Maximal Consistent Subsets

Robert Malouf*
San Diego State University

Default unification operations combine strict information with information from one or more defeasible feature structures. Many such operations require finding the maximal subsets of a set of atomic constraints that are consistent with each other and with the strict feature structure, where a subset is maximally consistent with respect to the subsumption ordering if no constraint can be added to it without creating an inconsistency. Although this problem is NP-complete, there are a number of heuristic optimizations that can be used to substantially reduce the size of the search space. In this article, we propose a novel optimization, ***leaf pruning,*** *which in some cases yields an improvement in running time of several orders of magnitude over previously described algorithms. This makes default unification efficient enough to be practical for a wide range of problems and applications.*

1. Introduction

In unification-based grammatical frameworks, it often desirable to combine information from possibly inconsistent sources. Over the years, a number of **default unification** operations have been proposed[1], which combine a strict feature structure with one or more defeasible feature structures. These operations preserve all information in the strict feature structure, while bringing in as much information as possible from the defeasible structures. Default unification has been used to address a wide variety of linguistic knowledge representation problems, including lexical inheritance hierarchies (Copestake 1993; Ginzburg and Sag 2001), lexical semantics (Lascarides and Copestake 1998), grammar induction (Briscoe 1999; Villavicencio 2002; Petersen 2004), anaphora resolution (Grover et al. 1994; Prüst, Scha, and van den Berg 1994), and discourse processing (Gurevych et al. 2003; Alexandersson, Becker, and Pfleger 2004), among many others.

Although the various default unification operators differ in their particulars, most involve something like Carpenter (1992)'s **credulous default unification** as one step. The result of credulously adding the default information in G to the strict information in F is:

$$\textit{cred-unify}(F, G) \equiv \{\textit{unify}(F, G'), \text{ where } G' \text{ subsumes } G \text{ and } G' \text{ is maximal such that } \textit{unify}(F, G') \text{ is defined}\}$$

* Department of Linguistics and Asian/Middle Eastern Languages, San Diego State University, 5500 Campanile Drive, San Diego, CA 92182-7727 USA, E-mail: rmalouf@mail.sdsu.edu.

1 For example, Bouma (1992), Carpenter (1992), Prüst (1992), Calder (1993), Lascarides and Copestake (1999), Alexandersson and Becker (2001), and Ninomiya, Miyao, and Tsujii (2002). See Bouma (2006) for a recent overview.

In other words, *cred-unify*(F, G) is the result of unifying F with the maximal consistent subset(s) of the atomic constraints in G. A subset of constraints is maximally consistent with respect to the subsumption ordering if no constraint can be added to it without creating an inconsistency. In general, there may be more than one subset of the constraints in G that is consistent with F and maximal, so the result of credulous default unification is a set of feature structures.

One example of the use of credulous default unification for the resolution of discourse anaphora comes from Grover et al. (1994). Consider the mini-discourse: *Jessy likes her brother. So does Hannah.* To resolve the anaphoric predicate in the second sentence, we can set up meaning representations for the source and the target:

$$\textbf{Source:} \begin{bmatrix} \text{REL} & \textit{like} \\ \text{AGENT} & \boxed{1}\textit{jessy} \\ \text{PATIENT} & \begin{bmatrix} \text{REL} & \textit{brother} \\ \text{THEME} & \boxed{1} \end{bmatrix} \end{bmatrix} \qquad \textbf{Target:} \begin{bmatrix} \text{AGENT} & \textit{hannah} \end{bmatrix}$$

and credulously unify the source with the strict information in the target.

