Word Segmentation, Unknown-word Resolution, and Morphological Agreement in a Hebrew Parsing System

Yoav Goldberg*
Ben Gurion University of the Negev

Michael Elhadad**
Ben Gurion University of the Negev

We present a constituency parsing system for Modern Hebrew. The system is based on the PCFG-LA parsing method of Petrov et al. (2006), which is extended in various ways in order to accommodate the specificities of Hebrew as a morphologically rich language with a small treebank. We show that parsing performance can be enhanced by utilizing a language resource external to the treebank, specifically, a lexicon-based morphological analyzer. We present a computational model of interfacing the external lexicon and a treebank-based parser, also in the common case where the lexicon and the treebank follow different annotation schemes. We show that Hebrew word-segmentation and constituency-parsing can be performed jointly using CKY lattice parsing. Performing the tasks jointly is effective, and substantially outperforms a pipeline-based model. We suggest modeling grammatical agreement in a constituency-based parser as a filter mechanism that is orthogonal to the grammar, and present a concrete implementation of the method. Although the constituency parser does not make many agreement mistakes to begin with, the filter mechanism is effective in fixing the agreement mistakes that the parser does make.

These contributions extend outside of the scope of Hebrew processing, and are of general applicability to the NLP community. Hebrew is a specific case of a morphologically rich language, and ideas presented in this work are useful also for processing other languages, including English. The lattice-based parsing methodology is useful in any case where the input is uncertain. Extending the lexical coverage of a treebank-derived parser using an external lexicon is relevant for any language with a small treebank.

1. Introduction

Different languages have different syntactic properties. In English, word order is relatively fixed, whereas in other languages word order is much more flexible (in Hebrew, the subject may appear either before or after a verb). In languages with a flexible word order, the meaning of the sentence is realized using other structural elements, like word

* Computer Science Department, Ben Gurion University of the Negev, Israel.
E-mail: yoav.goldberg@gmail.com.
** Computer Science Department, Ben Gurion University of the Negev, Israel.
E-mail: elhadad@cs.bgu.ac.il.

Submission received: 30 September 2011; revised submission received: 19 May 2012; accepted for publication: 3 August 2012.

inflections or markers, which are referred to as **morphology** (in Hebrew, the marker את is used to mark definite objects, distinguishing them from subjects in the same position. In addition, verbs and nouns are marked for gender and number, and subject and verb must share the same gender and number). A limited form of morphology also exists in English: the *-s* and *-ed* suffixes are examples of English morphological markings. In other languages, morphological processes may be much more involved. The lexical units (words) in English are always separated by white space. In Chinese, such separation is not available. In Hebrew (and Arabic), most words are separated by white space, but many of the function words (determiners like *the*, conjunctions such as *and*, and prepositions like *in* or *of*) do not stand on their own but are instead attached to the following words.

A large part of the parsing literature is devoted to automatic parsing of English, a language with a relatively simple morphology, relatively fixed word order, and a large treebank. Data-driven English parsing is now at the state where naturally occurring text in the news domain can be automatically parsed with accuracies of around 90% (according to standard parsing evaluation measures). When moving from English to languages with richer morphologies and less-rigid word orders, however, the parsing algorithms developed for English exhibit a large drop in accuracy. In addition, whereas English has a large treebank, containing over one million annotated words, many other languages have much smaller treebanks, which also contribute to the drop in the accuracies of the data-driven parsers. A similar drop in parsing accuracy is also exhibited in English when moving from the news domain, on which parsers have traditionally been trained, to other genres such as prose, blogs, poetry, product reviews, or biomedical texts, which use different vocabularies and, to some extent, different syntactic rules.

This work focuses on constituency parsing of Modern Hebrew, a Semitic language with a rich and productive morphology, relatively free word order,[1] and a small treebank. Several natural questions arise: Can the small size of the treebank be compensated for using other available resources or sources of information? How should the word segmentation issue (that function words do not appear in isolation but attach to the next word, forming ambiguous letter patterns) be handled? Can morphological information be used effectively in order to improve parsing accuracy?

We present a system which is based on a state-of-the-art model for constituency parsing, namely, the probabilistic context-free grammar (PCFG) with latent annotations (PCFG-LA) model of Petrov et al. (2006), as implemented in the BerkeleyParser. After evaluating the out-of-the-box performance of the BerkeleyParser on the Hebrew treebank, we discuss some of its limitations and then go on to extend the PCFG-LA parsing model in several directions, making it more suitable for parsing Hebrew and related languages. Our extensions are based on the following themes.

Separation of lexical and syntactic knowledge. There are two kinds of knowledge inherent in a parsing system. One of them is **syntactic knowledge** governing the way in which words can be combined to form structures, which, in turn, can be combined to form ever larger structures. The other is **lexical knowledge** about the identities of individual words, the word classes they belong to, and the kinds of syntactic structures they can participate in. We argue that the amount of syntactic knowledge needed for a parsing system is relatively limited, and that sufficiently large parts of it can be captured also

1 To be more precise, in Hebrew the order of constituents is relatively free, whereas the order of the words within certain constituents is relatively fixed.

based on a relatively small treebank. Lexical knowledge, on the other hand, is much more vast, and we should not rely on a treebank (small or large) to provide adequate lexical coverage. Instead, we should aim to find ways of integrating lexical knowledge, which is external to the treebank, into the parsing process.

We extend the lexical coverage of a treebank-based parser using a dictionary-based morphological analyzer. We present a way of integrating the two resources also for the common case where their annotations schemes diverge. This method is very effective in improving parsing accuracy.

Encoding input uncertainty using a lattice-based representation. Sometimes, the language signal (the input to the parser) may be uncertain. This happens in Hebrew when a space-delimited token such as בצל can represent either a single word ('[an] onion') or a sequence of two words or three words ('in shadow' and 'in the shadow,' respectively). When computationally feasible, it is best to let the uncertainty be resolved by the parser rather than in a separate preprocessing step.

We propose encoding the input-uncertainty in a word lattice, and use lattice parsing (Chappelier et al. 1999; Hall 2005) to perform joint word segmentation and syntactic disambiguation (Cohen and Smith 2007; Goldberg and Tsarfaty 2008). Performing the tasks jointly is effective, and substantially outperforms a pipeline-based model.

Using morphological information to improve parsing accuracy. Morphology provides useful hints for resolving syntactic ambiguity, and the parsing model should have a way of utilizing these hints. There is a range of morphological hints than can be utilized: from functional marking elements (such as the את marker indicating a definite direct object); to elements marking syntactic properties such as definiteness (such as the Hebrew ה marker); to agreement patterns requiring a compatibility in properties such as gender, number, and person between syntactic constituents (such as a verb and its subject or an adjective and the noun it modifies).

We suggest modeling agreement as a filtering process that is orthogonal to the grammar. Although the constituency parser does not make many agreement mistakes to begin with, the filter mechanism is effective in fixing the agreement mistakes that the parser does make, without introducing new mistakes.

Aspects of the work presented in this article are discussed in earlier publications. Goldberg and Tsarfaty (2008) suggest the lattice-parsing mechanism, Goldberg et al. (2009) discuss ways of interfacing a treebank-derived PCFG-parser with an external lexicon, and Goldberg and Elhadad (2011) present experiments using the PCFG-LA BerkeleyParser. Here we provide a cohesive presentation of the entire system, as well as a more detailed description and an expanded evaluation. We also extend the previous work in several dimensions: We introduce a new method of interfacing the parser and the external lexicon, which contributes to an improved parsing accuracy, and suggest incorporating agreement information as a filter.

The methodologies we suggest extend outside the scope of Hebrew processing, and are of general applicability to the NLP community. Hebrew is a specific case of a morphologically rich language, and ideas presented in this work are useful also for processing other languages, including English. The lattice-based parsing methodology is useful in any case where the input is uncertain. Indeed, we have used it to solve the problem of parsing while recovering null elements in both English and Chinese (Cai, Chiang, and Goldberg 2011), and others have used it for the joint segmentation and parsing of Arabic (Green and Manning 2010). Extending the lexical coverage of a treebank-derived parser using an external lexicon is relevant for any language with

a small treebank, and also for domain adaptation scenarios for English. Finally, the agreement-as-filter methodology is applicable to any morphologically rich language, and although its contribution to the parsing task may be limited, it is of wide applicability to syntactic generation tasks, such as target-side-syntax machine translation in a morphologically rich language.

2. Modern Hebrew

2.1 Lexical and Syntactic Properties

Some relevant lexical and syntactic properties of Modern Hebrew are highlighted in this section.

2.1.1 Unvocalized Orthography. Most vowels are not marked in everyday Hebrew text, which results in a very high level of lexical and morphological ambiguity. Some tokens can admit as many as 15 distinct readings, and the average number of possible morphological analyses per token in Hebrew text is 2.7, compared with 1.4 in English (Adler 2007). The word **כפיות** can be read in at least eight different ways ('spoons,' 'square cotton headkerchiefs,' 'coercions,' 'as mouths,' 'as spouts,' 'as fairies,' 'ungratefulness,' 'fun/adjective$_{\text{feminine,plural}}$'), the word **כתב** in at least six ways ('a journalist,' 'writing,' 'script,' 'wrote,' 'added someone as a recipient,' 'was added as a recipient') and the word **את** can be read as a very common case-marker (appearing before definite direct objects), a very common pronoun ('you/$_{\text{feminine}}$'), and a noun ('shovel').

2.1.2 Affixation. Eight common prepositions, conjunctions, and articles may never appear in isolation and must always be attached as prefixes to the following word.[2] These include the function words **מ** ('from'), **ש** ('which'/'who'/'that'), **כש** ('when'), **ה** ('the'), **ו** ('and'), **כ** ('like'), **ל** ('to'), and **ב** ('in'),. Several such elements may attach together, producing forms such as **ושמהשמש** (**ו־ש־מ־ה־שמש** 'and-that-from-the-sun'). Notice that when it appears by itself, the last part of the token, the noun **שמש** ('sun'), can also be interpreted as the sequence **ש־מש** ('who moved'). The linear order of such elements within a token is fixed (disallowing the reading **ו־ש־מ־ה־ש־מש** in the previous example).

The syntactic relations of these elements with respect to the rest of the sentence are rather free, however. The relativizer **ש** ('that'), for example, may attach to an arbitrarily long relative clause that goes beyond token boundaries. The attachment in such cases encompasses a long-distance dependency that cannot be captured by local-context (or Markovian) sequential processes that are typically used for morphological disambiguation. The same argument holds for resolving PP attachment of a prefixed preposition or marking conjunction of elements of any kind.

To further complicate matters, the definite article **ה** ('the') is not realized in writing when following the particles **ב** ('in'), **כ** ('like'), and **ל** ('to'). Thus, the form **בבית** can be interpreted as either **ב־בית** ('in house') or **ב־ה־בית** ('in the house').[3]

2 In what follows, we indicate the correct segmentations of the different forms. Naturally occurring Hebrew text does not have such indications.

3 This overt element is in fact indicated by vocalization, but is not realized in standard written text.

In addition, pronominal elements (clitics) may attach to nouns, verbs, adverbs, prepositions, and others as suffixes (e.g., **הביאן** [**הביא**+**הן**, 'brought-them'], **עליהם** [**הסעלי**+], 'on them']).

These affixations result in highly ambiguous token segmentations: **מספרו** ('[they] assigned numbers') vs. **מספרו** ('his number' or 'the one who cuts his hair') vs. **ומספר** ('from his book' or 'from his barber'), **הרכבת** ('putting together') vs. **הרכבת** ('the train'), and **בצל** ('an onion') vs. **בצל** ('in the shadow') are only a few examples of ambiguities that may arise. Quantitatively, 99,896 out of 567,483 forms (17%) in a wide-coverage lexicon of Hebrew can admit both segmented and unsegmented analyses.

In many cases the correct segmentation cannot be determined from local context alone, but can be disambiguated by more global syntactic constraints (in **ראיתי שמים כחולים**, the middle token is ambiguous between **שמים** ['sky'] and **ש**+**מים** ['that/$_{rel}$ water'], and the sequence can be interpreted as either 'I saw blue skies' or 'I saw that blue water.' On the other hand, **ראיתי שמים כחולים פרצו מן הבאר** is unambiguous because the past verb **פרצו** requires the relativizer **ש**, allowing only the segmented **ש**+**מים** reading 'I saw that blue water broke from the well'. In the other direction, **ראיתי שמים כחולים והלכתי לישון** is also unambiguous, allowing only the unsegmented reading 'I saw blue skies and went to sleep'.)

2.1.3 Rich Templatic Morphology. Hebrew words follow a complex morphological structure, which is based on a root + template system, with both derivational and inflectional elements. Word forms can encode gender, number, person, and tense, and in addition noun-compounding is also morphologically marked (see Section 2.1.7). Although the exact details of the system are irrelevant (but see Adler [2007] and Glinert [1989] for a good overview), we note that this word formation mechanism results in a very high number of possible word forms, and that it is hard to guess the part-of-speech of words based on prefixes and suffixes alone, a method frequently used in other languages.

2.1.4 The Participle Form. The Hebrew participle form (**בינוני**, literally the "middle form" of verbs) is a form that shares morphological and syntactic properties of nouns, verbs, and adjectives. This form causes many disagreements between human annotators, and large disagreement is found also between major Hebrew dictionaries regarding many word forms (see Adler et al. [2008b] for a discussion from tag set design and annotation guidelines, including many syntactic, semantic, and lexical considerations). For the purpose of this work, this form is of interest as it highlights the inherent ambiguity between adjectival, nominal, and verbal readings of many words, which are hard to disambiguate even in context.

