Representation and Inference in the Consul System1

William Mark

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291

Abstract

Users of interactive systems need a single cooperative interface for all of the services in their environment. The interface must behave in a consistent manner in understanding natural user requests and in providing explanation and help as required. The Consul system is designed to provide such an interface. Its natural interaction capability is achieved by mapping between detailed descriptions of users and systems in order to translate requests and provide explanations An interactive system of this Kind would be infeasible if the onus of constructing the knowledge base and inference techniques were placed on the individual service builders in Consul, service-dependent information is incorporated into the Knowledge base by semi-automatic acquisition, resulting in incorporation of the new Knowledge into the system's built-in abstract framework. This incorporation allows the service-dependent data to appropriately influence Consul's knowledge based mapping processes. The current Consul prototype demonstates natural request handling and explanation for a mail service

1. Introduction

An interactive system should provide a natural interface for its users and behave in a consistent manner across a wide range of functional services. The Consul system attempts to achieve these goals by providing natural language input and explanation facilities for users of interactive services such as electronic mail, automated appointment calendar, and document preparation.

Users' needs change constantly, and vary greatly from one environment to another. The needs of any particular group of users must be satisfied by service builders familiar with those needs. However, the task of building an interactive service that provides a natural interface and remains consistent with other services is too great to be left as a burden for individual service builders. What is needed is a system that has already solved the basic problems of providing a natural, consistent interface, and into which new services (or changes to existing services) can be incorporated with relative ease.

In Consul, a single knowledge base framework consistently represents not only the objects and actions of users and interactive systems, but also the inferential capability to map between the user and system worlds. This mapping mechanism is the foundation of Consuls natural language understanding and explanation techniques. The knowledge base is adapted to the domain of any particular service (e.g., electronic mail) by a system directed acquisition process which results in the association of built-in concepts with their appropriate counterparts in the service domain.

1.1. Understanding user requests

If the user is allowed to express himself in a "natural" manner, most of his requests will not correspond directly to input forms for the appropriate service actions. Some aspect of the interactive system must therefore have the capability of mapping user requests into appropriate system action invocations. For example, consider the following user request to a mail service2

Forward this message to Jones.

This request is simple enough from the user s point of view. However, from Consuls point of view, its interpretation depends on the circumstances of the request and the details of the particular service involved. In most mail services, if *this message* referred to a previously received message and if *Jones* were a valid user address, interpretation of the request would mean mapping the parse of the request into an invocation of the message forwarding function with the appropriate parameters. However, if *this message* referred to a just composed message, many (though not all) mail services would consider the request to be in error and would require restatement. As we will see below, the situation in a SIGMA-like mail service is somewhat more complex: "messages" are never actually forwarded (or sent in any form); only citations of messages are sent between users-the actual message remains in a central database. Therefore, the users request can never be taken literally, but must be redescribed in system terms. Even then the request is either ambiguous (the user must either "forward for action" or "forward for information") or in error (if *this message* is a "draft message").

Thus, understanding user requests requires mapping the input forms into the conceptual framework of the particular service that must fulfill the request. This process requires some understanding of both the context of the users request and the characteristics of the service.

1.2. Explaining the system

When the user requires information about some aspect of the system, he must be able to ask about it in a natural manner and receive a response that he can understand. Consul must therefore be ready to recognize and respond to "help requests" (e.g.. *What has to be in a message? What does forwarding do?).* This problem of "explanation" has several parts:

1 This research is supported by me Defense Advanced Research Projects Agency under Contract No DAHC15 72 C0308. ARPA Order No 2223 Views and conclusions contained in this paper are the author s and should not be interpreted as representing the official opinion or policy of DARPA. the U S Government or any person or agency connected with them.

All examples in this paper are baaed on a SIGMA-like service ([SIGMA 79]) currently being implemented under Consul

Consul must first decide whether a user utterance is a request for help or a request for action This is by no means straightforward, and ultimately relies on an understanding of the user's intent in making the request (as in *Could you send this message to Smith?).*

Even when a request for help is recognized as such, the system must understand the user s statement of what he wants to know in terms of those aspects of the system that are relevant to answering his question. For example, although *Why was message 17 deleted?* is a perfectly reasonable user help request, it makes no sense at all in the world of the SIGMA-like service. Messages do not have numbers associated with them nor are they ever deleted. Consul must map the user's question into a question that is appropriate from the mail system viewpoint-a question that takes into account that messages are accessed only through their citations, that citations have numbers by virtue of being in a user's mailbox, and that the relevant deletion process is the one that removes particular citations from a particular user's mailbox.

