Search Challenges in Natural Language Generation with Complex Optimization Objectives

Vera Demberg and **Jörg Hoffmann** and **David M. Howcroft**
Dietrich Klakow and **Álvaro Torralba**
Saarland University
Saarbrücken, Germany
{vera, howcroft}@coli.uni-saarland.de, {torralba, hoffmann}@cs.uni-saarland.de, dietrich.klakow@lsv.uni-saarland.de

Abstract

Automatic natural language generation (NLG) is a difficult problem already when merely trying to come up with natural-sounding utterances. Ubiquituous applications, in particular companion technologies, pose the additional challenge of flexible adaptation to a user or a situation. This requires optimizing complex objectives such as information density, in combinatorial search spaces described using declarative input languages. We believe that AI search and planning is a natural match for these problems, and could substantially contribute to solving them effectively. We illustrate this using a concrete example NLG framework, give a summary of the relevant optimization objectives, and provide an initial list of research challenges.

Introduction

As mobile devices are getting more and more ubiquitous, and speech recognition and synthesis have seen large performance improvements in recent years, NLG is necessary in an increasingly large number of applications and situations. Highly domain-specific template-based NLG approaches are becoming less viable, due to a lack of ability to adapt the generated utterances flexibly to a user or a situation, as would be especially important in companion technologies. For instance, a dialog system should generate utterances that are easier to comprehend and more redundant when a user is concentrating on another task (such as driving a car), but should generate concise utterances (which are often more complex) when the user can fully concentrate on the interaction with the dialog system (Demberg and Sayeed 2011).

Achieving flexible situation-adaptive NLG is a major challenge to, and an active research area in, Computational Linguistics, requiring the identification of suitable objectives and measures for controlling utterance complexity. It is also a major challenge to the design of search algorithms, for optimizing (combinations of) such objectives. We believe that the AI search community could make major contributions towards the latter. AI planning in particular is relevant given its focus on automation and powerful declarative models. Our mission in this paper is to provide a concise summary of the problem and the challenges ahead, in terminology accessible to the AI community, as a first step towards bringing the two communities together in this endeavor.

NLG traditionally proceeds in a pipeline comprising three phases: *document planning*, *microplanning*, and *surface realization*. During document planning, the system decides on the content to be conveyed, and what rhetorical structure connects said content. Microplanning then lexicalizes this content, choosing which words should be used to express it, and performs aggregation and referring expression generation. This results in a syntactico-semantic representation, forming the input to the surface realization component which generates the final natural language utterances from that input.

The boundary between microplanning and surface realization is fluid, varying the granularity of the microplanning output and, accordingly, the degrees of freedom assigned to the surface realization grammar. The surface realization process can include: lexical choice (e.g., restaurant vs. bar); structural choice (e.g., active vs. passive voice: "restaurant serves food" or "food is served by restaurant"); and choosing among adjective, prepositional phrase, or relative clause options as in "Almaz is an Eritrean restaurant", "Almaz is a restaurant with Eritrean food", or "Almaz, which is an Eritrean restaurant, . . . ".

The desired situation-adaptivity in NLG is a function not of *what* to say, but *how* to say it, and as such is naturally associated with the microplanning and surface realization phases. We here focus on the surface realization problem, the understanding being that, in applications, the surface realization grammar (and therewith the search) will be given sufficient freedom to encompass the relevant formulation differences.

Traditional optimization objectives for surface realization are to generate grammatically correct, natural-sounding sentences. Effectively optimizing these objectives is, already, not a solved problem, and could potentially benefit from AI search algorithms expertise. This is even more true for the complex optimization objectives required to achieve intelligent behavior in companion technologies and other ubiquitous applications. In what follows, to make matters concrete, we first consider a particular search-based surface realization framework, OpenCCG, overviewing its search algorithm in AI terms and in relation to AI search algorithms. We then summarize current research issus in the design of more complex NLG objectives & measures, towards the desired flexibility. We conclude the paper with a discussion

$$\begin{tikzcd} \textbf{[Eritrean]} \arrow[rd] & & \langle\textit{restaurant}\rangle & & \textbf{[good\_food]} \arrow[ld] \\ & \textbf{[be]} & \textbf{[Almaz]} \arrow[l] \arrow[r] & \textbf{[have]} & \end{tikzcd}$$

Figure 1: Example input for the realization algorithm representing the propositions be(Almaz, Eritrean) and have(Almaz, good_food).

of challenges to search algorithms, as well as possible approaches to address these.

