Utilizando linguagens de programação orientadas a objetos para codificar programas adaptativos

Paulo Roberto Massa Cereda
paulo.cereda@usp.br

João José Neto
jjneto@usp.br

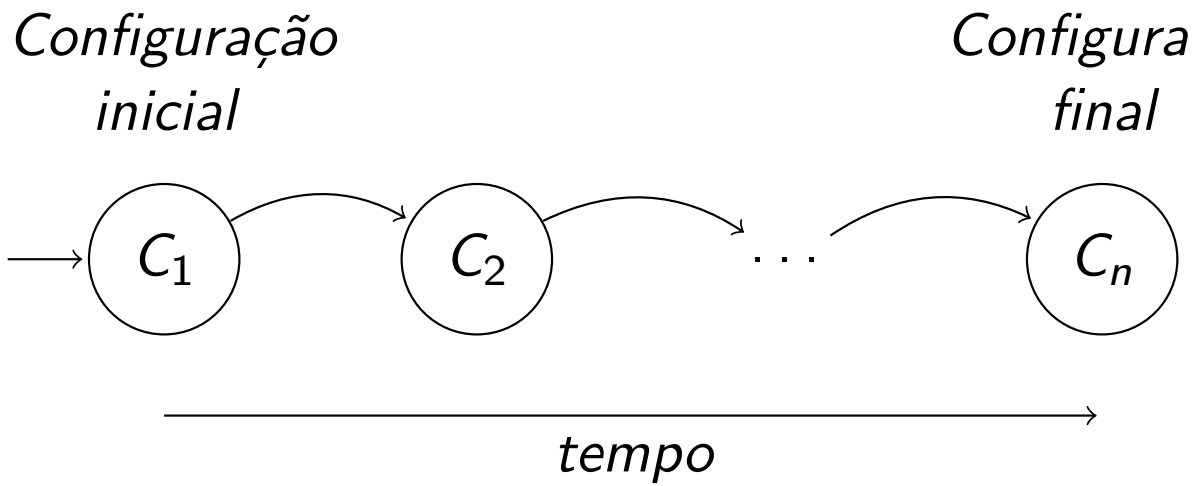
Escola Politécnica, Universidade de São Paulo

29 de Janeiro de 2015

WTA 2015

Motivação do trabalho

Um programa escrito em uma linguagem de programação com características adaptativas tem, no início de sua execução, um comportamento semelhante ao de um código não-adaptativo tradicional



Nossa contribuição

Estado da arte:

- Iniciativas baseadas em um subconjunto de código de montagem adaptativo
- Especificações para linguagens de alto nível

Nossa contribuição: técnicas para a incorporação dos conceitos adaptativos em linguagens de programação orientadas a objetos já conhecidas

Introdução

Técnicas para a escrita de código adaptativo

- Linguagens escolhidas: Java e C#
- Técnicas apresentadas: reflexão estrutural e blocos de código
- Qualquer linguagem de programação pode ser utilizada, desde que alguns requisitos mínimos estejam presentes nas especificações da linguagem escolhida

Reflexão estrutural

- O uso de reflexão permite a construção de programas auto-modificáveis
- Apenas ações de introspecção são permitidas
- **Reflexão comportamental:** interceptar uma operação tal como uma invocação de método e mudar o comportamento dessa operação
- **Reflexão estrutural:** oferecer a possibilidade de modificar estruturas de dados

Reflexão estrutural

Por exemplo, em Java:

