A Computational Model of Emergent Simple Syntax: Supporting the Natural Transition from the One-Word Stage to the Two-Word Stage.

Kris Jack, Chris Reed, and Annalu Waller
Division of Applied Computing,
University of Dundee,
Dundee, Scotland, DD1 4HN
[kjack | creed | awaller]@computing.dundee.ac.uk

Abstract

This paper introduces a system that simulates the transition from the one-word stage to the two-word stage in child language production. Two-word descriptions are syntactically generated and compete against one-word descriptions from the outset. Two-word descriptions become dominant as word combinations are repeatedly recognised, forming syntactic categories; resulting in an emergent simple syntax. The system demonstrates a similar maturation as children as evidenced by phenomena such as overextensions and mismatching, and the use of one-word descriptions being replaced by two-word descriptions over time.

1 Introduction

Studies of first language acquisition in children have documented general stages in linguistic development. Neither the trigger nor the mechanism that takes a child from one stage to the next are known. Stages arise gradually with no precise start or end points, overlapping one another (Ingram, 1989).

The aim of this research is to develop a system that autonomously acquires conceptual representations of individual words (the "one-word stage") and also, simultaneously, is capable of developing representations of valid multi-word structures i.e. simple syntax (the "two-word stage"). Two-word descriptions are expected to emerge as a result of the system state and not be artificially triggered.

The system accepts sentences containing a maximum of two words. It is designed to be scalable, allowing larger, more natural sentence sizes also. System input is therefore a mixture of both one-word and two-word sentences. The system is required to produce valid descriptions, particularly in the two-word stage. Rules that enforce syntactic order, and allow for the production of semantically correct descriptions from novel concepts, are desirable.

This paper is sectioned as follows; pre-one-word stage linguistic abilities in children are briefly discussed to explain why initial system functionality assumptions are made; the defining characteristics of both the one-word stage and two-word stage in children are introduced as possible benchmarks for the system; a detailed description of system design and implementation with examples of the learning process and games played by the system are presented; a discussion of current results along with their possible implications follows; a brief review of related works that have influenced this research, citing major influences; the direction and aims of future research is described briefly; and finally, conclusions are drawn.

2 Pre-One-Word Stage Children

Linguistic abilities can be found in children prior to word production. In terms of comprehension, children can distinguish between their mother's voice and a stranger's voice, male and female voices, and sentences spoken in their mother's native language and sentences spoken in a different language. They also show categorical perception to voice, can use formant transition information to mark articulation, and show intonation sensitivity (Pinker, 1994, Jusczyk, 1999).

In terms of production, children produce noises, such as discomfort noises (0-2 months), comfort noises (2-4 months), and "play" vocally with pitch and loudness variations (4-7 months) (Pinker, 1994). The babbling stage (6-8 months) is characterised with the production of recognisable syllables. The syllables are often repeated, such as [mamama] and [papapa], with the easiest to produce sounds often being associated with members of the family (Jakobson, 1971).

From this evidence it is reasonable to draw conclusions about linguistic abilities in the young child that can be used to frame assumptions for use in the system. It is assumed that the system can receive and produce strings that can be broken down into their component words. These words can be compared and equalities can be detected.

3 One-Word Stage and Two-Word Stages

The system is required to produce one-word descriptions in early stages that develop into two-word descriptions, where appropriate, in latter stages.. The recognition of each stage is based on the number of words that the system uses at a particular point. In children, the one and two-word stages have notable features.

The one-word, or holophrastic, stage (9-18 months), is characterised by one-word vocalisations that are consistently associated with concepts. These concepts can be either concrete or abstract, such as "mama", referring to the concrete concept of the child's mother, and "more", an abstract concept which can be applied in a variety of situations (Piaget, 1960).

Two phenomena that occur during this stage are underextensions and overextensions. An underextension is the formation of a word to concept association that is too narrow, such as "dog" referring to only the family dog. Overextension, similarly, is an association that is too broad, such as "dog" referring to all four legged animals. Mismatches, or idiosyncratic referencing also occur, resulting in a word being associated with an unrelated concept, such as "dog" referring to a table (Pinker, 1994). These associations change over time.

