Final Report

Grant Number: NAG2-1413

Title: **Motion Planning in a Society of Intelligent Mobile Agents**

Principal Investigator
Dr. Albert C. Esterline
Department of Computer Science
North Carolina A&T State University
Greensboro, NC 27411

Grant Period
6/15/00 to 6/14/02

Technical Officer
Dr. Michael Shafto
NASA Ames Research Center
Computer Sciences Division
IC:262-4
Moffett Field, CA 94035-1000

Administration
Office of Naval Research
Regional Office
100 Alabama St., SW, Ste. 4R15
Atlanta, GA 30303-3104

Abstract

The majority of the work on this grant involved formal modeling of human-computer integration. We conceptualize computer resources as a multiagent system so that these resources and human collaborators may be modeled uniformly. In previous work we had used modal for this uniform modeling, and we had developed a process-algebraic agent abstraction. In this work, we applied this abstraction (using CSP) in uniformly modeling agents and users, which allowed us to use tools for investigating CSP models. This work revealed the power of process-algebraic handshakes in modeling face-to-face conversation. We also investigated specifications of human-computer systems in the style of algebraic specification. This involved specifying the common knowledge required for coordination and process-algebraic patterns of communication actions intended to establish the common knowledge. We investigated the conditions for agents endowed with perception to gain common knowledge and implemented a prototype neural-network system that allows agents to detect when such conditions hold. The literature on multiagent systems conceptualizes communication actions as speech acts. We implemented a prototype system that infers the deontic effects (obligations, permissions, prohibitions) of speech acts and detects violations of these effects. A prototype distributed system was developed that allows users to collaborate in moving proxy agents; it was designed to exploit handshakes and common knowledge. Finally, in work carried over from a previous NASA ARC grant, about fifteen undergraduates developed and presented projects on multiagent motion planning.

The majority of the work on this grant involved formal modeling of human-computer integration. We conceptualize computing resources as a multiagent system so that these resources and human collaborators may be modeled uniformly. Section 1 presents background on multiagent systems. In previous work, we had developed a process-algebraic agent abstraction. Section 2 presents background on process algebras, and section 3 discusses our agent abstraction. In previous work, we had used modal logics for uniformly modeling agents and human collaborators. Section 4 discusses our application of our process-algebraic agent abstraction (using CSP) for this modeling; this allowed us to use tools for investigating CSP models. This work revealed the power of process-algebraic handshakes in modeling face-to-face conversation. We also investigated specifications of human-agent systems in the style of algebraic specification – see section 5. This involved specifying the common knowledge required for coordination and process-algebraic patterns of communication actions intended to establish the common knowledge. We investigated (see section 6) the conditions for agents endowed with perception to gain common knowledge and implemented a prototype neural-network system that allows agents to detect when such conditions hold. The literature on multiagent systems conceptualizes communication actions as speech acts. As discussed in section 7, we implemented a prototype system that infers the deontic effects (obligations, permissions, prohibitions) of speech acts and detects violations of these effects. The prototype distributed system discussed in section 8 allows users to collaborate in moving proxy agents; it was designed to exploit handshakes and common knowledge. Section 9 discusses the design of the negotiation mechanism for this testbed; this followed a promising new user-interface architecture and was expressed with Statecharts.

In work carried over from a previous NASA ARC grant, about fifteen undergraduate research assistants developed and presented projects on multiagent motion planning. Section 10 summarizes this work, and the Appendix lists the presentations on this work that were given at several conferences.

Section 11 gives the numbers and demographics on the students supported and graduated. The publication record for this effort is summarized in section 12, and section 13 lists the activities that were carried out in conjunction with this grant. Section 14 concludes and sketches future research directions.

1. Agents and Multiagent Systems

Wooldridge [Wo99] defines an agent as a computer system capable of autonomous action that meets its design objective in the environment in which it is situated. An *intelligent* agent in particular is capable of flexible autonomous action, where flexibility involves two somewhat opposing attributes: reactivity (the ability to perceive its environment and to respond to changes in it) and pro-activeness (aiming at goals by taking the initiative). In addition, since agents are autonomous, to achieve goals they must cooperate with other agents, so a third aspect of flexibility is social ability. Wooldridge notes that an object is superficially like an agent: it is a computational entity encapsulating a state in which it can perform actions (method execution) and communicate (message passing). The rather general notion of autonomy, however, has a clear application here since agents are distinguished from objects, for one thing, by the fact that the decision whether to execute an action lies with the object invoking the

method in an object system but with the agent receiving the request in an agent system. (The other two characteristics that Wooldridge identifies as distinguishing agents from objects are flexibility and the fact that a multiagent system is inherently multithreaded, which admittedly does not distinguish it from a distributed object system.) Note that autonomy relates to decision making, not to whether an implementation may execute in isolation.

Moving beyond individual agents to multiagent systems, one characteristic of multiagent environments identified by Huhns and Stephens [HS99] is that they "provide an infrastructure specifying communication and interaction protocols." The other characteristics of multiagent environments identified by Huhns and Stephens are that they are typically open (with no "centralized designer") and (echoing Wooldridge) the agents are autonomous and distributed. Abstractly, the open nature of these environments entails that smaller systems (agents) can be *composed* to form larger systems (multiagent systems), where composition involves coordinated concurrent activity. More concretely, it entails that agents that were not specifically designed for a given multiagent system may nonetheless participate in it. This requires flexibility on the part of the participating agents. Huhns and Stephens also see coordination as critical and, in the case of self-interested agents (where negotiation is involved), as a major challenge.

It is generally assumed that the agent communication language (ACL) is fixed for all agents that may participate in a given multiagent system. Messages in an ACL are generally conceptualized as speech acts. A message in an ACL generally distinguishes a performative, which denotes the kind of illocutionary act performed – for example, an assertion, request, or command. The message generally includes several predefined fields, including the message content, and fields indicating the language of this content and the ontology assumed. An ontology [Gu94] is a theory about what exists in a given domain and the basic properties and relations among these things. An ontology is critical for the semantics of the content of a message.

Coordination is essential for agents in a shared environment. A multiagent system must maintain global coherence without explicit global control. The agents must be able to determine common goals, enter into joint activities, reach agreements, pool knowledge, and know what the others know. Cooperation among non-antagonistic agents is coordination; negotiation is cooperation among competitive or simply self-interested agents. Negotiation is often framed in terms of game theory. Many cooperation protocols decompose and distribute tasks. The most widely used such protocol is the contract net protocol [Sm80], modeled on the contracting mechanism used by businesses. This protocol maintains a symmetry between a manager and a potential contractor in that the potential contractor has a say in whether it will take on a subtask announced by the manager. This is in sharp contrast to the RPCs used in frameworks. Indeed, while a client-server architecture is the norm in frameworks, a peer-to-peer architecture is more natural for multiagent systems. When, as is often the case, the decomposition of a task is not given in advance, multiagent planning is needed. To avoid communication bottlenecks and vulnerability to catastrophic failures, distributed planning is desirable even though it is much more difficult than centralized planning.

2. Process Algebras and Handshakes

A process algebra is a term algebra that can be used to describe communicating concurrent processes. The basic action in a process algebra is communication across an interface with a *handshake*. Two processes performing a handshake must be prepared at the time to perform complementary actions (usually thought of as output and input on a given channel, hence communication). So a handshake synchronizes processes. When a process in a system (itself a process, being a constellation of processes – see below) performs a handshake, the system undergoes a transition – some previously possible handshakes are no longer available, and new possibilities present themselves.

