Putting Things in Context:  
Situated Language Understanding for Human-Robot Dialog(ue)

Robert Ross  
Artificial Intelligence Group, Dublin Institute of Technology, Ireland.  
robert.ross@dit.ie

Abstract  
In this paper we present a model of language contextualization for spatially situated dialogue systems including service robots. The contextualization model addresses the problem of location sensitivity in language understanding for human-robot interaction. Our model is based on the application of situation-sensitive contextualization functions to a dialogue move’s semantic roles – both for the resolution of specified content and the augmentation of empty roles in cases of ellipsis. Unlike the previous use of default values, this methodology provides a context-dependent discourse process which reduces unnecessary artificial clarificatory statements. We detail this model and report on a number of user studies conducted with a simulated robotic system based on this model.

Introduction  
Situated systems such as service robots pose two noteworthy challenges for current dialogue management techniques:  
1. Situation Sensitivity: In the situated domains, physical context dependencies are frequent as people make exophoric reference to objects in their environment, and describe processes in highly elliptical ways.  
2. Agency: Situated applications such as robotics are agentic in nature, and thus have complex internal mental states, operate in a semi-autonomous manner, and perform actions that have clear temporal extent.  

Agency features minimally require mixed-initiative and multi-threading in dialogues, but also a coupling of dialogue management with rational agency that recognizes the disparate, yet tightly bound, nature of these elements. In recent work, we have presented a model of Agent-Oriented Dialogue Management (AODM) to address the issue of agency and dialogue management coupling (Ross 2009). In this paper meanwhile, we focus on the topic of situational contextualization.

By contextualization we are referring to the process in which the content of a domain model is used to resolve or augment user specified content. For non-situated domains, contextualization minimally involves reference resolution against discourse and domain models. While a unification-based approach is often sufficient for contextualization in these domains, robotics and the situated domain requires a more complex contextualization strategy. Situated contextualization must account for exophoric reference resolution as well as the assignment of meaning to spatial constructs such as spatial semantics denoting prepositional phrases. Moreover, the contextualization process must also account for the augmentation of elided but spatially-retrievable information. For example, considering the example of a robot which can perform physical tasks for a user, the content of spatial utterances such as the following must be smoothly resolved and integrated into discourse state with a minimal amount of unnatural clarification statements on the robot’s part:

- **User:** go forward 5 meters
- **User:** no, turn to my right
- **User:** enter the room after the lab
- **User:** it is the second room on the right
- **User:** left
- **User:** a bit more
- **User:** go out of the room, take a left, and then head towards the stairwell

Unfortunately spatial language interpretation is in itself a complex task (Bateman et al. 2010). While linking of keywords to specific behavior parameterizations can provide a rudimentary form of language understanding, hard-coded mechanisms do not scale well, nor do they provide a natural mechanism by which understood language can be integrated into a discourse state such that the content of understood utterances can be correctly and naturally referenced later in interaction. While a number of situated interactive systems have considered the need for a contextualization process (Winograd 1972; Lemon and Gruenstein 2004; Kruijff et al. 2007) there is as yet little consensus on the mechanisms behind such a process.

In light of this, here we report on a situated contextualization model that we have developed specifically for the needs of human-robot interaction. We begin the main body of the paper by briefly introducing the dialogue management model within which the contextualization process is couched. The situated contextualization model is then detailed before we report on its application and evaluation in an implemented dialogue system.
Agent Oriented Dialogue Management

The AODM model is an Information State Update (ISU) based dialogue management model (Traum and Larsson 2003; Poesio and Traum 1998). Following the ISU approach, a Dialogue Game Board style organization of dialogue structure is made use of, as too is a clean separation of dialogue integration and planning processes. However, the AODM model rejects broad dialogue-plan based interfaces to domain applications, and instead takes an agent-oriented perspective on the organization of the domain application. In the following, we briefly summarize the principle agentive data types, dialogue data types, as well as core processes for the AODM model.

