Probabilistic Refinement Algorithms for the Generation of Referring Expressions

Romina Altamirano[1] *Carlos Areces*[1,2] *Luciana Benotti*[1]
(1) FaMAF, Universidad Nacional de Córdoba, Argentina
(2) Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
`{ialtamir,areces,benotti}@famaf.unc.edu.ar`

ABSTRACT

We propose an algorithm for the generation of referring expressions (REs) that adapts the approach of Areces et al. (2008, 2011) to include overspecification and probabilities learned from corpora. After introducing the algorithm, we discuss how probabilities required as input can be computed for any given domain for which a suitable corpus of REs is available, and how the probabilities can be adjusted for new scenes in the domain using a machine learning approach. We exemplify how to compute probabilities over the GRE3D7 corpus of Viethen (2011). The resulting algorithm is able to generate different referring expressions for the same target with a frequency similar to that observed in corpora. We empirically evaluate the new algorithm over the GRE3D7 corpus, and show that the probability distribution of the generated referring expressions matches the one found in the corpus with high accuracy.

KEYWORDS: Generation of referring expressions, refinement algorithms, machine-learning.

Proceedings of COLING 2012: Posters, pages 53–62,
COLING 2012, Mumbai, December 2012.

1 Generation of referring expressions

In linguistics, a *referring expression* (RE) is an expression that unequivocally identifies the intended target to the interlocutor, from a set of possible distractors. The generation of referring expressions (GRE) is a key task of most natural language generation (NLG) systems (Reiter and Dale, 2000, Section 5.4). Depending on the information available to the NLG system, certain objects might not be associated with an identifier which can be easily recognized by the user. In those cases, the system will have to generate a, possibly complex, description that contains enough information so that the interlocutor will be able to identify the intended referent.

The generation of referring expressions is a well developed field in automated natural language generation. Building upon GRE foundational work (Winograd, 1972; Dale, 1989; Dale and Reiter, 1995), various proposals have investigated the generation of different kinds of referring expressions such as relational expressions ("the blue ball next to the cube" (Dale and Haddock, 1991)), reference to sets ("the two small cubes" (Stone, 2000)), or more expressive logical connectives ("the blue ball not on top of the cube" (van Deemter, 2002)). REs involving relations, in particular, have received increasing attention recently. However, the classical algorithm by Dale and Haddock (1991) was shown to be unable to generate satisfying REs in practice (see the analysis over the *cabinet corpus* in (Viethen and Dale, 2006)). Furthermore, the Dale and Haddock algorithm and many of its successors (such as (Kelleher and Kruijff, 2006)) are vulnerable to the problem of *infinite regress*, where the algorithm enters an infinite loop, jumping back and forth between descriptions for two related individuals, as in "the book on the table which supports a book on the table . . . "

Areces et al. (2008, 2011) have proposed low complexity algorithms for the generation of relational REs that eliminate the risk of infinite regression. These algorithms are based on variations of the partition refinement algorithms of Paige and Tarjan (1987). The information provided by a given scene is interpreted as a relational model whose objects are classified into sets that fit the same description. This classification is successively *refined* till the target is the only element fitting the description of its class. The existence of an RE depends on the information available in the input scene, and on the expressive power of the formal language used to describe elements of the different classes in the refinement. Refinement algorithms effectively compute REs for all individuals in the domain, at the same time. The algorithms always terminate returning a formula of the formal language chosen that uniquely describes the target (if the formal language is expressive enough to identify the target in the input model). Refinement algorithms require an ordered list of properties that can be used to described the objects in the scene, and the naturalness of the generated REs strongly depends on this ordering. The goal of this paper is twofold. First we show how we can add non-determinism and overspecification to the refinement algorithms, by replacing the fixed ordering over properties of the input scene by a *probability of use* for each property, and modifying the algorithm accordingly. In this way, each call to the algorithm can produce different REs for the same input scene and target. We will then show that given suitable corpora of REs (like the GRE3D7 corpora discussed in (Viethen, 2011)) we can estimate these probabilities of use so that REs are generated with a probability distribution that matches the one found in corpora.