In CCS [Mi89], the possibility for a handshake is represented by a prefix. For example, consider a process C that may hold a single data item. We can write

$$C \stackrel{def}{=} in(x).\overline{C}(x) \qquad (1)$$

$$\overline{C}(x) \stackrel{def}{=} \overline{out}(x).C$$

The prefix $in(x)$ stands for a handshake in which a value is received at port in and becomes the value of the variable x. Thus, C, that is,

$$in(x).\overline{C}(x),$$

performs this handshake then proceeds to the definition of $\overline{C}$, which has a parameter, x, indicating that the value to which x is bound in the handshake is remembered. $\overline{C}$, that is,

$$\overline{out}(x).C,$$

outputs the value of x at port $\overline{out}$ then proceeds to the definition of C. Note that the two definitions used to define C can be collapsed into a single definition:

$$C \stackrel{def}{=} in(x).\overline{out}(x).C \qquad (2)$$

More formally, where E and E_i (i in some index set) are process expressions and y is a label (referring to a link or channel), we have the following five kinds of process expressions (built with five different combinators)

1. A *prefix* is of the form $\bar{y}(x).E$, $y(x).E$, or $\tau.E$. $\bar{y}(x).E$ is a *negative* prefix; $\bar{y}$ can be thought of as an output port of a process that contains it. $\bar{y}(x).E$ outputs x on port $\bar{y}$ then behaves like E. $y(x).E$ is a *positive* prefix, where y is an input port of a process; it binds the variable x. At port y the arbitrary value z is input by $y(x).E$, which behaves like $E\{z/x\}$, where $E\{z/x\}$ is the result of substituting z for all free (unbound) occurrences of x in E. We think of the two complementary ports, y and $\bar{y}$, as connected by a channel (link), also called y. τ is the *silent action*; $\tau.E$ first performs the silent action and then acts like E.
2. A *summation* has the form $\sum_{i \in I} E_i$, where the set I is a finite index set. The process behaves like one or another of the E_i. The binary summation is written as $E_1 + E_2$.
3. A *composition*, $E_1 \mid E_2$, is a process consisting of E_1 and E_2 acting in parallel.
4. A restriction, $E \setminus L$, where L is a set of (link) labels, acts like E but prohibits actions at ports y and $\bar{y}$ for all $y \in L$, with the exception of communication between components of E along any link $y \in L$. Restriction, like positive prefix, binds variables.
5. A relabeling is of the form $E[f]$, where f (a relabeling function) maps labels to labels.

In addition, a defined process (with arity n), $A(x_1,x_2,\ldots x_n)$, has a unique defining equation,

$$A(x_1,x_2,\ldots x_n) \overset{def}{=} P,$$

where $x_1, \ldots, x_n$ are distinct variables and the only variables that may occur free in P. $A(y_1,y_2,\ldots y_n)$ behaves like $P\{ y_1 / x_1, \ldots, y_n / x_n \}$ for the simultaneous substitution of y_i for all free occurrences of x_i ($1 \le i \le n$) in P. Since a defining equation allows recursion, either direct (as in(2) above) or mutual (as in (1) above), it allows us to represent repetitive behavior.

A process E may perform a communication action (that is, transition) α, corresponding to a prefix, and evolve into another process E'. This is indicated by the notation $E \overset{\alpha}{\rightarrow} E'$. The meanings of the combinators are formally defined by transition rules. Each rule has a conclusion (stated below a line) and premises (above the line). Two rules are of particular interest, the first being (where y is a label):

$$\frac{E \overset{y}{\longrightarrow} E' \quad F \overset{\bar{y}}{\rightarrow} F'}{E \mid F \overset{\tau}{\rightarrow} E' \mid F'}$$

This indicates communication between E and F, resulting in a silent action, τ, whereby a value is communicated between E (now E') and F (now F'). For example,

$$\bar{y}(z).P \mid y(x).Q \overset{\tau}{\rightarrow} P \mid Q\{ z / x \}$$

We state the second rule for binary summation (where α is any communication action), but it generalizes to arbitrary summation:

$$\frac{E \overset{\alpha}{\rightarrow} E'}{E + F \overset{\alpha}{\rightarrow} E'}$$

This indicates a kind of choice: if a process can behave like E or like F but E takes the opportunity to handshake, then the process behaves like E (doing α and evolving into E', with no contribution from F). For example,

$$(\bar{y}(z).P_1 + P_2) \mid y(x).Q \overset{\tau}{\rightarrow} P_1 \mid Q\{ z / x \}$$

Since restriction hides a link from outside observation or influence, we can force a pair of complementary labels in distinct parallel components to participate in a handshake with each other (if they indeed handshake) by restricting over the label. For example,

$$(\bar{y}(z).P \mid y(x).Q) \setminus y$$

(where 'y' abbreviates '$\{ y \}$') can perform only the τ transition indicated above.

3. Modeling Multiagent Systems with a Process-Algebraic Framework

There are several reasons to consider agents as processes and to model multiagent systems with a process-algebraic framework. Compositionality allows process terms for agents to be combined simply to form a term for a multiagent system. A process algebra encourages one to think of a process as capable of certain interactions with its environment that could be discovered by experimenting with it. When convenient, the environment itself may be modeled as a (system of) process(es) and compositionality

allows the process-cum-environment to be viewed as a single process (system). Communication protocols required of a multiagent environment can be formulated straightforwardly in process algebras, which are often used for formalizing protocols. In fact, handshakes are remarkably like speech acts in face-to-face communication. (Suchman claims on p. 71 of [Su87] that "... analyses of face-to-face communication indicate that conversation ... is a joint action accomplished through the participants' continuing engagement in speaking and listening.") The symmetry of a handshake distributes control between the participating processes/agents hence respects their autonomy. Since there are handshakes with the environment, the reactive (in the sense opposed to pro-active) nature of agents can be accommodated.

A process-algebraic framework for modeling multiagent systems does not so directly capture aspects of the agent abstraction that are not directly related to communication. The pro-active aspect involves (internal) planning, and negotiation involves (internal) computation by the negotiators in addition to their communication. Process algebras, however, typically allow "silent" actions (handshakes not observable outside the process making them) among components that have had their communication ports in some sense internalized within a single process. Sequences of silent actions offer hope for modeling computations internal to agents.

But not all processes are agents. Generally, a process P can perform a communication action, or *transition*, then behave like the process resulting from reducing P in a certain way. Some of the operators (and the syntactic patterns they govern) persist through transitions; an example is the (parallel) composition operator. Others, such as the alternative operator, do not thus persist – once one alternative is selected, the others are no longer available. Think of a system as an encompassing process whose several component processes – agents – persist (perhaps in reduced form) through transitions. We picture a pair of agents P and Q as connected by a link – a possibility for a handshake – when P contains a name denoting an action and Q contains a name denoting the complementary action. This framework, then, gives a picture of the communication linkage of a multiagent system.

4. Modeling Societies of Agents and Users Using CSP

As computing resources become more readily available, certain tasks now done by groups of humans may be achieved more efficiently and reliably by allowing humans to collaborate with computing systems even as they collaborate among themselves. We have shown how such collaboration can be formally modeled so that human users and computational entities are modeled uniformly [ME01,Mo01]. We consider abstractly the communications that systems (users and computing resources) perform. This part of our research addresses an abstract aspect of the topic NASA has termed human-centered computing [NA00], and it is related to dialogue modeling (see [DF+98], sec. 8.1), which addresses the exchange of information (conversation) between a user and a computer system. We abstract away from the syntax of interfaces by modeling communication events and their content.

We use the agent abstraction described in [ER01,ERH02] – a process-algebraic process that persists through communication actions – in modeling both humans and computational resources. Developing human-centered systems has unique demands since humans must also be modeled. A competence model of a user predicts legal sequences of

behavior; a performance model also addresses what is required to produce each sequence (see [DF+98], sec. 6.2). The agent abstraction allows us to formulate a competence model of the participants.

For this part of our research we used the process algebra CSP [Ro98] to express the agent abstraction. Alexander [Al87]], addressing the design of software geared toward human-computer interaction, presents a collection of languages, tools, and methods that views dialogue in terms of discrete events. Events are modeled with a functional language. The overall structure of the dialogue is specified with a subset of CSP, which defines the possible sequences of events. We, however, use only CSP as supported by FDR [FS97], a tool for analyzing CSP models that identifies deadlocks and livelocks and verifies refinement relations. This version of CSP allows data types to be declared and functions to be defined and applied.