Agentive Types

Agentive types include traditional rational agency constructs such as beliefs, actions, plans, and intentions. Actions and plans are those capabilities that an agent is assumed to be endowed with, and are defined in the usual way in terms of an ontologically well specified signature. An intention on the other hand is a frame-like construct used as the primary unit of exchange between dialogue acts.

Dialogue Types

The agent-oriented perspective suggests the use of speech-act wrapped domain capabilities as the natural units of content communication between system and user. In natural communication however, such a ground move is the result of the communication management process rather than a product of perception. Thus, following the approaches to dialogue structure originally proposed by (Butler 1985) and later (Poesio and Traum 1998), we assume the Dialogue Act (act) as the primary unit of exchange at the surface/semantics interface, while assuming the Dialogue Move (move) as the coarse-grained unit of interaction established through grounding and contextualization at the semantics/pragmatics interface.

The act reflects a traditional pragmatic view of communicative function, and is used to model explicit contributions to a dialogue. All acts have an associated speech function, but not all acts need have an associated content. If present, an act’s content is defined in terms of the agent’s conceptual ontology. The move on the other hand is a frame-like construct used as the primary unit of exchange between dialogue management and agency processes. The licensed content of a move is directly coupled to the agent’s range of capabilities and potential mental states. Due to its use as a staging ground in meaningful update composition, we necessarily model the move in terms of three components:

1. **The Move Template**: defines the move type and content potential in terms of concept and role definitions extracted from a conceptual ontology.
2. **The Move Filler**: is the shallow descriptions provided by the user to fill out the roles in the move template.
3. **The Solution Set**: is the set of possible interpretations of the move filler following contextualization. While solution contents are defined in terms of the agent’s application ontology, solution contents also have associated interpretation likelihoods, and typically includes content which was not directly provided by the speaker.

Notable distinctions between the dialogue act and the dialogue move include: (a) the narrow range of speech function types for which a dialogue move may be composed in comparison to a dialogue act; and (b) the limitation that the propositional content of a move must be headed by an application state or capability, whereas an act’s content can be any consistent selection from the application ontology.

The AODM dialogue management process can be viewed as the mapping of dialogue acts in and out of dialogue move structures. In the following, we briefly expand on the dialogue management process.

Dialogue Management Processes

Following the ISU methodology as well as other contemporary models, the dialogue management process is split between a number of distinct sub-processes which monitor and manipulate the information state. Specifically, the following sub-processes are called cyclically:

- **Act Recognition**: Identification of user dialogue acts through perception, shallow analysis, and modality fusion when available.
- **Act Integration**: Integration of user dialogue acts into the information state. Successful integration of task-specific acts involves the update of new or already open dialogue moves.
- **Move Contextualization**: Directly following integration, all open user moves are contextualized against the current situational model to resolve anaphoric (in the general sense) references, elided content, and ambiguous features such as reference frame use.
- **Response Planning**: Examination of the information state – including open user and system dialogue moves – to determine what, if any, actions should be taken by the agent. Actions include the adoption of dialogue act goals, as well as the adoption of intentions in response to successfully contextualized user dialogue moves.
- **Intention Management**: A non-dialogic process which manages capability sequencing at an intentional level.
- **Message Planning**: Compose surface language contributions for planned dialogue acts.

Details on each of these processes are presented in Ross (2009). In the remainder of this paper, we focus on the language contextualization stage.

Dialogue Move Contextualization

In this section we describe the dialogue move contextualization process which we have developed to account for referent resolution, action interpretation, and elipsis resolution in human-robot dialogue. The responsibilities of the language contextualization process are essentially to take the shallow...
A solution parameter, present. Conversely, a solution is complete if one or more values for a minimal parameterization is not provided. We informally define a solution, \( \sigma \), as a set of values that parameterize a move specification. A solution is partial if one or more values for a minimal parameterization is not present. Conversely, a solution is complete if the solution set provides a minimal parameterization for the given move. A solution parameter, \( \psi \), is an entity defined in terms of the agent’s domain ontology. In general, any given contextualization function may return multiple possible values for a given parameter. Since these values will typically not all be equally likely, a likelihood is assumed assigned to each interpretation returned by a contextualization function.

