Respostas

Exercícios do Capítulo 2 – Agentes

1. Uma medida de desempenho é utilizada por um observador externo para avaliar o sucesso de um agente. Uma função de utilidade é utilizada por um agente para avaliar estados. A função de utilidade pode não ser igual à medida de desempenho; além disso, um agente pode não ter uma função de utilidade explícita, enquanto que sempre existe uma medida de desempenho.
2. Sim. Podemos criar um novo programa de agente através da modificação de um programa de agente existente, inserindo-se comandos inúteis que não alterem a saída do programa. Estes dois programas implementam a mesma função de agente.
3. O agente deve ter memória para guardar a informação de que um quadrado já foi limpo. Dessa forma ele só vai para o outro quadrado se ele ainda não verificou que o quadrado estava limpo. A memória que guarda essa informação é um estado interno.
4.

Agente	**Medida de Desempenho**	**Ambiente**	**Atuadores**	**Sensores**
Robô jogador de futebol	Ganhar jogo, saldo de gols	Campo, bola, trave, próprio time, outro time, próprio corpo	Dispositivos para locomoção e chute	Câmera, sensores de toque, acelerômetros, sensores de orientação
Agente de compras na Internet	Obter produtos requisitados, minimizar gastos	Internet	Seguir link, preencher dados em campos, mostrar para usuário	Páginas da Web, pedidos dos usuários
Robô explorador de Marte	Terrenos explorados, amostras recolhidas	Veículo lançador, Marte	Rodas, equipamento coletor de amostras, transmissor de rádio	Câmera, sensores de toque, acelerômetros, sensores de orientação, receptor de rádio

5.

Ambiente	**Observável**	**Determinístico**	**Episódico**	**Estático**	**Discreto**	**Agente**
Futebol	Parcialmente	Estocástico	Seqüencial	Dinâmico	Contínuo	Múltiplo
Internet	Parcialmente	Determinístico	Seqüencial	Estático	Discreto	Único
Marte	Parcialmente	Estocástico	Seqüencial	Dinâmico	Contínuo	Único
Teoremas	Totalmente	Determinístico	Seqüencial	Semi	Discreto	Múltiplo

Exercícios do Capítulo 3 – Busca

1. SUCESSOR(s) = { (a,s') | a $\in$ AÇÕES-VÁLIDAS(s) $\wedge$ s' = RESULTADO(a,s)}

   AÇÕES-VÁLIDAS(s) = { a | (a,s') $\in$ SUCESSOR(s)}

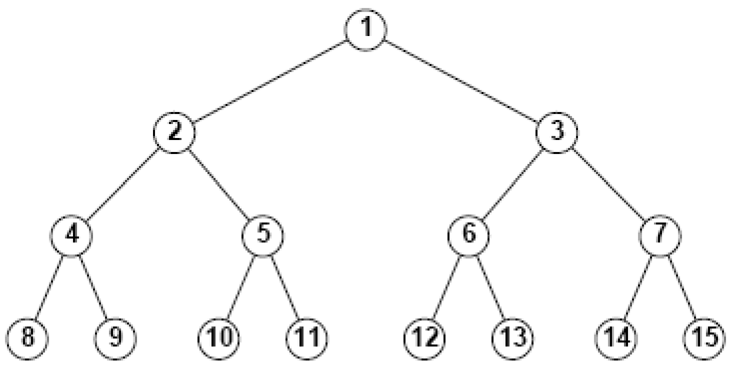
   RESULTADO(a,s) = { s' |(a,s') $\in$ SUCESSOR(s)}

2. Não. Um espaço de estados finitos nem sempre leva a uma árvore de busca finita. Considere um espaço de estados com dois estados, cada um deles com uma ação que leva ao outro. Isso gera uma árvore de busca infinita, porque podemos ir e voltar infinitas vezes. Porém, se o espaço de estados for uma árvore finita, ou, em geral, um DAG (grafo acíclico direcionado), não haverá loops e a árvore de busca será finita.

3.
   a. Estado inicial: nenhuma região colorida.
      Teste de objetivo: todas as regiões coloridas, e nenhuma região adjacente com a mesma cor.
      Função sucessor: atribuir uma cor a uma região que esteja sem cor.
      Função de custo: número total de atribuições (é igual para todas as soluções).

   b. Estado inicial: como descrito no enunciado.
      Teste de objetivo: macaco alcançou as bananas.
      Função sucessor: subir no engradado; descer do engradado; mudar engradado de lugar; andar de um lugar a outro; agarrar bananas
      Função de custo: número total de ações.

