

CAPÍTULO BASEADO NO MATERIAL DE CURSO DO PROF. STÉPHANE JULIA- FACOM-UFU

> Material gentilmente cedido pelo autor, coagido
por uma comunhão de bens ligada ao
matrimônio...

Bibliografia de base: Introduction à la
Calculabilité - Pierre Wolper - Ed. DUNOD

6. Problemas Indecidíveis (A Não Calculabilidade)

6.1 Introdução

- > Existem problemas (representados por certas linguagens) que não podem ser tratados por procedimentos efetivos (não decididos por uma TM).
- > O conjunto das TMs é enumerável, pois a héptupla que representa cada uma delas pode ser convertida em um programa (palavra ou sequência finita de caracteres) e o conjunto de programas é enumerável.
- > Mas, conforme demonstrando anteriormente no curso, o conjunto de todas as linguagens (problemas) não é enumerável !
- > Logo, existem mais linguagens do que TMs \implies Nem todas as linguagens podem ser decididas por uma TM !!

6.2 Provando a existência de linguagens indecidíveis

6.2.1 Classes de decidibilidade

Def: A classe de decidibilidade R (Recursiva) é o conjunto das linguagens decididas por uma TM.

Def: A classe de decidibilidade RE (Recursivamente Enumerável) é o conjunto das linguagens aceitas por uma TM.

> Se uma linguagem pertence à classe RE, então: existe uma TM que produz uma resposta positiva para as palavras desta linguagem e produz uma resposta negativa, ou um loop infinito, para as palavras que não pertencem à linguagem.

> Tal TM é então chamada de procedimento de decisão parcial.

OBSERVAÇÕES IMPORTANTES SOBRE LINGUAGENS PERTENCENTES À RE:

Conforme visto anteriormente, para qualquer linguagem L pertencente à RE, existe uma MT MT' que a aceita. Neste caso, a seguinte bi-implicação é verdadeira:

- **$\forall w, w$ é uma instância positiva de L se, e somente se, o processamento de w por MT' para e o resultado de tal processamento é “sim”;**

A Partir de tal bi-implicação, conclui-se o seguinte:

- ✓ Se uma palavra w pertence a L , O processamento de w por MT' SEMPRE vai parar, produzindo “sim” como resultado (pela relação de *suficiência* na implicação da esquerda para a direita);
- ✓ Se uma palavra w NÃO pertence a L , O processamento de w por MT' pode parar ou não (ou seja, pode entrar em “loop”). Caso MT' pare para tal w , o resultado do processamento será “não” (pela relação de *necessidade* na implicação da direita para a esquerda);
- ✓ Se o processamento por MT' de uma palavra w para e o resultado desse processamento é “sim”, então w pertence a L (pela relação de *suficiência* da implicação da direita para a esquerda);
- ✓ O fato de o processamento de uma palavra w por MT' ainda não ter parado em um dado momento t analisado (mesmo para um elevadíssimo valor de t), NÃO é suficiente para garantir que tal w não pertence a L , pois pode acontecer de a palavra processada pertencer a L , contudo, não ter havido tempo hábil ainda para se concluir o referido processamento;
- ✓ Caso se prove, de alguma forma, que o processamento de uma dada palavra w por MT' DEFINITIVAMENTE NÃO PARA (OU SEJA, MT' , DE FATO, ENTRA EM “LOOP” AO PROCESSAR w), conclui-se que w NÃO pertence a L (pois o “loop” viola a relação de *necessidade* expressa na implicação da esquerda para a direita).

Lema: A classe R é contida na classe RE

Prova: consequência direta das definições das classes R e RE .

6.2.2 Exemplo de linguagem indecidível que não pertence à classe RE : técnica da diagonalização

> As palavras finitas assim como as TM são enumeráveis.

> Existe então uma matriz infinita A onde cada linha "i" é rotulada por uma TM "Mi" e cada coluna "j" por uma palavra "Wj".

