# Referring Expression Generation under Uncertainty in Integrated Robot Architectures

Tom Williams and Matthias Scheutz
Human-Robot Interaction Laboratory
Tufts University
williams@cs.tufts.edu, matthias.scheutz@tufts.edu

***Abstract*—To engage in language-based social interaction with humans, robots must be able to refer to other agents, objects, and locations, also known as *referring expression generation (REG)*. Unfortunately, classic REG algorithms like the Incremental Algorithm assume that information is certain, centralized, and easily accessible: assumptions which commonly break down in realistic human-robot interaction scenarios. In this work, we show how the incremental algorithm can be extended to produce *DIST-PIA*: a domain-independent algorithm for REG under uncertain, heterogeneous, and distributed knowledge.**

## I. Introduction

To engage in language-based social interaction with humans, robots must be able to choose what properties to use to refer to other agents, objects, and locations. Traditionally, algorithms designed to solve this task (known as Referring Expression Generation (REG) algorithms) operate by selecting properties that appear in the *attribute set* of the target to be described, but do not appear in the attribute sets for one or more *distractors*, allowing those distractors to be ruled out.

Unfortunately, in most robot architectures, checking whether an entity has a certain property is not as simple as a set-membership check. Knowledge in robot architectures is commonly distributed across multiple architectural components which use different forms of representation. Moreover, because of the uncertainty inherent to realistic human-robot interaction scenarios, these components may be unable to definitively answer whether or not an entity has a certain property.

This presents a clear need for a general framework whereby architectural components query the *probability* that particular properties hold for particular entities, without requiring any prior knowledge of the *location* in the architecture where information about that entity is stored or *how* that entity is represented. Provided with such a framework, traditional REG algorithms could easily be modified to issue this type of query rather than making attribute-set set-membership checks.

In previous research, we presented just such a framework [9], and showed how it could be used to facilitate the inverse of the referring expression generation, known as *reference resolution*. In this work, we show how this framework can also be used for referring expression generation, and present *DIST-PIA*: a novel REG algorithm inspired by Dale and Reiter's classic *Incremental Algorithm*. Unlike previous robot-oriented approaches to referring expression generation under uncertainty, this algorithm can be used to generate expressions referring to entities other than objects. Furthermore, due to the use of this *consultant framework*, this algorithm does not mandate the use of any particular classifier.

We begin by discussing previous work on REG under uncertainty. We then describe how we have extended our consultant framework, and present *DIST-PIA*, our novel REG algorithm which leverages that framework. Finally, we conclude by discussing possible directions for future work.

## II. Previous Work

While much research has done on referring expression generation, three algorithms stand out the foundational algorithms of the field: *Full Brevity*, the *Greedy Algorithm* [1], and the *Incremental Algorithm* [2]. These algorithms, however, operate under a number of simplifying assumptions which are untenable in many real-world scenarios. Most crucially for human-centered robotics (HCR) applications, these algorithms assume certain knowledge on the part of both speaker and listener. More recently, there have been two classes of REG algorithm which have attempted to relax this assumption.

In the first class are algorithms that, like those foundational algorithms, are domain-independent, but which only take the *listener's* uncertainty into account [6]. This is problematic as this may lead a robot to unintentionally communicate beliefs it does not actually hold, reinforcing misconceptions and misaligning common ground. In the second class are algorithms which take the *speaker's* uncertainty into account, but which are highly domain-*dependent*, being only able to choose visually assessable properties and describe currently visible entities [3, 4, 7]. What is more, the approaches from both classes share another assumption with the foundational approaches which may not hold in all HCR contexts: that all relevant information is conveniently stored in a single format in a single centralized knowledge base.

It appears that what is needed is an REG algorithm that uses a framework for handling uncertain knowledge from different domains and of heterogeneous representation which is distributed throughout a robot architecture. If such a framework is available, a modified version of any of the foundational algorithms could be used, but we argue that the *Incremental Algorithm (IA)* is the best choice, as it has advantageous complexity guarantees, generates human-like REs through its use of a *preference ordering* over properties, and has generally been the most successful REG algorithm. In the next section, we describe just such a framework, and then describe how it can be extended to facilitate use with the *IA*.

