Tensors over Semirings for Latent-Variable Weighted Logic Programs

Esma Balkır[1] **Daniel Gildea**[2] **Shay B. Cohen**[1]
[1]ILCC, School of Informatics, University of Edinburgh
[2]Department of Computer Science, University of Rochester
esma.balkir@ed.ac.uk
gildea@cs.rochester.edu scohen@inf.ed.ac.uk

Abstract

Semiring parsing (Goodman, 1999) is an elegant framework for describing parsers by using semiring weighted logic programs. In this paper we present a generalization of this concept: latent-variable semiring parsing. With our framework, any semiring weighted logic program can be *latentified* by transforming weights from scalar values of a semiring to rank-n arrays, or tensors, of semiring values, allowing the modeling of latent variables within the semiring parsing framework. Semiring is too strong a notion when dealing with tensors, and we have to resort to a weaker structure: a partial semiring.[1] We prove that this generalization preserves all the desired properties of the original semiring framework while strictly increasing its expressiveness.

1 Introduction

Weighted Logic Programming (WLP) is a declarative approach to specifying and reasoning about dynamic programming algorithms and chart parsers. WLP is a generalization of bottom-up logic programming where proofs are assigned weights by combining the weights of the axioms used in the proof, and the weight of a theorem is in turn calculated by combining the weights of all its possible proof paths. The combinatorial nature of this procedure makes weighted logic programs highly suitable for specifying dynamic programming algorithms. In particular, Goodman (1999) presents an elegant abstraction for specifying and computing parser values based on WLP where the values could be drawn from any complete semiring. This generalizes the case of Boolean decision problems, probabilistic grammars with Viterbi search and other quantities of interest such as the best derivation or the set of all possible derivations. It is then possible to derive a general formulation of inside and outside calculations in a way that is agnostic to the particular semiring chosen.

Latent variable models have been an important component in the NLP toolbox. The central assumption in latent variable models is that the correlations between observed variables in the training data could be explained by unobserved, hidden variables. Latent variables have been used with grammars such as Probabilistic Context-Free Grammars (PCFGs), where each node in the parse tree is represented using a vector of latent state probabilities that further extend the expressiveness of the grammar (Matsuzaki et al., 2005).

The approach of adding latent variables to formal grammars have proven to be a fruitful one: in the context of PCFG parsing, Matsuzaki et al. (2005) show that latent variable PCFGs (L-PCFGs) perform on par with models hand-annotated with linguistically motivated features. Cohen et al. (2013) report that on the Penn Treebank dataset, L-PCFGs trained with either EM or a spectral algorithm provide a 20% increase in F1 over PCFGs without latent states. Gebhardt (2018) shows that the benefits of latent variables are not limited to PCFGs by successfully enriching both Linear Context-Free Rewriting Systems and Hybrid Grammars with latent variables, and demonstrates their applicability on discontinuous constituent parsing.

Given the usefulness of latent variables, it would be desirable to have a generic inference mechanism for any latent variable grammar. WLPs can represent inference algorithms for probabilistic grammars effectively. However, this does not trivially extend to latent-variable models because latent variables are often represented as vectors, matrices and higher-order tensors, and these taken together no longer form a semiring. This is because in the semiring framework, values for deduction items

[1]Our definition of a partial semiring is slightly different than those in the abstract algebra literature e.g. Steenstrup (1985).

Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task, pages 72–89

https://doi.org/10.18653/v1/P17

and for rules must all come from the same set, and the semiring operations must be defined over all pairs of values from this set. This does not allow for letting different grammar nonterminals be represented by vectors of different sizes. More importantly, it does not allow for a rule's value to be a tensor whose dimensionality depends on the rule's arity, as is generally the case in latent variable frameworks.

In this paper we start with a broad interpretation of latent variables as tensors over an arbitrary semiring. While a set of tensors over semirings is no longer a semiring, we prove that if the set of tensors have certain matching dimensions for the set of grammar rules they are assigned to, then they fulfill all the desirable properties relevant for the semiring parsing framework. This paves the way to use WLPs with latent variables, naturally improving the expressivity of the statistical model represented by the underlying WLP. Introducing a semiring framework like ours makes it easier to seamlessly incorporate latent variables into any execution model for dynamic programming algorithms (or software such as Dyna, Eisner et al. 2005, and other Prolog-like/WLP-like solvers).

We focus on CFG parsing, however the same latent variable techniques can be applied to any weighted deduction system, including systems for parsing TAG, CCG and LCFRS, and systems for Machine Translation (Lopez, 2009). The methods we present for inside and outside computation can be used to learn latent refinements of a specified grammar for any of these tasks with EM (Dempster et al., 1977; Matsuzaki et al., 2005), or used as a backbone to create spectral learning algorithms (Hsu et al., 2012; Bailly et al., 2009; Cohen et al., 2014).

2 Main Results Takeaway

We present a strict generalization of semiring weighted logic programming, with a particular focus on parser descriptions in WLP for context-free grammars. Throughout, we utilize the correspondence between axioms and grammar rules, deductive proofs and grammar derivations, and derived theorems and strings.

We assume that axioms/grammar rules come equipped with weights in the form of tensors over semiring values. The main issue with going from semirings to tensors over semiring values is that these weights need to be *well defined* in that any valid derivation should correspond to a sequence of well defined semiring operations. For CFGs, we give a straightforward condition that ensures this is the case. This essentially boils down to making sure that each non-terminal corresponds to a fixed vector space dimension. For example, if A corresponds to a space of d_1 dimensions, B to d_2 and C to d_3, then a rule $A \rightarrow B\ C$ would have a tensor weight in $d_2 \times d_3 \times d_1$.

As long as the weights are well defined, the standard definitions for the value of a grammar derivation and a string according to a semiring weighted grammar extend to the case of tensors of semirings. Weighted logic programming provides the means to declaratively specify an efficient algorithm to obtain these values of interest. In line with Sikkel (1998) and Goodman (1999) we present precise conditions for when a partial-semiring WLP describes a correct parser.

The value of the WLP formulation of parsing algorithms is that it provides a unified fashion in which dynamic programming algorithms can be extracted from the program description. This relies on the ability of a WLP to decompose the value of a proof to a combination of the values of the sub-proofs. Specifically, given a derivation tree, a WLP description automatically provides algorithms for calculating the inside and outside values. We provide analogous algorithms for calculating the inside and outside values for partial-semiring WLPs. Our outside formulation addresses the non-commutative nature of tensors themselves, and could be extended to cases where the underlying semiring is non-commutative using the techniques presented by Goodman (1998).

3 Related Work

"Parsing as deduction" (Pereira and Warren, 1983) is an established framework that allows a number of parsing algorithms to be written as declarative rules and deductive systems (Shieber et al., 1995), and their correctness to be rigorously stated (Sikkel, 1998). Goodman (1999) has extended the parsing as deduction framework to arbitrary semirings and showed that various different values of interest could be computed using the same algorithm by changing the semiring. This led to the development of Dyna, a toolkit for declaratively specifying weighted logic programs, allowing concise implementation of a number of NLP algorithms (Eisner et al., 2005).

The semiring characterization of possible values to assign to WLPs gave rise to the formulation of a number of novel semirings. One novel semiring of interest for purposes of learning parameters is the *generalized entropy semiring* (Cohen et al., 2008) which can be used to calculate the KL-divergence between the distribution of derivations induced by two weighted logic programs. Other two semirings of interest are *expectation* and *variance* semirings introduced by Eisner (2002) and Li and Eisner (2009). These utilize the algebraic structure to efficiently track quantities needed by the expectation-maximization algorithm for parameter estimation. Their framework allows working with parameters in the form of vectors in $\mathbb{R}^n$ for a fixed n, coupled with a scalar in $\mathbb{R}_{\geq 0}$. The semiring value of a path is roughly calculated by the multiplication of the scalars and (appropriately weighted) *addition* of the vectors. This is in contrast with our framework where weights could be tensors of arbitrary rank rather than only vectors, and the values of paths are calculated via tensor multiplication.

Finally, Gimpel and Smith (2009) extended the semiring framework to a more general algebraic structure with the purpose of incorporating non-local features. Their extension comes at the cost that the new algebraic structure does not obey all the semiring axioms. Our framework differs from theirs in that under reasonable conditions, tensors of semirings do behave fully like regular semirings.

4 Background and Notation

Our formalism could be used to enrich any WLP that implements a dynamic programming algorithm, but for simplicity, we follow Goodman (1999) and focus our presentation on parsers with a context-free backbone.[2]

4.1 Context-free Grammars

Formally, a Context-Free Grammar (CFG) is a 4-tuple $\langle N, \Sigma, \mathcal{R}, S \rangle$. The set of N denotes the non-terminals which will be denoted by uppercase letters A, B etc., and S is a non-terminal that is the special start symbol. The set of Σ denotes the terminals which will be denoted by lowercase letters a, b etc. $\mathcal{R}$ is the set of rules of the form $A \rightarrow \alpha$ consisting of one non-terminal on the left hand side (lhs), and a string $\alpha \in (N \cup \Sigma)^*$ on the right hand side (rhs). We will use $\alpha \Rightarrow \beta$ if β could be derived from α with the application of one grammar rule. We will say that a sentence $\sigma \in \Sigma^+$ could be derived from the non-terminal A if σ could be generated by starting with A and repeatedly applying rules in $\mathcal{R}$ until the right hand side contains only terminals, and denote this as $A \stackrel{*}{\Rightarrow} \sigma$. We will denote the language that a grammar G defines by $\mathcal{L}(G) = \{\sigma | S \stackrel{*}{\Rightarrow} \sigma\}$.