Next, the system must discover an appropriate answer to the reformulated question, requiring information gathering (as in *What has to be in a message?),* cause/effect reasoning (as in *Why was message 17 deleted?), or* other special processing depending on the type of question asked.

The answer, which is in system terms, must then be mapped back into terms that the user can understand in order to create a true response to the original question.

An additional explanation problem occurs because the system cannot always do what the user wants. Thus, although the intent of a user request may be that the system should perform some action, the response of the system may have to be an explanation of why it cannot meet this intent. For example, as discussed above, if *this message* in the request *Forward this message to Jones* refers to a just composed message, the system cannot do what the user is asking. Instead, it must respond with an explanation of why it cannot forward the message to Jones, and suggest a restatement or alternative course of action to the user

1.3. Acquiring domain-dependent knowledge

All of the above discussion assumes a carefully built, detailed model of what a particular interactive service can and cannot do This raises the question of how such a model is built into the Consul system. Builders of interactive services know about the characteristics of their service--not the higher level models and mapping knowledge required by Consul for cooperative behavior. Therefore. Consul must be responsible for the task of incorporating the details of a particular service into its knowledge base. It must figure out (or at least lead the service builder to tell it) how the operations and objects of a service can be seen in terms of its higher level model.

It is very likely that the relationship between specific service characteristics and Consul's built-in knowledge base will go beyond straightforward instantiation: some service characteristics will not instantiate *anything* in the knowledge base. For example. Consul's model of transfer operations assumes that the argument of the operation (say. a message) is what ends up being delivered to the specified destination. However, SIGMA send operations send citations of the message, not the message itself. SIGMA send operations thus do not directly instantiate Consul's model of transfer operations. But Consul will not be able to bring to bear its knowledge about transferring (e.g.. how explanations and user requests about sending map into transfer operations) unless it can relate the SIGMA send operations to its built-in model of transferring. Consul's acquisition task therefore comprises not only instantiation of its knowledge base, but also the construction of more complex mappings between what it knows and what the service builder tells it.

2. Approach

The main feature of Consul is its knowledge base of descriptions of users services, and interactive systems-Relationships between descriptions--mappings--are represented in terms of inference rules, which are "applied" to transform one kind of description into another. The handling of a natural language request from the user is treated as the reformulation (via the application of inference rules) of a description in the conceptual framework of the user as a description in the conceptual framework of the service model. This service description represents an actual function that can be executed in order to satisfy the user s request. Similarly, help and explanation are treated as reformulation of service descriptions in terms of user descriptions.

Consistent treatment of services is achieved by pre-building a service-independent model of interactive systems into Consul, and then acquiring knowledge about particular services in terms of that model. Acquisition is accomplished via a stylized dialogue with the service builder, directed by Consul based on the service builder's implementation of his service in terms of a programming formalism specially designed for this task.

2.1. The Knowledge Base

The basic structure of the Consul system is shown in Figure 1. The knowledge base, implemented in KLONE [Brachman 78]. is central to all system activities-parsing, explanation, mapping, execution, and acquisition. It contains several kinds of information:

<u>Knowledge</u> of Users a representation of the relation between English language constructs (e.g., "send request") and the actions and objects of the user s world that (he thinks) have some correspondence in the system world (e.g., "send", "message"). For example, since users have the concept of sending information from one user to another, the knowledge base includes User Send and User Message (see Figure 2). This representation is purely in terms of the user's