## III. Framework

In previous work, we presented a framework which makes uses a set of "consultants", each of which facilitates access to a *distributed heterogeneous knowledge-base* $k$ for a different domain [9]. Each such consultant must be capable of at least four functions:

1) providing a set $c_{domain}$ of atomic entities from $k$,
2) advertising a list $c_{constraints}$ of constraints that can be assessed with respect to entities from $c_{domain}$,
3) assessing constraints from $c_{constraints}$ with respect to entities from $c_{domain}$, and
4) adding, removing, or imposing constraints from $c_{constraints}$ on entities from $c_{domain}$.

These capabilities were originally chosen to facilitate *reference resolution*. However, the first three can also be used to facilitate REG: Capability 1 provides a set of distractors which must be ruled out, and Capability 2 provides a list of properties that can be checked using Capability 3 to determine if they (i) apply to the target referent, and (ii) rule out distractors.

The *IA*, however, considers constraints according to a *preference ordering*: for example, using an entity's *type* to describe it is typically preferable to its *color*, which is typically preferable to its *size*, and so on. To use this framework for REG, we thus modify the second capability to require such an ordering:

2) advertising a list $c_{constraints}$ of constraints that can be assessed with respect to entities from $c_{domain}$, *and that is ordered by descending preference*.

With this modification, we now have a framework which provides access to uncertain knowledge from different domains and of heterogeneous representation which is distributed throughout a robot architecture, and which is configured to interface well with the *IA*. In the next section, we describe how we have similarly modified the *IA* in order to take advantage of this framework.

## IV. Algorithm and Walkthrough

In this section we present our version of the *Incremental Algorithm (IA)* [2], modified to use the modified framework presented in the previous section. The standard *IA* incrementally proceeds through an ordered list of properties that could be used to describe a referent. For each such property $p$, if $p$ is in the list of properties attributed to the target referent, *IA* checks whether $p$ is *not* true of any distractors (initially all other possible referents). If any distractors are ruled out, $p$ is added to the *description*, i.e., the list of properties to communicate, and those ruled-out distractors are removed from the set of distractors. This process iterates until all distractors or properties are ruled out.

In order to make the *IA* usable in realistic HCR applications, we must adapt it to use the modified framework presented in the previous section so that it can make use of uncertain and distributed knowledge. This adaptation is necessary because when information is uncertain and distributed, we cannot assume that there will exist a precomputed set of sufficiently probable properties that hold for either the target or its distractors. Thus, instead of a simple set-membership check, we must use the framework to assess whether each property holds for the target or a given distractor by making a query to the relevant consultant.

In this section, we present *DIST-PIA*, the Distributed, Probabilistic Incremental Algorithm which does just that. In the remainder of this section, we first define the notation used throughout the remainder of the section. We then present as pseudocode the *DIST-PIA* algorithm and its helper function, *DIST-PIA-HELPER*. Finally, we provide a walkthrough of these algorithms on an example problem.

### A. Notation

| | |
|---|---|
| $C$ | A set of *consultants* $\{c_0, \dots, c_n\}$ |
| $c^{\Lambda}_m$ | The set of formulae $\{\lambda_0, \dots, \lambda_n\}$ advertised by the consultant $c \in C$ responsible for $m$. |
| $M$ | A robot's *world model* of entities $\{m_0 \dots m_n\}$ found in the domains provided by $C$. |
| $D$ | The incrementally built up description, comprised of mappings from entities $M$ to sets of pairs $(\lambda, \Gamma)$ of formulae and bindings for those formulae. |
| $D^M$ | The set of entities $m \in M$ for which sub-descriptions have been created. |
| $d^M$ | The set of entities $m \in M$ involved in sub-description $d$. |
| $P$ | The set of candidate $(\lambda, \Gamma)$ pairs under consideration for addition to a sub-description. |
| $Q$ | The queue of referents which must be described. |
| $X$ | The incrementally pruned set of distractors |

**Algorithm 1** *DIST-PIA*$(m, C)$

```
1: $D$ = new Map() // The Description
2: $Q$ = new Queue($m$) // The Referent Queue
3: while $Q \neq \emptyset$ do
4:     // Consider the next referent
5:     $m\prime$ = pop(Q)
6:     // Craft a description d for it
7:     $d$ = DIST-PIA-HELPER($m\prime, C$)
8:     $D = D \cup \{m \rightarrow d\}$
9:     // Find all entities used in d
10:    for all $m\prime\prime \in d^M \setminus keys(D)$ do
11:        // And add undescribed entities to the queue
12:        $push(Q, m\prime\prime)$
13:    end for
14: end while
15: return $D$
```

### B. Algorithm Walkthrough

In this section we will provide a walkthrough of our algorithms in an example scenario. Each step of this walkthrough is summarized in a row of Tab. I and denoted in the walkthrough in bold (e.g., **(1)**).

