A Preliminary Discussion of Fluents for Knowledge Representation in a Meta-Reasoner for Enterprise Information Systems

James D. Jones[1], Susanna Badiola[2] and Daley Seeker[3]

[1]Dept of Computer Science at Angelo State University, Box 10909, San Angelo, TX, 76909,USA james_d_jones@hotmail.com
[2]Dept of Philosophy at Angelo State University, San Angelo, TX, 76909, USA
[3]Dept of Accounting at Angelo State University, San Angelo, TX, 76909, USA

Abstract. It is a reasonable proposition to suggest that a high level, meta-reasoner for Enterprise Information Systems should exist. Such a reasoner could act as a monitor, manager, planner, and trouble shooter for the information resources of an organization. The lead author has built such a reasoner, which is now in its nascent stages. The software is based upon a logic programming paradigm, and employs powerful reasoning formalisms recently developed within the logic programming community. One of those formalisms is a theory of actions. A key component of an action theory is the ability to reason with respect to time. Another key feature of an action theory is the ability to reason about the value of variables that take on different values determined by the context (i.e., time). These issues (time, and the values of variables with respect to time) taken together comprise what is known in the field as *situation calculus*. The variables that vary with respect to time are called *fluents*. This paper discusses the use of fluents in this meta-reasoner. We will suggest fluents and assertions about what fluents that could be, or should be used (or made). This paper will also discuss the relevance of these fluents and assertions.

Keywords: *Meta-reasoning, Situation calculus, Logic programming, Artificial intelligence*

1. INTRODUCTION

This is a very narrowly focused paper on some issues related to knowledge representation and reasoning that a meta-reasoner for Enterprise Information Systems (EIS) may employ. Meta-reasoning (aka, meta-cognition) is a very recently re-emerging field of interest. Meta-reasoning "is any such strategy that involves the monitoring, modeling, and control of cognition" [1]. An excellent, short overview of this field can be found in [1].

The primary author of the present paper has designed an architecture and written programs employing this architecture that perform meta-reasoning for EIS [2, 3]. The programs are written in a logic programming framework (specifically, Answer Set Semantics [4-6]), and employ several recent powerful techniques for reasoning.

Please use the following format when citing this chapter:

Jones, J. D., Badiola, S., Seeker, D., 2007, in IFIP International Federation for Information Processing, Volume 255, Research and Practical Issues of Enterprise Information Systems II Volume 2, eds. L. Xu, Tjoa A., Chaudhry S. (Boston: Springer), pp. 1029-1035.

Some of those techniques include default reasoning with exceptions, an action theory, and negation as failure. A key feature of any logic program that has a semantics for negation as failure is that such a program possesses the power of introspection. Introspection is the ability to inspect one's own beliefs or knowledge. That is, it is the ability to know what one knows, and to know what one does not know. Not only can introspection provide the ability to reason about one's own beliefs, but it can provide the basis for additional inferences. This is particularly true in the case of default reasoning. We suggest that introspection is a valuable part of meta-reasoning.

A key formalism that these programs employ is a theory of actions [7, 8]. This action theory is built upon situation calculus, which is used to represent and reason with respect to time. Actions have consequences. Facts become true or false as a direct result of those actions. These "facts" are called *fluents*. A fluent is a time-varying variable. In traditional programming, variables are scalar[1] That is, they possess only one value at a time. The current value overwrites the previous value. By contrast, a fluent's value is with respect to time. We can simultaneously represent multiple values for the same fluent. For example, let x be a fluent representing the president of the United States. What is the value of x? It depends upon the time context about which the question is being asked. The value of x with respect to 2007 is George W. Bush. The value of x with respect to 1997 is Bill Clinton. Simultaneously, x instantiates to these values.

Fluents are important, because in a formalism that accounts for time and actions, fluents embody the knowledge of a system. Ultimately, all queries are reduced to a query about the value of a fluent (or the values of a group of fluents) with respect to certain time values. In this kind of a reasoning system, fluents reign supreme. Therefore, it is incumbent upon us to understand what fluents are, and to develop a taxonomy or a strategy of what fluents we need. This paper is an initial attempt in that direction.

2. POSSIBLE FLUENTS

If there was a meta-reasoner for EIS, what sorts of things might one want to reason about? The answer to that question is finitely large. For the sake of greatly simplifying this discussion, suppose we have one component system s which communicates one message m with our meta-reasoner. (The fact that this system communicates with the reasoner is not important. What is important is that the reasoner must have information in order to perform reasoning. We assume here without loss of generality that the source of that information is via message passing from a component system.) m could be any message. To give intuitive meaning, let's say that m is the message "the books are out of balance". This is a phrase used by accountants to say that the record keeping is not correct. In the following, we will examine several "aspects" about one single transmission. Each "aspect" will itself be a fluent. We will also consider assertions about the values of those fluents.

The message m is a fluent. This is represented as

[1] Even arrays (or vectors) are considered scalar.

fluent(i,m)[2].