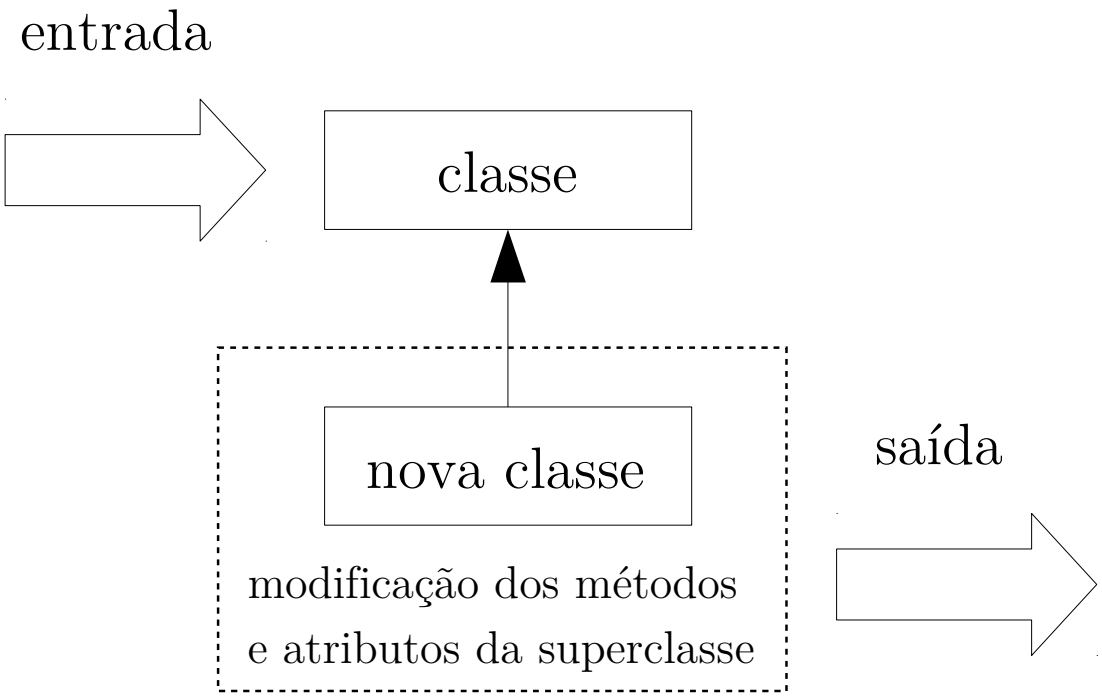
- modificações não são permitidas em tempo de execução
- uma vez que uma classe é instanciada na memória, todas as tentativas de modificar sua estrutura lançarão uma exceção

Reflexão estrutural

Solução:

1. Utilizar o conceito de herança do paradigma de programação orientada a objetos para estender a classe-alvo (já instanciada).
2. Alterar a estrutura da nova classe em tempo de execução.
3. Instanciar a nova classe.
4. Definir uma nova referência para a variável contendo o objeto ascendente original

Reflexão estrutural



É possível repetir a extensão de classes exaustivamente através da modificação de atributos e métodos, quando apropriado!

Reflexão estrutural

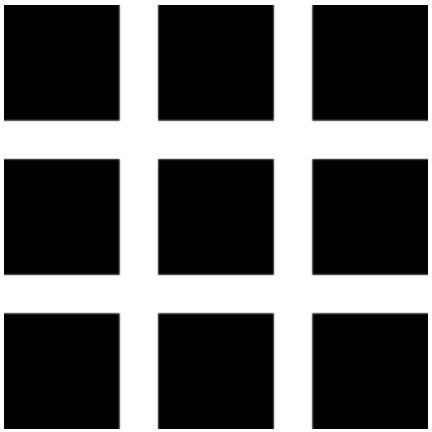
Vantagens:

- Permite modificações em tempo de execução!

Desvantagens:

