Grounding Formal Syntax in an Almost Real World

Joachim De Beule* **Joris Van Looveren*** **Willem H. Zuidema***
*Artificial Intelligence Laboratory
Vrije Universiteit Brussel
Pleinlaan 2, B-1050, Brussel, Belgium
{joachim, joris, jelle}@arti.vub.ac.be

Abstract

Human, syntactic language is one of the most intriguing behaviors and receives increasing attention from researchers in numerous fields. Here we present a model that goes an important step further than previous work because it explicitly connects low-level perception and categorization, hierarchical meaning construction and syntactic language. The model thus shows a solution to the 'symbol grounding problem' (Harnad, 1990): the meaning of the symbolic system – logical symbols and syntactic rules – is grounded in its relation with a simplified but realistic world. We discuss the different components of this collaborative effort: (i) a realistic simulation of Newtonian dynamics of objects in a 2D plane; (ii) schema-based event-perception and categorization; (iii) a semantics based on predicate logic; and (iv) a categorial grammar for the production and interpretation of language. The integration of the different components poses on the one hand novel and important constraints; on the other hand, it allows for experiments that help to identify the relations between the different levels. We note some important similarities and differences with SHRDLU (Winograd, 1976) and the Talking Heads experiment (Steels et al., 2002), and give an agenda for future experiments.

1. Introduction

Human language is one of the most intriguing adaptive behaviors that have emerged in evolution. With our language we can communicate about events that have happened in the past or will happen in the future; we can express complex causal relations, phrase questions or imperatives, and share in detail previous experiences. Language makes it possible to express an unbounded number of different messages, and it serves as the vehicle for transmitting knowledge that is acquired over many generations. Human language, with its grammar, is viewed as the last *major transition* in evolution (Maynard-Smith and Szathmáry, 1995): a transition that opens up a fundamentally new level of information transmission and selection.

In many fields, from anthropology to computational linguistics, human language is a central issue. In recent years also researchers in the field of simulation of adaptive behavior have studied the origins and nature of human language. Such studies have concentrated on the emergence of a lexicon of arbitrary form-meaning mappings, either through evolutionary optimization (e.g. MacLennan and Burghardt, 1994; Noble and Cliff, 1996) or through 'coupled' learning behavior in populations of agents (e.g. Hurford, 1989; Steels, 1996a; Oliphant, 1999). With the same approach, interesting results have also been obtained on the emergence of sound systems (e.g. De Boer and Vogt, 1999) and syntax (e.g. Kirby, 2000; Batali, 2002).

These studies have focused on issues that were largely ignored in more traditional approaches, in particular the question: how did human languages become the way they are? Moreover, they have put an emphasis on studying "complete agents", agents that not only have the ability to *interpret* language (the focus of most work in formal linguistics), but also to *produce* language, *acquire* language and use it to perform a *task* in an *environment*. This shift in emphasis has brought new criteria on what makes useful representations, formalisms and models, and it has brought new challenges.

Harnad (1990) defines one of these challenges as the 'symbol grounding problem': *how is symbol meaning to be grounded in something other than just more meaningless symbols?* Harnad argues that it is cognitive theory's burden to explain how "human beings (or any other devices) [...] can (1) discriminate, (2) manipulate, (3) identify and (4) describe the objects, events and states of affairs in the world they live in, and they can also (5) "produce descriptions" and (6) "respond to descriptions" of those objects, events and states of affairs." (Harnad, 1990). We here present a system that is, for a simplified world, capable of doing all these things.

But our ambitions go even further. If we are to seriously explore the functional and semantic constraints on the (1) use, (2) acquisition and (3) evolution of language, we need a sophisticated model of meaning that

is grounded in interactions with the world. Most existing models of the evolution of syntactic language (e.g. Batali, 2002; Kirby, 2000) presuppose the existence of a set of meanings, that is moreover extremely simple (of the type "john sees mary"). In this article we describe a system that was designed to investigate the acquisition and evolution of a language that is grounded in a rich interaction with the world: by giving the agents the possibility to observe and act upon a world, by letting the agents evolve their own meanings, thus letting the meanings really mean something in this world.

2. Previous work

2.1 SHRDLU (Winograd, 1976)

One of the first fully integrated system system was SHRDLU, consisting of a simulated blocks world, a semantic processing module, a planning algorithm, a grammar, etc. The currently described system bears many similarities with SHRDLU, but there are also important differences, the most important being that SHRDLU shows no grounding.

The blocks world in SHRDLU is a set of predicates, for example stating there is a red cube C and a triangle T. In addition there are relations like ON(T, C), stating that the triangle is currently positioned on the cube. Finally, the 'physics' of the world are a set of transition rules stating that, for example, if SHRDLU would perform the action PICK-UP(T), then the relation ON(T, C) will no longer be satisfied. All this information is preprogrammed, given in advance to SHRDLU. This means SHRDLU would be totally lost if a new kind of object enters the scene, for which it does not know any rules or relations. It would get even worse if this object, say a ball, would suddenly start to roll, a behavior not imaginable by SHRDLU because it was given no transition rules for it, and it has no means of creating new transition rules.

