A General Framework for Incremental Processing of Multimodal Inputs

Afshin Ameri E., Batu Akan, Baran Çürüklü, Lars Asplund
Mälardalens Högskola, Box 883, 721 23 Västeras, Sweden
afshin.ameri@mdh.se, batu.akan@mdh.se, baran.curuklu@mdh.se, lars.asplund@mdh.se
*

ABSTRACT

Humans employ different information channels (modalities) such as speech, pictures and gestures in their communication. It is believed that some of these modalities are more error-prone to some specific type of data and therefore multimodality can help to reduce ambiguities in the interaction. There has been numerous efforts in implementing multimodal interfaces for computers and robots. Yet, there is no general standard framework for developing them. In this paper we propose a general framework for implementing multimodal interfaces. It is designed to perform natural language understanding, multimodal integration and semantic analysis with an incremental pipeline and includes a multimodal grammar language which is used for multimodal presentation and semantic meaning generation.

1. INTRODUCTION

In-person communications between humans is a multimodal and incremental process [1]. These two features benefit the interaction between humans in several ways:

First, some modalities are more error-prone to some special types of information and can transfer other data types more precisely [8]. In a multimodal communication, humans can use the modality which is more reliable for the information to be transferred.

Second, in a multimodal interaction, different modalities are complementary to each other. This helps to reduce the ambiguity in perceived information and removal of vague data [8].

Third, the incremental nature of communication in humans helps to start processing of perceived inputs from different modalities as they are being received and build up the semantic meaning of them [7] . It means that humans build up their response or reaction as they perceive the inputs [7, 1]. In HCI/HRI domain, incremental processing helps to improve response times of computer systems. Multimodal systems should be very responsive, because (if not) they may lead to repetition of commands from the user, more ambiguity in the recognition process and user annoyance [2].

In this paper we propose a framework for rapid development of multimodal interfaces. The framework is designed to perform modality fusion and semantic analysis through an incremental pipeline. The framework also employs a new grammar definition language which is called COactive Language Definition (COLD). COLD is responsible for multimodal grammar definition and semantic analysis representation.

*This project is funded by Robotdalen, VINNOVA, Sparbanksstiftelsen Nya, EU European Regional Development Fund.

ICMI'11, November 14–18, 2011, Alicante, Spain.

2. BACKGROUND

Multimodal grammars were proposed in [6, 4] for definition of mixed inputs from different modalities. They have used unification-based approaches and later, finite-state transducers (FST) to parse and fuse different modality inputs [4, 5].

A more dynamic approach compared to FST model is presented in [10]. The model uses a graph with terminal and non-terminal nodes for grammar representation. This approach allows for on fly expansion of graph during the recognition [10].

A framework which uses spatial ontologies and user context for multimodal integration is described in [3]. The grammar is defined as a part of Microsoft Speech API, which means that object and other modalities' anchors are defined as part of the speech grammar. In other words, non-speech modalities need a specific speech anchor in order to be integrated into the multimodal interaction.

A framework for rapid development of multimodal applications is presented in [2]. Although authors accept the fact that system responsiveness is an important factor in such systems, they refuse to perform the fusion during the speech and discuss that such an approach may lead to constraints on user interaction.

An interesting multimodal system with an incremental pipeline is presented in [1]. Their system which is called RISE, can process syntactic and semantic information from audio/visual inputs incrementally and generates the feedback on-the-fly. RISE's architecture is based on incremental parsing of speech input with different sub-systems that try to resolve the references in the speech through visual data or the context of dialogue.

An abstract model for implementing an incremental dialogue manager is proposed in [9]. Although the model is

not designed for multimodal interactions, but it can easily be modified for that.

Like [3] and [1], we also use object references during fusion. In order to dismiss the problem of using speech as the main modality, we have defined a multimodal language (which we call it COLD) that allows us to define any mixture of different modalities. Pure non-speech communications are easily achievable through COLD. We also take a similar approach as [10] for grammar representation and parsing. Our contribution is in expanding this model to a more general one which can represent several modalities in the same graph.

3. ARCHITECTURE

An important part of the framework is COLD language. COLD definitions are used to (1) generate separate grammars for each modality, (2) define the fusion pattern, (3) define the semantic variables and calculations and (4) define dialogue patterns and dialogue turns.

3.1 COLD Language

In a similar approach to [4]; COLD language is capable of defining the grammar of multimodal sentences and their respective semantic representation. We have expanded Johnston's grammar to (1) include optional phrases and (2) support semantic variable handling.

