# Análise de Planejamento - Mundo dos Blocos

Grupo: Rafael Farias, Ana Flávia, Hanna Mesquita, Francisco Brilhante, Micael Gerrar, Erik Oliveira, Pedro Forte, Davi Emanuel

1. **s_inicial=S0 até um dos estados s_f1, s_f2, s_f3 e s_f4**
   **Estado inicial = S0**
   **Estado Final = s_f3**

```
MODULE main
VAR
   -- Posição do canto esquerdo de cada bloco
   pos_a: 0..7;
   pos_b: 0..7;
   pos_c: 0..7;
   pos_d: 0..7;

   -- Em que cada bloco está apoiado (table ou outro bloco)
   on_a: {table, a, b, c, d};
   on_b: {table, a, b, c, d};
   on_c: {table, a, b, c, d};
   on_d: {table, a, b, c, d};

DEFINE
   -- Tamanhos dos blocos
   size_a := 1;
   size_b := 1;
   size_c := 2;
   size_d := 3;

   -- Macro para verificar sobreposição entre dois blocos
   overlap(p1, s1, p2, s2) := (p1 <= p2 + s2 - 1) & (p2 <= p1 + s1 - 1);

   -- Definição de bloco livre (nada sobre ele)
   is_clear_a := on_b != a & on_c != a & on_d != a;
   is_clear_b := on_a != b & on_c != b & on_d != b;
   is_clear_c := on_a != c & on_b != c & on_d != c;
   is_clear_d := on_a != d & on_b != d & on_c != d;

   -- Ação de mover um bloco (X) para um destino (Y ou mesa)
   -- A lógica verifica se X está livre, se o destino está livre (se for um bloco)
   -- e se a nova configuração não causa colisões.
     move_block(mover, pos_mover, on_mover, size_mover, p_a, o_a, s_a, p_b, o_b, s_b,
p_c, o_c, s_c, p_d, o_d, s_d) :=
      case
         -- Mover para a mesa
         next(on_mover) = table &
            -- Sem colisão com outros blocos na mesa
            !(o_a = table & overlap(next(pos_mover), size_mover, p_a, s_a)) &
```

```
            !(o_b = table & overlap(next(pos_mover), size_mover, p_b, s_b)) &
            !(o_c = table & overlap(next(pos_mover), size_mover, p_c, s_c)) &
            !(o_d = table & overlap(next(pos_mover), size_mover, p_d, s_d)) : 1;

          -- Mover para o bloco 'a'
          next(on_mover) = a & next(pos_mover) = p_a & is_clear_a : 1;
          -- Mover para o bloco 'b'
          next(on_mover) = b & next(pos_mover) = p_b & is_clear_b : 1;
          -- Mover para o bloco 'c'
          next(on_mover) = c & next(pos_mover) = p_c & is_clear_c : 1;
          -- Mover para o bloco 'd'
          next(on_mover) = d & next(pos_mover) = p_d & is_clear_d : 1;
          TRUE: 0;
       esac;

   -- Definição da transição para cada bloco
   move_a := is_clear_a &
                move_block(a, pos_a, on_a, size_a, pos_a, on_a, size_a, pos_b, on_b, size_b,
pos_c, on_c, size_c, pos_d, on_d, size_d) &
            next(pos_b) = pos_b & next(on_b) = on_b &
            next(pos_c) = pos_c & next(on_c) = on_c &
            next(pos_d) = pos_d & next(on_d) = on_d;

   move_b := is_clear_b &
                move_block(b, pos_b, on_b, size_b, pos_a, on_a, size_a, pos_b, on_b, size_b,
pos_c, on_c, size_c, pos_d, on_d, size_d) &
            next(pos_a) = pos_a & next(on_a) = on_a &
            next(pos_c) = pos_c & next(on_c) = on_c &
            next(pos_d) = pos_d & next(on_d) = on_d;

   move_c := is_clear_c &
                 move_block(c, pos_c, on_c, size_c, pos_a, on_a, size_a, pos_b, on_b, size_b,
pos_c, on_c, size_c, pos_d, on_d, size_d) &
            next(pos_a) = pos_a & next(on_a) = on_a &
            next(pos_b) = pos_b & next(on_b) = on_b &
            next(pos_d) = pos_d & next(on_d) = on_d;

   move_d := is_clear_d &
                move_block(d, pos_d, on_d, size_d, pos_a, on_a, size_a, pos_b, on_b, size_b,
pos_c, on_c, size_c, pos_d, on_d, size_d) &
            next(pos_a) = pos_a & next(on_a) = on_a &
            next(pos_b) = pos_b & next(on_b) = on_b &
            next(pos_c) = pos_c & next(on_c) = on_c;

   -- Estado final S_f3: d sobre c+a, c na pos 0, a na pos 2, b na pos 5
    GOAL := on_d = c & pos_d = 0 & on_c = table & pos_c = 0 & on_a = table & pos_a = 2 &
on_b = table & pos_b = 5;
```

```
-- ESTADO INICIAL (S0)
INIT
   on_a = table & pos_a = 3 &
   on_b = table & pos_b = 5 &
   on_c = table & pos_c = 0 &
   on_d = a & pos_d = 3;

-- REGRAS DE TRANSIÇÃO
TRANS
   move_a | move_b | move_c | move_d;

-- ESPECIFICAÇÃO (encontrar um caminho para o GOAL)
SPEC EF(GOAL)
```