To proceed, we first decompose the source feature structure into the following five atomic ground constraints:

$$\left\{ \begin{bmatrix} \text{REL} & \textit{like} \end{bmatrix}, \begin{bmatrix} \text{AGENT} & \textit{jessy} \end{bmatrix}, \begin{bmatrix} \text{PATIENT|REL} & \textit{brother} \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} \text{PATIENT|THEME} & \textit{jessy} \end{bmatrix}, \begin{bmatrix} \text{AGENT} & \boxed{1} \\ \text{PATIENT|THEME} & \boxed{1} \end{bmatrix} \right\}$$

Then, we find the maximal subsets of the remaining constraints which are mutually consistent with the target. This yields two solutions:

$$\begin{bmatrix} \text{REL} & \textit{like} \\ \text{AGENT} & \boxed{1}\textit{hannah} \\ \text{PATIENT} & \begin{bmatrix} \text{REL} & \textit{brother} \\ \text{THEME} & \boxed{1} \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} \text{REL} & \textit{like} \\ \text{AGENT} & \textit{hannah} \\ \text{PATIENT} & \begin{bmatrix} \text{REL} & \textit{brother} \\ \text{THEME} & \textit{jessy} \end{bmatrix} \end{bmatrix}$$

corresponding to the sloppy identity and the strict identity readings of the anaphoric expression.

2. Algorithm

A key step for applying most default unification operators is finding the maximal consistent subsets of a set of constraints C. Unfortunately, finding these maximal consistent subsets is an expensive operation, and, in fact, is NP-complete. Let $T = \{T_1, \ldots, T_m\}$ be the set of **conflicts** in C, where a conflict is a minimal set of constraints that are mutually inconsistent with the target. For this example, T consists of the two conflicts:

$$\left\{ \left\{ \begin{bmatrix} \text{PATIENT|THEME} & \textit{jessy} \end{bmatrix}, \begin{bmatrix} \text{AGENT} & \boxed{1} \\ \text{PATIENT|THEME} & \boxed{1} \end{bmatrix} \right\}, \left\{ \begin{bmatrix} \text{AGENT} & \textit{jessy} \end{bmatrix} \right\} \right\}$$

Removing any one of the constraints from a conflict T_i would break that conflict, so if we could remove from C at least one member of each T_i, the remaining constraints would be

mutually consistent with the target information. Finding the maximal consistent subsets of C then is equivalent to finding the minimal subset $C' \subset C$ such that each T_i contains at least one member of C'. This is the **hitting subset problem,** a classic NP-complete problem (Karp 1972).

An algorithm to find the maximal consistent subsets of C must check each subset of C for consistency. One way to proceed is to construct a **spanning tree** of the boolean lattice of subsets of C. This takes the form of a binomial tree, as in Figure 1 (Bird and Hinze 2003). At each node, we keep track of k, the index of the constraint that was removed from the parent set to create that subset. For example, subset $\{c_1, c_3\}$ was formed by removing c_2 from $\{c_1, c_2, c_3\}$, so $k = 2$. The descendants of a node are constructed by successively dropping each of the constraints c_i where $k < i \leq |C|$. This ensures that we will visit every subset of C exactly once.

The algorithm described by Grover et al. (1994) performs a breadth-first search of the subset lattice, with one important optimization. Because the cardinality of the subsets at each level is one greater than those on the level below it, a breadth-first search of this tree will consider all larger sets before considering any smaller ones. Furthermore, because each subset is produced by removing constraints from its parent set, every node in a subtree is a subset of its root. This means that once a consistent set is found, no descendants of that set can be maximal, and that subtree can be pruned from the search space. However, consistent subsets that are maximal in their branch of the tree may turn out not to be globally maximal. For example, in Figure 1, if $\{c_1, c_2\}$ is consistent and $\{c_1, c_3\}$ is not, a breadth-first search would identify both $\{c_1, c_2\}$ and $\{c_1\}$ as consistent and (locally) maximal. A final post-check for set inclusion can remove pseudo-maximal results like $\{c_1\}$.