A Concrete Example: OpenCCG

OpenCCG is a prominent state-of-the-art method for surface realization via search (White 2006, White and Rajkumar 2012). As our purpose is to illustrate basic aspects of the search, we do not provide a comprehensive summary and present a simplified version only. OpenCCG is based on *combinatory categorial grammars (CCG)*. It uses a so-called *chart realization* algorithm (e. g. (Kay 1996, Cahill and van Genabith 2006, Carroll and Oepen 2005)). Chart realization is a dynamic programming approach to language generation that performs a best-first search in the space of (partial) sentences, storing partial results in a *chart* table, and generating new search nodes by combining expanded nodes with entries from that table.

The input to surface realization is a labeled directed graph, representing the so-called *semantics*, i. e., the content we wish to convey: objects, properties, activity, and how they are connected. The target is to find a valid sentence that contains all the semantics in the input. Figure 1 illustrates an example input. Valid sentences containing this semantics are, e. g.:

(a) Almaz has good food and is an Eritrean restaurant.

(b) The Eritrean restaurant Almaz has good food.

Search nodes in OpenCCG consist of a string, i. e. the partial sentence contained in the node, as well as a *grammar category* that determines how the node can be combined with other nodes. A category may be a grammar item (S stands for sentence, N for noun, NP for noun phrase, and so on), or a function that composes several atomic expressions with *forward concatenation* (/) or *backward concatenation* (\). For example, NP/N means that, if concatenated with another item of type N, we get an item of type NP; and NP concatenated with $S\backslash NP$ yields an item of type S.

We denote nodes as $\alpha(X)$, where α is the string and X the category. Nodes can be transformed and combined using different types of *rules*. Two strings can be concatenated via *application*, forward $[\alpha(X/Y)\ \beta(Y) \rightarrow \alpha\beta(X)]$ or backward $[\alpha(Y)\ \beta(X\backslash Y) \rightarrow \alpha\beta(X)]$. For example, we may concatenate "Eritrean" (NP/N) and "restaurant" (N) to obtain "Eritrean restaurant" (NP). Two strings can also be concatenated via *composition*, forward $[\alpha(X/Y)\ \beta(Y/Z) \rightarrow \alpha\beta(X/Z)]$ or backward $[\alpha(Y\backslash Z)\ \beta(X\backslash Y) \rightarrow \alpha\beta(X\backslash Z)]$. Additionally, there are unary rules changing the grammar type of a string in order to enable new combinations, e. g. "restaurant" (N) $\rightarrow$ "restaurant" $NP\backslash(NP/N)$. Importantly, rules combining two strings only ever allow to concatenate these, in either order – i. e., we can *not* insert another string later on in between the two. This is a design decision intended to keep the branching factor feasible.

In addition to the string and category $\alpha(X)$, search nodes are associated with information regarding how much of the input semantics is being conveyed, i. e., which parts of the input graph are *covered*. This is simply a bitvector that, for every element in the input, maintains the information whether or not a word covering that element has already been added to α.

The target of the search is to find a complete sentence, i. e. a node of category S covering the entire input semantics, maximizing the score with respect to the desired optimization metric (discussed further below).

