BIOLOGICAL CONCEPT FORMATION GRAMMARS
A Flexible, Multiagent Linguistic Tool for Biological Processes

Veronica Dahl[1,2], Pedro Barahona[3], Gemma Bel-Enguix[1] and Ludwig Krippahl[3]
[1]*Research Group on Mathematical Linguistics, Rovira i Virgili University, Avda. Catalunya 35, Tarragona Spain*
[2]*Logic and Functional Programming Group, Simon Fraser Unviersity, 8888 University Dr., Burnaby, Canada*
[3]*Centria - Centro de Inteligência Artificial, Departamento de Informática, Universidade Nova de Lisboa, Portugal*

Keywords: Biology, Cognitive sciences, Concept formation, Multi-agent systems, Molecular biology, Nucleic acid string analysis, Lung cancer detection, Logic programming, Constraint handling rules, Logic grammars, Constraint handling rule grammars, Language processing, RNA design.

Abstract: Constraint based models that are useful for processing biological information have been successfully put forward recently, e.g. for representing multi-disciplinary biological knowledge in view of cancer diagnosis, and for reconstructing RNA sequences from secondary structure. Here we generalize such results into a model for biological concept formation which can interact with heterogeneous agents through constraint-based reasoning. Our model includes linguistic agents, probabilistic agents for mining nucleic acid, and illness diagnosis agents. Information is selected automatically as a side effect of (the system) searching through applicable CHR rules, and automatically transformed when a rule triggers; decisions follow from the normal operation of the rules, and cognitive structure is given by properties that the concepts a given rule is trying to relate must satisfy. Moreover the user can declare under what circumstances a given property or properties can be relaxed. Concepts formed under relaxed properties result in output which signals not only what concepts were formed, but which of the properties associated with the construction of those concepts were satisfied and which were not. This allows us human-like flexibility while maintaining direct executability.

1 INTRODUCTION

Computational linguistics traditionally breaks up the tasks of processing language into separate components which are often applied in sequence: first a lexical analysis annotates the input with word category information, then a syntactic parser produces a parse tree or graph, next a semantic component translates the parse tree into a meaning representation form, and so on.

Sometimes some of these components intermingle, e.g. syntactico-semantic parsers build structure in parallel with meaning representations, in particular to exploit the fact that semantics can inform syntax, and viceversa. In such cases, the semantic information is either embedded into the grammar rules themselves, or forms part of a separate component, for instance a taxonomy, which the grammar rules consult.

Even in cases in which the components are separate, the master-servant modality of traditional computing science reigns, in that for instance grammar rules might invoke a taxonomy but never allow the taxonomy itself to decide when to intervene.

Modern language models and systems, in contrast, stress interactions between modularized components of a grammar which collaborate in a more democratic way. In this respect they approach modern software agents, which must act in a relationship of agency, that is, they must perform an action on behalf of another program module, unsolicited. In fact there have been several recent proposals to explicitly incorporate agents into grammars, in view of specific applications such as generating architectonical designs (Grabska et al., 2009), or designing 3-D structures (Jacob and Mammen, 2009).

An interesting recent development in logic grammars- Constraint Handling Rule Grammars, or CHRG (Christiansen, 2005)- propitiate the introduction of agents, in that they allow multi-headed rules to drive the parsing process in a daemon-like fashion: if all heads in a rule have matching counterparts within the working store (also called constraint store), then the rule applies, making no hierarchical distinction among the different heads of a rule, which contribute

Dahl V., Barahona P., Bel-Enguix G. and Krippahl L. (2010).
BIOLOGICAL CONCEPT FORMATION GRAMMARS - A Flexible, Multiagent Linguistic Tool for Biological Processes.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence*, pages 388-394
DOI: 10.5220/0002786203880394
Copyright © SciTePress

all equally. Thus, a CHRG rule can deal for instance with both syntactic and semantic information in one stroke, by having syntactic and semantic agents coexist as heads in the same rule.

This capability can be taken even further, since CHRG rules can pick up working store elements coming from a variety of disparate sources, and thus they lend themselves ideally for incorporating multiagents that collaborate in tasks that require intelligent interactions with non-grammatical kinds of agents.

In recent years, CHRG models that are useful for processing biological information have been successfully put forward, e.g. for representing multidisciplinary biological knowledge in view of cancer diagnosis (Barranco-Mendoza et al., 2004), and for reconstructing RNA sequences from secondary structure (Bavarian and Dahl, 2005).

