Extração de Modelos de Comportamento de Software

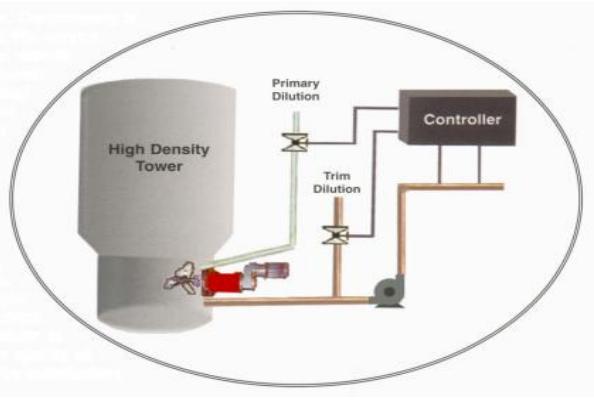
Lucio Mauro Duarte
lmduarte@inf.ufrgs.br
Grupo de VV & T
Instituto de Informática
Universidade Federal do Rio Grande do Sul

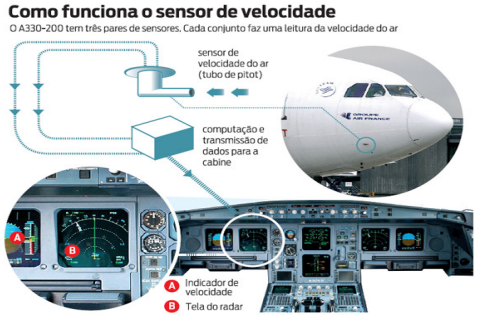
Workshop-Escola de Informática Teórica 2013
15/10/2013

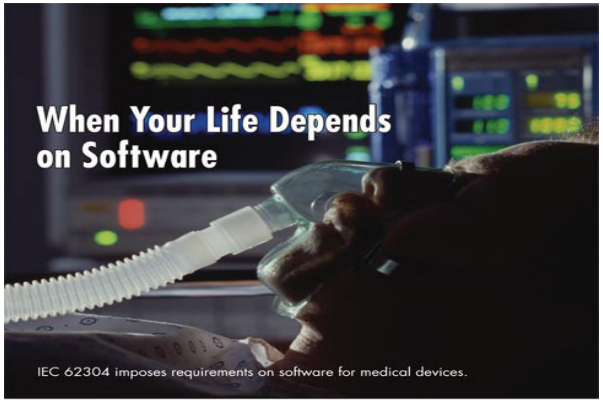
O que é software?

"Um **programa de computador**, a **documentação** sobre o seu modo de operação e os **dados de configuração** exigidos para que ele funcione corretamente." [Sommerville 2004]

Software na Vida Cotidiana







Software não é importante se...

- Você viaja assim:

Software não é importante se...

- E usa um computador deste modelo:

Desenvolvimento de Software

- Em geral
  1. Análise (requisitos, especificação)
  2. Projeto (modelagem)
  3. Validação (software é o correto?) e Verificação (software está correto?)
  4. Implementação
  5. Testes
  6. Manutenção (correção de erros e inclusão de modificações)

Comportamento Incorreto

- É possível afirmar-se que:
  - **Quase todas as partes** de um software apresentam **algum defeito** durante o desenvolvimento
  - Devido a limitações de orçamento e tempo, **maioria dos sistemas de software são postos em uso ainda com defeitos**
  - **Logo, há grande probabilidade de o software que usamos estar defeituoso!!!**

Software correto é possível?

Software correto é possível?

- Nem sempre é possível garantir correção total
- É possível, sim, aumentar a **confiança** na correção
  - Desenvolvimento rigoroso com o uso de técnicas de verificação

Principais Abordagens

- Teste de Software
- Verificação Formal
  - Análise Estática
  - Prova de Teoremas
  - Verificação de Modelos

Verificação de Modelos

- Totalmente **automática** ⇒ tecnologia *"push-button"*
- Exploração completa do espaço de estados
- Geração de **contraexemplo**
- Tem sido usada com sucesso tanto no âmbito acadêmico quanto por grandes empresas:
  - Intel
  - IBM
  - Microsoft
  - Boeing
  - NASA
  - ...

Processo de Verificação

- Etapas segundo [Clarke et al. 1999]:
  1. Especificação (requisitos ⇒ propriedades)
  2. Modelagem
  3. Verificação (Modelo respeita as propriedades do sistema?)

Resultados da Verificação

- Se verificação indicar violação das propriedades
  1. Geração do contraexemplo
  2. Realizar alterações
  3. Refazer verificação
- Processo termina quando não houver mais violações detectadas

