
UFAM

**Universidade Federal do Amazonas**
**Programa de Pós-Graduação em Engenharia Elétrica**

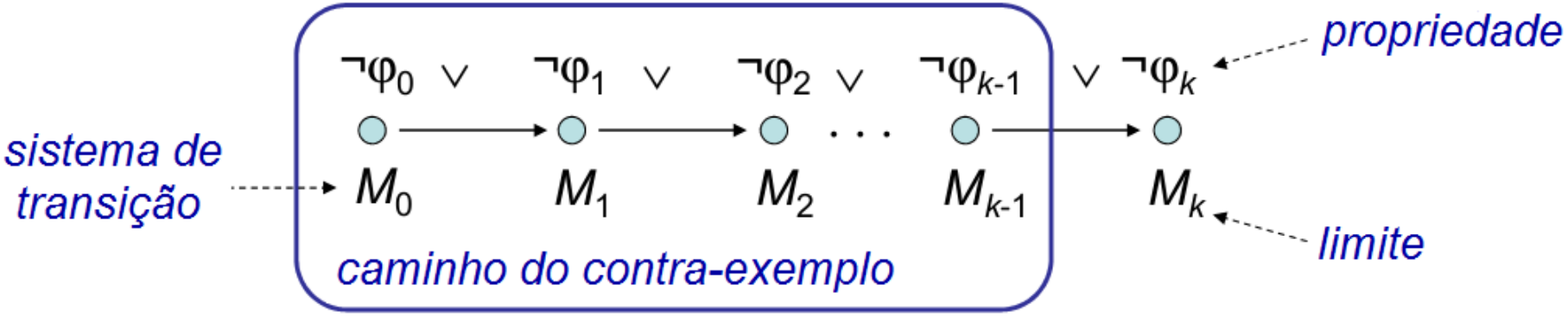
# Verificação de Códigos Lua Utilizando BMCLua

Francisco Januário, Lucas Cordeiro e Eddie Filho
franciscojanuario@ufam.edu.br, lucascordeiro@ufam.edu.br, eddie@ctpim.org.br

# Verificação de Modelo Limitada

UFAM

• Checa a negação de uma propriedade em uma dada profundidade




• Sistema de transição ***M*** com profundidade de ***k***

– **Estado:** contador de programa e valor de variáveis.

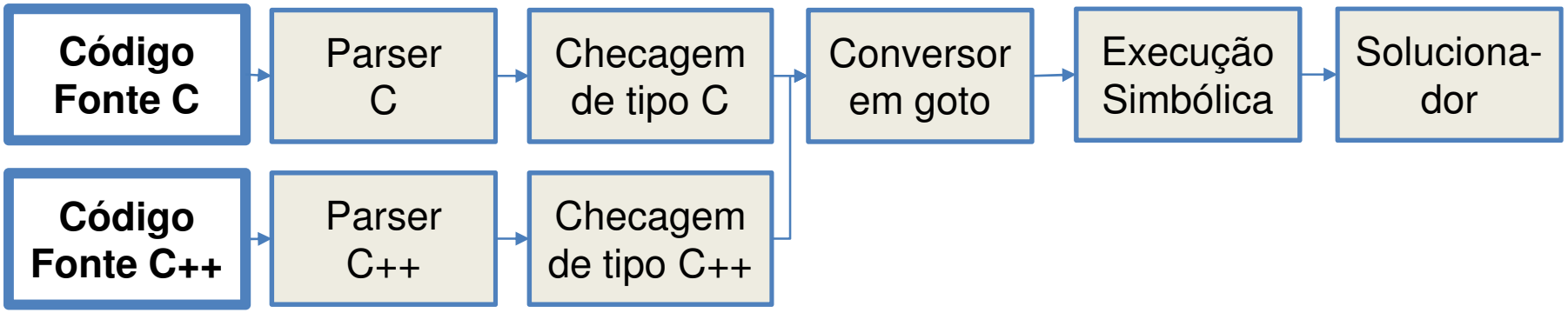
• Traduzido em uma condição de verificação ψ tal que:

| ψ é satisfatível se, e se somente se, houver um contra exemplo de profundidade máxima ***k***. |
|---|

# O Verificador ESBMC

UFAM

• Realiza a verificação de códigos ANSI-C e C++ usando solucionadores das teorias do módulo da satisfatibilidade




• Valida programas sequencias ou multi-tarefas

- Deadlocks
- Estouro aritmético
- Divisão por zero
- Limites de array
- Corrida de dados
- Atomicidade

# A Linguagem Lua (1)


• Utilizada em diversas aplicações, desde jogos até aplicações para TV Digital

- Adobe's Photoshop Lightroom
- Middleware Ginga
- World of Warcraft e Angry Birds


| Linguagem de extensão utilizada por outras linguagens | Interpretada, compacta e rápida, usada em diversos dispositivos |
|---|---|
| – C/C++<br>– NCL<br>– JAVA | – Mobile<br>– Set-Top Box |

# A Linguagem Lua (2)


- Defeitos em aplicações interativas para *set-top box* não detectados em outras fases do desenvolvimento

# A Linguagem Lua (2)


- Defeitos em aplicações interativas para *set-top box* não detectados em outras fases do desenvolvimento

```
local counter = 0
local dx, dy = canvas:attrSize()
function handler (evt)
    if evt.class=='ncl' then
        while dx ~= dy do
            counter = counter + 1
            canvas:drawText(10,10,'Progresso: '..counter)
        end
    end
end
event.register(handler)
```

# A Linguagem Lua (2)

UFAM

• Defeitos em aplicações interativas para *set-top box* não detectados em outras fases do desenvolvimento

```
local counter = 0
local dx, dy = canvas:attrSize()
function handler (evt)
    if evt.class=='ncl' then
        while dx ~= dy do
            counter = counter + 1
            canvas:drawText(10,10,'Progresso: '..counter)
        end
    end
end
event.register(handler)
```

*OverFlow* Aritmético

# A Linguagem Lua (2)

UFAM

• Defeitos em aplicações interativas para *set-top box* não detectados em outras fases do desenvolvimento

```
local counter = 0
local dx, dy = canvas:attrSize()
function handler (evt)
    if evt.class=='ncl' then
        while dx ~= dy do
            counter = counter + 1
            canvas:drawText(10,10,'Progresso: '..counter)
        end
    end
end
event.register(handler)
```

*OverFlow* Aritmético

• Impacto negativo na execução das aplicações interativas, como travamentos, etc.

