



Lemma J For all languages A, B, C the following hold

- i. $A \leq_m A$, (reflexive)
- ii. if $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$, (transitive)
- iii. if $A \leq_m B$ and B is Turing-recognizable, then so is A , and
- iv. if $A \leq_m B$ and B is decidable, then so is A

Proof.

- i. Let us choose $f(x) = x$ as the reduction.
- ii. Let f be reduction of A to B and g a reduction of B to C .
In other words, $f: A \leq_m B$, $g: B \leq_m C$.

We show that the composite function h , $h(x) = g(f(x))$ is a reduction $h: A \leq_m C$.



1. h is computable: Let M_f and M_g be the total Turing machines computing f and g . M_{REW} replaces all symbols to the right of the tape head with \square and moves the tape head to the beginning of the tape. The total machine depicted in the following figure computes function h .

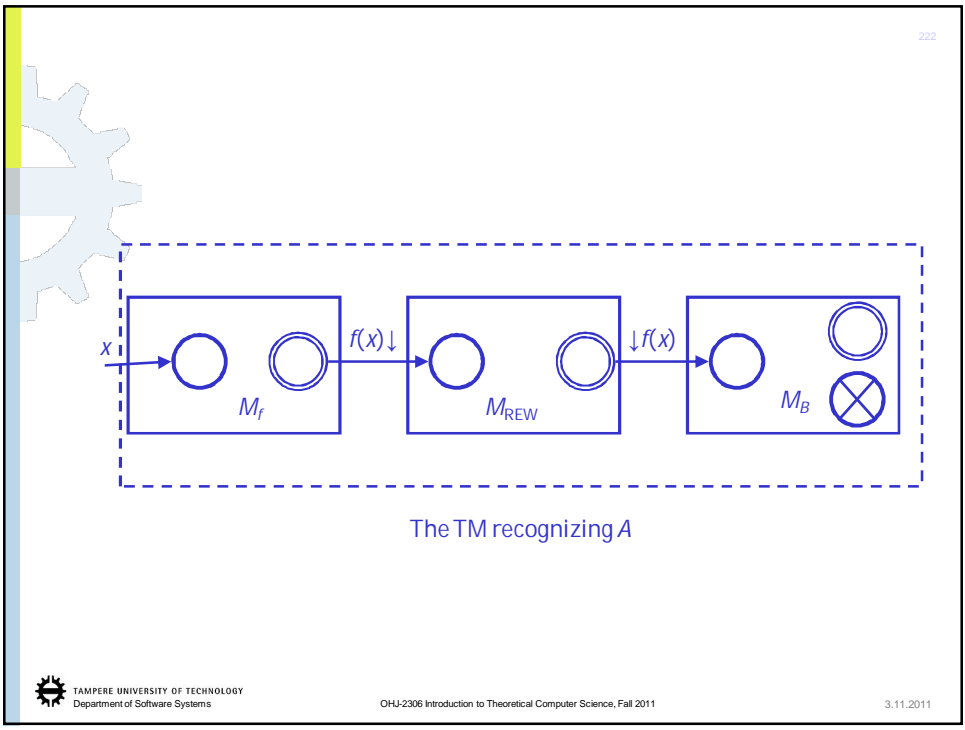
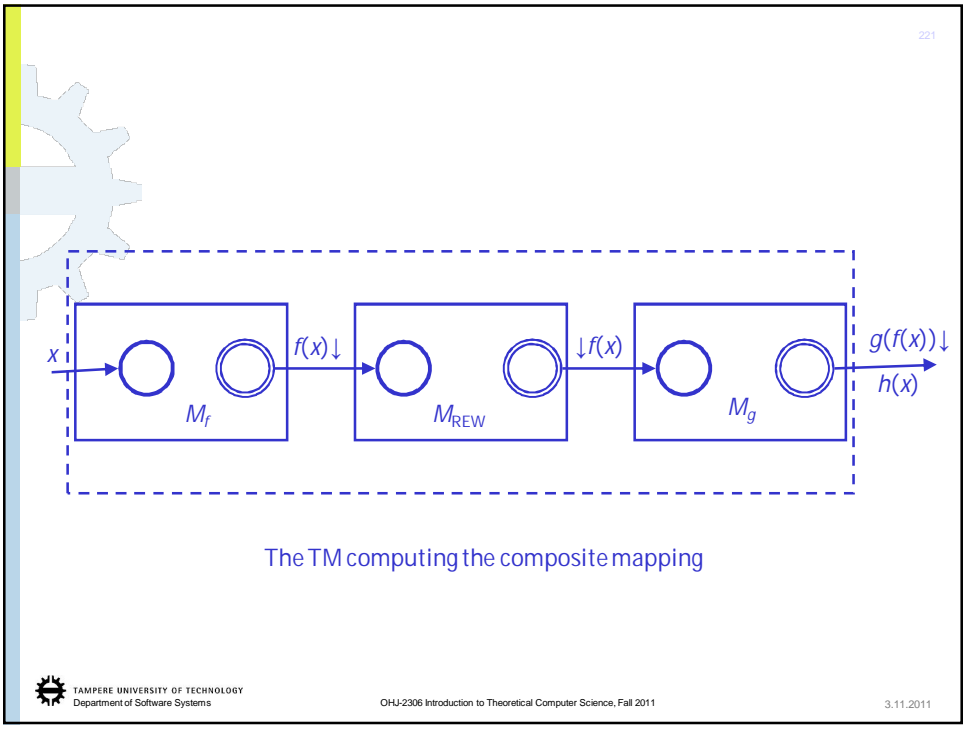
2. h is a reduction:

$$\begin{aligned} x \in A &\Leftrightarrow f(x) \in B \\ &\Leftrightarrow g(f(x)) = h(x) \in C, \end{aligned}$$

hence, $h: A \leq_m C$.

iii. (and iv.) Let $f: A \leq_m B$, M_B the recognizer of B and M_f the TM computing f . The TM depicted below recognizes language A and it is total whenever M_B is. \square





- We have already used the following consequence of Lemma J to prove undecidability.

Corollary 5.23 *If $A \leq_m B$ and A is undecidable, then B is undecidable.*

Let us call language $B \subseteq \{0, 1\}^*$ **TR-complete**, if

1. B is Turing-recognizable (TR), and
2. $A \leq_m B$ for all Turing-recognizable languages A

Theorem K *The universal language U is TR-complete.*

Proof. We know that U is Turing-recognizable. Let B be any Turing-recognizable language. Furthermore, let $B = L(M_B)$.

Now, B can be reduced to U with the function $f(x) = \langle M_B, x \rangle$, which is clearly computable, and for which it holds

$$x \in B = L(M_B) \Leftrightarrow f(x) = \langle M_B, x \rangle \in U. \quad \square$$



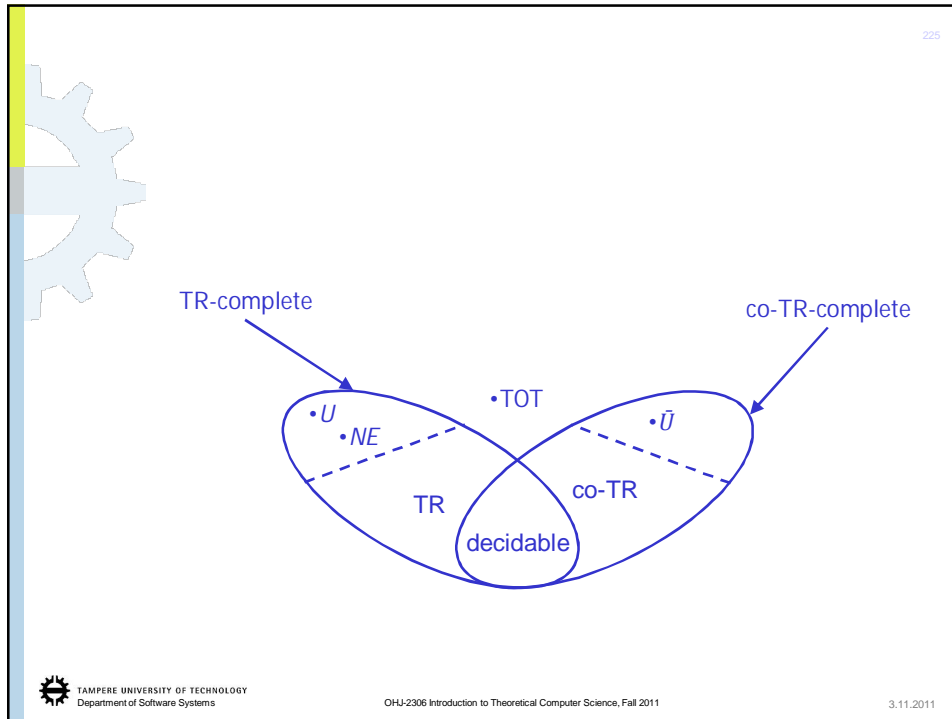
Theorem L *Let A be a TR-complete language, B TR, and $A \leq_m B$. Then also B is a TR-complete language.*