Ferramentas

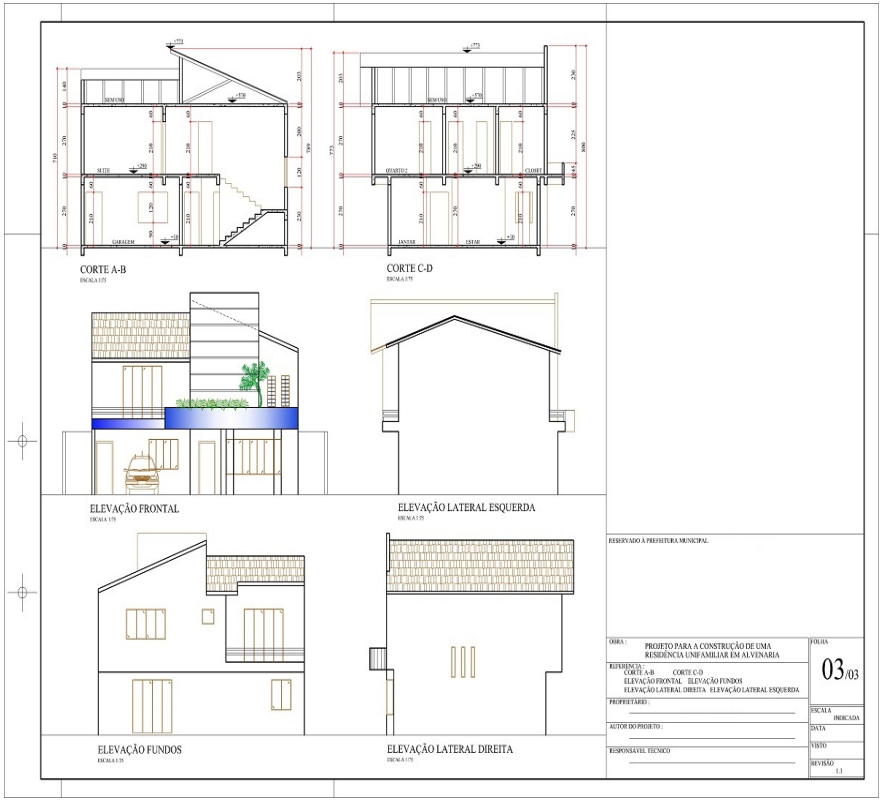
- Spin

  http://spinroot.com/spin/whatispin.html
- SMV

  http://embedded.eecs.berkeley.edu/Alumni/kenmcmil/smv/
- LTSA

  http://www.doc.ic.ac.uk/ltsa
- ...

O quê é um Modelo?

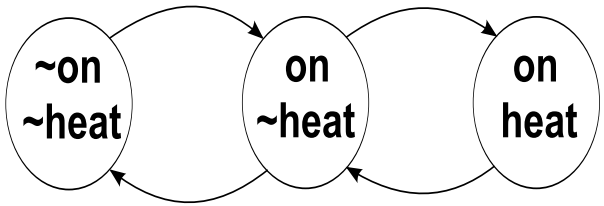


Modelos de Software

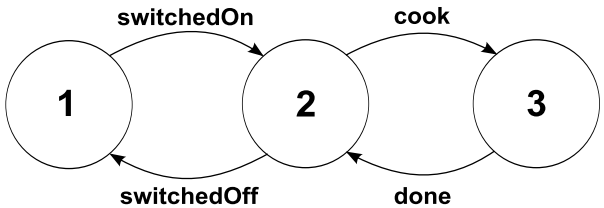
- São descrições **abstratas** de algum **aspecto** de um software
- Podem ser de diversos **tipos**, baseados em diversas **abstrações** e servirem a diferentes **propósitos**:
  - Diagrama de Classes
  - Control Flow Graph (CFG)
  - Data Flow Graph (DFG)
  - Message Sequence Chart (MSC)
  - Rede de Petri
  - Cadeias de Markov
  - Rede de Autômatos Estocásticos
  - Gramáticas de Grafos
  - Kripke Structure
  - Labelled Transition Systems (LTS)
  - ...

Modelos de Comportamento

- Foco em modelos de comportamento
  - Descrevem o comportamento esperado do software
  - Podem ser baseados em:
    - Estados (Kripke Structure)



    - Ações (LTS)

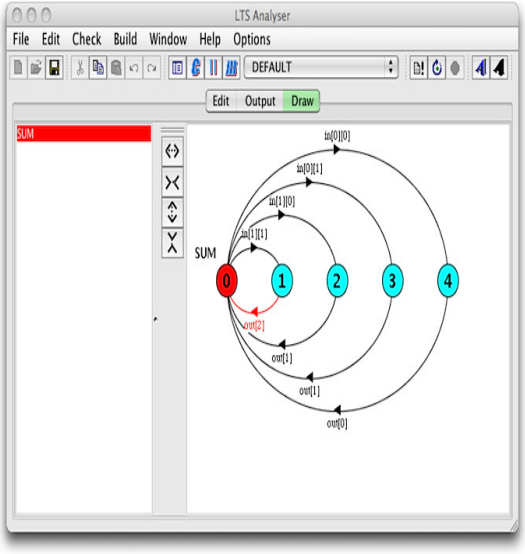


    - Estados e Ações (LKS)

Aplicação de Verificação de Modelos

- Questões:
  - De onde vem o modelo a ser verificado?
  - Como garantir que este modelo será corretamente implementado?

Código X Modelo



Geração de Código

Construção de Modelo

Construção de Modelos

- Ideal
  - Automática
  - Fácil e rápida
  - **Requisito essencial**: Modelo deve ser uma representação fiel do sistema
    - Modelo M é completo se inclui *todos* os comportamentos realizáveis do programa $Prog$
      $L(Prog) \subseteq L(M)$
    - Modelo M é correto se inclui *somente* comportamentos realizáveis do programa $Prog$
      $L(M) \subseteq L(Prog)$