In this article we generalize such results into a model for biological concept formation which can interact with heterogeneous agents through constraint-based reasoning, in view of fine-tuning the process to obtain richer and more accurate results. These results include traditional parsing results such as a sentence′s structure and meaning representation, but also less (for a parser) conventional results such as mined nucleic acid information, or medical diagnosis. In this sense, our model can be viewed as an executable, multi-agent based cross between a grammar and a biological expert system. It consists basically of a Constraint Handling Rule grammar that incorporates agents for syntax and semantics, but also for domain dependent biological information, including probabilities, and allows these agents to interact with grammatical information in a natural and expressive, while effective, way.

2 MOTIVATION

Intelligent systems such as those we aim ultimately at with our new paradigm must exhibit distributed intelligence, since they must represent and consult knowledge from different disciplines (e.g. linguistics, biology, parsing). Such knowledge is naturally, and more manageably, expressed in different modules, one or several for each discipline. However these modules must communicate between them in effective ways. This is no easy task, as anyone who has interacted with human experts in view of interdisciplinary collaboration will attest to. Where computers are involved, it becomes even more challenging.

Even within the one discipline of computational linguistics, the variety of subareas involved presents an already formidable challenge. Speech analysis requires different specialized knowledge, for instance, than text processing.

We typically divide in order to reign, and in many cases it is straightforward to do so because the processes involved can be done independently. For instance, we can first glean text from speech through one of the many currently available speech processors out there, and then utilize a text processing program to mine the text appropriately for our needs. In this article we will assume speech has already been translated into text, and focus on text processing.

Even by thus restricting our problem, however, we cannot always divide to reign in the sense of identifying independent processess that can be run in sequence. As mentioned in the Introduction, it may be desirable to intermingle some of the components. Thus, syntactico-semantic parsers build structure in parallel with meaning representations, so that semantics can inform syntax, and viceversa.

As an example of syntax informing semantics, when a sentence contains several natural language quantifiers, it is often easier to determine what their respective scopes -and hence, the sentence′s final meaning- should be once an entire syntactic structure is initially produced and can be looked at as a whole.

As an example of semantics informing syntax, semantic types associated to lexical items may help disambiguate what would otherwise be a structurally ambiguous sentence. For instance "find the price of a computer having Java" has a sensible reading, namely "find a computer that has Java and then calculate its price", as opposed to the nonsensical (to us, but not to machines) interpretation "find a computer′s price such that this price has Java". The nonsensical reading can be disallowed if we only incorporate the semantic type information that computers can have Java whereas prices cannot. Of course, if we allow the nonsensical reading it will fail at knowledge-based consultation time, but disallowing it at the parsing stage is more efficient than allowing the parser to generate a "meaning" representation which we know is doomed to failure.

Semantic type information, however, belongs epistemologically (and also from a practical point of view, since it can be consulted per se) in the knowledge base component of a traditional question-answering system, and therefore needs to be consulted from its grammar component for the purpose of blocking non-sensical readings- unless the type information were to be redundantly included in the grammar as well, as some systems dealing with semantic type checking do.

However in our multi-agent model, a single agent containing all the semantic type information of a particular application is enough, as it can be consulted either per se (e.g. for replying to questions such as: "Is a bank a financial institution?") or as part of an input sentence′s semantic correctness analysis process. In fact, it can be consulted, if desired, from the same grammar rule for both purposes.

More importantly, it can pop up in daemon-like fashion whenever needed, as opposed to depending on e.g. the parser for consulting it explicitly. For instance, the lexical entry for "price" could throw in the workspace not only the grammatical information that "price" is a noun with meaning representation "price(X-T,P)" (the price of an individual X of type T is P), but also the pragmatic constraint: "consistent(T,price)" (i.e., X′s type T must be consistent with X having a price). Likewise, the lexical entry for "computer" would attach the appropriate value to T (namely, "computer"), and once both are in, a biheaded rule would provoke a failure if T were not the type of an object allowed to have a price.

Such issues, which are mostly (computationally) linguistic in nature, first motivated our quest for a cognitive science formalism that could abstract the problems into a general paradigm, concept formation, that could be directly executable and yet flexible. Our incursions into computational molecular biology and other biological applications led, as we next recount, into interesting extensions of this framework.

