02. Ações Computacionais

Princípios de Engenharia de Software (Texto em Elaboração), v0.1.1

Italo S. Vega
italo@pucsp.br

Faculdade de Estudos Interdisciplinares (FACEI)

Pontifícia Universidade Católica de São Paulo

Sumário

2 Ações Computacionais

 — **Programas** de computador implementam modelos de **ações computacionais**.

2.1 Variável e Valor

 — Sempre tive dúvidas a respeito do conceito de variável.

O que se considera como variável depende do raciocínio que desejamos realizar. A escolha das variáveis é arbitrária.

 — É comum ouvirmos algo como: "sejam x e y duas variáveis na situação...".

A partir deste ponto, elas representam valores nas situações que seguem. Em Python, quando escrevemos um comando com a forma:

```
# CENÁRIO de declaração de variável e atribuição de valor
a = 5
```

1. introduzimos o identificador de variável a e
2. fazemos a variável referenciar o valor 5.

O conceito de variável neste material alinha-se com o apresentado por Sedgewick & Wayne (2009), p. 16:

> **Variável**: "um nome que se refere a um valor".

Naquele caso, podemos considerar que iniciou-se um raciocínio a partir da igualdade $a = 5$: a variável de nome a referencia o valor 5. Em uma representação inspirada no trabalho de Lamport (2002), escrevemos:

$$\text{início} \triangleq \wedge\, a = 5$$

para formalizar a introdução de um raciocínio que se inicia considerando ser verdade que $a = 5$.

 — E como represento valores em Python?

Valores são representados por **literais** na linguagem Python.

 — Em Python, o valor 5 pode ser representado pelo literal `5` do tipo `Integer`.

 — Entendo isso... mais ou menos.

O símbolo '`=`' não representa a ideia de comparação entre valores na linguagem Python. Trata-se de um ponto a ser observado nessa tecnologia, normalmente justificado pelo uso frequente quando programamos. Em um código, é bem mais comum a ocorrência de atribuições, ao invés da **relação de igualdade**, representada pelo símbolo '`==`'. Assim, para saber se o valor 5 foi atribuído à variável a, escrevemos:

```
# CENÁRIO de declaração do nome 'a' e atribuição de valor à variável
a = 5
# início
assert (a == 5)
```

2.1.1 Questão: igualdades e atribuições

Contexto Considere o fragmento em Python para implementar a expressão lógica $(3 = 3 \wedge 3 = 3)$:

```
# CENÁRIO com uma expressão inválida
(3 == 3 and 3 = 3)
```

Enunciado Assinale a alternativa que complete a seguinte sentença, tornando-a verdadeira:

A parte inválida da expressão é

1. `3 == 3` pois não se pode atribuir um valor a uma constante.
2. `3 = 3` pois não existe uma variável no lado esquerdo da atribuição.
3. `3 = 3` pois o operador `and` exige que os operandos sejam do tipo `boolean`.
4. `and` mas deveria ser `or`.

 — A separação que você está fazendo entre uma lógica e a sua implementação em Python ajuda a programar melhor?

 — Certamente! Vamos desenvolver alguma familiaridade com o uso de especificações lógicas em casos mais simples.

2.2 Estado de uma Máquina

 — Os livros de Sedgewick & Wayne (2009) e Eck (2021) contém vários trechos de código que posso utilizar nas nossas conversas.

Em particular, Sedgewick & Wayne (2020) implementam uma **lógica** para trocar os valores entre as variáveis a e b. Vamos considerar a seguinte atribuição de valores: $[a = 5, b = 3]$. Uma particular atribuição de valores em variáveis é chamado de **estado**.

Estado: uma atribuição de valores em variáveis.

2.2.1 Questão: estado de um quebra-cabeça

Contexto A quantidade de navios destruídos em um quebra-cabeça Batalha Naval[1] foi representada pelo valor da variável destruídos. Inicialmente, é verdade que destruídos $= 0$. Sabe-se que o quebra-cabeça contém cinco navios a serem destruídos.

