AN INTELLIGENT INTERFACE PROCESSOR FOR A BEHAVIOUR-BASED CONTROL ROBOT

Shaidah Jusoh * **Fangju Wang** ** **Simon X. Yang** *

* *School of Engineering*
** *Dept. of Computing and Information Science*
University of Guelph, N1G 2W1 Ontario, Canada

Abstract: A behaviour-based control is considered as the best approach to control autonomous robots. In the near future, autonomous robots like mobile service robots are expected to assist people in a common environment such as houses or offices. Such situations require natural interaction between people and the robots. One way to facilitate this is by using a natural language interface (NLI). Unfortunately the major problem with natural language is that it is always ambiguous. Up to now, there is no existing NLI processor that can well solve the ambiguity problems for human-robot interface. This paper presents a new methodology for creating an intelligent NLI processor for a mobile service robot that uses a behaviour-based control. The methodology uses a fuzzy approach and history knowledge. The history knowledge is an innovation in this work.

Keywords: natural language interface, behaviour-based control, fuzzy approach, ambiguity.

1. INTRODUCTION

In previous years, robots have become the most suitable tools in dangerous workplaces like a nuclear plant. However, in recent years, research on robotics has been focused on mobile service robots. These robots are expected to assist ordinary people and perform their tasks in human living areas such as in offices, houses, hospitals and so on. The existing development of this type of robots are: an office secretary robot (Asoh *et al.*, 2001), a purchaser robot that buys a cup of coffee (Nakauchi and Simmons, 2002) and a tour guide robot in a museum (Burgard *et al.*, 1999). The mobile service robots are autonomous robots. Nowadays, a behaviour-based control is seen as the best approach to control autonomous robots (Arkin, 1998; Hasegawa and Fukuda, 1999). The architecture of the approach can be divided into three levels: the highest, middle, and lowest. The highest level concerns with task-oriented behaviour, the middle with an obstacle-avoidance behaviour, and the lowest with an emergency behaviour. As the mobile service robots are expected to assist human users, the users should be able to instruct the robot to perform duties. An instruction from a user is defined as a user-task behaviour. Thus, a human-robot interface will be required to convert from a user-task behaviour into a task-oriented behaviour.

The interface is considered flexible if it allows a human user to convey the user-task behaviours in a natural language. This feature can be facilitated through the use of natural language interface (NLI). However, an NLI for a robot is more complex than a regular computing system, because the robot perceives its world and acts to

a sequence of user-task behaviours. The major problem of NLI is almost all natural language sentences are ambiguous. In a real life, people reduce this ambiguity by using their cognitive such as they recall what had happened in the past and so on. The NLI is considered as an intelligent interface if it is able to resolve the ambiguity problem by checking what had occurred in the past.

To the best of our knowledge, no attempt has been made to convert an ambiguous user-task behaviour into an ambiguous task-oriented behaviour, and NLIs for robots are still immature. Thus, a new methodology for creating an intelligent NLI processor that is able to convert an ambiguous user-task behaviour into an unambiguous task-oriented behaviour is presented in this paper. The new methodology is an integration of fuzzy grammar and history knowledge. This paper is organised as follows. Section 2 gives examples of user-task behaviours. Section 3 gives a brief background of natural language understanding. Fuzzy approach and history knowledge are presented in Section 4. The semantic interpretation is discussed as well. A summary and implementation issues are discussed in Section 5.

2. USER-TASK BEHAVIOUR

Two scenarios below demonstrate user-task behaviours that should be converted into task-oriented behaviours. As user-task behaviours are represented in sentences, for the rest of this paper, a *sentence* term is used in representing a user-task behaviour.

