# Verification of Multiple Agent Knowledge-Based Systems

Daniel E. O'Leary*
*University of Southern California, 3660 Trousdale Parkway, Los Angeles, California 90089-1421*

This paper's purpose is extending single agent verification tests to systems with multiple agent knowledge bases. Particularly, this paper identifies potentially anomalous situations occurring between agent knowledge bases. For example, consider one agent with rule "if $A$ then $B$" and another agent with rule "if $A$ then $C$," so agents have a "consequence conflict" for the same condition. In this setting, agents are constantly at odds, unable to reconcile differences. Alternatively, the following rules are found in one agent (if $A$ then $B$, "if $C$ then $A$"), while another agent has rule ("if $B$ then $C$"). With two interacting rule bases, a dialogue starting with "$A$" cycles indefinitely, $A \rightarrow B \rightarrow C \rightarrow A$, depending on relationships between these two agents. Additionally, with traditional problems, this paper also identifies new issues, like agent "isolation" property. One potential approach to identifying anomalies in multiple agent systems is comparing each subset of agents' knowledge base determining existence verification issues. Where agents' number is small this approach is feasible. However, for even medium size systems this approach explodes computationally. As a result, alternative approaches are needed. This paper finds many multiple agent verification tests are conducted on metarule set generated from all rules contained in each of agents' knowledge bases minimizing computational effort and minimizing need for new verification concepts development. A wide range of "anomalies" are identified for multiple agent systems. Anomalies are system properties not necessarily incorrect. An important part of system structure is anomalous knowledge. For example, multiple agent systems employ negotiations and other devices facilitating agent interaction, mitigating anomalies. However, anomalies signal a system error. Ultimately, determination to dispostion of the anomaly is likely the responsibility of the developer. 

## I. INTRODUCTION

Interagent verification generates different problems than intra-agent verification. Suppose the agents are represented using rule-based knowledge bases and that both agents are supposed to provide a solution to the same task, say, to ensure reliability. Suppose the knowledge bases include the following rules: Agent 1 (if $A$ then $B$) and agent 2 (if $A$ then $C$). If both receive the same information then if $A$ occurs the two will come to different conclusions, a

* e-mail: oleary@usc.edu.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 16, 361–376 (2001)

potential problem if they are both interested in solving the same problem. As another example, suppose that two agents must work together to solve the same problem. If agent 1 has the knowledge base (if $A$ then $B$ and if $C$ then $A$) and agent 2 has the knowledge base (if $B$ then $C$), then interaction between these two agents could cycle indefinitely. These interagent issues are not addressed in traditional verification. As a result, the purpose of this paper is to elicit some of the potential problems associated with verification of multiple agent knowledge-base systems and find efficient approaches to identify them.

In so doing, this paper generates a conceptual structure for viewing agents to isolate particular verification anomalies. These anomalous circumstances are seen as a signal indicating a potential error in the development of a multiple agent model. However, the anomalous knowledge may be an important part of the system design, not an error, where agents employ negotiations and other devices to facilitate agent interaction, mitigating potential anomalies. Ultimately, the task of determining whether or not an anomaly is an error is the task of the developer.

### A. This Paper

The remainder of this paper proceeds as follows. Section II provides a brief discussion of the scope of the paper and previous research. In addition, Section II briefly discusses three different design approaches for building multiple agent systems, including "top-down" and "bottom-up." Sections III and IV establish different characteristics of multiple agent systems. In particular, Section III investigates the relationship between information and task similarity, while Section IV analyzes the impact of the organization of agents (e.g., hierarchical) and negotiation between agents. Sections V, VI, and VII analyze the different aspects of verification: correctness, consistency, and completeness. Different types of anomalies are found to be more likely in particular types of multiple agent systems described in Sections III and IV. Section VIII provides a brief summary of the paper and discusses some extensions.