> Os elementos da matriz são assim definidos:

$A[M_i, w_j] = S$ (sim) se a TM "Mi" aceita a palavra "wj";

$A[M_i, w_j] = N$ (não) se a TM "Mi" não aceita a palavra "wj" (ou seja, ao processar "wj", "Mi" produz "não" ou entra em loop infinito).

A	w_0	w_1	w_2	w_3	...	w_j ...
M_0	S	N	N	S
M_1	N	N	S	S
M_2	S	S	N	N
M_3	-	-	-	-	-	-
⋮	-	-	-	-	-	-
M_i	N	N	S	N
⋮						

> Considere-se a linguagem L_0 definida por:

$$L_0 = \{w / w=w_i \text{ e } A[M_i, w_i]=N\},$$

composta por cada palavra " w_i " que não seja aceita por sua respectiva " M_i " (ou seja, por cada " w_i ", tal que $A(i,i) = N$ (não), tais como w_1 e w_2 na matriz exemplo A);

> L_0 não pertence à classe RE (e consequentemente não pertence também à classe R - linguagem não decidida por uma TM).

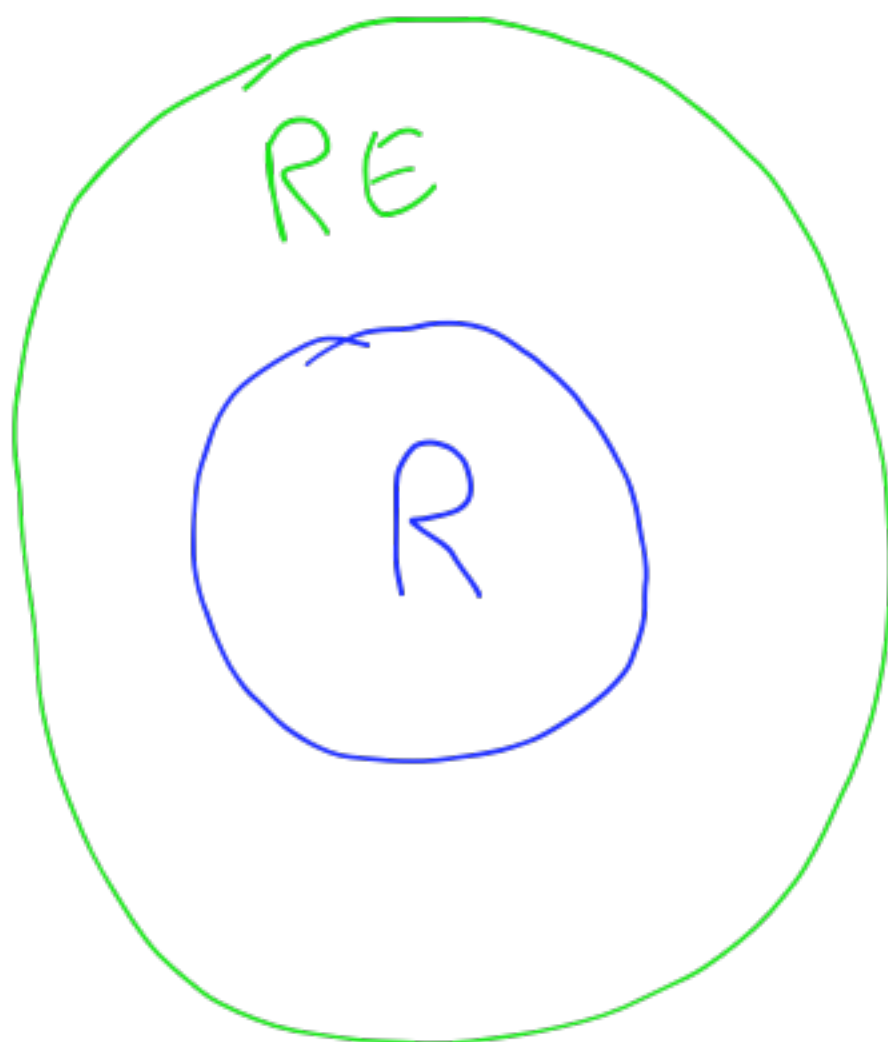
> Prova por contradição (ou absurdo):