Imagine a robot with three distributed consultants which use disparate representational schemes for representing people, locations, and objects respectively $(p, l, o)$. Suppose that this robot needs to refer to entity $p_5$. Provided with target referent $m = p_5$ and set of consultants $C = \{p, l, o\}$, *DIST-PIA* (*DP* hereafter) will begin (Tab. I Row **(1)**; Algorithm 1,Lines 1-2) by creating empty description $D = \emptyset$ and referent queue $Q = \{p_5\}$. Next, *DP* must determine the next referent for which a description must be created (Line 5), popping $p_5$ off of $Q$. Because a description for $p_5$ does not appear in $D$, will call *DIST-PIA-HELPER*) (*DPH* hereafter) with arguments $(p_5, \{p, l, o\})$ (Line 7). *DPH* is responsible for crafting the

**Algorithm 2** *DIST-PIA-HELPER*$(m, C)$

```
1: $d = \emptyset$ // The Sub-Description
2: $X = M \setminus m$ // The Distractors
3: // Initialize a set of properties to consider: those advertised by the consultant c responsible for m
4: $P = [\forall \lambda \in c_m^{\Lambda} : (\lambda, \emptyset)]$
5: // While there are distractors to eliminate or properties to consider
6: while $X \neq \emptyset$ and $P \neq \emptyset$ do
7:     $(\lambda, \Gamma) = pop(P)$
8:     // Find all unbound variables in the next property
9:     $V = find\_unbound(\lambda, \Gamma)$
10:    if $|V| > 1$ then
11:        // If there's more than one, create copies of that property under all possible variable bindings that leaving unbound exactly one variable of the same type as the target referent
12:        for all $\Gamma\prime \in cross\_bindings(\lambda, \Gamma, C)$ do
13:            // And push them onto the property list
14:            $push(P, (\lambda, \Gamma\prime))$
15:        end for
16:        // Otherwise, if it is sufficiently probable that the property applies to the target referent...
17:    else if $apply(c_m, \lambda, \Gamma \cup (v_0 \rightarrow m)) > \tau_{dph}$ then
18:        // And it's sufficiently probable that it does not apply to at least one distractor...
19:        $\bar{X} = [x \in X \mid apply(c_x, \lambda, \Gamma \cup (v_0 \rightarrow x)) > \tau_{dph}]$
20:        // Then bind its free variable to the target referent, and add it to the sub-description...
21:        if $\bar{X} \neq \emptyset$ then
22:            // And remove any eliminated distractors
23:            $d = d \cup (\lambda, \Gamma \cup (v_0 \rightarrow m))$
24:            $X = X \setminus \bar{X}$
25:        end if
26:    end if
27: end while
28: return $d$
```

sub-descriptions that comprise $D$, and begins by initializing sub-descriptor $d = \emptyset$, distractor set $X = \{p_1, p_2, p_3, p_4\}$ (assuming a set of five known people), and a set of properties $P$ to consider adding to $d$ (Algorithm 2, Lines 1- 4). Suppose $p$ (i.e., the consultant $c_m$ responsible for entity $m$) advertises the set of properties $c_m^{\Lambda}$ = jim(X-p), jill(X-p), man(X-p), woman(X-p), lives-in(X-p,Y-l).

