How to Understand and Create Mapping Reductions

What is a mapping reduction?

- A mapping reduction $A \leq_m B$ (or $A \leq_P B$) is an algorithm (respectively, polytime algorithm) that can transform any instance of decision problem $A$ into an instance of decision problem $B$, in such a way that the answer correspondence property holds.

- Answer correspondence property: The answer (yes/no) to any $A$ problem instance must be the same as the answer to the corresponding $B$ problem instance to which the reduction transforms it.\(^1\)

- Thus the following two-step algorithm can be used to solve any $A$ problem instance:
  1. Transform the given $A$ problem instance to a corresponding $B$ problem instance.
  2. Call a solver for $B$ problem instances and return whatever answer (yes/no) it gives.

How should one make sense of the $\leq$ notation?

- $A \leq_m B$ means “$A$ problems are no harder to solve than $B$ problems.”

- $A \leq_P B$ means “$A$ problems are no harder to solve in polytime than $B$ problems.”

- $A \leq_m B$ means “Being able to solve any $B$ problem $\Rightarrow$ being able to solve any $A$ problem.”

- $A \leq_P B$ means “Being able to solve any $B$ problem in polytime $\Rightarrow$ being able to solve any $A$ problem in polytime.”

Examples of decision problem instances:

- An $A_{TM}$ problem instance is (an encoding of) a given TM and a given string. The associated yes/no question is: *Does the given TM accept the given string?*

- A $\text{REGULAR}_{TM}$ problem instance is (an encoding of) a given TM. The associated yes/no question is: *Is the language recognized by the given TM regular?*

- A $\text{CLIQUE}$ problem instance is (an encoding of) a given undirected graph and a given number. The associated yes/no question is: *Does the given graph have a clique of the given size?*

- A $3\text{SAT}$ problem instance is (an encoding of) a 3CNF Boolean formula. The associated yes/no question is: *Does the given Boolean formula have a satisfying assignment?*

---

\(^1\)If the answer is just the opposite, for most purposes that’s okay too. In this case the mapping reduction is actually from $A$ to $B$. 

1
What are mapping reductions used for?

- To prove undecidability: If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable.
- To prove non-Turing-recognizability: If \( A \leq_m B \) and \( A \) is non-Turing-recognizable, then \( B \) is non-Turing-recognizable.
- To prove NP-completeness: If \( A \leq_P B \) and \( A \) is NP-complete (and \( B \in \text{NP} \)), then \( B \) is NP-complete.

Other, less common, uses for mapping reductions:

- To prove decidability: If \( A \leq_m B \) and \( B \) is decidable, then \( A \) is decidable.
- To prove Turing-recognizability: If \( A \leq_m B \) and \( B \) is Turing-recognizable, then \( A \) is Turing-recognizable.
- To prove membership in \( \text{P} \): If \( A \leq_P B \) and \( B \) is in \( \text{P} \), then \( A \) is in \( \text{P} \).

Important considerations when constructing mapping reductions

1. Make sure your reduction goes in the correct direction. For example:\(^2\)
   - If you’re trying to prove \( A \) is undecidable, will it help to construct a reduction \( A \leq_m B \) for some undecidable language \( B \)?
   - If you’re trying to prove \( A \) is NP-complete, will it help to construct a reduction \( A \leq_P B \) for some NP-complete language \( B \)?

2. Type match: A reduction (transformation) is an algorithm whose input and output must be of the right type. Examples:
   - \( A_{\text{TM}} \leq_m \text{REGULAR}_{\text{TM}} \)
     - \( A_{\text{TM}} \) problem instances have the form \( \langle M, w \rangle \), where \( M \) is a TM and \( w \) is a string.
     - \( \text{REGULAR}_{\text{TM}} \) problem instances have the form \( \langle M' \rangle \), where \( M' \) is a TM.
     - Therefore the input to the reducing function (transformation) must be (an encoding of) an arbitrary TM and an arbitrary string, and the output must be (an encoding of) some TM.
   - \( 3\text{SAT} \leq_P \text{CLIQUE} \)
     - \( 3\text{SAT} \) problem instances have the form \( \langle \phi \rangle \), where \( \phi \) is a 3CNF Boolean formula.
     - \( \text{CLIQUE} \) problem instances have the form \( \langle G, k \rangle \), where \( G \) is an undirected graph and \( k \) is a number.
     - Therefore the input to the reducing function (transformation) must be (an encoding of) an arbitrary 3CNF formula \( \phi \) and the output must be (an encoding of) some undirected graph \( G \) and some number \( k \).

\(^2\)Analogously, if you’re trying to prove a number is very large, will it help to prove that it’s less than or equal to some very large number?
3. **Answer correspondence:** The answer to the transformed problem instance should be the same as the answer to the original problem instance, for *any* instance of the original problem.\(^3\)

Examples:

- **\(A_{TM} \leq_m \text{REGULAR}_{TM}\)**
  - Let \(M'\) denote the corresponding TM whose encoding is produced as output by the reducing function when given as input (the encoding of) a TM \(M\) and a string \(w\).
  - If the answer is *yes* to the question *Does TM \(M\) accept string \(w\)?*, then the answer must be *yes* to the question *Does the TM \(M'\) recognize a regular language?*
  - If the answer is *no* to the question *Does TM \(M\) accept string \(w\)?*, then the answer must be *no* to the question *Does the TM \(M'\) recognize a regular language?*

- **3SAT \(\leq_P \text{CLIQUE}\)**
  - Let \(G\) and \(k\) denote the corresponding graph and number, respectively, whose encoding is produced as output by the reducing function when given as input (the encoding of) a 3CNF formula \(\phi\).
  - If the answer is *yes* to the question *Does \(\phi\) have a satisfying assignment?*, then the answer must be *yes* to the question *Does the graph \(G\) have a \(k\)-clique?*
  - If the answer is *no* to the question *Does \(\phi\) have a satisfying assignment?*, then the answer must be *no* to the question *Does the graph \(G\) have a \(k\)-clique?*

Proving answer correspondence always involves an “if and only if” proof.

---

\(^3\)As observed earlier, for most purposes it’s okay if the answers are always opposite. This just means the mapping reduction is actually from \(A\) to \(\overline{B}\).