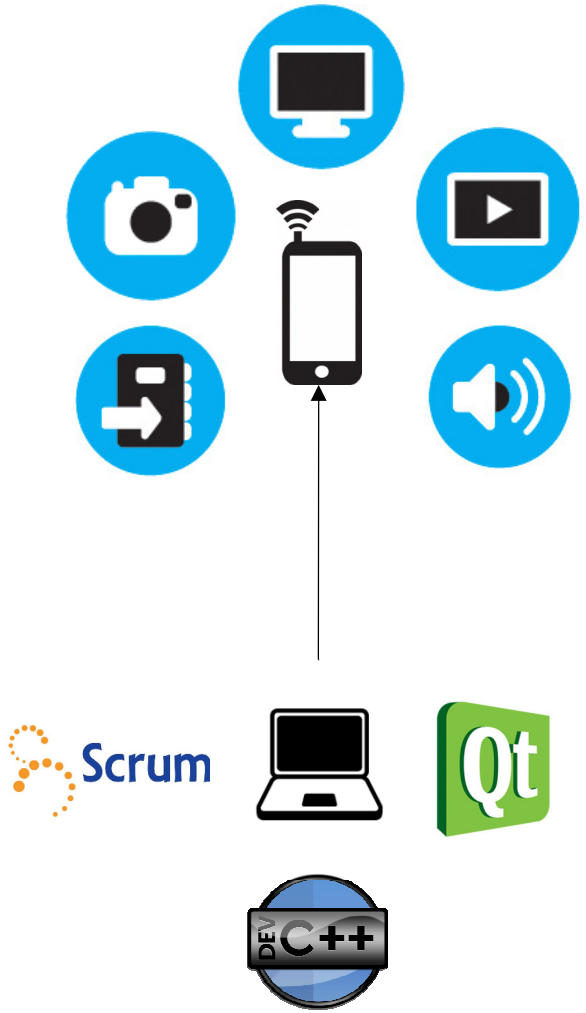
IV Encontro Regional de Computação e Sistemas da Informação

Verificação de Programas C++ Baseados no *Framework* Multiplataforma Qt

Felipe R. M. Sousa, Lucas C. Cordeiro e Eddie B. L. Filho
felipemonteiro@ufam.edu.br

Sistemas Embarcados

- Sistemas embarcados estão se tornando cada vez mais complexos
  - Processadores com vários núcleos de processamento
  - Memória compartilhada
- Empresas buscam diversas alternativas para acelerar o desenvolvimento de tais sistemas e potencializar seu desempenho
  - Conjunto reutilizável de classes: *framework* Qt



Motivação

- Empresas da área de sistemas embarcados procuram formas mais rápidas e baratas de verificar seus sistemas [Berard *et al.* 2010]
  - Em especial, o Pólo Industrial de Manaus tem focado grande parte dos recursos gerados através da lei de informática no desenvolvimento de aplicativos móveis
  - Verificação formal