Tenta-se provar que: $L_0 \in RE$

- > Se $L_0 \in RE$, então existe uma TM "Mk" que aceita L_0 .
- > A matriz infinita A representa todas as TMs existentes.
- > Para Mk, só existem duas possibilidades:
 - 1) Se $A[Mk, w_k] = S$, então Mk aceita uma palavra que não pertence à $L_0 \Rightarrow$ contradição
 - 2) Se $A[Mk, w_k] = N$ então Mk não aceita uma palavra que pertence à $L_0 \Rightarrow$ contradição
- > Logo, não existe uma MT que aceite L_0 ;

Teorema: A linguagem L_0 definida por $L_0 = \{w / w = w_i \text{ e } A[M_i, w_i] = N\}$ não pertence à classe RE, sendo, portanto, indecidível.

6.2.3 Exemplo de linguagem indecidível que pertence à classe RE



Lema 1: O complemento de uma linguagem da classe R é também uma linguagem da classe R

Se uma linguagem L pertence à R é porque existe uma TM M (onde Q :conjunto de estados; F : estados de aceitação de M) que a decide.

Podemos construir uma TM M' de tal modo que as respostas de M sejam invertidas (ou seja, conjunto de estados de $M' = Q$, e $F' = Q - F$);

A TM M' decide então o complemento de L .

Em particular, M' sempre para e aceitará todas as palavras que não pertencem a L (e somente elas).

Lema 2: Se uma linguagem L e seu complemento pertencem à classe RE, então tanto L quanto o seu complemento pertencem a R.

Prova: Precisamos provar que $L \in R$ e usar o Lema anterior.

Podemos construir uma TM M que simula em paralelo a execução de M_L (TM que aceita L) e de $M_{\bar{L}}$ (TM que aceita complemento de L).

M executa alternadamente uma etapa de M_L e uma etapa de $M_{\bar{L}}$ (TM de 2 fitas por exemplo).

Tal máquina M decide a linguagem L , pois:

- ou ela aceita uma palavra através da aceitação de M_L
- ou ela rejeita uma palavra através da aceitação de $M_{\bar{L}}$

Lema 3: A linguagem cL_0 (complemento da $L_0 = \{w / w=w_i \text{ e } A[M_i, w_i]=N\}$ indecidível e não pertencente à classe RE apresentada na seção 6.2.2) definida por:

$$cL_0 = \{w / w=w_i \text{ e } A[M_i, w_i] = S\}$$

pertence à classe RE (note que, na matriz exemplo A apresentada em 6.2.2, w_0 , por exemplo, pertence a cL_0).

Prova: Mostrar que existe uma TM M que aceita cL_0 .

Tal máquina M tem o seguinte funcionamento ao processar uma palavra w:

- 1) M percorre e enumera as palavras da primeira linha da matriz A da seção 6.2.2 até encontrar w, determinando o índice i da palavra w_i encontrada (sendo $w_i = w$);
- 2) M percorre e enumera as MTs da primeira coluna de A até encontrar a TM de mesmo índice que w_i , ou seja, M_i ;
- 3) M simula a execução da palavra w_i por M_i ;
- 4) M aceita w se, e somente se, M_i tiver aceitado w_i no passo 3.

Teorema: A linguagem cL_0 definida por :

$cL_0 = \{w / w = w_i \text{ e } A[M_i, w_i] = S\}$ é indecidível, apesar de pertencer à RE.

Prova: tal linguagem não pertence a R porque se fosse o caso, o seu complemento (L_0) pertenceria também à classe R (Lema 1) e já foi provado que L_0 não pertence à R (primeira linguagem indecidível).

6.2.4 Técnica da Redução: Outro Método de análise de Indecidibilidade de Linguagens

Conhecendo as duas linguagens indecidíveis L_0 e cL_0 apresentadas nas seções 6.2.2 e 6.2.3, respectivamente, o objetivo é de mostrar a indecidibilidade de outras linguagens (problemas concretos) através da técnica da redução.