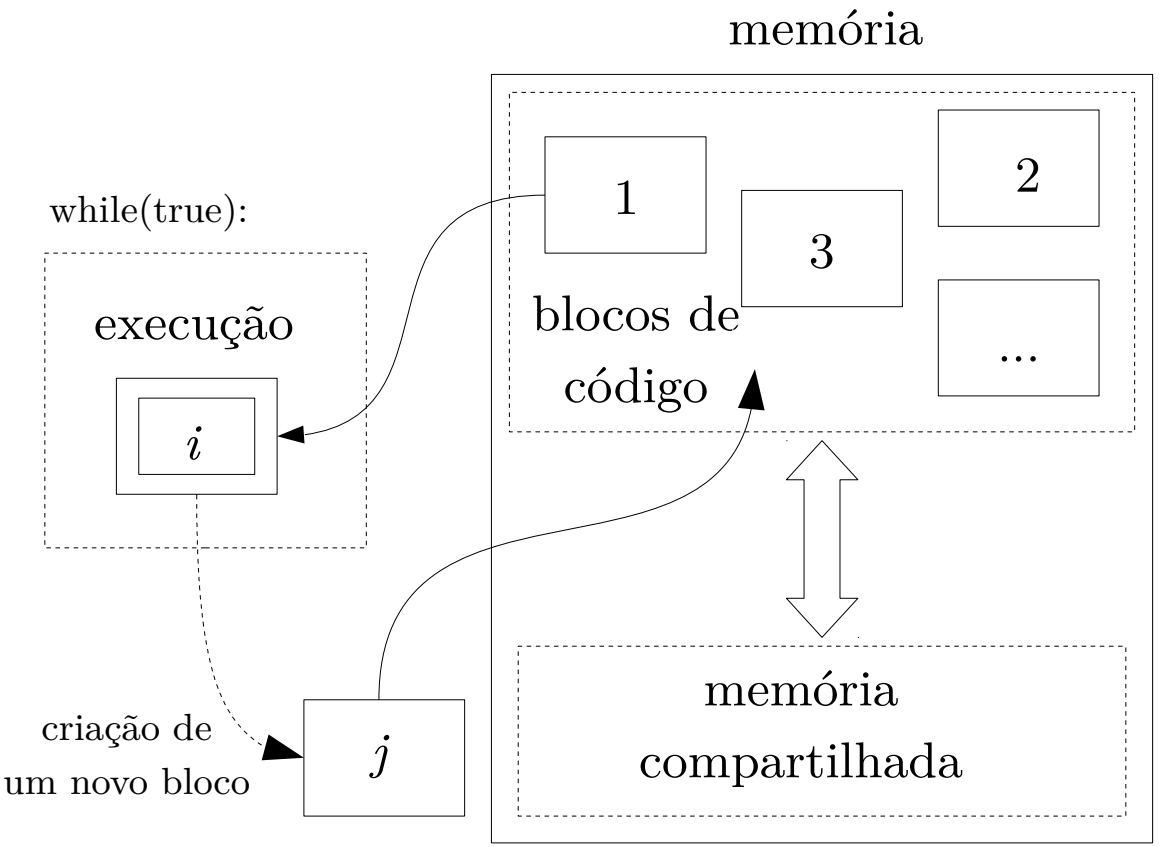
- Tempo de execução e consumo de memória consideráveis devido à ausência de otimizações do compilador
- Cada modificação simples requer descoberta
- Quando existe manipulação de bytecode, deve-se considerar também o processo de reconstrução

Blocos de código



- Conceito utilizado por simuladores:
  - conjunto de blocos de código, conectados por ligações simbólicas
  - executor que roda cada bloco, um por vez
- Linguagens de programação modernas possuem suporte a threading
  - Interfaces de threading oferecem um protocolo comum para objetos executarem um determinado trecho de código durante seus ciclos de vida

Blocos de código



Laço principal e a execução sequencial de cada bloco até que uma instrução de parada seja encontrada!

Blocos de código

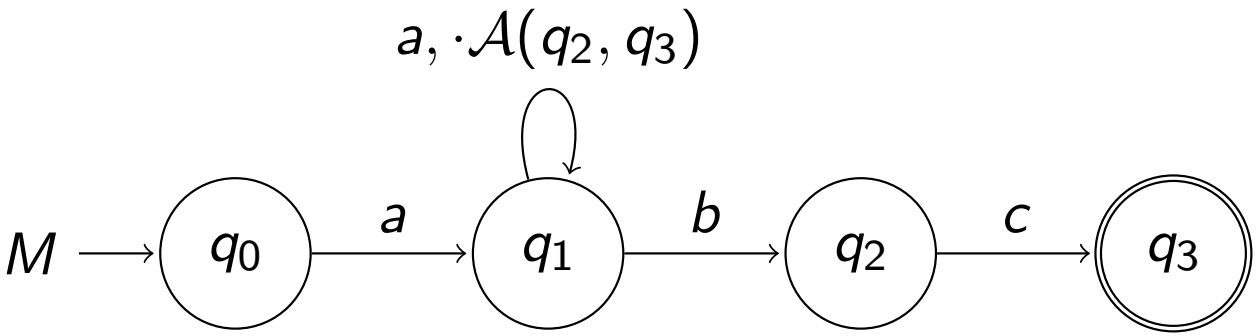
Vantagens:

- Solução viável para implementar características adaptativas em linguagens de programação que suportam threading
- Ótimo desempenho em relação a seu equivalente reflexivo

Desvantagens:

- Requer um alto nível de especificação do que seu equivalente reflexivo (blocos de código são definidos com operações básicas ao invés de métodos completos)

Introdução

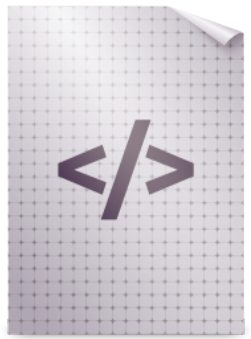


Autômato adaptativo que reconhece cadeias na forma $a^n b^n c^n$, com $n \in \mathbb{N},\ n \geq 1$

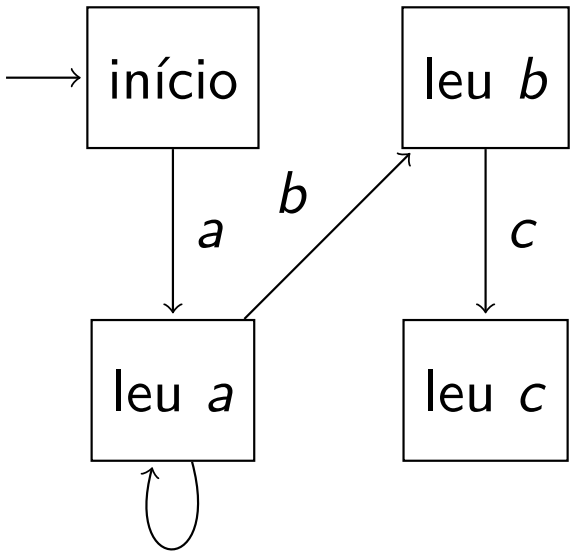
Introdução

Duas implementações:

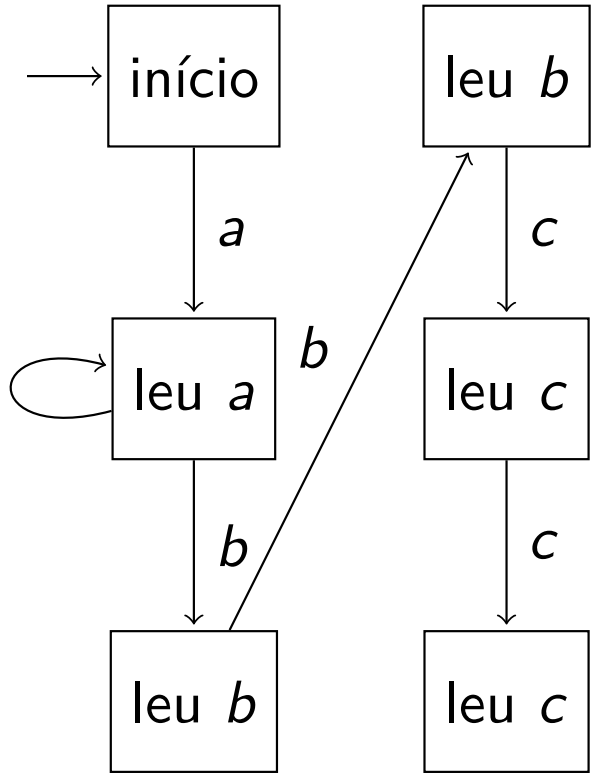
- Reflexão estrutural para realizar as ações adaptativas, disparadas por uma função adaptativa definida no conjunto de regras do dispositivo
- Blocos de código para representar cada estado do dispositivo, e ações adaptativas são disparadas através da criação de novos blocos e da modificação das ligações dinâmicas existentes entre blocos em tempo de execução