2.1.5 Relatively Free Constituent Order. The ordering of constituents inside a phrase is relatively free. This is most notably apparent in verbal phrases and sentential levels. In particular, whereas most sentences follow a subject-verb-object order (SVO), OVS and VSO configurations are also possible (counting in the Hebrew Treebank reveals 5,720 SV cases and 2,613 VS cases, compared with 81,135 SV and 3,018 VS constructions in the English WSJ Treebank). In addition, verbal arguments can appear before or after the verb, and in many orders. Such variations in constituent order are easy to capture using 'flat' S structures putting the verbs and all of its arguments on the same clausal level, and this is the annotation approach adopted by the Hebrew Treebank (as well as by treebanks of other languages, such as French [Abeillé, Clément, and Toussenel 2003]). These flat structures result in the grammar having more and longer rules and the treebank having fewer instances of each rule type, however, causing a data sparseness

problem for statistical estimation methods based on treebank counts, and making it more difficult to reliably estimate the grammar parameters.

2.1.6 Verbless Constructions. Several constructions in which the verb is not realized are common in Hebrew. These include the possessive constructions such as **לעידו צעצועים רבים** ('to-Ido toys many' meaning 'ido *has* many toys'), which also feature a flexible constituent order **צעצועים רבים לעידו** ('toys many to-Ido', 'ido *has* many toys'), and copular constructions such as **הילד חמוד** ('the-boy cute' 'the boy is cute') and **הילד משוגע** ('the-boy crazy' 'the boy *is* crazy').

2.1.7 NP Structure and Construct-State. Although constituent order may vary, NP internal structure is rigid. A special morphological marker (**construct state**, or **סמיכות**) is used to mark noun-compounds as well as similar phenomena (this is similar to the *idafa* construction in Arabic).[4] Noun compounding in Modern Hebrew is productive and very frequent—about a quarter of the noun tokens in the Hebrew Treebank are in the construct state. Construct-state nouns can be highly ambiguous with non-construct-state nouns. Some forms are morphologically marked but the marking is not present in unvocalized text (**בנות**/banot vs. **בנות**/bnot), and some forms are not marked at all (**עורך**). The construct-state marker, although ambiguous, is essential for analyzing NP internal structure. Where regular nouns are marked for definiteness using the definite marker **ה**, construct-nouns acquire the definite status of the noun-phrase they compound to. Construct constructions may be nested, as in **גוון צבע מכסה קופסת התפוחים** ('shade$_{\text{const}}$ color$_{\text{const}}$ lid$_{\text{const}}$ box$_{\text{const}}$ the apples,' meaning 'the shade of the color of the lid of the box of the apples').

2.1.8 Definiteness. Definiteness is spread across many elements in the NP. All elements in a definite NP, except for construct-nouns and proper-names, are explicitly marked using the functional element **ה** that is prefixed to the token. Proper-names are inherently definite and cannot take the definite marker, and construct-nouns acquire their definiteness status from the NP they dominate (definiteness is not explicitly marked on construct-nouns).

2.1.9 Case Marking. Definite direct objects are marked. The case marker in this case is the function word **את** appearing before the direct object. Subjects, indirect objects, and non-definite direct objects are not marked.

2.1.10 Agreement. Hebrew grammar forces morphological agreement between adjectives and nominals (adjectives appear after the noun, and agree in gender, number, and definiteness), and between subjects and verbs (including the verbless copular constructions), which agree in gender, number, and person. Agreement in the predicative case is a bit complex: When the verb is overt and the predicative-complement is a noun, as in **הנסיעה היא תירוץ** ('the-trip$_{\text{fem}}$ is$_{\text{fem}}$ an-excuse$_{\text{masc}}$'), gender and number agreement are required between the subject and the verb (but not the predicative-complement), but in the verbless case, the subject and the predicate-complement noun must agree (***הנסיעה תירוץ** 'the-trip$_{\text{fem}}$ an-excuse$_{\text{masc}}$'). When the predicate-complement is an adjective, gender and number agreement between the subject and the predicate-complement

4 The construct state is not restricted to nouns, and can also appear on numbers (e.g., **עשרות** ילדים/'tens-of kids') and adjectives (**נדול** הסופרים/'biggest-of authors').

is required regardless of the realization of the verb/copular element: הילד גבוה, הילדה גבוה*, הילד הוא גבוה, הילדה היא גבוה* ('the-boy $tall_{masc}$', '*the-boy $tall_{fem}$', 'the-boy is_{masc} $tall_{masc}$', '*the-girl is_{fem} $tall_{masc}$').

2.2 Implications for Parsing

After surveying some lexical and syntactic properties of Modern Hebrew, we turn to highlight some aspects in which Modern Hebrew differs from English from the perspective of parsing system design.

2.2.1 Small Amount of Annotated Data. Whereas the English Treebank is relatively large (49,208 sentences, or 1,173,766 words), the Hebrew Treebank (Guthmann et al. 2009) is much smaller, containing only 6,220 sentences, or 115,661 tokens (156,316 words[5]).

The small size of the Hebrew Treebank implies a smaller training set for learning-algorithms used to construct the parser.

2.2.2 Ambiguous Word Segmentation. Syntactic parsing systems treat the input sentence as observed data—the leaves (in constituency parsing) of the tree are known in advance, and the parser is expected to build a parse tree around them. This is not the case in Hebrew, where many function words are not separated by white space but instead are prefixed to the next word and appear within the same token. This makes the word sequence unobserved to the parser, which has to infer both the syntactic-structure and the *token segmentation*.[6]

One possible solution to the unobserved word-sequence problem is a pipeline system in which an initial model is in charge of token-segmentation, and the output of the initial model is fed as the input to a second stage parser. This is a popular approach in parsing systems for Arabic and Chinese (Jiang, Huang, and Liu 2009; Green and Manning 2010). As discussed in Section 2.1.2 (as well as in Tsarfaty [2006a], Goldberg and Tsarfaty [2008], and Cohen and Smith [2007]), however, the token-segmentation and syntactic-parsing tasks are closely intertwined and are better performed jointly instead of in a pipeline fashion, which is the approach we explore in this work.

2.2.3 Morphological Variation and High Out-of-Vocabulary Rate. The intrinsic deficiency caused by the small amount of training data is made even more severe due to Hebrew's rich morphological inflection patterns. The high amount of morphological variation means that many word forms will not be observed in the training data, making it harder to reliably estimate lexical probabilities based on the annotated resources alone.

Unlike English, where parts-of-speech for words are relatively easy to guess based on simple orthographic features (words starting with capital letters are proper nouns, words ending in *-ed* are usually verbs, etc.), this is not the case for Hebrew. Among the 773 words appearing in English test data but not in the training data, 269 start with a capital letter, 58 end with *-ed*, and 49 end with *-ing*. Together, these three simple heuristics cover almost half of the unobserved tokens. Such heuristics are not available for Hebrew in the common case of unvocalized text: Proper names are not marked

5 Because of agglutination, a Hebrew token may consist of several words, for example the token בבית comprises the two words ב('in') and בית('house').

6 *Token segmentation* is sometimes (erroneously) referred to as *morphological segmentation*.

in writing, and word prefixes and suffixes are not indicative of the part-of-speech tags.[7] Thus, the out-of-vocabulary (OOV) problem is much harder in Hebrew than in English and other European languages: On the one hand many words are unobserved in training, and on the other, it is more difficult to guess the analysis of such unknown words.

A system for handling automatic processing of Hebrew text cannot rely solely on manually annotated corpora, as such corpora cannot provide adequate lexical coverage. Systems that attempt to perform disambiguation on the lexical level (such as sequence-based morphological disambiguators, or syntactic parsers that perform morphological disambiguation as part of the parsing process) should be designed to incorporate lexical knowledge from sources external to the annotated corpora. We discuss methods of enhancing the system's performance based on a resource that is external to the treebank: A lexicon-based broad-coverage morphological analyzer enhanced with semi-supervised probability estimates based on expectation maximization (EM) training of a hidden Markov model (HMM) tagger on a large amount of unannotated text.

2.2.4 Morphological Agreement. The rich morphological system also means that words carry large amounts of extra information: definiteness, gender, number, tense, and person. Some of this information interacts with syntax through **agreement constraints**. Specifically, nouns and adjectives should agree in gender and number, and subjects and verbs should agree in gender, number, and person. Agreement constraints can provide useful hints for disambiguating the syntactic structure. Consider for example the sentence אשת האדם שאכלה את התפוח ('wife of the man who ate the apple'). The English sentence is ambiguous with respect to the entity who ate the apple, but the Hebrew version is not—the verb אכלה ('ate') is in feminine form, indicating that it was the wife who did the eating. Can a parsing system make use of such information? This issue is investigated further in Section 8.2.

2.3 Existing Resources for Hebrew Text Processing

Several linguistic resources are available for Hebrew, and are used as building blocks for the parsing systems described in this work.

2.3.1 The Hebrew Constituency-Treebank. A constituency treebank of Modern Hebrew, incrementally developed at the Technion over the course of more than eight years (Sima'an et al. 2001; Guthmann et al. 2009), is maintained by MILA, the knowledge center for processing Hebrew.[8] The current version of the treebank (Version 2) contains 6,220 sentences taken from the Israeli daily newspaper הארץ (*Ha'aretz*). The sentences are manually annotated on both the lexical and the syntactic levels. Each token[9] is segmented into words, and each word is assigned a part of speech tag that also captures,

7 Although the suffixes are good indicators of gender and number (ים is usually plural masculine, ה is usually singular feminine), they are not good at indicating the core part-of-speech (ה is a suffix can appear in adjectives יפה, verbs שמרה, nouns מחלה, and similarly for ים (יפים, מתחפשים, מנעולים). Furthermore, due to the root+template system, in most cases the first and last letters of the word are part of the root and not of the pattern שומר,ישמור, making the suffixes even less indicative.

8 http://www.mila.cs.technion.ac.il/mila/eng/index.html.

9 As discussed in Section 2.1.2, Hebrew *tokens* (entities separated by white space and punctuation symbols) do not necessarily correspond to Hebrew *words*. A single token may contain several words.

where applicable, the morphological properties of the word such as number, gender, and person. Then a constituency tree is built on top of the segmented words. The annotation of NPs is relatively nested, and the sentence level structures are relatively flat (the verb and all of its arguments reside on one level under S). The treebank has 115,661 tokens and 156,764 words.

The POS tagging scheme in the treebank is highly syntactic in nature: A part-of-speech is chosen to reflect the syntactic function of the given word in context. For example, demonstrative pronouns are tagged in the treebank as adjectives when appearing in an adjectival position (ילד זה, 'this/JJ child/NN'), and a special MOD tag is used to mark non-adverbial clausal level modification (that is, modifications that can be treated as adverbial, but that are used to modify something other than a verb). For a more detailed description of the Constituency Treebank see Sima'an et al. (2001), Guthmann et al. (2009), and Tsarfaty (2010, pages 199–216), as well as the annotation guidelines.[10]

2.3.2 Train/dev/test Splits. Throughout the article, we follow the established train/dev/test split for the treebank, namely, sentences 1–483 are used for development, sentences 484–5,740 are used for training the parser, and sentences 5,741 to 6,220 are used as the final test set.

2.3.3 The MILA Broad-Coverage Lexicon. Aside from the Constituency Treebank, Hebrew has a wide-coverage, lexicon-based morphological analyzer which can assign morphological analyses (prefixes, suffixes, core POS, gender, number, person, etc.) to Hebrew tokens. The lexicon (henceforth **the KC Analyzer**) is developed and maintained by the Knowledge Center for Processing Hebrew (Itai and Wintner 2008). It is based on a lexicon of roughly 25,000 word lemmas and their inflection patterns. From these, 562,439 unique word forms are derived. These are then prefixed (subject to constraints) by 73 prepositional prefixes. Even with this seemingly large vocabulary, the KC Analyzer's coverage is not perfect. In Adler et al. (2008a), we present a machine-learning method that is trained on the basis of the analyzer and that can guess possible analyses for words unknown to the analyzer with reasonable accuracies. Using this extension, the analyzer has perfect coverage (even though the quality is obviously better for words that are present in the analyzer's database).

The tag set used by the lexicon/analyzer is lexicographic in nature, and is discussed in depth in BGU Computational Linguistics Group (2008).

Creating a resource such as the morphological analyzer for a morphologically rich language is a worthwhile and cost-effective effort: After establishing the tag set, it is relatively straightforward to add lemmas to the lexicon, and the automatic inflection process guarantees good coverage of all the possible inflections. This is much more efficient than annotating enough text to obtain a similar coverage.

2.3.4 Hebrew Morphological Disambiguator. The morphological analyzer provides the possible set of analyses for each token, but does not disambiguate the correct analysis in context. A **morphological disambiguator** (henceforth "the Hebrew tagger" or "tagger") was developed by Meni Adler at Ben-Gurion University of the Negev (Adler and Elhadad 2006; Adler 2007; Goldberg, Adler, and Elhadad 2008). After the (extended) morphological analyzer assigns the possible analyses for each token in an

10 http://www.mila.cs.technion.ac.il/mila/files/treebank/Decisions-Corpus1-5001.v1.pdf.

input sentence, the tagger takes the output of the analyzer as input and chooses the single best analysis for the entire sentence (performing both token segmentation of words and part-of-speech assignment for each word). The tagger is an HMM-based sequential model that is trained in a semi-supervised fashion using EM based on the output of the morphological analyzer on a large amount (about 70M words) of unannotated Hebrew text. The tagger is described in Adler and Elhadad (2006) and Adler (2007).

The tagger is relatively accurate: It achieves 93% accuracy in predicting segmentation and tagging when measured on the POS accuracy, and 90% accuracy when measured on the complete tag set, which includes the complete set of morphological features. Because the tagger is not trained on a particular annotated training set but instead on a very large corpus of text spanning multiple genres, its performance is robust across domains.

The tagger's success is due in part to a smart initialization procedure to the EM training process. This initialization procedure takes the output of the analyzer and assigns a conditional probability distribution $P(tag|word)$ for each word. In other words, it assigns an a priori, context-free likelihood for each analysis of a word (although the word *broke* can be either a verb in the past tense or an adjective, it is more likely to be the former; such preferences can be modeled as probability distributions, and the initialization procedure attempts to learn the values of these distributions automatically from raw data). This initialization procedure is described in Goldberg, Adler, and Elhadad (2008).

