# Size matters: Grounding quantifiers in spatial perception

Pauw, S.

**Publication date**
2013

# Chapter 7

# Size Matters: Adjectival Origins

Gradable quantifiers (e.g., *many* and *few*) have a dual nature: They can behave both as quantifiers and adjectives. Using a framework in which agents can self-organize a language, I examine the hypothesis that their adjectival use can be explained by the cognitive overlap between these quantifiers and adjectives such as *big* and *small*. This chapter is a version of a forthcoming publication: Pauw (2013c).

## 7.1 Introduction

Words like *many* and *few* are syntactic hybrids: though traditionally analyzed as quantifiers ("many of the houses"), they can also behave like gradable adjectives ("few/fewer houses"). In fact, such terms pattern syntactically and semantically with both quantifiers and adjectives (see Solt, 2009, for an extensive analysis of these syntactic patterns). Why aren't they confined to one grammatical class? What is the cognitive basis for their dual behavior? And how might such conceptual and linguistic duality have evolved?

Cross-linguistic properties of these terms (henceforth, *gradable quantifiers*) suggest close ties between the functions of quantification and predication. Some languages have no separate grammatical class for expressing number, but instead use size-modifying adjectives, as in the extension of the Pirahã predicate *hi* 'small' to indicate 'a small number' (Everett, 2005). Historical evidence also suggests that gradable quantifiers typically derive from adjectives, as illustrated by the quantifier *few*, based on the Old English adjective *feawe* (Solt, 2009).

This chapter explores the hypothesis that the duality of gradable quantifiers has its roots in the close cognitive relationship between size and number. Judgments of size (underlying modifiers such as *big* and *small*) depend on perceptual features of objects (or sets of objects) in the environment. Judgments of approximate number (underlying terms like *few* and *many*) exploit a combination of spatial features that apply exclusively to sets of objects, such as their size and

density (Durgin, 1995). This cognitive overlap between the concepts of size and number may account for the duality observed in gradable quantifiers: the dependence on size motivates their adjectival uses, while their application to sets of objects motivates their quantificational uses.

In the next section of this chapter I introduce a computational model that captures the insight above within the evolutionary language games framework (Steels, 2012a), in which robotic agents self-organize the means for describing objects (or in this case, groups of objects) in their perceived environment. Within this framework, agents are encouraged to develop a system of gradable quantifiers. The following section of this chapter, introduces two innovation strategies: Agents can either develop new quantifiers from scratch or they can develop quantifiers by extending the meaning of cognitively similar lexical items (e.g., gradable adjectives). In the last part of the chapter it is shown that when left to their own devices, the agents will prefer the latter strategy and therefor converge on a set of quantifiers that find their origins in adjectives.

## 7.2 Robotic Language Games

The experiments we describe are based on communicative interactions between humanoid robots (see (Steels, 2012a) for specific details). Figure 7.1 shows an example scene with two robotic agents interacting in a shared environment. We use a setup that has been used in many other experiments (Spranger, 2012; Pauw and Hilfery, 2012; Pauw and Spranger, 2010). Each robot perceives the world through its own on-board sensors, e.g., a camera and proprioceptive sensors. From this multimodal sensory input, the robots build a world model, which reflects the robot's current belief about the state of the environment.

Conducting experiments with actual robots is time consuming. I therefore reuse recorded data from actual robotic interactions to speed up our experiments while at the same time preserving the realism of physical robot interactions. Every scene contains two robots. The recorded data of the scene comprises the world models that both robots have built from their visual input using standard machine vision algorithms. In the world model of a scene, every object is represented as a list of features such as their width, height, color and position. (See, for example, Figure 7.2.)

Using this data, we let a community of agents play *language games* (Steels, 2012a). The type of language game the agents play depends on the particular research question. For the purpose of this chapter we let them play *Multi-Word Guessing Games* (Steels, 2012b). This game uses the following script:

1. One of the agents takes the role of a speaker, the other takes the role of hearer.

2. The speaker has to pick an object or a group of objects as the *referent*. This

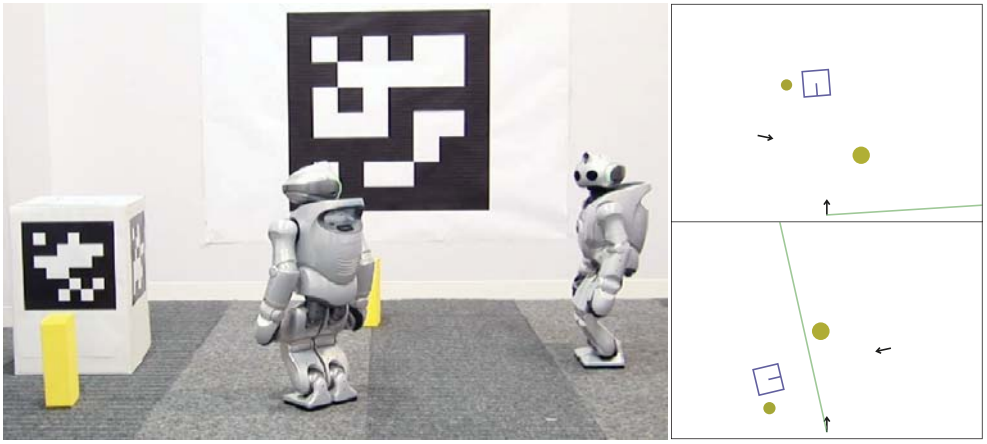


Figure 7.1: An example of a robotic interaction. Robots are placed in an office-like environment that contain different types of objects. This scene contains two yellow blocks, one box and two robots themselves. The world models of the robots are shown on the right. In the world models, the arrows represent the robots. The direction of the arrow marks the orientation of the robot. The circles represent the blocks and the blue square represents the box.

```
(object obj-228
 ((x 1868.59) (y 108.199) (z 0)
  (width 138.031) (length 131.142) (height 226.175)
  (average-y 85) (stdev-y 11.5768) (min-y 0) (max-y 112)
  (average-u 188) (stdev-u 14.1542) (min-u 0) (max-u 208)
  (average-v 122) (stdev-v 5.54406) (min-v 0) (max-v 128)))
```

Figure 7.2: An example representation of a block. Objects do not come pre-classified or predicated in any way. They are a list of continuous features. For this particular example we see its euclidean coordinates (in $mm$), its dimensions (in $mm$), and its color coordinates (in yuv color space).

is done randomly.

