Is Explicit Representation of Initiative Desirable?

Bradford W. Miller

MCC[1]
3500 W. Balcones Center Dr.
Austin, TX 78759
miller@mcc.com

Abstract

The TRAINS-96 system does not explicitly represent initiative, yet it is clearly a mixed-initiative system. As we move toward commercializing efforts in discourse systems, it is important to remember that having a system make inferences that could otherwise have been "compiled away" can lead to slow response time. This paper is a statement of need for a taxonomy of domains for various features a discourse system may need to support the domain and an initial attempt at specifics with respect to the symbolic representation and manipulation of initiative.

Introduction

Chess machines have long been able to deal with mixed-initiative in a game setting. These machines appear to represent initiative when thinking about the move they should make because they think ahead to their opponents' countermoves, their own countercountermoves, etc. Chess is "mixed-initiative" because an attack begun by one side need not be answered by the other—it may be ignored and a counterattack initiated, the opponent may continue with their own attack should the counterattack be thought feeble, etc. Whoever is attacking may be said to have the initiative, but in general, the potential for initiative is the same as who has the turn. Nevertheless, a chess machine actually constructs and explores a graph using a heuristic and an evaluation function. This evaluation function may in some sense operationally encode initiative, but the system does not symbolically represent or manipulate knowledge structures for initiate.

The same can be said for most two participant mixed-initiative dialogue systems. That is, whoever has the turn potentially has the initiative. If a system just reacts to user input, however, it doesn't need to explicitly represent initiative (or the turn): either it is reacting, or it is waiting for the other participant's input. In other words, we can have the general strategy of reacting to user input include the effect of "seizing the initiative" as a side effect. For instance, if my reaction to a question is to ignore the question and pose another, I have seized the initiative, but this action need not involve my having reasoned about it. We are thinking here about systems that are, indeed, mixed-initiative; that is, they allow the other participant to seize the initiative and do what she wants rather than just reacting to a menu of choices, and by the same token the system can take the initiative and do something that wasn't explicitly requested by the other participant. We want to count systems like TRAINS-96 (Allen et al, 1996), but not simple question answering systems or phone menu systems that only allow the user to respond to a limited list of choices, anything else being "an error".

One might say these systems have an operational encoding for initiative, and I want to contrast this with systems that explicitly reason about initiative, such as described in (Guinn, 1996). The essential question is: is there some feature(s) of a domain that requires us to explicitly represent and reason about initiative? Can mixed-initiative interaction be an "emergent property" of a system that does not actually "think" about initiative at all? Can we come up with a domain taxonomy that tells us what features of the domain (or the human machine interaction) may force us to build a system to reason explicitly about initiative?

Properties of the Domain

In order to properly examine this question, we need to restrict the range of systems we are going to think about. In particular, we will only deal with systems that have two participants, one a human, the other a computer (on which the system runs). The system should handle natural dialogue: dialogue as people use it. And our system should be able to run reasonably close to real-time. That is, we don't want to use techniques that require us to do something fundamentally hard, e.g., exponential search.

Parallels with Planning

In order to have a system account for the interactions with a user, it's hard to abandon a planning model (Allen et al, 1996; Pollack, 1992), though it is possible to have a sys-

[1] The author's current address is at MCC, however, most of the work mentioned in this note was accomplished at his prior affiliation with the Computer Science Dept. University of Rochester, Rochester, NY 14627-0226 with James Allen (james@cs.rochester.edu).

tem manage a mixed-initiative dialogue without doing planning of any sort (Baekgaard, 1996). The essential difference is the range of behaviors we expect the system to understand and undertake. Existing general plan recognition models (e.g. Allen & Perrault, 1980; Litman & Allen, 1987; Carberry, 1990), however, don't lend themselves to real-time interaction. It is important, then, that we do not force ourselves into having to solve the general plan recognition problem in order to reason about initiative. But parallels with planning remain: (Cohen and Levesque 1990) point out that planning systems that operationalize their intention (saying that the intention of a planning system is the content of its plans) have the problem of dealing with agents who form plans that are not adopted. Furthermore, "communication involves one's ability to reason about the intentions of others" (citing Grice, 1957; Perrault & Allen, 1980).

