Alloy

Rohit Gheyi

Tiago Massoni

Alloy

- Lógica
  - Lógica de 1ª ordem
  - Cálculo Relacional
- Linguagem
  - Sintaxe
- Análises
  - Exaustiva até um escopo, usando SAT Solvers

Visão Geral

module exemplo

assinaturas (conjuntos e relações)

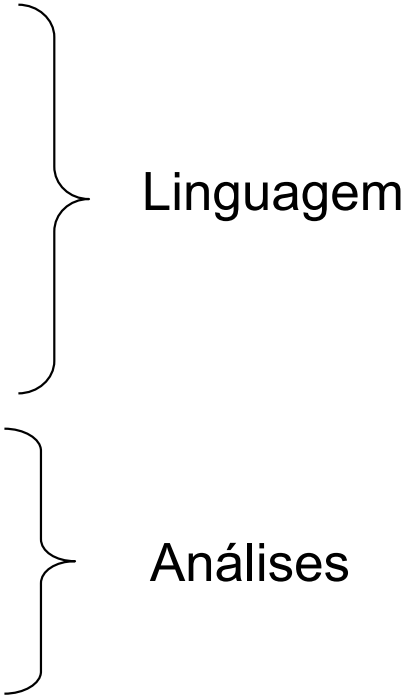
fatos (invariantes)

predicados e funções

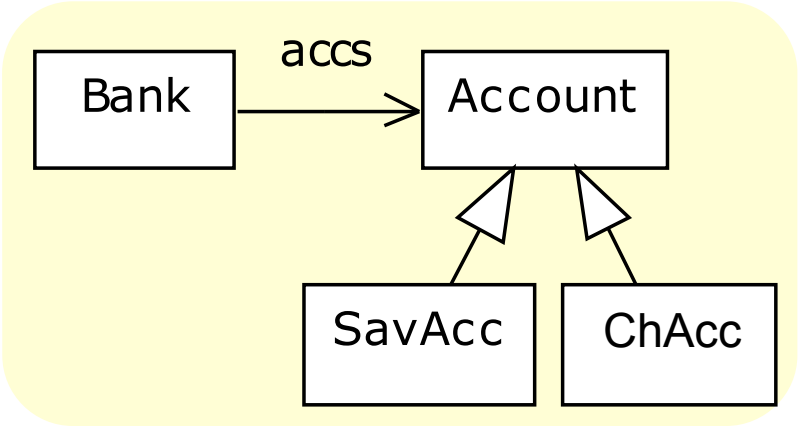
asserções

run

check



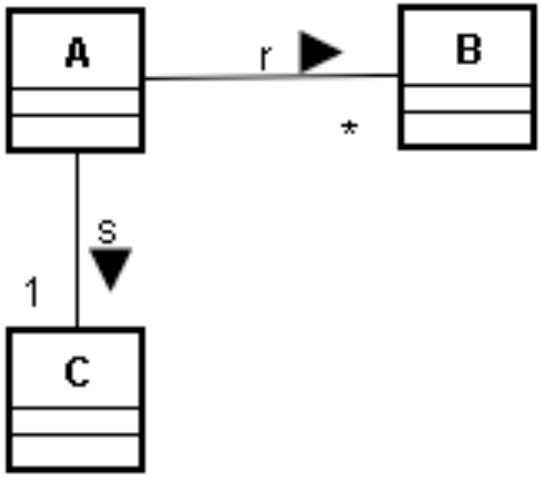
Exemplo: Assinaturas e Relações




```
module banco
sig Banco {
  contas: set Conta
}
sig Conta {}
sig ContaCorr extends Conta {}
sig ContaPoup extends Conta {}
```

Assinaturas e Relações

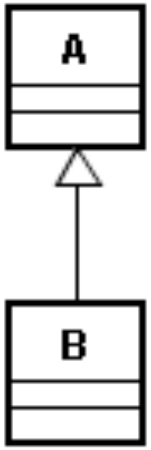
- assinatura = conjunto de objetos
- relação relaciona elementos entre conjuntos (globais)




```
sig A {
  r: set B,
  s: one C
}
sig B{}
one sig C{}
```

Herança

- B herda as relações e fórmulas sobre a A
- Em Alloy, A tem acesso as relações de B

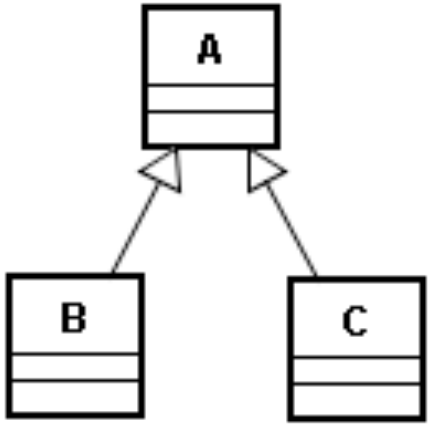



```
sig A { … }
sig B extends A {
  …
}
```

