Using Diagnosis to Learn Contextual Coordination Rules *†

Bryan Horling and Victor Lesser
Department of Computer Science
University of Massachusetts

UMass Computer Science Technical Report 99-15

Abstract

Knowing when and how to communicate and coordinate with other agents in a multi-agent system is an important efficiency and reliability question. Contextual rules governing this communication must be provided to the agent, or generated at runtime through environmental analysis. In this paper we describe how the TÆMS task modeling language is used to encode such contextual coordination rules, and how runtime diagnosis can be used to dynamically update them.

Overview

Communication and coordination is an essential component of most complex multi-agent systems. Contention over shared resources, the desire to employ remote information and the need to coordinate interrelated activities may each require some sort of information transfer between agents to be resolved. To this end, individual actors in a multi-agent system must be able to explicitly or implicitly communicate requests and results, desires and beliefs, to make the system an efficient and cohesive unit. Thus, a set of situation-specific guidelines must exist which give the agent knowledge about the types of inter-agent communication which should be performed, when the communication should take place, and which other entities the agent should interact with.

A range of alternatives exist for providing or creating these contextual rules. If the agents operate in a stable and well understood environment, the most efficient solution may be to fully specify the rules a priori. More generalized, domain-independent agents, or those operating in a more dynamic environment, may rely completely on runtime learning algorithms to provide the necessary information (Sugawara & Lesser 1993). A third option combines these two by using adaptive techniques to dynamically revise provided or cached rules as environmental conditions are seen to shift.

We propose that the third option, coupled with diagnostic techniques capable of recognizing environmental changes, is a valid and efficient strategy for realizing contextual coordination rules. Under optimal conditions, the existing rules will suffice, but as conditions change the local contextual representation may be updated appropriately, thus allowing the agent to adapt to new conditions.

To frame the problem more concretely, let us consider the following scenario, drawn from our previous work in the Intelligent Home domain (Lesser *et al.* 1999). In this environment we have several autonomous but cooperative agents, each with its own capabilities, objectives and constraints, which strive to achieve their own goals without adversely affecting others in the environment. Each agent represents an "intelligent appliance" in the house (e.g. a hot water heater, dishwasher, dryer, etc.), with corresponding objectives. Also represented in the house are a set of bounded resources, such as electricity and hot water, which agents may need to make use of to achieve their goals. The resources will have minimum and maximum bounds, and will have different effects on activity when they are depleted. In this particular instance, consider a pair of agents which concurrently make use of the hot water resource. Assume the system designer originally primed the agents with a conservative strategy requiring coordination over all resources at all times. This guideline, while safe, may not be the optimal one for all circumstances, because there

*Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreement number F30602-97-1-0249 and by the National Science Foundation under Grant number IIS-9812755 and number IRI-9523419. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory, National Science Foundation, or the U.S. Government. The effort depicted is also sponsored by the Dept. of the Navy, Office of the Chief of Naval Research, under Grant No. N00014-97-1-0591; the content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

