Isolating Cross-linguistic Parsing Complexity with a Principles-and-Parameters Parser: A Case Study of Japanese and English *

Sandiway Fong & Robert C. Berwick
NEC Research Institute, 4 Independence Way, Princeton, NJ, 08540 USA, sandiway@research.nec.com
Rm 838, MIT AI Laboratory, 545 Technology Sq., Cambridge, MA 02139, berwick@ai.mit.edu

1 Introduction

As parsing models and linguistic theories have broadened to encompass a wider range of non-English languages, a particularly useful "stress test" is to build a single theory/parser pair that can work for multiple languages, in the best case with minor variation, perhaps restricted to the lexicon. This paper reports on the results of just such a test applied to a fully operational (Prolog) implementation of a so-called principles-and-parameters model of syntax, for the case of Japanese and English. This paper has two basic aims: (1) to show how an implemented model for an entire principles-and-parameters model (essentially all of the linguistic theory in Lasnik & Uriagereka (1988)), see figure 2 for a computer snapshot, leads directly to both a parser for multiple languages and a useful "computational linguistics workbench" in which one can easily experiment with alternative linguistic theoretical formulations of grammatical principles as well as alternative computational strategies; (2) to use this system to uncover sources of parsing complexity in Japanese as opposed to English. In particular, we examine the "null hypothesis" that a single parsing design suffices for efficient processing of both Head-first and Head-final languages, in contrast to approaches that posit, e.g., a right-corner or other mirror-image strategy for parsing Japanese as compared to English (e.g., BUP; Mazuka (1990)). In this case we can confirm computationally and precisely, in accordance with much current psycholinguisitic work (Frazier and Raynert (1988); Inoue and J.D. Fodor (1991); Nagai (1991)) that it is *not* the Head-final character of Japanese that results in processing difficulty so much as the possibility of scrambling and free deletion of NPs (so-called "super Pro Drop"). We do this by empirically investigating the effects of 3 possible "optimizations" of the parsing system for Japanese: (1) the use of right-context information via automatic source transformations, using a programming language compiler technique to introduce dummy nonterminals and corresponding semantic actions; (2) modification of the Japanese grammar to put the specifier of CP ($= \bar{S}$) on the right and so eliminate unnecessary center-embedding; and (3) eliminating of scrambling and NP drop to isolate the separate effects of Head-final (e.g., Verb-final) phrase structure in Japanese.

By explicit construction, the implementation demonstrates that it is possible to build an efficient principle-and-parameters parser for multiple languages, using 25 principles that are expressed in a language quite close in form to that of the original linguistic theory. The English-Japanese differences handled include the basic Subject-Object-Verb (SOV) order of Japanese; free "scrambling" of Japanese noun phrases; topic-comment structure; nonappearance of noun phrases that are discourse recoverable; and lack of *wh*-word movement in Japanese questions. No rule reprogramming is required to accommodate these differences, but changes to only 4 binary switches and a minimally distinct lexicon with different thematic grids in some cases. The parser couples several already-known parsing design strategies to obtain efficient parsing times, e.g., type-checking; multiple-entry canonical LR(1) parsing; and automatic (source-to-source) grammar transformations.[1]

2 Principle-based parsing

In a principle-based parser, construction- and language-specific rules are replaced with broader principles that remain invariant aside from parametric variation (see below). The parser works by a (partially interleaved) generate-and-test technique that uses a canonical LR(1) covering grammar (derived from $\overline{\text{X}}$ theory plus the theory of movement) to first build an initial set of tree structures; these structures are then run through a se-

*This research has been supported by NSF Grant DCR85552543 under a Presidential Young Investigator Award to Professor Robert C. Berwick, and a grant from the Kapor Family Foundation. We would like to thank Howard Lasnik, Alec Marantz, Shigeru Miyagawa, David Pesetsky, and Mamoro Saito for valuable discussions and valiant attempts to tell us about Japanese.

