CÓDIGO PARA A SITUAÇÃO 2:

```
% --- SITUAÇÃO 2 ---
% Fatos estáticos (mesmos da situação 1)
size(a, 1).
size(b, 1).
size(c, 2).
size(d, 1).

% Estados Situação 2
% Estado Inicial S0
initial_s2_s0([
    pos(c, table(0)),
    pos(a, on(c)),
    pos(b, table(2)),
    pos(d, table(4)),
    clear(a), clear(b), clear(d)
]).

% Estado Final S5
goal_s2_s5([
    pos(c, table(2)),
    pos(a, table(0)),
    pos(b, on(c)),
    pos(d, table(4))
]).

% Plano manual S0 → S5
plan_s2_s0_to_s5([
    move(a, table(1)),
    move(c, table(2)),
    move(b, on(c))
]).
```

CÓDIGO PARA SITUAÇÃO 3:

```
% --- SITUAÇÃO 3 ---
% Fatos estáticos
size(a, 1).
size(b, 1).
size(c, 2).
size(d, 2).

% Estados Situação 3
% Estado Inicial S0
initial_s3_s0([
    pos(d, table(0)),
    pos(b, on(d)),
    pos(a, on(b)),
    pos(c, on(a)),
```

```
    clear(c)
]).

% Estado Final S7
goal_s3_s7([
    pos(b, table(2)),
    pos(a, on(b)),
    pos(c, on(a)),
    pos(d, on(c))
]).

% Plano manual S0 → S7
plan_s3_s0_to_s7([
    move(c, table(4)),
    move(a, table(1)),
    move(b, table(2)),
    move(a, on(b)),
    move(c, on(a)),
    move(d, on(c))
]).
```

Tabela de Conceito (Situação 2):

Conceito	STRIPS	Prolog Estendido	NuSMV	Justificativa
Block Properties	`block(X)`	`size(X, W)`	`DEFINE size_a=1, size_b=1, size_c=2, size_d=1`	Mesma base física
Mesa	`place(N)`	`table_slot(N)`	`slot: 0..6`	Mesma grade
Posição	`on(B,L)`	`pos(B,table(X))`	`on_c : {a,0,1,2,3,4,5,6}`	Bloco C muda de suporte
Mobilidade	`clear(B)`	`clear(B)`	`clear_b = TRUE`	B está livre na mesa
Estabilidade	-	`size(B,W1), size(C,W2), W1=<W2`	`size_b <= size_c`	B pode ir sobre C

Ocupação Espacial	-	`is_free(Slot, State)`	`free_slot(2) & free_slot(3)`	C ocupa 2 slots

Tabela de Restrições (Situação 2):

Tipo de Restrição	Destino	Regra em Linguagem Natural	Implementação NuSMV (Exemplo: move(B, C))	Implementação NuSMV (Exemplo: move(C, table(2)))
Mobilidade	Ambos	O bloco a ser movido deve estar livre (nada sobre ele)	`clear_b = TRUE`	`clear_c = TRUE`
Validade do Target	`on(Target Block)`	O target deve ser um bloco válido e diferente do bloco móvel	`b != c`	-
Acessibilidade do Target	`on(Target Block)`	O bloco alvo deve estar limpo para receber outro bloco	`clear_c = TRUE`	-
Estabilidade	`on(Target Block)`	O bloco móvel não pode ser mais largo que o bloco alvo	`size_b <= size_c`	-
Validade da coordenada	`table(X)`	A coordenada X deve ser um slot válido na mesa	-	`table_slot(2) = TRUE`
Disponibilidade de espaço	`table(X)`	Todos os slots de X a X+W-1 devem estar livres	-	`free_slot(2) & free_slot(3)`
Limite da mesa	`table(X)`	O bloco não pode ultrapassar os limites da mesa	-	`2 + size_c - 1 < table_width`

Tabela de Conceito (Situação 3):

Conceito	STRIPS	Prolog Estendido	NuSMV	Justificativa
Block Properties	`block(X)`	`size(X, W)`	`DEFINE size_a=1, size_b=1, size_c=2, size_d=2`	D também tem tamanho 2
Mesa	`place(N)`	`table_slot(N)`	`slot: 0..6`	Mesma grade
Posição	`on(B,L)`	`pos(B,on(Other))`	`on_d : {c,0,1,2,3,4,5,6}`	D muda de base
Mobilidade	`clear(B)`	`clear(B)`	`clear_c = TRUE`	C é o único acessível
Estabilidade	-	`size(C,W1), size(D,W2), W1=<W2`	`size_c <= size_d`	C pode ir sobre D
Ocupação Espacial	-	`is_free(Slot, State)`	`free_slot(4) & free_slot(5)`	C precisa 2 slots livres

Tabela de Restrições (Situação 3):

Tipo de Restrição	Destino	Regra em Linguagem Natural	Implementação NuSMV (Exemplo: move(C, table(4))	Implementação NuSMV (Exemplo: move(D, on(C)))
Mobilidade	Ambos	O bloco a ser movido deve estar livre (nada sobre ele)	`clear_c = TRUE`	`clear_d = TRUE` (após mover C)

Validade do Target	`on(Target Block)`	O target deve ser um bloco válido e diferente do bloco móvel	-	`d != c`
Acessibilidade do Target	`on(Target Block)`	O bloco alvo deve estar limpo para receber outro bloco	-	`clear_c = TRUE`
Estabilidade	`on(Target Block)`	O bloco móvel não pode ser mais largo que o bloco alvo	-	`size_d <= size_c`
Validade da coordenada	`table(X)`	A coordenada X deve ser um slot válido na mesa	`table_slot(4) = TRUE`	-
Disponibilidade de espaço	`table(X)`	Todos os slots de X a X+W-1 devem estar livres	`free_slot(4) & free_slot(5)`	-
Limite da mesa	`table(X)`	O bloco não pode ultrapassar os limites da mesa	`4 + 2 - 1 = 5 < 7`	-