Integrating Multimodal Cues Using Grammar Based Models

Manuel Giuliani and Alois Knoll

Robotics and Embedded Systems Group
Department of Informatics, Technische Universität München
Boltzmannstraße 3, D-85748 Garching bei München, Germany
giuliani@in.tum.de, knoll@in.tum.de

Abstract. Multimodal systems must process several input streams efficiently and represent the input in a way that allows the establishment of connections between modalities. This paper describes a multimodal system that uses Combinatory Categorial Grammars to parse several input streams and translate them into logical formulas. These logical formulas are expressed in Hybrid Logic, which is very suitable for multimodal integration because it can represent temporal relationships between modes in an abstract way. This level of abstraction makes it possible to define rules for multimodal processing in a straightforward way.

1 Introduction

Multimodal systems are programs that are able to process several input streams and produce multimodal output. They record the utterances of a human user, combine them to build a conjoined interpretation, and derive their output from that interpretation. Therefore multimodal systems are more flexible and error-tolerant than systems that have only one input modality. In the best case, multimodal systems can be very natural to interact with, because the users do not have to learn how to operate the system: They can simply interact with it as they would with another human. To realise their full potential, multimodal systems must not only be able to recognise multiple modes, they also have to process the input information fast and in a way that allows a computer to reason over the input and determine the action it should execute.

This paper describes an approach for a multimodal system that involves an input processing based on a grammar formalism that is mainly used for language processing: Combinatory Categorial Grammar (CCG). By using OpenCCG, a Java-based implementation of the CCG, the input of speech recognition and other modes can be translated into Hybrid Logic formulas. These formulas can then be used to compute reactions by the system on a user input.

The structure of this paper is as follows. In Section 2 we review related work to our approach. Sections 3 and 4 introduce CCG and OpenCCG. Hybrid Logic is explained in Section 5 before our approach for a multimodal system is depicted in Section 6. Section 7 shows an example input processing, while Section 8 concludes this paper.

C. Stephanidis (Ed.): Universal Access in HCI, Part II, HCII 2007, LNCS 4555, pp. 858–867, 2007.

2 Related Work

Using grammars for the processing of multimodal cues has been proposed in different variations. The first who reported a grammatical framework for a multimodal interface were Shimazu, Arita, and Takashima [1]. They developed a Multimodal Definite Clause Grammar (MM-DCG) that was able to handle an arbitrary number of modes and stored temporal information inside the grammar. The DCG formalism that they used is suitable for a straightforward transformation of the grammar rules into Prolog code. However, MM-DCG assumes that the several modes are known from the beginning and ties them strictly together in the grammar. This leads to very domain-specific grammars that are hard to extend.

Johnston and Bangalore [2] use a multimodal context-free grammar to represent the input streams of speech recognition and pen input. The word entries of their grammar are composed of $n + 1$ components for the n input modes and one component for their joint meaning. Therefore, the grammar is also a semantic module which adds meaning to the input stream while parsing it. This approach is very useful for small domains. Johnston and Bangalore describe a grammar that was created for an application that manages contact details with speech and pen input in a PDA. The possible speech-gesture combinations for this example are limited and known in advance. That makes it possible to store them together in a single grammar. Also, the application does not change very much, hence they do not need a grammar that is easy to extend.

Nevatia, Zhao and Hongeng [3] introduce an event ontology that aims to display complex spatio-temporal events in a simple way. For that, they propose an Event Recognition Language (ERL) to specify possible events for a system that recognises certain events like human movements in a video stream. The events are divided into three classes: primitive events build the basis of the hierarchy, single-thread events are composed of primitive events, and multi-thread events are a number of single-thread events with a temporal/spatial/logical relationship. An approach like the ERL makes it possible for every user to specify complex events without worrying about the low-level processing of the underlying system.

