Scalable Construction-Based Parsing and Semantic Analysis

John Bryant
Department of Computer Science
University of California at Berkeley
Berkeley, CA 94720
jbryant@icsi.berkeley.edu

Abstract

In ScaNaLU 2002, Chang et al presented a scalable natural language formalism called Embodied Construction Grammar (ECG) (Chang et al., 2002). ECG makes deep understanding systems possible because it is a rigorous, unified formalism that incorporates the semantic and pragmatic insights found in cognitive linguistics. The work described in this paper builds on (Chang et al., 2002) because it leverages the ECG formalism to perform deep, scalable construction-based parsing and semantic analysis.

1 Introduction

As described by (Chang et al., 2002), the semantic and pragmatic insights provided by cognitive linguistics must be incorporated into a language understanding system before deep understanding can take place. Embodied Construction Grammar (ECG) (Chang et al., 2002), (Bergen and Chang, 2002) is a rigorous, formalism incorporating such insight. It provides formal mechanism for describing cognitive primitives like linguistic constructions (Goldberg, 1995), image schemas (Lakoff, 1987), frames (Fillmore, 1982), and mental spaces (Fauconnier and Turner, 2002), as well as cross-domain mappings.

From a system-building point of view, however, the importance of ECG lies in its scalability. Within its unification-based framework, constructions, frames and mental spaces are combined compositionally, yielding a network of entwined semantic and pragmatic structures representing the overall interpretation. This makes it possible for ECG to scale to much more complex linguistic data that previous formalisms would allow.

The work described in this paper builds on this system-building perspective since it is a system that leverages the ECG formalism to perform deep, scalable construction-based parsing and semantic analysis. It incorporates an implementation of ECG's semantic and constructional primitives as well as integrating scalable language analysis algorithms like level-based parsing (Abney, 1996). This system is called the *constructional analyzer* and it fits into a larger framework for scalable, *simulation-based* language understanding.

In a simulation-based model of language understanding, interpretation of an utterance is split into two phases: analysis and enactment. Analysis is the process of mapping forms to context-independent meanings, providing the input parameters for enactment. Enactment uses an active, simulation-based model to generate context-specific inference. This process-level separation of analysis and inference provides further scalability.

The next three sections describe the ECG formalism, simulation-based understanding and partial parsing. Then the constructional analyzer is described along with an example. The paper closes with a description of a language analysis task requiring the deep semantic representation afforded by the constructional analyzer.

2 The Embodied Construction Grammar formalism

The grammar formalism that makes constructional analysis possible is the Embodied Construction Grammar. ECG combines a syntax and knowledge representation in a unification-based framework. This allows both constructions and frame-based, schematic knowledge to be expressed succinctly in the same formalism.

As usual for construction grammar, the grammar rules in ECG are pairs, mapping a particular lexical/morphological/syntactic pattern to a (partial) specification of a situation. In ECG, this description of a situation is known as a semantic specification (semspec). The semspec combines embodied semantic primitives like

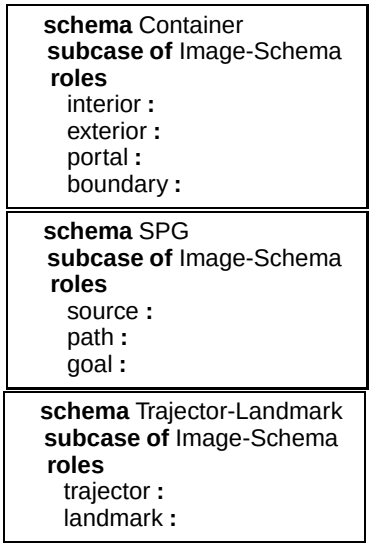

Figure 1: Some image schemas in ECG notation.

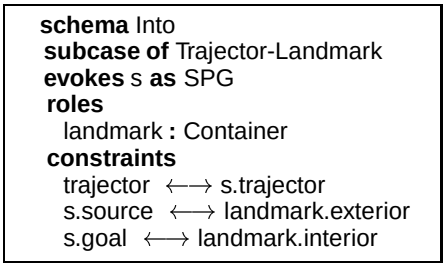

Figure 2: The Into schema.

