Introdução à Lógica Temporal das Ações

Aula para disciplina de Métodos Formais

Gabriela Moreira

Departamento de Ciência da Computação - DCC
Universidade do Estado de Santa Catarina - UDESC

02 de abril de 2025

Conteúdo

Outline

Introdução

A Lógica Temporal das Ações (*Temporal Logic of Actions* - TLA) foi proposta em (LAMPORT, 1994).

Seu principal conceito são **ações**. Uma ação é uma expressão booleana composta de variáveis, variáveis *primed* e constantes.

Nessa aula, vamos focar em como usar ações, com o operador *primed* (’), para definir estruturas de Kripke. Mais adiante, em outra aula, falaremos sobre outros conceitos.

Operador *primed*

f' (f *primed*) para uma função de estado f é o valor de f no **final de um passo**.

Em outras palavras, para um passo composto por uma **dupla de estados** (s, t), f' é o valor de f para t.

De forma semelhante, P' para um predicado P é o valor de P para o estado final de um passo. Assim, na avaliação da valoração de uma ação para um passo, predicados e variáveis **sem** o operador *primed* se referem aos seus respectivos valores no **primeiro estado** do passo, e sempre que **forem marcados** com o operador, fazem referência aos valores no **segundo** estado do passo.

Exemplos: operador *primed*

1. $x' = x + 1$
2. $x + 1 = x'$
3. $x = x' - 1$

Exemplos: operador *primed*

1. $x' = x + 1$
2. $x + 1 = x'$
3. $x = x' - 1$

Outros exemplos:

- $x' > x$
- $x'/ = x + 1$

Exemplos: operador *primed*

1. $x' = x + 1$
2. $x + 1 = x'$
3. $x = x' - 1$

Outros exemplos:

- $x' > x$
- $x'/ = x + 1$

PS: É necessário muito cuidado ao usar formas diferentes de (1).

Ações definem Transições

Combinando operações *primed* e predicados comuns, podemos definir as transições do modelo.

- Uma transição é permitida no modelo definido por uma ação sse a avaliação da ação pro passo da transição é verdadeiro.
  - Exemplo:
    - Ação $x' = x + 1$
    - Transição $1 \rightarrow 2$
    - Substituindo x por 1 e x' por 2, a igualdade é satisfeita

Ações definem Transições

Combinando operações *primed* e predicados comuns, podemos definir as transições do modelo.

- Uma transição é permitida no modelo definido por uma ação sse a avaliação da ação pro passo da transição é verdadeiro.
  - Exemplo:
    - Ação $x' = x + 1$
    - Transição $1 \rightarrow 2$
    - Substituindo x por 1 e x' por 2, a igualdade é satisfeita

Note que **1** ação define **N** transições. Assim, uma única ação pode definir todas as transições de um modelo!

Uma fórmula temporal define o modelo!

1. Um predicado comum define o(s) estado(s) inicial(is).
   - Geralmente, chamamos de *Init* ou `init`.
   - Exemplo: $Init \triangleq x = 0$

Uma fórmula temporal define o modelo!

1. Um predicado comum define o(s) estado(s) inicial(is).
   - Geralmente, chamamos de *Init* ou `init`.
   - Exemplo: $Init \triangleq x = 0$
1. Uma ação define as transições.
   - Geralmente, chamamos de *Next* ou `step`.
   - Exemplo: $Next \triangleq x' = x + 1$

Uma fórmula temporal define o modelo!

1. Um predicado comum define o(s) estado(s) inicial(is).
   - Geralmente, chamamos de *Init* ou `init`.
   - Exemplo: $Init \triangleq x = 0$
1. Uma ação define as transições.
   - Geralmente, chamamos de *Next* ou `step`.
   - Exemplo: $Next \triangleq x' = x + 1$
1. Uma fórmula temporal define o modelo.
   - $Spec \triangleq Init \wedge \Box Next$

Outline

TLA+ e Quint

- TLA+ (*Temporal Logic of Actions+*) combina a lógica de TLA com teoria de conjuntos, definindo uma linguagem de especificação formal.

TLA+ e Quint

- TLA+ (*Temporal Logic of Actions+*) combina a lógica de TLA com teoria de conjuntos, definindo uma linguagem de especificação formal.
- Quint é uma syntaxe alternativa à TLA+, que pode ser transpilada para TLA+, e portanto pode usar as mesmas ferramentas.
  - Quint tem alguns recursos adicionais que não existem em TLA+, com definições de testes e execuções (*runs*).
  - Quint (ainda) não suporta algumas coisas de TLA+ que serão vistas no final da disciplina, como refinamento.
  - No geral, Quint é **intencionalmente** mais restrito que TLA+.