## II. BACKGROUND, SCOPE AND PREVIOUS RESEARCH

This section establishes some basic background regarding the rule form of knowledge representation of the interacting knowledge bases, a brief discussion of the nature of verification, a brief summary of the type of multiple agent systems that are addressed, a review of previous literature on verification of multiple agent systems, a brief discussion of interagent vs. intra-agent verification, and a summary of the basic design approaches on which the results are based.

### A. Knowledge Representation

The author assumes that each agent's knowledge base is represented using rules of the type, "If condition then consequence." This is not a critical

assumption, however, it does facilitate explanation and presentation. Other forms of knowledge representation could be used, such as objects or multiple hybrid forms of knowledge representation could be used in different agents. However, the verification literature has focused on rule-based systems. As a result, this paper also focuses on rules, although the results presented here could be extended to other forms of knowledge representation.

### B. Verification

"Verification" was defined as "building the system right".[1] This view of the view of verification is one where the concern is with implementation of the "technology" of knowledge-based systems (e.g., rules, weights on rules, etc.) in a correct manner. Operationally, verification was defined by Adrion, Branstad, and Cherniavsky [p. 159][2] as "the demonstration of the consistency, completeness, and correctness of the software." As a result, the author's analysis of verification tests in multiple agent systems will concern itself with each of those elements.

### C. Multiple Agent Systems

The author uses multiple agent systems to refer to the existence of multiple independent agent rule bases. Agent rule bases are likely to represent different actors and their capabilities. The agents may have the exact same rule bases or their rule bases may be completely independent of each other. Sections III and IV discuss the nature of agent interactions and negotiations, and the role of information and task similarity in multiple agent systems as a means of defining the scope of agent actions in this paper.

### D. Previous Research on Verification and Validation of Multiple Agent Systems

There has been only limited investigation of verification and validation of multiple agent knowledge-based systems. O'Leary[3] investigated procedures for determining the existence of differences in expert judgment. O'Leary[4] presented results on determining the existence of conflicts between probability estimates of multiple experts in an influence diagram. Brown et al.[5] investigated the problem of validating heterogeneous and competing knowledge bases using a blackbox approach. Throughout that literature there has been little concern given to verification tests for those systems where knowledge bases interact, the focus of this paper.

### E. Intra-Agent vs. Interagent and Multiple Agent Systems

The focus in this paper is on some of the unique issues associated with "interagent," i.e., "between agent" verification problems. In particular, the concern is with unique verification problems deriving from the very nature of

having $n$ independent knowledge-based agents in the same system. The paper does not investigate "intra-agent" problems. In particular, it is assumed throughout that none of the anomalous verification situations (consistency, completeness, or correctness) occurs in the individual agent knowledge bases. Intra-agent verification issues can be addressed using procedures such as those of Nguyen et al.[6] and others.

### F. Design Approaches for Multiple Agent Systems

Verification inevitably is dependent on the explicit or implicit design of the multiple agents. If each agent is generated completely independently of each other with no concern for relating ontologies, then there is little hope for meaningful interagent verification. However, under other circumstances discussed in this paper verification can provide important insights. In particular, design of a multiple agent system typically would employ one of three basic approaches.

*Case* 1 (*Bottom-up*). Agents are designed independently of each other. However, the author assumes that if that is the case there is a common ontology across each of the agents so that each of the rule sets of different agents $A_i$, could be aggregated to form a metarule set of all the rules, $A'$. Although the resulting rule set would have some redundancies, removal of those redundancies could be used to generate the reduced rule set $A$. This approach is assumed implemented with each agent being a source of rule development.

*Case* 2 (*Top-Down*). Agents are designed, so that each agent $i$'s rule set $A_i$, is drawn from some meta rule set $A$, based on the same ontology, so that each $A_i$ is a subset of $A$. Thus, the set is developed and from $A$, rules for each agent are chosen. In the case where the agents are homogeneous, each set $A_i$ equals $A$. Ideally, this approach would be implemented by having each agent with the same rule drawing a referenced rule from a single source, rather than having rules independently, physically generated for each of the different agent knowledge bases.