A side effect of the EM–HMM training of the tagger is pseudo-counts for $\langle word, tag \rangle$ events, which are based on patterns observed in the unannotated training data. We use these counts in order to improve the lexical-disambiguation capacity of the parser.

2.3.5 A Resource Incompatibility Issue. Unfortunately, the KC Analyzer adopted a different tag set than the one used in the treebank, and analyses produced by the KC Analyzer (and hence by the morphological disambiguator) are incompatible with the Hebrew Treebank. These are not mere technical differences, but derive from different perspectives on the data. The Hebrew Treebank (TB) tag set is syntactic in nature ("if the word in this particular position functions as an adverb, tag it as an adverb, even though it is listed in the dictionary only as a noun"), whereas the KC tag set (Adler 2007; Netzer et al. 2007; Adler et al. 2008b) takes a lexical approach to POS tagging ("a word can assume only POS tags that would be assigned to it in a dictionary"). The lexical approach does not accommodate generic modification POS tags such as MOD, nor does it allow listing of demonstrative pronouns as adjectives.

These divergent perspectives are reflected in different guidelines to human taggers, different principles underlying tag definitions, and different verification procedures. This difference in perspective yields different performances for parsers induced from tagged data, and a simple mapping between the two schemes is impossible to define.

Some Hebrew forms, particularly the present participle and modal forms, are inherently hard to define, and the wide disagreement about their status is reflected in practically all Hebrew dictionaries. This kind of disagreement naturally appears also between the KC and TB. See Adler et al. (2008b) and Netzer et al. (2007) for further discussion on these two interesting cases.

Bridging the discrepancy between the two resources is an important aspect in the creation of a successful parsing system. On the one hand the syntactic annotations in the treebank are needed in order to train the parser, and on the other hand the information provided by the morphological analyzer is needed in order to provide a good lexical coverage. We discuss an approach to bridging this discrepancy in Section 6.

2.4 Section Summary

To summarize, the Hebrew language and its analysis poses several challenges to parser design: The amount of annotated material is relatively small, precluding the possibility of learning robust lexical parameters from the annotated corpora. The productive nature of the morphology results in many word forms, adding another obstacle to estimating lexical parameters from annotated data. The nature of the word-formation mechanism in Hebrew makes it hard to guess the morphological analysis of a word based on its prefix and suffix alone as is done in other languages, requiring the use of a more complex system for handling unknown words. Many function words in Hebrew are not separated by white space but are instead attached to the next token, making the observed word sequence ambiguous. Word segmentation needs to be performed in addition to syntactic disambiguation. Successful word segmentation may rely on syntactic disambiguation, suggesting that it is better to perform the segmentation and syntactic-disambiguation tasks jointly. Finally, Hebrew grammar requires various forms of morphological agreement, a fact which hopefully can help disambiguate otherwise ambiguous syntactic structures. The syntactic parser should be able to make use of agreement information.

In terms of existing resources, Hebrew has a small treebank annotated with constituency structure and a broad-coverage, manually constructed, lexicon-based morphological analyzer. The morphological analyzer is capable of providing the possible morphological analyses for many lexical forms, and it is extended using a machine-learning technique to also provide possible analyses for word-forms not covered by the lexicon. The extended lexicon provides a good lexical coverage of Hebrew. Also available is a morphological disambiguator that is capable of associating probabilities to the possible analyses of the lexical forms in the lexicon, and disambiguating the analyses of a sequence of lexical items in context based on a sequential model. The constituency treebank can be used to learn the parameters of a syntactic-model of Hebrew, and the morphological analyzer can be used to provide broad-coverage lexical knowledge. Unfortunately, the treebank and the lexicon/disambiguator follow different annotation schemes, and are therefore incompatible with each other. The annotation gap between the two resources must be bridged before they can be used together.

We now turn to survey the components of our Hebrew parsing system.

3. Latent-Annotation State-Split Grammars (PCFG-LA)

Klein and Manning (2003) demonstrated that linguistically informed splitting of non-terminal symbols in treebank-derived grammars can result in accurate grammars. Their work triggered investigations in automatic grammar refinement and state-splitting (Matsuzaki, Miyao, and Tsujii 2005; Prescher 2005), which was then perfected in work by Petrov and colleagues (Petrov et al. 2006; Petrov and Klein 2007; Petrov 2009).

State-split models assume that each non-terminal label has a latent annotation that should be recovered. Instead of a single NP symbol, these models hypothesize that there are many different NP symbols, $NP_1, \ldots, NP_k$, and each is used in a different context. The labels are hidden, however, and we can only observe the core category label (NP). The job of the training process is to come up with the hidden set of label assignments to non-terminals, such that the resulting grammar assigns a high probability to the observed treebank data. Such models are called PCFG with latent annotations (PCFG-LA) and are shown empirically to produce very accurate parsing results.

The model of Petrov et al. (2006) and its publicly available implementation, the BerkeleyParser,[11] learns the latent annotations by starting with a bare-bones treebank-derived grammar and automatically refining it in split-merge-smooth cycles, setting the parameters using EM. We provide a brief description of the model and learning process (refer to Petrov et al. 2006; Petrov and Klein 2007; Petrov 2009 for the full details).

The learning works by following an iterative split-merge-smooth cycle, in which the following steps are performed repetitively:

Splitting each non-terminal category in two All of the grammar symbols are split. In the first round, NP is split into NP_1 and NP_2. In the second round these are split into NP_{11}, NP_{12}, NP_{21}, NP_{22}, and so forth. Each splitting round results in new grammar in which a rule of the form $A \rightarrow BC$ is replaced by eight rules, the result of splitting each A, B, and C in two. An EM procedure is then used to set the probabilities of each of the split rules. The EM training is constrained by the grammar on the one hand and by the annotated tree structures on the other.

Merging back non-effective splits Not all of the splits are useful. For example, the punctuation POS tag will always result in punctuation, and there is no reason to split it into two punctuation POS tags. Having a grammar with too many states is difficult to manage in terms of memory, storage, and parsing time, and is also prone to overfitting the data. Thus, the model aims to undo splits if they are not useful. The splits are evaluated based on an information gain criteria, and splits that are not useful are merged back into their parent symbol, resulting in a smaller grammar (if the symbols B_1 and B_2 are merged back into B, the rules $A \rightarrow B_1\ C$ and $A \rightarrow B_2\ C$ are merged into $A \rightarrow B\ C$). The merging step is also followed by an EM procedure for setting the rule probabilities for the resulting grammar.

Smoothing the split non-terminals toward their shared ancestor Finally, split symbols may still share some information (although an NP in subject position and an NP in object position behave differently, they also retain some common properties). The smoothing procedure joggles the probability mass of the grammar and moves some probability from the split symbol to its parent. This step is also followed by parameter re-estimation using EM.

Performing five or six such split-merge-smooth cycles results in accurate grammars, with annotations that capture many latent syntactic interactions. Six cycles mean that symbols can have as many as 64 different substates.

At inference time, the latent annotations are (approximately) marginalized out, resulting in the (approximate) most probable unannotated tree according to the refined grammar (the score of the unsplit rule $A \rightarrow B\ C$ is taken to be $\sum_x \sum_y \sum_z A_x \rightarrow B_y\ C_z$).

The grammar learning process is applied to binarized parse trees, with first-order vertical and zeroth-order horizontal markovization (Klein and Manning 2003). This means that in the initial grammar, each of the non-terminal symbols is effectively conditioned on its parent alone, and is independent of its sisters. For example, the rule $S \rightarrow NP\ VP\ NP\ PP$ is binarized as:

$S \rightarrow NP\ @S$
$@S \rightarrow VP\ @S$
$@S \rightarrow NP\ @S$
$@S \rightarrow PP$

11 http://code.google.com/p/berkeleyparser/.

indicating that S rules start with an NP, can be followed by a sequence of zero or more NPs and VPs, and end with a PP. Such an extreme markovization suggests a very strong independence assumption, and is too permissive on its own. It allows the resulting refined grammar to encode its own set of dependencies between a node and its sisters, however, as well as ordering preferences in long, flat rules. For example, the binarized grammar allows the production S → NP NP PP, which may be incorrect. However, by annotating the symbols as follows:

$S \rightarrow NP\ @S_1$
$@S_1 \rightarrow VP\ @S_2$
$@S_2 \rightarrow NP\ @S_2$
$@S_2 \rightarrow PP$

the grammar now forces the VP to be produced before the NP, but still allows the NP to be dropped. Similarly, by annotating the symbols as:

$S \rightarrow NP\ @S_1$
$@S_1 \rightarrow VP\ @S_2$
$@S_2 \rightarrow NP\ @S_3$
$@S_3 \rightarrow PP$

the grammar effectively allows only the original rule to be produced.

Initial experiments on Hebrew confirm that moving to higher order horizontal markovization (encoding more context in the initial binarized rules) degrades parsing performance, while producing much larger grammars.

The PCFG-LA parsing methodology is very robust, producing state-of-the-art accuracies for English, as well as many other languages including German (Petrov and Klein 2008), French (Candito, Crabbé, and Seddah 2009), and Chinese (Huang and Harper 2009).

4. Baseline Experiments

The baseline system is an "out-of-the-box" PCFG-LA parser, as described in Petrov et al. (2006) and Petrov and Klein (2007) and implemented in the BerkeleyParser.[12] The parser is trained on the Modern Hebrew Treebank (see Section 9 for the exact experimental settings) after stripping all the functional and morphological information from the non-terminals.

We evaluate the resulting models on the development set, and consider three settings:

Seg+POS Oracle: The parser has access to the gold segmentation and POS tags.
Seg Oracle: The parser has access to the gold segmentation, but not the POS tags.
Pipeline: A POS-tagger is used to perform word segmentation, which is then used as parser input.

A better tag set. Glossing over the parses revealed that the parser failed to learn the distinction between finite and non-finite verbs. The importance of this linguistic

12 `http://code.google.com/p/berkeleyparser/`.

Table 1
Baseline: Out-of-the-box BerkeleyParser performance on the dev-set.

Setting	Tag set	F_1 (4 cycles)	F_1 (5 cycles)
Seg+POS Oracle	Core	89.7	89.5
Seg Oracle	Core	82.6	83.6
Pipeline	Core	76.3	77.2
Seg+POS Oracle	Core+Verbs	89.9	90.9
Seg Oracle	Core+Verbs	83.3	83.6
Pipeline	Core+Verbs	77.1	77.3

distinction for parsing is obvious, and was also noted in Klein and Manning (2003) for English and in our previous work on parsing Hebrew (Goldberg and Tsarfaty 2008). Finite and non-finite verbs are easily distinguishable from each other based on surface form alone. Although finiteness is clearly annotated in the treebank, it is not on the "core" part of the POS tags and was removed prior to training the parser. In a second set of experiments the core tag set of the parser was modified to distinguish finite verbs, infinitives, and modals.[13] The original core–tag set already includes some important distinctions, such as construct from non-construct nouns.

Results and discussion. Table 1 presents the parsing results on the development set. With gold POS tags and segmentation, the results are very high. Accuracy drops considerably when the parser is not given access to the gold tags (from about 90 to less than 84 F_1), indicating that the POS tags are both informative and ambiguous. Results drop even further (from 84 to 77) in the pipeline case where the gold segmentation is not available, indicating that correct segmentation also provides valuable information to the parser and that segmentation mistakes are costly.

Enriching the tag set to distinguish modals and finite and infinite verbs proved useful, with an increase of about 1 F_1 points (absolute) after four split-merge-smooth cycles, and a smaller increase after five cycles. This stresses the importance of the core representation: The automatic learning procedure goes a long way, but it can be aided by linguistically motivated manual interventions in some cases.

4.1 Analyzing the Learned PCFG-LA Grammar

4.1.1 Terminal-Level (Lexical) Splits. We begin by inspecting the splits at the part-of-speech level. Table 2 displays the number of splits learned for each of the parts-of-speech symbols. Prepositions are the most heavily split, followed closely by the somewhat-generic MOD tag and the nouns.

Nouns and adjectives. The noun and adjective splits are somewhat hard to decipher. Some of the groups are obvious (*things appearing after numbers*, *last names*, *parts-of-dates*, *time related*, *places*, etc.). Others are are much harder to interpret.

13 Unlike previous work, the distinction is retained only at the POS tag level and not propagated to the phrase level. The tag-level information is sufficient for the parser to learn the phrase-level distinctions on its own. Similar observations regarding the usefulness and sufficiency of linguistically motivated manual state-splitting of preterminals (as opposed to tree-internal nodes) prior to training a latent-variable grammar were also made by Crabbé and Candito (2008).

Table 2
Number of learned splits per POS category after five split-merge cycles.

Tag	# Splits	Tag	# Splits
H	1	CDT	6
HAM	1	CC	7
POS	1	DT	7
REL	1	JJ	7
VB	1	VB-INF	7
AT	2	PRP	8
COM	2	CD	10
JJT	2	RB	13
QW	2	NN	16
RBR	2	NNP	17
VBMD	2	NNT	22
WDT	2	MOD	24
AGR	4	IN	26
AUX	6		

MOD. For the general-modification POS tags, most categories clearly single out one or two words with very specific usage patterns, such as **לא** ('no'), **גם** ('also'), **רק** ('only'), **אפילו** ('even'), **לשעבר** ('former'), and so forth. The other categories are harder to interpret.

Verbs. Finite-verbs are not split at all, even though they form an open-class category. Modal verbs are split into two groups: One of them is dominated by nine modals (**אפשר**, **יש**, **נראה**, **קשה**, **אין**, **ניתן**, **דומה**, **חשוב**, **נכון**, roughly corresponding to the English *could*, *should*, *seem/appear*, *hard*, *shouldn't*, *possible*, *appear/seem*, *important*, *fitting/required*); and the second contains all the others. This is an interesting distinction, as the nine singled-out modals never take a subject, whereas the modals in the other group do.[14] Infinitive verbs are split into seven categories, six of which are dominated by one or two words each, and the last is a catch-all category.