Scenario A
An old lady is sitting on a chair and a robot is in its position. Suddenly, the old lady feels that she would like to have a cup of coffee
Lady: *"I: make a cup of coffee".*
The robot moves toward the target, that is a coffeemaker. The lady sees the robot's action.
Lady: *"II: bring it here"*

Scenario B
An old lady finds a dirty plate in the sink and requires the robot to wash it.
Lady: *"I: wash the plate in the sink".*
The robot moves toward the target (sink) and performs its duty. The lady sees the robot's actions.
Lady: *"II: store it into the cabinet"*
The robot stores the plate into the cabinet

In examples above, only sentence *I* in the **Scenario A** is not ambiguous. The sentence *wash the plate in the sink* is ambiguous because it can be understood as either *the plate should be washed in the sink* or *the plate in the sink should be washed*. Both of sentences *II* in both scenarios are ambiguous because of the pronoun *it*. Fig. 1 illustrates that natural language (NL) processor uses history knowledge to resolve ambiguity in sentences to create unambiguous task-oriented behaviours. Only the top application is concerned in this paper.

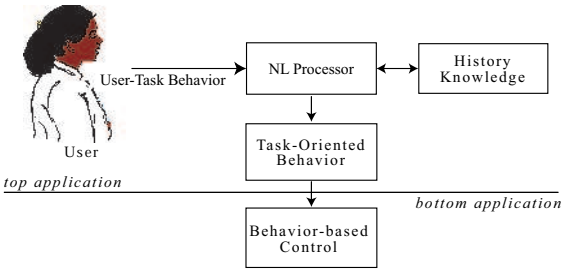

Fig. 1. Generating an unambiguous task-oriented behaviour

3. NATURAL LANGUAGE UNDERSTANDING

3.1 Syntactic Processing

Syntactic processing is a process of assigning a parse tree to a sentence. Syntactic processing requires a grammar. The grammar is a formal specification of the structures allowable in the language and it is usually represented as $G=(V_N, V_T, P, s)$. The symbols are explained below.

- V_N: a set *of non-terminal* symbols that do not appear in the input strings, but are defined in the grammar. Examples of *non-terminal* symbols are: *sentence (S), imperative sentence (IS), Noun Complement List (NCL), Noun Complement (NC), verb phrase (VP), noun phrase (NP)*, and *prepositional phrase (PP).*
- V_T: a set of *terminal* symbols that are primitives or classes of primitive symbols in input strings. Examples of *terminal* symbols are: *Noun, Verb, Preposition, Adjective, Determinant (Det)*, and *Adverb.*
- P: a set of *production rules*, each of the form $A \rightarrow \beta$, where α is a *non-terminal* symbol and a string of symbols from the infinite set of strings in $(V_T \cup V_N)^*$
- s: a starting symbol

Let α_1 be a string in $(V_T \cup V_N)^*$ and $\alpha \rightarrow \beta$ be a rule in P. If α is in α_1, we can obtain a new string from α_1 by substituting α with β. The new string is also a string in $(V_T \cup V_N)^*$. (The symbol * indicates a free manoid X* over the set X). Now, let α_2 denotes the new string. α_2 is said to be *derivable* from A_1 in G.

The derivation can be expressed as $\alpha_1 \rightarrow \alpha_2$. Let $\alpha_1, \alpha_2, \alpha_3,.., \alpha_m$ be strings in $(V_T \cup V_N)^*$ $(M \geq 2)$. If the are $\alpha_1 \rightarrow \alpha_2, \alpha_{m-1} \rightarrow \alpha_m$, then α_m is said to be derivable from α_1 in G. The sequence of derivations $\alpha_1 \rightarrow \alpha_2, ..., \alpha_{m-1} \rightarrow \alpha_m$ is referred to as a *derivation chain* from α_1 to α_m. A grammar G defines a *language* $L(G)$. A string s is a valid sentence in $L(G)$, if and only if $s \Longrightarrow S$.

3.2 Semantic Processing

Semantic processing is a process of converting a parse tree into a *semantic representation* that is precise and unambiguous representation of the meaning expressed by the sentence. Semantic processing can be conducted in two steps: *context independent interpretation* and *context dependent interpretation*. Context independent interpretation concerns what words mean and how these meanings combine in sentences to form sentence meanings. Context dependent interpretation concerns how the context affects the interpretation of the sentence. The context of the sentence includes the situation in which the sentence is used, the immediately preceding sentences, and so on.