Ryoo and Aggarwal [4] propose a general methodology for automated recognition of complex human activities. They use a representation scheme based on a context-free grammar (CFG). Similar to the ERL, this scheme can represent basic actions and complex actions. In this case the CFG allows the human actions that should be recognised by their system to be defined. In the program, they involve the CFG-based representation to recognise human activities in video streams automatically. The achieved recognition results are very high, but only for movements that correspond to the actions specified in the grammar. The probabilistic nature of human activity is not yet captured in their system.

3 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) was introduced by Ades [5] and Steedman [6]. It is an extension to the Categorial Grammar (CG) of Ajdukiewicz [7] and

Bar-Hillel [8]. Traditional context-free grammar formalisms use a top-down approach for parsing sentences. The structure of a language that should be parsed is stored in a set of rules, while the lexicon contains words with their assigned word categories. Categorial Grammars (CG) have a bottom-up approach in which the structure of a language is reflected in the lexicon of the grammar. Each word in the lexicon of a CG is assigned to a category that can be either atomic or complex. Example (1) shows the lexicon for a small grammar[1]:

(1) a. *loves* := (s\np)/np : $\lambda x.\ \lambda y.$**love**(x,y)
b. *Peter* := np : **Peter**
c. *Mary* := np : **Mary**

This publication follows Mark Steedman by using the "result leftmost" notation. This notation utilises the slash operators / and \. The rightward-combining functor over domain β into range α is written α / β, while the leftward-combining functor is written α \ β. For example, ditransitive verbs like *loves* use the category (s \ np) / np presented in (1a). They can be combined with a nominal phrase (np) that stands to the right of the verb. After that, they are combined with a np to the left of the verb to yield a sentence (s).

The combination of words to sentence constituents is done by a small set of rules. The simplest rules used in a CG are the rules of *Functional Application*, as displayed in (2):

(2) *Functional Application Rules*
a. $X/Y : f \quad Y : a \quad \Rightarrow \quad X : fa \qquad (>)$
b. $Y : a \quad X\backslash Y : f \quad \Rightarrow \quad X : fa \qquad (<)$

With the rules of Functional Application and the categories from the small grammar example in (1), the sentence "Peter loves Mary" can be parsed as shown in (3):

(3)
Peter	*loves*	*Mary*
np	(s\np)/np	np
	s\np (>)	
np (<)		

In addition to the rules of Functional Application, CCG adds a couple of extra rules to the basic set of rules. The first of these extra rules are the rules of *Harmonic Functional Composition* that are introduced in (4). They allow the combination of contiguous words that do not build a sentence constituent together. The rules of forward and backward composition are depicted by "**>B**" and "**<B**" respectively.

(4) *Harmonic Functional Composition Rules*
a. $X/Y : f \quad Y/Z : g \quad \Rightarrow \quad X/Z : \lambda x.f(gx) \qquad (>\mathbf{B})$
b. $Y\backslash Z : g \quad X\backslash Y : f \quad \Rightarrow \quad X\backslash Z : \lambda x.f(gx) \qquad (<\mathbf{B})$

[1] The CCG rules and the grammar example in (1) use the λ-calculus for the display of semantics. The λ-calculus will not be explained in this paper, because OpenCCG - that will be illustrated in Section 4 - uses Hybrid Logic.

The second type of rules that are introduced by CCG are the *Type-Raising* rules. These rules turn atomic categories into functions over functions over this category. The Type-Raising rules are displayed in (5); the variable *T* is a variable that stands for all categories.

(5) *Type-Raising Rules*

a. $X : a \Rightarrow T / (T \setminus X) : \lambda f.fa$ (>**T**)

b. $X : a \Rightarrow T \setminus (T / X) : \lambda f.fa$ (<**T**)

4 OpenCCG

OpenCCG [9] is a Java-based implementation of the CCG formalism. It is capable of parsing and realising sentences. That means it can translate utterances into a logical form as well as take a given logical form and convert it back to a sentence. OpenCCG emerged from the Grok system and was extended by Michael White. In OpenCCG rules, categories, and lexicon entries are stored in a set of mandatory and optional XML files. Traditionally CCG uses the λ-calculus to represent parsed sentences, but OpenCCG uses the more flexible *Hybrid Logic* as proposed by Baldridge and Kruijff [10] to implement a dependency-based perspective on meaning. Hybrid Logic will be explained in the next section.

