Tree-Adjoining Grammars

Michal Minárik

Dílna moderní teoretické informatiky 2010

15.12.2010

Motivation

- linguistic considerations
- construct formalism related directly to the strong generative capacity (structural description), more relevant to linguistic descriptions than the weak generative capacity (set of strings)
- lexicalization of grammar formalism

- CFGs
  - string generating system
  - weak generative capacity
- Tree-Adjoining Grammars (TAGs)
  - object generating (tree) system
  - strong generative capacity

Tree-adjoining grammar (TAG)

$$G = (\Sigma, N, I, A, S)$$

- where
  - Σ – finite set of terminal symbols
  - N – finite set of non-terminal symbols: $\Sigma \cap N = \emptyset$;
  - S – distinguished non-terminal symbol: $S \in N$;
  - I – finite set of finite trees, called initial trees:
    - interior nodes – labeled by non-terminal symbols
    - frontier nodes – terminals or non-teminals , non-terminals marked for substitution (anotated with a down arrow ↓)
  - A – finite set of finite trees, called auxiliary trees:
    - interior nodes – labeled by non-terminal symbols
    - frontier nodes – terminals or non-terminals, non-terminals markes for substitution except for one node , called foot note (anotated with an asterisk $*$)

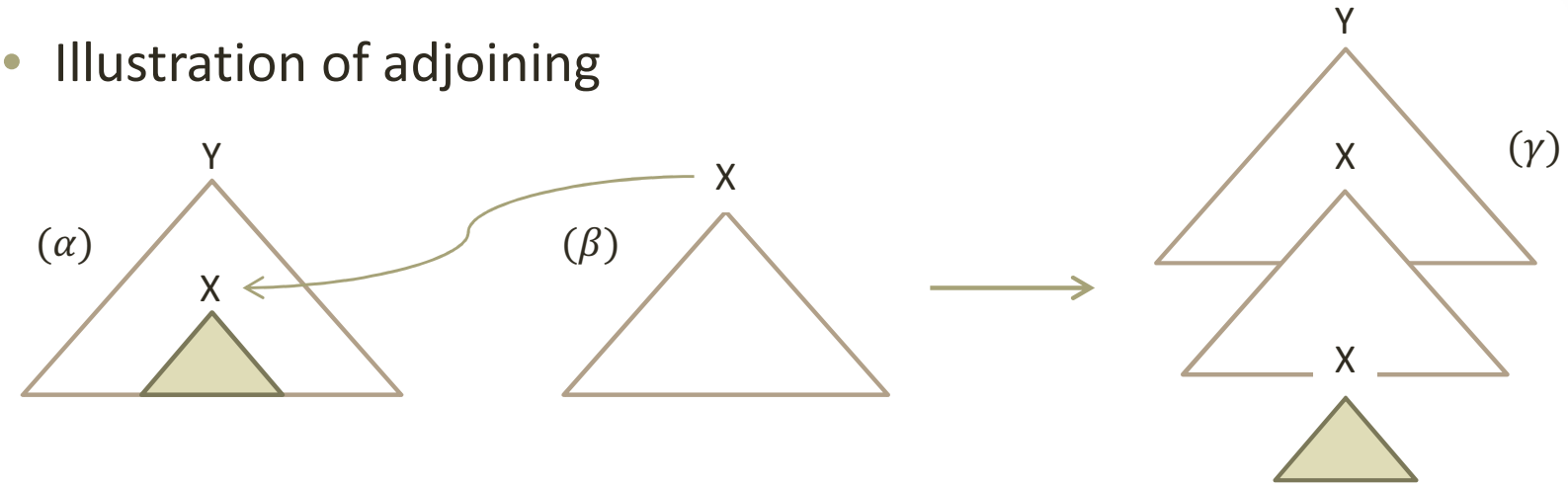
TAG – operations

- $I \cup A$ – set of elementary trees
- Elementary tree is called an X-type elementary tree if its root is labeled by the non-terminal X.
- A tree build by composition of two other tree is called a derived tree.
- Two composition operations – adjoining and substitution.