4. NATURAL LANGUAGE PROCESSOR

4.1 Fuzzy Approach

Fuzzy approach is used by extending a regular grammar into a fuzzy grammar. In *a fuzzy grammar*, the same *terminal* and *non-terminal* symbols as in a regular grammar is used, and an additional *label*, called a *plausibility function* is added in each grammar rule. A *regular grammar* rule is presented as

$$\alpha \rightarrow \beta \tag{1}$$

where α is a *nonterminal* symbol, β is a *nonterminal* or *terminal* symbol, or both symbols and $\rightarrow$ is a *production rule*. In a fuzzy grammar, a rule is modelled as

$$\alpha \xrightarrow{\rho} \beta \tag{2}$$

where ρ is a *plausibility function* in each fuzzy grammar rule, and $\rho \in [0, 1]$ indicates the plausibility for substituting α with β in a parsing process.
A fuzzy grammar (G) generates a fuzzy language $L(G)$ in the following manner. A string S of symbols in V_T is said to be in the language $L(G)$ if and only if $s \rightarrow S$, i.e. S is derivable from s. When Tr is a parse tree generating S, the plausibility of Tr is

$$min\{\mu(s \rightarrow \alpha_1), ..., \mu(\alpha_m \rightarrow S)\} > 0 \tag{3}$$

where $s \rightarrow \alpha_1, \alpha_1 \rightarrow \alpha_2, ..., \alpha_m \rightarrow S$ is the derivation chain from which Tr is constructed, and $\mu(\alpha_i \rightarrow \alpha_{i+1})$ is the non-zero $\rho i + 1$ such that $(\alpha_i \overset{\rho(i+1)}{\rightarrow} \alpha_{i+1}) \in P \ \forall i = 0, ..., m$,
if $\alpha_0 = s$ and $\alpha_{m+1} = S$. The restricting fuzzy set F_s is defined as

$$F_s = \{Tr\} \tag{4}$$

and its membership function is

$$\mu_{Fs}(Tr) = \begin{cases} min\{\mu(s \rightarrow \alpha_1), ..., \mu(\alpha_m \rightarrow S)\} \\ \text{ if } s \rightarrow_{Tr} S \\ 0 \text{ otherwise,} \end{cases} \tag{5}$$

where $\rightarrow_{Tr}$ is the chain $s \rightarrow \alpha_1, \alpha_1 \rightarrow \alpha_2, ..., \alpha_m \rightarrow S$ from which Tr is constructed. As previously mentioned, the sentence may be ambiguous. When the sentence is ambiguous, the syntactic processor can generates more than one Tr. The restricting fuzzy set F_s for a group of Tr can be defined as

$$F_s = \{G_{Tr}\} \tag{6}$$

and its membership function is

$$\mu_{Fs}(G_{Tr}) = \{max(Tr_1, ..., Tr_n)\} \tag{7}$$

In using a fuzzy grammar, it is important to define ρ for each grammar rule. In this work, for rule $\alpha \xrightarrow{\rho} \beta, \rho$ is defined as a function of the symbols in β and the symbols that have been processed. ρ is context dependent. Consider a *wash the plate in the sink* sentence is parsed. Using the fuzzy grammar mentioned above, the *syntactic processor* generates two parse trees (Fig. 2) with its plausibility values as shown in Table 1. By using Equations (5) and (7) consecutively, the processor decides that parse tree (B) as a unique parse tree.