5 Hybrid Logic

Hybrid logic goes back to the hybrid tense logic by Prior [11], which is a hybridised version of ordinary tense logic. Although classical logic and modal logic are quite powerful in expressing relations between entities and their properties, they lack the ability to directly reference specific states at which a proposition holds. In standard modal logic, truth is relative to a set of points. These points are usually taken to represent worlds, times, space or states in a computer. Therefore only expressions that are for example relative to a set of times, like the sentence

(6) The sun is shining.

can be formalised in modal logic. This statement has different truth values at different times. The problem is that some statements in natural language are true exactly at one time only. For example the statement

(7) It is 7 o'clock 11 October 2006.

is only true at exactly 7 o'clock on the 11th of October 2006 and false at all other times. Example (6) can be formalised in modal logic, but this is not possible with example (7).

Therefore, a logic is needed with the ability to name states (or points, worlds, times) and reference them later. Hybrid logic extends modal logic to reach that goal. By introducing a new class of formulas, called *nominals*. Nominals are used to name states; they are true at exactly one state and are used instead of ordinary propositional symbols. Formulas can be formed by using standard boolean operators and a new *satisfaction operator*, which is depicted by @. The satisfaction operator makes it

possible to formalise that a statement is true at a particular state. The formula $@_i p$ is defined as follows:

Definition 1. *$@_i p$ is true if and only if the proposition p is true in the unique state named by the nominal i*

Nominals and the satisfaction operator make it possible to formalise many relations that cannot be represented with standard modal temporal logic. The following sentence[2]

(8) Ed finished the book.

cannot be expressed by the modalities ⟨F⟩ and ⟨P⟩ (staying for future and past) alone, because these modalities do not state the exact point in time at which Ed finished the book. Through the addition of a nominal i it is possible to refer to that point in time:

(9) $\langle \mathrm{P} \rangle (i \wedge \mathbf{Ed\text{-}finish\text{-}book})$

Hybrid logic is also suitable to represent linguistic meaning of sentences. This is done by a conjunction of modalised terms, which are tied by a nominal that names the head proposition of that conjunction. In Example (10), the hybrid logic representation of the sentence *Ed wrote a long book in London* is shown:

(10) $@_{h_1}(\mathbf{write} \wedge \langle \mathrm{ACT} \rangle (d_0 \wedge \mathbf{Ed}) \wedge \langle \mathrm{PAT} \rangle (d_5 \wedge \mathbf{book} \wedge \langle \mathrm{GR} \rangle (d_7 \wedge \mathbf{long})) \wedge \langle \mathrm{LOC} \rangle (d_9 \wedge \mathbf{London}))$

The whole expression is dominated by the head proposition **write**, which is denoted by the nominal h_1. The modalities ⟨ACT⟩, ⟨PAT⟩, ⟨GR⟩ and ⟨LOC⟩ stand for the dependency relations *Actor*, *Patient*, *Locative* and *General Relationship* respectively. These relations are dependent on the head proposition. The hybrid logic term in Example (11) shows the general definition for the dependency relations:

(11) $@_h(\mathbf{proposition} \wedge \langle \delta_i \rangle (d_i \wedge \mathbf{dep}_i)$

The dependency relation $\langle \delta_i \rangle$ is associated with a nominal d_i. Each nominal d_i names a state where a dependent expressed as a proposition $\mathbf{dep}_i$, should be evaluated. δ_i is a successor of *h*, the nominal identifying the head of the hybrid logic term.

6 Multimodal System

This section describes the multimodal system we propose for the grammar-based integration of several modes. As discussed in the previous sections, the system has to meet the following design criteria:

– The system must be independent from specific input modes. This ensures that the system can always be extended by additional modes.