Enunciado Assinale a alternativa contendo um estado **inválido** do quebra-cabeça:

1. [destruídos $= 1$].
2. [destruídos $= 3$].
3. [destruídos $= 5$].
4. [destruídos $= 7$].

Em Python, o estado $[a = 5, b = 3]$ pode ser estabelecido pela execução dos seguintes **comandos** de atribuição:

```
# CENÁRIO de declaração de variáveis e atribuição de valores
a = 5
b = 3
# início
assert (a == 5 and b == 3)
```

 — Sob o ponto de vista lógico, representamos essa afirmação por um **predicado de estado**. Por ser o primeiro da nossa lógica, vamos chamá-lo de "início":

$$\begin{aligned} \text{início} \triangleq & \wedge a = 5 \\ & \wedge b = 3 \end{aligned}$$

[1] https://www.conceptispuzzles.com/index.aspx?uri=puzzle/battleships/history

 — Quer dizer que, após declarar as variáveis a e b e atribuir os respectivos valores 5 e 3, posso dizer que o predicado início é verdadeiro?

 — Exato! Se o computador não executar comandos subsequentes, o estado do computador não se modificará: $[a = 5, b = 3]$.

2.2.2 Questão: estado inicial de um quebra-cabeça

Contexto A quantidade de navios destruídos em um quebra-cabeça Batalha Naval[2] foi representada pelo valor referenciado por "destruídos". Também deve ser contabilizada a quantidade de tentativas realizadas pelo jogador para destruir os cinco navios. Escolheu-se a variável "tentativas" para capturar essa ideia, cujo valor deve ser incrementado a cada disparo de torpedo, embora o incremento de "destruídos" somente ocorra se o jogador indicar uma posição do campo de batalha contendo um navio.

Considere o seguinte predicado de estado inicial, dependente das variáveis $x_1, x_2 \in \{0, 1, 2, 3, 4, 5\}$:

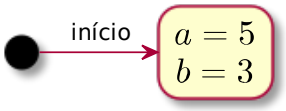
$$\begin{aligned} \text{início} \triangleq & \wedge \text{destruídos} = x_1 \\ & \wedge \text{tentativas} = x_2 \end{aligned}$$

Enunciado Assinale a alternativa contendo um par ordenado (x_1, x_2) que torne **verdadeiro** o predicado início:

1. $(x_1, x_2) = (0, 0)$.
2. $(x_1, x_2) = (5, 5)$.
3. $(x_1, x_2) = (0, 5)$.
4. $(x_1, x_2) = (5, 0)$.

 — Posso visualizar o estado de uma máquina usando um diagrama de máquinas de estado (Booch, Jacobson, & Rumbaugh, 1999)!

O estado $[a = 5, b = 3]$ também pode ser representado pelo seguinte diagrama:



Tratam-se de duas **formas** de representações distintas do mesmo estado. A primeira forma é chamada de **simbólica** enquanto a segunda, **diagramática**.

[2] https://www.conceptispuzzles.com/index.aspx?uri=puzzle/battleships/history

2.3 Atribuição de Valores e Ações

Em relação ao estado $[a = 5, b = 3]$, Fubã ainda está investigando a lógica para alcançar o estado $[a = 3, b = 5]$.

 — Para trocar os valores entre as variáveis a e b, precisaremos tornar verdade o estado $[a = 3, b = 5]$.

Esse novo estado resulta da **transformação** sofrida pelo primeiro, capturada pela ação "troca":

$$\begin{aligned} \text{troca} \triangleq \wedge\, a' &= b \\ \wedge\, b' &= a \end{aligned}$$

Aparece algo novo na ação troca. A presença do apóstrofe "′" seguido ao nome da variável indica um raciocínio que considera o estado posterior à ação. Ou seja, entende-se a' como o valor referenciado por a no estado seguinte da ação (Sedgewick & Wayne, 2009, p. 16). Logo, $a' = b$ significa que o valor de a no estado seguinte será igual ao de b no estado corrente. Isso que dizer que uma regra contendo variáveis marcadas com "′" especifica uma ação computacional.