Table 1. Generated rules and its plausibility values for each parse tree (parse tree (A) and parse tree(B))

Rules(A)	*I*	*Rules(B)*	*II*
$S \rightarrow IS$	1.0	$S \rightarrow IS$	1.0
$IS \rightarrow VP$	1.0	$IS \rightarrow VP$	1.0
$VP \rightarrow Verb\ NP\ PP$	0.4	$VP \rightarrow Verb\ NP$	0.8
$Verb \rightsquigarrow wash$	1.0	$Verb \rightsquigarrow wash$	1.0
$NP \rightarrow Det\ Noun$	0.8	$NP \rightarrow Det\ Noun\ NCL$	0.6
$Det \rightsquigarrow the$	1.0	$Det \rightsquigarrow the$	1.0
$Noun \rightsquigarrow plates$	1.0	$Noun \rightsquigarrow plates$	1.0
$PP \rightarrow Prep\ NP$	0.7	$NCL \rightarrow NC$	1.0
$Prep \rightsquigarrow in$	1.0	$NC \rightarrow PPL$	0.7
$NP \rightarrow Det\ Noun$	0.8	$PPL \rightarrow PP$	0.5
$Det \rightsquigarrow the$	1.0	$PP \rightarrow Prep\ NP$	0.7
$Noun \rightsquigarrow sink$	1.0	$Prep \rightsquigarrow in$	1.0
		$NP \rightarrow Det\ Noun$	0.8
		$Det \rightsquigarrow the$	1.0
		$Noun \rightsquigarrow sink$	1.0

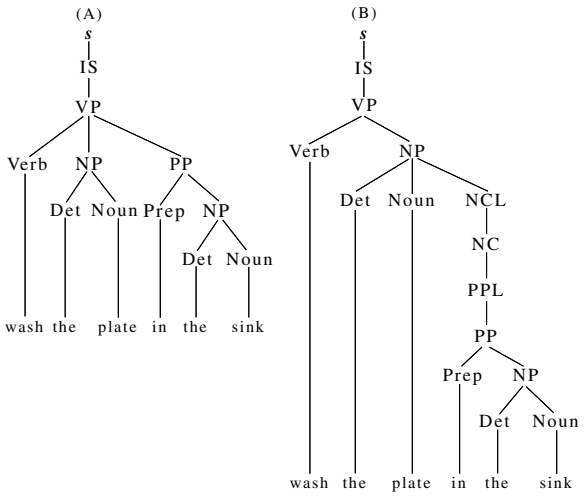

Fig. 2. Two parse trees for an ambiguous sentence. Parse (A) corresponds to the meaning that *the plate should be washed in the sink*, and parse (B) to *the plate in the sink should be washed*.

4.2 History Knowledge

History knowledge is a sequence of user-task behaviours that occurred in the past. For each of user-task behaviours, a unique parse tree is obtained and stored in the history database dynamically. Technically, the history database keeps three types of information: unique parse trees, a sequence number, and priority values.

Unique Parse Trees
A stack (*st*) of unique parse trees of all user-task behaviours are kept in a history database as history knowledge. For an example, when a user gives a sentence *wash the plate in the sink*, a parse tree (B) in Fig. 2 is considered as a unique one, thus it is kept in the *st*. If a user gives n sentences, where n is a number of required user-task behaviours to complete a certain duty, then n parse trees will be stored in each *st*. If there is a number of duties that the robot can perform, there will be a number of *st* in the database.

A sequence number
For each of the parse trees in the history database, it is given a sequence number that can be denoted as st_x, in which x represents the ordering number of the parse trees in the stack. For an example, the first parse tree in the stack (*st*) can be labelled as st_1, and if there is n parse trees in the stack, thus the last parse tree can be labelled as st_n.

