Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

Lecture 21

Mapping Reductions

slides by Graham Farr

Overview

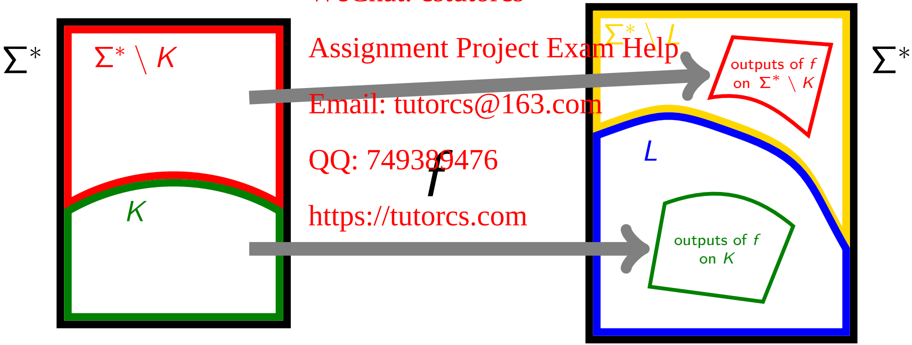
- Mapping reductions: relating one language to another
- Definition
- Properties
- Examples

Mapping reductions

Definition

A **mapping reduction** from language K to language L is a computable function $f : \Sigma^* \to \Sigma^*$ such that, for every $x \in \Sigma^*$,

$$x \in K \quad \text{if and only if} \quad f(x) \in L.$$



Mapping reductions

Notation:

$$K \leq_m L \quad \text{means} \ \ldots \ \text{mapping reduction from } K \text{ to } L.$$

A very simple property:
Every language is mapping-reducible to itself:

$$\forall L : L \leq_m L$$

Mapping reductions

Theorem

If there is a mapping reduction f from K to L, then:

If L is decidable, then K is decidable.

Symbolically:

$$(K \leq_m L) \wedge (L \text{ is decidable}) \implies (K \text{ is decidable})$$

Proof.

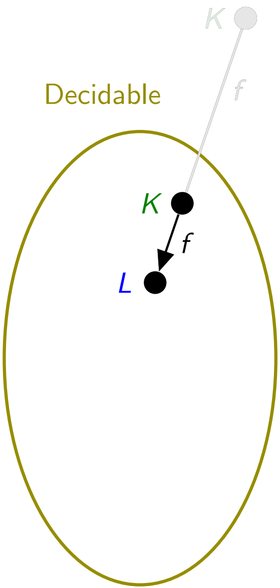
Decider for K:

Input: x.

Compute $f(x)$.

Run the Decider for L on $f(x)$.

// *This L-Decider accepts $f(x)$ if and only if $x \in K$, since f is a mapping reduction from K to L.* □



Mapping reductions

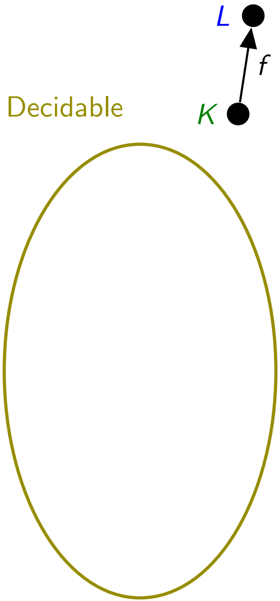
Corollary
If there is a mapping reduction from K to L, then:

If K is <u>un</u>decidable, then L is <u>un</u>decidable.

Symbolically:

$$(K \leq_m L) \wedge (K \text{ is } \underline{\text{un}}\text{decidable}) \implies (L \text{ is } \underline{\text{un}}\text{decidable})$$

Proof.
Contrapositive of previous Theorem. $\square$



EQUAL to HALF-AND-HALF

Mapping reduction f:

Input: a word w over alphabet $\{\mathtt{a},\mathtt{b}\}$

Sort w

Output the sorted word.

$$
\begin{aligned}
w \in \text{EQUAL} &\iff \text{it has the same number of } \mathtt{a}\text{'s as } \mathtt{b}\text{'s} \\
&\iff \text{after sorting, it has the same number of } \mathtt{a}\text{'s as } \mathtt{b}\text{'s} \\
&\qquad \text{(since sorting does not affect letter frequencies)} \\
&\iff f(w) \text{ consists of some number of } \mathtt{a}\text{'s followed by} \\
&\qquad \textit{the same number} \text{ of } \mathtt{b}\text{'s} \\
&\iff f(w) \in \text{HALF-AND-HALF}
\end{aligned}
$$

HALF-AND-HALF to PARENTHESES

Mapping reduction:

Input: a word w

For each letter of w in tur

If previous letter was `b` and current letter is `a`

// We have just seen `ba` which is impossible in HALF-AND-HALF.

