Replicação Orientada a Metaprogramação

Apresentação de Defesa de Mestrado

Fellipe Augusto Ugliara

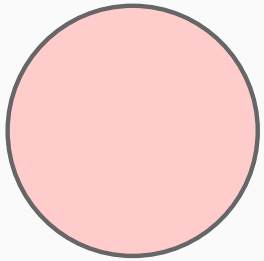
12 de Junho de 2018

Universidade Federal de São Carlos – UFSCar

Introdução

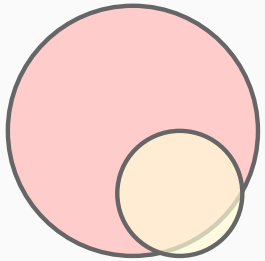
A **metaprogramação**, da linguagem de programação **Cyan**, foi usada para tornam o código-fonte dos programas **replicados**, desenvolvidos usam **Treplica**, mais **coesos** e menos **acoplados**.

Código-Fonte
- Replicação
- Metaprogramação



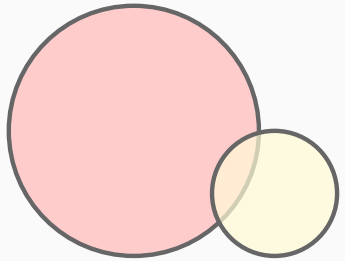
\+ Coeso
- Acoplado

Código-Fonte
\+ Replicação
