Formalizando as regras de jogos eletrônicos usando a notação Z

Douglas Fonseca dos Santos
Fatec Carapicuíba
Caverna Digital USP

Alvaro Gabriele Rodrigues
Fatec Carapicuíba

Resumo

Este artigo tem como objetivo demonstrar uma maneira de especificar formalmente as regras de jogos eletrônicos usando a notação Z. Apresenta também que inconsistências nos requisitos podem ser encontradas durante a fase de análise aumentando a confiabilidade da implementação.

Palavras-chave:: Notação Z, Especificação Formal, Regras de Jogos

Contato dos autores:

{douglas.fsantos89,alvaro.gabriele}@gmail.com

1 Introdução

De acordo com [Sus and Staffan 2003], a mecânica do jogo define um, e somente um, tipo de interação que ocorre durante o jogo, as regras específicas de cada jogo definem como implementar essa interação. Atualmente, existem vários trabalhos com o objetivo de identificar e documentar a mecânica dos jogos, no entanto, pouco se fala sobre documentar as regras específicas que serão implementadas.

Um dos métodos utilizados para documentar as regras do jogo é a especificação informal em forma de prosa. A especificação informal tem como vantagem ser de fácil compreensão para o leitor, pois utiliza a linguagem natural. No entanto, a forma como é organizada não permite passar facilmente para uma linguagem de programação. Uma especificação informal pode ser vaga, muitas vezes, omitindo detalhes importantes.

Métodos formais baseam-se em construções matemáticas, permitindo provar que as especificações atendem aos requisitos estabelecidos. A notação Z é uma notação formal para descrever sistemas de computador [Woodcock and Davies 1996].

O objetivo deste trabalho é especificar formalmente as regras do jogo usando a notação Z. Especificar formalmente as regras do jogo permite criar um documento claro e breve. Também é possível calcular inconsistências nos requisitos.

A próxima seção apresenta um trabalho usando um modelo formal, a seção 3 faz uma introdução à linguagem formal Z, a seção 4 apresenta um exemplo de como formalizar as regras para um sistema de equipamentos usando a notação Z e a seção 5 apresenta os resultados obtidos.

2 Trabalhos Relacionados

Um trabalho da Faculdade de Engenharia da Universidade do Porto propõe a elaboração de um jogo usando um modelo formal utilizando a ferramenta *Alloy Analyzer*, eles mostram que a especificação utilizando essa ferramenta produz modelos robustos e confiáveis. Este trabalho pode ser encontrado em [Ribeiro et al. 2010].

3 Notação Z

Especificações formais usam fórmulas matemáticas para dizer qual o comportamento de um sistema [Spivey 1989]. A utilização de métodos formais para especificar sistemas permite a utilização de inferência lógica e prova [Jacky 1996] para ajudar a entender os requisitos do sistema e a encontrar suposições escondidas [Woodcock and Davies 1996]. Métodos formais auxiliam a identificar partes ausentes de especificações incompletas permitindo considerar possibilidades alternativas antes do projeto ser implementado [Bowen 1996].

Z (pronuncia-se *zed*) é uma notação formal desenvolvida pelo Grupo de Pesquisa de Programação (Programming Research Group) na Universidade de Oxford (Oxford University) baseada na teoria de conjuntos e na lógica de predicados de primeira ordem. Além da notação matemática, ela possui uma notação que auxilia a estruturar a especificação chamada notação de esquemas.

3.1 Conceitos básicos da notação Z

A notação Z é uma notação baseada em modelos pois, modela-se um sistema representando seus estados - conjunto de variáveis e seus valores - e as operações que alteram este estado [Jacky 1996]. Para estruturar a descrição do sistema a notação Z utiliza a linguagem de esquemas. O texto do esquema é escrito em uma caixa aberta que possui duas partes separadas por uma linha, a declaração das variáveis que define o conjunto de elementos que a variável pode assumir e o predicado que impõe uma restrição sobre seus valores como no exemplo abaixo.