2 Adding non-determinism and overspecification

Refinement algorithms for GRE are based on the following basic idea: given a scene S, the objects appearing in S are successively classified according to their properties into finer and finer classes. A description (in some formal language $\mathscr{L}$) of each class is computed every time a class is refined. The procedure always stops when the set of classes stabilizes, i.e., no further refinement is possible with the information available in the scene[1]. If the target element is in a singleton class, then the formal description of that class is a referring expression; otherwise the target cannot be unequivocally described (in $\mathscr{L}$).

We present a modification of the algorithm in (Areces et al., 2008) where the fixed order of properties in the input scene is replaced by a finite probability distribution. The resulting algorithm (see Figure 3) is now non-deterministic: two runs of the algorithm with the same input might result in different REs for objects in the scene. The input to the algorithm will be a relational model $\mathscr{M} = \langle \Delta, \| \cdot \| \rangle$, where Δ is the non-empty domain of objects in the scene, and $\| \cdot \|$ is an interpretation function that assigns to all properties in the scene its intended extension. For example, the scene shown in Figure 1 could be represented by the model $\mathscr{M} = \langle \Delta, \| \cdot \| \rangle$ shown in Figure 2; where $\Delta = \{e_1, \ldots, e_7\}$, and $\|green\|$, for example, is $\{e_3, e_4, e_6\}$.

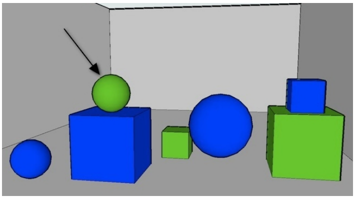

Figure 1: Input scene

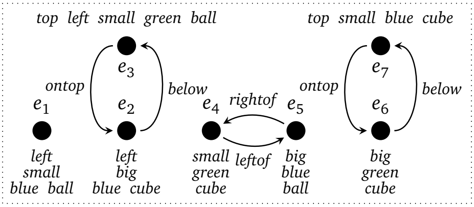

Figure 2: Scene as a relational model

On termination, the algorithm computes what are called the $\mathscr{L}$-similarity classes of the input model $\mathscr{M}$. Intuitively, if two elements in the model belong to the same $\mathscr{L}$-similarity class, then $\mathscr{L}$ is not expressive enough to tell them appart (i.e, no formula in $\mathscr{L}$ can distinguish them).

The algorithm we discuss uses formulas of the $\mathscr{EL}$ description logic language (Baader et al., 2003) to describe refinement classes[2]. The interpretation of the $\mathscr{EL}$ formula $\psi \sqcap \exists \mathsf{R}.\varphi$ is the set of all elements that satisfy ψ and that are related by relation R to some element that satisfy φ.

Algorithm 1 takes as input a model and a list Rs of pairs (R,R.p_{use}) that links each relation $\mathsf{R} \in \mathsf{REL}$, the set of all relation symbols in the model, to some probability of use R.p_{use}. The set RE will contain the formal description of the refinement classes and it is initialized by the most general description $\top$. For each R, we first compute R.rnd_{use}, a random number in [0,1]. If R.$\mathsf{rnd}_{use} \leq$ R.p_{use} then we will use R to refine the set of classes. The value of R.p_{use} will be incremented by R.inc_{use} in each main loop, to ensure that all relations are, at some point, considered by the algorithm. This ensures that a referring expression will be found if it exists; but gives higher probability to expressions using relations with a high R.p_{use}. While RE contains descriptions that can be refined (i.e., classes with at least two elements) we will call the refinement function $\mathsf{add}_{\mathscr{L}}$(R,$\varphi$,RE) successively with each relation in Rs. A change in one of the classes, can trigger changes in others. For that reason, if RE changes, we exit the **for** loop to start again with the relations of higher R.p_{use}. If after trying to refine the set with all relations in Rs, the set RE has not changed, then we have reached a stable state (i.e., the classes described in RE cannot be further refined with the current R.p_{use} values). We will then increment all

[1] Of course, if we are only interested in a referring expression for a given target we can stop the procedure as soon as the target is the only element of some of the classes.

[2] Notice, though, that the particular formal language used is independent of the main algorithm, and different $\mathsf{add}_{\mathscr{L}}$(R,$\varphi$,RE) functions can be used depending on the language involved.