3. The speaker has to find an utterance that most clearly singles out the referent.

4. (a) If the speaker was capable of finding an utterance, the hearer has to interpret the utterance and point to the object or group of objects that he thinks that speaker intended.

   (b) If the speaker could not find an utterance, the game is considered a failure and ends here.

5. (a) If the hearer points correctly, the game is considered a success and ends here.

   (b) Otherwise the game is considered a failure and the speaker will point at the intended referent (thus creating a learning opportunity for the hearer).

So, this game can either end in a success (5a) or in a failure (4b and 5b). In the case of failure, the agents are provided with two *repair operators*, a *learning* and a *innovation* operator. The learning operator is used case of a failure on the hearer side (5b). The innovation operators is used in the case of a failure on the speaker side (4b). This operator extends the language of the speaker. Together these operators allow us to look at how a language can gradually change over time to incorporate novel communicative needs.

## 7.3 Model

Before looking at the experimental data, we need to have a more detailed look at the communicative machinery of the agents. The experiments reported in this chapter are certainly not built from scratch. They rely on mechanisms that have been developed and tested in other related experiments (Spranger et al., 2010b; Pauw and Spranger, 2010). The model that is described in this section is an extension of an existing model that has been used to study spatial language (Spranger, 2012) and quantification (Pauw and Hilfery, 2012).

For conceptualization this model uses a degree-based semantics that is built on top of a prototype system (Rosch et al., 2004; Lakoff, 1987). This combination of prototype theory and degree-based semantics has proven particularly successful in dealing with the problems that are inherent to conceptualizing real-world perception. (See Spranger and Pauw (2012); Pauw and Spranger (2012) for an in depth discussion.)

This semantics is implemented in a formalism called Incremental Recruitment Language (IRL) (Spranger et al., 2012b; Van Den Broeck, 2008; Steels, 2000b), a procedural semantics that is designed for semantic planning in robotic language

games. For parsing and production of language the agents rely on Fluid Construction Grammar (FCG) (Steels and De Beule, 2006). IRL and FCG have been co-developed specially for the kind of grounded language games as described in this chapter. The rest of this section discusses the different technical aspects of this model in more detail.

### 7.3.1 Conceptual system - IRL

In the present scenario, the meaning of language is grounded in the real world. That is to say, the robotic interactions rely on data from the vision system of these robots. This data does not come pre-classified, but rather every entity in the environment is represented as a list of continuous features. (Figure 7.2 shows an example representation of a block in the world model of a robot.) The task of the conceptual system is to bridge the gap between the continuous perception of the real world and the discrete conceptual world.

#### Prototype Theory

A combination of IRL and prototype theory has proven very successful in conceptualizing real-world perception. In prototype theory, every concept is represented by a *prototype*. A prototype is a prototypical value (or vector) in the relevant domain. For example, the concept *big* is defined over the surface area of an entity. The closer the surface area is to the prototypical value for *big*, the more similar it is to the concept *big*.

This intuition of similarity is captured by a similarity function. The similarity function takes a prototype and an entity and returns a confidence score between 0 and 1. If the score is 0, the entity is not similar at all to the prototype, if it is 1, the entity is precisely like the prototype. Consider for example the following similarity function for area prototypes:

$$S_{area}(P, O) = e^{\frac{1}{\sigma_P}|\text{area}(P)-\text{area}(O)|} \tag{7.1}$$

This is a typical similarity function in prototype theory. In this case it is defined over the feature *area*, but similar functions can be defined over any feature or set of features. The prototype defines a prototypical value for the relevant feature (area($P$)) and a rate at which the confidence score decreases with the distance to this value ($\sigma_P$).

For gradable modifiers such as *big* and *small*, however, this sketches a slightly oversimplified picture. These modifiers require a comparison to some context-dependent norm. In Pauw and Hilfery (2012) it is shown in detail how the notions of prototype- and exemplar-based models can be adapted to model this norm dependence. For every type of object, the agents know what a normal amount is

(the average size of all objects of the specific object type)[1]. The prototypes are scaled with respect to this amount. On top of that, the prototypes have an open side (i.e., on one side of the prototype, the score is always 1). (See Figure 7.3 for an example of the scaled prototypes for the object type block.)

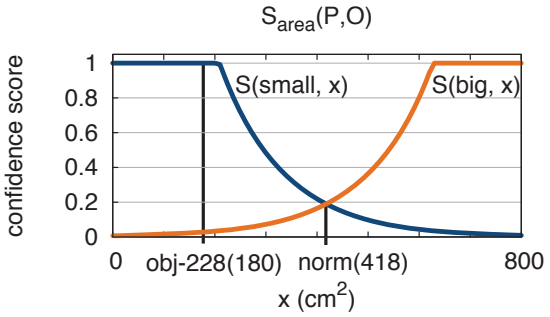



Figure 7.3: The area similarity function for blocks. The average block size is $418cm^2$. This is the norm. Any block that is bigger than that norm is gets higher confidence score for the *big*-prototype, any block that is smaller gets a higher confidence score for the *small*-prototype. The prototypes have an open side (i.e., on one side of the prototype, the score is always 1).

If we want to know if `obj-228` in Figure 7.2 is a small or a big block, we simply take its surface area and compare it to the *small* prototype using the above similarity function. The surface area of `obj-228` is $181cm^2$ (width = $138.031mm$ and length = $131.142mm$). If we enter this value in the similarity function above together with the *small* prototype we get a similarity value of 1. For *big* the similarity value is 0.03. So, the least we can say is that *small* is a much better way to describe object `obj-228` than *big* is.

