1. Compreender a estrutura de um programa fonte. Procura detectar se a forma com que as unidades léxicas (tokens) se compõem está correta. Olha apenas a forma, e não o significado.
2. Definir uma forma mais eficiente para descrever linguagens (gramáticas livres de contexto) e implementar parsers ou analisadores sintáticos (autômatos com pilha). Sem esse formalismo, o trabalho deve ser feito de forma manual, o que torna mais difícil a criação e manutenção dos compiladores.
3. Porque elas são simples o suficiente para serem concebidas e manipuladas. São adequadas para descrever a maioria das construções das linguagens utilizadas na prática. Além disso, permitem fácil implementação (autômatos com pilha). Outras classes não são adequadas. Linguagens regulares são simples demais, não permitindo recursividade. Linguagens sensíveis a contexto ou recursivamente enumeráveis são de difícil implementação (máquina de turing).
4. A árvore de análise sintática representa exatamente o processo de análise. Cada nó-folha da árvore é um terminal da gramática, e cada nó-interior é um não-terminal da gramática. O mapeamento é um-para-um. Já a árvore de sintaxe abstrata contém somente as informações necessárias para a análise semântica.