[1] To the best of our knowledge, this system is the first and broadest-coverage of its type to be able to parse Japanese and English by setting just a few parameter switches. Dorr (1987), under the supervision of the second author, developed a conceptually similar scheme to handle English, Spanish, and German. However, Dorr's system did not have the same broad coverage of English; did not handle Japanese; used hand rather than automatic compiling; and was approximately 15 times slower. Gunji's (1987) Japanese unification grammar comes closest to the principle-based model, but requires hand-modification from a set of core principles and does not really accommodate the important Japanese phenomenon of scrambling; see below. Other such systems work only on much smaller parts of English, e.g., Sharp (1985); Wehrli (1987); Crocker (1989); Correa (1988); Johnson, (1989); or are not in fact parsers, but proof-checkers, e.g., Stabler, (1991, forthcoming).

ries of predicates whose conjunction defines the remainder of the constraints the (sentence, phrase structure, LF) triple must satisfy. This is done using familiar machinery from Prolog to output LFs that satisfy all the declarative constraints of the linguistic theory. In practice, a straightforward generate-and-test mechanism is grossly inefficient, since the principles that apply to at the level of surface structure (S-structure) are but a fraction of those that apply in the overall system. The usual problems of lexical and structural ambiguity the the underconstrained nature of the initial $\overline{X}$ system means that the number of possible S-structures to hypothesize may be huge. To obtain an efficient parser we use a full multiple-entry table with backtracking (as in Tomita, 1986), extending it to a *canonical* LR(1) parser. The LR machine uses an automatically-built S-structure grammar that folds together enough of the constraints from other principles, parameters, lexical subcategory information offline to produce a 25-fold improvement over the online phrase structure recovery procedure originally proposed by Fong and Berwick (1989). Optimizations include extra conditions in action clauses to permit interleaving of other principles (like movement) with structure-building (the 'interleaving' noted by principles marked 'I' in the snapshot in figure 2 below); control structure flexibility in principle ordering; precomputation of the LR transition function; elimination of infinite recursion of empty elements by an additional stack mechanism, and so forth. We exploit the explicit modularity of the principle-based system in way that is impossible in an ordinary rule-based system: we can build a grammar for phrase structure that is small enough to make full, canonical LR(1) parsing usable, unlike large CFGs. The earlier error detection of full LR(1) parsing over LALR methods means that fail as early as possible, to avoid expensive tree constructions that can never participate in final solutions.[2]

3 The Japanese parser

We begin with a very simple parameterization of Japanese that will nonetheless be able to cover all the Lasnik and Saito *wh*-questions, scrambling, and so forth; see the table on the next page that follows the example sentences. The important point is that very little additional must be said in order to parse a wide variety of distinctive Japanese sentences; the principles as shown on the righthand side of the computer snapshot do *not* change. [3]

Consider first the example *wh*-movement sentences found in the linguistics paper *On the Nature of Proper Government* by Lasnik & Saito (1984).[4] These sentences (listed below) display many familiar typological Japanese–English differences, and cover a rather sophisticated set of differences between English and Japanese: for instance, why (6) is fine in Japanese but not in English; free omission of NPs; "scrambling" of subjects and objects; Verb-final (more generally, Head-final) constituent structure, and no overt movement of *wh*-phrases. We also consider a different set of Japanese sentences (also listed below) designed to illustrate a range of the same phenomena, taken from Hosokawa (1990). We stress that these sentences are designed to illustrate a range of sentence distinctions in Japanese, as well as our investigative method, rather than serve as any complete list of syntactic differences between the two languages (since they are obviously not).[5]

[Lasnik & Saito (1984)]

(2) Watashi-wa Taro-ga nani-o katta ka shitte iru
'I know what John bought'

(6) Kimi-wa dare-ni Taro-ga naze kubi-ni natta tte itta no
'To whom did you say that John was fired why'

(32) *Meari-wa Taro-ga nani-o katta ka do ka shiranai
'Mary does not know whether or not John bought what'

(37a) Taro-wa naze kubi-ni natta no
'Why was John fired'

(37b) Biru-wa Taro-ga naze kubi-ni natta tte itta no
'Why did Bill say that John was fired'

(39a) Taro-ga nani-o te-ni ireta koto-o sonnani okotteru no
'What are you so angry about the fact that Taro obtained'

(39b) *Taro-ga naze sore-o te-ni ireta koto-o sonnani okotteru no
'Why are you so angry about the fact that Taro obtained it'

(41a) Hanoko-ga Taro-ga nani-o te-ni ireta tte itta koto-o sonnani okotteru no
'What are you so angry about the fact that Hanoko said that Taro obtained'

(41b) *Hanoko-ga Taro-ga naze sore-o te-ni ireta tte itta koto-o sonnani okotteru no
'Why are you so angry about the fact that Hanoko said that Taro obtained it'

(60) Kimi-wa nani-o doko-de katta no
'Where did you buy what'