### Composition

Every concept in the agents' ontology is defined in the same way, using prototypes and similarity functions. And, if agents would only need one concept to describe an object this would be the end of the story. Most utterances, however, involve more than one concept. Take for example, the utterance "big block", this contains both the concept *big* and *block*. We need some mechanism to combine both concepts. This is done by first computing the similarity scores for both concepts individually and than using a mixture function that computes a total score.

[1]It should be noted that the norm has different definition here than in Chapter 6. In Chapter 6 norm is defined as the maximal observed amount. Here, the norm is taken to be an average amount. The difference is mainly a technical design choice. The reason to change the definition is to be in line with other semantic theories (e.g., Fernando and Kamp, 1996).

There are many possible ways to define such a mixture function (e.g., the product of the two scores, or their average) and in studies of degree-based semantics, this is subject to much debate (van Rooij, 2011). For this experiment I have adopted Zadeh's definition for conjunction by using the minimum of the scores (Zadeh, 1965). All philosophical ramifications aside, there is a very practical reason to use the minimum as mixture function: The scores are likely to get lower when there are more concepts involved, thus discouraging the agents to come up with overly long and convoluted descriptions.

In sum, a concept (using prototypes and similarity functions) assigns a score to every entity in the environment. Using the minimum as a mixture function the scores of multiple concepts can be combined. So, when interpreted, an utterance like "big block" assigns a score to every entity that reflects how well this entity is described by the utterance.

The question remains: How are these scores being used in communication? The goal for the hearer is to point at the referent he thinks the speaker intended. So the hearer can just simply take the entity that has the highest score. For the speaker the story is slightly more complicated: For conceptualization, IRL has a search mechanism that finds a set of prototypes that together best describe the referent. In order for a description to be good, it should not only have a high score for the referent, but at the same time a low score for all other entities. IRL finds the set of prototypes that together maximize the difference in score between the referent and all other entities, a process called *contrast maximization* (see Chapter 3 of this thesis for more information on the search mechanism and Chapter 4 for an extensive study of the principle of contrast maximization in real-world perception).

### Quantifiers

The mechanism described above shows how concepts in the form of prototypes can be used and combined to find descriptions of single objects. However, this chapter focuses on quantifiers and quantifiers typically do not describe single objects, but sets of objects. So, how are sets of objects modeled? And, how are quantifiers used to describe them?

For the experiments described in this chapter, it is assumed that there is a fixed set of object groups in the world model of the agent. So, all the groups the agents could talk about are know before any communication has taken place. The groups are computed directly from the visual data using a standard clustering algorithm called agglomerative clustering (Mitchell, 1997). The algorithm creates a hierarchy of groups based on their spatial arrangement. Figure 7.4 shows an example scene containing object groups that were created using this agglomerative clustering approach. The amount of groups that is created depends on the granularity of the algorithm, in this example it happens to create three groups.

More important than how the object groups are computed is their represen-

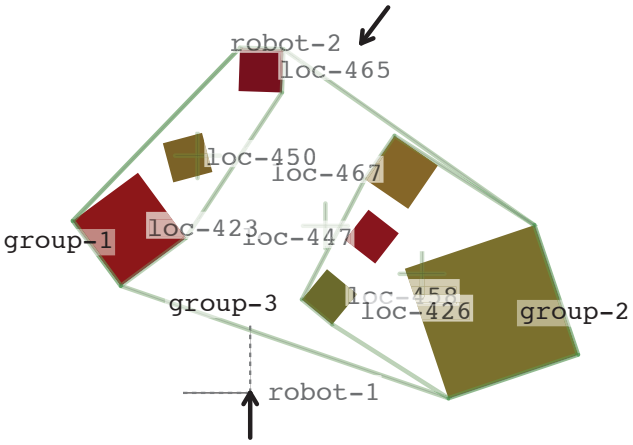
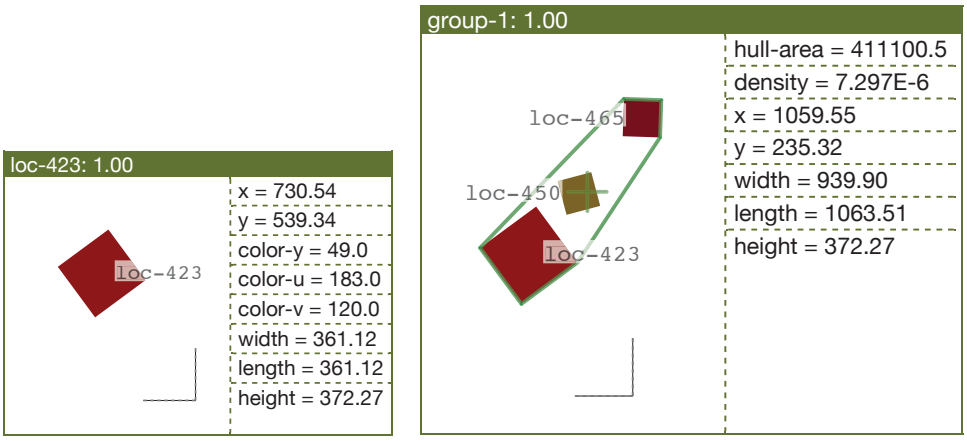



Figure 7.4: Top down view of an example scene. The arrows represent the speaker (*robot-1*) and the hearer (*robot-2*). The scene contains seven blocks (*loc*-x) and three object groups: *group-1*, containing the three leftmost blocks; *group-2*, containing the four rightmost blocks; and *group-3*, containing all blocks in the scene.

tation. Objects groups are represented in exactly the same way as single objects; i.e., a list of features. Most of the features that object groups have are the same as those of single objects (such as width, height, position, color parameters, ...). So, without any change to model, the agents can already use the concepts that are used for single objects to describe groups of objects, using utterances such as "the big [group of] objects", or "the [group of] objects to the left".