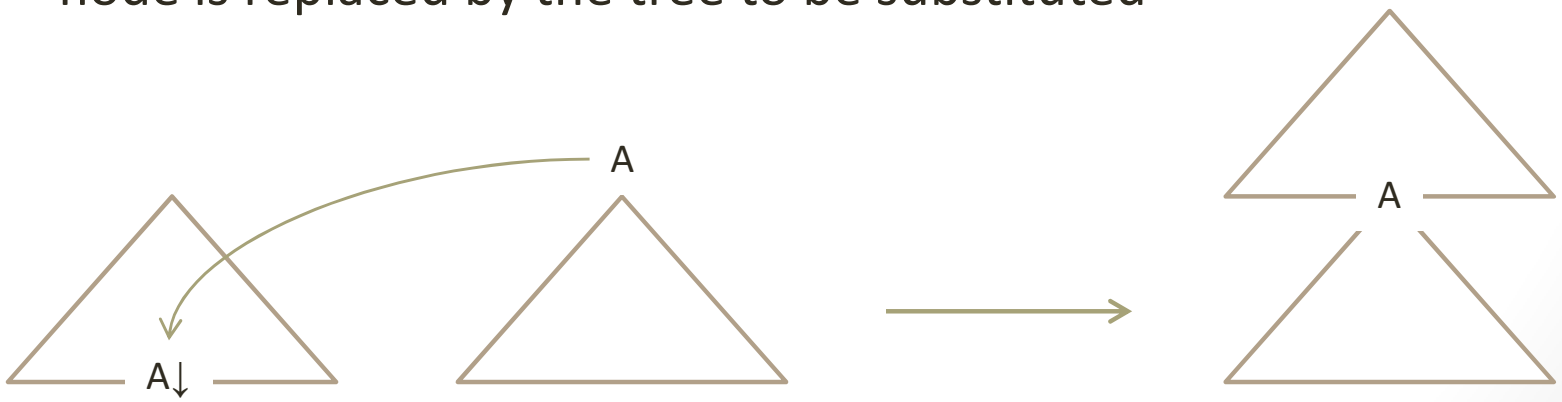
- Adjoining
  - Operators
    - auxiliary tree β – root labeled by X
    - any (initial, auxiliary or derived) tree α – contaning a non-substitution node n labeled by X
  - Resulting tree γ – obtained by adjoining β to α at node n
  - Adjoining on a node marked for substitution is disallowed.

TAG – operations

- Illustration of adjoining



- Substitution
  - on non-terminal nodes of the frontier of the tree
  - node is replaced by the tree to be substituted

