MINISTÉRIO DA EDUCAÇÃO

INSTITUTO FEDERAL DO ESPÍRITO SANTO

REITORIA

Avenida Rio Branco, 50 – Santa Lúcia – 29056-255 – Vitória – ES

27 3357-7500

CONCURSO PÚBLICO
EDITAL Nº 03 / 2015

Professor do Magistério do Ensino Básico, Técnico e Tecnológico

ÍNDICE DE INSCRIÇÃO	336
CAMPUS	Serra
ÁREA/SUBÁREA	Teoria da Computação

PROVA DE CONHECIMENTOS ESPECÍFICOS | DISCURSIVA
MATRIZ DE CORREÇÃO

QUESTÃO 01

A.i) O domínio da interpretação é o universo.

Símbolos predicados:

A(x) : "x é um atacante"
S(x) : "x é um servidor"
P(x,y,z) : "x pode persuadir y de que (o evento) z ocorreu com sucesso"
O(x) : "(o evento) x ocorreu com sucesso"
L(x) : "x é um (evento de) login"

Tradução:

$$\forall x : \forall y : \forall z : (A(x) \wedge S(y) \wedge L(z) \rightarrow P(x, y, z))$$

Como o texto não é específico sobre quais eventos de login podem ser forjados, assim outra tradução possível é:

$$\forall x : \forall y : \exists z : (A(x) \wedge S(y) \wedge L(z) \rightarrow P(x, y, z))$$

Note que não é necessário utilizar o predicado O(x) pois não importa se o login de fato ocorreu ou não.

A.ii) O domínio da interpretação é o universo.

Símbolos predicados:

C(x,u) : "x é a credencial de (o usuário) u"
U(x) : "x é um usuário"
D(x,t) : "x está carregável no momento t"
T(k) : "k é um 'token' de autenticação"

A(u,t,k) : "u foi autenticado no momento t gerando o token k"
K(x,k) : "x é decifrável utilizando o token k"
P(t1,t2) : "o momento t_1 ocorre após o momento t_2"

Tradução:

$$\forall u:\forall x:\forall t_1:(U(u)\wedge C(x,u)\wedge D(x,t_1)\rightarrow\exists t_2:\exists k:(T(k)\wedge A(u,t_2,k)\wedge P(t_1,t_2)))$$
$$\vee\forall u:\forall x:\forall k:(U(u)\wedge C(x,u)\wedge T(k)\wedge K(x,k)\rightarrow\exists t_2:A(u,t_2,k))$$

Note que não foi necessário mencionar explicitamente o formato da mensagem pois está implícito que o formato só pode ser decifrado com o uso do token de autenticação.

A.iii) O domínio da interpretação é o universo.

Símbolos predicados:

P(x) : "x é um protocolo de comunicação"
$G_s(x)$: "x é um algoritmo de criptografia simétrica"
$G_a(x)$: "x é um algoritmo de criptografia assimétrica"
$G_h(x)$: "x é um algoritmo de criptografia de hashing"
$G_m(x)$: "x é um algoritmo de criptografia MAC"
S(x,y) : "(o protocolo) x tem suporte a (o algoritmo) y"

Tradução:

$$\forall x:\big(P(x)\rightarrow\exists y:(G_s(y)\wedge S(x,y))\wedge\exists y:(G_a(y)\wedge S(x,y))$$
$$\wedge\exists y:(G_h(y)\wedge S(x,y))\wedge\exists y:(G_m(y)\wedge S(x,y))\big)$$

B.i)

