CS 7220 – Computational Complexity and Algorithm Analysis

Spring 2016

Section 7: Computability – Part III
Decidability

Pascal Hitzler
Data Semantics Laboratory
Wright State University, Dayton, OH
http://www.pascal-Hitzler.de

TOC: Decidability

Chapter 11 of [Sudkamp 2006].

1. Decision Problems
2. Problem Reduction
3. The Church-Turing Thesis
4. A Universal Turing Machine

Decision problems

A decision problem is a set of related questions each of which has a yes or no answer.

- Is the non-negative integer n a square number?
  [One question for each n.]
- Given a fixed language L, is the string x in L?
  [One question for each n.]
- Does the Turing Machine M halt on input x?
  [One question for each pair (M,x).]

Decision problems

A decision problem is a set of related questions each of which has a yes or no answer.

A decision problem P is *decidable* if there is an algorithm that, for every question p∈P, terminates and determines the appropriate answer.

A decision problem P is *semi-decidable* (or, *partially solvable*) if there is an algorithm that, for every question p∈P for which the answer is “yes”, terminates and determines the appropriate answer.

However, we have not defined the notion of *algorithm*. – and we’re not going to, as such.

Effective procedure

An *effective procedure* for a decision problem satisfies the following properties

- Complete: It produces the correct answer for each problem instance. [Note: this implies termination of the procedure.]
- Mechanistic: It consists of a finite sequence of instructions, each of which can be carried out without requiring insight, ingenuity, or guesswork.
- Deterministic: When presented with identical input, it always performs the same computation.

Note: If a decision problem P has an effective procedure, then P is decidable.

TMs and decision problems

- In the TM paradigm, effective procedures would be specified by TMs which always terminate.

- If we drop the completeness requirement (i.e., termination), then TMs are an appropriate paradigm. It is unproven (unprovable?) if TMs are an *exhaustive* paradigm for algorithmization.

- To invoke TMs for decision problems, the latter must be represented as input to TMs – and we will focus on language accepting TMs.

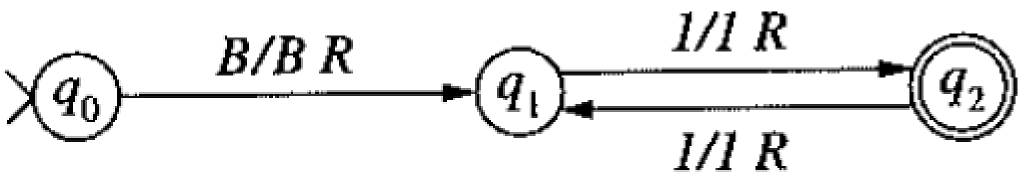
  I.e., we require a representation of the decision problem as a language acceptance problem.

Different representations

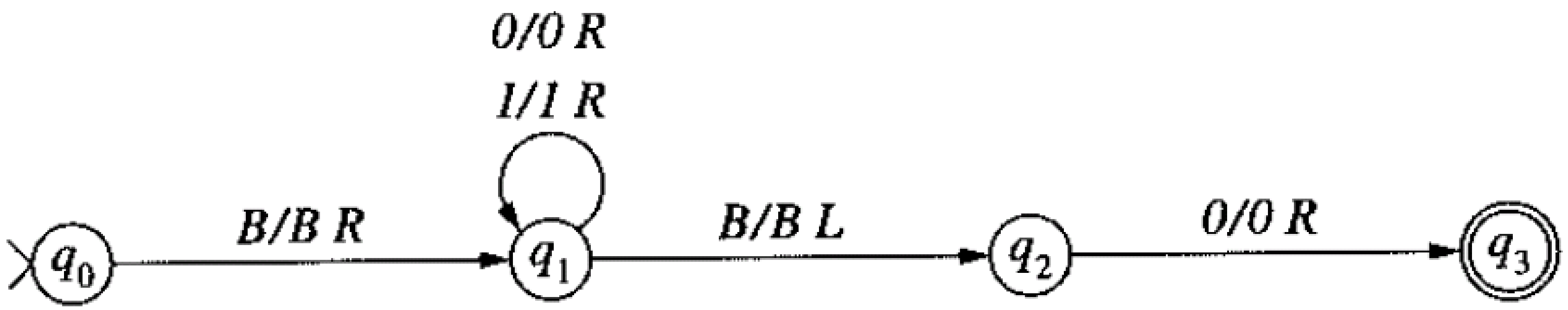
- **Different representations of the same decision problem result in different TMs for solving them.**

- **E.g. – determining whether a non-negative integer is even.**

  - **Unary representation of numbers:**



  - **Binary representation of numbers:**



The membership problem

- The membership problem for (a language) L:

  "Is the input string x in L?"

