3P17/4P17 COMPUTABILITY 2014-2015 Lecture Handout 10

Formalising reducibility: Mapping Reducibility

A language A is **mapping reducible** to a language B, written $A \leq_m B$, if there is a **computable** function $f : \Sigma^* \to \Sigma^*$ where, for every $\sigma \in \Sigma^*$,

$$\sigma \in A \iff f(\sigma) \in B.$$

The function f is called a **reduction**.

If you have f, you can test if $\sigma \in A$ by mapping σ to $f(\sigma)$ and testing if $f(\sigma) \in B$.

Theorem 1. *If $A \leq_m B$ and B is decidable, then A is decidable.*

Proof. Suppose Turing machine M decides B and f is a mapping reduction from A to B. A decider M' for A operates as follows on input σ:

(1) Compute $f(\sigma)$.

(2) Simulate M on $f(s)$. If it rejects, then reject, if it accepts then accept.

If $\sigma \in A$ then $f(\sigma) \in B$ because f is a reduction from A to B. Then M will accept $f(\sigma)$ and so M' will accept σ.

If $\sigma \notin A$ then $f(\sigma) \notin B$. Then M will reject $f(\sigma)$ and so M' will reject σ.

□

Corollary 2. *If $A \leq_m B$ and A is undecidable, then B is undecidable.*

Note that we have assumed the two languages are over the same alphabet Σ. This is not a problem because, for example, all languages can themselves be encoded into a scheme that uses $\Sigma = \{0, 1\}$.

Note that because f is a function, it must have a mapping for all input strings over Σ, even if some of them do not have the form of the elements of the language in question. In this case, we can just choose some e not in B (which will exist if B is not Σ^*) and use that as the value to map to. For the arithmetic Turing machines, we chose the empty string ε, which is often, but not always, a suitable error output. Of course, Turing machines computing functions must always halt with accept regardless of the output.

We will call e the error string, leaving its actual value (which is application-specific) unspecified.

Theorem 3. *If $A \leq_m B$ and B is recognisable, then A is recognisable.*

Proof. Similar to Theorem 1. □

Corollary 4. *If $A \leq_m B$ and A is not recognisable, then B is not recognisable.*

Mapping reduction examples

We assume all languages are over $\Sigma = \{0, 1\}$.

$A_{TM} \leq_m H_{TM}$
Define f as follows:

For a Turing machine M and an input string σ, set $f(\langle M, \sigma \rangle) = \langle M', \sigma \rangle$ where M' is defined below.

For any other $\sigma \in \Sigma^*$, set $f(\sigma) = \mathrm{e}$.

$\underline{M'}$
On input x, simulate M on x. If it accepts then accept. If it rejects then enter a loop.

Now if $\langle M, \sigma\rangle \in A_{TM}$, then M accepts σ so M' will accept $x = \sigma$. Therefore, $\langle M'\sigma\rangle \in H_{TM}$.

If $\langle M, \sigma\rangle \notin A_{TM}$ then M rejects or loops on σ, meaning M' loops on $x = \sigma$. Therefore $\langle M', \sigma\rangle \in H_{TM}$.

For σ of any other form, $f(\sigma) = \text{e} \notin H_{TM}$.

Thus, we have defined f and shown $\sigma \in A_{TM} \iff f(\sigma) \in H_{TM}$.

To complete the definition of mapping reduction, it remains to show that f is computable.

This is the case because there exists a TM which can check the form of the input. Also, constructing the string $\langle M', \sigma\rangle$ can be done because it merely involves altering the description of M (which it has) to loop instead of reject, and encoding M' along with σ.

Thus, f is a mapping reduction.

$\underline{E_{TM} \leq_m EQ_{TM}}$
Define f as follows:

For a Turing machine M, set $f(\langle M\rangle) = \langle M, M_\emptyset\rangle$.

For any other $\sigma \in \Sigma^*$, set $f(\sigma) = \text{e}$.

If $\langle M\rangle \in E_{TM}$ then $L(M) = \emptyset = L(M_\emptyset)$ so $\langle M, M_\emptyset\rangle \in EQ_{TM}$.