Variaveis

$$var : \mathbb{N}$$

$$var >= 5$$

O exemplo descreve um esquema nomeado *Variaveis* e declara que a variável *var* poderá assumir qualquer elemento do conjunto dos números naturais, o esquema restringe os valores que *var* pode assumir somente aos elementos que forem maiores do que 5. O predicado de um esquema também é chamado de invariante do sistema.

Para definir as operações que alteram o estado de um esquema é utilizado a linguagem de operadores lógicos de esquemas. Pode-se utilizar os operadores de conjunção e disjunção em esquemas para formar outros esquemas.

Supondo os esquemas S e T como abaixo:

S

$$a : A$$

$$P$$

T

$$b : B$$

$$Q$$

Pode-se escrever um esquema nomeado C como sendo uma conjunção entre os esquemas S e T da seguinte forma.

$$C \mathrel{\widehat=} S \land T$$

O exemplo acima define o seguinte esquema:

C

$$a : A$$
$$b : B$$

$$P \land Q$$

Semelhante ao exemplo acima, é possível escrever uma disjunção entre esquemas como abaixo:

$C \mathrel{\widehat=} S \vee T$

A disjunção define o seguinte esquema:

C
$a : A$
$b : B$

$P \vee Q$

Uma operação pode ser definida declarando um esquema que utiliza o esquema em que a operação será realizada, por exemplo, pode-se definir uma operação em S da seguinte forma:

OpS
ΔS
$i? : A$

R

A letra grega Δ é utilizada para dizer que a operação irá alterar o estado do esquema, ou seja, ela define o seguinte esquema:

ΔS
$a : A$
$a' : A$

A variável *a* é a variável do esquema *S* antes da operação ser completada e a variável *a'* é a variável após a operação ser completada. Os predicados envolvendo a variável *a'* são chamados de pós-condições e os outros predicados são chamados de pré-condições. Quando as pré-condições não são satisfeitas o comportamento da operação é indefinido. Operações com comportamento indefinido são chamadas de operações parciais, e operações que possuem um comportamento definido independente das pré-condições são chamadas de operações totais.

A variável *i?* é uma variável de entrada, a interrogação na frente da variável é uma convenção para variáveis de entrada, assim como uma exclamação na frente de uma variável é uma convenção para variáveis de saída.

Quando é necessário definir uma operação que não altera o estado do esquema utiliza-se a letra grega Ξ da seguinte forma:

OpS
ΞS

R

Para definir um novo tipo usa-se o nome do tipo entre colchetes como no exemplo abaixo.

$[TIPO]$

É possível definir uma abreviação para elementos de um determinado tipo, uma abreviação é um outro nome para um tipo já existente, uma abreviação não inclui novos símbolos. Podemos definir uma abreviação da seguinte forma:

$ABBR == \{t : TIPO \mid P\}$

O predicado P na abreviação acima define as propriedades dos elementos que pertencem a esse conjunto.

Quando é necessário definir restrições sobre os elementos de um determinado tipo usa-se uma definição axiomática. Assim como esquemas a definição axiomática possui duas partes, uma declaração e um predicado. Pode-se escrever uma definição axiomática da seguinte forma:

$a : A$

$\forall a : A \bullet a \neq \emptyset$

Esta definição define um tipo a que contém todos os elementos do tipo A exceto o elemento nulo. Em uma definição axiomática todo elemento do tipo deve satisfazer ao predicado.

Para relacionar elementos de um determinado tipo com elementos de outro tipo usa-se funções, por exemplo, pode-se definir uma função nomeada ab para mapear os elementos do tipo A com elementos do tipo B usando uma definição axiomática da seguinte forma.

$ab : A \rightarrow B$

Essa função diz que todo elemento em A possui um correspondente em B, por isso ela é chamada de função total.