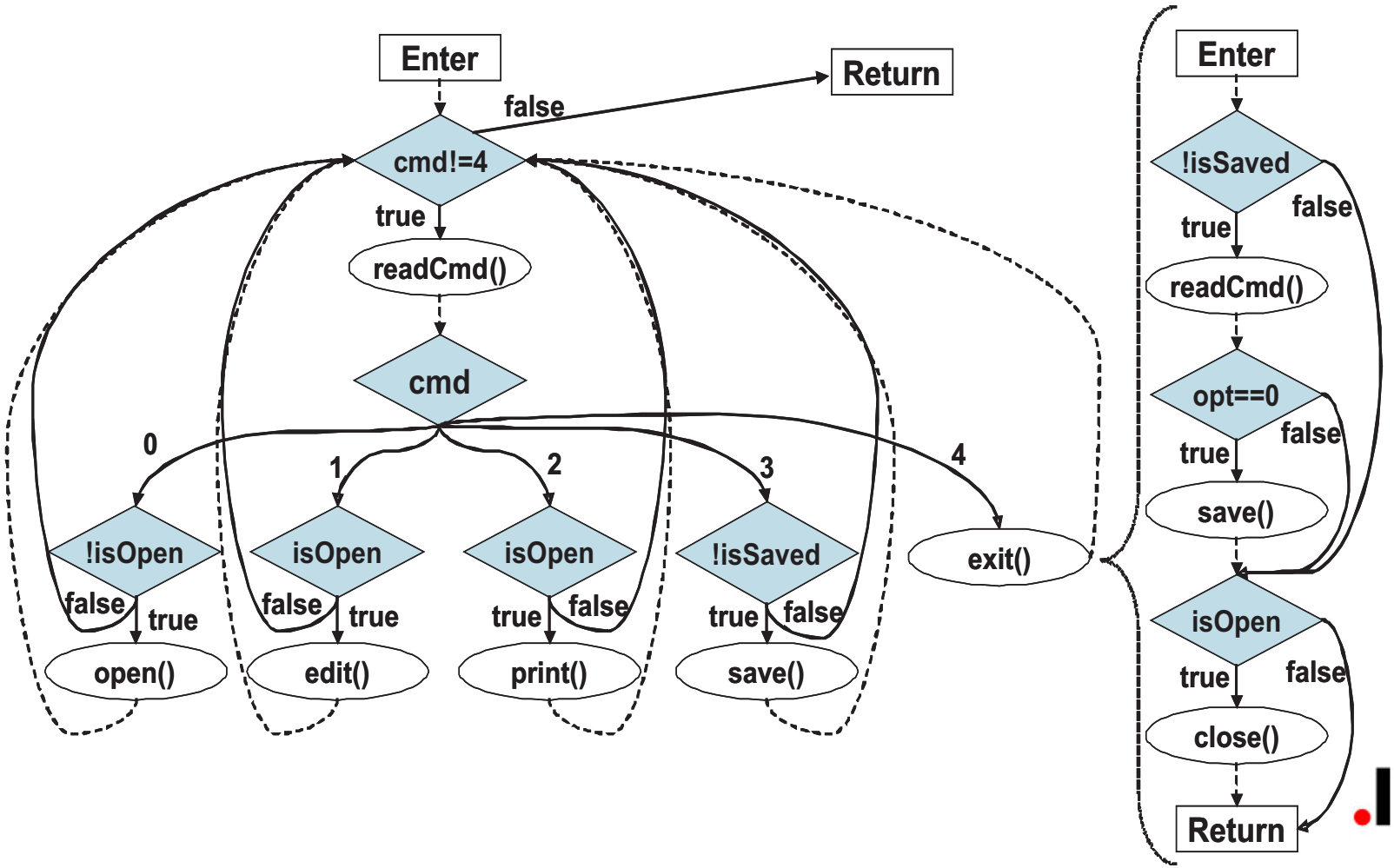
Extração de Modelos

- Processo de se obter um **modelo** (de comportamento) a partir de um **software existente** [Holzmann & Smith 1999]
- Possibilita a **geração automática do modelo**
- Modelo gerado descreve o **comportamento implementado pelo software**, dado um nível de abstração
- Esbarra no problema de construção de modelos [Corbett et al. 2000]

Exemplo

```
1  public class Editor {
2   private boolean isOpen;
3   private boolean isSaved;
4
5   public Editor () {
6    int cmd = -1;
7    isOpen = false;
8    isSaved = true;
9
10   while(cmd != 4){
11    cmd = readCmd();
12    switch (cmd) {
13     case 0: if(!isOpen)
14          name = open();
15        break;
16       case 1: if(isOpen)
17            edit(name);
18          break;
19     case 2: if(isOpen)
20          print(name);
21          break;
22     case 3: if(!isSaved)
23           save(name);
24          break;
25     case 4: exit(name);
26    }
27   }
28  }
29  ...
30  void exit (String n) {
31   if (!isSaved) {
32    int opt = readOpt();
33    if (opt == 0)
34     save(n);
35   }
36   if (isOpen) close(n);
37  }
38  ...
39 }
```

Abordagem Estática



Abordagem Estática (cont.)

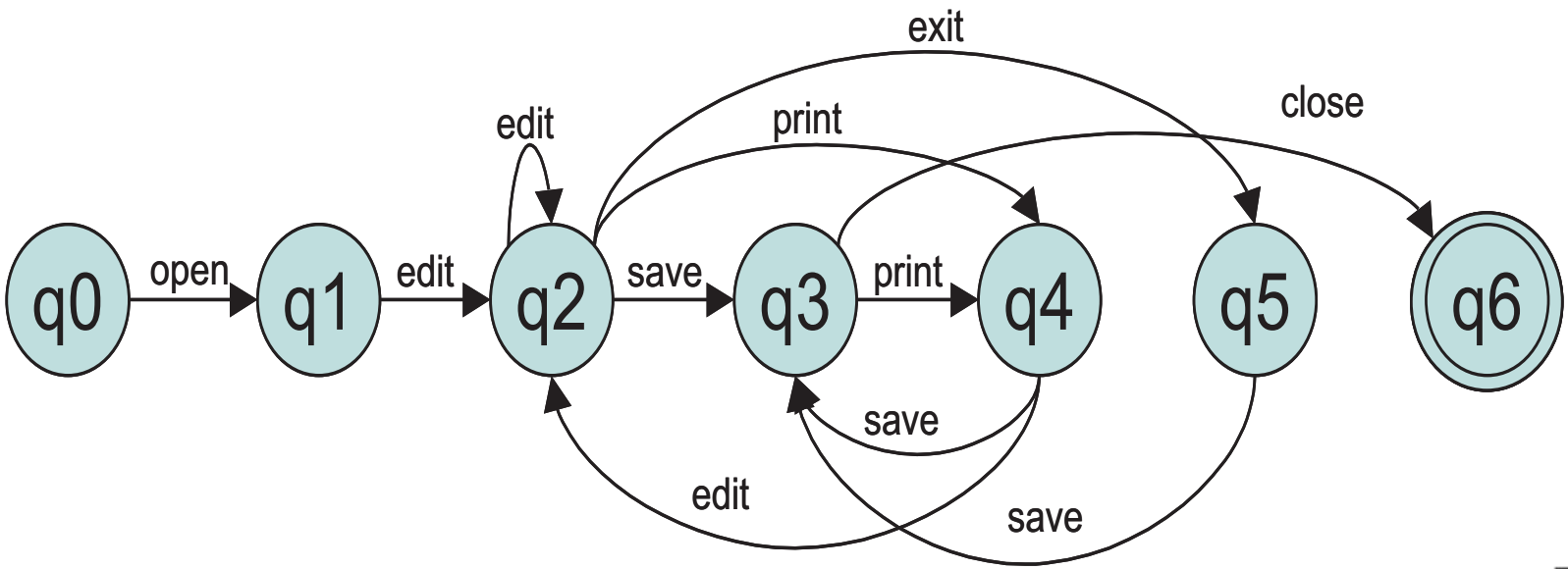
- **Comportamento completo** do sistema
- Alguns **comportamentos irrealizáveis** ⇒ falsos positivos
- Requer acesso ao código fonte
- Exemplos:
  - Bandera [Corbett et al. 2000]
  - Java PathFinder [Havelund et al. 2000]
  - SLAM [Ball & Rajamani 2002]
  - BLAST [Henzinger et al. 2002]

Abordagem Estática (cont.)