*Case* 3 (*Autonomous Chaos*). Agents are designed independently using different ontologies. In this situation, it would be virtually impossible to determine similarity of rules because of semantic and syntactic differences in development environments.

The research presented here focuses on the first two cases, either where there is an "*a priori*" metaset of rules or "*a postiori*" metaset of rules, based on a single ontology that could be constructed from the different rule sets.

## III. HOW ARE AGENTS' ACTIVITIES RELATED?

In multiple agent systems, the relationship between agents' activities can be related across two primary dimensions: Information and task, summarized in Figure 1. Agents can be structured to use the same or different information.

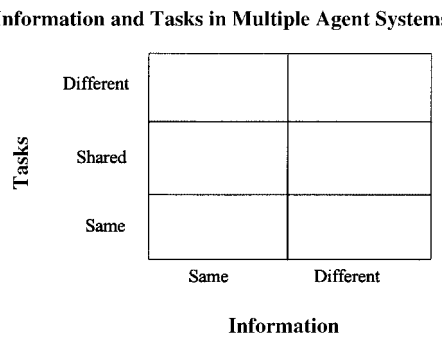


**Figure 1.** Dimensions of agent activities.

Similarly, agents can be asked to perform the same task, different tasks, or work together, sharing the task. Where a particular system fits, in terms of these dimensions, can be used to provide additional insight into the verification of that system. It will be seen that different anomalies are more likely in different sections of the figure.

### A. Same Task, Same Information

Agents designed to perform the same task with the same information, are typically employed in those settings where reliability is a critical issue. In this setting, agents may be homogeneous. For example, systems designed for use in space typically employ three homogeneous computing settings. Each computation is then performed by each of the agents, which are then polled to determine what consensus answer they recommend. The majority decision is then chosen.

### B. Same Task, But Different Information

In some problems, a number of agents are assigned the same task, but each has different information, e.g., in the case where there are multiple sensors with access to different information, depending on their location.[7] Agents designed to search out information on the web also could fall into this category, where agents are sent to different parts of the web to search for information of a particular type. In this situation, the agents may have homogeneous or heterogeneous knowledge bases, but, in any case, are designed with the same task outcome variable(s).

### C. Shared Task, Same Information

In this setting, multiple agents share generation of a solution for a task, while sharing information. Shared tasks may be done using a classic sequential assembly line approach, or there may be iterative solution. Shuttling small tasks back and forth between agents would provide a typical problem solving environment for this setting.

### D. Shared Task, Different Information

Rather than having access to the same information, task sharing agents may have access to different information, e.g., that corresponds to that necessary to solve the specific problem of concern. As a result, in this setting, different information may be available to or through different sensors.

### E. Different Tasks, Same Information

Blackboard systems provide one example where agents typically have different tasks but all have access to the same information, that on the blackboard. Because each agent is designed to get the same information, there can be concern for similarity of usage of that information, particularly if the agents are doing different tasks.

### F. Different Tasks, Different Information

Settings where agents are designed to get different information or information from different information sources probably are more likely than those with the same information to have problems, such as naming conventions, that inhibit communication between the agents. However, because tasks and information are different, it may be more difficult to identify those problems.

## IV. ORGANIZATION AND NEGOTIATION OF MULTIPLE AGENTS

Agents can be organized in many different ways, providing structure for interaction and resolution of potential conflicts between agents, as summarized in Figure 2. Organizational structures, ranging from flat to hierarchical, can provide agents with a means of resolving conflicts in knowledge or other anomalies, such as consequence conflicts between agents. For example, if an anomaly such as consequence conflict occurs, then the agent with the greatest level in the hierarchy can be chosen by the system as representing the correct

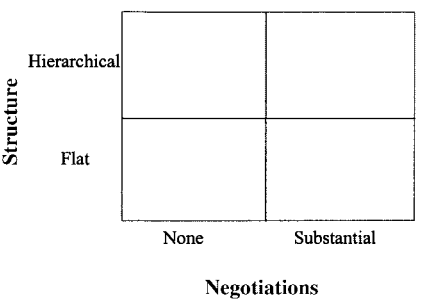


**Figure 2.** Structure and negotiations in multiple agent systems.

knowledge. In addition, multiple agent systems can be developed to allow a range of negotiation capabilities, ranging from no negotiations to substantial negotiations. Negotiation capabilities often are needed if the system is to assemble multiple capabilities.