CFG derivations can naturally be represented as trees. We will use the notation $\langle r : T_1 \dots T_k \rangle$ to represent a tree that has the node r as its root and $T_1, \dots, T_k$ as its direct subtrees. We will use $\mathcal{D}_G$ to denote the set of all derivation trees that can be constructed with the grammar G, and $\mathcal{D}_G(\sigma)$ for all valid derivation trees that generate the sentence σ in G.

4.2 Semirings

A semiring is an algebraic structure similar to a ring, except that it does not require additive inverses.

Definition 1. *A **semiring** is a set $\mathbb{S}$ together with two operations $+$ and $\times$, where $+$ is commutative, associative and has an identity element 0. The operation of $\times$ is associative, has an identity element 1 and distributes over $+$.*

The set of non-negative integers together with the usual $\times, +, 0, 1$ is a semiring, and so are probability values in $[0, 1]$. Booleans {TRUE, FALSE} also form a semiring with $\times := \vee$, $+ := \wedge$, $0 :=$ FALSE and $1 :=$ TRUE.

There are a few less common semirings that provide useful values in parsing. The *Viterbi* semiring calculates the probability of the best derivation. It has values in $[0, 1]$, $+ :=$ max and $\times, 0, 1$ as standard. The *Derivation forest*, *Viterbi derivation* and *Viterbi n-best* semirings calculate the set of all derivations, the best derivation and the n-best derivations respectively. Unlike the previous examples, the $\times$ operation of these semirings is not commutative. In general, if the $\times$ operation in a semiring is commutative, we refer to it as a *commutative semiring*, and otherwise it is referred to as *non-commutative*. For precise definitions and detailed descriptions of these semirings see Goodman (1999).

[2] Note that given a grammar G in a formalism F and a string α, it is possible to construct a CFG grammar $c(G, w)$ from G and α (Nederhof, 2003). This construction is possible even for range concatenation grammars (Boullier, 2004) which span all languages that could be parsed in poly-time.

4.3 Weighted Logic Programming

A logic program consists of *axioms* and *inference rules* that could be applied iteratively to prove theorems. Inference rules are expressed in the form $\frac{A_1 \ldots A_k}{B}$ where $A_1 \ldots A_k$ are antecedents from which B can be concluded. Axioms are inference rules with no antecedents.

One way to express dynamic programming algorithms such as CKY is as logic programs. This approach takes the point of view of *parsing as deduction*: terms consist of grammar rules and *items* in the form of $[i, A, j]$ that correspond to the intermediate entries in the chart. Grammar rules are taken to be axioms, and the description of the parser is given as a set of inference rules. These can have both grammar rules and items as antecedents and an item as the conclusion. A logic program in this form includes a special designated goal item that stands for a successful parse.

Continuing with the example of CKY, consider the procedural description for how to obtain a chart item from smaller chart items if we have the rule $A \rightarrow B\ C$ in the grammar:

$$chart[i,A,j] := chart[i,A,j] \vee (chart[i,B,k] \wedge chart[k,C,j])$$

The corresponding inference rule in a logic program would be:

$$\frac{A \rightarrow B\ C \qquad [i,B,k] \qquad [k,C,j]}{[i,A,j]}$$

Note that in the inference rule above, $A \rightarrow B\ C$ is a rule template with free variables A, B, C. In general, the terms in inference rules can contain free variables, however for a logic program to describe a valid dynamic algorithm, every free variable in the conclusion of an inference rule must appear in its antecedents as well.

A *weighted* logic program is a logic program where terms are assigned values from a semiring. When paired with semiring operations, inference rules provide the description of how to compute the value of the conclusion given the values of the antecedents. The result of an application of a particular inference rule is the semiring multiplication of all the antecedents. The value of a term B is then calculated as the semiring sum of values obtained from inference rules that have B as their the conclusion.

4.4 Semiring Parsing

In the context of parsing, Goodman (1999) presents a framework where a grammar G comes equipped with a function w that maps each rule in G to a semiring value. Then, a grammar derivation string E consisting of the successive applications of rules $e_1, \ldots, e_n$ is defined to have the value $V_G(E) = \prod_{i=1}^{n} w(e_i)$, and the value of a sentence $\sigma \in \mathcal{L}(G)$ is defined as $V_G = \sum_{j=1}^{k} V_G(E_j)$ where $E_1, E_2, \ldots, E_k$ are the derivations of σ in G.

A parser specification is given in the form of a weighted logic program, referred to as *item-based description*. From these, the value of a derivation D is calculated recursively as follows:

$$V(D) = \begin{cases} w(D) & \text{if } D \text{ is a rule} \\ \prod_{i=1}^{k} V(D_i) & \text{if } D = \langle b : D_1, \ldots, D_k \rangle \end{cases}$$

where $\prod$ is the semiring product.

Let $inner(x)$ represent the set of all derivation trees headed by the item x. Then the value of x is:

$$V(x) = \sum_{D \in inner(x)} V(D)$$

where $\sum$ is the semiring addition. The value of a sentence is then equal to $inner(goal)$.

Given the definitions of value according to the grammar and the parser, Goodman (1999) provides a theorem for conditions of correctness:

Theorem 4.1. *(Goodman 1999, Theorem 1; informal) An item-based description I is correct if for every grammar G there exists a one-to-one correspondence between the grammar and item derivations, and these derivations get the same value regardless of weight function used.*

One caveat with calculating based on item-based derivations is that there is an ordering of items: we cannot compute the value of an item unless the values of all its children are computed already. For this, Goodman (1999) assumes that each item is assigned to a *bucket* so that if an item b depends on a, then $bucket(a) \leq bucket(b)$. If a bucket depends on itself, then it is considered a special *looping* bucket. For all the formulas we present in this the main paper we assume that the items belong to non-looping buckets. The formulas for looping buckets are provided in Appendix B.

For an item x, calculating its value might require summing over exponentially many derivation trees.

To address this, it is possible to provide a general formula that efficiently computes the inner value for an item (Goodman 1999, Theorem 2):

$$V(x) = \sum_{a_1,\ldots,a_k \text{ s.t. } \frac{a_1,\ldots,a_k}{x}} \prod_{i=1}^{k} V(a_i)$$

The other important value associated with an item x is its *outside* value $Z(x)$, which is the sum of values of derivation trees, modified so that x is removed with all its subtrees. This value is complementary to the inside values $V(x)$ (Goodman 1999, Theorem 4):

$$V(x) \times Z(x) = \sum_{D \text{ a derivation}} V(D)C(D,x)$$

where $C(D,x)$ is the count of the occurrences of item x in derivation D.

$Z(x)$ can likewise be calculated using a recursive formula if the values are from a commutative semiring (Goodman 1999, Theorem 5):

$$Z(x) = \sum_{\substack{j,a_1,\ldots,a_k,b \text{ s.t.} \\ \frac{a_1\ldots a_k}{b} \text{ and } x=a_j}} Z(b) \times \prod_{i=1}^{j-1} V(a_i) \times \prod_{i=j+1}^{k} V(a_i)$$

4.5 Tensor Notation

We use the term *tensor* to refer to an n-dimensional array of semiring values. We use $\mathbb{S}$ to denote a semiring and $\mathbf{A}, \mathbf{B}$ etc. to denote tensors. The element $\mathbf{A} \in \mathbb{S}^{a_1 \times a_2 \times \ldots \times a_n}$ will denote that $\mathbf{A}$ is a rank-n tensor of values drawn from $\mathbb{S}$, with the ith rank having dimension a_i. The entry in index $k_1, \ldots, k_n$ will be denoted with subscripts $\mathbf{A}_{k_1,\ldots,k_n}$.

5 Latent-variable Parsing as Tensor Weighted Logic Programs

For semiring parsing to work for latent-variable models it should allow weights to be vectors, matrices and tensors. In this section we present a framework that generalizes that of Goodman (1999), and is able to capture tensors over semirings as weights. Note that this includes scalars as a special case.

5.1 Semiring Operations

The main reason why tensors over semirings are not semirings is that with tensor weights, $\oplus$ and $\otimes$ become partially defined – not all elements can naturally be added or multiplied to any other element anymore. We refer to these structures as *partial semirings*. With some reasonable constraints, we show that $\oplus$ and $\otimes$ obey the semiring axioms in cases that are relevant for the semiring parsing framework.

Let $\mathbb{S}$ be the chosen underlying semiring, $+, \times$ to be the semiring operations and $\mathbf{0}, \mathbf{1}$ be the additive and multiplicative identity of the semiring respectively. The set of possible weights are defined as $\{\mathbb{S}^{d_1 \times \ldots \times d_n}\}$ for $n \in \mathbb{N}$, and $d_i \in \mathbb{N}$ for all $i \leq n$. $\oplus$ is a partial addition that is defined on two tensors $\mathbf{A}, \mathbf{B} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ as long as the dimensions of each of their ranks match. Then, the addition is defined component-wise:

$$(\mathbf{A} \oplus \mathbf{B})_{i_1,\ldots,i_n} := \mathbf{A}_{i_1,\ldots,i_n} + \mathbf{B}_{i_1,\ldots,i_n}$$

The additive identity is now a class of tensors, one for each unique list of tensor dimensions. The additive identity for any $\mathbf{A} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ is the tensor $\mathbf{Z} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ with $\mathbf{0}$ in every entry.