- Metaprogramação



- Coeso
\+ Acoplado

Código-Fonte
\+ Replicação
\+ Metaprogramação

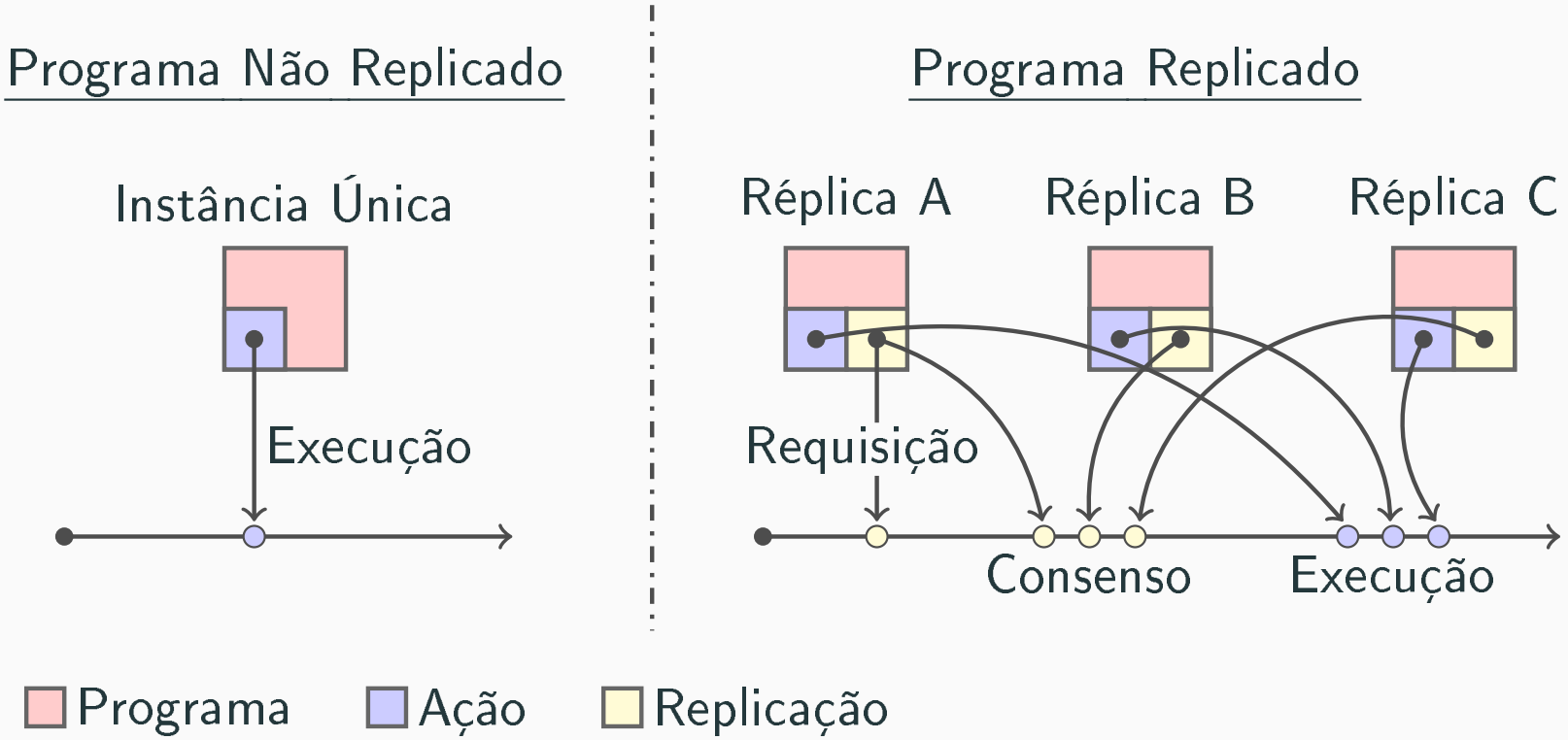


\+ Coeso
- Acoplado

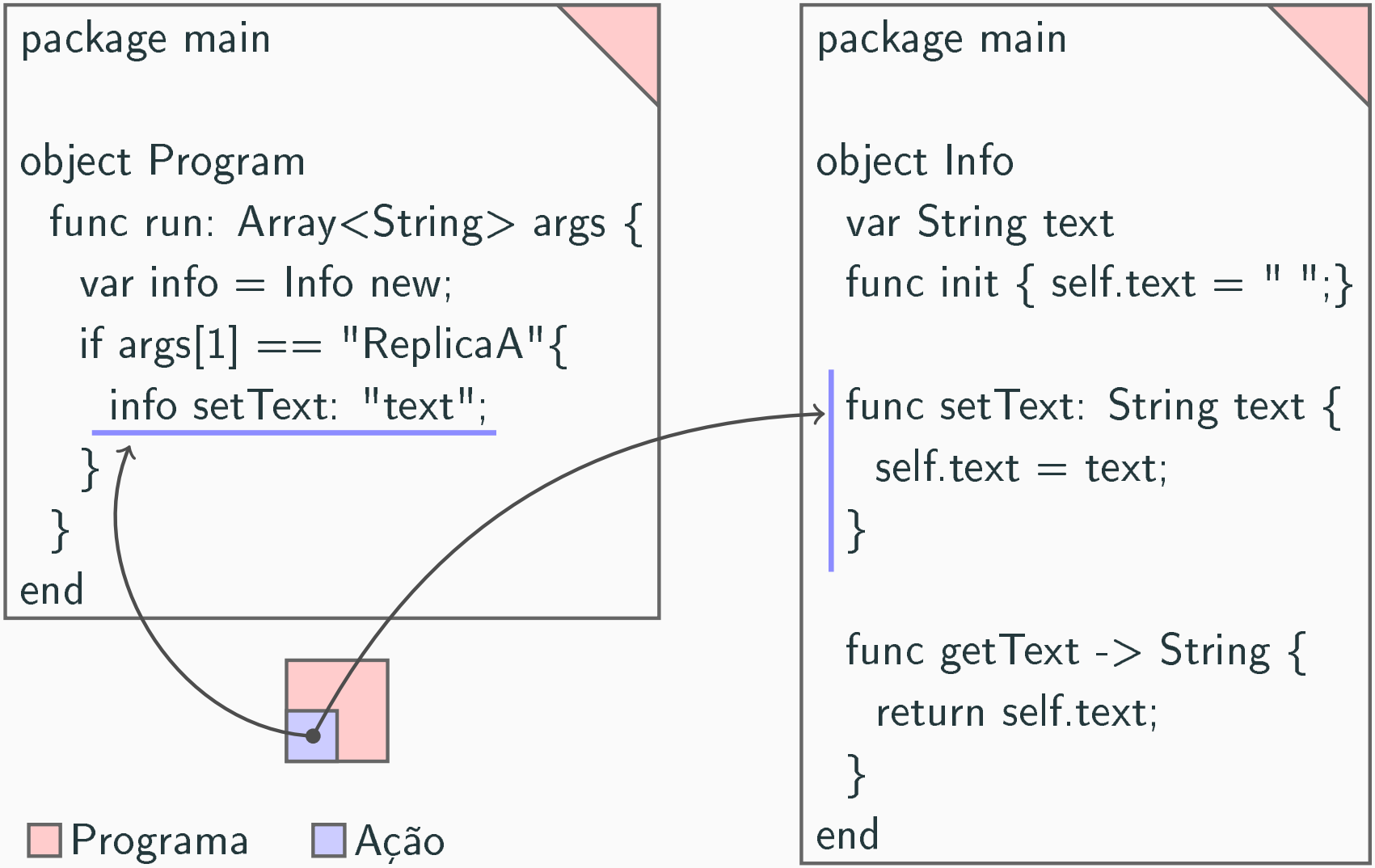
Programa Replicação

Replicação

Replicação é um método usado na implementação de programas de computador que podem ser tolerantes a falhas.



Programa em Cyan

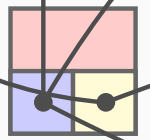


Programa Usando Treplica

```
package main
import treplica

object Program
  func run: Array<String> args {
    var info = Info new;
    var treplica = Treplica new;
    treplica runMachine:
      info numberProcess: 3 rtt: 200