The question that arises then is: can we reason about the intention of others *without* explicitly dealing with initiative? I argue that we can if the domain and our expectations of the system interaction meet certain criteria. I want to set out a subset of the properties that a domain and interaction may or may not have, in order to decide if an "explicit model" or an "operational encoding" is needed to build a system in that domain. This is necessary because in general, explicit symbolic models require far more computational effort to manipulate, as well as generally more programming infrastructure—something to be avoided if possible when actually developing a system to support commercial product. The rational proposed in this paper is theoretically "weak"; that is, I do not seek to formally prove that an explicit symbolic model is needed (since it is easy to prove the converse). Rather, I rely on what is currently possible to do simply, with available tools, and our current understanding of discourse systems.

The TRAINS-96 System

The TRAINS-96 system was designed to handle natural dialogue, run in near real-time, and have evaluable results (in our case, we used a metric of time to task completion and length of solution, as reported in (Allen et al, 1996)). The domain is a simple one, that of planning trips for engines on a displayed map of cities and tracks, but it is complex enough to show interesting discourse phenomena. The basic architecture of the TRAINS system include a multi-modal input component, a semantic parser driven by this input whose output is the surface speech-act recognized, a discourse manager including a reference module, a context manager, a problem solver, and other rule based reasoning components, and an output module that updates the display or hands off text to a text-to-speech program. This architecture is discussed in more detail elsewhere. The question here is: what characteristics of the domain and system allowed us to avoid needing to explicitly track initiative, while still dealing with mixed-initiative interaction?

1. The designer of the discourse manager was aware of the system's domain reasoning capabilities. That is, the system never needed to reason about its knowledge with respect to its ontology; it either could take a domain action, answer a question, etc., or it could not. This bias was built into the rules the system used to deal with its inputs.
2. Turn taking was strictly enforced. The system did not need to deal with interruption or "barge-in". So it could assume that once the human finished her turn, it would get a chance to respond completely before needing to begin processing the next input. The implicit assumption is that the human would not relinquish the turn until she was ready for the system to have the initiative (either because a problem had been presented to the system or because she needed further prompting as to what to do.)
3. There was only one class of "user"; that is, we did not deal with the notion of novice vs. expert users explicitly in the user model.
4. All domain requests had the same cost—we didn't need to deal with calculating if an ideon (the quantum unit) of information was cheaper to simply retrieve or to ask the human for. Basically information accessible to the system constituted everything the system "knew". Also, we did not model local vs. remote information.

The TRAINS-96 discourse manager used a rule-based system to interpret speech acts from the user (as analyzed by the parser). The rule-based interpretation of utterances allowed the system to be reactive. That is, once it finished its turn, it waited until there was more input, and then dealt with that input in light of the current environment (the environment as amended by the prior output). Because the rules basically match patterns in the input and the environment (Miller, 1997), it was not a matter of reasoning about the appropriateness of "seizing the initiative". The discourse manager's task was having the highest utility matching rule either respond to the users input, start a clarification subdialogue, switch the focus to a new (sub-)task, or otherwise try to carry on based on the current state of the discourse and the recognized plan. Mixed-initiative interaction was an effect planned and designed into the problem solving language (Ferguson & Allen, 1994; Ferguson et al, 1996), and by allowing all rules to match. In other words, the expectation of a particular response was not hard coded—the response was taken in the light of the expectation (a discourse state), and if the expectation was not met, it matched a different rule (Miller, 1997a).

So, in TRAINS, the system has the apparent initiative whenever it is its turn, that is, whenever it is actually processing what the human participant said during his turn. The system then does as much as it can, and "tosses the ball" back to the user.