- All "natural" languages belonging to the difference of TR and decidable languages are TR-complete, but it contains also other languages
- The class of TR languages is not closed under complementation, thus it has the dual class

$$\text{co-TR} = \{ \bar{A} \mid A \in \text{TR} \}$$

- $\text{TR} \cap \text{co-TR} = \text{decidable languages}$ (by Theorem 4.22)
- $B \subseteq \{0, 1\}^*$ is co-TR-complete, if $B \in \text{co-TR}$ and $A \leq_m B$ for all $A \in \text{co-TR}$
- A language A is co-TR-complete, if and only if the language \bar{A} is TR-complete
- Language $\text{TOT} = \{ \langle M \rangle \mid M \text{ halts on all inputs} \}$ does not belong to either TR or co-TR





6. Advanced Topics in Computability

- The Church-Turing thesis gives a universally acceptable definition of *algorithm*
- Another fundamental concept in computer science is *information*
- No equally comprehensive definition of information is known
- Several definitions of information are used – depending upon the application
- In the following we present one way of defining information, using computability theory

6.4 A Definition of Information

- Consider the following two binary sequences
 - $A = 01$
 - $B = 111001011101000111010100001110100111010111$
- Intuitively, A contains little information: it is merely a repetition of the pattern 01 twenty times
- In contrast, B appears to contain more information
- We define the quantity of information contained in an object to be the size of that object's smallest representation or description
 - a precise and unambiguous characterization of the object so that we may recreate it from the description alone
- Sequence A contains little information: it has a small description
- Sequence B apparently contains more information because it seems to have no concise description



- We may always describe an object (e.g., string) by placing a copy of the object directly into the description
- This type of a description is never shorter than the object itself
- Neither does it tell us anything about its information quantity
- A description that is significantly shorter than the object implies that the information contained within it can be compressed into a small volume
- Therefore, the amount of information cannot be very large
- *The size of the shortest description determines the amount of information*



Minimal Length Descriptions

- We describe a binary string x with a Turing machine M and a binary input w to M
- The length of the description is the combined length of representing M and w : $\langle M, w \rangle = \langle M \rangle w$

Definition 6.23 Let x be a binary string. The **minimal description** of x , denoted $d(x)$, is the shortest string $\langle M, w \rangle$ where TM M on input w halts with x on its tape. If several such strings exist, select the lexicographically first among them.

The **Kolmogorov complexity** (also Kolmogorov-Chaitin c . and descriptive c .) of x is

$$K(x) = |d(x)|$$



- In other words, $K(x)$ is the length of the minimal description of x
- It is intended to capture our intuition for the amount of information in the string x
- The Kolmogorov complexity of a string is at most a fixed constant more than its length
- The constant is universal, it is not dependent on the string

Theorem 6.24 $\exists c \forall x: K(x) \leq |x| + c$

Proof Consider the following description of the string x .