3.1.1 COLD Syntax

A sample of COLD code is shown in Figure 1. COLD's syntax can be divided into two subcategories: (1) grammar definition and (2) semantic representation. Semantic representation code is always inside curly braces. It can be any Prolog code which is syntactically correct. The rest of the code defines the grammar with the following syntax:

```
predicate (variable list) --> atom/predicate list
```

The "`-->`" sign, separates a predicate from its definition. Predicates can define variables that will be used in semantic analysis. Predicate definition is a space-separated list of atoms or reference to other predicates. In order to separate atoms for different modalities a colon ':' is used. For example the first definition of "pickup" in Figure 1 means that `pickup` is the atom for first modality and `singleObjectSelect` is an atom for the second modality.

Refernces to other predicates must be enclosed within "<>" signs with all the variables that should be passed to them. For example the second definition of `pickup` in the sample image refers to `object` predicate and passes `TargetObj` and `Props` variables to it.

Finally, optional phrases are enclosed in "[]". They are atoms or predicates which can be omitted from the interaction. For example the first definition of `putin` in figure 1 contains "`[here]`". It means that the user can omit the word "here" and still get the same response.

3.2 Incremental Multimodal Parsing

To make integration of new modalities easier, COLD performs parsing of inputs through dedicated parsers for each modality (unimodal parsers). All of them have access to a shared copy of the multimodal grammar graph and a pool of different hypothesis.

```
command(ObjetAnchor, Retval) --> <pickup(ObjetAnchor, Retval)>
{call(Retval)}.
command(ObjetAnchor, Retval) --> <putin(ObjetAnchor, Retval)>
{call(Retval)}.
command(ObjetAnchor, Retval) --> <execute>.

pickup(TragObj, Retval) --> pickup:singleObjectSelect
{TargObj=mouseReturnValue, Retval=pickup(TargObj)}.
pickup(TragObj, Retval) --> pickup <object(TargObj,Props)>
{Retval = (Props,pickup(TargObj))}.

object(TargObj, Retval) -->a wooden block {Retval = property
(TargObj,name,woodenblock)}.

putin(TargLoc, Retval) --> put it [here]:singleObjectSelect
{TargLoc=mouseReturnValue, Retval = (holding(CurObj), putin
(CurObj, TargLoc))}.
putin(TargLoc, Retval) --> put it in <object(TargLoc,Props)>
{Retval = (holding(CurObj), Props, putin(TargLoc, CurObj))}.

execute --> execute {execute}.
```

Figure 1: Some Sample code of COLD

3.2.1 Multimodal Grammar Graph

Multimodal grammar graph contains all of the data defined in COLD code. It includes three different type of nodes: rules, rule references and phrases. Rules contain a list of all the phrases and rule references that define its grammar. Rule references are references to other rules and phrases are atoms that can be matched and parsed by parsers.

Each phrase or rule reference node contains a Prolog code block object. The Prolog object stores related semantic code. This allows for incremental semantic analysis through incremental execution of the code.

Optional phrases are amrked with a boolean property. All the nodes also include a list of variables that they can access. Since variable values may be different for different competing hypotheses, the values are not defined at graph level.

3.2.2 Hypothesis

Hypotheses are objects that include parse related data. Their most important job is to store values for different semantic variables during a parse. Each hypothesis also contains a list of references to graph nodes that build up that hypothesis. Hypothesis objects have the ability to expand rule references when needed. This means that if a rule reference is to be parsed, its representation in the hypothesis is replaced by the complete representation of referred rule. In such cases, it may be possible that a rule expands into several different lists. Therefore, new copies of the hypothesis object and its related semantic variables are created and added to the pool.

3.2.3 Incremental Parser

Different components involved in parsing and their relations are shown in figure 2. Each modality needs an input registrar. Its job is to capture the data, pack it and send it to the central parser. The central parser receives data from registrars and sends it to its respective parser. It also detects completely parsed hypotheses and fires events in such cases. Unimodal parsers are only responsible for parsing inputs that are related to them.

Figure 2 shows the components as they are in our test setup, but there may be more input registrars and unimodal parsers for other scenarios.

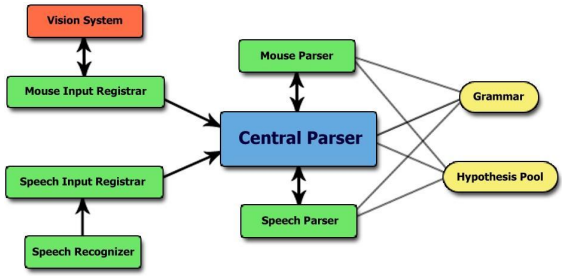

Figure 2: Different components involved in parsing.

3.3 Modality Fusion

A direct result of using multimodal grammar graph is that parsers will also contribute to fusion during their parse. Because different modalities are already represented in a unified format in the graph; as the incremental parsing goes forward, the results from different parsers, will be fused together consequently.