(63) Kimi-wa nani-o sagashiteru no
'Why are you looking for what'

Complement/noncomplement asymmetry, scrambling and unexpected parses

To see how the parser handles one Japanese example (see the actual computer output in figure 1 or figure 2), consider (39a) (and the corresponding illicit (39b)), where a complement *wh* but not a noncomplement *wh* can be extracted from a complex NP: (a) Taro-ga *nani-o* te-ni ireta koto-o sonnani okotteru no; (b) *Taro-ga *naze* sore-o te-ni ireta koto-o 'What/*Why are you so angry about the fact that Taro obtained'

This example illustrates several Japanese typological differences with English. The subject of the matrix clause (= *you*) has been omitted. *Nani* ('what') and *te* ('hand') have been scrambled; the direct object

[2] To provide a rough measure of the machine size for the phrase structure grammar of S-structure for both English and Japanese, the augmented CFG consists of about 74 productions derived from a schema of 30-34 rules. The resulting characteristic finite state automaton (CFSM) consists of 123 states with 550 transitions between the various states. The action table consists of a total of 984 individual (nonerror) entries.

[3] We will scramble only from direct object positions here, even though it is straightforward to scramble from indirect object positions. Informally, we have noted that scrambling from IO greatly increases computation time. A tighter set of constraints on scrambling seems called for.

[4] Average best parsing time for the Japanese sentences shown is 0.37secs/word on a Symbolics 3650 (64K LIPS) (σ = 1.52sec, n= 100). Parsing time on a Sun Sparcstation 2 is approximately an order of magnitude faster.

[5] E.g., the double-*o* constraint; case-overwriting, passive and causative constructions, etc. all remain to be fully implemented.

(marked *-o*) now appearing in front of the indirect object *te*. Phrase structure is Head final. Our relaxation of the Case Adjacency parameter and the rule that allows adjunction of NP to VP, plus transmission of Case to the scrambled NP will let this analysis through. The LF for this sentence should be something along the lines of: for what x, *pro* is so angry about [the fact that Taro obtained x]

In this example *pro* denotes the understood subject of *okotteru* ("be angry"). The LFs actually returned by the parser are shown in the snapshot in figure 1.[6]

[Hosokawa (1990)]

(1b)′ Gengogaku-no gakusei-ga tiizu-o tabeta
linguistics-gen student-nom cheese-acc eat-past
'A student of linguistics ate cheese'

(2b)′ Nagai kami-no gakusei-ga tiizu-o tabeta
long hair-gen student-nom cheese-acc eat-past
'A long haired student ate cheese'

(3b) Taro-ga hon-o katta
John-nom book-acc buy-past
'John bought a book'

(4b) Taro-ga Hanoko-ni hon-o ageta
John-nom Mary-dat book-acc give-past
'John bought Mary a book'

(5b) Taro-ga hon-o table-no ue-ni oita
John book-acc table-gen top-dat (top of table) put-past
'John put the book on the table'

(6b) Taro-wa gakkoo-ni itta
John-top school-dat go-past
'John went to school'

(15b) Watashi-wa taro-ga nani-o katta ka shiranai
I-top John-nom what-acc bought Q know-not
'I don't know what John bought'

(17b) Taro-wa Chomsky-no *Barriers*-o yomimashita ka
John-top Chomsky-gen *Barriers*-acc read-past Q
'Did John read Chomsky's *Barriers*'

(18b) Hanoko-wa
Biru-ga Chomsky-no *Barriers*-o yonda ka do ka shiranai
Mary-top Bill-nom Chomsky-gen *Barriers*-acc read Q know-not
'Mary does not know whether or not Bill read Chomsky's...'

The parametric differences that we need to accomodate all these differences between English and Japanese are quite few:

English and Japanese parameter settings

Parameter	English	Japanese
Spec order	specInitial.	specInitial.
	specFinal :- \+ specInitial.	specFinal :- \+ specInitial.
*Head order	headInitial.	headFinal.
	headFinal :- \+ headInitial.	headInitial :- \+ headFinal.
Agreement	agr(weak).	agr(weak).
Bounding	boundingNode(i2). boundingNode(np).	boundingNode(i2). boundingNode(np).
*Case Adjacency	caseAdjacency.	:- no caseAdjacency.
*Wh in Syntax	whInSyntax.	:-no whInSyntax.
*Pro-Drop	:- no proDrop.	proDrop.