Algorithm 1: Computing $\mathscr{L}$-similarity classes

```
Input: A model $\mathscr{M}$ and a list $Rs \in (REL \times [0,1])^*$ of relation symbols with their $p_{use}$ values, ordered by $p_{use}$
Output: A set of formulas RE such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathscr{L}$-similarity classes of $\mathscr{M}$
$RE \leftarrow \{\top\}$          // the most general description $\top$ applies to all elements in the scene
for $(R, R.p_{use}) \in Rs$ do
    $R.rnd_{use} = Random(0,1)$                   // $R.rnd_{use}$ is the probability of using R
    $R.inc_{use} = (1 - R.p_{use})$ / MaxIterations   // $R.p_{use}$ are incremented by $R.inc_{use}$ in each loop
repeat
    while $\exists(\varphi \in RE).(\#\|\varphi\| > 1)$ do          // while some class has at least two elements
        RE' $\leftarrow$ RE                                   // make a copy for future comparison
        for $(R, R.p_{use}) \in Rs$ do
            if $R.rnd_{use} \leq R.p_{use}$ then                    // R will be used in the expression
                for $\varphi \in RE$ do $add_{\mathscr{EL}}(R, \varphi, RE)$            // refine all classes using R
            if RE $\neq$ RE' then                              // the classification has changed
                exit                        // exit for-loop to try again highest $R.p_{use}$
        if RE = RE' then                                 // the classification has stabilized
            exit                                   // exit while-loop to increase $R.p_{use}$
    for $(R, R.p_{use}) \in Rs$ do $R.p_{use} \leftarrow R.p_{use} + R.inc_{use}$                  // increase $R.p_{use}$
until $\forall((R, R.p_{use}) \in Rs).(R.p_{use} \geq 1)$        // $R.p_{use}$ are incremented until they reach 1
```

Algorithm 2: $add_{\mathscr{EL}}(R, \varphi, RE)$

```
if FirstLoop? then                                            // are we in the first loop?
    Informative $\leftarrow$ TRUE                                      // allow overspecification
else Informative $\leftarrow \|\psi \sqcap \exists R.\varphi\| \neq \|\psi\|$;      // informative:  smaller than the original?
for $\psi \in RE$ with $\#\|\psi\| > 1$ do
    if $\psi \sqcap \exists R.\varphi$ is not subsumed in RE and     // non-redundant:  can't be obtained from RE?
      $\|\psi \sqcap \exists R.\varphi\| \neq \emptyset$ and                              // non-trivial:  has elements?
      Informative then
        add $\psi \sqcap \exists R.\varphi$ to RE                   // add the new class to the classification
        remove subsumed formulas from RE                         // remove redundant classes
```

Figure 3: Refinement algorithm with probabilities and overspecification for the $\mathscr{EL}$-language

the $R.p_{use}$ values and start the procedure again. Algorithm 2 almost coincides with the one in (Areces et al., 2008). The **for** loop will refine each descriptions in RE using the relation R and the other descriptions already in RE, under certain conditions. The new description should be *non-redundant* (it cannot be obtained from classes already in RE), *non-trivial* (it is not empty), and *informative* (it does not coincide with the original class). If these conditions are met, the new description is added to RE, and redundant descriptions created by the new description are eliminated. The **if** statement at the beginning of Algorithm 2 disregards the informativity test during the first loop of the algorithm allowing overspecification.

3 Learning to describe new objects from corpora

The algorithm presented in the previous section assumes that each relation R used in a referring expression has a known probability of use $R.p_{use}$. In this section, we describe how to learn these probabilities from corpora. We use the GRE3D7 corpus to illustrate our learning set up.

