Indecidibilidade

Rodrigo Gabriel Ferreira Soares

DEINFO - UFRPE

Julho, 2014

Motivação

- O que pode ser computado? E mais intrigantemente, o que não pode ser computado?

Motivação

- O que pode ser computado? E mais intrigantemente, o que não pode ser computado?
- Apresentamos modelos que decidem, semidecidem ou geram linguagens e computam funções.

Motivação

- O que pode ser computado? E mais intrigantemente, o que não pode ser computado?
- Apresentamos modelos que decidem, semidecidem ou geram linguagens e computam funções.
- Nosso estudo ateh agora, sugere que encontramos o limite superior natural para a capacidade que podemos esperar de um dispositivo computacional.

Motivação

- O que pode ser computado? E mais intrigantemente, o que não pode ser computado?
- Apresentamos modelos que decidem, semidecidem ou geram linguagens e computam funções.
- Nosso estudo ateh agora, sugere que encontramos o limite superior natural para a capacidade que podemos esperar de um dispositivo computacional.
- MTs que decidem linguagens e computam funções são algoritmos.

Motivação

- O que pode ser computado? E mais intrigantemente, o que não pode ser computado?
- Apresentamos modelos que decidem, semidecidem ou geram linguagens e computam funções.
- Nosso estudo ateh agora, sugere que encontramos o limite superior natural para a capacidade que podemos esperar de um dispositivo computacional.
- MTs que decidem linguagens e computam funções são algoritmos.
- A MT que para em resposta a todas as entradas eh a noção formal precisa a intuição sobre o que eh um **algoritmo**.

Motivação

- O que pode ser computado? E mais intrigantemente, o que não pode ser computado?
- Apresentamos modelos que decidem, semidecidem ou geram linguagens e computam funções.
- Nosso estudo ateh agora, sugere que encontramos o limite superior natural para a capacidade que podemos esperar de um dispositivo computacional.
- MTs que decidem linguagens e computam funções são algoritmos.
- A MT que para em resposta a todas as entradas eh a noção formal precisa a intuição sobre o que eh um **algoritmo**.

Tese de Church-Turing

- Tese de Church-Turing: Nada serah considerado como algoritmo se não puder ser expresso na forma de uma MT, cuja parada eh garantida em resposta a todas as possíveis entradas, e todas elas serão corretamente classificadas como algoritmos.

Tese de Church-Turing

- Tese de Church-Turing: Nada serah considerado como algoritmo se não puder ser expresso na forma de uma MT, cuja parada eh garantida em resposta a todas as possíveis entradas, e todas elas serão corretamente classificadas como algoritmos.
- Não sendo um teorema, essa tese não pode ser provada.

Tese de Church-Turing

- Tese de Church-Turing: Nada serah considerado como algoritmo se não puder ser expresso na forma de uma MT, cuja parada eh garantida em resposta a todas as possíveis entradas, e todas elas serão corretamente classificadas como algoritmos.
- Não sendo um teorema, essa tese não pode ser provada.
- Ter uma noção matemática rigorosa para algoritmos leva a questão de se provar que certos problemas computacionais não podem ser resolvidos por algoritmos.

Tese de Church-Turing

- Tese de Church-Turing: Nada serah considerado como algoritmo se não puder ser expresso na forma de uma MT, cuja parada eh garantida em resposta a todas as possíveis entradas, e todas elas serão corretamente classificadas como algoritmos.
- Não sendo um teorema, essa tese não pode ser provada.
- Ter uma noção matemática rigorosa para algoritmos leva a questão de se provar que certos problemas computacionais não podem ser resolvidos por algoritmos.
- Se cadeias de símbolos são usadas para representar linguagens, nem toda linguagem pode ser representada, pois enquanto existe somente uma quantidade enumeravelmente infinita de cadeias sobre um alfabeto, por outro lado uma quantidade infinita, mas não enumerável, de linguagens.