1	$P \to Q$	
2	$P \vee (R \wedge Q)$	
3	$S \to \neg R$	
4	$\neg(P \wedge Q)$	
5	$\quad S$	
6	$\quad \neg R$	$\Rightarrow$E, 3, 5
7	$\quad \neg R \vee \neg Q$	$\vee$I, 6
8	$\quad \neg(R \wedge Q)$	dm, 7
9	$\quad \neg\neg P \vee (R \wedge Q)$	$\neg\neg i$, 2
10	$\quad \neg P \to (R \wedge Q)$	cnd
11	$\quad \neg\neg P$	MT, 8, 10
12	$\quad P$	$\neg\neg e$, 11
13	$\quad \neg P \vee \neg Q$	dm, 4
14	$\quad P \to \neg Q$	cnd, 13
15	$\quad \neg Q$	$\Rightarrow$E, 12, 14
16	$S \to \neg Q$	$\Rightarrow$I, 5—15
17	$\neg S \vee \neg Q$	cnd, 16

Como o sistema de dedução natural permite a substituição de equivalências tautológicas, foram utilizadas as abreviações *dm*, para representar o uso das equivalência das leis de De Morgan, i.e., $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$ e $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$; e *cnd,* para representar o uso da equivalência condicional, i.e., $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$.

B.ii)

1	$\forall x : (P(x) \rightarrow (Q(x) \vee R(x)))$	
2	$\neg\exists x : (P(x) \wedge R(x))$	
3	$\forall x : \neg(P(x) \wedge R(x))$	QE, 2
4	$\neg(P(a) \wedge R(a))$	$\forall$E, 3
5	$\neg P(a) \vee \neg R(a)$	dm, 4
6	$P(a) \rightarrow \neg R(a)$	cnd, 5
7	$P(a) \rightarrow (Q(a) \vee R(a))$	$\forall$E, 1
8	$\quad P(a)$	
9	$\quad \neg R(a)$	$\Rightarrow$E, 6, 8
10	$\quad Q(a) \vee R(a)$	$\Rightarrow$E, 7, 8
11	$\quad \neg\neg Q(a) \vee R(a)$	$\neg\neg i$, 10
12	$\quad \neg Q(a) \rightarrow R(a)$	cnd, 11
13	$\quad \neg\neg Q(a)$	MT, 9, 12
14	$\quad Q(a)$	$\neg\neg e$, 13
15	$P(a) \rightarrow Q(a)$	$\Rightarrow$I, 8—14
16	$\forall x : (P(x) \rightarrow Q(x))$	$\forall$I, 15

B.iii)

1	$\forall x : \forall y : \forall z : (S(x,y) \wedge S(y,z) \rightarrow S(x,z))$	
2	$\forall x : \neg S(x,x)$	
3	$\neg S(a,a)$	$\forall$E, 2
4	$\forall y : \forall z : (S(a,y) \wedge S(y,z) \rightarrow S(a,z))$	$\forall$E, 1
5	$\forall z : (S(a,b) \wedge S(b,z) \rightarrow S(a,z))$	$\forall$E, 4
6	$S(a,b) \wedge S(b,a) \rightarrow S(a,a)$	$\forall$E, 5
7	$\neg(S(a,b) \wedge S(b,a))$	MT, 3, 6
8	$\neg S(a,b) \vee \neg S(b,a)$	dm, 7
9	$S(a,b) \rightarrow \neg S(b,a)$	cnd, 8
10	$\forall y : (S(a,y) \rightarrow \neg S(y,a))$	$\forall$I, 9
11	$\forall x : \forall y : (S(x,y) \rightarrow \neg S(y,x))$	$\forall$I, 10

QUESTÃO 02

A.i) Seja w tal que $w \in L(G_1)$.

Caso base, $|w| = 1$:

Então $w = a$ ou $w = b$, que trivialmente não tem *ab* como subpalavra.

Caso indutivo, $|w| = k$, onde $k > 1$:

Dado que, pela hipótese de indução, *w* não contém *ab* como subpalavra, então todos os *b*'s de *w* ocorrem apenas à direita de qualquer *a*, e todos os *a*'s ocorrem necessariamente à esquerda de qualquer *b*. Essa é a única forma de garantir que não pode aparecer *ab* como subpalavra de *w*.