An important factor in modality fusion is synchronization of different modalities. For example the `putin` command (Figure 1) is defined as `put it [here]:singleObjectSelect`, but it should not mean that the object selection can only be accepted after saying the word "here". Related inputs from different modalities have a relation in time. This time relation is reported to be 1000 ms in [10] and between 1000 ms to 2000 ms in [5]. We use a 2000 ms time difference.

In order to account for time relations, parsers are allowed to postpone a parse request if the next node in the graph does not have the same modality as the new input. In such cases, parsers store the information locally and return a value to the central parser indicating a postponed parse. Upon arrival of a new input, the central parser first sends the new input to the respective parser then asks the postponed parser to check for a new parse. The parser then decides to (1) accept the data (2) remove the data if the two seconds time difference does not approve; or (3) request for another postpone, if the two seconds difference still stands.

3.4 Semantic Analysis

Semantic analysis can be referred to as a search on the set of possible target objects and actions. In this sense each phrase of each modality can affect the final outcome of semantic analysis. An easy way of performing this search is to define different variables and narrow down their values based on the inputs from different modalities. Prolog language has proven to be the language of choice in such cases for many researchers, mostly due to its ease of use.

The COLD framework supports Prolog predicates and variables within COLD language. Each phrase or rule reference can have its related Prolog code and variables. When they are successfully parsed, the respective Prolog code is executed and variables are updated. We use SWI-Prolog as the core Prolog engine in the framework.

Because rule references act like a function call to a rule, all the variables that are defined within the rule reference should be passed to the rule. This happens as a call-by-reference function call. This design helps us to define semantic variables in the highest level of the grammar and allow lower parts of it to manipulate them. The final value of the variable is the semantic representation of that rule.

```
1
2  SpeechRegisterer:
3  New Input > Input: put , Time
       :634412439907006250
4
5  Central Parser : Hypothesis Pool Change
6  cputin(@putin putin(put . it here
       singleObjectSelect))
7  cputin(@putin putin(put . it in @object))
8  (_G17561,_G17562)
9  (_G17564,_G17565)
10
11 SpeechRegisterer:
12 New Input > Input: put it , Time
       :634412439909506250
13
14 Central Parser : Hypothesis Pool Change
15 cputin(@putin putin(put it . here
       singleObjectSelect))
16 cputin(@putin putin(put it . in @object))
17 (_G17573,_G17574)
18 (_G17576,_G17577)
19
20 MouseRegisterer:
21 New Input > Type: singleObjectSelect,
       Target:object119,
22        Time:634412439909662500
23
24 Central Parser : Hypothesis Pool Change
25 cputin(@putin putin(put it . in @object))
26 cputin(@putin putin(put it . here
       singleObjectSelect))
27 (_G17588,_G17589)
28 (object119, (holding(_G17602),putin(_G17602
       ,object119)))
29
30 SpeechRegisterer:
31 New Input > Input: put it here , Time
       :634412439914818750
32
33 Central Parser : Hypothesis Pool Change
34 cputin(@putin putin( put it in . @object))
35 cputin(@putin putin( put it here .
       singleObjectSelect))
36 (_G17768,_G17769)
37 (object119, (holding(_G17782),putin(_G17782
       ,object119)))
38
39 Central Parser : Input Recognized
40 (object119, (holding(_G17743),putin(_G17743
       ,object119)))
```

Figure 3: output of the system while parsing the sentence "put it here" and a click.

Prolog statements in COLD language are not used for semantic analysis only, but are also useful for real-time feedback to the user. For example, the `pickup` phrase can contain a predicate that highlights all pickable objects after receiving the word "pickup" from audio channel. This happens during the speech by the user and helps the user to develop a better interaction with the system.

Figure 3 shows system's output during a parse. In order to fit the output in the paper, it has been edited. At lines 2 and 3, SpeechRegistrar reports the input: "put". Central parser creates two hypothesis based on it (lines 6,7) and

reports their respective semantic representation (lines 8,9). Note that at this point the semantic representation only includes empty Prolog variables. After arrival of word "it", both hypothesis stand and since the word has no semantic code assigned to it, nothing changes (lines 11-18). At line 20, a mouse input arrives and it carries the selected object (object119) with it. Now the parser can parse the "singleObjectSelect" node, which belongs to mouse modality and this leads to a more meaningful semantic representation (lines 20-28). In this case, the parser did not wait for the word "here", because it is defined as an optional phrase in grammar. After arrival of the word "here", the parser fires a recognition event and reports its semantic meaning. The changes that are made in variable names during different steps of parse are due to variable copying that was described before.

