Lifetime Achievement Award

Computational Psycholinguistics*

Ronald M. Kaplan
Department of Linguistics
Stanford University
ron.kaplan@post.harvard.edu

I want to begin by expressing my deep appreciation to the Nominating and Executive Committees and the Association at large for this tremendous honor. I am especially proud to join my collaborators and friends who previously received the Lifetime Achievement Award: Bill Woods, Martin Kay, Lauri Karttunen, and Joan Bresnan. They have profoundly influenced the contributions for which I think I am being recognized today, as have many other colleagues and students that I have worked with closely over the years.

I will follow the tradition set down by most previous LTA recipients by describing how events in my personal history propelled me toward the concepts, theories, and algorithms I have helped to develop and have become known for. This historical perspective is quite different from a paper setting forth some current technical results.

Some former recipients, Martin Kay for example, were intrigued by language from a very early age. That was not the case for me. I started as an undergraduate at the University of California Berkeley with a major concentration in physics. I decided at one point that physical reality was a little too messy, and I also discovered that I was not interested in spending long hours in a laboratory. So after a year or two I switched to math. That carried me forward to the middle of my fourth and final year, when I began to contemplate a future as a mathematician. That also seemed a little dry and unappealing. I thought I should look around, even at that late date, and maybe find a specialization that was a little more social and, for me, a little more engaging.

1. Getting Started

I mentioned this to a friend during a brief ride in an elevator. He told me about a professor in the Psychology Department who was offering an individual major in what he called "language behavior." The major wasn't actually inside a department because it combined material from a bunch of separate disciplines: psychology, linguistics, philosophy, anthropology, and maybe even mathematics. That sounded pretty interesting, and an important property at that late stage of my undergraduate career was that it had no official prerequisite courses.

* An abridged version of this article was presented on receipt of the ACL's Lifetime Achievement Award in 2019.

https://doi.org/10.1162/COLI_a_00359

The professor, Dan Slobin, was an expert in child language acquisition, and, as I later learned, child language acquisition played a prominent ideological role in the newly popular approach to linguistics, transformational grammar. I looked at some of the courses that other students had put into their individual-major schedules and at some of the core readings in psycholinguistics that they had included. The field was brand new, it didn't have the long history of other disciplines, and therefore not a lot of material to be mastered before jumping in. And I realized that this was a neat application of formal mathematical methods to a collection of humanistic problems. All to the good—I signed up with Dan.

I also decided to take the few remaining math courses that I needed to complete that major as well. So at that point I guess I had become, if only by accident, a mathematical psycholinguist. I was awarded a double-major undergraduate degree, and I was admitted to graduate school in social psychology, which is where psycholinguistics was located at Harvard.

I encountered a few high-level ideas during that undergraduate period that have reverberated throughout the course of my career. Chomsky's (1965) *Aspects of the Theory of Syntax* had appeared just a few years before. It provided some formal details of the first main revision to the basic machinery of transformational grammar, but that wasn't what struck me. Rather, there was a simple statement at the beginning that set forth a particular view of the relation between a speaker's knowledge of language, the result of learning a language, and the ability to put that knowledge to use in comprehension and production. He said:

> No doubt, a reasonable model of language use will incorporate, as a basic component, the generative grammar that expresses the speaker-hearer's knowledge of the language; but this generative grammar does not, in itself, prescribe the character or functioning of a perceptual model or a model of speech production. (Chomsky 1965, page 9)

This is a clear statement of what became known as the Competence Hypothesis. I understood it as making two claims: First, that a representation of linguistic knowledge should play a causal role in the mental processes of everyday communication by language. And second, that linguistic knowledge was only one of many components of a larger model of language performance.

The first claim, I thought, distinguished this application of formal descriptions from the way that mathematical characterizations are deployed in most other sciences. The sun, for example, *behaves* according to the partial differential equations for energy transport, but presumably it does not solve a representation of those equations in order to decide what to do next. But the proposition here is that the psychological system encodes and then interprets something that correlates with generalizations about the properties of language that emerge from linguistic investigation. That would be a very neat state of affairs, if it were true.

The second claim places this in the context of another powerful conception, the notion of a "nearly decomposable system" that Herb Simon put forward in a short essay entitled *The Architecture of Complexity* (Simon 1962). Simon argued that apparently complicated systems, if created artificially or through an evolutionary process, are likely to be recursively decomposable into collections of separate components. Such systems appear complicated because these components interact in non-negligible ways. But those interactions are relatively simple compared to the interaction of elements within the individual components. It is thus fruitful, Simon would say, to construct scientific explanations of such systems by treating each of the components independently at first, and then layering on top an account of their interactions. Chomsky's statement was consistent with this kind of decomposable architecture, although he never got around to the layering-on-top part. That became the focus of some of my work.

You will have noticed that my story so far has not included a mention of computers or computation. I had been involved in computing during my physics period—I had a summer job computing the fluorescent signature of a high-altitude nuclear explosion, very topical in those days. And I had taken numerical methods courses in the math curriculum. But there was as yet no connection between computing and language behavior.

Because I had finished up at an odd time, in the spring of 1968, there was a six-month interlude between leaving Berkeley and starting up at Harvard. I went home to Los Angeles to look for a temporary job. In another chance encounter, another friend with connections to the Rand Corporation told me that Rand had a linguistic group that maybe I could hook up with. I sent in my papers and went for an interview. I found that they did have an interest in mathematical psycholinguistics, of all things, and that I could come on as a six-month intern.

Dave Hays was the head of a group that included Martin Kay, and Lauri Karttunen was there just finishing his dissertation. You may know that Dave was one of the founders of our field. He coined the term "computational linguistics," helped to create the American precursor organization of the ACL, and was involved in setting up the international COLING conferences. He was also a very smart man, in many ways ahead of his time. He was an early advocate for dependency grammar, for example, as a counter to transformational grammar and immediate constituent approaches (Hays 1964). And in my case I remember that he strongly urged me to learn about regular languages and finite-state machines. Having read *Syntactic Structures* (Chomsky 1957), I knew of course that regular languages were totally uninteresting. And coming from Berkeley I thought that the acronym FSM stood for Free Speech Movement. But I did try to learn at least a little bit.

For my first project, Dave sent me down the hall to work with Lauri on some semantic issues. As I recall, our computational task was to implement a semantic network within a Rand time-sharing system. I was also involved in Lauri's linguistic investigation into some properties of noun-phrase reference—the contexts in which definite and indefinite descriptions can be properly used. Semantics turned out not to be so hard. I went back to Dave after a few months and told him that we had finished that project. We now wanted to add on a front end to make it easier to interact with our system.