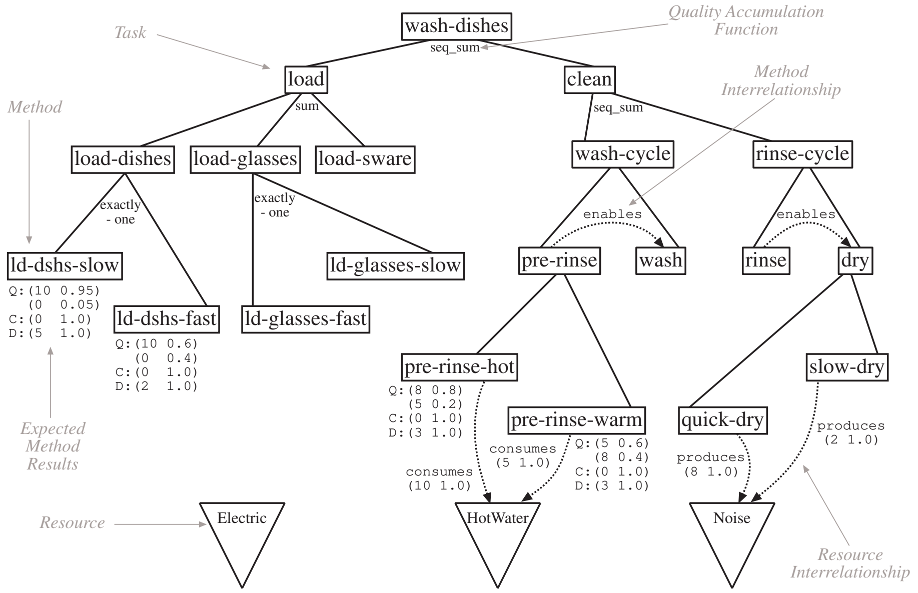

Figure 1: Abbreviated TÆMS task structure example for a dishwasher agent

may be sufficient resources available for both agents to complete their tasks regardless of the interactions between their local schedules. A diagnosis module, using knowledge about the execution characteristics and resource state, could determine that in this particular context, coordination is not necessary. If the environment changed, through the addition of competing actions or a reduction in the resource's level, the diagnosis could be revised to reflect the new circumstances, thereby re-activating the conservative strategy or recommending a new one.

The remainder of this paper will focus on our representation for these coordination rules and the mechanisms used to update them. We will conclude with a brief discussion of unresolved issues and future work.

Coordination Representation

The primary vehicle for the contextual representation of coordination is done with the domain-independent TÆMS task modeling language (Decker & Lesser 1993) (see Figure 1). A TÆMS task structure is essentially a goal decomposition tree, where leaf nodes represent executable primitive methods and internal task nodes provide a hierarchical organization. Tags indicating how quality is percolated up the tree, along with probabilistic descriptions of the executable methods allow a scheduler to reason about the traits and tradeoffs of a wide range of possible schedules. Interrelationships shown between internal tasks, methods and resources can be used to indicate a wide range of interactions, such as enables, facilitates, hinders and consumes (e.g. performing a task will *enable* the execution of another). Interrelationships may also span task structures between agents, and tasks and methods performed by remote agents may be represented locally.

TÆMS task structures are typically used to succinctly encode the different mechanisms for achieving a goal, and the constraints and tradeoffs associated with each plan. They may also be used, however, to describe those instances during execution where it has been determined that coordination is to take place. To do this, each agent will have two versions of its local task structure: *subjective* and *conditioned*. The subjective version contains what the agent believes to be the complete view of its local execution alternatives[1]. The conditioned view is a copy of the subjective which has gone through a process of *conditioning* - it may contain task, method or interrelationship deletions and modifications. When the conditioned view is used for plan construction, these modifications indirectly allow the problem solver performing the conditioning process to focus the attention of the scheduling and coordination mechanisms. These changes serve two purposes: to prevent certain subtrees from being considered during scheduling, and to remove interrelationships which are not to be coordinated over. In the first case, if the problem solver has determined that a particular action should not be performed, it can simply remove it from the conditioned view to prevent it from being considered. The second use is more germane to the issue at hand - by convention any interrelationships present in the conditioned view indicates that the interrelationship should be coordinated over when the related tasks or methods are scheduled for execution. Simply put, the subjective view defines all the existing relationships, while the conditioned view contains only the subset of

[1] We also have an *objective* view, inaccessible to the agent, which defines the real execution alternatives. We can engineer differences between these two views to create scenarios where the agent's expectations are not met.

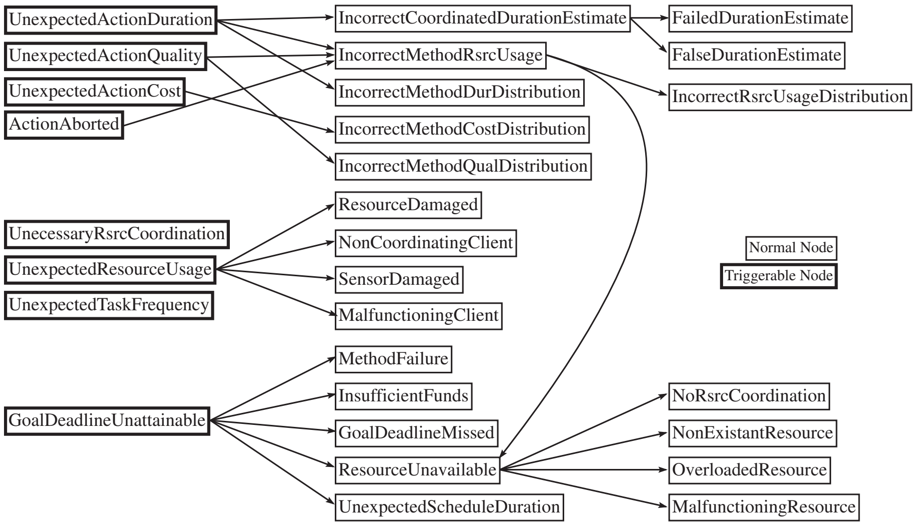

Figure 2: Example causal model structure for diagnosis in the Intelligent Home

relationships that require coordination.