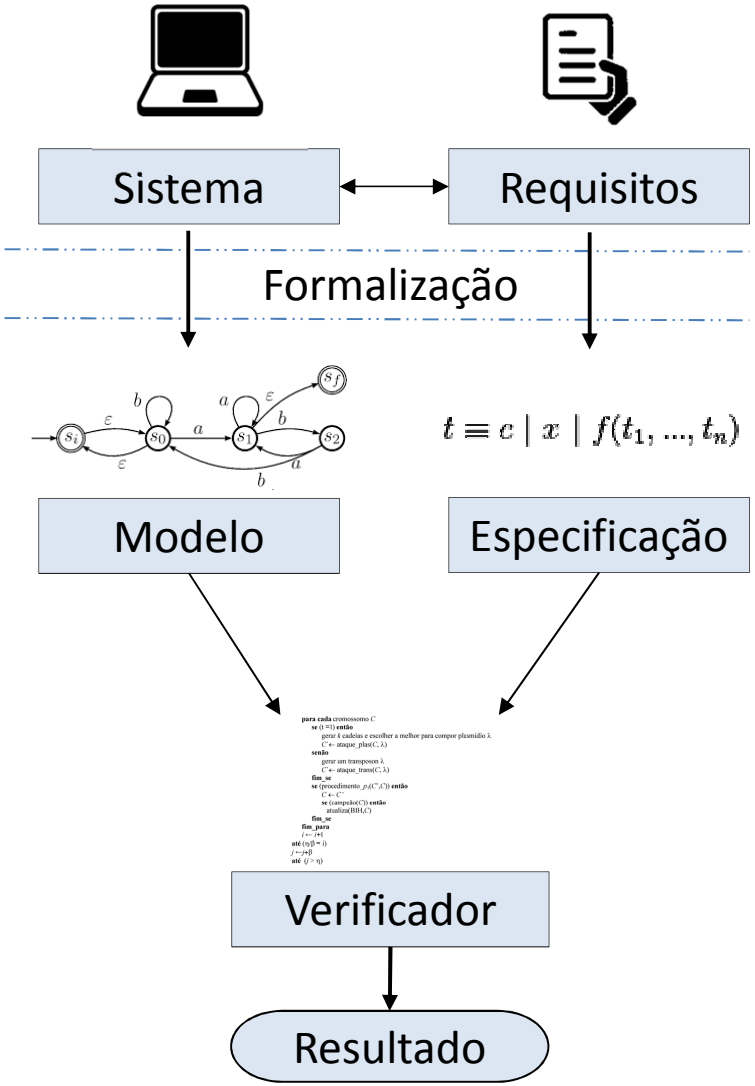
- Contudo, muitos sistemas não podem ser verificados de forma automática
  - O verificador *Java PathFinder* é capaz de verificar aplicações Java, mas não suporta a verificação das quais utilizam o sistema operacional Android [NASA, 2007]

Objetivos deste Trabalho

Implementar uma técnica capaz de verificar propriedades em programas C++ baseados no *framework* multiplataforma Qt

- Investigar propriedades específicas relacionadas ao *framework* Qt
- Estender as funcionalidades do verificador de software *Efficient SMT-based Context-Bounded Model Checker* (ESBMC) para o suporte do *framework* Qt
- Desenvolver uma suíte de teste automatizada para validar as implementações realizadas

Verificação Formal

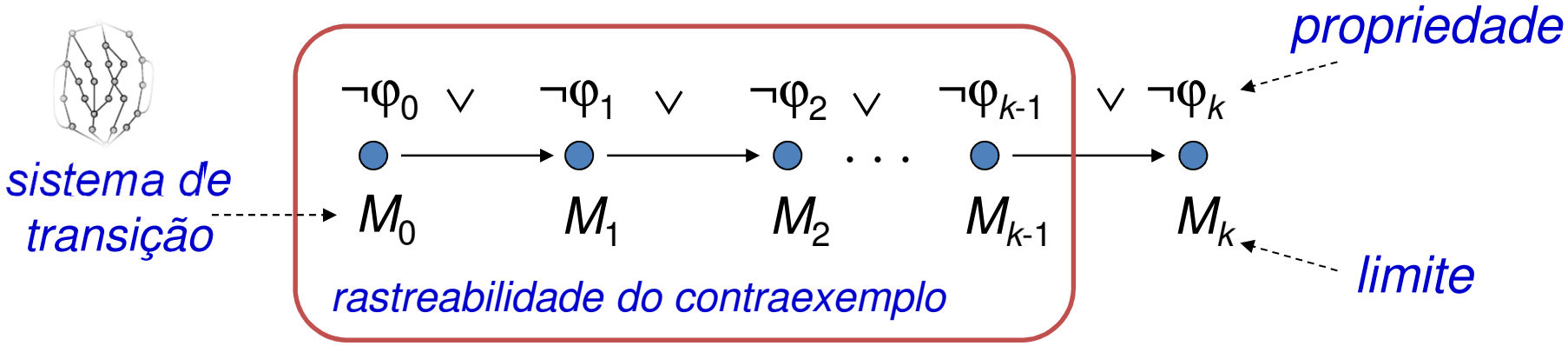


- **Ideia básica:** provar, matematicamente, a conformidade de um determinado algoritmo, com relação a uma determinada propriedade, através de métodos formais [Clarke *et al.* 1999]
  - Verificação Dedutiva
  - Verificação de Modelos

- Técnicas aplicadas na verificação formal:
  - ***Satisfiability Modulo Theories* (SMT):** refere-se ao problema de determinar se uma fórmula em lógica de primeira ordem pode ser satisfeita no que diz respeito a alguma teoria lógica
  - ***Bounded Model Checking* (BMC):** algoritmos para explorar o espaço de estado de um sistema de transição para determinar se ele obedece a certa especificação.

