2
Máquinas de Estados em Jogos Eletrônicos

Máquinas de Estados são um conceito importante em várias áreas da ciência. Em particular, a engenharia e a computação utilizam Máquinas de Estados como ferramentas no estudo de processos sistemáticos, na especificação de componentes lógicos e dispositivos de controle, e em muitas outras tarefas práticas. Além disso, Máquinas de Estados têm um papel crucial na Teoria da Computação, sendo freqüentemente utilizadas na formalização de muitos conceitos fundamentais, e no estudo formal sobre linguagens e autômatos.

Este capítulo apresenta alguns tipos de Máquinas de Estados que serão especialmente úteis no desenvolvimento de jogos eletrônicos. O discurso é iniciado com a apresentação de Máquinas de Estados como um conceito genérico; os tipos específicos de Máquinas utilizados em jogos são apresentados logo em seguida.

As estruturas apresentadas serão introduzidas na arquitetura de um módulo de inteligência artificial para jogos eletrônicos, discutindo-se a integração das Máquinas de Estados com os demais processos existentes na inteligência artificial para jogos.

2.1.
Máquinas de Estados

Máquinas de Estados são estruturas lógicas compostas por um conjunto de *estados* e um conjunto de *regras de transição* entre os estados. São ferramentas úteis em qualquer aplicação que envolva o controle de processos, na qual seja possível descrever cada uma das situações discretas em que os processos podem se encontrar a cada momento. Essas situações definem os estados da Máquina, e as regras de transição representam as condições necessárias para que o processo atinja uma nova situação a partir de um determinado *estado corrente*.

O funcionamento de uma Máquina de Estados compreende ainda a alimentação de alguns *dados de entrada* à Máquina, que serão utilizados na

avaliação das regras de transição. Adicionalmente, a execução da Máquina de Estados pode gerar *dados de saída* como resultado do andamento do processo.

Um dos estados da Máquina é marcado como *estado inicial*; a execução é iniciada com o estado inicial como estado corrente. Além disso, um ou mais estados podem ser marcados como *estados finais*; se, ao término da execução da Máquina, o estado corrente corresponder a um dos estados finais, diz-se que a Máquina *aceita* a entrada que lhe foi submetida; caso contrário, é comum dizer que a Máquina *rejeita* a entrada.

A cada iteração, a execução da máquina ocorre da seguinte maneira: dados de entrada são lidos; as regras de transição que saem do estado corrente são avaliadas; se alguma delas tiver suas condições satisfeitas, ocorre uma transição, e o estado corrente passa a ser o estado de chegada da transição efetuada; finalmente, dados de saída são escritos.

O funcionamento das Máquinas de Estados descrito acima é suficientemente genérico para se referir aos diversos tipos de Máquinas de Estados. Um exemplo dessa generalidade é a possibilidade de haver mais de uma transição possível a partir do estado corrente para uma mesma entrada: o mecanismo descrito no parágrafo anterior não apresenta restrições quanto à possibilidade de mais de uma regra de transição terem suas condições satisfeitas com uma dada entrada. Observa-se, portanto, que o funcionamento descrito se aplica tanto às Máquinas de Estados *Determinísticas* (que não permitem múltiplas transições de um mesmo estado com a mesma entrada) quanto às *Não-Determinísticas* (que, ao contrário, permitem essas transições). Máquinas de Estados Determinísticas e Não-Determinísticas são descritas em detalhes por Hopcroft & Ullman (1979).

Além disso, o funcionamento genérico das Máquinas de Estados apresentado anteriormente não versa sobre as estruturas de dados internas utilizadas durante a execução. Tipicamente, o marcador que indica o estado corrente estará presente na maioria dos tipos de Máquinas de Estados, mas é comum encontrar-se estruturas mais sofisticadas, como pilhas ou memória (Hopcroft *et al*. 2001). O funcionamento descrito se aplica a Máquinas de Estados que utilizam quaisquer dessas estruturas, indistintamente.