Coordination and question-words. Coordination words are heavily split, each of the categories dominated by one or two words, indicating different usage patterns. The question words **מה** ('what') and **מי** ('who') are singled out from the rest.

Gender/number agreement. The verbs are not split at all, indicating that the learned grammar cannot model subject–verb agreement. Pronouns are split by type (personals, demonstrative, and subtypes of demonstratives), but not by gender and number. Noun and adjective splits are sometimes hard to decipher, but they do not exhibit any grouping based on gender or number properties, indicating that the grammar cannot model adjective–noun agreement. Category splits for the AGR tag do show a clear division that follows gender and number, but it is unclear what is captured by this division as the information cannot interact with nouns, adjectives, verbs, or pronouns.

14 In fact, the nine modals are very similar in characterization to the words identified in Netzer et al. (2007) as modals, whereas many of the modals in the other group are not necessarily considered as modal outside of the treebank guidelines.

Table 3
Number of learned splits per NT-category after five split-merge cycles.

Tag	# Splits	Tag	# Splits
FRAGQ	1	ADVP	16
INTJ	6	S	16
FRAG	7	PP	22
SQ	7	VP	22
PRN	8	PREDP	25
ADJP	14	NP	32
SBAR	14		

4.1.2 Grammar-Level Splits. Table 3 shows the number of splits learned for each grammar non-terminal. The NP category is the most heavily split, followed by predicative phrases, verb phrases, and PPs. With the exception of the FRAGQ category, all symbols are split into at least six substates. What information is encapsulated in the state splits? As noted by Petrov et al. (2006), the latent state-splits learned for the grammar symbols are harder to analyze.

One way of shedding some light on the meanings of the split-states is by using the grammar in generation mode and by sampling word sequences from each of the states.[15] By looking at the resulting strings, one can sometimes infer the kind of information encoded in the grammar.

NP. The split-NPs encode phrase length (some splits result in very long NPs, some in very short, some in very specific one- or two-word patterns). They also encode the definiteness rules (either an NP is definite or not), the interaction between definiteness and the AT marker, and a limited interaction between definiteness and construct nouns. Other NP splits are dedicated to pronouns or to question words, or encode proper names, monetary units, and numbers.

SBAR. The split-SBARs are split according to the word introducing the SBAR. In addition, some split-SBARs encode quoted and parenthetical items.

S. The split-Ss differ by length. In addition, some S splits seem to be modeling verb-less sentences, variations in word order, and sentence-level coordination.

4.2 Limitation of PCFG-LA Parsing of Modern Hebrew

The PCFG-LA baseline is a strong one, and is substantially higher than all previous reported results for Hebrew parsing in each of the setups (Seg+POS oracle, Seg Oracle, and no Oracle). We also identify some of its limitations, namely:

Missed splits. The learning procedure is not perfect, and fails to capture some linguistically meaningful state-splits. When such splits are manually supplied (i.e., the trivial split of verbal types) accuracy improves.

15 Sampling a word sequence is performed by starting at a given state (a split grammar symbol), randomly choosing a right-hand-side based on the PCFG-induced distribution, expanding the state into the chosen right-hand side, and continuing recursively until we are left with only strings.

Sensitivity to non-gold POS. The substantial drop in accuracy when the POS tags are unobserved and need to be predicted is staggering, which suggests that it is difficult for the parser to assign part-of-speech tags. Of the 698 part-of-speech errors, 314 are on words not seen in training.

Sensitivity to non-gold segmentation. The accuracy drops even further when the parser is presented with predicted segmentation. Segmentation errors are detrimental to the parser.

Not encoding grammatical agreement. Finally, the learned grammar does not encode grammatical agreement. Whereas the majority of the parser mistakes are due to the flexible constituent order or "standard" ambiguities such as coordination and PP attachment, a handful of them could be resolved using agreement information.

In what follows, we address these four limitations, and substantially increase the parser accuracy for the realistic case where gold segmentation and POS tags are not available.

5. Manual State-Splits

We experimented with several linguistically motivated state-splits which were added as tree-annotations prior to running the parser. Most of them did not help on their own and slightly degraded parser performance when combined with other splits. These include splits which were proven useful in previous work, such as marking of definite NPs, and distinguishing possessive from other PPs. We also experimented with splits based on morphological agreement features, which are discussed in Section 8.1.

Overall, the learning procedure is capable of producing good splits on its own. We did, however, manage to improve upon it with the following annotation (the annotations were removed prior to evaluation).

Subject NPs. Hebrew phrase order is rather flexible, and the subject can appear before or after the verb. Identifying the subject can thus help in grounding the overall structure of the sentence. The subject is also dependent on agreement constraints with the verb. Following Johnson (1998), Klein and Manning (2003) implicitly annotate subject-NPs in English using parent annotation (distinguishing NPs under S from other NPs), with good results. When applied to English, the PCFG-LA also learns to model subject NPs well. Hebrew's non-configurationality, however, put both Subjects and Objects directly under S, making it much harder to learn the distinction automatically.

Explicit marking of subject NPs contributes slightly to the accuracy of the parser. Perhaps more important than the small increase in accuracy is the fact that the parser can identify subjects relatively well. In contrast, marking of object NPs did not help by itself and slightly degraded the parsing accuracy when combined with other annotations. Note, however, that Hebrew definite objects are already clearly marked using the את marker, making them an easy target for the parser.

6. Better Lexical Coverage with an External Lexicon

The drop in parsing accuracy when gold core POS tags are not available and need to be inferred by the parser is huge (from above 90 to less than 84 F_1).

The large number of possible word forms make it very difficult for manually annotated corpora to provide adequate lexical coverage. The problem is even more severe with the case of the Hebrew Treebank, which is especially small. Although it is big enough to learn meaningful syntactic generalizations (as demonstrated by the high performance of the baseline system) it is far too small to learn a good lexical model (as evidenced by the drop in accuracy when gold tags are not available).

We suggest increasing the lexical coverage of the parser using an external resource, namely, a lexicon-based morphological analyzer. We further extend the utility of the analyzer with lexical tagging probabilities learned from an unannotated corpus.

6.1 A Unified Lexical Probability Model

We would like to use the KC Analyzer (Section 2.3.3) to increase the lexical coverage of the treebank-trained parser. That is, we would like to improve the lexical model $P(T \rightarrow W)$ of the generative parser. As discussed in Section 2.3.5, however, the tag sets used by the two resources differ. How can this difference be reconciled?

One possibility is to re-tag the treebank with the KC tag set and then train on this unified resource. In Goldberg et al. (2009), we show that this procedure degrades parser performance. Instead, Goldberg et al. suggest a layered generative approach that retains the benefits of the treebank tagging for frequent words and resorts to the KC tag set only for rare and unseen words. Under this approach, frequent words are generated from treebank POS tags as usual, but rare words follow a generative process in which first the treebank tag generates a KC tag, and then the KC-tag generates the word. A sample derivation using this layered representation is presented in Figure 1.

The Treebank-to-KC tag generation probabilities represent a fuzzy, probabilistic mapping between the two resources. In Goldberg et al. (2009), the estimation of these probabilities was done based on a re-tagging of the treebank to use the KC tag set. The re-tagging process was far from trivial, and many tagging cases required extensive debates between human annotators.

Here, we present a new procedure which does not require the treebank to be re-tagged with a new tag set. It still uses the layered representation, but instead of forcing one unique KC analysis for each location, it embraces the uncertainty and allows all of them. This is done by treating the KC-tag assignments as hidden variables, learning the TB-KC mapping probabilities as part of the grammar training EM process, and marginalizing the KC tags out for the final tree. The procedure is based on the following assumptions:

- We have access to trees in which the POS tags t_{tb} are taken from a given tag set T_{TB}.

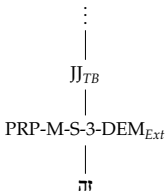

Figure 1
A layered POS tag representation.

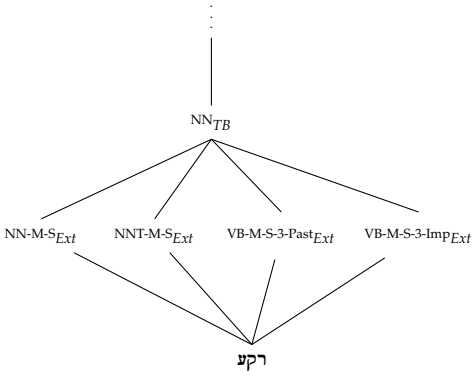

Figure 2
A latent layered POS tag representation.

- We have additional access to an external resource (lexicon) mapping words to tags t_{ext} from a different tag set T_{Ext}.
- Probabilities involving words which are frequent in the treebank can and should be based on treebank counts.
- Probabilities involving less frequent words should be smoothed in with information from the external lexicon.
- Smoothing should have a greater effect on less-frequent words.
- Probabilities for unseen words should be based solely on the external lexicon.

Figure 2 illustrates the representation used for words which are rare or unseen in the treebank training data. The treebank tag NN_{TB} (upper level) generates the word-form **רקע** (lower level) by considering all the possible KC POS tags allowed for the word in the morphological analyzer (the middle level). The probabilities related to generating the KC POS tags are summed, and all the other probabilities are multiplied. The exact equations are detailed in the following.

Although the needed quantity is the emission probability $P(T_{TB} \rightarrow W) = P(W|T_{TB})$, it is more convenient (for a reason which will be discussed later) to work with the tagging probability $P(T_{TB}|W)$. Once the tagging probabilites $P(T_{TB}|W)$ are available, they can easily be converted to emission probabilities using Bayesian inversion, based on the relative-frequency estimates of $P(W)$ and $P(T_{TB})$ which are calculated from the treebank:[16]

$$\frac{P(t_{tb}|w)P(w)}{P(t_{tb})} = P(w|t_{tb}) = P(t_{tb} \rightarrow w) \quad (1)$$

16 Our notation uses capital letters to denote random variables, and lower-case letters to denote specific events. Thus, $P(T|W)$ refers the distributions in which a tag $\in T$ is condition on a word $\in W$, $P(T|w)$ refers to the conditional distribution of tags $t \in T$ given a specific word w, and $P(t|w)$ refers to the probability mass of the specific tag t given word w.

Let us now focus on estimating the tagging probabilities $P(T_{TB}|W)$ for the cases of frequent, rare, and OOV words.

For frequent words that are seen more than K times in the treebank, we simply use treebank-based relative-frequency estimates:[17]

$$P_{\text{tb}}(t_{tb}|w) = \frac{c(w, t_{tb})}{c(w)} \tag{2}$$

where $c(\cdot)$ is a counting function.

For OOV words that are not seen in the treebank, the tagging probability is estimated using:

$$P_{\text{oov}}(t_{tb}|w) = \sum_{t_{\text{ext}} \in T_{\text{Ext}}} P(t_{\text{ext}}|w)P(t_{tb}|t_{\text{ext}}) \tag{3}$$

where $P(T_{\text{Ext}}|W)$ is a tagging probability using the external tag set, and $P(T_{TB}|T_{\text{Ext}})$ is a transfer probability relating the tags from the two tag sets (the estimation of these two probabilities is discussed subsequently). What this does is assume a process in which the word is tagged by first choosing a tag according to the external lexicon, and then choosing a tag from the TB tag set based on the external one. The external tag assignments are then treated as latent variables, and are marginalized out.

Finally, for rare words that are seen only a few times in the treebank, we interpolate the two quantities, weighted by the word's frequency in the treebank:

$$P_{\text{rare}}(t_{tb}|w) = \frac{c(w)P_{\text{tb}}(t_{\text{tb}}|w) + P_{\text{oov}}(t_{tb}|w)}{1 + c(w)} \tag{4}$$

We now turn to describing the estimation of the external tagging probability $P(T_{Ext}|W)$ and the tag transfer probability $P(T_{TB}|T_{Ext})$.

Estimating $P(T_{Ext}|W)$. The tagging probability follows the morphological analyzer. The analyzer provides the possible analyses, but does not provide probabilities for them. One simple option would be to assign each possible analysis (tag) a uniform probability, and assign 0 probability for tags not allowed by the lexicon for the given word. This method is referred to as $P_{\text{unif}}(T_{Ext}|W)$. We know that not all the possible analyses for a given word are equally likely, however, and in practice, the actual tagging distribution is usually biased toward one or two of the tags. These tagging preferences can be learned in an unsupervised manner given the lexicon and a large corpus of unannotated text, using EM training of an HMM tagging model. Adler and Elhadad (2006) suggest such a model for accurate tagging of Hebrew, and Adler (2007) and Goldberg, Adler, and Elhadad (2008) extend it to provide state-of-the-art tagging accuracies for Hebrew using a smart initialization. Here, we use the pseudo-counts from

17 In practice, a small amount of smoothing is added to allow tagging a word with open-class tags if it wasn't seen within the treebank: $P_{\text{tb}}(t_{tb}|w) = (c(w, t_{tb}) + 0.0001 * P(t_{tb}))/(c(w) + 0.0001)$.

the final round of EM training in this tagging model in order to compute $P_{\text{em}}(T_{\text{Ext}}|W)$. We show in Section 9 that this unsupervised lexical probabilities estimation does indeed provide better parsing results.