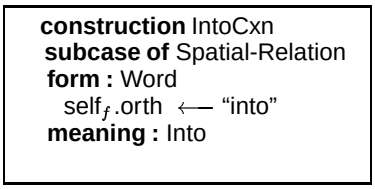

Figure 3: The Into lexical construction.

image schemas (Lakoff, 1987) and executing schemas (Narayanan, 1997) with frame-based knowledge (Fillmore, 1982) to completely specify the meaning of an utterance.

Meaning in ECG is represented by *schemas*. Schemas, much like frames, are schematic, role-based conceptual and semantic structures. But as they are intended to describe an embodied, parameter-based representation of language, the schema formalism is augmented by special semantic operators that make cognitive linguistic insights easier to express.

As an initial starting point into the formalism, figure 1 shows three canonical image schemas in ECG notation. Each schema is initially defined by the keyword **schema**, and after the name, an optional **subcase of** line denotes the structures from which the current schema inherits. Much like frames, ECG schemas (and constructions) are arranged into a type hierarchy from which a schema can inherit structure. In this case, each image schema inherits from the Image-Schema type. Following the **subcase of**, comes the optional roles block, denoted by the **roles** keyword. Just like the roles of a frame, the roles of a schema are the parameters of the concept being described.

These simple image schemas do not show off all of ECG's schema mechanism, however. For a more complete picture, we now focus on the Into schema shown in figure 2. The Into schema subcases the Trajector-Landmark schema, and thus inherits the trajector and landmark roles from its parent. The Into schema further constrains the landmark role by constraining it to be of type Container.

The Into schema also introduces the new **evokes** operator which makes the specified type locally accessible via the given alias. In this case, the evoked SPG schema acts as the background frame, capturing *into*'s notion of motion. This is the primary virtue of the **evokes** operator. It provides a way to place concepts such as *bachelor* into the context of their larger background frames, as described by (Fillmore, 1982).

After the roles block in the Into schema, comes the optional constraints block. Constraints act as the semantic glue with the $\leftrightarrow$ identifying its two argument slots. When a set of slots have been coindexed, a proposed filler to any one of those slots must satisfy the restrictions of all of them. In the Into schema, the locally defined landmark.interior[1] role[2] is identified with the evoked SPG's goal role, while the landmark.exterior role is coindexed with the SPG's source role. These constraints schematically describe how the use of *into* suggests motion from outside some container to the inside of that container.

Figure 3 shows the Into lexical construction. Every construction starts with the keyword **construction**, followed by the name of the construction. Then comes the optional **subcase of** keyword that relates constructions to the constructional type system. The ECG version of the Into construction has a form and meaning pole, notated by the keywords **form** and **meaning**.

Constructions can type their form and meaning poles. In the case of our Into construction, the form pole is of schema type Word[3] and the meaning pole is of type Into schema. A typed meaning pole indicates that a particular construction denotes an instance of that type. Thus

[1] ECG uses slot chain notation.

[2] The landmark role has an interior role because it was constrained to be of type Container.

[3] Form in ECG is also represented with schemas.

our Into construction claims that the word *into* means an instance of the complex relation described by the Into schema.

The Into construction also exhibits the assignment operator ($\leftarrow$). This operator fills a slot with an atomic value. In our Into construction's form pole, the orth[4] feature brought in from the Word schema is assigned the atomic string *into*.

Figure 4 shows the clausal Caused-Motion construction, an example of which is *The basketball player threw the ball into the basket.* This construction has an agent (the player) which acts upon a patient (throwing the ball) thereby moving it along a path (into the basket). Since the Caused-Motion construction links a particular syntactic form, that of a referring expression, a force-application verb, a referring expression and a path to a Caused-Motion-Scene[5], the construction is different from the ones we have covered so far in that it has constituents that are themselves constructions. Thus instead of typing the form block, the form block has constraints relating the constituents.

Each of the construction's four constituents are defined in the constructional block. Each constituent is assigned a local name, and then after the colon, the constructional type is defined. If necessary, like in the case of the Verb constituent, a semantic type is added in brackets.

The ordering of these constituents is specified by adding form constraints to the form block. When the constructional analyzer searches for instances of a construction, these form constraints must be satisfied. The two supported constraints are **before** which requires that the left argument be somewhere the left of the right argument in the input, and **meets** which requires the left argument to be directly before the right argument in the input.

In the Caused-Motion construction, the form constraints require that the agent be directly before the verb and the verb be before the path and patient. Notice that the relative order of the path and patient is left unspecified[6]. Because ECG allows constituent order to be unspecified like this, ECG can express with one rule what a CFG might have to express with an exponential number of rules.