3 BACKGROUND: FROM LINGUISTIC TO BIOLOGICAL CONCEPT FORMATION

Biological Concept Formation Grammars have evolved from parsing methods we first developed specifically for natural language (Dahl and Blache, 2004), then generalized into an executable cognitive model of knowledge construction inspired as well in constructivist theory (Dahl and Voll, 2004), and finally adapted to very different biological process modeling tasks: early cancer diagnosis (Barranco-Mendoza et al., 2004), RNA reconstruction from secondary structure (Bavarian and Dahl, 2005) and molecular biology text mining (Bel-Enguix et al., 2009). From each of these applications we have distilled both common threads and specific idiosyncracies which have served to fine-tune our initial, general-purpose tool of Concept Formation Rules into the flexible while specialized biological oriented model we present in this article.

Succinctly put, Concept Formation exploits the natural connections (discussed in (Dahl and Voll, 2004)) between constructivism, cognitive logic and logic programming′s recent new paradigm, Constraint Handling Rules (Fruhwirth, 2002), to develop a cognitive model of knowledge construction which can be directly executed through (a specialized system implemented in) CHR. In this model, information is selected automatically as a side effect of (the system) searching through applicable CHR rules, and automatically transformed (or simply augmented) when a rule triggers; hypotheses can be made in the form of assumptions; decisions follow from the normal operation of the rules, and cognitive structure is given by properties that the concepts a given rule is trying to relate must satisfy. Moreover some latitude is provided by which rather than rigidly having to satisfy all properties defined as necessary for a concept to form, the user can declare under what circumstances a given property or properties can be relaxed. Concepts formed under relaxed properties result in output which signals not only what concepts were formed, but which of the properties associated with that concepts construction were satisfied and which were not. This allows us human-like flexibility while maintaining direct executability.

3.1 Concept Formation Rules: The Basic Formalism

Human mind can be seen as a dynamically evolving store of Knowledge which constatly updates itself from new information built from previous information with some kind of reasoning (Dahl and Voll, 2004). Concept formation can be defined as the process of constructing new pieces of knowledge from previously known ones, process that might roughly look as follows: $c_1, c_2, ..., c_i \rightarrow newc$.

From the perpective of CHR, new pieces of knowledge are the Body of the rule, and the previously known ones the Head. Concept Formation Rules have the same general form as CHR rules, except that the guard may include any number of property calls for properties which have been defined by the user:

```
Head ==> Guard | Body
```

Head and Body are conjunctions of atoms and Guard a test constructed from (Prolog) built-in or system-defined predicates, including the reserved binary predicate "prop" (for "property"); the variables in Guard and Body occur also in Head; if the Guard is the constant true then it is omitted together with the vertical bar. Its logical meaning is the formula

$$\forall\,(\text{Guard} \rightarrow (\text{Head} \rightarrow \text{Body})$$

and the meaning of a program is given by conjunction.

Vagueness is expressed by relaxing properties between concepts in accordance with a user's criteria. The criteria can be flexibly and modularly adjusted for experimental purposes while maintaining direct executability.

Thus, rather than inflexibly allowing for a concept to be formed if a test succeeds and disallowing its formation if that test fails, we single out those tests for which we want to allow flexibility as properties. Properties are like any other test, except that their failure does not result in the rule itself necessarily failing: the concept will still be formed, and two lists will be associated with it: a list of the properties that the concept satisfies (S) and a list of those which it violates (V).

This allows us to construct possibly incorrect or incomplete concepts, plus the information on in which way they are not totally warranted. The user then has all the information pertaining to the construction of a particular concept and can therefore interpret these results in a much more informed, holistic way than if the degree of randomness or vagueness had been blindly computed from those assigned a priori to each individual piece of a reasoning puzzle.

Although the lists of satisfied (S) and unsatisfied (V) properties are not explicit, they are managed by the system. The notion of vaguenes lies in the distribution of the lists.

For instance, if we want to accept incorrect sentences in a parsing system that checks for number agreement, we might designate as properties all tests on rule applicability that would correspond to correct parses, and then relax some of them (e.g. the number agreement property). This would be useful for instance in a second language tutoring system which allows the user to make certain types of mistakes, while pointing out the reason why those are mistakes (i.e., which properties are not being satisfied).