      path: "/tmp"++ args[1];
    var action = UpdateAction new: "text";
    if args[1] == "ReplicaA"{
      treplica execute : action ;
    }
  }
end
```

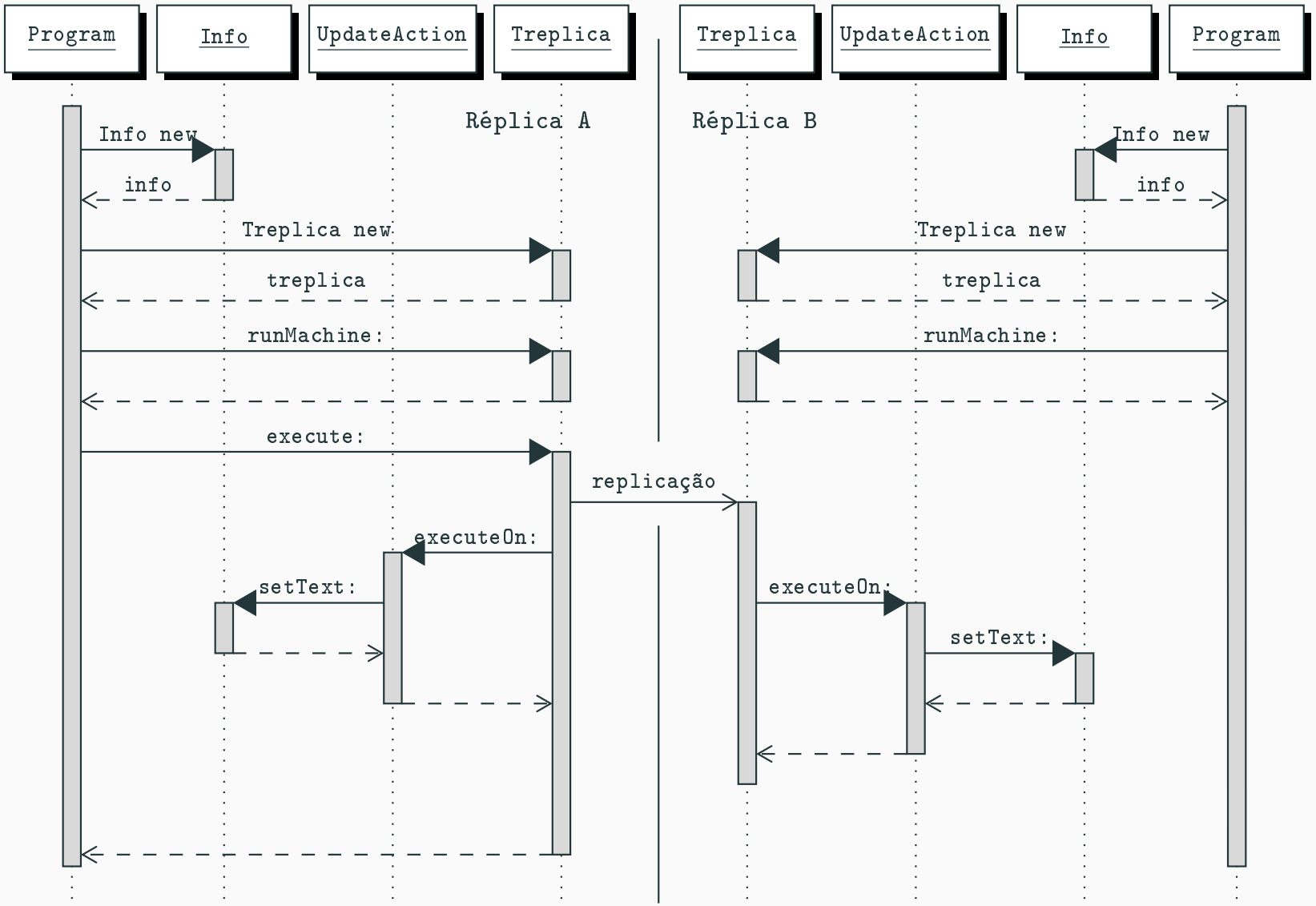
```
package main
import treplica

object UpdateAction extends Action
  var String updateText
  func init: String text {
    self.updateText = text;
  }

  func executeOn: Context context  {
    var info = Info cast: context;
    info setText: self.updateText;
  }
end
```

```
package main
import treplica

object Info extends Context
  ...
  func setText: String text {...}
  ...
end
```