The interpretations provided by any given contextualization function are dependent on both the shallow semantic categories provided through surface language (in the case of resolution) as well as the state of the interpreting agent. Such a state, \( s \), is assumed to be application dependent, but can be assumed to include a description of the agent’s physical as well as mental condition. While the interpreting agent has a real-world state, the state applicable to a given contextualization function may become disjoint from this state. For example, in the case of interpreting a move complex with a sequential relationship between two constituent moves, the second move is only interpretable in the context of the state set up by the interpretation of the first move. Moreover, even within the contextualization of a single move, the state applicable to a contextualization function can become disjoint from the real-world state of the interpreting agent. For example, in contextualizing a move which describes a physical process with two motion defining constraints, the second constraint can only be contextualized after the first constraint has been contextualized to derive an input state for the second constraint.

Moreover, rather than assuming that the application of a move results in a unique state, we assume that the performance of a capability can result in a distribution of possible states. This is particularly true for the case of spatial actions where a definite final state may not be known even for an unambiguous action description. We thus associate a distribution of possible states with a given move solution, and allow this state distribution to be manipulated by augmentation and resolution functions. In the case of a single move interpreted out of a discourse context, this state distribution variable is instantiated based on the agent’s current state. Whereas, in the case of a move being interpreted in the context of a move complex, a prior move’s final state distribution is applied as initial state for the secondary move.

Given each individual solution parameter, \( \psi \), has an associated likelihood \( P(\psi) \), we assign a solution’s total parameter likelihood as the product of individual parameter likelihoods. In the case of multiple possible solutions for a move, we can make use of this likelihood product to decide how to progress the interpretation or dialogue process forward. However, in the case of multiple solutions with equivalent or almost equivalent interpretation likelihoods, a second metric, the solution cost \( \epsilon \) is applied to help choose between solutions. This metric values the cost of performing the capability being raised by the given move. Rather than being associated directly with the solution itself, a specific cost is provided for each of the final states associated with a solution. The cost, \( \epsilon \), is thus assumed to be a function of both the individual solution parameters and a likely outcome state associated with that solution. Costs are assumed to be scalar and, like resolution and augmentation functions, are move type dependent.

In summary, a given solution contains both a set of parameter fillers along with their likelihoods, as well as a resultant state distribution for the given set of parameters. Specifically, we define a solution, \( \sigma \), for a given move, \( \mu \), as a 2-tuple:

\[
\sigma = \{ \Psi, S \}
\]

where \( \Psi \) is the ordered set of parameter assignments for the given solution, and \( S \) is the distribution of possible resultant states. These two values are in turn defined as follows:

\[
\Psi = \text{List}(\psi, P(\psi))
\]

\[
S = \text{Set}(s, P(s), \epsilon)
\]

where \( \psi \) is a parameter value, \( P(\psi) \) is a likelihood assigned to that parameter by a contextualization function, \( s \) is a state value, \( P(s) \) is a likelihood for that state value assigned by the contextualization function, and \( \epsilon \) is a solution cost.

Since any given resolution or augmentation function can introduce more than one parameter interpretation, multiple solutions may be maintained at any given time during interpretation. We thus introduce the solution set, \( \Sigma \), as the collection of solutions maintained for a given move:

\[
\Sigma = \text{Set}(\sigma)
\]

where the size of the solution set can be retrieved directly when necessary from \( \Sigma.\text{size} \).
The generalized contextualization process is the operation by which individual moves are contextualized against the situational state. Move contextualization can be viewed as a two step process:

1. **Contextualization Function Application:** For each of a move’s parameters, an associated augmentation or resolution function is selected and applied within the context of a specific state. One or more possible interpretations of that parameter along with associated state distributions are then used to update the solution set before the next parameter is similarly processed.

2. **Solution Set Analysis:** Following application of contextualization functions to all move parameters, the resultant solution sets are analyzed to determine overall likelihood.