Para demonstrar que w_1, onde $|w_1| = k+1$, não contém *ab* como subpalavra, lembramos que as únicas regras de produção da gramática que podem formar w_1 são $S \rightarrow aS|Sb$. Assim, ou w_1 é formada à partir de *w* por concatenação de um *a* à esquerda, ou é formada à partir de *w* por concatenação de um *b* à direita. Em qualquer dos dois casos, fica preservada a propriedade de que todos os *b*'s estarão à direita de qualquer *a*, e de que todos os *a*'s estarão à esquerda de qualquer *b*.

A.ii) Como citado no item anterior, as cadeias $w \in L(G_1)$, tem a propriedade de que todos os *b*'s estarão à direita de qualquer *a*, e de que todos os *a*'s estarão à esquerda de qualquer *b*. Assim, a linguagem $L(G_1)$ é formada por palavra que possuem zero ou mais *a*'s seguidos por zero ou mais *b*'s, mas com ao menos um caractere, i.e., $L(G_1)$ não contém a palavra vazia.

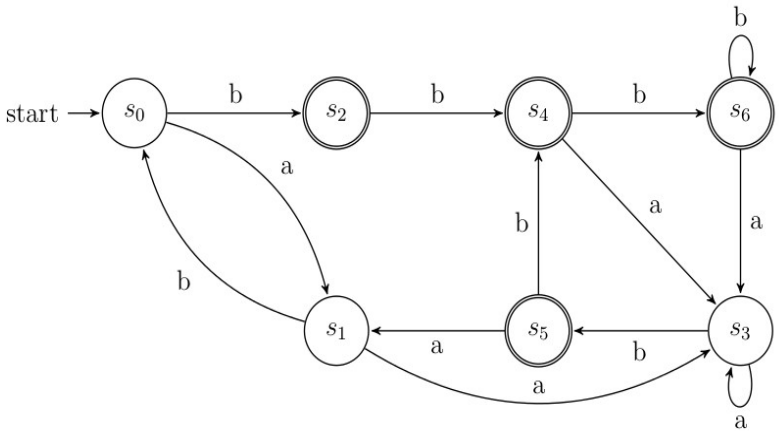
B) A tabela para a função de transição do autômato dado no enunciado é mostrada abaixo:

	a	**b**
→ q_0	$\{q_1\}$	$\{q_2\}$
q_1	$\{q_0,q_1\}$	$\{q_0\}$
*q_2	$\{\}$	$\{q_1,q_2\}$

A tabela que descreve o autômato determinístico equivalente é dada à seguir (entre parênteses, na primeira coluna, aparecem os nomes s_i dos novos estados):

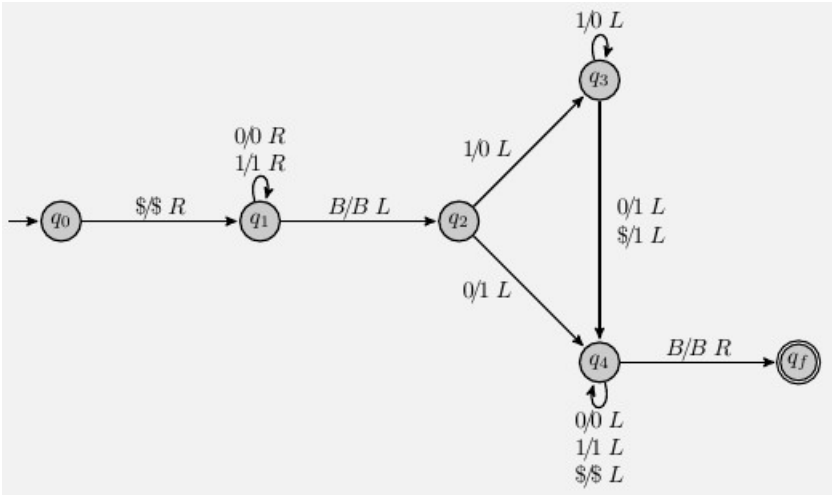
	a	**b**
→ $<q_0>$ (s_0)	$<q_1>$	$<q_2>$
$<q_1>$ (s_1)	$<q_0q_1>$	$<q_0>$
*$<q_2>$ (s_2)	–	$<q_1q_2>$
$<q_0q_1>$ (s_3)	$<q_0q_1>$	$<q_0q_2>$
*$<q_1q_2>$ (s_4)	$<q_0q_1>$	$<q_0q_1q_2>$
*$<q_0q_2>$ (s_5)	$<q_1>$	$<q_1q_2>$
*$<q_0q_1q_2>$ (s_6)	$<q_0q_1>$	$<q_0q_1q_2>$