But, the objects groups, have an additional feature that single objects do not have: the density of the object group (the total surface area of the constituents divided by the surface area of the entire group). And, although the agents already have means to describe groups of objects without taking this feature into account, the agents can come across situations where it is useful to have means to take this density feature into account. And, as we will see in a moment, it is precisely this density feature that is being used by (gradable) quantifiers.

In order to fit the definition of the quantifiers *many* and *few* within the prototype framework, Pauw and Hilfery (2012) employ a exemplar-based approach, where the similarity between the cardinality of an object group and the concept is determined. There is a problem with this approach however: It requires that the agents know the precise cardinality of object groups. From a cognitive point of view this is not very plausible. When we use an utterance like "many blocks", we do not explicitly count the number of relevant objects, but we estimate the number of object. The cognitive mechanism that is responsible for number estimation is often referred to as the *Approximate Number System* (ANS) (Halberda and Feigenson, 2008). The precise working of the ANS is subject to much debate,



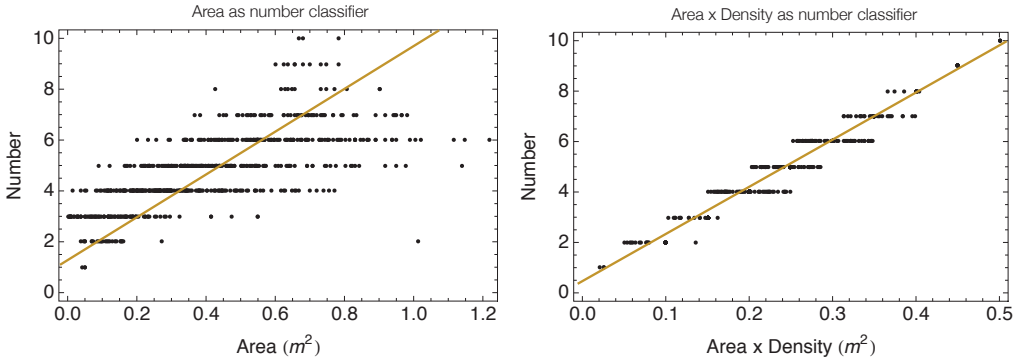

(a) Block (b) Object Group

Figure 7.5: Both single objects and groups of objects are represented as list of features. Most prototypes that apply to single objects can just as well be used for object groups.

but there seem to be two dominant features that influence the estimation of the amount of objects in a group: the *size* (area/volume) that the group of objects as a whole occupies (Hormann, 1983; Newstead and Coventry, 2000) and the *density* of the objects in the group (Durgin, 1995). (See Chapter 2 of this thesis for an in-depth discussion.)

In fact, when we look at the robot data, we can see that these features correlate very strongly with the number of objects in that group. Consider Figure 7.6(a). This figure shows both the surface area and number of objects for every possible group in all recorded scenes. There is a strong correlation between both variables ($\rho = 0.81$), making surface area a good classifier for number. But, we can do better: Figure 7.6(b), shows that, in line with the observation above, if we also take the density into account, we get an even better classifier. The density of an object here is computed by dividing the total surface area of all objects by the surface area of the group as a whole (including the space between the objects). By multiplying this density with the surface area of the group we get a value that correlates very strongly with the number of the group ($\rho = 0.96$).

Based on these observations, we can define the prototypes for the gradable quantifiers *many* and *few*. These prototypes do not have one dimension like *big* and *small*, but are defined over two dimensions: surface area and density. The similarity function for the Approximate Number System ($S_{\text{ANS}}$) then also has to take these two dimensions into account. The function $S_{\text{ANS}}$ is essentially an extension of $S_{\text{area}}$ that was used for the concepts *big* and *small*, but now also




(a) Area ($\rho = 0.81$) (b) Area×density ($\rho = 0.96$)

Figure 7.6: The correlations between number of objects, area and density. These measures are taken over all 900 recorded scenes. Every point represents a group of objects. The y-axis shows the number of objects in the group the x-axes shows its (a) area and (b) area times density. Both measures correlate with number, but the latter more strongly.

takes density into account:

$$S_{\text{ANS}}(P, O) = e^{\frac{1}{\sigma_p}|\text{area}(P)\cdot\text{density}(P) - \text{area}(O)\cdot\text{density}(O)|} \tag{7.2}$$

In sum, in this model, object groups are modeled in the same way as single objects. This means that they can be described using the same prototypes as used for describing single objects. However, object groups have the additional density feature that can be used to define concepts that exclusively apply to these object groups. The gradable quantifiers *many* and *few* are examples of such concepts. They extend the size-concepts *big* and *small* by taking the density feature into account.

It should be mentioned that this model of quantification is very different from most existing approaches. In most existing approaches quantifiers have a different functional status than adjectives. For example, in Generalized Quantifier Theory(Barwise and Cooper, 1981) adjectives are simple predicates and quantifiers are relations between predicates. Also earlier similar computational experiments assumed that quantifiers and adjectives are functionally different (Pauw and Hilfery, 2012; Spranger and Pauw, 2012). For the present experiment however, it is essential to let go of this functional difference because this allows gradable quantifiers to emerge as adjectives.

### 7.3.2 Grammar - FCG

The previous section describes how the agents can conceptualize the world using prototypes and similarity functions. Here we will look at how these concepts are communicated. In the experiments to follow the agents can use common nouns (e.g., "big blocks") and noun phrases (e.g., "two big blocks") to describe the (groups of) objects in their environment. The agents are provided with a basic context-free grammar that allows them to recursively build common nouns and noun phrases. The grammar is implemented in Fluid Construction Grammar (FCG) (Steels and De Beule, 2006), a construction grammar that links the syntactic structure of an utterance directly to the semantic operations of the agent. Figure 7.7 shows the syntactic structure of the utterance "two big blocks".

