Structural Coupling of Cognitive Memories Through Adaptive Language Games

Luc Steels
Sony Computer Science Lab
6 Rue Amyot 75005 - Paris
VUB AI Lab - Brussels
E-mail: steels@arti.vub.ac.be

Abstract

The paper investigates how a group of distributed agents may develop congruent cognitive memories in the form of networks of prototypes. Memories are congruent if they structure reality in a sufficiently similar way for the agents to cooperate and communicate. Each agent develops his own memory independently of the others and based on his own history of experiences. Agent memories are weakly coupled when the agents are in the same environment. They encounter the same sorts of objects and therefore make similar generalisations. Memories can also be more strongly coupled if the agents communicate and use their cognitive memories to structure and conceptualise reality for communication. The paper explores the second type of coupling. Experiments are reported where agents build autonomously internal cognitive memories and develop through self-organisation a shared lexicon.

1 Introduction

Even the simplest cognitive agent needs a memory to store experienced situations. There is a wide consensus in the cognitive sciences that (1) such a memory is *hierarchical*, i.e. that there are structures recording more specific and more general situations, and (2) the nodes in these possibly tangled hierarchies correspond to *prototypes*, as opposed to concepts defined in terms of necessary and sufficient features [?].The representation of a prototype is generally agreed to take the form of a *schema*, which contains the typical features of the prototype, possibly enriched with other information such as demons, defaults, associative links to other prototypes, etc.

An agent must autonomously develop his own memory based on a stream of situations presented by the environment. This raises the issue how a group of autonomous agents - which cannot inspect each other's brainstates directly and have each only limited experience with the environment - may arrive at a network of prototypes which is sufficiently shared that communication and cooperation between the agents is possible. This paper hypothesises that sharing may be achieved by *structural coupling*. Structural coupling is a concept introduced by Maturana and Varela for explaining how independent systems may nevertheless develop a mutual congruence because the activities of each impacts indirectly the other through the environment. They consequently become coordinated - without a coordinator.

Structural coupling is present, but in a relatively weak form, when different agents are presented with similar situations emanating from the same environment. A stronger form of coupling occurs when the structure of cognitive memory plays a role in communication. Rather than inventing new names for every possible situation that may occur, the agent conceptualises reality in terms of the prototypes in his memory, and verbalises this conceptualisation. The other agent is necessarily forced to adopt or hypothesise similar conceptualisations and hence structure his memory in a similar way. This way a co-evolution of language and cognitive memory is seen: The language lexicalises prototypes developed by the agents, and the agents adopt prototypes lexicalised by the language.

There is clear evidence for the presence of this phenomenon in human natural languages. A language lexicalises prototypes at different level of a prototype hierarchy. For example, English has words for animal, mammal, horse, and mare. Each of these words indicates progressively more specific prototypes for a female horse. There are differences between languages for which prototypes are lexicalised. For example, English does not have a single word for female elephant - some other language might. Also the inheritance relation between prototypes is not always simple.

This paper explores strong structural coupling of cognitive memories through an emergent lexicon. The next part of the paper presents a mechanism by which a single agent may build up his cognitive memory in the form of

a tangled hierarchy of prototypes. Then a mechanism is presented whereby a group of agents may build up and acquire a lexicon for describing situations in terms of words associated with prototypes. The next part of the paper then studies results of simulation experiments in which a group of agents progressively build up their cognitive memories and their lexicons. It is seen that congruence indeed arises. The results reported here are taken from software simulations but they will be ported to physical robots - as we have done with similar experiments in lexicon formation coupled to discrimination tasks [0].

The research discussed in this paper fits within a larger research effort to understand the origins of language and meaning [?]. Several researchers have adopted the viewpoint that language is a complex adaptive system and consequently that mechanisms from 'artificial life' and 'simulation of adaptive behavior' can be fruitfully applied to study the emergence and maintenance of language systems (see the overview in [0]).

2 An Adaptive Cognitive Memory

The main role of memory is to compactly store experiences in such a way that they can be retrieved later. Let us assume that each agent has a repertoire of attributes a_1, ... , a_n, each with a set of possible values, which can be combined to construct features of the form $[a\ v]$. Descriptions have a unique identifier d. The attributes could directly relate to sensory channels or data derived from low level sensory routines. The values could be subregions within the range of possible values for a channel. As shown in earlier papers, adaptive mechanisms exist that develop a repertoire of features from scratch using a selectionist mechanism driven by a discrimination task [?].