Priority values
For each of the parse tree in the *st*, if the parse tree contains constant nouns, each of the constant nouns will have its priority value t. The value t gives a degree of priority of the constant noun. The degree of t is decided based on its *step* in the parse tree. The syntactic processing starts from s which is a *zero* (0) *step*. The top of the parse tree is s. The 0 *step* is considered as the highest step. For example, in the parse tree (B), the constant noun *plate* has higher *priority* than the constant noun *sink*, because it appears in the higher step.The process of assigning a priority value to the constant noun is conducted during the syntactic processing. As previously mentioned, a fuzzy grammar rule is represented as in Equation 2. To assign a priority value, the fuzzy grammar rule is extended into

$$\alpha \xrightarrow{\rho} \beta_t \tag{8}$$

where t is a *priority function*. The priority function is activated only when a grammar rule of $\alpha \xrightarrow{\rho} \beta_t$ which α matches a *terminal symbol* $Noun$ and β a constant noun such as *plate*, is recognised by the *processor*. For example, when the processor replaces the Equation 8 with the following rule

$$Noun \xrightarrow{\rho} plate_t, \tag{9}$$

the value of t is calculated. When the processor starts the syntactic processing, it assigns an initial value of t as t_s, where $_s$ represents a starting point of t value. When the first grammar rule of $\alpha \xrightarrow{\rho} \beta_t$ is recognised, the t calculation can be conducted by using the following formula

$$t_1 = t_s - \theta \tag{10}$$

where θ is a constant value that is decided by the processor's designer. Therefore, the process of calculating t values can be formalised as

$$t_1 = t_s - \theta, t_2 = t_1 - \theta, ..., t_m = t_{m-1} - \theta \tag{11}$$

where $_m$ is the maximum number of noun constants in the sentence, if and only if $_{1,2,..,m}$ noun constants are attached at the different levels of subparse tree. However, if there is q *noun* constants attach to the same step in a parse tree, the process of calculating t values can be formalised as

$$t_q = t_m - \theta, \tag{12}$$

where a value of q is {m+1, m+2, ...,m+n}.

To understand the usage of history knowledge, a new sentence *store it into the cabinet* which represents the second user-task behaviour is parsed. Although the syntactic processor is able to generate a complete parse tree as illustrated in (i) of Fig. 3, the parse tree is still cannot be considered as an unambiguous one. This occurs because the pronoun constant like *it* causes ambiguity in the parse tree.

To resolve this ambiguity problem, the processor has to use history knowledge. The processes of using history knowledge and resolving the ambiguity are described in the following procedures:

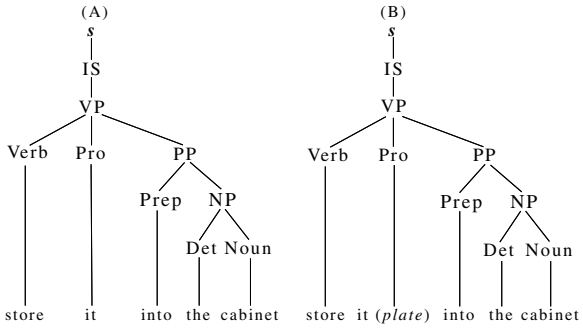

Fig. 3. Parse trees for the *store it into the cabinet* sentence.

(1) Identify the most recent parse tree.
In order to recognise what a pronoun constant *it* is referring to, the processor look up into parse trees in *st*. To determine the most recent parse tree, the processor uses an *ordering* number as a tool. If there is n parse trees in *st* and if the *st* starts counting from 1, thus the *ordering* of *st* for the stack would be $st_1, st_2, ..., st_n$. The processor can determine that the parse tree of st_n is the most recent parse tree.

(2) Identify the highest degree of priority.
For each of the parse trees, it has a set of priority values. The restricting fuzzy set F_s for a set of priority values can be defined as

$$F_s = t_i \tag{13}$$

and its membership function is

$$\mu_{Fs}(t_i) = \{max(t_1, ..., t_m)\}. \tag{14}$$

By using the Equation 14, the processor is able to identify the highest priority value of the most recent parse tree.

(3) Identify the noun constant/constants.
The processor then finds a noun constant or noun constants that its priority value matches to the highest priority value. Once a constant or constants has/have been identified, the processor modifies the complete parse tree by attaching the constituent noun or nouns to the constituent pronoun as illustrated in parse tree (ii) in Fig. 3. When this process has completed, the processor stores the unique parse tree into the dedicated *st*.