É muito comum associar-se o estudo das Máquinas de Estados ao das linguagens formais. De fato, Máquinas de Estados (ou *Autômatos*, seu nome mais comum nesse contexto) são ferramentas importantes na Teoria da Computação, na

especificação e construção de compiladores, e em muitas outras áreas que envolvem o reconhecimento de linguagens. Utilizados com esses propósitos, os autômatos geralmente recebem como entrada letras ou seqüências de letras, e também escrevem esses tipos de dados na saída. Nesta obra, entretanto, Máquinas de Estados serão apresentadas de maneira mais genérica, para que seja possível discutir sua utilização em uma aplicação que a rigor não envolve o reconhecimento de linguagens: o controle do comportamento de agentes inteligentes em jogos eletrônicos. Os dados de entrada e saída, nesse caso, não serão propriamente letras ou seqüências de letras, mas sim eventos e outros sinais importantes no jogo.

Esse tratamento diferenciado dos dados de entrada e saída não descaracteriza as Máquinas de Estados, nem tampouco impossibilita que a base teórica amplamente desenvolvida sobre o assunto seja utilizada no contexto dos agentes em jogos, uma vez que quaisquer eventos ou sinais existentes no sistema do jogo (e, na verdade, em qualquer sistema formal) podem ser codificados como um conjunto de seqüências de letras; as seqüências que representam ações e eventos possíveis no jogo (*teoremas* do sistema formal) podem ser agregados em uma linguagem, que será exatamente a linguagem aceita pela Máquina de Estados (Papadimitriou 1994).

2.1.1. Máquinas de Estados Finitos

Quando o número de estados da Máquina é finito, tem-se uma *Máquina de Estados Finitos* (ou simplesmente *FSM*, do inglês *Finite State Machine*). É o tipo de Máquina mais comum na literatura e nas aplicações. Sua definição simples facilita a implementação, compreendendo contudo uma ferramenta adequada na especificação e simulação de processos com um número finito de estados. Uma definição formal da FSM é mostrada a seguir.

Definição 2.1 Uma Máquina de Estados Finitos (FSM) é uma 5-tupla $\langle Q, \Sigma, \delta, q_0, F \rangle$, onde:

Q é o conjunto finito de estados;

Σ é o conjunto de possíveis dados de entrada, normalmente chamado de *alfabeto*;

$\delta : Q \times \Sigma \to Q$ é a *função de transição* entre estados[1];

q_0 é o estado inicial;

F é o conjunto de estados finais.

A função de transição $\delta(q, a) = p$ mapeia o estado q e o dado de entrada a ao estado p. Isso significa dizer que existe uma transição do estado q ao estado p, e que a condição para que essa transição aconteça é a estar presente na entrada lida.

O funcionamento de uma Máquina de Estados Finitos ocorre da seguinte maneira: a cada iteração, um dado de entrada a é lido; se houver uma transição da forma $\delta(q, a) = p$, onde q é o estado corrente, ocorre a transição, e o novo estado corrente passa a ser p. Esse processo continua recursivamente até que não haja mais dados de entrada para serem lidos; nesse caso, a Máquina *aceita* se o estado corrente for um dos estados finais (isto é, se pertencer a F), ou *rejeita*, caso contrário.

Se não houver transição possível a partir do estado corrente com a entrada lida, a máquina permanece no estado atual, e o processo continua. Formalmente, isso significa que devem ser adicionadas *auto-transições* a cada um dos estados, sempre que houver dados de entrada não referenciados pelas outras transições que saem desse estado (isto é, adiciona-se a transição $\delta(q, a) = q$ para todo a com o qual não haja transição partindo de q)[2]. Essa convenção é adotada nesta obra, e as auto-transições são omitidas em prol da legibilidade e concisão na representação

[1] Alguns autores apresentam uma definição da função de transição que inclui a escrita de um símbolo na saída. Segundo essa abordagem, a função é definida como $\delta : Q \times \Sigma \to Q \times \Omega$, sendo Ω o conjunto de símbolos de saída permitidos. A definição apresentada nesta obra, utilizada também por Hopcroft e Ullman (1979) e Rocha (1999), é mais concisa, considerando que *ações* podem ser executadas em transições ou associadas a estados, podendo essas ações incluir, por exemplo, a escrita de símbolos na saída. Essa associação de ações a transições e estados é discutida com detalhes na **Seção 4.1.1**.