[2] Examples (8), (9), and (10) are taken from [10].

- The data of the input modes must be accessible at a high level. For this the input processing and the complexity of the modes has to be transparent and the input streams must be represented on an abstract level.
- The time at which events like speech or gesture input occur must be captured and represented in the system. This happens both on a low level where timestamps of events are stored and also on a high level where the points in time are represented in an abstract representation.

Section 6.1 gives an overview, while several aspects of the proposed system are highlighted in Sections 6.2 to 6.4.

6.1 System Overview

Figure 1 shows an overview for our multimodal integration system. The input streams of the different modes are parsed by CCGs and represented on an abstract level by Hybrid Logic formulas. As mentioned in section 5, in Hybrid Logic formulas time points can be represented by nominals. These nominals are stored in the nominal-timestamp mapping module together with two timestamps that refer to the start and end time of an input stream. The Hybrid Logic formulas can be processed in two ways; on the one hand they are used to generate rules for the rule set of the system, while on the other hand the formulas can be processed directly in the online mode. For this, the set of rules must be already filled. The rules can be entered manually or learnt automatically from stored input data.

6.2 Input Modules

The several input modules have to register with the system before their input can be processed. The system has to make sure that an appropriate grammar is present for all registered input modules or that a new module can send well-formed Hybrid Logic formulas in any form. It is also imaginable that data sources from other multimodal systems may serve as input to our system. Since Hybrid Logic formulas can be represented in XML, any data that is in XML and involves information about input streams and timestamps can be transformed to Hybrid Logic formulas via XSLT stylesheets. The output of multimodal annotation tools like ELAN [12] or EXMARaLDA [13] can also be a source of data to the system.

The input modes that are used in our system are speech recognition, object recognition, face tracking, and gesture recognition. The use of grammars for parsing speech input is straightforward and common. For the other input modes we are planning to use a similar approach like in [3] and [4] that involves a hierarchy of actions/gestures. In this way, complex actions can be composed by primitive actions.

6.3 Nominal-Timestamp Mapping

The nominal-timestamp mapping module has two purposes. First, it stores every nominal with two timestamps for the start time and end time of the event that is denominated by the nominal. The method of determining timestamps for each event

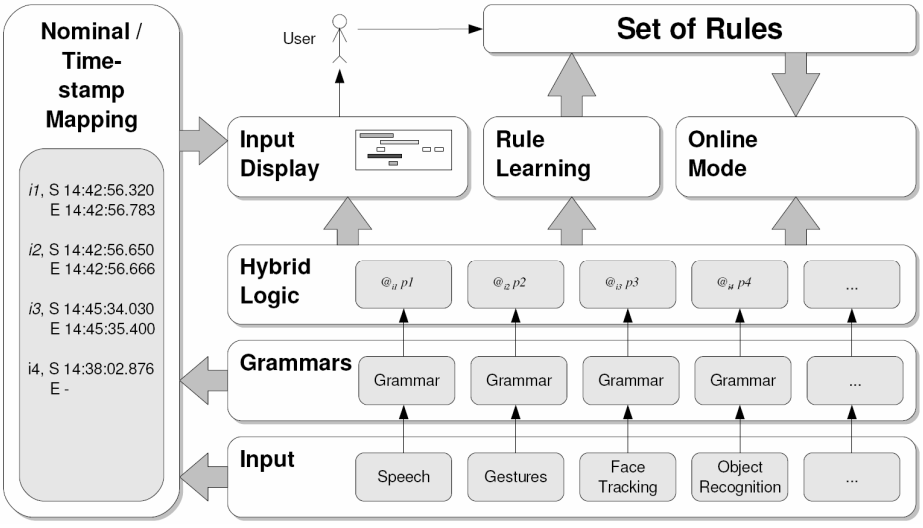

Fig. 1. Overview for the multimodal system

must be designed carefully. Especially as they differ for every input mode. For example, timestamps for speech recognition can be saved at the level of sentences or individual words, while the input from the face tracking component cannot be saved on a video frame level because the amount of data would be too large.