Tese de Church-Turing

- Tese de Church-Turing: Nada serah considerado como algoritmo se não puder ser expresso na forma de uma MT, cuja parada eh garantida em resposta a todas as possíveis entradas, e todas elas serão corretamente classificadas como algoritmos.
- Não sendo um teorema, essa tese não pode ser provada.
- Ter uma noção matemática rigorosa para algoritmos leva a questão de se provar que certos problemas computacionais não podem ser resolvidos por algoritmos.
- Se cadeias de símbolos são usadas para representar linguagens, nem toda linguagem pode ser representada, pois enquanto existe somente uma quantidade enumeravelmente infinita de cadeias sobre um alfabeto, por outro lado uma quantidade infinita, mas não enumerável, de linguagens.

Tese de Church-Turing

- Dispositivos atuais podem apenas decidir e semidecidir uma fração infinitesimal do conjunto de todas as linguagens sobre qualquer alfabeto.

Tese de Church-Turing

- Dispositivos atuais podem apenas decidir e semidecidir uma fração infinitesimal do conjunto de todas as linguagens sobre qualquer alfabeto.
- De acordo com a Tese de Church-Turing, tarefas computacionais que não podem ser executadas por MTs são impossíveis, desanimadoras, **indecidíveis**.

Maquinas de Turing universais

- Qual seria a base da computação? o hardware ou o software?

Maquinas de Turing universais

- Qual seria a base da computação? o hardware ou o software?
- MT eh uma peca de hardware não programável, especializada em resolver um problema em particular, usando instruções pré-fabricadas e imutáveis.

Maquinas de Turing universais

- Qual seria a base da computação? o hardware ou o software?
- MT eh uma peca de hardware não programável, especializada em resolver um problema em particular, usando instruções pré-fabricadas e imutáveis.
- Mas MTs também podem ser softwares.
- Uma certa MT de uso geral pode ser programada.

Maquinas de Turing universais

- Qual seria a base da computação? o hardware ou o software?
- MT eh uma peca de hardware não programável, especializada em resolver um problema em particular, usando instruções pré-fabricadas e imutáveis.
- Mas MTs também podem ser softwares.
- Uma certa MT de uso geral pode ser programada.
- O formalismo das MTs pode ser interpretado como uma linguagem de programação.

Maquinas de Turing universais

- Qual seria a base da computação? o hardware ou o software?
- MT eh uma peca de hardware não programável, especializada em resolver um problema em particular, usando instruções pré-fabricadas e imutáveis.
- Mas MTs também podem ser softwares.
- Uma certa MT de uso geral pode ser programada.
- O formalismo das MTs pode ser interpretado como uma linguagem de programação.
- Eh baseado na ideia de que um programa escrito em uma certa linguagem pode ser usado para interpretar/compilar um programa escrito naquela mesma linguagem.

Maquinas de Turing universais

- Qual seria a base da computação? o hardware ou o software?
- MT eh uma peca de hardware não programável, especializada em resolver um problema em particular, usando instruções pré-fabricadas e imutáveis.
- Mas MTs também podem ser softwares.
- Uma certa MT de uso geral pode ser programada.
- O formalismo das MTs pode ser interpretado como uma linguagem de programação.
- Eh baseado na ideia de que um programa escrito em uma certa linguagem pode ser usado para interpretar/compilar um programa escrito naquela mesma linguagem.

Maquinas de Turing universais

- Vamos propor uma forma geral para a especificação de MTs, de modo que suas descrições possam ser usadas como entradas para outras MTs.

Maquinas de Turing universais

- Vamos propor uma forma geral para a especificação de MTs, de modo que suas descrições possam ser usadas como entradas para outras MTs.
- Precisaremos codificar estados e símbolos como cadeias sobre um alfabeto fixo.