Multiplication is defined as the contraction of an index between two tensors with arbitrary number of ranks. Specifically, we consider the family $\otimes_{[k;l]}$ which contracts the kth rank of the first tensor with the lth rank of the second tensor. This is only defined if the two ranks to be contracted have the same dimension, as follows:

$$\begin{aligned}\left(\mathbf{A} \otimes_{[k;l]} \mathbf{B}\right)_{\substack{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1}, \\ j_{l+1},\ldots,j_m,i_{k+1},\ldots,i_n}} \\ := \sum_{i_k,j_l} \delta(i_k,j_l)\mathbf{A}_{i_1,\ldots,i_n} \times \mathbf{B}_{j_1,\ldots,j_m},\end{aligned}$$

where δ is the identity function that is equal to $\mathbf{1}$ if $i_k = j_l$ and $\mathbf{0}$ otherwise. Note that the ranks corresponding to $\mathbf{B}$ which are not contracted over go in between the ranks of $\mathbf{A}$, replacing where the contracted rank of $\mathbf{A}$ was. We will use $\otimes_j$ as a shorthand of $\otimes_{[j;1]}$, and in cases where $j = l = 1$, we will omit the subscript on $\otimes$ altogether.

More generally, we will allow multiplication operations that contract multiple consecutive dimensions. $\mathbf{A} \otimes^r_{[k;l]} \mathbf{B}$ will denote contracting rank k of $\mathbf{A}$ with rank l of $\mathbf{B}$, rank $k+1$ of $\mathbf{A}$ with rank $l+1$ of $\mathbf{B}$ and so forth until rank $k+r-1$ of $\mathbf{A}$ and $l+r-1$ of $\mathbf{B}$. Formally:

$$\begin{aligned}&\left(\mathbf{A} \otimes^r_{[k;l]} \mathbf{B}\right)_{\substack{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1}, \\ j_{l+r},\ldots,j_m,i_{k+r},\ldots,i_n}} := \\ &\sum_{\substack{i_k,\ldots,i_{k+r-1} \\ j_l,\ldots,j_{l+r-1}}} \left(\prod_{p=0}^{r-1} \delta\left(i_{k+p}, j_{l+p}\right)\right) \mathbf{A}_{i_1,\ldots,i_n}\mathbf{B}_{j_1,\ldots,j_m}\end{aligned}$$

We will use the notation $\mathbf{A} \otimes^* \mathbf{B}$ as a shorthand for $\mathbf{A} \otimes^{rank(\mathbf{A})} \mathbf{B}$ if $rank(\mathbf{A}) < rank(\mathbf{B})$ and $\mathbf{A} \otimes^{rank(\mathbf{B})} \mathbf{B}$ otherwise.

To make the presentation clearer, we will also use the notation $X \otimes [A_1, A_2, \ldots, A_k]$ to denote contraction of A_1 with the first rank of X, A_2 with the second and so forth. In other words $X \otimes [A_1, \ldots, A_n]$ is equivalent to $X \otimes_n A_n \otimes_{n-1} A_{n-1} \ldots \otimes_1 A_1$.

The multiplicative identity for $\mathbf{A} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ and $\otimes_k$ is the identity matrix $\mathbf{I} \in \mathbb{S}^{d_k \times d_k}$ where the diagonal entries are the multiplicative identity from the underlying semiring, and the non-diagonals are the additive identity. For $\mathbf{A} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ and $\otimes_k^r$ the multiplicative identity is a rank-$2r$ tensor $\mathbf{I} \in \mathbb{S}^{d_k \times \ldots \times d_{k+r-1} \times d_k \times \ldots \times d_{k+r-1}}$ and is defined as follows:

$$\mathbf{I}_{d_1,\ldots,d_r} = \prod_{i=0}^{\frac{n}{2}} \delta\left(d_i, d_{\frac{r}{2}+i}\right)$$

Lastly, as the higher order analogue of the transpose operator, we will define a permutation operator $\mathbf{A}^\pi$ where $\pi = [\pi_1, \pi_2, \ldots, \pi_r]$ is a permutation of $[1 \ldots r]$ and r is the rank of $\mathbf{A}$. The π_ith rank of $\mathbf{A}^\pi$ is equal to ith rank of $\mathbf{A}$.

The key property of semirings for purposes of efficient calculation of item values is the distributive property. This property also holds for tensors over semirings.

Lemma 5.1. *For any k, l, $\otimes_{[k;l]}$ distributes over $\oplus$*

A proof can be found in Appendix A.

5.2 Grammar Derivations

For a grammar G with a function w that provides a mapping from rules to tensor weights, we will define a value of a derivation via the derivation tree:

Definition 2. *Given a grammar G and a weight function w, the value of a derivation tree T is:*

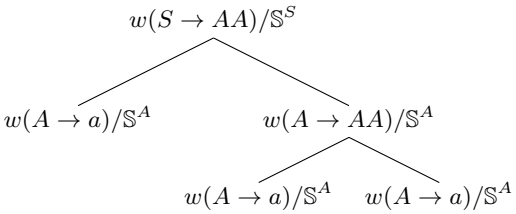
$$V_G^w(T) = \begin{cases} w(r) \\ \quad \textit{if } T = \langle r \rangle \\ w(r) \otimes [V_G^w(T_1), \ldots, V_G^w(T_k)] \\ \quad \textit{if } T = \langle r : T_1, \ldots, T_k \rangle \end{cases}$$

Note that there is no guarantee that this equation is defined for any arbitrary w. We will call a weight function w *well defined* for a grammar G if for all valid derivation trees T in G, $V_G^w(T)$ is defined. For CFGs there is a straightforward method to ensure that w is well defined:

Tensor dimensions of grammar rules:

$w(S \to AA) \in \mathbb{S}^{A \times A \times S}$ $\quad$ $w(A \to AA) \in \mathbb{S}^{A \times A \times A}$
$w(A \to a) \in \mathbb{S}^{A}$

Grammar derivation tree:



The value of the tree is given by the equation:

$$w(S \to AA) \otimes (w(A \to a), \\ (w(A \to AA) \otimes (w(A \to a), w(A \to a))))$$

Grammar derivation string:

$$S \xRightarrow[\mathbb{S}^{A \times A \times S}]{S \to AA} AA \xRightarrow[\mathbb{S}^{A \times S}]{A \to a} aA \xRightarrow[\mathbb{S}^{A \times A \times S}]{A \to AA} aAA \\ \xRightarrow[\mathbb{S}^{A \times S}]{A \to a} aaA \xRightarrow[\mathbb{S}^{S}]{A \to a} aaa$$

The value of the string is given by the equation:

$$w(S \to AA) \otimes w(A \to a) \otimes (A \to AA) \\ \otimes (A \to a) \otimes (A \to a)$$

Figure 1: Example derivation for the string "aaa". We illustrate the initial dimensions of the tensor values for the rules and also show the intermediate tensor dimensions during the calculation of the value of the grammar tree and the grammar string.

Lemma 5.2. *A set of weights w for a given CFG is well defined if there exist consistent dimensions d_i for each nonterminal A_i such that for all grammar rules $R : A_n \to \alpha_1 A_1 \alpha_2 \ldots \alpha_{n-2} A_{n-1} \alpha_n$, $w(R) \in \mathbb{S}^{d_1 \times \ldots \times d_n}$*

Proof is given together with Lemma 5.3.

Note that if a weight function for CFG is well defined, then the rank for the weights of rules with no non-terminals on their rhs is always 1.

Given a grammar derivation tree T, let us call the list of derivation rules $E : R_1, R_2, \ldots, R_n$ appearing in T ordered via depth-first, left-to-right manner a **grammar derivation string**.

Definition 3. *Given a CFG with tensor weights w, the **value of a grammar derivation string** is defined as:*

$$V_G^w(E) = \bigotimes_i w(R_i)$$

where the application of $\otimes$ proceeds from left to right as is standard.

For semirings, since the bracketing does not affect the final value of an expression, it is straightforward to show that the value of a grammar derivation tree corresponds to that of a grammar derivation string. With tensors over semirings this might fail with an arbitrary formalism F, and in the general we require the value of a derivation to be calculated with the bracketing induced by the derivation tree. However, for the special case of CFGs, the value of the grammar derivation tree and the value of its corresponding grammar derivation string are always equal. This means that for the computation of the value of the derivation, it is possible to replace the bracketing induced by the derivation tree by left-to-right bracketing without affecting the final value. Figure 1 demonstrates the calculation of the value of the tree and the string for the same derivation together with how the tensor dimensions of the intermediate results evolve with each step of the calculation.