The current system does not presuppose anything about its world, except that it provides the agent with inputs. The agent remains adaptive, keeps an open mind, and would not be totally lost if suddenly some new object would enter the scene. This makes our job a lot more difficult of course, and we do not claim we have solved the problem even in the restricted domain of a blocks world. But we do think that research on integrating and exploiting ideas from both the classical and new AI can provide an important step forward and maybe lead to a SHRDLU that would not be lost when facing an unknown experience, or even when placed in 'the real world.'

2.2 The Talking Heads (Steels, 1997)

Another important predecessor of the system described in this paper is known as the 'Talking Heads' (TH) experiment (Steels, 1997). This experiment is based on the notion of *language games* (Steels, 1996a; Wittgenstein, 1967). The goal of the TH experiment was to study the emergence of a shared lexicon through social interactions in a population of language-capable agents.

At the outset, the agents in the TH experiment have no concepts of the world and no language. Driven by their (preprogrammed) urge to be able to discriminate the different objects in their world they start dividing up their sensor data into regions, thus forming basic but nevertheless *useful* primitive concepts (discrimination games; Steels (1996b)). These concepts (*meanings*) are then communicated in language games. This way, through their social interactions, they collaboratively invent the language they need to communicate, and work towards a shared language. This in turn provides feedback on the value of created concepts.

There is no central entity in the experiment that decides on the language. In the beginning communication is chaotic, because different agents will independently try different words for expressing the same meanings, or even develop different concepts. The driving force towards convergence is every individual agent's desire to be able to communicate with as many other agents as possible, which causes it to adopt those words that (according to its own observations) most other agents use. This behavior generates a feedback loop in which a certain word, as soon as it gains a small edge on other words that express the same meaning, will be adopted by progressively more agents, until the whole population uses it (Steels et al., 2002).

An agent's learning mechanisms remain in force throughout the agent's life cycle, even when the population's language has been stable for a long time. This allows the population to remain resilient for changes in the environment; the agents will be able to create new language structures as necessary to accommodate any change in the environment as soon as it occurs.

The system described below bears many similarities to the Talking Heads system. The Talking Heads showed that the concept of an integrated system for language games, combining both classical and new AI ideas, can lead to successful experiments and important results, both in linguistics, machine learning and philosophy. It has therefore been an important step forward. But, in order to investigate more complex aspects of language (and cognition in general), and in order to validate these conclusions for more complex environments, it has to be extended in several ways:

- The TH's world is static. There is no time, no causation, nothing is really 'happening.' Clearly, cause-

consequence chains are fundamental to language (and intelligence), and a realistic world should reflect this.

- The conceptualization and learning mechanisms in the TH agents are not capable of representing causation, or indeed any other type of complex event. Concepts are, at best, combinations of regions in input space. Even static notions like 'tower' cannot be represented.

- The agents cannot perform actions in the world, and are therefore unable to achieve goals other than getting better in communication and discrimination. This also means it is not possible to let the evolution of concepts and language take advantage, of the agent's ability to perform actions, nor guide or reflect other goals.

- The language production and interpretation capacities of the TH do not allow for compositional meanings, as is clearly the case in natural languages (grammar). The utterance '*blue square left triangle*' is in no way distinguishable from '*left square blue triangle*'. This is because words and meanings are equivalent in the TH, no grammaticalization step is performed between the construction of a semantic description and the construction of an utterance.

3. Simulating Newtonian dynamics

As a first step towards our new system, we built a virtual and simulated world for our agents to 'live' in. On the one hand the world should not be so complex that it would be impossible to implement or run within a reasonable time frame. On the other, it should be complex enough to allow for hidden (state-)variables, time, causation, etc. A simple yet realistic model of part of the real world seems a good candidate. Therefore we chose to implement a blocks world that is two-dimensional and only consists of rigid polygonal bodies, but where the bodies actually behave as prescribed by the laws of Newtonian physics, including rotation, static and dynamic friction, gravity, etc.

There is a vast amount of literature and research on physically correct simulation of rigid bodies, mainly driven in recent years by the emerging possibilities to use these techniques in virtual reality and computer animations, for production process optimization and, of course, for computer games (e.g. Hanh, 1988; Moore and Wilhelms, 1988; Keller et al., 1993).

There are two main approaches to simulating the laws of physics for rigid bodies: either all interactions between contacting bodies, even interactions at so called resting contacts, are modeled by a collision (impulse-based methods, e.g. Mirtich and Canny (1995)), or resting contacts are modeled trough action/reaction forces (constraint-based methods, see e.g. Baraff (1994)). Some attempts have been made to combine the two approaches (e.g. Mirtich, 1995).

What is important for our purposes is that, with both approaches, it is possible to simulate real undeformable world objects very realistically. Although this is not the place to elaborate about technical and mathematical details of rigid body dynamics, we mention that colliding contacts are handled by impulse forces including friction and energy dissipation, parameterized by a coulomb friction coëfficient μ and a coëfficient of restitution ϵ (see Hanh (1988); Moore and Wilhelms (1988) and Chatterjee and Ruina (1998) for a more thorough analysis of the subject), while resting and sliding contacts are handled by action/reaction forces as described in (Baraff, 1994), slightly modified to allow for fast friction force calculation. Using this scheme, the behavior of simulated rigid bodies seems to be very realistic.