As one can see from the figure, the system does correctly recover the right LF, as the last one in snapshot. However, it also (surprisingly) discovers three *additional* LFs, illustrating the power of the system to uncover alternative interpretations that a proper theory of context would have the job of ruling out. Ignoring indices, they all have the same form: for what x, Taro is so angry about [the fact that *pro* obtained x]

Here the embedded subject *Taro* has been interchanged with the matrix subject *pro*. It turns out that the sentence happens to be ambiguous with respect to the two basic interpretations.[7] For completeness, here are the three variants of that correspond to the first three LFs reported by the parser. S. Miyagawa (p.c.) informs us that the last two, given proper context, are in fact possible. These include: (1) *pro* is coreferent with *koto* ("fact"):[8], i.e., for what x, Taro is so angry about [the fact that the fact obtained x]; (2) *pro* is coreferent with *taro*: for what x, Taro is so angry about [the fact that Taro obtained x]; and (3) *pro* is free in the sentence: for what x, Taro is so angry about [the fact that (someone else) obtained x].[9]

4 Parsing Japanese: the computational effects of scrambling, *pro*-drop, and phrase structure

Next we turn to the investigation of the *computational* differences between the two languages that we have explored, and show how to use the system in an exploratory mode to discover complexity differences between English and Japanese. In the discussion that follows, we shall need to draw on comparisons between the complexity of different parses. While this is a delicate matter, there are two obvious metrics to use in comparing this parser's complexity. The first is the total number of principle operations used to analyze a sentence—the number of S-structures, chain formations, indexings, the case filter and other constraint applications, etc. We can treat these individually and as a whole to give an account of the entire "search space" the parser moves through to discover analyses. However, this is

[6] We will not have room to describe in detail the derivation of these LFs. But, it should be noted that the derivation sequence is quite complex. Note, for example, that *nani* ('what') undergoes movement at two levels of phrase structure in order to get to the specifier position of the matrix Complementizer:

[*CP* nani[*IP* Taro[*NP*[*CP* *pro*[*VP* *t*′[*VP* te *t* ireta]]] koto]...]]

Furthermore, the LF trace *t*′ violates the so-called empty category principle unless it is deleted (as indicated by [] in the snapshot), under the present theory. The lack of *wh*-movement at S-structure in Japanese, and its presence in English, interacts with these constraints to bar examples like (6) in English; see Lasnik & Saito.

[7] This was pointed out by D. Pesetsky, and confirmed by M. Saito. However, presumably the use of *wa* rather than *ga* and intonational pauses could be exploited as a surface cue to rule out more generally ambiguity in this example and others like it. See Fong and Berwick (1989) for a discussion of how to integrate surface cues into the principle-based system.

[8] This interpretation can be eliminated by imposing selectional restrictions on the possible "agents" of *okotteru* (let us say that they must be animate).

[9] Having a parsing system that can recover *all* such linguistic alternatives is of interest in its own right, both to verify and correct the linguistic theory, as well as ensure that no possibilities are overlooked by human interpreters.

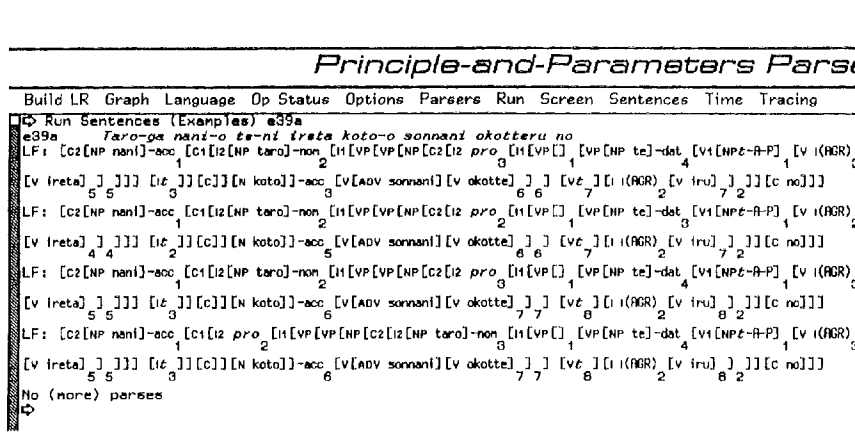

Figure 1: Computer snapshot from Lasnik & Saito.