Lemma 5.3. *Given a CFG G and a weight function w that fulfills the condition in Lemma 5.2, then w is well defined and $V_G^w(T) = V_G^w(E)$ for any grammar derivation tree T and corresponding grammar derivation string E.*

Proof. We will proceed by induction on the derivation tree. If T consists of only one rule r, then $V_G^w(T) = V_G^w(E)$. Furthermore, r does not have any non-terminals on its rhs, so $V_G^w(T) \in \mathbb{S}^{d_0}$ with $\mathbb{S}^{d_0}$ corresponding to the lhs non-terminal in r.

Otherwise, T has a labeled node r and the subtrees $T_1, \ldots, T_k$. Notice that if $A_0 \in \mathbb{S}^{d_1 \times \ldots \times d_n \times d_0}$, $A_1 \in \mathbb{S}^{d_2}$,..., $A_n \in \mathbb{S}^{d_n}$, then $A_0 \otimes [A_1, \ldots, A_n] = A_0 \otimes A_1 \otimes \ldots \otimes A_n$ due to all arguments within $[\ldots]$ being rank-1.

Because w fulfills the condition in Lemma 5.2, $w(r) \in \mathbb{S}^{d_1 \times \ldots \times d_k \times d_0}$ for some d_i where $\mathbb{S}^{d_0}$ is the space corresponding to the non-terminal on the lhs of r, and $\mathbb{S}^{d_i}$ is the space corresponding to the ith non-terminal appearing in the rhs of r for $i = 1, \ldots, k$. Then to complete the proof, it suffices to show that $V_G^w(T_i) \in \mathbb{S}^{d_i}$ for all subtrees T_i. This already holds for the base case. For each $T_i : \langle r_i : T_1', .., T_k' \rangle$, if $w(r_i) \in \mathbb{S}^{d_1^i \times \ldots \times d_k^i \times d_0^i}$ then by induction $V_G^w(T_i) \in \mathbb{S}^{d_0^i}$, where $\mathbb{S}^{d_0^i}$ is the space corresponding to the non-terminal in the lhs of R_i. For the derivation to be valid, this non-terminal needs to match the ith non-terminal in the rhs of R, hence $\mathbb{S}^{d_0^i} = \mathbb{S}^{d_i}$ □

5.3 Item-based Descriptions

Item-based descriptions are formal descriptions of various parsers for context-free grammars. Item-based descriptions consist of a set of deduction rules of the form $\frac{T_1 \ldots T_k}{Q} P_1 \ldots P_j$ where upper case letters could either be grammar rule templates (e.g. if $T_1 : A \rightarrow B\ C$ then any non-terminals from the grammar can be substituted for A, B, C) or for **items**. $T_1 \ldots T_k$ are referred to as antecedents, Q as the conclusion and $P_1 \ldots P_j$ are side conditions that the parser requires to execute the rule, but doesn't use the values of. Items correspond to chart elements in procedural descriptions of parsers, and are placeholders for intermediate results which can be combined to obtain the final result. The item-based description also provides a special **goal item** which is variable-free, and does not occur as a condition of any other inference rules.

Definition 4. *Given a grammar G and an item-based description I, a valid **item derivation tree** is defined as follows:*

- *For all $r \in G$, $\langle r \rangle$ is an item derivation tree.*
- *If $D_{a_1}, \ldots, D_{a_k}$ and $D_{c_1}, \ldots, D_{c_j}$ are derivation trees headed by $a_1, \ldots, a_k$ and $c_1, \ldots, c_j$ respectively, and $\frac{a_1 \ldots a_k}{b} c_1, \ldots, c_j$ is the instantiation of a deduction rule in I, then $\langle b : D_{a_1}, \ldots, D_{a_k} \rangle$ is also an item derivation tree.*

$inner_\sigma(x)$ denotes the set of all trees headed by x that occur in parses for σ. Formally, $D \in inner_\sigma(x)$ if D is headed by x and is a subtree of some $D' \in \mathcal{D}_{I(G)}(\sigma)$. The value of a derivation tree is calculated similarly to that of a grammar tree:

$$V_{I(G)}^w(D) = \begin{cases} w(D) & \text{if } D \text{ is a rule} \\ V_{I(G)}^w(D_1) \otimes [V_{I(G)}^w(D_2), \ldots, V_{I(G)}^w(D_n)] & \\ & \text{if } D = \langle b : D_1, \ldots, D_n \rangle \end{cases}$$

Notice that unlike the definition from Goodman (1999), the first antecedent in the inference rule has a special role in the calculation. Intuitively, our framework treats the value of the first antecedent as a *function*, and the trailing ones as the arguments. The interaction between the trailing antecedents is thus moderated through the value of the first antecedent, which corresponds to the requirement that

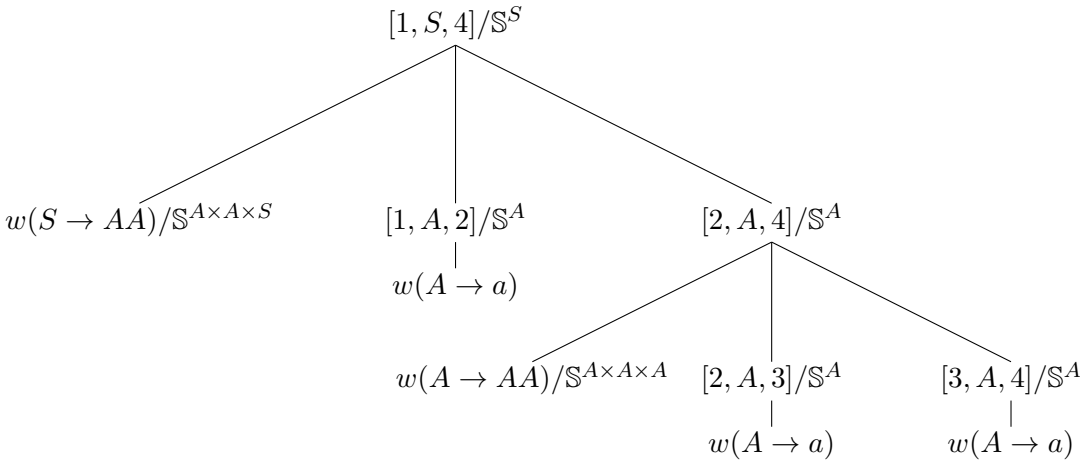


Figure 2: Item derivation corresponding to the derivation given in Figure 1 using the item-based description of CKY in Figure 3.

$$\frac{w(A \to w_i)}{[i, A, j]}$$

$$\frac{w(A \to BC) \quad [i, B, k] \quad [k, C, j]}{[i, A, j]}$$

Figure 3: Item-based description for CKY

the children nodes be independent of each other given the parent node.

Definition 5. *If for any $\sigma \in \mathcal{L}(G)$ and any $T, T' \in inner_\sigma(x)$, $V^w_{I(G)}(T)$ and $V^w_{I(G)}(T')$ are defined and $dim(V^w_{I(G)}(T)) = dim(V^w_{I(G)}(T'))$, then the weights w are well defined.*

Given an item-based derivation I, a grammar G, a well defined weight function w and a target sentence σ, the value of an item x is defined to be the sum of all its possible derivations. Formally:

$$V^w_{I(G)}(x, \sigma) = \bigoplus_{D \in inner_\sigma(x)} V^w_{I(G)}(D)$$

Definition 6. *For a given grammar G and item-based description I, the value of a sentence σ is equal to the value of the goal item which spans σ:*

$$V^w_{I(G)}(\sigma) = V^w_{I(G)}(goal, \sigma)$$

Definition 7. *An item-based description is **correct** if for all grammars G, complete semirings $\mathbb{S}$, well defined weight functions w and sentences σ, $V^w_{I(G)}(\sigma) = V^w_G(\sigma)$*

Now we are ready to state the equivalent theorem to Theorem 4.1. Let us introduce a special symbol $\perp$ and extend V^w_G and $V^w_{I(G)}$ to any weight function w so that if w is not-well defined for G, then $V^w_G(\sigma) = \perp$ and likewise for $V^w_{I(G)}$.

Theorem 5.4. *An item-based description I is correct if*

- *For every grammar G, the mapping $g : \mathcal{D}_{I(G)} \to \mathcal{D}_G$ that maps $d' \in \mathcal{D}_{I(G)}$ to the corresponding $d \in \mathcal{D}_G$ is a bijection with an inverse function f.*
- *For any complete semiring $\mathbb{S}$ and weight function w, g and f preserve the values assigned to a derivation:*

$$V^w_G(d) = V^w_{I(G)}(f(d)) \text{ and}$$
$$V^w_{I(G)}(d') = V^w_G(g(d'))$$

Proof proceeds similarly to that in (Goodman, 1999) and can be found in Appendix A.

6 Inside and Outside Calculations

In the following, we will omit the sentence σ from $inner_\sigma(x)$ and refer to this as $inner(x)$. Let $inner(\frac{a_1,..,a_k}{x})$ the set of derivation trees where the root note is x, and the direct children of x are $a_1, \ldots, a_k$.

For efficient computation of this value, we will assume that there is a partial order b on the items so that if the item y depends on x, then $b(x) \leq b(y)$.

Theorem 6.1.

$$V(x) = \bigoplus_{\substack{[a_1,\ldots,a_k] \\ s.t. \frac{a_1,..,a_k}{x}}} V(a_1) \otimes [V(a_2), \ldots, V(a_k)]$$

The proof uses the distributive property and follows that of Goodman (1999). It can be found in Appendix A.

