The fall-out: part 1

Reductions: Given languages

$$L_1 \subseteq \Sigma_1^*, \quad L_2 \subseteq \Sigma_2^*$$

A *reduction from* L_1 *to* L_2 is a function $f : \Sigma_1^* \to \Sigma_2^*$ satisfying:

(a) $x \in L_1 \iff f(x) \in L_2, \quad$ for all $x \in \Sigma_1^*$;

(b) there is a Turing machine transducer that computes f.

In other words: The question

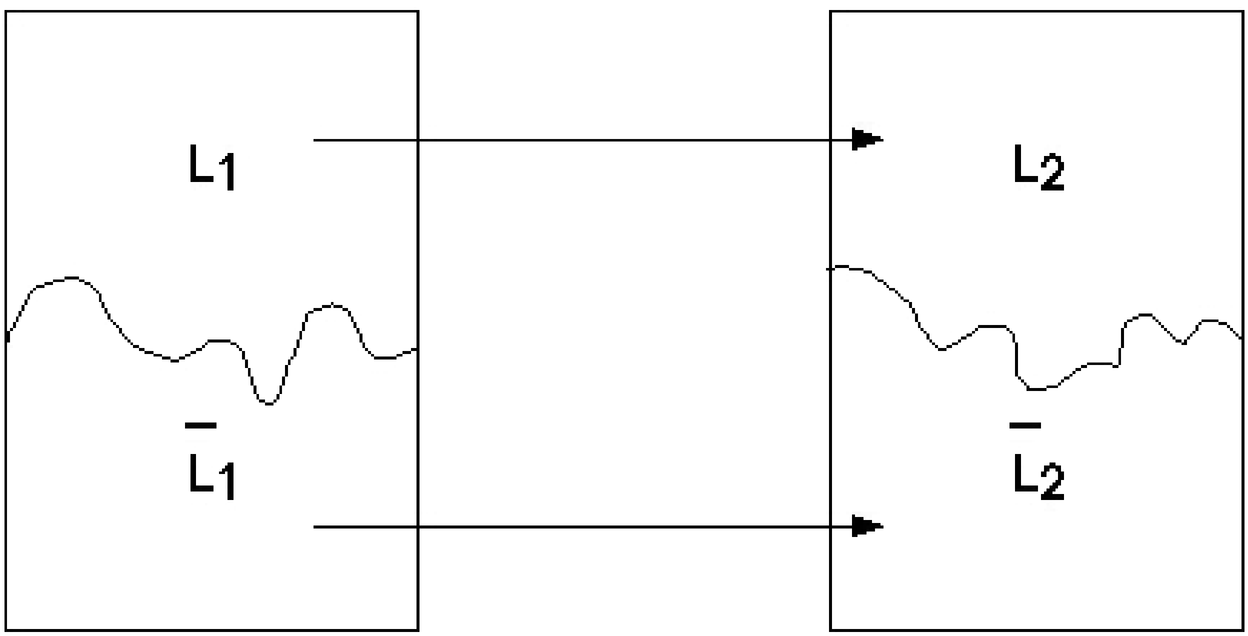
'is $x \in L_1$?'

has the *same* answer as

'is $f(x) \in L_2$?'

Moreover we have an algorithm for transforming first question to second.

Say that L_1 *is reducible to* L_2 if a reduction from L_1 to L_2 exists.


L_1
L_2
$\bar{L}_1$
$\bar{L}_2$

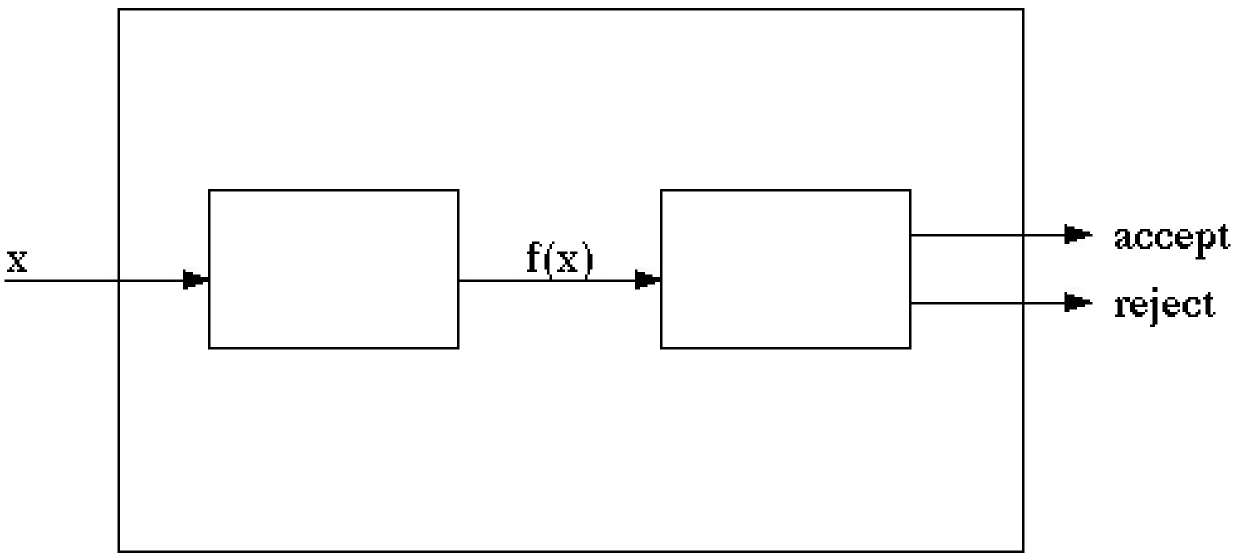
THEOREM Suppose $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ are languages. If L_1 is reducible to L_2, and L_2 is recursive, then L_1 is also recursive.

