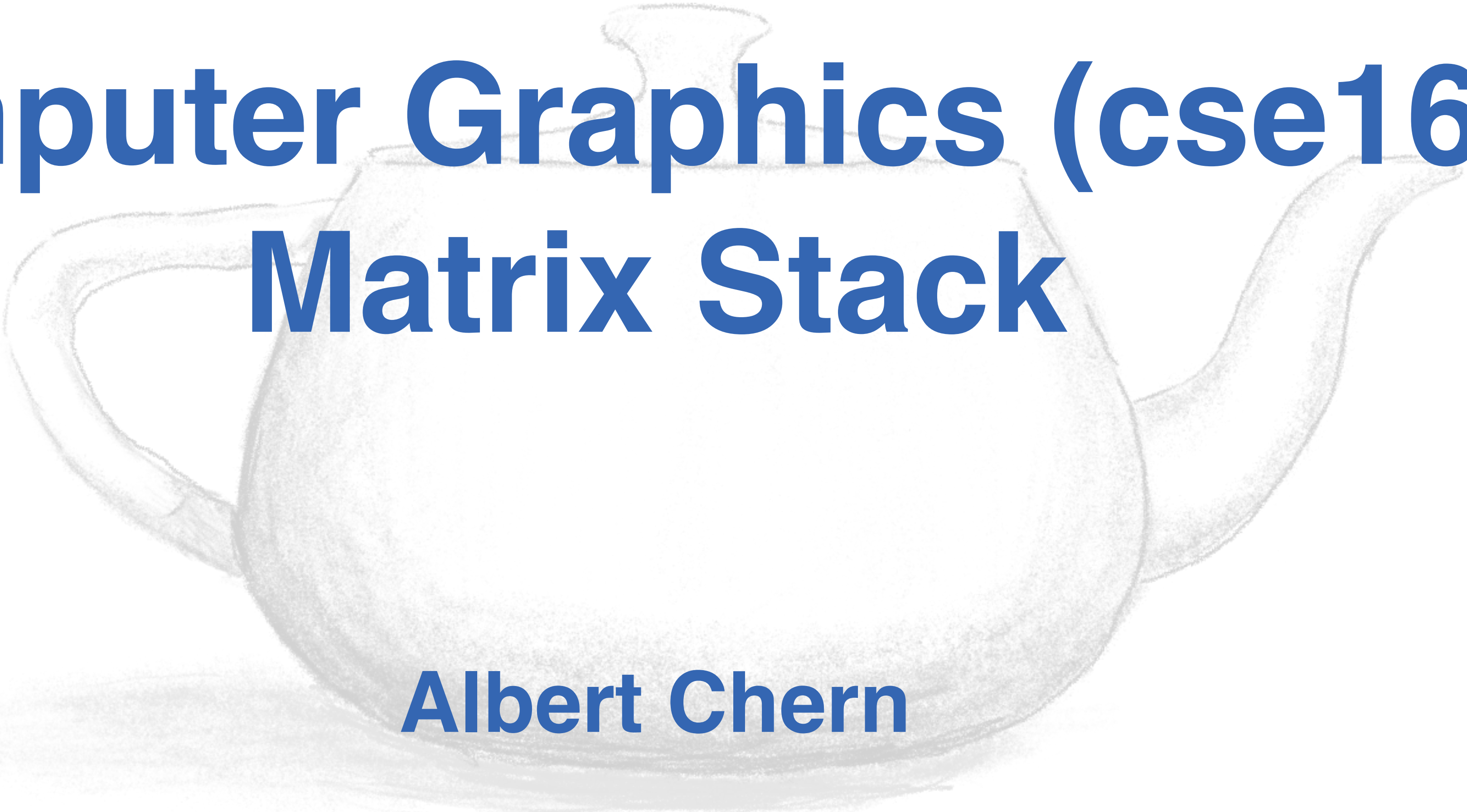




Computer Graphics (cse167)

Matrix Stack

Albert Chern

Basic setup

```
void init(void){
    ...// compile shader, link program etc
    GLuint projectionPos, modelviewPos; // Locations of uniforms
    projectionPos = glGetUniformLocation(program, "projection");
    modelviewPos = glGetUniformLocation(program, "modelview");
    ...
    // Set perspective projection P
    glUniformMatrix4fv(projectionPos, 1, GL_FALSE, &(P)[0][0]);
}
```

```
void display(void){
    ... // VAO describes our geometry
    ... // VM the modelview matrix for this geometry

    glUniformMatrix4fv(modelviewPos, 1, GL_FALSE, &(VM)[0][0]);
    glBindVertexArray(VAO);
    glDrawElements(GL_TRIANGLES, n, GL_UNSIGNED_BYTE, 0);
    ...
}
```

helper function “drawMyObj1”

Vertex shader

```
#version 330 core

// Inputs
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 color;

// Uniforms
uniform mat4 projection;
uniform mat4 modelview;

// Extra outputs, if any
out vec3 color;

void main() {
    gl_Position = projection * modelview
        * vec4(position, 1.0f);
}
```

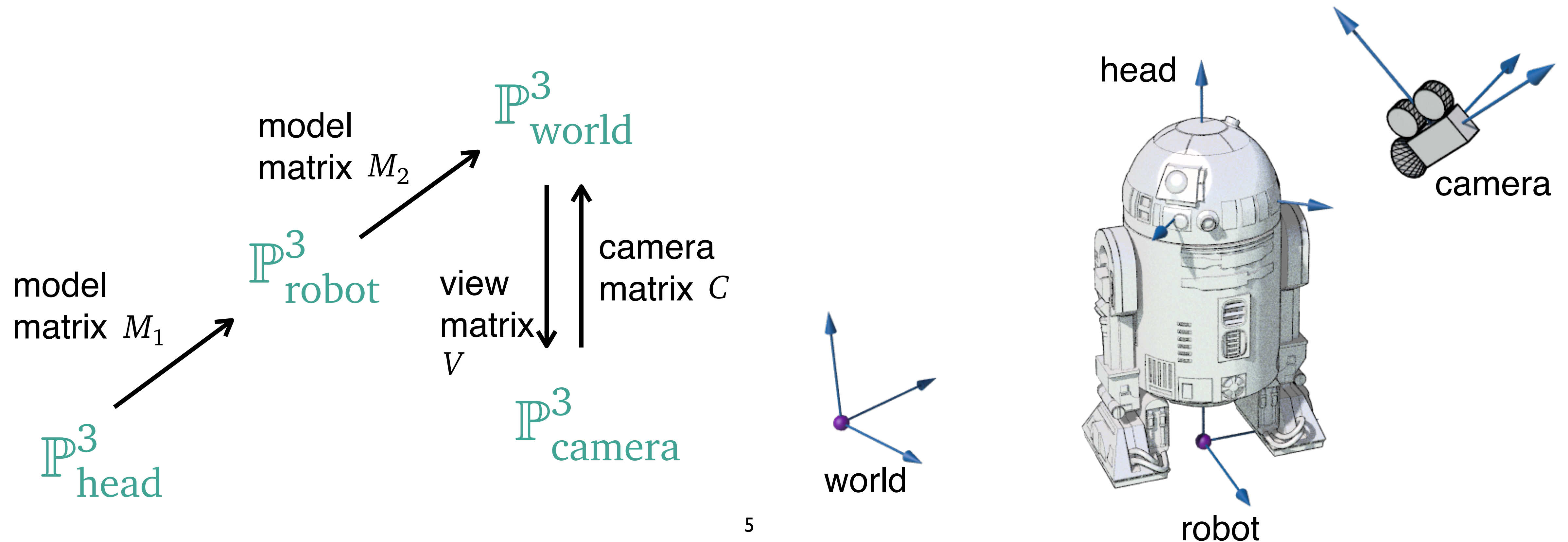
Basic setup

- We often need to draw a lot of objects, each of which need a **modelview** matrix

```
void display(void){  
    ... // compute VM1  
  
    glUniformMatrix4fv(modelviewPos, 1, GL_FALSE, &(VM1)[0][0]);  
    drawMyObj1;  
  
    ... // compute VM2  
  
    glUniformMatrix4fv(modelviewPos, 1, GL_FALSE, &(VM2)[0][0]);  
    drawMyObj2;  
  
    ...  
}
```

Scene Hierarchy

- We often need to draw a lot of objects, each of which need a **modelview** matrix



Matrix Stack

Matrix Stack

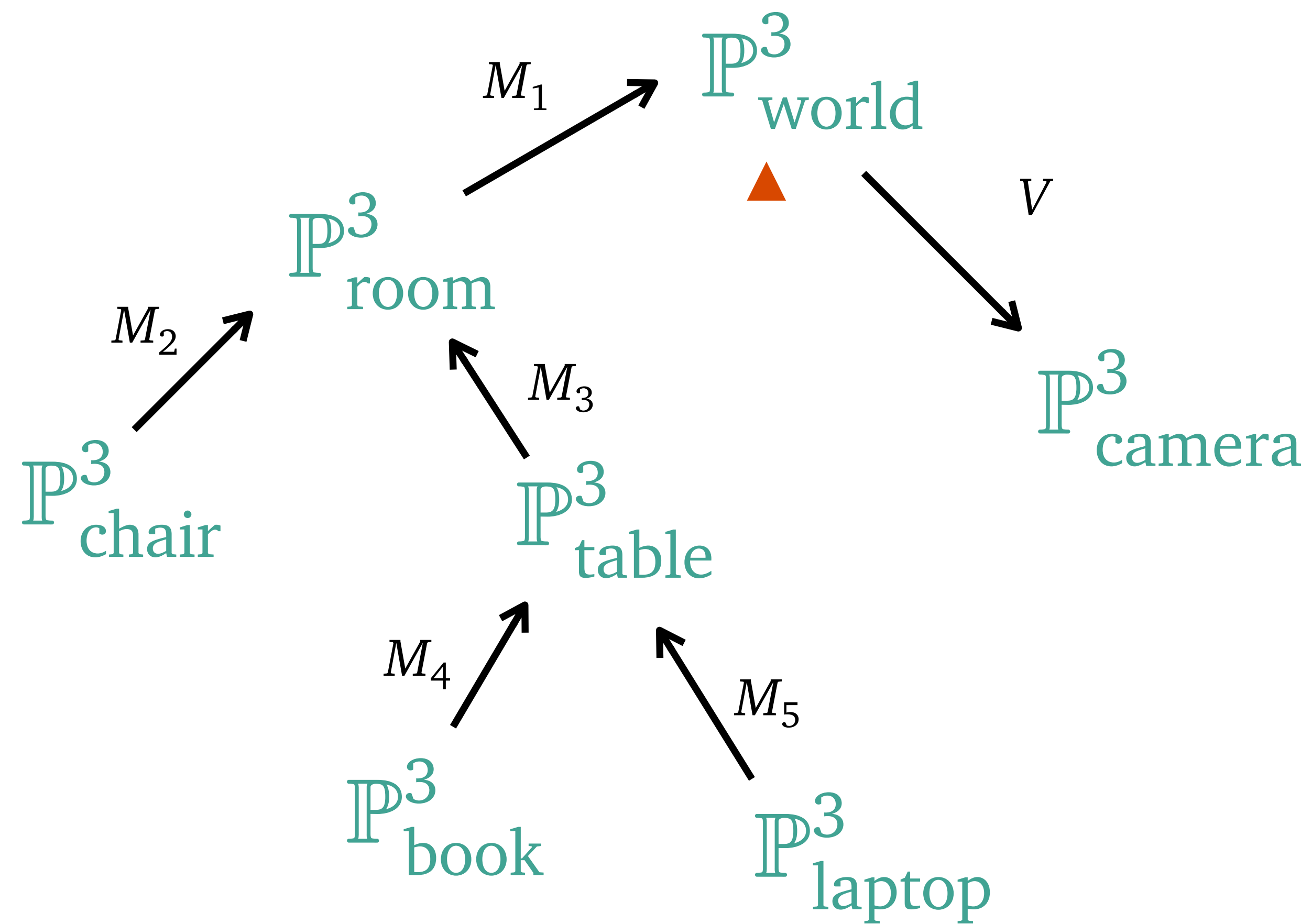
- In a large scene with many objects, whose modelview matrices are

$$VM_1M_2M_3M_5 \quad VM_1M_2M_3M_6 \quad VM_1M_2M_3M_5M_7 \quad \text{etc.}$$

- ▶ We don't want to recompute repetitively the same matrix multiplications.
- ▶ We use a stack to store the results of matrix multiplication of intermediate stages.
- ▶ A stack is an array with a **pop()** and a **push(obj)** function. Pop takes out the element in the array that was pushed in most recently (contrary to queue).

Matrix Stack

- ▶ Define a **modelviewStack** STACK
- ▶ Let **VM** be the current modelview matrix
- ▶ Initially $VM = V$



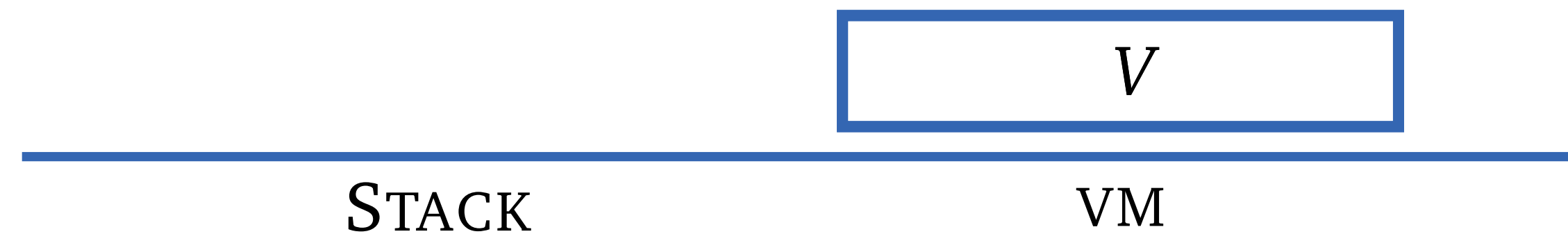
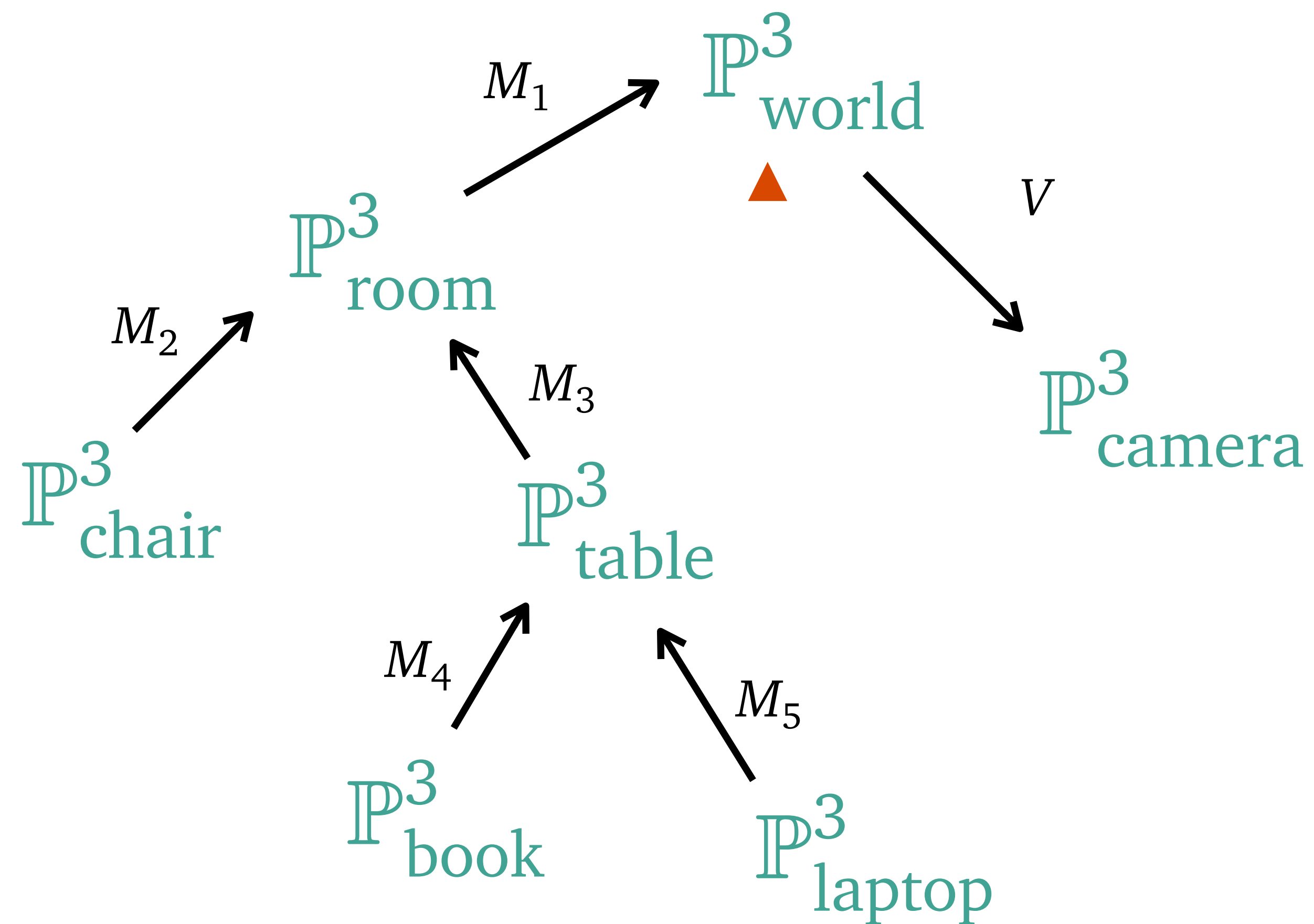
V

STACK

VM

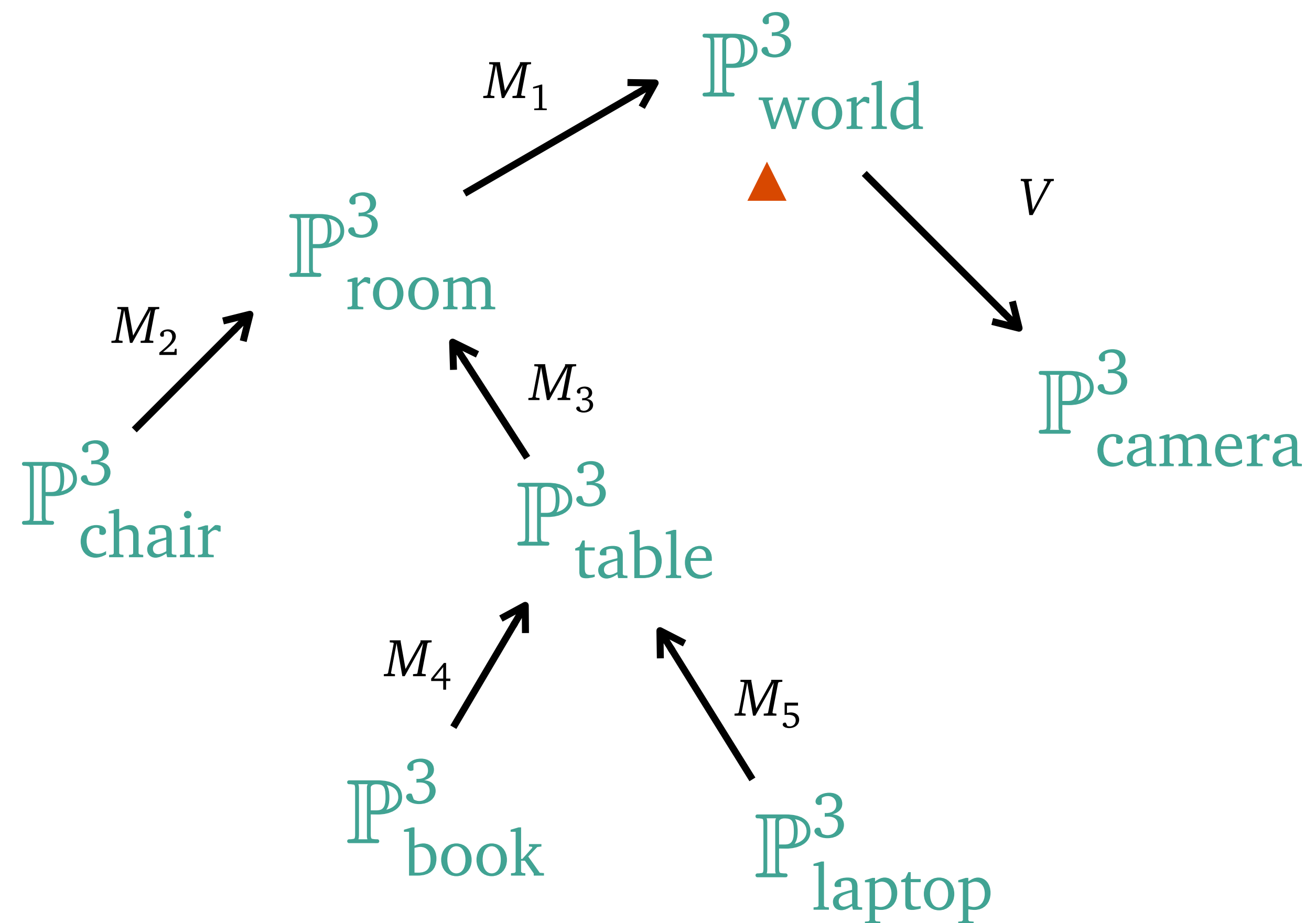
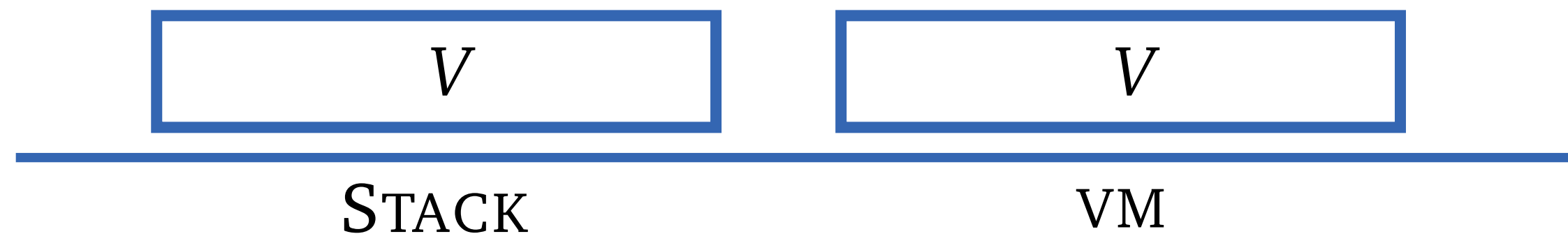
Matrix Stack

