Definição: uma máquina de Turing é M = (Σ, Q, Π, q0 , F, V, β,©), onde:
Σ é o alfabeto de símbolos de entrada (disjunto de {β,©} ∪ V
Q é o conjunto finito de estados
Π: (Q × S) ⇀ (Q × S × {Esquerda, Direita}) é uma função parcial denominada função de transição, onde S = Σ ∪ V ∪ {β,©}.
q0 ∈ Q é o estado inicial da máquina
F ⊆ Q é o conjunto de estados finais
V é o alfabeto auxiliar (disjunto de Σ ∪ {β,©})
β é o símbolo especial de espaço em branco
© é o símbolo especial de início de fita
Restrições para Π : (Q × S) ⇀ (Q × S × {E, D}):
• Π(q,©) = (q′ ,©, D) para quaisquer q, q′ ∈ Q
• se Π(q, a) = (q′ , b, x) e a ≠ ©, então b ≠ ©.
Podemos utilizar uma MT para determinar se w ∈ L ou w !∈ L (reconhecer linguagens). Aceita, Rejeita e Loop compunham os reconhecedores.

Definição: L ⊆ Σ∗ é uma linguagem enumerável recursivamente (linguagem semi-decidível) quando existe uma Máquina de Turing M que reconhece (aceita) L (Aceita(M) = L).
Definição: L ⊆ Σ∗ é uma linguagem recursiva (linguagem decidível) quando existe uma Máquina de Turing M que decide L (Aceita(M) = L, Rejeita(M) = Σ* - L , Loop(M)=vazio).
OBS: toda linguagem recursiva também é enumerável recursivamente.

Definição: uma função parcial f : Σ ∗ ⇀ Σ ∗ é chamada de **Turing-computável** se existe uma máquina de Turing M = (Σ, Q, Π, q0 , F, V, β,©) tal que ⟨M⟩ = f ou seja, ⟨M⟩(w) = f(w) para todo argumento w ∈ Σ ∗ .

Existem diversas formas diferentes de definir máquinas de Turing: • exatamente um estado de aceitação e um estado de rejeição • fita duplamente infinita • múltiplas fitas • múltiplos cursores de leitura-escrita por fita • fita multidimensional • cursor de leitura-escrita “eventualmente imóvel” • não-determinismo. (a maquina de Turing determinística de uma fita simula todas elas).
**Uma máquina não determinística apresenta mais de um comportamento possível para uma dada situação, havendo potencialmente diversas computações distintas para uma mesma entrada. A computação de uma máquina não determinística é uma árvore de configurações definidas a partir da configuração inicial.

Definição: um modelo de computação é denominado Turing-completo quando possui tanto poder computacional quanto a máquina de Turing (isto é, quando ele simula uma máquina de Turing).
Definição: um modelo de computação é denominado Turing-equivalente quando possui exatamente o mesmo poder que a máquina de Turing (simula e é simulado por ela).
Definição: uma máquina de Turing universal MTU é uma máquina de Turing que recebe como entrada uma codificação (M, w) onde: • M é uma máquina de Turing sobre o alfabeto Σ; • w ∈ Σ∗ é uma entrada para M (Uma Máquina de Turing é, na verdade, um programa para uma máquina universal.)

Tese (ou hipótese) de Church-Turing: A capacidade de computação representada por máquinas de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação. (Efetivamente computável = Turing-computável).

P é **decidível ou solucionável:** se existir alguma máquina de Turing M que, ao receber uma palavra W de entrada que codifica de forma válida uma instância de P, nunca entra em loop (tem um conjunto aceito e rejeita bem definidos).
P é **semi-decidível** se M aceita W se W codifica uma instância afirmativa de P e M rejeita W ou M entra em loop com W se W codifica uma instância negativa de P.

OBS: todo problema decidível também é semi-decidível, mas o inverso não é verdadeiro!

P é indecidível ou não solucionável se não existir máquina de Turing M que decide P.

Problemas indecidíveis:

- O problema da parada (PP) (indecidível) (semi-decidível)
- O problema da Aceitação (PA) (indecidível)
- O problema da Aceitação com palavra Vazia (PAPV) (indecidível)
- O problema da totalidade (PT) (indecidível)
- O problema da Parada com Entrada Vazia (PPEA) (indecidível)

O problema da aplicação (indecidível)
O problema da linguagem de aceitação vazia (PAV) (indecidível)
O problema das linguagens de aceitação iguais (indecidível)

Se P é decidível, então não P também é decidível
Se P e não P são semi-decidíveis, então P é decidível (logo, não P também é decidível).

