Decidibilidade

Preâmbulo

Objetivos

- O que os computadores podem fazer?

ou Quais linguagens podem ser definidas por qualquer dispositivo computacional?

- Lembre-se que reconhecer cadeias de uma linguagem é um modo formal de expressar problemas, e resolver problemas é uma representação daquilo que os computadores podem fazer.

Problemas Indecidíveis

- São aqueles que não podemos resolver usando um computador.
- AlanTuring propôs um formalismo (a Máquina de Turing) que é um modelo preciso daquilo que qualquer dispositivo físico de computação é capaz de fazer.
- A MT foi usada para desenvolver uma teoria da indecidibilidade.

Por que problemas indecidíveis têm que existir

- "Problema" = pertinência de uma cadeia a uma linguagem.
- O nro. de linguagens distintas sobre qualquer alfabeto não é enumerável (ou contável).
- Programas são cadeias finitas sobre um alfabeto finito; portanto são enumeráveis (contáveis).
- Então há infinitamente menos programas que problemas.
- Se escolhêssemos uma linguagem ao acaso, quase certamente ela seria um problema indecidível.
- Como examinamos em geral problemas simples, eles são quase sempre decidíveis.
- Mas mesmo problemas simples podem ser indecidíveis, como veremos a seguir.

Um conjunto é contável se ele é ou finito ou tem o tamanho dos Naturais

Uma introdução informal sobre problemas indecidíveis

- Considere o famoso programa C "hello, world":

```
main()
{
        printf("hello, world\n")
}
```

Vamos definir o *problema hello, world* como: descobrir se um dado programa em C, com uma determinada entrada, imprime **hello, world** como os 12 primeiros caracteres que ele imprime.

- Há programas, menos óbvios, que também podem imprimir `hello, world`.
- P.ex., um programa que procura todas as triplas de inteiros (x,y,z) tal que

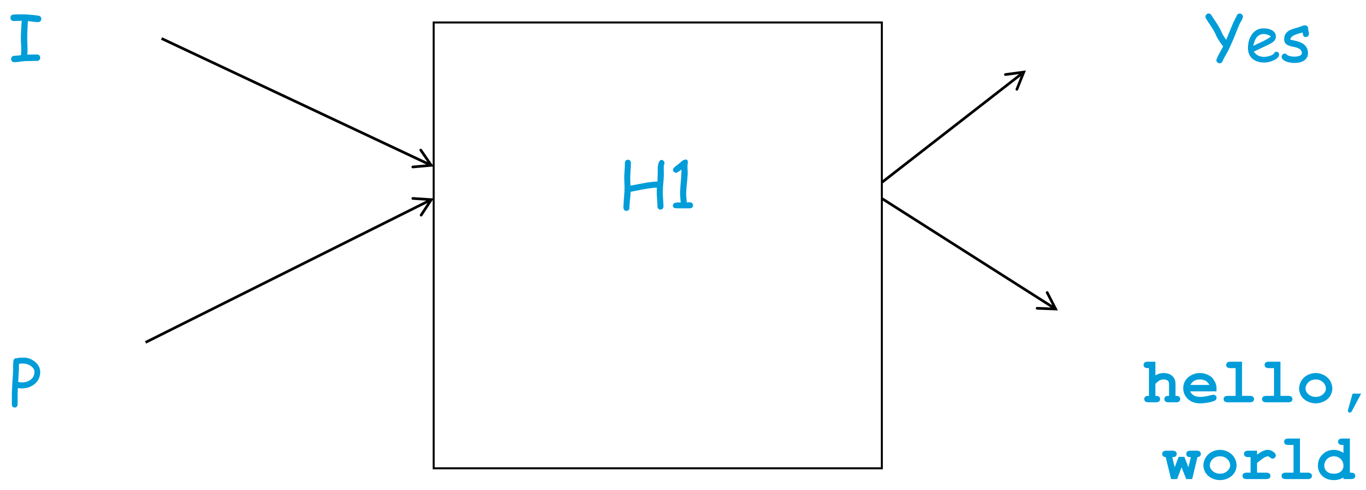
  $x^n + y^n = z^n$, para um dado n, e a cada resultado, imprime `hello, world`.