2. **s_inicial=S0 até S5 .**
   **Estado Inicial = S0**
   **Estado Final = S5**

```
MODULE main
VAR
   -- (O mesmo VAR do modelo anterior)
   pos_a: 0..7; pos_b: 0..7; pos_c: 0..7; pos_d: 0..7;
   on_a: {table, a, b, c, d}; on_b: {table, a, b, c, d};
   on_c: {table, a, b, c, d}; on_d: {table, a, b, c, d};
DEFINE
   -- (As mesmas definições do modelo anterior)
   size_a := 1; size_b := 1; size_c := 2; size_d := 3;
   overlap(p1, s1, p2, s2) := (p1 <= p2 + s2 - 1) & (p2 <= p1 + s1 - 1);
   is_clear_a := on_b != a & on_c != a & on_d != a;
   is_clear_b := on_a != b & on_c != b & on_d != b;
   is_clear_c := on_a != c & on_b != c & on_d != c;
   is_clear_d := on_a != d & on_b != d & on_c != d;
    move_block(mover, pos_mover, on_mover, size_mover, p_a, o_a, s_a, p_b, o_b, s_b,
p_c, o_c, s_c, p_d, o_d, s_d) := case next(on_mover) = table & !(o_a = table &
overlap(next(pos_mover), size_mover, p_a, s_a)) & !(o_b = table &
overlap(next(pos_mover), size_mover, p_b, s_b)) & !(o_c = table &
overlap(next(pos_mover), size_mover, p_c, s_c)) & !(o_d = table &
overlap(next(pos_mover), size_mover, p_d, s_d)) : 1; next(on_mover) = a &
next(pos_mover) = p_a & is_clear_a : 1; next(on_mover) = b & next(pos_mover) = p_b &
is_clear_b : 1; next(on_mover) = c & next(pos_mover) = p_c & is_clear_c : 1;
next(on_mover) = d & next(pos_mover) = p_d & is_clear_d : 1; TRUE: 0; esac;

   move_a := is_clear_a & move_block(a, pos_a, on_a, size_a, pos_a, on_a, size_a, pos_b,
on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_b) = pos_b &
next(on_b) = on_b & next(pos_c) = pos_c & next(on_c) = on_c & next(pos_d) = pos_d &
next(on_d) = on_d;
```

```
    move_b := is_clear_b & move_block(b, pos_b, on_b, size_b, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_c) = pos_c & next(on_c) = on_c & next(pos_d) = pos_d & next(on_d) = on_d;
    move_c := is_clear_c & move_block(c, pos_c, on_c, size_c, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_d) = pos_d & next(on_d) = on_d;

    move_d := is_clear_d & move_block(d, pos_d, on_d, size_d, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_c) = pos_c & next(on_c) = on_c;

   -- Estado final S5: torre a,b sobre c sobre d, com d na pos 2
   GOAL := on_a = c & on_b = c & on_c = d & on_d = table & pos_d = 2;

INIT
   -- (O mesmo INIT do modelo anterior)
    on_a = table & pos_a = 3 & on_b = table & pos_b = 5 & on_c = table & pos_c = 0 & on_d = a & pos_d = 3;
TRANS
   -- (A mesma TRANS do modelo anterior)
   move_a | move_b | move_c | move_d;

SPEC EF(GOAL)
```

3. **s_inicial = S0 até S7.**
   **Estado Inicial = S0**
   **Estado Final = S7**

```
MODULE main
VAR
   -- (O mesmo VAR do modelo anterior)
   pos_a: 0..7; pos_b: 0..7; pos_c: 0..7; pos_d: 0..7;
   on_a: {table, a, b, c, d}; on_b: {table, a, b, c, d};
   on_c: {table, a, b, c, d}; on_d: {table, a, b, c, d};
DEFINE
   -- (As mesmas definições do modelo anterior)
   size_a := 1; size_b := 1; size_c := 2; size_d := 3;
   overlap(p1, s1, p2, s2) := (p1 <= p2 + s2 - 1) & (p2 <= p1 + s1 - 1);
   is_clear_a := on_b != a & on_c != a & on_d != a;
   is_clear_b := on_a != b & on_c != b & on_d != b;
   is_clear_c := on_a != c & on_b != c & on_d != c;
   is_clear_d := on_a != d & on_b != d & on_c != d;
     move_block(mover, pos_mover, on_mover, size_mover, p_a, o_a, s_a, p_b, o_b, s_b, p_c, o_c, s_c, p_d, o_d, s_d) := case next(on_mover) = table & !(o_a = table & overlap(next(pos_mover), size_mover, p_a, s_a)) & !(o_b = table &
```