A particular situation or a prototype is described using a *schema*. Each schema contains a *description-set* which contains a set of descriptions. A schema is called a *situation-schema* when it describes a particular situation, it is a *prototype-schema* when it describes a class of situations. There is no structural difference between the two types of schemata.

A description contains a *content* in the form of a feature $[a\ v]$. A description also contains *justifications*. Each *justification* decomposes into an indication of the *strength* of a description and a *reason* for its adoption. The strength specifies how crucial the description is in the schema, in other words how certain it is that the description must be part of the schema. The strength can also be negative, implying that the description is known not to be part of the schema.

In the case of situation schemata, the sensory channels and sensory processing routines provide an indication of certainty and hence strength. In case of prototype schemata, the strength is progressively learned from the use and success of a schema in use. When a description has more than one justification, the one with the highest strength is assumed to be the main justification and determines the overall strength of the description.

Schemas can be related through a *use* relation. The relation s_1 *uses* s_2 holds between two schemas iff all the descriptions associated with s_2 are also assumed to be associated with s_1. They consequently do not have to be stored with s_1. Thus there could be a prototype-schema for 'horse' and a schema for 'mammal' and a use relation between them so that all descriptions associated with the mammal prototype are also associated with the horse prototype even though they are not stored with the horse prototype. A schema can use more than one other schema. For example, there can be a prototype-schema for 'female horse' which uses both the prototype for 'female' and for 'horse'. The use-relation implements specialisation/generalisation hierarchies, and corresponds to inheritance relations extensively used in AI and object-oriented programming. The justification mechanism relies on well-established techniques from truth-maintenance systems.

We now face two technical problems: (1) How can an agent retrieve the minimal set of prototype schemas that covers a situation, and (2) How can an agent organise his memory in such a way as to represent in the most compact way the situations he has encountered so far, in other words how can he build up his cognitive memory.

2.1 Retrieving prototype schemata

The retrieval is proposed to take place in two steps. In the first step all schemas are activated whose descriptions match, even partially. Then the schemas are ordered based on their score and the best schema-set, i.e. the minimal set with the highest score is chosen.

A description d_s, called the source-description, matches with another description d_t called the target-description if its contents are equal. The match also results in a *score*, which is designed to accomodate flexible matching of schemata, compatible with the notion of a prototype. The score $< l, g >$ decomposes into a *loss* l and a *gain* g. Let u_s be the strength of the source description and u_t that of the target description, and $m = 1$ if the content of d_s matches with the content of d_t and $m = 0$ otherwise. The score is equal to $m \times d_t \times d_s$. When the score is negative, loss is equal to the absolute value of the score, otherwise gain is equal to the score.

The result of matching a source-schema s_s with a target-schema s_t consists of pairs relating each target description with each source description. A global score is calculated by taking the average loss and the average gain of all matches. There are possibly descriptions in

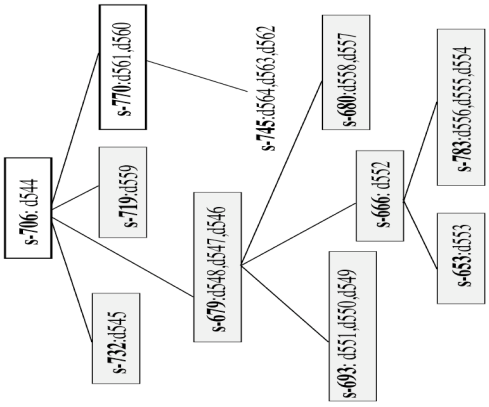

Figure 1: Partial prototype hierarchy in one agent. Each node specifies the name of the schema and the descriptions in it. There are occasionally use-links to other schemas. The incoming schema activates the squared schemas. There are some descriptions which are not yet matching.

the target-schema s_t which do not have any matching equivalent in the source-schema (in which case the loss is 1.0 for that description) and vice-versa.