To initialize the search space, a pre-process performs a lookup in a *dictionary*: a collection of all words – *lexical items* – that may be used, each associated with its grammar category and with the semantics it can cover. The pre-process retrieves all lexical items relevant to the input semantics at hand. For example, the semantics ⟨*restaurant*⟩ may be covered by the lexical items "restaurant"(N), "bistro"(N), and "café" (NP/N); the semantics ⟨*win*⟩ may be covered by different variants of that verb, differing with respect to tense, as well as the number of objects to be associated with the verb (intransitive, transitive, ditransitive verb). All these lexical items are inserted into (what the AI search community would refer to as) the open list, and search begins.

In the search, the chart serves as a dynamic programming cache. It stores the nodes that have already been expanded. Whenever a new node is expanded, successors are generated by *combining the node with every compatible node in the chart*. Two nodes are compatible if their semantics coverages are disjoint (we must not cover the same input element with more than one lexical entry), and the grammar categories can be concatenated by application or composition. Additional successors are generated as the result of applying unary rules, and adding connective words such as "that", "to", etc.

Duplicate elimination prunes nodes with the same semantics and category, even if their strings differ. However, in order to increase diversity, not all duplicates are pruned. Instead, the chart is divided into equivalence classes of same semantics and category, and the K-best nodes in each equivalence class are preserved, where "best" is according to optimization criterion score (see below), and K is a parameter that controls the trade-off between search efficiency and quality of the result.

The search is best-first, ordered by a *scoring function*. Contrary to heuristic search methods in AI, current OpenCCG scoring functions do not attempt to estimate "goal distance" (the number of steps until a complete sentence), nor the quality of the best completion of the partial solution at hand. The scoring functions do not attempt to predict the future at all, instead computing the optimization objective score solely on the content of the search node itself. In this sense, the use of scoring functions is akin to the use of evaluation functions in local search optimization methods, despite the fact that search nodes are not feasible solutions (i. e., do not correspond to grammatically valid sentences). This is a simple and feasible solution, but may obviously be detrimental to the search. One research challenge is to find remedies based on the methods devised for the generation of heuristic functions in AI.

A common optimization objective – a means of measuring "how natural-sounding" a sentence is – is based on *n-grams*, measures of how common particular word tuples (e. g. triples, *trigrams*) are in natural language. For each word tuple, we get a probability measure for this word tuple occuring in a natural language text. The score of a partial sentence α is the sum of the negative logarithms of these probabilities, over all word tuples contained in α (we get back to this in the next section). Note that, applied to partial sentences during search, this creates a strong bias towards longer strings, simply because these contain more word tuples. Note furthermore the crucial difference to optimization criteria commonly considered in AI search problems: these are typically described declaratively as a function of the solution structure, and satisfy particular decomposition properties such as additivity. In contrast, n-grams are defined outside the input grammar, and behave in irregular ways gleaned from natural language text bodies. This gap between search model and optimization objective is intended and necessary, as the objective – a "natural-sounding" sentence – is difficult to capture in terms of a formal sentence-generation model.

The algorithm has an anytime flavor: When a valid solution, i. e. a complete sentence, is found, we continue looking for other sentences of better quality; the search stops only when the state space is exhausted, or a time limit is reached. Then, all complete sentences are extracted from the chart, and are evaluated by a refined quality objective for the final ranking, such as a neural network trained with more features than the n-grams used during the search (Nakatsu and White 2006, White and Rajkumar 2009, Rajkumar and White 2010, White and Rajkumar 2012, Rajkumar and White 2014).

One characteristic of OpenCCG search spaces, and a huge source of performance difficulties, are *dead ends*, i. e., search nodes that cannot be completed into valid sentences. There are at least two major sources of dead-ends: (a) wrong grammar categories that cannot anymore be completed into a sentence conveying the desired semantics; (b) applying combination rules in the wrong order. The former arises, e. g., when selecting an intransitive verb, with a category allowing no direct object, for a sentence that requires such an object. An example for the latter is the partial sentence "this restaurant has", which is a dead-end in our example as we cannot concatenate it with anything expressing that the restaurant is Eritrean – recall that we cannot insert new strings *within* the partial sentence.