- Há programas, menos óbvios, que também podem imprimir `hello, world`.
- P.ex., um programa que procura todas as triplas de inteiros (x,y,z) tal que $x^n + y^n = z^n$, para um dado n, e a cada resultado, imprime `hello, world`.
- Sabe-se, no entando, que $x^n + y^n = z^n$ sse $n \leq 2$ (último Teorema de Fermat)
- Assim, o programa só irá imprimir `hello, world` para n=1 ou 2. Para outros valores de n, o programa nunca encontrará solução e não imprimirá `hello, world`.

- Vamos mostrar que não pode existir um programa H tal que, dado um programa qualquer P, e uma entrada I para P, informe se P, executado com I, imprime **`hello, world`**.

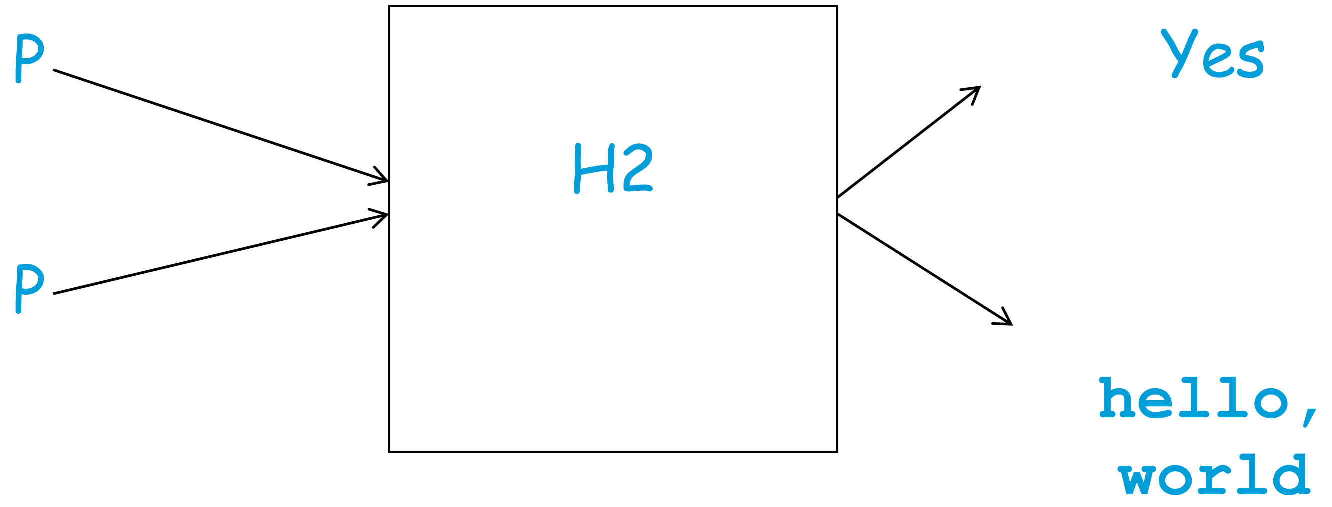
Se existir H para P, então P é decidível, caso contrário, P é indecidível.

- Vamos provar, por contradição, que H não pode existir.
- Para isso, faremos algumas transformações em H, que mantêm como questionável apenas a existência de H.