It is seen that different anomalies are more likely in different quadrants of Figure 2. However, system performance ultimately can be influenced by errors that are unrecognized or camaflouged by capabilities of agents to resolve differences in favor of hierarchy or using negotiations.

### A. Flat Agent Organizations

In a flat organization, agents may be differentiated by time, or information flows that they have access to or by knowledge bases. However, no one agent has pre-emptive priority over any other agent in a flat organization. In this setting, if there is a conflict, the organization structure provides no criteria to establish which agent is correct, possibly resulting in anomalous system behavior. Unfortunately, since flat organizations of agents provide no ready basis of resolving problems between agent knowledge bases, it is critical to identify conflicts and other potential anomalies between agents at the time of development.

### B. Hierarchical Agent Organizations

In a hierarchical organization, there is a structure governing agent interaction. For example, there may be a tree structure that establishes explicit governing relationships between agents. In such a setting, if there is a conflict between the knowledge of any two agents then the hierarchical structure could provide the basis of resolution of that conflict. In particular, the structure would resolve the conflict in favor of the hierarchically superior agent. As a result, conflicts of knowledge and other apparent problems that can exist between agent knowledge bases may not be an apparent issue in hierarchical settings. Accordingly, systems that employ hierarchical structures may be viewed as more robust. Agents can have conflicts at the time of development and yet the system may still function effectively because the hierarchy provides a basis for resolving conflicts. Unfortunately, although the existence of conflicts or cyclic knowledge may not result in unusual system performance, there may still be incorrect decisions being made because of, e.g., underlying conflicts in knowledge between agents. As a result, even in those settings where there is a hierarchy there is a need to establish the existence of conflicts between knowledge of agents.

Further, hierarchical organization structures can provide an additional vehicle for verification. For example, if the agents are designed to be organized in a tree structure, verification can test the actual agent governing relationships to determine if they are implemented in a tree structure. Some errors could be those found in classic single knowledge-based verification, such as cycles (e.g., Ref. 6).

### C. Agent Negotiations Capabilities

To facilitate agent interaction, agents may be provided with the capability to conduct negotiations with other agents. In case of conflicts, negotiation capabilities could help agents resolve their conflicting interests. There is some concern that negotiation capabilities can cover up major unintended conflicts and other anomalies between agents. As a result, even if agents have the capabilities to generate negotiations, it can be important to perform basic interagent verification.

Further, adding negotiations to agents adds another level of complexity that offers other verification opportunities. Unfortunately, those structures lack identifiable generality. As a result, there is no further discussion here.

## V. CORRECTNESS

Correctness between multiple agents is concerned with three primary issues: conflict, circularity, and subsumption conditions. Each of the three correctness results that are generated, can be accommodated through analysis of the metaset of rules, $A$.

### A. Conflict

Interagent comparisons may find conflicts in the agent's knowledge. Those conflicts can interfere with effective system operation for a number of reasons. First, conflicts can make different agents produce different answers to the same queries given the same information. Second, conflicts can make it "difficult" for agents to negotiate, since no matter what is negotiated there is a difference of agent knowledge (opinion). Such a conflict is probably not as important if there is probability or certainty factor information carried along with each rule, however, it may still cause difficulties.

This section addresses two different types of conflicts: consequence and condition conflicts. A consequence *conflict* anomaly occurs if two (or more) agents have different consequences for the same conditions. For example, if one agent has rule If $A$ then $B$ and another agent has the rule If $A$ then $C$ then there is a consequence conflict between the two agents. This situation is of primary interest in those settings where each of the agents is responsible for performing the same task, either with the same or different information. In addition, it is particularly critical when there is a flat structure, with no hierarchy to eliminate conflicts. The following result provides an approach to finding consequence conflicts in multiple agent settings.