Técnica da Redução ?

Queremos mostrar a indecidibilidade de uma linguagem L_2 sabendo da indecidibilidade de uma outra linguagem L_1 .

O princípio é baseado num raciocínio por contradição.

1) Mostrar que se existe um algoritmo que decide L2, então existe também um algoritmo que decide L1.

Para isso, produzir um algoritmo P1 (uma TM que sempre para) que decide L1 usando como sub-programa um algoritmo P2 que decide L2.

O algoritmo P1 é chamado de Redução de L1 a L2.

2) Se L2 é decidível então L1 é também decidível já que L1 pode ser reduzido a L2.

Como L1 é indecidível, recai-se em uma contradição que mostra que L2 também não é decidível.

Exemplo: A linguagem universal LU definida por $LU = \{ \langle M, w \rangle \mid M \text{ aceita } w \}$ é indecidível.

Tal linguagem é composta por todas as cadeias de caracteres que representam uma TM M e uma palavra w tais que M aceita w.

Trata-se de um tipo de TM Universal (uma TM que simula uma outra TM), conforme visto na seção 5.8.

Para provar que LU (L2) não pertence à classe R, podemos fazer uma redução a partir de cL0 (L1)

Suponha-se que existe um algoritmo (MT universal) que decide LU. Logo, pode-se usar tal algoritmo como um sub-programa apto a decidir cL0.

Como já foi provado que cL0 não pertence à R, teremos uma contradição. De fato:

$$cL0 = \{ w \mid w = w_i \text{ e } A[M_i, w_i] = S \}$$

> A redução é dada pelo seguinte algoritmo:

1) Dada uma palavra w , determinar na matriz A da seção 6.2.2 o índice i tal que $w = w_i$;

2) Determinar na matriz A a TM M_i ;

3) Aplicar o sub-programa que decide LU à palavra $\langle M_i, w_i \rangle$.

- se o resultado é sim, então a palavra w é aceita

- se o resultado é não, então a palavra w é rejeitada.

> Tal algoritmo decidiria então $cL0$, o que representaria uma contradição, já que foi demonstrado que $cL0$ não pertence à R .

> Logo, LU é indecidível (não pertence à R).

> Note que LU pertence à RE, pois existe uma TM Universal que simula o comportamento de M em w e que ACEITA $\langle M, w \rangle$ quando M aceita w .

> Mostrar que cLU não pertence à RE ! (Sugestão: consulte os lemas demonstrados neste capítulo).

6.2.5 Problema da Parada: provando pela Técnica da Redução que o problema (linguagem) da parada é indecidível

Tal problema consiste em Determinar se uma TM que recebe uma palavra de entrada w para.

Tal problema prático (detecção de loop infinitos) é indecidível !

Formalmente, temos de mostrar que a linguagem:

$H = \{ \langle M, w \rangle \mid M \text{ para ao processar } w \}$ não pertence à R.

Prova: aplicar uma redução da linguagem indecidível LU estudada na sub-seção 6.2.4 à linguagem H.

Suponha-se então que H é decidível.

Redução (algoritmo):

1) Aplicar o sub-programa que decide a linguagem H à palavra $\langle M, w \rangle$

2) Se o sub-programa produzir uma resposta negativa (provando assim que a TM M NÃO para ao processar a palavra w, ou seja, que M, de fato, entra em "loop" em tal processamento), isso permitirá concluir que $\langle M, w \rangle$ NÃO é uma instância de LU (ou seja, que M não aceita w) - conforme explicado nas observações relativas à bi-implicação referente às linguagens RE apresentada na seção 6.2.1;

3) Se o sub-programa que decide H produzir uma resposta positiva (indicando que a TM M para no processamento de w), então simular M sobre w.