- ▶ Define a **modelviewStack** `STACK`
- ▶ Let **VM** be the current modelview matrix
- ▶ Initially $VM = V$
- ▶ Push `STACK.PUSH(VM)`



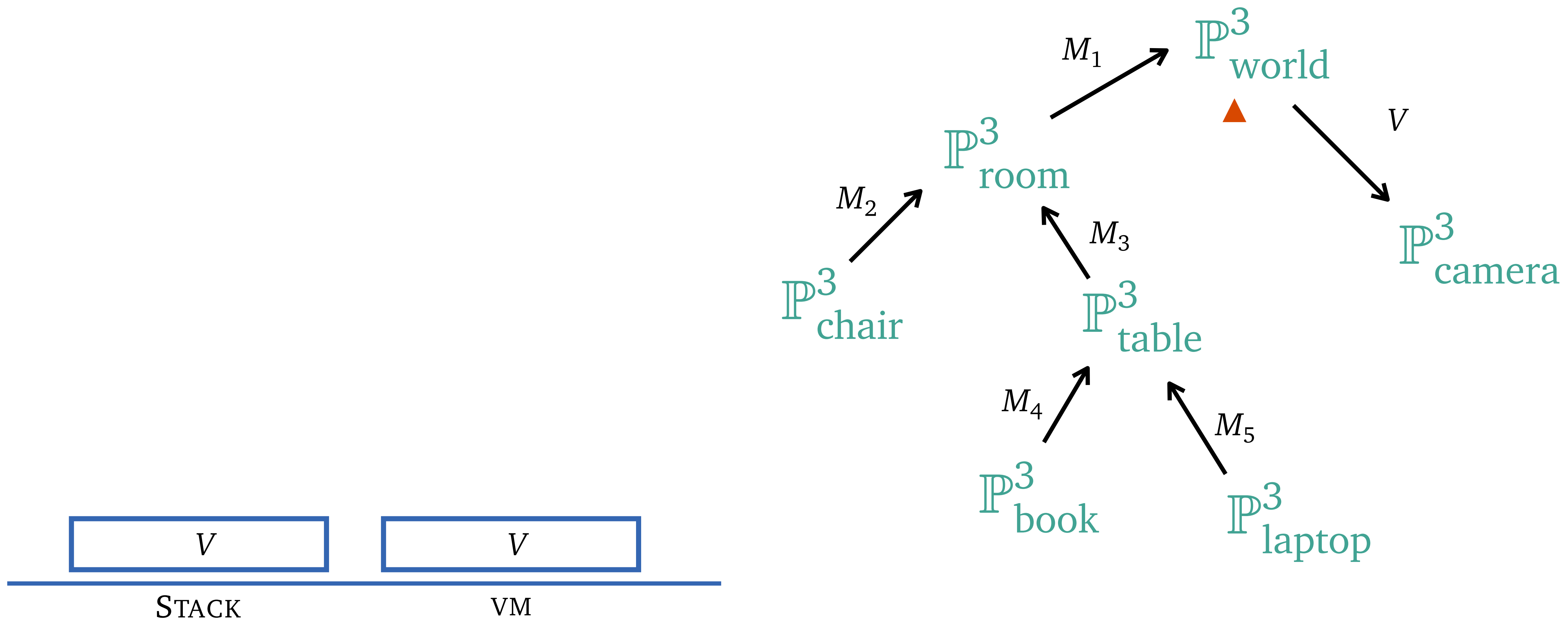
Matrix Stack

- ▶ Define a **modelviewStack** STACK
- ▶ Let **VM** be the current modelview matrix
- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$



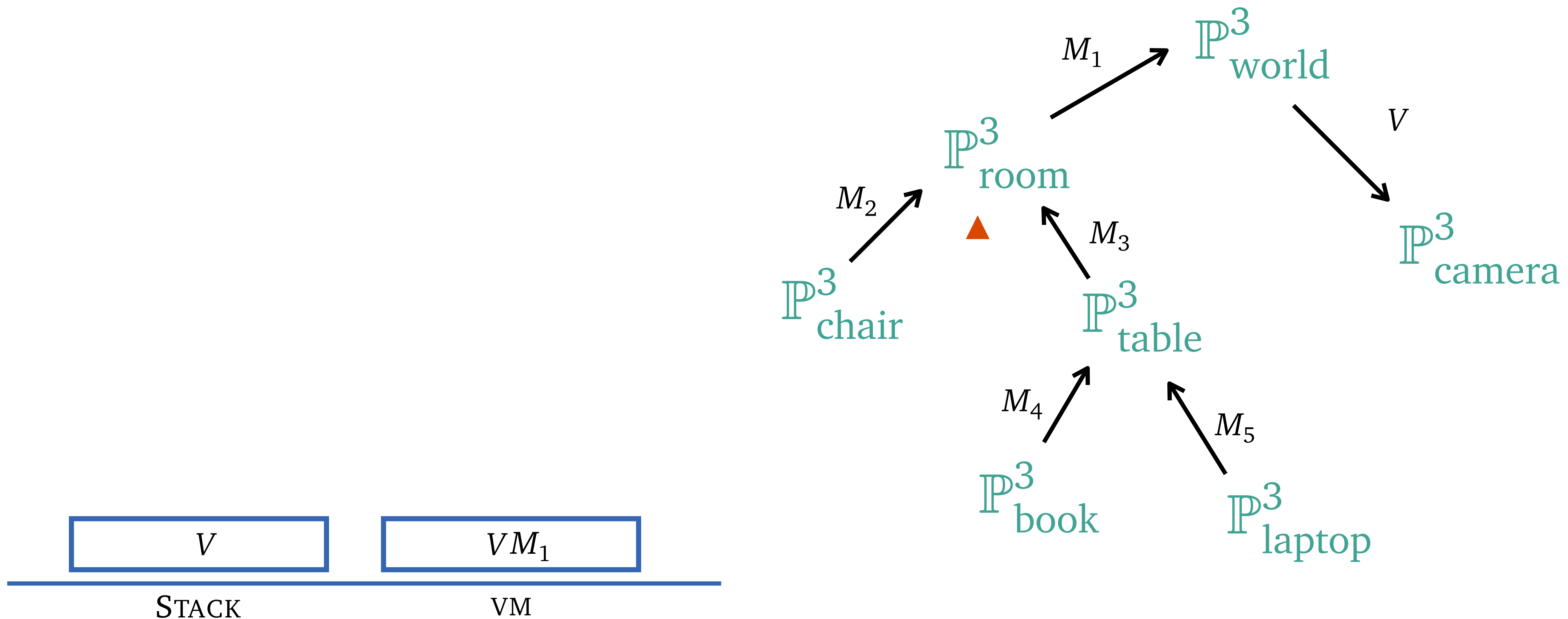
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$



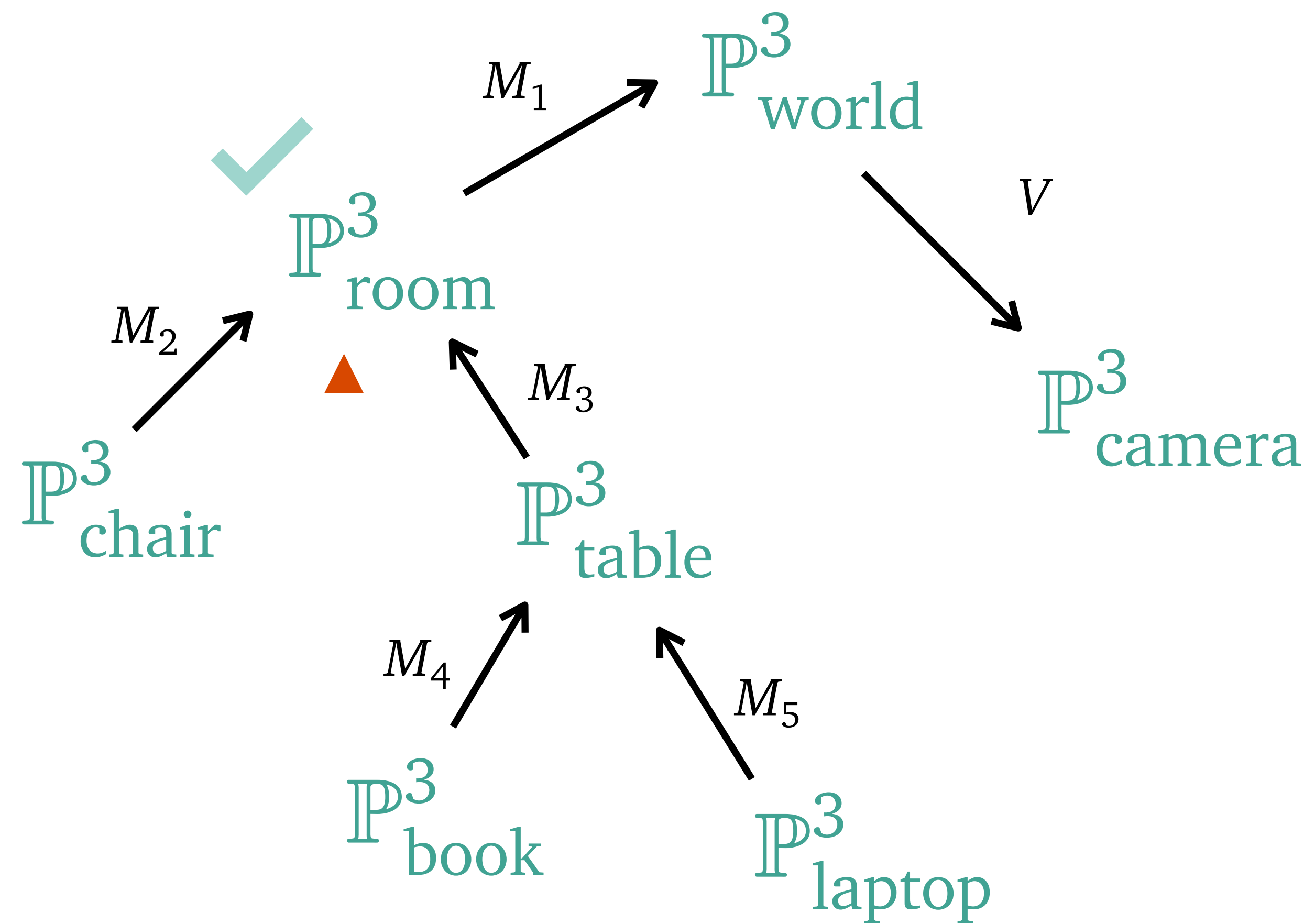
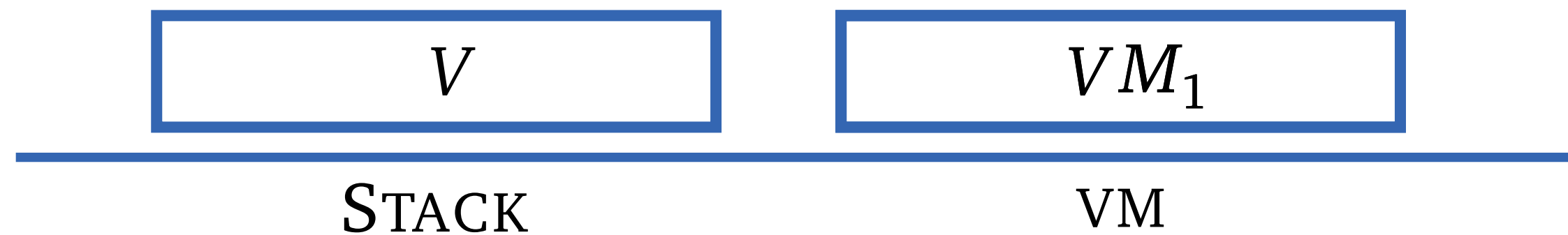
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$



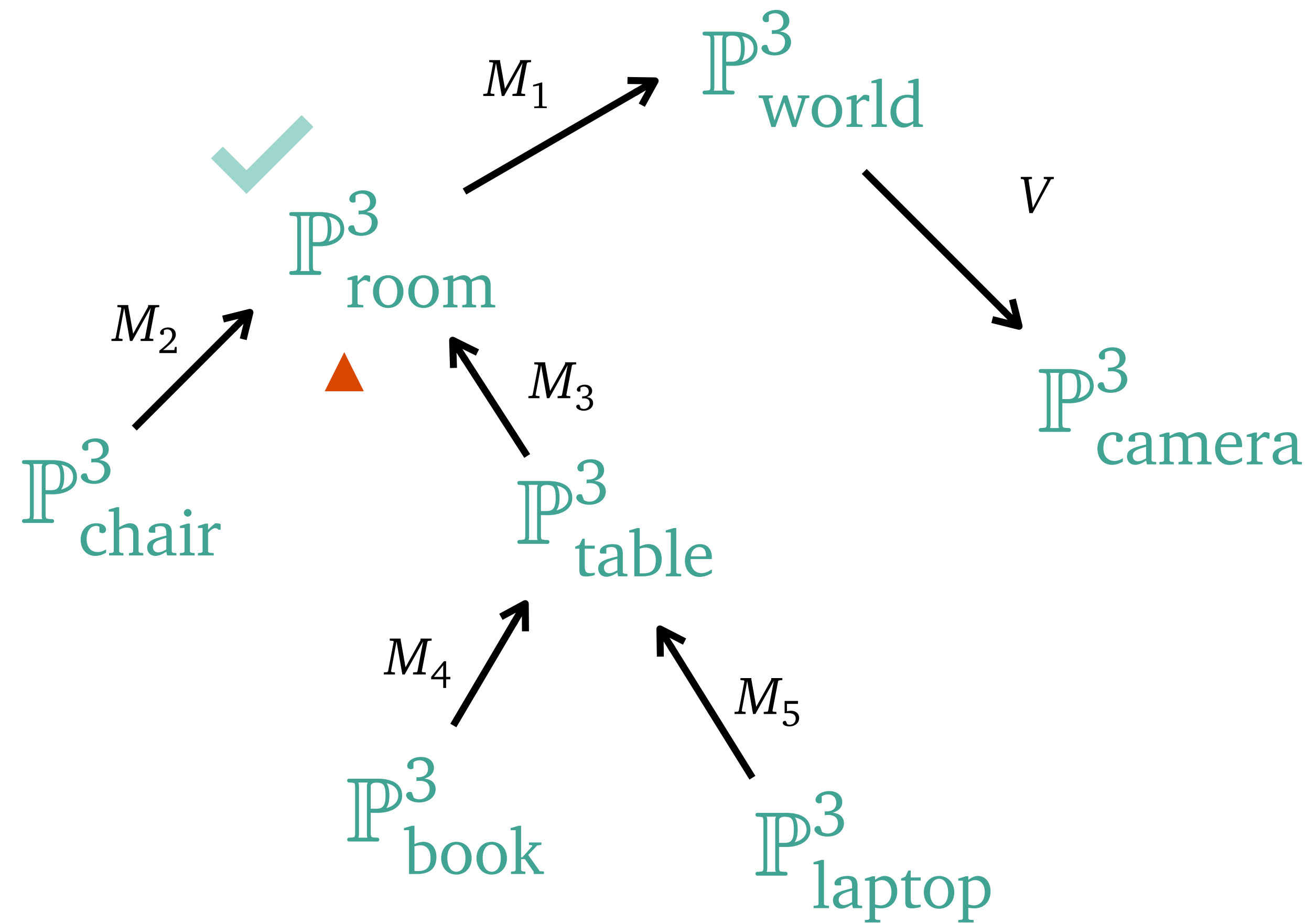
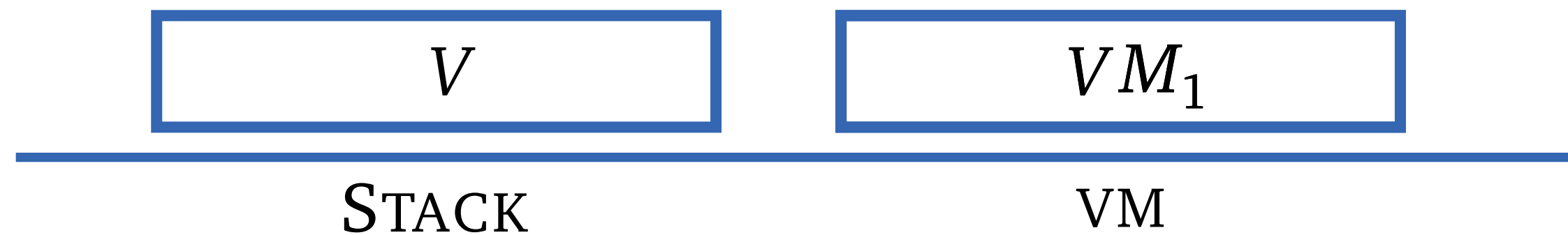
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom



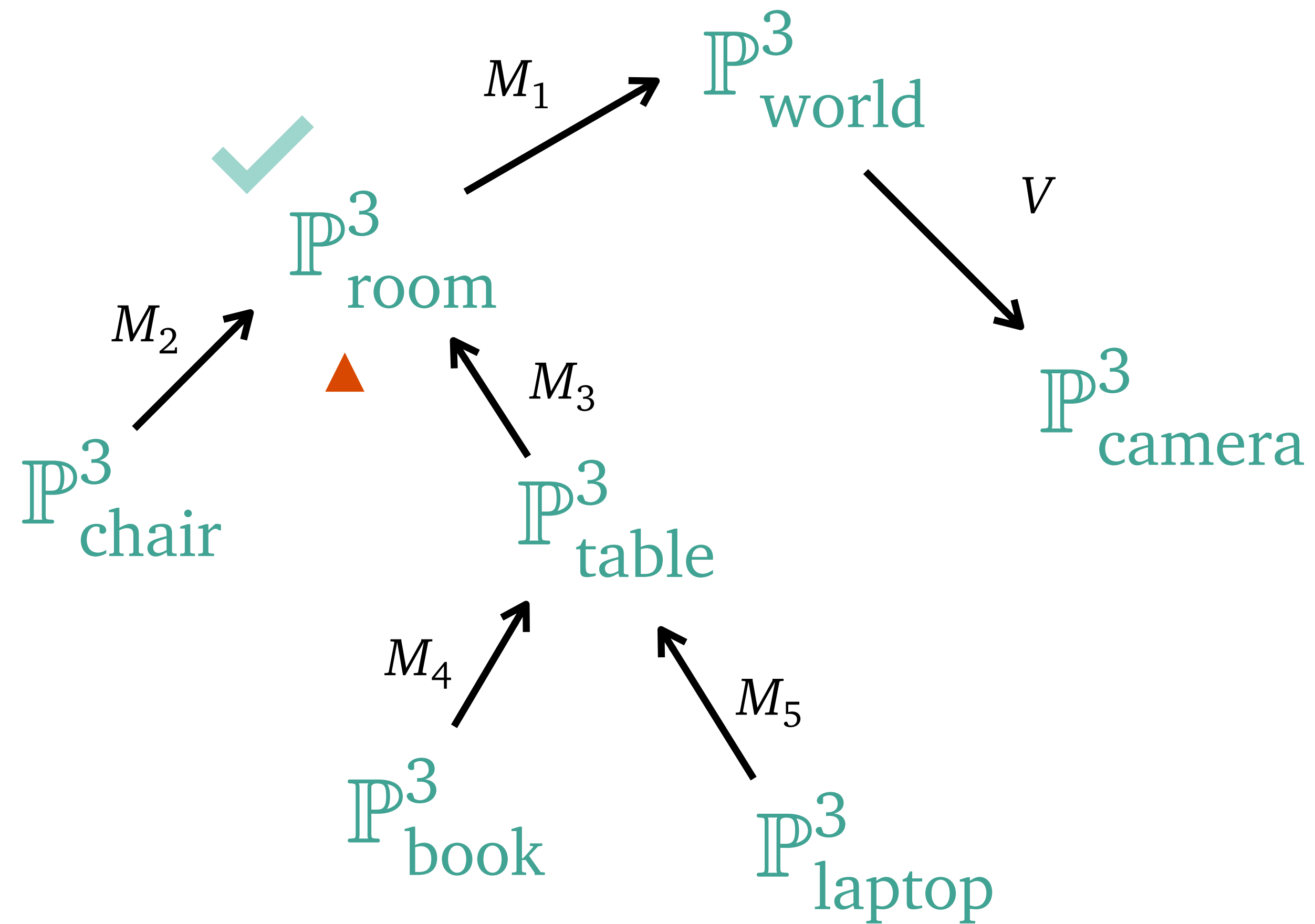
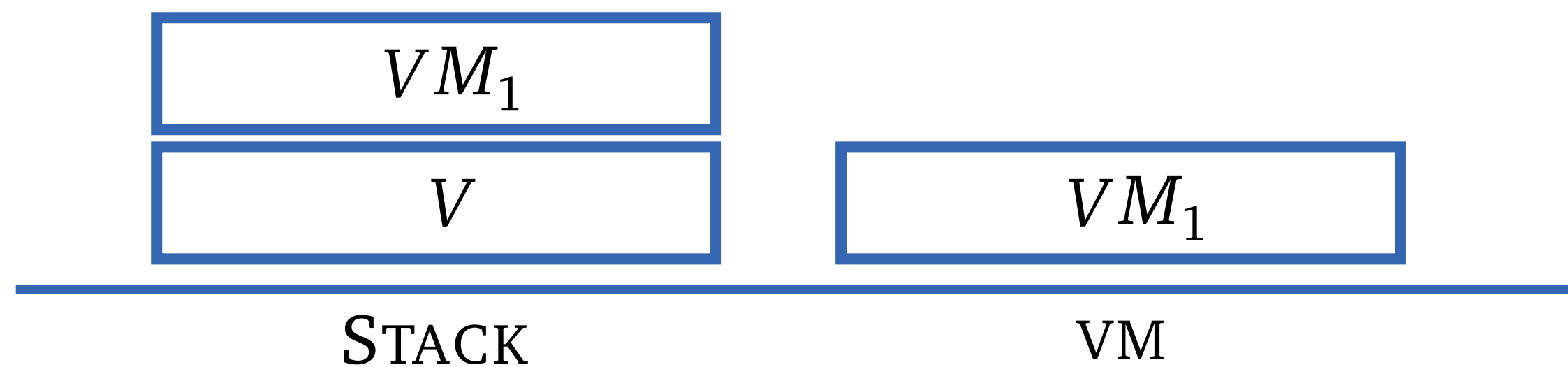
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$