The second purpose of the nominal-timestamp mapping module is to allow other modules to compare the temporal relation between nominals. The four temporal relations that should be available are defined in the following list, these are similar to the interval relations reported by Allen in [14]:

precedes(x,y) Returns true if the end time of nominal x is earlier than the start time of nominal y.

succeeds(x,y) Returns true if the start time of nominal x is later than the end time of nominal y.

includes(x,y) Returns true if the start time of nominal x is earlier than the start time of nominal y and the end time of nominal x is later than the end time of nominal y.

includedBy(x,y) Returns true if the start time of nominal x is later than the start time of nominal y and the end time of nominal x is earlier than the end time of nominal y.

6.4 Grammars, Hybrid Logic Formulas

A grammar for speech processing has already been developed for German and English. It is written with OpenCCG and is designed for a specific task where a user

instructs a robot to pick up and lay down objects like cubes, bolts and slats. See [15] for a complete description of the task and the robot assembly. The grammar can parse the following sentence types:

Statements	*"The bolt is yellow."*
Imperative Sentences	*"Take a slat!"*
Questions	*"Where is the red cube?"*
Confirmations	*"Yes." "No." "OK."*

The Hybrid Logic formulas representing the parsed sentences display their syntactic structure, as shown in Section 7.

The grammars for the other input modules like face tracking and gesture recognition are based on a hierarchy of actions. Thus, primitive actions build the basis of the hierarchy, while complex actions can be composed by combining those primary actions. For example, for the face tracking it is not suitable to generate logic formulas for every video frame that is captured by the video camera that is filming the user. It is rather desirable to join several frames to one action.

7 Processing Example

This section shows a full example how the processing of three input modes looks like. The three input modes are speech recognition, gesture recognition, and object recognition. The scenario of the example involves a robot, a user, and parts of a toy construction set. The user instructs the robot to pick up a certain object by pointing at a cube and saying *"Take this cube."*'. The sentence is passed by the speech recognition to the grammar, which yields the logical form presented in (12):

(12) $$@_{x_1}(\langle \text{VERB} \rangle \text{ take} \wedge \langle \text{NOUN} \rangle \text{ cube} \wedge \langle \text{DEICTIC} \rangle \text{ this})$$

(13) displays the logical form that is generated by the gesture recognition:

(13) $$@_{x_2}(\langle \text{GESTURE} \rangle \text{ pointAt})$$

The input by the object recognition is also parsed by a grammar. The corresponding logical form can be seen in (14):

(14) $$@_{x_3}(\langle \text{OBJECT} \rangle \text{ cube})$$

The processing of the Hybrid Logic formulas involves the following steps: First, the collected Hybrid Logic formulas are displayed for the user in an abstract way. Figure 2 shows one possible way to present the input modes.

From this presentation, and with the help of the time relation functions of the nominal-timestamp mapping module, the developer can generate rules in a pseudo code that are then translated back into Hybrid Logic formulas. These rule formulas are stored in the set of rules afterwards and can be applied for input processing in the online mode afterwards. Example (15) shows a rule in pseudo code for the example given here:

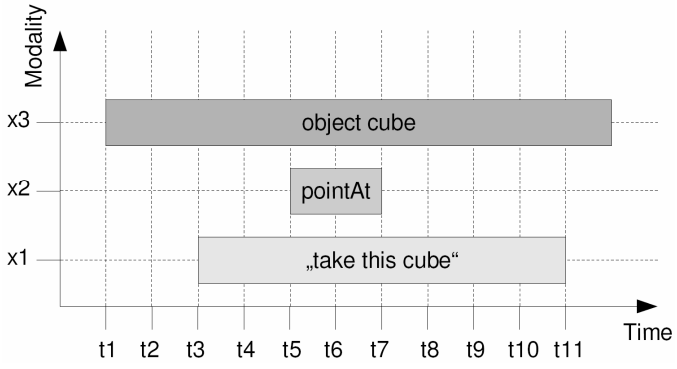

Fig. 2. Display of three input modes: Speech recognition x_1, gesture recognition x_2, and object recognition x_3

