	To ***prove*** that this is true…	If you ***assume*** this is true…
$\forall x.\ A$	Have the reader pick an arbitrary x. We then prove A is true for that choice of x.	Initially, ***do nothing***. Once you find a z through other means, you can state it has property A.
$\exists x.\ A$	Find an x where A is true. Then prove that A is true for that specific choice of x.	Introduce a variable x into your proof that has property A.
$A \to B$	Assume A is true, then prove B is true.	Initially, ***do nothing***. Once you know A is true, you can conclude B is also true.
$A \wedge B$	Prove A. Then prove B.	Assume A. Then assume B.
$A \vee B$	Either prove $\neg A \to B$ or prove $\neg B \to A$. *(Why does this work?)*	Consider two cases. Case 1: A is true. Case 2: B is true.
$A \leftrightarrow B$	Prove $A \to B$ and $B \to A$.	Assume $A \to B$ and $B \to A$.
$\neg A$	Simplify the negation, then consult this table on the result.	Simplify the negation, then consult this table on the result.

Languages

- An **alphabet** is a finite, nonempty set of symbols called characters. Typically, we use the symbol Σ.
- A **string** over Σ is a *finite* sequence of characters drawn from Σ.
- The **empty string** has no characters and is denoted ε.
- L is a **language over Σ** if it is a set of strings over Σ.
- The **set of all strings** composed from letters in Σ is denoted Σ^*.
- **Concatenation:**
  - Of strings: If $w \in \Sigma^*$ and $x \in \Sigma^*$, wx is the string formed by putting all the characters of x onto the end of w.
  - Of languages: $L_1L_2 = \{ x \mid \exists w_1 \in L_1.\ \exists w_2 \in L_2.\ x = w_1w_2 \}$
- **Language powers:** Defined recursively. $L^0 = \{\varepsilon\}$ and $L^{n+1} = LL^n$
- **Kleene closure:** $L^* = \{ w \in \Sigma^* \mid \exists n \in \mathbb{N}.\ w \in L^n \}$

	Is defined as…	To ***prove*** that this is true…	If you ***assume*** this is true…
$S \subseteq T$	$\forall x \in S.\ x \in T$	Pick an arbitrary $x \in S$. Prove $x \in T$	Initially, ***do nothing***. Once you find some $x \in S$, conclude $x \in T$.
$S = T$	$S \subseteq T \ \wedge\ T \subseteq S$	Prove $S \subseteq T$. Then prove $T \subseteq S$.	Assume $S \subseteq T$ and $T \subseteq S$.
$x \in A \cap B$	$x \in A \ \wedge\ x \in B$	Prove $x \in A$. Then prove $x \in B$.	Assume $x \in A$. Then assume $x \in B$.
$x \in A \cup B$	$x \in A \ \vee\ x \in B$	Either prove $x \in A$ or prove $x \in B$.	Consider two cases: Case 1: $x \in A$. Case 2: $x \in B$.
$X \in \wp(A)$	$X \subseteq A$.	Prove $X \subseteq A$.	Assume $X \subseteq A$.
$x \in \{ y \mid P(y) \}$	$P(x)$	Prove $P(x)$.	Assume $P(x)$.

Finite Automata

- A collection of **states** linked by **transitions**
- One state is the **start state**. The computation begins in that state.
- The computation proceeds from left to right over the **input string**. When the automaton sees a character, it follows the transition with that label.
- Some states are **accepting states** (double ring). If the device ends in an accepting state after seeing all the input, it accepts the input. Otherwise, it rejects.

DFA Rules

- DFAs are defined relative to some alphabet Σ.
- For each state in the DFA, there must be **exactly one** transition defined for each symbol in Σ.
- There is a unique start state.
- There are zero or more accepting states.