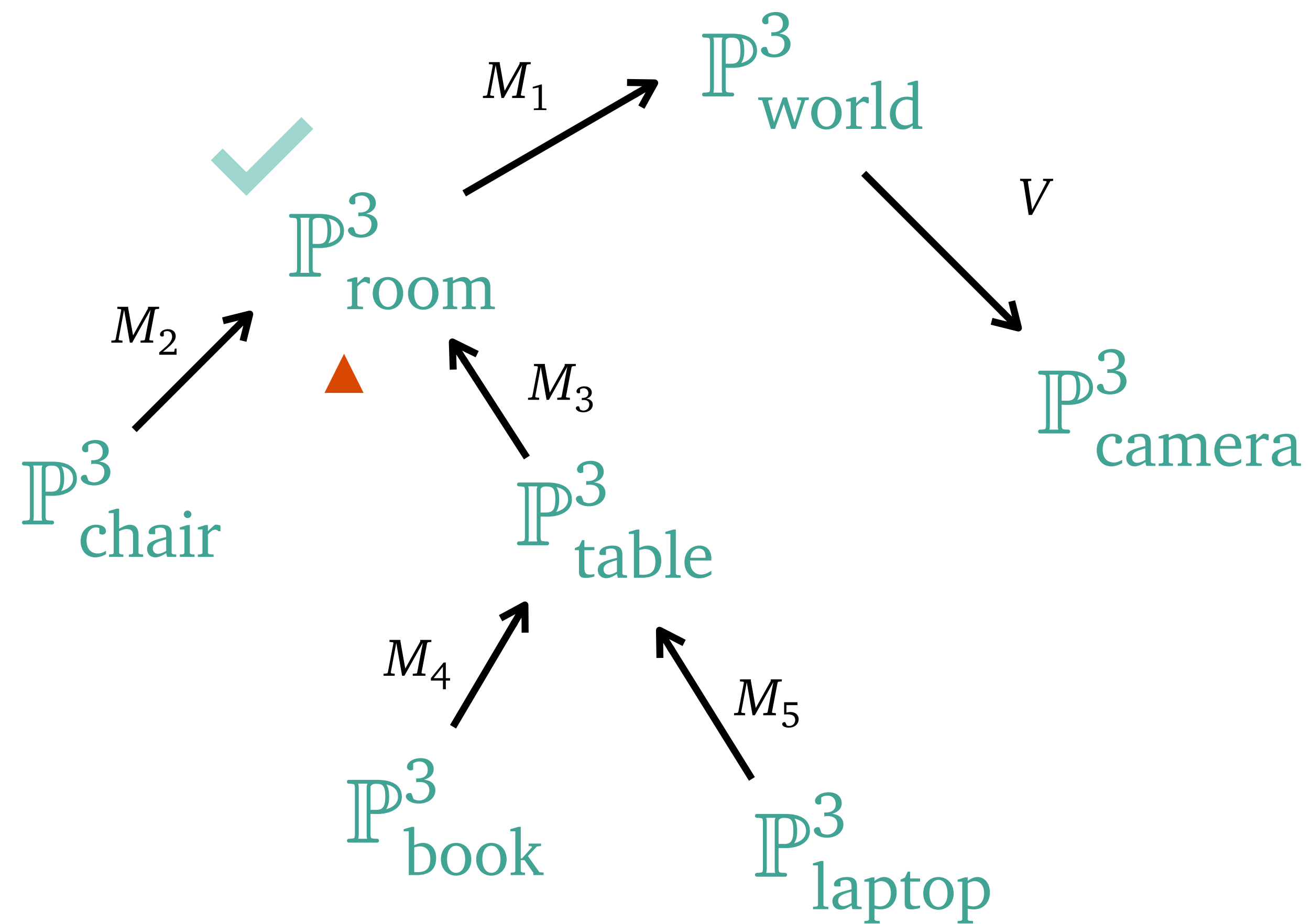
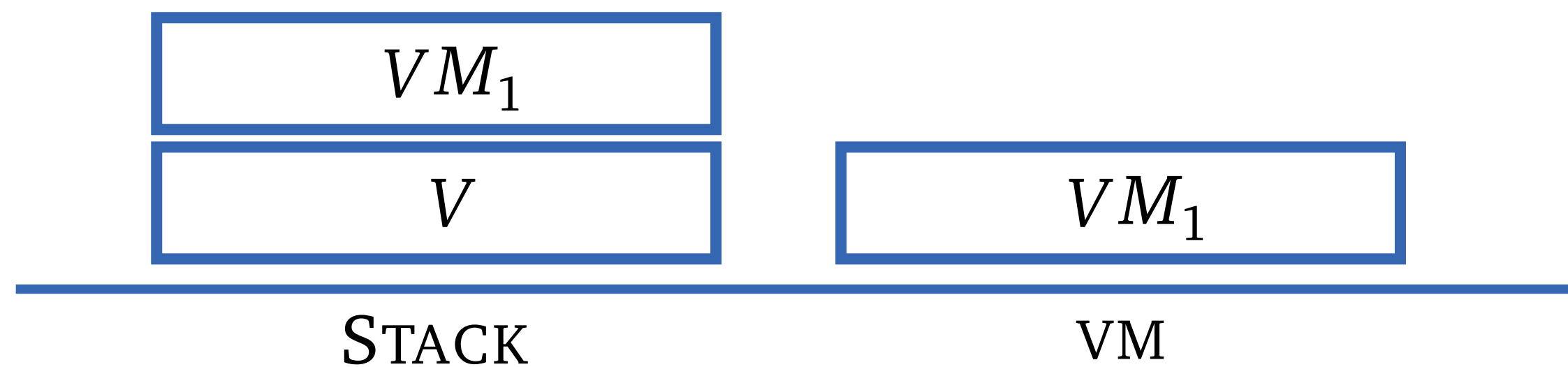
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$



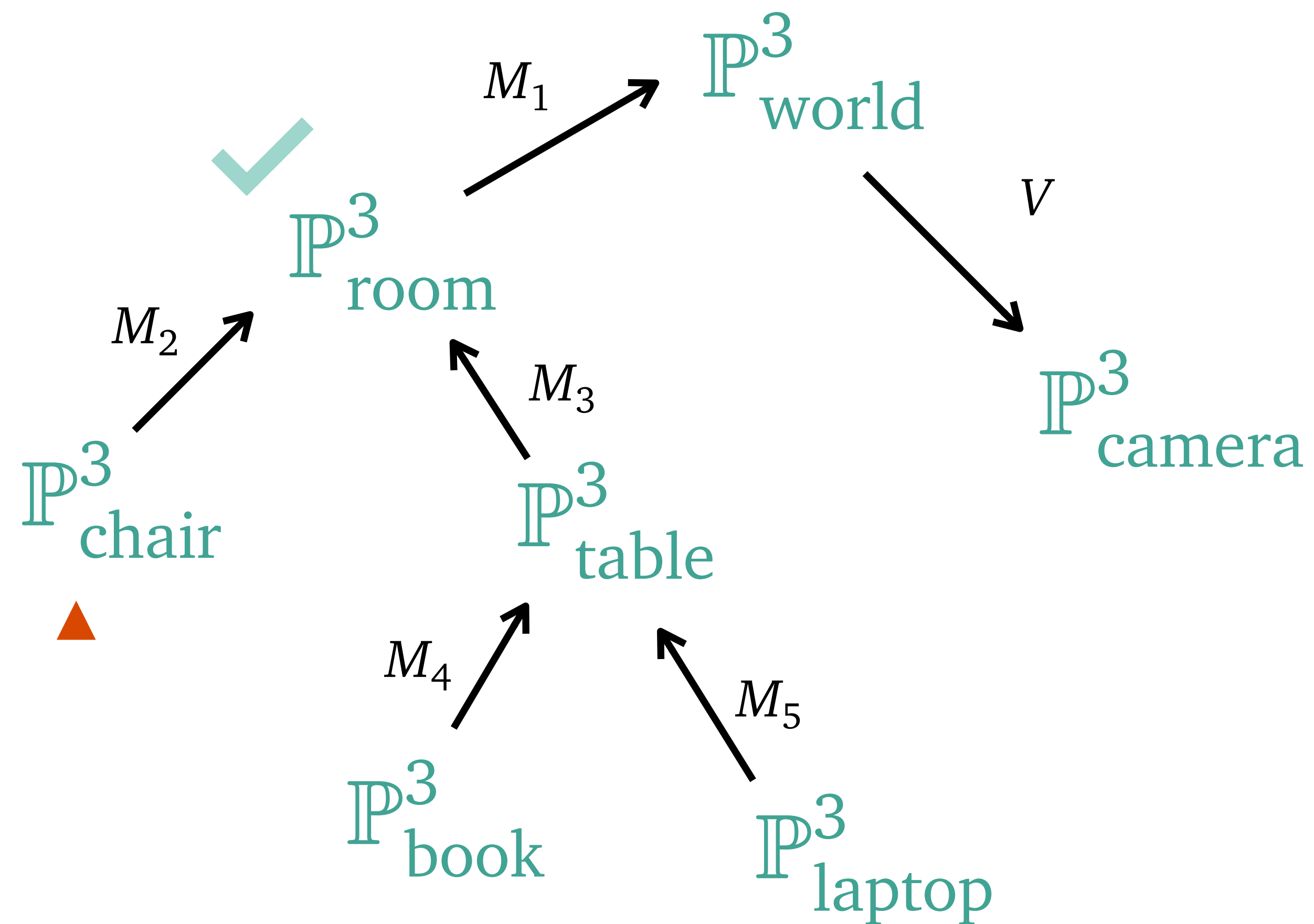
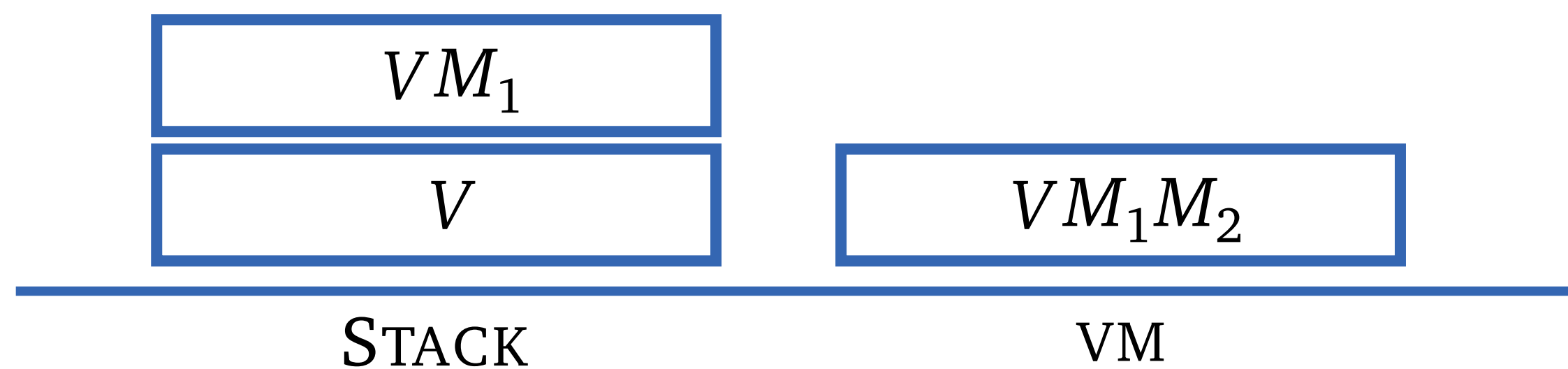
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$



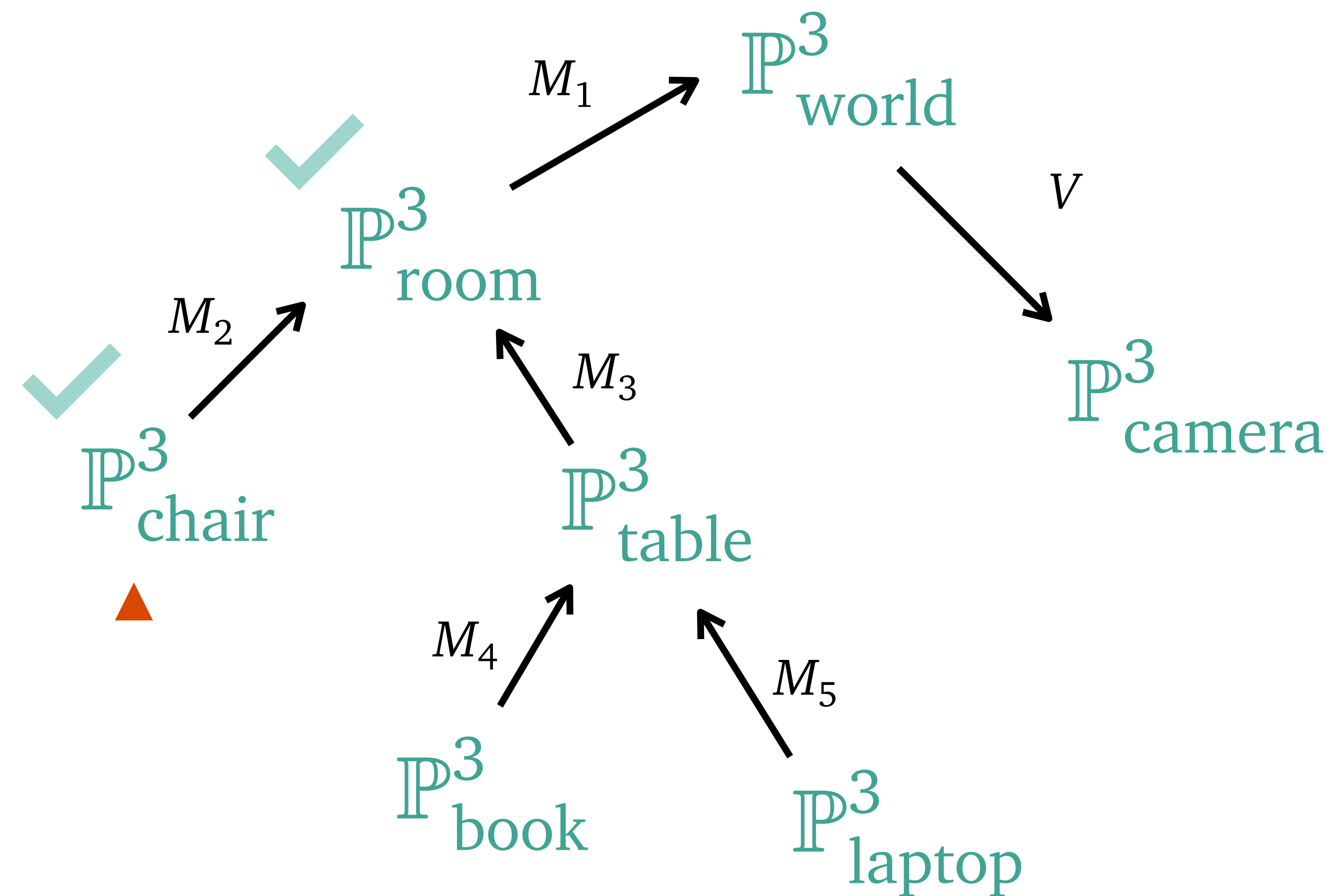
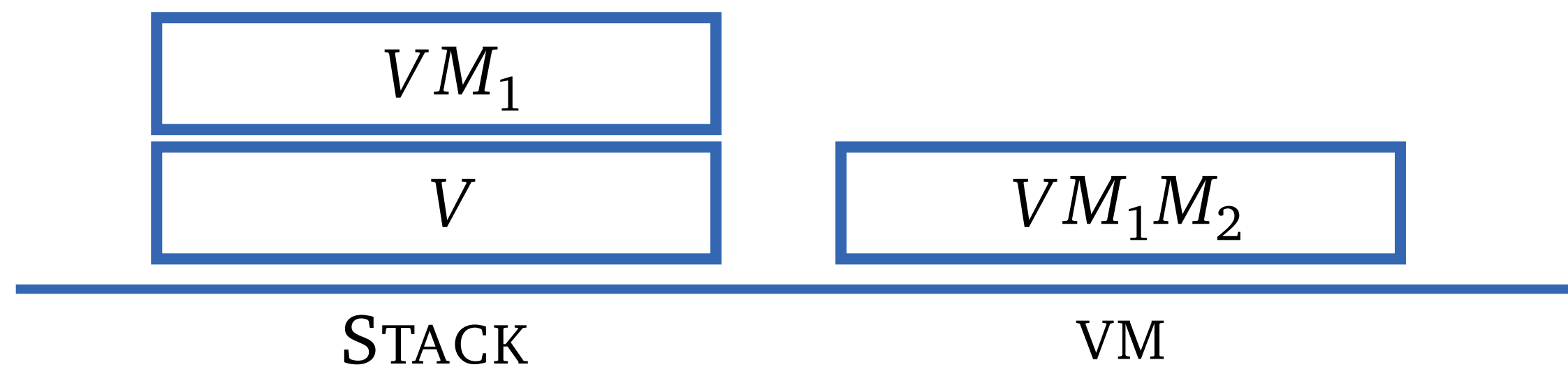
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$