Introdução



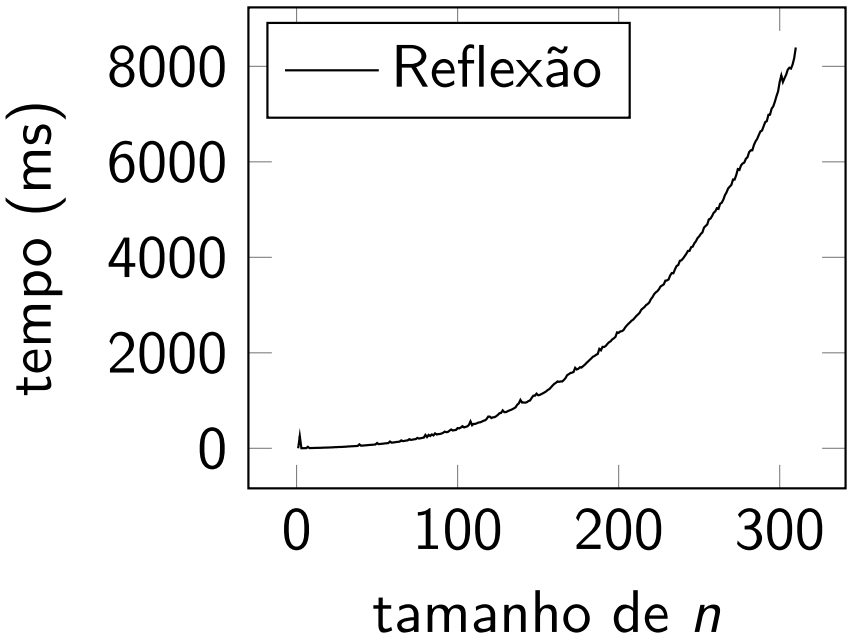
a, adicionar mais uma leitura de *b* e *c*



a, adicionar mais uma leitura de *b* e *c*

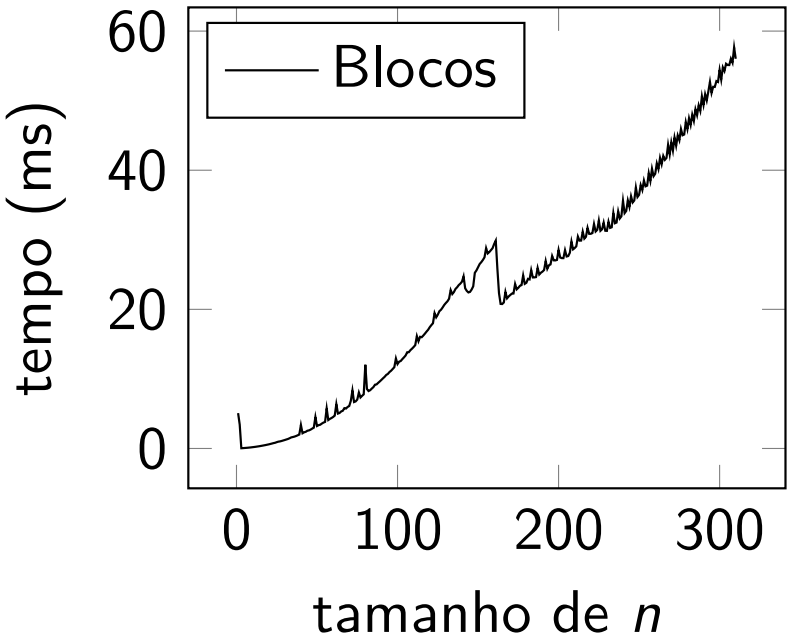
Primeiro experimento

Medição, em milissegundos, do tempo de reconhecimento de cada cadeia w, $w = a^n b^n c^n$, com $1 \leq n \leq 310$, durante 1000 iterações