RESULT 1. *There is interagent consequence conflict if and only if there is consequence conflict in the metaset of rules, $A$, using either a top-down or a bottom-up design.*

Condition conflicts occur if two or more agents have different conditions for the same consequences. For example, suppose that one agent has the rule "if $B$ then $A$" while another agent has the rule if $C$ then $A$. If both get the same information "$B$" only one will come to the conclusion $A$, while the other waits for additional information, possibly never to come. As a result, this situation is primarily a concern where multiple agents are designed to solve the same task using the same information. Condition conflicts can cause substantial difficulty in those situations where there are no hierarchy or negotiation capabilities to help resolve the conflicts. As a result, condition conflicts probably are primarily a problem in flat organizations of agents with no negotiation capabilities. The following result provides an approach to determining the existence of condition conflicts in multiple agent settings.

RESULT 1′. *There is interagent condition conflict if and only if there is condition conflict in the metaset of rules, $A$, using either a top-down or a bottom-up design.*

### B. Circularity

Circularity can occur between a pair (or more) of agents in the following types of situations. Assume that one agent has the rules If $A$ then $B$ and If $C$ then $A$ and another agent has the rule If $B$ then $C$. Dialogues between those agents can result in circularity, starting from agent 1 with $A$, to agent 2 with $B$, to agent 1 with $C$ and then back to $B$, etc. This can result in "negotiations" that "do not go anywhere" or agent behavior can be described as being in a "rut."

Circularity anomalies are likely to be errors in those settings where agents share tasks and the same information. In addition, where there is no hierarchy and no negotiations between agents, there is limited capability to resolve such circularities.

RESULT 2. *There can be circularity between agents if and only if there is circularity in the metaset of rules, $A$, using either a top-down or a bottom-up design.*

### C. Subsumption

Subsumption occurs when a subset of conditions of some rule for one agent results in the same conclusion in another agent. If one agent has a rule If $A$ then $C$ and the other agent has the rule "If $A$ and $B$ then $C$" then there is subsumption. In multiple agent settings, subsumption indicates that the agents require differential amounts of information to draw conclusions. It may be that such subsumption is by design, e.g., one agent is designed to come to conclusions more rapidly than others. However, it can indicate an error.

Subsumption anomalies are likely to indicate an error when agents are designed to solve the same task and using the same information. In the example, with the flow of information $A$ one agent solves the problem while the other agent waits for flows of information. In addition, if agents are sharing a task,

subsumption anomalies are also likely to indicate errors, since, e.g., the agents may have different views about when the task is completed. Subsumption is likely to be an issue primarily when there is no hierarchy to help resolve resulting conflicts. The following result provides one approach to finding subsumption anomalies.

RESULT 3. *There can be subsumption in the rules between agents if and only if there is subsumption in the metaset of rules*, $A$.

## VI. CONSISTENCY

Consistency between multiple agents relates primarily to ensuring that redundant rules, used by multiple agents, stay the same, in spite of activities like maintenance and development. It further requires that ontology and naming conventions between agents are the same, because otherwise it would be difficult or impossible for the agents to effectively communicate without consistency conventions.

Consistency is primarily a problem in those situations where agents share tasks or do the same task and share information. In these settings, consistency between agents is required to accommodate the task or to interact about the task.

### A. Redundancy

Assume two (or more) agents ($j$ and $k$) have identical rules, say rule $r$ from $A$, in their knowledge bases, denoted, $a_{jr}$ and $a_{kr}$, so that $a_{jr} = a_{kr}$. The existence of redundant rules in different agents is not an error, per se, (in fact in homogeneous agents there will be entire rule sets that are the same, and thus, redundant) but if there is maintenance to the rules then depending on the maintenance errors can be introduced. Redundancy is something that is to be preserved, and not eliminated in multiple agent systems. The question is "how to best preserve that redundancy between agent knowledge bases."