Verificação de Modelos Limitada

- Do inglês *Bounded Model Checking* (BMC), a técnica tem como ideia básica checar a negação de um determinada propriedade a uma determinada profundidade



- sistema de transição *M* desenrolado *k* vezes
  - para programas: *loops*, vetores, ...
- traduzido em uma condição de verificação ψ tal que

  ψ é satisfatível sse φ tem um contraexemplo de profundidade menor ou igual a *k*
- tem sido aplicada com sucesso na verificação de sistemas (embarcados)

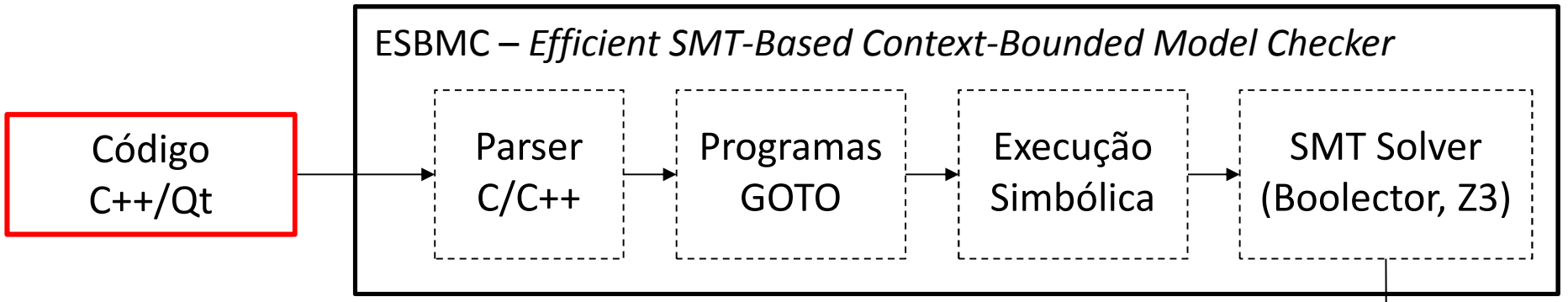
ESBMC

- Reconhecido internacionalmente pela sua robustez e eficácia na verificação de programas
  - Programas sequenciais e concorrentes em ANSI-C e C++
  - Competição Internacional de Verificação de Software
    - (SV-COMP 2012-2015)

- Utiliza teorias do módulo da satisfabilidade aliadas as técnicas de BMC para verificar propriedades em códigos ANSI-C/C++
  - Tempo de verificação pode variar dependendo da quantidade de *loops* e *interleavings*
  - *under*- e *overflow* aritmético, segurança de ponteiros, limite de *arrays*, divisão por zero, vazamento de memória, violações de atomicidade e ordem, *deadlock,* corrida de dados e **assertivas definidas pelo usuário**

- **Desafio:** suporte do *framework* Qt

Funcionamento do ESBMC




```
#include <QLinkedList>

#define UPPER_BOUND 2
#define LOWER_BOUND 0

int main(int argc, char *argv[]) {
   QLinkedList<int> integerList;

   for (int i = LOWER_BOUND; i < UPPER_BOUND; i++)
      integerList.push_back(i);

   for (int j = LOWER_BOUND; j <= UPPER_BOUND; j++)
      integerList.pop_back();
}
```