# Objetivos

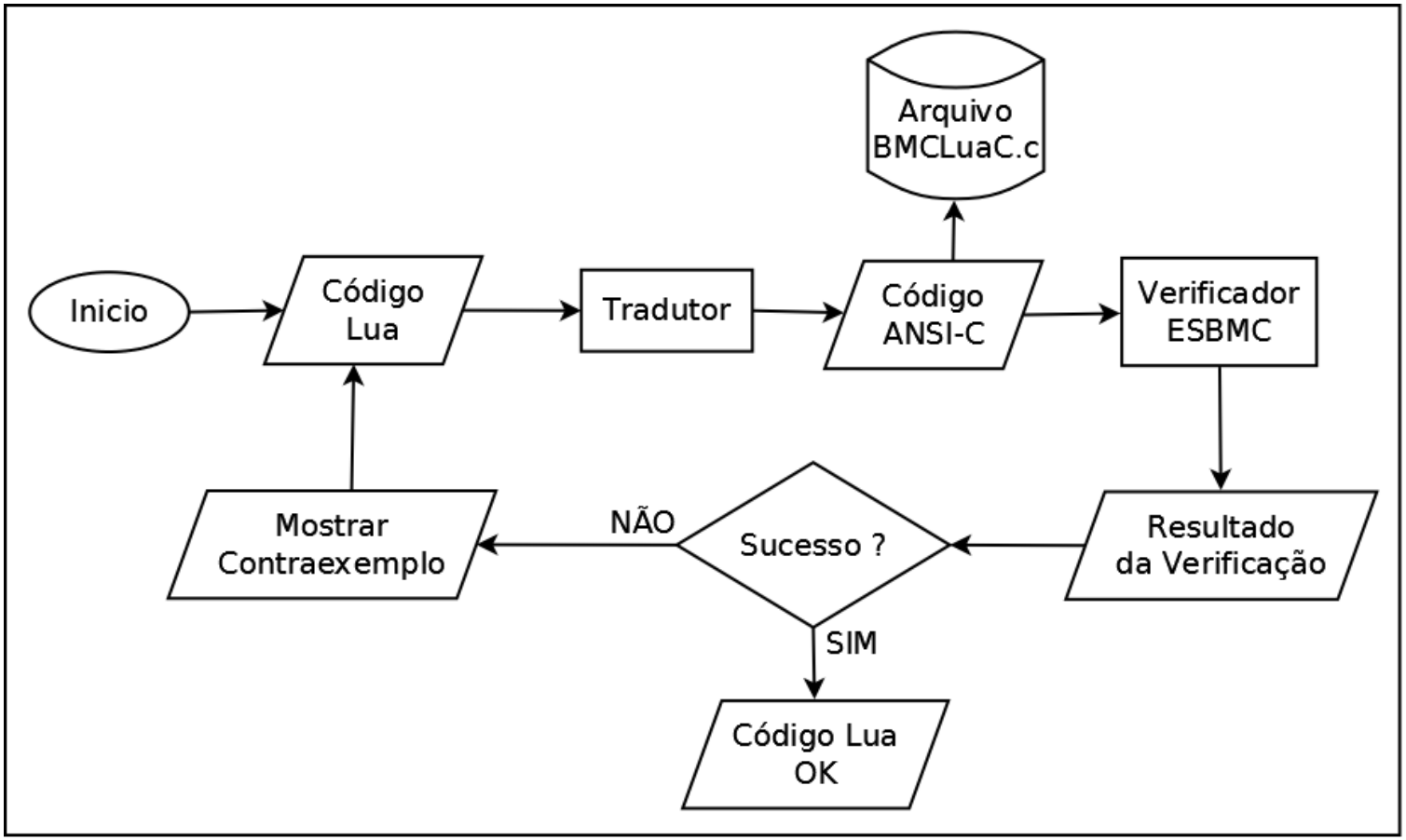
UFAM

- Estender os benefícios da verificação de modelos limitada para códigos da linguagem Lua:
  - Traduzir códigos Lua para uma linguagem modelo (ANSI-C)
  - Realizar a validação de códigos Lua através do verificador de modelos limitado ESBMC

- Implementar a metodologia BMCLua:
  - Desenvolver um Ambiente de Desenvolvimento Integrado
  - Incorporar um interpretador Lua
  - Incorporar o verificador de modelos ESBMC

