Recursion and complexity
(4th lecture)

Isabel Oitavem
CMA and DM, FCT-UNL

This work was partially supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the project UID/MAT/00297/2013 (Centro de Matemática e Aplicações).
Models of computation: Nondeterministic Turing machines

A deterministic Turing machine (TM) with \( k \) tapes is a four-tuple

\[ M = \langle Q, \Sigma, \delta, q_0 \rangle \]

where

- \( Q \) is a finite set of states;
- \( \Sigma \) is the tape alphabet;
- \( \delta \) is the transition function,
  \[ \delta : Q \times \Sigma^k \rightarrow Q \times \Sigma^k \times \{L, N, R\}^k; \]
- \( q_0 \in Q \) is the initial state.
Models of computation: Nondeterministic Turing machines

A nondeterministic Turing machine (NTM) with $k$ tapes is a five-tuple

$$M = \langle Q; \Sigma, \delta, q_0, F \rangle$$

where

- $Q$ is a finite set of states;
- $\Sigma$ is the tape alphabet;
- $\delta$ is the transition function,
  $$\delta : Q \times \Sigma^k \to \mathcal{P}(Q \times \Sigma^{k-1} \times \{L, N, R\}^k);$$
- $q_0 \in Q$ is the initial state;
- $F$ is the set of accepting final states.
An input $w$ is accepted by a nondeterministic machine $M$ if, and only if, there exits a computation of $M$ on $w$ ending in an accepting configuration.
Models of computation: Nondeterministic Turing machines

An input $w$ is accepted by a nondeterministic machine $M$ if, and only if, there exits a computation of $M$ on $w$ ending in an accepting configuration.

Or alternatively, we define a bottom-up labeling of the computation tree (or part of it) of $M$ on $w$ by the following rules:

- the accepting leaves are labeled 1;
- any node is labeled 1 if at least one of its sons is labeled 1.

The machine accepts $w$ if, and only if, the root is labeled 1.
Models of computation: Alternating Turing machines

A nondeterministic Turing machine (NTM) with \( k \) tapes is a five-tuple

\[
M = \langle Q; \Sigma, \delta, q_0, F \rangle
\]

where
- \( Q \) is a finite set of states;
- \( \Sigma \) is the tape alphabet;
- \( \delta \) is the transition function,
  \[
  \delta : Q \times \Sigma^k \rightarrow \mathcal{P}(Q \times \Sigma^{k-1} \times \{L, N, R\}^k);
  \]
- \( q_0 \in Q \) is the initial state;
- \( F \) is the set of accepting final states.
A alternating Turing machine (ATM) with $k$ tapes is a five-tuple

\[ M = \langle Q; \Sigma, \delta, q_0, \gamma \rangle \]

where

- $Q$ is a finite set of states;
- $\Sigma$ is the tape alphabet;
- $\delta$ is the transition function,
  \[ \delta : Q \times \Sigma^k \rightarrow \mathcal{P}(Q \times \Sigma^{k-1} \times \{L, N, R\}^k); \]
- $q_0 \in Q$ is the initial state;
- $\gamma : Q \rightarrow \{\lor, \land, \text{acc}, \text{rej}\}$. 
Models of computation: Alternating Turing machines

Given a tree in which internal nodes are either existential (∨) or universal (∧), we consider the following labeling procedure

- the accepting leaves are labeled 1;
- any existential node is labeled 1 if at least one of its sons has been labeled 1;
- any universal node is labeled 1 if all its sons are labeled 1.

The machine accepts the input if and only if the root of the computation tree is labeled 1.
Implicit recursion-theoretic approach: $\text{FPspace}$

A function $f$ (over $\mathbb{W}$) is **computable in polynomial space** if, and only if, $f$ is bitwise computable by an **alternating Turing machine** in polynomial time, and the length of the outputs of $f$ is polynomial in the length of the inputs.
FPtime and FPspace: models of computation

- Model of computation
  - **FPtime**: Deterministic TM;
  - **FPspace**: Alternating TM.