The meaning pole of the construction uses the semantic machinery that has already been described. It links the agent_m's category to the agent of the scene as well as setting patient_m.category to the trajector of the specified path. Notice that the constraints use the *m* and *f* subscripts when referring to the constructional constituents'

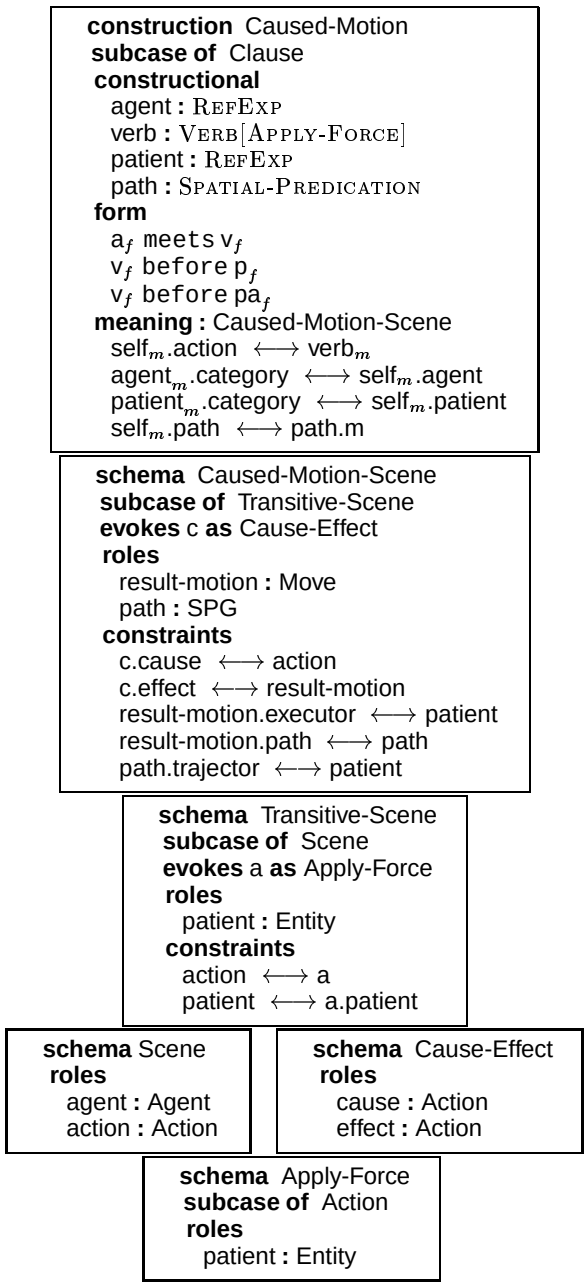

Figure 4: The Caused-Motion Construction and related schemas.

[4]Orth is short for *orthography*.

[5]A caused motion scene is one where the agent applies force to the patient resulting in a shift in the position of the patient.

[6]This might be a partial solution for dealing with what are called *heavy NPs*.

form and meaning poles, respectively, and can be applied to any construction as if they were just dotting into the structure.

With a formal language for describing constructions, many avenues are opened. The most important for the sake of this work, is that it is possible to translate ECG descriptions into feature structures. For the most part, this translation is straightforward. For example, schemas, constructions are represented as feature structures and their roles are represented (unsurprisingly) as roles.

The evokes operator, however, requires a little more care to properly model its nonlocal semantics. In this case, the evoked type is generated outside of the evoking structure. The evoked structure's alias is represented as a local role coindexed with the nonlocal structure[7].

3 Simulation-Based Language Understanding

Simulation-based language understanding draws inferences about actions and events by executing an active model. The active models are called *x-schemas* (short for executing schemas) which are extensions to stochastic petri nets. In order to draw inferences about an action like walking, the system performs a *simulation* by executing its internal model of walking.

To tailor the inference to a particular scenario, the simulator needs to set the *parameters* of the x-schema representation appropriately. These parameters are the free variables that control how the simulation executes. For example, the walking x-schema would have parameters for who the walker is and for the path (or direction) of motion. So in the sentence *Harry walked into the cafe*, the walker would be *Harry* and the path would be *into the cafe*.

But before a language understanding system can utilize the virtues of such a model, the parameters must be extracted from the utterance. This is where the constructional analyzer comes in. If the constructions have features designed to interact with the simulator, each constructional analysis will provide the parameters to the simulation, and the analyzer and simulator will interact to understand the utterance.

