Flipper: An Information State Component for Spoken Dialogue Systems

Mark ter Maat and Dirk Heylen

Human Media Interaction, University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
{maatm,heylen}@ewi.utwente.nl

Abstract. This paper introduces Flipper, an specification language and interpreter for Information State Update rules that can be used for developing spoken dialogue systems and embodied conversational agents. The system uses XML-templates to modify the information state and to select behaviours to perform.

1 Introduction

One of the main challenges in creating multi-model dialogue systems is how to respond on the perceived input. Several models have been proposed to make such decisions. For example, one could use a Finite State Machine (e.g. in the agent MACK [1]), or one could use statistical approaches to find the best match (e.g. in the agent Hassan [2]). Another approach presented in the literature is the Information State approach [3]. In this approach, all dialogue information is stored in one structured data-structure called the `Information State` (IS), and `update rules` are used to modify this information and to select behaviour to perform.

This paper introduces Flipper, an implementation of an Information State based template system for dialogue systems. With this system, it is possible to create an information state and templates — written in XML — that either modify the values in the IS or select a certain behaviour to perform.

2 Flipper

Flipper[1] is an Information State update system based on XML-tem-plates, and was developed for the SEMAINE project[2]. The rationale behind the rule system of Flipper is to have a flexible set of easily definable templates which specify exactly what kind of behaviour to perform under which circumstances. Each template has a set of preconditions that all need to be true to fire that template. These preconditions check certain conditions in the IS, usually by comparing

[1] `http://sourceforge.net/projects/hmiflipper/`
[2] `http://www.semaine-project.eu`

H. Högni Vilhjálmsson et al. (Eds.): IVA 2011, LNAI 6895, pp. 470–472, 2011.

values (=, <, >, etc), optionally with several simple arithmetic operations. Data in the IS can be structured by putting it in lists or records.

The second part of the templates contains the effects of the template. These can be modifications to the IS (remove, add or change variables), custom Java-functions, or certain behaviour to perform with optional parameters.

Here is an example template:

```
<template id="RespondToSmile1" name="A response to a smile">
 <preconditions>
  <compare value1="$face.nrOfSmiles" comparator="greater_than" value2="1" />
  <compare value1="$speakingIntention" value2="want_turn" />
 </preconditions>
 <effects>
  <update name="$nrResponses" value="$Agent.totalResponses + 1" />
  <update name="$responses._addlast" value="#Response129" />
 </effects>
 <behaviour class="ResponsePerformer" quality="0.5" />
  <argument name="response_id" value="#Response129" />
 </behaviour>
</template>
```

This template checks if there was a smile and if the agent wants the turn. If so, then the total number of the agent's responses is incremented by one, and the agent's response (#129) is put in a list of performed responses and is send to the ResponsePerformer (a custom Java class that has to be written for each project, since Flipper does not know how to perform a certain behaviour).

The result of the complete checking procedure is a list of templates that have all their preconditions met. In further processing, the first step is to execute all templates that do not result in actual behaviour but only update the IS. After that, a selection algorithm determines which of the templates that do result in actual behaviour will be executed, based on a quality-value that is specified in the template itself. The selected behaviour is then performed.

When incorporating Flipper into a new project, some things have to be done. First of all, an Information State has to be created, which should be filled with data from the input components. Secondly, the templates have to be written, which process this data, make interpretations, and decide how to behave. Finally, the component has to be written that forwards the data in the behaviour element in the templates to the output component of the system.

3 Conclusion

We described Flipper, an Information state based system for dialogue systems that uses XML-templates to modify the information state and select behaviour to perform. We have briefly explained how it works and shown a small example of a template. So far, Flipper has been used in several projects, for example in Semaine and in a demonstration system of the Sera project[3].

[3] http://project-sera.eu/

References

1. Cassell, J., Stocky, T., Bickmore, T., Gao, Y., Nakano, Y., Ryokai, K., Vaucelle, C., Vilhjálmsson, H.: MACK: Media lab Autonomous Conversational Kiosk. In: Proceedings of Imagina 2002 (2002)
2. Traum, D., Roque, A., Leuski, A., Georgiou, P., Gerten, J., Martinovski, B., Narayanan, S., Robinson, S., Vaswani, A.: Hassan: A Virtual Human for Tactical Questioning. In: 8th SIGdial Workshop on Discourse and Dialogue, pp. 71–74 (2007)
3. Traum, D.R., Bos, J., Cooper, R., Larsson, S., Lewin, I., Matheson, C., Poesio, M.: A model of dialogue moves and information state revision. Tech. rep., Deliverable D2.1, Trindi-project (1999)