For the notion of a value of a derivation to extend to outside trees, we will have to do some

modifications. This is because an outside tree will have one subtree $\langle b : A_1, \ldots, A_n \rangle$, such that $V(A_1) \otimes [V(A_2), \ldots, V(A_n)]$ will potentially not be defined since one of the subtrees A_k will be missing. Note that the missing A_k will be headed by an item. We will say the a tree $T \in outer(x)$ if T can be obtained by taking a tree T' headed by the goal item and removing any of its subtrees headed by the item x. Outer value $Z(T_k)$ is defined recursively as follows:

If T_k is headed by the goal item then $Z(T_k) = I_{d_S}$. Else, it has a direct parent tree T such that $T = \langle b : T_1, \ldots, T_k, \ldots, T_n \rangle$. In this case, $Z(T_k) =$

$$\Big(V(T_1) \otimes_k [I_{T_k \times d_S}, V(T_{k+1}), \ldots, V(T_n)]\Big)^{\pi} \otimes [V(T_2), \ldots, V(T_{k-1})] \otimes^* Z(T)$$

where $I_{T_k \times d_S}$ is the identity tensor for the space $\mathbb{S}^{d_1 \times \ldots \times d_i \times d_S}$, $T_k \in \mathbb{S}^{d_1 \times \ldots \times d_i}$, and d_s is the dimension assigned for the terminal symbol S. The permutation π is defined as follows:

$$[1, 2, \ldots, i, j+1, j+2, \ldots, n, i+1, i+2, \ldots, j]$$

where $i = k + \text{rank}(T_k) - 1$ and $j = k + 2 \times \text{rank}(T_k) + 1$

To understand the function of π it is useful to consider the dimensions of the term before and after it is applied. Let the term $V(T_0) \otimes_k [I_{T_k \times d_S}, V(T_{k+1}), \ldots, V(T_n)]$ have dimensions:

$$e_1 \times \ldots \times e_{k-1}, d_1 \times \ldots \times d_i \times d_S \times e_k \times d_1 \times \ldots \times d_i \times d_S \times d'_n \times \ldots \times d'_m$$

Here $e_1, \ldots, e_{k-1}$ are the dimensions that will be contracted with $V(T_1), \ldots, V(T_{k-1})$ with the second multiplication operation, and $d'_n, \ldots, d'_m$ are the dimensions that were either introduced by the contraction with $V(T_{k+1}), \ldots, V(T_n)$ or were trailing dimensions from $V(T_1)$. The result of the contraction with $I_{T_k \times d_S}$ are the dimensions in the middle: $d_1, \ldots, d_i, d_S, e_k, d_1, \ldots, d_i, d_S$. Unlike the original definition of I there is one dimension e_k missing from the beginning of the sequence since it got used up during the contraction operation. What the permutation does is to move one section of the dimensions introduced by I to the very end. The dimensions become:

$$e_1 \times \ldots \times e_{k-1}, d_1 \times \ldots \times d_i \times d'_n \times \ldots \times d'_m \times d_S \times e_k \times d_1 \times \ldots \times d_i \times d_S$$

Note that this has no effect on the next contraction with $V(T_1), \ldots, V(T_{k-1})$ since the first $k-1$ ranks are left in place. However, changing the order of the ranks allow the last contraction with $Z(T)$ to be well defined.

Lemma 6.2. *Let V and Z be defined on a commutative semiring $\mathbb{S}$ and let $O \in outer_\sigma(x)$ and $T \in inner_\sigma(x)$. If combining O and T in the obvious way results in the complete derivation D,*

$$V(D) = V(T) \otimes^* Z(O)$$

Proof. (Sketch) We proceed by induction on the parse tree. Base case is where $x = goal$, $T = D$ and O is empty. Then $V(T) = V(D)$ and $Z(O) = I_S$. $V(D) \otimes^* I_S = V(D)$ by the definition of I_S which proves the statement.

Otherwise T has a parent tree $T_p = \langle y : T_1, \ldots, T_n \rangle$ where $T = T_k$. Furthermore, $T_p \in \text{inner}_\sigma(y)$, $O_p \in \text{outer}_\sigma(y)$ and by the induction hypothesis $V(D) = V(T_p) \otimes^* Z(O_p)$.

Since $T_p \in \text{inner}_\sigma(y)$ we know that

$$V(T_p) = V(T_1) \otimes [V(T_2), \ldots, V(T_m)]$$

$$V(D) = (V(T_1) \otimes [V(T_2), \ldots, V(T_m)]) \otimes^* Z(O_p)$$

The proof progresses by calculating the value for $[V(D)]_i$ based on the above term and shows that this is equal to the value of $[V(T) \otimes^* Z(O)]_i$. Full proof can be found in Appendix A. □

In the general case, Goodman (1999) defines the reverse value of x as the sum of all its outer trees.

$$Z(x) = \bigoplus_{T \in outer(x)} Z(T)$$

We will see that for a well defined weight function w, any $D \in outer_\sigma(x)$ will be assigned a value with dimensions $d_1 \times \ldots \times d_n \times d_S$ where d_S is the dimension assigned to the start symbol S, and $d_1, \ldots, d_n$ are the dimensions for $inner_\sigma(x)$.

Lemma 6.3. *Let $C(D, x)$ represent the number of times x occurs in a derivation D. Then,*

$$V(x) \otimes^* Z(x) = \bigoplus_{D \in \mathcal{D}(\sigma)} V(D) C(D, x)$$

Proof.

$$V(x) \otimes^* Z(x) = \bigoplus_{T \in inner(x)} V(T) \otimes^* \bigoplus_{O \in outer(x)} Z(O)$$

$$= \bigoplus_{T \in inner(x)} \bigoplus_{O \in outer(x)} V(T) \otimes^* Z(O)$$

By Lemma 6.2, $Z(O) \otimes^* V(T) = V(D)$. For an item x, any $O \in outer(x)$ and $T \in inner(x)$ can be combined to form a successful derivation tree containing x, and thus the number $C(D, x)$ corresponds exactly to the number of derivation trees containing x. Hence,

$$V(x) \otimes^* Z(X) = \bigoplus_{\substack{T \in inner(x) \\ O \in outer(x)}} V(T) \otimes^* Z(O)$$
$$= \bigoplus_{D \in \mathcal{D}(\sigma)} V(D) C(D, x) \quad \square$$

Now we are ready to state how to calculate the outside value of an item. Following Goodman (1999) we will extend the notation for the set of outer trees and introduce $outer\left(k, \frac{a_1 \ldots a_n}{b}\right) \subseteq outer(a_k)$ to mean the subset of the outer trees in $outer(a_k)$ where a_k has parent b and the siblings a_i. In other words, this is the set of all outer trees where the rule from which a_k is removed is $\frac{a_1 \ldots a_n}{b}$.

Theorem 6.4. *If x is the goal item, then $Z(x) = I_s$. Else, $Z(x) =$*

$$\bigoplus_{\substack{j, a_1, .., a_k, b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x = a_j}} (V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), ..., V(a_n)])^{\pi}$$
$$\otimes [V(a_2), \ldots, V(a_{k-1})] \otimes^* Z(b)$$

Proof. (sketch) $Z(x) = \bigoplus_{D \in outer(x)} Z(D)$. Either x is a goal item, in which case $Z(x) = I_S$.

Otherwise the outer trees $outer(x)$ could be written as the union of outer trees $outer\left(k, \frac{a_1 \ldots a_n}{b}\right)$ for each rule $\frac{a_1 \ldots a_n}{b}$ where $a_k = x$ for some k. Hence:

$$Z(x) = \bigoplus_{\substack{j, a_1, .., a_k, b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x = a_j}} \bigoplus_{D \in outer\left(k, \frac{a_1 \ldots a_n}{b}\right)} Z(D)$$

Using the distributive property of the partial semiring, the inside part of the equation becomes:

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \ldots a_n}{b}\right)} Z(D) =$$
$$(V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \ldots, V(a_n)])^{\pi}$$
$$\otimes [V(a_2), \ldots, V(a_{k-1})] \otimes^* Z(b)$$

Replacing the inner part of the previous equation with this term gives the desired equality. $\square$

7 Conclusion

We have presented a general extension of the semiring parsing framework where the weights for the grammar rules are tensors of semiring values, with the motivation of extending semiring parsing framework to latent variable models. We hope that this work will enable streamlined development of EM-based or spectral learning algorithms for latent refinements of a number of grammar formalisms.

Acknowledgments

The authors thank the anonymous reviewers for feedback and comments on a draft of this paper, and acknowledge the support of NSF grant IIS-1813823.

References

Raphaël Bailly, François Denis, and Liva Ralaivola. 2009. Grammatical inference as a principal component analysis problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 33–40.

Pierre Boullier. 2004. Range concatenation grammars. In *New Developments in Parsing Technology*, pages 269–289. Springer.