$B \subseteq A$

Subset Signature

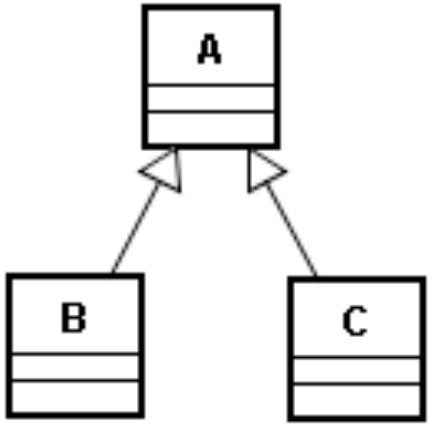
- A diferença é que as subassinaturas não são disjuntas
  - Fato implícito BIC = 0




```
sig A { … }
sig B,C in A {}
```

Assinatura Abstrata

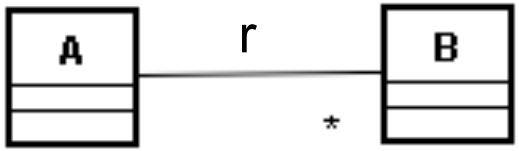
- Todas as instâncias da assinatura A pertencem a assinatura B ou C



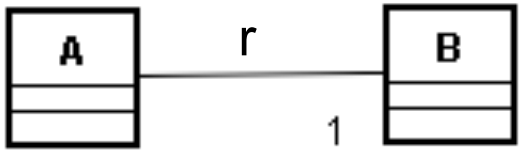
abstract sig A { … }
sig B,C **extends** A {}

Multiplicidades

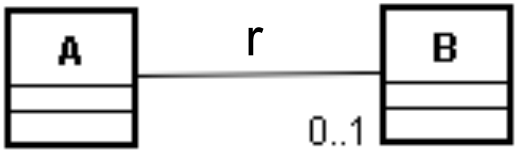
```
sig A {
  r: set B
}
sig B {}
```




```
sig A {
  r: one B
}
sig B {}
```




```
sig A {
  r: lone B
}
sig B {}
```



Relações tem aridade variada

```
sig Agenda {
  endereco: Nome -> Endereco
}
sig Nome, Endereco {}
```

Cardinalidade

r: A -> **one** B (função)

s: A **one** -> B (relação injetiva)

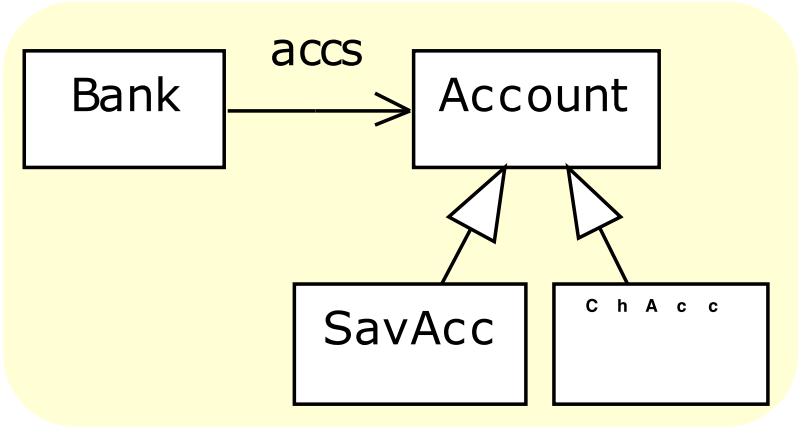
Fatos

- Declara um conjunto de invariantes do modelo

```
fact nome {
    f1
    f2 ...
}
```

Conjunção
$f_1 \wedge f_2 \wedge ...$

Exemplo: Fatos




```
module banco
sig Banco {
  contas: set Conta
}
sig Conta {}
sig ContaCorr extends Conta {}
sig ContaPoup extends Conta {}
fact AccPartition {
  Conta = ContaCorr + ContaPoup
}
```

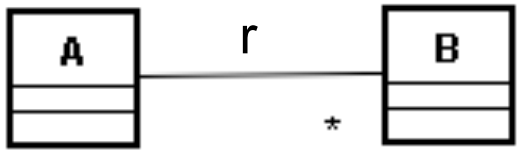
Toda conta ou é poupança ou é corrente

Cardinalidade

- **#**
  - #exp = 10
  - exp tem exatamente 10 elementos
- **one**
  - **one** exp
  - exp resulta em um elemento
- **some**
  - **some** exp
  - exp possui algum elemento
- **no**
  - **no** exp
  - exp não possui elemento