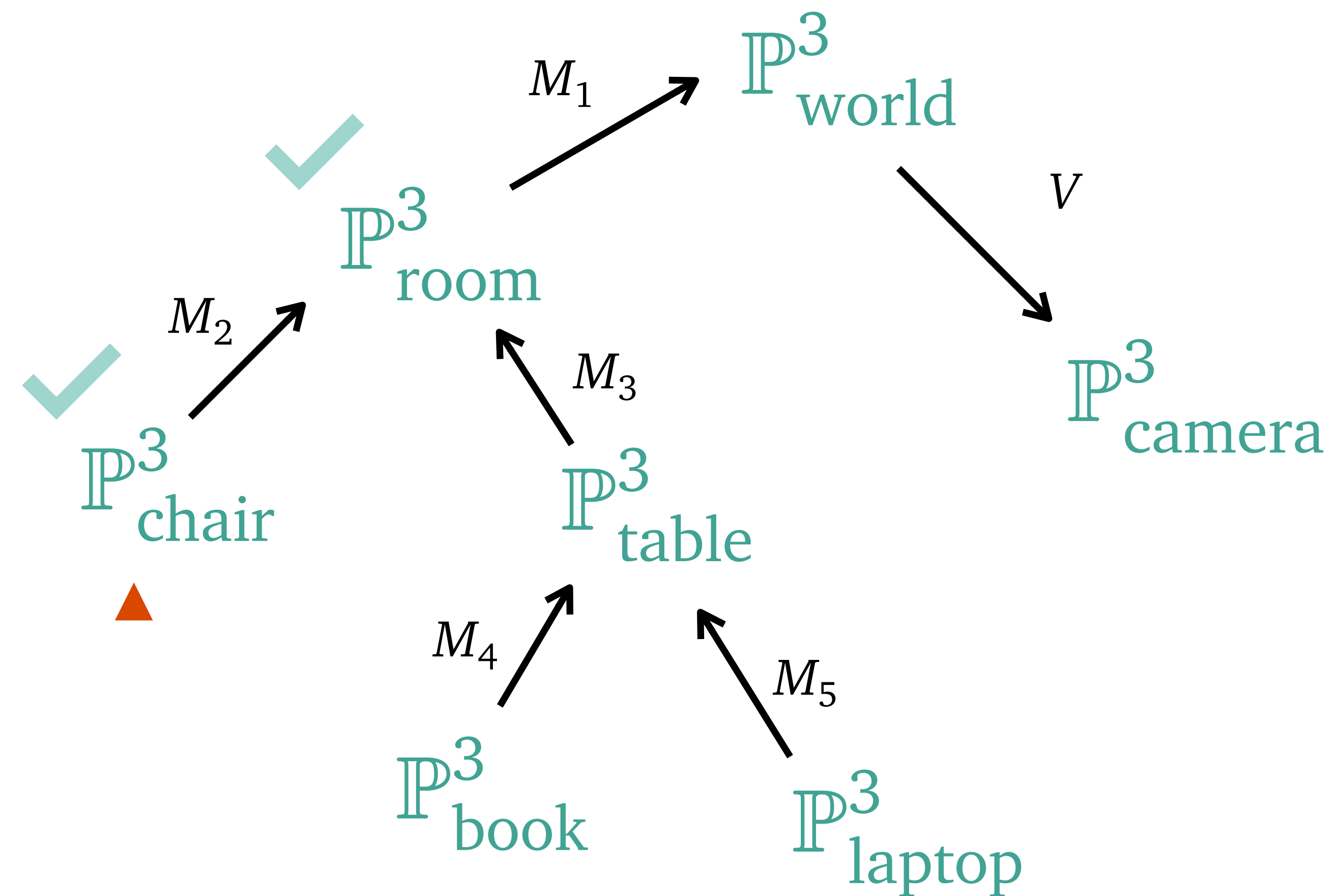
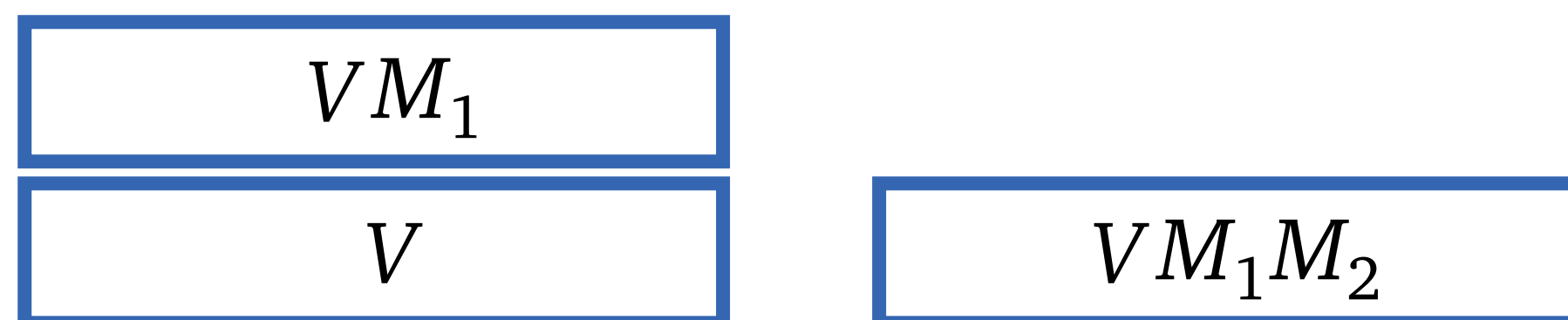
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair



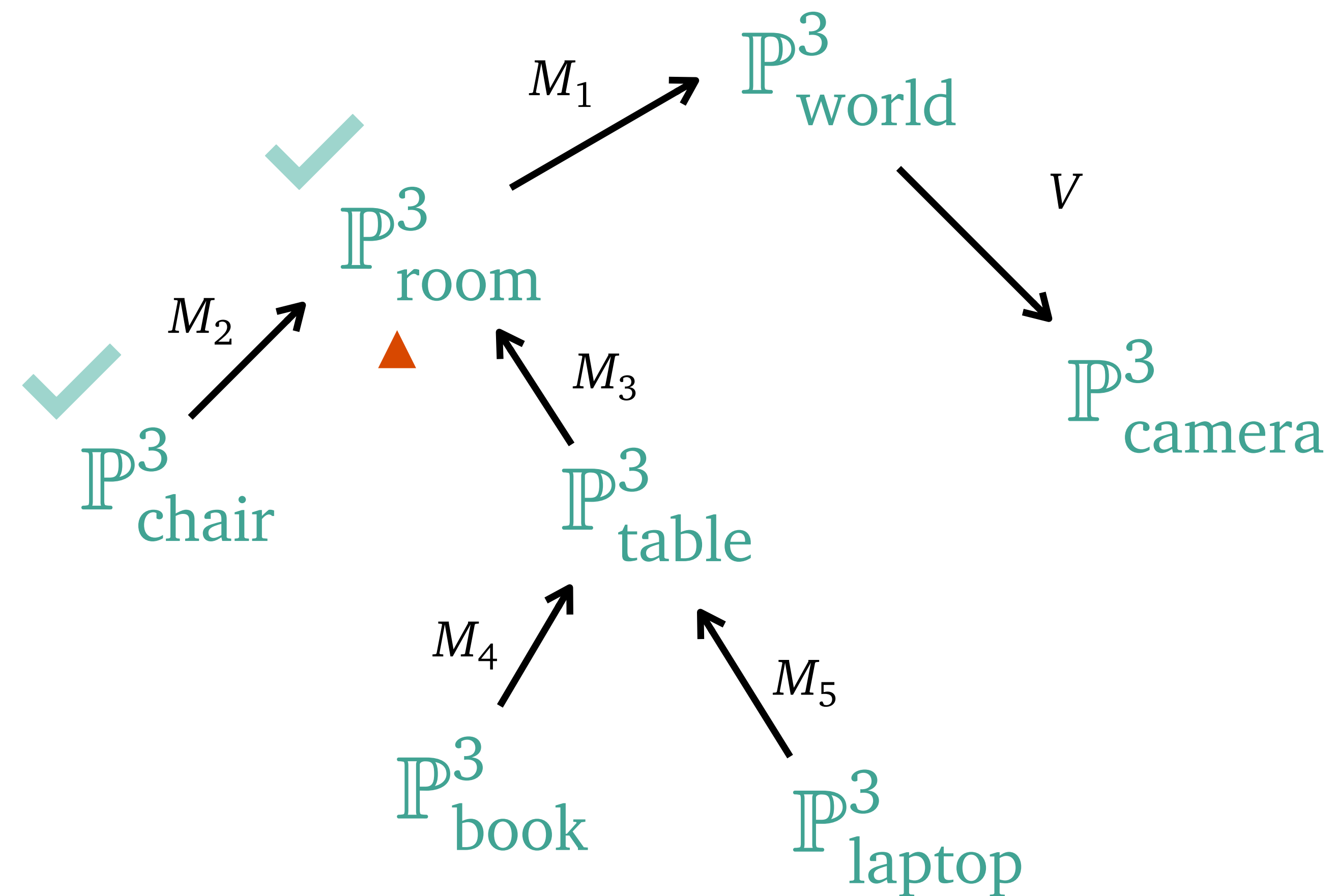
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$



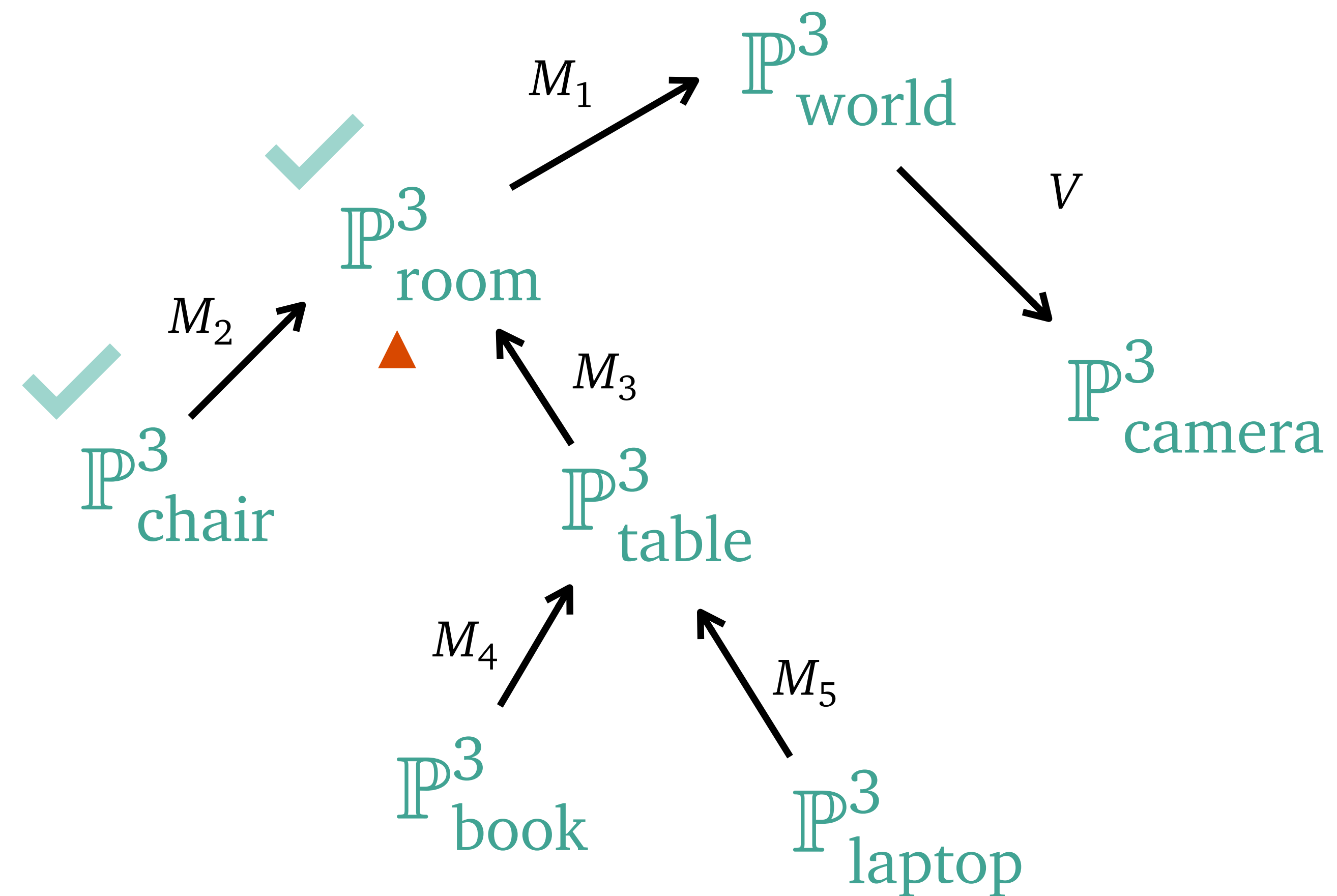
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$



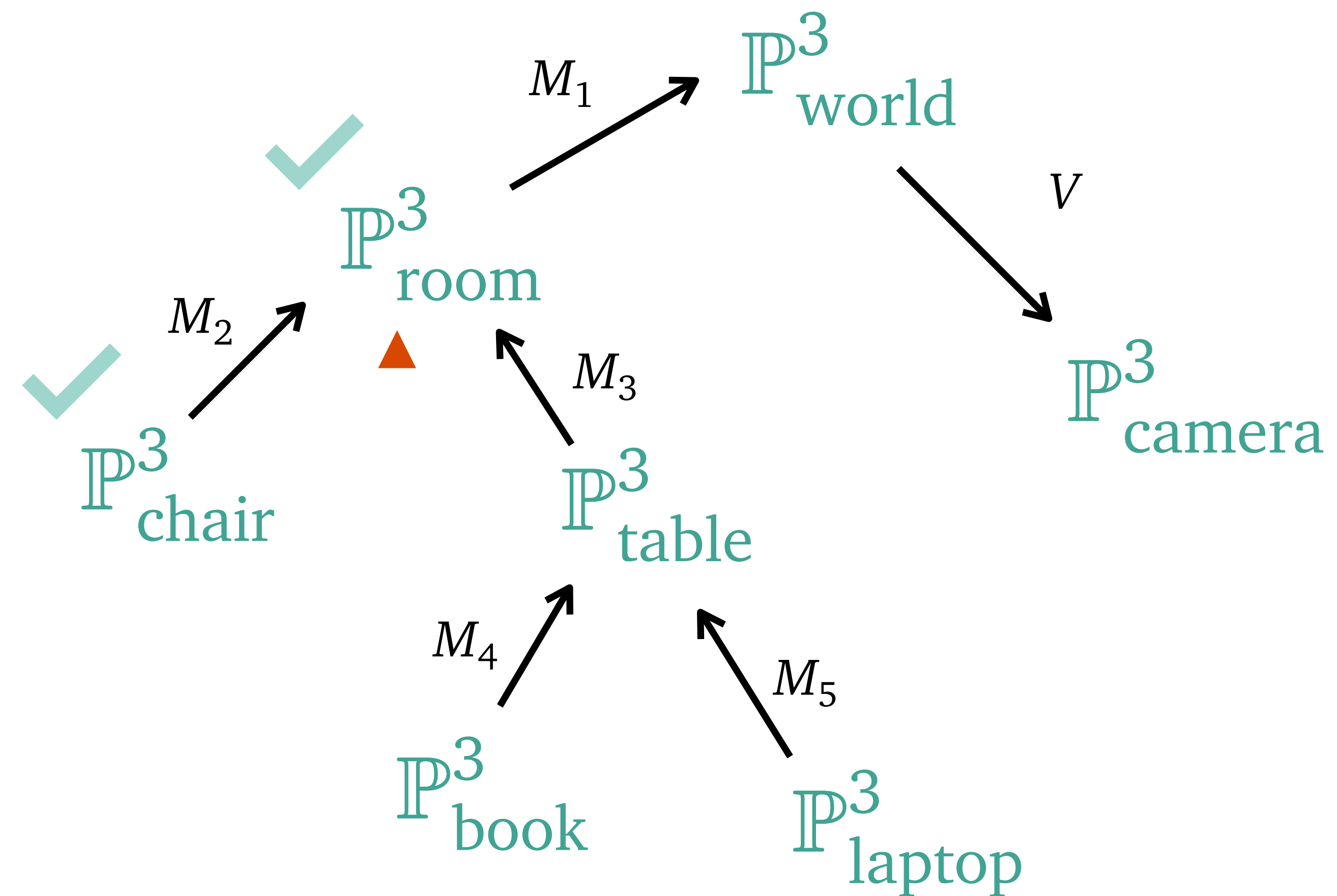
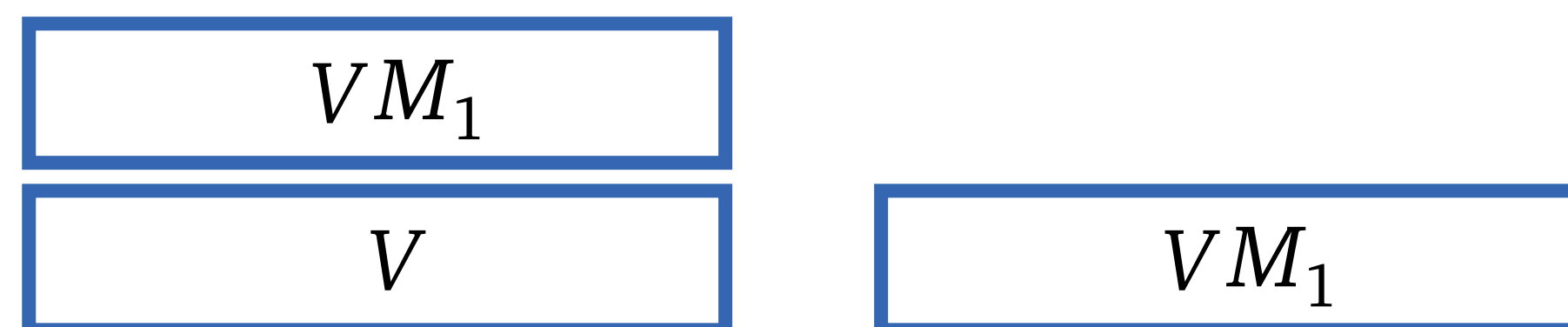
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$



Matrix Stack

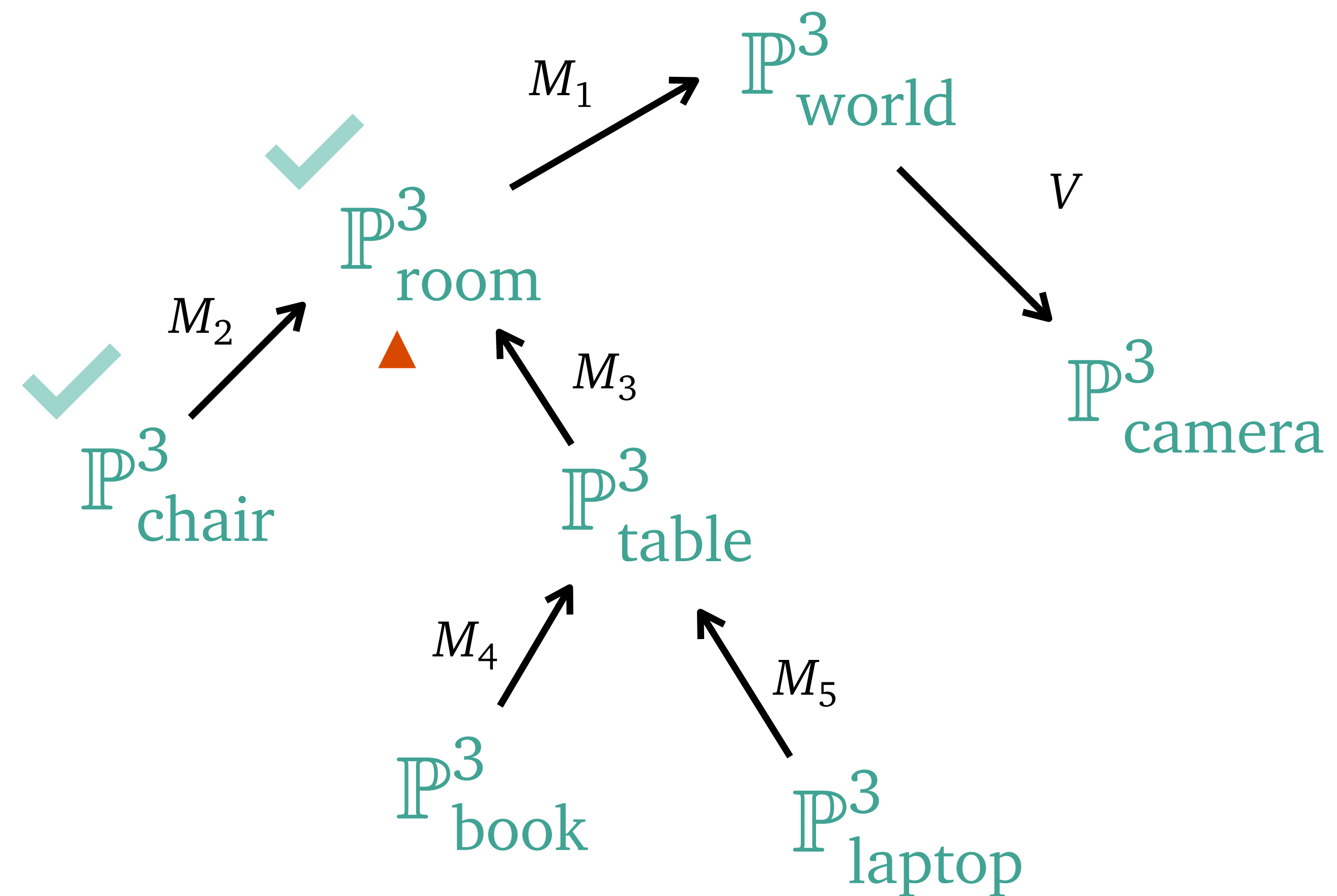
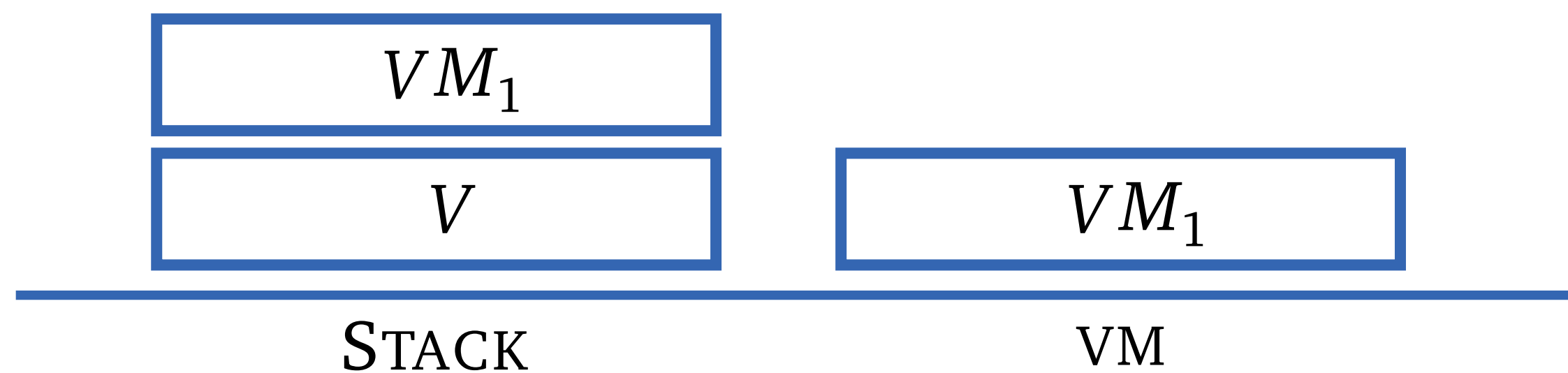
- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$



Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$

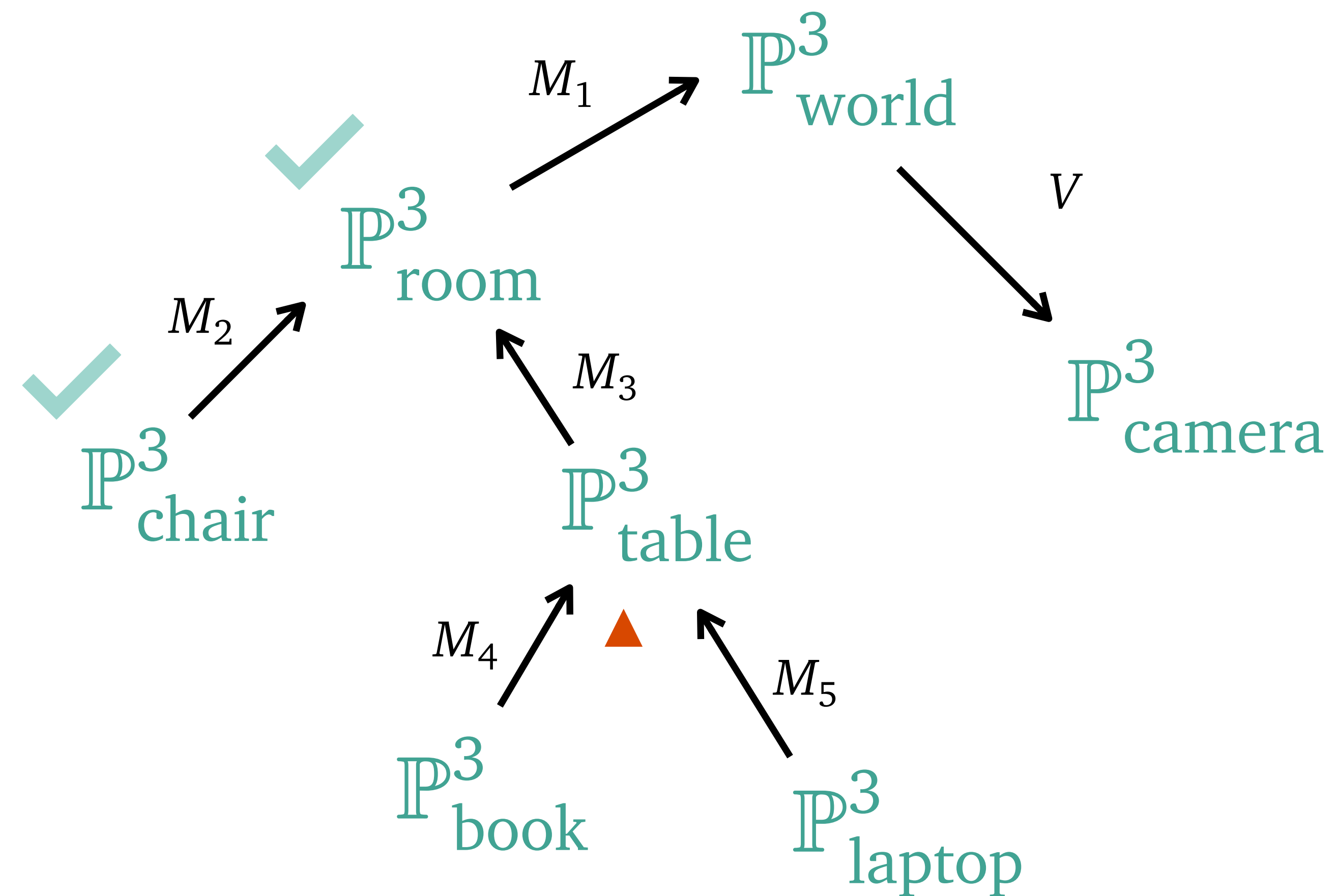
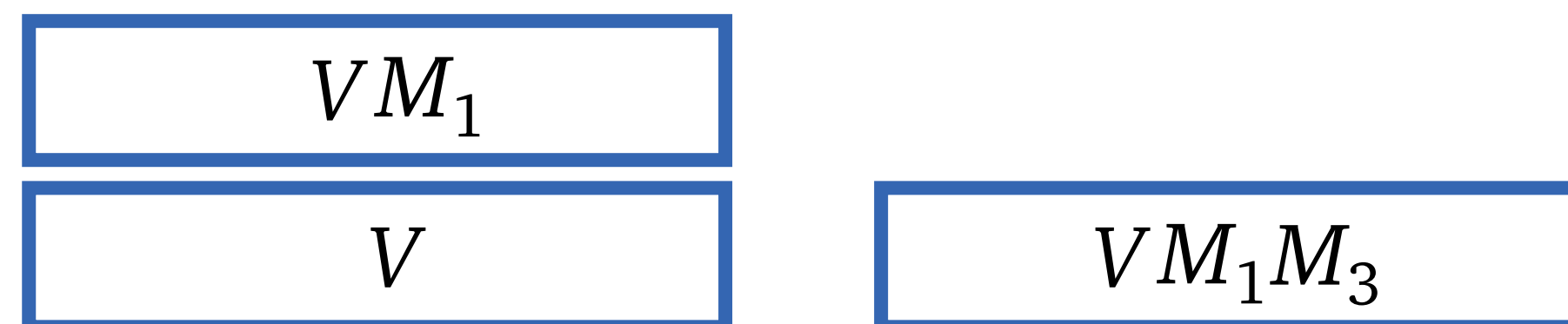
- ▶ Update $VM = VM * M_3$



Matrix Stack

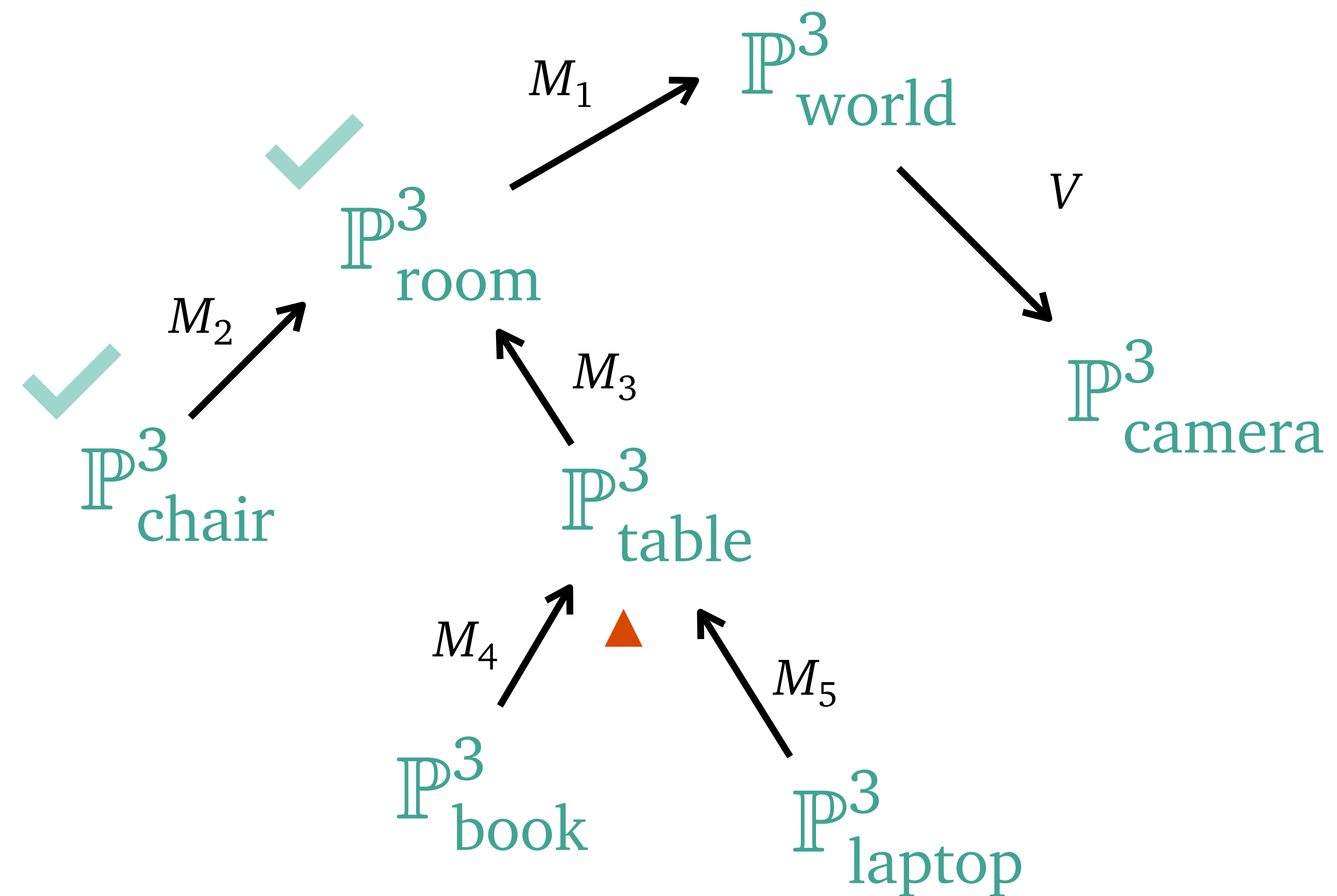
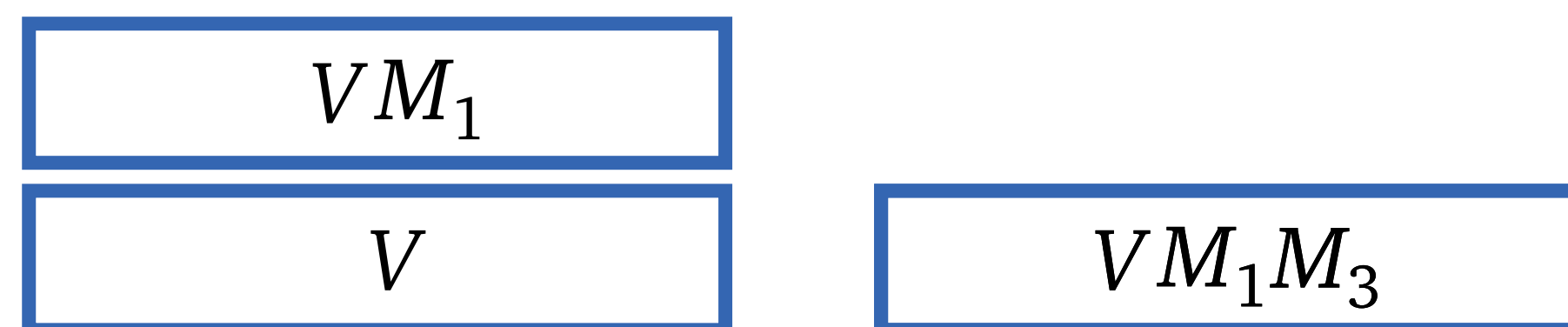
- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$

- ▶ Update $VM = VM * M_3$



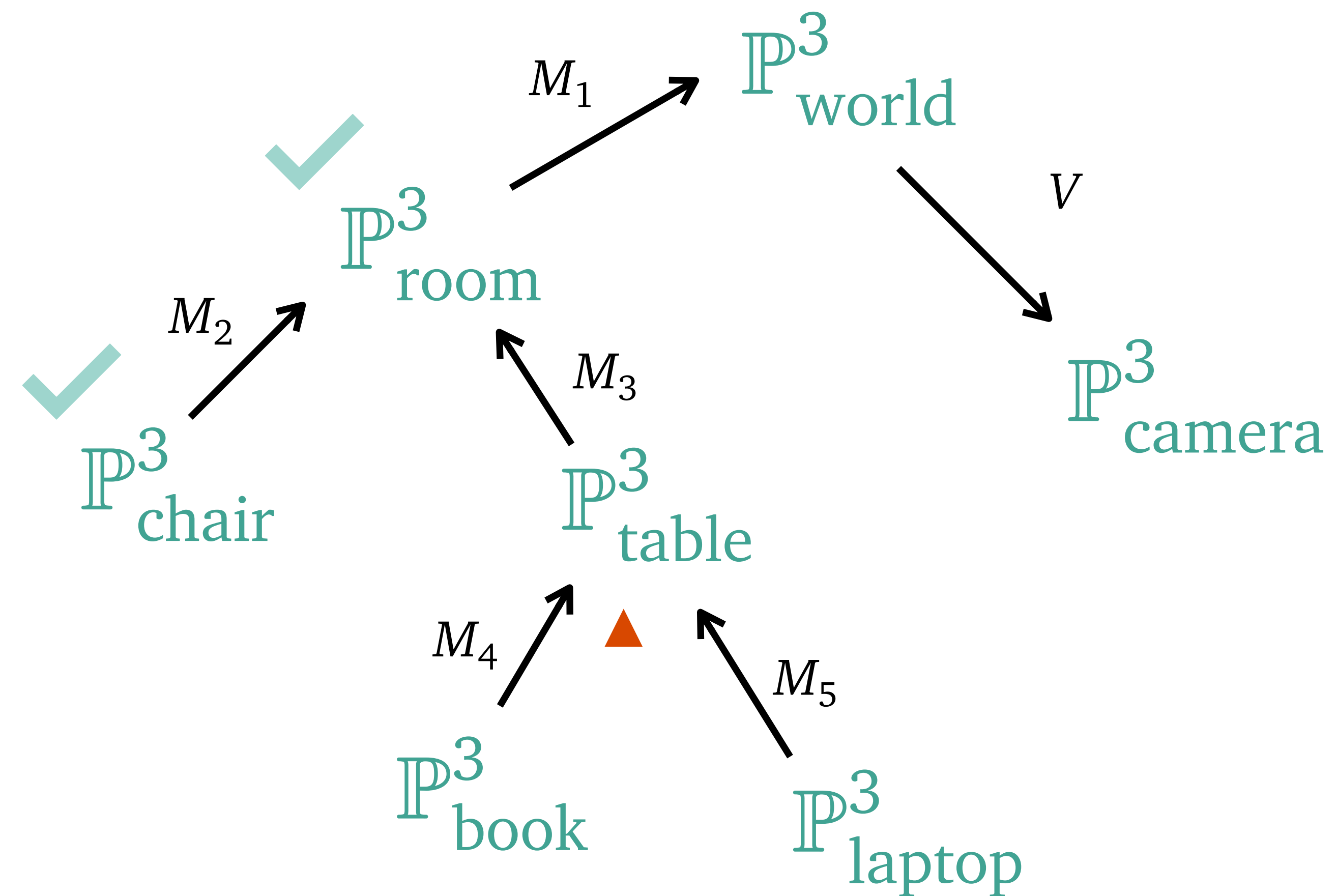
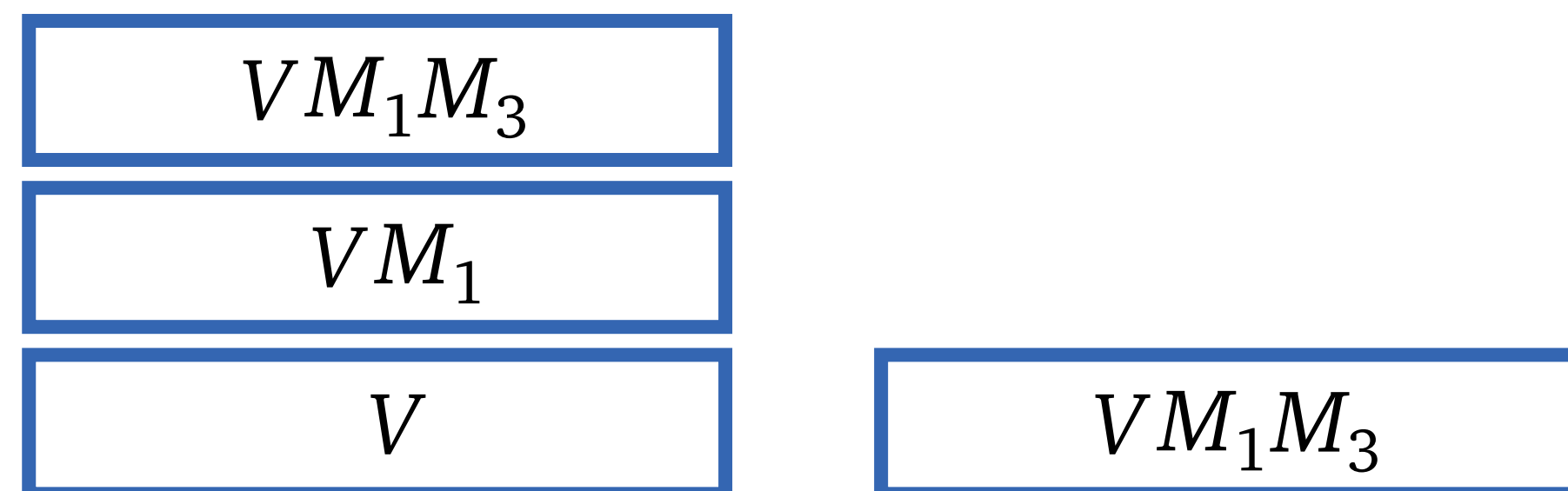
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$



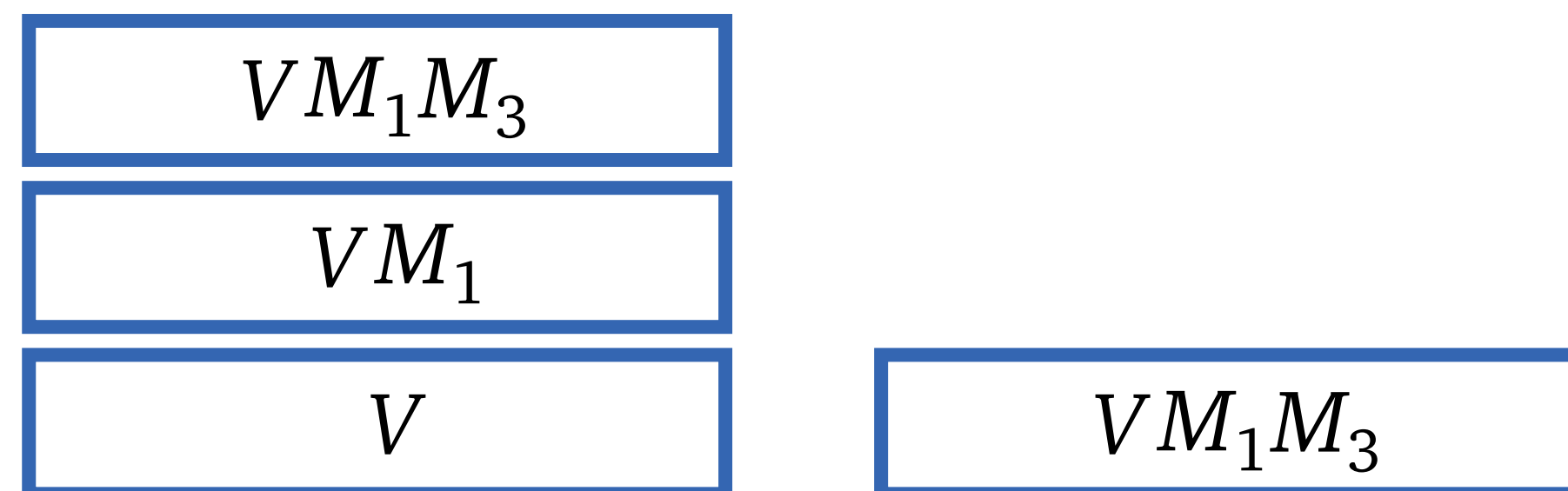
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$