often not a good measure of the total time spent in analysis. The second measure we use is more particular and precisely tailored to the specific backtracking-LR design we have built to recover structural descriptions: we can count the total number of LR finite-state control steps taken in recovering the S-structure(s) for a given sentence; indeed, this accounts for the bulk of parsing time for those cases, as in Japanese and many English sentences, where multiple quasi-S-structures are returned. Taken together, these two measures provide both a coarse and a more fine-grained way of seeing what is hard or easy to compute.[10]

5 Complexity of Japanese parsing

Given this initial set of analyses, let us now examine the complexity of Japanese sentence processing as compared to English. To do this, we initially examined sentences that we thought would highlight the ease of Japanese relative to English, namely, the "classic" English center-embedded vs. Japanese left-branching constructs from Kuno (1973), e.g., *The cheese the rat the cat John keeps killed*, =*Taro-ga katte-iru neko-ga korosita nezumi-ga*

On the conventional Chomsky-Miller account, the English construction is very difficult to parse, while the left-branching Japanese form is completely understandable. Interestingly, as shown in figure 2 the number of operations required to complete this parse correctly is enormous, as one can see from the righthand column numbers that show the structures that are passed into and out of each principle module.

It at first appears that left-branching structures are definitely *not* simpler than the corresponding center-embedded examples. Why should this be? On a modern analysis such as the one adopted here, recall that restrictive relative clauses, e.g. *the rat the cat killed*, are open sentences, and so contain an operator-variable structure coindexed with *the rat*, roughly:

(1) [*NP*[*NP* the rat]$_1$ [*CP* Op$_1$ ... the cat killed t_1]]

[10] Note that these two are metrics that are stable across compile-cycles and different platforms. This would be not true, of course, for simple parse times — the obvious alternative.

where the empty operator (Op) is base-generated in an A-position and subsequently fronted by Move-α (Chomsky, 1986:86).

Thus, the Japanese structures *are* center-embedded after all—the parser places a potentially arbitrary string of empty Operators at the front of the sentence. Perhaps, then, the *formal* accounts of why this sentence should be easy are incorrect; it is formally difficult but easy on other grounds. Of course, alternatively, the theory or parsing model could be incorrect, or perhaps it is scrambling, or *pro*-drop, or the Head-final character of the language makes such sentences difficult. In the rest of this paper we focus on 3 attempts to discover the source of the complexity.

To investigate these questions, we embarked on a series of optimization efforts that focused on the Spec positions of CP and the Head-final character of the language, with the goal of making the Japanese as easy, or easier than, the corresponding English sentences or determining why we could not make it easier. In all, we conducted three empirical tests: (1) using dummy nonterminals to "lift" information from the verb to the VP node, to test the Head-first/final hypothesis; (2) placing Spec of CP on the left rather than the right, to test the center-embedding hypothesis; and (3) building a "restricted" pseudo-Japanese that eliminated scrambling and free *pro*-drop, while *not* lifting the information up and to the left, leaving the Head-final character intact. We will next cover each computer experiment in turn. Figure 3 gives a bar-graph summary of the three experimental results in the form of times improvement (reduction) in LR state creation.

Optimization 1: Head-final information

Our first optimization centers on the Head-final phrase structure of Japanese. With Heads at the end, valuable information (subcategorization, etc.) may be unavailable at the time the parser is to make a particular decision. However, for our LR machine, there is a well-known programming language optimization: introduce dummy nonterminals on the left of a real nonterminal, e.g., VP→ X V NP, which, when reduced, call semantic action routines that can check the input stream for a particular property (say, the presence of a noun arbitrarily far to the right). Specifically, if verb