Difficulty (a) is inherent to surface realization, and poses an interesting challenge to dead-end detection (we get back to this later). Difficulty (b) is more harmless, in the sense that it is an artifact of the way the combination rules are designed. *Chunking* has been designed as an optimization to counter-act this artifact (amongst other things) (White 2004, White 2006). It identifies sub-sentences in a preprocessing stage, and forbids combining nodes that refer to different sub-sentences until the semantics within each sub-sentence have been fully covered. For example, chunking avoids combining "restaurant" with "has" until it has been combined with "Eritrean".

New Complex Optimization Objectives in Natural Language Generation

Traditionally, natural language generation systems have mostly focussed on generating text for some fixed quality objective (e.g., grammaticality). More recently however, there is an increased interest in more flexible types of generation targets. In particular, how to automatically generate utterances with an optimal trade-off between complexity and conciseness for a specific user in a given situation? An example use case would be an in-car spoken dialog system. In this kind of setting, what is "optimal" changes based on the driving situation. Passengers adapt to the difficulty of driving conditions, speaking less overall, using less complex utterances, and speaking more about traffic when drivers face a challenging task (Crundall et al. 2005, Drews, Pasupathi, and Strayer 2008). Remote conversation partners, e. g. on a cell phone, do not adapt to the driver's cognitive load (as they can't directly observe it), and are therefore more likely to exceed the driver's channel capacity, increasing the likelihood of an accident. Balancing communicative efficiency against, e.g., safety concerns therefore requires the development of adaptive natural language generation systems which can target different levels of information density in different contexts.

In psycholinguistics, (Hale 2001) has suggested the information-theoretic notion of *surprisal* for quantifying the processing difficulty caused by a word. Surprisal is defined as the Shannon Information (Shannon 1948), i. e., the negative log probability of a word in context (Equation 1), where context is usually operationalized as the preceding sequence of words (Equation 2):

$$\begin{aligned} surprisal(w_n) &= -logP(w_n|\text{CONTEXT}) \quad (1)\\ &= -logP(w_n|w_1 w_2 ... w_{n-1}) \quad (2)\end{aligned}$$

Here, w_i denotes the i^{th} word in the sentence. In this formulation, a word carries more *information*, and is more difficult to process, when it is less predictable based on the preceding words. Likewise, a word that is totally predictable carries no new information, and is easy to process. This formulation of surprisal has been shown to correlate with reading times (Levy 2008, Demberg and Keller 2008) and the N400 component of electroencephalographic (EEG) event-related potentials (ERPs) (Frank et al. 2015), suggesting that surprisal is a valid measure of comprehension difficulty.

On the production side, the task of the producer is to distribute information across the utterance. Human speakers appear to be sensitive to information density, altering their use of optional linguistic markers in a way that avoids large peaks in surprisal (Levy and Jaeger 2007). The *uniform information density* hypothesis (UID; (Jaeger 2006, Jaeger

2010)) observes that rational speakers should want to communicate as much information as possible without overwhelming their listener, leading speakers to use a relatively uniform information distribution near the *channel capacity*, the maximum amount of information comprehensible to their listener. Initial evidence suggests that NLG systems sensitive to information density produce outputs more highly rated by humans (Rajkumar and White 2011, Dethlefs et al. 2012).

For example, consider the case where we want to convey the message shown in Figure 1; the two alternative verbalizations shown in (a) and (b) differ in information density[1]: (a) has an average surprisal of 7.15 bits per word while (b) has 7.84 bits per word, meaning that (b) is more informationally dense than (a). This suggests that (a) should be preferred over (b) in situations where the user cannot give their full attention to the linguistic task. Note however, that utterance (b) is the one that conveys information more uniformly, with an average change in surprisal from word to word of 2.44 bits, versus 3.31 bits in (a).

More detailed information on UID and surprisal may be found in (Crocker, Demberg, and Teich 2015), published in this same issue.