Esses são alguns dos elementos básicos da notação Z.

4 Formalizando as regras do jogo

A notação Z é utilizada para especificar sistemas computacionais. Considerando o jogo como possuindo estados e as ações do jogador como operações que alteram esse estado pode-se especificar as regras do jogo utilizando a notação Z.

Considere o exemplo de um sistema de equipamentos para um jogo de rpg que pode ser especificado informalmente da seguinte forma:

- O jogador poderá equipar armaduras no corpo;
- O jogador poderá equipar espadas, escudos ou arcos nas mãos;
- O jogador não poderá ter equipado um arco e outra arma ao mesmo tempo;

A especificação informal não faz nenhuma referência ao que deve acontecer caso o jogador tente equipar um arco em uma mão e tenha uma arma na outra mão, ela somente diz que esta situação não deve ocrorrer. A notação Z auxilia a encontrar e especificar esses requisitos.

Essa especificação pode ser escrita em Z da seguinte forma:

$[EQUIP]$
$TIPO ::= ESPADA \mid ESCUDO \mid ARCO \mid ARMADURA$

$tipo : EQUIP \rightarrow TIPO$

$armadura == \{a : EQUIP \mid a = \emptyset \vee tipo\ a = ARMADURA\}$
$arma == \{a : EQUIP \mid a = \emptyset \vee a \notin armadura\}$

A primeira linha diz que existe um tipo nomeado EQUIP, que contém todos os equipamentos que podem ser utilizados no jogo, a segunda linha define um conjunto que possui apenas os elementos ESPADA, ESCUDO, ARCO e ARMADURA, a terceira linha define uma função que mapeia cada elemento do tipo EQUIP com um elemento do tipo TIPO, a quarta e quinta linhas definem dois subconjuntos do conjunto EQUIP, um para armaduras e outro para armas respectivamente.

O jogador pode ser definido usando um esquema da seguinte forma.

$Jogador$
$corpo : armadura$
$mao\_d, mao\_e : arma$

Este esquema diz que o jogador possui três variáveis, duas do tipo arma e uma do tipo armadura.

Agora á necessário definir as operações que alteram o estado do sistema. Uma operação para equipar uma arma na mão direita pode ser definida da seguinta forma.

EquiparArmaDireita

$\Delta Jogador$
$a? : arma$

$tipo\ a? = ARCO \Rightarrow mao\_e = \emptyset$
$mao\_d' = a?$
$mao\_e' = mao\_e$
$corpo' = corpo$

A primeira linha das pré-condições mostra exatamente o que a especificação informal estava dizendo, ou seja, ao equipar uma arma na mão direita e o tipo da arma for um arco é necessário que a mão esquerda esteja vazia. Pode-se notar que a operação tem uma pré-condição implícita, pois a? precisa ser um elemento do conjunto arma - lembrando que o conjunto arma é uma abreviação do conjunto EQUIP, portanto, todo elemento de arma é um elemento do tipo EQUIP.

Existem momentos em que as pré-condições da operação EquiparArmaDireita não são satisfeitas, portanto, ela é uma operação parcial, precisa-se de outras operações para tornar a operação EquiparArmaDireita uma operação total.

Para isso é necessário definir uma outra operação.

EquiparArco

$\Delta Jogador$
$a? : arma$

$tipo\ a? = ARCO$
$mao\_d' = a?$
$mao\_e' = \emptyset$
$corpo' = corpo$

A operação total para equipar uma arma na mão direita pode ser definida usando uma disjunção entre os esquemas EquiparArmaDireita e EquiparArco.

$$TEquiparArmaDireita = EquiparArmaDireita \lor EquiparArco$$

Usando as regras de inferência lógica pode-se calcular se as pré-condições serão sempre satisfeitas, como mostrado no exemplo abaixo, para faciltar a visualização será utilizado letra nos lugares dos predicados, e letra V será usada no lugar de verdadeiro:

$p = tipo\ a? = ARCO$
$q = mao\_e = \emptyset$
$r = a? \in arma$

$((p \Rightarrow q) \land r) \lor (p \land r)$
$r \land ((p \Rightarrow q) \lor p)$
$r \land ((\neg\, p \lor q) \lor p)$
$r \land ((\neg\, p \lor p) \lor q)$
$r \land (V \lor q)$
$r \land V$
r

Desta forma é possível ver que ainda falta o predicado de que a entrada precisa pertencer ao conjunto arma. Então definimos uma operação para tratar esse caso.