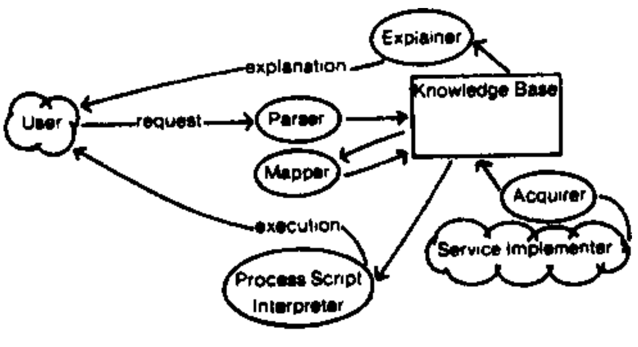

Figure 1: Consul

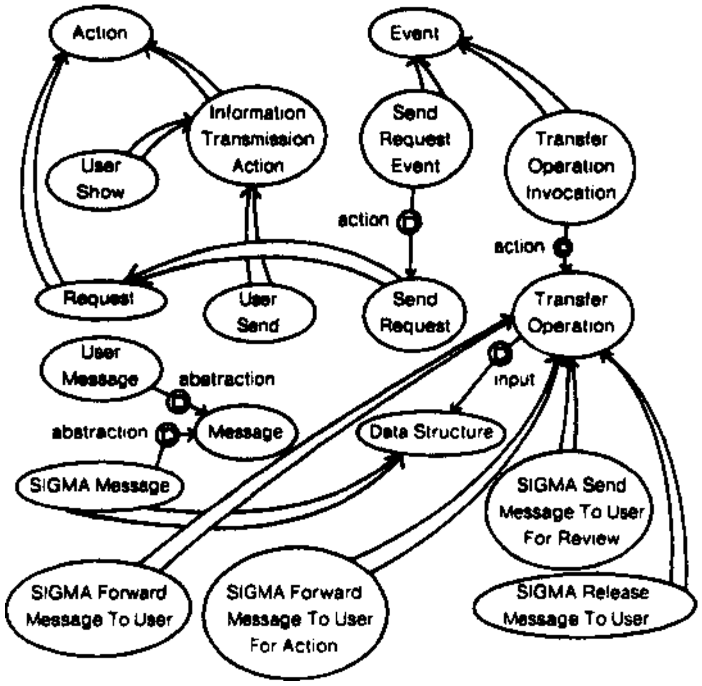
Action
Event
Information Transmission Action
Send Request Event
Transfer Operation Invocation
User Show
action
action
Request
User Send
Send Request
Transfer Operation
User Message
abstraction
input
abstraction
Message
Data Structure
SIGMA Message
SIGMA Send Message To User For Review
SIGMA Forward Message To User
SIGMA Forward Message To User For Action
SIGMA Release Message To User

Figure 2 Part of the Knowledge Base

conceptual structure-correspondence with the actions and objects of the system, if any. has yet to be established.

Systems knowledge: a service-independent representation of basic operations (e.g.. deletion, transfer, display) and the data structures these operations work on (files, tables, display lists, etc.). For example, any service will probably have a transfer operation of some sort. Consul's systems knowledge provides an organizational framework for structuring knowledge about transfer operations, based on the concept Transfer Operation (see Figure 2). The particular transfer operations of any service can be described in terms of that framework, and thus be "understood" by Consul (i.e.. seen in relation to the other things that Consul knows about).

Service knowledge: a particularization of systems knowledge to the actual operations and data structures of some interactive service that is implemented in Consul. For example, the model for a mail service would describe a variety of *specific* (actually executable) transfer operations for sending messages. SIGMA has several specific transfer operations for sending different kinds of message citations (see Figure 2).

Dynamic Environment: a model of system and user activities as events in time. i.e.. invocations of the actions defined in the user and service models. This event model serves as a dynamic environment for expressing the behavior of the user and the system. Thus Send Request Event and Transfer Operation Invocation in Figure 2 represent potential *occurrences* during an interactive session with Consul.

Figure 2 shows some of the instantiation relationships among Consul's descriptions of users, systems, and services. Consul must be able to map between these various descriptions in the knowledge base in order to understand requests, generate explanations, and acquire service specific information.

2.2. Mapping

In Consul, this mapping capability is provided by a rule-based inference process that can reformulate a given knowledge structure in terms of another. The rules are simple transformation relationships between modelled structures. When a given description instantiates a structure that is a rule condition, the mapper can reformulate the description in terms of the structure that is the rule conclusion. This process can be used to redescribe a wide class of incoming descriptions (user requests, explanation forms, and service specific information) in terms of Consul s built-in knowledge base[3].

This section shows the systems mapping activity on the request *Forward this message to Jones.* The first operation on any incoming request is parsing[4], i.e., classifying the request in the knowledge base in terms of Consul's model of what the user can express The new description is then redescribed by the mapping process until it can be seen to instantiate either a description of some service action (in which case the action is executed, thus satisfying the request) or some explanation form (in which case an explanation response is generated).

The parse of *Forward this message to Jones* is shown in Figure 3. The parser has described the request as a Request for the user action whose objective is Sending. The fact that the user has specified "forwarding" is represented by the intrinsic description of this Send Action as Resend The result of the action is that both the object (the message) and the intrinsic description arrive at Jones.