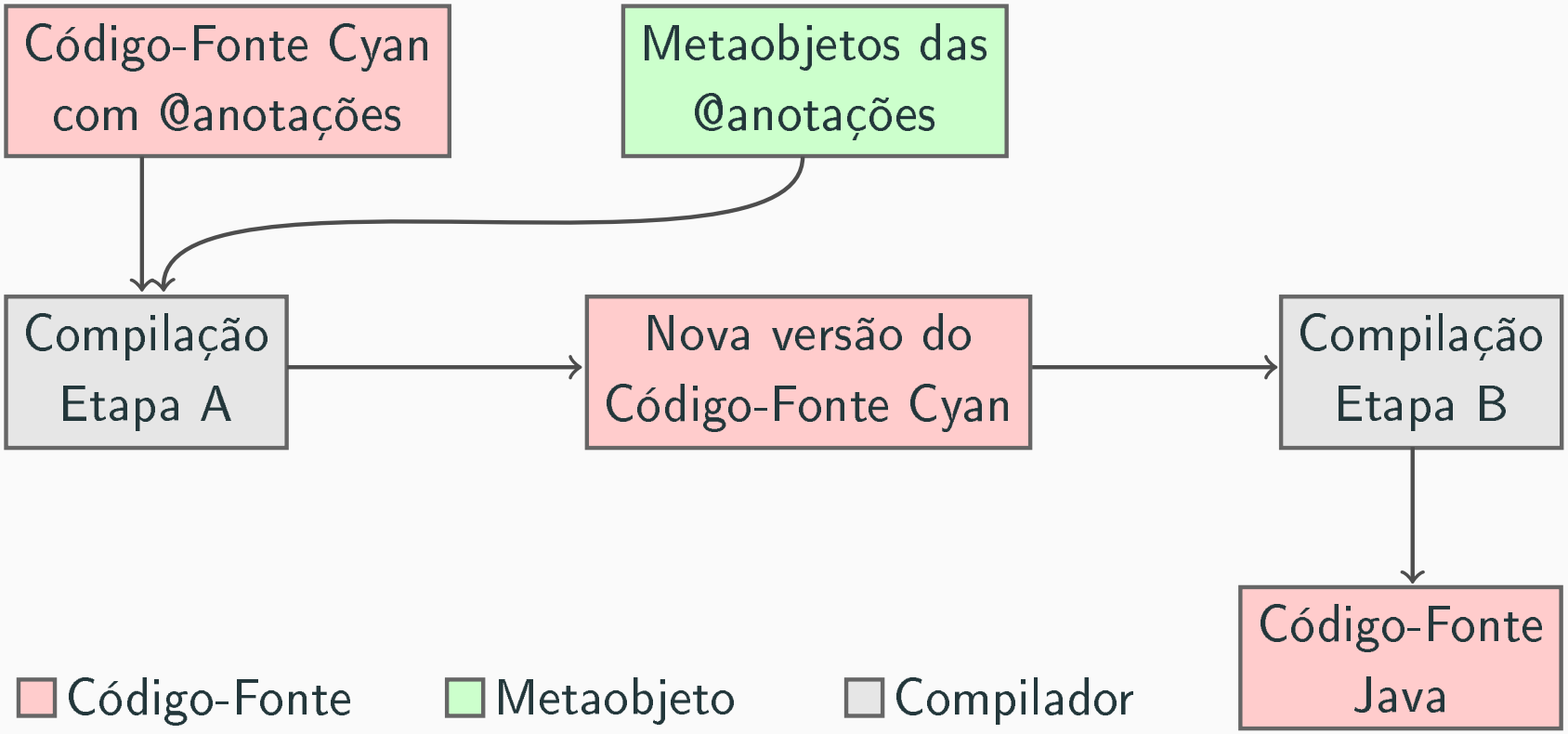
Programa Ação Replicação

Execução de Ação em Treplica

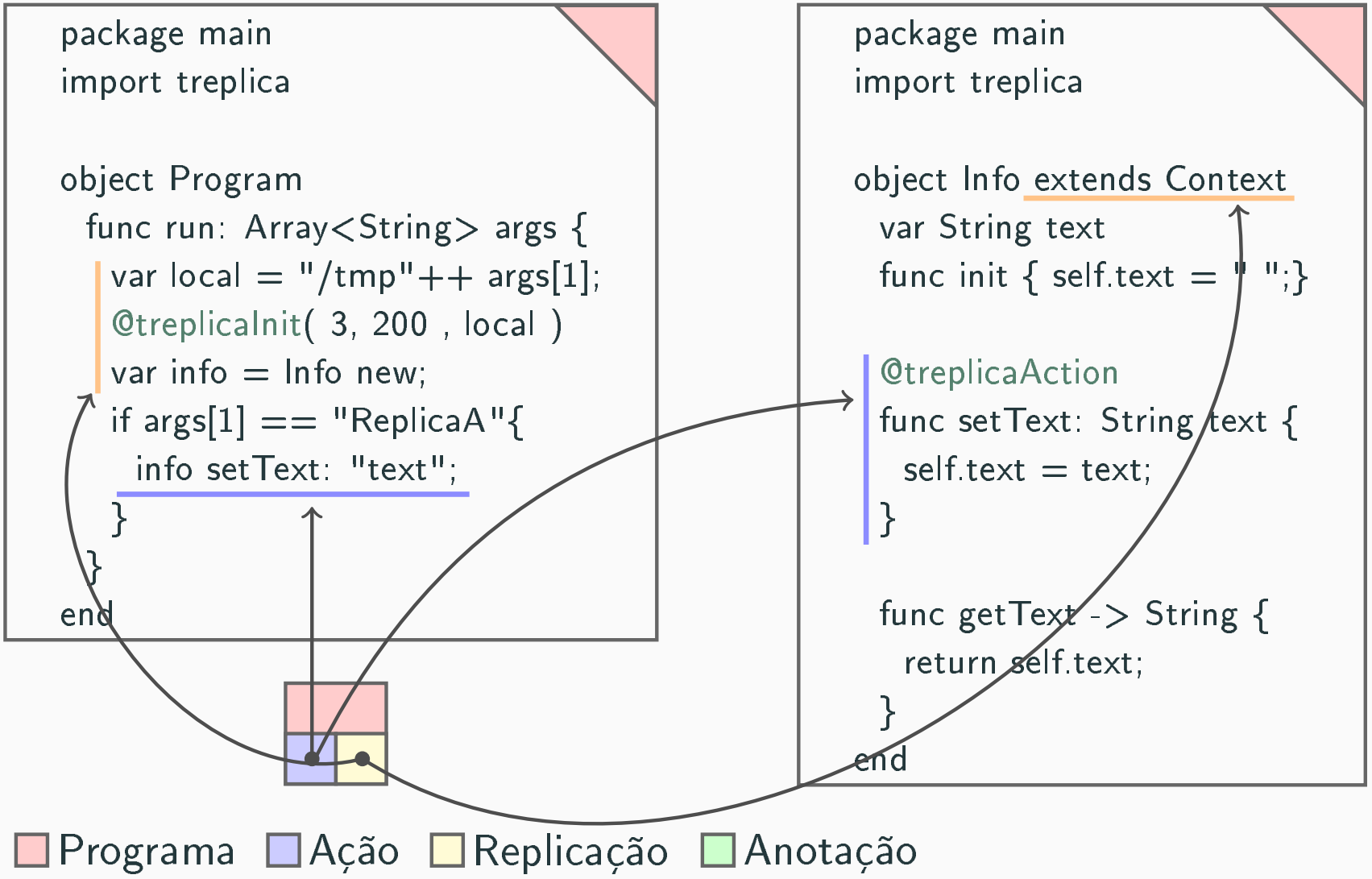


Metaprogramação em Cyan

A metaprogramação é a criação de um código-fonte (programa) que reflete seu significado sobre ele próprio. Em Cyan metaprogramação é realizada em tempo de compilação usando metaobjetos.



Treplica usando Metaobjetos



@treplicaInit

```
package main
import treplica

object Program
  func run: Array<String> args {
    var local = "/tmp"++ args[1];

    @treplicaInit( 3, 200 , local )
    var info = Info new;

    if args[1] == "ReplicaA"{
      info setText: "text";
    }
  }
end
```