$$
\begin{array}{lll}
(15) & \text{if} & @_{x_1}(\langle \textsc{verb} \rangle \text{ take} \wedge \\
 & & \quad \langle \textsc{noun} \rangle\ O_1 \wedge \\
 & & \quad \langle \textsc{deictic} \rangle \text{ this }) \\
 & \text{and} & @_{x_2}(\langle \textsc{gesture} \rangle \text{ pointAt }) \\
 & \text{and} & @_{x_3}(\langle \textsc{object} \rangle\ O_2\ ) \\
 & \text{and} & O_1 \equiv O_2 \\
 & \text{and} & includes(x_1, x_2) \\
 & \text{then} & take(robot, O_1)
\end{array}
$$

8 Conclusion

We described an approach for a multimodal system that uses Combinatory Categorial Grammars to parse input modes. The parsing process yields a representation of the modalities in Hybrid Logic, which is very useful for this task because it models temporal relationships. The proposed system displays the temporal connections between input modes on an abstract level, which allows the developer of the system to manually generate processing rules that enable the system to respond to user input appropriately.

Acknowledgements. This work was supported by the EU FP6 IST Cognitive Systems Integrated Project “JAST” (FP6-003747-IP), http://www.euprojects-jast.net/.

References

1. Shimazu, H., Arita, S., Takashima, Y.: Multi-modal definite clause grammar. In: Proceedings of the 15th Conference on Computational linguistics, pp. 832–836. Association for Computational Linguistics, Morristown, NJ, USA (1994)
2. Johnston, M., Bangalore, S.: Finite-state multimodal parsing and understanding. In: Proceedings of COLING-2000, Saarbruecken, Germany (2000)

3. Nevatia, R., Zhao, T., Hongeng, S.: Hierarchical language-based representation of events in video streams. In: IEEE Workshop on Event Mining (2003)
4. Ryoo, M.S., Aggarwal, J.K.: Recognition of composite human activities through context-free grammar based representation. In: CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, pp. 1709–1718. IEEE Computer Society Press, Los Alamitos (2006)
5. Ades, A.E., Steedman, M.J.: On the order of words. Linguistics and philosophy 4, 517–558 (1982)
6. Steedman, M.: The syntactic process. MIT Press, Cambridge, MA (2000)
7. Ajdukiewicz, K.: Die syntaktische konnexität. Studia Philosophica 1, 1–27 (1935)
8. Bar-Hillel, Y.: A quasi-arithmetic notation for syntactic description. Language 29, 47–58 (1953)
9. White, M.: Efficient realization of coordinate structures in combinatory categorial grammar. Research on Language & Computation 4(1), 39–75 (2006)
10. Baldridge, J., Kruijff, G.J.: Coupling ccg and hybrid logic dependency semantics. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 02), University of Pennsylania, Philadelphia, PA (2002)
11. Prior, A.: Past, Present and Future. Oxford University Press, Oxford (1967)
12. Brugman, H., Russel, A.: Annotating multi-media/multi-modal resources with elan. In: 4th International Conference on Language Resources and Evaluation (LREC2004), Lisbon (26.05.2004-28.05.2004 2004), pp. 2065–2068 (2004)
13. Schmidt, T., Wörner, K.: Erstellen und analysieren von gesprächskorpora mit exmaralda. Gesprächsforschung - Online-Zeitschrift zur verbalen Interaktion Ausgabe 6, 171–195 (2005)
14. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM 26(11), 832–843 (1983)
15. Foster, M.E., By, T., Rickert, M., Knoll, A.: Human-robot dialogue for joint construction tasks. In: Proceedings, Eighth International Conference on Multimodal Interfaces (ICMI 2006), Banff (November 2006)