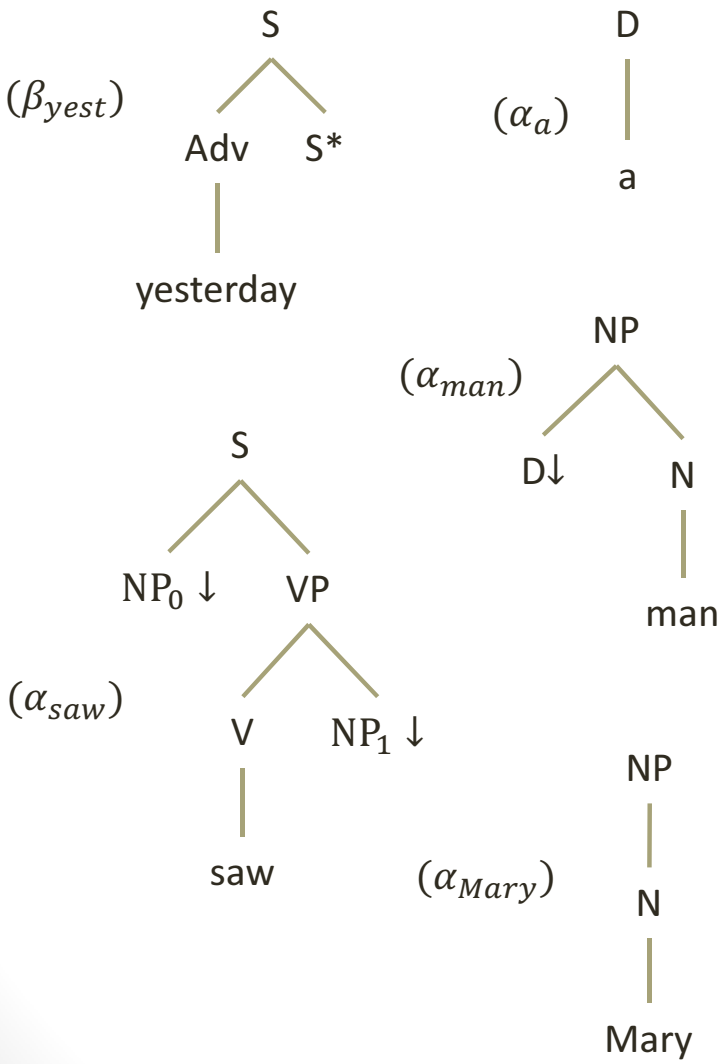


TAG – derivation

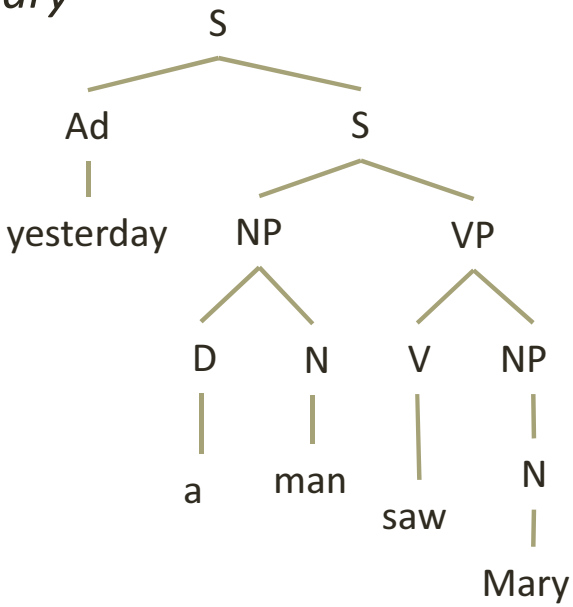
- Both operations, adjunction and substitution, are considered in a TAG derivation.
- Derived tree does not give enough information to determine how it was constructed, derivation tree specifies it uniquely.
- Derivation tree
  - Root node – labeled by an S-type initial tree
  - Other nodes – labeled by auxiliary trees in case of adjunction or initial trees in the case of substitution
  - A tree address is associated with each node – address of the node in the parent tree to which operation has been performed.
    - 0 – address of the root node, k – address of the k^{th} child of the root node, $p \cdot q$ – address of the q^{th} child of the node at address p.
  - Lines (convention)
    - unbroken line – joins tree adjoined to its parent
    - dashed line – joins tree substituted to its parent

TAG – derivation example

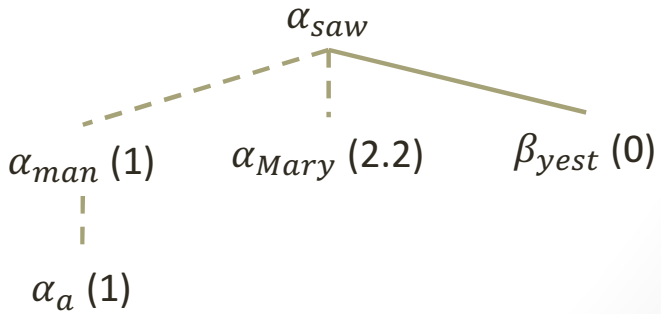
- Elementary trees of some TAG



- Derived tree for: *yesterday a man saw Mary*



- Derivation tree for: *yesterday a man saw Mary*



Properties of TAGs

- Tree set of a TAG:
  - *T_G = {t | t is derived from some S-rooted initial tree and t is completed}*
  - Initial tree is completed if there is no substitution nodes on the frontier of it.
- The string language of a TAG:
  - *L_G = {w | w is the yield of some t in T_G}*
- Properties of string languages of a TAG (TAL):
  - $CFL \subset TAL \subset Indexed\ Languages\ \subset CSL$
  - All closure properties of context-free languages also hold for tree-adjoining languages.
  - Tree-adjoining languages can be parsed in polynomial time, in the worst case in $O(n^6)$ time.