Another measure is *propositional idea density*, which has been shown to affect reading times and recall (Keenan and Kintsch 1973). The semantic representation can then be used to calculate the number of words used to convey a proposition. For example, (a) and (b) contain 9 and 7 words respectively though they encode the same 5 propositions from Fig. 1, resulting in idea densities of 0.555 and 0.714, respectively. On this analysis, therefore, (a) is less informationally dense and may be easier to read.

Another possible objective is minimizing the length of syntactic dependencies (Gibson 1998, Gildea and Temperley 2010): in a nutshell, how far apart related words are placed in the sentence. Such features have been shown to improve surface realization quality in a generate-and-rank approach (Rajkumar and White 2011) where complete solutions are first generated and then ranked. But they have yet to see use *during* the generation process coming up with the solutions in the first place.

Challenges for AI Search Algorithms

Current surface realizers do, generally speaking, exhibit reasonable performance, yet significant deficiencies remain to be overcome. In particular, they often do not succeed in generating grammatically valid sentences within the given runtime limit, which in practice is typically small, in the order of one second. In such cases, they have to resort to other approximate methods that relax the grammar rules. Apart from these basic issues relevant to sentence realization as it stands, new challenges are posed by the more complex optimization objectives as just discussed. We list some concrete challenges and ideas in what follows.

[1]Surprisal values based on (Demberg, Keller, and Koller 2013) obtained from http://tinyurl.com/pltagdemo

Dead-end detection. As we have outlined, dead-ends are a major issue in OpenCCG (and related) search spaces. The same can be said of the search spaces in manifold AI problems, and there is a wealth of ideas whose potential in sentence realization is worth exploring. The AI Planning literature in particular has recently considered a variety of approaches towards dead-end detection. In particular, many known heuristic functions (*critical paths* (Haslum and Geffner 2000), *abstraction* (Edelkamp 2001, Helmert et al. 2014, Hoffmann, Kissmann, and Torralba 2014), *partial delete-relaxation* (Keyder, Hoffmann, and Haslum 2014)) have this capability, and could be useful in sentence realization problems. Planning languages are powerful enough to capture the category (as well as semantic) aspects of CCG search nodes and node combination rules, so a compilation approach could be feasible in principle. In practice, implementing the techniques natively, and exploiting the particular structure of CCG specifications, seems more promising.

Partial-order reduction. As in many other search problems, surface realization search spaces may contain permutative transition paths leading to identical search states. Partial-order reduction techniques (e. g. (Valmari 1989, McMillan 1993, Bonet et al. 2008, Wehrle and Helmert 2014)), originating in Verification and well established also in AI Planning, are a possible remedy which, to our knowledge, remains unexplored in NLG.

Predictive scoring functions. Scoring functions in OpenCCG evaluate search nodes based solely on their partial-solution content, without any predicition of possible completions. This can (obviously) be detrimental. For example, if we want to keep information density below a threshold, or keep syntactic dependencies short, then the score of partial sentences should take into account how we may be able to cover the remaining semantics. The obvious remedy, from an AI heuristic search perspective, is to devise *relaxations* that complete a search node $\alpha(X)$ into an approximate full sentence $\overline{\alpha}$. Arbitrary optimization objectives can then be evaluated on $\overline{\alpha}$, and be taken as an estimate of the quality of the best possible completion of $\alpha(X)$. But, for this to make sense, we need to ensure that $\overline{\alpha}$ does indeed correspond to a *best possible* completion. This is a known problem in AI for good-natured objectives that decompose over the structure of the solution, like additive action costs. But what if the optimization objectives are *non-local*, not decomposing as easily?