As an example for applying the agent abstraction to human-computer integration, we consider an electronic purchasing system composed of human and non-human agents [Me01]. This allows (artificical) agents to negotiate the sale of a car without direct human intervention except to finalize the transaction. Each customer (buyer) is represented by a personal assistant, and each seller is represented by a sales representative. We would generally expect the customers and sellers to be humans and the other agents to be artificial, but this need not be the case. When a customer wants to buy a car, he sends information about the intended purchase to his personal assistant, who notifies the broker. If the broker is notified by a sales representative that its seller wants to sell the kind of car the customer in question is looking for, then the broker initiates a negotiation session between the personal assistant and that sales representative. If no sales representative informs the broker that it has the kind of car in question for sale, then the broker takes the initiative and asks the sales representatives whether they are selling such a car. If one replies positively, then the broker initiates a negotiation session between it and the personal assistant in question. If the broker cannot find a sales representative selling the kind of car the customer wants, then it waits until something becomes available. In a similar way but with the roles reversed, the attempted sale can also be initiated at the seller's end. The negotiation is handled by the personal assistant and the sales representative that the broker has paired. If they reach an agreed price, the respective customer and seller may concur and conclude the transaction, or one or both may cancel the transaction, in which case everyone starts over again.

Figure 1 is a flow graph representing our system specification with the exception of the channels used during negotiation between a personal assistant and a sales representative. In this particular specification, there are two customers (CU), each with its own personal assistant (PASS), and there are two sellers (SELLER), each with its own sales representative (SALESREP). We distinguish the two instances of each kind of agent by appending "I" to the name of the first and "II" to the name of the second. There is a single broker (BROKER) and a single database (DB), which contains information of the particular cars being sold. All channels are simplex, which makes the specification clearer. Where x is an abbreviation for agent X and y is an abbreviation for agent Y, the channel from X to Y is named x2y. The abbreviations for the kinds of agents are sr (SALESREP), cu (CUSTOMER), pa (PASS), sel (SELLER), br (BROKER), and db (DB). The abbreviation for the particular agent is formed by appending "i" or "ii" to the

abbreviation for its kind. Thus, for example, the channel from CUI to PASSI is named cui2pai.

The type of a channel is a sequence of more basic types. Any given channel may not use all the data fields available, allowing a single channel to be used for different kinds of communications. Every channel except those from the database begins, in the spirit of KQML, with a performative. On a channel from a customer to a personal assistant or from a seller to a sales representative, we communicate the negotiation parameters. Some channels communicate information about a vehicle, and some communicate economic actions in two fields: buy or sell and the model. When the broker pairs a personal assistant with a sales representative, it must tell the personal assistant the identity of the sales representative (sr1 or sr2) and the sales representative the identity of the personal assistant (pa1 or pa2). Finally, to conclude a deal, communications must include a response yes or no.

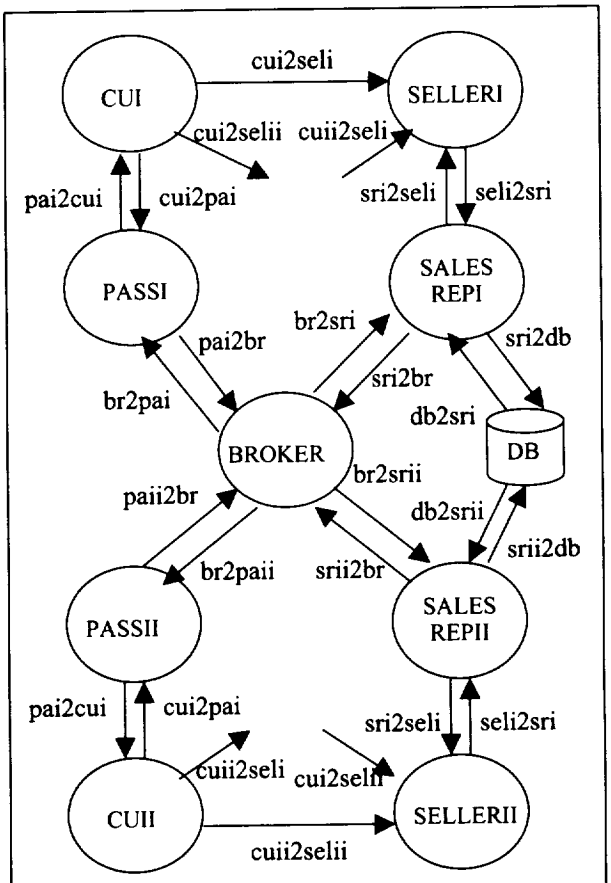

Figure 1. EPS Flow Graph

Since the full specification resulted in far too many states for FDR to handle, we scaled the specification down by reducing the range of values allowed for communicating data, by having only one customer, personal assistant, sales representative, and seller, by eliminating BROKER, and by generally simplifying the behavior of the retained agents. FDR confirmed that the resulting system, called BUYSELL, is deadlock free. It is not, however, livelock free. Investigation using the FDR Debug tool indicated that livelock arises when the CUI process keeps being told that the seller is not willing to sell and evolving back into itself. In fact, this is a possibility we anticipated, and no other source of livelock is present.

We abstracted out a component of the system and refined it so as to capture more focused aspects of human-computer integration. We refined the customer, CUI, expanding it to include an interface that allows humans at the periphery of the system to interact with the agents in the system. PERSON is a human and INT is the interface. INT communicates with the personal assistant, PASS, which is now a dummy. SELLER too is now a dummy; it comes into play only if and when the deal is arranged. FDR was used to confirm that indeed the resulting system failures-divergences refines BUYSELL.

5. Abstract Projects: Term Rewriting to Simulate Process-Algebraic Behavior and Attaining Common Knowledge

Bringing common knowledge into the account does not preclude a process-algebraic explication of the agent abstraction, as one branch of our research shows [Is01a,Is01b]. Here we are again concerned with how human users and automated systems may collaborate to complete projects. We again conceptualize the automated resources of the system as a multiagent system so as to represent users and system resources (or devices) as uniformly as possible. For now, we use the term "agent" to cover both (human) users ad automated resources. We attempted to specify a system of collaborating devices and a human user in the manner of algebraic specification. This involved representing a state of the system (thought of as a stage in a conversation) as a term with two components: one, in a restricted version of CCS, representing the behavior of the participants, and the other representing their common knowledge.

We use algebraic specification [vL89] to specify abstractly the situations in which agent communication takes place, the actions the agents may take in these situations, and the results of these actions. Software engineering techniques are generally structured around data rather than around functions since data are a more stable part of a system. So structuring around data yields specifications with a higher degree of continuity and reusability. The key point in structured design of software systems is to look for abstract data types (ADTs). A specification of an ADT describes a class of data structures by listing the services available on the data structures together with the intrinsic properties of these services. In specifying an ADT, we are not concerned with *how* a data structure is actually represented or *how* each operation is implemented. Rather, we are concerned with *what* the data structure signifies at the level of a customer who wants to instantiate the data type for use in his program. Specification modules abstract away all irrelevant details of data representation and procedure implementation. By avoiding side effects, properties of an ADT can be expressed in a simple and rigorous way. These properties are expressed as axioms and theorems in the form of identities (i.e., equations); the axioms formally describe the semantic properties of the algebraic specification.