O gráfico do autômato finito determinístico é mostrado abaixo:



QUESTÃO 03

A. A máquina abaixo realiza o incremento de 1 solicitado.



O estado inicial q_0 serve apenas para ler o caractere $ e posicionar a cabeça de leitura no início do número. A seguir, a entrada é varrida até o final pelo estado q_1. Quando um branco é lido, significa que o bit menos significativo do número foi encontrado e este pode ser incrementado de 1, que é a tarefa do estado q_2. Existem duas possibilidades: se o bit for 0, basta trocá-lo por 1 e retornar para o início da fita. Isto é feito pela transição de q_2 para q_4. Na outra situação, o bit menos significativo era 1, e a máquina entra no estado q_3, aonde todos os bits 1 à esquerda são invertidos. Assim que um caractere diferente de 1 for lido, q_3 escreve 1 e passa para o estado q_4, que é responsável por retornar a cabeça de leitura ao início da fita. Quando isso ocorre, a máquina para no estado final q_f.

B. A computação realizada pela máquina do Item A para a entrada $111 é dada abaixo.

$$
\begin{aligned}
& Bq_0\$111B \\
\vdash\ & B\$q_1111B \\
\vdash\ & B\$1q_111B \\
\vdash\ & B\$11q_11B \\
\vdash\ & B\$111q_1B \\
\vdash\ & B\$11q_21B \\
\vdash\ & B\$1q_310B \\
\vdash\ & B\$q_3100B \\
\vdash\ & Bq_3\$000B \\
\vdash\ & q_4B1000B \\
\vdash\ & Bq_f1000B
\end{aligned}
$$

QUESTÃO 04

Seguem as provas de cada um dos passos apresentados no enunciado.

A. É evidente que *4TA-SAT* está em NP uma vez que *SAT* está em NP. No entanto, para provar tal fato efetivamente, apresentamos uma Máquina de Turing Não-Determinística (MTND) que resolve o problema em tempo polinomial. Tal máquina é composta de duas partes:

(i) Inicialmente usamos a propriedade não-determinística da máquina para gerar quatro atribuições de valores para as proposições atômicas de φ.

(ii) A seguir, cada conjunto de quatro atribuições gerado é avaliado, de forma a testar se a atribuição torna a fórmula verdadeira. Se as quatro atribuições satisfazem a fórmula então a máquina para e aceita a entrada.

Se φ tem tamanho *n*, então a parte (i) pode ser realizada em tempo *O(n)* em uma MTND multi-fita. Já a avaliação de uma atribuição pode ser feita em $O(n^2)$ em uma máquina multi-fita. Como é necessário avaliar quatro atribuições, a computação da parte (ii) leva $T(n)=\left[4\,O(n^2)\right]\in O(n^2)$. Com isso, temos que a MTND resolve o problema em tempo polinomial e portanto, *4TA-SAT* está em NP.

B. Devemos incluir uma sub-fórmula em ψ para obtermos φ. Existem várias possibilidades para a escolha da sub-fórmula, no entanto, para simplificar a prova do item C, tomamos a tautologia $(p\rightarrow q)\vee(q\rightarrow p)$. Sendo uma tautologia, todas as quatro possíveis atribuições de valores para *p* e *q* são soluções. Assim, a redução simplesmente toma uma fórmula *3-SAT* ψ e cria φ fazendo: $\varphi=\psi\wedge((p\rightarrow q)\vee(q\rightarrow p))$. Nesta redução, assumimos sem perda de generalidade que $p,q\notin\psi$ (caso contrário, basta escolher outras variáveis que não ocorram em ψ). É imediato ver que a redução proposta é polinomial.

