# Planos formais - Mundo dos Blocos (Situações 1, 2 e 3)

Autor: Guilherme (conversação com ChatGPT). Documento gerado automaticamente.

**Situação 1: S0 -> Sf1 (Formalização)**

Dados e tamanhos: size(c)=2, size(a)=1, size(b)=1, size(d)=3. Mesa: slots 0..5 (slots inteiros).

Estado inicial S0 (interpretado):

```
pos(c, table(0)) % c occupies slots 0-1
pos(a, table(3)) % a at slot 3
pos(b, table(5)) % b at slot 5
pos(d, spanning(3,5), on(a and b)) % d supported by a and b (occupies 3-5)
```

Meta Sf1 (interpretação fornecida):

```
pos(d, table(3)) % d on table slots 3-5
pos(a, on(d), at_relative_slot=4) % a on d covering slot 4
pos(b, on(d), at_relative_slot=5) % b on d covering slot 5
pos(c, table(4)) % c occupying slots 4-5, supported by a and b
```

Plano (interpretação do roteiro fornecido). Observações sobre relaxações necessárias:

```
Plan = [
  move(d, on(c)) % relax size-check: 3 <= 2 (violação estrita)
  move(a, on(d)) % OK: size(a)=1 <= size(d)=3
  move(b, on(d)) % OK, assumes atomic/compound move with a or immediate sequence
  move(d, table(3)) % moving stack d[a,b] atomically to table slots 3..5
  move(c, table(4)) % c occupies slots 4..5 supported by a(slot4) and b(slot5)
]
```

Para que esse plano seja válido sem relaxações, é necessário permitir movimentos compostos (move_stack) ou disponibilizar slots livres adicionais na mesa.

## Situação 2: S0 -> S5 (Formalização)

Tamanhos: size(c)=2, size(a)=1, size(b)=1, size(d)=3. Mesa: slots 0..5.
Estados fornecidos:

S0:
pos(c, table(0)) % c occupies slots 0-1
pos(a, on(c), at_slot0) % a supported on c's left cell
pos(b, on(c), at_slot1) % b supported on c's right cell
pos(d, table(3)) % d occupies 3-5 on the table

S1:
pos(b, table(2)) % b moved to table slot 2

S2:
pos(a, on(b)) % a stacked onto b

S3:
pos(c, on(d)) at table-relative slots 4..5 % c placed covering d slots 4-5

S4:
pos(a, on(c), at_slot4) % a on c

S5:
pos(b, on(c), at_slot5) % b on c

Plano sugerido (ações mínimas entre estados):

1) move(b, table(2))
Preconditions: clear(b) (true if nothing above b), table_slot(2), is_free(2).
Effects: add pos(b,table(2)); delete pos(b,on(c)); add clear(c) if no other support.

2) move(a, on(b))
Preconditions: clear(a), clear(b), size(a)=1 <= size(b)=1.
Effects: add pos(a,on(b)); delete pos(a,on(c)); clear(b) false.

3) move(c, on(d)) % place c onto d covering relative slots 4..5
Preconditions: clear(c), clear(d), size(c)=2 <= size(d)=3, and d located so c fits on its surface (d occupies 3..5).
Effects: add pos(c,on(d)); delete pos(c,table(0)); clear(d) false.

4) move(a, on(c)) % a onto c (slot4)
Preconditions: clear(a), clear(c), size(a)=1 <= size(c)=2.
Effects: add pos(a,on(c)); delete previous pos(a,on(b) or table).

5) move(b, on(c)) % b onto c (slot5)
Preconditions: clear(b), clear(c) (as needed), size(b)=1 <= size(c)=2.
Effects: add pos(b,on(c)); delete previous pos(b,table(2)).

Observações:

- Este plano é válido sob as regras de tamanho e clear, considerando que c inicialmente suporta a e b em partes distintas (slots 0 and 1).
- A ação move(c,on(d)) assume que d está em mesa slots 3..5 e tem topo livre para receber c; size-check ok (2 <= 3).

## Situação 3: S0 -> S7 (Formalização)

Tamanhos: size(c)=2, size(a)=1, size(b)=1, size(d)=3. Mesa: slots 0..5.
Estados fornecidos:

S0:

pos(c, table(0)) % c occupies slots 0-1
pos(a, table(3)) % a at slot 3 (table)
pos(b, table(5)) % b at slot 5 (table)
pos(d, spanning(3,5), on(a and b)) % d supported by a and b (as in S0)

S1: d sobre c => move(d, on(c))
S2: a sobre b => move(a, on(b))
S3: d na mesa => move(d, table(3))
S4: a sobre c em 0 => move(a, on(c) at slot0)
S5: b sobre c em 1 => move(b, on(c) at slot1)
S6: igual a S5
S7: d na mesa ocupando 3,4,5 (final)

Plano sugerido (ações mínimas entre estados):

1) move(d, on(c))
Preconditions: clear(d), clear(c), size(d)=3 <= size(c)=2 ? (violação estrita).
Observation: Under strict size rules this is invalid (3 <= 2 false). If allowed, it requires relaxation or different physical model.

2) move(a, on(b))
Preconditions: clear(a), clear(b), size(a)=1 <= size(b)=1 OK.

3) move(d, table(3))
Preconditions: move stack allowed or slots 3..5 free; Effects: pos(d,table(3)).

4) move(a, on(c)) at slot0
Preconditions: clear(a), clear(c), size(a)=1 <= size(c)=2 OK.

5) move(b, on(c)) at slot1
Preconditions: clear(b), clear(c), size(b)=1 <= size(c)=2 OK.

Observações gerais:

- A sequência fornecida contém movimentos que, sob o formalismo estrito (size-check and is_free), requerem relaxações (notadamente move(d,on(c))).

- As ações que empilham blocos menores sobre blocos maiores (a on b, a on c, b on c) são válidas conforme os tamanhos.

- A ação move(d, table(3)) é tratada como um movimento de pilha (d com a,b) para a mesa; isto é modelado aqui como um passo atômico equivalente a move_stack(d,[a,b],table(3)).