In a first pass, a new situation-schema (the source) is matched with every prototype-schema (the target) resulting in a *match-matrix* for every prototype-schema, which indicates for every description what the outcome has been. Some of the descriptions in the target-schema are use-links to other schemata. These descriptions can only be matched when an examination of the corresponding prototype schemata have been compared as well. So in a consecutive series of passes, match-matrices are further completed, causing the results of matching to propagate downwards in the schema hierarchy.

Consider for example the prototype hierarchy in figure 1, and a new situation schema with the following description set:

```
S-496: d544, d561, d560, d566, d565
```

There are only two prototypes that completely match: s770 and s776. Many others partially match (because d544 is in every situation). Prototypes that inherit from s706 but have no other matching descriptions will progressively have less and less gains and higher losses. d566 and d565 do not match anywhere. The match-matrix for s770 is as follows. Total score is $< 0.0, 1.0 >$.

Schema	Description	Loss	Gain
s770	[use s-706]	0.0	1.0
s770	d561	0.0	1.0
s770	d560	0.0	1.0

Given match-matrices for every schema in memory, the next step is to find the best matching set. The best match is the minimal set of prototype-schemata that covers most of the descriptions in the situation-schema and had the highest score.

2.2 Reorganising memory

When there is a set of prototype schemata that completely cover the situation schema with a perfect match, then the new situation can be *assimilated* by the memory. No changes are necessary and the new situation is just another case of what has already been seen before.

A more complex case arises when there are no prototype schemata covering the new situation. We then say that the memory needs to *accomodate* the new situation, in the sense that it has to adapt the repertoire of available schemas and their interrelationships. The terms assimilation and accomodation have obviously been taken from Piaget [0]. Piaget was the first to see memory as an adaptive complex dynamical system with assimilation and accomodation as primary operators. Concretely, there are two cases.

1. When no matching target schema at all could be found, we are dealing with a completely new case. Hence the memory is expanded with a new prototype that contains the descriptions of the new situation.

2. The second case is more complex. A prototype (or chain of prototypes - see below) has been found but it is only a partial match. The following *lift operator* is used to repair memory.

Consider a source and target schema as in figure 2. There are some descriptions that match, there are some descriptions in the target that do not match with any in the source, and there are some descriptions in the source that do not match with any in the target. The lift-operator constructs a new prototype which contains all the descriptions common to the source and the target. It then constructs use-links between the source and the target stripped off from the common descriptions now put in the new prototype.

There are two optimisations possible. First of all it can be that there are no descriptions in the source left after all common descriptions have been absorbed into the new prototype. The source schema covering the situation is then equal to the new prototype (see figure 3a). Alternatively, it could be that all the descriptions in the target occured in the source, so that the existing target schema acts as the new prototype and only a use-link must be established between the source schema and the target (figure 3b).

The lift-operator needs to be applied recursively when repairs need to be made to different nodes in the hierarchy (figure 4). If a schema inherits from another schema that has been changed, then this requires another lifting

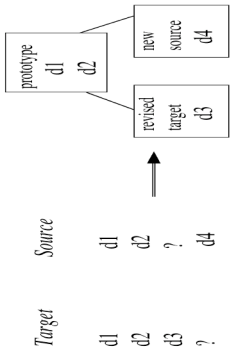

Figure 2: The lift-operator abstracts out the common descriptions from source and target and establishes use-links to the newly formed prototype.

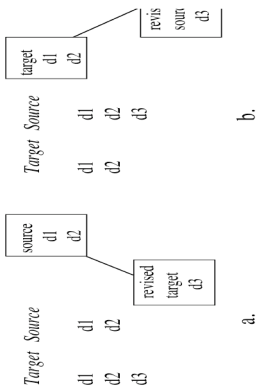

Figure 3: A new prototype need not always be constructed. 3a (left) occurs when all descriptions in the source are in the target schema. 3b occurs when all descriptions in the target were in the source.