By modeling ADTs with mathematical objects, we can reason rigorously about them. Rigorous reasoning in algebraic specifications is based on induction for general results. It is based on equational reasoning when we exploit the constructive nature [vL89, ch. 4] of these specifications to enable a kind of rapid prototyping. Equational reasoning is concerned with a restricted class of first-order languages: the only predicate symbol is equality. Equational logic is the foundation of term rewriting [BN99], but term rewriting uses equations as *directed* replacement rules, that is, the left-hand side can be replaced by the right-hand side but not vice versa. This constitutes a Turing-complete computational model that is close to functional programming. *Terms* are build from variables, constants, and function symbols. Given a rewrite rule

$$\alpha = \beta,$$

and a term t, one tries to match α with a sub-term of t, instantiating variables in α as required. A match with a sub-term t' of t defines a substitution $\theta = \{t_1/v_1, \ldots, t_n/v_n\}$, where $v_1, \ldots, v_n$ are the variables in α and $t_1, \ldots, t_n$ are sub-terms of t', such that $\alpha\theta = t'$. Then t' is replaced in t with $\beta\theta$. Note that any variables in t are not instantiated. In this respect, the pattern matching used in rewriting differs from unification. Variables in the target term cannot be used to transmit bindings and, in fact, are treated like constants.

We attempted to develop the notion of an abstract project, where a project is a step from an initial state of common knowledge to a state with additional, new common knowledge. We represent the state of the system with two components, one being the current common knowledge. The other is a term in a restricted version of CCS denoting the possible communication behaviors. In particular, the (parallel) composition operator may occur only as a top-level operator since the parallel components are the individual agents. This representation was encoded into the language of the automated reasoning system Otter [WP99], and equations were encoded for necessary CCS transition rules. Term rewriting (called demodulation in Otter) was then used to reduce the overall term to a normal form, where the behavior term is a parallel composition of process identifiers and a new common knowledge proposition has been conjoined to the old.

The scenario used to test this approach involved a human interacting with smart appliances for making breakfast. The agents are coordinated: they start together and remain in a steady state together; if any one fails or is interrupted, they all are. The state at the end of each step (joint project) must be common knowledge. We had to make assumptions about bounds on the time delay between some communication actions so that common knowledge may be achieved. Term rewriting reduced a term by simulating the CCS-specified behavior until a normal form was reached, where new common knowledge was asserted. Then another abstract project could be simulated, representing another step in the evolution of the system.

Several issues about term rewriting relative to process algebras were raised by this research. We are still working out these issues, and they are being addressed in a book chapter we are writing. These issues, however, should not obscure the general idea of this approach. An abstract specification of a joint project is initially specified in terms of the initial common knowledge and the new common knowledge reached. This specification is refined with a process-algebraic description of the possible behaviors that, given the initial common knowledge, will achieve the new common knowledge. Term rewriting is used to show that one of the designated states of new common knowledge must thus be reached. Note that other groups (systems) need consider the activity of the group whose joint activities are under consideration only in terms of the common knowledge of that group.

6. Heuristics for Inferring Common Knowledge via Agents' Perceptions

One graduate student investigated the conditions for agents endowed with perception to gain common knowledge and implemented a prototype neural-network system that allows agents to detect when such conditions hold [Wi00,WE01]. It has been proved that common knowledge is a prerequisite for reaching agreement and for coordinating actions among agents, whether human [CC82] or artificial [FH+95]. Since most environments are dynamic, the common knowledge agents have initially does not generally suffice for agents to coordinate their actions in all situations. So attaining common knowledge is a critical issue.

Common knowledge has been studied in several disciplines, such as philosophy, linguistics, game theory, and distributed systems. Philosophers and linguists have studied how humans can attain common knowledge in everyday life. In this context, the term "mutual belief" is often used instead of "common knowledge." We use these terms

interchangeably. Even though "*A* knows that φ" implies that φ is true, while "*A* believes that φ" does not, this difference is not critical here. The philosophical study has highlighted the role of special kinds of perceptual evidence and the mutual belief induction schema (formulated by Clark and Carlson [CC82]) for inferring mutual belief. In distributed systems, in contrast, the focus is on how artificial agents can attain common knowledge via available communication channels. The properties of the communication channels are the major factors for attaining common knowledge. It is concluded that agents can attain approximate common knowledge via reliable communication channels with bounded delivery time [FH+95].

This work showed how a special kind of perceptual evidence (physical co-presence) and the mutual belief induction schema may be applied to attain common knowledge in a group of artificial agents. The mode of perception we use as an example is vision since it supplies simple, clear examples. Our results, however, apply to any mode of perception that suffices to identify other agents and what they perceive. Most of the work addresses groups of two agents, but most of it easily generalizes to larger groups. Our results imply that coordination of multiagent systems need not depend on explicitly designed and implemented communication channels. Indeed, coordination may be implicit in the joint activities the agents pursue as long as they have shared perceptual abilities and common knowledge of these abilities. This is particularly significant since communication channels may fail, may be expensive to maintain, and may divert much of an agent's resources.

To focus on the episodic nature of physical co-presence evidence, we introduced a modal operator for seeing into the standard epistemic logic, and we introduced time parameters for modal operators. To attain common knowledge by physical co-presence, agents must model each other's perception, and this requires shared perceptual abilities and common knowledge of these abilities. Finally, we gave a procedure, which explores the structure of the perceptual models, for determining when common knowledge can be attained via perception in a group of two agents.

Common knowledge is critical in multiagent systems since it is a prerequisite for coordinated activity. It is not enough for agents to share information; they must detect when they have common knowledge so they may proceed safely with coordinated activity. A standard design would have rules for classifying perceived objects, including other agents, and rules for constructing perceptual models of other agents. We implemented a prototype of this sort and coupled it with a back propagation neural network [Wi00]. The behavior of the agent adapted to the feedback of the trainer so that the agent captured much of the "hidden knowledge," the facts, rules, probabilities, and other relations not captured by the knowledge engineer. We intend eventually to embed agents that detect common knowledge into systems endowed with sensors and actuators, and to connect the agents with communication channels, giving them additional opportunity for common knowledge. Fagin and Halpern [FH94] have provided a model for reasoning about knowledge and probability together, which is probably generally required in real-life situations.

7. Interpreting Speech Acts in Terms of Their Effects

One graduate student [Th01] developed a prototype program that tracks the effects of speech acts [Si98] performed by two interlocutors conversing via keyboard input. It

maintains a schedule for these effects (e.g., if A promises B to meet him somewhere at time T, this meeting is recorded in the schedule at time T), hence it is called the Performative Scheduler. (A speech act is a performative utterance.) A speech act consists of an illocutionary force and a propositional content. The illocutionary force determines the type of act, such as asserting or commanding. The propositional content is what is, for example, asserted or commanded. We explore in particular six categories of speech acts: assertives, directives, commissives, permissives, prohibitives and declaratives. Assertives are statements of fact; directives are commands, requests or suggestions; commissives are promises, commitments to the speaker's course of action; permissives give permission; prohibitives take permissions away; and declaratives entail occurrences of actions. The effects of speech acts of interest here are obligations (established by directives and commissives), permissions (established by permissives), and prohibitions (established by prohibitives); we call these effects deontic *effects*. In addition, assertives indicate beliefs, both on the part of the speaker and (as an effect) on the part of the addressee.

We have developed a context free grammar using definite clause grammar rules of the logic programming language LIFE [Ai94]. This grammar is able to handle a selected range of vocabulary items and syntactic structures used by the Performative Scheduler. To structure the values of attributes, we use feature structures [Ca92], which are the primary data structures of feature description languages and are supported by LIFE (known there as ψ-terms). These intensional, record-like structures represent partial information. Feature structures are also used as a repository of values for features, which help to interpret sentences in the language.