When this parse is classified in the knowledge base, it does not instantiate any service action or explanation form (i.e.. the system does not contain a description of the "Forward this message to Jones" action). However, as shown in Figure 4, the description does instantiate a (very general) knowledge structure that

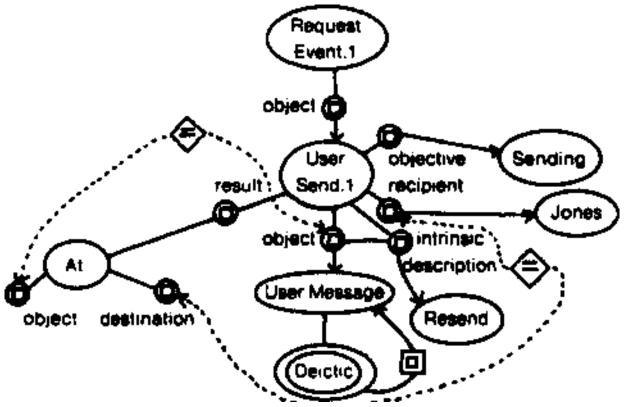

Figure 3: The parse of "Forward this message to Jones"

[3]The underlying philosphy and implementation of the rule application and redescription process is described in detail in (Mark 80]

[4]The PSIKLONE parser (Bobrow & Webber 80) has been integrated into the Consul system,

happens to be a rule condition. The rule simply says that a request for a user action with a particular result can be redescribed as's call to an operation whose effect is that result[5]. This does not imply that such an operation is really implemented in the system; it merely creates a description of a call to such an operation. In this case, rule application produces a call to an operation (Operation Invocation. 1) whose effect is "this message" and "resend" ending up at "Jones" (see Figure 4).

This redescription is then classified in the knowledge base (Figure 5). If, via this classification process, the new description is found to be a subconcept of a call to some actually executable operation, inference is complete-Operation Invocation. 1 can simply be passed on to the interpreter to produce the required execution results. Otherwise Consul will have to use additional rules to refine the description until it can be seen as some executable form.

As shown in Figure 5, Operation Invocation. 1 can be classified as a Transfer Operation Invocation, but not as any of the actually executable SIGMA send operation invocations This means that Consul will have to look for additional applicable rules However, in this case none of the structures instantiated by the current description happen to be rule conditions. In order to proceed, the mapper will have to change tactics.

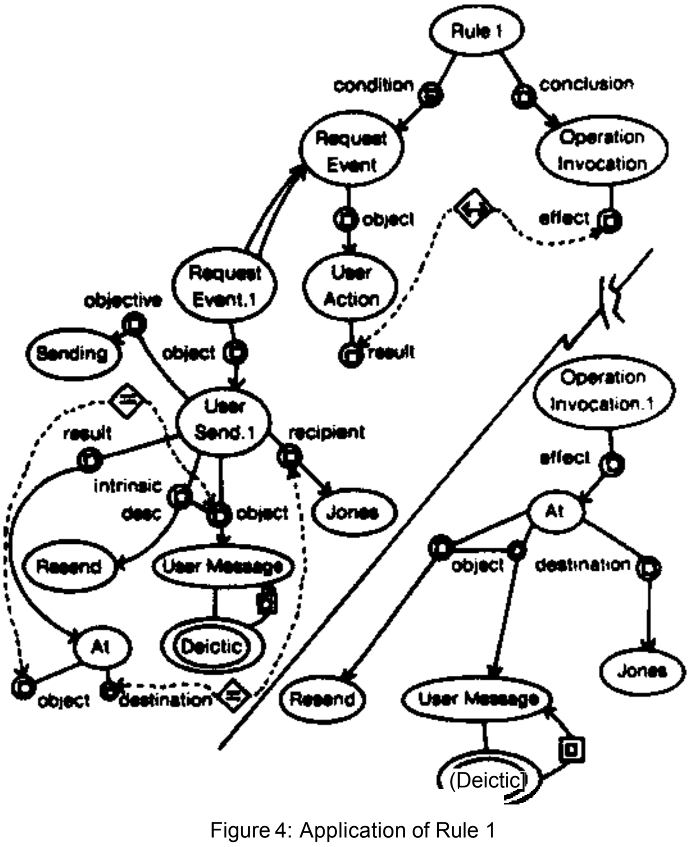

Figure 4: Application of Rule 1

H Consul had had a mora specific rule that applied to requests for send actions, that ruta condition would have been inatantieted. and that rule would have been applied instead