Interrelationships in the conditioned view therefore have a special meaning. A (consumes) relationship between a method and a resource would cause the usage of that resource to be appropriately coordinated over with other entities in the system prior to use. A hard or soft (enables, facilitates) interrelationship arising from a method or task represented at remote agent with a local target would cause the local agent to contact the remote one to coordinate their activities in a satisfactory manner. A relationship indicating potential negative effects on a remote agent (disables, hinders) would require the agent to first determine if performing the activity is acceptable before execution. The scheduling of a method flagged as being non-local would cause the agent to coordinate over the timely execution of that method, perhaps with a contract net protocol.

In this manner, the TÆMS task structure has proved itself capable for our needs thus far in defining contextual coordination rules. The underlying characteristic which has permitted this representation is the natural coordination independence of the TÆMS task structure. The component performing the conditioning can determine which relationships to exploit, and represent them concisely in the conditioned view. The scheduler and coordination component are then able to make simple inferences based on the location and content of elements in the structure to correctly determine and perform the necessary coordination actions. It remains to be seen how far this technique can be pursued. As our library of possible coordination techniques grows, or more data is needed to sufficiently quantify the characteristics of a particular situation, it may be that additional contextual information may need to be added to our representation.

Adapting Coordination

With a reasonable representation of coordination in hand, we must now determine how to create and update the structure at runtime. As mentioned earlier, agents will initially be seeded with preliminary coordination rules. Although it is quite possible to learn appropriate rules from scratch, as our early work on this problem indicates (Sugawara & Lesser 1998), the hybrid scheme using seeding and adaptation was chosen both for efficiency reasons and to reduce the time to convergence on a reasonable coordination model.

Our feedback structure is driven by a flexible, causal-model based diagnosis engine. The causal model is a directed, acyclic graph organizing a set of diagnosis nodes. Figure 2 shows such a graph, constructed to address issues brought up by faulty coordination and action scenarios in the Intelligent Home domain; complete graphs addressing broader topics can be found in (Bazzan, Lesser, & Xuan 1998). Each node in the graph corresponds to a particular diagnosis, with varying levels of precision and complexity. As a node produces a diagnosis, the causal model can be used to determine what other, typically more detailed, diagnoses can be used to further categorize the problem. Within the diagnosis system, the causal model then acts as a sort of road map, allowing diagnosis to naturally progress from easily detectable symptoms to more precise diagnostic hypotheses as needed. Implicit in this architecture is the existence of components within the agent capable of reacting to such diagnoses, i.e. something which effects the adaptive behavior at the appropriate time. This can be done either with a single subsystem responsible for all adaptation, or by enabling components governing behaviors subject to diagnosis to react themselves.

An important goal of the diagnosis engine is both domain and coordination independence. In other words, it is desirable for diagnosis to function correctly without any inherent knowledge of the domain the agent is working in, or the mechanisms it may potentially use to coordinate with. It was decided to pursue coordination independence, rather than directly integrating with existing coordination frameworks, such as GPGP (Lesser *et al.* 1998), as an exercise to see how well high level directives could be used in a general way to control any type of coordination. We have seen that the TÆMS structure, used both as input to the diagnostics process and to instantiate any adaptive measure taken as a result of diagnosis, possess both these qualities. The question, then, is weather the causal model design and its underlying analysis techniques do as well.

There is nothing inherent to the causal model design which would restrict its usage to a particular domain or coordination technique, the specificity lies wholly within the individual nodes, the faults they attempt to diagnose and the methods used to diagnose them. It is therefore feasible that a carefully designed causal model, working from appropriately independent knowledge or dynamically gathered domain specific information at runtime, can satisfy our needs. Reflectively, it is also quite possible to design nodes which have direct knowledge about the specific type of coordination being used, so that the model may easily be extended in this direction if our initial attempts prove to be too general.

