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 conjuntos de todas as linguagens (problemas) não é enumerável **!**

> Logo, existem mais linguagens do que TMs ==> 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:

- **∀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[Mi, wj] = S (sim) se a TM "Mi" aceita a palavra "wj";

A[Mi, wj]= 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 L0 definida por:

L0 = {w/ w=wi e A[Mi,wi]=N},

composta por cada palavra "wi" que não seja aceita por sua respectiva "Mi" (ou seja, por cada "wi", tal que A(i,i) =N (não), tais como w1 e w2 na matriz exemplo A);

> L0 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: L0 $\in$ RE

> Se L0 ∈ RE, então existe uma TM "Mk" que aceita L0.

> A matriz infinita A representa todas as TMs existentes.

> Para Mk, só existem duas possibilidades:

1) Se A[Mk,wk]=S, então Mk aceita uma palavra que não pertence à L0 ⇒ contradição

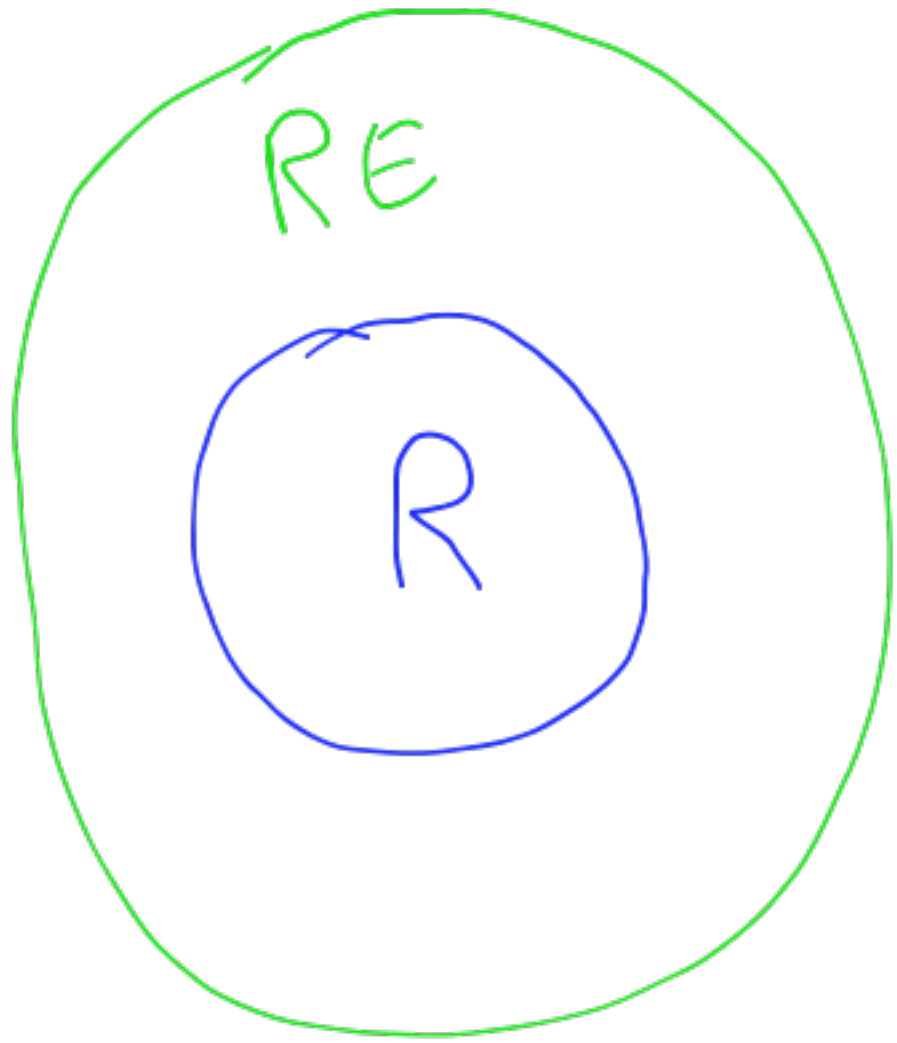
2) Se A[Mk,wk]=N então Mk não aceita uma palavra que pertence à L0 ⇒ contradição

> Logo, não existe uma MT que aceite L0, conforme figura do próximo "slide".