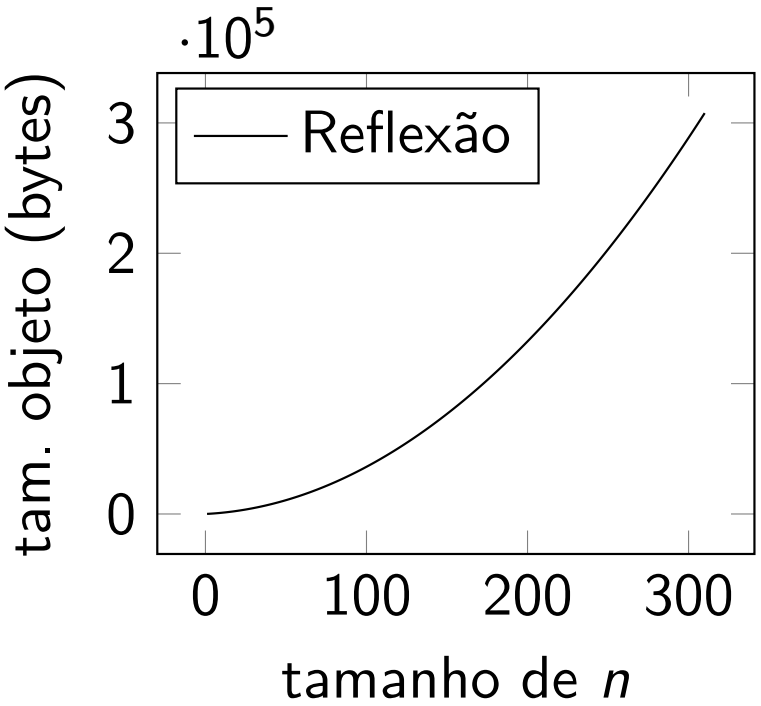
Primeiro experimento

Medição, em milissegundos, do tempo de reconhecimento de cada cadeia w, $w = a^n b^n c^n$, com $1 \leq n \leq 310$, durante 1000 iterações



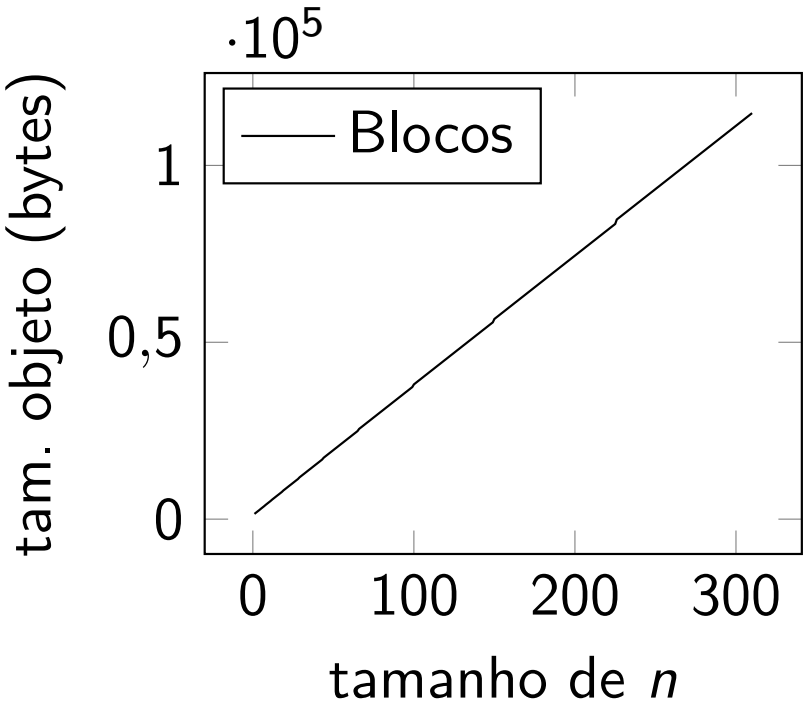
Segundo experimento

Medição, em bytes, do tamanho de objeto de cada componente implementando o motor adaptativo, e o consumo de memória para cada cadeia w, $w = a^n b^n c^n$, com $1 \leq n \leq 310$, durante 1000 iterações