Figure 1 gives pseudo-code for the single move contextualization process. The input to the process includes: (a) the move to be contextualized, \( \mu \); (b) the set of augmentation functions, \( \Delta \); (c) the set of resolution functions, \( \Psi \); and (d) the initial solution set, \( \Sigma \). As mentioned earlier, depending on whether a move is contextualized in the context of a move complex or not, the input initial solution set may be initialized to either the output solutions of the predecessor move, or to a new empty solution which consists only of an associated state that has been initialized to the agent’s current real-world state.

Referring to Figure 1, during contextualization the solution set \( \Sigma \) is updated for each contextualized parameter in \( \mu \). This solution set is first initialized to the input solution set, and then used as a basis for creating a revised solution set during the processing of each parameter. The contextualization process subsequently iterates through each of the move parameters and selects an augmentation or resolution function depending on whether the user supplied content for a given move parameter or not. This selection mechanism operates through the association of exactly one resolution and augmentation function with each parameter in a given move specification. Once selected, the augmentation or resolution function is then applied in the context of each currently held solution. Contextualization functions are assumed to return zero or more possible solutions. These solutions extend the input solution to which they were applied.

Following basic contextualization, the costs and likelihoods of solutions contained in the solution set, \( \Sigma \), are analyzed to aid subsequent dialogue planning and intention adoption strategies. First, the mean action cost for a given solution, \( \epsilon(\sigma) \), is calculated as an average of the cost values associated with each final state in a given solution \( \sigma \), i.e.:

\[
\epsilon(\sigma) = \frac{\sum_{j=0}^{\sigma_i.S.size} \sigma_j.S_.e}{\sigma_i.S.size}
\]  

(5)

The total parameter likelihood for a given solution, \( \Pi(\sigma) \), is then calculated as the product of individual parameter likelihoods, i.e.:

\[
\Pi(\sigma_i) = \prod_{w=0}^{\sigma_i.S.size} \sigma_i.P(\psi)\_w
\]  

(6)

Finally, the overall solution likelihood, \( \kappa(\sigma) \), is calculated as a product of the total parameter likelihood, \( \Pi(\sigma) \), and a normalized efficiency estimate based on the average cost values across all solutions, i.e.:

\[
\kappa(\sigma_i) = \Pi(\sigma_i) \cdot \left(1 - \frac{\sum_{j=0}^{\sigma_i.S.size} \epsilon_j}{\sum_{j=0}^{\sigma_i.S.size} \epsilon_j}\right)
\]  

(7)

The solution likelihood measure is thus dependent on both the actual likelihood of individual parameter interpretations and the cost of the likely resultant action. This solution likelihood measure contributes to determining the task progression strategy in the subsequent dialogue planning process. Namely, even if multiple solutions result from the interpretation of a given move, if the likelihood measure of one solution is notably greater than for the next most likely solution, then choosing the most likely solution without the need for direct clarification is possible. Even in the case that these solutions are similar in likelihood, it is essentially not important which interpretation is made if the two solutions share a similar final state.

The process just outlined describes the contextualization of individual moves. For contextualizing move complexes, i.e., constructions of atomic dialogue moves that capture more complex instructions, the basic process must be extended. Here we briefly comment on the extended process, but omit a more complete description due to space limitations.

The move complex contextualization process builds upon the individual move contextualization process, but also addresses asserted logical relation between moves – or rather
the implied relation between resulting intentions. Individual moves are contextualized in order—however, if a sequence of actions is implied, each solution of move i is used to instantiate a solution for move i+1. Meanwhile, if the implied relationship is a disjunction, then no such pass through of solutions is applied. Having processed the complete move complex in a linear order, best final solutions are then back-propagated through the trace of solution sets to eliminate ambiguous early moves that may have been clarified by subsequent dialogue moves.

