Computer Science 351

Simulations and Simple Turing Machine Variants

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #13

Goals for Today

- Review, and expand on, information about ***simulations***
- Application: Prove that every regular language is decidable
- Application: Consider several "variants" of Turing machines found in the computing literature

Simulations

Recall that a ***simulation*** is something that can be presented to relate the power of two models of computation. In order to show that the machines described by a ***second*** model of computation are (in some sense) at least as powerful or efficient as the machines described by a ***first*** model of computation, we generally do the following:

(a) Consider an arbitrary machine M, of the type described by the ***first*** model of computation.

(b) Use M to define another machine $\widehat{M}$, of the type described by the ***second*** model of computation.

(c) Prove that $\widehat{M}$ solves the same problem as M.

Simulations

Expanding the Middle Step

- ***Question:*** How can a "configuration" of the first machine, M, be represented when a machine $\widehat{M}$, of the second type, is being used?
- Stages in $\widehat{M}$'s computation will frequently include:
  (a) Initialization
  (b) Step-By-Step Simulation
  (c) Finishing Up

Application: Every Regular Language is Decidable

Claim #1: Every regular language is decidable.

How To Prove This:

- Let $L \subseteq \Sigma^\star$ — so there exists a ***deterministic finite automaton***

$$M = (Q, \Sigma, \delta, q_0, F)$$

  such that $L = L(M)$.
- Describe a Turing machine $\widehat{M}$ — with input alphabet Σ — that ***simulates*** M — and prove, furthermore, that $\widehat{M}$ ***decides*** L.
- The claim follows because L was an "arbitrarily chosen" regular language.

Simulating a DFA: Representing a "Configuration"

Consider an execution of a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ on a string

$$\omega = \alpha_1 \alpha_2 \ldots \alpha_n \in \Sigma^\star$$

with length $n \in \mathbb{N}$.

During M's execution on input ω we must remember:

- What *prefix* of ω has already been processed/what *suffix* of ω is left to process?
- What *state* is M currently in?

Simulating a DFA: Representing a "Configuration"

Let

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}})$$

that will simulate M. The following is true as steps of M are simulated:

- $\widehat{M}$'s tape should store the input string ω (with copies of "$\sqcup$" to the right).
- For $0 \leq i \leq n$, if symbols in the prefix $\alpha_1\alpha_2 \ldots \alpha_i$ of ω have already been processed, so that symbols in the suffix $\alpha_{i+1}\alpha_{i+2} \ldots \alpha_n$ remain to be processed, then $\widehat{M}$'s tape head should point to the $i + 1^{\text{st}}$ cell, which stores α_{i+1} — or "$\sqcup$", if $i = n$.
- So $\widehat{\Gamma} = \Sigma \cup \{\sqcup\}$.

Simulating a DFA: Representing a "Configuration"

$\widehat{M}$'s "finite control" should be used to remember M's current state:

- Choose (or change) names of states so that $\widehat{q}_{\text{accept}} \notin Q$ and $\widehat{q}_{\text{reject}} \notin Q$.
- Set
$$\widehat{Q} = Q \cup \{\widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}}\}.$$
- ***First Goal:*** For $0 \leq i \leq n$, if
$$\delta^\star(q_0, \alpha_1\alpha_2 \ldots \alpha_i) = q \in Q$$
then (in $\widehat{M}$)
$$q_0\omega \vdash^\star \alpha_1\alpha_2 \ldots \alpha_i q \alpha_{i+1}\alpha_{i+2} \ldots \alpha_n.$$

Simulating a DFA: Initialization

Choose $\widehat{M}$'s start state, $\widehat{q}_0$, to be M's start state, q_0.

- Then no "Initialization" phase is needed for this simulation, since $\widehat{M}$ is in configuration

$$\widehat{q}_0\omega = q_0\omega$$

before any symbols of ω have been processed.

Simulating a DFA: Step-By-Step Simulation

- Let $q \in Q$ and let $\sigma \in \Sigma$. Define (part of) $\widehat{M}$'s transition function, $\widehat{\delta}$, by setting

$$\widehat{\delta}(q, \sigma) = (\delta(q, \sigma), \sigma, \mathtt{R}). \tag{1}$$

- It is possible to prove — using this — that the ***First Goal*** is now satisfied.