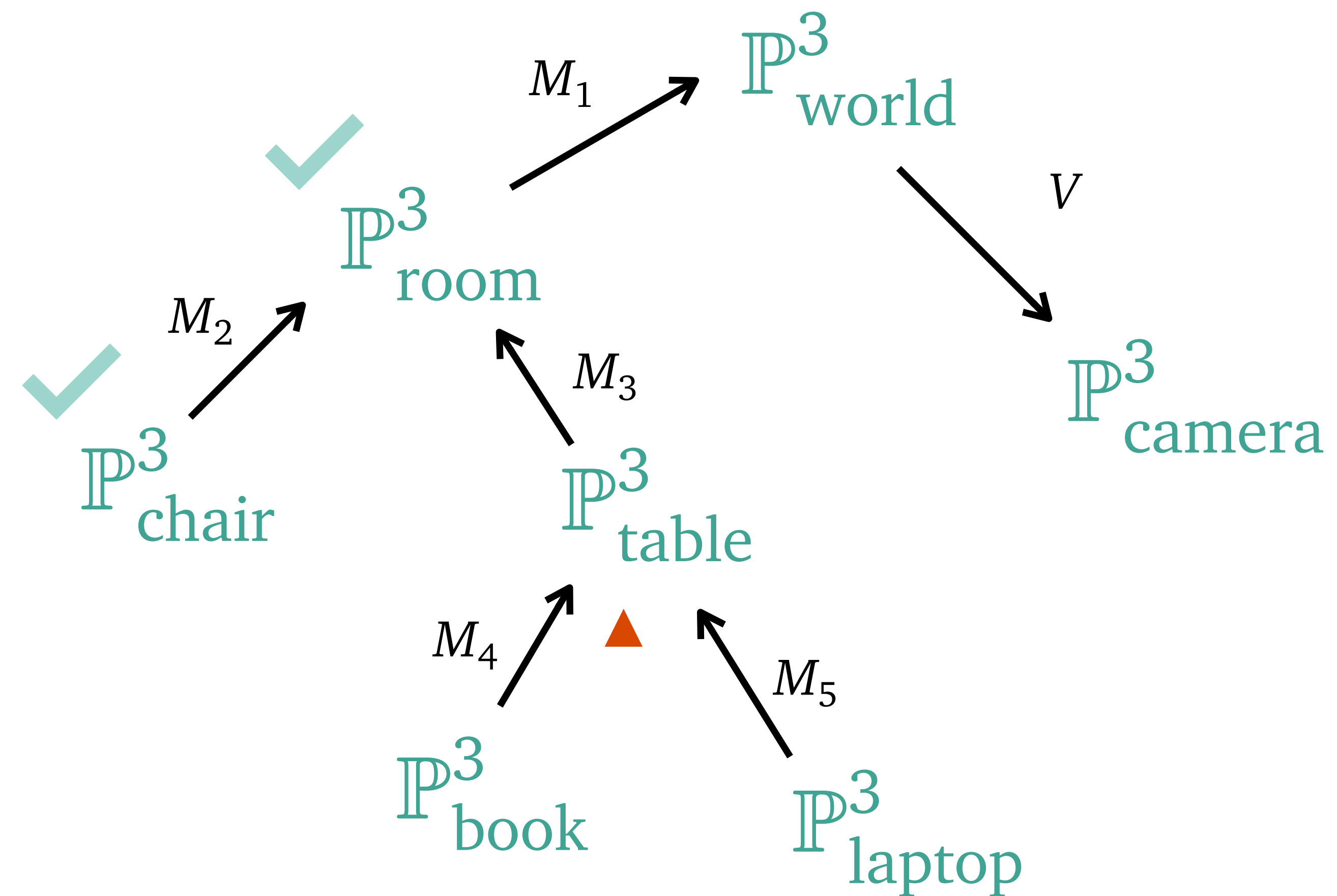
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$



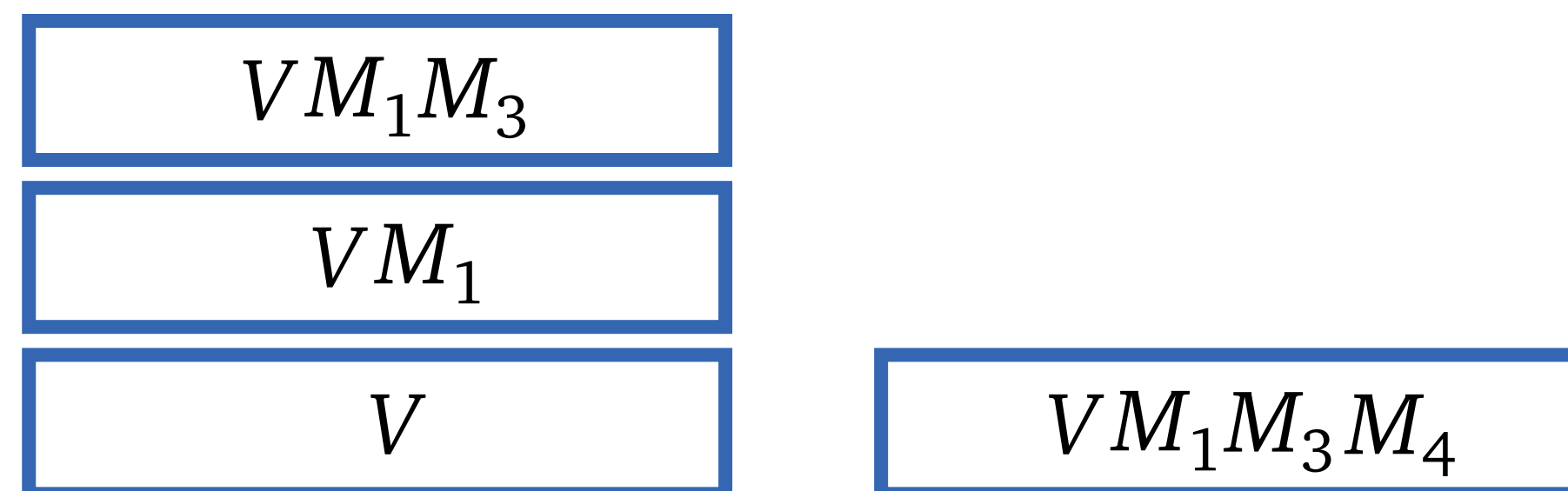
STACK

VM



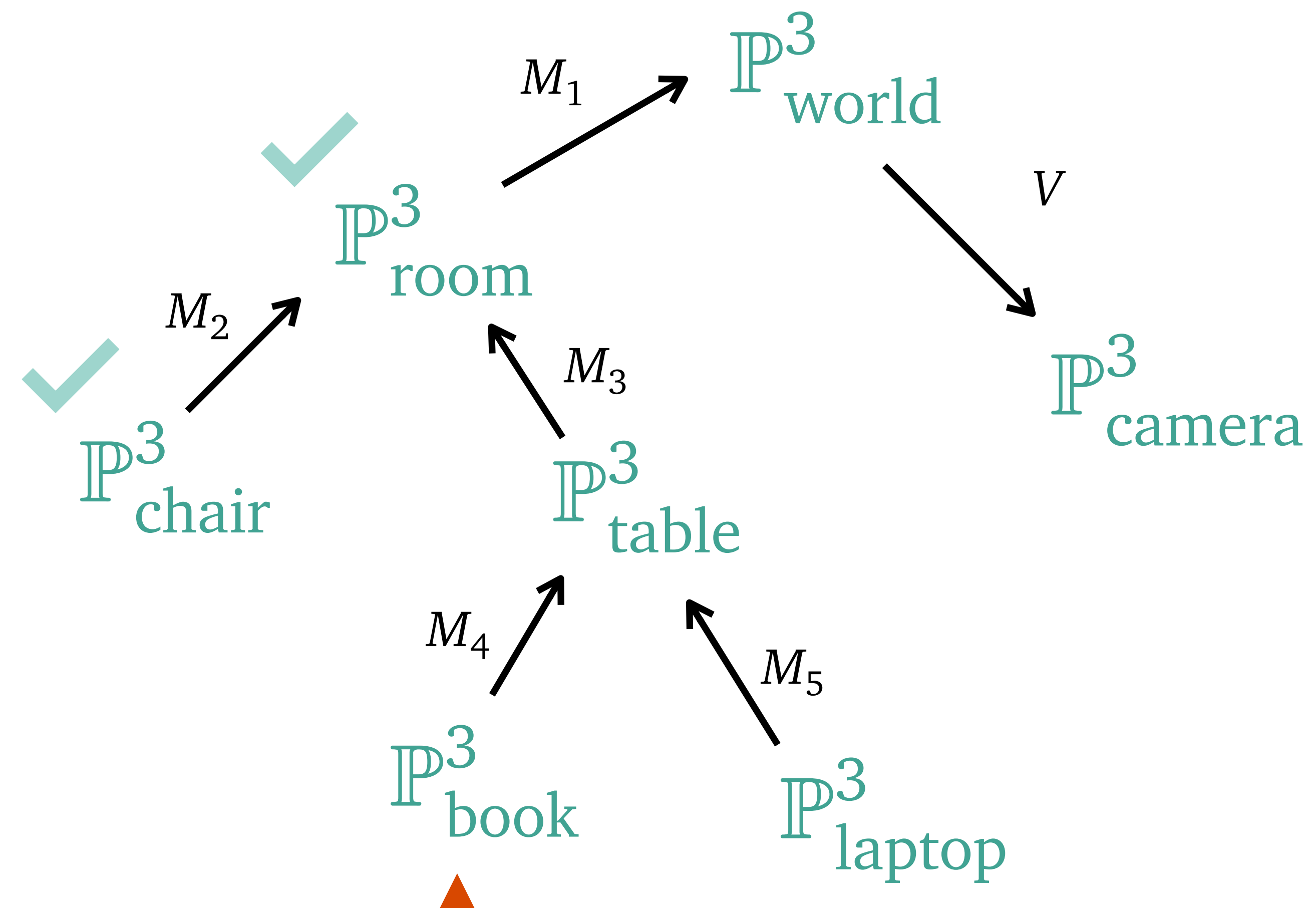
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$



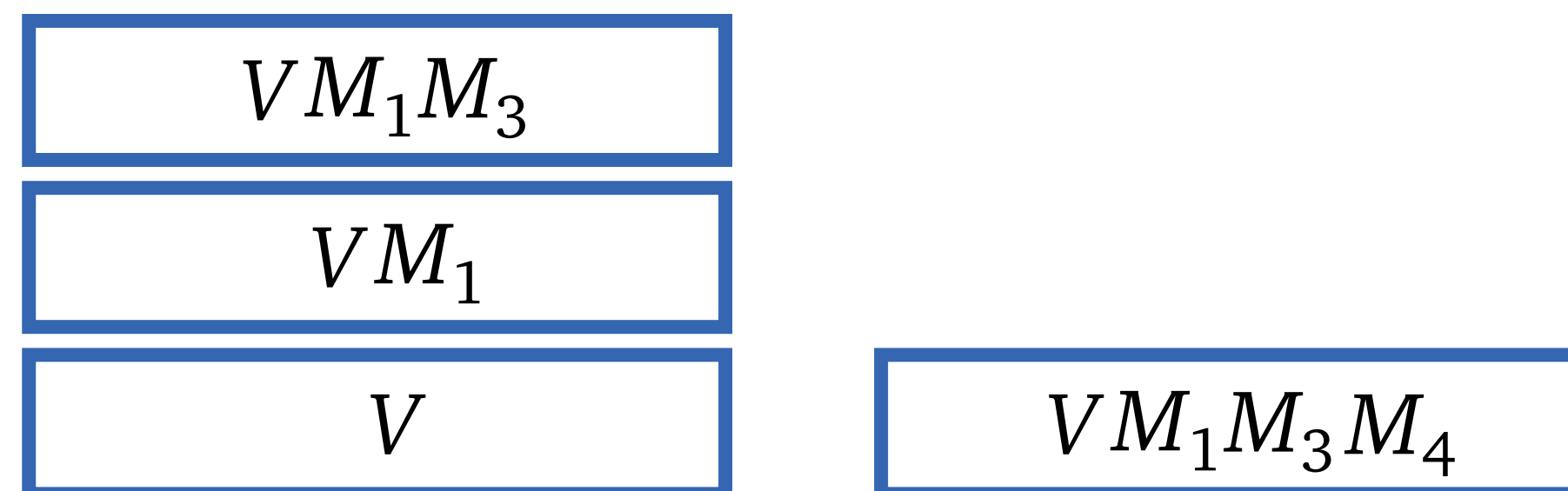
STACK

VM



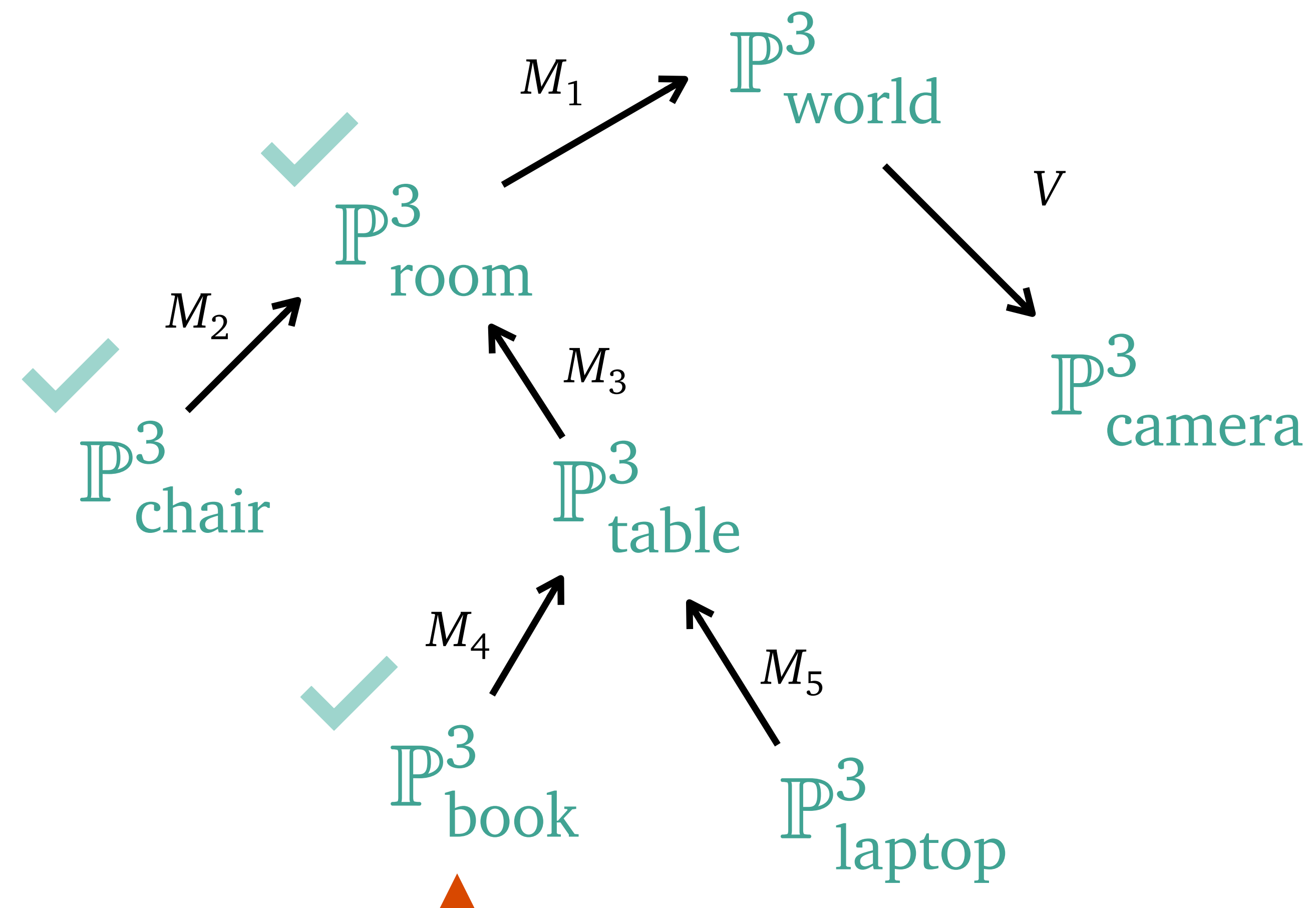
Matrix Stack

- Initially $VM = V$
- Push $STACK.PUSH(VM)$
- Update $VM = VM * M_1$
- drawRoom
- Push $STACK.PUSH(VM)$
- Update $VM = VM * M_2$
- drawChair
- Pop $VM = STACK.POP$
- Push $STACK.PUSH(VM)$
- Update $VM = VM * M_3$
- Push $STACK.PUSH(VM)$
- Update $VM = VM * M_4$
- drawBook



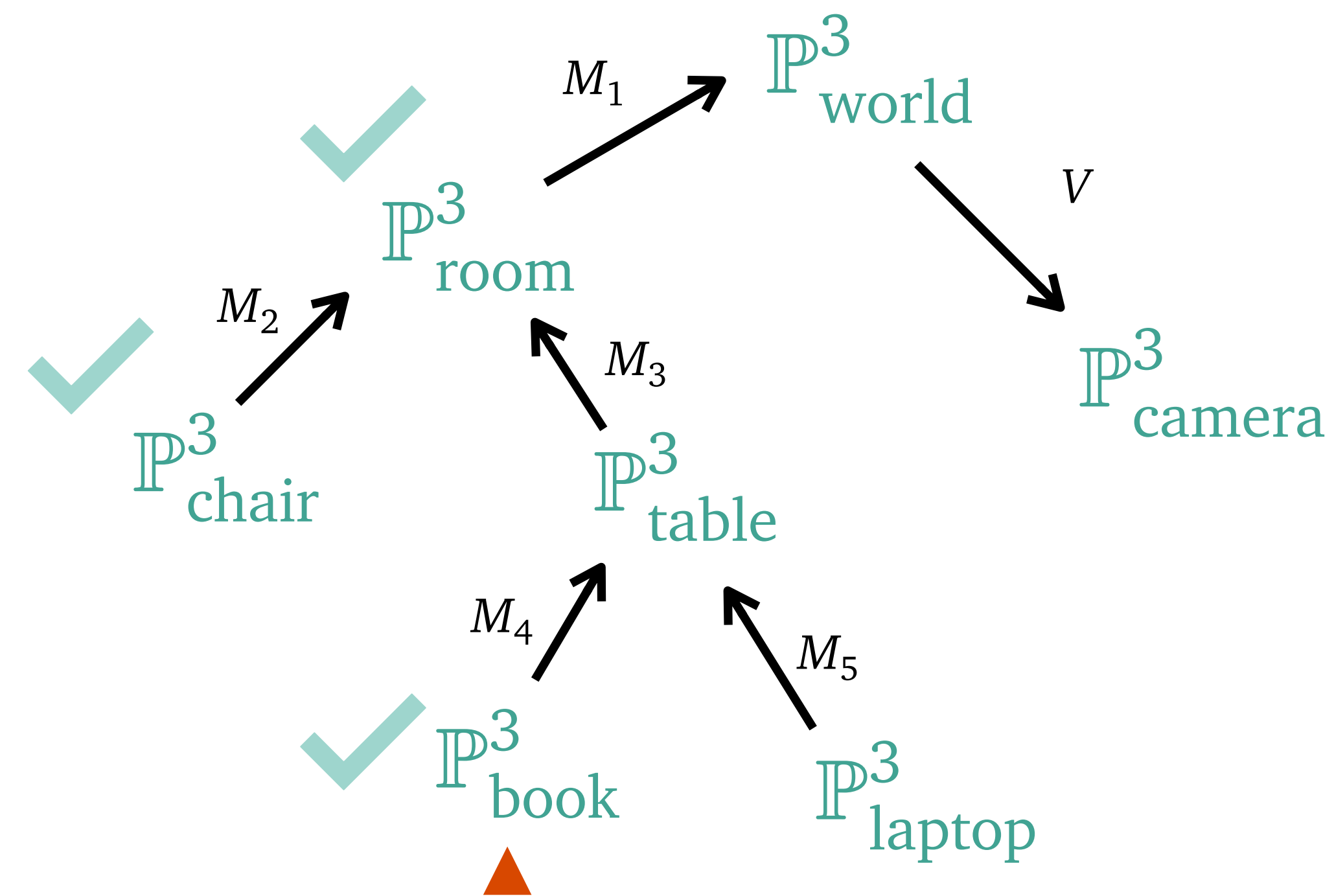
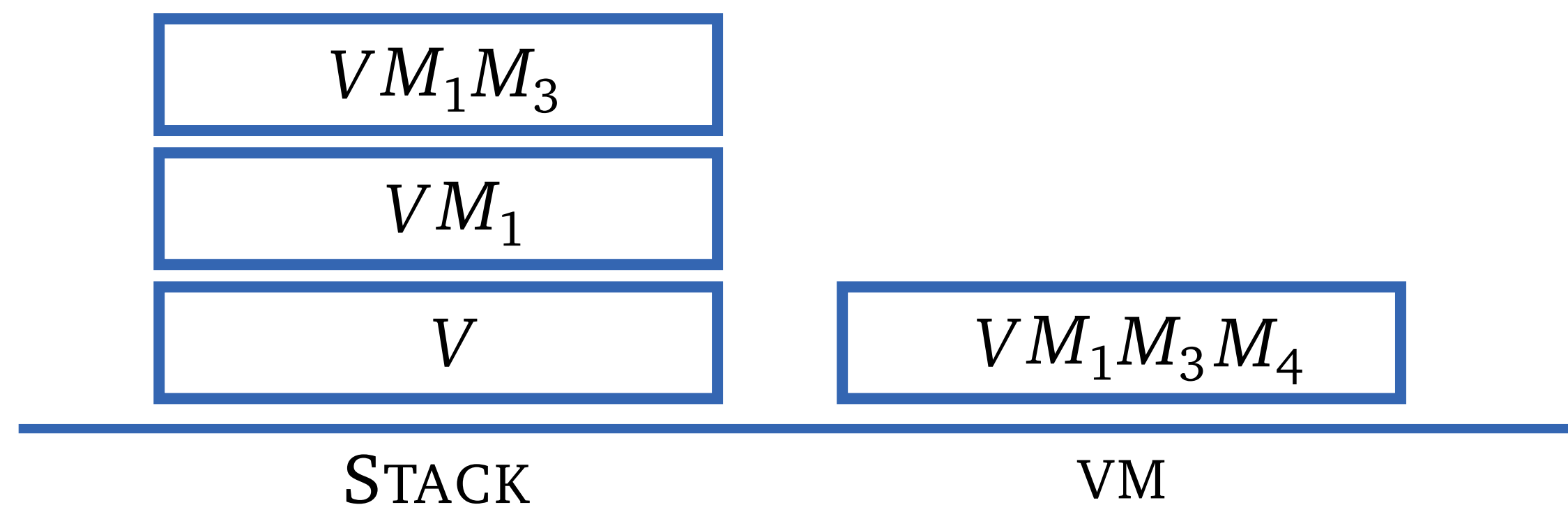
STACK