The language of D, denoted $\mathscr{L}(D)$, is $\{ w \in \Sigma^* \mid D \text{ accepts } w \}$

NFA Rules

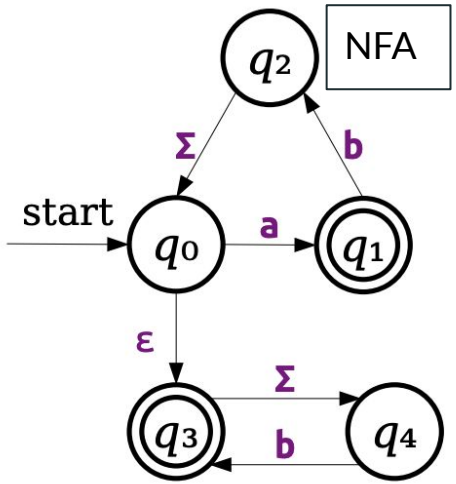
- NFAs have no restrictions on how many transitions are allowed per state.
- NFAs can use ε-transitions, which can be taken at any time.
- An NFA accepts a string if there is **some** sequence of choices that leads to an accepting state after every character from the string has been read.
- Two intuitions: “Perfect guessing”, “massive parallelism”

The Subset Construction

Process to convert an NFA into a DFA: Guide to the Subset Construction

Make a table of all the simultaneous states you can occupy, and the path options from each set of states

At most, the DFA will have $2^{|S|}$ states for an NFA with S states because $|\wp(S)| = 2^{|S|}$



Tabular DFA

	a	b
*$\{q_0, q_3\}$	$\{q_1, q_4\}$	$\{q_4\}$
*$\{q_1, q_4\}$	Ø	$\{q_2, q_3\}$
$\{q_4\}$	Ø	$\{q_3\}$
*$\{q_2, q_3\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_3, q_4\}$
*$\{q_3\}$	$\{q_4\}$	$\{q_4\}$
*$\{q_0, q_3, q_4\}$	$\{q_1, q_4\}$	$\{q_3, q_4\}$
*$\{q_3, q_4\}$	$\{q_4\}$	$\{q_3, q_4\}$
Ø	Ø	Ø

Regular Expressions

Here's a recap of the rules of regular expressions:

- $\varnothing$ is a regular expression that represents the empty language $\varnothing$.
- ε is a regular expression that represents the language $\{\varepsilon\}$.
- For any $\texttt{a} \in \Sigma$, the symbol $\texttt{a}$ is a regular expression for the language $\{\texttt{a}\}$.
- R^* is a regular expression for the Kleene closure of the language of R.
- For regular expressions R_1 and R_2, you can combine these regular expressions:
  - $R_1 R_2$ is a regular expression for the concatenation of the languages of R_1 and R_2.
  - $R_1 \cup R_2$ is a regular expression for the union of the languages of R_1 and R_2.
- $\texttt{R}^n$ is shorthand for $\texttt{RRR...R}$ (n copies of R). $\texttt{R}^0$ is defined as ε.
- Σ is shorthand for "any character in Σ".
- $\texttt{R?}$ is shorthand for $\texttt{R} \cup \varepsilon$, meaning "zero or one copies of $\texttt{R}$".
- $\texttt{R+}$ is shorthand for $\texttt{RR}^*$, meaning "one or more copies of $\texttt{R}$".

Regular Languages

The set of languages with an NFA

=

The set of languages with a DFA

=

The set of languages with a regular expression

=

The set of regular languages

Regular languages are closed under union, intersection, concatenation and Kleene closure. In other words, if L_1 and L_2 are both regular, then so are $L_1 \cup L_2$, $L_1 \cap L_2$, L_1L_2, and L_1^*.