In the case of bottom-up system design, if there is maintenance to individual rule sets then that could result in a change of either $a_{jr}$ and $a_{kr}$ but not the other (and other possibilities). As a result, a change to individual rule sets $A_j$ and $A_k$ from a bottom up approach could result in a new A (assuming it is formed), which would include the original rule and the rule that was supposed to be changed. The resulting system would be contrary to original intentions and thus in error.

Alternatively, using the top-down approach, if there is required maintenance to an individual rule then it should be done on a rule in the metaset $A$. This would eliminate problems of changing the rule in one rule set, but not another, or other similar problems. The resulting $A$ would be different than the $A$ derived from a bottom-up approach.

As a result, there is an implication for the design of multiple agent rule-based systems to preserve multiple agent redundancies: If there is a rule

that is used by more than a single agent then keep it in a repository and allow agents to reference the rule as part of their rule sets, whether the approach is top-down or bottom-up. This approach will limit errors due to changes in one agent but not in another for the same rule. Further, other additional redundancies are more likely to be preserved if virtually all the rules are maintained in a central repository.

### B. Ontologies and Related Issues

To ensure consistency between agents there is also a need for the agents to function under the same ontologies and to represent model variables using the same, e.g., naming conventions. The lack of a consistent ontology (or even the same "naming conventions") can limit the effectiveness of agent communications, since agents would not be talking using the same "language." As a result, system design and testing should be implemented so that there is a consistency in the ontology. However, if there is a situation where there is no such similarity, then for effective multiple agent communication to take place there must be individual agent capabilities with multiple ontologies or some superagent who has the ability to provide multiple ontology communication between the agents.

Problems with ontologies can derive from task differences and information differences. Hierarchical structures and negotiation capabilities can be used to generate solutions between agents without outside intervention.

### C. Naming Conventions

Perhaps the clearest impact of a lack of naming conventions in knowledge-based systems was presented in Landauer.[8] He demonstrated the need for and importance of establishing consistent naming conventions in his analysis of the manned maneuvering unit. For example, Landauer found 40 occurrences of "thrusters" and one occurrence of thruster, suggesting that the use of thruster is incorrect.

Consistent naming conventions are also critical in multiple agent systems. One design approach to ensure consistency of naming conventions is to establish a list of variable names that is used by all of the agents, facilitating communication between the agents. Otherwise, statistical tests of the occurrence of variable names, like those employed by Landauer[4] can be used to generate anomalous variable occurrences.

Naming conventions are probably most likely to result in anomalous situations when information differs between agents (e.g., Fig. 2).

## VII. COMPLETENESS

Completeness was characterized in single knowledge-base systems (e.g., Ref. 6) as deriving from four different characteristics. Those characteristics included unreferenced attributes (attributes that were declared as legal variables, but not used), illegal attributes (attributes that were not declared as legal

variables but were used), unreachable conditions (the condition should either match a goal or an if condition of another rule) or dead-end conditions (attributes must be askable or matched by a condition in another rule).

Finally, although it may be desirable to have agents that are virtually independent with no overlap, that situation should include criteria that is stated up front and is otherwise tested. As a result, the author defines "agency isolation" as a test of the completeness of agent development through determination of extent of isolation of particular agents. Ultimately, as with all anomalies, someone must decide if there is an error or just an anomaly.

### A. Unreferenced Attributes and Illegal Values

Development of a central repository of rules, attributes, and value sets expectations. Repositories of attributes and values can be generated and used in development to ensure that all attributes are legal (on the list) and that all attributes on the list are used, particularly in a top-down development environment.

### B. Unreachable Conditions

Next, consider the situation of determining the existence of unreachable conditions in different agents. As noted by Nguyen et al.[6] there is an unreachable condition in a goal driven production system when the condition of a rule does not match a goal or an "if" condition of another rule. By creating the set $A$ from the set of all agents, the set $A$ can be used to find some settings of unreachable conditions.