In the prototype, the user inputs a speech act and the prototype scheduler updates data structures that record the effects of the speech acts that have been addressed to it or that it has generated. The contents of the data structures are extracted from the feature terms constructed when a speech act is parsed. The prototype displays commitments and beliefs on the part of the users. The Performative Schuler maintains consistency among the deontic effects. This includes exploiting the following equivalences, which are standard theses in deontic logics [Åq84]. Let φ be any proposition, and let $O\,\varphi$, $P\,\varphi$, and $F\,\varphi$ mean that, respectively, it is obligatory that φ, it is permitted that φ, and it is forbidden (or prohibited) that φ. Then

$$P\,\varphi \equiv \neg O \neg \varphi \equiv \neg F\,\varphi$$

If one interlocutor performs a speech act whose effect is inconsistent with something already scheduled by him, then he is given the option to cancel the speech act or override the previous item. A speech act that is inconsistent with something already scheduled by the other interlocutor is disallowed. A simple interface allows events (especially actions by the interlocutors) to be tracked, and the Performative Scheduler flags violations of the deontic effects as they occur. Apparent violations due to the clash between beliefs and deontic effects are also flagged.

8. Human Collaboration Mediated by Handshakes

We have experimented with situations where humans collaborate in a handshake fashion [Hi01,HE01]. These situations involve several collaborators, each manipulating his own proxy agent. The collaborative aspect involves the participants signaling their intentions. This may lead to negotiation. At any rate, all parties agree to the broad

details of each other's plans. These agreements, however, are not enforced. What is crucial here is that communication is a sequence of handshakes. A handshake cannot happen until both parties are prepared to perform it. Also, a handshake is atomic in the sense that it cannot be partly done: an attempted handshake is either completed (successfully) or has no effect. This atomicity may be achieved by the occurrence of synchronizing events triggered by peripheral devices. For example, two collaborators may handshake by holding the left mouse button down when the mouse cursor is on the same button on their respective windows. Atomicity, however, can also be achieved by the analogue of what are called *adjacency pairs* in human conversation (as when the addressee shakes his head or says "Okay" at the end of the speaker's statement). These are similar to send-and-acknowledge communication protocols. For example, the addressee may click the **OK** button as the initiator elaborates his intentions. If the addressee does not click **OK**, say, because the message was garbled in transmission, then the handshake in which the message was meant to be communicated simply does not happen.

A computer-supported collaborative task that enforces handshake communication could allow activities that do not follow the handshake model. For example, an individual might access decision aids while collaborating with another who has no need or desire to see the details these aids provide. Or the second might want to have the details displayed on his screen to justify some step taken by the first. As long as the two are not jointly referring to some feature jointly displayed on their screens, agreeing, or disagreeing, there is no need for handshaking. Handshake communication between computer-supported collaborators forces them to attend concurrently to a sequence of communication actions. This is in the spirit of Clark's position on face-to-face conversation [Cl96], where the addressee is active – consider the nods, expressions, and interjections an addressee normally makes.

In one of our testbeds [HE01], the same grid is displayed on the screens of two collaborators (players). The grid displays obstacles and the start and goal positions of two proxy agents, each controlled by one of the players. Each player tries to move his agent from its start to its goal position, avoiding obstacles and the other agent, so as to minimize the sum of the single-cell moves made by both proxy agents. The obstacles tend to restrict motion along corridors. The players thus must have some high-level agreement about the paths of their proxy agents so that one is not forced to backtrack out of a long corridor. To this end, there is a whiteboard on which each player can portray the intended general path of his proxy agent. A proxy agent is not required to follow a path sketched for it, and an intended path may be elaborated at any time. Players can move freely from the grid to the whiteboard. Once a player has suggested a path segment, however, the players must eventually agree on something. If the other player initially disagrees, he can elaborate his own path or suggest an alternative for the first player. All communication relating to the whiteboard is handshake communication, and the protocol for turn taking while operating on the whiteboard is quite rigid.

Several undergraduate RAs developed projects around this testbed. One undergraduate presentation at the 2001 NC-LSAMP conference covered the general approach to this cooperative system [Hinds 01]. Several undergraduates have contributed to the development of the whiteboard and have given presentations on it [Hayes 01, Spears 01, Spears 02]. Several undergraduates have also given presentations on

identifying methods of using computer displays to facilitate highly cooperative tasks by making the evolving common ground manifest [Webb 01]. Other undergraduates gave presentations on the initial implementation of the distributed system using Tcl/Tk [Agyeman 01, Spratling 01]. Presentations also addressed the recent Java implementation [Gill 02]. Another undergraduate looked at ways to eliminate I/O bottlenecks in operating systems without sacrificing the structure of an operating system or applications [Khan 01].

9. A Statechart Design for a Whiteboard for Collaboration

One of our graduate students [Ma01a,Ma01b] produced a design for the interface for the whiteboard (in the testbed just mentioned) following the architectural approach presented by Horrocks [Ho99]. Horrocks advocates a three-layer architecture for user-interface design. In a usual user interface, event handler code is used to make individual objects work together. But, when a user interacts with an application, the interaction is with the group of objects making up the interface, not with individual objects. So the objects must work together; in the usual approach, the code making the objects work together is distributed throughout the event handlers.

A better approach is to centralize the control of the user interface in a small number of control objects, each responsible for coordinating the behavior of a group of related user-interface objects. Event handlers associated with interface objects are now used simply to forward user-supplied events to the appropriate control object. The control objects maintain the state of the user interface as a whole. They send messages to the user-interface objects as well as to the model objects. (A model object maintains the long-term information for the application and is not concerned with how the information is presented.) This gives a three-layer architecture – user interface, control, and model – whence the acronym UCM.

With direct manipulation, the user interface moves from one state to another, and the state, by defining the context in which an event occurs, defines the set of possible events that a user can supply. This is the event-state-action paradigm, which is followed by UCM architectures. Horrocks advocates Statecharts as an appropriate formalism for UCM architectures. One reason is that Statecharts allow concurrent components (substates of AND states), thus avoiding the state explosion experienced with products of finite state automata. Another reason is that Statecharts allow hierarchical decomposition (substates of XOR and AND states), thus allowing incremental refinement in the detail of a design.

Our design included all the detail required for negotiation among the players involved in using the whiteboard. An attempt was made to generalize the design beyond the two-player case. Most of the Statecharts represented the three-player case, which is enough to suggest clearly the structure required for any n-player case, $n \geq 2$.

10. Undergraduate Work on Motion Planning

Part of the effort for this grant carried on work from a previous grant from NASA ARC. This was a two-year (8/15/97 – 8/14/99) grant entitled "Motion Planning in a Society of Intelligent Mobile Agents" (Research Grant No. NAG2-1150) with the same PI, in the Department of Computer Science at North Carolina A&T State University. We addressed motion planning for a group of robotic agents. Several projects adapted the A*

algorithm to the case of multiple robots. Recall that the A* algorithm is a state-space search algorithm that maintains a priority queue of the states to select for expansion. Given a node n in the state space, the heuristic value associated with n is

$f(n) = g(n) + h(n)$,

where $g(n)$, which is known, is the number of steps from the start to n, and $h(n)$ is an estimate of the number of steps needed to reach the goal state. Candidate nodes are selected from smaller to larger values of f (hence the data structure for storing the children of expanded nodes is a priority queue). As long as $h(n)$ does not overestimate the number of steps, this best-first algorithm is guaranteed to find an optimal path in the state-space from the initial to the goal node and is then known as an A* algorithm.