Exercício 1

- Especifique os seguintes fatos
  - a relação r não possui relacionamentos.
  - O conjunto A possui 10 elementos
  - O conjunto B possui algum element




```
sig A {
  r: set B
}
sig B {}
fact card {
  no r
  #A =10
  some B
}
```

Operadores de Conjuntos

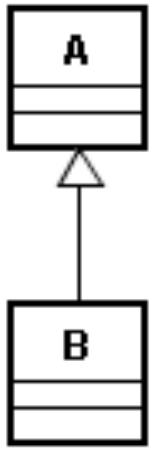
- União (U)
  - A+B
- Interseção (I)
  - A&B
- Diferença (/)
  - A-B
- Subconjunto (⊆)
  - A **in** B
- Negação (!)
  - ! (A **in** B)

Operadores Lógicos

- Conjunção ($\wedge$)
  - P **and** Q
  - P && Q
- Disjunção (v)
  - P **or** Q
  - P || Q
- Implicação ($\Rightarrow$)
  - P => Q
- Biimplicação (Û)
  - P <=> Q

Exercício

- Qual o significado das fórmulas a seguir?




```
sig A { … }
sig B extends A {}
fact fatos {
    A+B = A
    A&B = A
    A-B = A
    A != B
}
```

Quantificação

- Universal (∀)
  - **all** x:A | p(x)
  - Para todos os x do tipo A, tal que p(x) é verdade
- Existencial (∃)
  - **some** x:A | p(x)
  - Existe um x do tipo A, tal que p(x) é verdade

Exemplo

```
module banco
sig Banco {
  contas: set Conta
}
sig Conta {}
fact Exemplo {
  all b: Banco | some b.contas
  some b:Banco | some b.contas
}
```

Fatos Anexados à assinatura

```
sig Host {}
sig Link{ from,dest: Host}
fact {
    all l:Link | l.from != l.dest
    }
```