Nonregular Languages

Two strings $x \in \Sigma^*$ and $y \in \Sigma^*$ are called **distinguishable relative to L** if there is a string $w \in \Sigma^*$ such that exactly one of xw and yw is in L. This is denoted as $\mathbf{x \not\equiv_L y}$ and formally defined as $\mathbf{\exists w \in \Sigma^*. (xw \in L \leftrightarrow yw \notin L)}$

A **distinguishing set for L** is a set $S \subseteq \Sigma^*$ where the following is true:

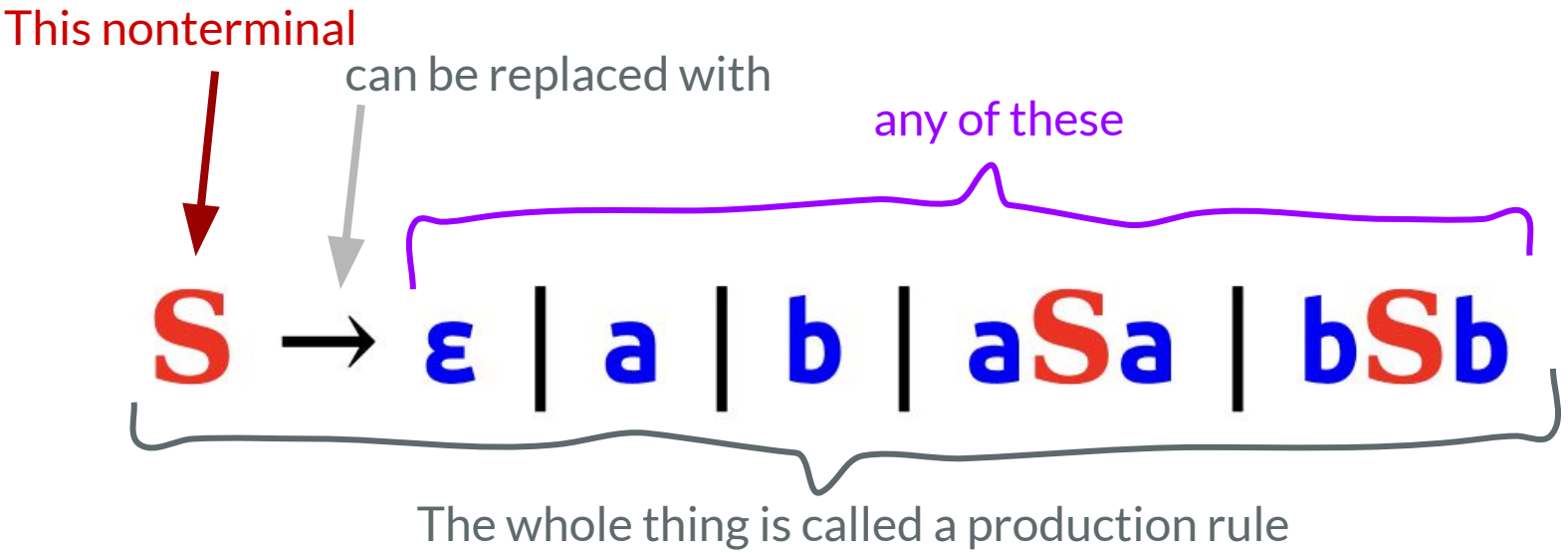
$$\mathbf{\forall x \in S.\ \forall y \in S.\ (x \neq y \rightarrow x \not\equiv_L y).}$$

"Every pair of distinct elements in the set is distinguishable."

Myhill-Nerode Theorem: If L has an infinite distinguishing set, then L is not regular.

Context-Free Languages

Guide to CFGs



- A CFG is a list of production rules
- A CFG's language = set of all **strings of terminals** derivable **from the start symbol** (the nonterminal from the **first** rule)

Turing Machines

- An idealized, infinite-memory computer
  - Has an infinite tape of symbols and a tape head
  - Can read or write only the symbol under the head
  - Operates based on a series of instructions and labels
- Takes a string as input (written on tape at start) and has three outcomes:
  - Accept
  - Reject
  - Loop
- Accepting or rejecting counts as halting
- Church-Turing Thesis claims: TMs are as powerful as, or more powerful than, any method of computation

Recognizability

A Turing Machine M is a **recognizer** for a language **L** if

$$\forall w \in \Sigma^*. (w \in L \leftrightarrow M \text{ accepts } w)$$

Here is an equivalent definition:

$$\forall w \in \Sigma^*. ((w \in L \rightarrow M \text{ accepts } w) \wedge (w \notin L \rightarrow M \text{ rejects } w \text{ or loops on } w))$$

A language is **recognizable** if there exists a recognizer for it.