He told me to go down the hall in the other direction and see what I could get from Martin Kay. Martin had developed a parsing program that was not based on transformational grammar but seemed expressive enough to recognize the patterns of natural language (Kay 1967). It was a system for rewriting sequences of trees into sequences of trees, with Turing machine power. But it used a very efficient data structure, called the chart, to hold the intermediate state of the computation. There was a missing ingredient, however: Martin had the program, but he didn't yet have an English grammar to drive it. That became my project for the rest of the summer, and in some sense, for the rest of my life. That was how I learned about parsing; that was my entry into computational linguistics.

2. Psycholinguistics

The summer came to an end and I reverted to my psycholinguistic studies at Harvard. My advisor was Roger Brown, probably the pre-eminent figure in child language acquisition at the time, and language acquisition was still a focus particularly at Harvard. As a first experiment, I decided to see whether two-year-olds were able to make some of the definite/indefinite reference distinctions that Lauri and I had worked on. The experiment failed—the children ignored both what I said to them and the rubber ducks that I gave them to play with. On the basis of that experience I decided I didn't have the patience or interest needed to work with children. I shifted to adult comprehension.

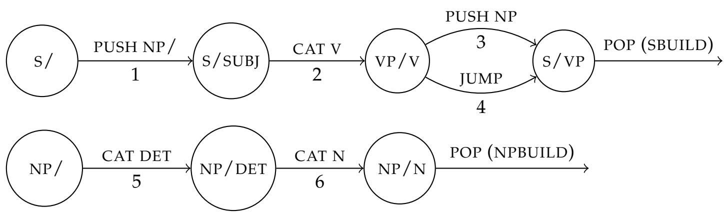
As per Chomsky's suggestion in *Aspects*, psycholinguists had investigated a theory of comprehension that incorporated transformational grammar as its repository of linguistic knowledge. They tested the simple assumption that, if that were the case, the complexity of sentence comprehension ought to be proportional to the number of grammatical transformations that applied in the linguistic derivation of a sentence. This idea, called the Derivational Theory of Complexity, received some initial empirical support. The measurement techniques of that era were crude at best, but it appeared in early studies that passive and negative sentences, involving two transformations, were more difficult for people to deal with than simple declaratives (Miller and McKean 1964). Later experiments, however, showed the opposite effect: application of an additional transformation had the effect of reducing psychological complexity. Slobin (1968), for example, observed this in the case of the transformation that deletes the agent by-phrase of a passive sentence. The sentence gets shorter, and easier to process.

By the late 1960s psycholinguists had more or less given up on the Derivational Theory of Complexity. In fact, they had basically given up on Chomsky's suggestion that a model of performance would incorporate a representation of the native speaker's linguist knowledge. They began to operate independently of linguistics, to form heuristic models of language behavior by generalizing just from experimental results, to summarize experimental results in a collection of what they called "perceptual strategies." One strategy, that the first Noun-Verb-Noun sequence in an English sentence is taken as the Agent-Action-Object, worked for simple actives but not for passives, thus accounting for the fact that passives seem to be harder to process. The problem, of course, was that this did not explain how passives could be understood at all—there was no obvious back-off when a strategy failed to apply.

Although the Derivational Theory of Complexity did not work out, retreating to a collection of disconnected perceptual strategies seemed like an overly weak response. The problem, I thought, was not in the general conception laid out in the Competence Hypothesis that systematic knowledge of language should be incorporated in performance models. Rather, the mistake was in choosing a grammatical framework, the transformational formalism, whose processing characteristics were so obviously unrealistic. Mainstream linguistics was locked onto that formalism, but I knew from my work with Martin's parser that there might be other sufficiently expressive alternatives and perhaps some that are psychologically more plausible.

My reputation as a grammar engineer somehow followed me from Rand to Cambridge. A project was starting up at Bolt Beranek and Newman to build a natural language interface that would enable lunar geologists to access geophysical data on the moon rocks that had been brought back from the Apollo missions. Bill Woods was heading the project, and one of the system's key components was the Augmented Transition Network parser that he had developed (Woods 1970).

As is well known, an ATN grammar, as illustrated in Figure 1, consists of a collection of finite-state transition networks, each of which describes how a phrase of a given category (S, NP, etc.) can be realized, and a recursive control that allows for one phrase to be embedded in another. The transitions are augmented with actions that can store



Arc	Condition	Action	Arc	Condition	Action
1		(SETR SUBJ *)	5		(SETR DET *)
2	(SVAGR SUBJ *)	(SETR ACTION *)	6		(SETR HEAD *)
3	(TRANS ACTION)	(SETR OBJ *)			
4	(INTRANS ACTION)				

Figure 1
A schematic ATN grammar for transitive and intransitive sentences.

information in local variables, called registers, and registers are passed forward to condition what may happen on later transitions. Whenever a network traversal reaches a final state, information in the local registers is used to build a tree intended to imitate the deep structure that a linguistically motivated transformational grammar might assign to that constituent.

I took over the task of extending an English grammar written in the ATN notation, and that was my contribution to the first version of the Lunar question-answering system (Woods and Kaplan 1971). Bonnie Webber joined us and extended the semantics and other features for the second version (Woods, Kaplan, and Nash-Webber 1972). Lunar and its ATN parser and grammar are described at greater length in Bill's LTA acceptance paper (Woods 2006) and elsewhere. The coverage and accuracy of the Lunar grammar demonstrated the power of the ATN formalism to provide relatively simple descriptions of complicated English sentences.

The ATN formalism also had psycholinguistic potential. In the natural order of execution for the transformational formalism, the entire input string is scanned once for every transformation. That seemed psychologically implausible on its face. To the extent that a transformation encodes separate syntactic generalizations, this direct interpretation pits the needs of the language learner—presumably to identify as many independent generalizations as possible—against the needs of the language understander—to focus on the words at hand. In contrast, the natural order of execution is reversed in the ATN set up. Discounting failures resulting from incorrect heuristic choices, the ATN parser makes a single pass through the surface phrase structure of the input string, applying only the grammatical options and constraints that are relevant at each phrasal position. This seemed to comport better with the view that the human language faculty is an evolutionary compromise between the requirements that languages be easy to learn, easy to produce, and easy to comprehend.

I wrote an early paper (Kaplan 1972) describing how the ATN parser and a simple ATN grammar could be organized to model the complexity predictions of some of the proposed perceptual strategies. I showed that those strategies could be simulated by carefully fixing the order of arcs leaving each state and using depth-first backtracking for recovery when a chosen arc does not lead to a successful traversal.

My colleagues Eric Wanner, Mike Maratsos, and I investigated the ATN modeling approach in a different way. The relative-clause sentences in Example (1) are equal in terms of their grammatical complexity and they both express essentially the same predicate argument relations. But clearly they differ in their psychological complexity.

a. The dog that the rat that the cat ate bit died. (1)

hold *hold-list memory* *retrieve*

b. The cat ate the rat that bit the dog that died.