Simulating a DFA: Finishing Up

- ***Second Goal:*** Ensure that, for every string $\omega \in \Sigma^\star$,

$$q_0\omega \vdash^\star \begin{cases} \omega \sqcup \widehat{q}_{\text{accept}} & \text{if } \omega \in L, \\ \omega \sqcup \widehat{q}_{\text{reject}} & \text{if } \omega \notin L. \end{cases}$$

- If this goal can be achieved then it will follow that $\widehat{M}$ ***decides*** L — establishing that L is a decidable language, as claimed.

Simulating a DFA: Finishing Up

- It follows by the *first* goal that, for $\omega \in \Sigma^\star$,

$$q_0\omega \vdash^\star \omega q \qquad \text{for } q = \delta^\star(q_0, \omega).$$

- For $q \in Q$, set

$$\widehat{\delta}(q, \sqcup) = \begin{cases} (\widehat{q}_{\text{accept}}, \sqcup, \mathrm{R}) & \text{if } q \in F, \\ (\widehat{q}_{\text{reject}}, \sqcup, \mathrm{R}) & \text{if } q \notin F. \end{cases} \tag{2}$$

Simulating a DFA: Proof of Correctness

Exercise: Suppose that $\widehat{M}$'s transition function, $\widehat{\delta} : Q \times \Gamma \to Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$ is as defined as shown at lines (1) and (2).

1. Prove that if $\mu, \nu \in \Sigma^\star$ and $q \in Q$ such that $\delta^\star(q_0, \mu) = q$, then (considering computations by $\widehat{M}$)

$$\widehat{q}_0 \mu\nu = q_0 \mu\nu \vdash^\star \mu q \nu.$$

2. Using the above result, prove, for all $\omega \in \Sigma^\star$, that

$$\widehat{q}_0 \omega \vdash^\star \begin{cases} \omega \sqcup \widehat{q}_{\text{accept}} & \text{if } \omega \in L, \\ \omega \sqcup \widehat{q}_{\text{reject}} & \text{if } \omega \notin L, \end{cases}$$

— so that $\widehat{M}$ decides the language L, as desired.

Application: Allowing the Tape Head Not to Move

A ***Turing machine whose tape head can stay*** is a variant of a Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}).$$

- Q, Σ, Γ, q_0, q_{accept} and q_{reject} are as for "regular" Turing machines.
- The transition function is a partial function

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}.$$

- If a transition

$$\delta(q, \sigma) = (r, \tau, \mathtt{S})$$

is applied, for $q, r \in Q$ and $\sigma, \tau \in \Gamma$, the state changes from q to r, the copy of σ seen on the tape is replaced with a copy of τ, and ***the tape head is not moved.***

An Easy Claim To Prove

Claim #2: Let $L \subseteq \Sigma^\star$.

(a) If L is recognizable then there exists a Turing machine, whose tape head can stay, with language L.

(b) If L is decidable then there exists a Turing machine, whose tape head can stay, that decides L.

Why This is True: Every "regular" Turing machine can be considered to be a Turing machine whose tape head can stay.

A Claim That is — Slightly — Harder to Prove

Claim #3: Let $L \subseteq \Sigma^\star$.

(a) If there exists a Turing machine whose tape head can stay, M, such that $L(M) = L$, then L is recognizable.

(b) If there exists a Turing machine whose tape head can stay, M, that decides L, then L is decidable.

How This is Proved: Simulate a Turing machine whose tape head can stay using a "regular" Turing machine.

Simulating a Turing Machine Whose Tape Head Can Stay: Representing a "Configuration"

Let

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

be a Turing machine whose tape head stay. Consider a Turing machine

$$\widehat{M} = (\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, q_0, q_{\text{accept}}, q_{\text{reject}})$$

which has the same tape alphabet and with a set of states $\widehat{Q}$ such that $Q \subseteq \widehat{Q}$ — and such that $\widehat{M}$ has the same start state, accept state and reject state as M.

- Each configuration of M can be represented by the same configuration of $\widehat{M}$ — with the same state, tape contents, and location of the tape head.

Simulating a Turing Machine Whose Tape Head Can Stay: Initialization

- Since M and $\widehat{M}$ have the same initial configuration, for an input string $\omega \in \Sigma^\star$, no "Initialization" stage is required.