é o mesmo de...

```
sig Host {}
sig Link{ from,dest: Host}{ from != dest }
```

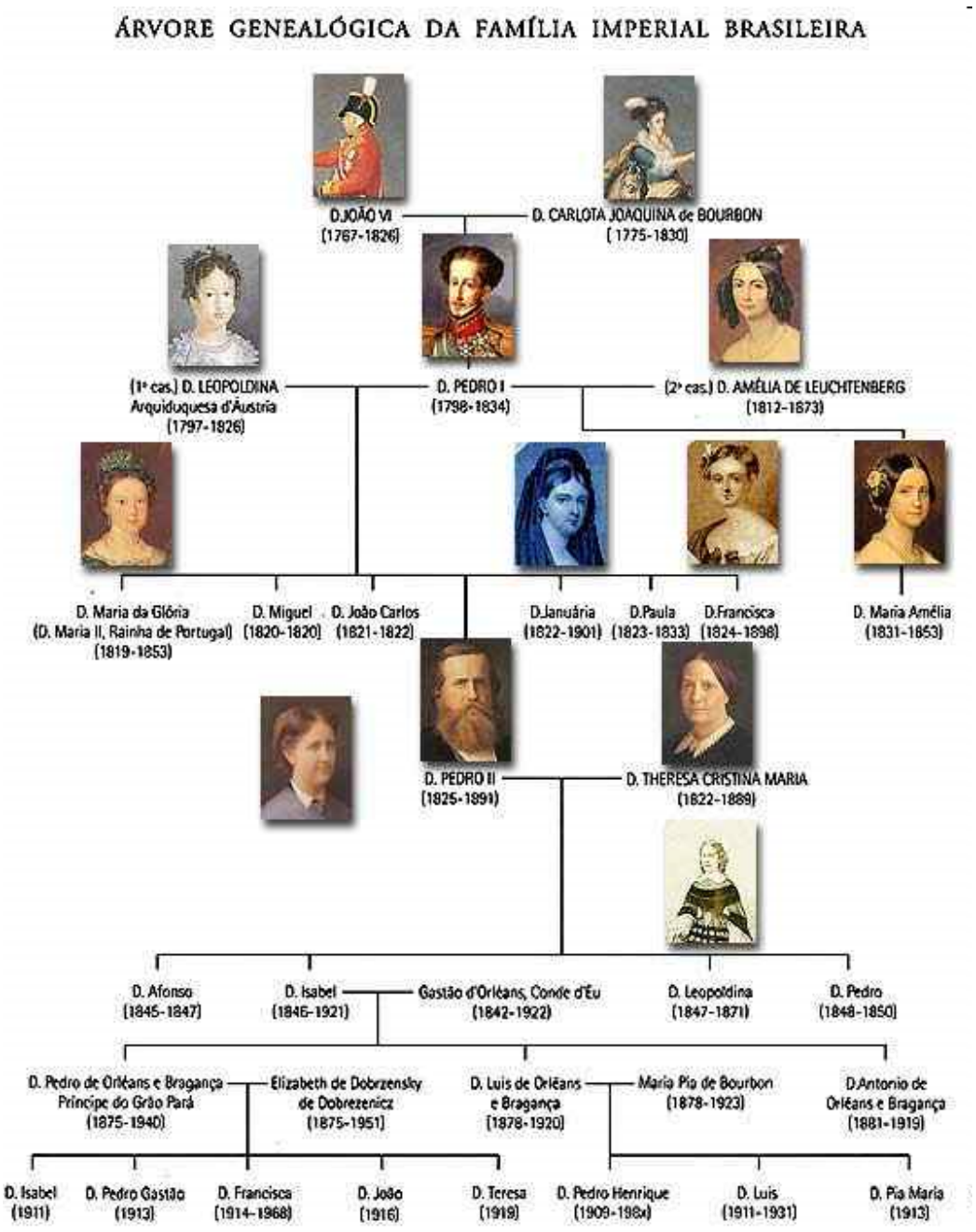
Operadores

- Join (.)
  - Composição relacional
  - (a,b).{(b,c), (d,e), (b,f)} = {(a,c), (a,f)}
- Transpose (~)
  - Reversa
  - Se r = {(b,c), (d,e)}, ~r = {(c,b), (e,d)}
- Transitive Closure (* e ^)
  - Aplicação de r uma ou mais vezes
  - Relações binárias de tipos relacionados
  - *r e ^r

Exemplo

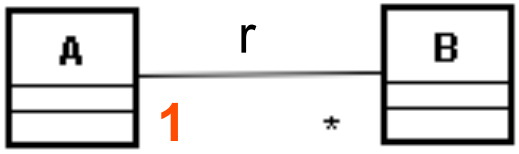
```
sig Pessoa {
  pais: set Pessoa
}
```

Como expressar todos os ancestrais?



Exercício 1

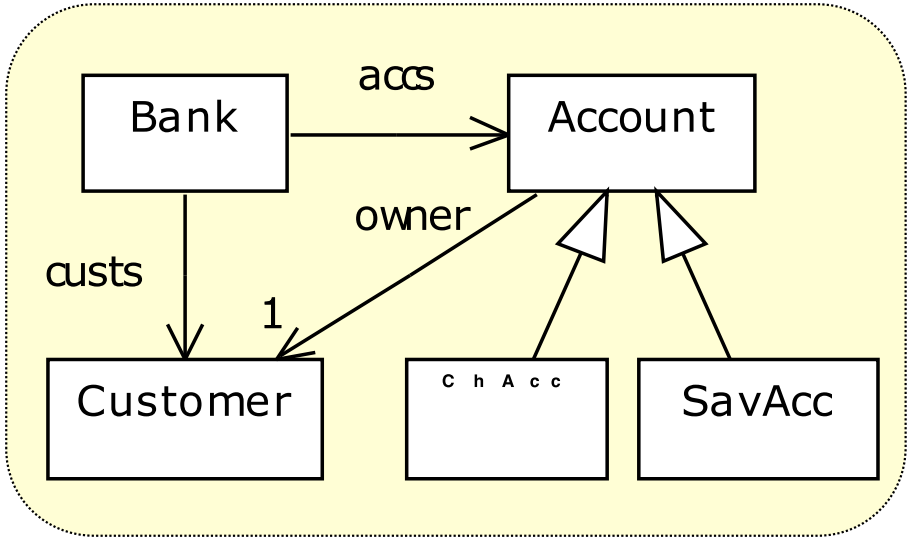
- Considere o modelo a seguir. Especifique a multiplicidade a seguir usando ~.




```
sig A {
  r: set B
}
sig B {}
fact mult {
  all b:B | one b.~r
}
```

Exercício 2

- Especifique em Alloy que os clientes do banco são os mesmos dos donos das contas do banco.



Exercício 3

- Especifique em Alloy parte de um metamodelo em Java para expressar:
  - Object é uma classe
  - Uma classe possui uma classe pai
  - Uma classe não pode estender ela mesma
    - Diretamente
    - Indiretamente

Solução

```
sig Class {
  extend: lone Class
}
one sig Object extends Class {}
fact {
  all c:Class | c !in c.extend
  all c:Class | c !in c.^extend
}
```

Predicados

- Declara um conjunto de fórmulas, sendo aplicado em outras fórmulas
- e ajuda a especificar operações...

```
pred temContas[b:Banco] {
    some b.contas
}
fact {
    all b: Banco | temContas[b]
}
```

Funções

- Declara um valor relacional, usado em outras fórmulas

```
fun contasDoBanco[b:Banco]: set Conta {
    b.contas
}
fact {
    all b: Banco |
        #contasDoBanco[b] > 1
}
```