Non-local optimization objectives. The traditional optimization objectives in sentence realization, and all the more so the new complex optimization objectives discussed in the previous section, depend on *context*. N-grams depend on the neighboring words. Surprisal depends on the sentence prefix. Uniform density distribution is a property of the sentence as a whole, so depends on the sentence parts already fixed during the search. The same applies to the length of syntactic dependencies. To approximate best possible completions, the computation of relaxed solutions must take into account such complex objectives. In some approaches, like abstractions (projections and merge-and-shrink), this might be relatively straightforward as we can apply non-local criteria to abstract solution paths. In other approaches, like (partial) delete-relexation methods, this is less clear. One possibility could be a limited post-optimization of relaxed solutions (within each call to the predictive scoring function), so that the objectives can, again, be applied to complete sentences.

Target-value search. In adaptive language generation systems, in particular when we wish to adapt information density as appropriate in the current user context, the objective often is not to minimize a function, but instead to *find a solution whose score is close to a particular value of that function*. This is known as *target-value search*, a topic that has been considered in AI graph search already (Kuhn et al. 2008, Linares López, Stern, and Felner 2014), but has not been given extensive attention. Substantial challenges still remain on the AI side itself, in particular pertaining to the design and computation of heuristic functions: The "best possible completion" is now no longer the cheapest possible path postfix, but instead one whose cost corresponds to the "remaining target cost" as closely as possible. This raises completely new challenges for AI Planning heuristics. It raises more complex challenges still for non-local optimization objectives, cf. above.

In conclusion, there is a lot of work still to do, but many ideas from AI Search and Planning appear to be promising for better NLG surface realization. We hope that our paper can contribute to making this happen.

Acknowledgements

This work was partially supported by the DFG excellence cluster EXC 284 "Multimodal Computing and Interaction", the DFG collaborative research center SFB 1102 "Information Density and Linguistic Encoding", as well as the EU FP7 Programme under grant agreement no. 295261 (MEALS). We thank Maximilian Schwenger for discussions. We are also grateful to Almaz for great Eritrean food.

References

Bonet, B.; Haslum, P.; Hickmott, S. L.; and Thiébaux, S. 2008. Directed unfolding of petri nets. *Transactions on Petri Nets and Other Models of Concurrency* 1:172–198.

Cahill, A., and van Genabith, J. 2006. Robust pcfg-based generation using automatically acquired LFG approximations. In Calzolari, N.; Cardie, C.; and Isabelle, P., eds., *Proceedings of the 21st International Conference on Computational Linguistics (ACL'06)*. ACL.

Carroll, J. A., and Oepen, S. 2005. High efficiency realization for a wide-coverage unification grammar. In *Natural Language Processing–IJCNLP*, 165–176.

Crocker, M. W.; Demberg, V.; and Teich, E. 2015. Information density and linguistic encoding (ideal). *KI - Künstliche Intelligenz*.

Crundall, D.; Bains, M.; Chapman, P.; and Underwood, G. 2005. Regulating conversation during driving: a problem for mobile telephones? *Transportation Research Part F: Traffic Psychology and Behaviour* 8(3):197–211.

Demberg, V., and Keller, F. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition* 109(2):193–210.

Demberg, V., and Sayeed, A. 2011. Linguistic cognitive load: implications for automotive uis. In *Adjunct Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2011)*.

Demberg, V.; Keller, F.; and Koller, A. 2013. Incremental, predictive parsing with psycholinguistically motivated tree-adjoining grammar. *Computational Linguistics* 39(4):1025–1066.

Dethlefs, N.; Hastie, H.; Rieser, V.; and Lemon, O. 2012. Optimising Incremental Dialogue Decisions Using Information Density for Interactive Systems. In *EMNLP-CoNLL*, 82–93. ACL.

Drews, F. A.; Pasupathi, M.; and Strayer, D. L. 2008. Passenger and cell phone conversations in simulated driving. *Journal of Experimental Psychology: Applied* 14(4):392–400.

Edelkamp, S. 2001. Planning with pattern databases. In Cesta, A., and Borrajo, D., eds., *Proceedings of the 6th European Conference on Planning (ECP'01)*, 13–24. Springer-Verlag.

Frank, S. L.; Otten, L. J.; Galli, G.; and Vigliocco, G. 2015. The erp response to the amount of information conveyed by words in sentences. *Brain and language* 140:1–11.