The ATN uses some special actions and a special memory, the so-called hold-list, to handle the long-distance filler-gap dependencies in relative clauses and questions. The different nouns in Example (1a) get stacked up in that memory, perhaps overloading it, while in Example (1b) the early noun is removed before the later one is added. We proposed that the hold-list is a particularly costly psychological resource and set out to test what we called the "hold hypothesis".

One set of experiments was enough to get me a Ph.D. The sentences in Example (2) are the same except for the relative clause verbs:

a. The driver that the patrolman told to stop had exceeded the speed limit. (2)

predicted complexity difference

b. The driver that the patrolman failed to stop had exceeded the speed limit.

By virtue of the different lexical properties of *told* and *failed*, the hold-list is empty at the word *stop* in Example (2a) but still contains a representation of *the driver* in Example (2b). If the hold-list memory is expensive, processing load should be higher at *stop* in Example (2b) than Example (2a) and the difference should die out at *exceeded*. As shown in Figure 2, this prediction was borne out using a reaction-time monitoring task as a measure of transient processing load and a variant of these sentences with more words in the region of contrast (Kaplan 1974). Wanner and Maratsos (1978) found further support for this hypothesis with other measures and a different syntactic contrast.

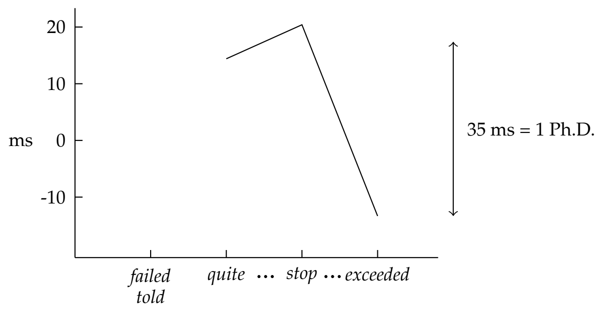


Figure 2
Hold hypothesis: *failed–told* reaction time differences from (Kaplan 1974).

In my thesis I also articulated a particular architecture for performance models with components that were relatively independent and could be studied separately, along the lines of (Simon 1962). Such a model would include

- a grammar in some formalism, as a repository of linguistic knowledge,
- a processor that allocates computational resources to interpret that formalism,
- an agenda of order and preference specifications to govern a presumably nondeterministic search, and
- a vector of coefficients that map processor resources into predictions of cognitive load.

The individual components should be evaluated as to how well they fit into an overall model, but they should also be evaluated as explanatory artifacts in their own domains. The grammar as a repository of linguistic knowledge should be a revealing encoding of linguistic generalizations, while the processor should admit of a well-engineered implementation.

A combination of linguistic and implementation considerations led me to propose some basic changes to Bill's ATN framework. As I mentioned, the internal state of an ATN network traversal was determined by mini-procedures on the arcs and recorded in a collection of local registers. These were converted to deep-structure trees at the end of a valid path, in imitation of conventional linguistic theories. I noticed that information was often lost in this conversion, because it was overlooked in the specification or because there was no natural place in a tree for some of the features to reside. Moreover, the parser contained extra routines for constructing trees from registers at the end of each traversal, and additional tree-walking code to inspect those trees at higher levels of recursion.

Information loss could be avoided and many subroutines could be eliminated by changing the underlying representation produced at each level of recursion and for the entire input string. Instead of converting to deep tree structures, I proposed (Kaplan 1974) simply taking the collection of registers, encoded as a hierarchical attribute-value list, as the output of the parsing process. Figure 3 illustrates the migration from a deep-structure tree to an attribute-value representation for a passive sentence with an embedded relative clause (Example (3)).

The rat that bit the dog was eaten by the cat (3)

Attribute-value matrices of this type are the original, primitive representations of what became the functional structures of Lexical Functional Grammar and the feature structures of other unification formalisms.

A second observation motivated a substantial reduction in the inventory of actions and conditions that could be used to set and test the values of registers during a network traversal. A parade example of ATN elegance is the customary way of relating the passive sentence in Example (3) to its corresponding active in Example (4).

The cat ate the rat that bit the dog. (4)

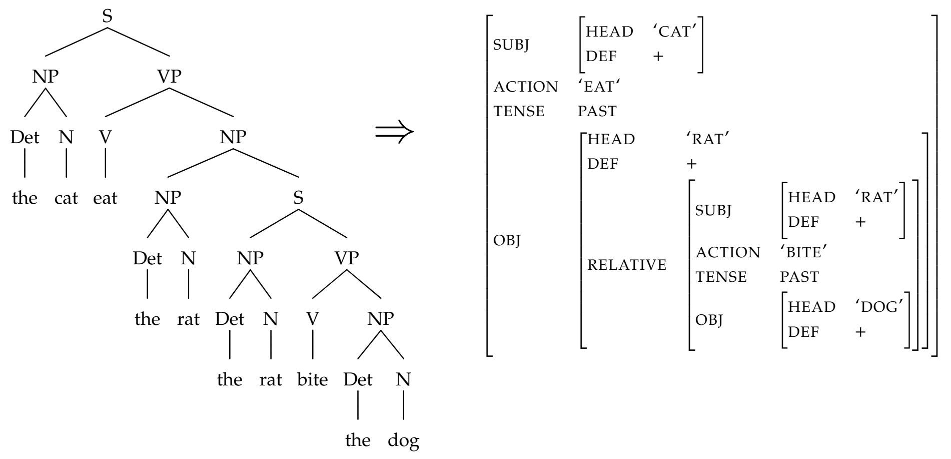


Figure 3
Migration from a deep-structure tree to a hierarchical attribute-value matrix.

Roughly, for both the active and passive the top-level sentence network first guesses that the initial NP is the subject and the immediately following verb is the action. This guess is corroborated when the verb is followed by an object NP, as in Example (4). If instead a participle like *eaten* is encountered after *was*, as in Example (3), the network branches to an alternative path set up to handle passives. On that path the register contents are rearranged so that the participle becomes the action, what was initially guessed as the subject becomes the object, and the NP after *by* becomes the final subject, correcting the initial guess. In the end the active and passive sentences are assigned essentially the same attribute-value structures with the subject and object values denoting the logical arguments of the action *eat*, as shown in Figure 3. An attractive aspect of this analysis is that it involves no backtracking over strings or recomputation of phrases, just the ability to modify register values. The perceptual strategy complexity difference could be attributed to this extra fix-up computation.

This simple solution breaks down in the face of tag questions:

a. Mary kissed John, didn't she? (5)

b. John was kissed by Mary, wasn't he?

The pronoun in the tag agrees with the original surface subject, not the logical subject, whether the main clause is active or passive, and the auxiliary verb in the tag is determined by the initial verb of the main clause. The initial settings of those registers must be preserved, if the tag is to be analyzed correctly. The motivation for the original ATN register operations is further weakened by the grammaticality contrast in Example (6).

a. The sheep that eats grass runs. (6)

b. *The sheep that eats grass run.