Uma redução de um problema-fonte P para um problema-alvo Q, denotada por r: P => Q.
Uma redução é um mapeamento entre instâncias de problemas que PRESERVA A RESPOSTA para as perguntas dos problemas em questão.

Teorema: Sejam P e Q problemas de decisão. Se Q é decidível e existe uma redução r: P=>Q, então P é decidível.
Teorema: Sejam P e Q problemas de decisão. Se P é indecidível e existe uma redução r: P=>Q, então Q é indecidível.
OBS: não dá para descobrir **NADA sobre Q** se for construída uma redução g: L => Q, onde L é um problema DECIDÍVEL!!

TEOREMA DE RICE: Seja X uma propriedade de linguagens semi-decidível. Então, o Problema do Teste da Propriedade X é decidível se, e somente se, a propriedade X em questão é trivial.

Problemas **não triviais** de linguagens semi-decidível: •conter uma palavra cujo tamanho seja um número primo; •conter uma quantidade infinita de palavras; •conter uma palavra que seja um palíndromo
(ou seja, existe pelo menos uma linguagem semi-decidível que satisfaz tal propriedade e de que também existe pelo menos uma que não satisfaz).

Testar qualquer uma das propriedades de linguagens semi-decidível a seguir é um **problema indecidível**: • "conter uma palavra cujo tamanho seja um número primo" • "conter apenas uma quantidade finita de palavras" • "conter uma palavra que seja um palíndromo" • "ser uma linguagem regular" • "ser decidível" • "conter a palavra vazia"
ATENÇÃO: o teorema de Rice versa **apenas sobre propriedades de linguagens aceitas** por máquinas de Turing.

Escreva uma redução válida r : PAPV ⇒ PT.

PAPV	PT
Entrada = M	Entrada = M'
Pergunta = ε E A(M)?	Pergunta = ∀w E ∑*, w E (A(M') U R(M')) ?
ε E A(M) Sim	w E A(M') Sim
ε E R(M) Não	w E R(M') Sim
ε E L(M) Não	w E L(M') Não

r(M) = M' onde, primeiro a M' faz uma "rotina" para limpar a palavra w de entrada, assim toda palavra seria a palavra vazia. Além disso, é adicionado dois novos estados **Π**(q1,s) e **Π**(q2,s) na tabela verdade de Mm (uma M modificada - Mm), que vai ficar alternando entre eles para criar um loop em todos os lugares em "branco". Assim, quando M' simular Mm (que é o próximo passo após M' apagar w) com a entrada vazia, o que antes era rejeitado agora entra em loop.
Fazendo isso, preservamos as respostas para as perguntas, pois estamos preservando o não do conjunto rejeição da máquina M, transformando os rejeitas em M em loops em M', dando a resposta não no PT . O resto já está devidamente "conectado".

ACEITAÇÃO-IGUAIS Instância: Um par (M1, M2), onde M1 e M2 são máquinas de Turing Pergunta: É verdade que ACEITA(M1) = ACEITA(M2)?

r : ACEITAÇÃO-VAZIA ⇒ ACEITAÇÃO-IGUAIS. Considere o seguinte algoritmo r que, ao receber de entrada uma instância M de ACEITAÇÃO-VAZIA, retorna uma instância r(M) => (M, M') e de ACEITAÇÃO-IGUAIS, onde M' é qualquer máquina de Turing sem estados finais previamente escolhida, ou seja, M' é alguma máquina de Turing particular tal que ACEITA(M') => ∅ .
Resta agora provarmos que a redução r descrita está correta:
• Suponha que M ∈ A(ACEITAÇÃO-VAZIA). Então, ACEITA(M) = ∅. Portanto, (M, M') e ∈ A(ACEITAÇÃO-IGUAIS), pois, por construção, ACEITA(M') = e ∅
• Suponha que M ∈ R(ACEITAÇÃO-VAZIA). Então, ACEITA(M) != ∅ = ACEITA(M') e . Consequentemente, (M, M') e ∈ R(ACEITAÇÃO-IGUAIS)