Lexicalized grammars

- A grammar is lexicalized if it consist (only) of:
  - a finite set of structures (lexicon) each associtated with a lexical item; each lexical item will be called the anchor of the coresponding structure;
  - an operation or operations for composing the structures.
  - Requirements:
    - anchor must be an overt lexical item (not the empty string)
    - structures be of finite size
    - combining operations combine a finite set of structures into a finite number of structures

- A structure is lexicalized if there is at least one overt lexical item that appears in it.
  - One is designated as the anchor or the subset of more are designated as multi-component anchor.

Lexicalization

- Set of selected structures during analysis of an arbitrary sentence of finite length is finite. ⇒ These structures can be combined in finitely many ways. ⇒ Lexicalized grammars are finitely ambiguous.
- A sentence of finite length can only be finitely ambiguous. ⇒ The search space used for analysis is finite. ⇒ It is decidable whether or not a string is accepted by a lexicalized grammar.

- Lexicalization
  - We say that a formalism F can be lexicalized by another formalism F', if for any finitely ambiguous grammar G in F there is a grammar G' in F' such that G' is a lexicalized grammar and such that G and G' generate the same tree set.

- Can context-free grammars be lexicalized?

Lexicalization of CFGs

- Chain rules obtained by derivation ($X \Rightarrow^* X$) or elementary ($X \rightarrow X$) are disallowed. (They generate infinitely ambiguous branches without introducing lexical items.)

- Lexicalized CFG
  - each production rule has a terminal symbol on its right side
  - combining operation – standard substitution

- Greibach Normal Form CFG
  - weak lexicalization – it does not give us the same set od trees as the original CFG

Lexicalization of CFGs - substitution

- Extension of the domain of locality of CFGs – making lexical items appear as part of the elementary structures by using a grammar on trees that uses substitution as comb. operation:

- Tree-Substitution Grammar (TSG)

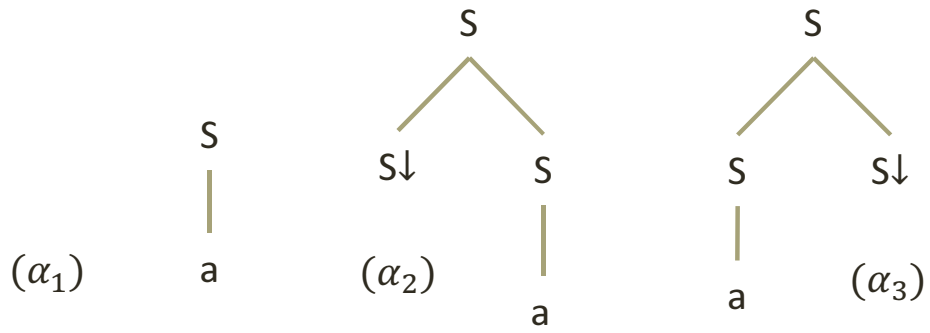
$$G = (\Sigma, N, I, S)$$

- where
  - Σ – finite set of terminal symbols
  - N – finite set of non-terminal symbols: $\Sigma \cap N = \emptyset$;
  - S – distinguished non-terminal symbol: $S \in N$;
  - I – finite set of finite trees, called initial trees:
    - interior nodes – labeled by non-terminal symbols
    - frontier nodes – terminals or non-teminals , non-terminals marked for substitution (anotated with a down arrow ↓)