- Tópicos Relacionados:
  - Execução Simbólica [King 1976]
  - Slicing [Corbett et al. 2000]
  - Counterexample-Guided Abstraction Refinement (CEGAR) [Kroening et al. 2004]

Abordagem Dinâmica

Trace: open edit save print edit edit print save print edit exit save close



Abordagem Dinâmica (cont.)

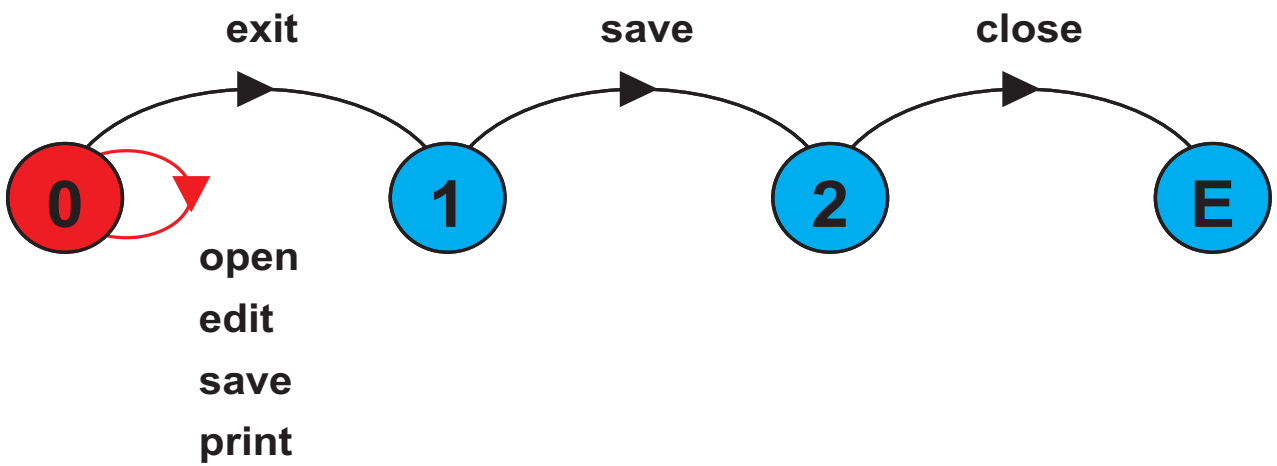
- Extração estilo "caixa preta"
- Contém somente **comportamentos reais**
- Requer **processo de inferência** $\Rightarrow$ possível inclusão de comportamentos inválidos
- **Não há garantia de modelo completo**
- Exemplos:
  - Grammar Inference [Cook & Wolf 1998]
  - GK-Tail [Lorenzolli et al. 2006]

Abordagem Dinâmica (cont.)

- Tópicos Relacionados:
  - Aprendizado de autômatos [Angluin 1987]
  - Análise Estatística [Cook & Wolf 1998]
  - Análise Parametrizada [Lorenzolli et al. 2006]

Abordagem Híbrida

Trace: open edit save print edit edit print save print edit exit save close



Abordagem Híbrida (cont.)

- Combina comportamentos reais com informações da estrutura do código
- Correção é garantida pelo apoio na informação estática
- **Completude depende dos comportamentos observados**
- Exemplos:
  - Daikon & ESC/Java [Nimmer & Ernst 2002]
  - LTSE [Duarte et al. 2006]

Abordagem Híbrida (cont.)

- Tópicos Relacionados:
  - Inferência dinâmica + análise estática [Nimmer & Ernst 2002]
  - Contextos [Duarte et al. 2006]

Abordagem de Modelo Implícito

- Não há geração de um modelo "manipulável"
- *Runtime monitoring*
- **Exploração do espaço de estados pode não ser completa**
- Exemplo:
  - Verisoft [Godefroid 2003]

Tópicos

- Abstração de construções de linguagens modernas (ADT, exceções, herança, aspectos, *inner classes*, etc.)
- Refinamento de modelos
- Análise de concorrência
- Modelos completos com respeito a algum critério (cobertura, propriedade, etc.)

Grupo de VV & T

- Grupo de pesquisa do INF/UFRGS dedicado à pesquisa em Validação, Verificação e Teste de Sistemas Computacionais
- Integrantes:
  - Profa. Dra. Leila Ribeiro
  - Profa. Dra. Érika Cota
  - Prof. Dr. Rodrigo Machado
  - Prof. Dr. Lucio Mauro Duarte
- Colaborações em projetos com pesquisadores da UFPel, UNIPAMPA e PUCRS
- Mais informações:

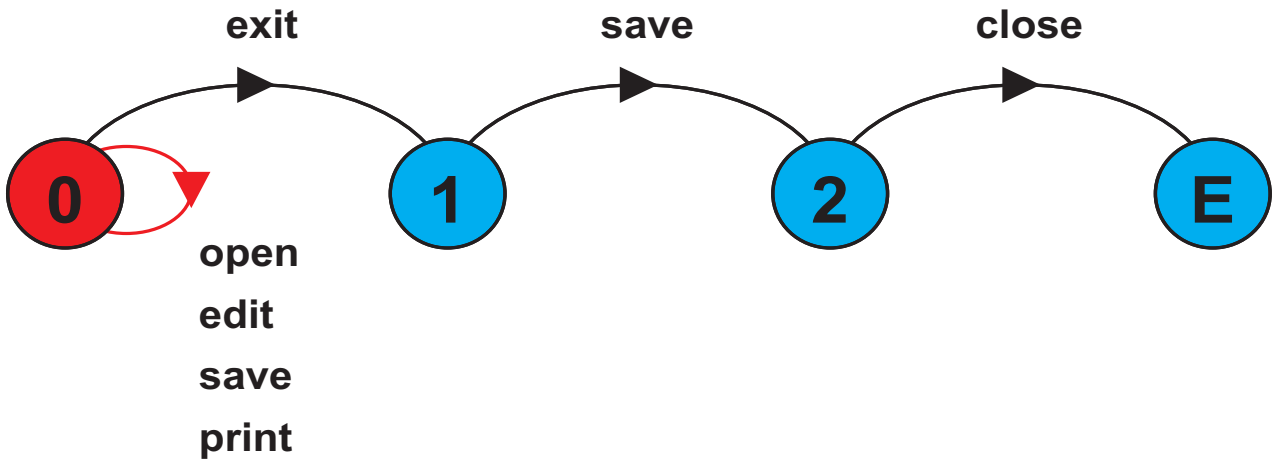
  https://wiki.inf.ufrgs.br/Verites (em construção)

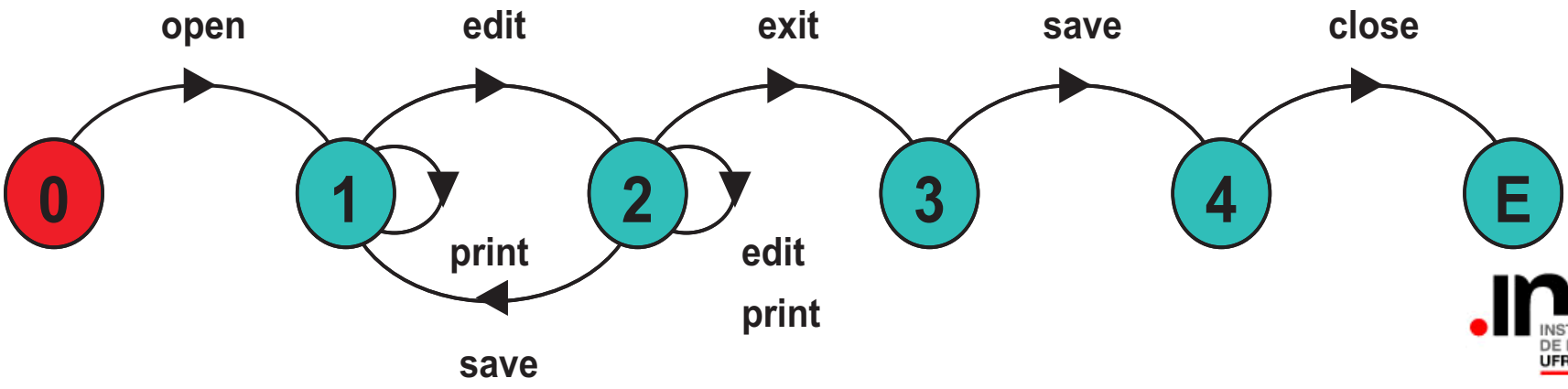
Extração de Modelos no Grupo de VV & T (cont.)

- Extração de modelos LTS baseados em contextos [Duarte et al. 2008]
  - **Contextos** combinam informações de controle de fluxo com informações sobre valores de atributos e cadeias de chamadas de métodos
  - Permite combinação segura de rastros de execução, inclusive com possibilidade de comportamentos extras
  - Independente de linguagem (regras de anotação)
  - Suporte a sistemas concorrentes
  - Completude depende dos rastros utilizados

Extração de Modelos no Grupo de VV & T (cont.)