This formula has two parameters (terms). The first parameter *i* identifies the type of fluent. There are two values for the type of fluent: *i* means the fluent is inertial, and *n* means the fluent is not inertial. The second parameter *m* identifies the fluent itself. A fluent is inertial if its truth value persists from one time period to the next, until something (an action) explicitly changes that truth value. A fluent is not inertial if it is true only during the time period in which an action is affecting it. Later examples will demonstrate the difference between these two types of fluents.

In our example here, *m*, which means "the books are out of balance" is an inertial fluent. This fluent (or proposition) can be true, false, or unknown. Assume that the truth value of this fluent is "true". Since the fluent is inertial, it will remain true that the books are out of balance until something causes it to change its truth value. An example of something that could cause it to change its truth value would be the action of making a journal entry to balance the books. Another example of something which could change the truth value of this fluent is that the error causing the imbalance could be discovered.

What causes this fluent to have any meaning at all is an assertion as to whether the fluent is true or not. It is the collection of these assertions which form our knowledge base, or our set of beliefs. The general form for such an assertion is

holds(Fluent, Time, Truth_vale).

Hence, *holds(m,1005,0)* means that fluent *m* is *false* at time *1005*. (To relate this back to our example, "it is false that the books are out of balance at time 1005".) Similarly, *holds(m, 1005, 1)* means that fluent *m* is *true* at time *1005*. A fluent does not ever have to asserted as either true or false; it could be "unknown". There is not a separate designation for "unknown". Rather, both formulae *holds(m,1005,0)* and *holds(m, 1005, 1)* would be missing from our set of beliefs.

We have just seen that a potential message *m* is a fluent. We have seen how that fluent can be asserted as true, or false (or left unspecified, in which case it is neither). We have seen that assertions can be made about the truth or the falsity of fluents. We see from our representation schema that the same fluent can take on different values at different times.

When message *m* becomes communicated, several other fluents will come into play. One such fluent could be the fact that the message is being uttered (a generic word meaning "communicated".) The *act* of uttering is an action. The *fact* (or *state*)

---

[2] We want to limit our discussion to only one message. However, if we had more messages, a more robust representation would be

fluent(i,X) :- message(X).

This is a rule which states that if *X* is a message, *X* is also a fluent. We could then enumerate messages, such as: message(m), message(n), message(o), etc. We could also identify more things as fluents, such as fluent(i,X) :- light_on(X) and fluent(i,X) :- running_water(X).

that it is being uttered is a fluent. The content of the message is also a fluent. The fact that the message was uttered at a specific moment in the past is a fluent.

Actions affect fluents; actions cause the values of fluents to be changed.) The fact that a message is being uttered is a non-inertial fluent, and is true only during the time in which it is being uttered[3]. This is a more complex formula, and we will look at the second parameter first. This is represented as

uttered(Message ,Entity, Truth_value)

Assuming the values of our example, and putting it all together yields the following

fluent(n,uttered(m,s,1))[4]

This represents that entity *s* "uttered" message *m* with some truth value. That is, entity *s* says that message *m* is true (or conversely, false). In particular, system *s* believes that the books are out of balance[5]. Notice that this fluent is not inertial. Uttering a message at time *T* does not imply uttering that same message at time *T*+1. Of course, that same message could be uttered again at time *T*+1, but that would be an additional action. So far, we have just defined the fluent. For the message to have actually been communicated, we would need an action to have happened which had the consequence an assertion similar to the following.

holds(uttered(m,s,1), 1000, 1)

This states that it is *true* that at time *1000*, system *s* uttered the *true* message "the books are out of balance". It is interesting to note that the above formula has two truth values: the truth of the fluent overall, and the truth of the message imbedded within the fluent. Consider the following contrasting formula.

holds(uttered(m,s,0), 1000, 1)

---

[3] Without loss of generality, we assume that actions are instantaneous. That is, they happen within one time period. In reality, depending upon the granularity of the time, actions can have duration. For example, a person can start to fall at one time period, and the act of falling is completed several time periods away. (This is particularly true if the granularity of time is milliseconds.)

[4] A more general representation would be:

fluent(n,uttered(M,S,Truth)) :- message(M), entity(S), truth_value(Truth).

[5] Actually, system *s* merely communicated that "the books are out of balance" is true. We do not know what the beliefs of *s* are. Nor do we know the truth of the message. What if *s* made an error in communication (believing one thing, and communicating another?) What if *s* was deceitful? What if *s* was genuinely wrong? That is, what if the books are NOT out of balance, and *s* made a wrong determination claiming that they are out of balance. We could expand our fabrication to include fluents to address these possibilities as well.

This states that it is *true* that at time *1000*, system *s* uttered the *false* message "the books are out of balance". Similar assertions could be made that neither utterance was made. For example,

holds(uttered(m,s,1), 1000, 0)

states that it is false that at time 1000, system *s* uttered the *true* message "the books are out of balance".