Taking each of the assumptions above in turn, I want to examine in the next section if relaxing them would require us to explicitly represent initiative in the system. This is

important because in the I3S project, where we are in effect building a version of TRAINS with tools suitable for interfacing it to different application programs (Rudnicky, 1996), we will not have the luxury of knowing what the ultimate domain will be when we build the discourse management components. We also want to be able to determine, for a particular domain, the set of discourse tools needed to implement that domain's discourse system. Or, more realistically, for a given set of discourse tools, what features of a domain or interaction we would (not) be able to support.

Generalizing the Observations

Given that the domain features detailed in the last section kept TRAINS-96 simple, let us look at the changing requirements when we violate those domain assumptions. First, suppose we want to build a general discourse management engine, which means that the designer will not be cognizant of all of the domain reasoning capabilities. To the extent that our problem solving language does not change, there would appear to be no immediate impact on the discourse manager (DM)—that is, either the plan recognizer can come up with a reasonable interpretation of an utterance or it cannot, and in the latter case, the DM would then attempt an alternate strategy. It does mean, however, that some of the strategies may be domain dependent, and our application engineer may need to add new rules to the DM to handle these application dependencies. This will allow us to continue to "compile in" our knowledge and not have to reason about "what we know". While reasoning about our knowledge is an interesting problem in its own right, it's beyond the capability of current real-time discourse agents. The basic rule of thumb seems to be (from practical experience): if you need to use an inference engine, you will not be able to produce a result in real-time.

Let's look at the second item then: barge-in or interruption support. From a discourse perspective, the phenomena of barge-in is a fairly natural occurrence and can happen for a variety of reasons including:

- The need to specify additional information the participant deems important to the completion of the task at hand (e.g., answering a query). Alternatively, the realization that the speaker is missing a crucial bit of information, possibly invalidating their prior utterances. Note that this is the form of the first interruption the manager makes in the dialogue of the appendix.
- The realization that the response being given is inappropriate for the task the participant had in mind. For instance, the first phrase from the system after a query may indicate a misunderstanding. In this case, the user would want to interrupt the system to correct the problem.
- The realization by a participant that the current (discourse) task is of less importance than other tasks (which may be serendipitously realized). For example, if the current task is to get a certain cargo to a certain city, and if the participant realizes that it can be accomplished as a side effect of a larger plan, it may be appropriate to immediately switch the topic to the larger plan and to mention the current task as a subgoal. This would be superior to having the current focus on accomplishing the subgoal necessitating further complications when trying to deal with it as a step in a larger plan (e.g. involving different participants).

In the case of fairly repetitive responses (e.g. navigating voice menus via telephones) the user often recognizes what the system is about to say from prior experience or through hearing the prefix of the entire utterance. (In this case the user may not listen to the entire utterance. This is the common usage of "barge-in" in the telephone industry, and is commonly described as "speaking during the prompt".) In the Appendix, I present a dialogue sample that demonstrates some of these phenomena.

As difficulties, we also need to recognize that handling barge-in in a conversational system may also lead to having critical information being suppressed. That is, we need to take account of the fact that while we may have intended to say something, we might not actually have had the chance to say it.

Here, we examine two cases: do we want the system to handle interruption by the human participant, and do we want the system to be able to barge in on the human's input. In the former case, I suspect the answer is again that the system does not need to explicitly handle the notion of representing initiative because it normally knows who has the turn (e.g., it's busy listening to the user). By simply giving the human priority, the system stops work on its recognition/response generation tasks whenever the user begins to speak, and either defer any work it has yet to complete, cancels it if the discourse state indicates an interruption and topic switch, and/or marks as uncommunicated any speech acts that have not yet been spoken (or displayed). The system can basically "carry on" given the extant discourse state, again taking up the initiative when the human has stopped giving us input (which might be typed or gestural—a speech centric interface is not presumed).