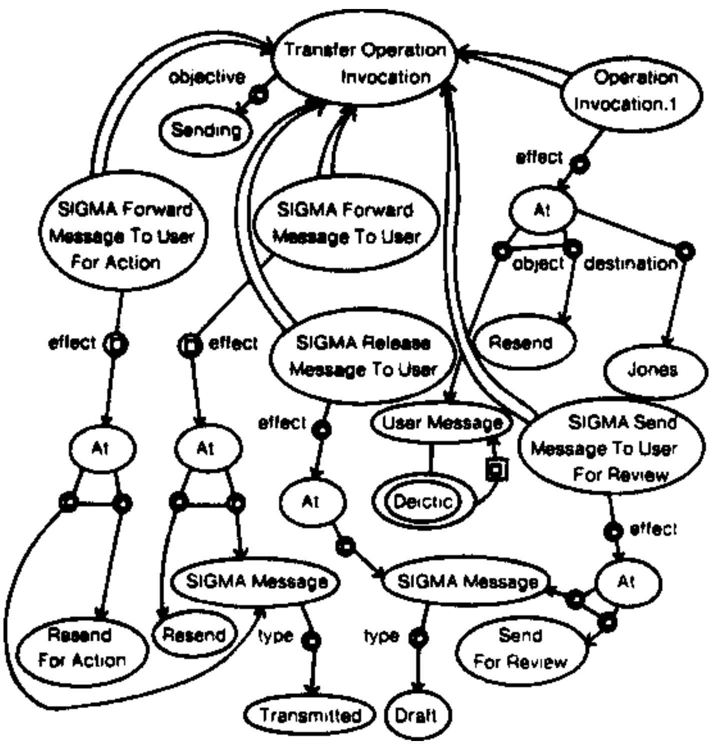

Figure 5: Classification in the Knowledge Base

When no additional rules are applicable, the system goes into a target-based reasoning mode The idea is to find rules that can redescribe parts of the description (rather than the whole thing at once, as before) until the whole description instantiates some "actionable entity"-executable function, explanation form, or other rule condition. Since the number of rules that can apply to parts of the description is potentially much larger than the number that can apply to the whole thing, target finding is used to determine whether redescribing the description part by part is worthwhile. Part-by-part redescription will be attempted only if the whole new description produced could be an instantiation of an actionable entity

Actionable targets for this redescription process are found by noting all parts of the current description that could possibly be redescribed by some rule (i.e., that instantiate some rule condition). Then, treating these parts as "wild cards" (concepts that instantiate anything), the classifier finds all of the actionable entities that the whole description instantiates; these are the targets. Once the targets have been found, the mapper knows that applying rules to parts of the description can produce an actionable entity. It can then apply the rules and see if the resulting new description actually instantiates a target actionable entity. If so, the mapping process either succeeds (if the entity is an executable function or explanation form) or continues (if the entity is a rule condition).

In this example, only one rule applies to a part of the current description. This rule, shown in Figure 6, attempts to resolve a deictic reference to an object (e.g.. *this message)* by looking for a similar object that appeared in a recent event. *Here* it is possible (depending on the actual resolution of the reference) that the rule will redescribe *this message* as a Transmitted message, thus making the current description a subconcept of the actually