- The membership problem for L is
  - decidable if and only if L is recursive.
    [We say "L is decidable" in this case.]
  - semi-decidable if and only if L is recursively enumerable.
    [We say "L is semi-decidable" in this case.]

A language or decision problem is *undecidable* if it is not decidable.

A note on nondeterminism

The membership problem for L is decidable if and only if there exists a nondeterministic TM M with L(M)=L, such that all computations of M terminate.

Why does this hold?

Because we have shown earlier that in this case we can construct an always terminating deterministic TM N with L(N)=L.

TOC: Decidability

Chapter 11 of [Sudkamp 2006].

1. **Decision Problems**
2. **Problem Reduction**
3. **The Church-Turing Thesis**
4. **A Universal Turing Machine**

Reduction

Definition 4.1

Let L be a language over Σ_1 and Q be a language over Σ_2.

L is (*many-to-one*) *reducible* to Q
if there is a Turing computable function r: $\Sigma_1^* \to \Sigma_2^*$
such that w$\in$ L if, and only if, r(w)$\in$ Q.

Note: If L is reducible to Q, then

- **if Q is decidable, so is L.**
- **if Q is semi-decidable, so is L.**
- **if L is undecidable, so is Q.**

Reducibility and Decidability

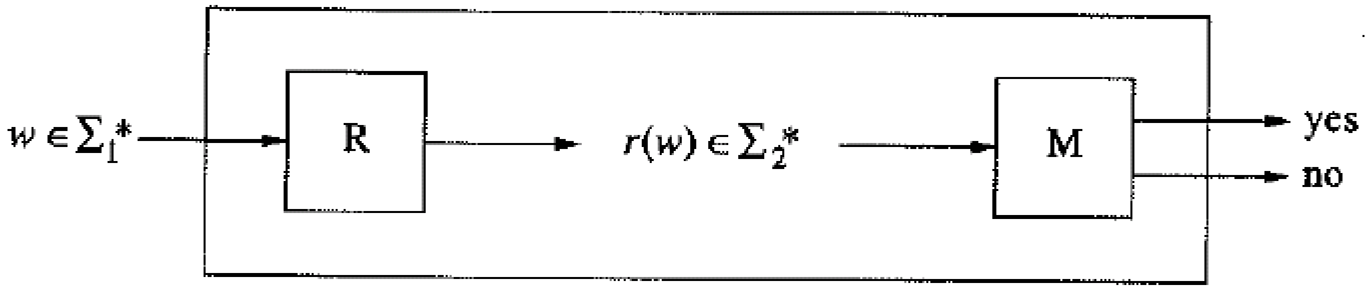
If Q is decidable, and L is reducible to Q, then L is decidable.

Proof: Since Q is decidable, there is an always terminating TM M with L(M)=Q.

Since L is reducible to Q, there is a TM R computing a function r s.t. w∈L iff r(w)∈Q.

The TM N resulting from the composition of R with M thus computes, for each w, whether or not w\in L, i.e., L(N)=L.

N always terminates since R terminates (for valid input) and M always terminates.



Exercise C10 [hand-in]

Prove the following:

If Q is semi-decidable, and L is reducible to Q, then L is semi-decidable.