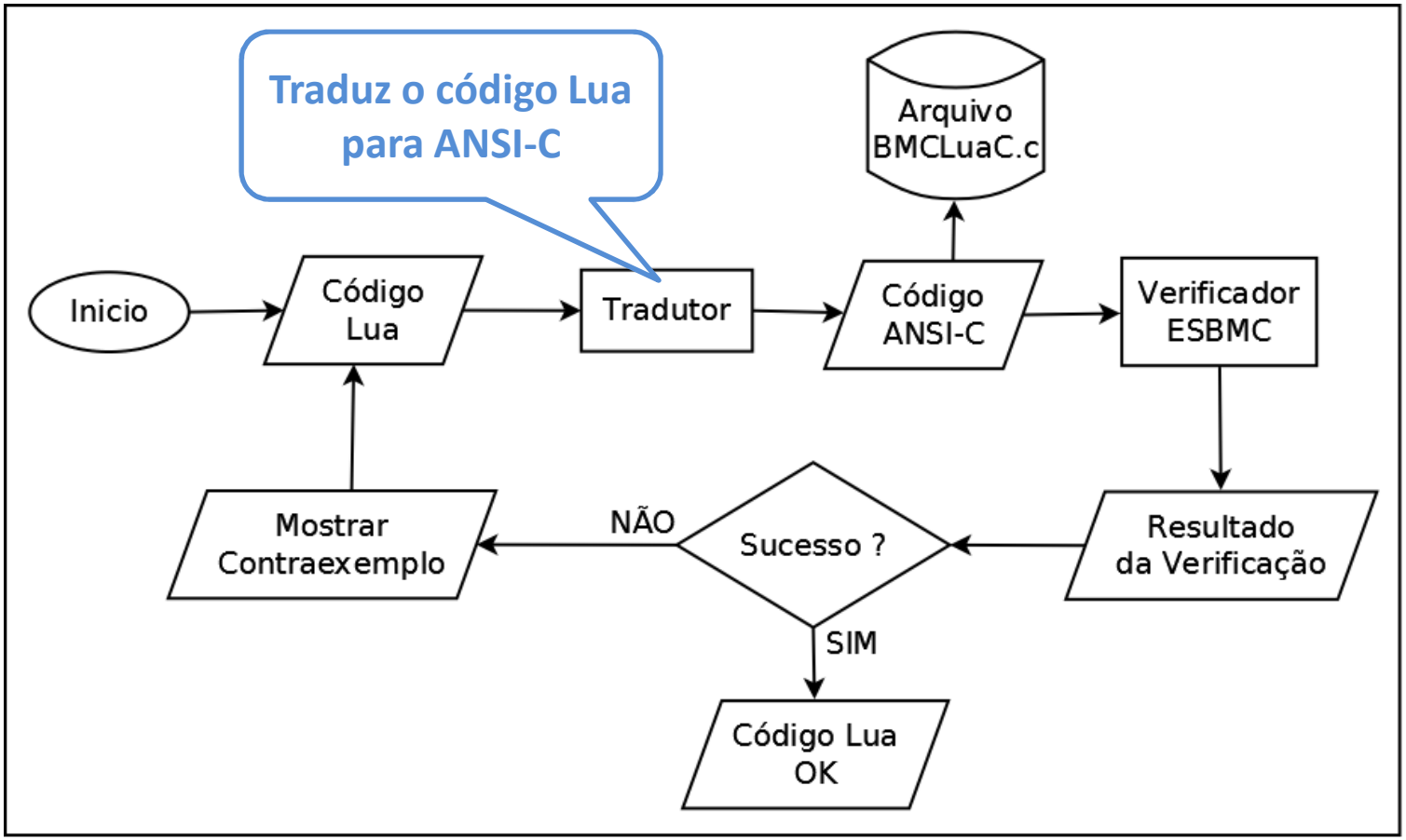
# A Metodologia BMCLua

- Composta basicamente de um tradutor e o verificador ESBMC



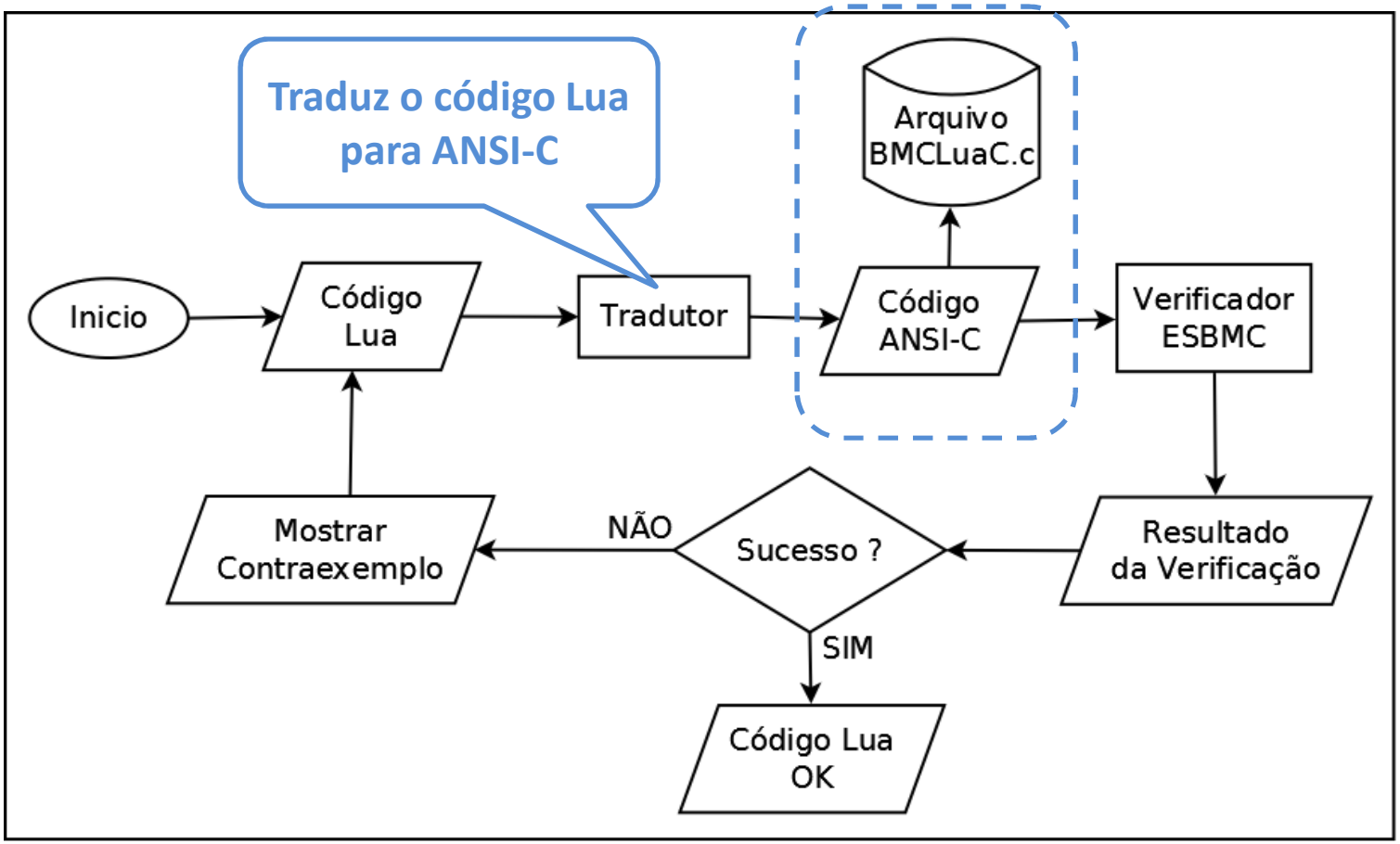
# A Metodologia BMCLua

- Composta basicamente de um tradutor e o verificador ESBMC



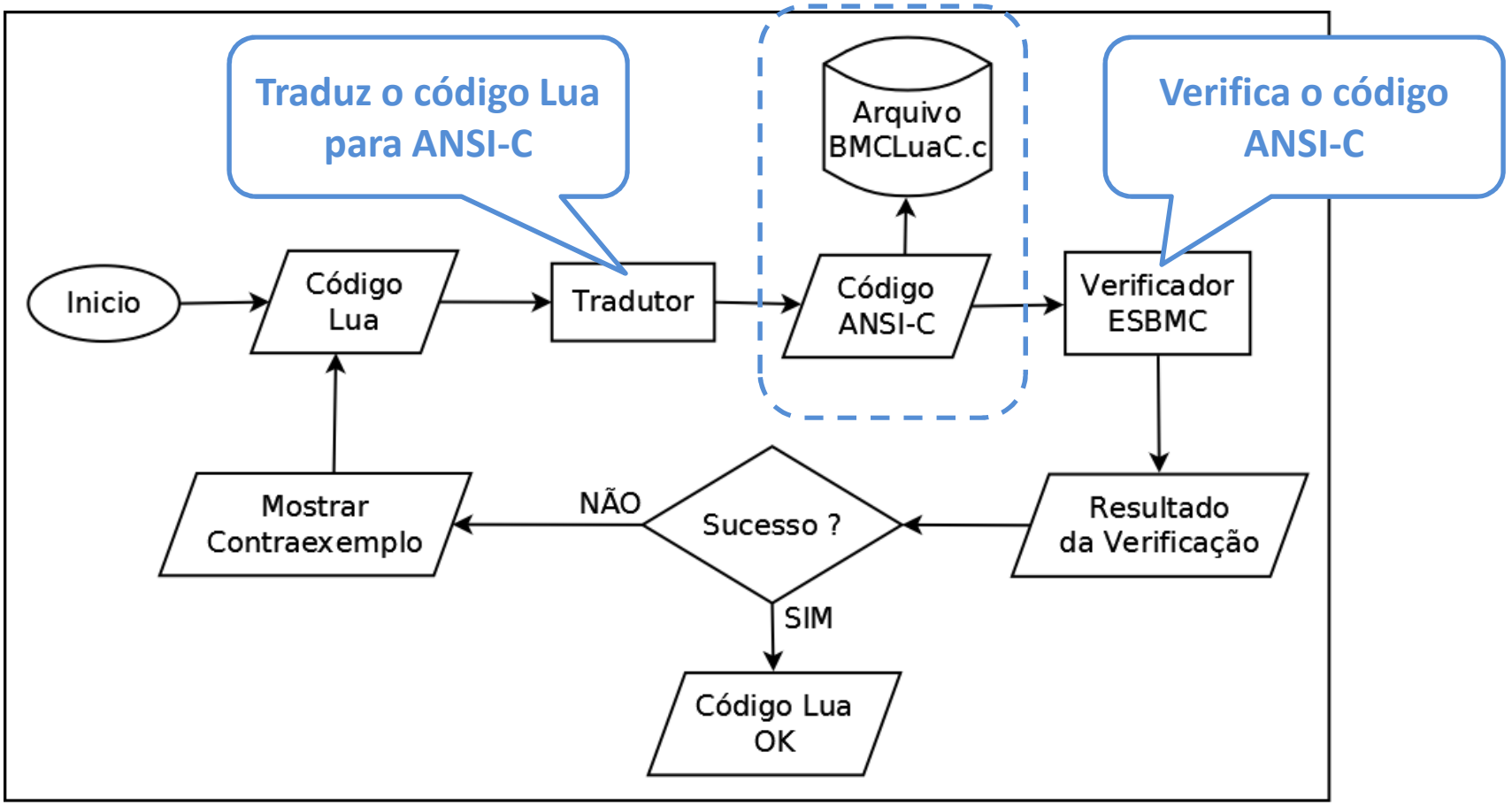
# A Metodologia BMCLua

- Composta basicamente de um tradutor e o verificador ESBMC



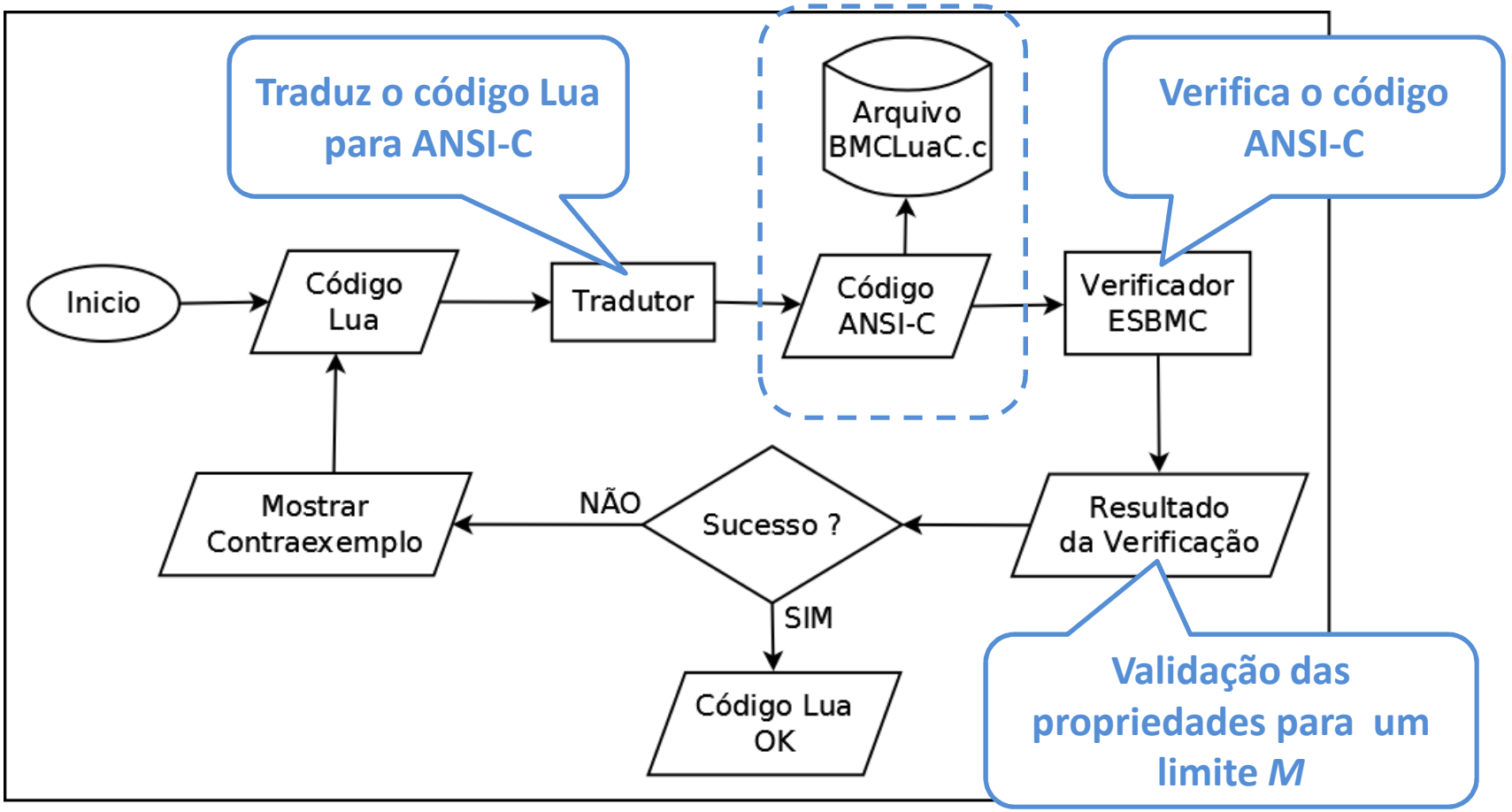
