PICARD: The Next Generator

Stephen Beale and Sergei Nirenburg
Computing Research Laboratory
Box 30001
New Mexico State University
Las Cruces, New Mexico 88003
sb,sergei@crl.nmsu.edu

Abstract

This paper[1] introduces a new line of research which ensures soundness and completeness in Natural Language text planners on top of an efficient control strategy. The work builds on the HUNTER-GATHERER analysis system (Beale, 96; Beale & Nirenburg, 96). That system employs constraint satisfaction, branch-and-bound and solution synthesis techniques to produce near linear-time processing for knowledge-based semantic analysis. PICARD enables similar results for the field of text planning by recasting localized means-end planning instances into abstractions connected by usage constraints that allow HUNTER-GATHERER to process the global problem as a simple constraint satisfaction problem. PICARD is currently being used to plan English and Spanish text in the Mikrokosmos Machine Translation Project.

HUNTER-GATHERER Overview

Implied information, background knowledge, ellipsis, coreference, figurative speech, lexical ambiguity; these are just a few of the immense challenges a computational-semantic system faces. Nevertheless, humans process language in real time every day with very little apparent misunderstanding. How can we make a computer do the same?

By constraining the problem. With search spaces that can reach 10^{18} and more just to deal with basic semantic dependencies in a sentence (disambiguating word senses and determining the semantic connections between them - or lexical choice and implementing semantic connections for generation), exhaustive search techniques are untenable. Constraint satisfaction problem (CSP) techniques allow early disambiguation and drastic search space reduction, while maintaining the integrity (soundness) of the solutions found.

[1] Research reported in this paper was supported in part by Contract MDA904-92-C-5189 from the U.S. Department of Defense.

Beale (1996) introduced a new control strategy for computational-semantic processing. The HUNTER-GATHERER methodology uses knowledge of constraint dependencies to identify small sub-problems which can be processed independently. Solution synthesis methods are then utilized to combine (gather) solutions to these sub-problems, or **circles**, into larger and larger solutions until the entire sentence is analyzed. As solutions for each circle are created, branch-and-bound and constraint satisfaction techniques are used to prune away (hunt down) non-optimal solutions.

HUNTER-GATHERER is a general control strategy that works particularly well for NL problems. Central to our application of this methodology to computational semantics, both in analysis and generation, is the hypothesis that such problems can almost always be viewed as bundles of tightly constrained sub-problems, each of which combine at higher, relatively constraint-free levels to produce a complete solution. Constraint dependency information, retrieved from the semantic co-occurrence information stored in the lexicon, which in turn exploits syntactic information, can be used to partition the complex disambiguation problem into simpler sub-problems, or "circles of dependency."

The concept of relatively independent "circles of dependency" can be exploited to allow inexpensive local analyses to be combined non-exponentially into global solutions. This is accomplished using a computational tool known as solution synthesis. Freuder (1978) introduced Solution Synthesis (SS) as a means to "gather up" all solutions for a CSP without resorting to traditional search methods. Freuder's algorithm created a set of two-variable nodes that contained all solutions for **every** two variable combination. These two-variable nodes were then combined into three-

variable nodes, and so on, until a node containing all the variables, i.e. the solution, was synthesized. At each step, constraints were propagated down and then back up the "tree" of synthesized nodes.

The HUNTER-GATHERER work extends and generalizes the solution synthesis methodology. The basic idea of synthesizing solution sets one order higher than their immediate ancestors is discarded. Instead, solution synthesis operates with maximally interacting groupings (circles) of variables of any order and extends to the highest levels of synthesizing. Freuder only creates second order nodes from adjacent variables in a list. After that, third and higher order nodes are blindly created from combinations of second order nodes. We redefine synthesis to operate on nodes of any size. Circles of co-constrained variables guide the synthesis process from beginning to end.