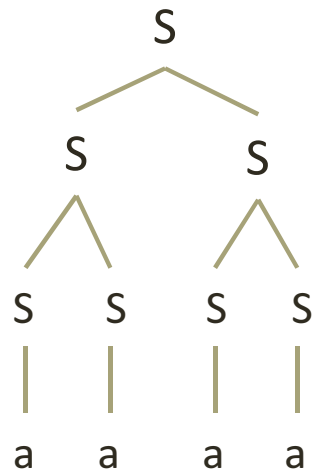
Lexicalization of CFGs with TSGs

- Finitely ambiguous context-free grammars cannot be lexicalized with a tree-substitution grammar.
- Proof by contradiction
  - Suppose that finitely ambiguous CFGs can be lexicalized with TSG.
  - Idea: Any derived tree from lexicalized TSG G includes at least one branch of bounded length from the root node to a node n labeled by a (node on the frontier of an arbitrary initial tree t in G).
  - Counter-example (CFG_1):
    - $S \rightarrow S\ S, S \rightarrow a$
    - a can occur arbitrary far from the root of the derivation tree.
  - Contradiction. ■

Lexicalization of CFGs with TSGs

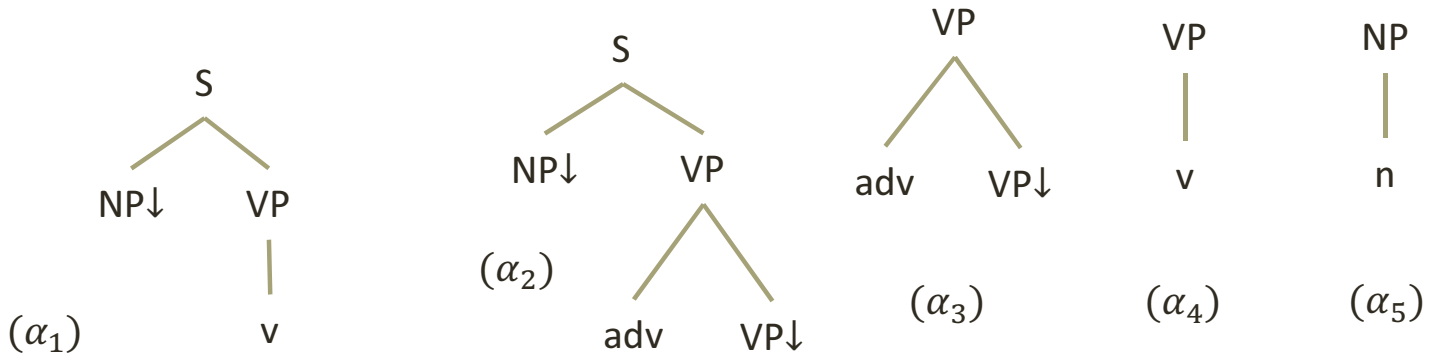


- Lexicalized TSG given above does not generate all the trees generated by the CFG (counter-example); for example:



Lexicalization of CFGs with TSGs

- Even if some CFGs can be lexicalized by using TSG, the choice of the lexical items that emerge as the anchor may be too restrictive.
- Example (CFG_2):
  - $S \rightarrow NP\ VP,\ VP \rightarrow \text{adv VP},\ VP \rightarrow v,\ NP \rightarrow n$
- Possible lexicalized TSG:

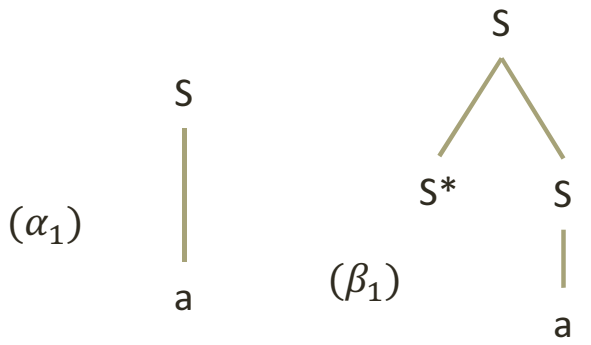


- This lexicalization forces to choose *adv* (or *n*) as the anchor of a structure α_3, and it cannot be avoided. (Choice is not linguistically motivated.)

Lexicalization of CFGs with TAGs

- Additional combining operation – adjunction.
- A tree-based system that uses substitution and adjunction coincides with TAG.