Teorema: A linguagem L0 definida por L0 = {w/ w=wi e A[Mi, wi]= N} não pertence à classe RE, sendo, portanto, indecidível.

A	w_0	w_1	w_2	w_3	...	w_j	w_K
M_0	S	N	N	S	...	S	N
M_1	N	N	S	N	...	S	N
M_2	S	S	N	N	...	N	S
M_3	N	N	S	S	..	S	N
⋮	...	..	..	..	..	..	...
M_j	N	N	S	S	..	N	S
M_k ⋮	N	S	S	N	...	S	S? N?

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 ∈ R e usar o Lema anterior.

Podemos construir uma TM M que simula em paralelo a execução de ML (TM que aceita L) e de McL (TM que aceita complemento de L).

M executa alternadamente uma etapa de ML e uma etapa de McL (TM de 2 fitas por exemplo).

Tal máquina M decide a linguagem L, pois:

- ou ela aceita uma palavra através da aceitação de ML

- ou ela rejeita uma palavra através da aceitação de McL

Lema 3: A linguagem cL0 (complemento da
L0 = {w/ w=wi e A[Mi,wi]=N}indecidível e não pertencente à classe RE apresentada na seção 6.2.2) definida por:

cL0 = {w/ w=wi e A[Mi , wi] = S}

pertence à classe RE (note que, na matriz exemplo A apresentada em 6.2.2, w0, por exemplo, pertence a cL0).

Prova: Mostrar que existe uma TM M que aceita cL0.

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 wi encontrada (sendo wi = w);
2) M percorre e enumera as MTs da primeira coluna de A até encontrar a TM de mesmo índice que wi, ou seja, Mi;
3) M simula a execução da palavra wi por Mi;
4) M aceita w se, e somente se, Mi tiver aceitado wi no passo 3.

Teorema: A linguagem cL0 definida por :

cL0 = {w/ w=wi e A[Mi , wi]= S} é indecidível, apesar de pertencer à RE.

Prova: tal linguagem não pertence a R porque se fosse o caso, o seu complemento (L0) pertenceria também à classe R (Lema 1) e já foi provado que L0 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 L0 e cL0 apresentadas nas seções 6.2.2 e 6.2.3, respectivamente, o objetivo é de mostrar a indecibilidade de outras linguagens (problemas concretos) através da técnica da redução.

Técnica da Redução **?**

Queremos mostrar a indecibilidade de uma linguagem L2 sabendo da indecibilidade de uma outra linguagem L1.

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={<M,w> | M 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 uma 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:

$$cL0=\{w \,/\, w = wi \text{ e } A[Mi, wi] = S\}$$

> 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 = wi ;

2) Determinar na matriz A a TM Mi ;

3) Aplicar o sub-programa que decide LU à palavra <Mi,wi>.

- 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 <M,w> quando M aceita w.

> Mostrar que cLU não pertence à RE ❗ *(Sugestão: consulte os lemas demontrados 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 (deteção de loop infinitos) é indecidível ❗

Formalmente, temos de mostrar que a linguagem:

H={<M,w> | M 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 <M,w>

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 <M,w> 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 <M,w> é uma instância de LU (M aceita w) (conforme previsto nas mesmas observações citadas no item "2" acima);

- Se M produzir um "não", então <M,w> 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.