First, *DPH* will pop from $P$ the first unconsidered property (i.e., $jim(X-p)$) in the form of a pair $(\lambda, \Gamma)$, where $\lambda$ is a formula and $\Gamma$ is a partial set of bindings for that formula (Line 7). Next, *DIST-PIA-HELPER* finds all variables $V$ in $\lambda^V$ that do not have bindings in $\Gamma$ (Line 9). $jim(X-p)$ has exactly one unbound variable, so *DPH* will use **(2)** the *apply* method (provided by each of *DP*'s consultants, as per capability three) to ask how probable it is that $jim(p_5)$ holds. Suppose the returned probability is above some threshold, say 60%. *DPH* will thus determine if this property also weeds out distractors. For each referent $x$ in $X$, *DPH* uses *apply* to ask how probable it is that $jim(x)$ holds. Suppose this is only sufficiently probable for $p_2$. The set of eliminated distractors $\bar{X}$ will equal $\{p_1, p_3, p_4\}$. Since this is nonempty, **(3)** $\{p_1, p_3, p_4\}$ will be removed from $X$ and **(4)** $jim(p_5)$ will be added to sub-description $d$ (Lines 17- 26).

*DPH* will next consider $jill(X-p)$. This has one unbound variable, so *DPH* will use **(5)** *apply* to ask how probable it is that $jill(p_5)$ holds. Suppose the returned probability is below 60%. *DPH* will move on, to consider $man(X-p)$. This predicate has one unbound variable, so *DPH* will use **(6)** *apply* to ask how probable it is that $man(p_5)$ holds. Suppose the returned probability is above 60%. *DPH* will thus determine if this property also weeds out distractors. For each referent $x$ in $X$, *DPH* will use *apply* to ask how probable it is that $man(x)$ holds. Suppose this is sufficiently probable for the lone remaining distractor, $p_2$. Since the set of eliminated distractors $\bar{X}$ is empty **(7)**, *DPH* will not add this property to $d$, but will instead move on.

*DPH* will next consider $woman(X-p)$. This has one unbound variable, so *DPH* will use **(8)** *apply* to ask how probable it is that $woman(p_5)$ holds. Suppose the returned probability is below 60%. *DPH* will move on, to consider $lives\text{-}in(X-p, Y-l)$. This has two unbound variables, so *DPH* will use **(9)** $cross\_bindings$ to create variable bindings that leave exactly one $p$-associated variable unbound. Since only $X$ is associated with $p$, it finds the set of candidate bindings to $Y$. If $l$ knows of three locations: $\{l_1, l_2,$ and $l_3\}$, *DPH* will add the following to $P$: $(lives\text{-}in(X-p, Y-l), \{Y \rightarrow ls_1\})$, $(lives\text{-}in(X-p, Y-l), \{Y \rightarrow ls_2\})$, and $(lives\text{-}in(X-p, Y-l), \{Y \rightarrow ls_3\})$ (Lines 10- 15).

*DPH* will next consider $lives\text{-}in(X-p, l_1)$. This has one unbound variable, so *DPH* will use **(10)** *apply* to ask how probable it is that $lives\text{-}in(p_5, l_1)$ holds. Suppose the returned probability is above 60%. *DPH* will thus determine if this property also weeds out distractors. For each referent $x$ in $X$, *DPH* will use *apply* to ask how probable it is that $lives\text{-}in(x, l_1)$ holds. Suppose that this is not sufficiently probable for the lone remaining distractor, $p_2$. The set of eliminated distractors $\bar{X}$ will equal $\{p_2\}$. Since this is nonempty **(11)**, $lives\text{-}in(X-p, Y-l)$ will be added **(12)** to $d$ and $\{p_2\}$ will be removed from $X$. Since $X$ is empty, $p_5 \rightarrow \{jim(p_5), lives\text{-}in(p_5, l_1)\}$ will be returned **(13)** to *DP* (Line 28).

Now, *DP* will add all entities mentioned in this set of properties that have not yet been described (i.e., $l_1$) to $Q$. Next, *DP* will pop $l_1$ off of $Q$, and call *DPH*$(l_1, \{p, l, o\})$. *DPH* will first initialize **(14)** sub-descriptor $d = \emptyset$, and set of distractors $X = \{l_2, l_3\}$, assuming for simplicity that $l$ only knows of three people. Next, let's suppose $l$ advertises the following properties: somerville(X-l),cambridge(X-l),massachusetts(X-l),in(X-l,Y-l).