2.3.1 Questão: contagem de disparos

Contexto A quantidade de tentativas realizadas pelo jogador para destruir os cinco navios de um quebra-cabeça Batalha Naval[3] foi representada pela variável "tentativas". O seu valor deve ser incrementado a cada disparo de torpedo, conforme a ação:

$$\text{disparo}_{\text{Efetuado}} \triangleq \wedge\, \text{tentativas}' = \text{tentativas} + 1$$

Enunciado Assinale a alternativa contendo uma afirmação verdadeira:

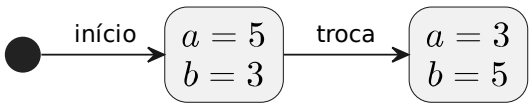
1. Se a ação $\text{disparo}_{\text{Efetuado}}$ for verdadeira então incrementou-se o valor de tentativas.
2. A cada disparo com destruição de navio incrementa-se o valor da variável tentativas.
3. O jogador resolve o quebra-cabeça quando tentativas $= 5$.
4. tentativas $= 0$.

---

[3] https://www.conceptispuzzles.com/index.aspx?uri=puzzle/battleships/history

No caso da troca de valores entre duas variáveis a e b, deve-se especificar uma ação que transforme o estado $[a = 3, b = 5]$ no estado $[a = 5, b = 3]$, por exemplo.

 — A ação troca é verdadeira se o estado $[a = 3, b = 5]$ vier imediatamente depois do estado $[a = 5, b = 3]$? Algo como:



Fubã está correto. Vale um destaque do conceito de ação:

> **Ação**: especificada por uma regra contendo variáveis com e sem o operador de **estado seguinte**, "′".

2.3.2 Questão: ações e variáveis

Contexto Considere as seguintes partes de ações:

I) $\wedge \text{tentativas}' = \text{tentativas} + 1$

II) $\wedge \text{destruídos}' = \text{destruídos} + 1$

III) $\wedge(x, y) = (3, 4)$

Sabe-se que destruídos $\in \{0, 1, 2, 3, 4, 5\}$.

Enunciado Assinale a alternativa contendo uma afirmação **verdadeira**:

1. tentativas′ refere-se a um valor no estado seguinte.
2. $(x, y) = (3, 4)$ não pode ser parte de uma ação.
3. Quando $x = 3$ uma ação foi realizada.
4. destruídos′ $\in \{0, 1, 2, 3, 4, 5\}$.

2.4 Passo de Ação Computacional

 — Associado ao conceito de ação, também é importante destacar o conceito de passo de ação.

Partindo-se de um estado qualquer e considerando-se uma ação verdadeira, calcula-se um novo estado, dito estado seguinte ou de chegada. Define-se o par formado pelo estado de partida e pelo estado de chegada como um passo de ação computacional. No exemplo da troca de valores entre as variáveis a e b, partindo-se do estado $[a = 5, b = 3]$, chega-se ao estado $[a = 3, b = 5]$ admitindo-se que a ação "troca" seja verdadeira:

$$\begin{aligned} \text{troca} \triangleq & \wedge a' = b \\ & \wedge b' = a \end{aligned}$$

 — Então o par de estados $([a = 5, b = 3], [a = 3, b = 5])$ satisfaz à ação e, por isso, é conhecido por passo de ação.

Fubã está correto novamente:

> **Passo de ação**: um par de estados que **satisfaz** uma ação.