Shay B Cohen, Robert J Simmons, and Noah A Smith. 2008. Dynamic programming algorithms as products of weighted logic programs. In *International Conference on Logic Programming*, pages 114–129.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2013. Experiments with Spectral Learning of Latent-Variable PCFGs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157, Atlanta, Georgia. Association for Computational Linguistics.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2014. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *The Journal of Machine Learning Research*, 15(1):2399–2449.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Jason Eisner. 2002. Parameter Estimation for Probabilistic Finite-State Transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jason Eisner, Eric Goldlust, and Noah A Smith. 2005. Compiling Comp Ling: Weighted Dynamic Programming and the Dyna Language. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 281–290, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Kilian Gebhardt. 2018. Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3049–3063, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kevin Gimpel and Noah A Smith. 2009. Cube Summing, Approximate Inference with Non-Local Features, and Dynamic Programming without Semirings. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 318–326, Athens, Greece. Association for Computational Linguistics.

Joshua Goodman. 1999. Semiring Parsing. *Computational Linguistics*, 25(4):573–606.

Joshua T Goodman. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University Cambridge, Massachusetts.

Daniel Hsu, Sham M Kakade, and Tong Zhang. 2012. A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Werner Kuich. 1997. Semirings and Formal Power Series: Their Relevance to Formal Languages and Automata. In Rozenberg Grzegorz and Arto Salomaa, editors, *Handbook of Formal Languages: Volume 1 Word, Language, Grammar*, pages 609–677. Springer, Berlin, Heidelberg.

Zhifei Li and Jason Eisner. 2009. First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Singapore. Association for Computational Linguistics.

Adam Lopez. 2009. Translation as Weighted Deduction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 532–540, Athens, Greece. Association for Computational Linguistics.

Takuya Matsuzaki, 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*, pages 75–82.

Mark-Jan Nederhof. 2003. Weighted Deductive Parsing and Knuth's Algorithm. *Computational Linguistics*, 29(1):135–143.

Fernando C N Pereira and David H D Warren. 1983. Parsing as Deduction. In *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, pages 137–144, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Stuart M Shieber, Yves Schabes, and Fernando C N Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming*, 24(1-2):3–36.

Klaas Sikkel. 1998. Parsing schemata and correctness of parsing algorithms. *Theoretical Computer Science*, 199(1-2):87–103.

Martha Edmay Steenstrup. 1985. *Sum-Ordered Partial Semirings*. Ph.D. thesis, University of Massachusetts Amherst.

Appendix A - Proofs of Theorems in Main Paper

Lemma 5.1. *For any k, l, $\otimes_{[k;l]}$ distributes over $\oplus$*

Proof. We will proceed by showing that:

$$A \otimes_{[k;l]} (B \oplus C) = (A \otimes_{[k;l]} B) \oplus (A \otimes_{[k;l]} C)$$

Firstly, note that for the left hand side of the equation to be defined, B and C needs to be of matching ranks, and that $B \oplus C$ will be the same rank as both B and C. Therefore, if the left hand side is well defined then both $A \otimes_{[k;l]} B$ and $A \otimes_{[k;l]} C$ is defined and has matching ranks. So the right hand side is defined if and only if the left hand side is defined as well.

$$\begin{aligned}
&[A \otimes_{[j;k]} (B \oplus C)]_{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1},j_{l+1},\ldots,j_m,i_{k+1},\ldots,i_n} \\
&\quad = \sum_{i_k,j_l} \delta(i_k, j_l) A_{i_1,\ldots,i_n} \times (B \oplus C)_{j_1,\ldots,j_m} \\
&\quad = \sum_{i_k,j_l} \delta(i_k, j_l) A_{i_1,\ldots,i_n} \times (B_{j_1,\ldots,j_m} + C_{j_1,\ldots,j_m}) \\
&\quad = \sum_{i_k,j_l} \delta(i_k, j_l)(A_{i_1,\ldots,i_n} \times B_{j_1,\ldots,j_m}) + \delta(i_k, j_l)(A_{i_1,\ldots,i_n} \times C_{j_1,\ldots,j_m}) \\
&\quad = [(A \otimes_{[k;l]} B) \oplus (A \otimes_{[k;l]} C)]_{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1},j_{l+1},\ldots,j_m,i_{k+1},\ldots,i_n}
\end{aligned}$$

□

Theorem 5.4. *An item-based description I is correct if*

- *For every grammar G, the mapping $g : \mathcal{D}_{I(G)} \to \mathcal{D}_G$ that maps $d' \in \mathcal{D}_{I(G)}$ to the corresponding $d \in \mathcal{D}_G$ is a bijection with an inverse function f.*
- *For any complete semiring S and weight function w, g and f preserve the values assigned to a derivation:*

$$V_G^w(d) = V_{I(G)}^w(f(d)) \text{ and } V_{I(G)}^w(d') = V_G^w(g(d'))$$

Proof.

$$V_{I(G)}^w(\alpha) = V_{I(G)}^w(goal, \alpha) = \bigoplus_{D \in \text{inner}_\alpha(goal)} V_{I(G)}^w(D) = \bigoplus_{D \in \mathcal{D}_{I(G)}(\alpha)} V_G^w(g(D))$$

Observe that $D \in \mathcal{D}_{I(G)}(\alpha)$ iff $g(D) \in \mathcal{D}_G(\alpha)$ since the rules that appear in the leaves of D, applied from left to right, determines the grammar derivation tree $g(D)$ uniquely via g, and vice versa. Hence,

$$V_{I(G)}^w(\alpha) = \bigoplus_{g(D) \in \mathcal{D}_G(\alpha)} V_G^w(g(D)) = V_G^w(\alpha)$$

□

Theorem 6.1.

$$V(x) = \bigoplus_{\substack{[a_1,\ldots,a_k] \\ s.t. \frac{a_1,\ldots,a_k}{x}}} V(a_1) \otimes [V(a_2), \ldots, V(a_k)]$$

Proof. Recall that by definition, $V(x) = \bigoplus_{D \in inner(x)} V(D)$. For any item derivation D, D is either an axiom or there is some $a_1, \ldots, a_k, b$ s.t. $D \in inner(\frac{a_1 \ldots a_k}{b})$. If D is an axiom, then $inner(D)$ is just a single rule a, and so $V(D) = V(a)$. Else, for each rule $\frac{a_1 \ldots a_k}{x}$

$$\bigoplus_{D \in inner(\frac{a_1 \ldots a_k}{x})} V(D) = \bigoplus_{\substack{D_{a_1} \in inner(a_1), \ldots, \\ D_{a_k} \in inner(a_k)}} V(D_{a_1}) \otimes [V(D_{a_2}), \ldots, V(D_{a_k})]$$

$$= \left(\bigoplus_{D_{a_1} \in inner(a_1)} V(D_{a_1}) \right) \otimes \left(\bigoplus_{\substack{D_{a_2} \in inner(a_2), \ldots, \\ D_{a_k} \in inner(a_k)}} \bigotimes_{i=2}^{k} V(D_{a_i}) \right)$$

$$= \left(\bigoplus_{D_{a_1} \in inner(a_1)} V(D_{a_1}) \right) \otimes \left(\bigoplus_{D_{a_2} \in inner(a_2)} V(D_{a_2}), \ldots, \bigoplus_{D_{a_k} \in inner(a_k)} V(D_{a_k}) \right)$$

$$= V(a_1) \otimes [V(a_2), \ldots, V(a_k)]$$

Where the last step holds due to the distributive property of the partial semiring.

Since the set $inner(x) = \bigcup_i D_i$ where $D_i \in inner(\frac{a_1 \ldots a_k}{x})$ for all inference rules $\frac{a_1 \ldots a_k}{x}$, we can write the summation over $D \in inner(x)$ as:

$$V(x) = \bigoplus_{D \in inner(x)} V(D)$$

$$= \bigoplus_{\substack{[a_1, \ldots, a_k] \\ \text{s.t.} \frac{a_1, \ldots, a_k}{x}}} \bigoplus_{D \in inner(\frac{a_1 \ldots a_k}{x})} V(D)$$

$$= \bigoplus_{\substack{[a_1, \ldots, a_k] \\ \text{s.t.} \frac{a_1, \ldots, a_k}{x}}} V(a_1) \otimes [V(a_2), V(a_3), \ldots, V(a_k)]$$

Where the last line is obtained by replacing the inner part of the expression with the equality obtained from the previous part of the proof. □

Lemma 6.2. *Let V and Z be defined on a commutative semiring $\mathbb{S}$ and let $O \in outer_\alpha(x)$ and $T \in inner_\alpha(x)$. If combining O and T in the obvious way results in the complete derivation D then*

$$V(D) = V(T) \otimes^* Z(O)$$

Proof. To simplify notation of the indices, let $\mathbf{i}$ stand for a list of indices $i_1, \ldots, i_n$ for some n. We will also use $\mathbf{d}^i$ to denote a list $d^i_1, \ldots d^i_{n_i}$ and $\mathbf{d}$ to denote $\mathbf{d}^1, \ldots, \mathbf{d}^n$. $\delta(\mathbf{i}, \mathbf{j}) = \prod_{k=1}^{n} \delta(i_k, j_k)$.

We will proceed by induction on the parse tree. Base case is where $x = goal$, $T = D$ and O is empty. Then $V(T) = V(D)$ and $Z(O) = I_S$. $V(D) \otimes^* I_S = V(D)$ by the definition of I_S which proves the statement.

