though its component phrases "高新技术" and "产品" are peer phrases. To avoid the conflict with the Rule (2), we just add some extra *virtual nodes* in the *n*-ary sub-trees to make sure that only binary sub-trees survive in the modified parse tree. Figure 2(b) is the modification result of the syntactic tree from Figure 2(a), where two virtual nodes with the new distinguishable POS of M are added.

In general, we add virtual nodes for each set of the continuous peer phrases and let them have the same height. Thus, for a *n*-ary sub-tree, there are $\sum_{i=1}^{n-1}(n-i) = (n-1)^2/2$ virtual nodes being added where $n>2$. The phrases exactly covered by the virtual nodes are called as *virtual peer phrases*.

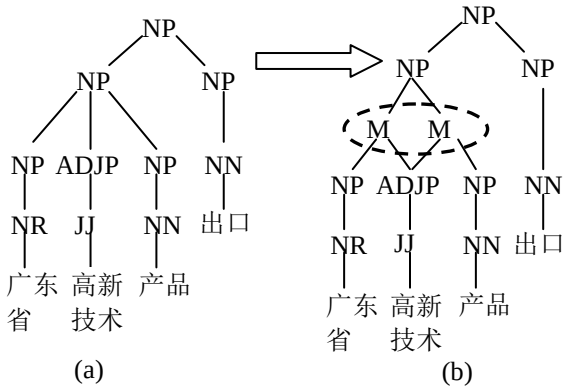

Figure 2: Example of syntax tree modification

4.2 Combination of Parse Trees

It is well known that parse errors in syntactic trees always are inescapable even if the state-of-the-art parser is used. Incorrect syntactic knowledge may harm the reordering probability estimation. To minimize the impact of parse error of a single tree, more parse trees are introduced. To support the combination of parse trees, the synchronous grammar rules are applied independently, but they will compete against each other with the effect of other models such as language model.

In our system, we combine the parse trees generated respectively by Stanford parser (Klein, 2003) and a dependency parser developed by (Zhou, 2000). Compared with the Stanford parser, the dependency parser only conducts shallow syntactic analysis. It is powerful to identify the base NPs and base VPs and their dependencies. Additionally, dependency parser runs much faster. For example, it took about three minutes for the dependency parser to parse one thousand sentences with average length of 25 words, but the Stanford parser needs about one hour to complete the same work. More importantly, as shown in the experimental results, the dependency parser can achieve the comparable quality of final translation results with Stanford parser in our system.

5 The Decoder

We developed a CKY style decoder to complete the sentence translation. A two-dimension array *CA* is constructed to store all the local candidate phrase translation and each valid cell CA_{ij} in *CA* corresponds to a foreign phrase where *i* is the phrase start position and *j* is the phrase end position. The cells in *CA* are filled in a bottom-up way. Firstly we fill in smaller cells with the translation in bilingual phrases learned from corpus. Then the candidate translation in the larger cell CA_{ij} is generated based on the content in smaller adjacent cells CA_{ik} and CA_{k+1j} with the *monotone combination* and *non-monotone combination*, where $i \le k \le j$. To reduce the cost of system resources, the well known pruning methods, such as histogram pruning, threshold pruning and recombination, are used to only keep the top *N* candidate translation in each cell.

6 Training

Similar to most state-of-the-art phrase-based SMT systems, we use the SRI toolkit (Stolcke, 2002) for language model training and Giza++ toolkit (Och and Ney, 2003) for word alignment. For reordering model training, two kinds of parse trees for each foreign sentence in the training corpus were obtained through the Stanford parser (Klein, 2003) and a dependency parser (Zhou, 2000). After that, we picked all the foreign linguistic phrases of the same sentence according to syntactic structures. Based on the word alignment results, if the aligned target words of any two adjacent foreign linguistic phrases can also be formed into two valid adjacent phrase according to constraints proposed in the phrase extraction algorithm by Och (2003a), they will be extracted as a reordering training sample. Finally, the ME modeling toolkit developed by Zhang (2004) is used to train the reordering model over the extracted samples.