In the above, message *m* was *uttered* by system *s*. *s* is an entity outside of our meta-reasoner. The fact that *s* utters the message does not implicate the meta-reasoners involvement in the message. The corresponding piece of the picture that is missing is the *reception* of that message. The fluent *received* is structurally the same as the fluent *uttered*, and is represented with the following.

fluent(n,received(m,s,1))

The above formula just defines the fluent. The following formula is needed to represent the fact that the message was indeed received. All of the corresponding discussion for *uttered* regarding the meaning, and the various truth values is the same for *received.*

holds(received(m,s,1), 1001, 1)

Our intuition leads us to believe that receiving a message is not enough. It needs to be evaluated, or understood. We will not deal with the meaning of a message here. There could be two strategies for arriving at the meaning of a message. One strategy would be that there is a predetermined set of messages which can be communicated. That is, with the exception of plugging in variable pieces of information, the message is somewhat static. In this case, the meaning could be easily ascertained.

The other strategy would be far more dynamic. There would not necessarily be a limit on what the message could be. In this case, the meaning cannot be anticipated in advance. For this situation, we would need some sort of natural language processing component to arrive at a meaning of the message. This could introduce the need for a wide range of fluents to represent the meaning (actually, the meta-reasoner's interpretation) of the message. There would also be the need for some sort of sentinel to signify that the message has been processed and understood.

Such a formula may be complex. Looking ahead to the possibility of numerous messages, we need to identify which message is understood. Further, assuming the same message could be issued multiple times, we need to identify which occurrence of the message is understood. The following representation will achieve this.

fluent(i,understood(received(m,s,1,),1001))[6]

[6] We want to understand only messages that have been communicated. We do not want to understand messages which have not been communicated. Hence, the truth value which occurs immediately after the time period is dropped. In other words, if the truth value was included, and if that value was 0, the following would be the result.

This formula is a fluent representing a specific message: the message *m* which was communicated by entity *s* at time 1001, which *s* believed to be true. In order to say that we understand this message, we need the following assertion.

holds(*understood*(*received*(*m*,*s*,*1*,),1001), 1005, 1)

This formula says that it is true that at time period 1005, there is a message which we understood. That message is a specific occurrence of the message "the books are out of balance". It is the occurrence which happened at time 1001, in which *s* communicated it as a true statement.

3. SUMMARY

The primary author of this paper has developed a meta-reasoner for EIS. This system employs certain reasoning techniques and knowledge representation techniques that are very robust. In this paper, we have very narrowly focused upon one aspect of the knowledge representation. This knowledge representation would be suitable for any meta-reasoner. We have focused upon fluents, and the corresponding assertions about those fluents. Further, rather than discussing potentially very many fluents, we have narrowly focused upon suggested fluents involving the communication of a single message.

It is hoped that the reader can discern that there are very many things about which we can successfully reason. In this paper, we demonstrated several fluents, and hinted at many more (especially in the footnotes.) All of these fluents were in the context of one system, communicating one message, communicating in only one fashion ("uttering"), and our meta-reasoner adopting a position of credulity. Imagine how much more complex this would be if we had many interconnected systems, communicating very many messages, performing very many additional actions besides communicating, communicating in many different ways (other than just "uttering"), and received in a multitude of ways by a skeptical meta-reasoner trying to establish "the truth". Even with these additional considerations, we have not even begun to scratch the surface of the available reasoning power. (For instance, this becomes even more complicated in a mature system that employs a vast array of defaults.) The technology to perform all this reasoning exists. Forgive the colloquialism, but we can reason about the "ifs, ands, and buts". It is our opinion that logic programming (in particular, the answer set semantics) coupled with a well developed theory of actions, provides a very powerful and robust reasoning platform. It is also our opinion that this platform is very suitable for meta-reasoning.

---

fluent(*n*,*understood*(*received*(*m*,*s*,*1*,),1001),0)

This is not an assertion, but a fluent. This would describe an occurrence of a message which was not communicated. Therefore, we do not need to represent that an uncommunicated message is understood. Therefore, we drop the final argument, since the only messages understood are those which are communicated, which are those which have a 1 for the final argument.

REFERENCES

1. M. Anderson and T. Oates, A Review of Recent Research in Metareasoning and Metalearning, *AI Magazine*. Volume 28, Number 1, pp.7-16, (2007).
2. J.D. Jones, S. Reames, and G. Pandzik, Skeleton of a Supervisor for Enterprise Information Systems, in *Proc. of International Federation for Information Processing (IFIP) International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS)* (Vienna, Austria, 2006).
3. J.D. Jones, Toward A High-Level Reasoning Module for Enterprise Information Systems, in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC 2006)* (Taipei, Taiwan, 2006).
4. M. Gelfond and V. Lifschitz, The Stable Model Semantics for Logic Programming, in *Proc. of 5th Intl Conference on Logic Programming* (1988).
5. M. Gelfond, Representing Knowledge in A-Prolog, Computational Logic: Logic Programming and Beyond, *Essays in Honour of Robert A. Kowalski*. Volume 2408, Part II, pp.413-451, (2002).
6. C. Baral, *Knowledge Representation, Reasoning, and Declarative Problem Solving* (Cambridge: 2003).
7. C. Baral and M. Gelfond, A. Provetti, Reasoning about actions: laws, observations and hypotheses, *Journal of logic programming*. Volume 31, pp.201-244, (1997).
8. C. Baral and M. Gelfond, Reasoning about effects of concurrent actions, *Journal of logic programming*. Volume 31, Number 1-3, pp.85-118, (1997).