The system we implemented allows one to define a world and add body definitions to it without further having to care about how the bodies should act or interact. The main simulation loop is as depicted below, although in reality it is a bit more complex because it involves detecting penetrations between bodies and possibly backing up the simulation (collision detection.)

```
simulate (simulation-definition) {
   initialize the simulation;
   loop {
         advance simulation state up to
         next contact or next display time;
         determine and apply contact forces;
         display the simulation;
   } }
```

The state of a simulation consists of the position X, orientation α, impulse P, angular momentum L, force F and torque τ for every rigid body in the simulation, together with some extra derived state variables to make for example collision detection faster.

Advancing the simulation means solving a system of coupled differential equations describing the change of all state variables with respect to time:

$$\begin{aligned} \dot{X} &= P/m, \\ \dot{\alpha} &= L/I, \\ \dot{P} &= F, \\ \dot{L} &= \tau. \end{aligned}$$

These are the laws of Newton, and they allow us, or an agent, to apply external forces to the bodies.

At every simulation step we can ask the system to provide us with information about its current state. For example we can get information about the position or color of blocks in the simulation. This information (or part of it, possibly after a number of transformations or noise adding) can now be used to supply an agent with input or observations. This will be discussed in the next

section; fig. 1 gives a sequence of simulation frames for a simple simulation of a ball bouncing down some stairs and colliding with some domino bricks.

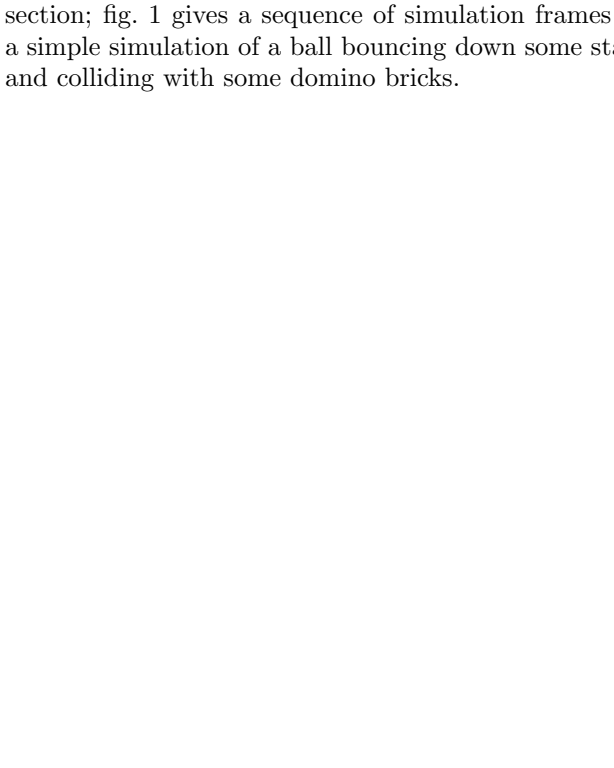

Figure 1: some subsequent views on an example simulation. Time increases from left to right before top down.

One can see that, although the ball does not hit the top stair block at an edge, it does start moving to the left after the first collision. This is because the ball is rotating counterclockwise, so that friction causes the ball to move to the left during the collision.

4. Conceptualization

This section describes the part of the system that processes the input an agent gets from the simulation (the raw data) and tries to transform it and detect *meaningful* or *useful* concepts. In the current and first experiment, these concepts consist of events like 'an approach' or 'a collision', but also objects like 'a red triangle' or 'a tower'. It is not our aim to build an 'event detection' system however, but to build a system that is able to construct and detect concepts, to give meaning to and make use of the raw data it receives trough observation.

In the first subsection we formulate and discuss some criteria the event perception system should obey. The second subsection describes the actual design of our system in more detail. The last subsection gives a short comparison with related systems.

4.1 Some criteria

The simulation described in the previous section provides input for an agent. It defines the world of interaction for an agent; the world the agent should observe, reason about and act upon. The first question at this point is what type of data from the simulation should be given to an agent. For example, we could provide an agent simply with all pixel values of a simulation window. The other extreme would be to give the agent access to the entire state of the simulation (positions, impulses, contacts, etc.) This would not be consistent with our aim and philosophy of *grounding* both the origin and meaning of an agent's concepts, for then, the world would again be part of the agent. For practical reasons, we chose a middle way. We transform (not copy) part (not everything) of the state of the simulation to a new set of observation variable-value pairs. This *raw data* represents the equivalent of data an animal receives trough its receptors. Of course, it *is* not equivalent, it only *represents* perceptional data as one would define it in a real animal. However, the agent does not get information from the simulation that could not be provided by applying segmentation and other standard image processing techniques on camera images (such as used in the TH, Steels et al., 2002).

The next question is: what should the event detection or *concept formation* system do with this raw data? The system should be able to detect meaningful and thus useful concepts with respect to some task the agent is to perform (e.g. a language or discrimination game, prediction, 'moving all red blocks to the left',...)