In addition to pruning branches rooted by a consistent subset (call this **root pruning**), the organization of the search space into a binomial tree allows another valuable optimization. The deepest leaf node in any subtree is the set formed from the root by removing all constraints $c_{k<i\leq|C|}$, and every set in the subset is a superset of that deepest leaf. Because no superset of an inconsistent set of constraints can be consistent, if the foot of a subtree is inconsistent then clearly no node in the tree can be consistent, and the entire tree can be skipped (call this **leaf pruning**). Taking both root pruning and leaf pruning together, the only subtrees that need to be explored are those whose root is

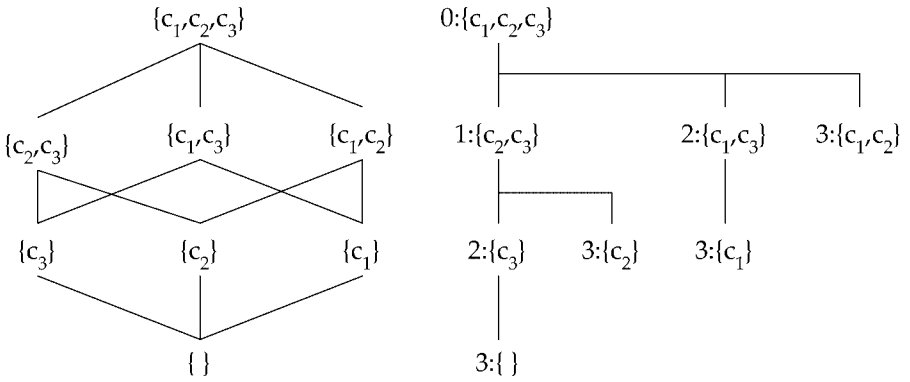

Figure 1
Boolean lattice and binomial spanning tree for $|C| = 3$.

inconsistent but whose deepest leaf is consistent. No other subtrees can possibly contain a solution.

Figure 2 gives a breadth-first search algorithm that implements these optimizations. Like Grover et al. (1994), this algorithm requires a post-check to remove pseudo-maximal subsets from *results*. A queue is used to keep track of subsets S that are yet to be checked for consistency, along with the index k of the constraint that was last dropped, and a flag *leftmost* that indicates whether that subset is the leftmost child. Because the deepest leaf node of the leftmost child is the same as the deepest leaf node of the parent, we are guaranteed that the deepest leaf of a leftmost child is consistent. Keeping track of leftmost children allows us to avoid a substantial number of redundant consistency checks.

3. Evaluation

The graphs in Figure 3 and Figure 4 show an empirical comparison between a breadth-first search with root pruning (BFS-R) and a breadth-first search with root and leaf pruning (BFS-RL) on randomly generated problems. The graphs show the number of subsets that were checked for consistency, as it relates to $|C|$, the number of constraints, and p, the probability that two members of C are consistent. Larger values for p generally lead to fewer but larger maximal consistent subsets. All counts are averaged across 100 randomly generated sets of ground constraints. In generating random problems, we make the simplifying assumptions that all constraints are consistent with any indefeasible information, and that a subset of constraints that are pairwise consistent is a consistent subset.

The first thing to note in these graphs is that root pruning by itself provides very little benefit. For most values of p, the number of subsets checked by BFS-R is very close to the worst case maximum $2^{|C|}$. A possible reason for this is that root pruning

```
MAXIMALCONSISTENTSUBSETS(C)
 1  results ← ∅
 2  Q ← {⟨C, 0, FALSE⟩}
 3  while Q ≠ ∅
 4        do ⟨S, k, leftmost⟩ ← head(Q)
 5           if Consistent(S)
 6              then if S ⊄ R for all R in results
 7                       then ▷ S is maximally consistent
 8                            APPEND(results,S)
 9              else  L ← S − {c_i : k < i ≤ |C|} ▷ L is S's deepest leaf
10                    if leftmost or Consistent(L)
11                       then leftmost ← TRUE
12                            for i ← k + 1 to |C|
13                                do ENQUEUE(Q, ⟨S − c_i, i, leftmost⟩)
14                                   leftmost ← FALSE
15           DEQUEUE(Q)
16  return results
```

Figure 2
Find the maximal consistent subsets of C. Performs a breadth-first search of the subset tree, with root and leaf pruning.

