A ***reduction*** $R$ from $L_1$ to $L_2$ is one or more Turing machines such that:

If there exists a Turing machine *Oracle* that decides (or semidecides) $L_2$, then the Turing machines in $R$ can be composed with *Oracle* to build a deciding (or a semideciding) Turing machine for $L_1$.

$L_1 \leq L_2$ means that $L_1$ is **reducible** to $L_2$.

Assume:

$(L_1 \leq L_2) \wedge (L_2 \text{ is in D}) \rightarrow (L_1 \text{ is in D})$

If ($L_1$ is in D) is false, then at least one of the two antecedents of that implication must be false. So:

If $(L_1 \leq L_2)$ is true,
then ($L_2$ is in D) must be false.

0. Assume *Oracle* that decides $L_2$ exists

1. Choose a language $L_1$:
    - that is already known not to be in D, and
    - that can be reduced to $L_2$.

2. Define the reduction *R.*

3. Describe the composition *C* of *R* with *Oracle*:

$$C(x) = Oracle(R(x))$$

4. Show that *C* does correctly decide $L_1$ if *Oracle* exists. We do this by showing:

- *R* can be implemented by Turing machines,
- *C* is correct:
    - If $x \in L_1$, then $C(x)$ accepts, and
    - If $x \notin L_1$, then $C(x)$ rejects.

$L_1$ is ***mapping reducible*** to $L_2$ ($L_1 \leq_M L_2$) iff there exists some computable function *f* such that:

$$\forall x \in \Sigma^* \ (x \in L_1 \leftrightarrow f(x) \in L_2).$$

To decide whether *x* is in $L_1$, we transform it, using *f*, into a new object and ask whether that object is in $L_2$.

**Note:** mapping reduction is a particular case of Turing reduction.

# The Steps in a Reduction Proof

1. Assume *Oracle* exists.
2. Choose an undecidable language to reduce from.
3. Define the reduction *R*.
4. Show that *C* (the composition of *R* with *Oracle*) is correct.