VM



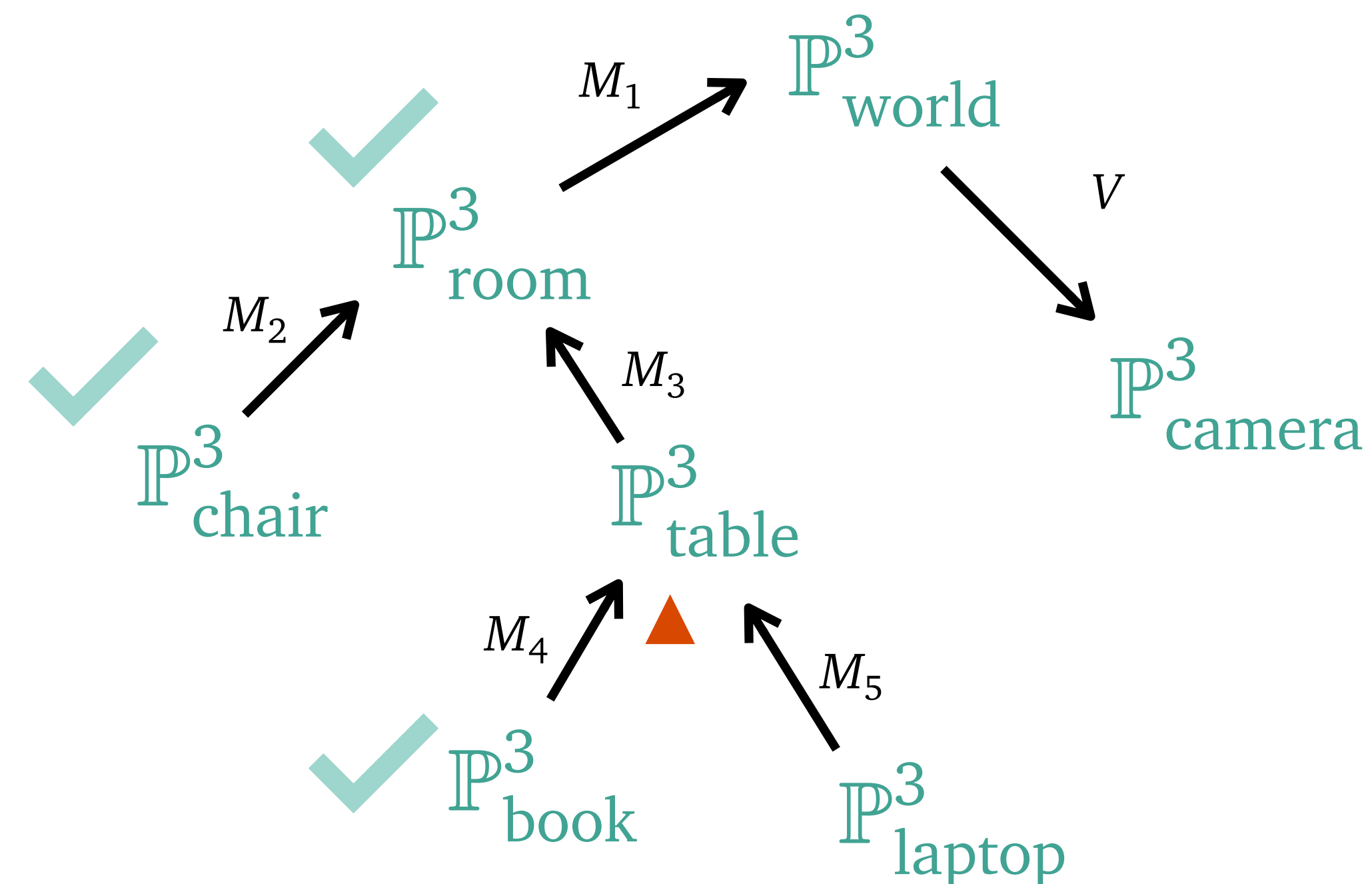
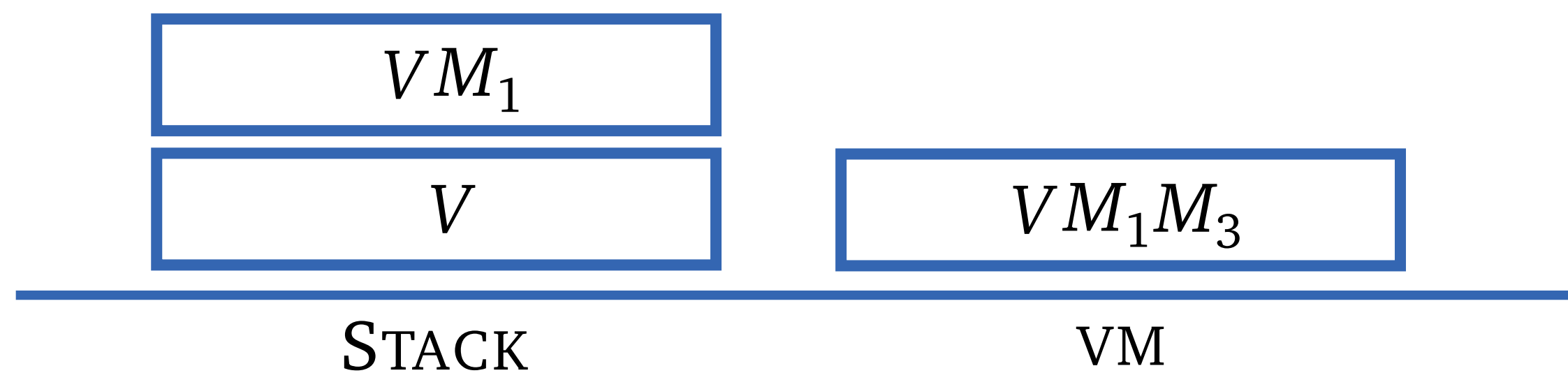
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$



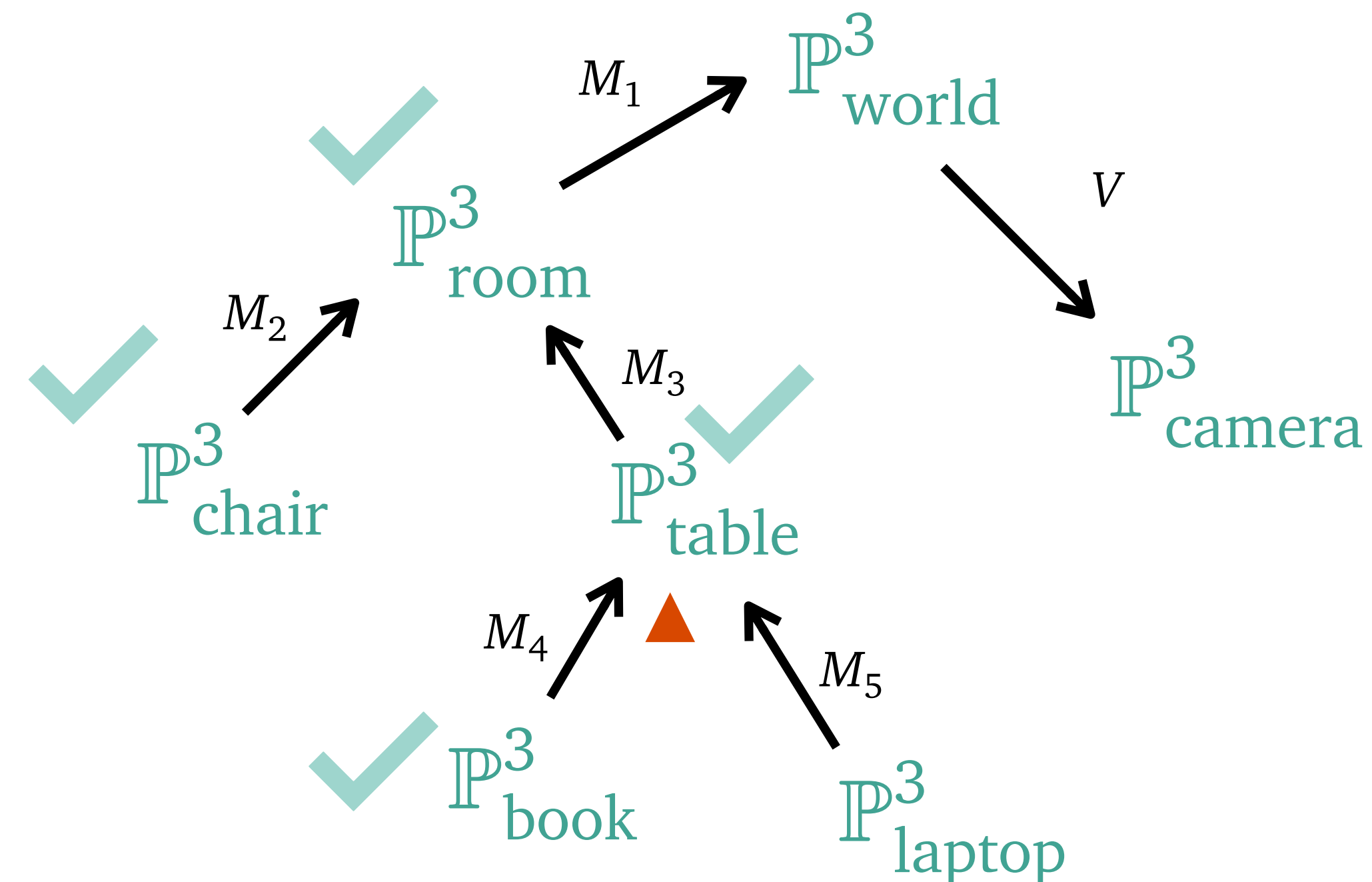
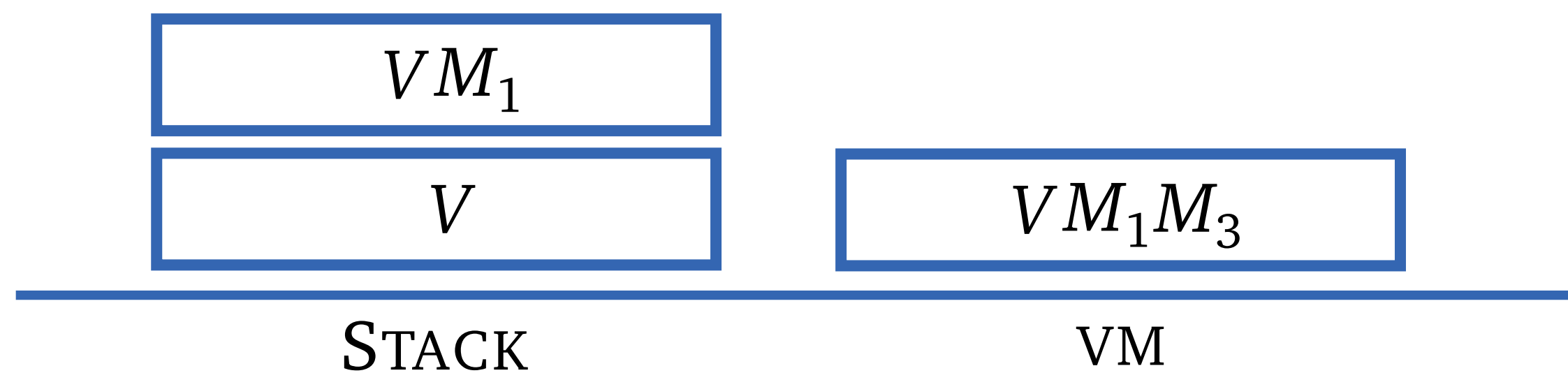
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$



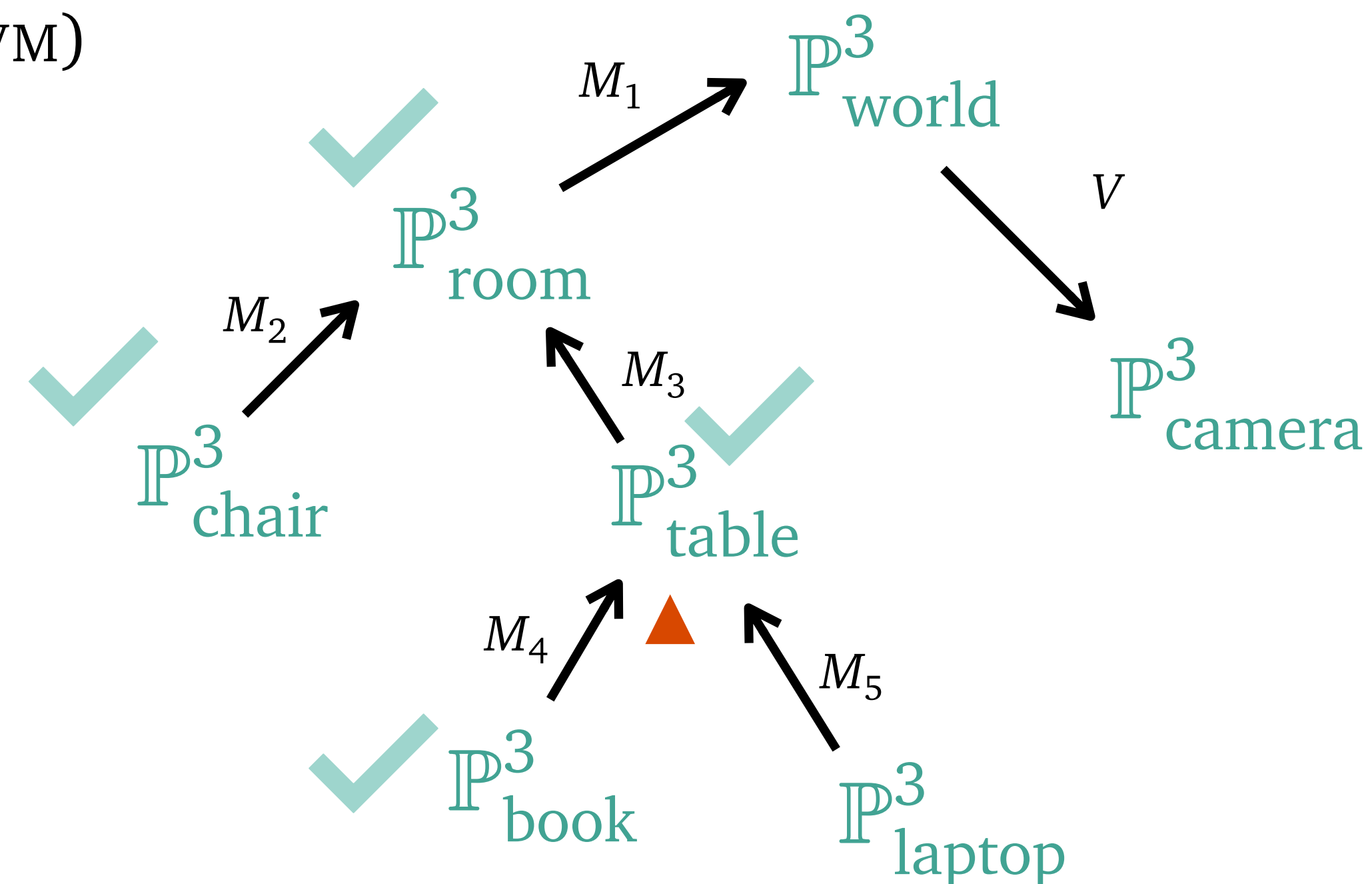
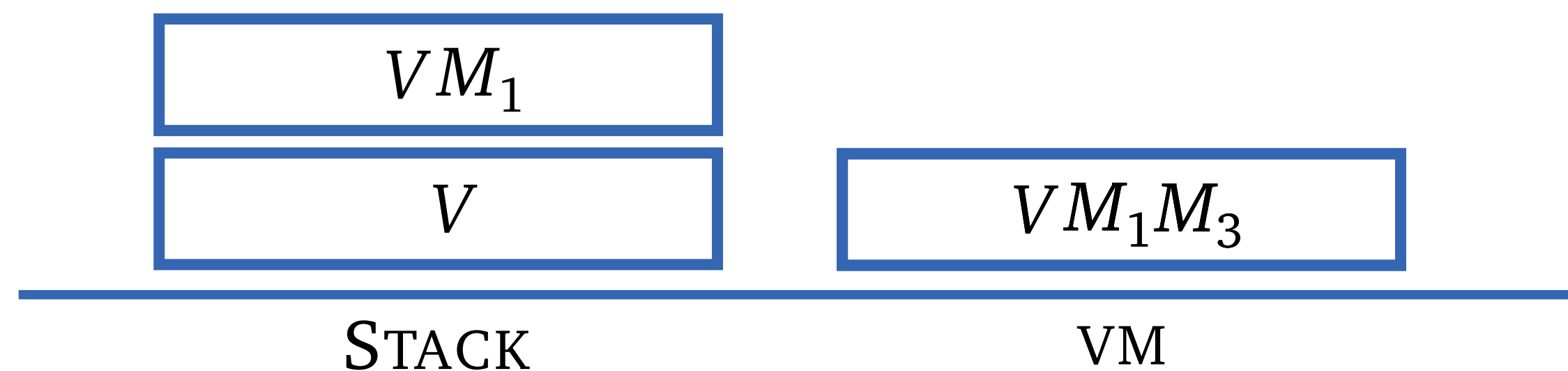
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable



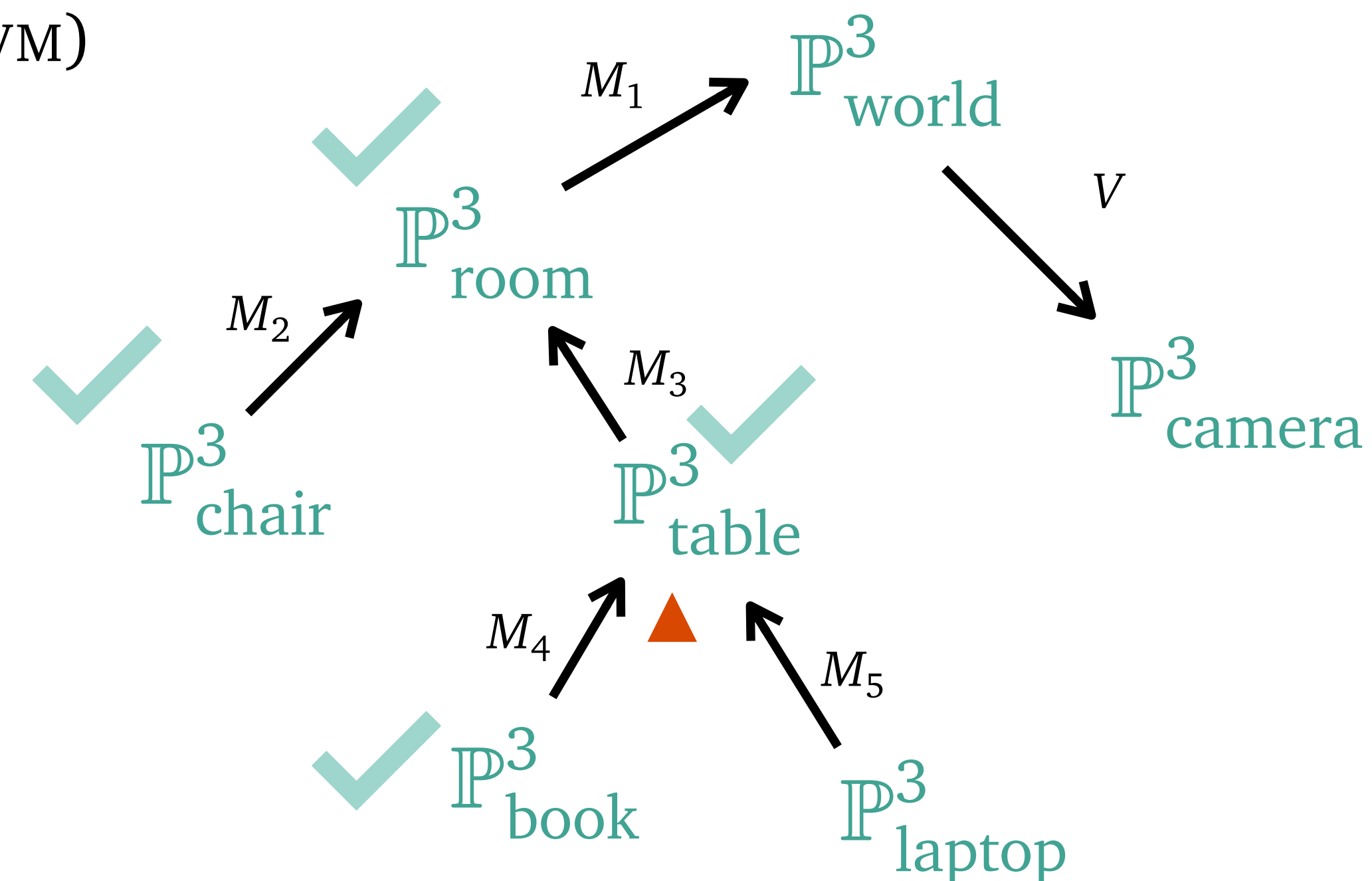
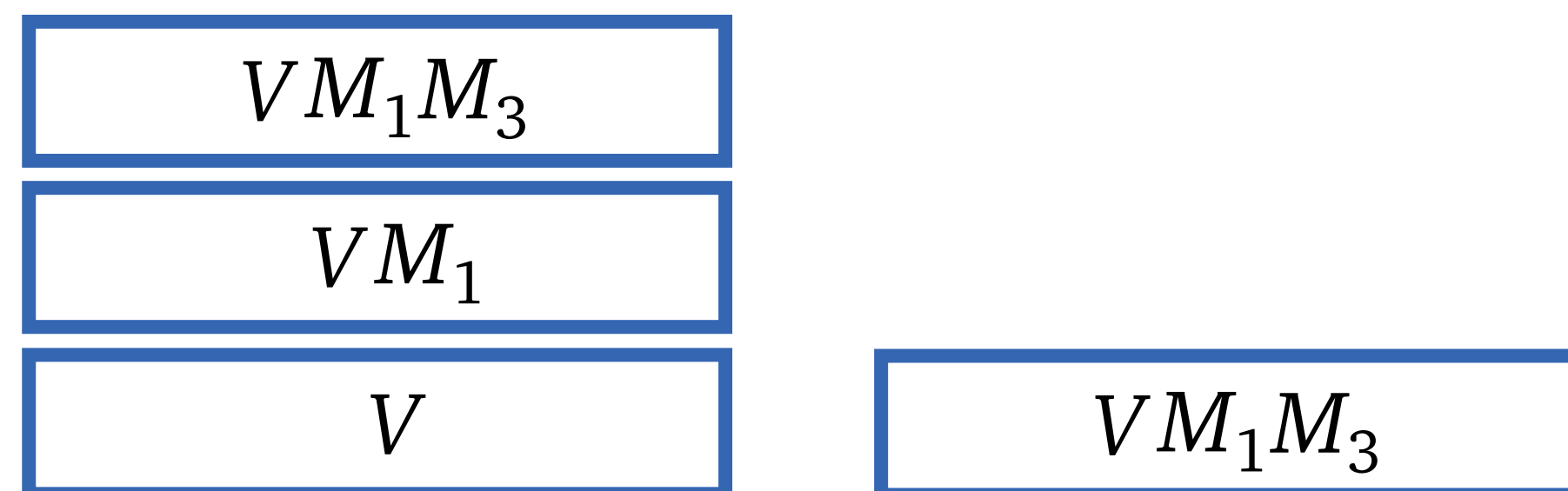
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$