There are engineering issues that need to be considered:

- We need to suppress the audio output of the system.
- We need to know what has been said; that is, how much of the text given to the text-to-speech (TTS) engine has been output.
- We need to be able to track the TTS output back to the set of speech acts that have been said, including the possibly of a partial act (much of the illocutionary force of an act may be contained in the prefix of a realization of an act).
- We need to be able to update the discourse context for what was actually said, rather than what was scheduled to be said.

None of these are trivial issues to be casually dismissed. However, none seems to require the explicit representa-

tion of initiative either. Initiative is still operationally represented in the state of the system, and does not need to be symbolically reasoned about.

Taking up the other question—namely, whether we want the system to be able to barge in on the human's diatribe—we first have to look at the requirements we are addressing. E.g., the system has something to say that bears on the task at hand and is of critical import (for instance, it may have just been informed of a train wreck). Or perhaps the system has a question that must be answered, which the human is apparently ignoring based on a prefix to her utterance. It would seem that the system needs to have the following capabilities:

- Rather than simply processing when it's the system's turn, it will have to do some work during the user's turn. For example, it may need to monitor its knowledge for new information the user needs to be apprised of, and/or use information from partial parses of the current input to see what the user is currently saying. (Is he addressing the problem at hand, or does he seem to be involved in a topic switch?)
- Given this situation, we now need to know whether the human is currently giving us input, if for no other reason than to avoid unintended interruption.

Is this a cheat? By requiring our system to multiprocess (think during the opponent's turn, as it were), do we need to track the turn since it is no longer implicit in the state of the system? This question, in some sense, answers immediately our title question in the affirmative, though we still haven't really answered the true question: do we need to reason about the initiative (and/or the turn)?

I suspect, in this case, the answer is yes. Our actions to initiate an interruption are different from our actions to initiate a response. These may vary from raising our voice to override the speakers current speech, to, more simply, putting something on the display (when available) that indicates the system is no longer listening to the human. However, this is not enough to require symbolic manipulation of initiative. If we can simply use patterns of speech acts and contexts to determine if we need to interrupt, we can deal with the problem operationally.

But here we run into a difficulty; even given we can recognize from a prefix of the input stream what topic the speaker is addressing, we need to somehow encode the "interruptablity" of the topic with respect to the specific speech act, the discourse state, and interaction with the problem solver (which is doing part of the topic analysis by way of plan recognition). Here my concern is that deriving this matrix of potential patterns is a much harder problem than simply reasoning more directly over the meta-representation of current topic, and interruption issue (e.g., we may not want to interrupt if the speaker is already addressing the interruption topic). It may be possible, however, to construct this matrix at some "compile time", that is, the programmer deals with the issue as a matter of reasoning over the interruption reason and input prefix topic, while some compiler turns this into specific code for each of the extant speech act patterns.

In the above cases we've looked at discourse reasons for needing to track initiative; there still remains the other main issue—domain reasons for needing to track initiative. That is, given we do not know our domain a priori, does the discourse system need to reason about what it knows (or is accessible to it) without the assistance of the human vs. those things it needs to request from the human? Or indeed, does the system need to reason about the knowledge of the human, given we have a good user model, and we may be taking on one of several roles in a conversation given the level of expertise of the system "user"?

An example may suffice to clarify this issue. If the system is some sort of trip planner, as in TRAINS, then the amount of information it can expect to elicit from the user without prompting differs between known experts and known novices. That is, we might want the system to take more initiative with a novice user, prompting for specific actions and question answers (even to get the result "I don't care"), because the novice may not be aware of the possible constraint space over which problems can be formulated. Experts on the other hand can be expected to supply all relevant constraints (they are aware of) when they formulate the problem, and we can then reasonably default the rest without needing to confirm them (since the expert might be considered to know what those defaults are). Of course, in a different domain, we might flip this reasoning—allowing the novice to default without prompting, and requiring the expert to state the problem in detail because it is expected to be more complex. The issue here is that given different notions of what the user is likely to know (in a domain), there are indeed different actions we may wish to take in terms of guiding a conversation and determining when we may want to try to throw initiative over to the user rather than trying to jump ahead and solve the problem, relying on the user to make any corrections to the plan as needed (the TRAINS approach). Note that these actions, per se, are not what require us to reason about initiative, but rather our model of dialogue participant's knowledge that brings us to doing it.

