Relatório final – Auction House simplificada

Guilherme de Oliveira Silva

Introdução

O projeto proposto pretende implementar uma versão simplificada da Auction House (AH) do jogo *World of Warcraft* em Alloy. No jogo, este recurso serve para que jogadores anunciem e vendam itens entre si através de leilões, que funcionam através de lances – ou compra instantânea (*buyout*) – e duram um tempo pré-determinado. Ao fim deste tempo, o jogador com o maior lance vence o leilão e recebe o item; caso não haja lances, o item é devolvido ao dono original.

No jogo, existem complicações a mais: existe uma série de regras para o comércio de categorias de itens, e é possível comprar itens como *commodities* – isto é, sem distinções entre os vendedores. Estes detalhes, entretanto, estão além do escopo proposto. Para este trabalho, pretendemos modelar as entidades e a dinâmica principal da AH: serão criadas representações para **Jogadores**, **Itens** e **Leilões**, e relações que replicam as interações que existem entre elas. Estas entidades, dinâmica e regras implementadas são discutidas a seguir.

Entidades

Item Holder

Esta é a representação básica de qualquer entidade que possua uma coleção de **Item** (*inventory*) e de moedas de ouro (*balance*). As assinaturas de **Player** e **Auction House** estendem esta definição, já que ambos possuirão as mesmas relações na dinâmica implementada.

Item

Representa itens do jogo. Esta *signature* possui apenas duas relações: *owner*, que associa cada **Item** a um **Item Holder**; e *itemStatus*, que indica se o item está à venda ou não (esta relação admite apenas um entre *ForSale* ou *NotForSale*).

Auction

Representa um leilão de itens. Esta *signature* possui, necessariamente, um *auctionStatus* (para controlar o estágio da venda), e pode estar relacionada a um jogador vendedor (*seller*), um item à venda (*forSale*), um jogador que deu o maior lance (*highestBidder*) e/ou um jogador que tenha comprado o item instanteamente (*buyoutBuyer*).

Dinâmica

O sistema é composto pelos *predicates* **init**, responsável pelo estabelecimento de condições iniciais, e **trans**, que abriga os possíveis passos entre estados. As transições permitem a criação de leilões (**createAuction**), e permite que jogadores deem lances (**bidOnAuction**) ou os compre instantaneamente (**buyoutAuction**). Ao final, o sistema lida com leilões que tenham sido vendidos com o *predicate* **endSoldAuction**, e com os leilões não vendidos com **endUnsoldAuction**.

init

As condições iniciais estabelecem

1. Há uma única instância de **Auction House**, que começa com saldo zerado e nenhum item em sua coleção.
2. Todas as instâncias de **Auction** são inicializadas com apenas um status, definido em *NotStarted*. As demais relações são, inicialmente, vazias.
3. Todas as instâncias de **Item** são distribuídas entre os jogadores, de modo que todo **Item** terá um *owner*.
4. Todas as instâncias de **Player** começam com saldo igual a 3.

createAuction

A criação de leilões exige três pré-condições:

1. Não existe outra **Auction** que já esteja vendendo o **Item** em questão.
2. O **Item** pertence ao **Player** que está criando a **Auction**.
3. A instância de **Auction** a ser utilizada não pode estar ativa.
4. O **Player** que está criando a **Auction** deve possuir, ao menos, 1 unidade de ouro para pagar a *taxa de depósito*.

E garante as seguintes pós-condições:

1. O **Item** anunciado é transferido para a **Auction House**, e deixa de pertencer à coleção de itens (*inventory*) do **Player** anunciante.
2. A *taxa de depósito* é paga à **Auction House**, que tem seu saldo incrementado em 1.
3. Os status de **Auction** e **Item** são atualizados conformemente (para *JustStarted* e *ForSale*, respectivamente).
4. A **Auction** é relacionada ao anunciante através do campo *seller*.

bidOnAuction

Para se dar lances em um leilão, exige-se que

1. A **Auction** de interesse esteja ativa.
2. O **Player** comprador não seja o vendedor, e nem dono do atual maior lance.
3. O **Player** comprador tenha pelo menos 1 unidade de ouro para gastar.