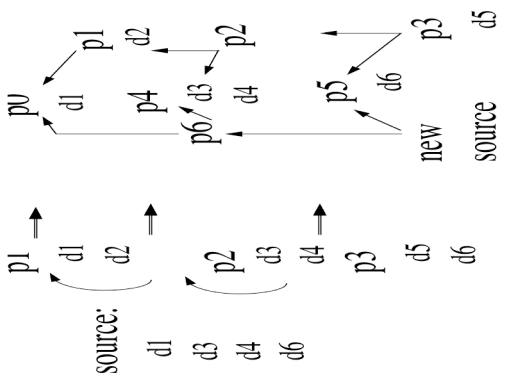

Figure 4: If there is a chain of prototypes of which some are only partially matching, a lift operation to one prototype recursively implies lifting operations to all its users. The figure shows that this operation can quickly lead to important changes. d1 is abstracted out from p1, which requires a revision of p2. The descriptions of p2 are abstracted out into p4. p3 is revised as well.

operation, and equally so for all downward prototypes in the hierarchy.

2.3 Experiments

In testing the mechanisms, an environment with a particular structure is generated by the experimenter. The environment consists of a set of possible situations with a particular structure. For example a hierarchical tree of certain dimensions is generated randomly and then a set of situations is generated conform with this tree. The structure is known to the experimenter but not to any one of the agents. Next situations from the environment are presented to the agent and the experimenter can follow in how far the agent manages to reflect the hidden structure introduced by the experimenter. The evolution of a cognitive memory with respect to a given hidden structure can be followed in a graph such as the one in figure 5. The hidden structure contains 35 nodes organised in a strict hierarchy. The graph shows how the network grows when new situations are encountered. It shows clearly that the repair operations proposed above are remarkably efficient in extracting the hidden structure. Even though the situations are presented in an arbitrary order, the network reflects completely the hidden structure imposed by the experimenter, but unknown to the agent, as soon as all situations have been seen at least once. The network is also maximally efficient because it does not contain any prototypes that are not relevant to cover at least one situation. Note that the cognitive memories of the agents will be different because they have seen different situations and may have

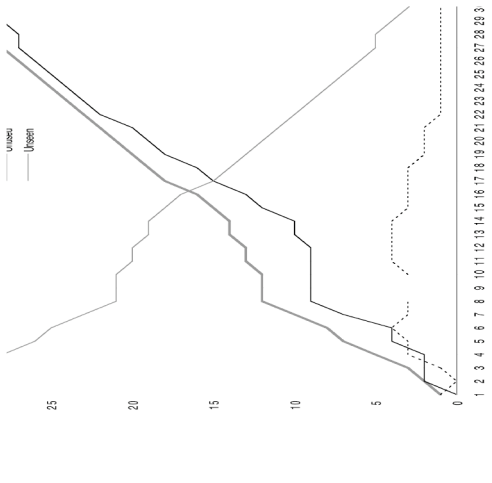

Figure 5: Evolution of cognitive memory for a single agent. *Net-size* plots the size of the network. *Same* measures how much the internal structure of the memory is equal to the hidden structure imposed by the experimenter. *Specialise* measures how many prototypes still need to be generalised or specialised. *Unseen* indicates how many situations are left that have not been seen yet. *Unused* is the number of prototypes that are superfluous. There are none in this example.

consequently constructed different abstract prototypes. This is particularly true for very large hidden structures, where different agents may have seen only parts of the set of possible situations, or when there are multiple possibilities for structuring memory.

3 Lexicon Formation

The lexicon formation process used in the experiments has been extensively described and researched in other papers [?], and is here only summarised. Each agent is assumed to have a lexicon L, which consists of a set of *associations* in the form of triples: <form, meaning, score>. The form is equal to a string of letters. The meaning is equal to a schema with the description-set defining the description associated with the form. The score $0.0 \leq u \leq 1.0$ indicates the strength of the association. There can be more than one association for the same form (homonymy) and more than one form for the same meaning (synonymy).

3.1 Naming Prototypes

Suppose that the speaker wants to identify a prototype to the hearer, further called the speaker topic f_s, and the hearer expects a prototype, f_h. The speaker retrieves all the associations whose meaning matches with the prototype, and orders them based on the score. The form which is part of the strongest association is used for transmission to the hearer. The hearer retrieves all the associations whose form matches with the form transmitted by the speaker. The game succeeds if the prototype expected by the hearer is the meaning of one of the associations. This association is called the winning association.

After each game the agents adapt their memories to be better in future naming games. The adaptation rules are as follows.