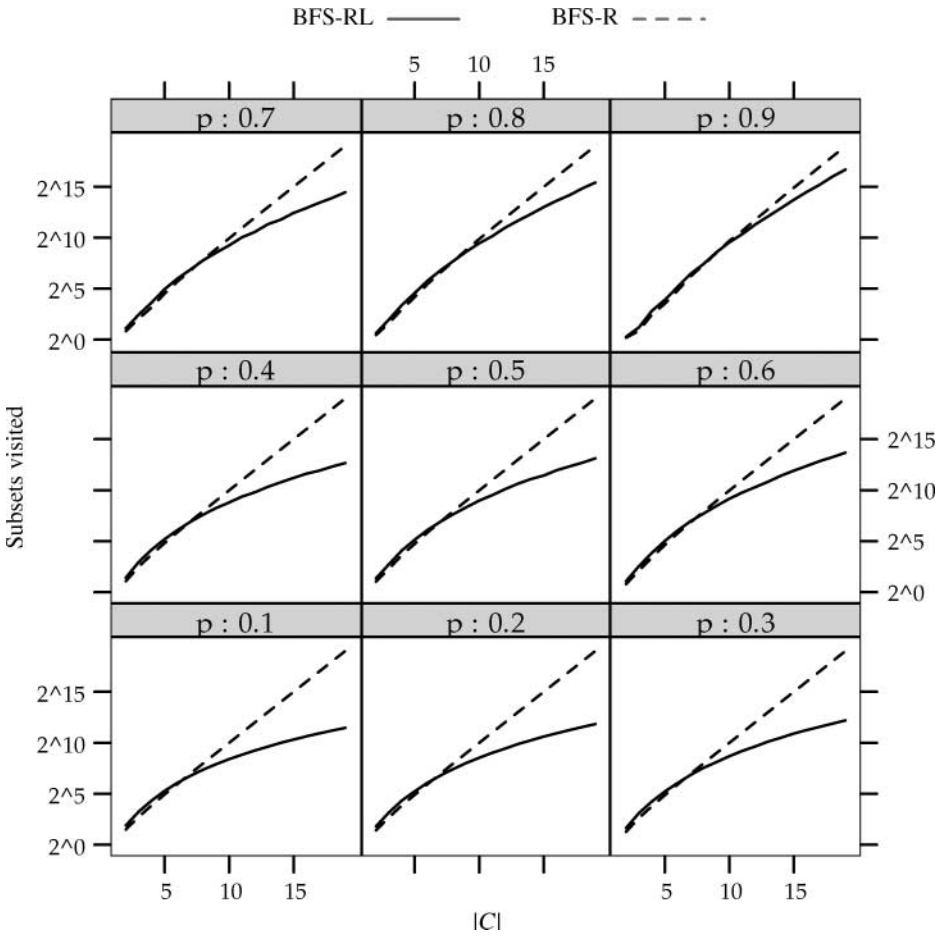

Figure 3
Comparison of breadth-first search using root pruning alone (BFS-R) and in combination with leaf pruning (BFS-RL). $|C|$ is the number of ground constraints, p is the fraction of the ground constraints which are pairwise consistent, and "Subsets visited" is the number of subsets of C which were checked for consistency (on a logarithmic scale). All counts are based on the average of 100 randomly generated problems.

will have the greatest effect when consistent subsets are found in the interior nodes of the binomial search tree. However, the configuration of the search space is such that most nodes are either leaves or very close to leaves, and only a few nodes have a large number of descendants. Therefore, root pruning mostly removes very small subtrees and has only a small effect on the overall cost of the algorithm.

The next observation to make is that for small values of $|C|$ (in these experiments, less than 7), BFS-RL is very slightly more expensive than BFS-R. In these cases, the advantages of leaf pruning do not outweigh the cost of the extra consistency checks required to implement it. As $|C|$ increases, though, leaf pruning can offer substantial improvements. For $|C| = 19$ and $p = 0.1$, leaf pruning eliminates more than 99.5% of the search space, leading to a 185-fold improvement in running time. As p increases, the benefits of leaf pruning do become more modest. Larger values of p mean fewer inconsistent leaf nodes, so fewer subtrees are able to be eliminated. Even so, the savings