Estimating $P(T_{TB}|T_{Ext})$. The tagset-transfer probabilities capture the patterns of transfer between the syntactic tagging scheme of the treebank and the other tagging scheme of the external resource. They are estimated using treebank counts and the tagging distribution $P(T_{Ext}|W)$:

$$P(t_{\text{tb}}|t_{\text{ext}}) = \frac{c(t_{\text{tb}}, t_{\text{ext}})}{c(t_{\text{ext}})} = \frac{\sum_w c(t_{\text{tb}}, w)P(t_{\text{ext}}|w)}{\sum_w c(w)P(t_{\text{ext}}|w)} \quad (5)$$

Integration into the PCFG-LA model. The estimation procedure is incorporated into the training process of the PCFG-LA model. Note that in the PCFG-LA model the treebank tag set T_{TB} is gradually split, and each tag takes the form $\langle \text{tag}, \text{substate} \rangle$, where *substate* is a latent variable indicating a specific split of the given tag. This means that the treebank tagging probability and the tag set–transfer probabilities are also defined over these split tags. Whereas the external tagging probabilities $P(T_{\text{Ext}}|W)$ are fixed prior to PCFG-LA training, the other distributions ($P(T_{TB_{substate}}|W)$ and $P(T_{TB_{substate}}|T_{Ext})$) are re-estimated in the EM process following each of the split, merge, and smooth stages. This is done by replacing the corpus counts $c(\cdot)$ in Equations (2) and (5) with pseudo-counts (expectations, marginal scores) of the same events in the E step of the EM procedure.

The main reason for using the Bayesian inversion (Equation (1)) instead of working with the emission probability $P(W|T)$ directly is that the emission probability is highly dependent on the vocabulary size. The treebank estimates are based on a small vocabulary, the external lexicon estimates are based on a very large vocabulary, and a proper combination of the two emission probabilities is not trivial. In contrast, the tagging probabilities do not depend on the vocabulary size, allowing a very simple combination. We can then base the counts for the emission probability on the treebank vocabulary alone, and estimate $P(W)$ for words unseen in training as if they were seen once.

7. Joint Segmentation and Parsing

When applied to real text (for which the gold word-segmentation is not available), the baseline PCFG-LA parser is supplied with word segmentation produced by a separate tagging process.[18] This seriously degrades parsing performance. A major reason for the performance drop is that the word-segmentation task and the syntactic-disambiguation task are highly related. Segmentation mistakes drive the parser toward wrong syntactic structures, and many segmentation decisions require long-distance information that is not available to a sequential process (Tsarfaty 2006a). For these reasons, we claim that parsing and segmentation should be performed jointly.

18 Although the tagger also produces POS tag assignments, we ignore them and use only the word segmentation. This is done for two reasons: first, the tag set of the tagger is the one used by the morphological analyzer, and is not compatible with the treebank. Second, we believe it is better for the parser to produce its own tag assignments.

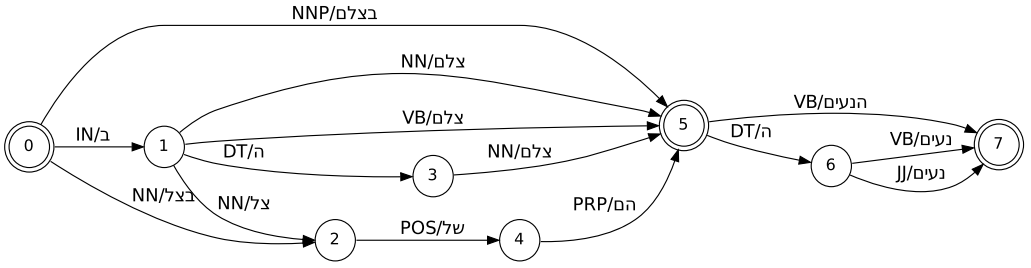

Figure 3
The lattice for the Hebrew sequence **בצלם הנעים** (see footnote 19).

Joint segmentation and parsing can be achieved using **lattice parsing**. Instead of parsing over a fixed input string, the parser operates on a lattice—a structure encoding all the possible segmentations.

7.1 Lattice Representation

Formally, a lattice is a directed acyclic graph in which all paths lead from the initial state to the end state.

For the Hebrew segmentation task, all word segmentations of a given sentence are represented using a lattice structure. Each lattice arc corresponds to a word and its corresponding POS tag, and a path through the lattice corresponds to a specific word-segmentation and POS tagging of the sentence. This is by now a fairly standard representation for multiple morphological segmentations of Hebrew utterances (Adler 2001; Bar-Haim, Sima'an, and Winter 2005; Adler 2007; Cohen and Smith 2007; Goldberg, Adler, and Elhadad 2008; Goldberg and Tsarfaty 2008; Goldberg and Elhadad 2011). It is also used for Arabic (Green and Manning 2010) and other languages (Smith, Smith, and Tromble 2005).

Figure 3 depicts the lattice for the two-words sentence **בצלם הנעים**.[19] Double-circles indicate the space-delimited token boundaries. Note that in this construction arcs can never cross token boundaries. Every token is independent of the others, and the sentence lattice is in fact a concatenation of smaller lattices, one for each token. Furthermore, some of the arcs represent lexemes not present in the input tokens (e.g., **ה**/DT, **של**/POS), although these are parts of valid analyses of the token. Segments with the same surface form but different POS tags are treated as different lexemes, and are represented as separate arcs (e.g., the two arcs labeled **נעים** from node 6 to 7).

A similar structure is used in speech recognition. There, a lattice is used to represent the possible sentences resulting from an interpretation of an acoustic model. In speech recognition the arcs of the lattice are typically weighted in order to indicate the probability of specific transitions. Given that weights on all outgoing arcs sum up to one, weights induce a probability distribution on the lattice paths. In sequential tagging models such as Smith, Smith, and Tromble (2005), Adler and Elhadad (2006), and Bar-Haim, Sima'an, and Winter (2008) weights are assigned according to a tagging model based on linear context. For the case of parsing, context-free weighting of lattice arcs is used: each arc

19 Whereas Hebrew is written right-to-left, the lattice is to be read left-to-right. The words on each arc follow the Hebrew writing directions, and are written right-to-left.

corresponds to a $\langle tag, word \rangle$ pair, and is weighted according to the emission distribution $P(tag \rightarrow word)$.[20]

7.2 Lattice Parsing

The CKY parsing algorithm can be extended to accept a lattice, instead of a predefined list of tokens, as its input (Chappelier et al. 1999). The CKY search then finds a tree spanning from the start-state to the end-state of the lattice, where the leaves of the tree are lattice arcs. The lattice extension of the CKY algorithm is performed by indexing lexical items according to their start- and end-states in the lattice instead of by their sentence position, and changing the initialization procedure of CKY to allow terminal and preterminal symbols of spans of sizes > 1. It is then relatively straightforward to modify the parsing mechanism to support this change: not giving special treatments for spans of size 1, and distinguishing lexical items from non-terminals by a specified marking instead of by their position in the chart.

Figure 4 shows the CKY chart for the lattice in Figure 3, together with an (incorrect) parse over the lattice. The chart is initialized with parts of speech corresponding to the lattice arcs. Phrase-structures are then built on top of the POS tags (in blue). The proposed structure must span the entire chart, and correspond to a path through the lattice from the initial state (0) to the last one (7).

At training time the correct segmentation is fully observed, and the generative parser is trained as usual over the treebank. At inference (test) time, the correct segmentation is unknown, and the decoding is applied to the segmentation lattice. The best derivation returned by the parser forces a specific segmentation. The returned parse tree is the most probable $\langle segmentation, tree \rangle$ pair according to the grammar.[21] We modified the PCFG-LA BerkeleyParser to accept lattice input at inference time.

Lattice parsing allows us to preserve the segmentation ambiguity and present it to the parser, instead of committing to a specific segmentation prior to parsing. This way segmentation decisions are performed in the parser as part of the global search for the most probable structure, and can be affected by global syntactic considerations. We show in Section 9 that this methodology is indeed superior to the pipeline approach.

Early descriptions of algorithms for parsing over word lattices can be found in Lang (1974, 1988) and Billott and Lang (1989). Lattice parsing was explored in the context of parsing of speech signals by Chappelier et al. (1999), Sima'an (1999), and Hall (2005), and in the context of joint word-segmentation and syntactic disambiguation in Cohen and Smith (2007), Goldberg and Tsarfaty (2008), and Green and Manning (2010).

20 Lattice parsing for Hebrew is explored also in Cohen and Smith (2007). There, lattice arc weights are assigned based on aggregate quantities (forward-backward tagging marginals) derived from a discriminative CRF tagging model. This approach is not ideal from a modeling perspective, as it makes each POS tag be accounted for twice: once by the syntactic model, and once by the sequential one. In this work, a sequential tagging model is not used at all. If the use of a sequential model is desired, an alternative method for integrating a sequence model and a syntactic model is making the models "negotiate" an agreed upon structure that maximizes the score under both models, using optimization techniques such as dual decomposition (Dantzig and Wolfe 1960), which was recently introduced into natural language processing (Rush et al. 2010).

21 Note that finding the most probable *segmentation* requires summing over all the trees resulting in each segmentation—a much harder task, proven to be NP-complete in Sima'an (1996).

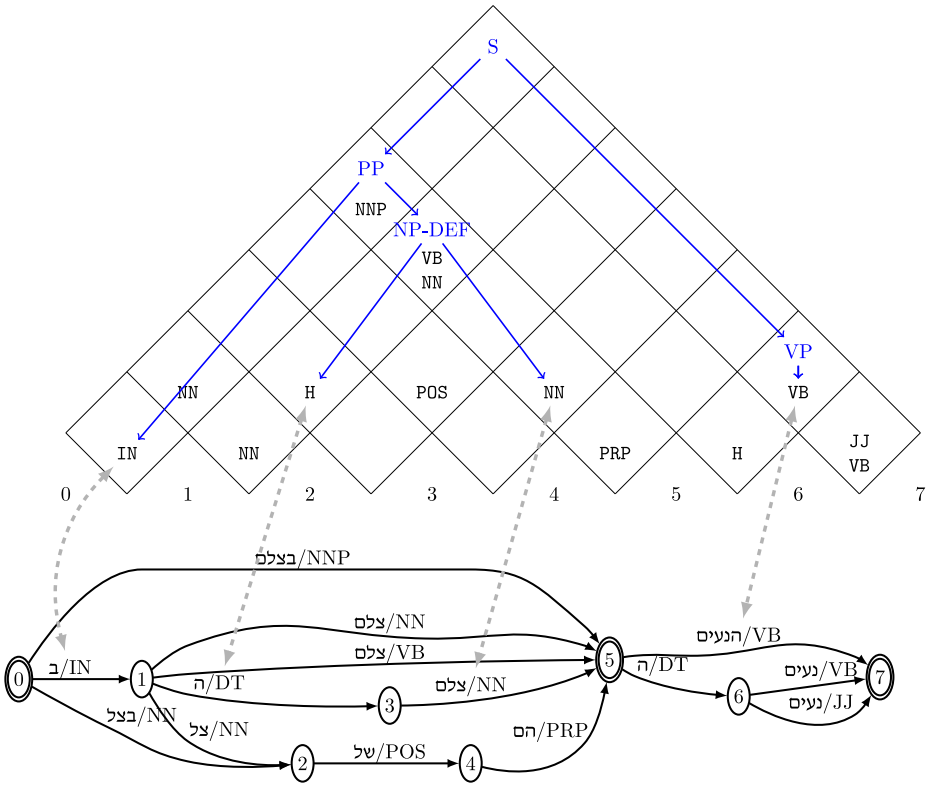

Figure 4
Lattice initialization of the CKY chart.

8. Incorporating Morphological Agreement

Inspecting the learned grammars reveal that they do not encode any knowledge of morphological agreement: The split categories for nouns, verbs, and adjectives do not group words according to any relevant morphological property such as gender or number, making it impossible for the grammar to model agreement patterns. At the same time, inspecting some of the bad parses reveals several clear cases of agreement mistakes. Can morphological agreement be incorporated in the parsing model?

8.1 Forcing Morphologically Motivated Splits

Our initial attempts focused on making the PCFG-LA learning procedure pick up on agreement-relevant state-splits. When neither the core tag set nor the non-terminals encode gender and number information, it is very hard for the parser to pick up on agreement patterns.[22]

We attempted to train the parser on trees which mark the agreement features (either the gender, the number, or both) either on the POS tags, the relevant constituents, or

22 In the external lexicon case, the external lexicon tags do encode the morphological features, making it possible in principle for the parser to learn to map certain substates to certain agreement features. This did not happen in practice, arguably because other structural factors were more powerful than the agreement ones.

both. Annotating agreement features on the POS tag–level made the parsing much slower, but did make the parser assign certain split categories to certain gender–number combinations, and sampling utterances from the learned grammar did indicate a notion of grammatical agreement. This did not improve parsing accuracy, however—and even slightly degraded it.

When propagating the agreement features and annotating them on the constituent level, parsing accuracy dropped considerably. When inspecting the learned grammar we observe that most of the agreement-annotated constituents (e.g., $NP_{Masc,Plural}$) were still fully split, indicating that the parser picked on patterns which were orthogonal to the agreement mechanism. The pre-splitting according to agreement-features properties caused data sparseness, aided over-fitting, and hurt parsing performance: The smoothing procedure of the BerkeleyParser shares some probability-mass between various splits of the same symbol, but was not applied in our case (no information flowed between, for example, $NP_{Masc,Plural}$ and $NP_{Masc,Singular}$). We attempted to counter this effect by changing the smoothing mechanism of the BerkeleyParser to share information also between the manually split symbols. This brought parsing accuracy back to the initial level, but also caused the parser to, again, not model agreement very well. The reason for this is clear in hindsight: Morphological agreement is an absolute concept, not a fuzzy one (things can either agree or not). Smoothing the probabilities between the different morphology-based split-licensed grammar rules that allow morphological disagreement, and made the grammar lose its discrimination power. This was then reinforced by the training process, which picked on other syntactic factors instead, and further phased out the agreement knowledge.