Unreachable conditions between agents are likely to most likely be a problem in those settings where agents share tasks and use the same information as other agents. The existence of unreachable conditions could limit the ability of agents to share task solution particularly in the case of sequential links between agents knowledge. This problem can be addressed using the following result.

RESULT 4. *There are unreachable conditions between agents only if there are unreachable conditions in the metaset of rules*, $A$.

### C. Dead-End If Conditions

Another multiple agent issue arises with dead-end if conditions. Nguyen et al.[6] define dead-end if conditions as occurring in those settings where conditions are not askable (i.e., the user provides information) or the condition does not match a consequence in some other rule.

Dead-end if conditions between agents are most likely an issue in those settings where agents share tasks, e.g., one agent solves one part of a task and then provides information about that subtask to another agent, that uses it to

solve a problem or to prepare information for another agent. Generally, this could be a problem because the tasks could sequentially link the agents knowledge. Dead-end conclusions can be addressed using the following result.

RESULT 5. *There are dead-end if conditions between agents only if there dead-end if conditions in the metaset of rules, A.*

### D. Agent Isolation

An important test of interagent consistency is the determination of whether or not there are any agents that are "isolated" from the other agents. Isolation is probably most anomalous in those situations where agents are designed to accomplish the same task, with the same information. In those settings, we would probably see agents have similar knowledge, with variations in the knowledge occurring as task and information differences were introduced. In addition, agents that were expected to perform negotiation would probably have at least a common set of variables that could be referenced as part of that negotiation. Further, hierarchically related variables would define a common set of variables for sets of agents, if a hierarchy existed.

There are at least two "levels" of isolation, representing different extremes. The author defines "level 1 isolation" as occurring if there is an agent with a knowledge base that has no rules that are the same as any other agent. The author defines "level 2 isolation" as occurring if there is an agent with a knowledge base that has no variables that are the same as any other agents. Given a top-down approach, with level 1 isolation, agents can at least talk about or infer about the same variables. In level 2 isolation, there is no basis of similarity between the agents. The occurrence of either form of isolation is likely to be anomalous in negotiation-based systems. Each form of isolation can be tested using a comparison of rules and variables of different agents.

## VIII. SUMMARY AND EXTENSIONS

This paper investigated verification issues unique to multiple agent systems, with particular interest in interagent verification issues. The author found that a number of classic intra-agent verification issues, such as correctness, consistency, and completeness also can manifest themselves as interagent anomalies, even if no individual agent has the problem. As a result, this indicates that verification of multiple agent systems must attend to interagent issues.

### A. Summary

This paper identified a number of anomalous situations and tied their existence to the nature of the information-task relationship, and the structure and negotiations used by the system. Anomalies are not necessarily an indication of errors because different agents may be performing different tasks or

have access to different information. Further, multiple agent systems may employ hierarchy and negotiation as a means of resolving and accommodating the anomalies.

Interagent verification was done using a metarule set resulting from combination of all agent's rules. A wide range of anomalies were identified, ranging from consequence conflicts to agent isolation.

### B. Extensions

There are a number of extensions of this research. First, this research can be extended to other forms of knowledge representation, such as frames or objects. Second, the results can be extended to different parts of the "information-task" matrix or the "structure-negotiations" matrix. Third, whether an agent is rational or irrational, is not likely to be a concern in most settings. Generally, agents are assumed to be rational. However, in some artificial life systems agent irrationality may be desirable. Ultimately, whether the agent's knowledge is rational or irrational, the results of this paper will be of direct application, unless irrationality is defined as inconsistent adherence to the knowledge in the knowledge base. As long as the knowledge base defines the scope of behavior, the results presented here can be used for verification of those systems where there are multiple agent knowledge bases interacting.