1. The game succeeds This means that speaker and hearer agree on the topic. To re-enforce the lexicon, the speaker increments the score u of the association that he preferred, and hence used, with a fixed quantity δ. And decrements the score of the n competing associations with δ. 0.0 and 1.0 remain the lower and upperbound of u. An association is competing if it associates the topic f_s with another word. The hearer increments by δ the score of the association that came out with the best score in the score-matrix, and decrements the n competing associations with δ. An association is competing if it associates the transmitted word with another meaning. These changes implement an excitation-exhibition dynamics similar to the one used in Kohonen networks, except that the change is constant.

2. The game fails There are several cases:

1. The Speaker does not know a word

It could be that the speaker failed to retrieve from the lexicon an association covering the topic. In that case, the game fails but the speaker may create a new word w' and associate this with the topic f_s in his lexicon. This happens with a word creation probability w_c.

2. The hearer does not know the word.

In other words there is no association in the lexicon of the hearer involving the word heard. In that case, the game ends in failure but the hearer may extend his lexicon with a word absorption probability w_a. He associates the word to the topic f_h he was expecting.

3. There is a mismatch between f_h and f_s.

In this case, both speaker and hearer have to adapt their lexicons. The speaker decrements with δ all the associations that have a word-form for f_h, and the hearer decrements with δ all associations that have f_s as meaning.

A group of agents will arrive at a shared lexicon because of the positive feedback loop between score and use. The more an association is successful, the higher its score and thus the more agents use it. This causes success even more to increase and a winner-take-all situation arises where one form starts to dominate for every meaning. We have also been able to show that the lexicon continues to grow if new meanings are added and is resilient to a change in the population. This is illustrated

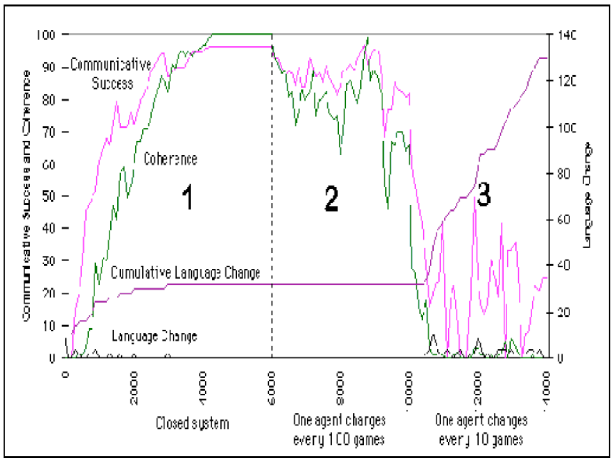

Figure 6: Results of experiments with a group of 20 agents who name 10 different items. A language quickly forms itself, as seen by the rapid increase in average game success and language coherence. A language once formed remains stable even if there is an in- and outflow of agents in the population. This graph shows both language change and the average game success. In a first phase, the language forms itself in a closed population. In a second phase, an in- and outflow of agents (1 in/outflow per 100 games) is introduced, the language stays the same and success is maintained. In the third phase the flux is increased to 1 per 10 games and the language disintegrates. Average game success rapidly reaches very low levels.

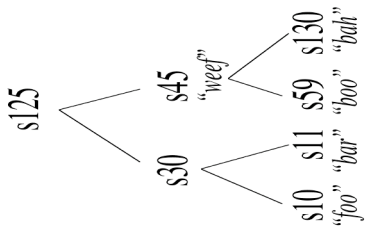

Figure 7: The coverage tree represents the hierarchies of prototypes that describe a given situation-schema. For this particular tree, the cover-set, which is the minimal set of words covering s125, is equal to {foo, bar, weef}

in figure 6, which shows the results of experiments with naming single items.

3.2 Expressing conceptualisations

In the previous paragraph, the meanings were assumed to be single prototypes. We now examine what is needed when conceptualisations based on a prototype hierarchy are used to construct the utterance. The utterance will now consist of several words naming the different prototypes involved. Of course, it is possible to define a unique name for every prototype but it is more desirable to reuse as much as possible existing words. For example, if the two prototypes from which a new prototype p inherits are lexicalised, there is no need to introduce a new word for p. At the same time, it is also desirable to have a minimum number of words in the message itself. Often used prototypes should therefore be lexicalised even if the prototypes from which they inherit have their own names.