C. Devemos provar que "ψ é satisfatível" $\Leftrightarrow$ "ψ possui ao menos quatro soluções".

Prova de $\Rightarrow$: se ψ é satisfatível então a fórmula possui ao menos uma solução. Seja $v(\psi)$ a atribuição de valores para as variáveis de ψ que corresponde a tal solução. Utilizando $v(\psi)$, podemos criar quatro soluções para φ, variando os valores de *p* e *q*. Isto é, $\forall X\in\psi$:

$$v_1(X)=\begin{cases} v(X) & \text{se } X\in\psi \\ F & \text{se } X=p \\ F & \text{se } X=q \end{cases} \qquad v_2(X)=\begin{cases} v(X) & \text{se } X\in\psi \\ T & \text{se } X=p \\ F & \text{se } X=q \end{cases}$$

$$v_3(X)=\begin{cases} v(X) & \text{se } X\in\psi \\ F & \text{se } X=p \\ T & \text{se } X=q \end{cases} \qquad v_4(X)=\begin{cases} v(X) & \text{se } X\in\psi \\ T & \text{se } X=p \\ T & \text{se } X=q \end{cases}.$$

Prova de $\Rightarrow$: Se φ possui quatro soluções, elas necessariamente possuem uma sub-solução $v(\psi)$ comum, conforme ilustrado na prova do item anterior. (Em outras palavras, como φ possui uma tautologia como o último termo da conjunção, a satisfatibilidade de φ depende unicamente da satisfatibilidade de ψ.

Se $v(\psi)$ é uma solução então ψ é satisfatível e a prova está completa.

QUESTÃO 05

A. A função abaixo realiza uma busca linear em uma lista conforme solicitado no enunciado.

```
def busca_linear(v, x):
  for i in range(len(v)):
    if v[i] == x:
      return i # x encontrado
  return -1 # x nao encontrado
```

B. A função abaixo realiza uma busca binária em uma lista com ordenação decrescente conforme solicitado.

```
def busca_binaria(vo, x):
  inf = 0
  sup = len(vo) - 1
  while (inf <= sup):
    meio = inf + ((sup - inf) / 2)
    v_meio = vo[meio]
    if v_meio > x:
      inf = meio + 1
    elif v_meio < x:
      sup = meio - 1
    else:
      return meio # x encontrado
  return -1 # x nao encontrado
```

C. Assumindo-se que o acesso a qualquer elemento de uma lista em Python é realizado em *O(1)*, temos as complexidades assintóticas abaixo.

Função	Melhor Caso	Pior Caso	Caso Médio
Busca linear	*O(1)*	*O(n)*	*O(n)*
Busca binária	*O(1)*	$O(\log_2 n)$	$O(\log_2 n)$

Para a busca linear, o melhor caso ocorre quando o valor *x* se encontra na primeira posição da lista, sendo necessária somente uma única comparação. Por outro lado, o pior caso ocorre quando *x* não se encontra na lista, o que leva a função a percorrer toda a lista, realizando *n* comparações. Para o caso médio, assumindo que *x* é igualmente provável de ocorrer em qualquer posição da lista, temos que o número esperado de comparações é $(n + 1) / 2$, que está em *O(n)*.

Para a busca binária, o melhor caso ocorre quando o valor *x* se encontra na primeira posição testada da lista, sendo necessária somente uma única comparação. Por outro lado, o pior caso ocorre quando *x* não se encontra na lista, o que leva a função a realizar $\log_2 n$ comparações. Para o caso médio, temos que o número esperado de comparações é $(\log_2 n) - 1$, que está em $O(\log_2 n)$.

_____ Assinatura Presidente

_____ Assinatura Membro

_____ / _____ / 2015