The REs in the corpus were produced by 294 participants, each producing 16 referring expressions for 16 scenes. In this way, 140 descriptions for 32 different scenes were obtained, resulting in a corpus of 4480 REs describing a target in a 3D scene containing seven objects. Each description was elicited in the absence of a preceding discourse. A sample scene is shown in Figure 1 (the target is marked with an arrow). For more details on the corpus see (Viethen, 2011, Chapter 5). Importantly for our purposes, the corpus not only contains propositional REs (as other benchmark corpora in the area, e.g., (Gatt et al., 2008)) but also relational REs naturally produced by people. For example, the RE “small ball on top of cube” is used to describe the target in Figure 1. As our algorithm is one of the few that can generate relational REs in an efficient and reliable way, a corpus of relational REs is needed to test its full potential. It is worth mentioning that, although people only used 16 propositional properties and 4 relational properties in their REs, and converged to between 10 and 30 different descriptions of the same target, the possible different correct *relational REs* for a generation algorithm are in the order of several hundred. Hence, reproducing the corpus distribution is a complex task.

We calculate $\mathrm{R.p}_{use}$ values for each training scene in the corpus in the following way. First, we use the REs in the corpus C to define the relational model $\mathscr{M}$ used by the algorithm. Then we calculate the value of p_{use} for each relation R in the model as the percentage of REs in which the relation appears. I.e., $\mathrm{R.p}_{use}=$ (# of REs in C in which R appears)/(# of REs in C). The values $\mathrm{R.p}_{use}$ obtained in this way should be interpreted as the probability of using R to describe the target in model $\mathscr{M}$, and we could argue that they are correlated to the *saliency* of R in the scene. For that reason, for example, in the scene in Figure 1 the value of $\mathit{ball}.\mathrm{p}_{use}$ is 1, while the value of $\mathit{cube}.\mathrm{p}_{use}$ is 0.178. These probabilities will not be useful to describe different targets in different scenes. We will now see how we can use them to obtain values for new targets and scenes using a machine learning approach.

We selected eight different scenes for testing from the GRE3D7 corpus, and for each, we used the rest of the corpus for training. We used linear regression (Hall et al., 2009) to learn a function estimating the value of p_{use} for each relation in the domain. We used simple, domain independent features that can be extracted automatically from the relational model:

target-has(R) := true if the target is in R
#relations := number of relations the target is in
#bin-relations := number of the binary relations the target is in
landmark-has(R) := true if a landmark (i.e., an object directly related to the target) is in R
discrimination(R) := 1 divided the number of objects in the model that are in R

Despite its simplicity, the functions obtained by linear regression are able to learn interesting characteristics of the domain. E.g., they correctly model that the saliency of a color depends strongly on whether the target object is of that color, and it does not depend on its discrimination power in the model. They also correctly predict that the *ontop* relation is used more frequently than the horizontal relations (*leftof* and *rightof*), as reported in (Viethen, 2011). Interestingly, they also indicate a characteristic of the GRE3D7 corpus not mentioned in previous work: size is more frequently used for overspecification when the target and landmark have the same size (it is used in overspecified REs in 49% of the descriptions for scenes where target and landmark have the same size, and only 25% of the time when target and landmark have different size).

4 Evaluation

We present a quantitative evaluation of the algorithm proposed. In particular, we show that the probabilistic refinement algorithm with overspecification is able to generate a distribution of REs similar to that observed in corpora. We discuss in detail the experiments we run for the

Referring Expressions	Corpus		Algorithm		Accuracy
	#Cor	%Cor	#Alg	%Alg	%Acc
ball,green	91	65.00	6376	63.76	63.76
ball,green,small	23	16.43	3440	34.40	16.43
ball,green,small,on-top(blue,cube,large)	8	5.71	0	0.00	0.00
ball,green,on-top(blue,cube)	5	3.57	0	0.00	0.00
ball,green,on-top(blue,cube,large)	5	3.57	0	0.00	0.00
ball,green,small,on-top(blue,cube)	2	1.43	0	0.00	0.00
ball,on-top(cube)	1	0.71	27	0.27	0.27
ball,green,small,on-top(blue,cube,large,left)	1	0.71	0	0.00	0.00
ball,small,on-top(cube,large)	1	0.71	2	0.02	0.02
ball,green,top	1	0.71	0	0.00	0.00
ball,small,on-top(cube)	1	0.71	3	0.03	0.03
ball,green,on-top(cube)	1	0.71	0	0.00	0.00
ball,front,green	0	0.00	97	0.97	0.00
ball,front,green,small	0	0.00	13	0.13	0.00
ball,front,top	0	0.00	12	0.12	0.00
ball,green,left	0	0.00	11	0.11	0.00
ball,top	0	0.00	10	0.10	0.00
ball,green,left,small	0	0.00	5	0.05	0.00
ball,left,top	0	0.00	2	0.02	0.00
ball,small,top	0	0.00	1	0.01	0.00
ball,front,on-top(cube,left)	0	0.00	1	0.01	0.00
Total	140	100.00	10000	100	80.51