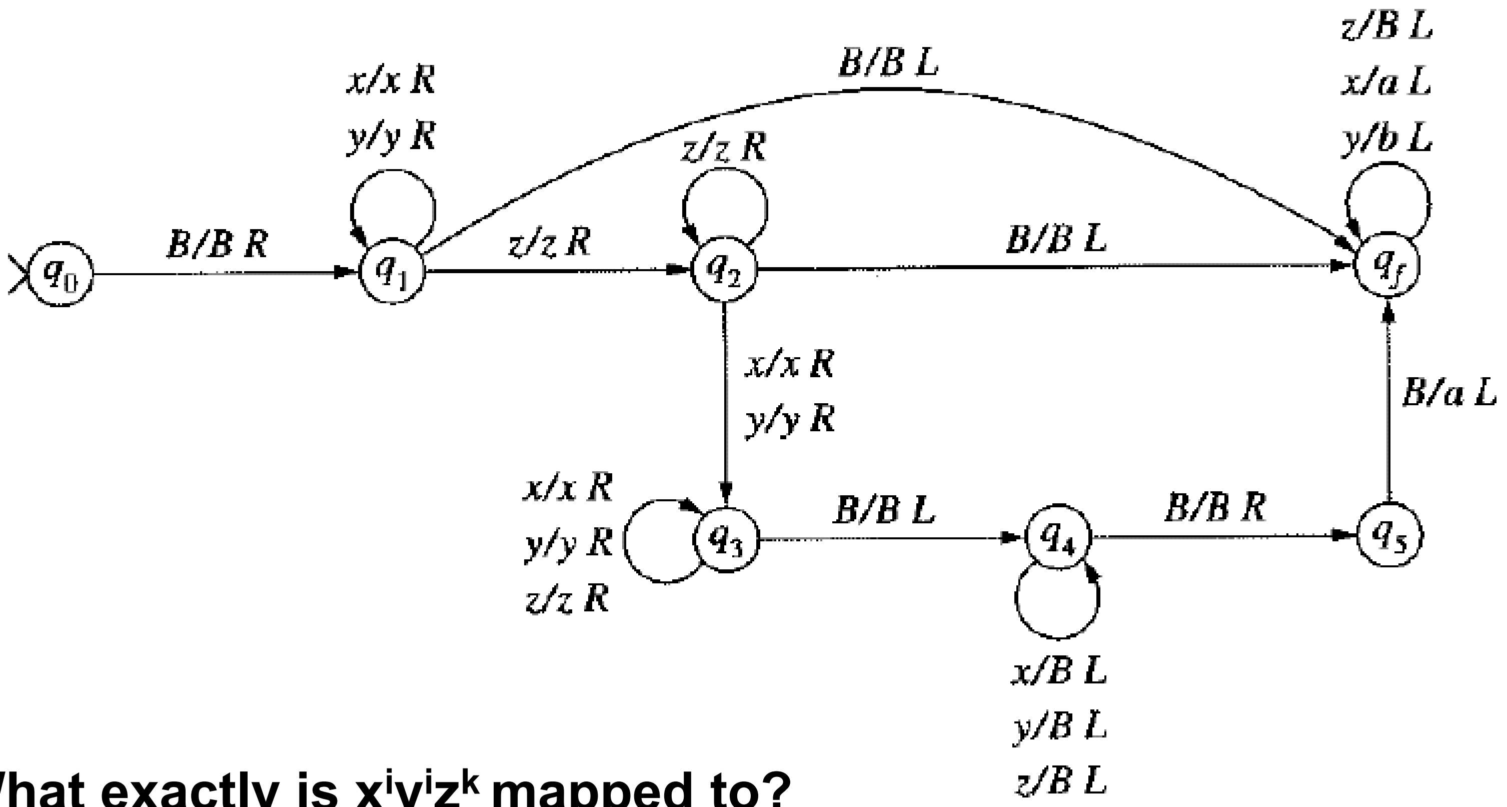
Exercise C11

Prove the following:

If L is undecidable and reducible to Q, then Q is undecidable.

Reduction example – Example 4.2

TM reducing $L = \{x^i y^i z^k | i \geq 0, k \geq 0\}$ to $Q = \{a^i b^i | i \geq 0\}$



- What exactly is $x^i y^i z^k$ mapped to?
- What are the other strings mapped to?
- Why is this sufficient?

Exercise C12

Construct a TM which reduces $\{a^ib^ia^i | i>0\}$ to $\{c^id^i | i>0\}$.

Example 4.3

We show that L = {uu | u=$a^ib^ic^i$ for some i≥0} is decidable.

Proof:

From Example 1.7 we know that Q = {$a^ib^ic^i$ | i ≥ 0} is decidable.