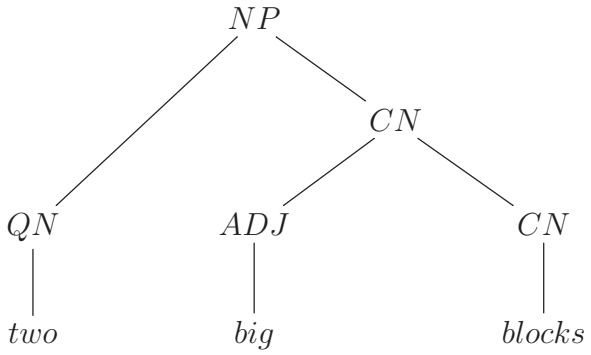
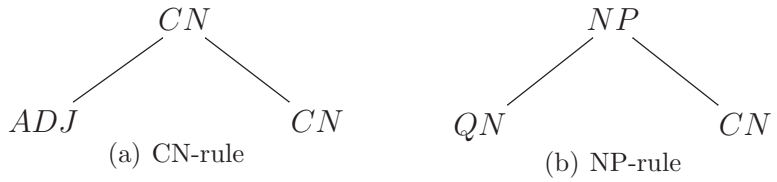



Figure 7.7: Syntactic structure of "two big blocks". FCG combines the adjective (ADJ) 'big' and the common noun (CN) 'blocks' into the CN "big blocks", which together with the quantifier (QN) 'two' forms the noun phrase (NP) "two big blocks".

The grammar consists of lexical rules that link the labels to the concepts (prototypes) and their respective grammatical category. For example, there is a lexical rule which links the label 'big' to the category adjective and the *big*-prototype. There are three grammatical categories for lexical items: adjectives (ADJ ) such as *big* and *left*, quantifiers (QN) such as *two* and *three*, and common nouns (CN) such as *block* and *box*.

Furthermore, the grammar contains two grammatical constructions that recursively combine the lexical items into more complex CN's and NP's (as shown in Figure 7.8). Note that these constructions are defined in a construction grammar, so they do not only define the syntactic patterns, but they also have semantic contents (Goldberg, 1995). They instruct the agent how the concepts of its constituents should be interpreted.

The CN rule instructs the agents to apply both the prototypes of the ADJ




(a) CN-rule (b) NP-rule

Figure 7.8: Grammatical constructions. The CN rule The NP rule combines a QN and a CN into a NP.

and of the CN to the context and use the mixture function (as defined above) to combine the two. For example, in Figure 7.7, the CN instructs the agent to apply both the *block*-prototype and the *big*-prototype to the context and combine the results. The NP rule does the same thing, but additionally instructs the agents to ignore single objects and only consider groups (sets) of objects. In theory the length of the CN's that this grammar produces is unbounded. However, the amount of used modifiers are limited by the conceptual system. So, in practice, the agents will mostly construct noun phrases containing only one or two modifiers.

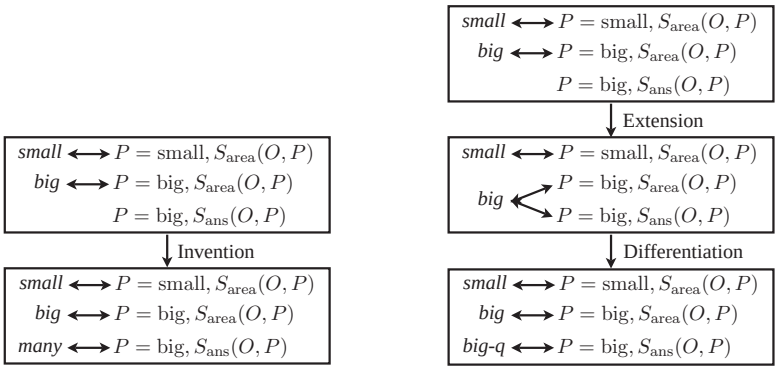
## 7.4 Language Innovation

Now we can turn to the main topic of this chapter: The evolution of gradable quantifiers. We provide the agents with lexical items to express spatial relations (*left*, *right*, *front*, *back*, *near* and *far*), size (*big* and *small*), object class (*block*, *box* and *robot*) and the grammatical rules to combine those items into nouns and noun phrases such as "block in front of you". We let the agents play language games as described above in which the agents have to describe (groups of) objects to each other. One of the salient properties of groups of objects can be the (approximate) number of objects it contains. Since the agents do not have any predefined vocabulary for expressing number they will have to develop this themselves.

To this end, I provided the agents with three operators: an innovation operator, which allows an agent to invent a new language item when required; an adoption operator, with which agents can learn the meaning of unfamiliar language items; and an alignment operator, which keeps track of the success of a language item.

### Innovation

The *innovation operator* is used when an agent wants to express a meaning for which it does not yet have a lexical item, so it has to establish a relation between a (new) lexical item and the concept that could previously not be expressed. This




(a) Invention Strategy (b) Extension Strategy

Figure 7.9: Two contrasting formation strategies. When the need arises to express a new meaning ($P$ = big, $S_{\text{ans}}(O, P)$), the agents can either (a) create a new random lexical item for it (*many*), or (b) extend an existing lexical item (*big*). The latter strategy introduces homonyms. If the ambiguity that arises from such a homonym proves to be problematic, agents can invent a marker to differentiate (*-q*).

can be done in many ways. In most existing experiments that model language evolution, new language items are invented from scratch. I.e., as soon a novel concept needs to be expressed, the innovation operator randomly generates a label, which is than linked to that concept. Although fine for the purpose of those experiment, this innovation strategy is too much of a simplification for the present experiments. This chapter aims to test the idea that gradable quantifiers are derived from adjectives, this means that we cannot assume that they are invented from scratch. So, in the present chapter we compare two formation strategies (see also Figure 7.9):

1. The (traditional) invention strategy: Every time an agents wants to express a new meaning, a new random lexical items is created and mapped to this meaning.

2. The extension strategy: The agent tries to find existing lexical items that have a meaning that is *similar* to the new one. If it can find such a lexical item, it will extend its meaning rather than inventing a new lexical item. Remember that the meaning of a word is defined as a prototype and a similarity function. Thus, two meanings are identical when they have both the same prototype and similarity function. We say that two meanings are *similar* if they contain either the same prototype or the same similarity function.