The property calls are automatically handled by the system provided that the user defines the properties as follows:

a) a property must be named and defined through the binary predicate prop, whose first argument is the property's name and whose second argument is the list of arguments involved in checking, and in signalling the results of checking, the property. For instance, in a grammar that needs to check for number agreement between a determiner and a noun, say, and to produce either the (agreeing) number of both, or an indication of mismatch, we can choose the name agreement for the property, and define it as follows:

```
prop(agreement,[Ndet,Nn,N]):- Ndet=Nn,
                              !,  N=Nn.
prop(agreement,[Ndet,Nn,mismatch]).
```

b) Acceptability of a property that has thus been defined must be checked in the concerned rule through the binary system predicate "acceptable", whose first argument is the prop atom with all its arguments and whose second argument will evaluate to either true, false, or a degree of acceptability, according to whether (or how much of) the property is satisfied. For our example, we can write:

```
determiner(Ndet), noun(Nn) = =>
acceptable(prop(agreement,[Ndet,Nn,N]),B) |
         noun_phrase(N).
```

c) In order to relax a property named Name (i.e. to allow the derivation of concepts that require it but for which it is not satisfied), we simply write the following:

```
relax(Name).
```

Degrees of acceptability can be defined through a binary version of the relaxing primitive, where L is the prop atom with all its arguments and D is a measure of acceptability:

```
relax(L,D).
```

A list of satisfied and violated properties, together with the degree of violation if appropriate, will be output for each property defined in a given CF program.

3.2 Biological Concept Formation Rules

Although it was not emphasized in the first stages of our work, Concept Formation Rules involve agents, in the sense of cooperating processes that trigger autonomously upon need, i.e. when the constraint store acquires (either through a user′s query or through the normal working of the rules) enough data to trigger a given rule, it will trigger on its own and produce, if all involved properties hold with an acceptable tolerance level, the appropriate new concepts to be added to the working store. The process continues until no more new concepts can be added.

Since Concept Formation is a Cognitive Sciences model, note that although our examples so far have been grammatical in nature, it can also be used to model any other problem domain.

Having the main mechanism for Concept Formation, we need to introduce the agents that can help to design efficienty applications for BioMedicine. In particular, we can identify three main types of agents:

- A *Property Agent*: which manages the user-defined properties.
- A *Concept Formation Agent*: which invokes it in order to enforce or relax those properties according to the user′s specifications as explained above.

- A *Probabilistic Agent*: which is needed for the application of formalism to fuzzy domains, like a multidisciplinary approach to cancer diagnosis, as well as from our work on RNA sequence discovery from secondary structure.
- A *BioMining Agent*: which can identify substrings of interest within given strings of nucleic acid. The definition of such agent derives from our work on mining molecular biology texts.

In the next section we describe the resulting new model of concept formation in its rightfully earned conceptualization as a multiagent system for biological concept formation.

4 OUR MODEL′S LINGUISTIC AGENTS

Our model comprises two types of linguistic agent systems: those that can process human language input and those which can analyse nucleic acid sequences. We next explain the general idea of Human Language processing Agent Systems and describe Nucleic Acid Language Agent Systems, stressing their probabilistic agents and suggesting an application to illnes diagonis.

4.1 Human Language Processing Agent Systems

A linguistic agent that can process human language input allows non-computer specialists to pose questions directly in natural language, thus largely removing the need for them to either learn a specialized computing language, or depend on computer specialists who may not be too conversant with the biological side of things. The subset of language admitted by this agent will vary according to the application, but a core, extensible parser conforms a basic linguistic agent which, because expressed in Constraint Handling Rule format, allows for smooth interaction between the grammar and the biological knowledge base agents (by allowing one of them to inform the other one through integrity constraints in CHR). The main features of this agent, described in (Dahl, 2009), are that it allows for eager discarding of wrong lines of reasoning, and for paraphrases of a given question without ill-effects in the execution, thus accepting fairly rich input.

4.2 Nucleic Acid Language Agent Systems

In (Bel-Enguix et al., 2009), we proposed a CHR based mining technique- the Parallel Matching methodology- for problems that are interesting both in molecular biology and in linguistics, such as identifying subsequences (of English words, for instance, or of nucleotides) that are common to a group several sequences, matching ambiguous subsequences, finding a substring′s frequency, finding gapped patterns. The resulting set of primitives can be viewed as conforming a nucleic acid decoding agent, which has moreover been expanded to include further nucleic acid decoding primitives, and into a grammatical formulation (Dahl, 2009) allowing interaction with human language processing agents, so that the nucleic acid agents can be consulted from human language commands.