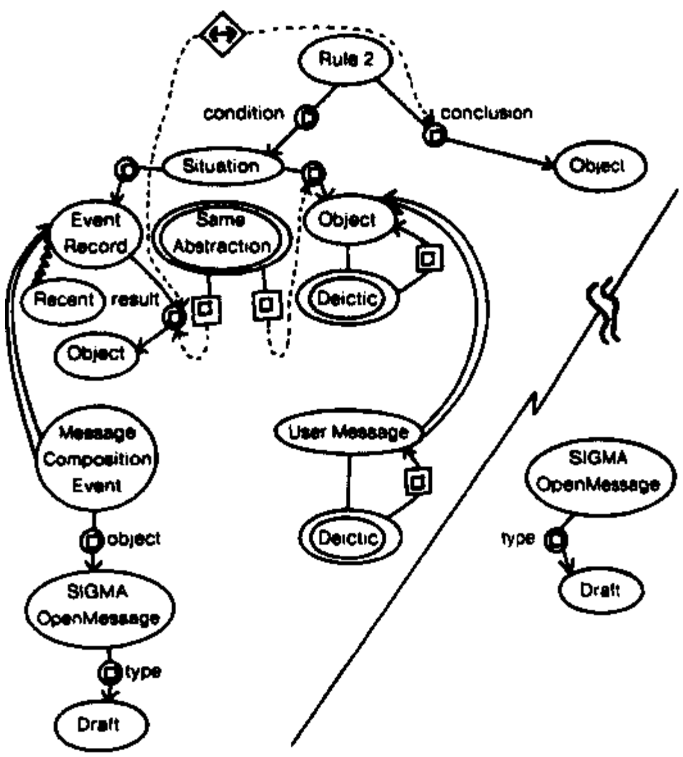

Figure 6: Resolving the Deictic Reference

executable SIGMA Forward Message To User shown in Figure 5. However, let us assume in this case that when Rule 2 is applied, *this message* turns out to be a Draft, as shown in Figure 6 SIGMA Forward Message To User is therefore not applicable, and target-based reasoning fails. The mapper must now conclude that the user's request cannot be satisfied, and that an explanation response must be generated instead. It creates an explanation form noting the structure it gave up on (Figure 7) and passes the problem on to the explainer.

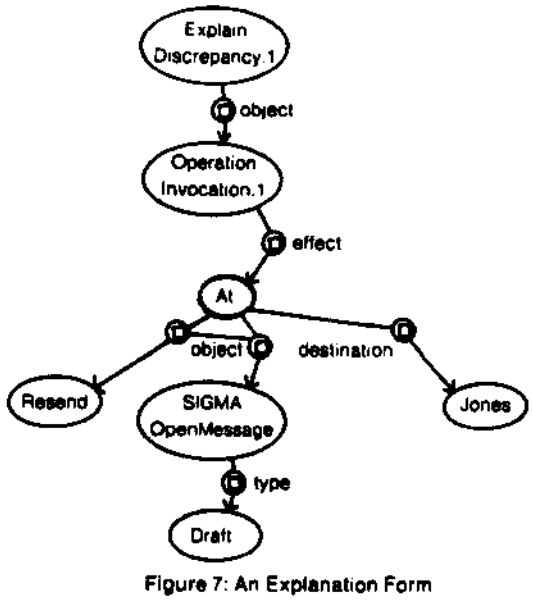

Figure 7: An Explanation Form

2.3. Explanation

Now that Consul has given up on the attempt to perform the user's request for action, it must explain why it cannot perform the action, and perhaps suggest alternatives to the user. This requires finding the service operation or operations that the user was probably trying to get Consul to perform, and explaining to the user in his own terms why his request did not cause their invocation. Then, since Consul assumes that the user is really making an effort to do something with the system, it tries to make reasonable suggestions for corrections of the original request that *win* make something happen. Of course, the system must make sure that the user knows what that something is and how it differs from what he was requesting.

The first step of the explainer is therefore to find a reasonable set of targets--service operations that the user could have been trying for. This target finding operation is somewhat different than the one used during target-based mapping. For mapping purposes the only reasonable targets are those that could be reached by rule application. The explainer assumes that no such targets are available (otherwise the user s request would have been satisfied). Instead, its notion of potential targets are those service operation invocations that have the same objective as the user action specified in the original request.

As shown in Figure 3, the user's request specified the User Send action, that is, the action whose objective is the abstract concept of Sending. The explainer assumes that any service operations (or their invocations) that share this objective are close enough to the intent of the users request to be considered potential targets. All service operations constructed as instances of the systems Transfer Operation will inherit the objective Sending. Figure 5 shows that four SIGMA operation invocations. Forward. Forward For Action, Release, and Send For Review, are instances of the systems Transfer Operation Invocation[6].

This initial list of targets is then examined to determine the differences between the targets and the user's requested action. A comparison of the desired operation invocation in Figure 7 with each of the potential targets (of which only a few relevant parts are shown in Figure 5-the full model of SIGMA's forwarding function is shown in Figure 9) reveals that Forward has the right intrinsic description but sends the wrong type of message. Release and Send For Review send the right type of message but have the wrong intrinsic description (Release has none at all), Forward For Action both sends the wrong type of message and has the wrong intrinsic description. Since Forward For Action happens to differ in *all* of the ways that any of these targets differ, it is pruned from the target list. This leaves only Forward, Release. and Send For Review to be used in the explanation.