2.4.1 Questão: passo da ação de disparo efetuado

Contexto A quantidade de tentativas realizadas pelo jogador para destruir os cinco navios de um quebra-cabeça Batalha Naval[4] (ideia representada pela variável "*tentativas*") deve ser incrementada a cada disparo de torpedo, conforme a ação:

$$\text{disparo}_{\text{Efetuado}} \triangleq \wedge \text{tentativas}' = \text{tentativas} + 1$$

Sabe-se que o passo $[\text{tentativas} = 2, \text{destruídos} = 3] \rightarrow [\text{tentativas} = x_1, \text{destruídos} = x_2]$ satisfaz a ação "$\text{disparo}_{\text{Efetuado}}$". Em Python:

```
# CENÀRIO de um passo computacional
# Estado de partida
[tentativas, destruidos] = [2, 3]
print ([tentativas, destruidos])
# Torno verdadeira a AÇÂO disparoEfetuado
tentativas += 1
# Estado de chegada
print ([tentativas, destruidos])
```

Enunciado Assinale a alternativa contendo valores x_1 e x_2 válidos:

1. $x_1 = 3$ e $x_2 = 3$.

[4] https://www.conceptispuzzles.com/index.aspx?uri=puzzle/battleships/history

2. $x_1 = 3$ e $x_2 = 4$.
3. $x_1 = 4$ e $x_2 = 3$.
4. $x_1 = 4$ e $x_2 = 4$.

2.5 Lógica e Regras

 — Mas como projeto um código que implemente a ação "troca"?

A execução do seguinte trecho conduz a uma afirmação falsa. O código não implementa a especificação lógica da ação. Ao ser executado, produz um absurdo:

```
# CENÁRIO de troca de valores entre 'a' e 'b' (absurdo lógico)
a = 5
b = 3
# início
a = b
b = a
# troca
assert (a == 3 and b == 5)
```

 — Isso significa que as instruções executadas pelo computador levaram a um estado indesejado: a máquina não realizou um passo de ação computacional!

O efeito da execução das instruções mostra que elas não implementaram a ação de troca especificada. Por vezes, a maneira de se escrever uma ação induz a um erro de implementação. Convém reescrevê-la para que tal tipo de erro tenha uma chance menor de ocorrer. Neste caso, ajuda a criação de uma variável t de modo que $t' = a$. A ideia é "salvar" o valor corrente de a em t, para depois copiá-lo em b:

$$\begin{aligned} \text{início} \triangleq & \wedge a = 5 \\ & \wedge b = 3 \\ \text{troca}_2 \triangleq & \wedge t' = a \\ & \wedge a' = b \\ & \wedge b' = t' \end{aligned}$$

 — Agora sim! A ação "troca$_2$" torna-se verdadeira, se eu admitir que "início" também é:

```
# CENÁRIO de troca de valores entre 'a' e 'b'
a = 5
b = 3
# início
t = a
a = b
b = t
# troca2
assert (a == 3 and b == 5 and t == 5)
```

 — Embora o código seja diferente, a lógica da "troca" é uma consequência da "troca$_2$"...

E Fubã interrompe o seu colega mostrando uma **tabela de estados**.

2.6 Tabela de Estados

 — Aprendi a montar uma **tabela de estados** mostrando os valores das variáveis após a execução de cada instrução.

Em cada linha da tabela, Fubã apresenta uma instrução em Python cuja execução afeta os valores das variáveis que aparecem nas colunas seguintes. No caso da instrução `a=5`, por exemplo apenas o valor de a sofre alteração. O símbolo $\perp$ indica a ideia "não faz sentido", "absurdo"; as variáveis b e t referenciam um valor absurdo:

Instrução	Fórmula	a	b	t
`a = 5`	$\wedge a = 5$	5	$\perp$	$\perp$
`b = 3`	$\wedge a = 5$	5	3	$\perp$
`t = a`	$\wedge t' = a$	5	3	5
`a = b`	$\wedge a' = b$	3	3	5
`b = t`	$\wedge b' = t$	3	5	5

Fubã pensa em um **programa** de computador como um conjunto de instruções que implementam uma sequência de ações.

 — Além disso, a sequência de estados das variáveis é o que se chama de **comportamento** computacional.

É verdade que $[a = 3, b = 5, t = 5]$ no comportamento representado pela tabela de estados criada pelo Fubã.

Comportamento: uma sequência de estados.