In addition to this modified solution synthesis, HUNTER-GATHERER employs branch-and-bound and constraint satisfaction methods to prune away non-optimal or impossible solutions from the search space. Beale, Nirenburg & Mahesh (1996) report "near" linear time processing for semantic analysis, with exhaustive search spaces in the **trillions** reduced to **hundreds**. The PICARD system extends the HUNTER-GATHERER analysis capabilities to the field of text planning by using constraint circles to identify localized means-end plan combinations. Constraint satisfaction techniques can then be used to ensure that only consistent plans are used together, while the other mechanisms in HUNTER-GATHERER, solution synthesis and branch-and-bound, efficiently find the optimal solution.

Using Constraint Satisfaction to Enable Abstractions

Figure 1 is a representation of the semantic content of a simple natural language sentence. In English the sentence could be rendered "Grupo Roche acquired Dr. Andreu through a subsidiary in Spain." The node names are semantic concepts taken from a language-independent ontology. Arc labels correspond to relations between concepts. The ontology defines for each concept the set of arcs that are allowed/expected, as well as the appropriate filler concepts. For simplicity, additional semantic information such as temporal relationships are not shown. Please consult (Beale, Nirenburg & Mahesh, 1995) for more information about semantic representation in the Mikrokosmos system. For our purposes, the

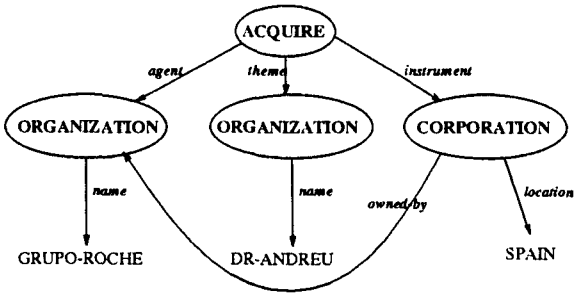

Figure 1: Example Semantic Representation

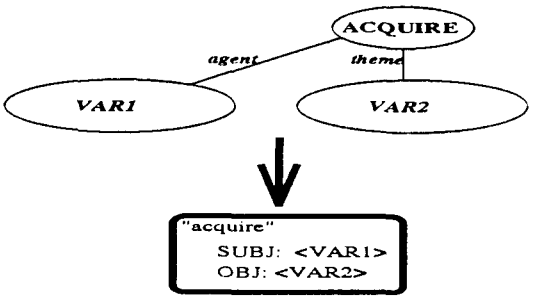

Figure 2: Lexicon Entry for *acquire*

details of the semantic representation and generation lexicon entries to follow are unimportant; they serve only as simple examples of control concepts that will apply to more complex problems.

Generation lexicon entries attempt to match parts of the input semantic structures and map them into target language surface structures. For instance, a lexicon entry for the concept **ACQUIRE** might look like Figure 2. The VARs in the entry will be bound to the corresponding semantic structures in the input, and their target realization will be planned separately and inserted into the output structure as shown. Typically, lexicon entries also contain semantic and pragmatic constraints. For instance, VAR1 might be constrained to be HUMAN. The entry could also be constrained to apply only to texts with certain stylistic characteristics. Collocational constraints are also important in generation. Any of these constraints can apply locally or can be propagated down to the VARs. The interplay of constraints is a major factor in determining the best overall plan.

Planning for Machine Translation comes in when we try to combine information in various lexicon entries to best match the input semantics with as little redundancy as possible and maximal adherence to the constraints.