Decidability

A Turing Machine M is a **decider** for a language **L** if

$$\textbf{M is a recognizer for L, AND}$$
$$\forall w \in \Sigma^*. \textbf{(M halts on w)}$$

Here is an equivalent definition:

$$\forall w \in \Sigma^*. ((w \in L \rightarrow \text{M accepts w}) \wedge (w \notin L \rightarrow \text{M rejects w}))$$

A language is **decidable** if there exists a decider for it.

Verifiers

A TM V is a **verifier** for a language L if:

$$\textbf{V halts on all inputs, AND}$$

$$\forall w \in \Sigma^*. (w \in L \leftrightarrow \exists c \in \Sigma^*. \text{V accepts } \langle w, c \rangle)$$

Here is an equivalent definition:

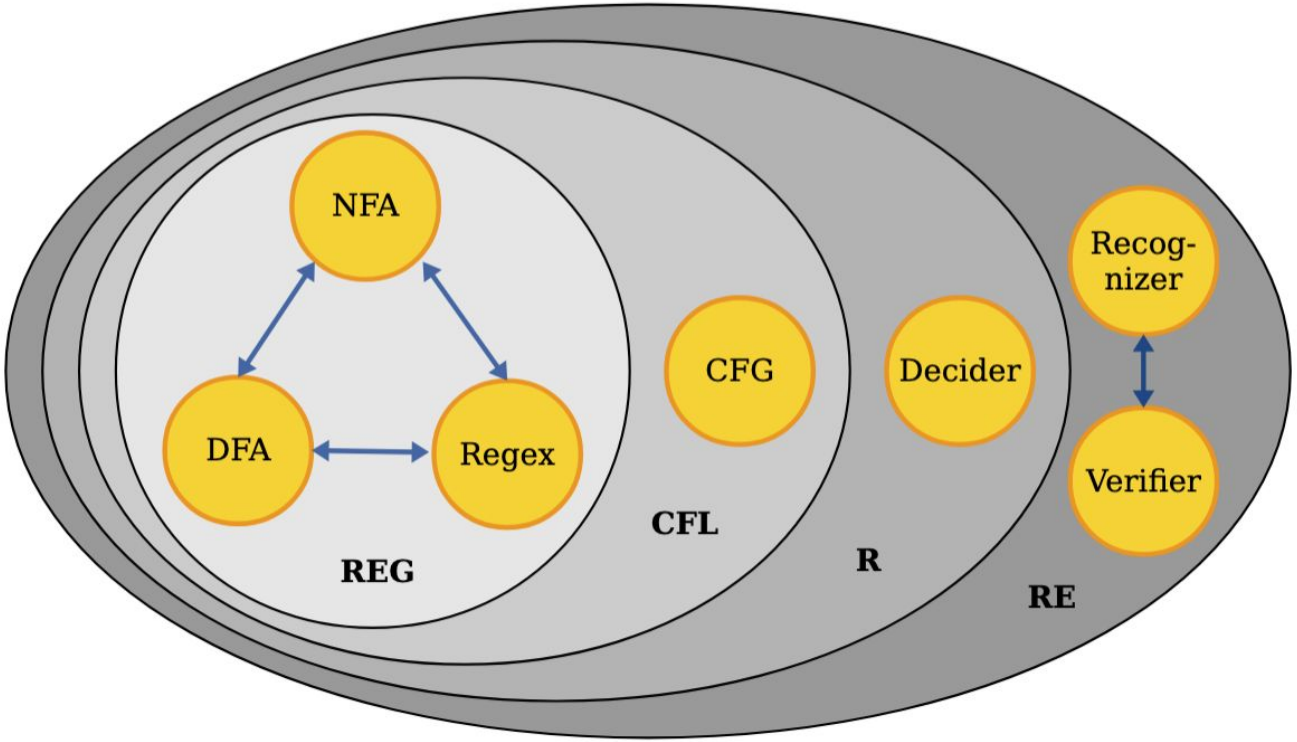
- V takes inputs of the form $\langle w, c \rangle$ where w is an input to L
- $\forall w \in \Sigma^*. (w \in L \rightarrow \exists c \in \Sigma^*. \text{V accepts } \langle w, c \rangle)$
- $\forall w \in \Sigma^*. (w \notin L \rightarrow \forall c \in \Sigma^*. \text{V rejects } \langle w, c \rangle)$