Researchers have integrated the constructional analyzer with a simulation engine, creating a unique and powerful method for language understanding. Such a system would even be able to do "exotic" metaphorical inference with ease, since metaphorical inference is just a special kind of parameterization[8].

[7] With this implementation, ECG does not require any extensions to standard unification algorithms.

[8] See (Narayanan, 1997) for more information about metaphorical inference.

4 Syntactic Chunking

Traditional chunking is a simple parsing algorithm where each syntax rule is transformed into a finite state recognizer. The recognizers are arranged into levels, controlling the order of the reductions. A chunker starts at the lowest level (and hopefully most-certain reductions) which are done by the part of speech tagger. Then it proceeds up the levels toward the less certain reductions. This bottom up approach assumes (obviously incorrectly) that reductions can be made locally and bottom-up without any notion of global context, giving the algorithm speed and state savings.

Within a single level, the reductions are done greedily. Going left to right through the input, whenever a particular pattern is recognized, the longest match of that pattern is reduced. Figure 5 shows the steps that an Abney chunker goes through for the sentence, *The woman in the lab coat thought you were sleeping*.

This figure exposes both the virtues and weaknesses of chunking–the weakness being the fact that the parse for the sentence does not attach the prepositional phrase *in the lab coat*, treating it instead as a sister to the *the woman* noun group. But it also illustrates the robustness of the system in that while the grammar has no rule for reduced relative, it still correctly labels the syntactic groups. In other words, the incompleteness of the grammar does not cause phrases to be rejected since there is no requirement that a successful parse converge to a complete sentence. Thus such a parser is well-suited for spoken language (Hinrichs et al., 2000), where Hinrichs, et al were able to get 93% "correctness" [9] at finding the various syntactic groups from an error prone spoken language input. Unsurprisingly without any semantics, they were only able to achieve 54% "correctness" when generating the complete parse tree from the chunk analysis.

5 The Constructional Analyzer

Because the approach to constructional analysis we describe in this report uses a level-based processing model, it can be considered a relative of such shallow information extraction tools. But instead of doing shallow semantic analysis for the purposes of information extraction, it utilizes both the semantic richness of construction grammar and extended computational mechanisms to do full constructional analysis, resulting in both a complete syntactic analysis and a deep semantic analysis. The constructional analyzer integrates the following computational capabilities to make deep analysis possible.

- Support for unification

[9] They do not give a notion of what correctness means in their paper.

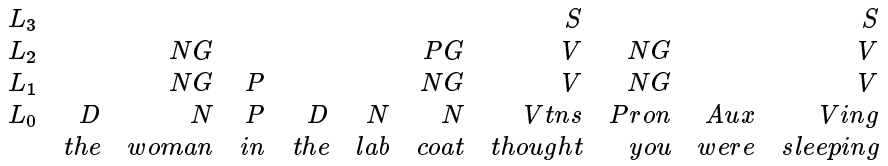

Figure 5: A Partial Parse from (Abney, 1996). Since this processing model is bottom up, first the level 0 reductions are performed, then the level 1 reductions, then level 2 and finally level 3. In the nonterminals, *G* is used (for group) instead of *p* (for phrase) to clearly indicate that these are non-recursive chunks. Each grammar symbol spans all the input to its left until another grammar is encountered.

- Support for multiple concurrent analyses with a chart
- Support for the more flexible form relations found in ECG
- Support for ECG's semantic expressiveness

5.1 The Basics of the Constructional Analyzer

Since ECG is a unification-based formalism, supporting unification is a necessary first step. Along with unification, a chart is employed by the system to keep track of the many possible analyses generated during rule-based language analysis.

On the form side, the analyzer cannot use finite state machines or even context free grammars to do matching because of ECG's more relaxed notion of form. ECG does not require the grammar writer to specify a total ordering on a construction's constituents, and thus recognizers requiring a total ordering (like CFG parsers) in each rule are unusable. The constructional analyzer instead uses a computational unit called a *construction recognizer*.

A construction recognizer is the chunk of active knowledge into which a construction gets transformed. Each construction recognizer is designed to check both the form and meaning constraints of just the construction it models. In other words, instead of a monolithic parser that takes the grammar rules as input, the constructional analyzer itself is a collection of active construction recognizers working together to generate the constructional analysis.