# A Metodologia BMCLua

- Composta basicamente de um tradutor e o verificador ESBMC



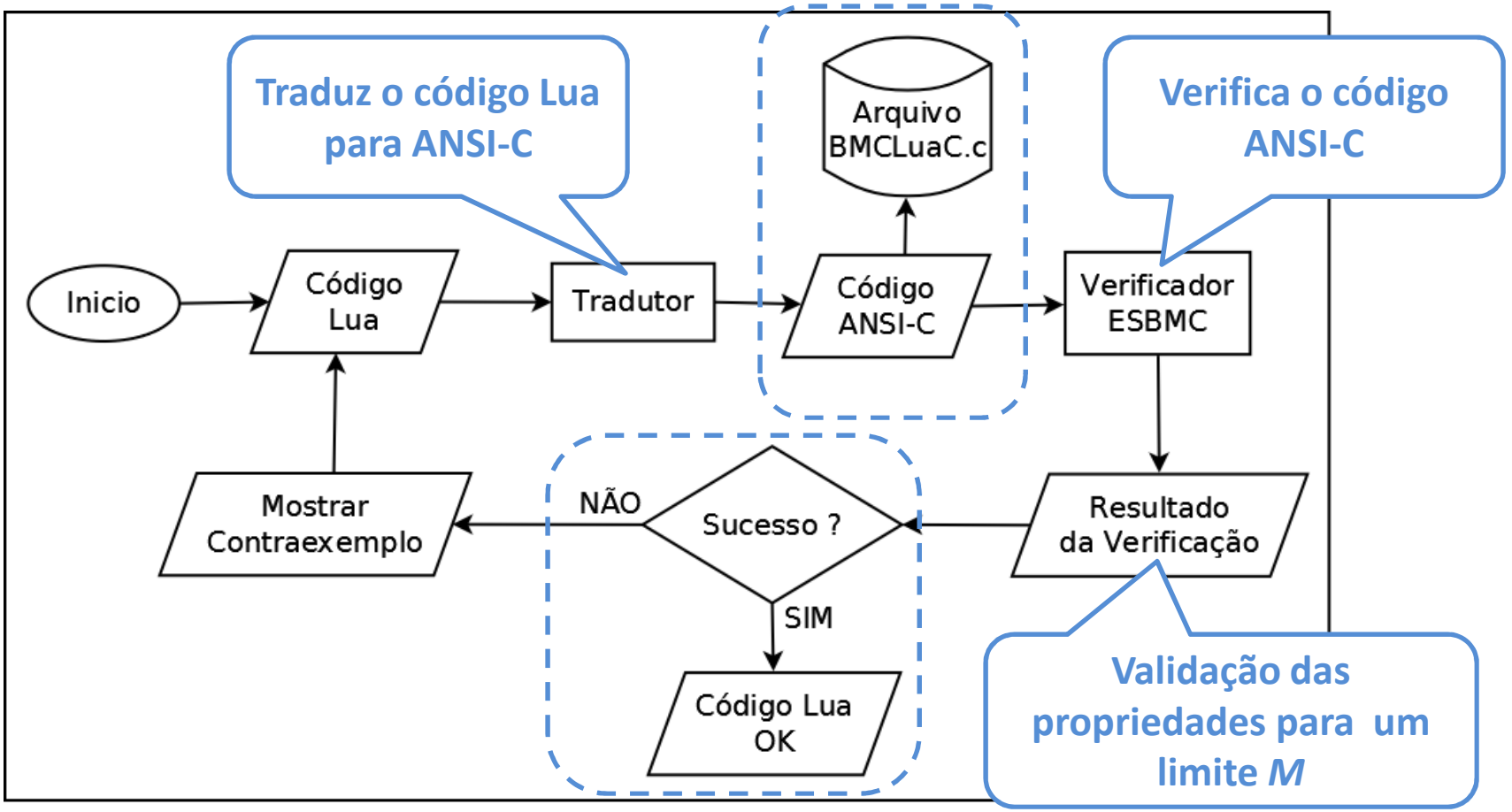
# A Metodologia BMCLua

- Composta basicamente de um tradutor e o verificador ESBMC



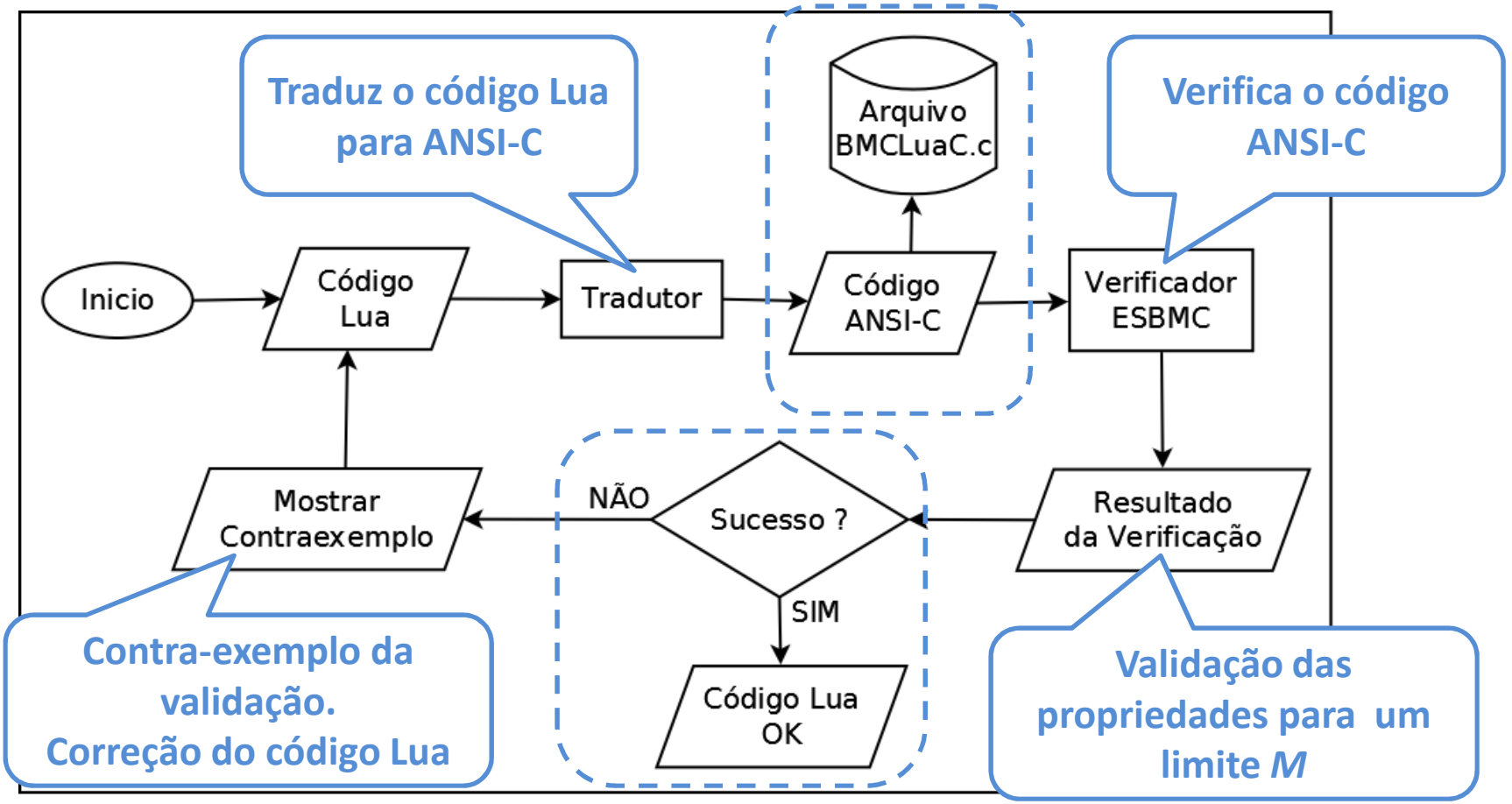
# A Metodologia BMCLua

- Composta basicamente de um tradutor e o verificador ESBMC



# A Metodologia BMCLua

- Composta basicamente de um tradutor e o verificador ESBMC



# Tradução e Verificação no BMCLua

```