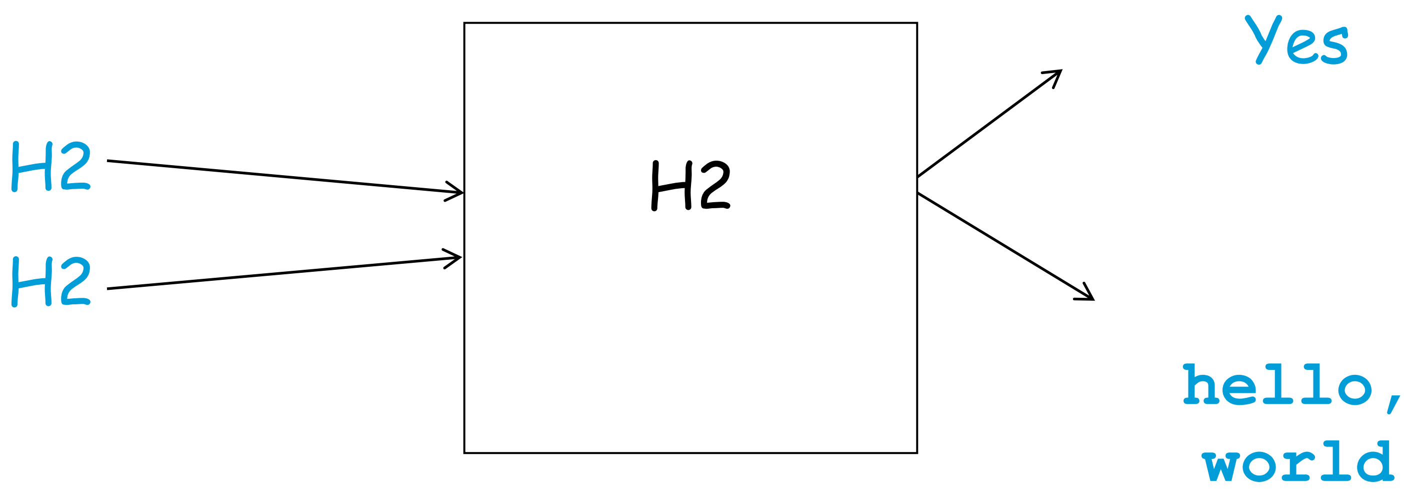
H1 se comporta como H, mas imprime **hello, world** no lugar de No.

- Agora suponha que a entrada I seja o próprio código P.



Agora podemos imaginar o que H2 faria quando sua entrada fosse o próprio H2.

Mostraremos que H2 não pode existir. E, portanto, tampouco podem existir H1 e H.



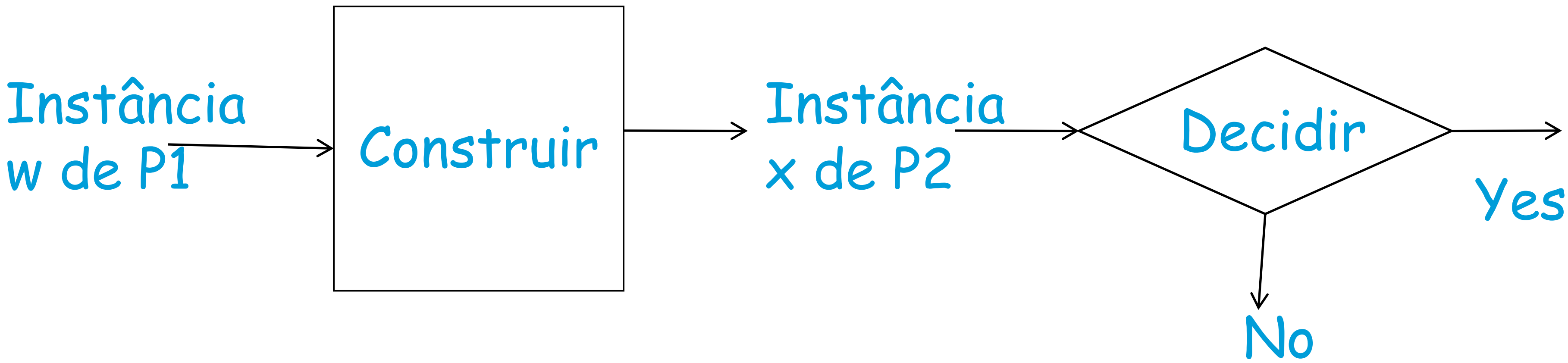
• Suponha que H2 gera a saída Yes. Então ele está informando que sua entrada H2, recebendo a si mesmo como entrada, imprime **hello, world** como sua primeira saída. Todavia, acabamos de supor que H2 gera nessa situação a saída Yes, em vez de **hello, world**.

• Ao supor, por outro lado, que a saída do bloco é **hello, world**, informamos que H2, recebendo a si mesmo como entrada, gera como primeira saída **hello, world**, logo a saída do bloco deveria ser Yes, e não **hello, world** como supusemos.