Our CHR plus CHRG formulation more readily allows us to use eagerly any constraints of the problem which could serve to early prune the search space. For instance, if the presence of phenylalanine precluded that of leucine and we had detected phenylalanine in our input string, there would be no point in searching for leucine or for the sequence that encodes it. An integrity constraint that provokes failure if both are found in the working store (a single line of code) is all that is needed. In a straight Prolog formulation, in contrast, adding something globally would not be possible: we would have to enter the definition of the substring finding predicate to include a test in some appropriate place within it.

4.2.1 Probabilistic Agents

To mine more complex biological structures, such as RNA secondary structure, in order for instance to reconstruct the RNA sequence that folds into a given secondary structure, we also use CHR but add a probabilistic agent that follows the methodology developed by (Bavarian and Dahl, 2005). This agent operates in the guard of a CHR rule, and uses the probabilities that are believed to govern the proportion of base pairs within RNA sequences. Bavarian and Dahl calculated these probabilities by comparing several RNAs together from Gutell lab′s comparative RNA website (Cannone et al., 2002), a database of known RNA secondary structures. After comparing 100 test cases with various length from 100 to 1500 bases, they found the following probabilities for each base pair:

$$P_{CG} = 0.53, P_{AU} = 0.35, P_{GU} = 0.12$$

The other probabilities which are of interest are the probabilities for an unpaired base to be one of A,

C, G, or U. The results are as follows:

$$P_G = 0.18, P_A = 0.34, P_C = 0.27, P_U = 0.21$$

Inserting the probabilities into the grammar rules is done by generating a random variable in the guard section of the rules, which is the only part that accepts Prolog predicates. This random variable then is tested according to the probabilities: for instance if the random variable in the guard of a rule that assigns nucleotides to positions known to be paired is less than 0.53, it will assign a GC pair. The average error is estimated to be about 18%, meaning that 18% of the nucleotides might be paired with a nucleotide in a wrong position (in the original structure they might be either unpaired or be paired with another nucleotide).

4.2.2 Illness Diagnosis Agents

In previous work of one of the authors with Alma Barranco-Mendoza, specialized concept formation rules were used for representing knowledge in view of diagnosing diseases such as lung cancer (Barranco-Mendoza et al., 2004). Following this work, yet another kind of probabilistic agent materializes as an additional parameter of each constraint in the specialized concept formation rules of our multi-agent system.The application introduced in this paper aims to aid in early stage detection of some types of cancer, like lung and oral, which have poor prognosis because they are very difficult to diagnose at the early stages.

Our concept formation methodology assists in the integration and analysis of multidisciplinary agents containing genetic and molecular information along with the radiological, serum and sputum data. In particular, it provides some kind of diagnosis even if given incomplete patient information, as not all tests can or will be done on a given patient at a given time. This is achieved by relaxing certain properties, whereupon the analysis will be completed even if the information is not complete. The list of violated properties can provide a list of suggested follow-up tests to improve the accuracy of the diagnosis.

As part of the input concepts it accepts the patients age, smoking history, malignancy history, radiological, serum and sputum data. The knowledge store includes the properties that should be evaluated for each input data element as well as the relations amongst them. The diagnosis is given as a probability of cancer that is calculated as a function of the concepts used in the analysis. As well, the diagnosis will list those diagnostic properties that were satisfied and those that were not. For example:

```
const(Prob),age(x,A),history(x,smoker,T),
serum_data(x,marker_type,in_range)<=>
marker(x,marker_type,in_range,P,B),
acceptable(marker(x,marker_type,in_range, P),B),
probability(P,Prob,x, B),
acceptable(probability(P,Prob,x),B)|
  possible_lung_cancer(yes,Prob,x).

  relax(marker(x,marker_type,in_range,P,B)).
```

This rule evaluates for a patient x if a specific biomarker, marker-type, found in serum data is within a certain value range for a patient with an age of A who is a type T smoker (T depends on the number of cigarettes or cigars smoked daily). If true, then the diagnosis of possible lung cancer is going to be true with a probability increase of P (where P is a function of the patient's age, health history, and this particular biomarker presence). But if we relax the requirement of the presence of the biomarker, then the system can evaluate patient records that do not have this particular information and report in the diagnosis listing that this information was not included in the record, which could be valuable information as recommended follow-up tests for that particular patient.