Simulating a Turing Machine Whose Tape Head Can Stay: Step-By-Step Simulation

- Transitions of M that move the tape head can be used for $\widehat{M}$ without change: If $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and $\sigma \in \Gamma$ such that

$$\delta(q, \sigma) = (r, \tau, D)$$

for $r \in Q$, $\tau \in \Gamma$, and $D \in \{\mathtt{L}, \mathtt{R}\}$, then

$$\widehat{\delta}(q, \sigma) = (r, \tau, D)$$

as well.

Simulating a Turing Machine Whose Tape Head Can Stay: Step-By-Step Simulation

- Transitions of M that do not move the tape head are replaced by a *pair* of transitions, for $\widehat{M}$:
- Let $\dot{Q} = \{\dot{q} \mid q \in Q\}$ — so that $Q \cap \dot{Q} = \emptyset$, and let

$$\widehat{Q} = Q \cup \dot{Q}$$

- For $q \in Q$, the new state $\dot{q}$ will be used to remember that one of M's transitions to state q is "in progress" but not completed.

Simulating a Turing Machine Whose Tape Head Can Stay: Step-By-Step Simulation

- Suppose $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and $\sigma \in \Gamma$ such that

$$\delta(q, \sigma) = (r, \tau, \mathtt{S})$$

for a state $r \in Q$ and symbol $\tau \in \Gamma$. Then set

$$\widehat{\delta}(q, \sigma) = (\dot{r}, \tau, \mathtt{R})$$

— and set $\widehat{\delta}(\dot{s}, \alpha)$ to be $(s, \alpha, \mathtt{L})$ for *every* state $w \in Q$ and for *every* symbol $\alpha \in \Gamma$.

Simulating a Turing Machine Whose Tape Head Can Stay: Finishing Up

- No phase to “Finish Things Up” is needed for this simulation.

Simulating a Turing Machine Whose Tape Head Can Stay: Proof of Correctness

- Using the above definitions of $\widehat{Q}$ and $\widehat{\delta}$ one can prove the following: For every input string $\omega \in \Sigma^\star$, for all strings $\mu, \nu \in \Gamma^\star$, and for every state $q \in Q$,

  $$q_0\omega \vdash^\star \mu q \nu \qquad \text{during an execution of } M$$

  if and only if

  $$q_0\omega \vdash^\star \mu q \nu \qquad \text{during an execution of } \widehat{M}.$$

Simulating a Turing Machine Whose Tape Head Can Stay: Proof of Correctness

- It follows that, for every input string $\omega \in \Sigma^\star$,

$$M \text{ accepts } \omega \iff \widehat{M} \text{ accepts } \omega,$$

$$M \text{ rejects } \omega \iff \widehat{M} \text{ rejects } \omega,$$

and

$$M \text{ loops on } \omega \iff \widehat{M} \text{ loops on } \omega.$$

- Thus $L(\widehat{M}) = L(M)$ and $\widehat{M}$ *decides* a language $L \subseteq \Sigma^\star$ if and only if M decides L — establishing Claim #3.

Application: Handling "Falling Off the Tape" Differently

A ***Turing machine with left failure*** is a variant of a Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

where Q, Σ, Γ, δ, q_0, q_{accept} and q_{reject} are all as before, but the transition function, δ, is applied differently: Whenever a transition

$$\delta(q, \sigma) = (r, \tau, \mathtt{L})$$

is to be applied, for $q, r \in Q$ and $\sigma, \tau \in \Gamma$, and the tape head is already at the leftmost cell then — instead of continuing without moving the location of the tape head — the computation halts and the input string is ***rejected***.

Simulating a Turing Machine That "Falls Off the Tape" Differently: A Claim in One Direction

Claim #4: Let $L \subseteq \Sigma^\star$.

(a) If L is recognizable then there exists a Turing machine, with left failure, with language L.

(b) If L is decidable then there exists a Turing machine, with left failure, that decides L.

Simulating a Turing Machine That "Falls Off the Tape" Differently: A Claim in One Direction

How a Proof Starts:

- Let

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

be a standard Turing machine (being simulated) and let

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}})$$

be a Turing machine with left failure, that will simulate executions of M.

Simulating a Turing Machine That "Falls Off the Tape" Differently: Representing a "Configuration"

- Shift everything on the tape over by one symbol, using a special marker at the left end of the tape — so that the *simulating* machine, $\widehat{M}$ can know when the *simulated* machine is trying to move past the left end of the tape.
- Let "#" be a symbol such that $\# \notin \Gamma$ and let

$$\widehat{\Gamma} = \Gamma \cup \{\#\}.$$

- For strings $\mu, \nu \in \Gamma^\star$ and a state $q \in Q$ a configuration "$\mu q \nu$", of M, will be represented using the configuration

$$\#\mu q \nu$$

of $\widehat{M}$.