First, *DPH* will consider $somerville(X-l)$. This has one unbound variable, so *DPH* will use **(15)** *apply* to ask how probable it is that $somerville(l_1)$ holds. Suppose the returned probability is above 60%. *DPH* will thus determine if this property also weeds out distractors. For each referent $l_x$ in $X$, *DPH* will use *apply* to ask how probable it is that $somerville(l_x)$ holds. Suppose that it is not sufficiently probable that any of the distractors have this property. The set of eliminated distractors $\bar{X}$ will equal $\{l_2, l_3\}$. Since this is nonempty **(16)**, $somerville(l_1)$ will be added **(17)** to $d$ and $\{l_2, l_3\}$ will be removed from $X$. Since $X$ is empty, $l_1 \rightarrow \{somerville(l_1)\}$ will be returned **(18)** to *DP*. Since $Q$ is empty, *DP* will return $\{p_5 \rightarrow \{jim(p_5), lives\text{-}in(p_5, l_1)\}, l_1 \rightarrow \{somerville(l_1)\}\}$

| # | Act | Description | m | Sub-description | Distractors | Property | Property List |
|---|---|---|---|---|---|---|---|
| 1 | P | $\emptyset$ | $p_5$ | $\emptyset$ | $\{p_1, p_2, p_3, p_4\}$ | $\emptyset$ | $\{jim(X\text{-}p), jill(X\text{-}p), man(X\text{-}p), wom(X\text{-}p), l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 2 | A | $\emptyset$ | $p_5$ | $\emptyset$ | $\{p_1, p_2, p_3, p_4\}$ | $jim(X\text{-}p)$ | $\{jill(X\text{-}p), man(X\text{-}p), wom(X\text{-}p), l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 3 | E | $\emptyset$ | $p_5$ | $\emptyset$ | $\{p_2\}$ | $jim(X\text{-}p)$ | $\{jill(X\text{-}p), man(X\text{-}p), wom(X\text{-}p), l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 4 | d | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\{p_2\}$ | $\emptyset$ | $\{jill(X\text{-}p), man(X\text{-}p), wom(X\text{-}p), l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 5 | A | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\{p_2\}$ | $jill(X\text{-}p)$ | $\{man(X\text{-}p), wom(X\text{-}p), l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 6 | A | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\{p_2\}$ | $man(X\text{-}p)$ | $\{wom(X\text{-}p), l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 7 | E | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\{p_2\}$ | $man(X\text{-}p)$ | $\{wom(X\text{-}p), l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 8 | A | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\{p_2\}$ | $wom(X\text{-}p)$ | $\{l\text{-}in(X\text{-}p, Y\text{-}l)\}$ |
| 9 | B | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\{p_2\}$ | $l\text{-}in(X\text{-}p, Y\text{-}l)$ | $\{l\text{-}in(X\text{-}p, l_1), l\text{-}in(X\text{-}p, l_2), l\text{-}in(X\text{-}p, l_3)\}$ |
| 10 | A | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\{p_2\}$ | $l\text{-}in(X\text{-}p, l_1)$ | $\{l\text{-}in(X\text{-}p, l_2), l\text{-}in(X\text{-}p, l_3)\}$ |
| 11 | E | $\emptyset$ | $p_5$ | $\{jim(p_5)\}$ | $\emptyset$ | $l\text{-}in(X\text{-}p, l_1)$ | $\{l\text{-}in(X\text{-}p, l_2), l\text{-}in(X\text{-}p, l_3)\}$ |
| 12 | d | $\emptyset$ | $p_5$ | $\{jim(p_5), l\text{-}in(p_5, l_1)\}$ | $\emptyset$ | $\emptyset$ | $\{l\text{-}in(X\text{-}p, l_2), l\text{-}in(X\text{-}p, l_3)\}$ |
| 13 | D | $\{jim(p_5), l\text{-}in(p_5, l_1)\}$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 14 | P | $\{jim(p_5), l\text{-}in(p_5, l_1)\}$ | $l_1$ | $\emptyset$ | $\{l_2, l_3\}$ | $\emptyset$ | $\{som(X\text{-}l), cam(X\text{-}l), mass(X\text{-}l), in(X\text{-}l, Y\text{-}l)\}$ |
| 15 | A | $\{jim(p_5), l\text{-}in(p_5, l_1)\}$ | $l_1$ | $\emptyset$ | $\{l_2, l_3\}$ | $som(X\text{-}l)$ | $\{cam(X\text{-}l), mass(X\text{-}l), in(X\text{-}l, Y\text{-}l)\}$ |
| 16 | E | $\{jim(p_5), l\text{-}in(p_5, l_1)\}$ | $l_1$ | $\emptyset$ | $\emptyset$ | $som(X\text{-}l)$ | $\{cam(X\text{-}l), mass(X\text{-}l), in(X\text{-}l, Y\text{-}l)\}$ |
| 17 | d | $\{jim(p_5), l\text{-}in(p_5, l_1)\}$ | $l_1$ | $\{som(l_1)\}$ | $\emptyset$ | $\emptyset$ | $\{cam(X\text{-}l), mass(X\text{-}l), in(X\text{-}l, Y\text{-}l)\}$ |
| 18 | D | $\{jim(p_5), l\text{-}in(p_5, l_1), som(l_1)\}$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |

TABLE I: WALKTHROUGH SUMMARY. Column Two summarizes action taken: **P**repare, **A**ssess, **E**liminate, **B**ind, **d**-append, or **D**-append. Some predicates are abbreviated, and predicate/binding tuples are rewritten as bound predicates.

(Algorithm 1, Line 15), with the expectation that natural language generation will use these properties to craft a referring expression along the lines of "Jim, who lives in Somerville".

## V. Conclusion

In this paper, we make three main contributions. First, we presented a domain independent algorithm for REG under uncertainty, which separates the problems of referring expression generation and reference resolution from the task of *property assessment*. Second, we have taken the realities of modern integrated agent architectures into account by using our consultant framework [9], which allows information to be distributed across multiple heterogeneous KBs.

In future work, we must evaluate this approach to experimentally verify that the REs it generates are both useful and natural. Once this is done, we would like to adapt our approach to learn its thresholds from data, to use uncertainty representations that better handle ignorance [10], to incorporate audience design considerations, similar to Horacek [6], to use Givenness-Hierarchy Theoretic mechanisms [11] to generate deictic and anaphoric REs, and to extend the approach to generate references to *sets* of objects as well [5, 8].

## Acknowledgments

This research was in part funded by grant N00014-14-1-0149 from the US Office of Naval Research.

## References

[1] Robert Dale. Cooking up referring expressions. In *Proc. of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 68–75, 1989.

[2] Robert Dale and Ehud Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[3] Rui Fang, Changsong Liu, Lanbo She, and Joyce Y Chai. Towards situated dialogue: Revisiting referring expression generation. In *Proc. of Empirical Methods on Natural Language Processing*, pages 392–402, 2013.

[4] Rui Fang, Malcolm Doering, and Joyce Y Chai. Embodied Collaborative Referring Expression Generation in Situated Human-Robot Interaction. In *Proc. of the 10th Int'l Conference on Human-Robot Interaction*, 2015.

[5] Nicholas Fitzgerald, Yoav Artzi, and Luke Zettlemoyer. Learning Distributions over Logical Forms for Referring Expression Generation. In *Proc. of Empirical Methods in Natural Language Processing*, pages 1914–1925, 2013.

[6] Helmut Horacek. Generating referential descriptions under conditions of uncertainty. In *Proc. of the 10th European Workshop on NLG*, pages 58–67, 2005.

[7] Amir Sadovnik, Andrew Gallagher, and Tsuhan Chen. Not everybody's special: Using neighbors in referring expressions with uncertain attributes. In *Proc. of the Conference on Computer Vision and Pattern Recognition Workshops*, pages 269–276, 2013.

[8] Kees Van Deemter. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52, 2002.

[9] Tom Williams and Matthias Scheutz. A framework for resolving open-world referential expressions in distributed heterogeneous knowledge bases. In *Proc. of the 30th AAAI Conference on Artificial Intelligence*, 2016.

[10] Tom Williams, Gordon Briggs, Bradley Oosterveld, and Matthias Scheutz. Going beyond command-based instructions: Extending robotic natural language interaction capabilities. In *Proc. of 29th AAAI Conference on Artificial Intelligence*, pages 1387–1393, 2015.

[11] Tom Williams, Saurav Acharya, Stephanie Schreitter, and Matthias Scheutz. Situated open world reference resolution for human-robot dialogue. In *Proc. of the 11th Int'l Conference on Human-Robot Interaction*, 2016.