Table 1: REs in the corpus and those produced by our algorithm for Figure 1

scene shown in Figure 1 (Scene 3 in the GRE3D7 corpus), then summarize the results for the other seven scenes we used for testing.

Using p_{use} learned as described in Section 3 and running our algorithm 10000 times, we obtain 14 different referring expressions for Figure 1. It is already interesting to see that with the p_{use} values learned from the corpus the algorithm generates only a small set of RE with a high probability. Of these 14 different REs, 5 are the most frequent REs found in the corpus of 140 REs associated to the Scene; indeed, 98% of the utterances generated by the algorithm for this scene appear in the corpus. The remaining 9 REs generated by the algorithm, not present in the corpora, are very natural as can be observed in Table 1. The table lists the REs in the corpus and the REs generated by the algorithm using the learned p_{use}. For each RE, we indicate the number of times it appears in the corpus (#Cor), the proportion it represents (%Cor), the number of times it is generated by our algorithm (#Alg) and the proportion it represents (%Alg). Finally, the accuracy (%Acc) column compares the REs in the corpus with the REs generated by the algorithm. The accuracy is the proportion of perfect matches between the algorithm output and the human REs from the corpus. The accuracy metric has been used in previous work for comparing the output of an RE generation algorithm with the REs found in corpora (van der Sluis et al., 2007; Viethen, 2011) and it is considered a strict comparison metric for this task.

To put our results in perspective we compare in Table 2 our algorithm with a number of possible variations. All numbers shown in the table represent accuracy with the corresponding corpus. The first column shows the values obtained when we run the algorithm over the scene with the values of p_{use} obtained *from the scene itself*. As we could expect, this column has the highest average accuracy. The second column shows the results of the algorithm runs with p_{use} learned

	Scene p_{use}	Learned p_{use}	Random p_{use}	Uniform p_{use}
Scene 1	85.75%	84.49%	17.95%	5.37%
Scene 3	82.81%	80.51%	9.89%	4.40%
Scene 6	90.11%	83.30%	4.13%	4.16%
Scene 8	86.52%	64.06%	16.32%	9.75%
Scene 10	89.49%	75.80%	7.56%	3.70%
Scene 12	80.21%	81.29%	57.09%	6.68%
Scene 13	89.98%	50.79%	9.30%	3.59%
Scene 21	92.13%	80.01%	8.45%	6.77%
Average	87.13%	75.03%	16.34%	5.55%

Table 2: Accuracy between the REs in the corpus and those generated using p_{use} values computed from the scene, machine learned, random and uniform.

from corpora as explained in Section 3. In most cases the accuracy is rather high and the average accuracy is still high. The relatively low accuracy obtained in Scene 13 is explained mostly by the poor estimation of the p_{use} value for the *large* relation. In the corpus, relations *small* and *large* are used much more when the target cannot be uniquely identified using taxonomical (*ball* and *cube*) and absolute (*green* and *blue*) properties, but the features we used for machine learning do not capture such dependencies. In spite of this limitation, the average of the second column is 75%, indicating that p_{use} values learned from the corpus are good enough to be used to generate REs for new scenes from the domain. The last two columns can be considered as baselines. In the first one we generate random values for p_{use}. The accuracy obtained is in most cases poor, but with a noticeable variation due to chance. In addition to poor accuracy, when random p_{use} values were used many of the generated REs where unnaturally sounding like "small on the top of a blue cube that is below of something that is small." In the last column we present the accuracy for an artificial run, where all the REs generated in any of the previous columns were assigned the same probability.