In addition, we don't want an agent to 'get lost' when a previously un-encountered event happens. It should therefore be able to create new detector channels, new concepts triggered by other primitive or previously created event detectors. For example, assume an agent is at some point able to measure, trough observation, positions of objects in the blocks world and it would be useful for the agent to create a detector for approaching objects. A new 'approach' detector could be built, looking for pairs of objects of which the positions are getting closer. Note that, because newly created detectors in their turn become building blocks for other event detectors, arbitrarily abstract concepts could come to existence. These will however still be grounded as is also explained in Harnad (1990).

Finally, it should be possible to attach actions and predictions to the occurrence of an event.

4.2 Implementation

Our implementation consists of *item* and *template* datastructures, together with a system for processing them. Items can represent virtually anything, they are data structures that have a unique ID and a set of features, every feature having a name, a value and a history (see later). Observation of the simulation can for example result in a set of object items having features for position, color, etc.

Apart from observation, other items representing more abstract or newly created channels should be created. This is the task of templates. Templates are detectors for various things. They can become active for simple object items, but also for configurations of objects (e.g. 'tower') or events (such as approaches or touches). Templates consist of an activation slot, an action slot and a prediction slot. The activation slot is a set of conditions on items the template needs to get activated. For example, the approach template mentioned above, could have an activation slot

```
(and (has-feature-p position ?x)
     (has-feature-p position ?y)
     (decreasing-p (distance ?x ?y))),
```

where `has-feature-p` and `decreasing-p` are predefined predicates and `distance` could be a newly evolved detector. The symbols starting with a question mark represent variables, to be bound to items for activation of the template: the template can be activated when it can find a binding for its activation slot consisting of items that, when filled in, make the activation predicate become true. The action slot of a template could for example be to create a new item.

As mentioned, next to a (current) value, features also have a history. This reflects the change the feature value has made before it got its current value and is the way our event detection system handles time and change. Because it is impossible to record every change, some way of filtering, some way of abstraction has to be done. We therefore adopted some ideas from qualitative physics (see e.g. Weld and de Kleer (1990) for an overview or Kuipers (1994) for a particular but comprehensive model): only the direction of change of a feature's value is recorded. In addition, a new history epoch is started whenever this direction changes itself. For example, suppose a bath tub is filling with water, starting of empty. The height of the water in the tub starts at zero. From there on it increases, but only until it reaches the edges, from where it is steady again. Thus, the history of a water-level feature would after the whole experiment consist of three epochs: (1) zero and *steady* before start time, (2) between zero and full and *increasing* after start time, and (3) full and *steady* after filling time. Of course, a template's activation slot should be able to test on the history of a feature, as is the case with the approach template above.

4.3 Relation with other systems

Neural Networks: Concept formation in the sense of pattern recognition and learning of new patterns is currently probably best done by neural networks (NN). At the lowest (least abstract) level, the criteria we put forward could be seen as a description of a neural network where concepts would be frequently re-occurring activation patterns that proved useful in some way or another. At this point, however, it would be difficult to replace the techniques we have used with a neural network. In a NN it can be hard to identify individual concepts, because they can be distributed all over the network and may never be entirely the same. In addition it is not clear how, for example, a neural network can be constructed that can handle an input of a variable number of objects or can represent something like 'an object' as a well bounded collection of time varying features and properties in the first place (the binding problem, von der Malsburg 1981) without making use of e.g. standard image processing techniques to preprocess network input.

Frame Systems and Semantic Networks also have some of the characteristics mentioned above. They have been extensively studied in the field of knowledge representation and computational linguistics. In Brachman and Levesque (1985), a comprehensive collection of some of the most influential papers on the history of knowledge representation is given. However, it is not well understood how time and change can be incorporated, or how such frame systems or networks could be learned (a question we also in part still have to answer). One particular interesting implementation of empirical learning of a kind of frame system is described in Drescher (1991). Drescher implements a mechanism to learn *schema's* where a schema is a tripartite structure comprising a context, action, and result. Learning is done by altering and chaining together schema's to improve the agents performance in predicting its living environment. Next to prediction, we think language can also be an important constraining principle guiding a learning agent to a shared and useful set of concepts.

Event Detection is another related field. There are two main approaches to event detection, that both define a set of primary events that can be detected in video-streams using e.g. segmentation and that both assume every event can be formulated in terms of the primitive events. In the *rule based approach* (Baillie and Ganasca, 2000), the logical and temporal relations are formulated explicitly by he designer in a set of rules. These methods resemble our approach but are not designed to learn new events[1]. Our system is capable of representing such rules. In the *inductive approach* (Siskind, 1992), definitions of events are not explicitly represented but learned by a neural network or a hidden Markov model. After training, the system is able to respond to the occurrence of

[1] supervised learning of event definitions is currently investigated and seems successful

certain events. But in such a system it is difficult to reason about events or give information about the internal structure of an event, thus making it less suitable for language production and other symbolic cognitive tasks.

5. Semantics descriptions

The conceptualization module recognizes and filters events from the huge stream of events that can be retrieved from the raw data. Only those events are retained that have some relevance to the agent, according to certain criteria. These criteria depend on the different tasks that the agent has to perform within the world, ranging from basic 'survival' tasks such as charging batteries (for a mobile robot), to higher-level tasks such as communication and cooperation with other agents.