n = 5
while n >= 0 do
   print(4/n)
   n = n - 1
end
```

```
#include <stdio.h>
void main(void){
   int n = 5;
   while(n >= 0){
      printf("%f",4/n);
      n = n - 1;
   }
}
```

```
Violated property:
  file /home/BMCLua/BMCLuaC.c
  division by zero
  n != 0

  VERIFICATION FAILED
```

- A versão atual traduz apenas um subconjunto de comandos da linguagem Lua.
  - print
  - break
  - return
  - if ... else ... end
  - while ... do ... end
  - for ... do ... end
  - repeat ... until
  - do ... end

# Tradução e Verificação no BMCLua

```
n = 5
while n >= 0 do
   print(4/n)
   n = n - 1
end
```

Tradução

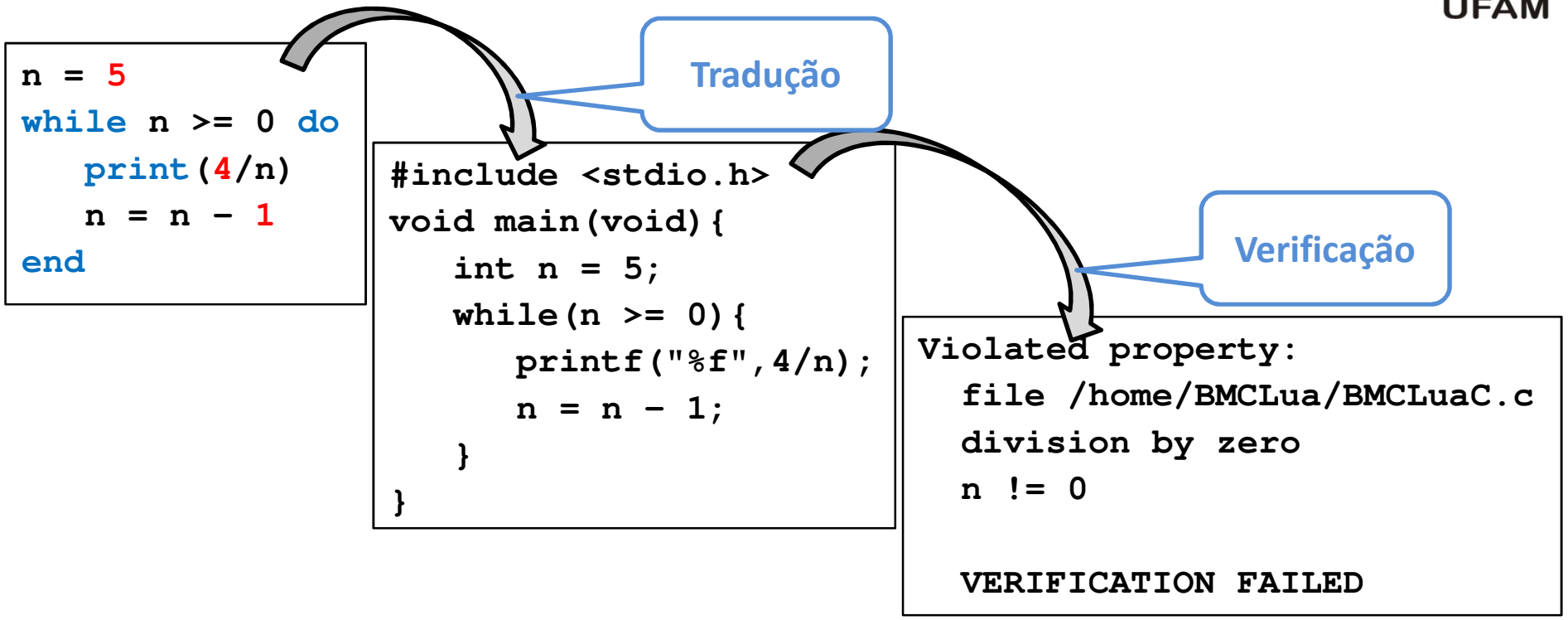
```
#include <stdio.h>
void main(void){
   int n = 5;
   while(n >= 0){
      printf("%f",4/n);
      n = n - 1;
   }
}
```

```
Violated property:
  file /home/BMCLua/BMCLuaC.c
  division by zero
  n != 0

  VERIFICATION FAILED
```