TLC e Apalache

Temos dois *model checkers* disponíveis para TLA+ e Quint.

- TLC
  - Primeiro *model checker* para TLA+. Faz enumeração explícita de estados.
  - Funciona apenas com TLA+, então especificações em Quint precisam ser transpiladas
  - Open source, mantido pela Microsoft Research

TLC e Apalache

Temos dois *model checkers* disponíveis para TLA+ e Quint.

- TLC
  - Primeiro *model checker* para TLA+. Faz enumeração explícita de estados.
  - Funciona apenas com TLA+, então especificações em Quint precisam ser transpiladas
  - Open source, mantido pela Microsoft Research
- Apalache
  - *Model checker* limitado simbólico. Traduz a especificação para restrições SMT e resolve-as usando o *solver* Z3.
  - Exige anotações de tipo para variáveis e constantes
  - Suporte nativo a TLA+ e Quint
  - Open source, desenvolvido majoritariamente pela Informal Systems

Model checking TLA+

1. Apalache
   - Lembre-se de anotar os tipos
   - Use a linha de comando

Model checking TLA+

1. Apalache
   - Lembre-se de anotar os tipos
   - Use a linha de comando
1. TLC
   - Crie um arquivo `.cfg`
   - Use a linha de comando ou a extensão para VSCode

Model checking TLA+

1. Apalache
   - Lembre-se de anotar os tipos
   - Use a linha de comando
1. TLC
   - Crie um arquivo `.cfg`
   - Use a linha de comando ou a extensão para VSCode
1. Toolbox (TLC)
   - A IDE cria as configurações para você, porém você fica dependente da IDE para rodar o *model checker.*

Model checking Quint

1. Linha de comando: `quint verify spec.qnt`
   - Isso vai baixar e usar o Apalache por baixo
   - Suporte a fórmulas temporais é limitado

Model checking Quint

1. Linha de comando: `quint verify spec.qnt`
   - Isso vai baixar e usar o Apalache por baixo
   - Suporte a fórmulas temporais é limitado
1. TLC:
   - Ainda não há suporte completo, mas há um script pra ajudar: `tlc/check_with_tlc.sh`.
   - O script tem várias dependências, não recomendo usar ainda nesse estado.

Instalando as ferramentas - Dependências

- NodeJS >= 18
- Java Development Kit >= 17

Quint

1. Com acesso de administrador
   - Instalar: `npm i @informalsystems/quint -g`
   - Executar: `quint --help`
2. Sem acesso de administrador
   - Instalar: `npm i @informalsystems/quint --user`
   - Executar: `npx quint --help`
   - Essa instalação é local, então você vai precisar instalar de novo se trocar de pasta

Quint

1. Com acesso de administrador
   - Instalar: `npm i @informalsystems/quint -g`
   - Executar: `quint --help`
2. Sem acesso de administrador
   - Instalar: `npm i @informalsystems/quint --user`
   - Executar: `npx quint --help`
   - Essa instalação é local, então você vai precisar instalar de novo se trocar de pasta

Também temos binários para download direto na release do GitHub

- Isso é novo e não foi muito testado ainda, mas deve ser tranquilo usar

TLC

- TLC para linha de comando
  - Opcional, se quiser usar pela linha de comando
- Extensão no VSCode (TLA+ Nightly)
  - Após instalar, abra um arquivo `.tla`, aperte F1 e procure o comando "TLA+: Check model with TLC"

Apalache

- Apalache para linha de comando
- É usado internamente pelo Quint quando invocamos `quint verify`. Só precisa baixar separadamente se quiser utilizar com TLA+.
  - Se não funcionar (comando ficar "travado" e não termina nunca), abra outra terminal e execute:

```
~/.quint/apalache-dist-0.47.2/apalache/bin/apalache-mc server
```

Outline

Semáforos

Vamos ver novamente o exemplo dos semáforos e verificar propriedades em Quint e TLA+

- Arquivo no moodle/site da disciplina

Testando as ferramentas

Atividade EaD valendo presença na aula do dia 24/03
Descrição no moodle: Usando as ferramentas: TLA+ e Quint

Referências

LAMPORT, L. The temporal logic of actions. **ACM trans. program. lang. syst.**, v. 16, n. 3, p. 872–923, 1994.

Introdução à Lógica Temporal das Ações

Aula para disciplina de Métodos Formais

Gabriela Moreira

Departamento de Ciência da Computação - DCC
Universidade do Estado de Santa Catarina - UDESC

02 de abril de 2025