At any time however, the agent will be performing only a limited subset of all tasks that it should pay attention to while it is in operation. For example, when a mobile robot just charged its batteries, the importance of the battery-charging task will be at a very low level. Accordingly, the events that relate to this task, such as detecting the battery level, should be perceived but not attended to unless they become important.

The event stream generated by the event perception has to be filtered so that those events that are important to the agent at a certain time step are immediately available for further processing (action planning, verbalization, etc). This is the semantic subsystem's responsibility: one of its tasks is to act as an attention focusing system, providing a concrete representation (semantic descriptions) for those aspects of the perceived environment that require immediate processing.

5.1 Semantic descriptions

In our system, semantic descriptions are combinations and manipulations of concepts that express a certain aspect of the environment; things like '*the blue square approaches the circle.*' They take the form of simple, Prolog-inspired programs, and are thus based on predicate logic. Each statement can represent a predicate (for example, `(blue ?x)`, or a function to perform on an object or an event (for example `(patient ?y ?x)` which extracts the patient from event *?x* and binds it to *?y*).

The semantic descriptions are relatively similar to procedural semantics-type descriptions such as introduced by Woods (1968) and later in slightly different form by Winograd (1976). Both Woods' and Winograd's systems were mainly aimed towards language understanding, and not language production. Especially Woods' system uses explicit procedures that can only be used to do queries on a data base. In Winograd's system there is language production, but his solutions to the problems he faced were ad-hoc and specific to the small blocks world. For example, the semantic descriptions that SHRDLU uses, are based on *semantic markers*: a predefined hierarchy of concepts that the system knows beforehand. One of the things we want to do with our system however, is precisely to study how such a concept hierarchy emerges from the interactions of the system with the world. This makes it impossible to use a closed set of predefined semantic markers, and emphasizes the need for a mechanism of abstraction, so that the system can create concepts in a hierarchical way by itself.

```
(?x | (approach ?x)
      (agent ?y ?x)
      (blue ?y)
      (square ?y)
      (patient ?z ?x)
      (circle ?z))
```

Figure 2: Semantic description for *the blue square approaches the circle*

An example of a semantic description is shown in fig. 2. This sequence of predicates says that `?x` must be an 'approach' event that has an agent `?y` which is blue and square and a patient `?z` which is a circle. The agent has an evaluator that is able to process these sequences of predicates within the set of perceptions that the agent acquired from the current context, and construct a set of bindings for `?x`, `?y` and `?z` such that the entire predicate is true if and only if such a set exists.

Variable `?x` is the head of the meaning; this is indicated by the explicit mention to the left of the operations. In this case, the 'approach' event is the head. The same sequence of predicates can also be used to express the head `?y`, in which case a natural language rendering could be '*the blue square approached by the circle.*'

5.2 Meaning Construction

Since our agent lives in a complex, changing world, we want it to adapt to this world. This means that it is not desirable to give it in advance the semantic descriptions that it will need to be able to perform its tasks. Hence, we must include a mechanism that allows the agent to create new semantic descriptions on-the-fly, as it needs them to describe something.

The mechanism that allows an agent to construct meanings of arbitrary complexity is *abstraction* or naming. The agent is not only capable of constructing complex semantic descriptions, as shown in the previous section, but it is also capable of giving new descriptions names and incorporating them in its repertoire of operations that it uses to construct semantic descriptions. A description added in this way can subsequently be used in new descriptions in the same way that primitive operations are used to construct lower-level semantic descrip-

tions. For example, the description of fig. 2 with topic *?y*, abstracted as `(operation1 ?y)` could be used as the topic in a description for *the blue square approaching the circle is large*:

```
(?w | (operation1 ?w)
      (large ?w))
```

This mechanism also completes the compositionality of the semantic system: descriptions are functions of their elements, which in their turn can be composed of several other descriptions, and so on.

5.3 Relation with conceptualization

There is an important interplay between the conceptualization and the construction of semantic descriptions. Events are constructed from raw perception data by using detectors that find correlations between the data (e.g. the distance between two objects becoming monotonically smaller), and identify them as 'events' (e.g. an 'approach' event). Semantic descriptions are constructed in a similar way, by finding relations between objects and events. When semantic descriptions are used often, this indicates that they are important, which might trigger a process to move the detection of the meaning one stage earlier, i.e. the conceptualization phase. For example, an agent might notice several times that an 'approach' event is often followed by a 'touch' event. If this happens frequently enough, the agent's semantic subsystem could instruct the event perception module to combine those two events into a new 'collision'-event detector that watches for approach-touch sequences. This could be compared to e.g. learning to dance: in the beginning one has to consciously think of every step you one makes, but after a while the whole process becomes fluent and automatic.

6. Grammar

The final component of the system deals with transforming the semantic descriptions to natural language and vice versa. This transformation is the work of a *grammar*. Many grammar formalisms exist, each with strong and weak points. For the purposes of our system we needed a formalism that can deal with the semantic descriptions that the semantic component uses. Some consensus criteria for such a formalism are (see e.g. Gamut, 1991):

Compositionality: The formalism should support compositionality, i.e. the meaning of a sentence is a (systematic) function of the meaning of its parts. Simply storing complete sentences and their meaning is not enough, nor is simply adding up the meaning of different words (as in the multiple word games, Van Looveren 2000). Instead, the formalism should be able to build-up sentences by combining lexicon entries, and attribute the *argument structure* (the "who did what to whom") based on word order or morphological markers. However, the formalism should not be restricted to only build-up sentences from words: it should be able to deal with larger units such as complete idioms as well.