**Application & Evaluation Studies**

The model outlined in the last section was generalized to avoid considering the specifics of augmentation and resolution functions. Such functions are of course domain specific. An implementation of the generalized contextualization process has however been instantiated as part of a complete ISU-based dialogue system for human-robot interaction (Ross 2009; Tenbrink et al. In Press). Contextualization functions for moves in this domain address both spatial and non-spatial semantic roles. Non-spatial roles include features such as actor, temporal placement, and so forth. Spatial functions on the other hand interpret semantic roles such as placement, direction, movement trajectory, extent and so forth. These spatial contextualization functions provide the essential mapping between linguistic-semantic categories found in dialogue move instances, and the non-linguistic spatial representation and reasoning models that account for the relative nature of spatial knowledge.

The spatial contextualization functions used are based on the notion of spatial templates such as the Attention Vector Sum model (Regier and Carlson 2001). Such templates provide a mapping between situated interpretation of spatial terms and their symbolic counterparts. We cannot consider the details of these contextualization functions here. However, Figure 2 provides a short worked example of the interpretation of a single move for this domain. We assume an instructor instructs an instructee to “turn” as the instructee reaches a t-junction. Since all but a verb indicating the process was omitted, augmentation functions, e.g., $\delta_{\text{actor}}$, are applied in succession to construct a solution for the under-specified dialogue move. In this highly simplified example, only one possible solution is maintained through to the augmentation of the placement role. However, since two turning direction possibilities are available at the t-junction, augmentation of the direction role results in two distinct solutions for the move being introduced in the last augmentation step. If the instructor provides further information on
this dialogue move, or indeed indicates a successor dialogue move, then this solution set may be reduced. However, in the event that the instructor passes the dialogue turn, a clarification question is subsequently composed by the dialogue planning process on the basis of this solution set.

With implemented contextualization functions, the language understanding approach has been implemented to construct contextually ground interpretations for spatial instructions such as those illustrated in the introduction. Coupled with a dialogue management strategy provided by an information-state methodology, this results in a robust and natural style of interaction for human-robot dialogues. As indicated, the AODM model and the contextualization process reported on here, has been fully implemented and evaluated in the context of a research project investigating features of situated language and dialogue phenomena in human-robot interaction for both German and English speakers (Tenbrink et al. In Press; Ross 2009). Most recently, Tenbrink et al. report on an evaluation of a simulated text-based dialogue enabled wheelchair system based on the contextualization and dialogue model presented here. While it was found that limitations in parser coverage for complex spatial constructions continue to be a sizable challenge, results obtained with a group of 21 native German speakers found that task performance and timing was comparable to that of a human-human gold standard (Tenbrink et al. In Press).

Related Work
Our function-based approach to spatial language contextualization shares motivations and some basic assumptions with work by Winograd (1972) and more recently Tellex & Roy (2007). Whereas Winograd focused on very idealized domains we have attempted to push the basic notion of function-based contextualization to more complex domains and integrated it with a complete discourse model. Similarly, whereas Tellex & Roy focused on the resolution of explicit language in a monologue setting, we have applied a function-based strategy to both resolution and augmentation in a full dialogue setting. The AODM model as a whole is also similar in approach to Gruenstein and Lemon’s Conversational Intelligence Architecture (Lemon and Gruenstein 2004). Specifically, both models advocate a tight coupling between dialogue management and agency features – although in our work we moved from the use of static defaults towards a function-driven contextualization process which dynamically looks to the situated domain in language resolution and augmentation.

Conclusions & Outlook
In this paper we presented a language contextualization process to compliment existing language integration and dialogue planning models in dialogue management for human-robot interaction. Though the model only touches on some of the issues relevant to further developing the semantics/pragmatics interface, we believe models like this to be significant in exploring the relationship between linguistic and extra-linguistic processes in situated intelligent systems. The necessarily integrated nature of this contextualization process however poses a challenge for evaluation. Therefore in future work we aim to provide a specific testbed which allows the contextualization process to be replaced with, and hence benchmarked against, alternative contextualization strategies.

Acknowledgments The author gratefully acknowledges the support of the DFG-funded Research Center on Spatial Cognition at the University of Bremen, and the Irish Research Council for Science, Engineering and Technology.

References