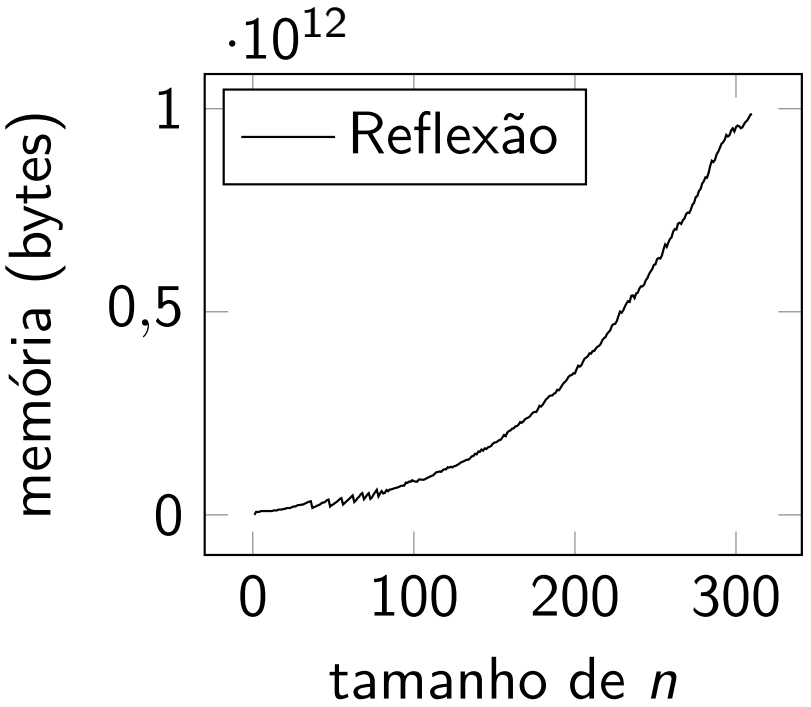
Segundo experimento

Medição, em bytes, do tamanho de objeto de cada componente implementando o motor adaptativo, e o consumo de memória para cada cadeia w, $w = a^n b^n c^n$, com $1 \leq n \leq 310$, durante 1000 iterações



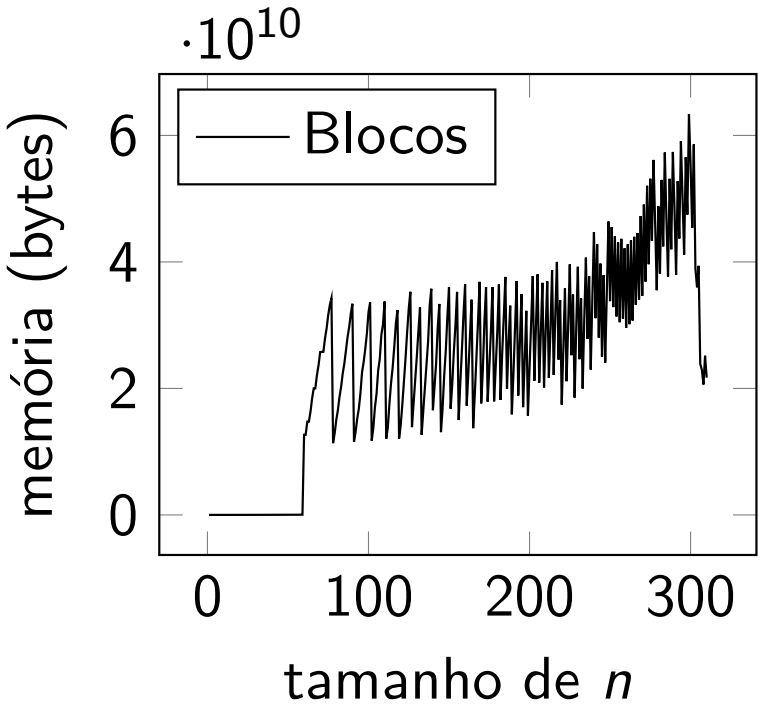
Segundo experimento

Medição, em bytes, do tamanho de objeto de cada componente implementando o motor adaptativo, e o consumo de memória para cada cadeia w, $w = a^n b^n c^n$, com $1 \leq n \leq 310$, durante 1000 iterações



Segundo experimento

Medição, em bytes, do tamanho de objeto de cada componente implementando o motor adaptativo, e o consumo de memória para cada cadeia w, $w = a^n b^n c^n$, com $1 \leq n \leq 310$, durante 1000 iterações



Considerações finais

- Viabilidade da implementação de programas adaptativos através da inserção de trechos de código com características de auto-modificação
- Apesar da reflexão ter problemas de desempenho, ainda assim é uma solução viável para a inspeção de código já existente e injeção de blocos de código com características de adaptatividade quando necessário

Obrigado!

Paulo Roberto Massa Cereda
paulo.cereda@usp.br

João José Neto
jjneto@usp.br