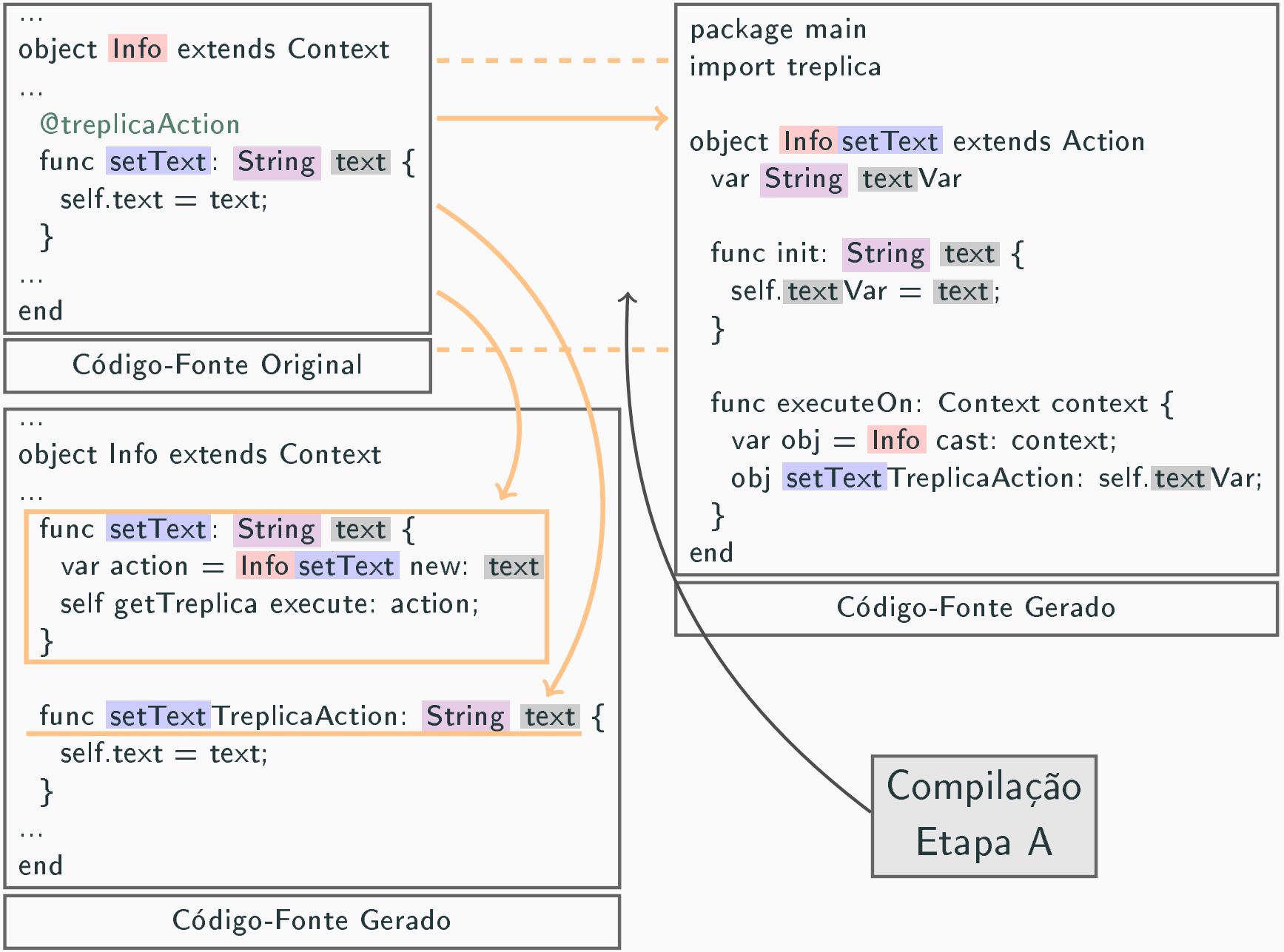
Código-Fonte Original

Compilação
Etapa A

```
package main
import treplica

object Program
  func run: Array<String> args {
    var local = "/tmp"++ args[1];

    var info = Info new;
    var treplicainfo = Treplica new;
    treplicainfo runMachine:
      info numberProcess: 3 rtt: 200
      path: local;
    info setTreplica: treplicainfo;

    if args[1] == "ReplicaA"{
      info setText: "text";
    }
  }
end
```

Código-Fonte Gerado

@treplicaAction