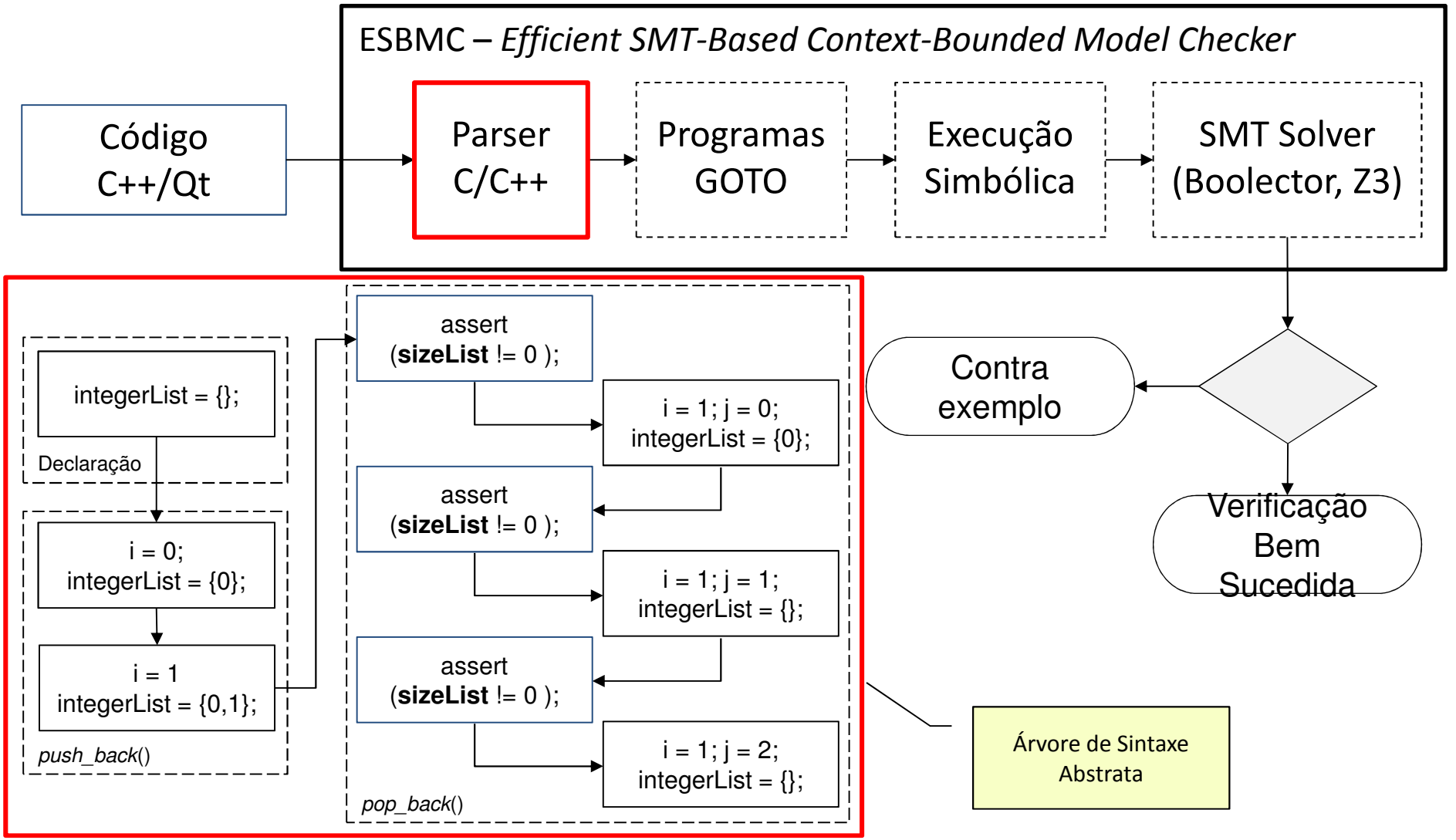
Definindo os limites dos *loops*

Inserindo elementos na lista

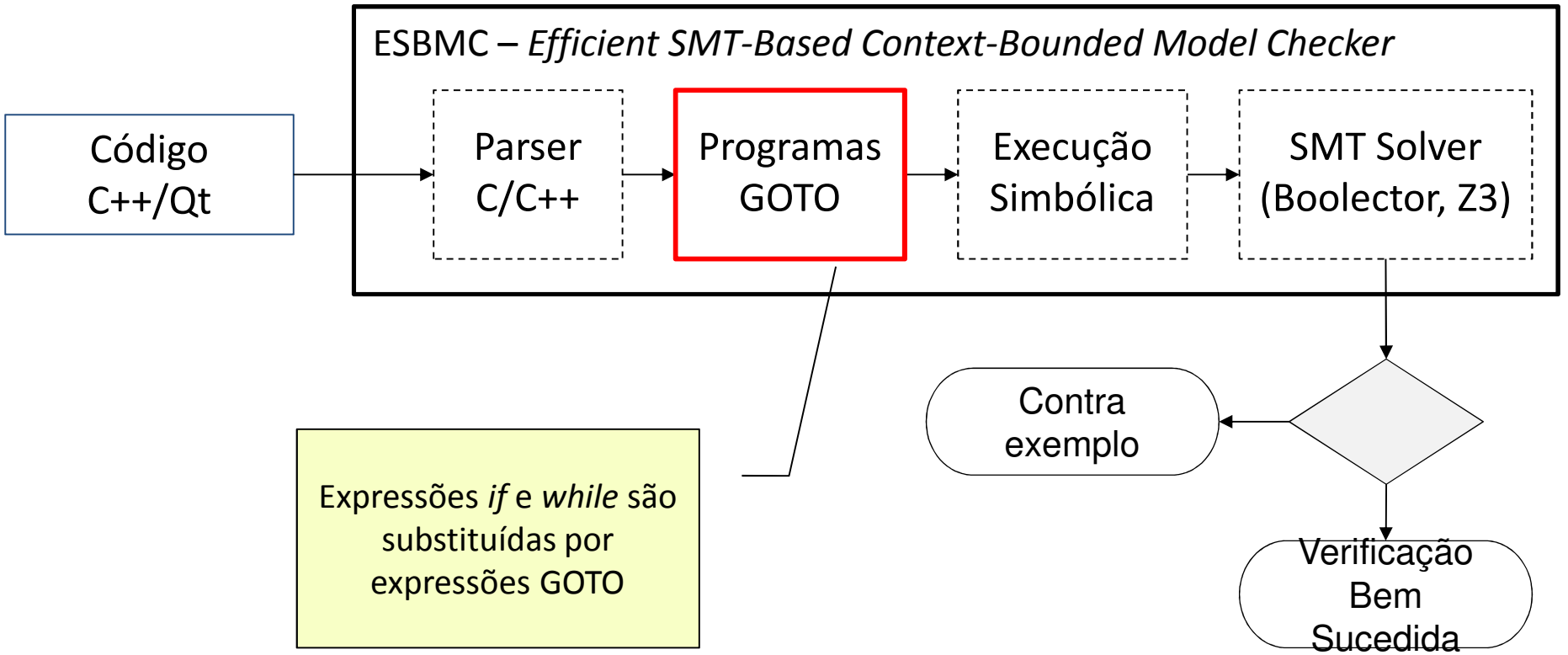
Removendo Elementos da lista

Tentando remover mais elementos do que foram inseridos

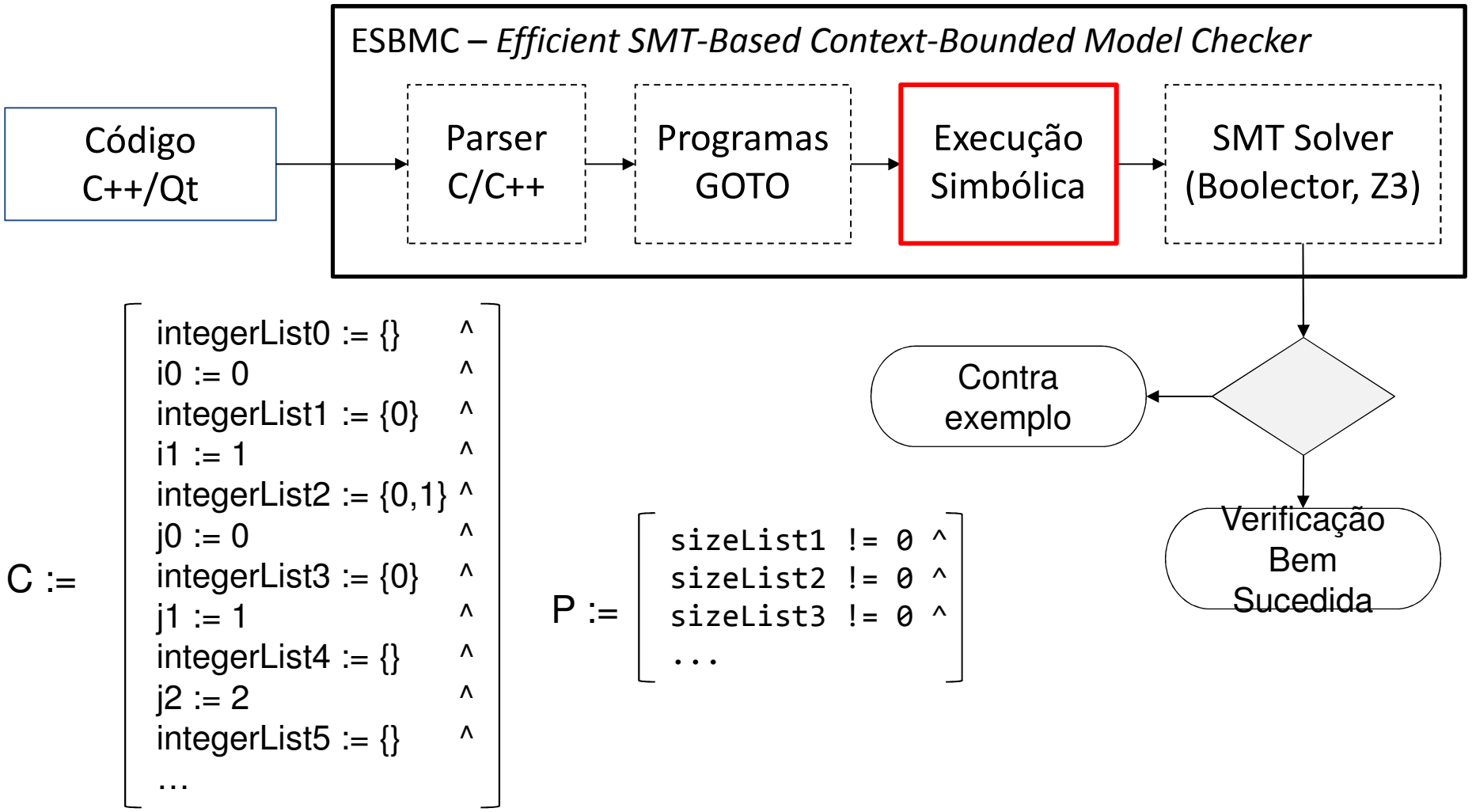
Funcionamento do ESBMC



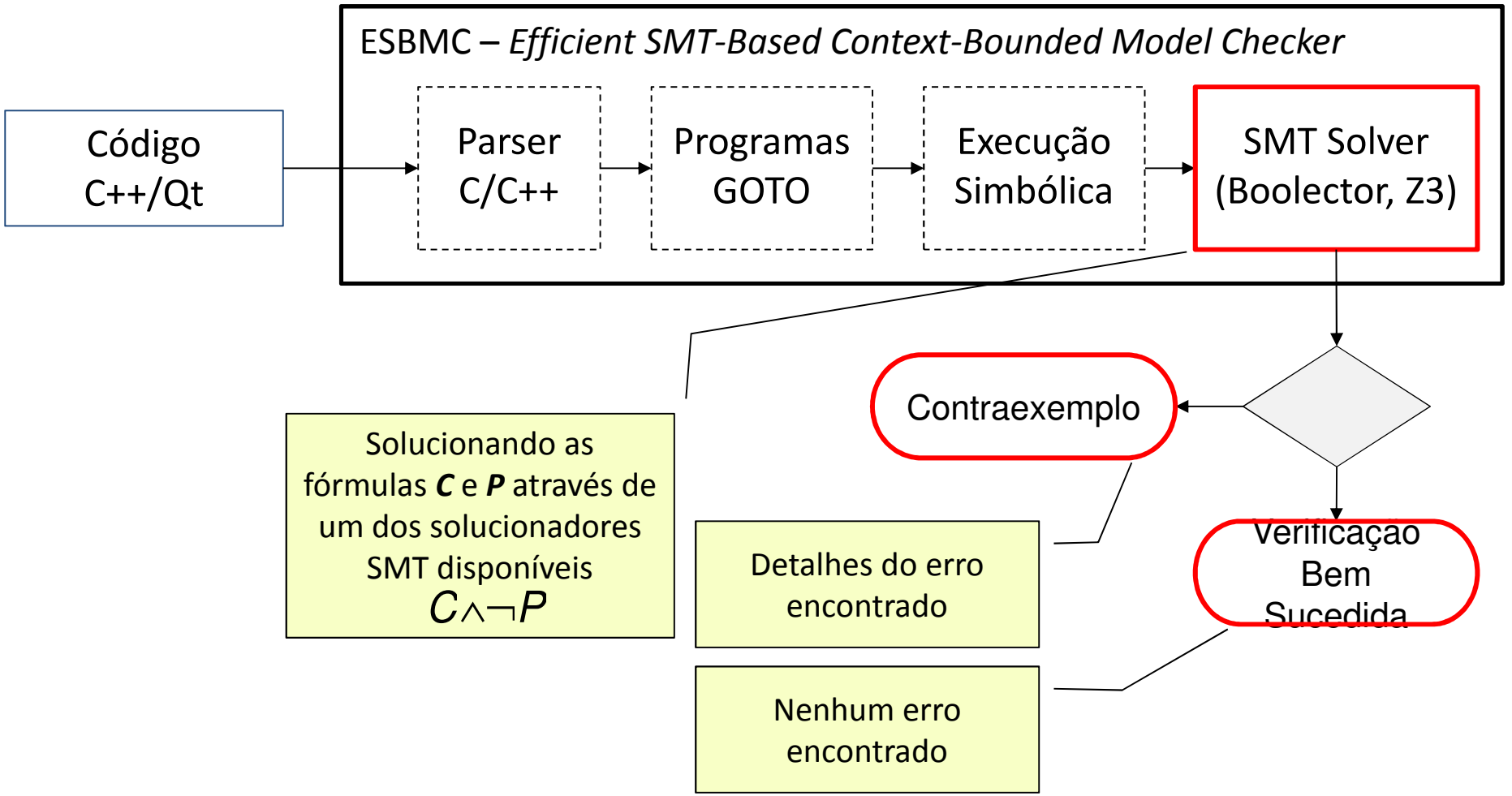
Funcionamento do ESBMC



Funcionamento do ESBMC



Funcionamento do ESBMC



Modelo Operacional

- A utilização de um modelo operacional, representando o *framework* Qt, configura uma abordagem mais viável para verificação de tais programas