A note on product-grammars. In recent work, Petrov (2010) showed that a committee of latent-variable grammars encoding different grammatical preferences can be combined into a product-grammar that is better than the individual ensemble members. Petrov created the ensemble by training several PCFG-LA parsers on the same data, but using different random seeds when initializing the EM starting point. We attempted to create a similar ensemble by providing the learning process with different linguistically motivated tree annotations (with and without encoding agreement features, with and without encoding definiteness, etc.). The combined parser did increase the performance level over that of the individual parsers, but an ensemble with the same number of components that was produced using the random-seeds approach produced far superior results. This reinforces the findings of Petrov (2010) who also reports that the ensemble creation using random initialization is exceptionally strong and outperforms other methods of ensemble creation.[23]

8.2 Agreement as Filter

We now turn to suggest an approach to modeling agreement, which rests on the following principles:

- Agreement can be modeled as a set of hard (not probabilistic) constraints.
- Agreement is completely orthogonal to the other aspects of the grammar.

23 The product grammar approach with random seeds works well and is effective for improving the accuracy of Hebrew parsing. As it is completely orthogonal to the approaches presented in this article, however, we chose not to discuss it further other than commenting on its applicability.

Based on these principles, we suggest treating agreement as a filter, a device that can rule out illegal parses. Under the agreement-as-filter framework, we want the parser to produce the most probable parse according to its grammar *and subject to hard agreement constraints*. This approach completely decouples the grammar from the agreement verification mechanism. The agreement information is not modeled in the grammar and is not used to guide the search for the best parse. Instead, it is a separate process that imposes hard constraints on the search space and rules out parts of it completely. That is, agreement is a part of the parser and not of the grammar. This is similar in spirit to ideas from constraint-based grammars such as LFG (Falk 2001) and HPSG (Pollard and Sag 1994), which also model aspects of the syntax as Boolean constraints.

Grammatical agreement is a relation between constituents. The relevant morphological features are propagated from one of the leaves up to the constituent level. When constituents are combined to form a larger constituent, their morphological features are assigned to the newly created constituent according to language-specific rules (it is possible that different morphological features will be assigned by different constituents). An agreement violation occurs when two or more constituents assign conflicting features to their parent.

Implementation. In the implementation, an agreement-verification mechanism is manually constructed (not learned) based on a set of simple, language-dependent rules. First, we provide a set of rules to propagate the morphological agreement features from the leaves to the constituents. Then, we specify an additional set of rules to inspect local tree configuration and identify agreement violations (the Hebrew set of rules is described later, along with a concrete example). The feature-propagation mechanism works bottom–up and the agreement verification rules are very local, making it possible to integrate the filtering mechanism into a bottom–up CKY parsing algorithm (refusing to complete a constituent if it violates an agreement constraint). We did not pursue this route for the experiments in this work, however. Instead, we opted for an approximation in which we take the 100-best trees for each sentence, and choose the first tree that does not have an agreement violation (this is an approximation because the 100-best trees may not contain a valid tree, in which case we accept the agreement violation and choose the first-best tree). The specific details of the Hebrew agreement filter are given in the appendix.

Verifying the hard-constraint property. We verified that the hard constraint assumption works and that the agreement verification mechanism is valid by applying the procedure to the gold-standard trees in the training-set and checking that (1) the propagated features agree with the manually marked ones, and (2) none of the training-set trees were filtered due to agreement violation. We did find a few cases in which the propagated features disagreed with the manually marked ones, and a few gold-standard trees that the mechanism marked as containing an agreement violation. All of these cases were due to mistakes in the manual annotation.

Connections to parse-reranking. Our implementation is similar to parse-reranking (Charniak and Johnson 2005; Collins and Koo 2005). Indeed, if we were to model agreement as soft constraints, we could have incorporated this information as features in a reranking model. The filter approach differs in that it poses hard constraints and not soft ones, pruning away parts of the search space entirely. Thus, the use of k-best list is merely a technical detail in our implementation—the agreement information is

easily decomposable and the hard constraints can be efficiently incorporated into the CKY search procedure.

9. Evaluation and Results

Data set. For all the experiments we use Version 2 of the Hebrew Treebank (Guthmann et al. 2009), with the established test-train-dev splits: Sentences 484–5,740 are used for training, sentences 1–483 are the development set, and sentences 5,741–6,220 are used for the final test set.

Evaluation Measure. In the cases where the gold segmentation is given, we use the well-known `evalb` F_1 score. Namely, each tree is treated as a set of labeled constituents.[24] Each constituent is represented as a 3-tuple $\langle i, j, L \rangle$, in which i and j are the indices of the first and the last words in the constituent, respectively, and L is the constituency label. For example, $(2, 4, \text{NP})$ indicates an NP spanning from word 2 to word 4. The performance of a parser is evaluated based on the amount of constituents it recovered correctly. Let G denote the set of constituents in a gold-standard constituency tree, and P denote the set of constituents in a predicted tree. Precision (P), recall (R), and F_1 are defined as:

$$\text{precision} = \frac{|G \cap P|}{|P|} \qquad \text{recall} = \frac{|G \cap P|}{|G|}$$

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

F_1 ranges from 0 to 1, and it is 1 iff both precision and recall are 1, indicating the trees are identical. We report numbers in precentages rather than fractions.

When measuring the performance of models in which the token-segmentation is predicted and can contradict the gold-standard, a generalization of these measures is used. Instead of representing a constituent by a triplet $\langle i, j, L \rangle$, each constituent is represented by a pair containing the concatenation of the words at its yield, and its label L. This measure was suggested by Tsarfaty (2006a) and used in subsequent work (Tsarfaty 2006b; Goldberg and Tsarfaty 2008; Goldberg et al. 2009; Goldberg and Elhadad 2011). This is equivalent to reassigning the i and j indices to represent character positions instead of word numbers. When the yields of the gold standard and the predicted trees are the same, this is equivalent to the standard evaluation measure using the $\langle i, j, L \rangle$ triplets of word indices and a label, and it will produce the same precision, recall, and F_1 as above.

Effect of external lexicon. We start by evaluating the effect of extending the parser's lexical model with an external lexicon, as described in Section 6.1. The rare-word threshold is set to 100. We use the morphological analyzer described in Section 2.3.3. We test two conditions: UNIFORM, in which the $P(T_{\text{ext}}|w)$ distribution is uniform over all the

24 This assumes unary-chains do not contain cycles.

Table 4
Dev-set results when incorporating an external lexicon.

Setting	Ext-Lexicon/Probs	F_1 (4 cycles)	F_1 (5 cycles)
Seg Oracle	NONE	83.13	83.39
Pipeline	NONE	75.98	76.65
Seg Oracle	UNIFORM	84.92	84.56
Pipeline	UNIFORM	77.53	77.35
Seg Oracle	HMM-BASED	86.17	85.79
Pipeline	HMM-BASED	78.75	78.78

analyses suggested by the morphological analyzer for the word, and HMM-BASED in which the $P(T_{\text{ext}}|w)$ distribution is based on pseudo-counts from the final round of EM–HMM training of the semi-supervised POS tagger described in Section 2.3.4. Results are presented in Table 4.

Incorporating the external lexicon helps both in the case where the correct segmentation is assumed to be known, as well as in the pipeline case where the segmentation is automatically induced by a sequential tagger. Incorporating the semi-supervised lexical probabilities learned over large unannotated corpora (HMM-BASED) further improves the results, up to 86.1 F_1 for the gold-segmentation case and 78.7 F_1 for the pipeline case. The pipeline model still lags behind the gold-segmentation case, indicating that the correct segmentation is very informative for the parser.

Joint segmentation and parsing. Having established that the external lexicon can be effectively incorporated into the parser, we turn to evaluate the method for joint segmentation and parsing. We follow the same conditions as before (UNIFORM and HMM-BASED lexical probabilities), but in this set of experiments the parser is allowed to choose its preferred segmentation using the lattice-parsing methodology presented in Section 7.2. The lattice is constructed according to the analyses licensed by the morphological analyzer. Table 5 lists the results. Lattice parsing is effective, leading to an improvement of about 2–3 F_1 points over the pipeline model.

Agreement filter. We now turn to add the agreement filtering on top of the lexicon-enhanced models. In this setting, the model outputs its 100-best trees for each sentence, agreement features are propagated, and agreement violations are checked as described

Table 5
Dev-set results when using lattice parsing on top of an external lexicon/analyzer.

Setting	Ext-Lexicon/Probs	F_1 (4 cycles)	F_1 (5 cycles)
Pipeline	UNIFORM	77.53	77.35
Lattice (Joint)	UNIFORM	80.35	80.31
Pipeline	HMM-BASED	78.75	78.78
Lattice (Joint)	HMM-BASED	80.91	80.46

Table 6
Dev-set results of using the agreement-filter on top of the lexicon-enhanced parser (starting from gold segmentation).

Setting	Ext-Lexicon/Probs	F_1 (4 cycles)	F_1 (5 cycles)
No Agreement	UNIFORM	84.92	84.56
Agreement as Filter	UNIFORM	85.30	84.52
No Agreement	HMM-BASED	86.17	85.79
Agreement as Filter	HMM-BASED	86.55	86.25

in Section 12, and the first tree that does not contain any agreement violation is returned as the final parse for the sentence (or the first-best tree in case that all of the output trees contain an agreement violation). Table 6 lists the results when agreement filtering is performed on top of parses based on gold segmentation, and Table 7 lists the results when agreement filtering is performed on top of a lattice-based parsing model that does not assume gold segmentation is available.

Discussion of agreement filter results. Although the agreement filter does not hurt the parser performance, the benefits from it are very small. To understand why that is the case, we analyzed the 1-best parses produced by the 5-cycles-trained grammar on the gold-segmented development set (these conditions corresponds to the last column of the third row in Table 6). The analysis revealed the following reasons for the low impact of the agreement filter: (1) The grammar is strong enough to produce fairly accurate structures, which have very few agreement mistakes to begin with, and (2) fixing an agreement mistake does not necessarily mean fixing the entire parse—in some cases it is very easy for the parser to fix the agreement mistake and still produce an incorrect parse for other parts of the structure.

The 1-best trees of the 480 sentences of the development set contain 22,500 parse-tree nodes. Of these 22,500 nodes, 2,368 nodes triggered a gender-agreement check: about 10% of the parsing decisions could benefit from gender agreement. Of the 2,368 relevant nodes, however, 130 nodes involved conjunctions or possessives, and were outside of the scope of our agreement verification rules. Of the remaining 2,238 parse-tree nodes, 2,204 passed the agreement check, and only 34 nodes (1.5% of the relevant nodes, and 0.15% of the total number of nodes) were flagged as gender-agreement violations. Similarly for number agreement, 2,244 nodes triggered an agreement check, of which 2,131 nodes could be handled by our system. Of these relevant nodes, 2,109 nodes passed the gender-agreement check, and only 23 nodes (1.07% of relevant nodes,

Table 7
Dev-set results of using the agreement-filter on top of the lexicon-enhanced lattice parser (parser does both segmentation and parsing).

Setting	Ext-Lexicon/Probs	F_1 (4 cycles)	F_1 (5 cycles)
No Agreement	UNIFORM	80.35	80.31
Agreement as Filter	UNIFORM	80.55	80.74
No Agreement	HMM-BASED	80.91	80.46
Agreement as Filter	HMM-BASED	81.04	80.72

Table 8
Numbers of parse-tree nodes in the 1-best parses of the development set that triggered gender or number agreement checks, and the results of these checks.

	Gender Agreement	Number Agreement
Triggered agreement check	2,368	2,244
Could be handled by the system	2,238	2,131
No agreement violation	2,204	2,109
Agreement violation	34	23

and 0.1% of the total nodes) were flagged as agreement violations. The numbers are summarized in Table 8. It is clear that the vast majority of the parser decisions are compatible with the agreement constraints.

Turning to inspect the cases in which the agreement filter caught an agreement violation, we note that the agreement filter marked 51 of the 480 development sentences as having an agreement violation in the 1-best parse—about 10% of the sentences could potentially benefit from the agreement filter. For 38 of the 51 agreement violations, the agreement violation was fixed in the tree suggested in the 100-best list. We manually inspected these 51 parse trees, and highlight some the trends we observed. In the 13 cases in which the 100-best list did not contain a fix to the agreement violation, the cause was usually that the 1-best parse had many mistakes that were not related to the agreement violation, and diversity in the 100-best list reflected fixes to these mistakes without affecting the agreement violation. Another cause of error was an erroneous agreement mistake due to an omission in the lexicon. Of the 38 fixable agreement violations, 25 were local to a noun-phrase, 10 were cases of subject–verb agreement, and the remaining three were either corner-cases or harder to categorize. The *subject–verb agreement* violations were handled almost exclusively by keeping the structure mostly intact and changing the `NPSUBJ` label to some other closely related label that does not require verb agreement, usually `NP`. This is a good strategy for fixing subject-less sentences (about half of the cases), but it is only a partial fix in case the subject should be assigned to a different NP (which does not happen in practice) or in case a more drastic structural change to the parse-structure is needed. In one of the 10 cases, the subject–verb agreement mistake indeed resulted in a structural change that improved the overall parse quality. The NP internal agreement violations include many cases of noun-compound attachments, and some cases involving coordination. The corrections to the agreement violation were mostly local, and usually resulted in correct structure, but sometimes introduced new errors. Figure 5 presents some examples of the different cases. Our overall impression is that for NP internal mistakes the agreement-filtering method was mostly doing the right thing.

To conclude, the agreement filter is useful in overcoming some errors and providing better parses, especially with respect to noun-compound construct-state constructions. Due to the limited number of parsing mistakes involving agreement violations, however, and because of the local nature of the agreement-violation mistakes, the total effect of the agreement filter on the final parsing score is small.