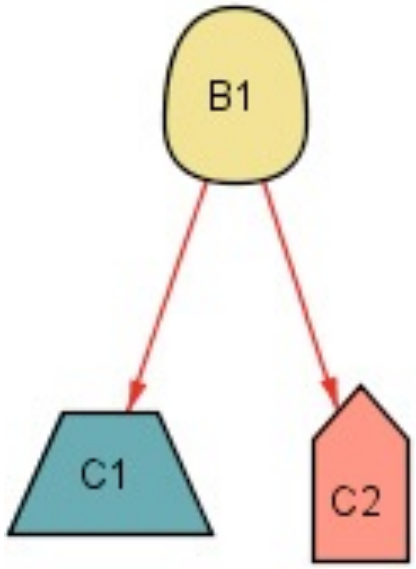
Análises

- Run
  - Encontra uma instância válida para o predicado ou função
  - Satisfaz aos invariantes do modelo e as fórmulas do predicado ou função
- Check
  - Checa em um dado escopo se as fórmulas de um asserção são válidas a partir da especificação (todos os casos)

Exemplo run

- Especifique que existem duas contas C1 e C2, e um banco B1 que só as contém
- O escopo 2 indica no máximo 2 elementos para todas as assinaturas

```
module banco
sig Banco {
  contas: set Conta
}
sig Conta {}
one sig B1 extends Banco {}
one sig C1,C2 extends Conta {}
fact { C1+C2 = B1.contas }
pred show[]{}
run show for 2
```



Exemplo check

- Checa uma propriedade no escopo
- Todos os bancos possuem uma conta?

```
module banco
sig Banco {
  contas: set Conta
}
sig Conta {}
assert todoBancoTemConta {
   all b:Banco | some b.contas
}
check todoBancoTemConta for 2
```

inteiros em Alloy

- número inteiro dentro de um objeto

  Int

```
sig Node { adj: Node -> lone Int }

fact {
    all n: Node |
    let w= n.(n.adj) |
        some w => int[w] = 0
}
```

cast de Int (objeto) para int (valor)

escopo (número de bits de representação de inteiros)

```
run show for 3 Int
```

boolean em Alloy

```
sig Fone {
    foraGancho, tocando: Boolean
    }
```

não existe!
evitar...

melhor classificar

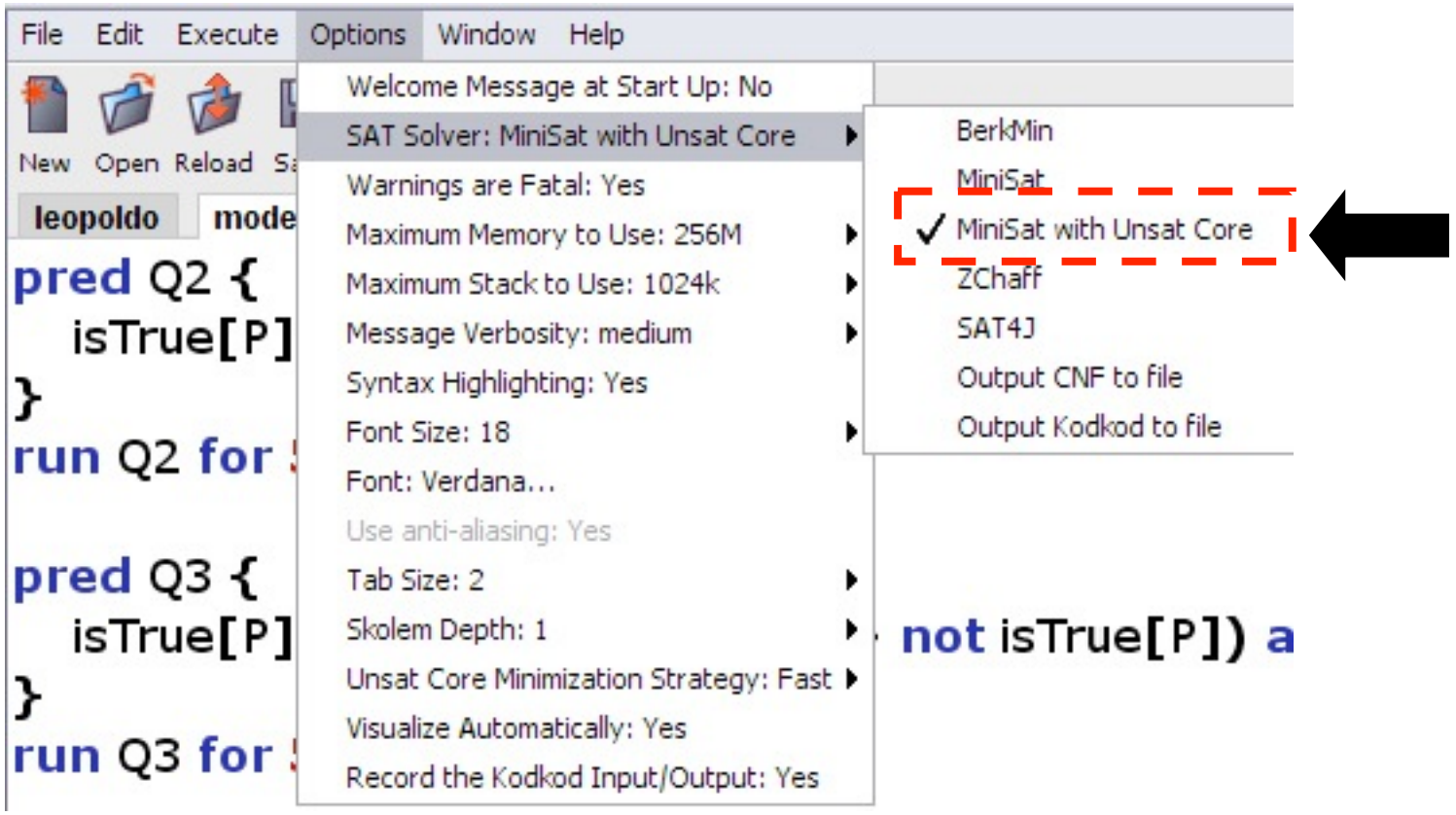
```
sig Fone { }
sig foraGancho, tocando in Fone {}
fact {
    no foraGancho & tocando
}
```