Trace: open edit save print edit edit print save print edit exit save close

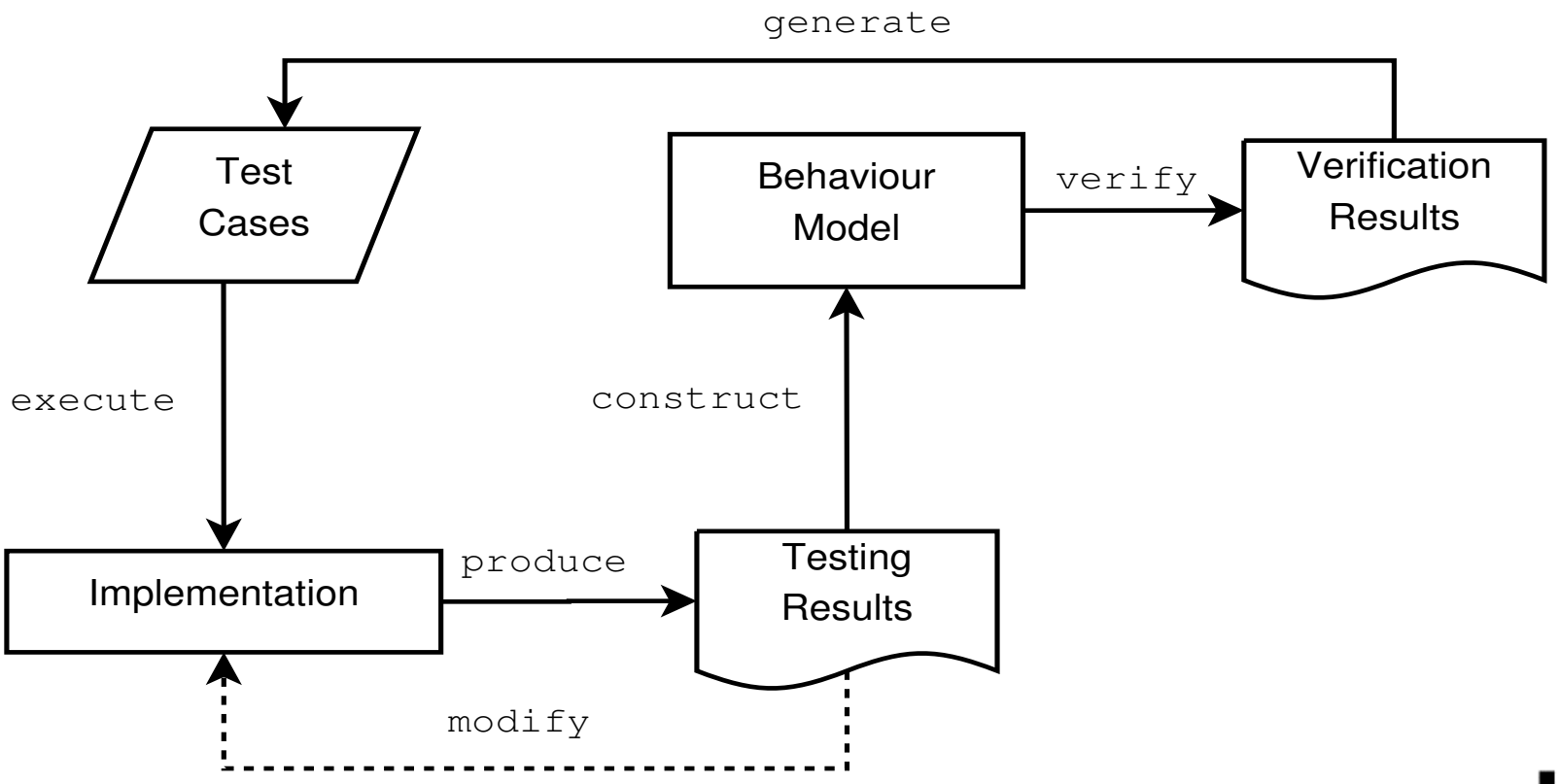




Extração de Modelos no Grupo de VV & T (cont.)

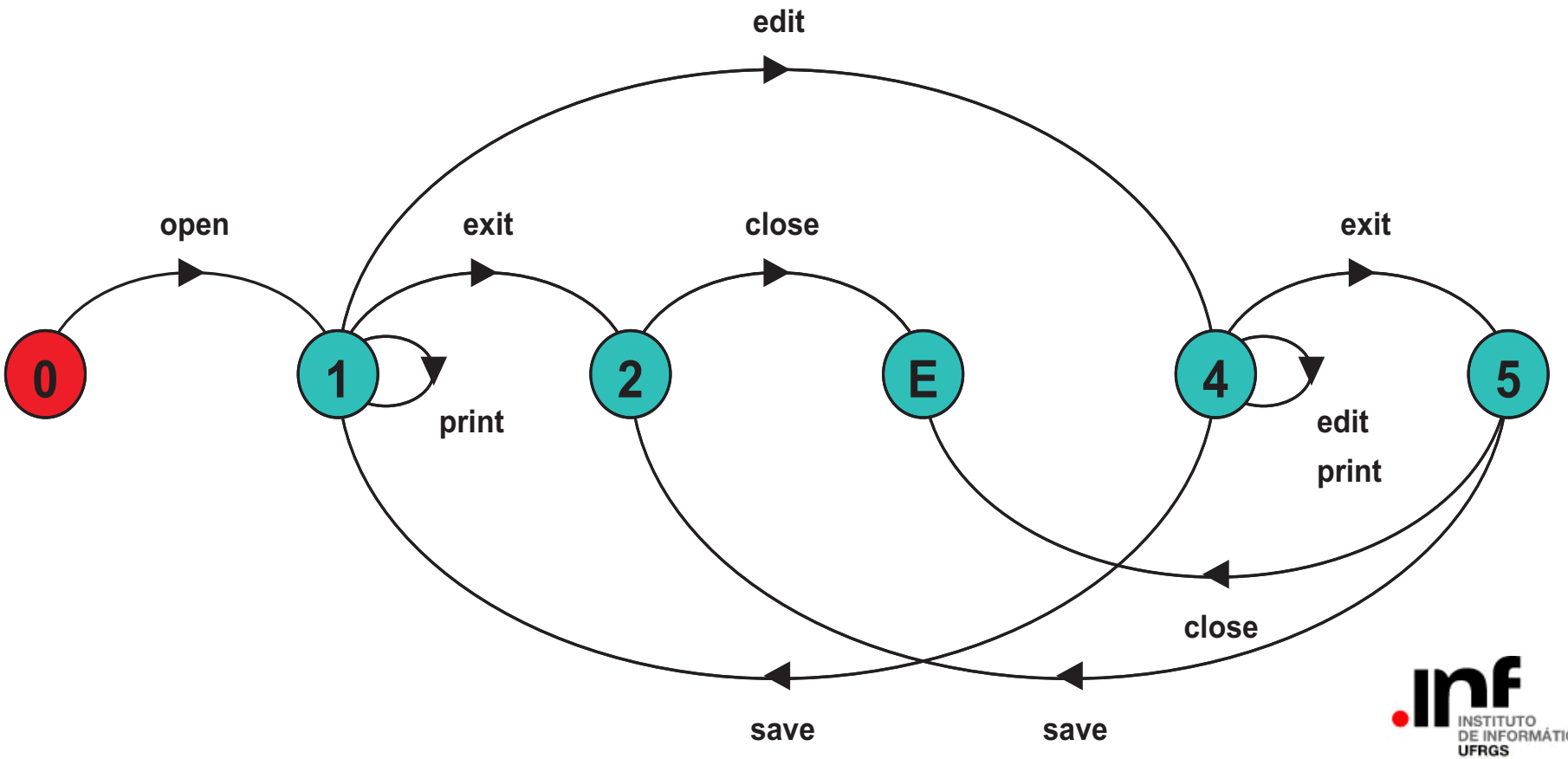
- Combinação de Verificação de Modelos e Teste de Software [Duarte 2011]
  - Abordagem incremental para garantir um modelo completo
  - Uso de casos de teste para guiar a extração do modelo (e.g., baseado em uma propriedade - *Property-Oriented Testing* - ou em um critério de cobertura)
  - Modelo criado usado para gerar novos casos de teste (*Model-Based Testing*)

Extração de Modelos no Grupo de VV & T (cont.)



Extração de Modelos no Grupo de VV & T (cont.)