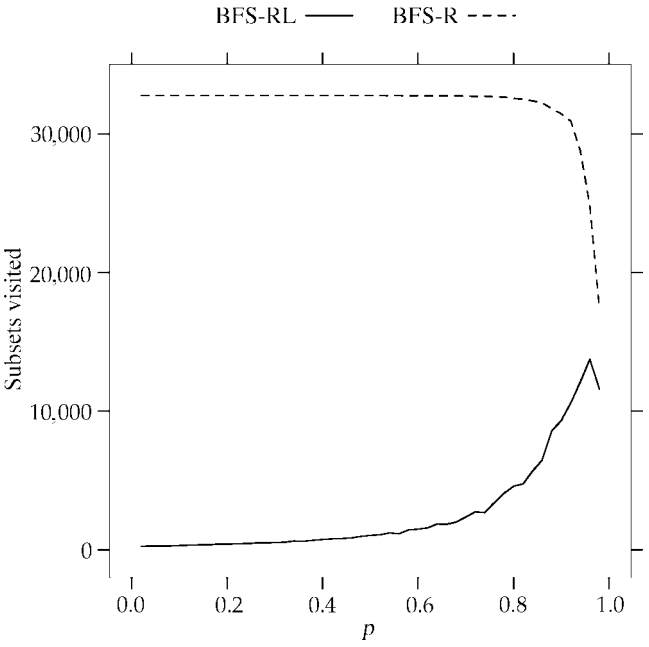

Figure 4
Comparison of subsets visited by BFS-R and BFS-RL as a function of p for $|C| = 15$. All counts are based on the average of 100 randomly generated problems.

from leaf pruning can still be dramatic: at $|C| = 19$ and $p = 0.9$, leaf pruning yields a nearly five-fold improvement in speed.

Values of $|C|$ and p that can be realistically expected will vary widely from application to application. An anonymous reviewer reports that in one application, the resolution of non-monotonic lexical inheritance for constraint-based grammars, p is generally greater than 0.7. This may be due in part to the fact that most constraint-based grammar development platforms do not support defaults (Copestake's [2002] LKB is a notable exception), and so grammar engineers tend to avoid the use of default overriding. Ginzburg and Sag (2001) propose a more comprehensive use of defaults, and grammars written following these principles would likely have a much lower value of p. To my knowledge, however, these ideas have not yet made their way into any large-scale grammar implementations.

Ninomiya, Miyao, and Tsujii (2002) describe experiments using default unification for robust parsing and automatic grammar augmentation via a kind of explanation-based learning. For this application, all features of a rule in the base XHPSG grammar (Tateisi et al. 1998) are considered defaults that can be overridden if necessary to get a successful parse of a sentence. In this case, $|C|$ is likely very large and grows quickly with the length of the sentences being parsed. The value of p will depend on the coverage of the base grammar, but can be expected to be fairly close to 1.0 for most domains. In situations such as this, where p is expected to fall close to the worst case for leaf pruning, one could consider inverting the search direction of the algorithm in Figure 2. Rather than beginning with C and removing constraints until a consistent subset is found, we could instead begin with the empty subset and add constraints until an inconsistency is found. In either case, the frontier in the search space between consistent and inconsistent subsets is where maximally consistent subsets will be found,

and leaf pruning can be used to eliminate regions of the search space that contain only consistent or inconsistent subsets.

4. Conclusions

Finding the maximal consistent subsets of a set of ground constraints is an important sub-problem for many natural language processing and knowledge representation tasks. Unfortunately, the problem is NP-complete, and in the worst case requires checking all $2^{|C|}$ subsets of C for consistency. Previously proposed algorithms have produced approximate solutions (Boppana and Halldórsson 1992), or have weakened the requirements to make finding a solution easier (Ninomiya, Miyao, and Tsujii 2002).

By using deepest leaf pruning, the algorithm described in the previous sections improves on the method of Grover et al. (1994) and is able to achieve substantial gains over the worst case running time for a large class of problems. An efficient method for finding maximal consistent subsets can make default unification practical for problems such as large-scale lexical representation, on-line discourse processing, or ontology construction.