It thus suffices to reduce L to Q. We do this as follows:

From Example 1.12 we know how to check if an input string w is of the form uu (for some string u). Hence,

1. if w ≠ uu (for some u), then erase the tape and output a single a.
2. if w = uu (for some u), then the copy and the second u in the input string are erased, leaving u in the input position.

TOC: Decidability

Chapter 11 of [Sudkamp 2006].

1. Decision Problems
2. Problem Reduction
3. The Church-Turing Thesis
4. A Universal Turing Machine

The Church-Turing Thesis (CTT)

Intuitively: “Everything that is computable is computable by a TM.”

CTT for Decision Problems:

There is an effective procedure to solve a decision problem if, and only if, there is a TM that halts for all input strings and solves the problem.

CTT for Recognition Problems:

A decision problem P is semi-decidable if, and only if, there is a TM that accepts precisely the instances of P whose answer is yes.

CTT for Computable Functions:

A function f is effectively computable if, and only if, there is a TM that computes f.

Notes on the CTT

- It’s not a mathematical theorem, and has not been (cannot be) formally proven.
- However, it could be disproved!
- We actually have already invoked the CTT frequently in this lecture, namely whenever we refrained from giving a TM in detail, but instead gave a high-level description of a TM – with the understanding that a TM could be explicitly constructed in these cases, if desired.

TOC: Decidability

Chapter 11 of [Sudkamp 2006].

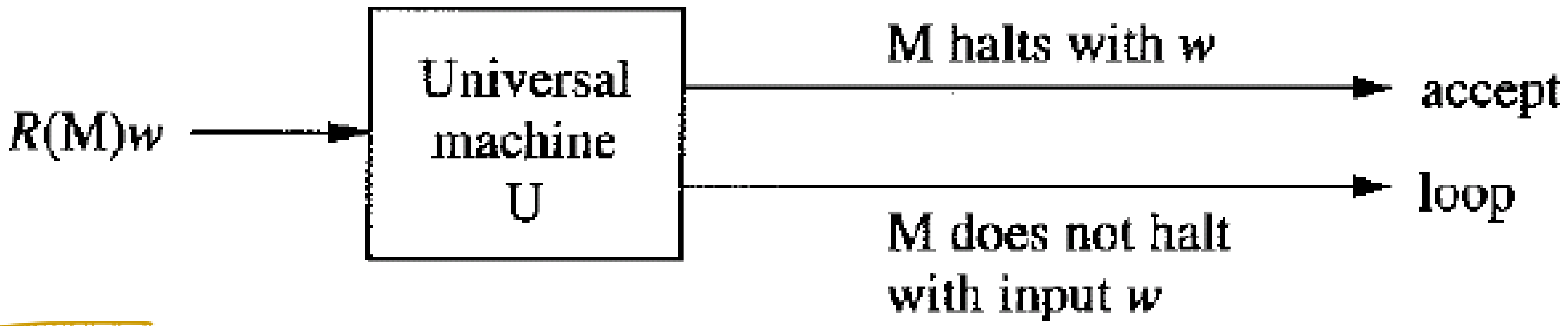
Universal TM

A universal TM simulates the computations of an arbitrary TM.
$\rightarrow$ think of the simulated TM as an algorithm specified by software, while the universal TM is the analogy to hardware.

We focus on TMs which accept by halting.

We need to be able *to give a TM as input* to a computation, i.e., we need to identify each TM M with a string R(M), the *representation* of M.

Schematically:



Representation

We consider only TMs with $\sum = \{0,1\}$ and ¡ = $\{0,1,B\}$.

States are called $\{q_0, \ldots, q_n\}$ with q_0 the start state.

A transition has the form $\delta(q_i,x) = [q_j,y,d]$
where $q_i,q_j \in Q$; $x,y \in \Gamma$, $d \in \{L,R\}$.