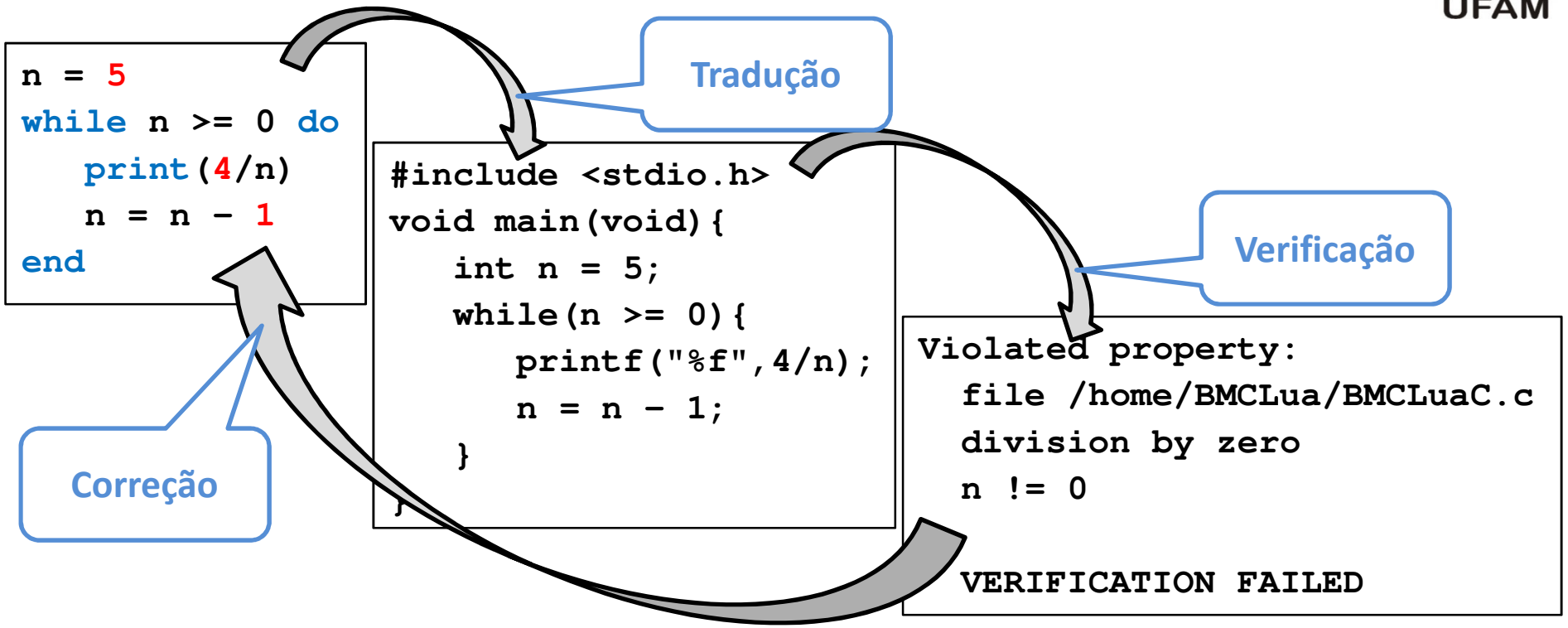
- A versão atual traduz apenas um subconjunto de comandos da linguagem Lua.
  - print
  - break
  - return
  - if ... else ... end
  - while ... do ... end
  - for ... do ... end
  - repeat ... until
  - do ... end

# Tradução e Verificação no BMCLua




- A versão atual traduz apenas um subconjunto de comandos da linguagem Lua.
  - print
  - break
  - return
  - if ... else ... end
  - while ... do ... end
  - for ... do ... end
  - repeat ... until
  - do ... end

# Tradução e Verificação no BMCLua




- A versão atual traduz apenas um subconjunto de comandos da linguagem Lua.
  - print
  - break
  - return
  - if ... else ... end
  - while ... do ... end
  - for ... do ... end
  - repeat ... until
  - do ... end

# Configuração e *Benchmarks*


- Ambiente:
  - Intel Core i3 2.5 GHz com 2 GB de RAM na plataforma Linux Ubuntu 32-bits
  - ESBMC v1.21 com solucionador z3 v3.2

- Utilizado *benchmarks* para testes de desempenho e precisão:
  - Bellman-Ford
  - Prim
  - BubbleSort
  - SelectionSort

# Resultados Experimentais (1)

UFAM

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 6 | 1 | < 1 | < 1 |
| | 10 | 40 | 11 | 1 | < 1 | < 1 |
| | 15 | 40 | 16 | 1 | < 1 | < 1 |
| | 20 | 40 | 21 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 5 | 1 | < 1 | < 1 |
| | 5 | 61 | 6 | 1 | < 1 | < 1 |
| | 6 | 61 | 7 | 1 | < 1 | < 1 |
| | 7 | 61 | 8 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (1)

Total de elementos do array

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 6 | 1 | < 1 | < 1 |
| | 10 | 40 | 11 | 1 | < 1 | < 1 |
| | 15 | 40 | 16 | 1 | < 1 | < 1 |
| | 20 | 40 | 21 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 5 | 1 | < 1 | < 1 |
| | 5 | 61 | 6 | 1 | < 1 | < 1 |
| | 6 | 61 | 7 | 1 | < 1 | < 1 |
| | 7 | 61 | 8 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (1)

Total de elementos do array

Total de linhas de código traduzido

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 6 | 1 | < 1 | < 1 |
| | 10 | 40 | 11 | 1 | < 1 | < 1 |
| | 15 | 40 | 16 | 1 | < 1 | < 1 |
| | 20 | 40 | 21 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 5 | 1 | < 1 | < 1 |
| | 5 | 61 | 6 | 1 | < 1 | < 1 |
| | 6 | 61 | 7 | 1 | < 1 | < 1 |
| | 7 | 61 | 8 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (1)

Total de elementos do array

Total de linhas de código traduzido

Limite de iterações de loops realizada

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 5 | 1 | < 1 | < 1 |
| | 10 | 40 | 1 | 1 | < 1 | < 1 |
| | 15 | 40 | 3 | 1 | < 1 | < 1 |
| | 20 | 40 | 4 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 6 | 1 | < 1 | < 1 |
| | 5 | 61 | 6 | 1 | < 1 | < 1 |
| | 6 | 61 | 7 | 1 | < 1 | < 1 |
| | 7 | 61 | 8 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (1)