- se M produzir um "sim", então $\langle M, w \rangle$ é uma instância de LU (M aceita w) (conforme previsto nas mesmas observações citadas no item "2" acima);

- Se M produzir um "não", então $\langle M, w \rangle$ não é uma instância de LU (pois M estará em um estado final que não é de aceitação, ainda conforme previsto nas mesmas observações citadas no item "2" acima);

> Logo, o algoritmo que hipoteticamente decide H também permitiria decidir LU, o que é uma contradição, já que foi demonstrado que LU não pertence à R.

6.2.6. Exemplos de Problemas práticos Indecidíveis

Exemplo 1: O problema da validade do cálculo dos predicados é indecidível. Tal problema consiste em determinar se uma fórmula do cálculo dos predicados é verdadeira em toda e qualquer interpretação possível.

As provas para tais problemas práticos são também baseados na técnica da Redução (ou, eventualmente, em certos casos, na técnica da diagonalização).

Exemplo 2: O décimo problema de Hilbert é indecidível. Tal problema consiste em determinar se a equação:

$$P(x_1, x_2, \dots, x_n) = 0,$$

onde $P(x_1, x_2, \dots, x_n)$ é um polinômio a coeficientes inteiros, tem uma solução nos inteiros \mathbb{Z} .

Por exemplo: $x^2 - 4 = 0$ tem uma solução inteira;

$x^2 - 2 = 0$ não tem solução inteira.

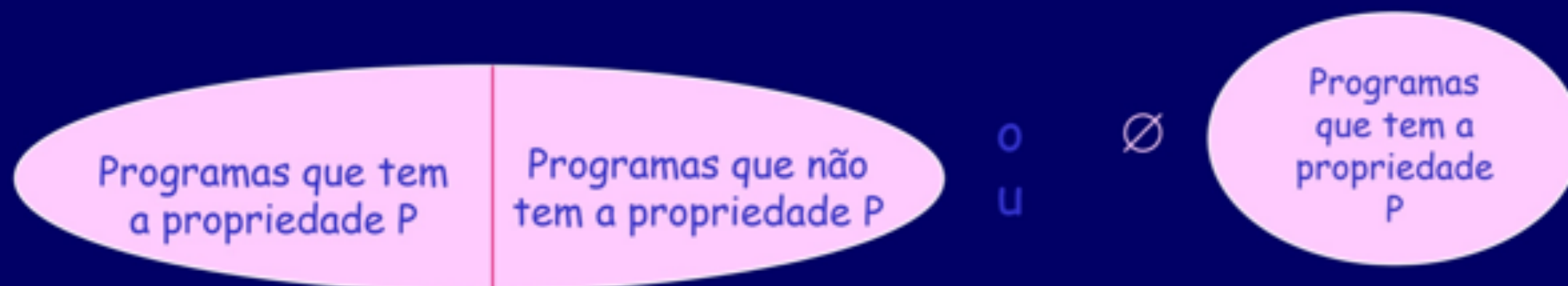
É interessante ver que quando se considera uma única instância do problema, é possível geralmente encontrar uma solução. O que não se consegue é encontrar uma solução genérica para qualquer instância do problema.

O EXEMPLO 3 SEGUINTE FOI EXTRAÍDO
DE UM MATERIAL DO ICMC/USP
DISPONÍVEL EM:

https://pt.m.wikipedia.org/wiki/Teorema_de_Rice

3) Usando o Teorema de Rice para Provar Indecidibilidade

- Este teorema mostra que a indecidibilidade do Problema da Parada não é um fenômeno isolado!
- Teo: Se A é uma **propriedade extensiva não-trivial** de programas, então A é indecidível.
- Uma **propriedade** de programas é especificada pela divisão do mundo dos programas em duas partes:



- Uma propriedade **não-trivial** é uma que é satisfeita por pelo menos um, mas não todos os programas.
- Uma propriedade **extensiva** (externa) depende exclusivamente do **comportamento de entrada e saída do programa**,
 - e assim é independente da aparência, tamanho e outras propriedades chamadas de **intensivas** (interna).
- Exemplo: autoria, local e data da publicação são critérios externos de um **TEXTO**, sendo que o assunto e o tipo de texto são critérios internos.