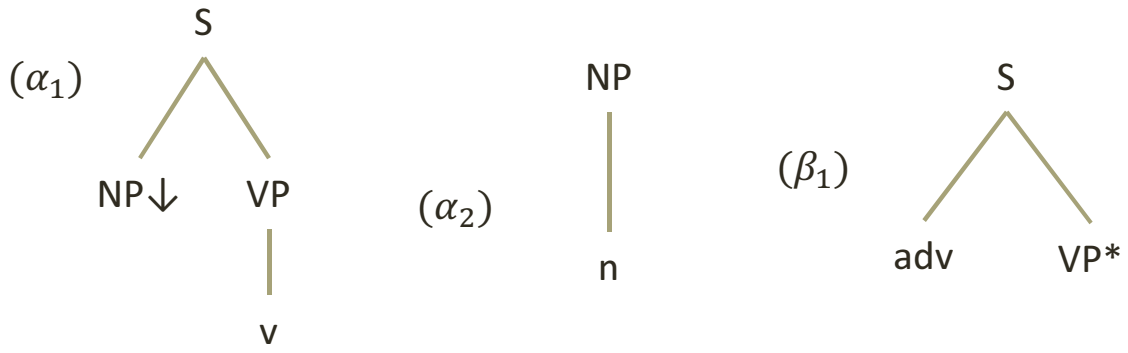
- Example
  - CFG_1 lexicalized with TAG:



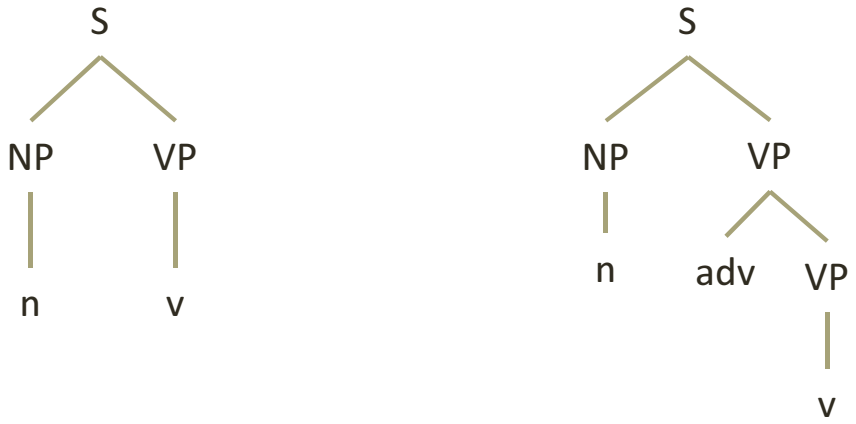
  - This lexicalization can derive all possible derived trees of CFG_1.

Lexicalization of CFGs with TAGs

- Example
  - CFG_2 lexicalized with TAG:



  - Now its possible to choose the anchor freely (α_1).
  - The following trees can be derived:



Lexicalization of CFGs with TAGs

- If $G = (\Sigma, N, P, S)$ is a finitely ambiguous CFG which does not generate the empty string, then there is lexicalized tree-adjoining grammar $G_{lex} = (\Sigma, N, I, A, S)$ generating the same language and tree set as G. Furthermore G_{lex} can be chosen to have no substition nodes in any elementary trees.

- TAGs are closed under lexicalization
  - If G is a finitely ambiguous TAG that uses substition and adjunction as combining operation, s.t. $\lambda \notin L(G)$, then there exists a lexicalized TAG G_{lex} which generates the same language and the same tree set as G.

Embedded push-down automaton (EPDA)

