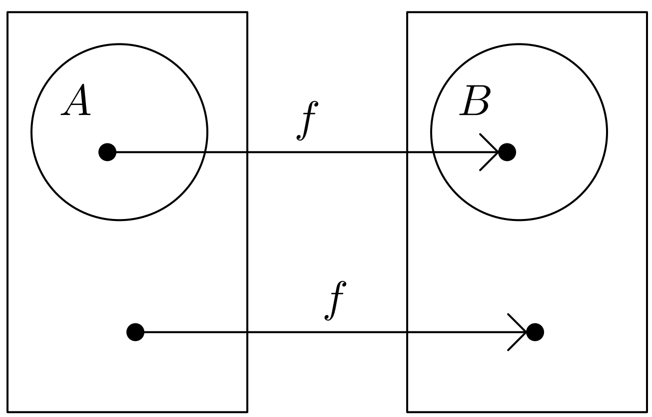
Reductions

Recall that a T-computable function is a function from strings to strings for which there is a TM. Let A, B be languages. We say that A is ***reducible*** to B, written $A \leq_m B$, if there is a T-computable function f such that $w \in A$ exactly when $f(w) \in B$.



Reductions Preserve Hardness

Fact.
a) If A is reducible to B and B is recursive, then A is recursive.
b) If A is reducible to B and A is not recursive, then B is not recursive.

Proof (of a). Let TM R decide language B, and let function f reduce A to B. Construct TM S as follows: On input w, it computes $f(w)$ and submits this to R; then it accepts if R accepts. So S decides A.

Why The Notation $\leq$?

The above fact shows if one writes $A \leq_m B$, then B is as least as hard as A. This relationship behaves as one would expect. For example:

Fact. *For any languages A, B and C: If $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.*

If f reduces A to B and g reduces B to C, then h defined by $h(w) = g(f(w))$ reduces A to C.

Practice

Show that for any languages A and B: If $A \leq_m B$ then $\bar{A} \leq_m \bar{B}$.

Solution to Practice

The same reduction works! If function f reduces A to B, then it maps A to B and $\bar{A}$ to $\bar{B}$.