Otherwise T has a parent tree $T_p = \langle y : T_1, \ldots, T_n \rangle$ where $T = T_k$. Furthermore, $T_p \in \text{inner}_\alpha(y)$, $O_p \in \text{outer}_\alpha(y)$ and by induction hypothesis $V(D) = V(T_p) \otimes^* Z(O_p)$.

Since $T_p \in \text{inner}_\alpha(y)$ we know that

$$V(T_p) = V(T_1) \otimes [V(T_2), \ldots, V(T_m)]$$

So

$$V(D) = (V(T_1) \otimes [V(T_2), \ldots, V(T_m)]) \otimes^* Z(O_p)$$

The proof progresses by calculating the value for $[V(D)]_i$ based on the above term and shows that this is equal to the value of $[V(T) \otimes^* Z(O)]_i$.

Let:

$$V(T_1) \in \mathbb{S}^{\mathbf{e},\mathbf{f}} \qquad V(T_i) \in \mathbb{S}^{e_i,\mathbf{d}^i}$$
$$Z(O_p) \in \mathbb{S}^{\mathbf{d},\mathbf{f},s} \qquad V(D) \in \mathbb{S}^{s}$$

Then:

$$\begin{aligned} V[(T_p)]_{\mathbf{d},\mathbf{f}} &= [V(T_1) \otimes (V(T_2), \ldots, V(T_m))]_{\mathbf{d},\mathbf{f}} \\ &= \sum_{\mathbf{e},\mathbf{e}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{i=2}^{m} \delta(e_i, e_i') V(T_i)_{e_i',\mathbf{d}^i} \end{aligned}$$

$$\begin{aligned} [V(D)]_s = [V(T_p) \otimes^* Z(O_p)]_s = \\ \sum_{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}'\mathbf{f},\mathbf{f}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \left(\prod_{i=2}^{m} \delta(e_i, e_i') V(T_i)_{e_i,\mathbf{d}^i} \right) \\ \times \delta(\mathbf{d}, \mathbf{d}') \delta(\mathbf{f}, \mathbf{f}') Z(O_p)_{\mathbf{d},\mathbf{f},s} \end{aligned}$$

Now we will proceed to prove that this term is equal to $V(T_k) \otimes^* Z(O)$. Let $I_{T_k} \in \mathbb{S}^{e_k',\mathbf{d}^k,s,e_k,\mathbf{d}^k,s}$. We will calculate the value of the outside term in sections. Let $A = V(T_1) \otimes_k (I_{T_k}, V(T_{k+1}), \ldots, V(T_n))$. Then,

$$\begin{aligned} &A_{e_1,\ldots,e_{k-1},\mathbf{d}^k,s,\hat{e}_k,\hat{\mathbf{d}}_k,\hat{s},\mathbf{d}^{k+1},\ldots,\mathbf{d}^n,\mathbf{f}} = \\ &A^{\pi}_{e_1,\ldots,e_{k-1},\mathbf{d}^k,\mathbf{d}^{k+1},\ldots,\mathbf{d}^n,\mathbf{f},s,\hat{e}_k,\hat{\mathbf{d}}_k,\hat{s}} = \\ &\sum_{\substack{e_k,\ldots,e_n \\ e_k',\ldots,e_n'}} V(T_1)_{\mathbf{e},\mathbf{f}} \times \delta(e_k, e_k') \delta(\mathbf{d}^k, \hat{\mathbf{d}}^k) \delta(s, \hat{s}) \times \prod_{i=k+1}^{m} \delta(e_i, e_i') V(T_i)_{e_i',\mathbf{d}^i} \\ &[A^{\pi} \otimes (V(T_2), \ldots, V(T_{k-1}))]_{\mathbf{d},\mathbf{f},s,\hat{e}_k,\hat{\mathbf{d}}^k,\hat{s}} = \\ &\sum_{\mathbf{e},\mathbf{e}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{\substack{i=2 \\ i \neq k}}^{n} V(T_i)_{e_i',\mathbf{d}^i} \times \\ &\qquad \delta(\mathbf{e}, \mathbf{e}') \times \delta(e_k, \hat{e}_k) \times \delta(\mathbf{d}^k, \hat{\mathbf{d}}^k) \times \delta(s, \hat{s}) \\ &[Z(O)]_{\hat{e}_k,\hat{\mathbf{d}}^k,\hat{s}} = \sum_{\substack{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}' \\ \mathbf{f},\mathbf{f}',s,s'}} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{\substack{i=2 \\ i \neq k}}^{n} V(T_i)_{e_i',\mathbf{d}^i} \times Z(O_p)_{\mathbf{d}',\mathbf{f}',s'} \\ &\qquad \times \delta(\mathbf{e}, \mathbf{e}') \times \delta(e_k, \hat{e}_k) \times \delta(\mathbf{d}^k, \hat{\mathbf{d}}^k) \times \delta(s, \hat{s}) \\ &\qquad \times \delta(\mathbf{d}, \mathbf{d}') \times \delta(\mathbf{f}, \mathbf{f}') \times \delta(s, s') \\ &[V(T_k) \otimes^* Z(O)]_{\hat{s}} = \\ &\qquad \sum_{\substack{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}' \\ \mathbf{f},\mathbf{f}',s,s' \\ \hat{e}_k,\hat{\mathbf{d}}^k,e_k'',\mathbf{d}^{k''}}} V(T_k)_{e_k'',\mathbf{d}^{k''}} \times V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{\substack{i=2 \\ i \neq k}}^{n} V(T_i)_{e_i',\mathbf{d}^i} \times Z(O_p)_{\mathbf{d}',\mathbf{f}',s'} \\ &\qquad \times \delta(\mathbf{e}, \mathbf{e}') \times \delta(e_k, \hat{e}_k) \times \delta(\mathbf{d}^k, \hat{\mathbf{d}}^k) \times \delta(s, \hat{s}) \\ &\qquad \times \delta(\mathbf{d}, \mathbf{d}') \times \delta(\mathbf{f}, \mathbf{f}') \times \delta(s, s') \times \delta(e_k'', \hat{e}_k) \times \delta(\mathbf{d}^{k''}, \hat{\mathbf{d}}^k) \\ &= \sum_{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}'\mathbf{f},\mathbf{f}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{i=2}^{m} V(T_i)_{e_i,\mathbf{d}^i} \times Z(O_p)_{\mathbf{d},\mathbf{f},\hat{s}} \\ &\qquad \times \delta(\mathbf{e}, \mathbf{e}') \times \delta(\mathbf{d}, \mathbf{d}') \times \delta(\mathbf{f}, \mathbf{f}') \end{aligned}$$

Which completes the proof. The last simplification step is obtained by replacing $\hat{e}_k$ and e''_k with e_k, $\hat{\mathbf{d}}^k$ and $\mathbf{d}^{k''}$ with $\mathbf{d}^k$ and s and s' with $\hat{s}$ since these need to be equal for any term to contribute to the final sum. The commutativity of $\mathbb{S}$ then allows $V(T_k)_{e_k,\mathbf{d}^k}$ to be moved to its place in the sequence.

□

Theorem 6.4. *If x is the goal item, then $Z(x) = I_s$. Else,*

$$Z(x) = \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \dots a_k}{b} \text{ and } x=a_j}} (V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \dots, V(a_n)])^\pi \otimes (V(a_2), \dots, V(a_{k-1})) \otimes^* Z(b)$$

Proof. by definition $Z(x) = \bigoplus_{D \in outer(x)} Z(D)$. Either x is a goal item, in which case $Z(x) = Z() = I_S$.

Otherwise the outer trees $outer(x)$ could be written as the union of outer trees $outer\left(k, \frac{a_1 \dots a_n}{b}\right)$ for each rule $\frac{a_1 \dots a_n}{b}$ where $a_k = x$ for some k. Hence:

$$Z(x) = \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \dots a_k}{b} \text{ and } x=a_j}} \bigoplus_{D \in outer\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D)$$

For the inner part of this equation we have:

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D) = \bigoplus_{D_b \in outer(b)} \bigoplus_{\substack{D_{a_1} \in inner(a_1), \dots, \\ D_{a_{k-1}} \in inner(a_{k-1})}} \bigoplus_{\substack{D_{a_{k+1}} \in inner(a_{k+1}), \dots, \\ D_{a_n} \in inner(a_n)}} \left(V(D_{a_1}) \otimes_k \left[I_{D_{a_k} \times d_S}, V(D_{a_{k+1}}), \dots, V(D_{a_n})\right]\right)^\pi \otimes \left(V(D_{a_2}), \dots, V(D_{a_{k-1}})\right) \otimes^* Z(D_b)$$

Since $\oplus$ distributes over $\otimes$, this can rewritten as

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D) = \left(\bigoplus_{\substack{D_{a_1} \in \\ inner(a_1)}} V(D_{a_1}) \otimes_k \left[I_{D_{a_k}}, \bigoplus_{\substack{D_{a_{k+1}} \in \\ inner(a_{k+1})}} V\left(D_{a_{k+1}}\right), \dots, \bigoplus_{\substack{D_{a_n} \in \\ inner(a_n)}} V\left(D_{a_n}\right) \right] \right)^\pi \otimes \left(\bigoplus_{D_{a_2} \in inner(a_2)} V(D_{a_2}), \dots, \bigoplus_{D_{a_{k-1}} \in inner(a_{k-1})} V(D_{a_{k-1}}) \right) \otimes^* \bigoplus_{D_b \in outer(b)} Z(D_b)$$

And since $V(a_i)$ and $Z(D_b)$ are defined as the summation of their inner and outer trees respectively

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D) = (V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \dots, V(a_n)])^\pi \otimes (V(a_2), \dots, V(a_{k-1})) \otimes^* Z(b)$$

Replacing the inner part of the previous equation with this term gives us the desired equality, completing the proof. □

Appendix B - Inside and Outside Calculations for Looping Buckets

In computing the inside and outside values with an item-based description, we assume a pre-computed ordering over items in the form of *buckets*. For items x and y, we write $bucket(x) \leq bucket(y)$ if the value of y depends on the value of x. So far we have assumed that items could be simply sorted so that no item directly or indirectly depends on itself, and given the inside and outside formulas accordingly. In this section we give the equivalent formulas for items in *looping buckets*. Items in a looping bucket depend on each other and computing their values might require an infinite sum. Our presentation and proofs both follow that of Goodman (1998).

