Problema da parada

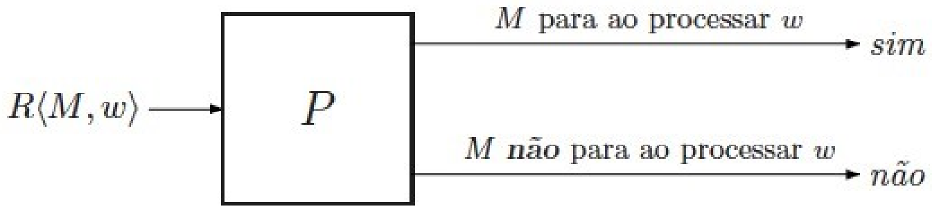
Vamos discutir agora o conhecido problema da parada. Embora iremos discuti-lo a partir de MTs, ele se aplica diretamente a linguagens de programação pela tese de Church-Turing.

O problema da parada é enunciado como:

- Dadas uma MT M e uma palavra w arbitrárias, determinar se M para ao processar w.

A existência da MTU U_p prova que a linguagem desse problema é LRE. A pergunta agora é se ele é decidível. Será que existe uma MT equivalente a U_p que sempre pare com qualquer palavra de $\{R\langle M,w\rangle \mid \ M \ \ para \ \ ao \ \ computar \ \ w\}$?

Vamos demonstrar que essa linguagem não é decidível. A demonstração será por contradição. Para isso, vamos supor a existência de uma MT P que decida o problema. Essa máquina seria da seguinte forma:

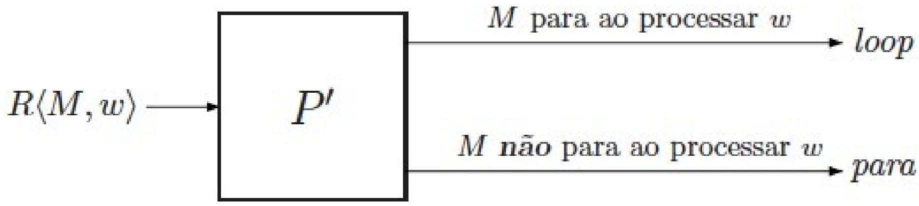


A MT P, portanto, sempre para e dá a resposta correta. Se M para com w, então ela diz *sim* (para em estado final), caso contrário diz *não* (para em estado não final).

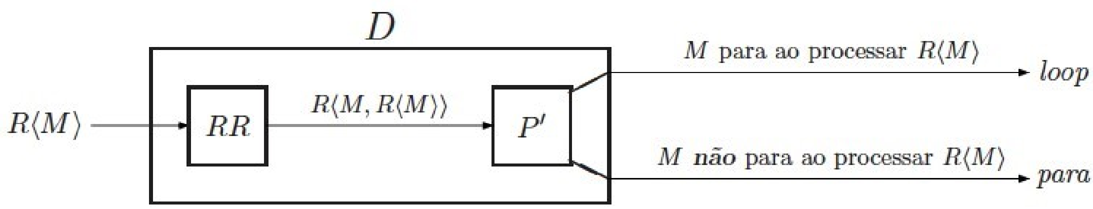
Dada a existência de P, podemos construir a máquina P' com o seguinte comportamento:

- P' entra em loop se M para ao processar w
- P' para se M entra em loop ao processar w

Para isso, basta fazer com que P' entre em loop sempre que P parar em estado final (adicionamos um novo estado a P' em que sempre se lê qualquer símbolo da fita e se move o cabeçote para direita). Assim, P' seria da seguinte forma:



Agora, vamos criar uma terceira máquina a partir de P'. Essa máquina, ao contrário das duas anteriores, receberá apenas a representação de uma máquina como entrada. Ela então produz uma entrada para a máquina P', contendo a própria máquina recebida como entrada e sua representação como palavra. Esquematicamente seria como na figura abaixo:



A máquina D:

1. Recebe a representação de uma máquina como entrada;
2. Produz uma instância do problema de determinar se uma MT M para com uma palavra w, em que a MT é a própria máquina recebida na entrada, e a palavra sua representação;
3. Roda a máquina P' com essa nova instância gerada.

Como D aceita qualquer representação de MT, podemos entregar a ela sua própria representação $R\langle D\rangle$. Agora, o que aconteceria nesse caso?

- Se D parar, então D não para ao processar $R\langle D\rangle$
- Se D não parar, então D para ao processar $R\langle D\rangle$

Ou seja, D para ao processar $R\langle D\rangle$ se, e somente se, D não para ao processar $R\langle D\rangle$.

Contradição! Como D foi construída a partir de P', que foi construída a partir de P, temos que P não pode existir. Portanto, o problema da parada é indecidível!

Dessa forma, demonstramos que a linguagem $L_p = \{R\langle M, w\rangle \mid \ M \ \ para \ \ com \ \ w\}$ não é recursiva. Portanto, demonstramos que $LRec \subset LRE$. Existe ao menos uma linguagem recursivamente enumerável que não é recursiva.

Além disso, temos que $\overline{L_p} \notin LRE$. Isso porque, como L_p é LRE, se seu complemento também fosse, então L_p seria recursiva. Bastaria criar uma máquina que acionasse as máquinas que reconhecessem cada elas para decidir L_p (uma das duas pararia com a palavra de entrada). Como acabamos de demonstar que L_p é não decidível, temos que $\overline{L_p}$ **não pode ser computada**! Portanto, demonstramos a existência de problemas não computáveis pela máquina de Turing. (Essa afirmação também pode ser verificada através da diagonalização de Cantor. O conjunto das MTs pode ser enumerado -- ordenação lexicográfica das representações -- enquanto $\mathcal{P}(\Sigma^*)$ não é enumerável.)

Assim temos que $LR \subset LLC \subset LRec \subset LRE \subset \mathcal{P}(\Sigma^*)$.