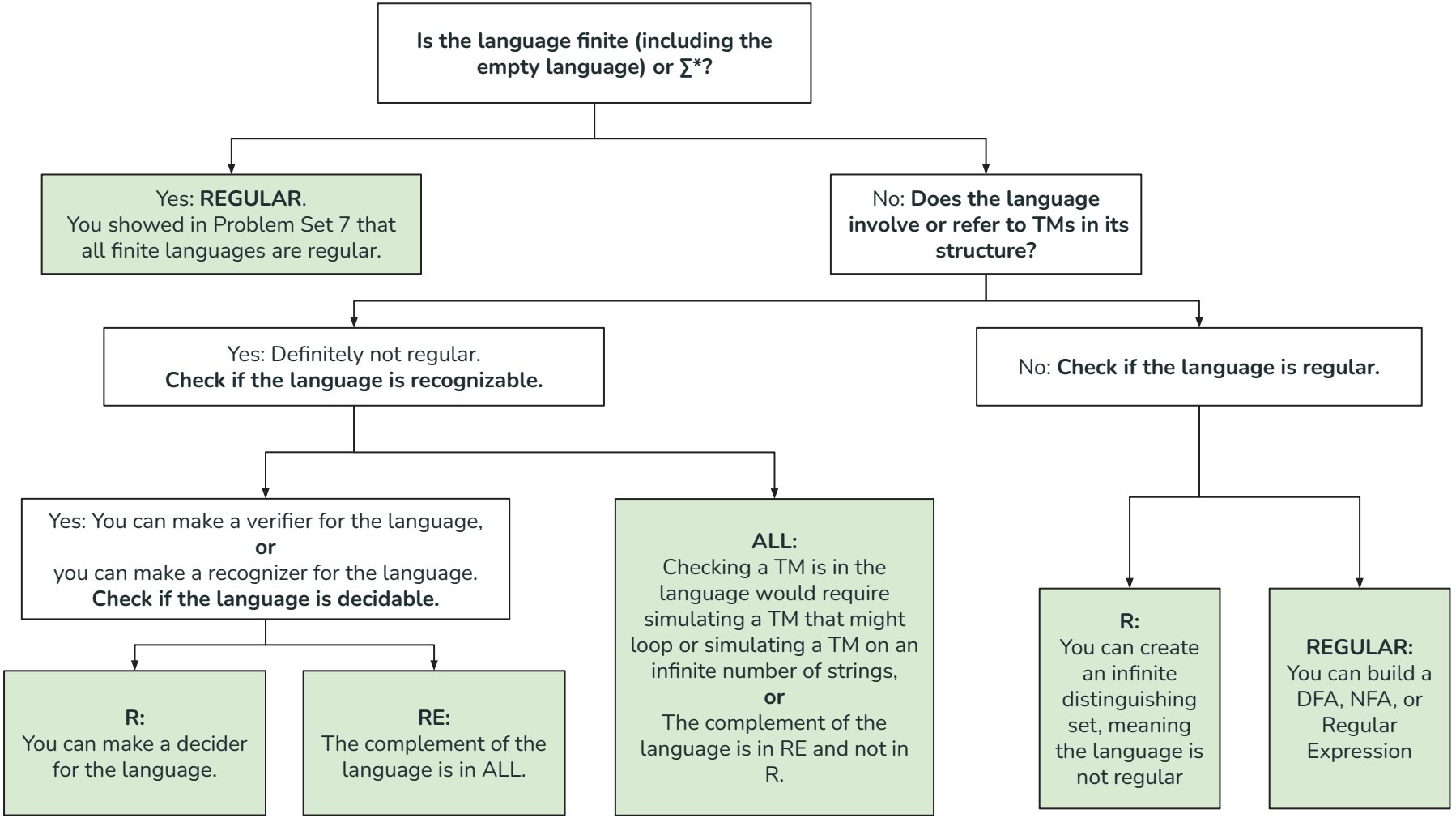
Intuitively, V “checks” whether a certain string is in the language, given some “proof” (the certificate c).

