedings of the IJCAI-99 Workshop on Practical Reasoning and Rationality*, Stockholm, Sweden, 1999.

[12] J. F. Allen and G. Ferguson, "Actions and Events in Interval Temporal Logic," *Journal of logic and computation*, vol. 4, no. 5, pp. 531-579, 1994.

[13] N. Rescher and A. Urquhart, Temporal Logic, New York, NY: Springer-Verlag, 1971.

[14] M. Nirkhe, S. Kraus, M. Miller, and D. Perlis, "How to (plan to) meet a deadline between now and then," *Journal of logic and computation*, vol. 7, no. 1, pp. 109-156, 1997.

[15] M. Miller and D. Perlis, "Presentations and this and that: logic in action," in *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, Boulder, Colorado, USA, 1993.

[16] J. Gurney, D. Perlis, and K. Purang, "Interpreting presuppositions using active logic: From contexts to utterances," *Computational Intelligence*, vol. 13, no. 3, pp. 391-413, 1997.

[17] K. Purang, "Systems that detect and repair their own mistakes, Ph. D. dissertation," Department of Computer Science, University of Maryland, College Park, Maryland, 2001.

[18] J. Doyle, "A Truth Maintenance System," *Artificial Intelligence*, vol. 12, no. 3, pp. 231-272, 1979.

[19] K. Purang, "Alma/Carne: Implementation of a Time-situated Meta-reasoner," in *Proceedings of the Thirteenth International Conference on Tools with Artificial Intelligence (ICTAI-01)*, Dallas, Texas, USA, 2001.

[20] S. Fox and D. Leake, "Using Interospective Reasoning to Refine Indexing," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Quebec, Canada, 1995.

[21] F. Dylla, A. Ferrein, E. Ferrein, and G. Lakemeyer, "Acting and Deliberating using Golog in Robotic Soccer - A Hybrid Architecture," in *Proceedings of the 3rd International Cognitive Robotics Workshop (CogRob 2002)*, Edmonton, Alberta, Canada, 2002.

[22] D. Josyula, F. Hughes, H. Vadali, and B. Donahue, "Modeling Emotions for Choosing Between Deliberation and Action," in *Proceedings of the IEEE World Congress on Nature and Biologically Inspired Computing - NABIC*, Coimbatore, India, 2009.

 ISBN: 978-1-61208-273-8