Phrase structure: To attribute argument structure correctly (and in a later stage deal with e.g. stress patterns), the formalism should be able to recognize the phrase structure of sentences. I.e. it should identify "the block" as a phrase (a noun phrase) in the sentence "the circle approaches the block", and observe that "approaches the" is not such a phrase.

Recursion: It should be possible to nest phrases in other phrases, as in "the block approaches the block that just hit the triangle" (the triangle's phrase is nested in the block's phrase). Only a system that is both compositional and recursive can "make infinite use of finite means", which is seen as a fundamental property of human language (Chomsky, 1999).

Moreover, we prefer a formalism (i) that is simple, but easily extendible; (ii) that is well understood; (iii) for which learning algorithms exist; (iv) where semantic and syntactic operations are intricately tied together; and (v) that is largely lexicalized, i.e. all information is in the lexicon and words, combinations of words, and sentences (e.g. idioms) can be treated in the same way.

6.1 Categorial Grammar

Categorial grammar is such a formalism. A categorial grammar implies that every syntactic entity of a language (i.e. a word) has a grammar category assigned to it. In the simplest case, there are only two basic categories: n and s (for noun and sentence, respectively). All other categories for a sentence part p_1 can be constructed by combining the basic categories by considering the sentence part p_2 that can come immediately before or after a word. The combination of two sentence parts will be assigned a resulting category.

For example, "block" and "circle" are both of the basic category n. Now, if we want to say "the block" or "the circle", we need "the" to be of a category that must be followed by an n, and which results in something of category n. So, "the" is of category n/n: it results in a category n (the first one), if it is followed (forward slash) by a sentence part of category n (the second one).

Similarly, we can define a verb "approaches" as a sentence part that produces a complete sentence s if they are both preceded and followed by something of category n: $n\backslash s)/n$. This means that, if the verb is followed by an n, the result r is something that is of category $n\backslash s$; this means in turn that r should be preceded (backslash) by something of category n to produce an s. Table 3 shows

the	n/n
block	n
approaches	$(n\backslash s)/n$
the	n/n
circle	n

Figure 3: Syntactic categories for the words in "the block approaches the circle".

the lexicon for the words in the sentence 'the block approaches the circle'.

In our system we use a variant of categorial grammar that is slightly more general than this pure form. We represent n/n (a category that needs an n on the right side to yield an n) as `(n n r)`. A category is thus either one of the basic categories `n` or `s`, or of the structure `(yields needs constraint)`, where `yields` and `needs` are categories. The `constraint` is in the present implementation either an `l` (left) or an `r` (right), but can be extended to other types of constraints. This will be necessary to model free word-order languages (such as Latin or German).

6.2 Semantics

In the categorial grammar tradition the usual way to deal systematically with the meanings of *combinations* of lexical entries, uses Church's lambda calculus (see e.g. Gamut, 1991, for a discussion). We adopt this solution, which means that we have to extend the semantic description with the possibility to include lambda (λ) terms. Lambda terms can be seen as listing the variables that still need to be substituted; they disappear when a complete semantic description is reached.

E.g. the following is the semantic description for "x approaches y", where x and y still need to be filled in:

(1)
```
(?x | λ?x λ?y | (approach ?z)
                (argument1 ?x)
                (argument2 ?y))
```

When applied to the following semantic description

(2)
```
(?p || (circle ?p)
```

the resulting description is as follows:

(3)
```
(?p | λ?y | (approach ?z)
            (argument1 ?p)
            (argument2 ?y))
            (circle ?p))
```

I.e. the variable `?x` in (1) is replaced by the head of (2), and the `λ?x` is removed. (3) means something like "the circle approaches y", where y still needs to be filled in. (3) can in turn be applied to e.g. the description of a block, yielding a description meaning "the circle approaches the block".

6.3 Production & Interpretation

We have implemented a production algorithm and an interpretation algorithm that, given the proper lexicon, map semantic descriptions on natural language expressions and vice versa. These algorithms are rather straightforward:

production starts with a target semantic description; the system selects all its partial matches in the lexicon and searches for a way to combine these entries that yields a correct sentence (e.g. of type `s`), with a semantics that is identical to the target description. In the current version it is implemented as a standard constraint-satisfaction problem.

interpretation starts with a natural language sentence; the system finds all partial matches in the lexicon and searches a way to combine these entries such that it matches the complete sentence and yields a consistent interpretation (a semantic description, without λ's and with all variables bound). In the current version it is implemented as standard depth-first, exhaustive search.

7. An example of the system 'at work'

In this section we give an example of how the system behaves on input from a simple simulation shown in fig. 4, where two red squares are moving in opposite direction.

Figure 4: Some subsequent views on an example simulation, Time increases from left to right before top down.