Problema

Se uma especificação estiver inconsistente, o que fazer?

Core Unsat

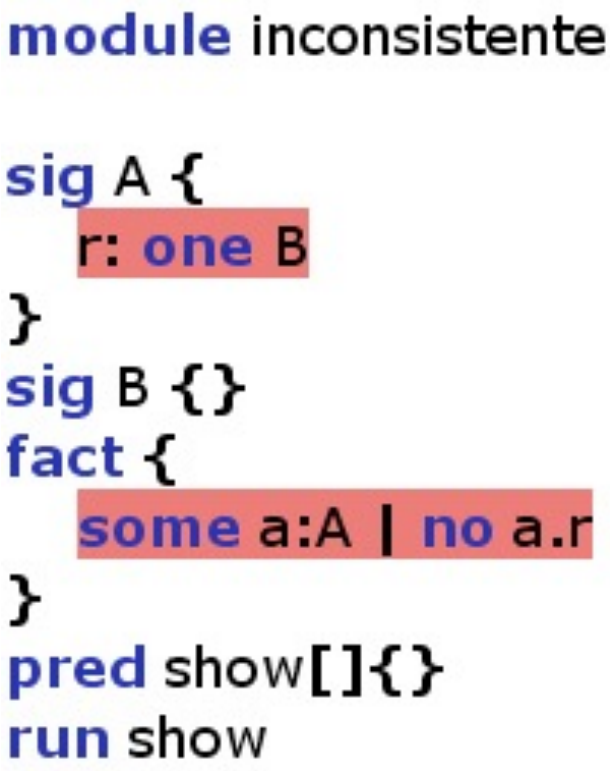
- Tem alguns solvers que ajudam



Exemplo

No instance found. Predicate may be inconsistent. 47ms.
Core contains 2 top-level formulas. 16ms.

```
module inconsistente
sig A {
  r: one B
}
sig B {}
fact {
  some a:A | no a.r
}
pred show[] {}
run show
```



Instalação

- Home Page
  - http://alloy.mit.edu
- Requer JRE
- Rodar
  - java –jar alloy.jar
- API
  - http://alloy.mit.edu/alloy4/api.html

Comportamento e Padrões

Sistema Bancário

- Comportamento
- Operações
  - Adicionar Conta
  - Remover Conta

Estrutura: Padrão Local State

```
open util/ordering[Time] as to
sig Time {}
one sig Banco {
    contas: Conta->Time
}
sig Conta {}
```

Ordenar elementos de uma assinatura

Padrão Local State (relação ternária)

Operação: Adicionar Conta

```
pred addConta[b:Banco, c:Conta,t,t':Time] {
    c !in (b.contas).t
    (b.contas).t' = (b.contas).t + c
}
```

Operação: Remover Conta

```
pred removeConta[b:Banco, c:Conta, t,t':Time] {
    c in (b.contas).t
    (b.contas).t' = (b.contas).t - c
}
```