  - Totalmente desenvolvido em C++
  - Diminui o custo computacional
  - Baseado na documentação do *framework* Qt 5.5
- Garantir a verificação de propriedades relacionadas com as estruturas do *framework* Qt
  - Inclusão de pré- e pós-condições no modelo operacional

Pré-condições

- Verificar as condições mínimas para executar um determinado método/função
  - Trecho de código retirado da aplicação *Animated Tiles* [Qt Project Hosting, 2012]

```
QTimer timer;
timer.start(125);
timer.setSingleShot(true);
trans = rootState->addTransition(&timer,
                                 SIGNAL(timeout()),
                                 ellipseState);
trans->addAnimation(group);
```

Pré-condições

- Verificar as condições mínimas para executar um determinado método/função
  - Modelo operacional referente a classe *QTimer*

```
class Qtimer {
public:
    Qtimer (){}
    QTimer ( QMainWindow* window ){}
    void start ( int i ) {
        __ESBMC_assert(i >= 0 , “Specified time invalid.”);
    }
    void setSingleShot( bool b ){}
    void start(){}
        ...
};
```

Pós-condições

- Verificar todas as propriedades após a execução de um determinado método/função
  - Código exemplificando o uso do *container QList* presente no módulo *QtCore* do *framework* Qt

```
#include <cassert>
#include <QList>
using namespace std;
int main () {
    QList<int> mylist;
    mylist.push_front(200);
    mylist.push_front(300);
    assert(mylist.front() == 300);
    return 0;
}
```