AS PROPRIEDADES P1, P2 e P3 mostradas a seguir são exemplos de propriedades EXTENSIVAS E NÃO TRIVIAIS que são INDECIDÍVEIS, segundo o teorema de Rice.

- Prop P1: O número de variáveis no programa P é maior que 100?
- Prop P2: Uma MT M tem mais que 7 estados?
 - Vejam que nesses 2 casos falo da máquina propriamente dita.
- Prop P3: Uma MT M roda em tempo polinomial?

(aqui falo do comportamento e é não-trivial: pois existem linguagens decidíveis por MT em tempo exponencial de vários graus e outras nem sabemos se estão em P)

Portanto, não existem Checadores de Complexidade!!!
Vejam que aqui falo de comportamento de um algoritmo.

Quase todas as propriedades da linguagem de MT são indecidíveis:

$\text{Regular}_{\text{TM}} = \{ \langle M \rangle \mid L(M) \text{ é uma linguagem regular} \}$

$\text{Finita}_{\text{TM}} = \{ \langle M \rangle \mid L(M) \text{ é uma linguagem finita} \}$

$\text{LC}_{\text{TM}} = \{ \langle M \rangle \mid L(M) \text{ é uma linguagem LC} \}$

Teorema de Rice nos diz que não
podemos prever os aspectos
interessantes do comportamento de
um algoritmo de forma algorítmica!

6.3 Interpretação da indecibilidade

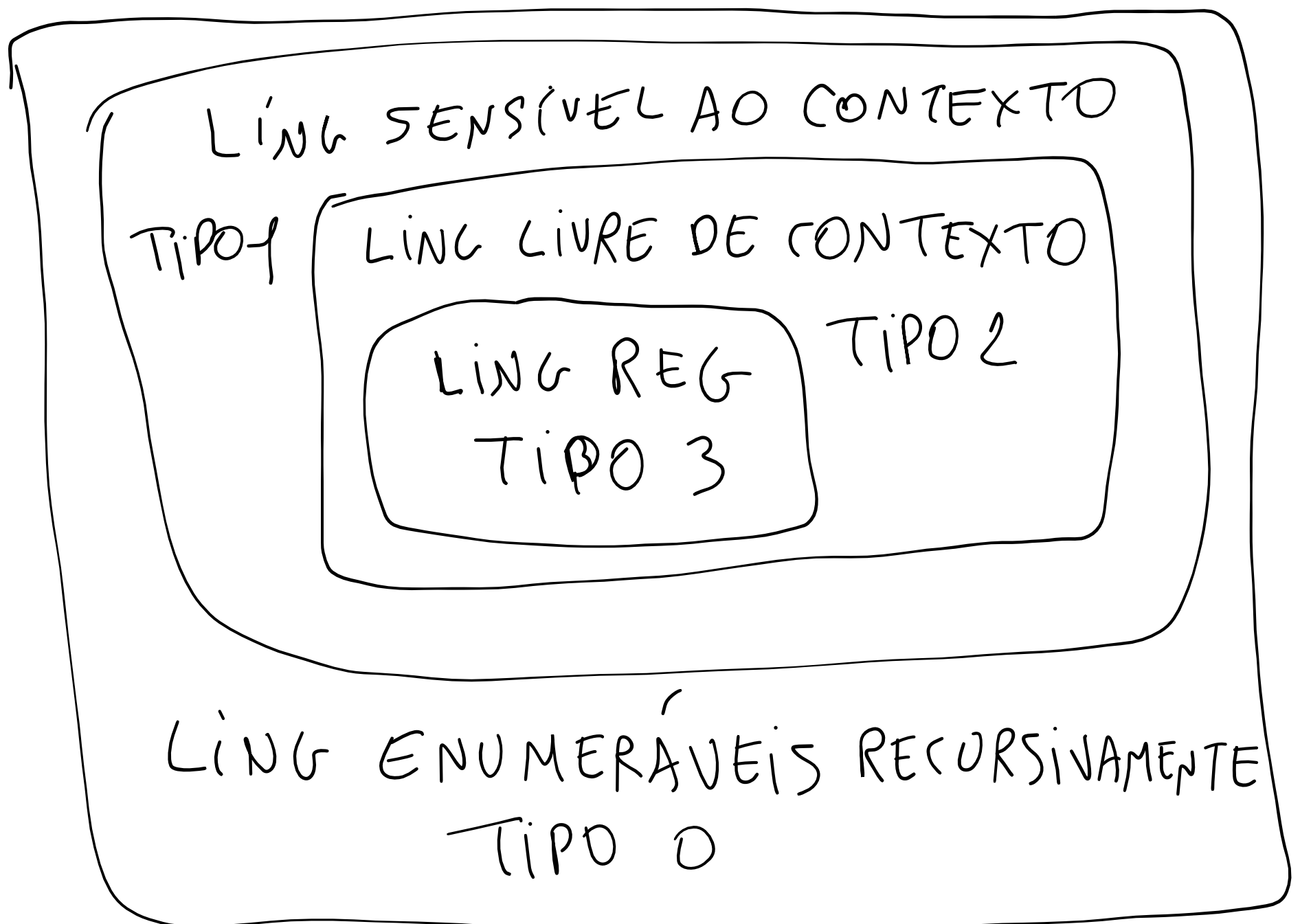
A indecibilidade está relacionada com problemas que tenham uma infinidade de instâncias.