   The extension strategy introduces an additional problem: It creates ambiguity. If a word has multiple meanings this can lead to communicative

failure. If such is the case, the speaker will notice this and he can decide to introduce a marker to differentiate the two interpretations.

### Adoption

Once an agent has created a new lexical item, this his to be learned by other agents in the community. This is taken care of by the *adoption operator*. This operator is used every time an agent hears an unfamiliar language item. In essence the adoption operator is the mirror-image of the innovation operator. The operator relies on a mechanism in IRL called *flexible interpretation*. When part of an utterance is missing, the agent can only derive part of the conceptual structure. The flexible interpretation mechanism can recover the missing part of the conceptual structure by trying out which combination of available operations and prototypes results in the discrimination of the referent, in the given context. The reconstructed part of the meaning is taken to be the meaning of the unfamiliar lexical item. Most of the time this operator creates many possible solutions.

The question is remains of how to pick one of those solutions. First of all, the amount possible solutions can be further limited. The adoption operator looks in the lexicon of the agents for similar lexical items. If such an item is found it is assumed that the new item was created by the extension strategy and therefore, the meaning of the new lexical item should be similar to the meaning of the similar lexical item. Secondly, if multiple solutions are found they can all be added to the lexicon. In such a case, the agents will have multiple language constructions that stand in competition.

### Alignment

Using the invention operator, a community of agents will introduce new lexical item. Most of the time, a community of agents comes up with multiple lexical items that describe the same concept. These language items are in competition with each other. This is where the *alignment operator* comes into play. The language items are all provided with a score between 0 and 1. When the items are newly created (either through innovation or adoption), the score is set to 0.5. After a successful interaction the operator increases the score of all the language constructions that were involved and, conversely, after every unsuccessful interaction the scores are decreased[2]. The scores are updated using the following update function:

$$S_{n+1}(l) = S_n(l) + \lambda S_n(l)(1 - S_n(l))$$

Where $S_n(l)$ is the original score and $S_{n+1}(l)$ is the updated score of lexical item $l$ and $\lambda$ is a rate at which a single update influences the score. For successful

[2]Note that the alignment operator in this chapter differs from the one described in Chapter 6. Here the alignment operator only adapts the score of the language item, whereas the alignment operator in Chapter 6 can also modify the underlying concept.

interactions $\lambda = 0.1$ for unsuccessful interactions $\lambda = -0.1$. The function is essentially a numerical approximation of an S-curve. This means that when the score gets closer to its extreme values (0 or 1), the rate at which the score changes gets lower. Update functions with this property have proven to be more stable than linearly updating scores for similar experiments. (I.e., a false positive or negative has less influence on the confidence score.)

So, when a community of agents is confronted with a novel problem (e.g., describing a group of objects in terms of its estimated number), these three operators assure that the community extends the existing language to accommodate this. The innovate operator creates new lexical items, and the adoption and alignment operator assure that the new items are spread through the community.

## 7.5 Experiments and Results

We can now look at how a community can use the described model and the innovation strategies to bootstrap a system of graded quantifiers. In what follows I will describe three experiments: A first baseline experiment establishes the expected communicative success of the quantification system. In this experiment the agents do not yet have to invent gradable quantifiers themselves. In a second experiment, the performance of both innovation strategies are tested individually. The last experiment finally shows that when the choice of strategy is left to the agent, they will be inclined to develop quantifiers that derive from adjectives.

### 7.5.1 Baseline

The first part of experimental section consists of a baseline experiment. Figure 7.9 describe different stages of development for both the extension and the invention strategy. Before looking at how agents can bootstrap a quantifier system, I implemented the language at each stage manually, considering four different stages of language development:

1. The agents have no gradable quantifiers. This is the initial state for both the invention and the extension strategy.

2. The agents are provided with size words that are extended to incorporate the ANS meaning. This is the second stage of the extension strategy.

3. The polysemy of the size markers is resolved by marking the different interpretations. This is the final stage of the extension strategy.

4. The estimate number is expressed using novel lexical items. This is the second and final stage of the invention strategy.

On top of these four different language conditions, there are two different environmental conditions: *Salient number*—the context and topic is chosen such that there is a guaranteed difference in the number of objects in the topic and any other group of objects in the context. In this condition, quantity always the most (and often, the only) salient feature of an object group; *No manipulation*—the topics and contexts are chosen completely at random from the data set. Figure 7.10 shows the average communicative success of 2000 interactions for each of these $2 \times 4$ different experimental conditions.

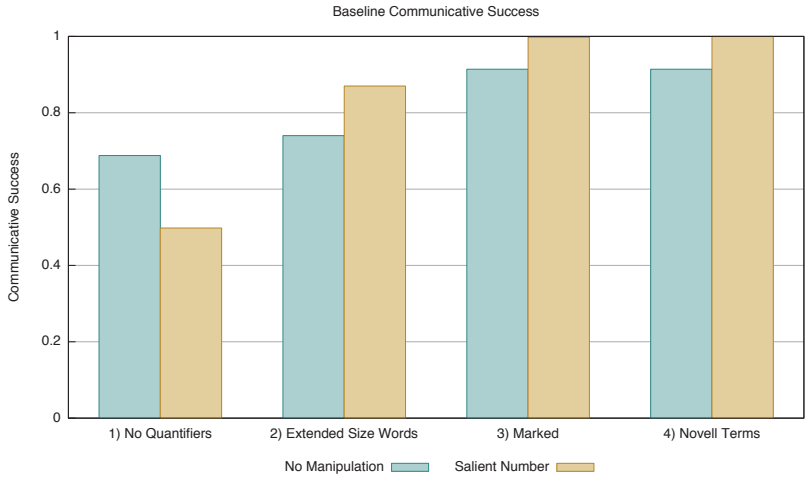



Figure 7.10: Results baseline, 2000 interactions, 2 environmental conditions, 4 linguistic stages. The height of the bar indicates the average communicative success over 2000 interactions.

