Especificação Formal (Z notation) — Jogo de Reciclagem

---

1. Introdução

Objetivo: fornecer uma especificação formal em Z (estilo clássico: conjuntos, esquemas de estado, operações) que modele o núcleo do jogo de reciclagem: fases, itens de lixo, lixeiras, ações de colocar item em lixeira, feedback e pontuação.

Escopo: foco na lógica de jogo e nas propriedades que desejamos provar (por exemplo: correção do feedback, preservação de invariantes, ausência de deadlocks simples).

---

2. Tipos básicos e conjuntos

Usamos conjuntos abstratos (given sets) para representar entidades do domínio.

```
[ITEM, BIN, LEVEL, PLAYER]
```

- `ITEM` — identificador abstrato para cada item de lixo.
- `BIN` — identificador abstrato para cada lixeira (categoria: plÃ¡stico, vidro, orgÃ¢nico, etc.).
- `LEVEL` — identificador de fase/nível do jogo.
- `PLAYER` — identificador de jogador (pode ser singleton se o jogo for single-player).

Predicados/funcções auxiliares (total/partial):

```
Category : ITEM ↔ BIN              -- categoria correta de cada item
ItemsInLevel : LEVEL ↔ ℙ ITEM      -- itens pertencentes a cada nível
TimeLimit : LEVEL ↔ ℕ              -- tempo limite por fase
```

`Category` associa cada `ITEM` à `BIN` correta. `ItemsInLevel` diz quais itens compõem cada fase. TimeLimit Liga cada **nível** a um número natural (`ℕ`), que representa o tempo máximo da fase.

---

3. Esquema de estado

Definimos o estado global do jogo: items ainda não reciclados, pontuações, estado da fase, e estado do jogo.

```
GameState
    itemsRemaining : LEVEL ↔ ℙ ITEM
    placed         : ITEM ⇸ BIN
```

```
    score          : PLAYER ↔ ℤ
    feedback       : ITEM ↔ {ok, erro}
    currentLevel   : LEVEL ↔ {running, finished, not_started}
    timer          : LEVEL ↔ ℕ
    player         : PLAYER
---------------------------------
    dom itemsRemaining = dom currentLevel
    ∀ lv : dom itemsRemaining • itemsRemaining lv ⊆ ItemsInLevel lv
    dom timer = dom currentLevel
```

Explicação das componentes principais

- `currentLevel` guarda o estado (running/finished/not_started) de cada nível. (Em um jogo simples pode-se ter só um nível corrente; aqui permitimos um mapeamento.)
- `itemsRemaining` mapeia para os itens ainda por classificar em cada nível.
- `placed` registra onde cada item foi colocado (se já colocado). É uma função parcial de `ITEM` para `BIN`.
- `score` mantém a pontuação por jogador.

Invariantes importantes (dentro do esquema):

- O domínio de `itemsRemaining` coincide com os níveis conhecidos em `currentLevel`.
- `itemsRemaining lv` é subconjunto de `ItemsInLevel lv` (não posso ter itens que não pertençam ao nível).

---

4. Esquemas de inicialização

Inicializamos o sistema com nenhum nível iniciado, todas as pontuações zeradas e `itemsRemaining` iguais aos itens do nível quando o nível for iniciado.

```
InitGameState
    GameState'
---------------------------------
    itemsRemaining' = ∅
    placed' = ∅
    feedback' = ∅
    score' = { player ↦ 0 }
    currentLevel' = ∅
    timer' = ∅
    player' = player
```

4.1 Função auxiliar para pontuação:

```
Points : ITEM × BIN → ℤ
∀ i: ITEM; b: BIN •
```

```
    (Category i = b ⇒ Points(i,b) = 1) ∧
    (Category i ≠ b ⇒ Points(i,b) = -1)
```

A seguir, operações principais modeladas com pré/post-condições.

5.1 StartLevel — inicia um nível

```
StartLevel
    ΔGameState
    lv? : LEVEL
--------------------------------
    lv? ∉ dom currentLevel
    currentLevel' = currentLevel ∪ { lv? ↦ running }
    itemsRemaining' = itemsRemaining ∪ { lv? ↦ ItemsInLevel lv? }
    timer' = timer ∪ { lv? ↦ TimeLimit lv? }
    placed' = placed
    feedback' = feedback
    score' = score
```

Pré-condições:

O nível ainda não foi iniciado (lv? ∉ dom currentLevel).

Pós-condições:

O nível é marcado como “rodando”.

Os itens desse nível são carregados em itemsRemaining.

O tempo limite é definido.

Pontuação e feedback permanecem como estavam.

5.2 PlaceItem — jogador coloca um item numa lixeira

```
PlaceItem
    ΔGameState
    lv? : LEVEL
    i?  : ITEM
    b?  : BIN
--------------------------------
    lv? ∈ dom currentLevel
    currentLevel lv? = running
    i? ∈ itemsRemaining lv?

    placed' = placed ∪ { i? ↦ b? }
    itemsRemaining' = itemsRemaining ⊕ { lv? ↦ (itemsRemaining lv? \
{ i? }) }
```

```
    feedback' = feedback ∪ ∀ i: ITEM; b: BIN •
    (Category i = b ⇒ feedback' = feedback ∪ { i ↦ ok }) ∧
    (Category i ≠ b ⇒ feedback' = feedback ∪ { i ↦ erro })
    score' = score ⊕ { player ↦ (score player + Points(i?, b?)) }
    currentLevel' = currentLevel
    timer' = timer
```

Pré-condições

O nível está em execução (currentLevel lv? = running).

O item ainda não foi reciclado (i? ∈ itemsRemaining lv?).

Pós-condições

O item i? é registrado na lixeira b?.

Esse item some da lista de itens restantes.

O sistema gera um feedback (ok se acertou, erro se errou).

A pontuação aumenta em +1 se o lixo foi colocado certo, ou diminui em –1 se colocado errado.

5.3 FinishLevel — marca nível como terminado quando não há itens restantes

```
FinishLevel
    ΔGameState
    lv? : LEVEL
---------------------------------
    lv? ∈ dom currentLevel
    (itemsRemaining lv? = ∅ ∨ timer lv? = 0)

    currentLevel' = currentLevel ⊕ { lv? ↦ finished }
    itemsRemaining' = itemsRemaining
    placed' = placed
    score' = score
    feedback' = feedback
    timer' = timer
```

Pré-condições

O nível está ativo (lv? ∈ dom currentLevel).

Esse nível não tem mais lixos restantes ou o tempo acabou.

Pós-condições

O nível passa a ser marcado como “finalizado”.

O estado do jogador, pontuação e feedback não mudam.

5.4 NextLevel — quando o nível é marcado como concluido passamos para o proximo.

```
NextLevel
    ΔGameState
    lv? : LEVEL
    lv_next? : LEVEL
--------------------------------
    lv? ∈ dom currentLevel
    currentLevel lv? = finished
    lv_next? ∉ dom currentLevel

    currentLevel' = currentLevel ∪ { lv_next? ↦ running }
    itemsRemaining' = itemsRemaining ∪ { lv_next? ↦ ItemsInLevel lv_next? }
    timer' = timer ∪ { lv_next? ↦ TimeLimit lv_next? }
    placed' = placed
    feedback' = feedback
    score' = score
```