10. The Final Model

Finally, we evaluate the best performing model on the test set. Table 9 presents the results of parsing the test set while incorporating the external lexicon and using the

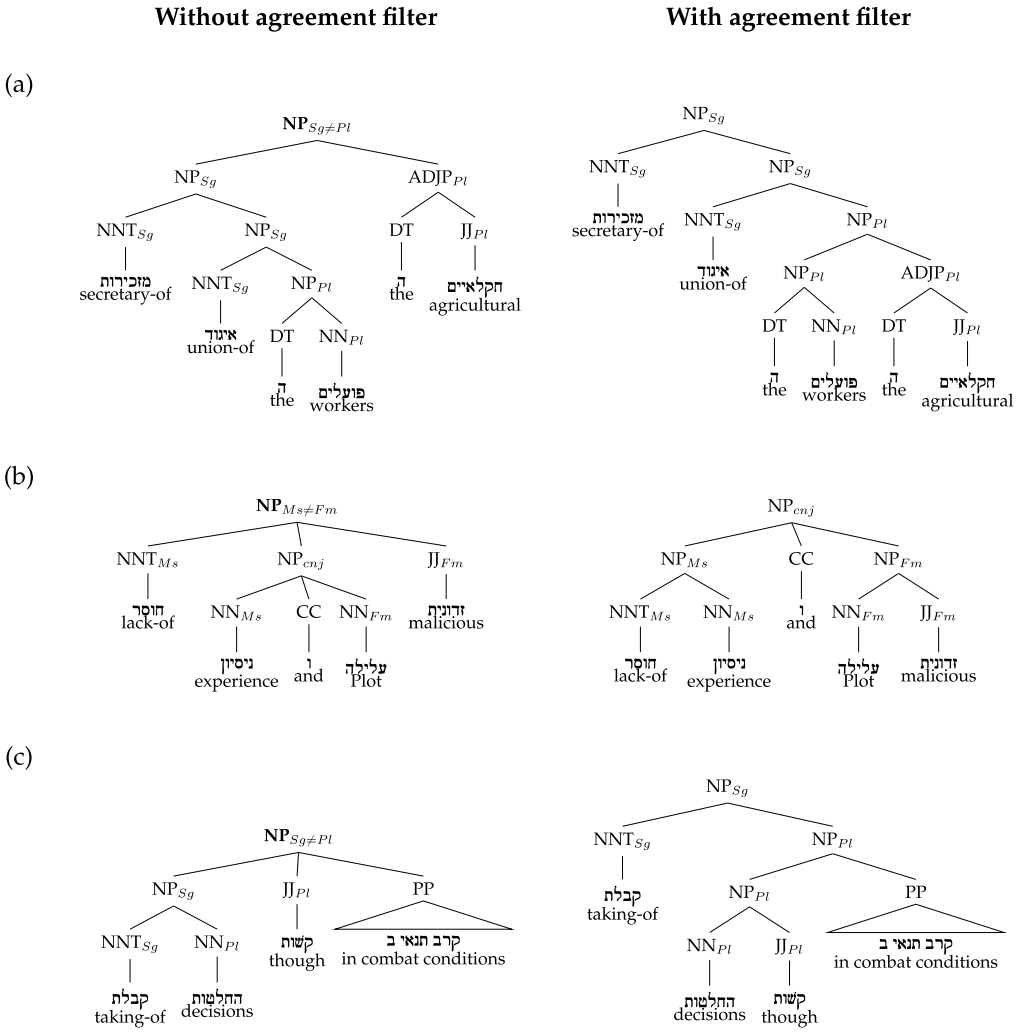

Figure 5
NP agreement violations that were caught by the agreement filter system. (a) Noun-compound case that was correctly handled. (b) Case involving conjunction that was correctly handled. (c) A case where fixing the agreement violation introduces a PP-attachment mistake.

Table 9
Test-set results of the best-performing models.

Setting	Model	F_1 (4 cycles)
Gold Segmentation	HMM-Based External Lexicon	85.67
	+ Agreement	85.70
Lattice-parsing	HMM-Based External Lexicon	76.87
	+ Agreement	76.95

HMM-based probabilities, for a grammar trained for four split-merge iterations. This grammar is applied both to the gold-segmentation case and to the realistic case where segmentation and parsing are performed jointly using lattice-parsing. We also test the effectiveness of the agreement-filter in both situations.

Agreement information does not hurt performance, but contributes very little to the final accuracy—additionally on the test sentences, the parser makes very few agreement mistakes to begin with.

Consistent with previous reports (Tsarfaty 2010), the test set is somewhat harder than the development set. With gold-segmentation, the models achieve accuracies of 85.70% F_1. In the realistic scenario in which the segmentation is induced by the parser, the accuracies are around 76.9% F_1. We verified that the HMM-based lexical probabilities also outperform the Uniform probabilities on the test set (the F_1 scores when using uniform lexical probabilities are 84.06 and 76.30 for the gold and induced segmentations, respectively). These are the best reported results for parsing the test-set of the Hebrew Treebank.

11. Related Work in Parsing of Morphologically Rich Languages

Coping with unknown words. Several papers show that the handling of unknown words is a major component to be considered when adapting a parser to a new language. For example, the work in Attia et al. (2010) uses language-specific unknown-word signatures for several languages based on various indicative prefixes and suffixes, and Huang and Harper (2009) suggest a Chinese-specific model based on the geometric average of the emission probabilities of the individual characters in the rare or unknown word. Another method of coping with lexical sparsity is word clustering. In Candito and Crabbé (2009), the authors demonstrate that replacing words by a combination of a morphological signature and a word-cluster (based on the linear context of a word in a large unannotated corpus) improves parsing performance for French. The technique provides more reliable estimates for in-vocabulary words (a given cluster appears more frequently than the actual word form), and it also increases the known vocabulary: Unknown words may share a cluster with known words.

Arabic. Arabic is similar to Hebrew in the challenges it presents for automatic parsing. Most early work on constituency parsing of Arabic focused on straightforward adaptations of Bikel's parser to Arabic, with little empirical success. Attia et al. (2010) show that parsing accuracies of around 81% F_1 can be achieved for Arabic (assuming gold word segmentation) by using a PCFG-LA parser with Arabic-specific unknown-word signatures. Recently, Green and Manning (2010) report on an extensive set of experiments with several kinds of tree annotations and refinements, and report parsing accuracies of 79% F_1 using the Stanford-parser and 82% F_1 using the PCFG-LA BerkeleyParser, both when assuming gold word segmentation. The work of Green and Manning also explored the use of lattice-parsing as suggested in Section 7 of this article, as well as earlier in Goldberg and Tsarfaty (2008) and Cohen and Smith (2007), and report promising results for joint segmentation and parsing of Arabic (an F_1 score of 76% for sentences of up to 70 words). The best reported results for parsing Arabic when the gold word segmentation is not known, however, are obtained using a pipeline model in which a tagger and word-segmenter is applied prior to a manually state-split constituency parser, resulting in an F-score of 79% F_1 (for sentences of up to 70 words) (Green and Manning 2010).

Hebrew and relational-realizational parsing. Some related work deals directly with constituency parsing of Modern Hebrew. The work of Tsarfaty and Sima'an (2007) experiments with grammar refinement for Hebrew, and shows that annotating definiteness and accusativity of constituents, together with parent annotation, improves parsing accuracy when gold word segmentation is available.

The Relational Realizational (RR) line of work presented in Tsarfaty et al. (Tsarfaty and Sima'an 2008; Tsarfaty, Sima'an, and Scha 2009; Tsarfaty and Sima'an 2010; Tsarfaty 2010) handles the constituent-order variation in Hebrew by presenting a separation between the *form* and *function* aspects of the grammar. Briefly, whereas plain treebank-derived grammars have rules such as S → NP VP PP NP PP that are applied in a single step, the RR approach suggests a generative model in which the generation of flat clausal structures is decomposed into three distinct steps. First, in the *projection* step, a non-terminal generates the kinds of its children without specifying their form or the order between them, using rules of the form S → {OBJ,SBJ,PRED,COM,Adjunct}@S. Second, in the *configuration* step, an order is chosen based on a separate ordering distribution, using rules of the form

{OBJ,SBJ,PRED,COM,Adjunct}@S → SBJ@S PRED@S Adj@S OBJ@S COM@S.

Third, in the *realization* step, each functional element receives a specific form, using rules of the form SBJ@S → NP or Adj@S → PP. The realization rules can encode syntactic properties that are required by the grammar for the given function—for example, a rule such as OBJ@S → $NP_{def,acc}$ captures the requirement that definite objects in Hebrew must be marked for accusativity using the את marker, and the rest if the generative process will generate the object NP according to this specified constraint. This kind of linguistically motivated separation of form and function is shown to produce models with fewer parameters and result in better parsing accuracies than plain (or head-driven) PCFGs derived from the same trees.

The relational-realizational model can accommodate agreement information. It is shown in Tsarfaty and Sima'an (2010) that, given gold-standard POS tags that include the gender and number information for individual words, RR models enriched with gender and number agreement information can provide Modern Hebrew parsing accuracies of 84% F_1 for sentences of up to 40 words, the highest reported number for Modern Hebrew parsing based on gold POS tags and word-segmentation by the time of its publication.

Although the RR framework is well motivated linguistically and appealing aesthetically, in the current work we chose to rely on the extreme markovization employed by the PCFG-LA BerkeleyParser in order to cope with the constituent order variation, and to model agreement as an external filter that is orthogonal to the grammar. The approach taken in this article provides state-of-the-art results for Hebrew constituency parsing. We leave the question of integrating the RR approach with the approach presented here to future work.

12. Conclusions

We presented experiments on Hebrew Constituency Parsing based on the PCFG-LA methodology of Petrov et al. (2006). The PCFG-LA model performs well out-of-the-box, especially when the gold POS tags are available to the parser. It is possible to improve the learned grammar, however, by specifying some manual state-splits, specifically

distinguishing between modal, finite, and infinitive verbs, and explicit marking of subject-NPs.

Parsing accuracies drop considerably when the gold POS tags are not available, and drop even further when using non-gold segmentation. A large part of the drop when the gold POS tags are not available is due to the large percentage of lexical events that are unseen or seen only a few times in the training set. This drop can be mitigated by extending the lexical coverage of the parser using an external lexical resource such as a wide-coverage morphological analyzer for mapping lexical items to their possible POS tags. The POS-tagging schemes assumed by the treebank and the morphological analyzer need not be compatible with each other: We present a method for bridging the POS tags differences between the two resources. The morphological analyzer does not provide lexical probabilities. Parsing accuracies can be further improved by using lexical probabilities which are derived in a semi-supervised fashion based on the morphological analyzer and a large corpus of unannotated text.

The correct token-segmentation is very important for achieving high-quality parses, and when the gold segmentation is not available, parsing results drop considerably. It is better to let the parser induce its preferred segmentation in interaction with the parsing process rather than to use a segmentation based on an external sequence model in a pipeline fashion. The joint induction of both the syntactic structure and the token-segmentation can be performed by representing the possible segmentations in lattice structure, and using lattice parsing. Joint parsing and segmentation is shown to outperform the pipeline approach. The parsing accuracies with non-gold segmentation are still far below the accuracies when the gold-segmentation is assumed to be known, however, and accurate parsing with non-gold segmentation remains a challenging open research problem.

The learned PCFG-LA grammar is not capable of modeling agreement information. We considered methods of using morphological agreement information to improve parsing accuracy. We propose modeling agreement information as a filtering process that is orthogonal to the grammar used for parsing. The approach works in the sense that, in contrast to other methods of using agreement information, it does not degrade parsing accuracy and even improves it slightly. The benefit from the agreement filtering is small, however: With the strong grammar induced by the PCFG-LA training procedure, the parser makes very few agreement mistakes to begin with. Modeling morphological agreement is probably more useful in syntactic generation than in syntactic parsing. We expect the filtering approach we propose to be proven useful for tasks involving syntactic generation, such as target-side-syntax machine translation into a morphologically rich language.

Overall, we presented four enhancements to the PCFG-LA mechanism in order to adapt it to parsing Hebrew: the introduction of manual, linguistically motivated state-splits; extending the lexical coverage of the parser using an external morphological analyzer; performing segmentation and parsing jointly using a lattice parser; and incorporating agreement information in a filtering framework. Together, these enhancements result in the best published results for Hebrew Constituency Parsing to date.

Appendix A: The Hebrew Agreement Filter

Hebrew syntax requires agreement in gender, number, and person. The implementation considers only the gender and number features, which are the most common. Each of

the features can take one of five values `Masculine`, `Feminine`, `Both`, `Unknown`, and `NA` for *Gender*, and `Singular`, `Plural`, `Both`, `Unknown` and `NA` for *Number*. Masculine, Feminine, Singular, and Plural are self-explanatory, and are assigned when the feature value is obvious. NA means that the feature is irrelevant for the given constituent (adverbs and PPs do not carry gender or number features). Both and Unknown are assigned when we are uncertain about the corresponding feature value. Both and Unknown are identical in the sense that they leave the feature value unspecified, and have the same effect on the filtering process. From a practical perspective they could be collapsed into the same category. We chose to maintain the distinction between the two cases because they have slightly different semantics. Both indicates that both options are possible (for example, the form **ילדות** is ambiguous between the plural *girls* and the singular *childhood*, and the titular **דר**, *Dr.* can refer both to males and females), whereas Unknown means that the feature value could not be computed due to a limitation of the model (for example, there is no clear rule as to the gender of a conjunction which coordinates masculine and feminine NPs, and we are currently unable to accurately infer the gender and number associated with certain complex quantifiers such as **רוב** (*most*). Compare: **נאכל רוב העוגה, רוב העוגה נאכלה, רוב הכיתה נשאר** ('most of the class$_{fem}$ stayed$_{masc}$, most of the cake$_{fem}$ was eaten$_{fem}$, most of the cake$_{fem}$ was eaten$_{masc}$').

Feature values are said to **agree** if they are compatible with each other. Feminine is compatible with NA, Both, and Unknown but not with Masculine. Similarly, Singular is compatible with NA, Both, and Unknown, but not with Plural.