Estado inicial do sistema

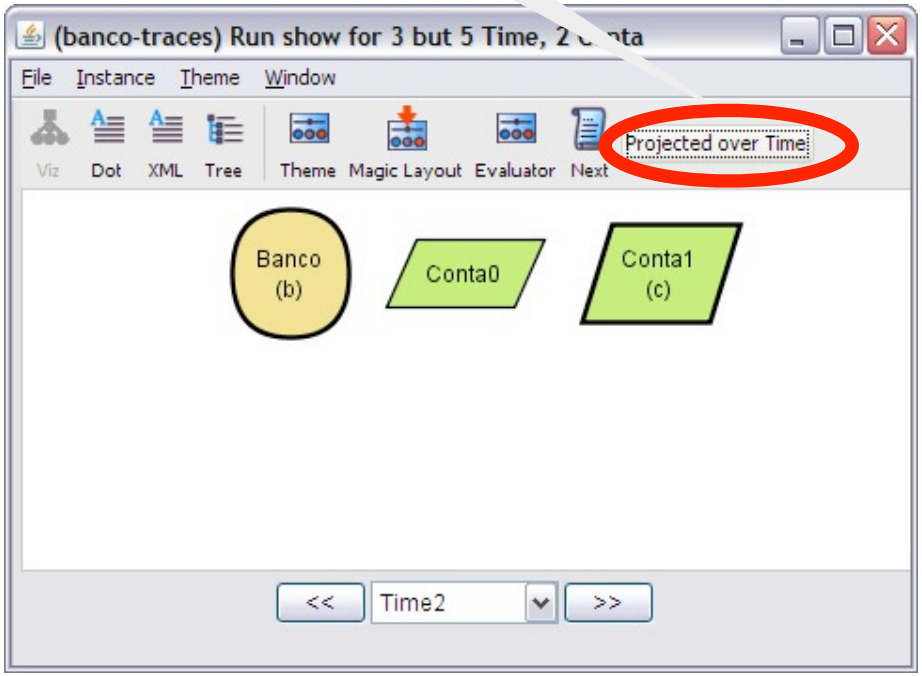
```
pred init [t: Time] {
    no (Banco.contas).t
}
```

