CS375: Logic and Theory of Computing

Fuhua (Frank) Cheng

Department of Computer Science
University of Kentucky
Table of Contents:

- **Week 1: Preliminaries** (set algebra, relations, functions) (read Chapters 1-4)
- **Weeks 3-6: Regular Languages, Finite Automata** (Chapter 11)
- **Weeks 7-9: Context-Free Languages, Pushdown Automata** (Chapters 12)
- **Weeks 10-12: Turing Machines** (Chapter 13)
Table of Contents (conti):

- Weeks 13-14: Propositional Logic (Chapter 6), Predicate Logic (Chapter 7), Computational Logic (Chapter 9), Algebraic Structures (Chapter 10)
The Church-Turing Thesis: Anything that is intuitively computable can be computed by a Turing machine.
It is a thesis rather than a theorem because it relates the informal notion of intuitively computable to the formal notion of a Turing machine.

Computational Models
A computational model is a characterization of a computing process that describes the form of a program and describes how the instructions are executed.

Example. The Turing machine computational model describes the form of TM instructions and how to execute them.
Example. If X is a programming language, the X computational model describes the form of a program and how each instruction is executed.

Equivalence of Computational Models

Two computational models are equivalent in power if they solve the same class of problems.

Any piece of data for a program can be represented by a string of symbols and any string of symbols can be represented by a natural number.

So even though computational models may process different kinds of data, they can still be compared with respect to how they process natural numbers.
Assumption: there is an unlimited amount of memory available. So we can represent any natural number or any finite string.

Each of the following models of computation is equal in power to the TM model.

The Simple Programming Language

This imperative programming model processes natural numbers. The language is defined as follows:

• Variables have type $\mathbb{N}$.
• Assignment statements: $X := 0$; $X := \text{succ}(Y)$; $X := \text{pred}(Y)$.
  (assume $\text{pred}(0) = 0$)
• Composition of statements: $S1; S2$.
• while $X \neq 0$ do $S$ od.
8. Turing Machines and Equivalent Models — The Church-Turing Thesis

This simple language model has the same power as the Turing machine model.

For input and output use the values of the variables before and after execution.

Example. To demonstrate the power of this language, define the following macros.

Some Macros

\[ X := Y \]
\[ X := 2 \]
\[ Z := Z + X \]
\[ Z := X + Y \]
\[ Z := \text{X monus } Y \]

Macro Expansion

\[ X := \text{succ}(Y); \ X := \text{pred}(X). \]
\[ X := 0; \ X := \text{succ}(X); \ X := \text{succ}(X). \]
\[ C := X; \ \textbf{while } C \neq 0 \ \text{do } Z := \text{succ}(Z); \ C := \text{pred}(C) \ \textbf{od}. \]
\[ Z := X; \ C := Y; \ \textbf{while } C \neq 0 \ \text{do } Z := \text{succ}(Z); \ C := \text{pred}(C) \ \textbf{od}. \]
\[ Z := X; \ C := Y; \ \textbf{while } C \neq 0 \ \text{do } Z := \text{pred}(Z); \ C := \text{pred}(C) \ \textbf{od}. \]
Example. To demonstrate the power of this language, define the following macros:

**Some Macros**

```plaintext
while X < Y do S od
if X ≠ 0 then S₁ else S₂ fi
```

**Macro Expansion**

```plaintext
T := Y monus X; while T ≠ 0 do S; T := Y monus X od.
U := X; V := 1;
while U ≠ 0 do S₁; V := 0; U := 0 od;
while V ≠ 0 do S₂; V := 0 od.
```
Partial Recursive Functions

This model consists of a set of functions that take natural numbers as arguments/values. The functions are defined as follows:

• Initial functions: zero\( (x) = 0 \), succ\( (n) = n + 1 \), and projections (e.g., \( p_2(a, b, c) = b \)).

• Composition: e.g., \( f(x) = h(g_1(x), \ldots, g_m(x)) \), where \( h, g_1, \ldots, g_m \) are partial recursive.

• Primitive recursion:
  
  \[
  f(x, 0) = h(x) \quad \text{(\( h \) is partial recursive)}
  \]
  
  \[
  f(x, \text{succ}(y)) = g(x, y, f(x, y)) \quad \text{(\( g \) is partial recursive)}
  \]
Partial Recursive Functions (cont.)

- Initial functions: \( \text{zero}(x) = 0 \), \( \text{succ}(n) = n + 1 \), and projections (e.g., \( p_2(a, b, c) = b \)).
- Composition: e.g., \( f(x) = h(g_1(x), \ldots, g_m(x)) \), where \( h, g_1, \ldots, g_m \) are partial recursive.
- Primitive recursion:
  \[
  f(x, 0) = h(x) \\
  f(x, \text{succ}(y)) = g(x, y, f(x, y))
  \]
  \( (h \) is partial recursive\( ) \)
  \( (g \) is partial recursive\( ) \).
- Unbounded search (minimization):
  \[
  f(x) = \min(y, g(x, y) = 0)
  \]
  \( (g \) is total partial recursive\( ) \).

This means that \( f(x) = y \) is the minimum \( y \) such that \( g(x, y) = 0 \), if such a \( y \) exists.
8. Turing Machines and Equivalent Models – The Church-Turing Thesis

Markov Algorithms
This model processes strings. An algorithm consists of a finite, ordered, sequence of productions of the form \( x \rightarrow y \), where \( x, y \in A^* \) for some alphabet \( A \). Any production can be suffixed with (halt) although this is not required.

Execution
Given an input string \( w \in A^* \), perform the following execution step repeatedly:

Scan the productions \( x \rightarrow y \) sequentially to see whether \( x \) occurs as a substring of \( w \). If so, replace the leftmost occurrence of \( x \) in \( w \) by \( y \) and reset \( w \) to this string. Otherwise halt. If the \( x \rightarrow y \) is labeled with (halt), then halt.

Markov Algorithm model has the Same power as the TM model.
8. Turing Machines and Equivalent Models – The Church-Turing Thesis

Markov Algorithms

**Example.** The Markov algorithm consisting of the single production \( a \to \Lambda \) will delete all \( a \)'s from any string.

**Example.** A more instructive Markov algorithm to delete all \( a \)'s from any string over \( \{a, b\} \) can be written as the following sequence of productions (\( # \) is an extra symbol).

\[
\begin{align*}
1. & \ #a \to \ # \\
2. & \ #b \to b# \\
3. & \ # \to \Lambda \quad \text{(halt)} \\
4. & \ \Lambda \to \#.
\end{align*}
\]

An example trace:

\[
\begin{array}{c}
\text{abab} \\
\text{#abab} \\
\text{#bab} \\
\text{b#ab} \\
\text{b#b} \\
\text{bb#} \\
\text{bb}
\end{array}
\begin{array}{c}
\text{(input)} \\
\text{(by 4)} \\
\text{(by 1)} \\
\text{(by 2)} \\
\text{(by 1)} \\
\text{(by 2)} \\
\text{(by 3, halt)}.
\end{array}
\]
Example. Find a Markov algorithm to delete the rightmost $b$ from strings over \{a, b\}.

Solution:

1. $\#a \rightarrow a\#$
2. $\#b \rightarrow b\#$
3. $\# \rightarrow @$
4. $a@ \rightarrow @a$
5. $b@ \rightarrow \Lambda \ (\text{halt})$
6. $@ \rightarrow \Lambda \ (\text{halt})$
7. $\Lambda \rightarrow \#.$
End of Turing Machines II