In this example, the agent could be the speaker in a language game. It therefore has to pick a subject from the simulation (actually, from the items the simulation triggered into existence) to talk about, find a semantic description for it and finally verbalize this description in a grammatical correct utterance.

Thus, the first step the system takes when the simulation starts is to try to detect events. For this example, we gave the system definitions for various moving events

form	meaning			category
	head	substitution list	meaning	
"approaches"	?x	(λ?y λ?x)	((APPROACH ?z) (ARGUMENT1 ?z ?x) (ARGUMENT2 ?z ?y))	((S N R) N L)
"approached by"	?y	(λ?y λ?x)	idem	((S N R) N L)
"approach of"	?z	(λ?y λ?x)	idem	((S N R) N L)

Figure 5: The agent's lexicon entries for 'approach.'

(e.g. moving left, right, falling down) for various kinds of objects (square, rectangle) for various features (color) and for some other events (approach) For example the move left template looks for items which have a feature x-position which is decreasing.

At the end of the frame sequence shown in fig. 4, observation resulted in 5 body-items, two contact items, one move-left and one move-right item, two falling items, three approach items and three move-away items. These correspondent respectively with the five objects in the scene, the two 'walls' contacting the 'ground', the two squares moving to the left and right and falling, and the movements towards each-other and away from each-other of all combinations of objects of which at least one is moving.

The next step for the system is to pick an item as a subject to talk about and find a (preferably unique) description for it. Suppose the system picks the square moving to the right. Some descriptions found by the system for this item were:

```
(?x | ((SQUARE ?x)
       (LOW ?x)))
(?x | ((MOVING-RIGHT ?y)
       (ARGUMENT1 ?y ?x)))
(?x | ((MOVING-RIGHT ?y)
       (ARGUMENT1 ?y ?x)
       (RED ?x)
       (SQUARE ?x)))
(?x | ((ARGUMENT2 ?y ?x)
       (MOVE-AWAY ?y)
       (ARGUMENT1 ?y ?z)
       (ARGUMENT1 ?y ?u)
       (MOVING-LEFT ?u)
       (SQUARE ?x)
       (SQUARE ?y)))
```

The following step is to transform the semantic descriptions to a grammatical sentence. For this the system needs a lexicon. The lexicon entries containing the `approach` predicate for example are given in fig. 5.

The above semantic descriptions are translated to

```
"the low square"
"the moving to the right"
"the red square moving to the right"
"the red square moving away from the square
 moving to the left"
```

The system is also able to interpret sentences like the above, thereby creating semantic descriptions consistent with the utterance, finding bindings for it etc. The above utterances are not sentences but noun phrases. It is also possible to let the system describe what *is* happening in a running simulation, thereby forming correct sentences like "the red square moves to the right."

8. Discussion and future work

Is this whole setup a return to classical, symbolic AI? We think it is in the first place an attempt to combine ideas from both classical AI and new, adaptive AI. Both approaches have their difficulties and merits and both answer different questions. In many cases the problems of one approach are just the answers provided by the other. A major problem of classical AI is that is produces carefully engineered non-adaptive systems. A major problem of distributed, dynamically complex systems it that they are hard to engineer, unpredictable and, if a successful system is built it is hard to define what precisely was 'the good idea' that made it all work. In addition, by leaving it *all* up to the agent by not providing any abstractions away from the huge and vicious world to start from, we leave ourselves with an enormously difficult problem to solve, namely 'the whole thing' at once.

An important thing that should be added to the Harnad's list of challenges for cognitive theory (Harnad, 1990) is how such natural language users can learn and adapt themselves to changes in the environment. We argue language plays an important role in this and have designed a system that provides us with the means to test this hypothesis. At the same time, it allows us to investigate important issues on the origins and evolution of language.

While building an integrated and open system we had to make each subsystem powerfully enough to meet other subsystems' requirements. For example on the one hand, the semantic description language we developed needs to be able to represent and handle all things it gets from the conceptualization module. At the same time, it needs to be able to provide the input required by the grammar module. This way, requirements and specifications in

one module also produce requirements and specifications for another. If we want to incorporate tense and aspect in the language, the conceptualization module needs to know about time as well and in such a way that appropriate information can be propagated upward to the grammar module and vice versa. The mechanism of integration thus provided us at each level with some design guidelines.

But this mechanism can also be used by the system itself. If the agent 'feels' at the grammatical and semantical level that it would be useful to have a concept for something that represents an 'approach+touch' event, it can instruct its conceptualization module to create such a 'collision' notion according to the requirements and specification of the higher levels.

In the future we plan to use the system to investigate some specific aspects of language like tense, grammar, causality, etc. The main goal is to answer questions about the origins and evolution of language. We will therefore have to extend the system with good learning algorithms. In addition, we plan to replace the simulation by a camera and a robot arm. The simulation could still be used as an 'imagination' module by an agent (see Siskind (1992) for a successful implementation of this idea and related papers for a psychological evidence for such a module in humans).

9. Acknowledgments

The authors would like to thank Luc Steels, who has created the theoretical framework underlying this research, and Frederic Vannieuwenhuyse and Eefje Leydesdorff for their contributions in the design and implementation of the grammar model. JDB is funded by the Vrije Universiteit Brussel (VUB), JDB and WHZ are funded through the Concerted Research Action fund (G.O.A.) of the Flemish Government and the VUB. JVL is sponsored by a grant from the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT).