EXEMPLOS DE ALGUMAS VARIANTES DA INDECIDIBILIDADE DO PROBLEMA DA PARADA:

> O problema de determinar se um programa escrito em uma linguagem de programação para ao executar qualquer valor fixado para suas entradas é indecidível (provado por redução sobre o problema da parada);

> O problema de determinar se uma MT para ao executar a palavra vazia (problema da parada sobre a palavra vazia) é indecidível (provado por redução sobre o problema da parada);

> O problema de determinar se uma MT para ao executar alguma palavra de entrada (problema da parada existencial) é indecidível (provado por redução sobre o problema da parada sobre a palavra vazia;

> O problema de determinar se uma MT para ao executar qualquer palavra de entrada (problema da parada universal) é indecidível (provado por redução sobre o problema da parada sobre a palavra vazia).

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(x1,x2,...,xn) = 0,

onde P(x1,x2,...,xn) 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.

6.2.7. Complemento da Análise de Indecidibilidade nas Linguagens RE :

> Uma propriedade de uma linguagem RE é simplesmente um conjunto de linguagens REs. Logo, a propriedade de ser uma linguagem livre de contexto é representada por um conjunto S composto por todas as linguagens livres de contexto. A propriedade de ser a linguagem vazia é representada, então, por um conjunto S igual a $\{ \emptyset \}$. Note que tal definição é extensional.

> Uma propriedade P é trivial se o conjunto S que a representa é vazio (ou seja, P não é satisfeita por linguagem alguma) ou, então, se S é composto por todas as linguagens REs;

> Note que uma propriedade "vazia", $\emptyset$ (trivial), difere da propriedade não trivial "ser vazia" $\{ \emptyset \}$.

> Não se pode reconhecer um conjunto de linguagens da mesma forma com que se reconhecem as linguagens que os compõem. Motivo: linguagens típicas, sendo infinitas, não podem ser escritas como uma string finita (palavra) que pode ser uma entrada de uma MT;

> Neste caso, devem-se reconhecer, ao invés dessas linguagens, as MTs que as aceitam (cujos códigos são finitos, ainda que as linguagens que aceitam não o sejam);

> Logo, se P é uma propriedade das linguagens REs, a linguagem L-P é o conjunto de códigos das MTs Mi, tais que L(Mi) é uma linguagem em P. Neste caso, falar em decidibilidade de uma propriedade P se refere à decidibilidade da linguagem L-P.

TEOREMA DE RICE NAS LINGUAGENS REs:

> Analisemos agora algumas propriedades que se referem EXCLUSIVAMENTE às próprias linguagens aceitas por MTs, ou seja, propriedades das linguagens REs que são independentes das próprias máquinas que as aceitam. Logo, estão excluídas propriedades que não são uniformemente verdadeiras ou falsas para todas as MTs que aceitem uma mesma linguagem (ex: excluem-se aqui propriedades não triviais das linguagens REs tais como "a MT que aceita a linguagem tem 75 estados" - as quais, apesar de serem não triviais, são decidíveis). Excluamos, também, as propriedades triviais (ou seja, as propriedades que são verdadeiras para todas as linguagens RE ou para nenhuma). Temos, assim, o seguinte teorema:

> Teorema de Rice nas linguagens REs: Toda propriedade não trivial das linguagens REs (considerando as restrições acima) é indecidível.

Forma útil de enunciar o Teorema de Rice em TC:

> Sendo P uma propriedade não trivial das linguagens REs, isso significa que: há um conjunto de linguagens REs que satisfazem P, pertencendo, assim , ao conjunto S que representa P; há linguagens REs que não pertencem a S (todas as remanescentes que não satisfazem P); para cada linguagem que pertence a S ou que não pertence a S, existe uma MT que a aceita;

> Enunciado alternativo do Teorema de Rice: o problema de determinar se a linguagem aceita por uma MT arbitrária encontra-se em S é indecidível. Na prática, isso significa que não existe uma MT capaz de decidir sempre se uma linguagem aceita por uma MT arbitrária satisfaz uma propriedade não trivial;

> O Teorema NADA afirma sobre propriedades que não se refiram, EXCLUSIVAMENTE, às funções ou às linguagens. Logo, por exemplo, o Teorema não se aplica às seguintes propriedades NÃO TRIVIAIS : 1) "A MT que aceita a linguagem gasta mais do que 100 passos para operar uma dada entrada" ; 2) " A MT que aceita a linguagem tem 10 estados" (inclusive, no caso deste último exemplo, a propriedade não trivial é decidível).