But the explainer also notes that one of these targets. Forward. differs from the requested invocation in a different way than the other two. The specification *this message* in the user's request has been resolved into a description of a unique object in the world, i.e.. a particular message. The explainer will not suggest alternatives to the user's specification of unique objects (i.e.. it will not respond to the users request with "You can t forward this message, would you like to forward a different one?"). Therefore.

[6]The way that these service specific operations came to be so modelled is discussed in the following section

given that the user really wants to send *this message,* only two of the possible targets, Release and Send For Review, remain as viable alternatives.

All of the necessary information for the response has now been gathered. Comparison of the explanation form in Figure 7 with the target descriptions shows why the user's requested action could not be applied to his requested object: the indicated intrinsic description and message type are incompatible with respect to the target set (while the recipient Jones, for example, is compatible, and will not enter into the explanation). The response will therefore explain that the user's request for forwarding could not be satisfied because this operation cannot be applied to messages of type "draft". And, since there are other targets in the set that can still satisfy the user's objective for his requested object, the response will go on to suggest Release and Send For Review as alternatives.

Once the content of the response has been decided upon, the explainer must determine whether there are user concepts that correspond to the system concepts ("forwarding", "sending for review", "releasing", and "draft message") in the planned explanation. Those concepts that have a direct correspondence in the user model can simply be incorporated into the response (e.g.. since the user's concept of message and the systems concept of message share a common abstraction, Consul can make the direct translation). Those concepts that do not have a direct correspondence in the user world must be redescribed in terms of component parts that do have user equivalents (eg. "forwarding for action is like forwarding except that the user receives the note 'requires action' with the message"). We have already seen that the user has the concepts of "forwarding" and "message"; for simplicity, we will assume that the other system concepts to be explained also have user equivalents The explanation response resulting from the user request *Forward this message to Jones* is therefore

Draft messages cannot be forwarded. You can release this message or send it for review.

2.4. Acquisition

It should be clear from the preceding sectors that Consuls ability to understand user requests and produce appropriate responses relies crucially on proper classification of service dependent information in the knowledge base. This means that the service dependent information *must* be seen in relation to the built-in concept structures representing Consul's systems knowledge. Consul's reliance on precise service modelling would be in feasible if model construction were the responsibility of the individual service builder: the service builder is unaware of Consul's needs, and certainly unaware of the intricacies of its KL ONE model. Instead, Consul provides an acquisition mechanism to incorporate the builders description of his service into the Consul framework.

The service builder is never directly aware of the system knowledge base. He implements his service as a set of process script programs [Lingard 81]. The process script language is a programming formalism specially designed for the acquisition task. Programs consist of a procedure and some descriptive information about that procedure. The descriptive part is in the form of a small number of categories of information required by Consul in order to see how the function represented by the process script fits into its knowledge base. The process scripts, along with descriptions of service-dependent data structures, are acquired into the knowledge base to form the service model. The process script to implement the *SIGMA Forward* operation is shown in Figure 8

```
ProcessScript   SIGMAForward;
  Input              u:SIGMAUser;
  Output             none;
  DataAcceased       SiGMAOpenMessage. SIGMALoggedOnUser:
  Preconditions      SIGMAOpenMessageSV = true:
  SideEffects        none:
  Undo               none:
  Error Conditions   eNoMailBoxForRecipient,
    Call SIGMASend(u. "Forwarded"):
```

Figure 8: A Process Script

The process script is expressed completely in terms of the service builder s programming environment. It must therefore be translated into the appropriate KLONE representation, called SIGMA Forward Process Script in Figure 9. This description must then be incorporated into the knowledge base As I mentioned earlier, the service builder s program (embodied in *SIGMA Send)* transfers citations of messages rather than the messages themselves This means that the operation defined by this process script is not an instance of the systems Transfer Operation, which would expect *SIGMA Open Message* to actually be sent However. Consul will not be able to understand *SIGMA Forward* unless it can see it in relation to Transfer Operation

The required relationship is constructed by the acquirer in accordance with a stylized dialogue with the service builder (see [Wilczynski 81] in this *Proceedings* for a detailed description). Briefly, the acquirer uses its built-in model of the Transfer Operation to ask the service builder questions about the parameters of *SIGMA Forward.* The service builder s answers to these questions are used to construct an instantiation of Transfer Operation with the same functionality as the process script The result is called SIGMA Forward Operation in Figure 9