5 CONCLUSIONS

We have abstracted, from recent different realizations of the linguistically inspired Concept Formation paradigm, a multi-agent model for Biological Concept Formation which can be considered as a computational metaphor for the (biological) mind, with direct executability implications. Due to the generalized use of Constraint Handling Rules or their grammatical counterpart, we are able to integrate human language processing techniques into our approach which are not only useful for all types of concept formation but also allow us a smooth integration of human language processing agents, as well as their interactions with the knowledge base agents. Another interesting feature of our proposal is its robustness: due to the capability of relaxing some of the properties involved in concept formation, results that can be useful are provided even in the absence of all the information "necessary" to form the concepts in question.

Concept formation rules are applicable to many other AI and cognitive problems as well, most notably, those involving the need to reason with incomplete or incorrect concepts.

The flexibility allowed by relaxing properties was argued in our initial paper (Dahl and Voll, 2004) to provide a more appealing solution to the need for flexibility than the two main alternatives out there, namely probabilities and fuzzy logic. The probabilistic approach had been discounted as inappropriate for measuring the meaning of information, although adequate to measure the randomness of information. However, after our work on reconstructing RNA sequences from the structure into which they fold, plus our work on using Concept Formation as an aid for early diagnosis of lung cancer, we must rectify that statement. We are now convinced that probabilistic agents, both of the randomness measure kind and in the form of combining the probabilities of individual biomarkers into an overall probability of a disease developing, are very useful agents that have a rightful place in a biological rendition of the Concept Formation paradigm.

Admittedly, the range of disparate biological problems we have attempted to cover under a single paradigm is perhaps too ambitious to allow us as homogeneous a model as we would have liked. However we feel it is an important step towards achieving an encompassing and robust multi-agent model for these various tasks, in that it allows for autonomous triggering of the agents needed at a given time, for easy synchronization between the various agents, mainly through integrity constraints, and for flexibility, through property relaxation, in the face of either incomplete or erroneous data- a problem that biological systems aspiring to deal with real life problems must absolutely face.

ACKNOWLEDGEMENTS

This paper is supported by the European Commission in the form of V. Dahl′s Marie Curie Chair of Excellence, the Canadian National Sciences Research Council, the project “Constraint - and Hypothetical - based reasoning for BioInformatics”, reference HP2008-0029, and “Logic, Automata and Formal Languages”, MTM2007-63422.

REFERENCES

Barranco-Mendoza, A., Persaoud, D., and Dahl, V. (2004). A property-based model for lung cancer diagnosis. In *RECOMB (2004) 27–31*.

Bavarian, M. and Dahl, V. (2005). Rna secondary structure design using constraint handling rules. In *WCB'05, Workshop on Constraints for Bioinformatics*.

Bel-Enguix, G., Jiménez-López, D., and Dahl, V. (2009). Dna and natural languages: Text mining. In *Proc. International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, KDIR 2009*, pages 140–145.

Cannone, J., Subramanian, S., Schnare, M., J.R. Collett, L. D., Du, Y., Feng, B., Lin, B., Madabusi, L., Muller, K., Pande, N., Shang, Z., Yu, N., and Gutell, R. (2002). The comparative rna web (crw) site: An online database of comparative sequence and structure information for ribosomal, intron, and other rnas. In *BioMed Central Bioinfo. 3:2*.

Christiansen, H. (2005). CHR grammars. In *Journal on Theory and Practice of Logic Programming, vol. 5, number 4-5*.

Dahl, V. (2009). Decoding nucleic acid strings through human language. In *Manuscript Submitted for Publication*.

Dahl, V. and Blache, P. (2004). Directly executable constraint based grammars. In *Proc. Journees Francophones de Programmation en Logique avec Contraintes*.

Dahl, V. and Voll, K. (2004). Concept formation rules: An executable cognitive model of knowledge construction. In *NLUCS'04, International Workshop on Natural Language Understanding and Cognitive Sciences*.

Fruhwirth, T. (2002). Theory and practice of constraint handling rules. In *Special Issue on Constraint Logic Programming (P. Stuckey and K. Marriot, Eds.), Journal of Logic Programming*.

Grabska, E., Strug, B., and Slusarczyk, G. (2009). A multi-agent distributed design system. In *7th International Conference on PAAMS'09, AISC 55*. Springer-Verlag Berlin Heidelberg.

Jacob, C. and Mammen, S. V. (2009). Swarm grammars: growing dynamic structures in 3d agent spaces. In *Digital Creativity, Volume 18, Issue 1, March 2007*. Routledge.