The two-word stage (18-24 months) introduces simple syntax into the child's language faculty. Children appear to determine the most important words in a sentence and, almost all of the time, use them in the same order as an adult would (Gleitman and Newport, 1995). Brown (1973) defines a typology to express semantic relations in the two-word stage. It contains ten sets of relations, but only one will be considered in this paper; attribute + entity ("red circle"). During this stage, children already demonstrate a three word comprehension level (Tomasello and Kruger, 1992). The concepts relating to their sentences may therefore be more detailed than the phrases themselves.

The system is expected to make the transition from the one-word stage to the two-word stage without changes to the functionality of the system. Once the system begins to run, input is restricted to that of sensory (concept based) and vocal (string representation) data.

4 System Design and Implementation

4.1 Introduction

The system is designed to learn phrase-to-concept associations and demonstrate it through playing games: a guessing game and a naming game. Games are often used to test, and encourage system learning (Steels and Kaplan, 2001). The learning process involves a user selecting an object in a scene and naming it. The guessing game involves a user saying a phrase, and the system pointing to the object that the phrase refers to. The naming game involves a user pointing to an object and the system naming it The system is not physically grounded, so all games are simulated.

The learning process allows the system to acquire associations between phrases and concepts while the games test system comprehension and system production respectively. The learning process takes a **string** and **concept** as input, and produces no output. Comprehension takes a string as input, and produces a concept as output, whereas production takes a concept as input, and produces a string as output.

4.2 Strings and Concepts

A string is a list of characters with a fixed order. A blank space is used to separate **words** within the string, of which there can be either one or two. The system can break strings down into their component words.

A concept is a list of **feature values**. The system recognises six feature values; red, blue, green, white, circle, and square. There are no in-built associations between any of the feature values. This form of learning is supported by the imageability theory (Paiviom 1971). No claims concerning concept acquisition and formation are made in this paper. All concepts are hard coded from the outset.

The full list of objects used in the games are derived from shape and colour combinations; *red square*, *red circle*, *blue square*, *blue circle*, *green square*, *green circle*, *white square*, and *white circle*. Individual feature values can also act as concepts, therefore the full list is concepts is the list of object plus the list of feature values.

4.3 Groups

To associate a string with a concept, the system stores a list of **groups**. Each group contains an **ID**, one or more **description pairs**, an **observed frequency**, and zero or more **occurrence supporter links**.

The ID acts as a unique identifier, allowing the group to be found. A description pair is a string and a concept. Groups must have at least one description pair since their primary function is to relate a string to a concept. The observed frequency represents the number of times that the description pair's components have been associated through system input.

The occurrence supporter links are a set of group IDs. Each ID in the set refers to a group that

contains a superset of either the description pair, or the same value for one component of the description pair and a superset of the other e.g. The description pair ["red"; *red*] [1] would be supported by the description pair ["red square"; *red square*]. A worked example is provided in the next section. The links therefore record the number of occurrences of the group's description pair. The occurrence supporter link reinforces the description pair's association and increases the **total frequency** of the group. The total frequency is the group's observed frequency plus the observed frequency of all of its supporters, never including a supporter more than once.

Finally, group equality is defined by groups sharing the same description pair.

4.4 The Learning Process

At each stage in the learning process, a description pair is entered into the system. The system does not attempt to parse the correctness of the description. All data is considered to be positive. The general learning process algorithm is detailed in the rest of this section. Specific examples are also provided in Table 1, showing the groups' values; ID, description pair, occurrence frequency (OF), occurrence supporter links (OSLs), and total frequency (TF). Five steps are followed to incorporate the new data:

1. Identify the description pair.
2. Find equal and unequal parts.
3. Update system based on equal parts..
4. Update system based on unequal parts.
5. Re-enter new groups into the system.

4.4.1 Identify the description pair

If the description pair exists in a group that is already in the system, then that group's observed frequency is incremented. Otherwise, the system creates a new group containing the new description. It is given a unique ID and an observed frequency of one. Assume that the system already contains a group based on the description pair ["red circle"; *red circle*]. This has an ID of one. Assume also that the new description pair entered is ["red square"; *red square*]. Its group has an ID of two (group #2).

All description pairs entered into the system are called **concrete description pairs**, this is, the system has encountered them directly as input. The new group is referred to as a **concrete group**, since it contains a concrete description pair.