If $\langle M\rangle \notin E_{TM}$ then $L(M) \neq \emptyset = L(M_\emptyset)$ so $\langle M, M_\emptyset\rangle \notin EQ_{TM}$.

For σ of any other form, $f(\sigma) = \text{e} \notin EQ_{TM}$.

Thus, we have defined f and shown $\sigma \in E_{TM} \iff f(\sigma) \in EQ_{TM}$.

To complete the definition of mapping reduction, it remains to show that f is computable.

f is clearly computable since it involves merely checks and constructing the string $\langle M, M_\emptyset\rangle$ from the string $f\langle M\rangle$.

Thus, f is a mapping reduction.

$\underline{A_{TM} \leq_m E_{TM}}$
No mapping reduction exists from A_{TM} to E_{TM}.

In Handout 9, Theorem 1, M_1 accepts σ if and only if $L(M_\sigma)$ is non-empty. This suggests a mapping reduction f from A_{TM} to $\overline{E_{TM}}$.

For a Turing machine M and an input string σ, set $f(\langle M, \sigma\rangle) = \langle M_\sigma\rangle$ where M_σ is defined below (same as in Handout 9, Theorem 1).

For any other $\sigma \in \Sigma^*$, set $f(\sigma) = \text{e}$.

$\underline{M_\sigma}$
On input string x do the following:

(1) Compare x to σ. If $x \neq \sigma$ the reject. If $x = \sigma$ go to (2)

(2) Simulate M on σ. If M accepts, then accept. If it rejects, then reject.

If $\langle M, \sigma\rangle \in A_{TM}$ then $L(M_\sigma) = \{\sigma\} \neq \emptyset$, so $\langle M_\sigma\rangle \in \overline{E_{TM}}$.

If $\langle M, \sigma\rangle \notin A_{TM}$ then $L(M_\sigma) = \emptyset$, so $\langle M_\sigma\rangle \notin \overline{E_{TM}}$.

For σ of any other form, $f(\sigma) = \mathrm{e} \notin E_{TM}$.

Thus, we have defined f and shown $\sigma \in A_{TM} \iff f(\sigma) \in E_{TM}$.

To complete the definition of mapping reduction, it remains to show that f is computable.

This was justified in Handout 9, proof Theorem 1.

Thus, f is a mapping reduction.

Theorem 5. *EQ_{TM} is not recognisable.*

Proof. Not recognisable:

We will show A_{TM} is mapping reducible to $\overline{EQ_{TM}}$.

Since A_{TM} is recognisable (Handout 8, Theorem 2) but undecidable (Handout 8, Theorem 1), it follows by Handout 8 Theorem 3 that $\overline{A_{TM}}$ is not recognisable.

For languages A and B, $A \leq_m B$ if and only if $\overline{A} \leq_m \overline{B}$ (justification is left as an exercise).

Therefore $A_{TM} \leq_m \overline{E_{TM}}$ implies $\overline{A_{TM}} \leq_m EQ_{TM}$.

Hence, $\overline{A_{TM}}$ is not recognisable implies EQ_{TM} is not recognisable, by Corollary 4.

For TM M and input σ, set $f(\langle M, \sigma\rangle) = \langle M_\sigma, M_\emptyset\rangle$, where M_σ is as in the above.

For σ of any other form, $f(\sigma) = \mathrm{e} \notin \overline{EQ_{TM}}$.

If $\langle M, \sigma\rangle \in A_{TM}$ then $L(M_\sigma) = \{\sigma\} \neq \emptyset = L(M_\emptyset)$ so $\langle M_\sigma, M_\emptyset\rangle \in \overline{EQ_{TM}}$.

If $\langle M, \sigma\rangle \notin A_{TM}$ then $L(M_\sigma) = \emptyset = L(M_\emptyset)$ so $\langle M_\sigma, M_\emptyset\rangle \notin \overline{EQ_{TM}}$.

Strings of any other form map to e.

Thus, we have defined f and shown $\sigma \in A_{TM} \iff f(\sigma) \in \overline{EQ_{TM}}$.

To complete the definition of mapping reduction, it remains to show that f is computable.

This is clearly the case by the above justifications.

□