Let M be a TM that halts as soon as it is started. This machine computes the identity function – its output is the same as input. A description of x is simply $\langle M \rangle x$. Letting c be the length of $\langle M \rangle$ completes the proof. \square



- Theorem 6.24 conforms to our intuition: the information contained by a string cannot be (substantially) more than its length
- Similarly, the information contained by the string xx is not significantly more than the information contained by x

Theorem 6.25 $\exists c \forall x: K(xx) \leq K(x) + c$

Proof Consider the following Turing machine, which expects an input of the form $\langle N, w \rangle$, where N is a TM and w is an input for it.

$M =$ "On input $\langle N, w \rangle$, where N is a TM and w is a string:

1. Run N on w until it halts and produces an output string s .
2. Output the string ss ."

A description of xx is $\langle M \rangle d(x)$. Recall that $d(x)$ is the minimal description of x . The length of this description is $|\langle M \rangle| + |d(x)|$, which is $c + K(x)$, where c is the length of $\langle M \rangle$. \square



- By the above results we might expect the Kolmogorov complexity of the concatenation xy of two strings x and y to be at most the sum of their individual complexities (plus a fixed constant)
- That bound, however, cannot be reached
- The cost of combining two descriptions leads to a greater bound:

Theorem 6.26 $\exists c \forall x, y: K(xy) \leq 2K(x) + K(y) + c$

Proof We construct a TM M that breaks its input into two separate descriptions. The bits of the first description $d(x)$ are doubled and terminated with string 01 before the second description $d(y)$ appears.

Once both descriptions have been obtained, they are run to obtain the strings x and y and the output xy is produced.

The length of this description of xy is clearly twice the complexity of x plus the complexity of y plus a fixed constant describing M . \square





- A more efficient method of indicating the separation between the two descriptions improves this theorem somewhat
- Instead of doubling the bits of $d(x)$, we may prepend the length of $d(x)$ as a binary integer that has been doubled to differentiate it from $d(x)$
- The description still contains enough information to decode it into the two descriptions of x and y
- The description now has length at most

$$2\log_2 K(x) + K(x) + K(y) + c$$



Optimality of the Definition

- Kolmogorov complexity $K(x)$ has an optimality property among all possible ways of defining descriptive complexity with algorithms
- Consider a general description language to be any computable function $p: \Sigma^* \rightarrow \Sigma^*$
- E.g., p could be the programming language LISP (encoded in binary)
- Define the minimal description of x w.r.t. p – denoted $d_p(x)$ – to be the lexicographically shortest string s where $p(s) = x$
- Let $K_p(x) = |d_p(x)|$
- If, for example, $p = \text{LISP}$, then
 - $d_{\text{LISP}}(x)$ is the minimal LISP program that outputs x
 - $K_{\text{LISP}}(x)$ is the length of the minimal program



Theorem 6.27 For any description language p , a fixed constant c exists that depends only on p , where

$$\forall x: K(x) \leq K_p(x) + c$$

Proof: Take any description language p and consider the following Turing machine:

$M = \text{"On input } w: \text{ Output } p(w).\text{"}$

Then $\langle M \rangle d_p(x)$ is a description of x whose length is at most the fixed constant $|\langle M \rangle|$ greater than $d_p(x)$. \square

Let $p = \text{LISP}$. Suppose that x has a short description w in LISP.

Let M be a TM that can interpret LISP and use the LISP program w for x as input to M .

Then $\langle M, w \rangle$ is a description of x that is only a fixed amount $|M|$ larger than the LISP description of x

Incompressible Strings and Randomness

- For some strings the minimal description may be much shorter than the string itself (if the information in the string appears sparsely or redundantly)
- Do some strings lack short descriptions?
- I.e., is the minimal description of some strings actually as long as the string itself?
- Such strings exist: They cannot be described more concisely than simply writing them out explicitly

Definition 6.28 Say that string x is c -compressible if

$$K(x) \leq |x| - c.$$

If x is not c -compressible, we say that x is *incompressible by c* .

If x is incompressible by 1, we say that x is **incompressible**.