ArmaInvalida

$\Xi Jogador$

$a? \notin arma$

Agora é possível definir a função total TEquiparArmaDireita da seguinte forma:

$$TEquiparArmaDireita \mathrel{\widehat{=}} EquiparArmaDireita \lor EuiparArco \lor ArmaInvalida$$

Calculando as pré-condições, conforme descrito acima, obtemos:

$((p \Rightarrow q) \land r) \lor (p \land r) \lor \neg\, r$
$(r \land ((p \Rightarrow q) \lor p)) \lor \neg\, r$
$(r \land ((\neg\, p \lor q) \lor p)) \lor \neg\, r$
$(r \land ((\neg\, p \lor p) \lor q)) \lor \neg\, r$
$(r \land (V \lor q)) \lor \neg\, r$
$(r \land V) \lor \neg\, r$
$r \lor \neg\, r$
V

Portanto, pode-se concluir que a função TEquiparArmaDireita é uma função total.

É possível definir uma função para equipar armas na mão esquerda definindo um esquema com os predicados relativos à mão esquerda, e reutilizando os esquemas utilizados na operação para equipar uma arma na mão direita, como no exemplo abaixo:

$$TEquiparArmaEsquerda \mathrel{\widehat{=}} EquiparArmaEsquerda \lor EquiparArco \lor ArmaInvalida$$

5 Conclusão

Este artigo apresentou uma forma de utilizar a notação formal Z para especificar as regras de jogos e como encontrar inconsistências nos requisitos antes de implementar o sistema.

A documentação gerada usando uma notação formal é breve, e informa de modo preciso o comportamento do sistema, podendo ser usado como um meio de comunicação entre os programadores e o desenvolvedor do jogo. No entanto, a documentação formal não auxilia a definir as regras do jogo, é necessário que as regras sejam conhecidas antes de poder fazer uma especificação formal útil.

A notação formal permite que as regras do jogo sejam definidas de forma que possam facilmente serem implementadas em uma linguagem de programação, no entanto, tem pouca utilidade como forma de comunicação com os usuários, como um manual por exemplo, a notação formal não deve ser usada como um substituto para a especificação informal, mas como um complemento.

Agradecimentos

Agradeço ao núcleo de Realidade Virtual Caverna Digital do Laboratório de Sistemas Integráveis da Universidade de São Paulo pelo apoio e ao Rodrigo pelas dicas que foram de grande ajuda para a conclusão deste trabalho.

Referências

BOWEN, J. 1996. *Formal Specification and Documentation Using Z: A Case Study Approach*. International Thomson Computer Press, Feb.

JACKY, J. 1996. *The way of Z: practical programming with formal methods*. Cambridge University Press, New York, NY, USA.

RIBEIRO, J. P. M., PINTO, P. A. T., AND CAMPOS, R. R. C., 2010. Modelo formal em alloy masmorras e dragões. `http://paginas.fe.up.pt/~ei06029/paginapessoal/files/MasmorrasDragoes.pdf`, Abril.

SPIVEY, J. M. 1989. *The Z notation: a reference manual*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

SUS, L., AND STAFFAN, B. 2003. Game mechanics: Describing Computer-Augmented games in terms of interaction.

WOODCOCK, J., AND DAVIES, J. 1996. *Using Z: specification, refinement, and proof*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.