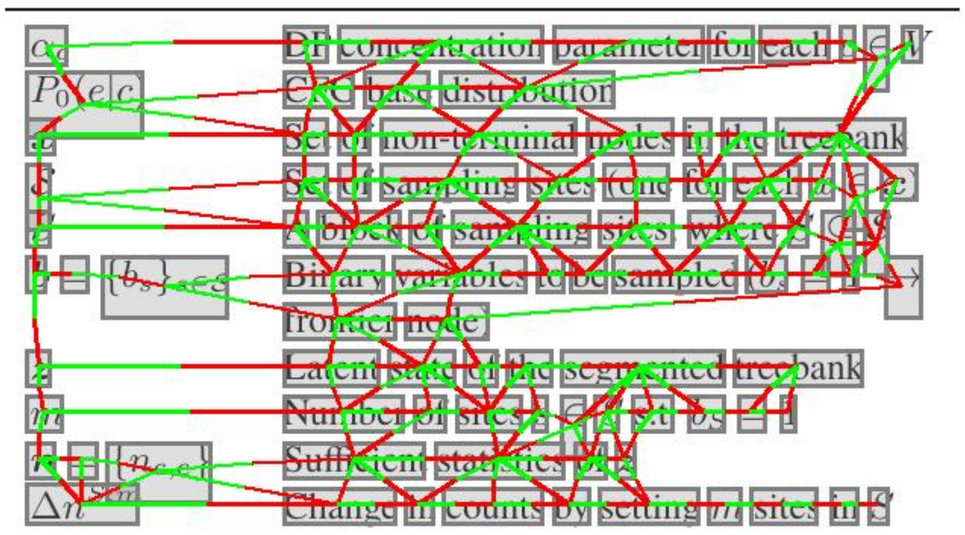

Table 5: DP-TSG model notation. For consistency, we largely follow the notation of Liang et al. (2010). Note that $\boldsymbol{z} = (\boldsymbol{b}, \boldsymbol{x})$, and as such $z = \langle c, e \rangle$.

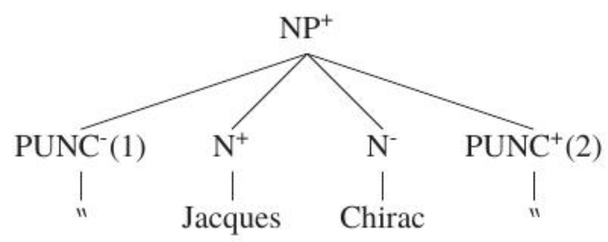

Figure 1: Example of two conflicting sites of the same type. Define the *type* of a site $t(\boldsymbol{z}, s) \stackrel{\text{def}}{=} (\Delta n^{s:0}, \Delta n^{s:1})$. Sites (1) and (2) above have the same type since $t(\boldsymbol{z}, s_1) = t(\boldsymbol{z}, s_2)$. However, the two sites *conflict* since the probabilities of setting b_{s_1} and b_{s_2} both depend on counts for the tree fragment rooted at NP. Consequently, sites (1) and (2) are not exchangeable: the probabilities of their assignments depend on the order in which they are sampled.

$t \in \Sigma$ are terminals; $e \in R$ are elementary trees;[5] $\diamondsuit \in V$ is a unique start symbol; and $\theta_{c,e} \in \boldsymbol{\theta}$ are parameters for each tree fragment. A PTSG derivation is created by successively applying the substitution operator to the leftmost *frontier node* (denoted by c^+). All other nodes are *internal* (denoted by c^-).

In the supervised setting, DP-TSG grammar extraction reduces to a segmentation problem. We have a treebank T that we segment into the set R, a process that we model with Bayes' rule:

$$p(R \mid T) \propto p(T \mid R)\, p(R) \tag{1}$$

Since the tree fragments completely specify each tree, $p(T \mid R)$ is either 0 or 1, so all work is performed by the prior over the set of elementary trees.

The DP-TSG contains a DP prior for each $c \in V$ (Table 5 defines further notation). We generate $\langle c, e \rangle$ tuples as follows:

$$\theta_c | c, \alpha_c, P_0(\cdot|c) \sim DP(\alpha_c, P_0)$$
$$e | \theta_c \sim \theta_c$$

The data likelihood is given by the latent state $\boldsymbol{z}$ and the parameters $\boldsymbol{\theta}$: $p(\boldsymbol{z}|\boldsymbol{\theta}) = \prod_{z \in \boldsymbol{z}} \theta_{c,e}^{n_{c,e}(\boldsymbol{z})}$. Integrating out the parameters, we have:

$$p(\boldsymbol{z}) = \prod_{c \in V} \frac{\prod_e (\alpha_c P_0(e|c))^{\overline{n_{c,e}(\boldsymbol{z})}}}{\alpha_c^{\overline{n_{c,\cdot}(\boldsymbol{z})}}} \tag{2}$$

where $x^{\overline{n}} = x(x+1)\ldots(x+n-1)$ is the rising factorial. (§A.1 contains ancillary details.)

Base Distribution The base distribution P_0 is the same maximum likelihood PCFG used in the Stanford parser.[6,7] After applying the manual state splits, we perform simple right binarization, collapse unary rules, and replace rare words with their signatures (Petrov et al., 2006).

For each non-terminal type c, we learn a stop probability $s_c \sim \text{Beta}(1, 1)$. Under P_0, the probability of generating a rule $A^+ \to B^-\ C^+$ composed of non-terminals is

$$P_0(A^+ \to B^-\ C^+) = p_{\text{MLE}}(A \to B\ C) s_B (1 - s_C) \tag{3}$$

For lexical insertion rules, we add a penalty proportional to the frequency of the lexical item:

$$P_0(c \to t) = p_{\text{MLE}}(c \to t) p(t) \tag{4}$$

where $p(t)$ is equal to the MLE unigram probability of t in the treebank. Lexicalizing a rule makes it very specific, so we generally want to avoid lexicalization with rare words. Empirically, we found that this penalty reduces overfitting.

Type-based Inference Algorithm To learn the parameters $\boldsymbol{\theta}$ we use the collapsed, block Gibbs sampler of Liang et al. (2010). We sample binary variables b_s associated with each non-terminal node/site in the treebank. The key idea is to select a block of exchangeable sites S of the same *type* that do not *conflict* (Figure 1). Since the sites in S are exchangeable, we can set $\boldsymbol{b}_S$ randomly so long as we know m, the number of sites with $b_s = 1$. Because this algorithm is a not a contribution of this paper, we refer the reader to Liang et al. (2010).

[5] We use the terms *tree fragment* and *elementary tree* interchangeably.

[6] The Stanford parser is a product model, so the results in §5.1 include the contribution of a dependency parser.

[7] Bansal and Klein (2010) also experimented with symbol refinement in an all-fragments (parametric) TSG for English.