# CS/ENGRI 172, Fall 2002

## 11/6/02: Lecture Twenty-Nine Handout

**Topics**: Push-down automata (PDAs).
**Announcements**: The final exam is scheduled for December 20th, 9-11:30am, Olin 165.

### The PDA formalism

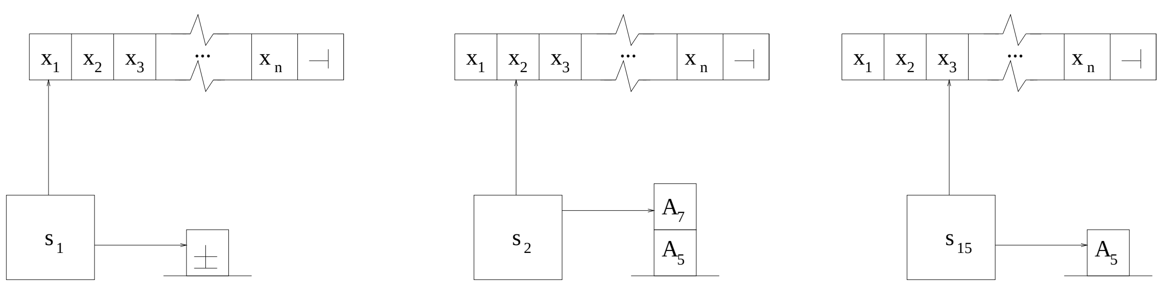
Push-down automata are essentially limited versions of Turing machines. We only consider *deterministic* PDAs; that is, for any given configuration, at most one move, and perhaps no move, is possible.

Suppose we have a PDA $P$ with

- $m$ distinct states $s_1, s_2, \dots, s_m$, where $s_1$ is the initial state and $s_m$ is the accept state;
- an input alphabet consisting of $\ell$ distinct symbols $a_1, a_2, \dots, a_\ell$, with $a_1$ being the right-end marker $\dashv$;
- a stack alphabet consisting of $k$ distinct symbols $A_1, A_2, \dots, A_k$, with $A_1 = \pm$, the initial stack symbol

(naturally, we assume $\ell \geq 2$ and $k, m \geq 1$). Then, a legal input to $P$ would be $x = x_1 x_2 \dots x_n$, each $x_i$ drawn from among $a_2, \dots, a_\ell$ (so the input can't contain the end marker, but repeats are allowed).

$P$'s rules must all be of the form $(s, a_i, A_j) \to (s', \alpha)$ where $s$ and $s'$ are states, $a_i$ is a single input symbol, $A_j$ is a single stack symbol denoting what symbol is on top of the stack, and $\alpha$, designating a replacement for $A_j$ on the stack, is either a sequence of stack symbols or the word "pop". No two rules can have the same left-hand side. If $P$ had rules $(s_1, x_1, \pm) \to (s_2, A_7 A_5)$ and $(s_2, x_2, A_7) \to (s_{15}, \text{pop})$, then the first three *configurations* of $P$ on input $x$ would be as follows:




$P$ *accepts* $x$ if it can start in the initial configuration corresponding to $x$ and, obeying its rules, have the input head fall off the tape while changing to its accept state. If it would halt in any other configuration — i.e., it gets stuck somewhere on the input tape or falls off the tape but ends up in a state other than the accept state — it does not accept $x$.