[2] Nos casos de Máquinas de Estados que efetuam transições tendo como entrada a palavra vazia (comumente denotada ε), assume-se que haja auto-transições com a palavra vazia em todos os estados onde não haja uma transição explícita com ε.

dos exemplos. Hopcroft *et al*. (2001) comentam sobre a necessidade dessa completude das Máquinas de Estados Finitos.

2.1.2. Máquinas de Estados *Fuzzy*

Lógica Fuzzy é um ramo da Lógica concebido inicialmente por Lofti Zadeh (1965), inspirado na *Teoria Nebulosa dos Conjuntos* (*Fuzzy Set Theory*), cuja principal característica é a quebra da *dicotomia* existente nas relações de pertinência. Na Teoria Nebulosa, é possível que um elemento pertença apenas parcialmente a um conjunto, e dá-se o nome de *grau de pertinência* a um valor numérico relativo associado a cada par elemento-conjunto, que denota o quanto o elemento pertence ao conjunto (Tanaka 1997).

Analogamente, uma sentença na Lógica *Fuzzy* pode ser parcialmente verdadeira. Associa-se à sentença um *valor* ou *grau de verdade*, que é um valor numérico dentro de um intervalo determinado que representa o quão verdadeira é a sentença. Os limites inferior e superior desse intervalo representam graus de verdade correspondentes a "totalmente falso" e "totalmente verdadeiro", respectivamente[3]. Essa propriedade adiciona à Lógica o poder de tratar situações de *incerteza*, nas quais não esteja clara a verdade das sentenças, mas seja possível atribuir-lhes valores relativos de verdade (Klir & Yuan 1995).

Nesse ínterim, pode-se definir um tipo de Máquina de Estados que associe o conceito de estado corrente à incerteza da Lógica *Fuzzy*. Uma *Máquina de Estados Fuzzy* (ou simplesmente *FuSM*, do inglês *Fuzzy State Machine*) exibe essa propriedade associando a cada estado um valor de verdade, que corresponde ao quanto esse estado se aproxima de ser o estado corrente na situação atual do processo. O estado corrente passa a ser um conjunto de estados, e cada estado

[3] A rigor, o intervalo de valores pode ser qualquer, bastando apenas que os valores associados às sentenças pertençam ao intervalo e correspondam à gradação entre "totalmente verdadeiro" e "totalmente falso". Os limites do intervalo, que correspondem aos valores dicotômicos, poderiam, por exemplo, estar invertidos, com o valor correspondente a "totalmente verdadeiro" sendo o limite inferior do intervalo. Nesse caso, seria mais adequado chamar os valores de *graus de falsidade*. A associação dos limites aos valores citados são uma mera convenção do autor, adotada no texto por razões didáticas.

possui um valor de verdade que corresponde ao grau de pertinência desse estado ao *conjunto estado corrente*.

Cada estado de uma Máquina de Estados representa uma "situação ideal" em que o processo pode se encontrar em um dado momento. Enquanto nas Máquinas de Estados Finitos a representação do processo está restrita a figurar *exatamente* em uma dessas "situações ideais" num dado momento, na Máquina de Estados *Fuzzy* o processo pode se encontrar numa situação intermediária, que mistura características de mais de uma das "situações ideais". Isso sigifica dizer que a Máquina de Estados *Fuzzy* pode não se encontrar exatamente em um determinado estado em cada momento; seu estado corrente pode representar múltiplos estados.

A definição formal das Máquinas de Estados *Fuzzy* é idêntica à das Máquinas de Estados Finitos. As diferenças entre as Máquinas encontram-se na descrição de seu funcionamento e nas estruturas de dados internas que são utilizadas durante sua execução, bem como no tratamento da entrada (Brough 1997). Os dados de entrada submetidos a uma Máquina de Estados *Fuzzy* também são tratados como eventos incertos, e as condições associadas às transições representam uma *tendência* dos dados de entrada, numa abordagem utilizada também por Brubaker (1990). Esses dados substituem as variáveis presentes na condição, que passa a ser avaliada como uma sentença da Lógica *Fuzzy*.

O funcionamento de uma Máquina de Estados *Fuzzy* envolve a avaliação das regras de transição que saem de todos os estados cujo valor de verdade seja não-nulo (isto é, que possuam um grau positivo de pertinência ao conjunto estado corrente). Quando uma transição é efetuada, parte do grau de pertinência ao conjunto estado corrente do estado de origem é transferida ao estado destino.