In our applications, we use a grid that contains obstacles as well as a starting and goal cell for each robot. The aim is to get the robots from their starting points to their goal points in the least number of steps while avoiding collisions with the obstacles and other robots. For the single-robot case, at a node n in the search space, the estimate $h(n)$ is calculated as the Euclidean distance from the robot's current position to the robot's goal position. As the robot must move horizontally or vertically (from one cell to another that shares a side with it), the best the robot can actually do is the Manhattan distance from its current to its goal position (i.e., the sum of the difference between the rows and the difference between the columns of these positions). This is clearly greater than or equal to the Euclidean distance between these two positions (with equality only when the goal and current positions are in the same column or row and there is no intervening obstacle). For the multiple-agent case, suppose there are m robots labeled 1, ..., m. The variable part of the state is recorded as a vector $\mathbf{v}$ of m coordinates, where v_i is the coordinate of robot i. Let the vector $\mathbf{g}$ be the vector of goal positions and $\mathbf{b}$ be the vector of initial positions. The object is to find a path through the $2\times m$-dimensional space that takes the robots from $\mathbf{b}$ to $\mathbf{g}$ without colliding with objects or each other and such that the sum of the single-cell moves by the m robots is a minimum. Where n is the node in the search space corresponding to $\mathbf{v}$, $h(n)$ is calculated as $\sum_{i=1}^{m} d(g_i, v_i)$, where $d(g_i,v_i)$ is the Euclidean distance from robot i's position at n to robot i's goal position. By the fact that a summation preserves inequalities, it follows that $h(n)$ as thus calculated does not overestimate the sum of the distances the robots must travel.

Much of our work here was restricted to the two-robot case, but it generalizes in obvious ways to the m-robot, $m \geq 2$, case [Barnette 02]. Search-space nodes where robots come close to each other generate many descendants where the robots collide. Such nodes can be biased against by including a third positive term, $k(n)$, in the calculation of $f(n)$. $k(n)$ is a decreasing function of the distance between the agents [Saddler 01]. We have looked closely at the appropriate form for $k(n)$ [Adams 02]. One undergraduate considered improving the performance of the algorithm by detecting situations where all agents could take several steps at once without threat of collision [White 02]. Another undergraduate considered how this scheme may be enhanced by allowing some human control [Faulcon 01, Faulcon 02].

Some projects have adapted Lee's algorithm to multiple-robot path planning [Johnson 01, Coombs 02]. Recall that Lee's algorithm for a single robot finds a minimal path from an initial position to a goal position on a grid in the presence of obstacles. It does so in two phases. In the first, forward-sweep phase, cells are numbered by their distances (allowing for obstacles) from the source. This is done by expanding a frontier

with incrementing numbers out from the source. This phase finishes when the frontier reaches the goal. In the second, back-trace phase, a path is found by starting at the goal and following cells with decreasing numbers back to the source. A simple induction shows that this is guaranteed to find a source-to-goal minimum path. We adapt Lee's algorithm to obtain a parallel algorithm for multiple robots. The forward-sweep phase is done in parallel for all $m \geq 2$ robots. (This is appropriate for a multiprocessor system with shared memory; imagine one plane of memory dedicated to each robot.) The sweep is changed so that it goes from the goal to the source for each agent. In the back-trace phase, heuristics are used to avoid collisions in such a way that the total number of steps for each agent is increased as little as possible. This involves finding an appropriate function for determining when two agents are sufficiently close to threaten collision [Player 02]. By reversing the roles of the goals and sources, the actual robots may begin following the paths before the algorithm has completed. We have looked at ways to exploit priorities among agents and to group agents. Groups can be extracted from single-agent runs of Lee's algorithm. This also allows us to identify channels through which the groups travel. By exploiting the range of speeds less than the maximum for each agent, groups can be routed along channels in such a way that we avoid collisions among groups as well as within groups.

Other projects have looked at cell decomposition for motion planning [Ore 01, Manning 01]. In cell decomposition methods [La91], the space, including obstacles, through which a robot moves from source to goal is represented as a tree. Each node of the tree is a rectangloid cell, labeled as empty, full, or mixed. The children of a node represent its decomposition. Only a mixed cell may have children. We used quadtrees (appropriate for two-dimensional problems), where a rectangle is partitioned into four sub-rectangles. We use breadth-first search to find a source-to-goal channel of empty cells that on demand decomposes a mixed rectangle in search of adjacent empty rectangles to complete promising initial segments of channels.

11. Students Supported and Graduated

Three students completed masters theses [An02,Mo01,Wi00]. Of these, two were African-American and were partially supported by the grant, and one was unsupported. Three students, all African Americans, produced masters projects [Is01,Ma01,Th01]. Of these, one was fully supported by the grant, one was partially supported, and the third completed while employed full time. Twelve undergraduate students were supported at various levels during the 2000-2001 academic year, ten were African Americans, and eight have graduated. Ten more were supported at various levels during 2001-2002; nine were African Americans, and one has graduated.

12. Publications

The grant resulted in two book chapters [ER01,ERH02] and ten papers in conference proceedings [BE02,Es02,HE01,Hi01,Is01a,JE01a,JE01b,Jo01,ME01,WE01].

13. Activities

The following activities were pursued as part of the grant.

- One of two co-directors of the NASA ACE Center at North Carolina A&T State University.
- Took six students to the ADMI Minorities Computing Conference in Hampton, VA, May/June 2001. Three gave oral presentations of their papers and two had poster presentations. One oral presentation [Hi01] was awarded the prize for the best oral presentation.
- Moderator for the panel discussion "Communication and Coordination" at the NASA Goddard/JPL Radical Agent Architectures Conference, Tyson's Corner, VA, Jan. 2002.
- Assistant moderator for the "Formal Models of Multiagent Systems" session at the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001), Orlando, FL, July 2001. (The moderator was a former student.)
- Advised eleven students who gave oral presentations at the NC-LSAMP Undergraduate Minorities Research Conference in April 2001 at Greensboro, NC. One of the students was awarded first place in the category of computer science oral presentations. One of two recipients from the seven participating universities of the Outstanding Faculty Mentor award.
- Advised six students who gave oral presentations and one student who gave a poster presentation at the NC-LSAMP Undergraduate Minorities Research Conference in April 2002 at Raleigh, NC. One of my undergraduate RAs, Renard Spratling, won third place for his oral presentation in the computer science section at the Florida-Georgia LSAMP in January 2002. He won third place again for his oral presentation in the computer science/math section at the National Association for Equal Opportunity in Higher Education (NAFEO) Conf. in Washington, D.C. in March 2002.
- Mentor for two NASA SHARP Plus (high-school) students in the summer of 2000 and again in the summer of 2001.

14. Conclusion and Future Work

Our approach is to conceptualize computing resources as a multiagent system so that these resources and human collaborators may be modeled uniformly using an appropriate formalism. The work supported by this grant has demonstrated that process algebras are an appropriate formalism for this modeling. A process algebra offers rigorously defined operators for defining rich patterns of communication behavior in a straightforward way. Also, the handshake communication characteristic of process algebras – which happens only when both parties are prepared and where both parties play an active role – is a good model for speech acts in human face-to-face conversation. Still, a process-algebraic agent abstraction, as we have argued at length elsewhere [ER01,ERH02], must be supplemented with epistemic notions (especially common knowledge) and deontic notions (such as obligation and permission). This is partly because what agents or humans know about each other generally is not at the level of detail of behavior patterns. It is also because obligations, common knowledge, and so on hold in the situation at hand, and one must capture their influence. Thus, it is important also to model human-agent systems using epistemic and deontic logic, as we have already done [BE02]. Note

that, following the usual approach to software development, one begins with an abstract specification from which one develops a more concrete design. We have followed this approach in work where we relate a process-algebraic model to a modal-logic specification. We have also found that Statecharts are generally an easily used formalism for designing human-agent systems. (Process-algebraic semantics have been given for Statecharts.)

The work supported by this grant resulted in a significant number of publications. Six masters students, all but one of whom were African Americans, finished projects or theses by working on this grant. And a large number of undergraduate presentations resulted from this grant.

Future work will include close scrutiny of the adequacy of the process-algebraic notion of a handshake as a model for the signaling aspect of speech acts in face-to-face conversation. In everyday dialogue, there are frequently situations where more than two participants are involved in a speech act (as when the speaker is addressing a group). Theoretical foundations would be shored up if there were a multiagent process-algebraic handshake. The group in CCS generated by the set of particulate actions and with the commutative product $\times$ (on which the synchronous calculus is based) [Mi89] would seem to offer something close to what is needed. Often the allowable pattern of communication behavior in a group does not depend on the specific number of participants. This suggests an n-agent handshake, where $n \geq 2$ but is otherwise indeterminate. The connection between epistemic logic (especially common knowledge) and process algebras warrants detailed formal development. The obvious approach is to attempt to relate the epistemic modal operators to the modal operators of Hennessey-Milner logics [Mi89], which are similar to dynamic logic but with process-algebraic terms as the underlying computation model.

