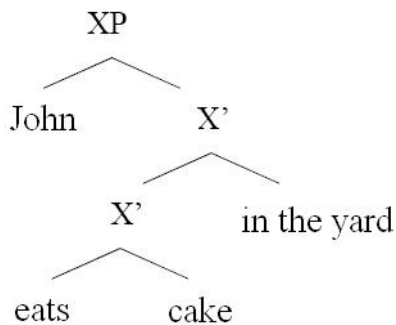
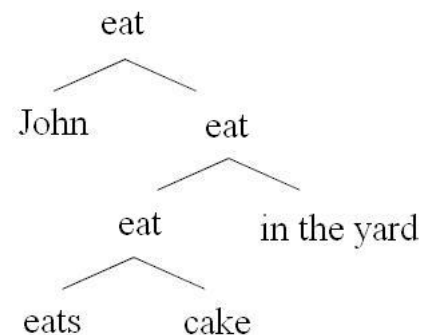


How do we represent adjuncts in BPS?

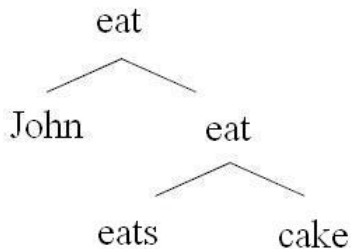
Recall from X-bar theory that the complement in the verb phrase shown below is “cake,” and that the node “in the yard” is an adjunct.



The fact that BPS eliminates labels makes it more clear that the head “eat” projects its properties onto its parent nodes. However, this complicates the issue of adjunction by making complement and adjunct nodes much less distinct. There’s no way to tell from looking at the tree which nodes are adjuncts and which are complements.

**A/A Condition**

The A-over-A condition states that operation such as movement or do-so replacement can only operate on maximal projections. This means that a node with a certain label cannot be targeted by an operation if its parent has the same label. So in the example below, movement or replacement of the V node containing “eat” (ignoring tense) is not allowed because its parent has the same label.

**“Do so” substitution**

[[_{VP}Mike ate [_{comp}cake] [_{adj}in the yard]], and [_{VP}John did so too]].

[[_{VP}Mike ate [_{comp}cake] [_{adj}in the yard]], and [_{VP}John did so [_{adj}in the park]].

*[[_{VP}Mike ate [_{comp}cake] [_{adj}in the yard]], and [_{VP}John did so [_{comp}pizza] [_{adj}in the kitchen]].

In X' theory, we can explain this phenomenon by saying that do-so substitution targets X' nodes.

BPS doesn't have X-bar nodes, so how do we explain do-so substitution? We can't. ☹️

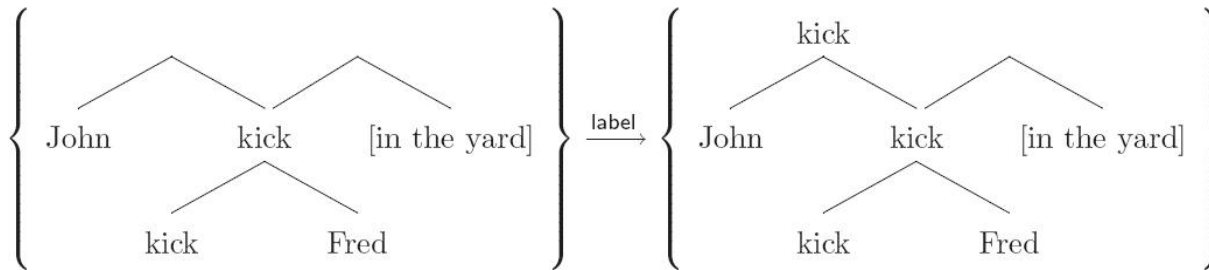
Solution (Hornstein and Nunes 2008)

BPS uses a MERGE operation to join nodes. We could separate MERGE into two distinct steps:

1. Concatenate operation: joining two nodes from the tree
2. Label operation: applying a name to the new node

We'll discuss the motivation for this later.

The critical difference is that complements require immediate labeling in order to integrate them into the tree structure, while adjuncts do not require immediate labeling. Adjuncts are allowed to “dangle”: they can be concatenated without being immediately labeled.



A labeling operation: the node containing John has been concatenated with its sister node, and it is now being labeled with “kick.” Notice the adjunct has been concatenated, but not labeled.

Multiple adjuncts

If multiple adjuncts are present in a sentence, it's possible to replace zero, one, or more of them via do-so replacement (or another operation like fronting). In the below example, the internal node [_v ate[^]the-cake] has been labeled, but its parent node has not. The adjuncts are dangling.

[[_v ate[^]the-cake][^]in-the-afternoon]]
[^]in-the-yard
[^]with-a-fork

Motivation

Adjuncts modify events directly, while complements only modify them indirectly. Event semantics, etc. Labeling is necessary to define the relationship between, for instance, a verb and its complement. Recall from event semantics that the Patient(e, x) relation defines the relationship of a verb and its complement.

[stabbed] := λxλyλe.[stabbing'(e) ^ Agent(e; y) ^ Patient(e; x)]
 [violently] := λe.violent'(e)

It's necessary for the Patient(e, x) relation to exist in order for complements to modify an event. (Who was stabbed?) Adjuncts, however, can modify an event directly, without the help of relations like Patient, and without requiring labeling (Was the event violent?).