Total de elementos do array

Total de linhas de código traduzido

Limite de iterações de loops realizada

Total de propriedades verificadas

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 5 | 1 | < 1 | < 1 |
| | 10 | 40 | 1 | 1 | < 1 | < 1 |
| | 15 | 40 | 1 | 1 | < 1 | < 1 |
| | 20 | 40 | 1 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 3 | 1 | < 1 | < 1 |
| | 5 | 61 | 3 | 1 | < 1 | < 1 |
| | 6 | 61 | 3 | 1 | < 1 | < 1 |
| | 7 | 61 | 3 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (1)


Total de elementos do array

Tempo (seg) de processamento no BMCLua

Total de linhas de código traduzido

Limite de iterações de loops realizada

Total de propriedades verificadas

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 5 | 1 | < 1 | < 1 |
| | 10 | 40 | 1 | 1 | < 1 | < 1 |
| | 15 | 40 | 1 | 1 | < 1 | < 1 |
| | 20 | 40 | 1 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 3 | 1 | < 1 | < 1 |
| | 5 | 61 | 3 | 1 | < 1 | < 1 |
| | 6 | 61 | 3 | 1 | < 1 | < 1 |
| | 7 | 61 | 3 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (1)


Total de elementos do array

Tempo (seg) de processamento no BMCLua

Total de linhas de código traduzido

Limite de iterações de loops realizada

Tempo (seg) de processamento no ESBMC

Total de propriedades verificadas

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 6 | 1 | < 1 | < 1 |
| | 10 | 40 | 1 | 1 | < 1 | |
| | | | | 1 | | |
| | 20 | 40 | | 1 | | |
| Prim | 4 | | | 1 | < 1 | < 1 |
| | 5 | | | 1 | < 1 | < 1 |
| | 6 | 61 | | | | < 1 |
| | 7 | 61 | | | | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (2)

UFAM

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | 5 | 40 | 6 | 1 | < 1 | < 1 |
| | 10 | 40 | 11 | 1 | < 1 | < 1 |
| | 15 | 40 | 16 | 1 | < 1 | < 1 |
| | 20 | 40 | 21 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 5 | 1 | < 1 | < 1 |
| | 5 | 61 | 6 | 1 | < 1 | < 1 |
| | 6 | 61 | 7 | 1 | < 1 | < 1 |
| | 7 | 61 | 8 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

# Resultados Experimentais (2)

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| Bellman-Ford | | | | 1 | < 1 | < 1 |
| | | | | 1 | < 1 | < 1 |
| | 15 | 40 | 16 | 1 | < 1 | < 1 |
| | 20 | 40 | 21 | 1 | < 1 | < 1 |
| Prim | 4 | 61 | 5 | 1 | < 1 | < 1 |
| | 5 | 61 | 6 | 1 | < 1 | < 1 |
| | 6 | 61 | 7 | 1 | < 1 | < 1 |
| | 7 | 61 | 8 | 1 | < 1 | < 1 |
| | 8 | 61 | 9 | 1 | < 1 | < 1 |

Os tempos são equivalentes do programa Lua e o C original

# Resultados Experimentais (3)

UFAM

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| BubbleSort | 12 | 28 | 13 | 1 | < 1 | < 1 |
| | 35 | 28 | 36 | 1 | 2 | 2 |
| | 50 | 28 | 51 | 1 | 5 | 5 |
| | 70 | 28 | 71 | 1 | 10 | 10 |
| | 140 | 28 | 141 | 1 | 56 | 52 |
| | 200 | 28 | 201 | 1 | 203 | 163 |
| SelectioSort | 12 | 31 | 13 | 1 | < 1 | < 1 |
| | 35 | 31 | 36 | 1 | 1 | 1 |
| | 50 | 31 | 51 | 1 | 2 | 2 |
| | 70 | 31 | 71 | 1 | 5 | 4 |
| | 140 | 31 | 141 | 1 | 39 | 25 |
| | 200 | 31 | 201 | 1 | 175 | 89 |

# Resultados Experimentais (3)

UFAM

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| BubbleSort | 12 | 28 | 13 | 1 | < 1 | < 1 |
| | 35 | 28 | 36 | 1 | 2 | 2 |
| | 50 | 28 | 51 | 1 | 5 | 5 |
| | 70 | 28 | 71 | 1 | 10 | 10 |
| | 140 | 28 | 141 | 1 | 56 | 52 |
| | 200 | 28 | 201 | 1 | 203 | 163 |
| SelectioSort | 12 | 31 | 13 | 1 | < 1 | < 1 |
| | 35 | 31 | 36 | 1 | 1 | 1 |
| | 50 | 31 | 51 | 1 | 2 | 2 |
| | 70 | 31 | 71 | 1 | 5 | 4 |
| | 140 | 31 | 141 | 1 | 39 | 25 |
| | 200 | 31 | 201 | 1 | 175 | 89 |

# Resultados Experimentais (3)

UFAM

| Algoritmo | E | L | B | P | TL | TE |
|---|---|---|---|---|---|---|
| BubbleSort | 12 | 28 | 13 | 1 | < 1 | < 1 |
| | 35 | 28 | 36 | 1 | 2 | 2 |
| | | | 51 | 1 | 5 | 5 |
| | | | 71 | 1 | 10 | 10 |
| | | | | 1 | 56 | 52 |
| | 200 | 28 | 201 | 1 | 203 | 163 |
| SelectioSort | 12 | 31 | 13 | 1 | < 1 | < 1 |
| | 35 | 31 | 36 | 1 | 1 | 1 |
| | 50 | 31 | 51 | 1 | 2 | 2 |
| | 70 | 31 | 71 | 1 | 5 | 4 |
| | 140 | 31 | 141 | 1 | 39 | 25 |
| | 200 | 31 | 201 | 1 | 175 | 89 |

Maior número de variáveis
Maior o tempo de processamento

# Conclusões


• O IDE BMCLua foi capaz de verificar violações de propriedades em códigos Lua

• Falta o tradutor do BMCLua incorporar outras estruturas e funcionalidades da linguagem Lua / NCLua

# Trabalhos Futuros

• Adicionar uma gramática formal para analisadores (*parser* e *lexer*) de trechos de códigos com o ANTLR

• Incorporar ao *middleware* Ginga e implementar um *plug-in* para o IDE Eclipse