Maquinas de Turing universais

- Vamos propor uma forma geral para a especificação de MTs, de modo que suas descrições possam ser usadas como entradas para outras MTs.
- Precisaremos codificar estados e símbolos como cadeias sobre um alfabeto fixo.
- Convensao:
  1. O estado devera ter a forma $\{q\}\{0,1\}^*$
  2. Um simbolo da fita devera ter a forma $\{a\}\{0,1\}^*$

Exemplo

- Considere a MT $M = \{K, \Sigma, \delta, s, \{h\}\}$ onde $K = \{s, q, h\}$, $\Sigma = \{\sqcup, \triangleright, a\}$ e δ eh dado pela tabela

q	σ	$\delta(q, \sigma)$
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\triangleright$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\triangleright$	$(q, \rightarrow)$

Exemplo

estado/simbolo	representação
s	$q00$
q	$q01$
h	$q11$
$\sqcup$	$a000$
$\triangleright$	$a001$
$\leftarrow$	$a010$
$\rightarrow$	$a011$
a	$a100$

- A representação da cadeia $\triangleright aa \sqcup a$ eh

$$\text{“} \triangleright aa \sqcup a'' = a001a100a100a000a100$$

Exemplo

- A representação de M serah a seguinte cadeia

$$\begin{aligned} \text{“}M\text{”} \quad = \quad & (q00, a100, q01, a000), (q00, a100, q01, a000) \\ & (q00, a001, q00, a011), (q01, a100, q00, a011) \\ & (q01, a000, q00, a011), (q01, a001, q01, a011) \end{aligned}$$

- MT universal U usa como programas as codificacoes de outras maquinas para direcionar sua operação.

Exemplo

- A representação de M serah a seguinte cadeia

$$\begin{aligned} \text{“}M\text{”} \quad = \quad & (q00, a100, q01, a000), (q00, a100, q01, a000) \\ & (q00, a001, q00, a011), (q01, a100, q00, a011) \\ & (q01, a000, q00, a011), (q01, a001, q01, a011) \end{aligned}$$

- MT universal U usa como programas as codificacoes de outras maquinas para direcionar sua operação.
- U tem dois argumentos: uma descrição “M” de certa maquina M e uma descrição “w” de uma cadeia de entrada w.

Exemplo

- A representação de M serah a seguinte cadeia

$$\begin{aligned} \text{“}M\text{”} \quad = \quad & (q00, a100, q01, a000), (q00, a100, q01, a000) \\ & (q00, a001, q00, a011), (q01, a100, q00, a011) \\ & (q01, a000, q00, a011), (q01, a001, q01, a011) \end{aligned}$$

- MT universal U usa como programas as codificacoes de outras maquinas para direcionar sua operação.
- U tem dois argumentos: uma descrição “M” de certa maquina M e uma descrição “w” de uma cadeia de entrada w.
- U para em resposta a entrada “M” “w” se e somente se M parar em resposta a entrada w.

Exemplo

- A representação de M serah a seguinte cadeia

$$\begin{aligned} \text{“}M\text{”} \quad = \quad & (q00, a100, q01, a000), (q00, a100, q01, a000) \\ & (q00, a001, q00, a011), (q01, a100, q00, a011) \\ & (q01, a000, q00, a011), (q01, a001, q01, a011) \end{aligned}$$

- MT universal U usa como programas as codificacoes de outras maquinas para direcionar sua operação.
- U tem dois argumentos: uma descrição “M” de certa maquina M e uma descrição “w” de uma cadeia de entrada w.
- U para em resposta a entrada “M” “w” se e somente se M parar em resposta a entrada w.
- $U(\text{“}M\text{”},\text{“}w\text{”}) = \text{“}M(w)\text{”}$

O problema da parada

- Suponha que você escrevesse um programa em sua linguagem de programação favorita. Ele recebe como entrada um programa P escrito na mesma linguagem e uma entrada X para esse programa.