4.
   a.

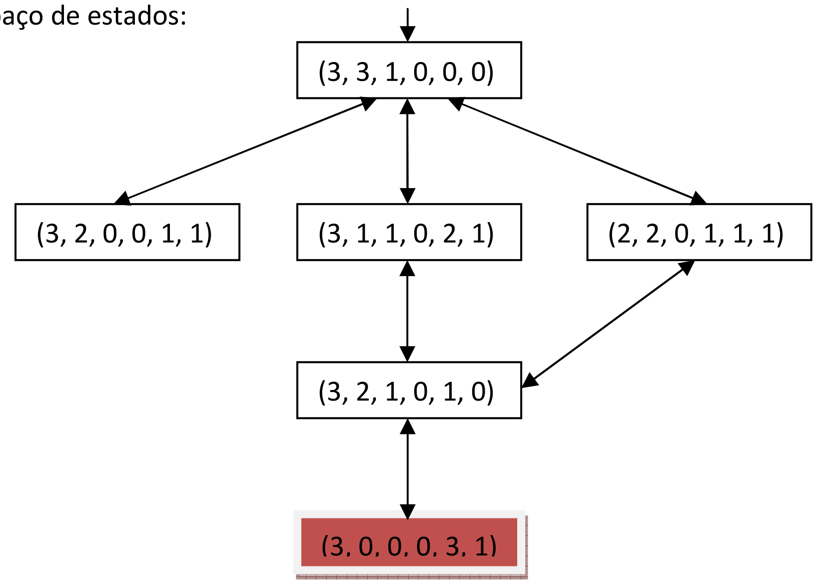


   b. Busca em extensão: 1 2 3 4 5 6 7 8 9 10 11
      Busca em profundidade limitada: 1 2 4 8 9 5 10 11
      Aprofundamento iterativo: 1; 1 2 3; 1 2 4 5 3 6 7; 1 2 4 8 9 5 10 11

5.

a. Uma representação possível: Um estado é um vetor com seis inteiros listando o número de missionários, canibais e barcos em cada lado. O teste de objetivo verifica se o estado tem três missionários de um lado e três canibais do outro. A função de custo tem valor um para cada ação. Os sucessores de um estado são todos os estados que movem uma ou duas pessoas e um barco de um lado para o outro, sem que o número de missionários de um lado fique menor que o número de canibais.

Espaço de estados:



b. Qualquer algoritmo de busca funciona bem, porque o espaço de estados é muito pequeno. Basta eliminar estados repetidos e estados inválidos (com maior número de canibais do que missionários do mesmo lado).

Exercícios do Capítulo 4 – Busca Heurística

1. Seqüência de nós na borda:

L[0+244=244]

M[70+241=311], T[111+329=440]

L[140+244=384], D[145+242=387], T[111+329=440]

D[145+242=387], T[111+329=440], M[210+241=451], T[251+329=580]

C[265+160=425], T[111+329=440], M[210+241=451], M[220+241=461], T[251+329=580]

T[111+329=440], M[210+241=451], M[220+241=461], P[403+100=503], T[251+329=580], R[411+193=604], D[385+242=627]

M[210+241=451], M[220+241=461], L[222+244=466], P[403+100=503], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627]

M[220+241=461], L[222+244=466], P[403+100=503], L[280+244=524], D[285+242=527], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627]

L[222+244=466], P[403+100=503], L[280+244=524], D[285+242=527], L[290+244=534], D[295+242=537], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627]

P[403+100=503], L[280+244=524], D[285+242=527], M[292+241=533], L[290+244=534], D[295+242=537], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627], T[333+329=662]

B[504+0=504], L[280+244=524], D[285+242=527], M[292+241=533], L[290+244=534], D[295+242=537], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627], T[333+329=662], R[500+193=693], C[541+160=701]

2. w=0 faz com que f(n) = 2g(n), que equivale à busca de custo uniforme (a multiplicação por 2 não modifica a ordem em que os nós são expandidos).

   w=1 faz com que f(n)= g(n)+h(n), que equivale à busca A*.

   w=2 faz com que f(n) = 2h(n), que equivale à busca gulosa pela melhor escolha.