Suponha-se, por exemplo, uma transição $\delta(q,a) = p$ de uma Máquina de Estados *Fuzzy*, onde a condição associada a a seja "$x \approx 5$" ("x é aproximadamente 5") [4], onde x é uma variável que será substituída pelo dado lido. Suponha-se também que os estados q e p possuam, num determinado momento, graus de pertinência ao conjunto estado corrente μ_q e μ_p,

[4] É importante observar que as condições a que se referem as regras de transição podem sempre ser codificadas como seqüências de letras (Papadimitriou 1994); assim, a função de transição das Máquinas de Estados *Fuzzy* é definida de maneira idêntica à das Máquinas de Estados Finitos.

respectivamente, com μ_q não-nulo. Na avaliação dessa regra de transição, o dado de entrada lido (digamos, o valor numérico 4) substitui a varável x na condição. É avaliado então o grau de verdade da sentença "4 é aproximadamente 5", e esse valor servirá como referência para determinar a *intensidade* da transição. Em outras palavras, o grau de verdade da sentença determina qual a porção de μ_q que será transferida a μ_p. Se, por exemplo, o grau de verdade da sentença "4 é aproximadamente 5" for μ_s, com $0 \le \mu_s \le 1$, o valor resultante de μ_q e μ_p serão, respectivamente:

$$\mu_q = \mu_q - (\mu_q \mu_s) = \mu_q (1 - \mu_s)$$
$$\mu_p = \mu_p + (\mu_q \mu_s)$$

Dessa maneira, a porção de grau de pertinência diminuída do estado de origem da transição é adicionada no estado destino, e o valor somado dos graus de pertinência de todos os estados da máquina se mantém constante. Se for considerado que, no início da execução, o grau de pertinência do estado inicial é igual a 1, sendo os graus dos demais estados todos iguais a zero, tem-se que essa quantidade será distribuída pelos estados durante a execução, mas o somatório de todos os graus permanecerá com o valor 1. Isso permitirá a interpretação do grau de pertinência dos estados como probabilidades, facilitando a utilização de Máquinas de Estados *Fuzzy* em aplicações como jogos eletrônicos. Essa abordagem será discutida na **Seção 2.2.3**.

A **Seção 2.2.3** adiante exibe exemplos de Máquinas de Estados *Fuzzy* e ilustra seu funcionamento com mais detalhes em situações típicas de sua utilização em agentes inteligentes para jogos eletrônicos.

Variantes de Máquinas de Estados *Fuzzy* podem ser definidas modificando-se o mecanismo de cálculo operado nas transições. Brubaker (1990) e Dybsand (2001) apresentam alternativas à técnica descrita acima, indicando algumas de suas aplicações mais adequadas.

2.2. Módulo de Inteligência Artificial em Jogos Eletrônicos

Jogos eletrônicos são um tipo de aplicação que demanda a utilização de técnicas presentes em várias áreas da Computação. Engenharia de *Software*, Computação Gráfica, Inteligência Artificial, Interface Homem-Máquina e Redes de Computadores são apenas algumas das disciplinas envolvidas em um típico projeto de jogo eletrônico atual.

Para possibilitar que essa multiplicidade de técnicas seja organizada e condensada em um sistema compacto, é comum dividir-se o sistema que executa o jogo eletrônico em módulos, com cada módulo sendo responsável pelas soluções de uma determinada área de conhecimento da Computação. Dessa forma, é comum haver módulos distintos dedicados às tarefas de renderização, interface com o jogador e comunicação via rede (em jogos muilti-jogador), que executam suas tarefas de forma independente.

Da mesma maneira, é natural que as tarefas da Inteligência Artificial sejam desenvolvidas em um módulo independente. O Módulo de IA de um jogo eletrônico típico é responsável pela construção do comportamento dos agentes inteligentes, que participam do jogo como adversários ou aliados do jogador, de maneira independente com relação aos demais módulos que compõem o sistema do jogo (Lopes 2002a). Algumas tarefas típicas que são implementadas no Módulo de IA são: busca de caminhos no espaço da cena do jogo, percepção do mundo do jogo, tomadas de decisões sobre as próximas ações do agente, etc.