O problema da parada

- Suponha que você escrevesse um programa em sua linguagem de programação favorita. Ele recebe como entrada um programa P escrito na mesma linguagem e uma entrada X para esse programa.
- Suponha que seu programa consiga determinar se P ira parar em resposta a entrada X, retornando "sim" se isso acontecer e "não" caso contrario.

O problema da parada

- Suponha que você escrevesse um programa em sua linguagem de programação favorita. Ele recebe como entrada um programa P escrito na mesma linguagem e uma entrada X para esse programa.
- Suponha que seu programa consiga determinar se P ira parar em resposta a entrada X, retornando "sim" se isso acontecer e "não" caso contrario.
- Nomeamos nosso programa como para(P, X). Ele teria um valor inestimável.

O problema da parada

- Poderíamos escrever o seguinte programa
  1. diagonal(X)
  2. se para(X, X), então vá para 2, senão pare

O problema da parada

- Poderíamos escrever o seguinte programa
  1. diagonal(X)
  2. se para(X, X), então vá para 2, senão pare
- A função diagonal(diagonal) para?

O problema da parada

- Poderíamos escrever o seguinte programa
  1. diagonal(X)
  2. se para(X, X), então vá para 2, senão pare
- A função diagonal(diagonal) para?
- Ela para se e somente se ela não parar. O que eh uma contradição, assim a única hipótese que levantamos eh falsa, ou seja, para(P, X) não existe.

O problema da parada

- Poderíamos escrever o seguinte programa
  1. diagonal(X)
  2. se para(X, X), então vá para 2, senão pare
- A função diagonal(diagonal) para?
- Ela para se e somente se ela não parar. O que eh uma contradição, assim a única hipótese que levantamos eh falsa, ou seja, para(P, X) não existe.
- Então não pode haver algoritmo para resolver o problema que o programa "para" resolveria: decidir se programas arbitrários irão parar ou entrar em execução infinita.

O problema da parada

- Poderíamos escrever o seguinte programa
  1. diagonal(X)
  2. se para(X, X), então vá para 2, senão pare
- A função diagonal(diagonal) para?
- Ela para se e somente se ela não parar. O que eh uma contradição, assim a única hipótese que levantamos eh falsa, ou seja, para(P, X) não existe.
- Então não pode haver algoritmo para resolver o problema que o programa "para" resolveria: decidir se programas arbitrários irão parar ou entrar em execução infinita.

Linguagens recursivamente enumeráveis

- O argumento desenvolvido anteriormente eh familiar.

Linguagens recursivamente enumeráveis

- O argumento desenvolvido anteriormente eh familiar.
- Temos uma notação completa para algoritmos, na forma de linguagem de programação: a MT.

Linguagens recursivamente enumeráveis

- O argumento desenvolvido anteriormente eh familiar.
- Temos uma notação completa para algoritmos, na forma de linguagem de programação: a MT.
- Podemos agora definir uma linguagem que não seja recursiva.

Linguagens recursivamente enumeráveis

- O argumento desenvolvido anteriormente eh familiar.
- Temos uma notação completa para algoritmos, na forma de linguagem de programação: a MT.
- Podemos agora definir uma linguagem que não seja recursiva.

$H = \{\text{“}M\text{”}\text{“}w\text{”} : \text{a MT para em resposta a cadeia de entrada } w\}$

Linguagens recursivamente enumeráveis

- O argumento desenvolvido anteriormente eh familiar.
- Temos uma notação completa para algoritmos, na forma de linguagem de programação: a MT.
- Podemos agora definir uma linguagem que não seja recursiva.

$H = \{\text{“}M\text{”}\text{“}w\text{”} : \text{a MT para em resposta a cadeia de entrada } w\}$

Problemas indecidíveis

1. Qualquer algoritmo pode ser convertido em uma MT que sempre para para todas as entradas.

Problemas indecidíveis

1. Qualquer algoritmo pode ser convertido em uma MT que sempre para para todas as entradas.
2. Então, não há algoritmo que decida, para uma dada maquina M e uma cadeia w, se M aceita w ou não.

