Fundamentos de IA - Planejamento - Prof Edjard Mota
Estendendo a estrutura de planejamento STRIPS para um mundo de blocos com tamanhos heterogêneos e restrições espaciais: uma abordagem de programação lógica

Tabela de Conceitos e Representação

Conceito	STRIPS	Prolog estendido	Proposta de modelo NuSMV	Justificativa para projeto NuSMV
Block Properties	`block(X)`	`size(X, W)`	`DEFINE size_a := 1; DEFINE size_b := 2; DEFINE size_c := 3;`	Codifica dimensões físicas imutáveis como constantes de tempo de compilação, permitindo representar blocos de tamanhos variados (1, 2, 3) para maior flexibilidade e eficiência, evitando variáveis de estado.
Table Representation	`lugar(N)`	`table_slot(N)`	`DEFINE table_slot := {0..6};`	Estabelece uma grade fixa de 0 a 6 como conjunto constante, facilitando o raciocínio sobre posições absolutas e ocupação espacial, superando a abstração do STRIPS.
Block Position	`on(Block, Object)`	`pos(Block, table(X)) or pos(Block, on(OtherBlock))`	`VAR pos_C : {table(0)..table(6), on(A), on(B), on(C)}; VAR pos_A : {table(0)..table(6), on(B), on(C)};`	Permite rastrear posições absolutas na mesa ou relativas a múltiplos blocos (A, B, C), resolvendo a limitação relacional do `on/2` e suportando cenários complexos com vários objetos.
Clear Status	`clear(Object)`	`clear(Block)`	`VAR clear_C : boolean; VAR clear_A : boolean; VAR clear_B : boolean;`	Representa o estado de desobstrução por bloco de forma binária, adaptando-se a múltiplas interações e pré-condições de mobilidade, como exigido pelo operador `move`.
Stability Constraint	Não representado	`size(Block1, W1), size(Block2, W2), W1 <= W2`	`DEFINE stability_rule := (size(C) <= size(A) & size(A) <= size(B));`	Garante estabilidade ao limitar a largura do bloco superior à do inferior em empilhamentos múltiplos, corrigindo a ausência de propriedades físicas no STRIPS.
Spatial Occupancy	Não representado	`occupies(Block, Start, End)`	`VAR occupied_slots : array[0..6] of boolean; VAR occupied_by : array[0..6] of {none, A, B, C};`	Registra ocupação de slots e proprietário (A, B, C) na grade, permitindo verificar disponibilidade contígua para blocos largos e rastrear alocações, essencial para restrições espaciais.
Action Definition	`move(Block, From, To)`	`can(move(Block, From, To), [clear(Block), clear(To), on(Block, From)])`	`DEFINE move_action := (pos_C = table(2) & clear_C & forall(S in {2..2+size(C)-1}, is_free(S)));`	Define ações com pré-condições espaciais e de mobilidade, expandindo o esquema tripartite do STRIPS para incluir verificação de espaço livre na mesa.
State Transition	`adds/move(X,From,To), deletes/move(X,From,To)`	`updates_state(move(X,From,To), [on(X,To), clear(From)])`	`NEXT(occupied_slots) := update_occupied(pos_C, size_C); NEXT(clear_A) := update_clear(pos_A);`	Atualiza dinamicamente ocupação e status de desobstrução após movimentos, refletindo transições espaciais e lógicas no ambiente.

Conceito	STRIPS	Prolog estendido	Proposta de modelo NuSMV	Justificativa para projeto NuSMV
Goal Representation	`[on(a,b), clear(a)]`	`goal_state([on(a,b), clear(a)])`	`SPEC AG (on_goal_state => goal_reached); SPEC EF (pos_C = on(A));`	Especifica objetivos como propriedades verificáveis e estados futuros desejados, integrando planejamento com verificação de modelo em NuSMV.
Environment Constraints	Não representado	`max_height(N)`	`DEFINE max_height := 3; DEFINE max_width := 6;`	Impõe restrições ambientais como altura máxima e largura total da mesa, atendendo a limitações implícitas nos cenários e garantindo viabilidade física.
Initial State	`[clear(2), clear(4), on(a,1)]`	`initial_state([clear(2), clear(4), on(a,1)])`	`INIT(occupied_slots) := {false, false, false, false, false, false}; INIT(pos_C) := table(1);`	Define um estado inicial explícito, permitindo simulações consistentes e rastreamento desde o início, superando a abstração inicial do STRIPS.
Error Handling	Não representado	`error_check(move(X,from(F),to(T)), error_condition(E)).`	`DEFINE error_condition := (pos_C = on(C));`	Incorpora verificação de erros (ex.: movimento inválido como sobre si mesmo), aumentando a robustez do modelo.
Temporal Consistency	Não representado	`time_step(T)`	`VAR time : 0..100; NEXT(time) := time + 1;`	Adiciona uma dimensão temporal para rastrear a sequência de ações, essencial para planejamento dinâmico e análise de regressão.