Ao fim, estabelece-se que o **Player** se tornará o *highestBidder* do estado seguinte.

buyoutAuction

As pré-condições exigidas são idênticas às de **bidOnAuction**, com exceção de (3): é necessário ter 3 unidades de ouro (o saldo máximo inicial) para gastar. As pós-condições são quase idênticas, porém, com as seguintes adições:

1. O **Player** que compra o **Item** instantaneamente também é nomeado *buyoutBuyer*.
2. A **Auction** é encerrada, e seu status é alterado para *AuctionHasBeenSold*.

endSoldAuction

Este *predicate* gerencia todos os leilões que foram vendidos. Para tal, apenas exige que seu status seja *AuctionHasBeenSold*. Como consequência, estabelece que

1. O status seguinte da **Auction** será *Ended*.
2. Os saldos dos jogadores envolvidos (*highestBidder* e *seller*) são ajustados de acordo com o valor a pagar/receber.

3. O vendedor paga uma comissão à **Auction House** de 1 unidade de ouro.
4. O **Item** vendido será transferido do inventário da **Auction House** para a coleção de itens do jogador *highestBidder*, e terá seu status revertido para *NotForSale*.

endUnsoldAuction

Este *predicate* é a contraparte do anterior, pois é o responsável pelo gerenciamento de **Auctions** terminadas, mas não vendidas – isto é, que não possuem *highestBidder*. Para tal, exige apenas que o status do leilão seja *AuctionHasNotBeenSold* (o que já garante a inexistência de lances). Suas pós-condições são

1. A **Auction** é marcada como encerrada (*Ended*).
2. O **Item** anunciado é devolvido ao vendedor, e seu status é revertido a *NotForSale*.

Condições adicionais

As regras que norteiam o sistema são listadas a seguir.

1. Um **Item Holder** (isto é, válido tanto para **Player** quanto **Auction House**) não pode possuir saldo negativo.
2. Um **Player** deve possuir o **Item** que está vendendo.
3. A relação de propriedade (**Item.***owner*) é reflexiva com a relação de coleção de **Item** (**ItemHolder.***inventory*)
4. Cada **Auction** pode listar um único **Item**.
5. **Auctions** terminadas não podem ser reativadas, isto é, voltar seu status para *Active*.
6. Cada **Item** deve pertencer a somente um **Item Holder** em cada estado.
7. Um **Player** não pode dar um lances (ou comprar instantaneamente) em seu próprio **Auction**.
8. Uma **Auction** acaba após 3 rodadas (*FirstRound*, *SecondRound*, *ThirdRound*). Ao fim, o **Player** com maior lance tornará-se dono do **Item**. Caso não haja lances, o **Item** continuará a pertencer ao dono original. Independentemente do resultado, o status imediatamente posterior será "não está à venda".
9. O *auctionStatus* de todas as **Auctions** é atualizado em todos os estados. Leilões recém iniciados são atualizados para *FirstRound*; *FirstRound* para *SecondRound*; *SecondRound* para *ThirdRound*; e *ThirdRound* para *Ended*.

10. Caso algum **Player** faça *buyout*, a **Auction** acaba instantaneamente.

Em termos de implementação, foram utilizadas pré-condições, *quantifiers* e *facts* para garantir que o sistema siga esta especificação. A seção de asserções, ao fim do código, testa a validade absoluta destas diretrizes.

Conclusões

O modelo captura praticamente todas as relações anteriormente descritas. Entretanto, o sistema de atualização automática do estado dos leilões não funciona corretamente – o que impossibilita que múltiplos lances sejam feitos –, e algumas *frame conditions* (como o saldo de jogadores em algumas execuções do modelo) não se sustentam sempre. Ainda assim, as regras foram corretamente provadas pelos *asserts*, e o sistema funciona como o esperado em leilões comprados instantaneamente.
Existe espaço para trabalhos futuros. Para além da correção do *bug* anteriormente reconhecido, sugere-se que edições futuras

1. Não fixem o valor dos lances ou do *buyout*.
2. Tenham *frame conditions* mais robustas.
3. Armazenem o histórico de lances de cada jogador, de modo que seja possível visualizar o atual lance máximo de cada leilão para além do jogador *highestBidder*.