Simulating a Turing Machine That "Falls Off the Tape" Differently: Initialization

- Let

$$\omega = \alpha_1 \alpha_2 \ldots \alpha_n$$

be a string in $\Sigma^\star$ with length $n \in \mathbb{N}$. During the *Initialization* stage, $\widehat{M}$'s initial configuration for ω

$$\widehat{q}_0 \alpha_1 \alpha_2 \ldots \alpha_n$$

must be used to derive the representation of M's initial configuration for ω

$$\#q_0 \alpha_1 \alpha_2 \ldots \alpha_n$$

Simulating a Turing Machine That "Falls Off the Tape" Differently: Initialization

Exercise:

- Give an "initial" part of $\widehat{M}$ which carries out this initialization stage.

Note: The details of this simulation, and a sketch of a proof of its correctness, are available in a supplemental document.

Simulating a Turing Machine That "Falls Off the Tape" Differently: Step-By-Step Simulation

- Let $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$, let $\sigma \in \Gamma$, and suppose that

$$\delta(q, \sigma) = (r, \tau, D)$$

for a state $r \in Q$, symbol $\tau \in \Gamma$, and direction $D \in \{\mathtt{L}, \mathtt{R}\}$. Let

$$\widehat{\delta}(q, \sigma) = (r, \tau, D)$$

as well.

Simulating a Turing Machine That "Falls Off the Tape" Differently: Step-By-Step Simulation

Two Things Might Happen When this Transition is Used:

1. The symbol "#" is *not* visible on $\widehat{M}$'s tape — so *M did not* try to move its tape head off the left end of the tape — and there is nothing more to do, to simulate this step.
2. The symbol "#" *is* visible on $\widehat{M}$'s tape — so *M did* try to move its tape head off the left end of the tape.
   Add the transition
   $$\widehat{\delta}(r, \#) = (r, \#, \mathrm{R})$$
   for every state $r \in Q$ to ensure that — after one more step — $\widehat{M}$ is in a configuration representing the configuration that *M* is in, once again.

Simulating a Turing Machine That "Falls Off the Tape" Differently: Finishing Up

- Adding transitions

$$\widehat{\delta}(q_{\text{accept}}, \sigma) = (\widehat{q}_{\text{accept}}, \sigma, \mathrm{R})$$

and

$$\widehat{\delta}(q_{\text{reject}}, \sigma) = (\widehat{q}_{\text{reject}}, \sigma, \mathrm{R})$$

— for all $\sigma \in \Gamma$ — ensures that $\widehat{\delta}(q, \sigma)$ is defined whenever

$$q \in \widehat{Q} \setminus \{\widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}}\}$$

and $\sigma \in \widehat{\Gamma}$ — completing the definition of $\widehat{M}$.

Simulating a Turing Machine That "Falls Off the Tape" Differently: Proof of Correctness

- It is possible to prove that, during an execution on an input string $\omega \in \Sigma^\star$ $\widehat{M}$ never carries out a transition indicating that the tape head should move ***left*** when the tape head is already at the leftmost cell of the tape.
- Using this, one can show that, for all $\omega \in \Sigma^\star$,

$$\widehat{M} \text{ accepts } \omega \iff M \text{ accepts } \omega,$$

$$\widehat{M} \text{ rejects } \omega \iff M \text{ rejects } \omega,$$

and

$$\widehat{M} \text{ loops on } \omega \iff M \text{ loops on } \omega,$$

as needed to establish the claim.

Simulating a Standard Turing Machine: A Claim in The Other Direction

Claim #5: Let $L \subseteq \Sigma^\star$.

(a) If there exists a Turing machine with left failure, whose language is L, then L is recognizable.

(b) If there exists a Turing machine with left failure, that decides L, then L is decidable.

Exercise: Modify the proof of Claim #4 to prove Claim #5.

Other Turing Machine Variants

- Additional Turing machine variants — which you might find in the computing literature, or which can simplify Turing machine design and analysis — will be considered in the lecture presentation for this lecture, and in the corresponding tutorial exercise.