   Este algoritmo é ótimo quando h(n) é admissível e w≤1.

3.
   a. Quando todos os custos são iguais, temos que g(n) α profundidade(n), logo a busca de custo uniforme reproduz a busca em extensão, já que os nós são expandidos em ordem de menor profundidade (menor custo) para maior profundidade.
   b. Busca em extensão equivale à busca pela melhor escolha com f(n)=profundidade(n); busca em profundidade é busca pela melhor escolha com f(n)=-profundidade(n); busca de custo uniforme é busca pela melhor escolha com f(n)=g(n).
   c. Busca de custo uniforme equivale a A* com h(n)=0.

4. A heurística dos blocos mal posicionados é exata para o problema em que um bloco pode ser movido de um quadrado A para qualquer quadrado B. Como esse problema é um relaxamento da condição que um quadrado pode ser movido do quadrado A para um quadrado B se B estiver vazio, o valor da heurística de Gaschnig não pode ser menor que o valor da heurística dos blocos mal-posicionados. Como ela também é admissível (por ser um relaxamento do problema original), ela é mais precisa.

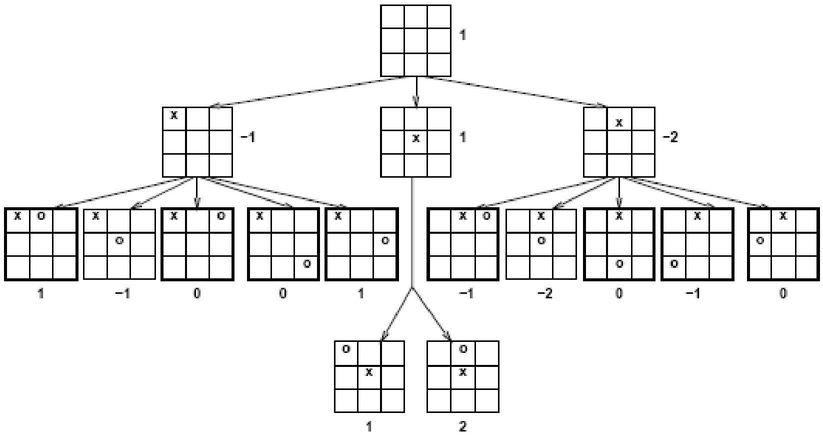
   Se permutarmos dois blocos adjacentes no estado objetivo, teremos um estado em que a heurística dos blocos mal-posicionados e a heurística da distância Manhattan terão valor 2, e a heurística de Gaschnig terá valor 3.

   Para calcular a heurística de Gaschnig, repita o seguinte até que o estado objetivo seja atingido: seja B a posição atual do espaço vazio; se B for ocupado pelo quadrado X (não vazio) no estado objetivo, mova X para B; senão, mova qualquer bloco mal posicionado para B.

5.
   a. Busca de subida de encosta.
   b. (Não faz sentido)
   c. Busca de subida de encosta.
   d. Busca aleatória.

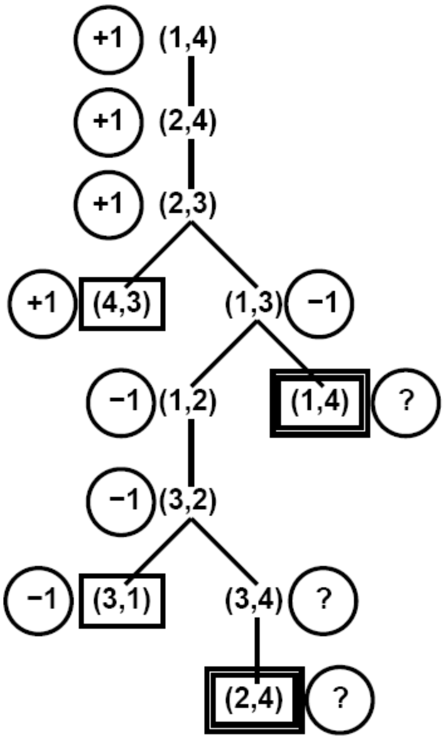
Exercícios do Capítulo 6 – Busca Competitiva

1. (nós que não seriam avaliados com a poda alfa-beta estão em negrito)



   A melhor jogada inicial de acordo com a estratégia minimax é colocar o X no centro.