The verbs of the main and relative clauses must match in number when the first noun-phrase is understood as the subject of both clauses. If the number of the first noun is explicitly marked, this would follow indirectly from the local agreement of each verb with its own subject in the normal left-to-right order of register calculations (the SVAGR condition in Figure 1). But the number of *sheep* is naturally unspecified and there is no common value for the verbs to agree with separately through standard register operations.

I concluded from these and other examples that the set of register actions and conditions could and should be replaced by one composite operation defined to merge two attribute-value structures provided they do not contain conflicting values, and that that operation must be transitive and order-free. I called this the SAME predicate: It asserts that two structures have exactly the same attributes and values. This was the seed of the equality predicate of LFG and the unification operator in Martin's Functional Unification Grammar (Kay 1979, 1984) and other formalisms in that tradition.

My early ATN account of perceptual strategies relied on a carefully tuned, fixed ordering of arcs leaving each state in the grammar. Mehler and Carey (1967) conducted a really quite trivial experiment that suggested that this was not a good arrangement. Still thinking about the psychological reality of transformational grammar, they wanted to see whether listeners were sensitive to differences in surface-structure grouping, as in Example (7a,b), and differences in deep-structure relations (Example (7c,d)).

a. They [are forecasting] cyclones. (7)

b. They are [conflicting desires].

c. They are delightful to embrace. (= Someone embraces them.)

d. They are hesitant to travel. (= They travel.)

Without getting into the details of their primitive experiment, Mehler and Carey found that a sentence of one type (say Example (7a)) is easier to process after presenting ten sentences of the same type rather than priming with ten sentences of the contrasting type (Example (7b)). It seems that comprehension strategies can be influenced by very immediate, short-term syntactic experience, yet we would not want to say that the listener's knowledge of English is so unstable and changes so quickly. Rather, preference order should not be embedded in the grammar but encoded as an independent and perhaps rapidly fluctuating overlay. This is the separate agenda component of the modeling architecture I have sketched. As an aside, I think such short-term variability poses a conceptual challenge for psycholinguistic models that might be based on modern probabilistic and deep learning approaches, if preferences and structural constraints are tightly coupled or even inseparable.

My original account also depended on the default top–down depth-first search policy of the ATN parser. Back-tracking from a failed hypothesis could be very expensive, and fluctuating preferences might increase the likelihood that subsequent choices would be incorrect. It was well known that cubic-time performance could be achieved for any reasonable context-free parser simply by incorporating a memory for previously analyzed constituents and partial constituents. That is a very good engineering trade-off: quadratic space for exponential time. It seemed reasonable to think that the psychological system would also make such a trade-off, to mitigate the damaging effect of incorrect heuristic choices. I stripped down the ATN parser to its bare essentials and then combined it with Martin's chart data structure to represent the well-formed partial constituents. Martin and I later developed a simplified and more abstract implementation of this idea, which we later called an "active-chart parser." He describes this evolution also in his LTA acceptance paper (Kay 2006).

That was the end of my stint at Harvard. I have mentioned a few core ideas that emerged from these psycholinguistic and computational considerations: the compositional architecture for performance models, with separate knowledge, process, and strategy components; hierarchical attribute-value structures as an underlying representation for syntax; a single SAME operation for imposing constraints on attributes and values; and the integration of memory for previously recognized constituents. These ideas are the background for much of my later research.

3. Fundamental Issues

I met Danny Bobrow while working on the Lunar project. He was the vice president for Artificial Intelligence at BBN, and nominally Bill's boss. He left BBN to create a language understanding group at Xerox's new research center in Palo Alto. He invited me to join the group when it looked like I was about to finish my degree. Being a little wary of industry, I asked him how long we would have before Xerox expected us to impact their products. He replied that we would be one of their long-term investments, and that we had a 15-year horizon. I thought I could manage that. I suggested that he also invite Martin Kay, so that we could continue our productive collaboration. We arrived at the same time, in the fall of 1974. I expected to carry forward with psycholinguistic modeling at PARC, but I found that it was much easier to have an impact in linguistic theory and computational linguistics. Experimental psychology is hard.

As the language understanding project started up, we set to work on different components of an overall system. Danny and Terry Winograd worked on knowledge and reasoning, while Martin and I focused on syntax and morphology. This was all in service of developing a Conversational User Interface, in contrast to the graphical interface that others at PARC were exploring. We expected to assemble the separate components into a prototype system, after a few years of work. We eventually did lash the components together in, again, a Simon-esque way. A student, Henry Thompson, volunteered to deal with the non-negligible interactions of components so that we could protect the simplicity of our individual modules. The result was GUS, the early mixed-initiative frame-based dialog system described by Bobrow et al. (1977).

Martin and I had the luxury at PARC of being able to examine fundamental computational and linguistic issues over several years without otherwise showing much incremental progress. We were lucky that our work was not being peer-reviewed and that we did not have to apply for grants. Although ATNs had become a standard platform for many natural language applications, Martin and I argued at great length to identify and converge on the most primitive and abstract operations for parsing. And we had long, stimulating, and enjoyable debates about the nature of grammatical formalisms and syntactic representations. We agreed on attribute-value matrices as the basic encoding of underlying structure and something akin to the SAME predicate as the primitive descriptive device. We found further computational motivation for SAME: Its order-free property made it easy to configure an active-chart parser to implement bidirectional, island-driving parse strategies of the sort that were being proposed for some early approaches to speech recognition. But we had different and strongly held views about how to proceed from there.

Roughly, Martin was attracted to the idea that the grammar itself could be coded as elaborated feature structures, and that parsing consisted of applying implicitly a SAME-like operator—unification—to combine the grammar matrices recursively with attribute-value representations of the input words. The effect is to minimize the number of representations and entities, as Occam's razor would suggest.

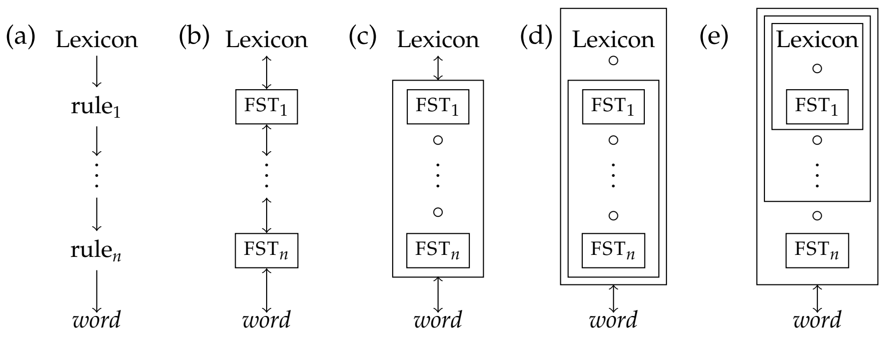


Figure 4
Compilation and composition of morphological rules.