overlap(next(pos_mover), size_mover, p_b, s_b)) & !(o_c = table & overlap(next(pos_mover), size_mover, p_c, s_c)) & !(o_d = table & overlap(next(pos_mover), size_mover, p_d, s_d)) : 1; next(on_mover) = a & next(pos_mover) = p_a & is_clear_a : 1; next(on_mover) = b & next(pos_mover) = p_b & is_clear_b : 1; next(on_mover) = c & next(pos_mover) = p_c & is_clear_c : 1; next(on_mover) = d & next(pos_mover) = p_d & is_clear_d : 1; TRUE: 0; esac;

move_a := is_clear_a & move_block(a, pos_a, on_a, size_a, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_c) = pos_c & next(on_c) = on_c & next(pos_d) = pos_d & next(on_d) = on_d;

move_b := is_clear_b & move_block(b, pos_b, on_b, size_b, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_c) = pos_c & next(on_c) = on_c & next(pos_d) = pos_d & next(on_d) = on_d;

move_c := is_clear_c & move_block(c, pos_c, on_c, size_c, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_d) = pos_d & next(on_d) = on_d;

move_d := is_clear_d & move_block(d, pos_d, on_d, size_d, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_c) = pos_c & next(on_c) = on_c;

-- Estado final S7: a,b sobre c na pos 0, e d na mesa na pos 3
GOAL := on_a = c & on_b = c & on_c = table & pos_c = 0 & on_d = table & pos_d = 3;

INIT
-- (O mesmo INIT do modelo anterior)
on_a = table & pos_a = 3 & on_b = table & pos_b = 5 & on_c = table & pos_c = 0 & on_d = a & pos_d = 3;
TRANS
-- (A mesma TRANS do modelo anterior)
move_a | move_b | move_c | move_d;

SPEC EF(GOAL)

4. Gere planos negando os estados objetivo com propriedades lógicas não realizáveis.

MODULE main
VAR
-- (O mesmo VAR dos modelos anteriores)
pos_a: 0..7; pos_b: 0..7; pos_c: 0..7; pos_d: 0..7;
on_a: {table, a, b, c, d}; on_b: {table, a, b, c, d};
on_c: {table, a, b, c, d}; on_d: {table, a, b, c, d};

```
DEFINE
   -- (As mesmas definições dos modelos anteriores)
   size_a := 1; size_b := 1; size_c := 2; size_d := 3;
   overlap(p1, s1, p2, s2) := (p1 <= p2 + s2 - 1) & (p2 <= p1 + s1 - 1);
   is_clear_a := on_b != a & on_c != a & on_d != a;
   is_clear_b := on_a != b & on_c != b & on_d != b;
   is_clear_c := on_a != c & on_b != c & on_d != c;
   is_clear_d := on_a != d & on_b != d & on_c != d;
     move_block(mover, pos_mover, on_mover, size_mover, p_a, o_a, s_a, p_b, o_b, s_b, p_c, o_c, s_c, p_d, o_d, s_d) := case next(on_mover) = table & !(o_a = table & overlap(next(pos_mover), size_mover, p_a, s_a)) & !(o_b = table & overlap(next(pos_mover), size_mover, p_b, s_b)) & !(o_c = table & overlap(next(pos_mover), size_mover, p_c, s_c)) & !(o_d = table & overlap(next(pos_mover), size_mover, p_d, s_d)) : 1; next(on_mover) = a & next(pos_mover) = p_a & is_clear_a : 1; next(on_mover) = b & next(pos_mover) = p_b & is_clear_b : 1; next(on_mover) = c & next(pos_mover) = p_c & is_clear_c : 1; next(on_mover) = d & next(pos_mover) = p_d & is_clear_d : 1; TRUE: 0; esac;

    move_a := is_clear_a & move_block(a, pos_a, on_a, size_a, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_c) = pos_c & next(on_c) = on_c & next(pos_d) = pos_d & next(on_d) = on_d;

    move_b := is_clear_b & move_block(b, pos_b, on_b, size_b, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_c) = pos_c & next(on_c) = on_c & next(pos_d) = pos_d & next(on_d) = on_d;

    move_c := is_clear_c & move_block(c, pos_c, on_c, size_c, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_d) = pos_d & next(on_d) = on_d;

    move_d := is_clear_d & move_block(d, pos_d, on_d, size_d, pos_a, on_a, size_a, pos_b, on_b, size_b, pos_c, on_c, size_c, pos_d, on_d, size_d) & next(pos_a) = pos_a & next(on_a) = on_a & next(pos_b) = pos_b & next(on_b) = on_b & next(pos_c) = pos_c & next(on_c) = on_c;

   -- GOAL NÃO REALIZÁVEL: Bloco 'c' e 'a' colidindo na mesa.
   GOAL := on_c = table & pos_c = 0 & on_a = table & pos_a = 1;

INIT
   on_a = table & pos_a = 3 &
   on_b = table & pos_b = 5 &
   on_c = table & pos_c = 0 &
   on_d = a & pos_d = 3;

TRANS
```

```
    move_a | move_b | move_c | move_d;

-- A especificação agora deve ser falsa.
SPEC EF(GOAL)
```