ID	Description Pair	OF	OSLs	TF
#1	["red circle"; *red circle*]	1	[]	1
#2	["red square"; *red square*]	1	[]	1
#3	["red"; *red*]	0	[#1,# 2]	2
#4	["#3 circle"; *#3 circle*]	0	[#1]	1
#5	["#3 square"; *#3 square*]	0	[#2]	1
#6	["circle"; *circle*], ["square"; *square*]	0	[]	0
#7	["#3 #6"; *#3 #6*]	0	[#2]	1

Table 1: Sample data

4.4.2 Find equal and unequal parts

The new group is compared to all of the groups in the system. Comparisons are based on the groups' description pairs alone. Strings are compared separately from concepts. A string match is found if one of the strings is a subset, or exact match, of the other. Subsets of strings must contain complete words. Words are regarded as atomic units. Concepts are compared in the same fashion as strings, where feature values are the atomic units. Successful comparisons create a set of equal parts and unequal parts. Comparison results are only used when equal parts exist. This approach is similar to alignment based learning, but with the additional component of concepts (van Zaanen, 2000).

In comparing the new group, group #2, to the existing group, group #1, the equal part ["red"; *red*] and the unequal part ["circle"; *circle*], ["square"; *square*] are found. The comparison algorithm is essential to the operation of the system. It is used in the learning process and in the games. Without it, no string or concept relations could be drawn[2].

4.4.3 Update system based on equal parts

When an equal part is found, a new group is created. In the example, an equal part is found between group #1 and group #2. Group #3 is created as a result. The new group is given an observed frequency of zero. The IDs of the groups that were compared (group #1 and group #2) are added to the new group's (group #3) occurrence supporter links. If the group already exists, then as well as the existing group's observed frequency being incremented, the IDs of the groups that were compared are added to the occurrence supporter links. IDs can only appear once in the set of occurrence supporters links, so if an ID is already in it, then it is not added.

[1] The convention of strings appearing in quotes ("red"), and concepts appearing in italics (*red*) is adopted throughout this paper.

[2] The system assumes full compositionality. Idioms and metaphors are not considered at this stage.

Up until this point, all groups' description pairs have contained a string and concept. Description pairs can also contain links to other groups' strings and groups' concepts. These description pairs are referred to as **abstract description pairs**. If all elements of the abstract description pair are links to other groups then it is **fully abstract**, else it is **partially abstract**. A group that contains an abstract description pair is called an **abstract group**. The group is fully abstract if its abstract description pair is fully abstract, else it is a **partially abstract group**. Once a group has been created (as group #3 was), based on a description comparison, the system attempts to make two abstract groups.

The new abstract groups (group #4 and group #5) are based on substitutions of the new group's ID (group #3) into each of the groups that were originally compared. Group #4 is therefore created by substituting group #3 into group #1. Similarly, group #5 is created by substituting group #3 into group #2.

The new abstract groups are given an observed frequency of zero (ID's equal four and five). Note that abstract groups always have an observed frequency of zero as they can never been directly observed. The ID of the appropriate group used in comparison and later creation is added to the occurrence supporters links. Each abstract group therefore has a total frequency equal to that of the group of which it is an abstract form.

4.4.4 Update system based on unequal parts

Unequal parts are only considered if equal parts are found in the comparison. Otherwise, the unequal parts would be the complete set of data from both groups, which does not provide useful information for comparisson. For every set of unequal parts that is found, a new group is created. If there is more than one unequal part then the group will contain more than one description pair. Such a group is referred to as a **multi-group**. Two unequal parts were found earlier in comparing group #1 and group #2. They are ["circle"; *circle*] and ["square"; *square*]. Group #6 is therefore created using these two description pairs.

The creation of a multi-group allows for a fully abstract group to be created. The system uses the data from the new multi-group (group #6) and the group created through equal parts (group #3). Both groups are substituted back into the group that was originally being compared (group #1). The resulting group (group #7) is fully abstract as both equal parts and unequal parts have been used to reconstruct the original group (group #1).

4.4.5 Re-enter new groups into the system

All groups that have been created through steps 3 and 4 are compared to all other groups in the system. Results of comparisons are dealt with by repeating steps 3-5 with the new results. By use a recursive step like this, all groups are compared to one another in the system. All group equalities are therefore created when the round is complete. The amount of information available from every new group entered into the system is therefore maximised.

4.5 The Significance of Groups Types

Four different types of group have been identified in the previous section. Although all groups share the same properties, they can be seen to represent difference aspects of language. It is the combination and interaction of these groups that gives rise to emergent simple syntax. This syntax is bi-gram collocations, but since the system is scalable, it is referred to as simple syntax.