Gibson, E. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition* 68(1):1–76.

Gildea, D., and Temperley, D. 2010. Do Grammars Minimize Dependency Length? *Cognitive Science* 34:286–310.

Hale, J. T. 2001. A Probabilistic Earley Parser as a Psycholinguistic Model. In *NAACL*.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In Chien, S.; Kambhampati, R.; and Knoblock, C., eds., *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS-00)*, 140–149. Breckenridge, CO: AAAI Press, Menlo Park.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery* 61(3).

Hoffmann, J.; Kissmann, P.; and Torralba, Á. 2014. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Schaub, T., ed., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*. Prague, Czech Republic: IOS Press.

Jaeger, T. F. 2006. *Redundancy and Syntactic Reduction in Spontaneous Speech*. Unpublished dissertation, Stanford University.

Jaeger, T. F. 2010. Redundancy and reduction: speakers manage syntactic information density. *Cognitive Psychology* 61(1):23–62.

Kay, M. 1996. Chart generation. In Joshi, A. K., and Palmer, M., eds., *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 200–204. Morgan Kaufmann / ACL.

Keenan, J., and Kintsch, W. 1973. Reading Rate and of Propositions Retention as a Function of the Number in the Base Structure of Sentences. *Cognitive Psychology* 5:257–274.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2014. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research* 50:487–533.

Kuhn, L.; Price, B.; de Kleer, J.; Do, M.; and Zhou, R. 2008. Heuristic search for target-value path problem. In *Proceedings of the 1st International Symposium on Search Techniques in Artificial Intelligence and Robotics*.

Levy, R., and Jaeger, T. F. 2007. Speakers optimize information density through syntactic reduction. *Advances in Neural Information Processing Systems* (20).

Levy, R. 2008. Expectation-based syntactic comprehension. *Cognition* 106(3):1126–77.

Linares López, C.; Stern, R.; and Felner, A. 2014. Solving the target-value search problem. In Edelkamp, S., and Bartak, R., eds., *Proceedings of the 7th Annual Symposium on Combinatorial Search (SOCS'14)*. AAAI Press.

McMillan, K. L. 1993. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In von Bochmann, G., and Probst, D. K., eds., *Proceedings of the 4th International Workshop on Computer Aided Verification (CAV'93)*, Lecture Notes in Computer Science, 164–177. Springer.

Nakatsu, C., and White, M. 2006. Learning to say it well: Reranking realizations by predicted synthesis quality. In Calzolari, N.; Cardie, C.; and Isabelle, P., eds., *Proceedings of the 21st International Conference on Computational Linguistics (ACL'06)*. ACL.

Rajkumar, R., and White, M. 2010. Designing agreement features for realization ranking. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 1032–1040.

Rajkumar, R., and White, M. 2011. Linguistically Motivated Complementizer Choice in Surface Realization. In *UCNLG+Eval: Language Generation and Evaluation Workshop*, 39–44. ACL.

Rajkumar, R., and White, M. 2014. Better surface realization through psycholinguistics. *Language and Linguistics Compass* 8(10):428–448.

Shannon, C. E. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal* 27(3):379–423.

Valmari, A. 1989. Stubborn sets for reduced state space generation. In *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets*, 491–515.

Wehrle, M., and Helmert, M. 2014. Efficient stubborn sets: Generalized algorithms and selection strategies. In Chien, S.; Do, M.; Fern, A.; and Ruml, W., eds., *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*. AAAI Press.

White, M., and Rajkumar, R. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, 410–419.

White, M., and Rajkumar, R. 2012. Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 244–255.

White, M. 2004. Reining in CCG chart realization. In Belz, A.; Evans, R.; and Piwek, P., eds., *Proceedings of the 3rd International Conference atural Language Generation*, volume 3123 of *Lecture Notes in Computer Science*, 182–191. Springer.

White, M. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation* 4(1):39–75.