2.
   a.



   b. Os valores “?” são tratados supondo que um agente que possa escolher entre ganhar o jogo e entrar num estado “?” sempre escolherá ganhar. Isto quer dizer que min(-1,?) é -1 e max(+1,?) é +1. Se todos os sucessores são “?”, o valor propagado é “?”.
   c. O minimax padrão é uma busca em profundidade e entraria em loop infinito. Isto pode ser corrigido comparando-se o estado atual e o estado na pilha; se o

estado for repetido, deve-se retornar o valor “?”. A propagação de valores “?” é feita como descrito no item (b).

Exercícios do Capítulo 7 – Agentes Lógicos

1. Lembre de considerar o valor de todas os 4 símbolos (A, B, C e D) na hora de fazer a tabela-verdade.
    a. 6
    b. 12
    c. 4
2.
    a. Válida
    b. Satisfatível
    c. Satisfatível
    d. Válida
    e. Válida
    f. Válida
    g. Válida
    h. Satisfatível

3. Dadas as duas primeiras sentenças vemos que se o unicórnio é mítico, então ele é imortal; senão ele é um mamífero. Logo ele deve ser ou imortal ou mamífero, e logo tem chifre. Isso significa que ele também é mágico. Porém, no podemos deduzir nada sobre se ele é mítico ou não.
4. Cada mundo possível pode ser descrito como uma conjunção de símbolos, isto é, $(A \wedge C \wedge E)$. Podemos afirmar que um mundo possível não pode ocorrer negando essa conjunção, isto, é, $\neg(A \wedge C \wedge E)$, que pode ser reescrita como $\neg A \vee \neg C \vee \neg E$. Uma conjunção dessas cláusulas é uma sentença em FNC, e pode listar todos os mundos possíveis para a sentença.

Exercícios do Capítulo 8 – Lógica de Primeira Ordem

1. A base de conhecimento não tem como conseqüência lógica $\forall x\, P(x)$. Para mostrar isso, devemos encontrar um modelo onde $P(a)$ e P(b) são verdadeiros mas $\forall x\, P(x)$ é falso. Considere qualquer modelo com três elementos no domínio, onde *a* e *b* se referem aos dois primeiros elementos e a relação correspondente a *P* somente é verdadeira para esses dois elementos, e não para o terceiro elemento.
2. A sentença $\exists x,y\ x=y$ é válida. Uma sentença é válida se for verdadeira para todos os modelos possíveis. Uma sentença existencialmente quantificada é verdadeira em um modelo se ela for verdadeira para qualquer interpretação estendida na qual elementos do domínio são atribuídos às variáveis. De acordo com a semântica padrão da lógica de primeira ordem, cada modelo deve conter pelo menos um elemento, logo, para qualquer

modelo, há uma interpretação estendida na qual o primeiro elemento é atribuído a *x* e *y*. Nessa interpretação, *x* = *y* é verdadeira.

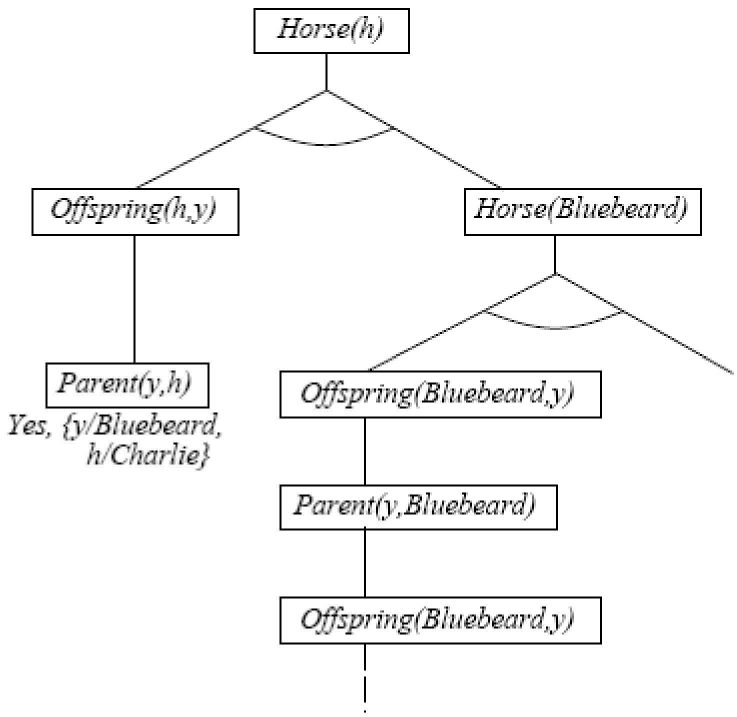
3. $\forall x,y\ x=y$ estipula que existe apenas um objeto no domínio. Se existirem dois objetos, então existe uma interpretação estendida na qual objetos diferentes são atribuídos a *x* e *y*, logo a sentença seria falsa.
4. $\forall x,y,l\ Alemão(x) \land Alemão(y) \land Fala(x,l) \Rightarrow \text{Fala}(y,l)$
5. $\forall x,y\ Cônjuge(x,y) \land Masculino(y) \Rightarrow Feminino(y)$
6. Há vários problemas com essa definição. Ela permite provar Adjacente([1,1],[1,2]), mas não Adjacente([1,2],[1,1]), então precisamos de um axioma adicional de simetria. Também não é possível provar que Adjacente([1,1],[1,3]) é falso, então devemos reescrevê-lo como $\forall$x, y $\Leftrightarrow$ Adjacente([x,y],[x+1,y]) $\land$ Adjacente([x,y],[x,y+1])
   Finalmente, ele não funciona para os quadrados das paredes e dos cantos, então algumas condições extras têm que ser incluídas.