4.5.1 Concrete Groups

Concrete groups acquire the meaning of individual lexemes (associate concepts with strings). They are verifiable in the real world through the use of scene based games.

4.5.2 Multi-Groups

Multi-groups form syntactic categories based on similarities between description pair usage. Under the current system, groups can only have a maximum of two description pairs. If this were to be expanded, it is clear that large syntactic categories such as noun and verb equivalents would arise.

4.5.3 Partially and Fully Abstract Groups

Partially and fully abstract groups act as phrasal rules in the system. Abstract values contained within the group's description pairs can relate to both concrete groups and multi-groups. Abstract groups that relate to multi-groups offer a choice of substitutions.

For example, group #7 (Table 1) relates a single group to a multi-group. By substitution of groups #3 and #6 into group #7, the concrete pairings of ["red circle"; *red circle*] and ["red square"; *red square*] are produced. The string data are directly equivalent to:

S -> Adj. N,
where Adj. = {"red"}
and N = {"circle", "square"}

When a description pair is entered into the system, the process of semantic bootstrapping takes place. Lexical items (strings) are associated with their meanings (concepts). When group

comparisons are made, syntactic bootstrapping begins. Associations are made between all combinations of lexical items throughout the system, and all combinations of meanings throughout the system.

The system stores lexical item-meaning associations, lexical item-lexical item associations and meaning-meaning associations. This basic framework allows for the production of complex phrasal rules.

4.6 Comprehension and Production Through Games

The guessing game tests comprehension while the naming game test production. Comprehension takes a string as input, and produces a concept as output, whereas production takes a concept as input, and produces a string as output. The comprehension and the production algorithms are the same, except the first is string based, and the second is concept based.

The algorithm performs two tasks: finding concrete groups with exact matches to the input, and finding abstract groups with possible matches to the input. Holophrastic matching uses only concrete groups. Syntactic matching performs holophrastic matching, followed by further matches using abstract groups. Note that the system only performs syntactic matching, which includes holophrastic matching. Holophrastic matching is never performed alone, unless in testing stages.

For holophrastic matches, the system searches through its list of groups. Their description pairs are compared to the input being searched for. There is therefore re-use of the comparison algorithm introduced in the learning process. When a match is found, the group is added to a list of possible results.

If holophrastic matching is being performed alone, then this list of possible results is sorted by total frequency. The group with the highest total frequency is output by the system.

Syntactic matching begins by performing holophrastic matching, but does not output a result until all abstract groups have been matched too. It is therefore an extension of holophrastic matching. Once a first fun of holophrastic matching is performed, the input is converted into abstract form. This is performed at the word/feature value level. The most likely element is found by searching through the groups, comparing it to the description pair, and selecting the group with the highest total frequency from those found.

The group IDs replace the appropriate element in the input (just as substitutions were made during the learning process). All multi-groups that contain any of the abstract forms are found. Each multi-group's description pair becomes a replacement for the appropriate input's abstract value.

The new input, which is still in abstract form, is searched for, using holophrastic matching again. Since the groups found are not exact matches of the original input, their total frequency is multiplied by an **abstract factor**. The abstract factor is a value between zero and one inclusive. The higher the factor, the greater the effect that abstract groups have on the results. Syntactic matches can therefore produce different results based on the value of abstract factor. The abstract factor is not changed from the initiation to termination of the system.

Groups found during the search are added to a new list of possible results. The appropriate elements are substituted into the groups abstract values to make them concrete. If an abstract value is acting as a substitute (by being found originally in a multi-group) then the original input value is used, not the replacement element. This allows the abstract group to act as a syntactic rule, but it is penalised by the abstract factor so it does not have as much influence as concrete groups, that have been found to occur through direct input associations.

The groups found throughout the entire syntactic search are now contained in a second list of possible results. This list is reduced by removing duplicate groups. For each group that is removed, its observed frequency and occurrence supporter links are added to the duplicate that is kept in the list.

The two lists from each matching routine are merged and sorted by total frequency. The string\concept of the group with the highest total frequency is outputted by the system.