5.2 An Example

In order to make the previous discussion more concrete, let's analyze an example sentence using the Caused-Motion construction described earlier. The sentence we will consider is *The basketball player threw the ball into the basket.* Given a grammar that can handle simple referring expressions of the form (Det) Adj* Noun+ (making sure to add the appropriate semantics) and spatial phrases, we can arrange the rules into levels (see figure 6) and generate analyses that use the Caused-Motion construction.

0. Lexical constructions
1. Noun noun compounds
2. Adjectives
3. Referring expressions
4. Spatial Predications
5. Clausal constructions

Figure 6: The levels used in the example analysis.

construction Noun-Noun-Compound
subcase of Category
constructional
a : CATEGORY
b : CATEGORY
form
a_f meets b_f
meaning
$self_m \longleftrightarrow b_m$

Figure 7: A Generic Noun-Noun-Compound construction that just sets the meaning of the construction as a whole to be that of the second constituent. It relies on the structure merging process (Bryant, 2003) to infer the correct relation between the two Category constituents.

Figure 7 shows an example noun-noun compound construction used in the analysis that puts constituents of type category[10] together to generate an instance of type Noun-Noun-Compound which is itself subtype of category. Thus the rule is recursive with itself and any other category requiring rule. Notice that it is on the same level that all other category constructions are assigned. The constructional analyzer allows the constructions assigned to the same level to be mutually recursive.

After the Category constructions are processed, simple

[10] An instance of the category construction can either be what is usually considered a noun like *dog* or a noun-noun compound like *vinyl siding salesman* or *gas meter turn-off valve.*

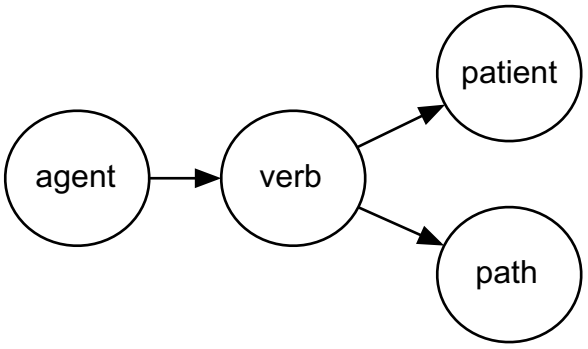

Figure 8: The constituent graph structure for the Caused-Motion construction. Each constituent corresponds to a node in the constituent graph. At any particular point, the construction recognizer is only allowed to search for constituents with no incoming edges. When a constituent is found, its node is removed from the graph along with any outgoing edges from that node. After removing a node, the construction recognizer is now allowed to search for different, newly-released constituents or if there are no nodes left, then a valid instance of the construction (at least with respect to the form constraints) has been found.

referring expressions are generated. After the referring expressions, Spatial-Predication constructions are recognized on the next level, and the constructional analyzer is finally ready to match the Caused-Motion construction. Figures 8 and 9 (and the rest of this section) describe the matching process schematically.

In frame A of figure 9, the construction recognizer is in its initial state and it has not found any of the constituents for the Caused-Motion construction. In B, the recognizer has found a referring expression corresponding to *the basketball player*, and since it unifies with the agent role of the Caused-Motion construction, it is accepted as the first constituent. Notice how the node in the graph corresponding to the agent is removed indicating that it has been found.

In frames C and D, the same scenario takes place except it is the verb *threw* and the referring expression *the ball* that satisfy the form and meaning requirements of their corresponding constituents. Notice in C that the construction recognizer is now allowed to find either the patient or the path since both nodes have no incoming edges. In E, we see a completely matched Caused-Motion construction with a complete Caused-Motion scene and an empty constituent graph indicating that a complete instance of this construction has been found.

In short, the construction recognizer builds up a graph data structure to keep track of the constituents and an in-progress semspec to keep track of the semantics. Each constituent that satisfies the form and semantic constraints updates the constituent graph and the in-progress partial semspec. The final result for a successful match has the agent of the caused motion scene to be the player, the patient being the ball, and the goal of the path being the interior of the basket.

5.3 Computational Complexity

At its core, the constructional analyzer is just a unification parser that needs to support the quirks of a particular grammar formalism. The quirk that most affects the computational complexity of the constructional analyzer is ECG's support for partial constituent order.

