A Grammar Formalism for Specifying ISU-based Dialogue Systems

Peter Ljunglöf, Department of Linguistics, University of Gothenburg

We describe how to give a full specification of an ISU-based dialogue system as a grammar. For this we use Grammatical Framework (GF), which separates grammars into abstract and concrete syntax. All components necessary for a complete GoDiS dialogue system are specified in the abstract syntax, while the linguistic details are defined in the concrete syntax. Since GF is a multilingual grammar formalism, it is straightforward to extend the dialogue system to several languages.

The information-state update approach

The GoDiS dialogue manager [1] is based on formal semantic and pragmatic theories of dialogue, and provides general and fairly sophisticated accounts of several common dialogue phenomena such as interactive grounding (a.k.a. verification), accommodation, keeping track of multiple conversational threads, and mixed initiative. General solutions to general problems allow modularity, re-use and rapid prototyping.

GoDiS is based on the Information State Update (ISU) approach to dialogue management [4]. The ISU approach, which has been developed over the last 10 years in several EU-funded projects, provides a generalization over previous theories of dialogue management and allows exploring a middle ground between sophisticated but brittle research systems, and robust but simplistic commercial systems. In the ISU approach, a dialogue manager is formalized as: (i) an information state (IS) type declaration, (ii) a set of dialogue moves, and (iii) information state update rules.

In GoDiS, which is based on a theory of Issue-Based Dialogue Management (IBDM), a single script (called a *dialogue plan*) can be used flexibly by the dialogue manager to allow for a wide range of dialogues. The main benefit of the IBDM account as implemented in GoDiS is the combination of advanced dialogue management and rapid prototyping enabled by cleanly separating generic dialogue principles from application-specific domain knowledge.

GoDiS enables rapid prototyping of systems with advanced dialogue behavior. However, the GoDiS dialogue manager only communicates with the outside world using semantic representations called *dialogue moves*. The designer of the dialogue system must implement a translation between natural language utterances and dialogue moves, be it through a simple lookup table, or an advanced feature-based grammar. Furthermore, a speech-based system also needs a statistical language model or a speech recognition grammar. In this paper we show how a GoDiS dialogue system can be specified as a single grammar in the Grammatical Framework. All components necessary for a ISU-based dialogue system are then automatically generated from the grammar.

The GoDiS dialogue manager

The GoDiS system communicates with the user via *dialogue moves*. There are three main dialogue moves – requesting actions, asking questions and giving answers. Apart from the three main moves there are also different kinds of feedback moves – confirmations, failure reports and interactive communications management.

The basic building blocks in GoDiS are individuals, sorts, one-place predicates and actions. From these all necessay dialogue moves can be built, such as questions, answers, requests and feedback. To specify a GoDiS dialogue system, we have to give the following information: (i) the sortal hierarchy, (ii) the individuals and the sorts they belong to, (iii) the predicates and their domains, (iv) the actions, and (v) the dialogue plans. Furthermore, there has to be an interface to each external device.

Dialogue plans convey what the system can do and/or give information about. A dialogue plan is a receipt for the system, so it knows how to answer a specific question, or how to perform a given action. The dialogue plans can roughly be divided into three different kinds – actions, issues and menus. An *action plan* is when the user wants the system to perform an action, e.g., to call a contact in the address book. An *issue plan* is when the user wants the system to give information, such as telling the phone number of a contact in the address book. A special kind of action plan is the *menu*, where the user can select from any of a given number of sub-plans which the system then performs.

Grammatical Framework

Grammatical Framework [2] is a grammar formalism based on type theory. The main feature is the separation of *abstract* and *concrete* syntax, which is crucial for our treatment of dialogue systems. The abstract syntax of a GF grammar consists of declarations of categories and functions. Function declarations correspond to rules in a context-free grammar.

The concrete syntax consists of *linearizations* of the abstract functions. Linearizations are written in a typed functional programming language, which is very expressive but still decidable.

It is possible to define different concrete syntaxes for one particular abstract syntax, making GF a multilingual grammar formalism. Furthermore, the abstract syntax of one grammar can be used as a concrete syntax of another grammar, which makes it possible to implement grammar resources to be used in several different application domains. These points are currently exploited in the GF Resource Grammar Library [3], which is a multilingual GF grammar with a common abstract syntax for 13 languages, including Arabic, Finnish and Russian.

GoDiS specification as GF abstract syntax

Action and issue plans are specified as functions with result categories Action(m) and Issue(m) respectively, where m specifies which menu they belong to:

fun callContact : Name → Action(MakeCall)

fun searchForNumber : Name → Number →
Issue(ManageContacts)

The first specification states that callContact is an action plan in the MakeCall menu. It takes one argument, which is the Name of the contact to call. The second specification is the issue plan searchForNumber. It also takes one Name argument, which the system will ask for if not already said by the user. The final Number argument represents the system's answer, and will be filled by the system when it knows the answer.

Everything else in the GF grammar specifies the ontology of the dialogue system. From the grammar we can extract the sorts and the sortal hierarchy, the individuals and the predicates.

Dialogue utterances as GF concrete syntax

The concrete syntax is responsible for translating everthing the user says into dialogue moves, and what the system might want to say into natural language. For each function in the abstract syntax, there has to be a corresponding linearization. E.g., the callContact action might have the following linearization:

lin callContact(x) = "call" ++ *variants*{x ; "a contact"}

This linearization will be used in several places by the dialogue system. First, we can use it in parsing the user utterances "call anna" (or "call a contact"): The result is the GF term callContact(anna) (or callContact(?)), which will be automatically translated into GoDiS dialogue moves. Second, the system will use it when presenting the MakeCall menu: "do you want to call a contact or call a number?" . Third, the system will generate different kinds of feedback moves containing the GF term: "so, you want to call a contact" or "I'm sorry, I cannot call a contact at this moment".

A short example

Assume that the user says "I'd like to call a contact please". This is recognized by the system as the dialogue move request(callContact), which means that GoDiS loads the associated action plan. In this plan it sees that it needs a value for callContact:Name, so it utters the dialogue move ask(?x.callContact:Name(x)). There is an extra linearization for callContact for handling wh-questions (not shown above), translating the dialogue move into "Which name do you want to call?". GoDiS incorporates the user answer as answer(Name(Kim)), and then tells the phone to look up Kim's number in the phone book and call. A confirmation dialogue move, confirm(callContact), is at the same time presented to the user.

References

[1] Staffan Larsson. *Issue-based Dialogue Management*. PhD thesis, Department of Linguistics, Gothenburg University, 2002.

[2] Aarne Ranta. Grammatical Framework, a type-theoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189, 2004.

[3] Aarne Ranta, Ali El-Dada, and Janna Khegai. *The GF Resource Grammar Library*, 2006. Can be downloaded from the GF homepage `http://www.cs.chalmers.se/~aarne/GF`

[4] David Traum and Staffan Larsson. The information state approach to dialogue management. In Smith and Kuppevelt, editors, *Current and New Directions in Discourse and Dialogue*, pages 325–353. Kluwer Academic Publishers, 2003.