5 Testing and Results

The system is tested within the following areas:

1. Comprehension and production of all fourteen concepts. The rate at which full comprehension and full production are achieved is compared.
2. Correctness of production matches for compound concepts. The correctness of production matches are studied over a number of rounds.
3. Type of production matches for compound concepts. The type of production matches favoured, holophrastic or syntactic, are compared over a number of rounds

A match of concept to word or word to concept is considered correct if the string describes the concept fully. For example, ["red"; *red*] and ["red

square"; *red square*] are correct, but ["red"; *red square*] and ["red square"; *red*] are incorrect. One point is given for each correct match, zero for each incorrect match.

Note that all test results are based on the average of ten different system trials. Each result shows a broad tendency that will likely be smoothed if more trials are run. All input is randomly generated. The abstract factor is set to 0.4 for all tests.

5.1 Comprehension Vs. Production

Full comprehension occurs much sooner (see Figure 1), on average, than full production. This result is found in children also. Although production and comprehension compete quite steadily in early stages of the system, comprehension reaches its maximum, on average, in 20% of the time that production takes to reach its maximum.

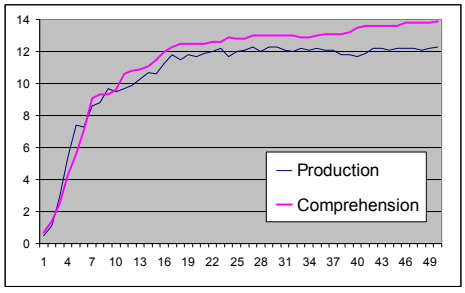

Figure 1: Shows number of correct comprehension and production matches

Full comprehension (fourteen points) is achieved, on average, by round 50, while full production comes at round 250. Both holophrastic data and syntactic data contribute to the successes. Underextensions are found during comprehension. For example, in early rounds, "green" is used to describe only *green squares*. This phenomena is quickly eliminated in the trials but with a larger set of concepts and vocabulary, it is likely to persist for more than a few rounds.

5.2 Correctness of Holophrastic Vs Syntactic Matches

At the end of each round, production is tested using the eight compound concepts alone. These are based on the eight observable objects in the simulated scene. Only compound concepts can demonstrate simple syntax in this system, as singular concepts have associations to single word strings.

The system uses syntactic matching alone, but syntactic matching includes holophrastic matching, as discussed earlier. To determine whether holophrastic data is being used, or syntactic data when a syntactic match is run, the matching algorithm has been split. The number of correct strings produced using holophrastic data and the number of correct strings produced using syntactic data alone are compared (see Figure 2).

The data demonstrate that the system uses mostly holophrastic matches in early rounds (comparable to the one-word stage). This is eliminated in further rounds, in favour or syntactic matches alone (the two-word stage). Note that although the holophrastic stage may appear to be producing two-words, these words are considered to be one-word. For example, "allgone" is considered to be one-word in early stages of linguistic development, as opposed to "all gone" (Ingram, 1989).

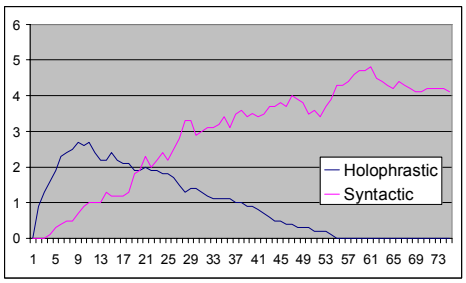

Figure 2: Shows number of correct holophrastic and syntactic matches.

The syntactic data continues to rise, until it achieves full production. The holophrastic stage never achieves full production, but peaks, then reduces to zero. This trend occurs as holophrastic underextensions such as "red" representing *red square* become more likely than "red square" representing *red square*.

Early syntactic matches are based on novel string productions for novel string concepts. Holophrastic matching is incapable of producing novel strings from novel concepts, as it deals with concrete concepts. Abstract concepts however, allow new string combinations to be produced, such as "blue square", from *blue square* even though neither then string nor concept have been encountered before. Such an abstraction may come from a multi-group that associates "blue" with "red", while containing a group that contains "red square" also. The novel string "blue square" is therefore abstracted.

5.3 Use of Holophrastic Vs Syntactic Matches

The system does not always produce the correct strings when a concept is entered. The strings that are produced are a result of either holophrastic or syntactic matching. Regardless of correctness, the amount of times that holophrastic matches are made over syntactic matches can be compared (see Figure 3).

