Pomset mcfgs

Michael J Pan
University of California Los Angeles
mjpan@cs.ucla.edu

Abstract

This paper identifies two orthogonal dimensions of context sensitivity, the first being context sensitivity in concurrency and the second being structural context sensitivity. We present an example from natural language which seems to require both types of context sensitivity, and introduce partially ordered multisets (pomsets) mcfgs as a formalism which succintly expresses both.

Introduction

Researchers in computer science and formal language theory have separately investigated context sensitivity of languages, addressing disjoint dimensions of context sensitivity. Researchers in parallel computing have explored the addition of concurrency and free word order to context free languages, i.e. a concurrency context sensitivity (Gischer, 1981; Warmuth and Haussler, 1984; Pratt, 1985; Pratt, 1986; Lodaya and Weil, 2000). Computational linguistis have explored adding crossing dependency and discontinuous constituency, i.e. a structural context sensitivity (Seki et al., 1991; Vijay-Shanker et al., 1987; Stabler, 1996).

Research considering the combination of two dimensions of expressing context sensitivity have been sparse, e.g. (Becker et al., 1991), with research dedicated to this topic virtually nonexistent. Natural languages are not well expressed by either form of context sensitivity alone. For example, in Table 1, sentences 1-8 are valid, but 9, 10 are invalid constructions of Norwegian. In addition to the crossing dependency between the determiner and adverb phrase, this example can be described by either Bobaljik's paradox (Bobaljik, 1999), which asserts that relative ordering of clausal constituents are not unambiguously determined by the phrase structure, or shape conservation (Müller, 2000), i.e. that linear precedence is preserved despite movement operations. In other words, the two structurally context sensitive components (due to the crossing dependency between them) can be shuffled arbitrarily, leading to concurrent context sensitivity.

Derfor	ga	**Jens Kari kyllingen**	tydeligvis	ikke	lenger	kald	
Therefore	gave	**Jens Kari the chicken**	evidently	not	longer	cold	

Derfor ga **Jens Kari** tydeligvis **kyllingen** ikke lenger kald
Derfor ga **Jens** tydeligvis **Kari kyllingen** ikke lenger kald
Derfor ga **Jens** tydeligvis **Kari** ikke **kyllingen** lenger kald
Derfor ga **Jens** tydeligvis **Kari** ikke lenger **kyllingen** kald
Derfor ga **Jens** tydeligvis ikke lenger **Kari kyllingen** kald
Derfor ga tydeligvis **Jens** ikke lenger **Kari kyllingen** kald
Derfor ga tydeligvis ikke **Jens** lenger **Kari kyllingen** kald
* Derfor ga **Jens** ikke tydeligvis **Kari** lenger **kyllingen** kald
* Derfor ga **Jens** ikke tydeligvis **kyllingen** lenger **Kari** kald

Table 1: Bobaljik's paradox/shape conservation example

This paper proposes pomset mcfgs as a formalism for perspicuously expressing both types of context sensitivity. [1] The rest of the paper is organized as follows. Section 1 introduces pomsets, pomset operations, and pomset properties. Section 2 provides a definition of pomset mcfgs by extending the standard definition of mcfgs, defined over tuples of strings, to tuples of pomsets. Section 3 discusses pomset mcfg parsing.

[1] Other pomset based formalisms (Lecomte and Retore, 1995; Basten, 1997; Nederhof et al., 2003) have been limited to the use of pomsets in context free grammars only.

Proceedings of the 10th Conference on Parsing Technologies, pages 106–108,
Prague, Czech Republic, June 2007. ©2007 Association for Computational Linguistics