Exemplos de problemas indecidíveis (em conformidade com o Teorema de Rice) ligados aos conjuntos recursivamente enumeráveis (linguagens REs):

> Determinar se a linguagem aceita por uma MT é vazia (problema da linguagem aceita vazia) é indecidível. A prova é por redução sobre a linguagem cLU (complemento da linguagem universal apresentado em 6.2.4);

> Determinar se a linguagem aceita por uma MT é regular (problema da linguagem aceita regular) é indecidível. A prova é feita por redução a partir de cLU (complemento da linguagem Universal);

> Determinar se uma linguagem aceita por uma MT é decidível é indecidível (problema da linguagem aceita indecidível). A prova é feita por redução a partir de cLU.

> Como as linguagens RE são uma fonte abundante de problemas indecidíveis, apresentaremos, a seguir, 3 alternativas distintas para caracterizá-las (além da estudada que consiste no fato de serem aceitas por uma MT). As linguagens REs são :

1- As linguagens calculadas por uma MT;

2- As linguagens geradas por uma gramática;

3- As linguagens enumeradas por um procedimento efetivo.

> Os próximos "slides" esclarecem um pouco mais cada uma dessas três caracterizações.

1- As REs são linguagens calculadas por uma MT:

> Definição: seja uma MT M que para sobre uma palavra u de entrada. Denotemos por f-M(u) a palavra calculada por M ao processar u. A linguagem calculada por M é, então:

{ w / existe u tal que M para sobre u e w = f-M(u) }

> Teorema: uma linguagem é calculada por uma MT se, e somente se, ela é RE.

2- As REs são linguagens geradas por uma gramática

> Existe uma ligação estreita entre os diferentes tipos de autômatos e as gramáticas de Chomsky: as gramáticas regulares (tipo 3) correspondem aos autômatos finitos; as gramáticas livres de contexto (tipo 2) correspondem aos autômatos à pilha; as gramáticas gerais (tipo 0) correspondem às MTs;

> Teorema: uma linguagem é gerada por uma gramática se, e somente se, ela é RE.

OBS: é natural de se perguntar se existe uma classe de autômatos que aceitam exatamente as linguagens sensíveis ao contexto (geradas pelas gramáticas do tipo 1). Tal classe existe e é denominada "autômatos de bordas lineares", que são MTs não determinísticas para as quais se limita a utilização da fita. Em tais autômatos, existem constantes k1 e k2 tais que, em toda a execução de uma palavra w de entrada, à maquina não utiliza nunca mais do que k1 x |w| + k2 casas de sua fita, ou seja, a fração utilizada da fita é uma função linear do tamanho da palavra de entrada.

3- As linguagens RE são linguagens enumeradas por um procedimento efetivo.

> O termo "recursivamente enumerável" sugere a possibilidade de uma enumeração por um procedimento efetivo. De fato, para toda linguagem RE existe um procedimento efetivo (MT M) que gera sucessivamente as palavras dessa linguagem.