The minimal set of words covering a situation-schema is called its *cover-set*. To discuss the principles for constructing this set, it is useful to represent the set of prototypes relevant for a given situation-schema as a tree, further called the *coverage tree*, with the most specific prototypes near the top of the tree. This tree represents the relevant parts of the prototype network, but upside down compared to figure 1.

Let n be the top-node of a coverage tree. If n has been lexicalised, then its associated word is used, otherwise the set of lexicalisations of its sisternodes. If this procedure is followed recursively, we end in principle with a set of lexicalisations for the bottom nodes of the tree, the most abstract nodes in the hierarchy. In practice there are often words for more specific prototypes (see figure 7).

3.3 Constructing a Lexicon

As in the single-word version of the game, the score of the associations used by the speaker and the hearer increases if the game is successful, and the score of competing words decreases. The game can fail because words refer to different prototypes for different agents, or because words are missing to cover all prototypes. In the first case, the score decreases. When the lexicon of the speaker fails to yield a set of words covering a minimal set of prototypes, the following cases arise:

1. If there is no lexicalisation for n and n has no sisternodes, then n itself must be lexicalised. This happens probabilistically based on the word creation probability w_c.

2. If there is no lexicalisation for n and n has sisternodes, then the different sisternodes are either lexicalised or their associations retrieved. If n has in addition descriptions that are not present in any of its prototypes, a lift operation is performed and the new prototype is in turn also lexicalised. This last action implies that the language may itself cause a restructuring of memory.

3.4 Acquiring a Lexicon

When the lexicon of the hearer fails to cover all the words transmitted by the hearer, repair actions need to be performed, possibly both to the memory and to the lexicon. In general, changes are only made when the hearer can make reasonably educated guesses. Otherwise it is better to wait for a situation that has less uncertainty. More specifically the following cases can be seen:

1. If the word is completely unknown, the hearer stores a new association, with a probability w_a, the word absorption probability, between the most specific prototype schema and the word.

2. If there are two words, of which one is known and the other one is not known, an attempt is made to find a conceptualisation that would map onto the two words. This is either because there are two prototype schemata from which is inherited or a lift operation can be performed to assign some of the descriptions of the existing prototype to the new word.

3. In other cases, no further attempt is made to extract a word or a new prototype.

4 Congruence of Memory

We are now in a position to consider the growth of cognitive memories independently constructed by individual agents but coupled through a shared environment *and* through the adaptive language games described in the previous section. The first step is to develop a measure to identify whether memories are indeed congruent. This is not easy because each agent develops different schemas, and an agent may have seen parts that other agents did not see yet.

We say that two prototype schemas are congruent iff *all* their descriptions (including the ones they inherit through use-relations) match. This means that two prototypes are congruent if they cover the same set of situations. The schemas are not necessarily equal because the prototypes from which they inherit may be structured differently. Next, we pair the test network to the the memory of an agent, by constructing a mapping from all nodes in the test network to their congruent schema in the agent, if there is one. The coverage of a network is equal to the percentage of nodes that have a congruent schema. The coverage for a group of agents is equal to the number of nodes in the test network for which at least one agent has a congruent schema. The group congruence of a test network schema s is equal to the percentage of agents that have a congruent prototype schema for s. The congruence for the memory of a group of agents is equal to the average congruence.

Figure 8 shows the results of experiments with a group of 10 agents. The test network contains 43 nodes with a hierarchy of maximum depth 5 and maximum width

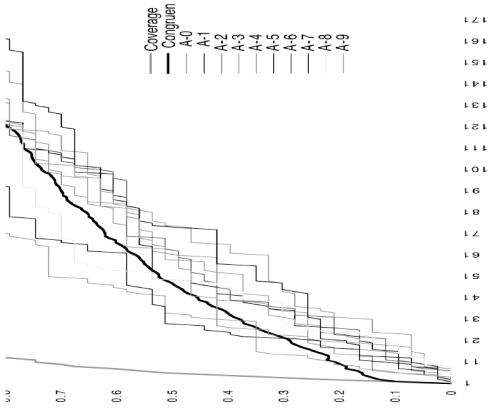

Figure 8: This diagram shows the evolution towards memory congruence, per cycle of 10 games. The thick line going up fastest is the coverage graph. Very quickly, at least one agent has a congruent node for every schema in the test network. The second thick line corresponds to the average memory congruence for all agents. The other lines indicate the congruence between the memory of a single agent and the memory of the total network.