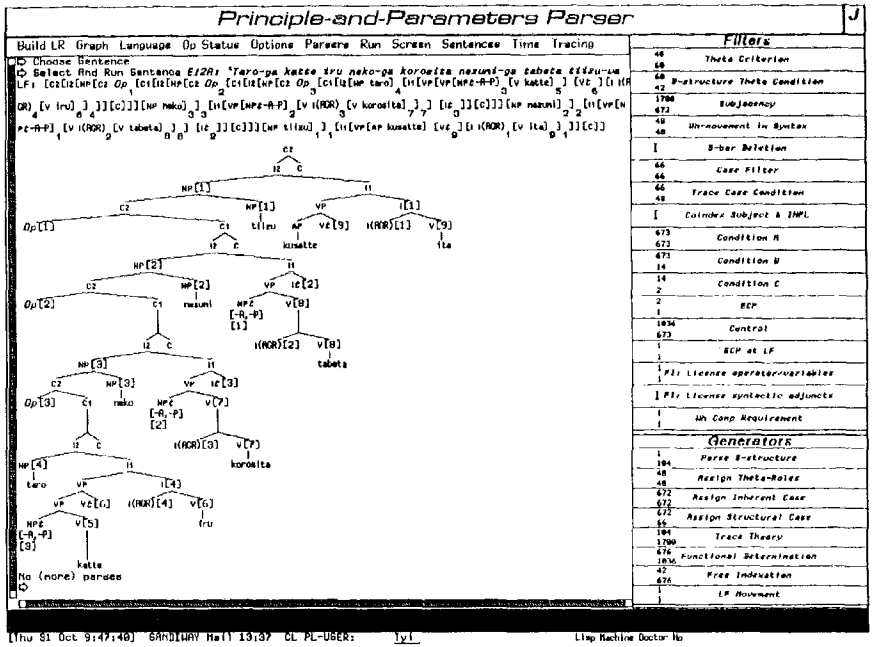

Figure 2: The parse of the Japanese counterpart of the English center-embedded question. Tracing out the left-hand fringe of the tree, note the string of empty operators, as well as, on the right-hand column, the large number of parser operations required to build this single correct LF as compared to English (in the text). Still, a single parse is correctly returned.

information occurs on the right we can offline "lift" that information up to the VP node, where it can then influence the LR state transitions that are made when examining material to the left of the head. For example, for each V subcategory, the LR machine will contain in effect a a new LR state; the system will add a command to look as far into the input as needed to determine whether to branch to this new state or another V subcategory state. This is precisely the mechanism we used to determine whether to insert an empty category or not in a Head-first language. For instance, in Japanese relative clauses this is of importance because the parser may get valuable information from the verb to determine whether a preceding NP belongs to that relative clause or not. the action and transition tables of the resulting Japanese machine, which we will call "optimized," will be far larger than its base case counterpart (more precisely: the action table is 3 times larger, or about 380K to 980K, while the transition table is about twice as large, 72K to 142K).

The advantages accrued by this optimization are substantial, 2-10 times better; see the table below. (This also holds across other sentences; see the bar graph summary at the end of the paper.) The unoptimized number of LR state transitions grows astonishingly rapidly. For example, the transitions needed to parse ce4 is exactly as shown—over 20 million of them, compared to 1 million for the optimized version.[11]

Sentences:
ce1. The cheese was rotten;
ce2. The cheese the rat ate was rotten;
ce3. The cheese the rat the cat killed ate was rotten.
ce4. The cheese the rat the cat John keeps killed ate was rotten. (*=12a on snapshot*)
(See figure 2 for computer output of the corresponding Japanese sentence.)

Total number of LR state transitions

Sentence	Jp, Unopt.	Jp, Opt.	Times better(E.)	English
ce1	232	122	1.9(6.1)	745
ce2	7122	1518	4.7(1.6)	2431
ce3	257,042	25,246	10.18(.19)	4979
ce4	20,360,664	966,114	21.07(.03)	32101

The same basic trend also holds, though not as strongly, when we look at these and other sentences in terms of the total number of principle operations required; while we do not have space to review all of these here, as an example, sentence (15b) takes 4126 operations in the base case, and 455 when optimized in this fashion; while ce3 takes 1280 operations and 667 when optimized, respectively.

[11] We should point out that in all cases, about a two-thirds of these transitions occur before the LR machine reaches a point in the search space where the solutions are "clustered" enough that the remaining solutions do not take so much effort.

Optimization 2: Spec of CP on the right

A second obvious strategy is to remove the center-embedding itself. Here there is a grammatical move we can make. Evidently, in Japanese the only elements that appear in Spec of CP are put there by LF movement. Thus, these elements can never be visible in this position on the surface. If this is so, then there is really nothing to prevent us from placing just the Spec of CP on the right, rather than the left. This is an example of the "testbed" property of the system; this change takes two lines of code. Given this change, the resulting structures will have their Operators on the right, rather than the left, and will *not* be center-embedded. In addition, in this test the parser will *not* take advantage of right-hand information, thus eliminating this as a possible source of speedup.