4. RESULTS

To demonstrate the use of the proposed framework, we are presenting a test case. Our setup consists of an ABB IRB140 industrial robot with a gripper and a camera attached to the 6th joint, therefore the user can view the world through the robots eyes. The images from the camera are overlaid with the transparent virtual objects creating an augmented reality (AR) user interface. Using the mouse, the user can select objects and give voice commands to pick them up and to put them in a location. The semantically analysed voice and mouse commands are converted to simple Prolog commands such as `pickup(ObjectName)`, which are later expanded to approach, and grasping patterns and finally converted to RAPID language (programming language for ABB robots), and sent to the robot controller for execution.

We used a kitting theme including a palette with empty places. Red, blue and green colored objects in 3 different stacks are placed on the workbench. The task is to create a robot program for re-arranging the colored objects on the palette. The Prolog statements generated at the end of the semantic analysis can be saved to a file, and recalled later on to recreate the robot program.

To assist the user, we highlight possible objects that are applicable to the given command. For example in the pickup case, when the user says the words "pickup" all pickable objects are highlighted. As the user continues to describe which object he wants the robot to pickup, irrelevant highlights are removed. In Figure 4 samples of this highlighting can be seen. The incremental nature of our system can be used to generate early feedback and assistance to the user even before the voice commands end. It is also useful in conditions when user commands are ambiguous; the feedback can help the user to recognize the ambiguity before finishing the sentence.

5. CONCLUSION

We have designed and implemented an incremental framework for development of multimodal interactions and tested it in a kitting scenario. Incremental processing of inputs allows for in-time interaction between the robot and its human colleague. As a part of this framework we introduced COLD language, which we use to define the multimodal grammar and semantic analysis procedures. Semantic analysis code is directly written in COLD with a Prolog-like syntax, making development easier.

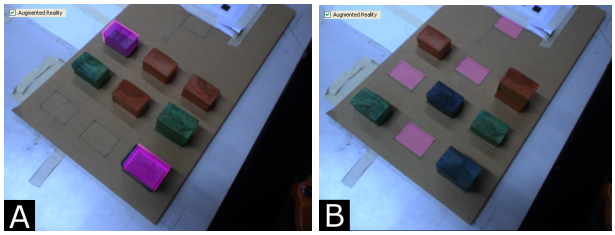

Figure 4: AR UI. (a) Blue objects are highlighted after user says "pickup a blue object". (b) While the robot is holding an object, the user says "put", and all empty locations are highlighted.

We plan to add context-manager and dialogue manager to the pipeline in the next steps. The context-manager will try to resolve inter-dialogue relations. Dialogue manager will get the outputs of the semantic analysis routines and builds up a dialogue response for the user.

6. REFERENCES

[1] T. Brick and M. Scheutz. *Incremental natural language processing for HRI*. ACM Press, New York, New York, USA, 2007.

[2] F. Flippo, A. Krebs, and I. Marsic. A framework for rapid development of multimodal interfaces. *Proceedings of the 5th international conference on Multimodal interfaces - ICMI '03*, page 109, 2003.

[3] S. Irawati, D. Calderón, and H. Ko. Spatial ontology for semantic integration in 3D multimodal interaction framework. In *Proc. ACM VRCAI2006*, volume 1, pages 129–135, New York, New York, USA, 2006. ACM.

[4] M. Johnston. Unification-based multimodal parsing. *Proceedings of the 36th annual meeting on Association for Computational Linguistics* -, page 624, 1998.

[5] M. Johnston and S. Bangalore. Finite-state multimodal integration and understanding. *Natural Language Engineering*, 11(02):159–187, 2005.

[6] M. Johnston, P. R. Cohen, D. McGee, S. L. Oviatt, J. a. Pittman, and I. Smith. Unification-based multimodal integration. *Proceedings of the 35th annual meeting on Association for Computational Linguistics* -, pages 281–288, 1997.

[7] Y. Kamide, G. T. M. Altmann, and S. L. Haywood. The time-course of prediction in incremental sentence processing: Evidence from anticipatory eye movements. *Journal of Memory and Language*, 49(1):133–156, 2003.

[8] S. L. Oviatt. Ten myths of multimodal interaction. *In CACM*, 42(11):74–81, 1999.

[9] D. Schlangen and G. Skantze. A general, abstract model of incremental dialogue processing. *Proc. EACL '09*, (April):710–718, 2009.

[10] R. Stiefelhagen, C. Fogen, P. Gieselmann, H. Holzapfel, K. Nickel, and a. Waibel. Natural human-robot interaction using speech, head pose and gestures. *IROS 2004*, pages 2422–2427, 2004.