Again taking Herb Simon as a guide, I maintained a separate grammar as a system of constraints in its own formalism, the use of explicit trees to represent the hierarchical traversal of surface constituents, and attribute-value matrices only as the underlying representation. I thought of the SAME predicate as just that, a predicate that candidate feature structures did or did not satisfy, and not an operation defined by a sequence of computations. I would say that I tended towards the model theory side of the logic while Martin tended towards the proof theory side. Martin developed his conception into what became Functional Unification Grammar. I will describe next how the formalism of Lexical-Functional Grammar derived from the approach that I was pursuing.

4. Finite State Morphology

First, on either approach we knew that the parser needed a front end for morphological analysis and dictionary look-up. Suffix-stripping routines for English and similar languages were commonplace, but we set ourselves a more ambitious goal: We wanted a universal system, one that would work not only for suffixing languages but also for languages that displayed the full range of morphological variation. We wanted an approach that would handle the complexities of Turkish and Finnish just as easily as English.

Linguistic treatments at that time were specified in the generation/production direction, as cascades of context-sensitive rewriting rules. They started from abstract lexical and morphological formatives and applied rules in sequence until an actual word emerged (see Figure 4 a). These rules were of the form

$$\alpha \rightarrow \beta / \lambda \_ \rho \tag{8}$$

indicating that a string in a regular language α can be rewritten to any string in β if it appears immediately after a left context in a regular language λ and before a right context in ρ. We needed an efficient algorithm for running individual rules backwards and then reversing the cascade sequence. This turned out to be a hard problem. The code for interpreting individual rules, let alone the entire cascade, looked to be much more complicated than the code needed for syntactic parsing. We knew that couldn't be right: Syntax is hard, morphology is easy. There had to be a trick.

We realized that the expressive power of such a morphological grammar is drastically reduced if an individual rule is not allowed to apply to its own output, either at a single level in a cascade or in a cyclic repeat of the cascade. In fact, we realized that the input–output mapping under that restriction of each individual rule could be computed exactly by a finite-state transducer (Figure 4b), and that a finite cascade of rules could be simulated by the composition of their equivalent transducers (Figure 4c).

Knowing that there exists a transducer equivalent to each rule was not the same as knowing how to construct such a transducer for a rule automatically. We struggled with that problem on and off for a number of years. For any rule, we could go to the white board, fiddle around with states and transitions, and eventually end up with what looked like the target transducer. But even a simple rule would end up with quite a few states and a large number of criss-crossing transitions, and we almost always would leave out a transition or two, or maybe even a state.

Standing back, we saw that the actual rewrite, the $\alpha \rightarrow \beta$ part, was easy to implement. The difficult part was to recognize all and only the occurrences of the context strings in λ and ρ, and to ensure that the rewrite only took place when surrounded by the proper contexts. This is what took several years of fooling around.

I had one of those "aha" moments, lying in bed on a Saturday morning. Our problem, abstractly, was to translate a universally quantified specification, a rule, into basically an existential device, a finite-state machine. We needed a machine that would identify and mark all substrings ending in a left context or beginning with a right context, not just some of them. Framed in that way, there was an obvious analogy to the simple and direct equivalence between universally and existentially quantified formulas in first order logic (Example (9a)), or to simple conditionals in the propositional calculus (Example (9b)).

a. $\forall x\, \phi(x) \text{ iff } \neg\, \exists x\, \neg\phi(x)$ (9)

b. $P \rightarrow Q \text{ iff } \neg\, (P \;\&\; \neg Q)$

Double negation is the key in both of these logical identities, and these relationships carry over by analogy to properties of strings and substrings. Because the regular languages are closed under complementation and the contexts in rewriting rules are regular, expression (10a) defines a regular language that includes a string only if every prefix ending in λ is followed immediately by a bracket $<$ unmistakably marking a left-context occurrence. Conversely, for the strings in the language defined by Example (10b), every left-context-marking bracket is immediately preceded by a substring in λ.

a. $\overline{\Sigma^*\lambda\, \overline{<\Sigma^*}}$ (10)

b. $\overline{\overline{\Sigma^*\lambda}\, <\Sigma^*}$

Regular languages are closed under intersection, and the intersection of these two languages contains all and only strings where left-contexts are properly marked. And of course there is a finite-state machine that accepts exactly that intersection language. With this formulation we were able easily to implement an automatic compiler for rewriting rules of arbitrary complexity, and prove that the resulting transducers had the desired input/output properties. This high-level insight enabled us also to construct a compiler for Koskenniemi's (1983) two-level formalism for morphology (Karttunen, Koskenniemi, and Kaplan 1987).

We solved the rule translation and composition problems in the early 1980s. At the same time we realized that the lexicon could also be treated as a regular relation and could be composed at the top of the rule cascade (Figure 4d). There was then no need for separate dictionary lookup code. This worked well for most of the languages we investigated, with simple suffixing or prefixing morphology. Unsurprisingly, Finnish turned out to be a computational challenge. It was impractical, in space and in time, to combine the productive rules of Finnish morphology, whether presented as a rewriting cascade or as a collection of two-level constraints.

Lauri joined us at PARC and later made a crucial observation. The problem becomes tractable if the inventory of lexical items is introduced as a constraint early in the construction process rather than as the very last step. The order of combination can be inverted (Figure 4e) because of the associativity of composition (and intersection), and the result is a lexicalized morphological transducer typically with orders of magnitude fewer states and arcs than the lexical and morphological transducers taken separately (Karttunen, Kaplan, and Zaenen 1992). The end result, even for Finnish, is an efficient one-step reversible mapping from surface strings to lexemes and morphological features. From a psycholinguistic perspective, this exemplifies an alternative way of relating competence, the morphological generalizations expressed in a succinct collection of rules, to performance models embodied in equivalent transducers: compilation instead of interpretation. Dual spaces with exactly the same input/output mappings but with completely different principles of optimization, different strategies for run-time execution, and different profiles of computational resources.

We languished in our efforts to write up these results, because of other distracting targets of opportunity. As it turned out, this technology was not only useful as a parsing component, it had many stand-alone applications. We carefully engineered the finite-state algorithms and together with John Maxwell we developed exceedingly compact ways of encoding large transducers. In 1985 we worked with a spin-off company, Microlytics, to deploy the technology in after-market spell checkers for the installed base of Xerox electronic typewriters, to meet a competitive threat from other manufacturers. With respect to Danny Bobrow's description of the 15-year product horizon, I like to point out that we came in about 5 years ahead of schedule. Together with Microlytics, we subsequently created a family of hand-held spell-checkers, conjugators, thesauruses, and even electronic books, all based on this technology, as illustrated in Figure 5. Microlytics also embedded the technology in multiple-language spell-checkers that were licensed to most of the mainstream word-processors of that era. Another PARC spin-out, Inxight Software, licensed morphological analyzers for indexing and retrieval in search engine applications. Eventually, in 1994, Martin and I got around to publishing a description of our theoretical results (Kaplan and Kay 1994). We are pleased that it was recognized as a finalist for the 25-year test-of-time award.