Parsing complexity is reduced by this move, by a factor of just about one-half, if one considers either LR state transitions or principle operations; not as good as the first optimization; see below for some representative results. Also, with the most deeply center-embedded sentence the total number of principle operations actually is worse than in the base case. Evidently we have not located the source of the parser's problems in center-embedding alone.

Complexity for Spec on the right

Sentence	LR trans.	Total ops
ce1	122	32
ce2	4930	97
ce3	209,980	721
ce4	16,290,667	12605

Optimization 3: Factoring out the effects of scrambling and *pro*-drop

While it appears that Head-final information helps the most, we must also remember that part of the complexity of Japanese is the result of free scrambling and *pro*-drop. To factor apart these effects, we ran a series of computer experiments on a quasi-Japanese grammar, J*, that was just like Japanese except scrambling and *pro*-drop were barred. The changes were again simple to make: one change was automatic, just turning off a parameter value, while the second involved 3 lines of hand-coding in the $\overline{X}$ schemas to force the system to look for a lexical NP in DO (and IO) positions Further, we did *not* optimize for right-hand information (so that the Head-final character was left intact). Of course, we now can no longer parse sentences with scrambled direct objects.

The table below shows the results. This was the *best* optimization of all. Without scrambling, and hence no movement at all compared to English, the Head-final quasi-Japanese was for the most part parsed 5-10 times more efficiently than English, and at worst (for the triply-embedded sentence) with three times fewer LR transitions and only about 30% more principle operations than English. Thus, this was even more efficient than the righthand information optimized Japanese parser. (The first column gives the number of LR transitions and the second gives the total number of principle operations for this "no scramble/drop" version, while the last two columns give the same information for English.)

No scrambling/drop vs. English

Sentence	LR trans.	No. ops	Eng. LR	Eng. ops
ce1	93	32	745	109
ce2	274	88	2431	168
ce3	1241	445	4979	558
ce4	8241	3719	21,074	2874

As before, with a short sentence, there is little difference between optimization methods, but over a range of sentences and with longer sentences, the no-scramble or *pro*-drop optimization works better than any other. Evidently, given the framework of assumptions we have made, the Head-final character of Japanese does not hurt the most; rather, it is scrambling and *pro*-drop that does, since if we remove these latter two effects we get the biggest improvement in parsing efficiency. We can confirm this by looking at the LR transitions for the other sentences (1b)–(18b) across methods, summarizing our tests. We can summarize the three experiments across sentences in figure 3.

Summary of complexity across tests

Sentence	Unopt.	Opt.	Spec-Final	No Scramble/drop
1b	1196	730	602	216
2b	1894	790	957	298
3b	345	289	185	103
4b	601	422	307*	149
5b	1742	1051	878*	370
6b	526	377	267	138
15b	32,955	19,998	11,205	1681
17b	1287	1789	685	272
18b	210,036	84,727	43,745	5306

6 Conclusions

Given our limited set of test sentences, our results must be correspondingly tentative. Nonetheless, we can draw several initial conclusions:

- One *can* parse Japanese by parametrically varying a grammar, much as expected. The limits of the method are theory-bound: we can accommodate just as much as we understand about Japanese syntax.
- Attempting to parse more than one language with the same grammar and parser can quickly reveal what is wrong with one's theory for either language. In our case, we discovered omissions in the implementation relating to Case transmission, the Wh-Comp Requirement, and trace deletion, among other items.
- A single parser suffices for very distinct languages. The grammar is parameterized, but not the parser, confirming much recent other research in Japanese sentence processing cited in the introduction. Japanese at first appears much more complex to parse than corresponding English sentences. We suggest, tentatively, that complexity is introduced by scrambling and omission of NPs, rather than Head-final properties. Unoptimized, the system is too slow. Some efficiency is obtained if one can "lift" information from the right for use in parsing with an LR machine. From a heuristic standpoint, this suggests that strategies limiting what may appear in a scrambled position or dropped in a certain context will aid such an LR-based device more than switching to a parser based presumably geared for a different branching direction.
- The principle-based system affords a new and generally straightforward way to precisely explore different grammatical theories, structural assumptions, and parsing methods and their computational consequences