1 Pomsets

In this section, we define pomsets as a model for describing concurrency. A labelled partial order (LPO) is a 4 tuple (V, Σ, $\preceq$, μ) where V is a set of vertices, Σ is the alphabet, $\preceq$ is the partial order on the vertices, and μ is the labelling function μ:V$\rightarrow$ Σ. A pomset is a LPO up to isomorphism. The concatenation of pomsets p and q is defined as ;(p,q) = ($V_p \cup V_q, \Sigma_p \cup \Sigma_q, \preceq_p \cup \preceq_q \cup V_p \times V_q, \mu_p \cup \mu_q$). The concurrency of pomsets p and q is defined as $\|$(p,q) = ($V_p \cup V_q, \Sigma_p \cup \Sigma_q, \preceq_p \cup \preceq_q, \mu_p \cup \mu_q$). Pomset isolation ($\iota$) is observed only in the context of concurrency. The concurrence of an isolated pomset with another pomset is defined as $\|(\iota$p,q) = ($\{v_p\} \cup V_q, p_\lambda \cup \Sigma_q, \preceq_q, \{(p_\lambda, v_p)\} \cup \mu_q$), where λp is the set of linearizations for p, and p_λ is a function which returns an element of λp. Let $\|_i$ be a pomset concurrency operator restricted to an arity of i. Because concurrency is both associative and commutative, without isolation, $\|_m\|_n$ = $\|_n\|_m$ = $\|_{m+n}$, defeating any arity restrictions. Isolation allows us to restrict the arity of the concurrency operator, guaranteeing that in all linearizations of the pomset, the linearizations of the isolated subpomsets are contiguous.[2] A mildly concurrent operator $\iota\ \|_n$, i.e. an n-concurrent operator, is a composite operator whose concurrency is isolated and restricted to an arity of n, such that it operates on at most n items concurrently.

2 Pomset mcfgs

There are many (structural) mildly context sensitive grammar formalisms, e.g. mcfg, lcfrs, mg, and they have been shown to be equivalent (Vijay-Shanker et al., 1987). In this section we construct mcfgs over pomsets (instead of strings) to define grammars with both types of context sensitivity.

A pomset mcfg G is a 7-tuple (Σ,N,O,P,F,R,S) such that Σ is a finite non-empty set of atoms, i.e. terminal symbols, N is a finite non-empty set of nonterminal symbols, where N$\cap\Sigma=\emptyset$, O is a set of valid pomset operators, P is a set of i-tuples of pomsets labelled by $\Sigma \cup$N, F is a finite set of pomset rewriting functions from tuples of elements of P into elements in P, F$\subseteq\{$ g:$P^n \rightarrow$P $|$ n>0 $\}$, R is a finite set of rewrite rules which pair n-ary elements of F with n+1 nonterminals, and S$\in$N is the start symbol, and d(S) = 1.

This definition extends the standard mcfg definition (Seki et al., 1991), with two main differences. First, strings have been generalized to pomsets, i.e. P is a set of i-tuples of pomsets instead of i-tuples of strings. It follows that F, the set of functions, operate on tuples of pomsets instead of tuples of strings, and so forth. Second, pomset mcfgs explicitly specify O, the set of possible operators over the pomsets, e.g. $\{;, \iota\ \|_2\}$; string mcfgs have an implied operator set O=$\{;\}$ (i.e. just string concatenation).

Additionally, just as in mcfgs, where the arity of string components are limited, we can limit the arity of the concurrency of pomsets. A n-concurrent pomset mcfg is a pomset mcfg such that for all concurrency operators $\|_i$ in the grammar, i$\leq$n. A pomset mcfg with no concurrency among its components is a 1-concurrent pomset mcfg, just as a cfg is a 1-mcfg.

3 Parsing

In this section we propose a strategy for parsing pomset mcfgs, based on IDL parsing (Nederhof and Satta, 2004). We define pomset graphs, which extend IDL graphs and pom-automata and are defined over tuples of pomsets (or tuples of idl expressions), rather than single pomsets or idl expressions. An informal analysis of the computational complexity for parsing pomset mcfgs follows.

Pomset graphs The construction is quite straight forward, as pomsets themselves can already be considered as DAGs. However, in the pomset graph, we add two vertices, the start and end vertices. We then add precedence relations such that the start vertex precedes all minimal vertices of the pomset, and that the end vertex succeeds all maximal vertices of the pomset. For any nonempty pomset, we define $V_{min} \subseteq$V and $V_{max} \subseteq$V to be the minimal and maximal, respectively, vertices of V. Informally, no vertex in a pomset precede V_{min} and none succeed any in V_{max}. Formally, $\forall$ v$\in$V, v'$\in$V,v'$\neq$v, V_{min} = $\{$ v $|$ (v',v) $\notin \preceq \}$ and V_{max} = $\{$ v $|$ (v,v') $\notin \preceq \}$. The start vertex is then labelled with the empty string, ϵ, and the end vertex is labelled with σ', a symbol not in Σ.