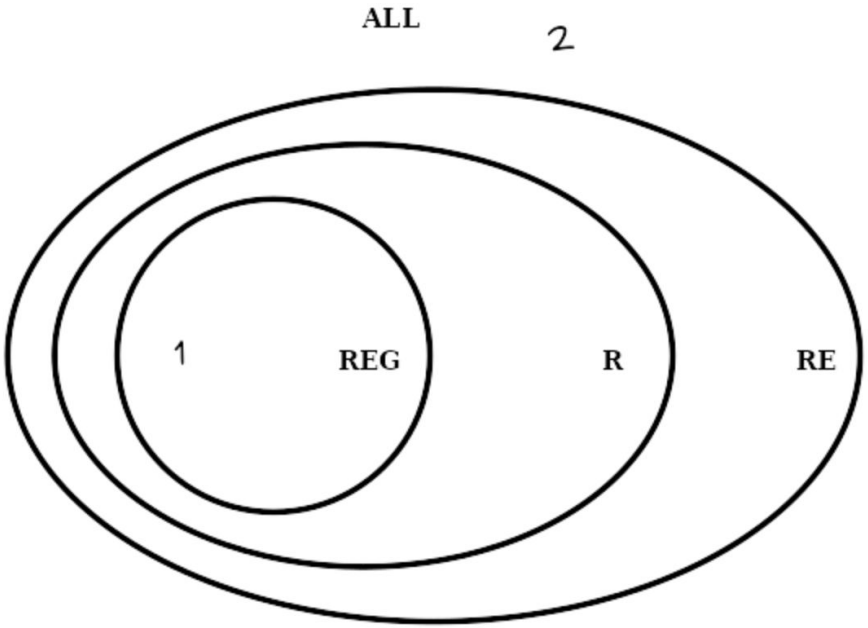
A language that has a verifier is recognizable.

Lava Diagram

Guide to the Lava Diagram Guide to Self-Reference




Is the language finite (including the empty language) or Σ*?
Yes: REGULAR.
You showed in Problem Set 7 that all finite languages are regular.
No: Does the language involve or refer to TMs in its structure?
Yes: Definitely not regular.
Check if the language is recognizable.
No: Check if the language is regular.
Yes: You can make a verifier for the language,
or
you can make a recognizer for the language.
Check if the language is decidable.
ALL:
Checking a TM is in the language would require simulating a TM that might loop or simulating a TM on an infinite number of strings,
or
The complement of the language is in RE and not in R.
R:
You can create an infinite distinguishing set, meaning the language is not regular
REGULAR:
You can build a DFA, NFA, or Regular Expression
R:
You can make a decider for the language.
RE:
The complement of the language is in ALL.



1. $\Sigma^\star$
2. L_D
3. $\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, and } M \text{ accepts } w\}$
4. $\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, and } M \text{ does not accept } w\}$
5. $\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, and } M \text{ recognizes } \{w\}\}$
6. $\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, and } M \text{ does not recognize } \{w\}\}$
7. $\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, } M \text{ accepts } w, \text{ and } M \text{ does not recognize } \{w\}\}$
8. $\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, } M \text{ does not accept } w, \text{ and } M \text{ recognizes } \{w\}\}$