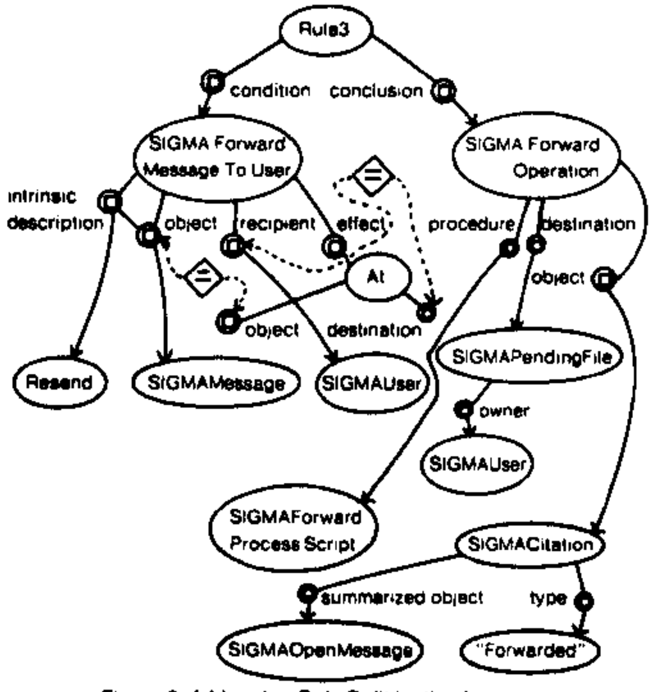

Figure 9: A Mapping Rule Built by the Acquirer

Unfortunately, this is not a Transfer Operation in the sense intended by the service builder: it does not transfer *SIGMA Open Message*, but instead a citation of that message. In order to capture this intended meaning of *SIGMA Forward,* the acquirer must use the results of its dialogue with the service builder to generate the mapping rule shown in Figure 9.

The description generated as the condition part of the rule, SIGMA Forward Message To User, is the appropriate representation of *SIGMA Forward* as a function for forwarding messages; the mapping rule shows how its invocation can be mapped into an invocation of the actual process script. As shown in Figures 2 and 5. it is the generated description SIGMA Forward Message To User that is found during the mapping process (because the user also speaks of messages rather than citations being sent). If such a description had not been generated by the acquirer. Consul would not have been able to find the *SIGMA Forward* process script during its natural language understanding and explanation activities, and the users request could not have received the correct response. The acquirer will not allow process scripts to be entered into the system until it has related them to system model concepts either by instantiation or by mapping rules.

3. Conclusion

The Consul system is still in the early stages of development. It currently demonstrates all of the capabilities discussed in this paper for a small class of user requests and service functions. Only part of the SIGMA-like mail service has actually been modelled and implemented. Immediate plans (the next year or so) call for the expansion of the knowledge base and service implementation to allow cooperative interaction with the full mail service. Then (over the next several years), in order to prove the generality of the system as a cooperative interaction environment, the knowledge base will be extended to model the characteristics of other interactive services. The major problems now facing the Consul project are the acquisition, maintenance, and efficient use of large amounts of organized knowledge. In part, the solution to these problems awaits the higher speed, larger address space machine architectures that are now becoming available.

Acknowledgements

Thomas Lipkis is responsible for the explainer and the classification algorithm, Robert Lingard for the process script programming methodology, and David Wilczynski for the acquisition mechanism.

References

[Bobrow&Webber 80] Robert Bobrow and Bonnie Webber, "Knowledge Representation for Syntactic /Semantic Processing." in *Proceedings* of *the National Conference on Artificial Intelligence,* AAAI. August 1960.

[Brachman 78] Ronald Brachman. *A Structural Paradigm for Representing Knowledge.* Bolt, Beranek and Newman, Inc., Technical Report. 1978.

(Lingard 81] Robert Lingard, "A Software Methodology for Building Interactive Tools." in *Proceedings of the Fifth International Conference on Software Engineering,* 1981.

[Mark 80] William Mark. "Rule-Based Inference In Large Knowledge Bases," in *Proceedings of the National Conference on Artificial intelligence,* American Association for Artificial Intelligence, August 1980.

[SIGMA 79] R. Stotz. R. Tugender, D. Wilczynski, and D. Oestreicher, "SIGMA: An Interactive Message Service for the Military Message Experiment," in *Proceedings of the National Computer Conference,* AFIPS, May 1979.

[Wilczynski 81] David Wilczynski, "Knowledge Acquisition in the Consul System," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* IJCAI, 1981.