Common knowledge is being recognized as a key concept not only in economics, philosophy, linguistics and the psychology of language, and the theory of distributed systems but also in the social sciences in general. A recent book by Chwe [Ch01] identifies widespread social conventions and rituals for establishing common knowledge. All of this work can be tapped to establish empirically-grounded yet rigorous foundations for the design of systems (including human groups) that integrate humans with computational resources.

It is generally assumed in the literature on multiagent systems that inter-agent communication is asynchronous (in the sense that a sender does not wait for the receiver's reply, not in the sense in which a process algebra is asynchronous, i.e., without a global clock). If, however, communication among agents should be fundamentally similar to human face-to-face human conversation, then one would expect inter-agent communication to be synchronous in the first instance. One should undertake a reconstruction of agent communication to see how asynchronous communication can be founded on principles similar to face-to-face conversation. The starting point is recognition that patterns of transmitted bits are interpreted according to the conventions agreed upon by system designers. One should consider how asynchronous non-electronic communication (as, for example, enabled by writing) arose and depends on carrier reliability, bounds on delivery time, and time-stamping. The conditions for common knowledge arising for any variety of asynchronous communication should be investigated.

An important topic we have only touched on in the period of the grant (but see [Jo01,JE01a]) is game theory, which we have been finding more and more relevant. Given rationality assumptions, game theory explains how agents coordinate with a minimum of communication and without communicating plans. Game theory does, however, rely on the fact that the strategies available to agents are common knowledge. In fact, common knowledge is an important topic in game theory. Game theorists look for (Nash) equilibria of games, which are strategy profiles (each consisting of a strategy for each player) where no player could do any better (in terms of payoff) given the strategies selected by the other players. Of particular interest is a subgame-perfect equilibrium, where the strategy profile is a Nash equilibrium in every subgame, including those never reached along the path of play (viewing a game now in extensive form). A subgame-perfect equilibrium thus requires that threats or promises be credible, and such strategy profiles are essentially self-enforcing. Game theory has become an important tool for political scientists and political economists, especially in developing what are called analytic narratives [BG+98]. An analytic narrative pays close attention to the facts of an historical situation but gains insight from applying game theory to the alternatives available to the players involved. A given narrative may have several perspectives and frequently benefits from negative results that show that various perspectives are unfruitful. Game theory fits well into our approach. Actions always involve communication in some sense, and one should be able to describe players' strategies with a process algebra given that the appropriate participants can be identified and n-agent ($n \geq 2$) handshakes are available. Game theory has also received considerable attention in the literature on multiagent systems (e.g., [Kr01]), especially regarding negotiation.

References

[Ai94] H. Ait-Kace, *The Wild LIFE Handbook* (prepublication edition). Paris: DEC Paris Research Laboratory, 1994.

[Al87] Heather Alexander, *Formally-Based Tools and Techniques for Human Computer Dialogues*. Chichester, England: Ellis Horwood Limited, 1987.

[An01] Dominic Anderton, *A Formal Representation of the Spheres of Commitments*, MS Thesis, Dept. of Comp. Sci., North Carolina A&T State Univ., Greensboro, NC, 2000.

[Åq84] L. Åqvist, "Deontic Logic," in D. M. Gabbay and F. Guenther (eds.), *Handbook of Philosophical Logic*, Vol. II, Reidel: Dordrecht/Boston, 1984, pp. 607-704.

[BE02] Jamika Burge and Albert Esterline, "Using Epistemic And Deontic Logic To Model Societies of Agents", to appear in *Proc. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*

[BG+98] R.H. Bates, A. Greif, M. Levi, J.-L. Rosenthal, and B. R. Weingast, *Analytic Narratives*. Princeton, NJ: Princeton University Press, 1998.

[BN99] F. Baader and T. Nipkow *Term Rewriting and All That.* Cambridge: Cambridge University Press, 1999.

[Ca92] Bob Carpenter, *The Logic of Typed Feature Structures*. Cambridge, UK: Cambridge University Press, 1992.

[CC82] H. H. Clark and T.B. Carlson, "Speech Acts and Hearers' Beliefs", in N.V. Smith (ed.), *Mutual Knowledge*, New York: Academic Press, 1982, pp. 1-37.

[Ch01] Michael S.-Y. Chwe, *Rational Ritual: Culture, Coordination, and Common Knowledge*. Princeton, NJ: Princeton University Press, 2001.

[Cl96] Herbert H. Clark, *Using Language*, Cambridge, England: Cambridge University Press, 1996.

[DF+98] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-Computer Interaction* (2nd ed.), Hertfordshire, UK: Prentice Hall, 1998.

[ER01] A. C. Esterline and T. Rorie, "Using the π-Calculus to Model Multiagent Systems," in C. Rouff et al. (eds.), *Formal Approaches to Agent-Based Systems*, Springer-Verlag, 2001

[ERH02] A. C. Esterline, Toinette Rorie, and Abdollah Homaifar, "A Process-Algebraic Agent Abstraction," chapter in a book on multiagent systems edited by J. Rash to be published by Kluwer

[Es02] A. C. Esterline, "Handshakes, Common Knowledge, Obligations, and Agents," to appear in *Proc. NASA Goddard/JPL Conf. on Radical Agent Architectures*, 2002.

[FH+95] R. Fagin, J.Y. Halpern, Y. Moses, and Y. Vardi, *Reasoning About Knowledge*, Cambridge, MA: The MIT Press, 1995.

[FH94] R. Fagin and J.Y. Halpern, "Reasoning about Knowledge and Probability", *Journal of the ACM*, 1994, pp. 340-367.

[FS97] Formal Systems, *Failure-Divergences Refinement: FDR2 User Manual.* Formal Systems (Europe) Ltd, 1992-97.

[Gu94] M. Guarino, "The Ontological Level," in R.Casati, B. Smith, and G, White (eds.), *Philosophy and the Cognitive Sciences* (Proc. 16th Int. Wittgenstein Symposium), Vienna: Verlag Hölder-Pichler-Tempsky, 1994, pp. 443-457.

[HE01] O. Hinds and A. Esterline, "Joint Activity Coordination and Common Knowledge in Multiagent/Multi-person Environments," *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, Orlando, FL, July 2001.

[Hi01] O. Hinds (Advisor: A. Esterline), "Joint Activity Coordination and Planning in Multiagent Environments through Computer Integrated Communication," *ADMI 01*, Hampton, VA, May 2001.

[Ho98] Ian Horrocks, *Constructing the User Interface with Statecharts*. Harlow, U.K.: Addison-Wesley, 1998.

[HS99] M.N. Huhns and L.M. Stephens, "Multiagent Systems and Societies of Agents," in G. Weiss (ed.), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Cambridge, MA: The MIT Press, 1999, pp. 79-120.

[Is01a] A. Issa (Advisor: A. Esterline), "Rigorous Specification of Joint Activity in Human-Computer Collaboration," *ADMI 01*, Hampton, VA, May 2001.

[Is01b] Abdulcadir Issa, *Rigorous Specification of Joint Activity in Human-Computer Collaboration*, MS Thesis, Dept. of Comp. Sci., North Carolina A&T State Univ., Greensboro, NC, 2001.

[JE01a] R. Johnson and A. Esterline, "Game Theory Applied to Agents for Contracting Employment," *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, Orlando, FL, July 2001.

[JE01b] E. Johnson and A. Esterline, "Hierarchical Multiagent Motion Planning using Lee's Algorithm for Static Environments," *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics* , Orlando, FL, July 2001.