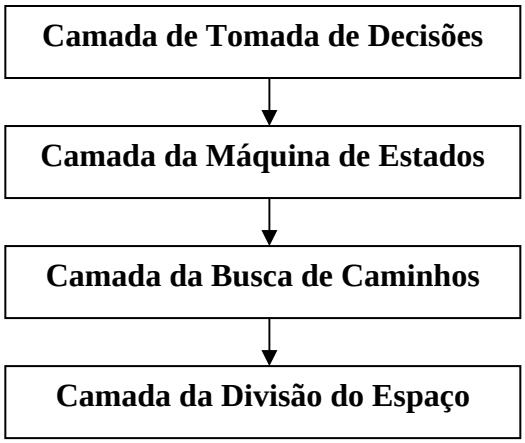

Figura 2.1. Divisão em camadas do Módulo de Inteligência Artificial.

Com o reconhecimento de uma gradação na complexidade dos problemas e de uma relação hierárquica entre as tarefas a serem desenvolvidas, é possível propor uma organização do Módulo de IA para jogos eletrônicos em camadas, de forma a espelhar essas características e isolar processos autônomos dentro do módulo. Uma das propostas mais aceitas de divisão do Módulo de IA em camadas (Lopes 2003) pode ser observada na **Figura 2.1**.

Na arquitetura proposta, a camada inferior é responsável pelo pré-processamento de cada cena do jogo para a subdivisão de seu espaço em unidades lógicas sobre as quais seja possível aplicar um algoritmo de busca de caminhos; a camada imediatamente superior é responsável por executar essa busca a partir da subdivisão pré-computada. A camada da Máquina de Estados, na qual se concentra a contribuição proveniente desta obra, é responsável por implementar o comportamento dos agentes inteligentes através de estados que representam sua situação em cada momento do jogo. O estado corrente determinará os caminhos a serem buscados pelo agente na cena do jogo. Finalmente, a camada superior é encarregada da percepção do mundo do jogo, produzindo dados sobre a situação atual da partida que servirão de entrada para a Máquina de Estados. Essa relação, onde uma camada utiliza-se das soluções oferecidas pela camada imediatamente inferior para realizar suas tarefas, está explicitada pelas setas existentes na **Figura 2.1**.

Os processos de subdivisão do espaço das cenas de jogos 3D são bastante discutidos na literatura da área. Em particular, as obras de Watt & Policarpo (2000, 2002) e Eberly (2000) incluem esses processos como parte fundamental do desenvolvimento de jogos 3D. Busca de caminhos para jogos também é um assunto bastante explorado; particularmente, o autor da presente obra desenvolveu vários trabalhos anteriores focalizados nessa camada (Lopes 2002a, 2002b, 2003).

2.2.1. Camada da Máquina de Estados

Na Camada da Máquina de Estados estão concentrados os processos e estruturas de dados que atribuem comportamento aos agentes inteligentes de jogos. As possíveis situações de jogo relevantes à participação do agente são

mapeadas em estados, e as condições de transição dependem da avaliação de dados de entrada provenientes da percepção do agente sobre o estado do jogo.

Um exemplo de Máquina de Estados, com estados e transições representando situações e eventos típicos de jogos, pode ser observado na **Figura 2.2**.

O exemplo apresentado representa o comportamento de um agente que está inicialmente ocioso, e permanece nessa situação até que um inimigo seja percebido. A partir daí, o agente começa a persegui-lo até que esteja ao alcance de seu ataque, passando a atacá-lo em seguida. Durante a perseguição e o ataque, uma medida de "saúde" do agente sofre decréscimos; se a saúde se tornar muito baixa, o agente deverá abandonar sua tarefa corrente e descansar para recuperar a saúde.