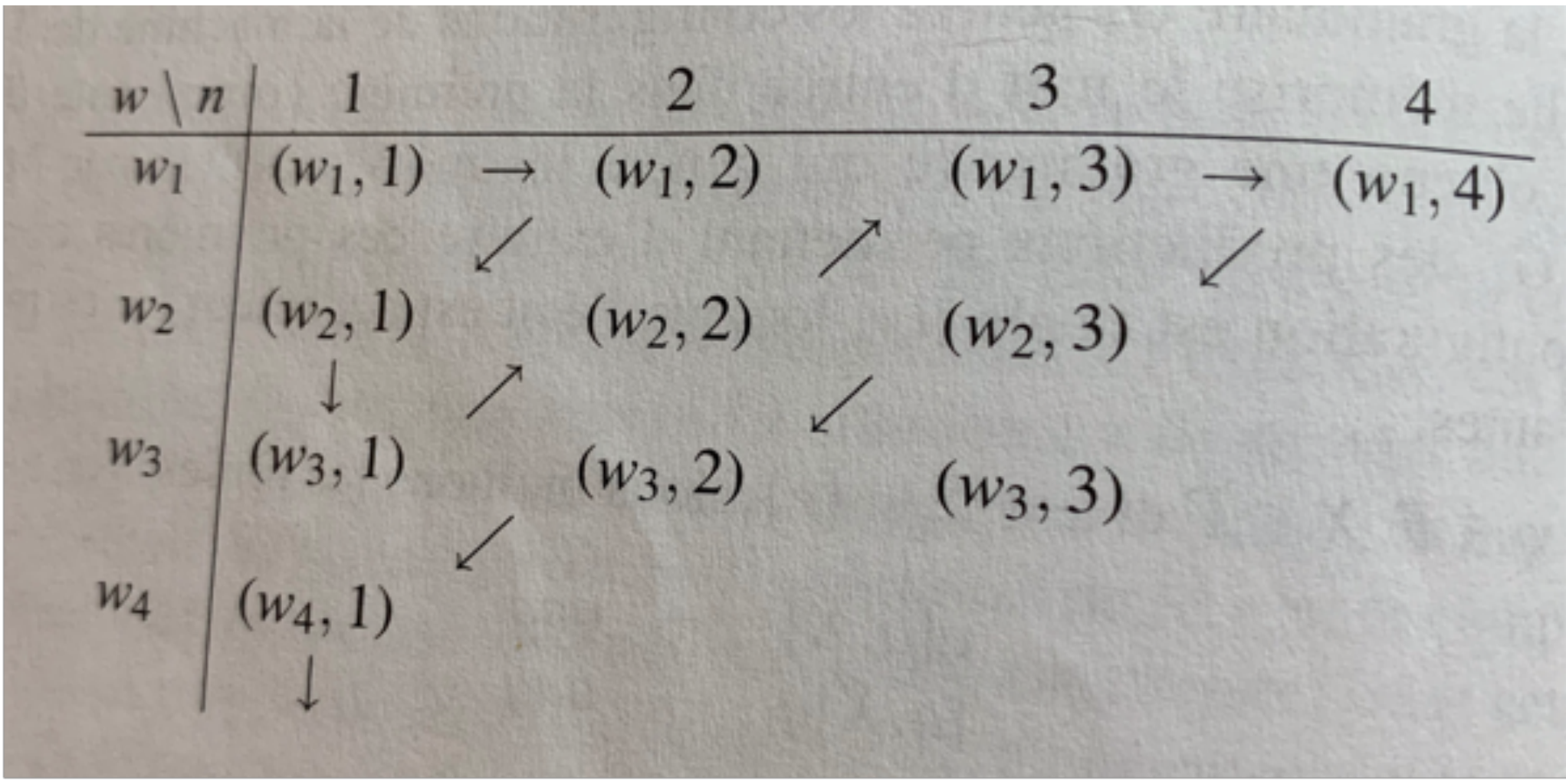
> uma primeira ideia intuitiva (e incorreta!) para enumerar as palavras aceitas por M seria:

- Gerar todas as palavras por ordem lexicográfica e de comprimento crescente;
- Simular M sobre cada nova palavra gerada e conservá-la, unicamente, se ela for aceita por M;

> A proposta acima é incorreta pois a enumeração será interrompida na primeira palavra sobre a qual M tem uma execução infinita.