2.7 Resumo

1. Variável é "um nome que se refere a um valor".
2. Estado: uma atribuição de valores em variáveis.
3. Ação: especificada por uma regra contendo variáveis com e sem o operador de estado seguinte, "′".
4. Passo de ação: um par de estados que satisfaz uma ação.
5. Comportamento: uma sequência de estados.
6. Tabelas de estados representam comportamentos.

2.8 Exercícios

2.8.1 Estados e Comportamentos

Contexto O rastreio de estados em conclusões intermediárias expõe valores de um comportamento computacional. Considere o seguinte código em Python ornamentado com instruções para mostrar estados em pontos de interesse:

```
# CENÁRIO de teste
a = 5
b = 3
# início
t = a
print (a, b, t); # <-- valor 1
a = b
print (a, b, t); # <-- valor 2
b = t
print (a, b, t); # <-- valor 3
# troca
print (a == 3, b == 5, t == 5)
```

Enunciado Assinale a alternativa contendo uma afirmação **verdadeira**:

1. `Valor 1` corresponde a $[a = 5, t = 3]$.
2. `Valor 2` corresponde a $[b = 5, t = 5]$.
3. `Valor 3` corresponde a $[a = 3, t = 5]$.
4. A afirmação da conclusão é falsa.

2.8.2 Alternância de Estados

Contexto Considere o código em Python que troca o valor entre duas variáveis:

```
# CENÁRIO de programas e provas lógicas
a = 5
b = 3
t = a
a = b
b = t
print (a, b)
```

Um dos comportamentos deste código corresponde à sequência de estados $[a = 5, b = 3] \rightarrow [a = 3, b = 5]$.

Enunciado

1. Modifique o código de modo a ser possível o comportamento: $[a = 5, b = 3] \rightarrow [a = 3, b = 5] \rightarrow [a = 5, b = 3]$.
2. Elabore uma tabela de estados do comportamento exibido pelo código.

2.8.3 Lógica (parcial) de um Quebra-cabeça

Contexto Em um quebra-cabeça Batalha Naval[5] a seguinte lógica foi desenvolvida:

$$\begin{aligned} \text{início} \triangleq & \wedge \text{destruídos} = 0 \\ & \wedge \text{tentativas} = 0 \end{aligned}$$

$$\text{disparo}_{\text{Efetuado}} \triangleq \wedge \text{tentativas}' = \text{tentativas} + 1$$

$$\text{navio}_{\text{Destruído}} \triangleq \wedge \text{destruídos}' = \text{destruídos} + 1$$

Enunciado Assinale uma alternatica contendo uma afirmação verdadeira:

1. $\text{disparo}_{\text{Efetuado}}$ é uma ação.
2. início exemplifica um estado.
3. $\text{navio}_{\text{Destruído}}$ é um predicado de estado.
4. Sempre que um disparo for efetuado é verdade que um navio foi destruído.

2.8.4 Lógica (parcial) de um Quebra-cabeça em Python

Contexto Considere a lógica do exercício anterior.

Enunciado Implemente início, $\text{disparo}_{\text{Efetuado}}$ e $\text{navio}_{\text{Destruído}}$ em Python.

[5] https://www.conceptispuzzles.com/index.aspx?uri=puzzle/battleships/history

Referências

Booch, G., Jacobson, I., & Rumbaugh, J. (1999). *The Unified Modeling Language User Guide*. Addison Wesley.

Eck, D. J. (2021). *Introduction to Programming Using Java*. Hobart; William Smith Colleges.

Lamport, L. (2002). *Specifying Systems: The TLA+ language and tools for hardware and software engineers*. Addison-Wesley. Recuperado de https://lamport.azurewebsites.net/tla/book-21-07-04.pdf

Sedgewick, R., & Wayne, K. (2009). In A.-W. Pearson (Org.), *Introduction to Programming in Java An Interdisciplinary Approach*. Recuperado de https://introcs.cs.princeton.edu/java/home/chapter1.pdf

Sedgewick, R., & Wayne, K. (2020). Lecture 1: Basics. Recuperado de https://introcs.cs.princeton.edu/java/lectures/keynote/CS.1.Basics.pdf