A indecibilidade não impede a elaboração de algoritmos usados para tratar instâncias (ou conjuntos finitos de instâncias) específicas .

Um exemplo prático é o problema da prova de teorema: apesar de o problema de um provador geral ser indecidível, teoremas específicos são provados regularmente. Existem algoritmos para provar teoremas específicos, mas não existe um único algoritmo para provar qualquer teorema !

6.4 Conclusão : Classificação das Linguagens

Hierarquia de Chomsky baseada em gramáticas geradoras : o objetivo de tal classificação era o mero estudo das linguagens - inclusive das linguagens naturais, no caso das linguagens sensíveis ao contexto e enumeráveis recursivamente (construção de frases) - sem se preocupar com a noção de procedimento efetivo:



Analizando a computabilidade das linguagens definidas por Chomsky:

- > As linguagens Regulares (tipo 3) são decididas por autômatos finitos determinísticos;
- > As linguagens livres de contexto (tipo 2) são aceitas por um autômato à pilha não determinístico;
- > As linguagens sensíveis ao contexto (tipo 1) são aceitas por autômatos com bordas lineares, que são MTs não determinísticas em que se limita a utilização da fita de modo que o tamanho dela que pode ser usado em cada processamento seja limitado a uma função linear do tamanho da palavra de entrada;
- > As linguagens Gerais (Tipo 0) são aceitas por MTs.

MOSTRA-SE A SEGUIR UMA NOVA
DEFINIÇÃO DE HIERARQUIA ENTRE
LINGUAGENS QUE É ALTERNATIVA
ÀQUELA PROPOSTA POR CHOMSKY,
SENDO DEFINIDA DE FORMA A SE
ALINHAR À NOÇÃO DE PROCEDIMENTO
EFETIVO:

Hierarquia baseada nas Máquinas de Turing e na noção de procedimento efetivo (aqui a classificação se baseia na resolução de problemas em computação):



EM TAL HIERARQUIA:

> O conjunto das Linguagens Recursivas (decididas por MTs determinísticas) tem, dentre seus subconjuntos:

- As Linguagens Regulares (decididas por autômatos finitos determinístico);

- As Linguagens Livres de Contexto (decididas pelos autômatos à pilha determinísticos usados nos Compiladores, que NÃO são equivalentes aos autômatos à pilha não determinístico);

> As Linguagens Recursivamente Enumeráveis são semi-decidíveis, ou seja, são ACEITAS por MTs determinísticas.