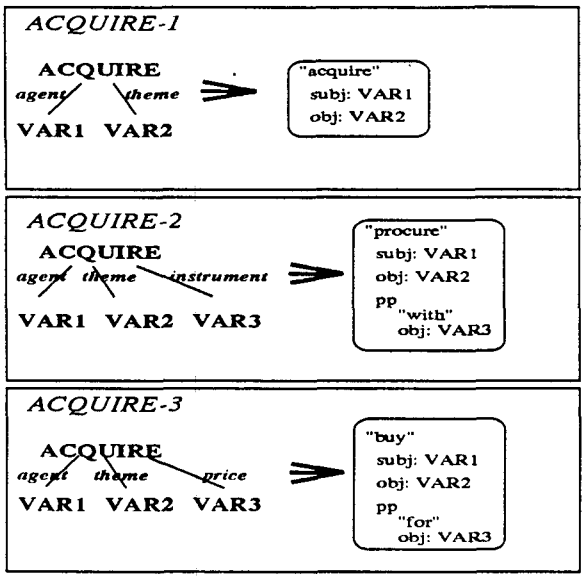

Figure 3: Three entries for ACQUIRE

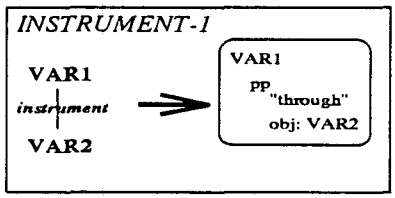

Figure 4: An entry for INSTRUMENT

The case study given here exemplifies the issues in lexical choice. Obviously text planning involves much more than simply picking words, but the principles outlined below apply to other components of text planning as well. Figures 3, 4 and 5 represent some possible lexicon entries that might be used in planning target English sentences for Figure 1.

It would be useful if we could divide text planning problems into relatively independent sub-problems and use HUNTER-GATHERER's solution synthesis to efficiently combine the smaller solutions. The problem is that solution synthesis requires an unchanging, orderly set of variables to start with. In text planning, as in all types of means-end planning systems, there is no fixed number of variables. "Variable," in this context, refers to a set of possible plans from which one **must** be chosen. A variable can be set up for **ACQUIRE**, which has three possible plans. One of them must be chosen. On the other hand, sometimes a plan for *instrument* is needed, and sometimes not. For instance, if *ACQUIRE-1*

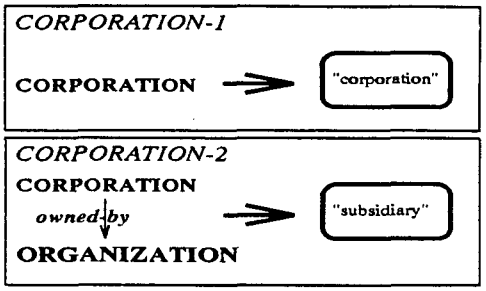

Figure 5: Two entries for CORPORATION

(Figure 3) is used, a separate sub-plan must be made for the *instrument* relation. Two "variables" would be needed, one for **ACQUIRE** and one for *instrument*. If the *ACQUIRE-2* is used, the *instrument* plan and variable are unnecessary. Lexicon entries which have different set of VARS, different preconditions and/or contain more or fewer relations all create differing amounts of sub-plans. These differences are compounded as different paths through the space of possible plans are taken.

PICARD solves this problem in a simple way. Means-end planning is carried out **locally** to determine, for each lexicon entry, the additional sub-plans that are needed. Again, these sub-plans correspond to VARs and missing relations or preconditions in the lexicon entry. For instance, the *ACQUIRE-1* entry requires a sub-plan for the missing *instrument* relation. For each needed sub-plan, a "usage constraint" is added to the lexicon entry that will "request" some "non-dummy"[2] sub-plan to be used that fulfills the need. The *ACQUIRE-1* entry, for example, would receive a usage constraint that requires it to use one of the sub-plans for *instrument*. In addition, for each of the sub-plans that can fill the need, a usage constraint is added such that those entries can only be used if "requested" by some other plan.

The main benefit this gives is that a stable set of "variables" can be created. There will be an **ACQUIRE** variable, from which one of the three lexicon entries must be selected. There will be an *instrument* variable, from which either the entry shown in Figure 4 will be used or the newly created dummy entry. These variables can then be processed by a solution synthesis algorithm. Whenever a choice is made, for instance selecting *ACQUIRE-1* for **AQUIRE**, the constraint satisfaction mechanism in HUNTER-GATHERER will eliminate all conflicting sub-plans. Picking entry

[2] "Dummy" plans are explained next.