Equivalently: If L_1 is *not* recursive then L_2 is not recursive.

Use this version to prove new things are non-recursive starting with L_{halt}.

Proof of Theorem: High level description

On input x
  call M and compute $f(x)$;
  run M_2 on $f(x)$ and give its decision;


x
f(x)
accept
reject

The uniform halting problem:

INSTANCE: A binary Turing machine M.

QUESTION: Does M halt on *all* inputs $x \in \{0,1\}^*$?

Language:

$L_{\text{uhalt}} = \{\langle M \rangle \mid M \text{ halts on all inputs } x \in \{0,1\}^*\}$.

Looks unsolvable but looks can be deceptive!

THEOREM The language L_{uhalt} is not recursive.

PROOF Reduction from L_{halt} to L_{uhalt}.

$$L_{\text{halt}} \subseteq \{\mathtt{0}, \mathtt{1}, \mathtt{\$}\},$$
$$L_{\text{uhalt}} \subseteq \{\mathtt{0}, \mathtt{1}\}$$

Only *really* interested in well formed strings $\langle M \rangle \mathtt{\$} x$ but must deal with all.

- send every badly formed string to $\mathtt{0}$ (or any string of odd length).

Send a well formed string $\langle M \rangle \mathtt{\$} x$ to M_x described by:

Given input $w \in \{\mathtt{0}, \mathtt{1}\}^*$
 if $w \neq x$ then halt
 else simulate M on x and do what it does

Clearly

M halts on $x \iff M_x$ halts on all inputs

Formally reduction is

$$y \mapsto \begin{cases} 0, & \text{if } y \text{ badly formed} \\ \langle M_x \rangle, & \text{if } y = \langle M \rangle \$ x. \end{cases}$$

But this just sums up in symbols what we said above!

Detailed construction of M_x from M and x:
Let

$$x = x_0 x_1 x_2 \cdots x_{n-1}, \text{each } x_i \in \{0, 1\}.$$

(1) Add $2n$ new states to M:

$$\begin{array}{l} q'_0, q'_1, \ldots, q'_{n-1}, \\ q''_1, q''_2, \ldots, q''_n. \end{array}$$

(2) Extend transition function to new states by:

(a) Adding right-sweeping quintuples

$$\begin{array}{l} (q'_0, x_0, q'_1, x_0, R), \\ (q'_1, x_1, q'_2, x_1, R), \\ \quad \vdots \\ (q'_{n-2}, x_{n-2}, q'_{n-1}, x_{n-1}, R), \\ (q'_{n-1}, x_{n-1}, q''_n, x_{n-1}, R) \end{array}$$

(b) Adding left-sweeping quintuples

$$(q''_n, \not{b}, q''_{n-1}, \not{b}, L),$$
$$(q''_{n-1}, x_{n-1}, q''_{n-2}, x_{n-1}, L),$$
$$\vdots$$
$$(q''_2, x_2, q''_1, x_2, L),$$
$$(q''_1, x_1, q_I, x_1, L)$$

Note: $\langle M_x \rangle$ easy to compute given $\langle M \rangle$ and x (actually $\langle M \rangle \$ x$).

Non-emptiness problem for r.e. languages:

INSTANCE: A binary Turing machine M.

QUESTION: Is the language $L(M)$ non-empty?

Language:

$$L_{\mathsf{ne}} = \{\langle M \rangle \mid L(M) \neq \emptyset\}.$$

THEOREM The language L_{ne} is not recursive.

PROOF Again reduction from L_{halt}.

Given $\langle M \rangle\$x$ we construct a Turing machine M_x s.t.

$$M \text{ halts on input } x \iff L(M_x) \neq \emptyset.$$

Can assume M never falls off left hand end of tape. Now M_x behaves as:

Given input x
  simulate M on x
  if this halts then accept

Easy to deal with badly formed strings for the reduction.

Number theory: a simple first-order theory:

Sentences formed from the following entities, according to 'appropriate syntactic rules':

(a) the constants 0 and 1;

(b) variables (denoted by lower case roman letters);

(c) the binary arithmetic operators $+$ and $\times$;

(d) the relational operators $<$ and $=$;

(e) the logical connectives $\wedge$, $\vee$, and $\neg$;

(f) the quantifiers $\exists$ (there exists) and $\forall$ (for all).

Sentences with no free variables interpreted as statements about $\mathbb{N}$.

Examples:

$$\begin{array}{ll} \forall x\, \exists y\, [x < y], & \text{true} \\ \forall x\, \exists y\, [x = y + y], & \text{false} \end{array}$$

Can make quite complicated assertions:

prime(x) $= \forall u\, \forall v\, [(u = 1) \vee (v = 1) \vee \neg(u \times v = x)]$.

$\forall x\, \exists y\, [(x < y) \wedge \text{prime}(y)]$, infinitely many primes (true).

$\forall x\, \exists y\, [(x < y) \wedge \text{prime}(y) \wedge \text{prime}(y + 1 + 1)]$, infinitely many prime pairs (only a conjecture).

L_{num} the set of *true* sentences. Gödel's Incompleteness theorem yields

THEOREM The language L_{num} is not recursive.