Mapping Reducibility

Alyssa Lytle

Fall 2025

1 Mapping Reducibility

The goal of this lesson is formalize the concept of reducibility

We will define the *mapping reducibility* of problem A to problem B as the ability to define a function mapping from A to B.

1.1 Computability

First let's define the type of function we will be talking about here.

Definition: Computable Function

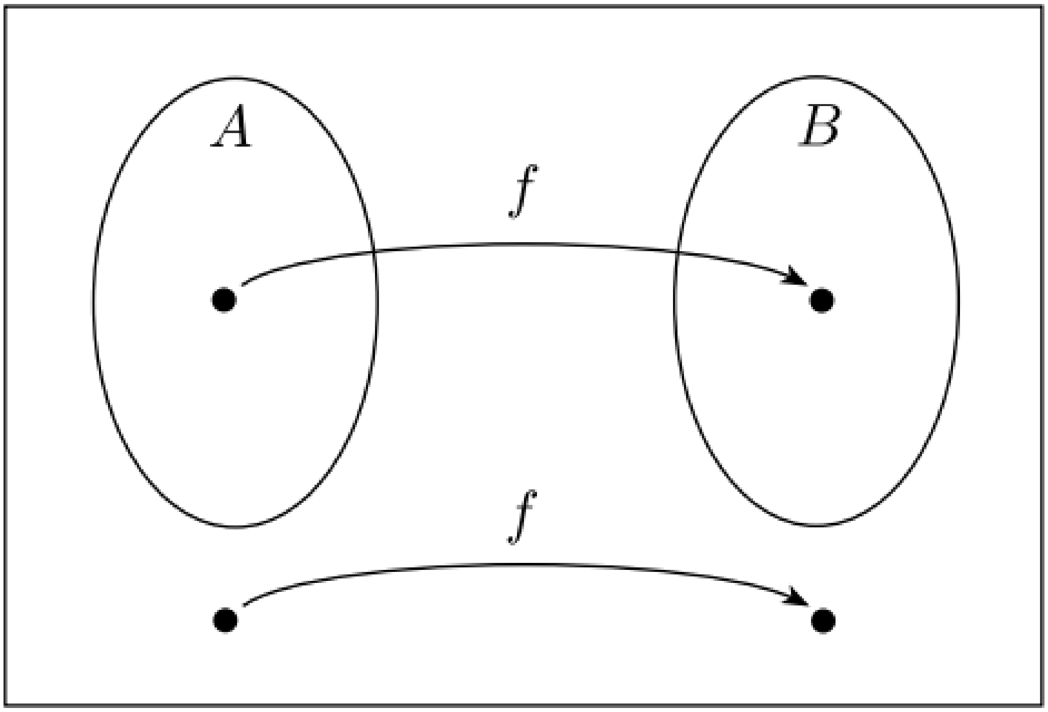
A function $f : \Sigma^* \to \Sigma^*$ is a *computable function* if some turing machine M, on every input w, halts with just $f(w)$ on its tape.

Example:

Show that $m + n$ is a computable function.

One thing to note: a computable function can be a transformation of a machine description. In other words, a TM can take a string description of a turing machine M as input and return a string description of another machine M'.

1.2 Mapping Reducibility



Definition: Mapping Reducible

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

$$w \in A \iff f(w) \in B$$

.
The function of f is called the *reduction* from A to B.

If A is the solution set of one problem, and B is the solution set to another problem, we can convert questions about membership in A to questions about membership in B!

1.3 Reducibility + Decidability

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

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

Example: The Halting Problem (Again)

Let's again show $HALT_{TM}$ is undecidable by showing A_{TM} can be reduced to $HALT_{TM}$.
Recall:
$A_{TM} = \{\langle M, w\rangle | M \text{ is a Turing Machine and accepts } w\}$
$HALT_{TM} = \{\langle M, w\rangle | M \text{ is a Turing Machine and halts on input } w\}$
For a reminder, here is our previous proof: We know that A_{TM} is undecidable. So, we need to show that A_{TM} is reducible to $HALT_{TM}$.
Assume we have a TM R that decides $HALT_{TM}$. We will use R to construct a TM S that decides A_{TM}.

$S =$ "On input $\langle M, w\rangle$:

1. Run TM R on input $\langle M, w\rangle$
2. If R rejects, *reject*.
3. If R accepts, simulate M on w until it halts.
4. If M has accepted, *accept*; if M has rejected, *reject*."

Our solution is very similar.
We construct a machine F that computes a reduction f.
In this case, we're inputting a suggested solution to the A_{TM}, $\langle M, w\rangle$, and outputting a suggested solution to $HALT_{TM}$, $\langle M', w'\rangle$.
In order for f to be a reduction, it must be true that $\langle M, w\rangle \in A_{TM} \iff f(\langle M, w\rangle) \in HALT_{TM}$.
Essentially this means f will have to construct a machine M' such that:
"M accepts w if and only if M' halts on w"'
and moreover
"M rejects w if and only if M' loops forever on w"'.
Here, we actually don't need to change the input, so we will see that
$\langle M, w\rangle \in A_{TM} \iff \langle M', w\rangle \in HALT_{TM}$.

$F =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M'.
   $M' =$ "On input x :
   1. Run M on x
   2. If M accepts, *accept*.
   3. If M rejects, enter a loop."
2. Output $\langle M', w\rangle$.

Example:

Let's do a basic reducibility proof and then show that it's mapping reducible.
Recall:
$E_{TM} = \{\langle M\rangle | M$ is a Turing Machine and $L(M) \neq \emptyset\}$
and
$EQ_{TM} = \{\langle M_1, M_2\rangle | M_1$ and M_2 are Turing Machines and $L(M_1) = L(M_2)\}$
Let's assume we know E_{TM} is undecidable, and let's use that to show EQ_{TM} is also undecidable.
So, we assume there exists a TM R that decides EQ_{TM} and show that it can be used to build a TM S that solves E_{TM}.
Here, we will design it so $L(S)$ is the set of all TMs with an emtpty language. (All $\langle M\rangle$ such that M is a Turing Machine and $L(M) = \emptyset$).
This design is pretty straightforward:

$S =$ "On input $\langle M\rangle$:

1. Run TM R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.
2. If R accepts, *accept*.
3. If R rejects, *reject*."

Doing this as a mapping reducibility just takes a few extra steps.
We want to construct machine F that computes reduction f such that $\langle M\rangle \in E_{TM} \iff f(\langle M\rangle) \in EQ_{TM}$

$F =$ "On input $\langle M\rangle$:

1. Construct the machine M_1 that rejects all inputs.
2. Output $\langle M, M_1\rangle$."