> A seguir, mostramos uma alternativa correta de enumeração:

> Sejam pares (w, n) onde w é uma palavra e n um inteiro natural. Como os conjuntos de palavras e dos naturais são ambos enumeráveis, o conjunto dos pares (w, n) também o é, conforme tabela abaixo:



> A enumeração efetiva das palavras aceitas por M pode ser feita pelo seguinte algoritmo:

- Para cada par (w, n), na ordem de suas enumerações, faça:

    - Simule a execução de M sobre w limitando a execução, contudo, a n etapas;
        - Se tal execução aceita w, produza w

    - Passe ao par (w, n) seguinte, seguindo a ordem da numeração.

> Note no algoritmo que, no processo de numeração, uma mesma palavra pode ser simulada diversas vezes, até que, caso pertença à linguagem, seja produzida ou enumerada. Contudo, como a ordem de simulação dos pares segue a ordem crescente da tabela, as simulações de uma mesma palavra podem ser intercaladas com as simulações de outras palavras.

> Note na tabela anterior apresentada que para cada palavra w há infinitas alternativas de pares, cada um deles estabelecendo uma dada quantidade n de etapas no processo de simulação do algoritmo apresentado (onde n cresce na ordem da numeração indicada na tabela).

> Assim sendo, cada palavra pertencente à linguagem seria simulada por meio de diversos pares (cada um deles permitindo uma quantidade maior de etapas), até que se simule um par em que ela seja aceita.

> Além disso, o algoritmo jamais vai numerar uma palavra não aceita pela linguagem, pois as suas simulações continuariam indefinidamente, em quantidades crescentes de etapas.

> Note que este último procedimento de enumeração não enumera as palavras aceitas por M em ordem lexicográfica nem em qualquer outra ordem natural, o que, aliás, seria impossível. De fato, suponha que exista tal tipo de enumeração. Nesse caso, para decidir se uma palavra w pertence à linguagem aceita por M, bastaria seguir a numeração até encontrar a primeira palavra que sucede w lexicograficamente (ou na ordem natural adotada). Se, neste momento, w ainda não tiver aparecido, w não pertence à linguagem aceita por M. Isso nos daria um procedimento de decisão para toda linguagem aceita por uma MT, o que demonstramos não existir.

6.2.8. Exemplos de Problemas Indecidíveis Relativos às Gramáticas:

> O fato de as linguagens geradas pelas gramáticas coincidirem com as linguagens REs permite demonstrar a indecidibilidade de problemas relativos às gramáticas, tais como:

- O problema de determinar se uma palavra w pertence à linguagem gerada por uma gramática G (problema da pertinência à linguagem de uma gramática) é indecidível. A prova é feita por redução a partir do problema LU;

- O problema de determinar se duas gramáticas G1 e G2 geram a mesma linguagem (problema da igualdade de gramáticas) é indecidível. A prova é feita por redução a partir do problema da pertinência à linguagem de uma gramática.

6.2.9. Teorema de Rice Aplicado aos programas:

> Na ótica dos programas, o enunciado do Teorema de Rice garante que são indecidíveis as propriedades que dependem exclusivamente das entradas e das saídas dos programas e que são não triviais, ou seja, propriedades para as quais há programas que as satisfazem e há programas que não as satisfazem. Exemplos de funções computáveis indecidíveis:

- A classe de funções computáveis que retornam zero para cada entrada, bem como seu complemento;

- A classe de funções computáveis que retornam zero para pelo menos uma entrada, bem como seu complemento;

- A classe de funções computáveis que são constantes, bem como seu complemento.

> De um modo prático, o Teorema de Rice afirma que não há algoritmos capazes de predizer aspectos interessantes de algoritmos.