5. Lexical-Functional Grammar

Finite-state morphology was a productive and valuable detour from the central concerns of syntax. A year after I moved to PARC, I went back to Cambridge to participate in a series of cross-disciplinary workshops that George Miller and Morris Halle were organizing at MIT. They had received a large grant from AT&T to stage a symposium to celebrate the 100th anniversary of the invention of the telephone. They invited me and my Harvard psycholinguistic colleagues, other psychologists, philosophers, and computer scientists, and a number of traditional linguists. That's where Joan Bresnan and I met for the first time.

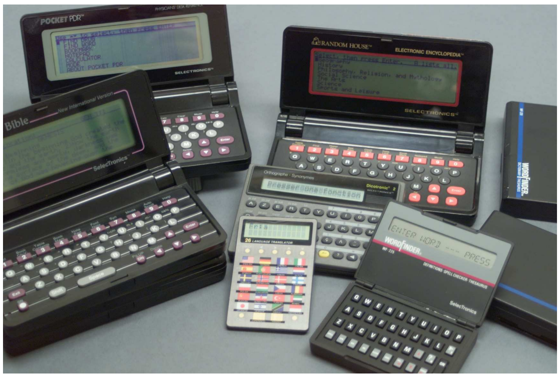


Figure 5
A sample of hand-held products created with PARC finite-state technology, circa 1990.

The workshop met once a month over the course of a year, with each group having a day or so to present their work and to lead a discussion. Eric Wanner, Mike Maratsos, and I described our psychologically motivated alternative to transformational grammar in one session. In another session Joan introduced her conception of lexical redundancy rules as a better way of describing many syntactic phenomena within a transformation framework (Bresnan 1978). We chatted in the breaks and at breakfast in the hotel, and gradually came to see some common interests and develop some mutual respect. She saw that psycholinguistic considerations and perhaps even the non-derivational organization of ATN models could serve as possible supports for her redundancy approach. I realized that lexical redundancy rules could solve some residual descriptive and explanatory problems that surfaced in ATN grammars after register operations had been reduced to SAME.

We met again in Italy, in the summer of 1977. Antonio Zampolli organized the IV International Summer School in Computational Linguistics in Pisa, and Joan, Lauri, Martin, and I, along with quite a few others, were invited to teach. My course was called Computational Psycholinguistics, and I covered the psycholinguistic modeling approach within the simplified and reduced formal framework that I was evolving from the early ATN. Joan's course was on realistic transformational grammar. I sat in her course and she sat in mine, and we went out drinking together at the end of the day. Our interactions grew intense, as we saw our work connect in more and more ways. If any of you were there, 40 years ago, you may remember how upset Antonio was when Joan and I got deep into a theoretical discussion when we should have been looking out at the countryside from the rooftops at San Gimignano. But that's when Joan and I decided to team up.

I arranged for her to visit PARC several times during the following year, and she arranged for me to have a mini-sabbatical at MIT in the fall of 1978. We decided to combine forces and together teach a course called Computational Psycholinguistics. We agreed that we would teach psychology, computation, and linguistics from a single formalism no matter what. We started with the formalism I had developed at PARC: surface-structure traversals producing feature structure representations by means of a SAME predicate. The challenge was to demonstrate convincingly that linguistic generalizations could be captured in that formalism and those representations. Week by week

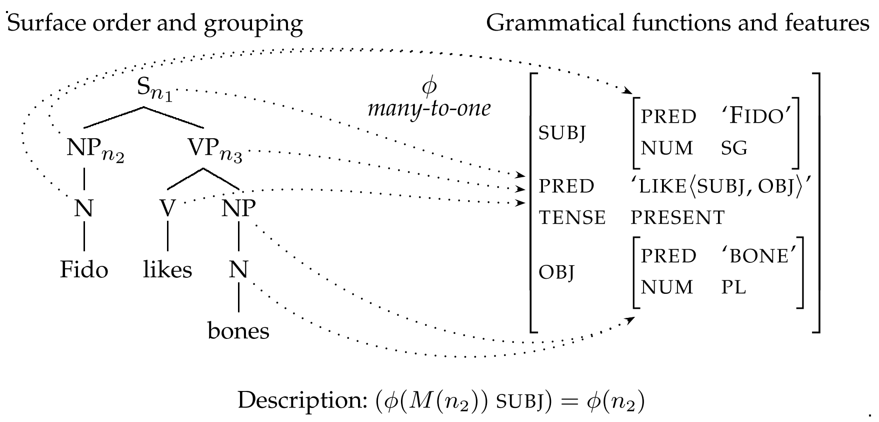


Figure 6
The description and correspondence architecture of Lexical-Functional Grammar.

we had to make adjustments. In some cases we had to sneak in adjustments to the adjustments we had announced just the week before.

The pressure from linguistics helped to clarify the fundamental notions that lay at the core of the syntactic architecture. I developed a completely declarative interpretation to replace the lingering procedural aspects of the formal set up. The surface constituent structure (c-structure) and the deep functional structure (f-structure) in Figure 6 are representations of formally different types, again in a Simon-esque way, and together they are responsible for characterizing the syntactic aspects of the sound-to-meaning mapping. But both are seen as models, in the mathematical sense, for type-specific collections of constraints. Thus, grammatical rules (11a) and the lexicon provide conventional context-free admissibility conditions that mother/daughter/sister nodes of the surface tree must satisfy. The annotations on the rules are instantiated at the c-structure nodes to produce a description that an associated f-structure must satisfy, given the assumption of a piece-wise correspondence function ϕ that projects from nodes of the tree to units of the attribute-value matrix.

$$\text{a.}\quad S \rightarrow \underset{(\uparrow \text{SUBJ}) = \downarrow}{NP} \quad \underset{\uparrow = \downarrow}{VP} \tag{11}$$

$$\text{b.}\quad (\phi(n_1)\ \text{SUBJ}) = \phi(n_2) \wedge \phi(n_1) = \phi(n_3)$$

The meta-variable ↓ on a category denotes the f-structure unit corresponding to the matching node (e.g., ↓ on the NP category instantiates to $\phi(n_2)$ in this case) and ↑ refers to $\phi(n_1)$, the f-structure corresponding to the mother S node. Any f-structure projected from this tree must be a minimal model for the description composed of the equations in (11b) and the constraints instantiated from the rules admitting all other nodes. The attribute-value matrix in Figure 6 meets this condition. The power to characterize non-context-free dependencies comes from the fact that the historical SAME predicate is respecified here as ordinary mathematical equality, and the c-structure to f-structure correspondence ϕ is many-to-one.