4.3 Semantic Interpretation

Once a unique parse tree is obtained, the processor is able to process the parse tree semantically. In this work, a λ reduction technique (Jurafsky and Martin, 2000) is used for the purpose of semantic interpretation. The λ reduction technique has been used as an extension of First Order Predicate Calculus (FOPC). FOPC is the most commonly used method for representing semantic attachments; that is, semantic attachments are expressed as FOPC predicates. A FOPC predicate may contains a sequence of constants, variables, and functions that are connected by connectives and quantified by quantifiers. A constant refers to a specific object in the world such as a noun like "water". A variable represented as a symbol x is used to infer about the object without having to make reference to any particular named object. Syntactically, it is the same as a single argument predicate. In semantic attachment, each grammar rule is attached with a semantic expression:

$$\mathcal{A} \longrightarrow \alpha_1, ...\alpha_n\{\{(\alpha_1.sem, ...\alpha_n.sem)\} \tag{15}$$

where the expression in $\{\}$ is the semantic attachment, $\alpha_i.sem$ is the semantics of the i constituent in this rule, and $\{$ is a function that constructs the semantics of $\mathcal{A}$ out of the semantics of the constituents on the right hand side of the rule.

The semantics for constituents using λ expressions are represented. An important operation in the semantic attachment technique is the replacement of the variable by the constants that have been evaluated. The λ *reduction* technique is used for such replacement operations. Syntactically, λ reduction is represented by λ notation that extends the syntax of FOPC to include expressions of the following form:

$$\lambda x \mathcal{F}(x) \tag{16}$$

where $\mathcal{F}$ is a predicate containing one or more variables x. The generic form of the λ reduction is

$$\lambda x \mathcal{F}(x)(X) = \mathcal{F}(X) \tag{17}$$

where X is a constant. Informally, when the λ reduction technique is used, a grammar rule and its semantic attachment can be:

$$\mathcal{A} \longrightarrow \alpha_1, ...\alpha_n\{\lambda x_1, ..., \lambda x_m \mathcal{F}(x_1, ..., x_m)\} \tag{18}$$

where $x_1, ..., x_m$ are variables, $\mathcal{F}$ is a predicate containing the variables. It is assumed that a sentence has been parsed into a parse tree, in which the $\mathcal{A}$ in the above rule is the root of a sub-parse tree. In semantic analysis of the instruction (represented by the parse tree), the semantics of the sub-parse tree rooted by $\mathcal{A}$ is calculated by

$$\lambda x_1, ..., \lambda x_m \mathcal{F}(x_1, ..., x_m)(X_1), ..., (X_m) \tag{19}$$

where $X_1, X_2 ... X_m$ are constants or predicates that do not contain variables, which have been calculated in analysing the sub-parse trees rooted by α_1,α_2,...α_n when the semantics of sub-parse tree rooted by $\mathcal{A}$ analysed.

The process of semantic attachment takes place when a a unique parse tree is obtained. The semantic attachment for parse tree (B) of the Fig. 2 is illustrated in Fig. 5. In this case, we ignore a determinant such as *the* because it does not give a significant meaning in the semantic interpretation. Fig. 5 illustrates how the *terminal symbol* **Pro** has a semantic attachment of *plate*.