Agreement cases. The system is designed to handle the following cases of morphological agreement:

NP level agreement between nouns and adjectives. **ארגז תפוחים ירוקים גדולים** ('box-of$_{Sg}$ apples$_{Pl}$ green$_{Pl}$ big$_{Pl}$') , **ארגז תפוחים ירוקים גדול** ('box-of$_{Sg}$ apples$_{Pl}$ green$_{Pl}$ big$_{Sg}$')

S level agreement between subject and verbs. **אחד הילדים הלך** ('[one-of the-kids]$_{Sg}$ walked$_{Sg}$')

Predicative agreement between the subject, ADJP, and copular element. **הוא חכם** ('he [is] smart$_{masc}$'), **היא היתה מדהימה** ('she was amazing/$_{fem}$'), but not with nouns **סמל היא היתה** ('she was a-symbol$_{masc}$').

Agreement between the Verb in a relativized SBAR and the realization of the Null-subject in the external NP.
הוועדה ש דנה בנושא ('the-committee$_{fem}$ which [*] discussed$_{fem}$ the-matter')

Morphological feature propagation. The first step of determining agreement is propagating the relevant features from the leaves up to the constituent level.

The procedure begins by assigning each leaf gender and number features. These are assigned based either on the TB tag assigned for the word if training on gold POS tags, or on the morphological analyzer entries for the given word (in most cases the number and gender features are easy to predict, even in cases where the core POS is not clear. In the relatively rare cases where the analyzer contains both a feminine and masculine (alt. singular and plural) analyses, feature value is marked as Both).

Table A.1
Gender and number percolation rules. FC = first child with non-NA gender/number. Rules for each constituent type are applied in order, until a condition holds. Rules for gender and number are applied independently of each other.

Constituent	Condition	Feature Values
SBAR	has REL and S children	S.features
SBAR	otherwise	NA
PREDP	has ADJP child	ADJP.features
PREDP	has AGR child and no NP child	AGR.features
PREDP	otherwise	NA
S	has VP child and no NP-Subj child	VP.features
S	has VB child and no NP-Subj child	VB.features
S	otherwise	NA
NNPG	always	U
NP	has NNT child	NNT.features
NP	has CDT and NP children	CDT.number NP.gender
NP	is a conjunction	gender=U number=Plural
NP	has a " child	U
NP	first child is NP, second is POS	NP.features
NP	has IN child	FC.gender number=U
NP	has child with non-NA gen/num	FC.gender FC.number
NP	otherwise	NA
ADJP	has JJT child	JJT.features
ADJP	has child with non-NA gen/num	FC.gender FC.number
ADJP	otherwise	NA
VP	has VB child	VB.features
VP	has VB-Modal child	VB-Modal.features
VP	has VP child	VP.features
VP	otherwise	NA
other	always	NA

After each leaf is assigned feature values, the features are propagated up the tree according to a set of rules such as the following (the complete set of rules is given in Table A.1):

- If the constituent is an NP and has a Construct-noun child, it is assigned the gender of the Construct-noun.
- If the constituent is a coordinated NP (has a CC child), set its number feature to plural.
- If the constituent is an S and it has VP child but no NP-Subject child, take the gender from the VP.

Agreement rules. Once the features are propagated from the leaves to a constituent, agreement is verified at the constituent level according to the following rules:
NP agreement rules:

- Agreement for coordinated NPs and Possessive NPs is not checked.

- If NP has an SBAR child, all the children up to the SBAR whose type is nominal or adjectival must agree in gender and number.
- If NP has an ADJP child, all the children up to the ADJP whose type is nominal or adjectival must agree in gender and number.

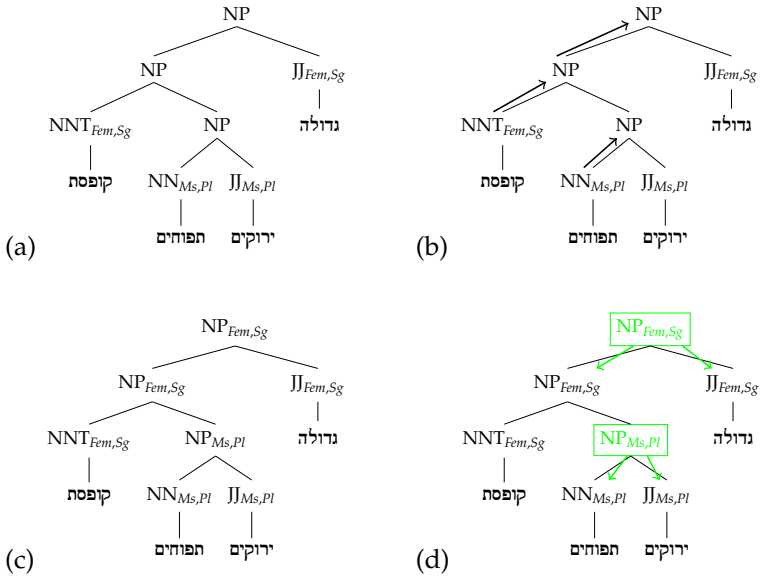

Figure A.1
Agreement annotation and validation example: correct tree. The sentence words translate to *box-of apples green big*, literally, *a big box of green apples*.

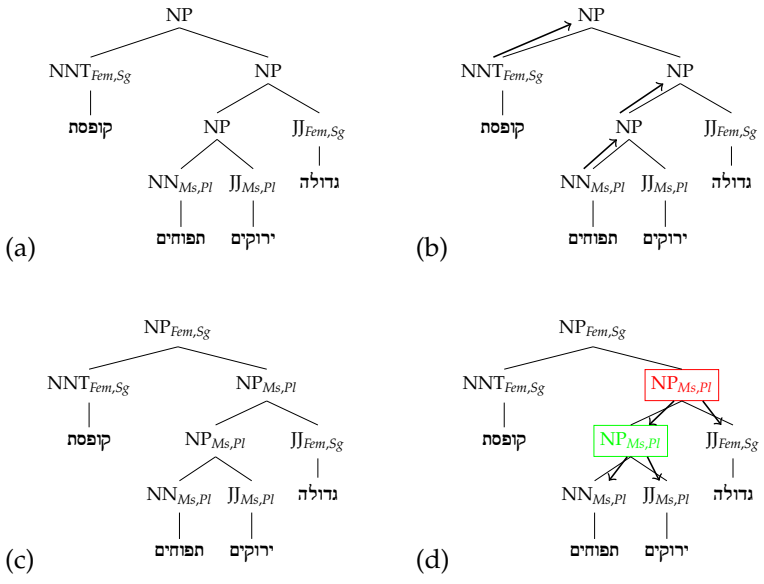

Figure A.2
Agreement annotation and validation example: incorrect tree, agreement violation. *box-of apples green big*, literally, *a big box of green apples*, though the parse tree suggests the interpretation *a box of big green apples*.

S agreement rule:

- All children of S with type in {NP-Subject, VP, VB, AUX, PREDP} must agree in their gender and number features.

ADJP agreement rule:

- All children of ADJP with type in {NP, NP-Subject, NN, JJ, ADJP} must agree in their gender and number features.

An example. Consider the tree in Figure A.1a. In the first stage (Figure A.1b), agreement features are propagated according to the rules in Table A.1, resulting in the annotated tree in Figure A.1c. Agreement is then validated in Figure A.1d (nodes in which an agreement rule applied and passed are marked in green). In contrast, the tree in Figure A.2a has an agreement mistake. As before, the agreement features are propagated according to the rules (Figure A.2b) resulting in Figure A.2c. Agreement validation fails at Figure A.2d (the node in which agreement validation was applied and failed is marked in red).

References

Abeillé, Anne, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In A. Abeillé, editor. *Treebanks: Building and Using Parsed Corpora*. Springer, Berlin, pages 165–188.

Adler, Meni. 2001. Hidden Markov model for Hebrew part-of-speech tagging. Master's thesis, Ben-Gurion University of the Negev.

Adler, Meni. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev.

Adler, Meni and Michael Elhadad. 2006. An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 665–672, Sydney.

Adler, Meni, Yoav Goldberg, David Gabay, and Michael Elhadad. 2008a. Unsupervised lexicon-based resolution of unknown words for full morphological analysis. In *Proceedings of ACL-08: HLT*, pages 728–736, Columbus, OH.

Adler, Meni, Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2008b. Tagging a Hebrew corpus: The case of participles. In *Proceedings of LREC 2008*, pages 3167–3174, Marrakech.

Attia, Mohammed, Jennifer Foster, Deirdre Hogan, Joseph Le Roux, Lamia Tounsi, and Josef van Genabith. 2010. Handling unknown words in statistical latent-variable parsing models for Arabic, English and French. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 67–75, Los Angeles, CA.

Bar-Haim, Roy, Khalil Sima'an, and Yoad Winter. 2005. Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 39–46, Ann Arbor, MI.

Bar-Haim, Roy, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering*, 14(2):223–251.

Billott, Sylvie and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151, Vancouver.

BGU Computational Linguistics Group. 2008. Hebrew morphological tagging guidelines. Technical report, Ben Gurion University of the Negev.

Cai, Shu, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 212–216, Portland, OR.

Candito, Marie and Benoît Crabbé. 2009. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International*

Conference on Parsing Technologies (IWPT'09), pages 138–141, Paris.
Candito, Marie, Benoît Crabbé, and Djamé Seddah. 2009. On statistical parsing of French with supervised and semi-supervised strategies. In *EACL 2009 Workshop Grammatical Inference for Computational Linguistics*, pages 49–57, Athens.
Chappelier, J., M. Rajman, R. Aragues, and A. Rozenknop. 1999. Lattice parsing for speech recognition. In *Sixth Conference sur le Traitement Automatique du Langage Naturel (TANL'99)*, pages 95–104, Cargése.
Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, MI.
Cohen, Shay B. and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 208–217, Prague.
Collins, Michael and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.
Crabbé, Benoît and Marie Candito. 2008. Expériences d'analyses syntaxique statistique du français. In *Proceedings of TALN*, pages 45–54, Avignon.
Dantzig, G. B. and P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research*, 8:101–111.
Falk, Yehuda N. 2001. *Lexical-Functional Grammar: An Introduction to Parallel Constraint-Based Syntax*. CSLI Publications, Stanford, CA.
Glinert, Lewis. 1989. *The Grammar of Modern Hebrew*. Cambridge University Press.
Goldberg, Yoav, Meni Adler, and Michael Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of ACL-08: HLT*, pages 746–754, Columbus, OH.
Goldberg, Yoav and Michael Elhadad. 2011. Joint Hebrew segmentation and parsing using a PCFGLA lattice parser. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 704–709, Portland, OR.
Goldberg, Yoav and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, OH.
Goldberg, Yoav, Reut Tsarfaty, Meni Adler, and Michael Elhadad. 2009. Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 327–335, Athens.
Green, Spence and Christopher D. Manning. 2010. Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 394–402, Beijing.
Guthmann, Noemie, Yuval Krymolowski, Adi Milea, and Yoad Winter. 2009. Automatic annotation of morpho-syntactic dependencies in a Modern Hebrew Treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT)*, pages 1–12, Groningen.
Hall, Keith. 2005. *Best-first Word-lattice Parsing: Techniques for Integrated Syntactic Language Modeling*. Ph.D. thesis, Brown University.
Huang, Zhongqiang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841, Singapore.
Itai, Alon and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98.
Jiang, Wenbin, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging—a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 522–530, Suntec.
Johnson, Mark. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.
Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo.
Lang, Bernard. 1974. Deterministic techniques for efficient non-deterministic parsers. In J. Loeckx, editor, *Automata, Languages and Programming*, volume 14 of

Lecture Notes in Computer Science. Springer, Berlin Heidelberg, pages 255–269.
Lang, Bernard. 1988. Parsing incomplete sentences. In *Proceedings of COLING*, pages 365–371, Budapest.
Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82, Ann Arbor, MI.
Netzer, Yael, Meni Adler, David Gabay, and Michael Elhadad. 2007. Can you tag the modal? You should! In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 57–64, Prague.
Petrov, Slav. 2009. *Coarse-to-Fine Natural Language Processing*. Ph.D. thesis, University of California at Berkeley.
Petrov, Slav. 2010. Products of random latent variable grammars. In *Proceedings of NAACL*, pages 19–27, Los Angeles, CA.
Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney.
Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, NY.
Petrov, Slav and Dan Klein. 2008. Parsing German with latent variable grammars. In *Proceedings of the Workshop on Parsing German*, pages 33–39, Columbus, OH.
Pollard, Carl and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.
Prescher, Detlef. 2005. Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 292–304, Porto.
Rush, Alexander M, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*, pages 1–11, Cambridge, MA.
Sima'an, Khalil. 1996. Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proceedings of COLING*, pages 1175–1180, Copenhagen.
Sima'an, Khalil. 1999. *Learning Efficient Disambiguation*. Ph.D. thesis, ILLC Dissertation Series, University of Amsterdam.
Sima'an, Khalil, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a tree-bank of Modern Hebrew text. *Traitement Automatique des Langues*, 42(2):1–32.
Smith, Noah A., David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of EMNLP*, pages 475–482, Vancouver.
Tsarfaty, Reut. 2006a. Integrated morphological and syntactic disambiguation for Modern Hebrew. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*, pages 49–54, Sydney.
Tsarfaty, Reut. 2006b. The Interplay of Syntax and Morphology in Building Parsing Models for Modern Hebrew. In *Proceedings of ESSLI Student Session*, pages 263–274, Malaga.
Tsarfaty, Reut. 2010. *Relational-Realizational Parsing*. Ph.D. thesis, ILLC Dissertation Series, University of Amsterdam.
Tsarfaty, Reut and Khalil Sima'an. 2007. Three-dimensional parametrization for parsing morphologically rich languages. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 156–167, Prague.
Tsarfaty, Reut and Khalil Sima'an. 2008. Relational-realizational parsing. In *Proceedings of CoLING*, pages 889–896, Manchester.
Tsarfaty, Reut and Khalil Sima'an. 2010. Modeling morphosyntactic agreement in constituency-based parsing of Modern Hebrew. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 40–48, Los Angeles, CA.
Tsarfaty, Reut, Khalil Sima'an, and Remko Scha. 2009. An alternative to head-driven approaches for parsing a (relatively) free word-order language. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 842–851, Singapore.