Barton (1985) showed that parsing with unordered CFG rules is NP-complete. Since an ECG grammar can leave all constituency unordered, the worst case runtime must be exponential with respect to the utterance. An actual grammar of this sort is unlikely, however, and thus it is useful to symbolically bound the runtime of the analyzer in terms of the number of unifications it performs.

For purposes of the complexity analysis, call the number of levels l, the number of recognizers at each level r, and assume an input of k words. Further assuming that there are $O(k)$ states in each chart entry, and that the number of constituent orderings allowed by a grammar rule is j for the max of c constituents used in any rule. The worst-case runtime would then be $O(lrk^{cj})$ since there would be k^c combinations of unifications for each of the j constituent orderings associated with a particular recognizer.

Clearly any product of c and j much over 2 makes the algorithm intractable for all but the smallest grammars. Consequently, an open area of research is a mechanism for concentrating effort on the most promising syntactic and semantic combinations using methods of the sort described in (Narayanan and Jurafsky, 1998) and (Bryant, 2003).

6 Applications

The constructional analyzer is currently being put to use in two tasks that require the deep semantics that it provides.The first task is that of *Simulation-Based Language Understanding* (Narayanan, 1997) has already been described. The second is the task of inductive learning of child constructions (Chang and Maia, 2001).

6.1 Child Language Learning

The language learning model used by Chang (Chang and Maia, 2001) is a comprehension-based model of language learning built on the following assumptions:

- There is significant prior knowledge going into the learning process.