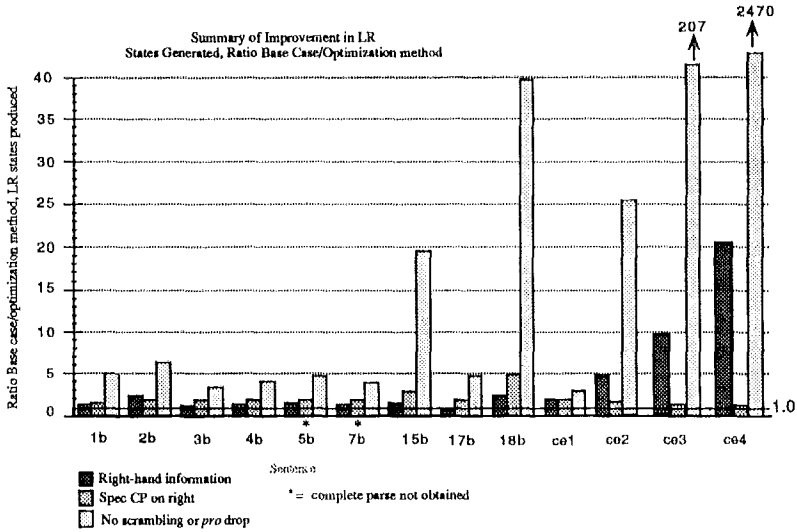

Figure 3: A bar graph showing the improvement in total LR transitions when parsing Japanese examples 1b–18b, and ce1–ce4, compared against the original base case unoptimized parser, across the 3 experiments described here. The horizontal line drawn at 1.0 indicates improvement over the base case.

in a precise way, without extensive hand coding. All of the experiments we tried took no more than a few lines of modification. Of course, the difficult part is to come up with a universal set of principles in the first place—so that in fact, English looks just about like Japanese, and vice-versa.

7 References

Baltin, M.R., and A.S. Kroch (eds.), 1989. *Alternative Conceptions of Phrase Structure*, The University of Chicago Press.

Chomsky, N.A., 1986. *Knowledge of Language: Its Nature, Origin, and Use.* Prager.

Correa, N., 1988. Syntactic Analysis of English with respect to Government-binding Grammar, Ph.D. dissertation, Syracuse University.

Crocker, M.W., 1989. *A Principle-Based System for Syntactic Analysis, (m.s.).*

Dorr, B.J., 1987. UNITRAN: A Principle-Based Approach to Machine Translation. S.M. thesis. MIT Department of Electrical Engineering and Computer Science.

Fong, S. & R.C. Berwick, 1989. The computational implementation of principle-based parsers. *International Workshop on Parsing Technologies.*, Carnegie Mellon University, in M. Tomita (ed) *Current Issues in Parsing Technologies*, Kluwer.

Fong, S., 1991. Computational Properties of Principle-based Grammatical Theories. Ph.D., dissertation, MIT Department of Computer Science and Electrical Engineering.

Frazier, L., and K. Rayner, 1988. Parameterizing the language processing system: Left- vs. right-branching within and across languages. In J. Hawkins (ed.) *Explaining Language Universals*, Basil Blackwell, Oxford, pp. 247–279.

Hosokawa, H. 1991. Syntactic differences between English and Japanese. *Georgetown Journal of Languages and Linguistics*, 1:4, 401–414.

Inoue, N. and Fodor, J.D., 1991. Information-paced processing of Japanese sentences. Paper presented at the International Workshop on Japanese Syntactic Processing, Duke University.

Johnson, M., 1989. Use of the Knowledge of Language, *Journal of Psycholinguistic Research.* 18(1).

Kuno, S., 1973. *The Structure of the Japanese Language*, Cambridge, MA: MIT Press.

Lasnik, H. & M. Saito, 1984. On the nature of proper government. *Linguistic Inquiry*, 15:2.

Lasnik, H. & J. Uriagereka, 1988. *A Course in GB Syntax: Lectures on Binding and Empty Categories*, Cambridge, MA: MIT Press.

Mazuka, R., 1990. Processing of empty categories in Japanese. Manuscript, Duke University.

Nagai, N., 1991. Paper presented at the International Conference on Japanese Syntactic Processing, Duke University.

Sharp, R.M., 1985. A Model of Grammar Based on Principles of Government and Binding Theory. M.S. thesis. Department of Computer Science. University of British Columbia.

Stabler, E.P., Jr., 1991 forthcoming. *The Logical Approach to Syntax: Foundations, Specifications and Implementations of Theories of Government and Binding.*, Cambridge, MA: MIT Press.

Tomita, M., 1986. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems.* Kluwer.

Wehrli, E., 1986. *A Government-Binding Parser for French.* Working Paper No. 48. University of Geneva.