Now, our expectations of participant's knowledge may not have to be in great detail. This process might just involve having a set of general characteristics (e.g., novice; beginner; layman; journeyman; master) and a corresponding level of prompting, interruption, etc. we are willing to generate with respect to the specific user "classes". We might explicitly represent initiative, but not really do anything "interesting" (or at least complicated) with it.

It's a reasonable assumption that a large number of simple domains can be handled by analyzing these interaction features, and I suspect the key attribute a domain or implementation shall have when this heuristic will fail, is letting the system have available to it multiple sources of knowledge and actions (and these have different, or at least non-zero, costs associated with them). Then it begins to make sense that one needs to essentially do a utility analysis to determine if it's more useful to turn the prob-

lem back to the other participant, or expend additional effort by using another source that may or may not be able to provide a solution. This is my take on the fundamental insight in (Guinn, 1996). In other words, the other participant becomes an information source that also has a cost and utility associated with it. We would need to compare the cost and expected utility of trying to tease that knowledge out of our user with the cost and utility of accessing another available knowledge source. This would be influenced by our user model, i.e. dealing with a novice is likely to have a lower utility and higher cost than an expert. The trick is to be able to classify expertise on potentially multiple subjects from simple conversation.

Conclusion

There can be computational and representational cost to dealing with initiative other than operationally. Certainly a wide variety of domains do not require such a facility, but it is clear that certain features of the domain (e.g. multiple non-zero cost knowledge sources) or the specifics of the desired interaction with the user (e.g., the ability to interrupt the user) may require dealing with tracking initiative and reasoning about it in an explicit symbolic manner.

I believe it is important to know for a given domain and interaction if this facility will be needed, and this paper has provided, I hope, a step in this direction.

I think the next step might be to analyze a number of interactions that contain interruptions, and other forms of seizing the initiative, and classify the type of interruption, for instance, answering what are the general classes of interruption and initiative seizure. Other areas that need further investigation include how one determines expertise in an area from interaction features, and dealing with multiple knowledge sources that have different costs. Last, we need to answer the question of how much can be compiled in from a given symbolic representation used by an application programmer, into one that has no explicit representation at run-time.

Acknowledgments

Thanks to Alex Rudnicky, Chung Hee Hwang, and the reviewers for helpful comments on drafts of this paper. Thanks also to Kathy Hoemeke for editing and presentation assistance. Last I'd like to thank the MCC shareholders and associates who fund the I^{3}S project, making this paper possible.

References

Allen, J. F.; Miller, B. W.; Ringger, E. K.; and Sikorski, T. 1996. A Robust System for Natural Dialogue. in Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96), 62-70. Santa Cruz, Calif.: Morgan Kaufmann Publishers

Allen, J. F., and Perrault, C. R. 1980. Analyzing Intention in Utterances. *Artificial Intelligence* 15(3):143-178

Baekgaard, A. 1996. Dialogue Management in a Generic Dialogue System. in Proceedings of Eleventh Twente Workshop in Language Technology, (CD-ROM).

Carberry, S. 1990. *Plan Recognition in Natural Language Dialogue.* Cambridge, Mass.: MIT Press.

Cohen, P. R., and Levesque, H. J. 1990. Intention is Choice with Commitment. *Artificial Intelligence* 42:213-261

Ferguson, G. and Allen, J. F. 1993. Cooperative Plan Reasoning for Dialogue Systems. AAAI Fall Symposium on Human-Computer Collaboration, Raleigh NC.

Ferguson, G. and Allen, J. F. 1994. Arguing About Plans: Plan Representation and Reasoning for Mixed-Initiative Planning. in Proceedings, Second International Conference on AI Planning Systems. Chicago, IL.