Exercícios do Capítulo 9 – Inferência em Lógica de Primeira Ordem

1. Tanto *b* quanto *c* são legítimas, *a* não é legítima porque introduz um símbolo que já tinha sido usado (*Everest*). Note que *c* não implica que existam duas montanhas tão altas quanto o *Everest*, porque em *nenhum* lugar está escrito que *BenNevis* é diferente de *Kilimanjaro*, nem de *Everest*.
2.
   a. {x/A,y/B,z/B}
   b. Nenhum unificador (x não pode ser atribuído a A e B simultaneamente).
   c. {y/João, x/João}.
   d. Nenhum unificador (porque a verificação de ocorrência impede a unificação de y e Pai(y)).
3.
   a. Cavalo(x) $\Rightarrow$ Mamífero(x)
      Vaca(x) $\Rightarrow$ Mamífero(x)
      Porco(x) $\Rightarrow$ Mamífero(x)
   b. Descendente(x,y)$\land$Cavalo(y) $\Rightarrow$Cavalo(x)
   c. Cavalo(Bluebeard)
   d. Pai(Bluebeard,Charlie)
   e. Descendente(x,y) $\Rightarrow$ Pai(y,x)
      Pai(x,y) $\Rightarrow$ Descendente(y,x)
      (não podemos usar $\Leftrightarrow$ porque não poderia ser usado no Modus Ponens)
   f. Mamífero(x) $\Rightarrow$ Pai(G(x),x) (aqui G é uma função de Skolem)

4.

a.



b. Temos um loop infinito por causa da regra (b). O loop específico que aparece na figura é causado pela ordem das cláusulas. Seria melhor ordenar Cavalo(Bluebeard) antes da regra (b). Porém, um loop ocorrerá de qualquer maneira se quisermos encontrar todas as soluções.

c. Deveríamos ser capazes de provar que tanto Bluebeard quanto Charlie são cavalos.

d. Sempre que um loop ocorrer num ramo da árvore de prova, devemos parar de expandir a árvore nesse ramo e continuar com todos os outros ramos, coletando as soluções. Então, devemos usar essas soluções para tentar provar o ramo suspenso.

5.

a. $\forall x$ Cavalo(x) $\Rightarrow$ Animal(x)
   $\forall x, h$ Cavalo(x) $\wedge$ CabeçaDe(h,x) $\Rightarrow$ $\exists y$ Animal(y) $\wedge$ CabeçaDe(h,y)

b. A. $\neg$Cavalo(x) $\vee$ Animal(x)
   B. Cavalo(G)
   C. CabeçaDe(H,G)
   D. $\neg$Animal(y) $\vee$ $\neg$CabeçaDe(H,y)
   (H e G são constantes de Skolem)

c. Resolver D e C para obter $\neg$Animal(G). Resolver $\neg$Animal(G) com A para obter $\neg$Cavalo(G). Resolver $\neg$Cavalo(G) com B para obter uma contradição.