For an item x in a looping bucket B, let the *generation* of a derivation tree x to be the maximum number of items in B that could appear in a single path from the root to a leaf. This intuitively provides an ordering for processing a potentially infinite number of trees by starting from generation 0 and incrementally adding larger and larger trees. We will denote the set of inner trees of x with generation at most g with $inner_{\leq}(x, B)$ Adding up the values of all inner trees of x that have generation at most g then gives us an approximation for the true inner value of x, and the approximation gets better as g gets larger. Formally, we define a *g generation value* for an item x in bucket B as:

$$V_{\leq g}(x, B) = \bigoplus_{D \in inner_{\leq g}(x,B)} V(D)$$

For ω-continuous semirings, the infinite sum is equal to the supremum of the partial sums (Kuich 1997, 613), hence (Goodman 1999, 589):

$$V(x) = \bigoplus_{D \in inner(x)} V(D) = \sup_g V_{\leq g}(x, B)$$

Fortunately, tensors of semirings of set dimensions are ω-continuous as long as the underlying semiring is ω-continuous. We give the necessary definitions to establish this property:

Definition 8. *(Kuich 1997, 611) A semiring is **naturally ordered** if there is a partial ordering $\sqsubseteq$ such that $x \sqsubseteq y$ iff there is a z s.t. $x \oplus z = y$.*

Definition 9. *(Kuich 1997, 612) A naturally ordered complete semiring is ω-continuous if for any sequence $x_1, x_2, \ldots$ and for any constant y, if for all n, $\bigoplus_{0 \leq i \leq n} x_i \sqsubseteq y$ then $\bigoplus_i x_i \sqsubseteq y$*

Notice that for the set of tensors in $\mathbb{S}^{\mathbf{d}}$ where $\mathbf{d}$ is an arbitrary list of positive integers, if the underlying semiring has a natural ordering then this could be extended straightforwardly to $\mathbb{S}^{\mathbf{d}}$ by the following rule: $\mathbf{X} \sqsubseteq \mathbf{Y}$ iff $\mathbf{X_i} \sqsubseteq \mathbf{Y_i}$ for all indices $\mathbf{i}$. It is straightforward to check that if the underlying semiring is ω-continuous, then $\mathbb{S}^{\mathbf{d}}$ is ω-continuous as well.

Goodman (1999) gives a formula for $V_{\leq g}(x, B)$ in order to compute or approximate the supremum. Below we give the analogous formula for partial semirings:

Theorem B.1. *For items x in a looping bucket B and the generation $g \geq 1$*

$$V_{\leq g}(x, B) = \bigoplus_{\substack{[a_1,\ldots,a_k] \\ s.t. \frac{a_1,\ldots,a_k}{x}}} K_g(a_1, B) \otimes [K_g(a_2, B), \ldots, K_g(a_k, B)]$$

Where

$$K_g(a, B) = \begin{cases} V(a) & \text{if } a \notin B \\ V_{\leq g-1}(a, B) & \text{if } a \in B \end{cases}$$

Proof.

$$
\begin{aligned}
V_{\leq g}(x,B) &= \bigoplus_{D \in inner_{\leq g}(x,B)} V(D) \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} \bigoplus_{\substack{D_{a_1} \in inner_{\leq g-1}(a_1,B),\ldots, \\ D_{a_k} \in inner_{\leq g-1}(a_k,B)}} V\left(\langle x : D_{a_1}, \ldots D_{a_k} \rangle\right) \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} \bigoplus_{\substack{D_{a_1} \in inner_{\leq g-1}(a_1,B),\ldots, \\ D_{a_k} \in inner_{\leq g-1}(a_k,B)}} V(D_{a_1}) \otimes [V(D_{a_2}), \ldots, V(D_{a_k})] \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} \bigoplus_{D_{a_1} \in inner_{\leq g-1}(a_1,B)} V(D_{a_1}) \\
&\qquad \otimes \left[\bigoplus_{D_{a_2} \in inner_{\leq g-1}(a_2,B)} V(D_{a_2}), \ldots, \bigoplus_{D_{a_k} \in inner_{\leq g-1}(a_k,B)} V(D_{a_k}) \right] \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} V_{\leq g-1}(a_1,B) \otimes [V_{\leq g-1}(a_2,B), \ldots, V_{\leq g-1}(a_k,B)]
\end{aligned}
$$

Note that if a_i is not in the bucket B then $V_{\leq g-1}(a_i, B) = V(a_i)$, hence $V_{\leq g-1}(a_i, B)$ can be replaced with $K_g(a_i, B)$, completing the proof. □

We will follow a similar strategy for computing the outside values of items that belong to a looping bucket. The only difference is the slight difference in the definition of the generation of of the tree. If $D \in outer(x)$ where x belongs to a looping bucket B, then the generation of D is maximum number of items that could appear in a single path from the root to x, where x is included in the count. Let

$$
Z_{\leq g}(x,B) = \bigoplus_{D \in outer_{\leq g}(x,B)} Z(D)
$$

Theorem B.2. *For items x in a looping bucket B and the generation $g \geq 1$*

$$
\begin{aligned}
Z_{\leq g}(x,B) = \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} &(V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \ldots, V(a_n)])^{\pi} \\
&\otimes [(V(a_1), \ldots, V(a_{k-1})] \otimes^* H_g(b,B)
\end{aligned}
$$

Where π is defined as in Theorem 6.4 and

$$
H_g(b,B) = \begin{cases} Z(b) & \text{if } b \notin B \\ Z_{\leq g-1}(b,B) & \text{if } b \in B \end{cases}
$$

Proof.

$$
\begin{aligned}
Z_{\leq g}(x,B) &= \bigoplus_{D \in outer_{\leq g}(x,B)} Z(D) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \dots a_k}{b} \text{ and } x=a_j}} \; \bigoplus_{D \in outer_{\leq g-1}\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \dots a_k}{b} \text{ and } x=a_j}} \; \bigoplus_{D_b \in outer_{\leq g-1}(b)} \; \bigoplus_{\substack{D_{a_1} \in inner(a_1),\dots, \\ D_{a_{k-1}} \in inner(a_{k-1})}} \; \bigoplus_{\substack{D_{a_{k+1}} \in inner(a_{k+1}),\dots, \\ D_{a_n} \in inner(a_n)}} \\
&\qquad \left(V(D_{a_1}) \otimes_k \left[I_{D_{a_k} \times d_S}, V(D_{a_{k+1}}), \dots, V(D_{a_n})\right]\right)^{\pi} \\
&\qquad \otimes \left(V(D_{a_2}), \dots, V(D_{a_{k-1}})\right) \otimes^* Z_{\leq g}(D_b, B) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \dots a_k}{b} \text{ and } x=a_j}} \left(\bigoplus_{\substack{D_{a_1} \in \\ inner(a_1)}} V(D_{a_1}) \otimes_k \left[I_{D_{a_k}}, \bigoplus_{\substack{D_{a_{k+1}} \in \\ inner(a_{k+1})}} V(D_{a_{k+1}}), \dots, \bigoplus_{\substack{D_{a_n} \in \\ inner(a_n)}} V(D_{a_n}) \right] \right)^{\pi} \\
&\qquad \otimes \left(\bigoplus_{D_{a_2} \in inner(a_2)} V(D_{a_2}), \dots, \bigoplus_{D_{a_{k-1}} \in inner(a_{k-1})} V(D_{a_{k-1}}) \right) \\
&\qquad \otimes^* \bigoplus_{D_b \in outer_{\leq g-1}(b)} Z_{\leq g-1}(D_b, B) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \dots a_k}{b} \text{ and } x=a_j}} (V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \dots, V(a_n)])^{\pi} \\
&\qquad \otimes [(V(a_2), \dots, V(a_{k-1})] \otimes^* Z_{\leq g-1}(b, B)
\end{aligned}
$$

Like the inner case, note that for an item b not in the looping bucket b, $Z_{\leq g-1}(b,B) = Z(b)$, hence we can replace $Z_{\leq g-1}(b,B)$ with $H_g(b,B)$, completing the proof. □