Non-deterministic Turning Machine

Definition of Computation:

- The computation of a non-deterministic Turing machine is a tree whose branches correspond to different possibilities for the machine. If some branch of the computation leads to the accept state, the machine accepts its input.

- The computation of a non-deterministic Turing machine is defined in the expected way:

\[ (\{q_f, q_T\} \times \Gamma) \leftarrow q_f \times \Gamma \ : \delta \]

Overview

- Notation
- Algorithms
- Enumerators
- Non-determinism
Proof Continued

Equivalence

For nondeterministic TM $N$, make equivalent deterministic TM $D$.

Construction:
- Simulate all possible branches of $N$'s nondeterminism.
- Accept if any one of the computation branches of $N$ accepts.

What do we mean by equivalent?

- View nondeterminism as a bunch of choice points.
- For each transition, pick an alternative from $\delta(q, a) \subseteq P(\Sigma \times \Gamma \times \{L, R\})$.
- Thus simulate them all in parallel (breadth-first), rather than depth-first.
- Must simulate them all in parallel (breadth-first) to ensure they are equivalent.
- View nondeterminism as a bunch of choice points.

Let $b$ be the maximum number of choices.
- Given a sequence, we can increment it base $b$ to get the next set of choices.
- So a computational path is just a number in base $b$.
- We can interpret a sequence of choices using $\{1, 2, \ldots, b\}^*$.
- View nondeterminism as a bunch of choice points.

While

address tape (sequence of choice points we are currently trying)
- simulation tape used to simulate a branch of computation
- input tape (never altered)
- use 3 tapes:
  - Copy input tape to simulation tape
  - Run computation according to choices on address tape
  + Running computation from start to one step after what we previously did
  + If computation goes into an accept state, then accept
  - Otherwise, a choice point is invalid, or we finish the sequence
  - If computation gets into an accept state, then accept
  + Running computation from start to one step after what we previously did
  - Copy input tape to simulation tape

Copy input tape to simulation tape
- Run computation according to choices on address tape
- While
- Copy input tape to simulation tape
- Run computation according to choices on address tape
- Address tape (sequence of choice points we are currently trying)
- Simulation tape used to simulate a branch of computation
- Input tape (never altered)
Can we define a notion of halting for a nondeterministic TM?

- Definition: Halts (accepts or rejects) on every computation branch. Halts if all branches halt.
- If we halt by accepting on any path, we accept. If we halt by rejecting on all paths, we reject.
- Can we define a notion of halting for a nondeterministic TM?

Nondeterminism and Decidability

Corollary 3.18: A language is Turing-recognizable if and only if some nondeterministic TM recognizes it.
Overview

- Notation
- Algorithms
- Enumerators
- Nondeterminism

Equivalence for Deciders

Corollary 3.19: A language is Turing-decidable if a nondeterministic TM decides it.

Previous construction for recognizing a language:

Equivalence for Deciders
Can we enumerate all strings in $\Sigma^*$?

Can we enumerate all numbers, writing them out in base 2?

Can we enumerate all numbers, writing them out in base $b$?

**Recursively Enumerable Languages**

- Languages that can be recognized by an Enumerator are called recursively enumerable languages.

- Enumerator TMs enumerate the members of its language:
  - Don't worry about repetitions, or order.
  - Can output an infinite list of strings.
  - Can output strings to the printer.
  - Have a work tape and a printer.

**Enumerators**
We re basically running \( V \) in parallel on all inputs + So, \( A \) will print it out as often as necessary \( \forall m \) \( \forall n \) \( \forall i \) \( \forall j \) \( \forall k \)
• If \( \forall j \) accepts a particular \( w \), it will do it in say \( f \) steps.
  's computation accepts, print our \( s \)
  Run \( \forall j \) for \( f \) steps on input \( s \)
  For \( i = 1, 1, 2, \ldots \), say \( \forall j \),
  's output is \( A \)
'\( A \) is Turing enumerable
• Construct \( E \) as follows:

We can make a list of them as \( A \) is enumerable
- Say \( \forall j \), \( \forall w \), is a list of all possible strings in \( A \)
- We have a deterministic TM \( A \) that recognizes \( L \)

Proof (Turing-recognizable language can be enumerated)

\[ \text{Equivalence} \]

Equivalence

\[ \text{Theorem 3.21:} \ \text{A language is Turing-recognizable iff it some} \]

\[ \text{Continued} \]

\[ \text{Continued} \]

\[ \text{Continued} \]
Overview

Nondeterminism

Algorithms

Enumerators

Nondeterminism

Variants

• Pretty well any model with unrestricted access to unlimited memory has the same power

- Turing-recognizable and Turing-decidable
- So, many variants, but all still same class of languages
  - One reasonable requirement is only perform a finite amount of work in a single step
  - Any two computational models that satisfy certain reasonable requirements can simulate one another and hence are equivalent
  - Build an interpreter in one language for another one
  - Compile one language into another
  - Similar to how you can
  - Pretty well any model with unrestricted access to unlimited
Church-Turing Thesis

**Church-Turing Thesis**

- Intuitive notion of algorithm is the same as what can be done with a Turing machine or with lambda calculus.

- Hilbert's tenth problem was to devise an algorithm that tests whether a polynomial has an integer root.
  - Actually, he said, a process according to which it can be determined by a finite number of operations.

- What is even an algorithm?
- But how do we go about proving this? That no such algorithm exists?
- As it turns out, there is no such algorithm.

**Motivation**

- Why is even an algorithm?
- What do we mean by an algorithm?
- How can we prove that there is no such algorithm?
- Hilbert's tenth problem was to devise an algorithm that tests if a given polynomial has an integer root.
- The number of operations is finite, and it is a process according to which it can be determined by a finite number of operations.

- A problem is a process problem if all variables have integer values.
- A tool is an assignment of values to its variables so that the value is 0.
- A tool is an assignment of values to its variables so that the value is 0.

- Integral roots of a polynomial are:

- Integral roots of a polynomial are:
Describing TM

• Formal description
  - States and transitions

• Implementation level
  - How the head moves, how it stores data on the tape

• High-level description
  - Describe the algorithm

• Abstract away from how machines works. Only use this when comfortable with how TMs work.

Overview

• Nondeterminism
• Algorithms
• Enumerators
• Notation
Input

- Encode a TM so another TM can use it as input.

- Encode a DFA so a TM can use it as input.

- Encode a graph G so a TM can use it as input.

- If we want to encode two objects O₁ and O₂ as input:\n  ⟨⟨O₁, O₂⟩⟩

- For object O₁, we will refer to its string encoding as a string.

If you want to give it some object, you must encode it as a string.

TM's always take a string as input.