[Jo01] R. Johnson (Advisor: A. Esterline), "Match Maker for Contracting Employment," *ADMI 01*, Hampton, VA, May 2001.

[Kr01] Sarit Kraus, *Strategic Negotiation in Multiagent Environments*. Cambridge, MA: The MIT Press, 2001.

[La91] J.C. Latombe, *Robot Motion Planning*. Boston: Kluwer Academic Publishers, 1991.

[Ma01a] Karen Martin, *Communication in a Human-Controlled Multiagent System to Reach Common Ground*, MS Project Report, Dept. of Computer Science, North Carolina A&T State Univ., Greensboro, NC, 2001.

[Ma01b] Karen Martin, "A Statechart-based Design of a Whiteboard to Facilitate Common Ground in Computer-Based Collaboration," Poster Presentation, *ADMI 01*, Hampton, VA, May 2001.

[ME01] K. Mosley and A. Esterline, "Modeling Societies of Agents and Users Using CSP," *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, Orlando, FL, July 2001.

[Mi89] R. Milner, *Communication and Concurrency*, New York: Prentice-Hall, 1989.

[Mo01] Kevin Mosley, *Uniformly Modeling Societies of Agents and Users Using Formal Methods: CSP and VDM*, MS Thesis, Dept. of Computer Sci., North Carolina A&T State Univ., Greensboro, NC, 2001.

[NA00] NASA Intelligent Systems Program, *Intelligent Systems Program Plan*, Moffett Field, CA: NASA Ames Research Center, 2000.

[Ro98] A.W. Roscoe, *The Theory and Practice of Concurrency*. Prentice Hall PTR, 1998.

[Si98] M.P. Singh, "A Semantics for Speech Acts, in M. N. Huhns and M. P. Singh (eds.), *Readings in Agents*. San Francisco, CA: Morgan Kaufmann, 1998, pp. 458-470.

[Sm80] R.G. Smith, "The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers, C-29(12)*, pp. 1104--1113, 1980.

[Su87] Suchman, L.A., *Plans and Situated Actions: The Problem of Human-Machine Communication*, New York: Cambridge University Press, 1987.

[Th01] Donald Thaxton, *Interpreting Speech Acts with a Definite Clause Grammar and Feature Structures*, MS Project Report, Dept. of Computer Science,

North Carolina A&T State Univ., Greensboro, NC, 2001.

[vL89] I. van Horebeck and J. Lewi, *Algebraic Specifications in Software Engineering*. Berlin: Springer-Verlag, 1989

[WE01] S. Wiriyacoonkasem and A. Esterline, "Heuristics for Inferring Common Knowledge via Agents' Perceptions in Multiagent Systems," *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics* , Orlando, FL, July 2001.

[Wi00] S. Wiriyacoonkasem, *Adapting the Co-Presence Heuristic for Inferring Common Knowledge via Agents' Perceptions in Multiagent Systems through the Use of Neural Networks*, MS Thesis, Dept. of Comp. Sci., North Carolina A&T State Univ., Greensboro, NC, 2000.

[Wo99] *M. Wooldridge, "Intelligent Agents," in G. Weiss (ed.), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Cambridge, MA: The MIT Press, 1999, pp. 27-77.*

[WP99] Lary Wos and Gail W. Pieper, *A Fascinating Country in the World of Computing. Your Guide to Automated Reasoning*. Singapore: World Scientific Publishing Co.Ple., Ltd., 1999

Appendix: Undergraduate Presentations at Conferences

The NC-LSAMP (North Carolina Louis Stokes Alliance for Minority Participation) Conference for minority undergraduates in the areas of science, mathematics, engineering, and technology is held annually in April. The conferences in 2001 and 2002 were held during the period of this grant. The following lists the presentations given by my undergraduate RAs at these conferences. All were relevant to the topics of the grant. All the presentations were oral presentations except for the one given by White in 2002, which was a poster presentation. African-American students are indicated with an asterisk ('*').

NC-LSAMP 2001

Agyeman, Kwadwo*, "Multi-user Task Integration."

Faulcon, Michael C.*, "Robot Motion Planning using the A* Algorithm in the Presence of Multiple Agents."

Hayes, Juan*, "A Hierarchical Internet Whiteboard for Joint Task Planning of Mobile Agents."

Hinds, Oliver, "Joint Activity Coordination and Planning for Multi-Agent Environments through Computer Integrated Communication."

Johnson, Edgar*, "Hierarchical Multi-Agent Motion Planning using Lee's Algorithm for Groups of Multiple Agents within a Static Environment."

Khan, Muntasir Ahmed, "Operating Systems Support for High-Speed Communication."

Manning, Que'Shetta*, "Using Quadtrees to Find Channels for Robot Motion Planning."

Ore, Jenelle*, "Cell Decomposition in Quadtrees for Robot Motion Planning."

Saddler, Edward T.*, "Multiple-Robot Motion Planning using the A* Algorithm with an Enhanced Heuristic Function."

Spears, Patrick*, "Rules of Engagement for Computer Facilitated Cooperation."

Spratling, Renard*, "Multi-user Distributed Workspace for Joint Motion Planning."

Webb, Kimberly*, "Computer Displays of Common Ground to Facilitate Joint Activitites."

NC-LSAMP 2002

Adams, Beth*, "Modifying a Multiagent A* Algorithm to Favor Paths that Maintain Distance between Agents."

Barnette, Kenshasa*, "A Multiagent A* Algorithm."

Coombs, Nickolia*, "Multi-Agent Motion Planning using Lee's Algorithm."

Faulcon, Michael C.*, "Coordinating the Paths of Multiple Agents Independently Planned Using an A* Algorithm."

Gill, Jaspreet, "Joint Activity Coordination and Planning in Multi-Agent Environments through Computer Integrated Communication."

Manning, Que' Shetta*, "Two-Agent Implementation of the A* Algorithm."

Player, Malcolm*, "Using a Distance Function in Collision Avoidance in an Application of Lee's Algorithm for Multiagent Path Planning."

Spratling, Renard*, "Multi-user Distributed Workspace for Human-Computer Integration and Joint Motion Planning."

Spears, Patrick*, "Rules of Engagement for Computer Facilitated Cooperation: An Update."

White, Marcus*, "A Big-Step Heuristic to Increase the Efficiency of the Multi-Agent Implementation of the A* Algorithm."

Florida-Georgia LSAMP 2002

Spratling, Renard*, "Multi-user Distributed Workspace for Human-Computer Integration and Joint Motion Planning." (Awarded third place oral presentation in the computer science section.)

National Association for Equal Opportunity in Higher Education (NAFEO) Conference (Washington, D.C.) 2002

Spratling, Renard*, "Multi-user Distributed Workspace for Human-Computer Integration and Joint Motion Planning." (Awarded third place oral presentation in the computer science/math section.)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY)	2. REPORT DATE// Type	3. DATES COVERED (From - To)
04-11-2002	Summary of Research - Final	6/15/2000 - 6/14/2002

4. TITLE AND SUBTITLE
Motion Planning in a Society of Intelligent Mobile Agents

5a. CONTRACT NUMBER

5b. GRANT NUMBER
NAG 2-1413

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Dr. Albert C. Esterline, Project Director

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
North Carolina Agricultural and Technical State University
1601 E. Market St., Greensboro, NC 27411

8. PERFORMING ORGANIZATION REPORT NUMBER
4-42107

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Same as No. 7

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA ARC

11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Per §1260.22, NASA encourages the widest practicable dissemination of research results at any time during the course of the investigation.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

See Attached

15. SUBJECT TERMS
Human - computer integration, multiagent systems, process algebras, common knowledge, multiagent motion planning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Albert Esterline
					19b. TELEPONE NUMBER (Include area code) (336) 334-7245, Ext. 462