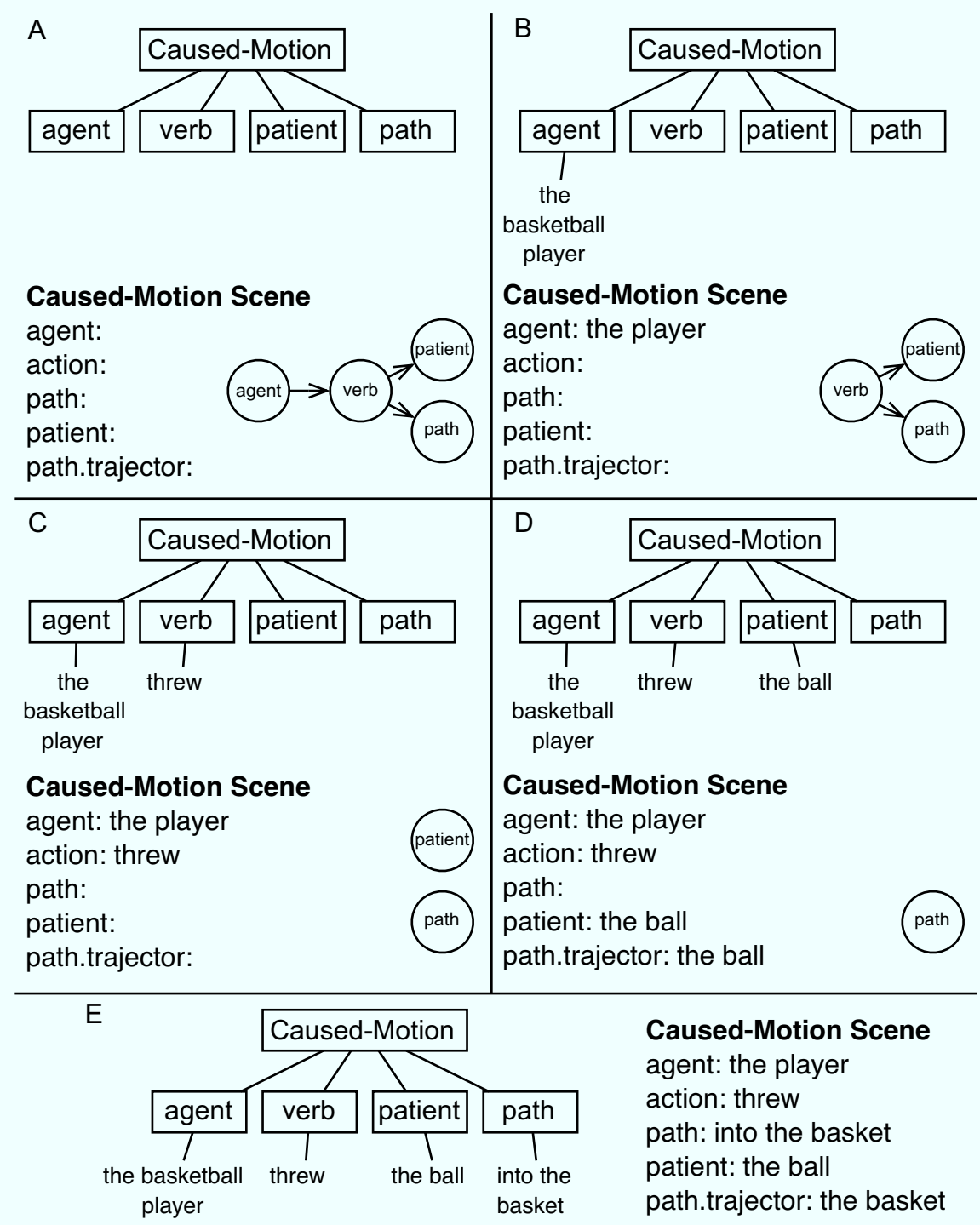

Figure 9: Snapshots of the internal state of the Caused-Motion construction recognizer on the sentence *The basketball player threw the ball into the basket.*

- The learning is incremental and based on experience.

- The learning is tied to language use. i.e. Frequency of language data affects the learning.

The model analyzes the incoming utterance using the current set of constructions that have been learned. If the current analysis generated by the constructional analyzer cannot explain all of the semantic content found in the current scenario associated with the utterance, the model hypothesizes new constructions. This hypothesis process pairs up the unused form relations with the missing semantic relations to produce constructions that fill in the semantic/pragmatic gap. Since the hypothesis process is under-constrained, the model generates multiple constructions in an attempt to explain the same missing semantic content. The more useful of these constructions in later analyses are the ones that get reinforced while the others wither away.

This model of learning depends on the language analyzer to initially try and explain the semantics of the scene. But for such an analyzer to be useful it needs to be semantically focused. It also needs to be capable of incremental analysis as well as tolerant of noise and missing constructions. These requirements line up perfectly with the constructional analyzer's virtues primarily because the language learning task heavily influenced the design of the constructional analyzer. In effect, the constructional analyzer was built on the assumption that all grammars, not just grammars in the process of being learned, will lack coverage when faced with real language.

7 Conclusion

Cognitive linguistics has provided the theoretical basis for turning natural language systems into broad coverage systems, but a formal mechanism to describe these theories was a necessary first step before natural language systems could take advantage. ECG stepped in to provide such formal mechanism, and the first system to profit is the constructional analyzer.

The work is far from over, however. While the current system does (theoretically) scale with respect to linguistic coverage, it still does not scale with respect to computational performance. Thus further computational work is necessary. The deep semantics must be leveraged to make the systems computationally faster and more robust. Initial work in this direction has already begun (Narayanan and Jurafsky, 1998) (Bryant, 2003).

References

Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.

Benjamin Bergen and Nancy Chang. 2002. Embodied construction grammar in simulation-based language understanding. Technical Report TR-02-004, ICSI. To appear in Oestman and Reid, eds., Construction Grammar(s): Cognitive and Cross Lingusitic Dimensions. John Benjamins.

John Bryant. 2003. Constructional analysis. Master's thesis, UC Berkeley.

Nancy Chang and Tiago Maia. 2001. Learning grammatical constructions. In *Proceedings of the Conference of the Cognitive Science Society*.

Nancy Chang, Jerome Feldman, Robert Porzel, and Keith Sanders. 2002. Scaling cognitive linguistics:formalisms for language understanding.

Gilles Fauconnier and Mark Turner. 2002. *The Way We Think:Conceptual Blending and The Mind's Hidden Complexities*. Basic Books.

Charles Fillmore. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–138. Linguistics Society of Korea.

Adele Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press.

Erhard Hinrichs, Sandra Huebler, Valia Kordoni, and Frank Mueller. 2000. Robust chunk parsing for spontaneous speech. In Wolfgang Wahlster, editor, *Verbmobil:Foundations of Speech-to-Speech Translation*, pages 163–182. Springer.

George Lakoff. 1987. *Women, Fire, and Dangerous Things*. University of Chicago Press.

Srini Narayanan and Daniel Jurafsky. 1998. Bayesian models of sentence processing. In *Proceedings of the Conference of the Cognitive Science Society*.

Srini Narayanan. 1997. *Knowledge-Based Action Representations for Metaphor and Aspect*. Ph.D. thesis, University of California at Berkeley.