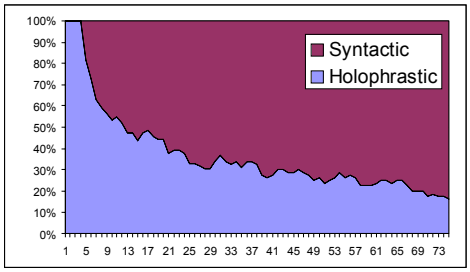

Figure 3: Shows distribution of holophrastic and syntactic matches.

The system relies completely on one-word descriptions at the outset, but soon syntactically derived two-word descriptions become prevalent. It is likely that the one-word stage will last longer if larger concept and vocabulary sets are in use.

The system shows the same form of transition as can be seen in children from the one-word stage to the two-word stage, without the use of an artificial trigger. The shift is gradual although the use of larger concept and vocabulary sets, plus different abstract factor values will affect the transition. The greater the number of words in multi-groups (the greater the size of syntactic categories), the lower the abstract factor is required to encourage the emergence of simple syntax.

6 Related Works

Supporters of computational modelling in language acquisition, often promote the practical importance of running simulations, where evolutionary effects can be recreated in short time periods (Zuidema, 2001).

Although this paper is focussed on an individual system, or agent, acquiring language, it is been influenced by research into social learning (Oliphant and Batali, 1997; Kirby, 1999; Steels and Kaplan, 2002). Social learning demonstrates the convergence upon a common language, or set of languages, from an uncoordinated proto-language, within a population of agents. Social learning allows for the playing of games between agents, similar to those in this paper, with the results being used as further system input, to support, or deny associations. This research can be viewed as a form of social learning with one agent (string and concept generator) performing the teacher role, and the other agent (the system) performing the learner role.

Simulations of both the babbling stage and the one-word stage have been developed (Scheler, 1997; Abidi, 1999). ACCLAIM, a one-word stage simulator, demonstrates that systems can react appropriately to changes in situations. For example, when a cessation event is triggered, it produces “Stop”, and when an object is requested, it produces “More”. Both examples are typical of children during the one-word stage (Bloom, 1973).

Several systems exist that use perceptions to encourage language acquisition (Howell, Becker, and Jankowicz,, 2001; Roy, 2001). ELBA learns both nouns and verbs from video scenes, starting with a blank lexicon. Such systems have helped in the selection of both appropriate input sources and feature values to use in this research. This system will also be physically grounded in future.

The research presented in this paper describes a system that drives linguistic development. Other systems have used similar techniques, based on syntactic and semantic bootstrapping (Howell and Becker, 2001), but have not explained how multiple word acquisition is achieved from a single word basis.

Steels (1998) introduces frames that group lexical elements together by the roles that they play, very similar to groups in this paper. Frames are more dynamic than groups however, structurally adapting when words reoccur. Groups do not adapt in this way. New groups are created to describe similarities rather than adapting existing ones. Steels also introduces multiple word sentences, but it is unclear as to why agents invent a multiple word description over creating a new single word description. The invention is triggered and does not emerge. This research is based on real multiple word inputs, so the reason for invention is not necessary, unlike the reason for adoption i.e. why the system adopts two-word descriptions.

The comparison algorithm, as previously noted, is similar to alignment based learning (van Zaanen, 2000). The system in this research performs perfect alignment requiring exact word matches when finding equal parts and unequal parts. This system also uses concepts, reducing the number of incorrect groupings, or constituents, when there is ambiguity in text. Unsupervised grammar induction can also be found in EMILE (van Zaanen and Adriaans, 2001). EMILE identifies substitution classes by means of clustering. These classes are comparable to this system’s groups although no concepts are used.

7 Future Research

As the system stands, it uses a small input set. Further developments are focussed on expanding the system. All ten of Brown’s relations should be implemented. Larger concept and vocabulary sets are therefore required. Extensions to these sets are likely to affect underextensions, mismatches, the length of pre-syntactic usage time, and the overall growth pattern of simple syntax.

8 Conclusion

This paper offers a potential explanation of the mechanism by which the two-word stage emerges from the one-word stage. It suggests that syntactic data is sought out from the beginning of language acquisition. This syntactic data is always competing with the associations of holophrastic data. Syntax is strengthened when patterns are consistently found between strings and concepts, and is used in favour of holophrastic data when it is sufficiently frequent. The simple syntax continues to grow in strength, ultimately being used in favour of holophrastic data in all production and comprehension tasks.

