Type-based Human-Computer Interaction

Peter Ljunglöf

Department of Computer Science and Engineering,
University of Gothenburg and Chalmers University of Technology, Sweden

A dialogue system is a computer system that can engage in dialogue with a human in order to complete tasks such as answering questions or performing actions. The information state update (ISU) approach [2] is a linguistically motivated theory of how to design dialogue systems. The approach is capable of advanced dialogue behaviour, and is based on a structured information state to keep track of dialogue context information.

The underlying logic of ISU-based systems tend to get very complicated, making it difficult to foresee the side effects of changing, adding or deleting inference rules. Ranta and Cooper [5] describe how a dialogue system can be implemented in a syntactical proof assistant based on type theory. Metavariables in the proof tree represent questions that needs to be answered by the user so that the system can calculate a final answer. However, the backbone of their theory is a very simple form-based dialogue system that cannot handle underspecified answers, anaphoric expressions, or ambiguous utterances.

In [3], we extended their theory with ideas from ISU and Dynamic Syntax [1], to make the system handle more flexible dialogues. Ranta and Cooper use *focus nodes* to know which of the nodes in the tree that is the current *question under discussion*. To this we add the idea of *unfixed nodes* from Dynamic Syntax. An unfixed node is a subtree which we know should be attached somewhere below a given node, but we do not yet know exactly where. We use unfixed nodes for representing *underspecified* answers, which is a very common phenomenon occuring in everyday dialogue between human participants.

In figure 1 the top node is the current focus, meaning that the system wants to build a tree of the type Action, which can be transliterated as the question "what do you want to do?". The user has given a partial answer to that question, by saying that (s)he wants to "go to London". This answer is of the type Route, which is not the correct type. Therefore the system adds this partial tree as an *unfixed* descendant of the focus node. The meaning is that in the final tree, the Action node will contain the Route tree as descendant.

We use a type-theoretical grammar formalism [4] to specify all aspects of the dialogue domain: tasks, issues, plans and forms, as well as the ontology of individuals, properties and predicates. We then make use of type checking for constraining the dialogue trees. Type checking is also used when interpreting user utterances and when providing the user with suggestions of what to say next.

The goal of the dialogue is to build a complete type-correct tree, and when this tree is completed, it represents a task which the user wants the system to perform in some way. In our theory the system tries to build the tree by successive refinement. In the middle of the dialogue, the uninstantiated parts of the tree are represented with metavariables. There is always exactly one of these metavariables that has focus.

The general idea of the dialogue manager is to ask the user to refine the current focus node. User utterances are then translated to (possibly incomplete) subtrees, which the system tries to incorporate into the dialogue tree. If the user utterance is of the same type as the focused metavariable, the tree can be extended directly. Otherwise the system will try to add the utterance as an unfixed node below the focus, or it will try to change focus to another metavariable that has a compatible type.

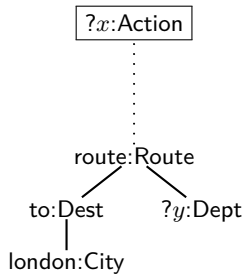

Figure 1: Example dialogue tree.

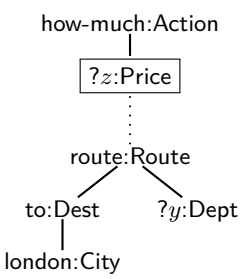

Figure 2: Refinement top-down.

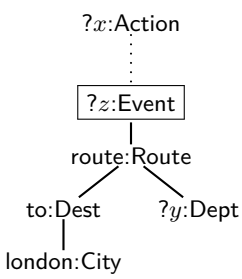

Figure 3: Refinement bottom-up.

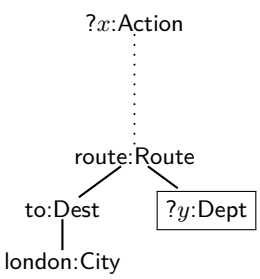

Figure 4: Refinement bottom-down.

There are at least three possible refinement strategies for deciding which node to select as the next focus. These strategies correspond to different dialogue behaviour:

- In *top-down* refinement (figure 2), the system tries to refine the focus node by finding possible children that can be ancestors to the unfixed trees.
- In *bottom-up* refinement (figure 3), the system tries to refine the unfixed node upwards by finding possible parents that can be descendants to the ancestor node.
- In *bottom-down* refinement (figure 4), the system tries to complete the unfixed tree first, and then finds its way up towards the ancestor node.

Issues that can be discussed within our proposed framework include the following:

- If the system is a question-answer (QA) system, the final complete tree specifies a question from the user, and the system can use type-theoretic function definitions for transforming the question into a suitable answer.
- Apart from unfixed nodes, we also borrow the idea of *linked trees* from Dynamic Syntax, and use them for sub-dialogues and anaphoric expressions.
- We have not yet incorporated dialogue feedback such as clarification questions and corrections into the theory. Feedback plays an importand role in dialogue, and it is necessary to be able to handle it in an advanced dialogue system.
- Ranta and Cooper borrowed the ideas of metavariables and focus nodes from the Agda proof assistant. Hopefully the new additions of unfixed nodes and refinement strategies can be borrowed back, and be used for making proof assistants more intuitive to use.

References

[1] Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell, 2001.

[2] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3–4):323–340, 2000.

[3] Peter Ljunglöf. Dialogue management as interactive tree building. In *DiaHolmia'09, 13th Workshop on the Semantics and Pragmatics of Dialogue*, Stockholm, Sweden, 2009.

[4] Aarne Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford, 2011.

[5] Aarne Ranta and Robin Cooper. Dialogue systems as proof editors. *Journal of Logic, Language and Information*, 13(2):225–240, 2004.