References

Alexandersson, Jan and Tilman Becker. 2001. Overlay as the basic operation for discourse processing in a multimodal dialogue system. In *Proceedings of the IJCAI Workshop "Knowledge and Reasoning in Practical Dialogue Systems,"* pages 8–14, Seattle, WA.

Alexandersson, Jan, Tilman Becker, and Norbert Pfleger. 2004. Scoring for overlay based on informational distance. In *Proceedings of KONVENS 2004*, pages 1–4, Vienna, Austria.

Bird, Richard and Ralf Hinze. 2003. Functional pearl: Trouble shared is trouble halved. In *Proceedings of the 2003 ACM SIGPLAN Workshop on Haskell*, pages 1–6, Uppsala, Sweden.

Boppana, Ravi and Magnús M. Halldórsson. 1992. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196.

Bouma, Gosse. 1992. Feature structures and nonmonotonicity. *Computational Linguistics*, 18(2):183–204.

Bouma, Gosse. 2006. Unification: Classical and default. In Keith Brown, editor, *Encyclopedia of Language and Linguistics*. Elsevier, New York.

Briscoe, E. J. 1999. The acquisition of grammar in an evolving population of language agents. *Electronic Transactions in Artificial Intelligence. Special Issue: Machine Intelligence*, 3(035):47–77.

Calder, Jonathan. 1993. Feature-value logics: Some limits on the role of defaults. In C. J. Rupp, Mike Rosner, and Rod Johnson, editors, *Constraints, Language and Computation*. Academic Press, London, pages 20–32.

Carpenter, Bob. 1992. Skeptical and creduluous default unification with applications to templates and inheritance. In Ted Briscoe, Anne Copestake, and Valerie de Paiva, editors, *Default Inheritance within Unification-Based Approaches to the Lexicon*. Cambridge University Press, Cambridge, UK, pages 13–37.

Copestake, Ann. 1993. Defaults in lexical representation. In E. J. Briscoe, A. Copestake, and V. de Paiva, editors, *Inheritance, Defaults and the Lexicon*. Cambridge University Press, Cambridge, UK, pages 223–245.

Copestake, Ann. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, CA.

Ginzburg, Jonathan and Ivan A. Sag. 2001. *Interrogative Investigations*. CSLI Publications, Stanford, CA.

Grover, Claire, Chris Brew, Marc Moens, and Suresh Manandhar. 1994. Priority union and generalisation in discourse grammar. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 17–24, Las Cruces, New Mexico.

Gurevych, Iryna, Robert Porzel, Elena Slinko, Norbert Pfleger, Jan Alexandersson, and Stefan Merten. 2003. Less is more: Using a single knowledge representation

in dialog systems. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 14–21, Edmonton, Alberta.
Karp, Richard. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*. Plenum Press, New York, pages 85–103.
Lascarides, Alex and Ann Copestake. 1998. Pragmatics and word meaning. *Journal of Linguistics*, 34:387–414.
Lascarides, Alex and Ann Copestake. 1999. Default representation in constraint-based frameworks. *Computational Linguistics*, 25:55–105.
Ninomiya, Takashi, Yusuke Miyao, and Jun'ichi Tsujii. 2002. Lenient default unification for robust processing within unification based grammar formalisms. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 1–7, Taipei, Taiwan.
Petersen, Wiebke. 2004. A set-theoretic approach for the induction of inheritance hierarchies. *Electronic Notes in Theoretical Computer Science*, 53:296–308.
Prüst, Hub. 1992. *On Discourse Structuring, VP Anaphora and Gapping*. Ph.D. thesis, University of Amsterdam.
Prüst, Hub, Remko Scha, and Martin van den Berg. 1994. Discourse grammar and verb phrase anaphora. *Linguistics and Philosophy*, 17(3):261–327.
Tateisi, Yuka, Kentaro Torisawa, Yusuke Miyao, and Jun'ichi Tsujii. 1998. Translating the XTAG English grammar to HPSG. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 172–175, Philadelphia, PA.
Villavicencio, Aline. 2002. *The Acquisition of a Unification-Based Generalised Categorial Grammar*. Ph.D. thesis, Cambridge University.