As an example of how a coordination independent model could function, let us revisit the over-coordination scenario described in the overview. The pertinent node in the causal model shown in Figure 2 is UnnecessaryRsrcCoordination (URC), and our acting agent will be the dishwasher, using the task structure shown in Figure 1. The URC node takes a long term view of resource coordination. It begins analysis when coordination is attempted over a particular resource, which can be detected by listening to the events produced by the coordination component in the dishwasher[2]. Once coordination over a resource has been noticed, the node will monitor whether or not these requests are being accepted or rejected by the other resident agents. In this case, diagnosis sees that coordination is being performed over the hot water resource, whose usage is shown by the two consumes interrelationships arising from the pre-rinse methods in the task structure. Once a sufficient amount of data has been gathered on coordination over hot water, the node then compares the accept-request ratio to a specified acceptance threshold. The ratio is determined at that time to be above the threshold, so a diagnosis is produced indicating that coordination may not be necessary over hot water. A confidence value is also attached to the diagnosis, which is currently determined by the relative distance between the observed acceptance ratio and the threshold. It is at this point that other components can react to the diagnosis in an appropriate manner. In the dishwasher, a problem solving component will update the conditioned view by removing the interrelationships between methods and hot water, which would in turn stop coordination over that resource. This will produce a conditioned view where all consumes interrelationships terminating at the hot water resource have been removed. When this is done, URC switches modes to listen for diagnoses arising from the NoRsrcCoordination node, which lies at the end of a chain of nodes which determine if a method's poor execution performance may be caused by a missing resource which has not been coordinated over. If at some later time this node determines that a method has performed incorrectly due to an insufficient level of hot water, URC may react by reducing the confidence on its original diagnosis. The problem solving component could then detect this change in confidence and update the hot water coordination rules accordingly.

[2]We use the JAF component-oriented agent architecture for agent construction. More information on this can be found in (Horling 1998).

This relatively simple heuristic can also be augmented to support a sliding window of relevant data, or indicate that only certain methods do not require coordination (e.g. ones that operate at certain times, or use less than some amount of resource) to produce more situation-specific rules. The important point, however, is that the diagnosis and reaction are being performed in a domain and coordination independent manner. The diagnosis technique knows only that a resource is being used, not how or why, and has only general knowledge that the resource may be coordinated over in some manner. Through passive monitoring of generic coordination events and performance data for related methods, we may also infer correct general policies for several of the other types of interrelationships mentioned in the previous section.

An alternative method to diagnosis is to start with a conditioned view devoid of interrelationships, and use conventional learning techniques to determine through exploration where interrelationships exist (Jensen *et al.* 1999). The fact that the learning algorithm can detect such interrelationships would indicate that it would potentially be advantageous to coordinate over them. If such a system could also learn where interrelationships did not exist, it might prove to be equivalent to the diagnosis-based one described above. This technique could provide a more accurate model of when to coordinate than the heuristic based one, but might do so at the cost of responsiveness.

Future Work

A working version of the system described above has been implemented, primarily targeting inefficiencies and faults in resource coordination. We plan on extending our causal model representation to better diagnose coordination over actions and results, and explore the effectiveness of different diagnostic techniques.

As mentioned previously, more research needs to be done exploring how well the TÆMS model can effectively represent coordination context. In addition, we also suffer from a lack of a formal model for several other environmental characteristics used during the diagnosis process (i.e. expected external resource usage, remote agent plans, etc.).

References

Bazzan, A. L.; Lesser, V.; and Xuan, P. 1998. Adapting an Organization Design through Domain-Independent Diagnosis. Computer Science Technical Report TR-98-014, University of Massachusetts at Amherst.

Decker, K. S., and Lesser, V. R. 1993. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management* 2(4):215–234. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

Horling, B. 1998. A Reusable Component Architecture for Agent Construction. UMASS Department of Computer Science Technical Report TR-1998-45.

Jensen, D.; Atighetchi, M.; Vincent, R.; and Lesser, V. 1999. Learning quantitative knowledge for multiagent coordination. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. Orlando, FL: AAAI.

Lesser, V.; Decker, K.; Carver, N.; Garvey, A.; Neiman, D.; Prasad, N.; and Wagner, T. 1998. Evolution of the GPGP Domain-Independent Coordination Framework. Computer Science Technical Report TR-98-05, University of Massachusetts at Amherst.

Lesser, V.; Atighetchi, M.; Horling, B.; Benyo, B.; Raja, A.; Vincent, R.; Wagner, T.; Xuan, P.; and Zhang, S. X. 1999. A Multi-Agent System for Intelligent Environment Control. In *Proceedings of the Third International Conference on Autonomous Agents*. Seattle,WA, USA: ACM Press.

Sugawara, T., and Lesser, V. 1993. Learning control rules for coordination. In *Multi-Agent and Cooperative Computation '93*, 121–136.

Sugawara, T., and Lesser, V. R. 1998. Learning to improve coordinated actions in cooperative distributed problem-solving environments. *Machine Learning*.