Theorem 6.29 Incompressible strings of every length exist.

Proof: The number of distinct binary strings of length n is 2^n . Each description is a binary string, so the number of descriptions of length $< n$ is at most the sum of the number of strings of each length up to $n-1$:

$$\sum_{i=0}^{n-1} 2^i = 1 + 2 + 4 + \dots + 2^{n-1} = 2^n - 1$$

The number of short descriptions is less than the number of strings of length n . Therefore at least one string of length n is incompressible. \square

Corollary 6.30 At least $2^n - 2^{n-c+1} + 1$ strings of length n are incompressible by c .



- Incompressible strings have many properties we would expect to find in randomly chosen strings:
 - For example, any incompressible string of length n has roughly an equal number of 0s and 1s
 - The length of its longest run of 0s is approx. $\log_2 n$
- Any computable property that holds for “almost all” strings also holds for all sufficiently long incompressible strings
 - A property of strings is simply a function that maps strings to the truth values
 - A property holds for almost all strings if the fraction of strings of length n on which it is false approaches 0 as n grows large
- Also a randomly chosen long string is likely to satisfy a computable property that holds for almost all strings



Theorem 6.31 Let f be a computable property that holds for almost all strings. Then for any $b > 0$, the property f is False on only finitely many strings that are incompressible by b .

Proof: Let us consider the following algorithm:

$M =$ "On input i , a binary integer:

1. Find the i th string s where $f(s) = \text{False}$, considering the strings ordered lexicographically.
2. Output string s ."

- We can use M to obtain short descriptions of strings that fail to have property f as follows.
- For any such string x , let i_x be the index of x on a list of all strings that fail to have property f , ordered lexicographically.
- Then $\langle M, i_x \rangle$ is a description of x with length $|i_x| + c$.
- Because only few strings fail to have property f , the index of x is small and its description is correspondingly short.



- Fix any number $b > 0$. Select n such that at most a $1/2^{b+c+1}$ fraction of strings of length n or less fail to have property f .
- All sufficiently large n satisfy this condition because f holds for almost all strings.
- Let x be a string of length n that fails to have property f .
- We have $2^{n+1}-1$ strings of length n or less, so

$$i_x \leq \frac{2^{n+1}-1}{2^{b+c+1}} \leq 2^{n-b-c}.$$

- Therefore, $|i_x| \leq n-b-c$, so the length of $\langle M, i_x \rangle$ is at most $(n-b-c) + c = n-b$, which implies that $K(x) \leq n-b$.
- Thus, every sufficiently long x that fails to have property f is compressible by b .
- Hence, only finitely many strings that fail to have property f are incompressible by b . \square



Examples of incompressible strings?

- Unfortunately, Kolmogorov complexity is not computable (exercises next week)
- No algorithm can decide in general whether strings are incompressible (exercises next week)
- Indeed, no infinite subset of them is Turing-recognizable
- So we have no way of obtaining long incompressible strings
- In any case, we would have no way to determine whether a string is incompressible even if we could somehow obtain such
- The following theorem describes certain strings that are nearly incompressible, but doesn't provide a way to exhibit them explicitly



Theorem 6.32 For some constant b , for every string x , the minimal description $d(x)$ of x is incompressible by b .

Proof: Consider the following TM:

- $M =$ "On input $\langle R, y \rangle$, where R is a TM and y is a string:
1. Run R on y and reject if its output isn't of the form $\langle S, z \rangle$
 2. Run S on z and halt with its output on the tape."

Let b be $|\langle M \rangle| + 1$. We show that b satisfies the claim. Suppose to the contrary that $d(x)$ is b -compressible for some string x . Then

$$K(d(x)) = |d(d(x))| \leq |d(x)| - b.$$

But then $\langle M \rangle d(d(x))$ is a description of x whose length is at most

$$|\langle M \rangle| + |d(d(x))| \leq (b - 1) + (|d(x)| - b) = |d(x)| - 1.$$

This description of x is shorter than $d(x)$, contradicting the latter's minimality. \square

