Nondeterministic (operational) models (1)

• Usually one thinks of an algorithm as a determined sequence of operations
  – In a certain state with a certain input there is no doubt on the next step

• Example: let us compare

  \[
  \text{if } x > y \text{ then } \text{max} := x \text{ else } \text{max} := y
  \]

  with

  \[
  \text{if}
  \begin{align*}
  x &\geq y \text{ then } \text{max} := x \\
  x &\leq y \text{ then } \text{max} := y
  \end{align*}
  \]

  \[
  \text{fi}
  \]
Nondeterministic (operational) models (2)

• Is it only a matter of elegance?
• Let us consider the `case` construct of Pascal: why not having something like the following?

```
case
    x=y   then  S1
    z>y+3 then  S2
...  then
endcase
```
Another form of nondeterminism that is usually “hidden”
Search algorithms

• Search algorithms are a “simulation” of basically nondeterministic algorithms
  – Is the element searched for in the root?
    If yes, ok
    Otherwise
      Search the left subtree
      Search the right subtree

• Choice of priority among paths is often arbitrary
In conclusion

- Nondeterminism (ND) is a model of computation or at least a model of parallel computing
  - Ada and other concurrent languages exploit it
- It is a useful abstraction to describe search problems and algorithms
- It can be applied to various computational models
- Important: ND models must not be confused with stochastic models
Adding nondeterminism

\[ \delta(q_1, a) = \{q_2, q_3\} \]
Nondeterministic FSAs

• A nondeterministic FSA (NDFSA) is a tuple  
  \(<Q, I, \delta, q_0, F>\), where  
  – \(Q, I, q_0, F\) are defined as in (D)FSAs  
  – \(\delta: Q \times I \rightarrow \mathcal{P}(Q)\)  

• What happens to \(\delta^*\)?
Move sequence

• $\delta^*$ is inductively defined from $\delta$

\[
\delta^*(q, \varepsilon) = \{q\}
\]
\[
\delta^*(q, y.i) = \bigcup_{q' \in \delta^*(q, y)} \delta(q', i)
\]

• Example:

$\delta(q_1, a) = \{q_2, q_3\}$,
$\delta(q_2, b) = \{q_4, q_5\}$,
$\delta(q_3, b) = \{q_6, q_5\}$

$\rightarrow \delta^*(q_1, ab) = \{q_4, q_5, q_6\}$
Acceptance condition

\[ x \in L \iff \delta^*(q_0, x) \cap F \neq \emptyset \]

Among the various possible runs (with the same input) of the NDFSA, it is sufficient that one of them succeeds to accept the input string

- **Existential nondeterminism**
  - There exists a universal interpretation, too: \( \delta^*(q_0, x) \subseteq F \)
**DFSA vs NDFSA**

- Starting from $q_1$ and reading ab the automaton reaches a state that belongs to the set \{q_4, q_5, q_6\}
- Let us call again “state” the set of possible states in which the NDFSA can be during the run
Formally

- NDFSAs have the same power as DFSAs
- Given a NDFSA, an equivalent DFSA can be automatically synthesized as follows

If \( A_{ND} = \langle Q, I, \delta, q_0, F \rangle \) then

\( A_D = \langle Q_D, I, \delta_D, q_{0D}, F_D \rangle \), where

- \( Q_D = \mathcal{P}(Q) \)
- \( \delta_D(q_D,i) = \bigcup_{q \in q_D} \delta(q,i) \)
- \( q_{0D} = \{q_0\} \)
- \( F_D = \{q_D \mid q_D \in Q_D \land q_D \cap F \neq \emptyset\} \)
Example (1)

Transform the following NDFSA into the equivalent DFSA

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q2</td>
</tr>
<tr>
<td>q0</td>
<td></td>
<td>q1</td>
</tr>
<tr>
<td>q0</td>
<td>q1</td>
<td>q3</td>
</tr>
<tr>
<td>q1</td>
<td>q3</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td></td>
<td>q0</td>
</tr>
<tr>
<td>q3</td>
<td>q2</td>
<td>q0</td>
</tr>
</tbody>
</table>

Nondeterminism
Example (2)

- Let us proceed recursively

<table>
<thead>
<tr>
<th>$q_0$</th>
<th>0</th>
<th>$q_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>0</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_0$</td>
<td>1</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>0</td>
<td>$q_3$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>1</td>
<td>$q_3$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>0</td>
<td>⊥</td>
</tr>
<tr>
<td>$q_2$</td>
<td>1</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>0</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>1</td>
<td>$q_0$</td>
</tr>
</tbody>
</table>
Example (3)

Graphically

Nondeterminism
Why ND?

• NDFSAs are no more powerful than FSAs, but they are not useless
  – It can be easier to design a NDFSA
  – They can be exponentially smaller w.r.t. the number of states

• Example: a NDFSA with 5 states becomes in the worst case a FSA with $2^5$ states
Nondeterministic TM

• To define a nondeterministic TM (NDTM), we need to change the transition function and the translation mapping

• All the other elements remain as in a (D)TM

• The transition function is

$$\delta: (Q-F) \times I \times \Gamma^k \rightarrow \mathcal{P}(Q \times \Gamma^k \times \{R,L,S\}^{k+1})$$

and the output mapping

$$\eta: (Q-F) \times I \times \Gamma^k \rightarrow \mathcal{P}(O \times \{R,S\})$$
Nondeterminism
Acceptance condition

- A string $x \in I^*$ is accepted by a NDTM if and only if there exists a computation that terminates in an accepting state.
- It would seem that the problem of accepting a string can be reduced to a visit of a computation tree:
  - How should we perform the visit?
  - What about the relationship between DTM$s$ and NDTM$s$?
Visiting the computation tree

• We know different kinds of visits:
  – Depth-first visit
  – Breadth-first visit

• A depth-first visit cannot work in our problem because the computation tree could have infinite paths and the algorithm might “get stuck” in it

• We should adopt a breadth-first visit algorithm
DTM vs NDTM

Can we build a DTM that visits a tree level by level?