6.3 Interpretação da indecibilidade

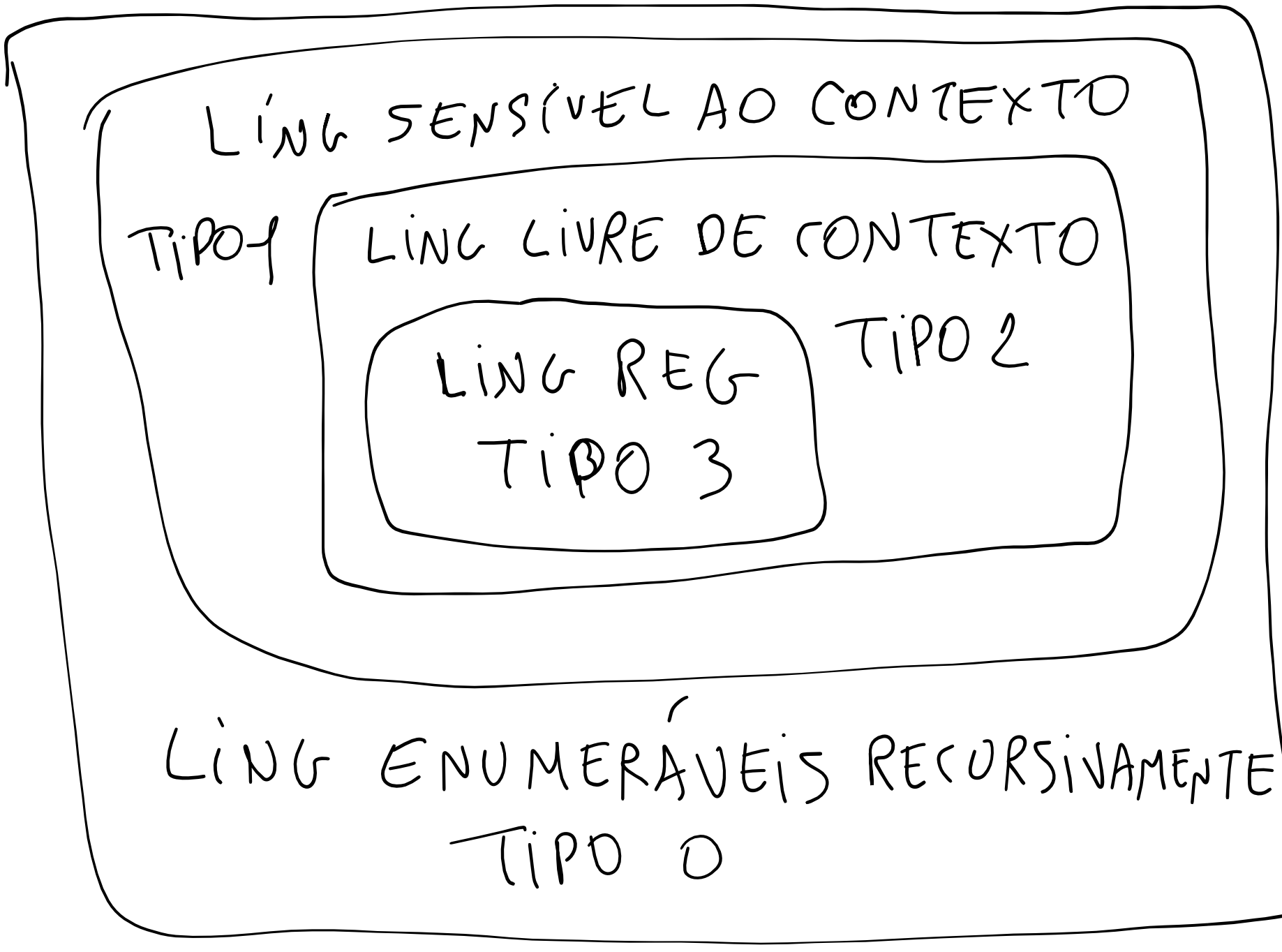
A indecibilidade está relacionada com problemas que tenham uma infinidade de instâncias.

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:



Analisando 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 Máquinqs 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.