Rascunho

1. Introdução

O problema do Mundo dos Blocos é clássico em Inteligência Artificial e planejamento automático. Nele, blocos devem ser movimentados da posição inicial para uma configuração final, obedecendo regras de mobilidade, estabilidade e ocupação de espaço.

Neste trabalho, utilizamos a versão com blocos de tamanhos variáveis, o que adiciona restrições adicionais de estabilidade e ocupação espacial. O objetivo é:

- Representar o problema em diferentes linguagens (STRIPS, Prolog estendido, NuSMV).
- Comparar as representações e justificá-las.
- Mapear restrições de planejamento em forma textual e lógica.

2. Situações Modeladas

Foram consideradas três situações:

- **Situação 1**: Configuração inicial `i1` deve ser transformada em `i2a`, `i2b` ou `i2c`.
- **Situação 2**: Duas torres (`a-b` e `c-d`) precisam ser reorganizadas.
- **Situação 3**: Torre `d-b-a-c` deve ser transformada em `b-a-c-d`.

3. Representação Conceitual

Cada situação foi descrita em tabelas de conceitos com STRIPS, Prolog estendido e NuSMV.

📌 **Tabela 1 – Conceitos e Representação**

Conceito	STRIPS	Prolog estendido	Proposta modelo NuSMV	Justificativa
Block Properties	`block(a).`	`tamanho(a,1).`	`DEFINE size_a := 1;`	Cada bloco tem dimensão fixa e imutável, definida como constante para eficiência.

📌 **Tabela 1 – Conceitos e Representação**

Conceito	STRIPS	Prolog estendido	Proposta modelo NuSMV	Justificativa
Block Properties	`block(a).`	`tamanho(d,2).`	`DEFINE size_d := 2;`	Diferentes larguras afetam as regras de estabilidade.

📌 **Tabela 1 – Conceitos e Representação**

Conceito	STRIPS	Prolog estendido	Proposta modelo NuSMV	Justificativa
Block Properties	`block(d).`	`tamanho(d,2).` `tamanho(a,1).`	`DEFINE size_d := 2; size_a`	A ordem da torre depende dos tamanhos fixos, constantes.

4. Restrições de Planejamento

As restrições foram mapeadas em linguagem natural e formalizadas no NuSMV.

📌 **Tabela 2 – Restrições**

Tipo de Restrição	Destino	Regra em Linguagem Natural	Implementação NuSMV (move(a,b))	Implementação NuSMV (move(a,table(2)))
Mobility	Bloco a	Só pode mover a se não houver outro bloco em cima dele	`TRANS move_a_b -> clear_a;`	`TRANS move_a_table2 -> clear_a;`
Target Accessibility	Bloco b ou c	Só é possível colocar a em cima de b ou c se estes estiverem livres	`TRANS move_a_b -> clear_b;`	—
Stability	Empilhamento	a só pode ser colocado sobre blocos de tamanho ≥ tamanho de a	`TRANS move_a_b -> size_a <= size_b;`	`TRANS move_a_c -> size_a <= size_c;`
Spatial Occupancy	Slots da mesa	Para colocar c na mesa em posição 3, slots 3 e 4 precisam estar livres	—	`TRANS move_c_table3 -> free(3) & free(4);`
Logical Validity	—	Um bloco não pode ser colocado sobre si mesmo	Implícito	Implícito

📌 Tabela 2 – Restrições

Tipo de Restrição	Destino	Regra em Linguagem Natural	Implementação NuSMV (move(c,d))	Implementação NuSMV (move(a,table(1)))
Mobility	Bloco a,c	Só podem ser movidos se não houver outro bloco em cima	`TRANS move_c_d -> clear_c;`	`TRANS move_a_table1 -> clear_a;`
Target Accessibility	Bloco b,d	Só pode empilhar `a` em `b` ou `c` em `d` se estes estiverem livres	`TRANS move_a_b -> clear_b;`	—
Stability	Empilhamento	Só é permitido colocar `c` sobre `d` se `size(c) ≤ size(d)`	`TRANS move_c_d -> size_c <= size_d;`	—
Spatial Occupancy	Slots mesa	Para colocar `d` na mesa, slots necessários devem estar livres	—	`TRANS move_d_table3 -> free(3) & free(4);`
Logical Validity	—	Não pode mover bloco para cima de si mesmo	Implícito	Implícito