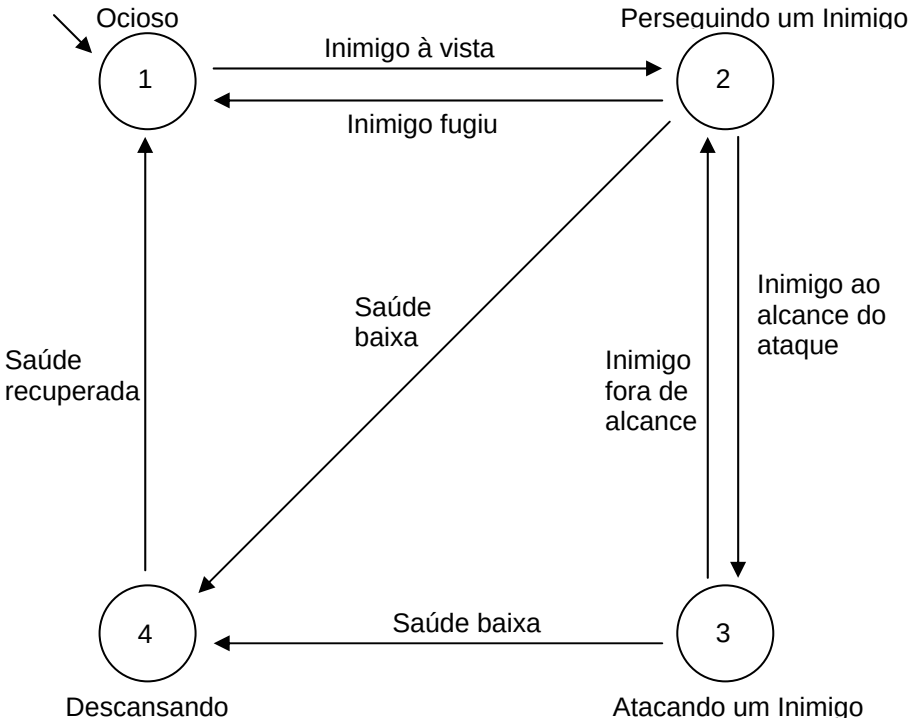

Figura 2.2. Exemplo de Máquina de Estados com situações típicas de jogos.

Merecem especial atenção as regras de transição no exemplo da **Figura 2.2**: a determinação de seu valor de verdade depende de dados de entrada que serão fornecidos à Máquina pelos processos da Camada de Tomada de Decisões. Esses dados provêm da percepção que o agente tem do jogo, que determina se existe algum inimigo visível, a que distância se encontra, etc. A percepção

também envolve a consulta de valores de variáveis que representam a condição atual do agente, como a “saúde” no exemplo.

O exemplo da **Figura 2.2** é propositadamente concebido para se aplicar tanto a Máquinas de Estados Finitos quanto a Máquinas de Estados *Fuzzy*. As regras de transição são especificadas de maneira genérica, possibilitando sua interpretação em ambos os tipos de Máquina. As seções a seguir apresentam a adaptação do exemplo para esses dois tipos de Máquinas de Estados.

2.2.2. Máquinas de Estados Finitos em Jogos

A simplicidade das Máquinas de Estados Finitos as tornam ferramentas apropriadas para a representação do comportamento de agentes de jogos cujas ações sejam simples e as situações possíveis sejam fáceis de serem determinadas. Agentes cujo comportamento é representado por Máquinas de Estados Finitos apresentam o comportamento dicotômico característico das FSMs.

A **Figura 2.3** apresenta uma adaptação do exemplo da **Figura 2.2** para o caso de Máquinas de Estados Finitos. É importante observar que as condições de transição assumiram um caráter dicotômico: tornaram-se sentenças da lógica proposicional, cujo valor de verdade a cada momento pode ser apenas *verdadeiro* ou *falso*. Com isso, determinar se uma transição pode ser efetuada consiste apenas em determinar o valor de verdade da condição.

Pode haver casos onde exista mais de uma transição possível a partir do estado corrente em um dado momento. Por exemplo, se o estado corrente na Máquina da **Figura 2.3** for *Perseguindo um Inimigo*, pode acontecer que tanto a condição $distancia(inimigo) < 30$ quanto a condição $saude < 25\%$ sejam verdadeiras. Nesse caso, pode ser escolhida qualquer uma das opções (aleatoriamente, por exemplo). Apesar dessa possibilidade de aleatoriedade, o comportamento de agentes controlados por Máquinas de Estados Finitos é muito previsível, o que representa uma característica negativa para os agentes de jogos. Essa previsibilidade é muitas vezes percebida pelo jogador, que utiliza sua experiência no jogo para antever as ações dos inimigos e, dessa forma, obter vantagens durante a partida.