Trace: open edit save print edit edit print save print edit exit save close



Extração de Modelos no Grupo de VV & T (cont.)

- Extração de modelos em Gramáticas de Grafos [Oliveira et al. 2013]
  - Adaptação da abordagem de extração de LTS
  - Mudança de paradigma, visto que GG é baseada em estados (dados) e não em ações

Extração de Modelos no Grupo de VV & T (cont.)

- Extração de modelos quantitativos
  - Possibilidade de extrair modelos que possam ser usados para análises quantitativas, tais como Redes de Autômatos Estocásticos

Extração de Modelos no Grupo de VV & T (cont.)

- Verificação de implementações de sistemas
  - Uso do modelo extraído para comparação com um modelo da especificação
  - Análise de equivalência para identificar possíveis discrepâncias

Referências

- **Angluin, D. "Learning regular sets from queries and counterexamples", Inf. Comput., Academic Press, Inc., n,75, 1987.**
- Ball, T. & Rajamani, S. K. "The SLAM Project: Debugging System Software via Static Analysis", POPL, 2002.
- Clarke, E. M.; Grumberg, O. & Peled, D. A. "Model Checking", The MIT Press, 1999.
- Cook, J. E. & Wolf, A. L. "Discovering Models of Software Processes from Event-Based Data", 1998.
- **Corbett, J. C. et al. "Bandera: extracting finite-state models from Java source code", ICSE, 2000.**
- Duarte, L. M.; Kramer, J. & Uchitel, S. "Model Extraction Using Context Information", MoDELS, 2006.
- Duarte, L.; Kramer, J. & Uchitel, S. "Towards Faithful Model Extraction Based on Contexts", FASE, 2008.
- Duarte, L. "Integrating Software Testing and Model Checking Through Model Extraction", SBMF : short papers, 2011.
- Godefroid, P. "Software Model Checking: The Verisoft Approach", Bell Laboratories, Lucent Technologies, 2003.

Referências (cont.)

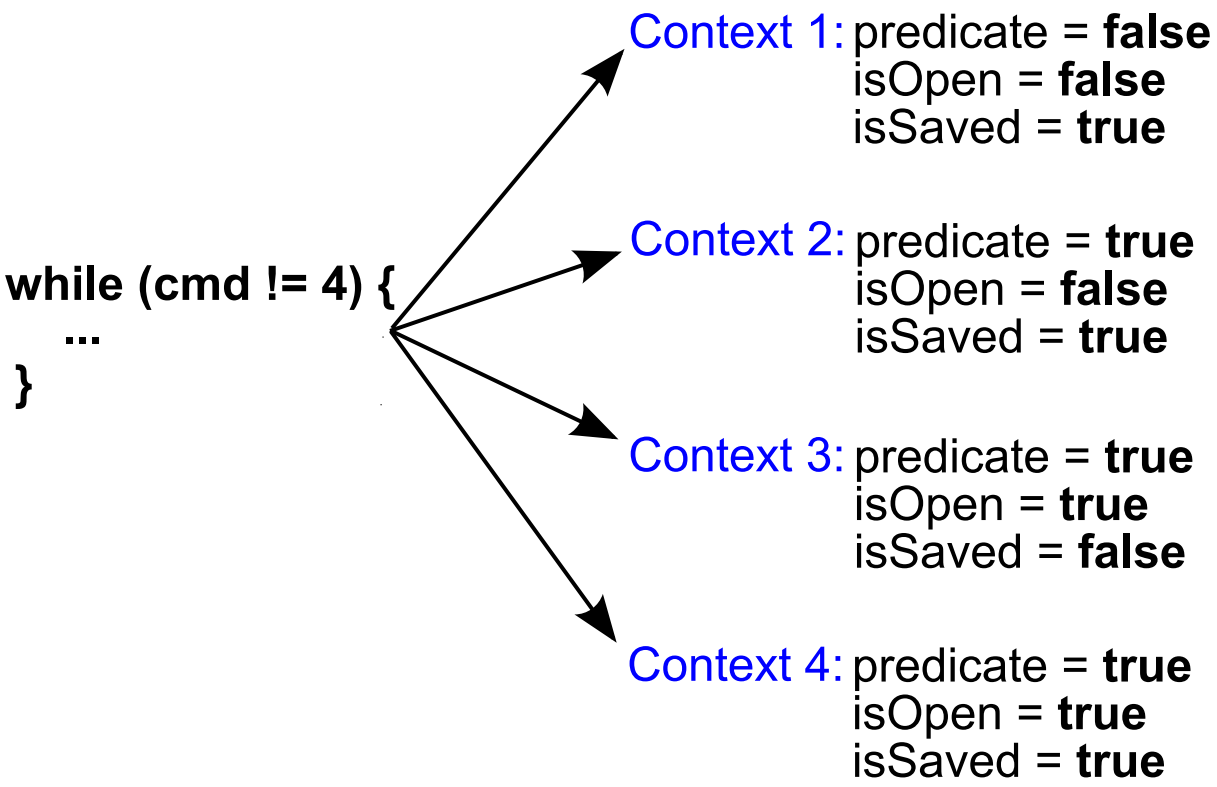
- Havelund, K. & Pressburger, T. "Model checking JAVA programs using Java PathFinder", Springer-Verlag, n.2, 2000.
- Henzinger, T.; Jahla, R.; Majumdar, R. & Sutre, G. "Lazy Abstraction", POPL, 2002.
- **Holzmann, G. & Smith, M. "A Practical Method for Verifying Event-Driven Software", ICSE, 1999.**
- King, J. "Symbolic Execution and Program Testing", Communications of the ACM, n. 19, 1976.
- Kroening, D. et al. "Counterexample Guided Abstraction Refinement via Program Execution", LNCS 3308, 2004.
- Lorenzoli, D.; Mariani, L. & Pezzè, M. "Inferring State-Based Behavior Models", WODA, 2006.
- Nimmer, J. & Ernst, M. Automatic Generation of Program Specifications", ISTTA, 2002.
- Sommerville, I. "Software Engineering", Addison-Wesley, 2004.