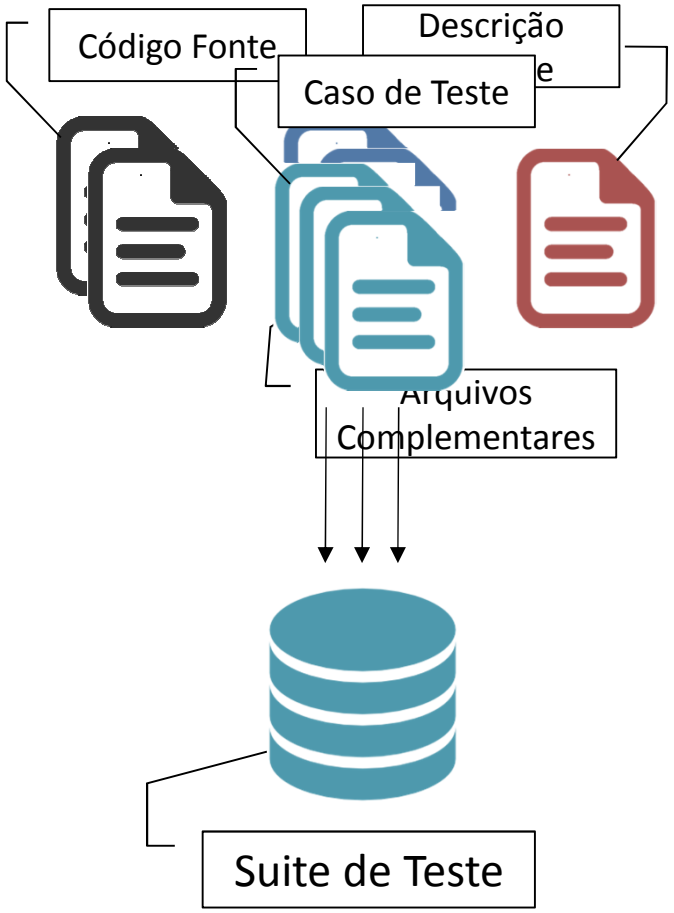
Pós-condições

- Verificar todas as propriedades após a execução de um determinado método/função
  - Modelo operacional com a simulação do comportamento do método *push_front*

```
void push_front( const value_type& x ){
    if(this->_size != 0) {
        for(int i = this->_size -1; i > -1; i++)
            this->_list[i+1] = this->_list[i];
    }
    this->_list[0] = x;
    this->_size++;
}
```

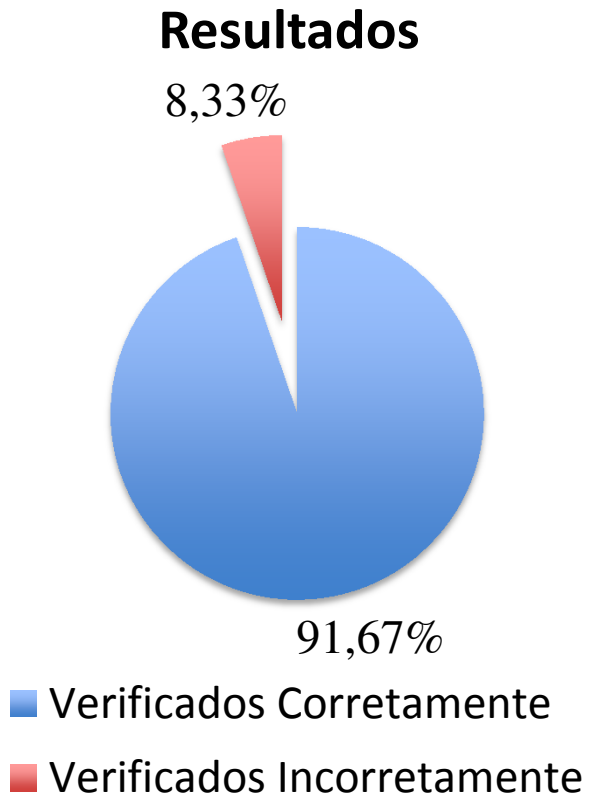
Avaliação Experimental

- **Objetivo:** validar a corretude e desempenho do modelo operacional
- Criação de uma suíte de casos de teste automatizada
  - 52 *benchmarks*
  - 26 casos de teste com *bug*
  - 26 casos de teste sem *bug*
- *Setup* dos experimentos
  - ESBMC v1.20
  - Computador *Intel Core* i7-2600 com 3,40 GHz de *clock* e 24 GB de RAM



Avaliação Experimental

- A suíte de teste é verificada por completo em aproximadamente 7 minutos (414 segundos)
- Apresenta uma taxa de aproximadamente 92% de acerto
- Os casos não verificados corretamente apresentam um resultado “falso negativo”
  - Representação interna de ponteiros do verificador



Conclusão

- ESBMC representa o estado da arte em verificação
  - Único verificador de modelos para o *framework* Qt
- O modelo operacional aborda dois módulos do *framework* Qt: *QtGui* e *QtCore*
  - Modelos referentes a 128 bibliotecas do *framework* Qt
  - 91,67% de 52 *benchmarks* verificados corretamente em aproximadamente 7 minutos
- Trabalhos futuros
  - Expandir o modelo operacional e a suíte de casos de teste
- Todo trabalho está disponível em ***www.esbmc.org***

Dúvidas?

Muito obrigado pela atenção de todos.

felipemonteiro@ufam.edu.br