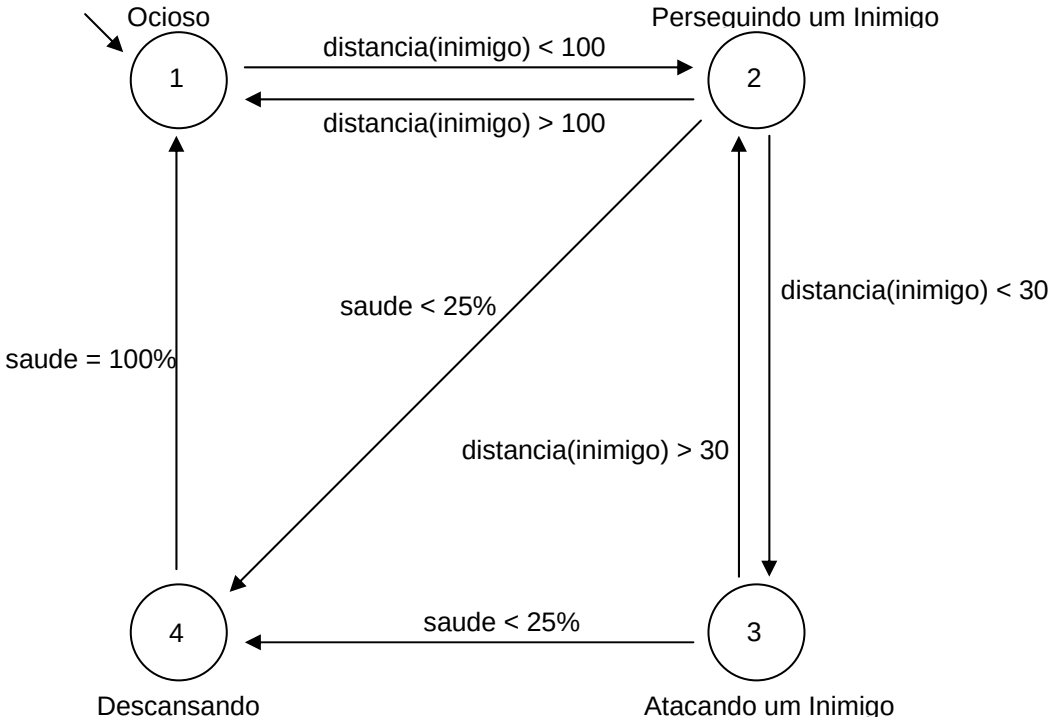

Figura 2.3. Exemplo de Máquina de Estados Finitos com situações típicas de jogos.

Os parâmetros das condições (*distancia(inimigo)*, *saude*) são variáveis e funções cujos valores são calculados na Camada de Tomada de Decisões e disponibilizados à Máquina de Estados como dados de entrada.

2.2.3. Máquinas de Estados *Fuzzy* em Jogos

Máquinas de Estados *Fuzzy* são utilizadas em jogos como uma alternativa às Máquinas de Estados Finitos, quando for desejável que os agentes apresentem comportamento diversificado e menos previsível. A simplicidade das situações possíveis em que o agente pode estar envolvido ainda é necessária, mas o poder de expressão adicional fornecido pelas Máquinas de Estados *Fuzzy* permite que o agente esteja em mais de uma dessas situações em um dado momento. Formalmente, as Máquinas de Estados *Fuzzy* permitem que mais de uma dessas situações pertençam ao conjunto estado corrente.

A **Figura 2.4** apresenta uma adaptação do exemplo da **Figura 2.2** para a utilização de Máquinas de Estados *Fuzzy*. É importante notar que as condições de

transição tornaram-se sentenças da Lógica *Fuzzy*, cujo valor de verdade está contido em um intervalo contínuo. A avaliação dessas condições envolve o cálculo de seu valor de verdade a partir dos dados de entrada.

A cada iteração da Máquina de Estados *Fuzzy*, devem ser avaliadas todas as transições que partem de estados cujo grau de pertinência ao conjunto estado corrente seja não-nulo. Isso significa que, no pior caso, todas as transições da Máquina serão avaliadas. Esse *overhead* de processamento é um dos fatores negativos da adoção de Máquinas de Estados *Fuzzy* em relação às Máquinas de Estados Finitos.