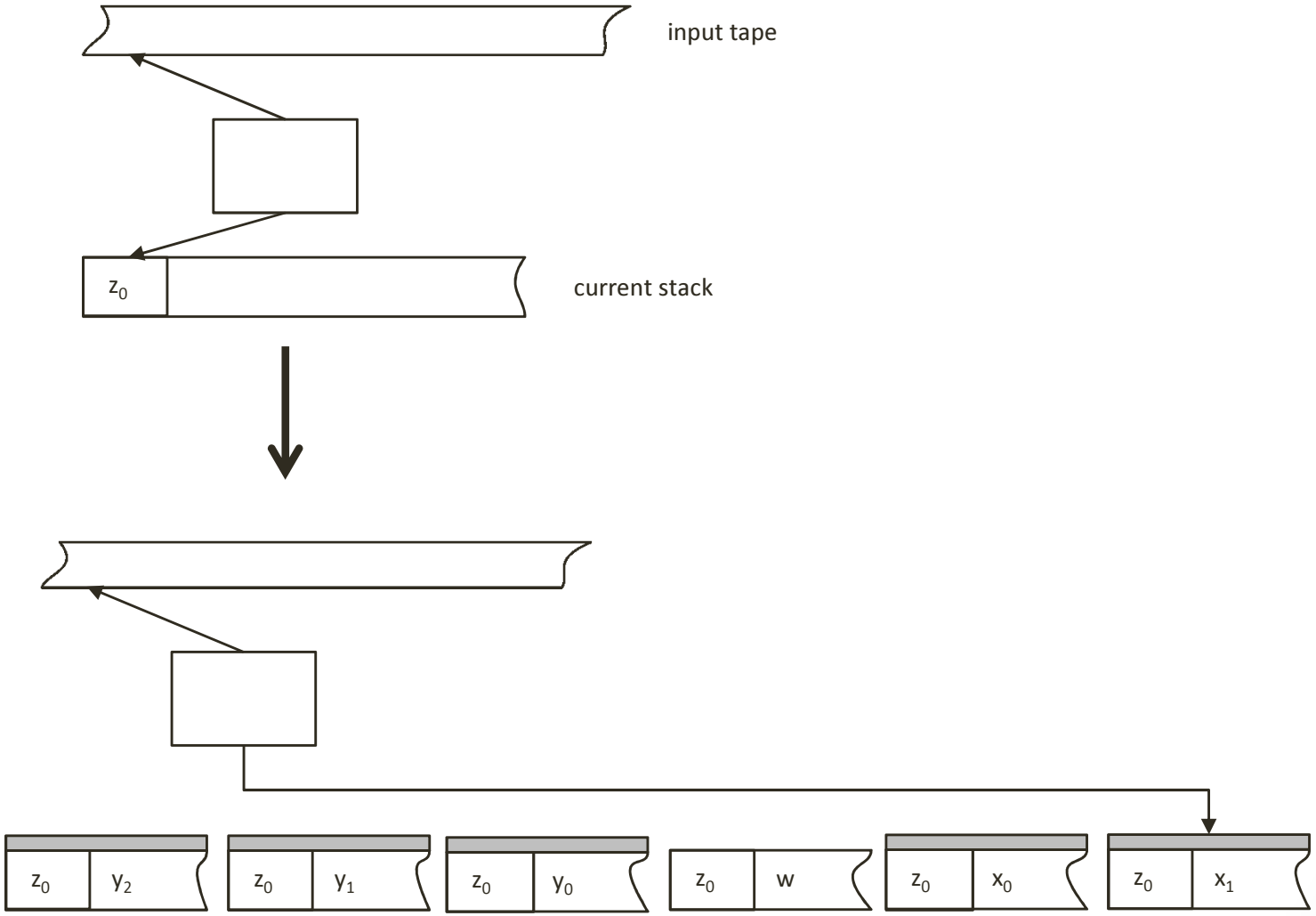
- Accepts TAL
- Similar to a PDA except that the push-down store is a sequence od stacks.
- The stack head is always at the top of a stack, if it reaches the bottom of a stack, it automatically moves to the top of the stack below (left of) the current stack.
- M starts with one stack and may create new stack above and below the current stack.
- Transition function δ:

$$\delta = (\text{current state, input symbol, stack symbol}) =$$
$$(\text{new state, } sb_1, sb_2, \ldots, sb_m, \text{ push/pop on current stack, } st_1, st_2, \ldots, st_n)$$

  - where $sb_1, sb_2, \ldots, sb_m$ are the stacks introduced below the current stack
  - and $st_1, st_2, \ldots, st_n$ are the stacks introduced above the current stack
- In each of the new stacks specified information may be pushed.

Embedded push-down automaton (EPDA)

- Illustration of the move of EPDA



Literature

- G. Rozenberg and A. Salomaa.
  Handbook of Formal Languages, volume 3.
  Springer, Berlin, 1997.