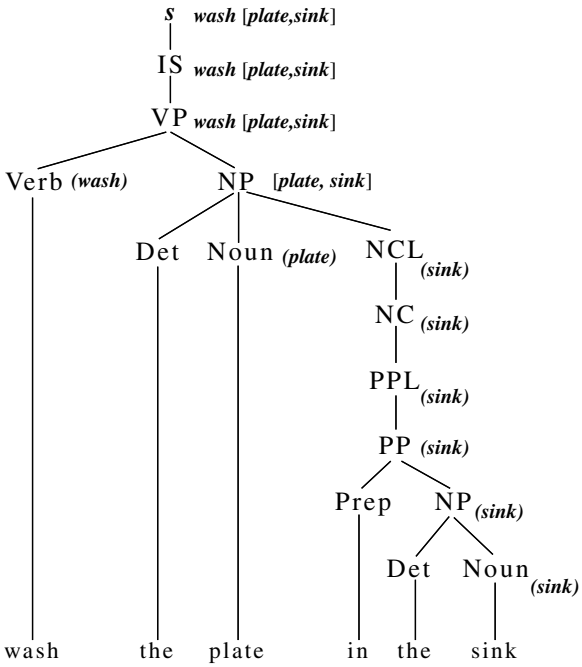

Fig. 4. A semantic attachment for parse tree (B) of the Fig. 2

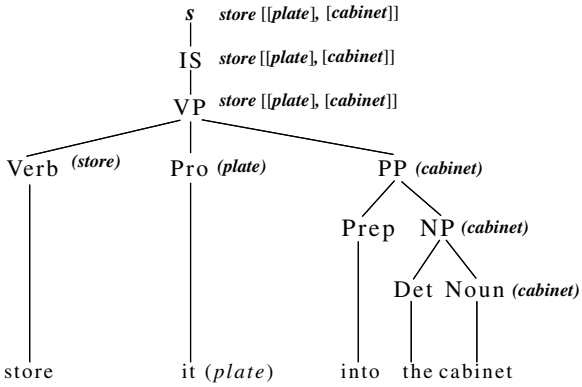

Fig. 5. A semantic attachment for a parse tree (ii) of the Fig. 2

5. IMPLEMENTATION ISSUES AND SUMMARY

After unambiguous semantic interpretation is obtained, the processor converts the semantic parse tree into an agent communication language (ACL) message, in which a task-oriented behaviour is represented in ACL. The details of the conversion technique was presented in the previous paper (Jusoh *et al.*, 2003). In implementing this technique, the parser for searching parse trees of each sentences is developed by using a dynamic programming technique. The implementation work is conducted in C language on Linux operating system. To guide the searching process, 70 fuzzy grammar rules have been used. The lexicon for input sentences is developed in a standard English. Because there is no existing dataset for human-robot interaction, a dataset to test this technique is created. In generating the dataset, interaction between a mobile service robot and a user occurs in a house is assumed. The dataset consist of 113 sentences in which are categorised into various applications such as cleaning, cooking, and so on.

As a summary, this paper has demonstrated a new methodology that can be used to better resolve the ambiguity problem in natural language sentences for autonomous robots that use behaviour-based control. However, only a part of our ongoing research in creating an intelligent natural language processor for human-robot interfacing is reported in this paper.

REFERENCES

Arkin, R. C. (1998). *Behavior-based Robotics: Intelligent Robots and Autonomous Agents*. MIT Press. Cambridge, Mass.

Asoh, H., Y. Motomura, F. Asano, I. Hara, S. Hayamizu, K. Itou, T. Kurita, T. Matsui, N. Vlassis, R. Bunschoten and B. Kröse (2001). Jijo-2: An office robot that communicates and learns. *IEEE Intelligent Systems* **16**(5), 44 – 55.

Burgard, W., A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner and S. Thrun (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence* **114**(1-2), 3–55.

Hasegawa, Y. and T. Fukuda (1999). Motion coordination of behavior-based controller for brachiation robot. In: *IEEE International Conference on Systems, Man, and Cybernetics*. pp. 896 –901.

Jurafsky, D. and J. H. Martin (2000). *Speech and Language Processing*. Prentice-Hall. United States of America.

Jusoh, Shaidah, Fangju Wang and Simon X. Yang (2003). An intelligent interface for a mobile service robot. In: *Proceeding of IEEE Intelligent Automation Conference (IAC'2003)*. pp. 567–572.

Nakauchi, Y. and R. Simmons (2002). A social robot that stands in line. *Autonomous Robots* **12**(3), 313–324.