This briefly summarizes the formal foundation of Lexical-Functional Grammar as it emerged from that course (Kaplan and Bresnan 1982), and this picture of the underlying grammatical architecture is still accurate after 40 years. This conception of formal descriptions derived from structural correspondences and projection functions was extended and refined in subsequent research (Kaplan 1987, 1989), and it now stands as a general framework for characterizing the interactions between syntax and other relatively independent modules of linguistic analysis. Halvorsen and Kaplan (1988) and Halvorsen (1988) make use of a correspondence between f-structure and semantic structure to create descriptions of logical formulas, and a projection function maps f-structure units to linear–logic meaning constructors in the "glue" approach to semantic interpretation (Dalrymple 1999). As another application of this architecture, Dalrymple and Mycock (2011) have formalized the three-way interaction of syntax, semantics, and prosody by means of a particular configuration of structural correspondences. Descriptions and correspondences are cornerstones of the LFG formalism and distinguish it from Montague Grammar, HSPG, Categorial Grammar, and other grammatical systems.

6. High-Speed Parsing and Large-Scale Grammars: XLE and Pargram

In this completely declarative, constraint-based formulation, grammatical specifications were even less connected to a particular implementation. That created the need and opportunity to investigate new strategies for parsing. This is the problem that I started to work on after I returned to PARC and that I continued in close collaboration with John Maxwell. Equality was originally implemented as an attribute-value unification algorithm, and a standard context-free chart parser was used to interpret the surface-structure grammar. Those algorithms were combined in the obvious way, by using unification to prune the chart-parser constituents in a bottom–up fashion. This was effective, but not efficient, even though unification and chart parsing are both fast.

It was easy to identify the potential sources of exponential behavior. The formalism encoded certain kinds of syntactic ambiguity by allowing quickly solvable equations to appear under disjunction and negation, in Boolean combinations. And those combinations emerge from the enumeration of all subtrees determined by a context-free grammar that could be exponentially ambiguous. We first attacked the Boolean satisfiability problem. A straightforward Boolean solver might convert to disjunctive normal form so that all inconsistencies, even from distant parts of a sentence, can be easily recognized. But this is a bad strategy if distant equations are unlikely to interact, as we thought should be the case for natural language. We arrived at a method that optimizes for the independence of disjunctions in distant equations rather than their inconsistency. After several years of hard work, we reduced the method to a very simple lemma (Example (12)) with a trivial two-line proof (Maxwell and Kaplan 1991).

$$\alpha \vee \beta \text{ iff } p \rightarrow \alpha \wedge \neg p \rightarrow \beta, \text{ for } p \text{ a new Boolean variable.} \tag{12}$$

This is a variant of conjunctive normal form for which there is a sound and complete satisfiability procedure. The complexity of that procedure depends only on the size of subsets of inconsistent equations. When distant equations are in fact relatively independent, the effect for a Boolean combination of size n is to reduce the complexity from 2^n to roughly $n2^k$ for $k \ll n$.

This method applies not just to speed up syntactic computations. It also allows for efficient ambiguity management when independent components of different formal types are composed in a nearly decomposable architecture. It is not necessary to unpack and enumerate representations of alternative structures when they are passed across a component boundary. Crouch and King (2006) and Bobrow et al. (2007) take advantage of this to preserve compact encodings of ambiguity as they translate first from syntax to semantics and then to knowledge representations that support inferences of entailment and contradiction. Systematic ambiguity management also reduces the incentive for premature attempts at ambiguity resolution based on soft probabilities within one component before the hard constraints of other components or of nonlinguistic context can be taken into account.

John and I also investigated strategies for intermixing equation solving with the ambiguities of chart-parser constituent formation (Maxwell and Kaplan 1993). We eventually arrived at a technique that optimizes in a different way for the intuition that natural languages are mostly, though not always, context-free (Maxwell and Kaplan 1996). The result was a high-speed, largely polynomial, commercial-grade parser and generator for interpreting broad-coverage LFG grammars, and composing them with finite-state morphologies (Crouch et al. 2008). We called it the Xerox Linguistic Environment—XLE. I am currently working with Jürgen Wedekind on a mathematical analysis of the LFG formalism that might provide a formal explanation, retrospectively, for the effectiveness of the heuristic techniques in XLE. Previously we investigated the mathematical and algorithmic properties of the generation problem for LFG and similar formalisms (Wedekind and Kaplan 2012).

Together with colleagues initially at the University of Stuttgart and the Xerox research center in Grenoble, we created the Parallel Grammar project, Pargram. Its goal was to build, in XLE, large-scale LFG grammars for a variety of languages and to assign parallel f-structures to related syntactic constructions (Butt et al. 1999; 2002). Miriam Butt, Mary Dalrymple, Helge Dyvik, Tracy King, Hiroshi Masuichi, Christian Rohrer, and Annie Zaenen are among the many people in many institutions who worked to produce those commercial-grade grammars and otherwise contributed to the Pargram effort. I might also mention that Chris Manning worked with us as a student and wrote one of the early versions of the English grammar. And an important later addition to XLE's techniques for ambiguity management and robustness was an efficient module for probabilistic ranking of alternatives, contributed by Stefan Riezler and Mark Johnson (Riezler et al. 2002). XLE and the English Pargram grammar were the key language technologies for the semantic search engine developed years later at Powerset, a PARC spin-out subsequently acquired by Microsoft.

7. Conclusion

I have been fortunate in my career to work on intriguing problems in psychology, linguistics, and computation, with some of the very best collaborators. In the mid-1980s I formed the Natural Language Theory and Technology group at PARC. The NLTT group included Martin, Lauri, and Joan—previous LTA recipients—and many of the other people I have mentioned. The name I chose for the group turned out to be very prophetic: I have recited here just some of the contributions to both theory and technology that group members were able to make over the 25 years of its life. That is quite a long time in research-group years.

I left PARC for Powerset when our language technology seemed ready to scale for practical applications, and we calculated that the high margins of search advertising made the extra cost of semantic search economically attractive. I have suggested more recently, as a technical leader at Nuance and Amazon, that sophisticated natural language technologies are crucially needed to improve the fairly poor performance of the conversational systems that ordinary people are now encountering in their everyday lives. The rapid advances in machine learning, particularly the varieties of deep learning models coupled with the massive amounts of data that can now be collected, are obviously having a major positive impact. But to be truly effective, such systems must bring together information of many different types and from many different sources, and much of it cannot be observed directly in end-to-end stimulus/response interactions. Learning by observation, even when possible, may not be the most complete, effective, or efficient way of acquiring the necessary understanding of language, human interaction styles, specific application domains, or the world at large. Some influences on conversational interaction are more accessible, or perhaps only accessible, from the knowledge and wisdom that experts have already accumulated by introspection, scientific analysis, and reasoning, and from curated repositories that may already be available.