Output the string `)`.

else

replace current letter as follows:

$\texttt{a} \mapsto$ `(`

$\texttt{b} \mapsto$ `)`

Output: the string obtained from w by doing all these replacements.

EQUAL to PARENTHESES

Is there a mapping reduction fr[illegible]AL to PARENTHESES?

Yes! Compose the two previous[illegible] reductions.

This is a special case of:

Theorem.
Mapping reducibility is <u>transitive</u>:

$$K \leq_m L \leq_m M \implies K \leq_m M.$$

Mapping reductions: transitivity

Theorem.
Mapping reducibility is transitive:

$$K \leq_m L \text{ and } L \leq_m M \implies K \leq_m M.$$

Proof.
Let f be a mapping reduction from K to L, and
let g be a mapping reduction from L to M.
We claim that the composition $g \circ f$, defined for all w by $g \circ f(w) = g(f(w))$,
is a mapping reduction from K to M.
Since f and g are both computable, $g \circ f$ must be too.

$$\begin{aligned} w \in K &\iff f(w) \in L && \text{(since } f \text{ is a mapping reduction from } K \text{ to } L\text{)} \\ &\iff g(f(w)) \in M && \text{(since } g \text{ is a mapping reduction from } L \text{ to } M\text{)} \\ &\iff (g \circ f)(w) \in M && \text{(by definition of } g \circ f\text{).} \end{aligned}$$

$\square$

FA-Empty $\longrightarrow$ No-Digraph-Path

From previous lecture:

$$\text{FA-Empty} := \{\langle A \rangle \ldots \text{FA and } L(A) = \emptyset\}$$

$$\text{Digraph-Path} := \{\langle G, s, t\rangle : G \text{ is a directed graph, } s, t \text{ are vertices in } G\text{, and there exists a directed } s\text{–}t \text{ path in } G.\}$$

$$\text{No-Digraph-Path} := \{\langle G, s, t\rangle : G \text{ is a directed graph, } s, t \text{ are vertices in } G\text{, and there } \underline{\text{does not exist}} \text{ a directed } s\text{–}t \text{ path in } G.\}$$

We give a mapping reduction from FA-Empty to No-Digraph-Path.

FA-Empty $\longrightarrow$ No-Digraph-Path

Mapping reduction

Input: $\langle A \rangle$ where A is a [illegible]omaton.

1. Construct the directed gra[illegible] A:
   - initially, vertices of G := states of A
   - every transition $v \xrightarrow{x} w$ in A becomes a directed edge (v, w) from v to w in G.
   - then add a new vertex t
   - for every Final State v of A, add a new directed edge (v, t) from v to t in G.
2. Specify s and t:
   - s := vertex of Start State of A.
   - t is as created above (the new vertex).
3. Output: $\langle G, s, t \rangle$

FA-Empty $\longrightarrow$ No-Digraph-Path

$A \in$ FA-Empty $\iff$ there is no sequence of transitions in A leading from Start State to a Final State

$\iff$ there is no path in G leading from s to a vertex representing a Final State

$\iff$ there is no path in G leading from s to t

$\iff$ $\langle G, s, t \rangle \in$ No-Digraph-Path

RegExpEquiv $\longrightarrow$ FA-Empty

From previous lecture:

$$\begin{aligned} \text{RegExpEquiv} &:= \{\langle A, B\rangle : A, B \text{ are regular expressions and } L(A) = L(B)\} \\ \text{FA-Empty} &:= \{\langle A\rangle : A \text{ is a FA and } L(A) = \emptyset\} \end{aligned}$$

We give a mapping reduction from RegExpEquiv to FA-Empty.

RegExpEquiv $\longrightarrow$ FA-Empty

Mapping reduction:

Input: $\langle A, B\rangle$ where A and B are regular expressions

1. Construct a FA, C, that defines the language

$$(L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)).$$

2. Output: C

Reducing *from* a decidable language

Let's reduce from EnglishPalinc YearsOfTransitsOfVenus:

$$\text{YearsOfTransitsOfVenus} := \quad \text{a Transit of Venus occurs in year } n \,\}$$
$$:= \{\ldots, 1761, 1769, 1874, 1882, 2004, 2012, 2117, \ldots\}$$

Mapping reduction:

Input: a string w over the English alphabet

If w is a palindrome

output 2012

else

output 2021.

Reducing *from* a decidable language

Theorem.
If L_1 is decidable and L_2 is *any* language except $\emptyset$ and Σ^* then

$$L_1 \;\leq_m\; L_2\,.$$

Proof. Let D be a decider for L_1.

Let $x^{(\text{yes})}$ be any specific word in L_2.
Let $x^{(\text{no})}$ be any specific word in $\overline{L_2}$.

Mapping reduction from L_1 to L_2:

Input: a string w

1. Run D on w.
2. If D accepts w then output $x^{(\text{yes})}$
   else output $x^{(\text{no})}$.

There's not much point in a mapping reduction that *decides* L_1 !

Revision

Reading: Sipser, pp. 234–238.