We also computed the entropy of the probability distribution of REs found in the corpus, and the cross-entropy between the corpus distribution of REs and the execution of each algorithm we just described (see (Jurafsky and Martin, 2008) for details on cross-entropy evaluation). Figure 4 shows the results for the eight scenes we are considering. The cross-entropies from the first two runs (*scene* and *learned*) are, in general, much closer to the corpus entropy than *random*'s and *uniform*'s cross-entropies, and to each other. Only in Scene 12 *random* approaches, by chance, the other two.

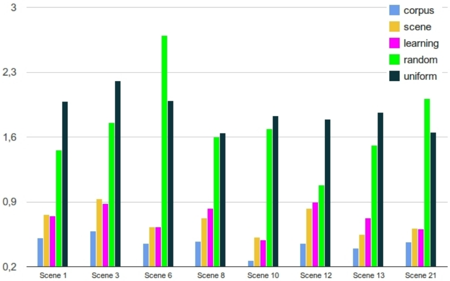

Figure 4: Cross-entropy between the corpus distribution and different runs of the algorithm

5 Discussion and Conclusions

We extend Areces et al. (2008) algorithm to generate REs similar to those produced by humans. The modifications we proposed are based on two observations. First, it has been argued that no fixed ordering of properties is able to generate all REs produced by humans and, second, humans frequently overspecify their REs (Engelhardt et al., 2006; Arts et al., 2011; Viethen, 2011). We tested the proposed algorithm on the GRE3D7 corpus and found that it is able to generate a large proportion of the overspecified REs found in the corpus without generating trivially redundant referring expressions. Viethen (2011) trains decision trees that achieve 65% average accuracy on the GRE3D7 corpus. This approach is able to generate overspecified relational descriptions, but they might fail to be referring expressions. Indeed, because the method does not verify the extension of the generated expression over a model of the scene, the generated descriptions might not uniquely identify the target. As we have already discussed, our algorithm ensures termination and it always finds a referring expression if one exists. Moreover, it achieves an average of 75% of accuracy over the 8 scenes used in our tests.

Different algorithms for the generation of overspecified referring expressions have been recently proposed (de Lucena and Paraboni, 2008; Ruud et al., 2012). To our knowledge, they have not been evaluated on the GRE3D7 corpus and, hence, comparison is difficult. de Lucena and Paraboni (2008) and Ruud et al. (2012) algorithms have been evaluated on the TUNA-AR corpus (Gatt et al., 2008) where they have achieved a 33% and 40% accuracy respectively. As the TUNA-AR corpus includes only propositional REs, it would be interesting future work to evaluate how these algorithms perform in corpora with relational REs such as GRE3D7.

The way we introduce overspecification is inspired by the work of Keysar et al. (1998) on egocentrism and natural language production. Keysar et al. argue that when producing language, considering hearers point of view is not done from the outset but it is rather an afterthought; adult speakers produce REs egocentrically, just like children do, but then adjust REs so that the addressee is able to identify the target unequivocally. The first, egocentric, step is a heuristic process based in a model of saliency of the scene that contains the target. Our definition of p_{use} is intended to capture the saliences of the properties for different scenes and targets. The p_{use} of a relation changes according to the scene. This is in contrast with previous work where the saliency of a property is constant in a domain. Keysar et al. argue that the reason for this generate-and-adjust procedure may have to do with information processing limitations of the mind: if the heuristic that guides the egocentric phase is well tunned, it succeeds with a suitable RE in most cases and seldom requires adjustments. Interestingly, we observe a similar behavior with our algorithm: when p_{use} values learned from the domain are used, the algorithm is not only more accurate but also much faster than when using random p_{use}values.

Besides testing our algorithm over the rest of the scenes in the GRE3D7 corpus, as future work we plan to evaluate our algorithm on more complex domains like those provided by Open Domain Folksonomies (Pacheco et al., 2012). We will also explore corpora obtained through interaction such as the GIVE Corpus (Gargett et al., 2010) where it is common to observe multi shot REs. Under time pressure, subjects will first produce an underspecified expression that includes salient properties of the target (e.g., “the red button”). And then, in a following utterance, they add additional properties (e.g., “to the left of the lamp”) to make the expression a proper RE identifying the target uniquely. The source code and the documentation for the algorithm are distributed under the GNU Lesser GPL and can be obtained at `http://code.google.com/p/bisimulation-gre`.