- Resource constraint: polynomial time.

```
DTM       NTM and ATM

<table>
<thead>
<tr>
<th>c0</th>
<th>cε</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>∧</td>
</tr>
<tr>
<td>c1</td>
<td>c0</td>
</tr>
<tr>
<td></td>
<td>∧</td>
</tr>
<tr>
<td>c2</td>
<td>c00</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Implicit recursion-theoretic approach

$\text{FPtime} = [SI; SC, SR]$  \hspace{1cm} (Bellantoni-Cook 1992)

$\text{FPspace} = [SI; SC, ? ]$

$\text{SR}$ (Input-sorted recursion over $\mathbb{W}$):

\[
\begin{align*}
  f(\epsilon, \bar{x}; \bar{y}) &= g(\epsilon, \bar{x}; \bar{y}) \\
  f(z_0, \bar{x}; \bar{y}) &= h(z_0, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y})) \\
  f(z_1, \bar{x}; \bar{y}) &= h(z_1, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))
\end{align*}
\]

Example: $f(11)$ leads to $h(11, h(1, g(\epsilon)))$
Implicit recursion-theoretic approach

\[ \text{FPtime} = [SI; SC, SR] \quad (\text{Bellantoni-Cook 1992}) \]
\[ \text{FPspace} = [SI; SC, ?] \]

**SR** (Input-sorted recursion over \( \mathbb{W} \)):

\[
\begin{align*}
  f(\epsilon, \bar{x}; \bar{y}) &= g(\epsilon, \bar{x}; \bar{y}) \\
  f(z0, \bar{x}; \bar{y}) &= h(z0, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y})) \\
  f(z1, \bar{x}; \bar{y}) &= h(z1, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))
\end{align*}
\]

Example: \( f(11) \) leads to \( h(11, h(1, g(\epsilon))) \)

\[
\begin{array}{c}
  h \\
  \downarrow \\
  h \\
  \downarrow \\
  g \\
\end{array}
\]

SR reproduces the **sequential structure** of deterministic computations.
Implicit recursion-theoretic approach

\[
\text{FPtime} = [SI; SC, SR] \quad \text{(Bellantoni-Cook 1992)}
\]
\[
\text{FPspace} = [SI; SC, STR] \quad \text{(O. 2008)}
\]

\textbf{SR} (Input-sorted recursion over } \mathbb{W}:\n
\[
f(\epsilon, \bar{x}; \bar{y}) = g(\epsilon, \bar{x}; \bar{y})
\]
\[
f(z_0, \bar{x}; \bar{y}) = h(z_0, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))
\]
\[
f(z_1, \bar{x}; \bar{y}) = h(z_1, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))
\]

\textbf{STR} is defined analogously to \textbf{SR}, but

\begin{itemize}
  \item double the recursive call
  \item distinguish the recursive calls from each other via a pointer } p. \end{itemize}
Implicit recursion-theoretic approach

**FPtime** = $[SI; SC, SR]$  
**FPspace** = $[SI; SC, STR]$  
(Bellantoni-Cook 1992)  
(O. 2008)

**SR** (Input-sorted recursion over $\mathbb{W}$):

$$f(\epsilon, \bar{x}; \bar{y}) = g(\epsilon, \bar{x}; \bar{y})$$

$$f(z0, \bar{x}; \bar{y}) = h(z0, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))$$

$$f(z1, \bar{x}; \bar{y}) = h(z1, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))$$

**STR**:

$$f(\epsilon, p, \bar{x}; \bar{y}) = g(\epsilon, p, \bar{x}; \bar{y})$$

$$f(z0, p, \bar{x}; \bar{y}) = h(z0, p, \bar{x}; \bar{y}, f(z, p0, \bar{x}; \bar{y}), f(z, p1, \bar{x}; \bar{y}))$$

$$f(z1, p, \bar{x}; \bar{y}) = h(z1, p, \bar{x}; \bar{y}, f(z, p0, \bar{x}; \bar{y}), f(z, p1, \bar{x}; \bar{y}))$$
Implicit recursion-theoretic approach

**STR:** $f(\epsilon, p, \bar{x}; \bar{y}) = g(\epsilon, p, \bar{x}; \bar{y})$

\[ f(z_0, p, \bar{x}; \bar{y}) = h(z_0, p, \bar{x}; \bar{y}, f(z, p_0, \bar{x}; \bar{y}), f(z, p_1, \bar{x}; \bar{y})) \]

\[ f(z_1, p, \bar{x}; \bar{y}) = h(z_1, p, \bar{x}; \bar{y}, f(z, p_0, \bar{x}; \bar{y}), f(z, p_1, \bar{x}; \bar{y})) \]