Problemas indecidíveis

1. Qualquer algoritmo pode ser convertido em uma MT que sempre para para todas as entradas.
2. Então, não há algoritmo que decida, para uma dada maquina M e uma cadeia w, se M aceita w ou não.
3. Os problemas para os quais não existe algoritmos são ditos **indecidíveis** ou **insolúveis**.

Problemas indecidíveis

1. Qualquer algoritmo pode ser convertido em uma MT que sempre para para todas as entradas.
2. Então, não há algoritmo que decida, para uma dada maquina M e uma cadeia w, se M aceita w ou não.
3. Os problemas para os quais não existe algoritmos são ditos **indecidíveis** ou **insolúveis**.
4. O problema indecidível mais famoso e fundamental eh responder se uma MT para ou não.

Problemas indecidíveis

1. Qualquer algoritmo pode ser convertido em uma MT que sempre para para todas as entradas.
2. Então, não há algoritmo que decida, para uma dada maquina M e uma cadeia w, se M aceita w ou não.
3. Os problemas para os quais não existe algoritmos são ditos **indecidíveis** ou **insolúveis**.
4. O problema indecidível mais famoso e fundamental eh responder se uma MT para ou não.
5. Tal problema eh conhecido como o **Problema da parada da maquina de Turing**.

Reduções

Definição

Sejam $L_1, L_2 \subseteq \Sigma^*$ duas linguagens. Uma redução de L_1 para L_2 eh uma função recursiva $\tau : \Sigma^* \to \Sigma^*$, tal que $x \in L_1$ se e somente se $\tau(x) \in L_2$.

Reduções

Definição

Sejam $L_1, L_2 \subseteq \Sigma^*$ duas linguagens. Uma redução de L_1 para L_2 eh uma função recursiva $\tau : \Sigma^* \to \Sigma^*$, tal que $x \in L_1$ se e somente se $\tau(x) \in L_2$.

- Teorema: Se L_1 eh uma linguagem não-recursiva, e se houver uma redução de L_1 para L_2, então L_2 também não eh recursiva.

Problemas indecidíveis

São indecidíveis os seguintes problemas acerca de MTs: Dada uma MT M,

Problemas indecidíveis

São indecidíveis os seguintes problemas acerca de MTs: Dada uma MT M,

1. M para em resposta a entrada w?

Problemas indecidíveis

São indecidíveis os seguintes problemas acerca de MTs: Dada uma MT M,

1. M para em resposta a entrada w?
2. M para em resposta a uma cadeia vazia?

Problemas indecidíveis

São indecidíveis os seguintes problemas acerca de MTs: Dada uma MT M,

1. M para em resposta a entrada w?
2. M para em resposta a uma cadeia vazia?
3. há alguma cadeia para a qual M para?

Problemas indecidíveis

São indecidíveis os seguintes problemas acerca de MTs: Dada uma MT M,

1. M para em resposta a entrada w?
2. M para em resposta a uma cadeia vazia?
3. há alguma cadeia para a qual M para?
4. M para em resposta a qualquer cadeia?

Problemas indecidíveis

São indecidíveis os seguintes problemas acerca de MTs: Dada uma MT M,

1. M para em resposta a entrada w?
2. M para em resposta a uma cadeia vazia?
3. há alguma cadeia para a qual M para?
4. M para em resposta a qualquer cadeia?
5. Dadas duas MTs, M_1 e M_2, elas param para as mesmas cadeias?

Problemas indecidíveis

São indecidíveis os seguintes problemas acerca de MTs: Dada uma MT M,

1. M para em resposta a entrada w?
2. M para em resposta a uma cadeia vazia?
3. há alguma cadeia para a qual M para?
4. M para em resposta a qualquer cadeia?
5. Dadas duas MTs, M_1 e M_2, elas param para as mesmas cadeias?
6. a linguagem que M semidecide eh regular? Livre de contexto? Recursiva?