Some of the results summarized here were part of a presentation at the International Joint Conference on Multiple Agent Systems, San Francisco, June 11, 1995. The author thanks Les Gasser for his extensive comments on that version of this paper. Earlier versions of this paper were also presented at the Workshop on Verification and Validation of Intelligent Systems, at American Association for Artificial Intelligence (AAAI), RI, August 1997 and at Central Taiwan University, June 1998, as part of a two day seminar by the author on Intelligent Agents. The author thanks the participants for their comments in those presentations. In addition, the author thanks the referees from the AAAI Workshop for their substantive comments. Finally, the author thanks the referees for the Database and Expert System Applications (DEXA) workshop on Verification and Validation for their comments on an earlier version of this paper.

### References

1. O'Keefe R, O'Leary D. Expert system verification and validation. Artif Intell Rev 1993;7:3–42.
2. Adrion W, Branstad M, Cherniavsky J. Validation, verification and testing of computer software. ACM Computing Surveys 1982;14(2):159–192.
3. O'Leary D. Determining differences in expert judgment: Implications for knowledge acquisition and validation. Decision Sci 1993;24(2):395–407.
4. O'Leary D. Validation of computational models based on multiple heterogeneous knowledge sources. Comput Math Org Theory 1997;3(2):75–90.
5. Brown C, Nielson N, O'Leary D, Phillips M. Validating heterogeneous and competing knowledge bases using a black box approach. Expert Syst Appl 1995;9(4):591–598.
6. Nguyen T, Perkins W, Laffey T, Pecora D. Knowledge base verification. AI Magazine 1987;8(2):69–75.

7. Cammarata S, McArthur D, Steeb R. Strategies of cooperation in distributed problem solving. In Proceedings of the 1983 International Joint Conference on Artificial Intelligence, 1983, p 767–770.
8. Landauer C. Correctness principles for rule-based expert systems. Expert Syst Appl 1990;1(3):291–316.

## APPENDIX

RESULT A.1. *There is interagent condition conflict if and only if there is condition conflict in the metaset of rules, $A$, using either a top-down or a bottom-up design.*

*Proof.* If there is a conflict in the metaset of rules there are at least two rules where the same conditions result in different consequences. Since intra-agent conflicts are assumed to have been removed, this means that the rules must have come from different agents. As a result, there is an interagent conflict.

If there is an interagent conflict then there are at least two rules with the same conditions and different consequences. If that is the case, then there will be a conflict when the rules are aggregated into a metaset $A$.

RESULT A.2. *There can be circularity between agents if and only if there is circularity in the meta set of rules, $A$, using either a top-down or a bottom-up design.*

*Proof.* If there is a circularity in the metaset of rules then that circularity must have derived from the rules of at least two agents, since intra-agent circularities are assumed to have been removed. As a result, there is an interagent circularity.

If there is an interagent circularity then it occurs with rules in two or more agents. If that is the case, then there will be a conflict when the rules are aggregated into a metaset $A$.

RESULT A.3. *There can be subsumption in the rules between agents if and only if there is subsumption in the metaset of rules.*

*Proof.* If there is a subsumption in the metaset of rules there are at least two rules where the one rule subsumes the other. Since intra-agent subsumption is assumed to have been removed, this means that the rules must have come from different agents. As a result, there is an interagent subsumption.

If there is an interagent subsumption then there are at least two rules where one subsumes the other. If that is the case, then there will be subsumption when the rules are aggregated into a metaset $A$.

RESULT A.4. *There are unreachable conditions between agents only if there are unreachable conditions in the metaset of rules.*

*Proof.* If there are unreachable conditions in the metaset, then that means that for any agent that includes that rule, there is an unreachable condition in their rule base, since even if all the rules are included the condition still exists.

RESULT A.5. *There are dead-end if conditions between agents only if there are dead-end if conditions in the metaset of rules*, $A$.

*Proof.* If there are dead-end if conditions in the metaset then that means that for any agent that includes that rule, there is a dead-end if condition in their rule base, since even if all the rules are included the condition still exists.