[2] Pomset isolation is similar to proposals in for string isolation in linear specification language (Goetz and Penn, 2000), locking in idl-expressions (Nederhof and Satta, 2004), and integrity constraints in fo-tag (Becker et al., 1991).

Given a pomset p= (V_p,Σ,$\preceq$,μ_p), a pomset graph for p is a vertex labelled graph γ(p) = (V_γ,E,μ_γ) where V_γ and E are a finite set of vertices and edges, where V_γ=$V_p \cup \{v_s, v_e\}$ and E= $\preceq \cup v_s \times V_{min} \cup V_{max} \times v_e$, Σ_γ=$\Sigma \cup \{\epsilon, \sigma'\}$, where σ' is a symbol not in Σ, and μ_γ=$\mu_p \cup \{(v_s,\epsilon),(v_e,\sigma')\}$ is the vertex labelling function. Having defined the pomset graph, we can apply the IDL parsing algorithm to the graph.

Complexity While the complexity of the membership problem for pomset languages in general is NP-complete (Feigenbaum et al., 1993), by restricting the context sensitivity of the pomset grammars, polynomial time complexity is achievable. The complexity of the parsing of IDL graphs is O(n^{3k}) (Nederhof and Satta, 2004) where k is the width of the graph, and the width is a measurement of the number of paths being traversed in parallel, i.e. the arity of the concurrent context sensitivity. Our intuition is that the parameterization of the complexity according to the number of parallel paths applies even when structural context sensitivity is added. Thus for a k-concurrent m-structural mcfg, we conjecture that the complexity is O(n^{3km}).

4 Conclusion

In this paper we identified two types of context sensitivity, and provided a natural language example which exhibits both types of context sensitivity. We introduced pomset mcfgs as a formalism for describing grammars with both types of context sensitivity, and outlined an informal proof of the its polynomial-time parsing complexity.

References

Twan Basten. 1997. Parsing partially ordered multisets. *International Journal of Foundations of Computer Science*, 8(4):379–407.

Tilman Becker, Aravind K. Joshi, and Owen Rambow. 1991. Long distance scrambling and tree adjoining grammars. In *Proceedings of EACL-91, the 5th Conference of the European Chapter of the Association for Computational Linguistics*.

Jonathan David Bobaljik. 1999. Adverbs: The hierarchy paradox. *Glot International*, 4.

Joan Feigenbaum, Jeremy A. Kahn, and Carsten Lund. 1993. Complexity results for pomset languages. *SIAM Journal of Discrete Mathematics*, 6(3):432–442.

Jay Gischer. 1981. Shuffle languages, Petri nets, and context-sensitive grammars. *Communications of the ACM*, 24(9):597–605, September.

Thilo Goetz and Gerald Penn. 2000. A proposed linear specification language. Technical Report 134, Arbeitspapiere des SFB 340.

A. Lecomte and C. Retore. 1995. Pomset logic as an alternative categorial grammar. In Glyn Morrill and Richard Oehrle, editors, *Formal Grammar*, pages 181–196.

K. Lodaya and P. Weil. 2000. Series-parallel languages and the bounded-width property. *Theoretical Computer Science*, 237(1–2):347–380.

Gereon Müller. 2000. Shape conservation and remnant movement. In *Proceedings of NELS 30*.

Mark-Jan Nederhof and Giorgio Satta. 2004. IDL-expressions: A formalism for representing and parsing finite languages in natural language processing. *Journal of Artificial Intelligence Research*, 21:287–317.

Mark-Jan Nederhof, Giorgio Satta, and Stuart M. Shieber. 2003. Partially ordered multiset context-free grammars and ID/LP parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies*, pages 171–182, Nancy, France, April.

Vaughan R. Pratt. 1985. The pomset model of parallel processes : Unifying the temporal and the spatial. Technical report, Stanford University, January.

Vaughan R. Pratt. 1986. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context free grammars. *Theoretical Computer Science*, 88:191–229.

Edward P. Stabler. 1996. Derivational minimalism. In Christian Retoré, editor, *LACL*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer.

K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the ACL*, pages 104–111, Stanford, CA.

Manfred K. Warmuth and David Haussler. 1984. On the complexity of iterated shuffle. *J. Comput. Syst. Sci.*, 28(3):345–358.