Extração de Modelos de Comportamento de Software

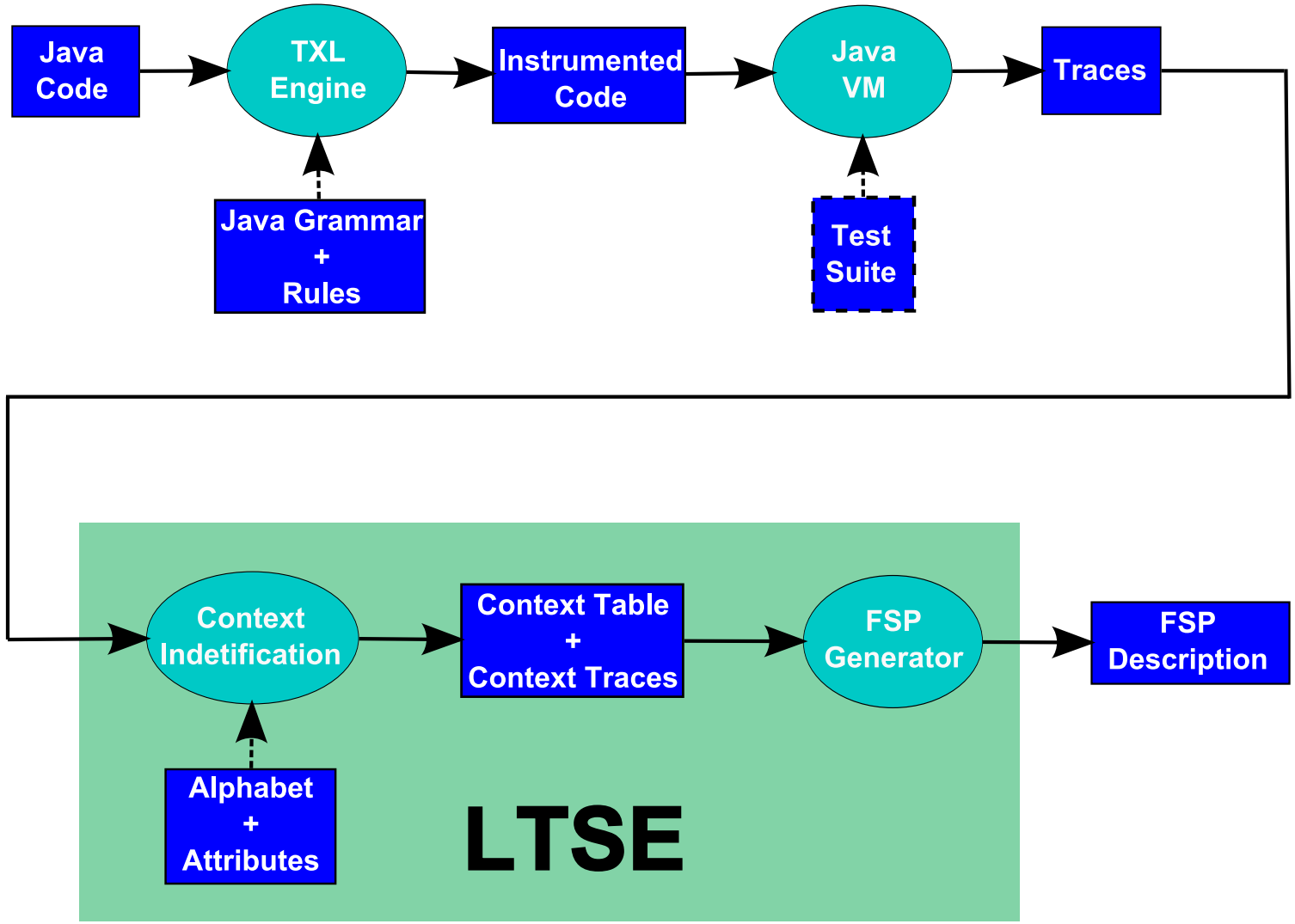
Lucio Mauro Duarte
lmduarte@inf.ufrgs.br
Grupo de VV & T
Instituto de Informática
Universidade Federal do Rio Grande do Sul

Workshop-Escola de Informática Teórica 2013
15/10/2013

Contexto


while (cmd != 4) {
...
}
Context 1: predicate = false
isOpen = false
isSaved = true
Context 2: predicate = true
isOpen = false
isSaved = true
Context 3: predicate = true
isOpen = true
isSaved = false
Context 4: predicate = true
isOpen = true
isSaved = true

Visão Geral


Java Code
TXL Engine
Instrumented Code
Java VM
Traces
Java Grammar + Rules
Test Suite
Context Indetification
Context Table + Context Traces
FSP Generator
FSP Description
Alphabet + Attributes
LTSE

Identificação de Contextos

Trace: [open edit save print edit edit print save print edit exit save close]

```
REP_ENTER:(cmd!=4)#true#{false,true}
 SEL_ENTER:(cmd)#0#{false,true}
  SEL_ENTER:(!isOpen)#true#{false,true}
   MET_ENTER:open#true#{false,true}
   MET_END
  SEL_END
 SEL_END
REP_END
REP_ENTER:(cmd!=4)#true#{true,true}
 SEL_ENTER:(cmd)#1#{true,true}
  SEL_ENTER:(isOpen)#true#{true,true}
   MET_ENTER:edit#true#{true,true}
   MET_END
  SEL_END
 SEL_END
REP_END
REP_ENTER:(cmd!=4)#true#{true,false}
...
```

Contexto	Pred.	Val.	Estado
0	-	T	{}
0.1	(cmd!=4)	T	{F,T}
0.1.1	(cmd)	0	{F,T}
0.1.1.1	(!isOpen)	T	{F,T}
0.1.1.1.1	open	T	{F,T}
0.2	(cmd!=4)	T	{T,T}
0.2.1	(cmd)	1	{T,T}
0.2.1.1	(isOpen)	T	{T,T}
0.2.1.1.1	edit	T	{T,T}
0.3	(cmd!=4)	T	{T,F}
...	....	...	{...}

Geração de FSP

Rastro: [open edit save print edit edit print save print edit exit save close]