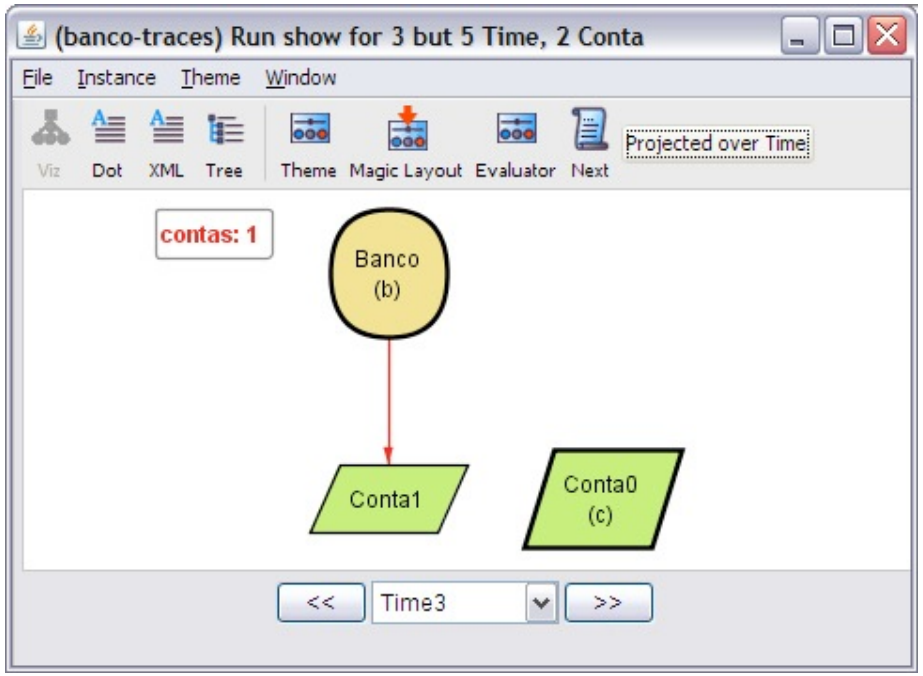
Padrão Traces

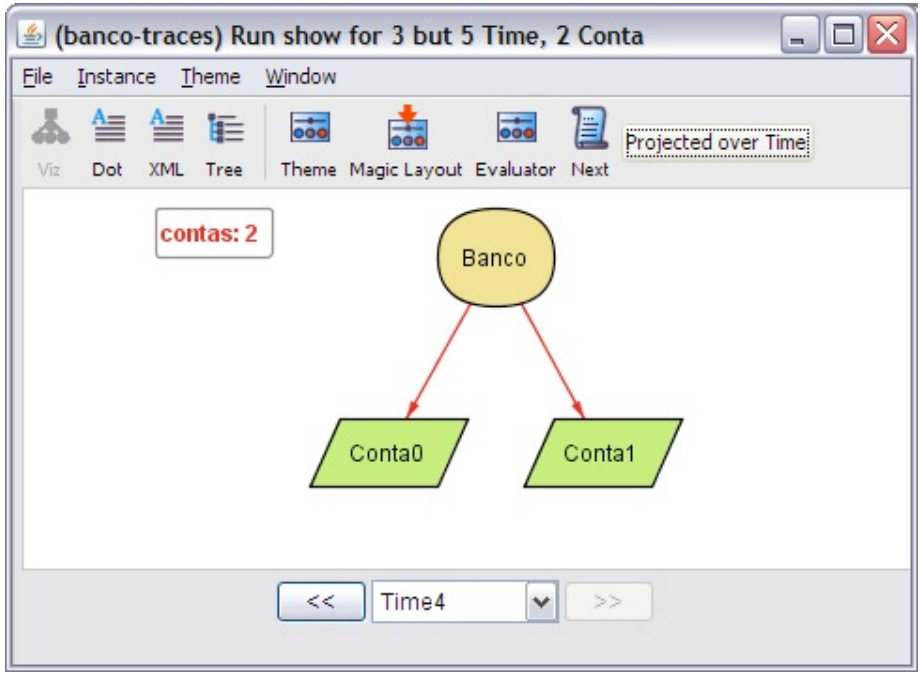
```
fact traces {
    init [first]
    all pre: Time-last | let pos = pre.next |
          some b: Banco, c: Conta |
                addConta[b,c,pre,pos] or
                removeConta[b,c,pre,pos]
}
```

Análises

Projetar sobre o Time







Uma instância com projeção:

comportamento

Outra opção

- Assinatura que representa o sistema

```
open util/ordering[Banco] as bo
one sig Banco {
    contas: set Conta
    clientes: set Cliente
}
sig Conta {}
sig Cliente {}
```

Outra opção

```
pred addConta[b,b':Banco, c:Conta] {
    c !in b.contas
    b'.contas = b.contas + c
}
```

Outra opção

```
fact traces {
    init [first]
    all pre: Banco-last | let pos = pre.next |
          some c: Conta |
                addConta[pre,pos,c] or
                removeConta[pre,pos,c]
}
```

Bibliografia

- Daniel Jackson. Software abstractions: logic, language and analysis. MIT Press, 2006.
  - http://softwareabstractions.org/
  - Alloy 3 para Alloy 4
- Alloy Analyzer
  - http://alloy.mit.edu
- Tutorial
  - http://alloy.mit.edu/fm06/

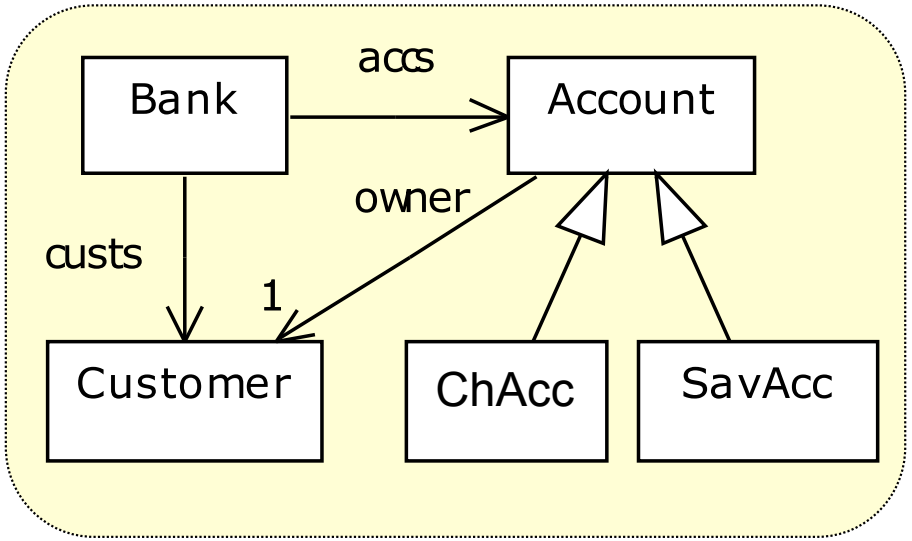
Escreva estes fatos em Alloy

os clientes do banco são os mesmos dos donos das contas do banco

Se um banco tem contas, ele tem clientes

Nunca o banco terá mais clientes do que contas

Todo cliente do banco tem que ter pelo menos uma poupança



Escreva em Alloy usando assinaturas, fatos, predicados e asserções

Maria admira todos os professores.
Alguns professores admiram Maria,
Nenhum aluno assistiu a todas as aulas.

Queremos testar:
Maria admira ela mesma.
Nenhuma aula foi assistida por todos os alunos.