**Example:**

$f(11, \epsilon)$ leads to $h(\epsilon, h(0, g(00), g(01)), h(1, g(10), g(11))).$
Implicit recursion-theoretic approach

**STR:** \( f(\epsilon, p, \bar{x}; \bar{y}) = g(\epsilon, p, \bar{x}; \bar{y}) \)

\[
f(z_0, p, \bar{x}; \bar{y}) = h(z_0, p, \bar{x}; \bar{y}, f(z, p_0, \bar{x}; \bar{y}), f(z, p_1, \bar{x}; \bar{y}))
\]

\[
f(z_1, p, \bar{x}; \bar{y}) = h(z_1, p, \bar{x}; \bar{y}, f(z, p_0, \bar{x}; \bar{y}), f(z, p_1, \bar{x}; \bar{y}))
\]

**Example:**
\( f(11, \epsilon) \) leads to \( h(\epsilon, h(0, g(00), g(01)), h(1, g(10), g(11))) \).

\[
\begin{array}{c}
h\epsilon \\
\land \\
\h0 \land h1 \\
\land \\
g00 \land g01 \land g10 \land g11
\end{array}
\]

The mentioned input is the **pointer**, and it gives the **address from the root of the tree to the current node**.

**STR** trivially extends **SR**.
Implicit recursion-theoretic approach

Example:
\[ f(11, \epsilon) \text{ leads to } h(\epsilon, h(0, g(00), g(01)), h(1, g(10), g(11))). \]

\[
\begin{array}{c}
h_{\epsilon} \\
\land \\
\land \\
\land \\
g_{00} \quad g_{01} \\
\quad \land \\
g_{10} \quad g_{11}
\end{array}
\]

Bottom-up labeling:
(assuming that non-terminating configurations have two successor configurations)
Implicit recursion-theoretic approach

Example:
$f(11, \epsilon)$ leads to $h(\epsilon, h(0, g(00), g(01)), h(1, g(10), g(11)))$.

Bottom-up labeling:
(assuming that non-terminating configurations have two successor configurations)
$g$ and $h$ execute the computation determined by the pointer and read the state of the last computed configuration:
Implicit recursion-theoretic approach

Example:
\( f(11, \epsilon) \) leads to \( h(\epsilon, h(0, g(00), g(01)), h(1, g(10), g(11))) \).

\[
\begin{aligned}
\hskip 0.5 in h \epsilon \\
\wedge \\
\hskip 0.5 in h 0 \quad h 1 \\
\wedge \\
\hskip 0.5 in g 00 \quad g 01 \quad g 10 \quad g 11
\end{aligned}
\]

Bottom-up labeling:
(assuming that non-terminating configurations have two successor configurations)

\( g \) and \( h \) execute the computation determined by the pointer and read the state of the last computed configuration:

- \( g \) returns 1 if it is an accepting state; 0 otherwise.
Implicit recursion-theoretic approach

Example:

\[ f(11, \epsilon) \text{ leads to } h(\epsilon, h(0, g(00), g(01)), h(1, g(10), g(11))). \]

\[
\begin{array}{c}
h_\epsilon \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\end{array}
\]

Bottom-up labeling:

(assuming that non-terminating configurations have two successor configurations)

\textit{g} and \textit{h} execute the computation determined by the pointer and read the state of the last computed configuration:

\begin{itemize}
  \item \textit{g} returns 1 if it is an accepting state; 0 otherwise.
  \item \textit{h} does \lor or \land of its last two inputs, depending on the read state.
\end{itemize}
\[ \text{FPtime} = [SI; SC, SR] \]
\[ \text{FPspace} = [SI; SC, STR] \]

(Bellantoni-Cook 1992)

(O. 2008)
$$\text{FPtime} = [SI; SC, SR]$$  
$$\text{FPspace} = [SI; SC, STR]$$  

(Bellantoni-Cook 1992)  
(O. 2008)

<table>
<thead>
<tr>
<th>Class</th>
<th>Model of Computation</th>
<th>time bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPtime</td>
<td>DTM</td>
<td>poly</td>
</tr>
<tr>
<td>NP</td>
<td>NTM</td>
<td>poly</td>
</tr>
<tr>
<td>FPspace</td>
<td>ATM</td>
<td>poly</td>
</tr>
<tr>
<td>PP</td>
<td>PTM</td>
<td>poly</td>
</tr>
<tr>
<td>BPP</td>
<td>PTM</td>
<td>poly + bounded error</td>
</tr>
</tbody>
</table>