References

Baillie, J. C. and Ganasca, J.-G. (2000). Segmentation qualitative sur des series de donnees. Technical Report LIP6, Université Pierre et Marie Curie, Paris.

Baraff, D. (1994). Fast contact force computation for nonpenetrating rigid bodies. *Computer Graphics*, 28(Annual Conference Series):23–34.

Batali, J. (2002). The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In Briscoe, T., (Ed.), *Linguistic evolution through language acquisition: formal and computational models*. Cambridge University Press.

Brachman, R. J. and Levesque, H. J., (Eds.) (1985). *Readings in Knowledge Representation*. Morgan Kaufmann, San Mateo.

Chatterjee, A. and Ruina, A. (1998). A new algebraic rigid body collision law based on impulse space considerations. *ASME Journal of Applied Mechanics*.

Chomsky, N. (1999). On the nature, use and acquisition of language. In Ritchie, W. C. and Bhatia, T. K., (Eds.), *Handbook of Child Language Acquisition*. Academic Press.

De Boer, B. and Vogt, P. (1999). Emergence of speech sounds in changing populations. In Floreano, D., Nicoud, J.-D., and Mondada, F., (Eds.), *ECAL'99 Advances in artificial life*, Lecture Notes in Artificial Intelligence 1674, pages 664–673.

Drescher, G. L. (1991). *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, Ma.

Gamut, L. (1991). *Logic, language and meaning*, volume 2. The University of Chicago Press.

Hanh, J. K. (1988). Realistic animation of rigid bodies. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):299–308.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, pages 335–346.

Hurford, J. (1989). Biological evolution of the saussurean sign as a component of the language acquisition device. *Lingua*, 77(2):187–222.

Keller, H., Stolz, H., and Ziegler, A. (1993). Virtual mechanics: Simulation and animation of rigid body systems. Technical Report TR-1993-08, Universität Stuttgart, Fakultät Informatik.

Kirby, S. (2000). Syntax without natural selection: How compositionality emerges from vocabulary in a population of learners. In Knight, C., Hurford, J., and Studdert-Kennedy, M., (Eds.), *The Evolutionary Emergence of Language: Social function and the origins of linguistic form*. Cambridge University Press.

Kuipers, B. (1994). *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, MA.

MacLennan, B. J. and Burghardt, G. M. (1994). Synthetic ecology and the evolution of cooperative communication. *Adaptive Behavior*, 2:151–188.

Maynard-Smith, J. and Szathmáry, E. (1995). *The major transitions in evolution*. Morgan-Freeman.

Mirtich, B. (1995). Hybrid simulation: Combining constraints and impulses. In *Proceedings of the First Workshop on Simulation and Interaction in Virtual Environments*.

Mirtich, B. and Canny, J. F. (1995). Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, pages 181–188, 217.

Moore, M. and Wilhelms, J. (1988). Collision detection and response for computer animation. *Computer Graphics*, 22(4).

Noble, J. and Cliff, D. (1996). On simulating the evolution of communication. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. MIT Press, Cambridge, MA.

Oliphant, M. (1999). The learning barrier: Moving from innate to learned systems of communication. *Adaptive Behavior*, 7:371–384.

Siskind, J. M. (1992). Naive physics, event perception, lexical semantics, and language acquisition. Ph.D. diss. (Elec. Eng'g. & Comp. Sci., MIT).

Steels, L. (1996a). Emergent adaptive lexicons. In Maes, P., (Ed.), *Proceedings of the Simulation of Adaptive Behaviour Conference*. The MIT Press, Cambridge, Ma.

Steels, L. (1996b). Perceptually grounded meaning creation. In Tokoro, M., (Ed.), *Proceedings of ICMAS-96, Kyoto*. AAAI Press, Calif.

Steels, L. (1997). The origins of syntax in visually grounded robotic agents. In Pollack, M., (Ed.), *Proceedings of the 15th International Joint Conference on Artificial Intelligence*. Morgan Kauffman.

Steels, L., Kaplan, F., McIntyre, A., and Van Looveren, J. (2002). Crucial factors in the origins of word-meaning. In *The Transition to Language*. Oxford University Press, Oxford, UK.

Van Looveren, J. (2000). An analysis of multiple-word naming games. In Van den Bosch, A. and Wiegand, H., (Eds.), *Proceedings of the 12th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC '00)*.

von der Malsburg, C. (1981). The correlation theory of brain functions. Max-Planck-Institut, Biophys. Chem., Internal Report 81-2, Göttingen FRG.

Weld, D. S. and de Kleer, J. (1990). *Readings in Qualitative Reasoning About Physical Systems*. Morgan Kaufmann, San Mateo, CA.

Winograd, T. (1976). *Understanding Natural Language*. Academic Press.

Wittgenstein, L. (1967). *Philosophische Untersuchungen*. Suhrkamp, Frankfurt, Germany.

Woods, W. A. (1968). Procedural semantics for a question-answering machine. In *AFIPS Conference Proceedings, Fall Joint Computer Conference, Montuole, NJ*, pages 457–471.