📌 **Tabela 2 – Restrições**

Tipo de Restrição	Destino	Regra em Linguagem Natural	Implementação NuSMV (move(c,a))	Implementação NuSMV (move(b,table(2)))
Mobility	Bloco c	Só o topo da torre (c) pode ser movido inicialmente	`TRANS move_c_a -> clear_c;`	—
Target Accessibility	Bloco a,b	Para reconstruir a torre, só é possível empilhar em blocos livres	`TRANS move_c_a -> clear_a;`	`TRANS move_b_table2 -> clear_b;`
Stability	Empilhamento	`d` só pode ser colocado sobre `c` se `size(d) ≤ size(c)`	`TRANS move_d_c -> size_d <= size_c;`	—
Spatial Occupancy	Slots mesa	Para desmontar a torre, deve haver espaço na mesa	—	`TRANS move_b_table2 -> free(2);`
Logical Validity	—	Nenhum bloco pode ser colocado sobre si mesmo	Implícito	Implícito

5. Discussão

- **STRIPS**: fornece uma forma declarativa simples de descrever ações, mas não representa bem restrições de estabilidade e espaço.
- **Prolog estendido**: mais expressivo, porém de execução mais complexa.
- **NuSMV**: robusto para verificação formal, permite assegurar que o plano gerado nunca viole estabilidade ou ocupação.

6. Conclusão

O modelo do Mundo dos Blocos com tamanhos variáveis mostra que o aumento da complexidade requer uma modelagem mais rica em restrições. O uso do NuSMV se mostra adequado para garantir validade formal do planejamento, evitando estados inválidos.

7. Divisão de Atividades (Equipe de 6 pessoas)

1. **Levantamento Teórico (2 pessoas)** – Revisar STRIPS, Prolog estendido e NuSMV.

2. **Modelagem das Situações (2 pessoas)** – Criar tabelas de conceitos para cada situação.

3. **Modelagem de Restrições (2 pessoas)** – Preencher tabelas de restrições em linguagem natural e NuSMV.

4. **Integração e Redação Final (Todos)** – Juntar tabelas em um único relatório e revisar.

8. Modelagem Inicial em NuSMV

Para complementar a análise, seguem modelos iniciais simplificados em NuSMV para cada situação. Eles servem como base conceitual e podem ser refinados posteriormente para incluir verificação completa de estabilidade e ocupação de slots.

Situação 1:

```
MODULE main
VAR
  pos_a : {table0, table2, on_b, on_c};
  pos_b : {table2};
  pos_c : {table3};
  pos_d : {table5};

INIT
  pos_a = table0 & pos_b = table2 & pos_c = table3 & pos_d =
table5;

TRANS
  next(pos_a) = on_b | next(pos_a) = on_c;

DEFINE
  goal := (pos_a = on_b) | (pos_a = on_c);

CTLSPEC EF goal
```

Situação 2:

```
MODULE main
VAR
  pos_a : {on_b, table1};
  pos_b : {table0};
  pos_c : {on_d, table2};
  pos_d : {table2};

INIT
  pos_a = on_b & pos_b = table0 & pos_c = on_d & pos_d = table2;

TRANS
  next(pos_a) = table1 | next(pos_c) = table2;

DEFINE
  goal := (pos_a = table1 & pos_b = table0 & pos_c = table2 &
pos_d = table2);

CTLSPEC EF goal
```

Situação 3:

```
MODULE main
VAR
  pos_a : {on_b, table1};
  pos_b : {on_d, table2};
  pos_c : {on_a, table3};
  pos_d : {table0, on_c};

INIT
  pos_d = table0 & pos_b = on_d & pos_a = on_b & pos_c = on_a;

TRANS
  next(pos_c) = table3;
  next(pos_a) = table1;
  next(pos_b) = table2;
  next(pos_a) = on_b;
  next(pos_c) = on_a;
  next(pos_d) = on_c;

DEFINE
  goal := (pos_b = table2 & pos_a = on_b & pos_c = on_a & pos_d
= on_c);

CTLSPEC EF goal
```