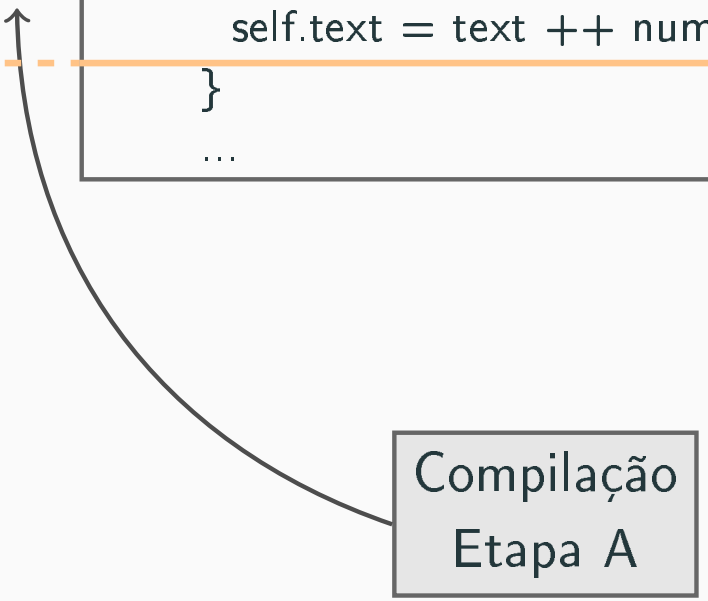


Restrições e Verificações

```
package main
...
object Info extends Context
  ...
  @treplicaAction
  func setText: String text {
    var num = getRandom;
    self.text = text ++ num;
  }

  func getRandom -> String {
    return Math random;
  }

  func getFix -> String {
    return "23";
  }
  ...
end
```

```
main,Info,getRandom-main,Info,getFix
```

```
...
func setText: String text {
  var num = main.Info getFix;
  self.text = text ++ num;
}
...
```



Conclusão
e
Questões?