Ferguson, G. M.; Allen, J. F.; Miller, B. W.; Ringger, E. K. 1996. The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant. TRAINS Technical Note, 96-5, Department of Computer Science, University of Rochester.

Grice, H. P. 1957. Meaning. *Philosophy Review* 66:377-388

Guinn, C. I. 1996. Mechanisms for Mixed-Initiative Human-Computer Collaborative Discourse. in Proceedings 34th Annual Meeting of the Association for Computational Linguistics (ACL-96), 278-285. Santa Cruz, Calif.: Morgan Kaufmann Publishers

Litman, D., and Allen, J. F. 1987. A Plan Recognition Model for Subdialogues in Conversation. *Cognitive Science* 11(2):163-200

Miller, B. W. 1997. The Lymphocyte Pattern-matching Engine. Technical Report, Department of Computer Science, University of Rochester. Forthcoming.

Miller, B. W. 1997a. The TRAINS-96 Discourse Manager. TRAINS Technical Note, Department of Computer Science, University of Rochester. Forthcoming.

Perrault, C. R., and Allen, J. F. 1980. A Plan-based Analysis of Indirect Speech Acts. *American Journal of Computational Linguistics* 6(3):167-182.

Pollack, M. E. 1992. The Uses of Plans. *Artificial Intelligence* 57:43-68

Rudnicky, A. I. 1996. MCC I3S Intuitive Interfaces for Information Systems: Goals and Objectives. (On-line) http://www.mcc.com/projects/I3S/I3S_Overview/

Sample Dialogue

This is a transcript of a conversation between two humans in the TRAINS-93 domain (which includes boxcars and cargoes). A human manager tries to develop a plan for delivering cargoes to cities, aided by a planning system, here played by a human in a Wizard of Oz scenario. It is reproduced from (Ferguson & Allen, 1993).

M: We have to ship a boxcar of oranges to Bath by 8 AM and it is now midnight.
S: Okay.
M: Okay. Um... all right. So, there are two boxcars at ... Bath and one at ... Dansville and ... [4 sec]
S: And there's ...
M: [interrupts] Wait. I've forgotten where the oranges are. Where are the oranges?
S: The oranges are in the warehouse at Corning.
M: Okay. So we need to get a boxcar to ... Corning.
S: Right.
M: All right. So why don't we take one of the ones from Bath?
S: Okay.
M: So...
S: [interrupts] We need ... Okay, which engine do you want to bring it?
M: Oh. How about ... Well, Let's see. What's shorter the distance between Avon and Bath or Elmira? It looks like it's shorter from Elmira to Corning, so why don't you send E2 to Corning?
S: Okay. [4sec]
M: In fact ... What did I say? Did I say ... Did I send a boxcar from Bath to Corning? It also looks like it's shorter from Dansville to Corning, so why don't you send that boxcar to ... Corning ...
S: [interrupts] Okay, with ...
M: [interrupts] instead ...
S: [interrupts] with engine E2.
M: Yes.
S: Okay, so you want ...
M: [interrupts] Oh wait. Okay, I ... All right, I misunderstood. So you have to have an engine in order to move a boxcar, right?
S: Right. [3sec]
M: What's ... Is it shorter to move an engine from Elmira? [3sec] Can you figure things out for me? Can you figure out what's the shortest? [3sec] What would be faster: to send an engine from Elmira to ... one of the boxcars or from Avon?
S: Well there's a boxcar already at Elmira [3sec] and you mean to go to Corning.
M: Yes. All right. I didn't see that boxcar. Okay. Great. Send E2 with the boxcar from Elmira to Corning.
S: Okay. [2sec] Okay. We can do that.
M: All right. And [3sec] once it ... gets there [2sec] What am I allowed to do? Okay. I can make a whole plan at this point. Right?
S: Yeah.
M: Okay. Once it gets there ... have it fill up with oranges and go straight to Bath. [2sec]
S: Okay. Then we get to Bath at ... 5 AM.
M: Great.
S: So we're done.
M: Okay.