The results show first of all that even without quantifiers, the agents can communicate. The number of objects is only one of the features they can use to describe a group of objects. Also, without quantifiers the agents are much less successful in the salient-number condition (50% success) then when there is no environmental manipulation (73%). Secondly, for both environmental conditions these results show that extending the size words strongly increases the communicative success (89% for the salient-number condition) and even more so when this difference is marked (100% for the salient-number condition). This effect is stronger for the salient-number environmental condition.

It should also be noted that there is no difference at all in communicative success between the third and fourth language condition. In other words, the end stages of both innovation strategies are equally successful (86% for the no-manipulation condition, 100% success for the salient-number condition). It is only the intermediary stage of the extension strategy which makes a difference.

### 7.5.2 Strategy Comparison

Now we can look at how agents can bootstrap a language using the proposed innovation strategies. In Figure 7.11, both the extension and invention strategies are tested individually. The Figure contains one a graph for both environmental manipulation: no manipulation and salient number. Both graphs show the results for both innovation strategies. For every condition I ran ten series of 1200 language game interactions. The graphs show the average progression of communicative success and lexicon size over those ten series.

These graphs show several things. To begin with, for both environmental conditions the community of agents reach the communicative success that was established in the baseline experiment (86% for the no-manipulation condition, 100% success for the salient-number condition). For the final communicative success it does not matter whether the agent employ the extension or invention strategy. Using either strategy they converge on the same communicative success.

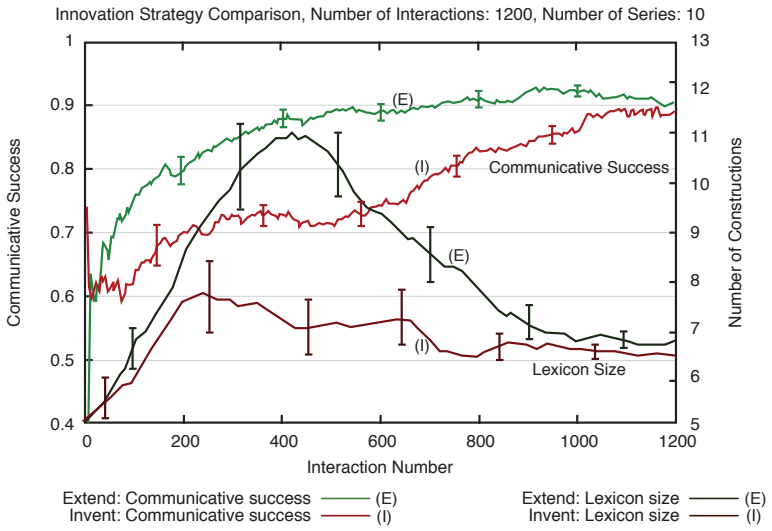
Second, the graphs show that there is an initial overshoot of the lexicon size. Initially all agents invent there own way of expressing quantity. Only when the lexicon becomes shared by the entire population (through the adoption operator) the unsuccessful lexical items gradually disappear and the language system converges on a smaller but more effective lexicon.

Of most interest is that, although the final success of both strategies is the same, there is a big difference in the rate at which this success is reached. Using the extension strategy a community of agents converges on a shared quantifier system much faster. For example in Figure 7.11(a), using the extension strategy the community of agents reach 80% communicative success after 200 interactions, whereas with the invention strategy this takes 800 interactions.

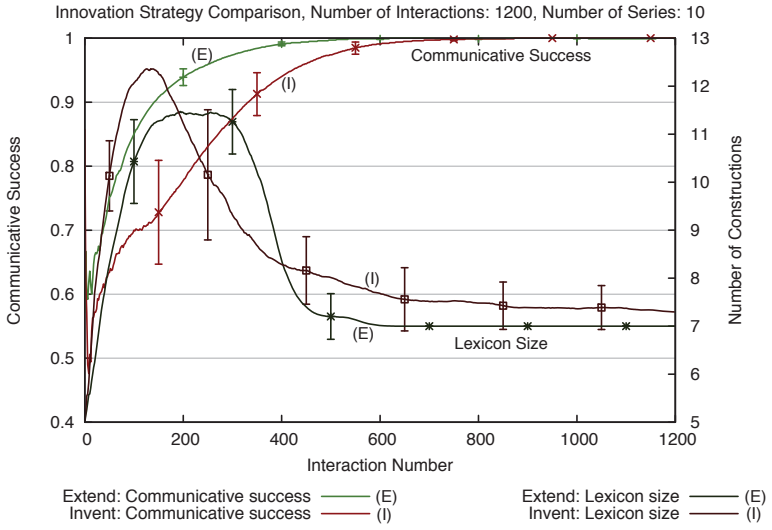
Finally, there is a clear difference between the two environmental conditions in this graph. Under the salient-number condition the agents start at only 50% success and reach 100% success in the end. This effect is much less spectacular without manipulation of the environment (73% to 86% communicative success). However, the advantage of the extension strategy become much more apparent for the no-manipulation environmental condition.

### 7.5.3 Strategy Competition

Figure 7.12 shows what happens if we leave the agent to his own devices in choosing a innovation strategy. Every time an agent needs to express a new meaning, it can choose to either follow the invention strategy and invent a completely new lexical item, or follow the extension strategy and adapt an existing lexical item. When agents randomly pick a strategy every time they require a new language item, they will wind up with a lexicon containing both gradable quantifiers that are derived from adjectives and quantifiers that are created from scratch. Figure 7.12(a) shows and example of an experiment where initially both types of quan-




(a) Strategy Comparison. No Manipulation




(b) Strategy Comparison. Salient Number

Figure 7.11: Results strategy comparison. These graphs show the average progression of the communicative success and lexicon size of the agents over 10 series of 1200 interactions. Both graph show the results of both the extension and invention strategy. Figure (a) shows the results for the no-manipulation condition and Figure (b) the results for the salient-number condition. The horizontal axis shows the number of interactions; left vertical axes, the communicative success; and the right vertical axis the lexicon size.

tifiers are invented, but the quantifiers that survive in the end have adjectival origins. Figure 7.12(b) shows a similar experiment, but here the community of agents converge on quantifiers that were created from scratch.