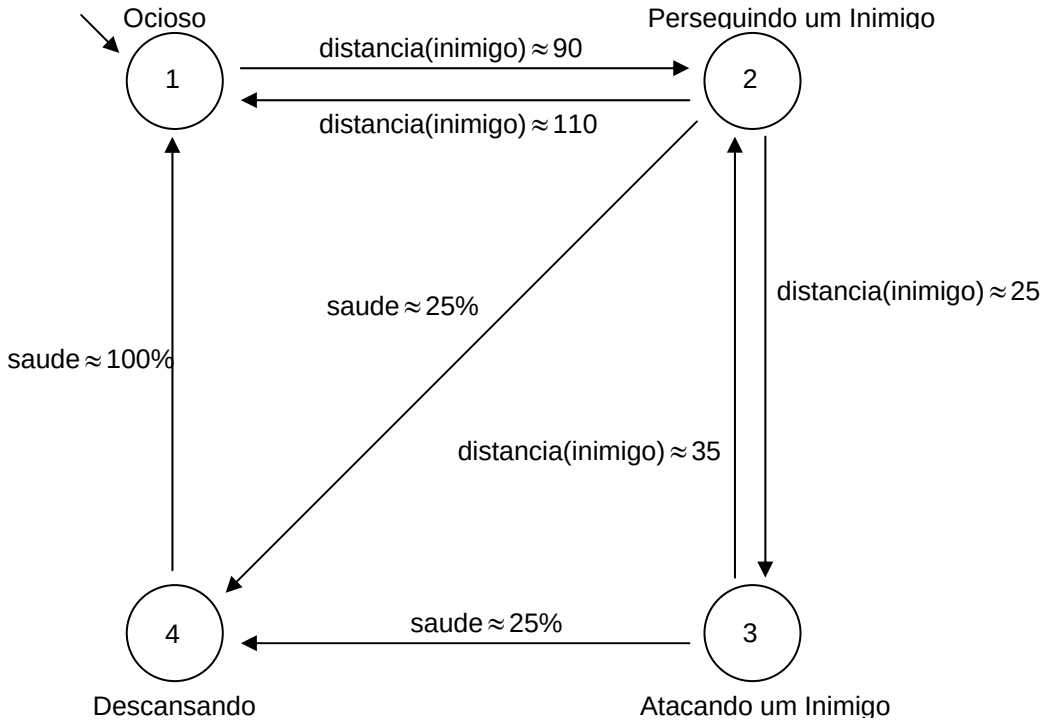

Figura 2.4. Exemplo de Máquina de Estados *Fuzzy* com situações típicas de jogos.

O valor de verdade de uma condição determina a proporção de grau de pertinência ao conjunto estado corrente que será transferida do estado origem da transição para o estado destino, conforme descrito na **Seção 2.1.2**, de forma que o valor somado dos graus de pertinência de todos os estados permaneça constante.

Apesar de o estado corrente representar um conjunto de estados, não é possível mesclar o comportamento representado por mais de um estado para determinar as ações que o agente realizará no jogo. Isso se deve ao fato de que cada estado representa ações bem definidas que devem ser realizadas pelo agente

(como, por exemplo, perseguir um inimigo ou descansar para recuperar saúde), e que se referem à execução de blocos de instruções que implementam as ações desejadas. Não é possível, portanto, misturar instruções correspondentes a mais de uma ação, e por isso é necessário que o agente decida, a cada momento, qual ação será realizada, dentre as ações que correspondem aos estados pertencentes ao conjunto estado corrente.

Uma das soluções para esse problema, utilizada no contexto de jogos com bastante freqüência, é o sorteio ponderado. Nesse caso, o grau de pertinência ao conjunto estado corrente de cada estado é interpretado como uma probabilidade, e um sorteio dirigido é realizado utilizando essas probabilidades. A ação correspondente ao estado sorteado indica as instruções que serão executadas. É necessário definir também um tempo durante o qual o agente permanecerá executando as ações do estado sorteado antes de realizar um novo sorteio, pois, se esse sorteio for realizado a cada *frame* de jogo, o comportamento do agente se tornará predominantemente aleatório.