We encode $\delta(q_i,x) = [q_j,y,d]$ as

$$en(q_i)0en(x)0en(q_j)0en(y)0en(d)$$

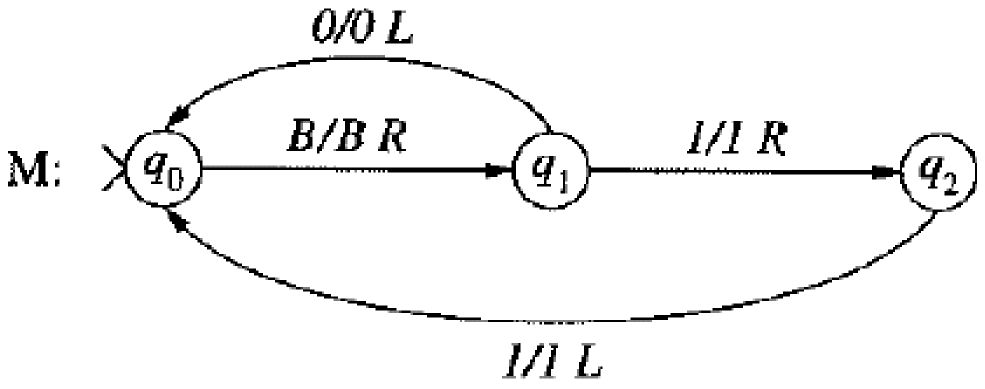
where	$en(0) = 1$	$en(1) = 11$	$en(B) = 111$	
	$en(q_0) = 1$	$en(q_1) = 11$	…	$en(q_n) = 1^{n+1}$
	$en(L) = 1$	$en(R) = 11$		

Two consecutive 0's separate transitions.

The beginning and end of the representation have three 0's.

Example 4.4

M does not terminate on strings starting with 0, and accepts all others.



Encoded transitions:

Transition	Encoding
$\delta(q_0, B) = [q_1, B, R]$	101110110111011
$\delta(q_1, 0) = [q_0, 0, L]$	1101010101
$\delta(q_1, 1) = [q_2, 1, R]$	110110111011011
$\delta(q_2, 1) = [q_0, 1, L]$	1110110101101

R(M):

000101110110111011001101010101001101101110110110011101101011101000

Determining whether a string is a TM

Given a string u∈{0,1}*.

Check if u consists of

- a prefix 000,
- followed by a finite sequence of encoded transitions separated by 00's, all transitions being of the specified form,
- followed by 000.

If yes, then u is the representation of a TM M.

M is deterministic if the combination of the state and input symbol in every encoded transition is distinct.

A universal 3-tape deterministic TM U

Computation begins with input on tape 1. Idea: If the input has the form R(M)w, then the computation of M is simulated on tape 3.

A computation of U does the following:

1. If input is not of the form R(M)w for a det. TM M, then U loops.
2. Write w on tape 3 from position 1. Reposition tape head on left.
3. "1", encoding state q_0, is written on tape 2.
4. Simulate transition on tape 3: Let x be the current symbol on tape 3 and q_i be the state encoded on tape 2.
   a. Scan tape 1 for an applicable transition. If there is none, halt and accept the input.
   b. If applicable transition is found on tape 1, then
      i. Put new state on tape 2
      ii. Put output symbol on tape 3
      iii. Move tape head on tape 3 as specified.
5. Computation continues with step 4

Theorem 4.5

The language

L_H = {R(M)w | M halts with input w}

is recursively enumerable.

Proof idea?

Proof

The language L_H = {R(M)w | M halts with input w} is recursively enumerable.

Proof:

The universal machine U accepts strings of the form R(M)w, where R(M) is the representation of a TM and M halts when run with input w.

For all other strings, the computation of U does not terminate.

Thus, $L(U)=L_H$.

Example 4.5

The decision problem

Halts on n'th Transition

Input: TM M, string w, integer n

Output: yes, if the computation of M with input w performs exactly n transitions before halting

no, otherwise

is decidable.

Proof idea?

Proof

Modify U to become U’, by adding a frouth tape to record the number of transitions in the simulation of M.

Represent a problem instance as R(M)w0001^{n+1}.

Computation of U’ on input u:

1. If input does not end with 0001^{n+1}, U’ halts rejecting the input.
2. 1^n is written on tape 4 (from pos. 1); 000^{n+1} is erased from the end of the string on tape 1; head of tape 4 returns to left.
3. If string on tape 1 is not of the form R(M)w, U’ halts rejecting the input.
4. w is copied to tape 3; the en(q_0) is written on tape 2.
5. Now simulate M, using tape 4 as transition counter (move right).
   – If M is found to terminate while a blank is read on tape 4, halt and accept.
   – If M terminates without a blank on tape 4, or if M would not terminate but there is a blank read on tape 4, halt and reject.