ACQUIRE-1 will eliminate the dummy entry for *instrument*. Choosing entry *ACQUIRE-2* will eliminate all of the non-dummy *instrument* plans, as well as all the sub-plans that are created by the *instrument* plans. In this way, local plans can be linked together, but can be processed globally by an efficient solution synthesis control.

Conclusion

To summarize, a means-end planner is used **locally** to set up possible sub-plans. The sub-plans are connected with a system of usage constraints that inhibit or allow usage depending on the other sub-plans being used. The HUNTER-GATHERER system can then efficiently process the collection of sub-plans to find the best overall plan. Constraint satisfaction techniques described in (Beale, 96) automatically control the combination of sub-plans. Constraint satisfaction also ensures the soundness of **all** preconditions used in the lexicon entries, including those which are not related to the ideas presented above. Efficiency is gained by restricting the means-end planning component to local subproblems. Solutions to these sub-problems are then combined, utilizing solution synthesis, branch-and-bound and constraint satisfaction, by HUNTER-GATHERER.

The HUNTER-GATHERER control architecture has been used extensively in the Mikrokosmos semantic analyzer. The following table shows actual results of semantic analyses of various size problems (from sentences selected randomly from our corpus):

	Sent A	Sent B	Sent C
# plans	79	95	119
exhaustive	7.8 Mil	56.7 Mil	235 Bil
HUNT-GATHER	179	254	327

It is interesting to note that a 20% increase in the number of total plans[3] (79 to 95) results in a 626% increase (7.8M to 56M) in the number of exhaustive combinations possible, but only a 42% increase (179 to 254) in the number of combinations considered by HUNTER-GATHERER. As one moves on to even more complex problems, a 25% increase (95 to 119) in the number of plans catapults the exhaustive complexity by 414,600% (56M to 235B) and yet only increases the HUNTER-GATHERER complexity by 29% (254 to 327). As the problem size increases, the minor effects of non-local interactions diminish with respect to the size of the problem. We expect, therefore, the behavior of this algorithm to move even closer to linear with larger problems (for example, ones involving discourse).

Generation in the Mikrokosmos project is a relatively new development. Currently we are developing methods to reverse multilingual analysis lexicons. PICARD has been used to back-translate the semantic analyses of the Mikrokosmos analyzer using these reversed lexicons. Efficiency results similar to those reported above were obtained.

The HUNTER-GATHERER algorithms are complete with respect to the set of monotonic solutions. Currently, solutions with plans that temporarily violate preconditions of other plans (with the "violation" corrected by a later plan) will not be allowed. Besides this limitation, HUNTER-GATHERER is guaranteed to find the same solution(s) as an exhaustive search. In addition, the constraint satisfaction component of HUNTER-GATHERER ensures soundness. By converting means-end planners into a format that can be used by HUNTER-GATHER, PICARD achieves efficient processing with guaranteed soundness and completeness without sacrificing the generality of means-end planning.

[3]The total number of "plans" corresponds to the total number of word senses for all the words in the sentence.

References

Stephen Beale. 1996. HUNTER-GATHERER: Applying Constraint Satisfaction, Branch-and-Bound and Solution Synthesis to Natural Language Semantics. Technical Report, MCCS-96-289, Computing Research Lab, New Mexico State Univ.

Stephen Beale, Sergei Nirenburg and Kavi Mahesh. 1996. HUNTER-GATHERER: Three Search Techniques Integrated for Natural Language Semantics. To appear in the Thirteenth National Conference on Artificial Intelligence (AAAI96), Portland, Oregon.

Stephen Beale, Sergei Nirenburg and Kavi Mahesh. 1995. Semantic Analysis in the Mikrokosmos Machine Translation Project. In *Proceedings of the 2nd Symposium on Natural Language Processing*, 297-307. Bangkok, Thailand.

E.C. Freuder. 1978. Synthesizing Constraint Expressions. *Communications ACM* 21(11): 958-966.