It is a long (and boring) exercise, but YES

– We can build a DTM that establishes whether a NDTM recognizes a string x
– Given a NDTM, we can build an equivalent DTM

ND does not add power to TMs
ε Moves and PDAs

• ε-moves came with the following constraint:
  \[
  \text{If } \delta(q, \epsilon, A) \neq \bot, \text{ then } \delta(q, i, A) = \bot \quad \forall i \in I
  \]

• Without this constraint the presence of ε-moves would make PDAs intrinsically nondeterministic
Adding nondeterminism to PDAs

• Removing the constraint already makes the PDA nondeterministic

• Additionally, we can have nondeterminism by changing the transition function of a PDA and consequently:
  – transitions among configurations
  – acceptance condition
Definition

A nondeterministic PDA (NDPDA) is a tuple \( <Q, I, \Gamma, \delta, q_0, Z_0, F> \)

- \( Q, I, \Gamma, q_0, Z_0, F \) as in (D)PDA
- \( \delta \) is the transition function defined as

\[
\delta: Q \times (I \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}_F(Q \times \Gamma^*)
\]

- What is the \( F \) in \( \mathcal{P}_F \)?
- Why \( F \)?
Transition function

\[ \delta: Q \times (I \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}_F(Q \times \Gamma^*) \]

- \( \mathcal{P}_F \) indicates the *finite* subsets of \( Q \times \Gamma^* \)
  - Why did we not specify it for NDTM?
- Graphically:
Transition among configurations

Relation $\vdash$ on $Q \times I^* \times \Gamma^* \times Q \times I^* \times \Gamma^*$ is defined by $c=<q,x,y> \vdash c’=<q’, x’, y’>$ if and only if

– Case 1
  • $x=iy$, $x’=y$
  • $y=A\beta$, $y’=\alpha\beta$
  • $<q’, \alpha> \in \delta(q,i,A)$

– Case 2
  • $x’=x$
  • $y=A\beta$, $y’=\alpha\beta$
  • $<q’, \alpha> \in \delta(q,\varepsilon,A)$
Acceptance condition

- Given a NDPDA P

\[ \forall x \in I^* \ (x \in L(P) \iff c_0=\langle q_0,x,Z_0 \rangle \vdash^* c_F=\langle q,\varepsilon,\gamma \rangle \]  
and \( q \in F \)

- Informally, a string is accepted by a PDA if there is a path coherent with x on the PDA that goes from the initial state to a final state
  - The input string has to be read completely
Effects of nondeterminism

• ND does not add expressive power to
  – TMs
  – FSAs
• Does ND add expressive power to DPDAs?
Nondeterminism

NDPDAs vs DPDAs (1)

• Obviously a NDPDA can recognize all the languages recognizable by DPDAs
• ND allows

\[
\text{q}_1 \xrightarrow{i, A/\alpha} \text{q}_2 \quad \text{q}_1 \xrightarrow{i, A/\beta} \text{q}_3
\]

so NPDAs can recognize \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}
\{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}
Languages recognizable by NDPDAs are called context-free languages.
Nondeterminism

NDPDA vs TM

- (a) and (c): NO!
  - A (N)DTM can simulate a NDPDA by using the tape as a stack
- (d): NO!
  - The stack is still a destructive memory
  - $a^n b^n c^n$ is not recognizable by a NDPDA
The bull’s-eye

- Regular languages
- Deterministic context-free languages
- Context-free languages
- Recursively enumerable languages

Nondeterminism
Closure properties in DPDAs

• In DPDAs we have
  – Closure w.r.t. complement
  – Non-closure w.r.t. union, intersection, difference

• Does changing the power of the automata change their behavior w.r.t. operations?
  – It can happen
  – New power, new language
Union (1)

- NDPDAs are **closed** under union
  - Intuition:

- Given two NDPDAs, $P_1$ and $P_2$, we can always build a NDPDA that represents the union by creating a new initial state that is connected to both initial states of $P_1$ and $P_2$ with an $\epsilon$-move.
Union (2)

Nondeterminism
Intersection

• The closure w.r.t. intersection still does not hold
• Consider
  – \{a^n b^n c^*\}
  – \{a^* b^n c^n\}
both are recognizable by (N)DPDAs, but \{a^n b^n c^*\} \cap \{a^* b^n c^n\} = \{a^n b^n c^n\} is not recognizable by any NDPDA
Complement (1)

- If a class of languages is closed w.r.t. union, but not w.r.t. intersection it cannot be closed w.r.t. complement
  - We can write intersection in terms of union and complement
- NDPDAs are not closed w.r.t complement
Remarks

• If a machine is deterministic and its computation terminates, the complement can be obtained by
  – Completing the machine
  – Swapping accepting and non-accepting states

• Nondeterminism, like non-termination (infinite computation), makes this approach unfeasible
Complement (2)

- For NDPDAs, computations can always be made terminating (as for DPDAs)
- However, ND can cause this problem:

  One can have two computations:

  - $c_0=\langle q_0, x, Z_0 \rangle \vdash^* c_1=\langle q_1, \epsilon, \gamma \rangle$
  - $c_0=\langle q_0, x, Z_0 \rangle \vdash^* c_2=\langle q_2, \epsilon, \gamma \rangle$

  with $q_1 \in F$ and $q_2 \notin F$

  → even if we swap accepting and non-accepting state, $x$ is still accepted