I have conducted this experiment many times, checking at the end of each series on which type of quantifiers the agents converged. The result strongly depends on the likelihood of either strategy being used. Figure 7.12(c) shows the likelihood that the converged quantifiers have adjectival origins as a function of the likelihood that the extension strategy is being used for innovation.

The graph shows no surprising results for the extreme cases: If the agents always use the extension innovation strategy the quantifier origins are always adjectival. And, conversely if the agents never use the extension strategy (in other words, always use the invention strategy) the resulting quantifiers are always created from scratch.

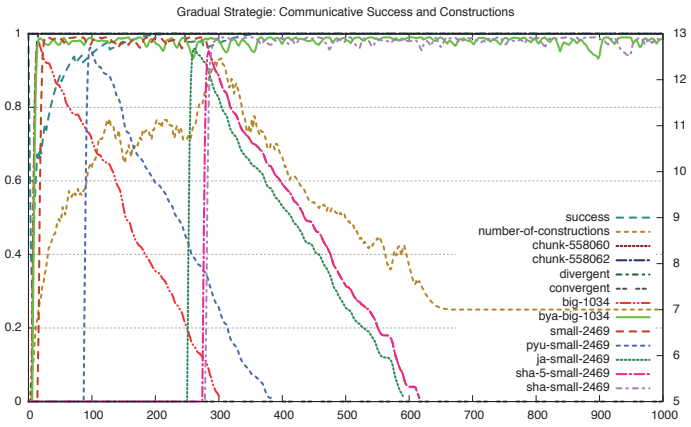
Of more interest is that Figure 7.12(c) shows an overall preference for quantifiers that have adjectival origins. At a likelihood of 50% for using the extension strategy, the chance that the agents converge on quantifiers with adjectival origins is between 70% and 85% (depending on the environmental manipulation). Overall, the likelihood of adjectival quantifiers is much higher than the likelihood of using the extension strategy (i.e., quantifiers that have adjectival origins).

In sum, both innovation strategies, extension and invention, permit a community of agents to develop a language containing gradable quantifiers. And for both strategies the resulting languages are equally apt in expressing number. However, the extension strategy exploits the cognitive similarity between size and quantifying terms. This results in an intermediary language stage in which the community of agents use size terms to express quantity. This intermediary stage makes it easier for novel terms to become shared in a population. So, leaving the choice of strategy up to the agents results in a preference for quantifying terms that are created by the extension strategy. In other words, all other things being equal, the community of agents has a strong preference for quantifiers with adjectival origins.

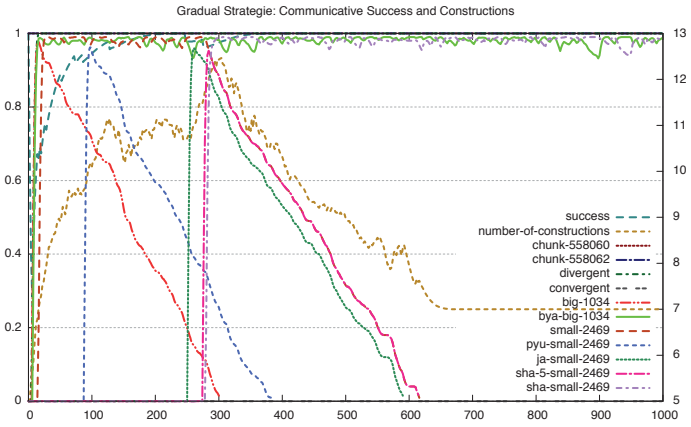
## 7.6 Conclusion

The present results provide a motivation for adjectival origins of gradable quantifiers. It was argued that these origins might be explained by a cognitive overlap between size terms and gradable quantifiers. I provided our agents with a cognitively plausible approximate number system and innovation strategies that can exploit the cognitive overlap for the generation of new lexical items. And it is shown that under these conditions, a community of agents is indeed inclined to derive gradable quantifiers from gradable adjectives.

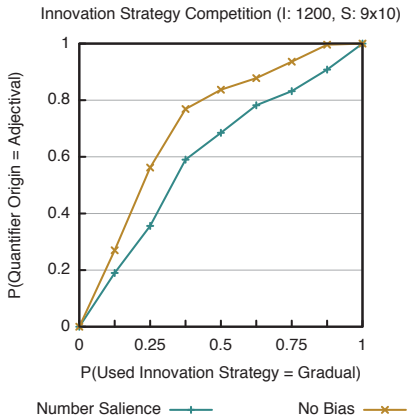
Of course, this is not the be-all and end-all answer to the duality of gradable quantifiers. Most importantly, this chapter only addresses part of the proposed




(a) Single experiment. Converged on quantifiers that were invented from scratch




(b) Single experiment. Converged on quantifiers with adjectival origins.




(c) Strategy Competition

Figure 7.12: Results strategy competition. Figure (b) shows an example of an experiment that results in graded quantifiers from scratch. In Figure (a) the agents converge on graded quantifiers with adjectival origins. Figure (c) shows the likelihood that the converged quantifiers have adjectival origins as a function of the likelihood that the extension strategy is being used for innovation. The result reflect the average convergence of 10 runs for every experimental condition.

puzzle. The chapter starts with pointing out the dual use of gradable quantifiers: Sometimes they act like adjectives and sometimes like quantifiers. This chapter only provides some insight on the adjectival origins. The question remains of why they can also occur as quantifiers. This could have to do with semantic overlap: Just as more canonical quantifiers, gradable quantifiers can only be applied to sets of objects, not to single objects. For now, I leave this as point of future investigation.