This returns to one of my persistent themes. Just as for cognitive models of human language performance, we should assume that conversational systems, and other highly capable language-based applications, are nearly decomposable and that their modules are of widely varying pedigrees. Going forward, I think the outstanding scientific and practical challenge will be to characterize and implement the non-negligible interactions of these components. It will take every arrow in our quiver to construct conversational systems that are helpful, informative, and pleasant to interact with.

I want to close by again thanking the selection committees and the Association for this tremendous honor.

References

Bobrow, Daniel G., Bob Cheslow, Cleo Condoravdi, Lauri Karttunen, Tracy Holloway King, Rowan Nairn, Valeria de Paiva, Charlotte Price, and Annie Zaenen. 2007. PARC's bridge and question answering system. *Proceedings of the GEAF07 Workshop*, pages 46–66, Stanford University.

Bobrow, Daniel G., Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson, and Terry Winograd. 1977. GUS, a frame-driven dialog system. *Artificial Intelligence*, 8:155–173.

Bresnan, Joan. 1978. A realistic transformational grammar. In Morris Halle, Joan Bresnan, and George Miller, editors, *Linguistic Theory and Psychological Reality*, MIT Press, Cambridge, MA, pages 1–59.

Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar project. In *Proceedings of COLING 2002: Workshop on Grammar Engineering and Evaluation*, pages 1–7, Stroudsburg, PA.

Butt, Miriam, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications, Stanford University.

Chomsky, Noam. 1957. *Syntactic Structures*. Mouton & Co., The Hague.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.

Crouch, Dick. 2005. Packed rewriting for mapping semantics to KR. In *Proceedings of the Sixth International Workshop on Computational Semantics*, Tilburg.

Crouch, Dick, Mary Dalrymple, Ronald M. Kaplan, Tracy Holloway King, John T. Maxwell, III, and Paula S. Newman. 2008. *XLE Documentation*. Palo Alto Research Center, Palo Alto, CA. Available at `http://ling.uni-konstanz.de/pages/xle/doc/xle_toc.html`.

Crouch, Dick and Tracy Holloway King. 2006. Semantics via f-structure rewriting. In *Proceedings of the LFG'06 Conference*, pages 145–165, University of Konstanz.

Dalrymple, Mary, editor. 1999. *Semantics and Syntax in Lexical-Functional Grammar*. MIT Press, Cambridge, MA.

Dalrymple, Mary and Louise Mycock. 2011. The prosody–semantics interface. In *Proceedings of the LFG'11 Conference*, pages 173–193, University of Hong Kong.

Halvorsen, Per Kristian. 1988. Projections and semantic description in Lexical-Functional Grammar. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 471–478, Tokyo.

Halvorsen, Per Kristian and Ronald M. Kaplan. 1988. Projections and semantic description in Lexical-Functional Grammar. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1116–1122, Tokyo.

Hays, David G. 1964. Dependency theory: A formalism and some observations. *Language*, 40:511–525.

Kaplan, Ronald M. 1972. Augmented transition networks as psychological models of sentence comprehension. *Artificial Intelligence*, 3:77–100.

Kaplan, Ronald M. 1974. Transient Processing Load in Relative Clauses. Ph.D. thesis, Harvard University.

Kaplan, Ronald M. 1987. Three seductions of computational psycholinguistics. In Peter Whitelock, Mary McGee Wood, Harold L. Somers, Rod Johnson, and Paul Bennett, editors, *Linguistic Theory and Computer Applications*, Academic Press, London, pages 149–188.

Kaplan, Ronald M. 1989. The formal architecture of Lexical-Functional Grammar. *Journal of Information Science and Engineering*, 5:305–322.

Kaplan, Ronald M. and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, pages 173–281.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Karttunen, Lauri, Ronald M. Kaplan, and Annie Zaenen. 1992. Two level morphology with composition. In *Proceedings of COLING 92*, volume I, pages 141–148, Nantes.

Karttunen, Lauri, Kimmo Koskenniemi, and Ronald M. Kaplan. 1987. A compiler for two-level phonological rules. In *Tools for Morphological Analysis*, Report No. CSLI-87-108. Center for the Study of Language and Information, Stanford University.

Kay, Martin. 1967. Experiments with a powerful parser. In *Proceedings of the Second International Conference sur le Traitement Automatique des Langues*, pages 1–20, Grenoble, France.

Kay, Martin. 1979. Functional grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society*, pages 142–158, Berkeley, CA.

Kay, Martin. 1984. Functional unification grammar: A formalism for machine translation. In *Proceedings of COLING84*, pages 75–78, Stanford University.

Kay, Martin. 2006. A life of language. *Computational Linguistics*, 31(4):425–438.

Koskenniemi, Kimmo. 1983. Two-level morphology: A general computational model for word-form recognition and production, Technical Report 11, Department of General Linguistics, University of Helsinki.

Maxwell, John T., III and Ronald M. Kaplan. 1991. A method for disjunctive constraint satisfaction. In Masaru Tomita, editor, *Current Issues in Parsing Technology*. Kluwer Academic Publishers, Boston, pages 173–190.

Maxwell, John T., III and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19:571–589.

Maxwell, John T., III and Ronald M. Kaplan. 1996. Unification parsers that automatically take advantage of context freeness. In *Proceedings of the LFG'96 Conference*, Stanford University.

Mehler, Jacques and Peter Carey. 1967. Role of surface and base structure in the perception of sentences. *Journal of Verbal Learning and Verbal Behavior*, 6(3):335–338.

Miller, George A. and Kathryn O. McKean. 1964. A chronometric study of some relations between sentences. *Quarterly Journal of Experimental Psychology*, 16(4):297–308.

Riezler, Stefan, Tracy Holloway King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278, Philadelphia, PA.

Simon, Herbert A. 1962. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482.

Slobin, Dan I. 1968. Recall of full and truncated passive sentences in connected discourse. *Journal of Verbal Learning and Verbal Behavior*, 7(5):876–881.

Wanner, Eric and Michael Maratsos. 1978. An ATN approach to comprehension. In Morris Halle, Joan Bresnan, and George Miller, editors, *Linguistic Theory and Psychological Reality*, MIT Press, Cambridge, MA, pages 119–161.

Wedekind, Jürgen and Ronald M. Kaplan. 2012. LFG generation by grammar specialization. *Computational Linguistics*, 38(4):867–915.

Woods, William A. 1970. Transition network grammars for natural language analysis.

Communications of the ACM, 13(10):591–606.

Woods, William A. 2006. The right tools: Reflections on computation and language. *Computational Linguistics*, 36(4):601–630.

Woods, William A. and Ronald M. Kaplan. 1971. The lunar sciences natural language information system. Technical Report 2378, Bolt Beranek and Newman, Inc., Cambridge, MA.

Woods, William A., Ronald M. Kaplan, and Bonnie Nash-Webber. 1972. The lunar sciences natural language information system: Final report. Technical Report 2378, Bolt Beranek and Newman, Inc., Cambridge, MA.