5. The descriptions in each node vary from 0 to 5. The agents start with no cognitive memory and no shared lexicon. They are within the same environment (partly explaining similar generalisations), but also engage in language games in which they conceptualise and then verbalise situations. We see clearly that congruence is progressively arising.

Figure 10 shows the build up of the lexicon of the agents for the same experiment. We see that the lexicon slowly builds up in parallel with the formation of memory. The experiment runs 2000 games longer.

5 Conclusions

The paper has investigated how cognitive memories of distributed autonomous agents may become congruent based on structural coupling. Weak structural coupling happens already when agents are within the same environment and encounter similar situations. A stronger form occurs when coupling takes place through language. The agents use the structure of their memories to conceptualise situations and transmit the lexicalisation of these conceptualisations rather than unique names for every situation. Language also influences how memory is organised in order to economise on the number of words.

These experiments are a step towards understanding how shared language and meaning may evolve in groups of distributed autonomous agents located in open, changing environments. Solid results have been achieved by several researchers in lexicon formation ([?], [?]) but how

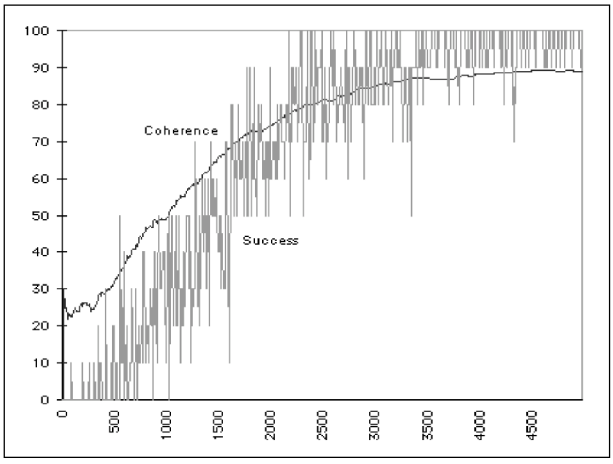

Figure 9: This diagram shows the game success reaching almost total once the memories have stabilised and language coherence building up slowly.

syntax could evolve is less certain (see the discussion in [?], [?], [?]). This paper contributes to the discussion by showing how multiple-word sentences may arise and how different words have different functions in terms of their access to memory. Our next goal is to show how these multiple-word sentences become syntactically structured.

6 Acknowledgement

The research described in this paper was financed and conducted at the Sony Computer Science Laboratory in Paris. The simulations presented have been built on top of the BABEL toolkit developed by Angus McIntyre [4] of Sony CSL. Without this superb toolkit, it would not have been possible to perform the required investigations within the time available.

References

[1] Batali, J. (1998) Computational Simulations of the Emergence of Grammar. To appear in Hurford, J., et.al. (1998).

[2] Kirby, S. (1997) Competing Motivations and emergence: explaining implicational hierarchies. *Language Typology*, 1:5-32.

[3] Lakoff, G. (1987) Women, Fire and Dangerous Things: What Categories Reveal about the Mind. University of Chicago Press, Chicago.

[4] McIntyre, A. (1997) The Babel Toolkit for Studying Language Formation and Evolution. Submitted for Publication.

[5] Hurford, J. (ed.) (1996) Evolution of Human Language. Edinburgh Univ. Press. Edinburgh.

Piaget, J. (1965) L''equilibration des structures cognitives. PUF, Paris.

Steels, L. (1996a) Self-organizing vocabularies. In: Langton, C. (ed.) (1996) Proceedings of Alife V, Nara Japan. MIT Press, Cambridge Ma.

Steels, L. (1996b) Emergent Adaptive Lexicons. In: Maes, P. (ed.) (1996) Proceedings of the Simulation of Adaptive Behavior Conference. The MIT Press, Cambridge Ma.

Steels, L. (1997a) The synthetic modeling of language origins. *Evolution of Communication*, 1(1):1-35. Walter Benjamins, Amsterdam.

Steels, L. and P. Vogt (1997) Grounding adaptive language games in robotic agents. In: Harvey, P. and P. Husbands (eds.) (1997) Proceedings of European Conference on Artificial Life. The MIT Press, Cambridge MA.