This system provides the foundation for more complex, hierarchical, syntax to emerge. The type and volume of input is the only constraint upon the system. The entry into post two-word stages is predicted from the system's robust architecture.

9 Acknowledgements

The first author is sponsored by a studentship from the EPSRC.

Thanks to the workshop reviewers for their helpful and much appreciated advice.

References

S. Abidi, 1996. A Neural Network Simulation of Child Language Development at the One-word Stage. *In proceedings of IASTED Int. Conf. on Modelling,* Simulation and Optimization, Gold Coast, Australia.

L. Bloom, 1973. One Word at a Time. The use of single-word utterances before syntax The Hague, Mouton.

R.W. Brown, 1986. Language and categories. In "*A Study of Thinking*", ed. J.S. Bruner, J.J. Goodnow, and G.A. Austin, pages 247-312. New York: John Wiley, 1956. Reprint, New Brunswick: Transaction.

L.R.. Gleitman and Elissa L. Newport, 1995. *The Invention of Language by Children: Environmental and Biological Influences on the Acquisition Language.* In "An Invitation to Cognitive Science", L.R. Gleitman and M. Liberman, 2nd ed., Vol.1, Cambridge, Mass., London, MIT Press.

S.R. Howell and S. Becker, 2001. Modelling language acquisition: Grammar from the Lexicon? In *Proceedings of the Cognitive Science Society*..

S.R. Howell, S. Becker, and D. Jankowicz, 2001. *Modelling Language Acquisition: Lexical Grounding Through Perceptual Features*. In Proceedings of the 2001 Workshop on Developmental Embodied Cognition

J.R. Hurford, M. Studdert-Kennedey, and C. Knight, 1998. The Emergence of Syntax. In "*Approaches to the evolution of language: social and cognitive bases*", Cambridge, Cambridge University Press.

D. Ingram, 1989. *First Language Acquisition. Method, Description and Explanation.* Cambridge: Cambridge University Press.

R. Jakobson, 1971. *Why "mama" and "papa"?* In "Child Language: A Book of Readings", by A. Bar-Adon and W. F. Leopold, ed., pages 213-217. Englewood Cliffs, NJ:Prentice-Hall.

P.W. Jusczyk, 1999 How infants begin to extract words from speech. *Trends in Cognitive Science*, 3 (9, September):323-328.

S. Kirby, 1999. *Syntax out of learning: The cultural evolution of structured communication in a population of induction algorithms.* In Proceedings of ECAL99 European Conference on Artificial Life, D. Floreano et al. ed. pages 694-703, Berlin: Springer-Verlag,

M. Oliphant and J. Batali 1997. Learning and the emergence of coordinated communication. *Centre for Research in Language Newsletter*, 11(1).

A. Paivio, 1971, *Imagery and Verbal Processes.* New York: Holt, Rinehart & Winston.

J. Piaget, 1960. *The Language and Thought of the Child.* Routledge and K. Paul, 3rd ed.,. Routledge Paperbacks.

S. Pinker, 1994. *The Language Instinct. The New Science of Language and Mind*. Allen Lane, Penguin Press.

D. Roy, 2001. Grounded spoken language acquisition: Experiments in word learning. *IEEE Transactions on Multimedia.*

G. Scheler, 1997d. The transition from babbling to the one-word stage: A computational model. In *Proceedings of GALA '97.*

L. Steels and F. Kaplan, 2001. AIBO's first words: The social learning of language and meaning. *Evolution of Communication*, vol. 4(1):3-32. John Benjamin's Publishing Company, Amsterdam, Holland.

L. Steels, 1998. The Origins of Syntax in visually grounded robotic agents. *AI* 103, 1-24.

M. Tomasello, and A.C. Kruger, 1992. Joint attention in action: Acquiring verbs in ostensive and non-ostensive contexts. *Journal of Child Language* 19:311-333.

M. van Zaanen, 2000. Learning structure using alignment based learning. In *Proceedings of the Third Annual Doctoral Research Colloquium (CLUK)*, pages 75-82.

M. van Zaanen and P. Adriaans, 2001. Alignment-based learning versus EMILE: A comparison. In *Proceedings of the Belgian-Dutch Conference on AI (BNAIC)*.

W.H. Zuidema, 2001. Emergent syntax: the unremitting value of computational modelling for understanding the origins of complex language. *ECAL01*, 641-644. Springer, Prague, Sept. 10-14, 2001.