- Essa situação é paradoxal, e concluímos que H2 não pode existir. Consequentemente, H não pode existir.
- Isto é, provamos que nenhum programa H pode saber se um dado programa P com entrada I imprime ou não **`hello, world`** como sua primeira entrada.
- Repare que, ao invés da condição de "imprime ou não **`hello, world`** como sua primeira entrada", podemos pensar em qualquer outra.
- O Problema da Parada: é decidível saber se um programa qualquer P sempre pára com uma entrada I?

Redução de um problema a outro para mostrar indecidibilidade

- Se sabemos que P1 é indecidível, e queremos mostrar que P2 é indecidível, podemos tentar:
- reduzir P1 a P2 e,
- se pudéssemos resolver P2 (ou seja, se P2 fosse decidível), então poderíamos usar essa solução para resolver P1.
- Mas como P1 é indecidível, então P2 não pode ser decidível.



A construção do bloco deve converter instâncias de P1 em instâncias de P2 que têm a mesma resposta. E:

1. Dada uma instância de P1, ou seja, uma dada cadeia w que pode ou não estar na linguagem P1, aplique o algoritmo de construção para produzir uma cadeia x.

2. Teste se x está em P2 e dê a mesma resposta sobre w e P1.

   Se w está em P1, então x está em P2, e assim esse algoritmo imprime Yes. Se w não está em P1, então x não está em P2, e o algoritmo imprime No. Ou seja, ele decide P2 e P1. Como P1 é sabidamente indecidível, então temos uma prova por contradição de que o algoritmo de decisão para P2 não pode existir; isto é, P2 é indecidível.

Exemplo

- Vamos mostrar que a pergunta “o programa Q, dada a entrada y, chamará a função foo?” é indecidível.
- Se Q não tem uma função foo, a resposta é direta.
- Se Q tem a função, ele pode ou não chamá-la para a entrada y.
- Seja P1 o problema de hello, world (indecidível), e P2 o problema de chamar foo.
- Supomos que existe um programa que decide P2.

Exemplo

- Nosso trabalho é projetar um algoritmo que reduza P1 a P2, ou seja, que converta o problema de hello, world ao problema de chamar foo.
- Ou seja, dado o programa Q e sua entrada y (P1), devemos construir um programa R e uma entrada z (P2) tais que R, com entrada z, chame foo se e somente se Q com a entrada y imprimir hello, world:

1. Se Q tem uma função chamada foo, renomeie essa função e todas as chamadas a ela. É claro que o novo programa Q1 faz exatamente o que Q faz.
2. Adicione a Q1 uma função foo. Essa função não faz nada e não é chamada. O programa resultante é Q2.
3. Modifique Q2 para memorizar os 12 primeiros caracteres que ele imprime, armazenando-os num vetor global A. Esse programa é o Q3.
4. Modifique Q3 de forma que, sempre que executar qualquer instrução de saída, ele verifique em seguida no vetor A se escreveu 12 caracteres ou mais e, se for o caso, se hello, world são esses 12 caracteres. Nesse caso, chame a nova função foo que foi adicionada no item (2). O programa resultante é R e a entrada z é igual a y.

- Agora assuma que Q, com entrada y, imprima hello, world como sua primeira saída. Então R chamará foo. Porém, se Q com entrada y não imprimir hello, world como primeira saída, R nunca chamará foo. Se pudermos descobrir se R com a entrada z chama foo, então também saberemos se Q com a entrada y (y=z) imprime hello, world.
- Mas sabemos que P1 é indecidível, portanto R não pode existir e P2 é indecidível.

Decidibilidade e Intratabilidade

- Distinguir problemas indecidíveis é importante também para orientar programadores sobre o que podem fazer via programação.
- No entanto, alguns problemas, embora decidíveis, exigem tempo demais para sua resolução. São chamados "intratáveis", e mais do que os indecidíveis, são enfrentados diariamente e apresentam muitos desafios.
- Precisamos, assim, de ferramentas que nos ajudem a decidir se um problema é indecidível ou intratável e o que fazer nesse último caso.

Indecidibilidade e Máquinas de Turing

- A máquina de Turing é um modelo de um computador muito simples, que é essencialmente um AF que tem uma única fita de comprimento infinito na qual ele pode ler e gravar dados.
- Essa simplicidade possibilita provar, no entanto, vários resultados sobre decidibilidade