Acknowledgments. This work was partially supported by grants ANPCyT-PICT-2008-306, ANPCyT-PICT-2010-688, the FP7-PEOPLE-2011-IRSES Project “Mobility between Europe and Argentina applying Logics to Systems” (MEALS) and the Laboratoire Internationale Associé “INFINIS”.

References

Areces, C., Figueira, S., and Gorín, D. (2011). Using logic in the generation of referring expressions. In Pogodalla, S. and Prost, J., editors, *Proceedings of the 6th International Conference on Logical Aspects of Computational Linguistics (LACL 2011)*, volume 6736 of *Lecture Notes in Computer Science*, pages 17–32, Montpelier. Springer.

Areces, C., Koller, A., and Striegnitz, K. (2008). Referring expressions as formulas of description logic. In *Proceedings of the 5th International Natural Language Generation Conference (INLG'08)*, pages 42–49, Morristown, NJ, USA. Association for Computational Linguistics.

Arts, A., Maes, A., Noordman, L., and Jansen, C. (2011). Overspecification facilitates object identification. *Journal of Pragmatics*, 43(1):361–374.

Baader, F., McGuiness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.

Dale, R. (1989). Cooking up referring expressions. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 68–75.

Dale, R. and Haddock, N. (1991). Generating referring expressions involving relations. In *Proceedings of the 5th conference of the European chapter of the Association for Computational Linguistics (EACL'91)*, pages 161–166.

Dale, R. and Reiter, E. (1995). Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

de Lucena, D. J. and Paraboni, I. (2008). USP-EACH frequency-based greedy attribute selection for referring expressions generation. In *Proceedings of the 5th International Conference on Natural Language Generation (INLG 2008)*, pages 219–220. Association for Computational Linguistics.

Engelhardt, P., Bailey, K., and Ferreira, F. (2006). Do speakers and listeners observe the gricean maxim of quantity? *Journal of Memory and Language*, 54(4):554–573.

Gargett, A., Garoufi, K., Koller, A., and Striegnitz, K. (2010). The give-2 corpus of giving instructions in virtual environments. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*, Malta.

Gatt, A., Belz, A., and Kow, E. (2008). The tuna challenge 2008: Overview and evaluation results. In *Proceedings of the 5th International Conference on Natural Language Generation (INLG 2008)*, pages 198–206. Association for Computational Linguistics.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Jurafsky, D. and Martin, J. (2008). *Speech and Language Processing*. Pearson Prentice Hall, second edition.

Kelleher, J. and Kruijff, G.-J. (2006). Incremental generation of spatial referring expressions in situated dialog. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1041–1048.

Keysar, B., Barr, D. J., and Horton, W. S. (1998). The Egocentric Basis of Language Use. *Current Directions in Psychological Science*, 7(2):46–49.

Pacheco, F., Duboue, P., and Domínguez, M. (2012). On the feasibility of open domain referring expression generation using large scale folksonomies. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 641–645, Montréal, Canada. Association for Computational Linguistics.

Paige, R. and Tarjan, R. (1987). Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

Ruud, K., Emiel, K., and Mariët, T. (2012). Learning preferences for referring expression generation: Effects of domain, language and algorithm. In *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*, pages 3–11, Utica, IL. Association for Computational Linguistics.

Stone, M. (2000). On identifying sets. In *Proceedings of the 1st International Natural Language Generation Conference (INLG'00)*, pages 116–123.

van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52.

van der Sluis, I., Gatt, A., and van Deemter., K. (2007). Evaluating algorithms for the generation of referring expressions: Going beyond toy domains. In *Proceedings of Recent Advances in Natural Language Processing*.

Viethen, H. A. E. (2011). *The Generation of Natural Descriptions: Corpus-Based Investigations of Referring Expressions in Visual Domains*. PhD thesis, Macquarie University, Sydney, Australia.

Viethen, J. and Dale, R. (2006). Algorithms for generating referring expressions: Do they do what people do? In *Proceedings of the 4th International Natural Language Generation Conference (INLG'06)*, pages 63–70.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1):1–191.