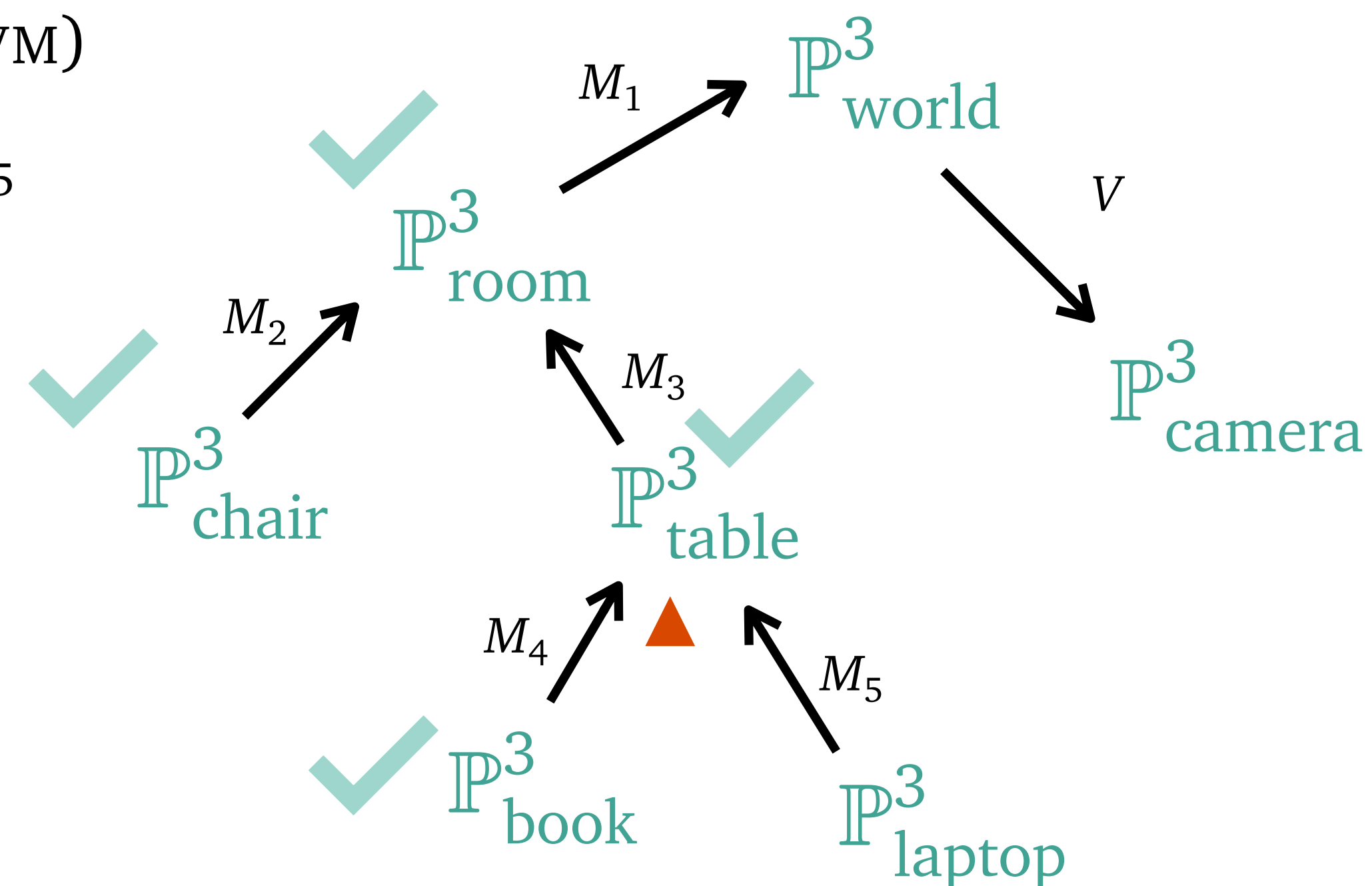
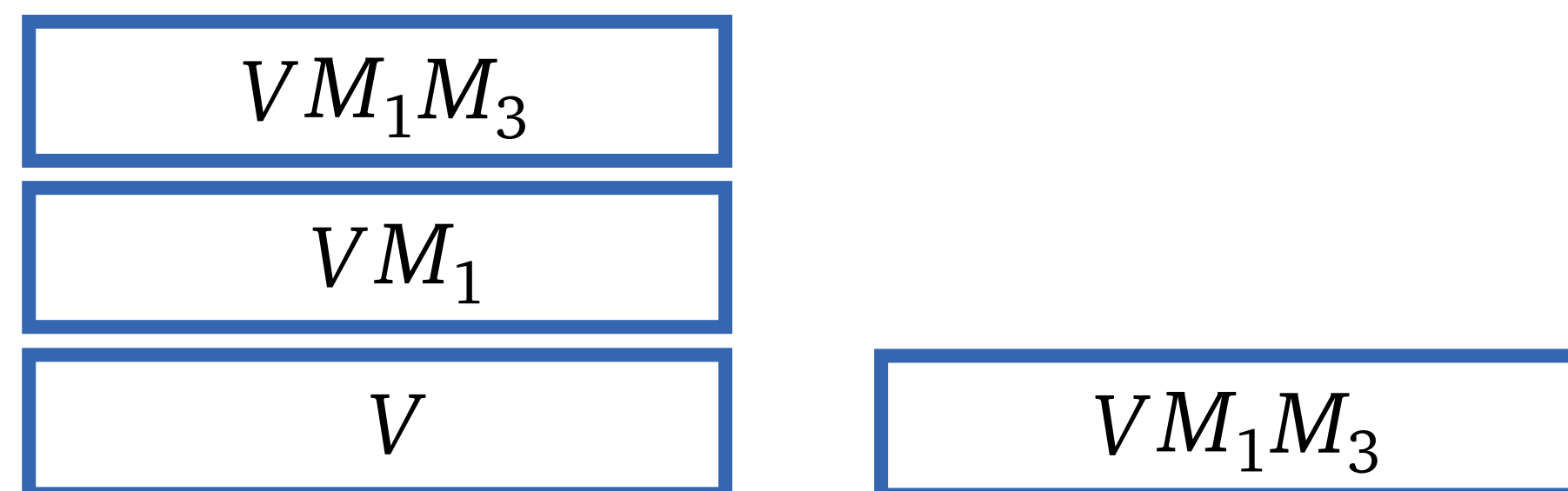
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$



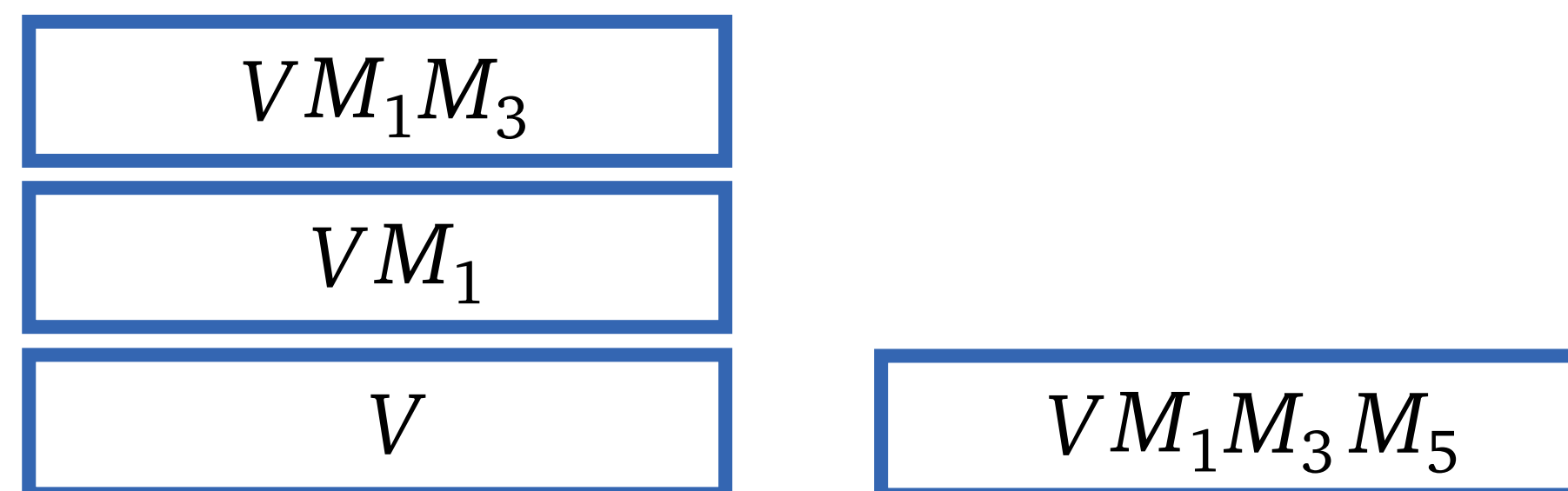
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_5$



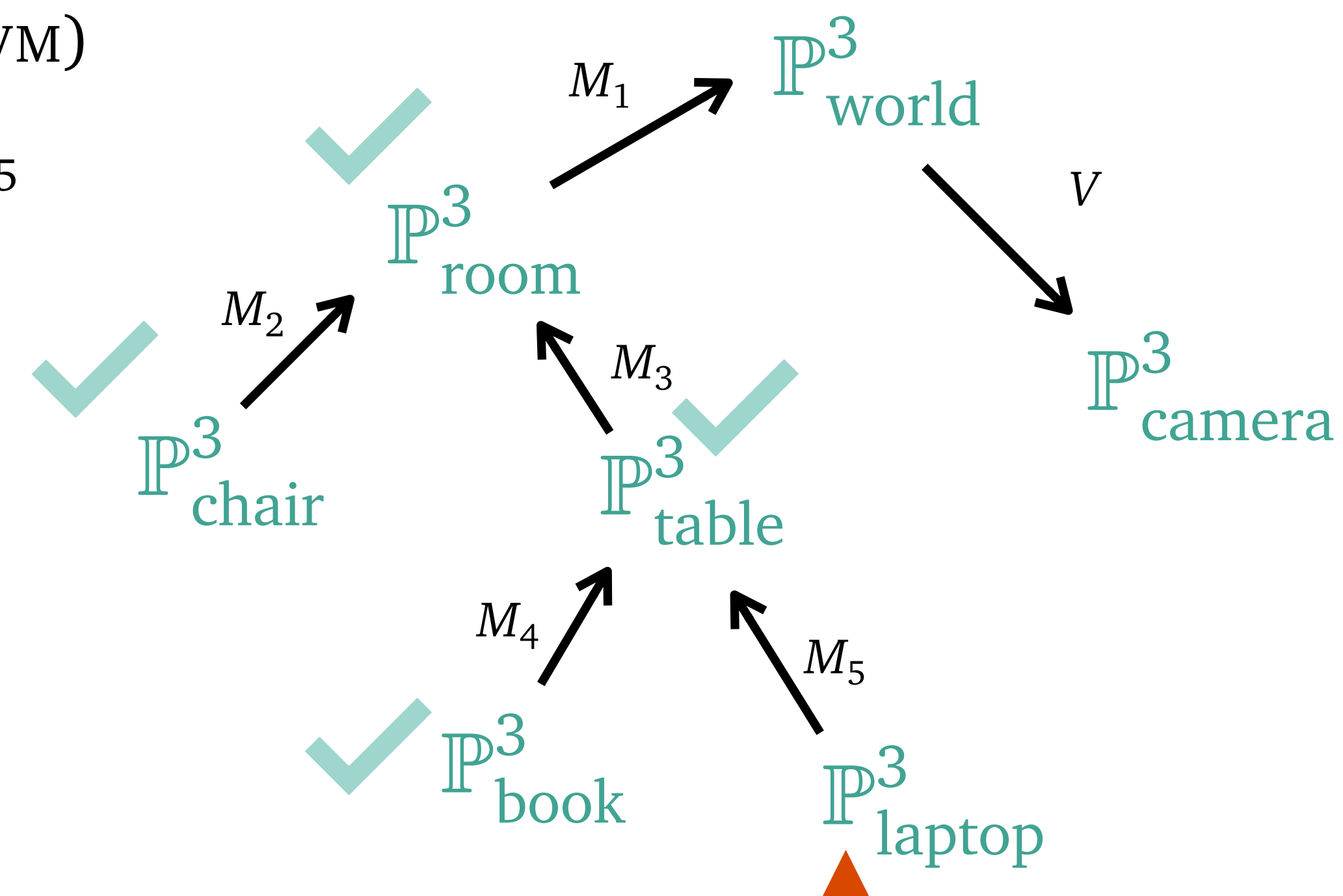
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_5$



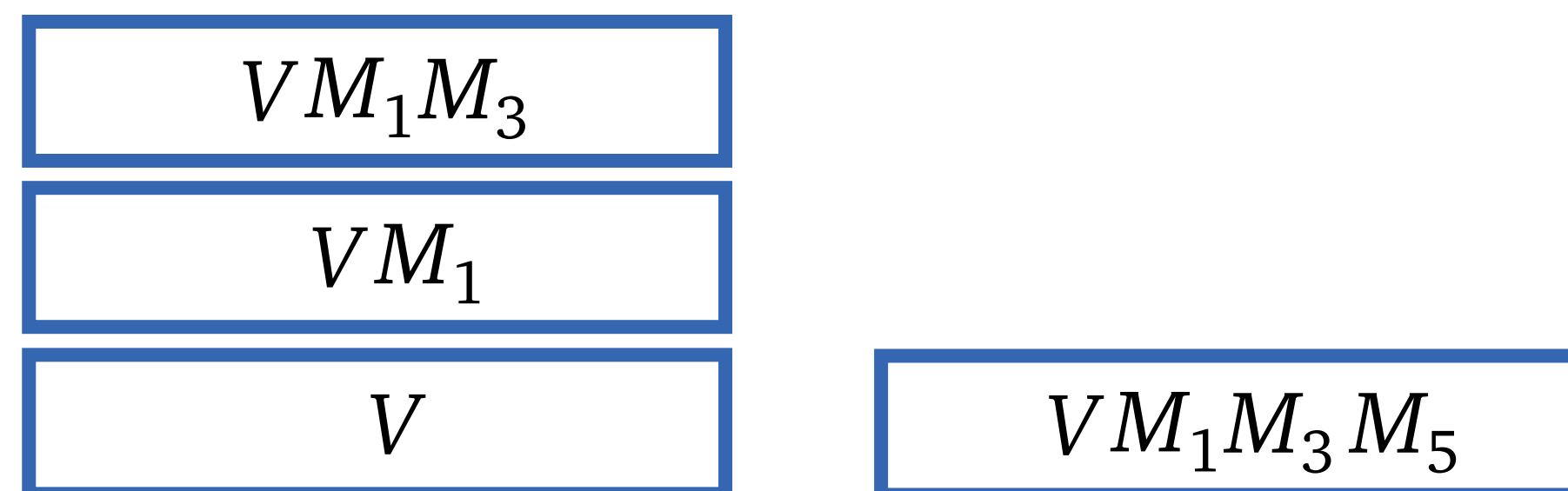
STACK

VM



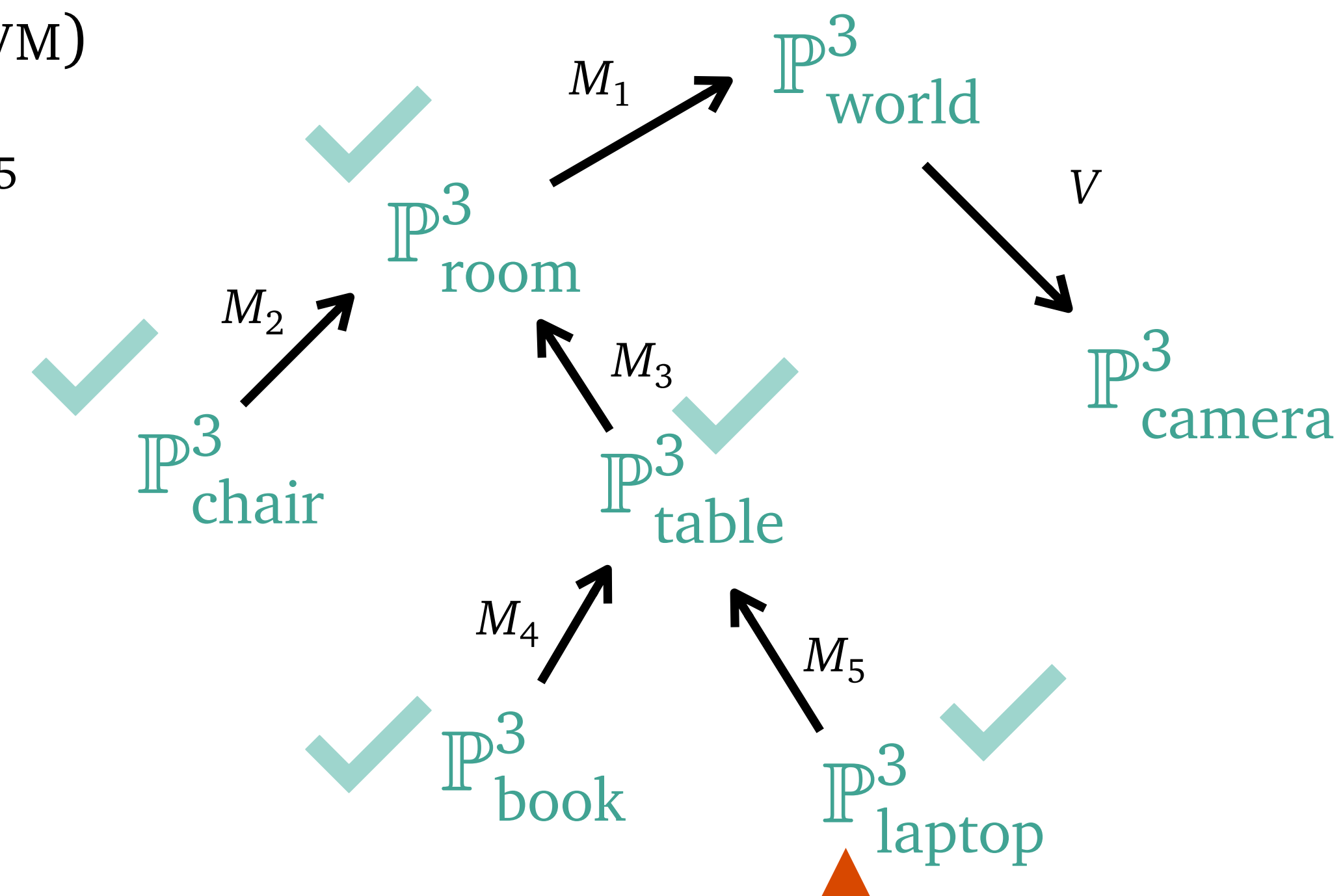
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_5$
- ▶ drawLaptop



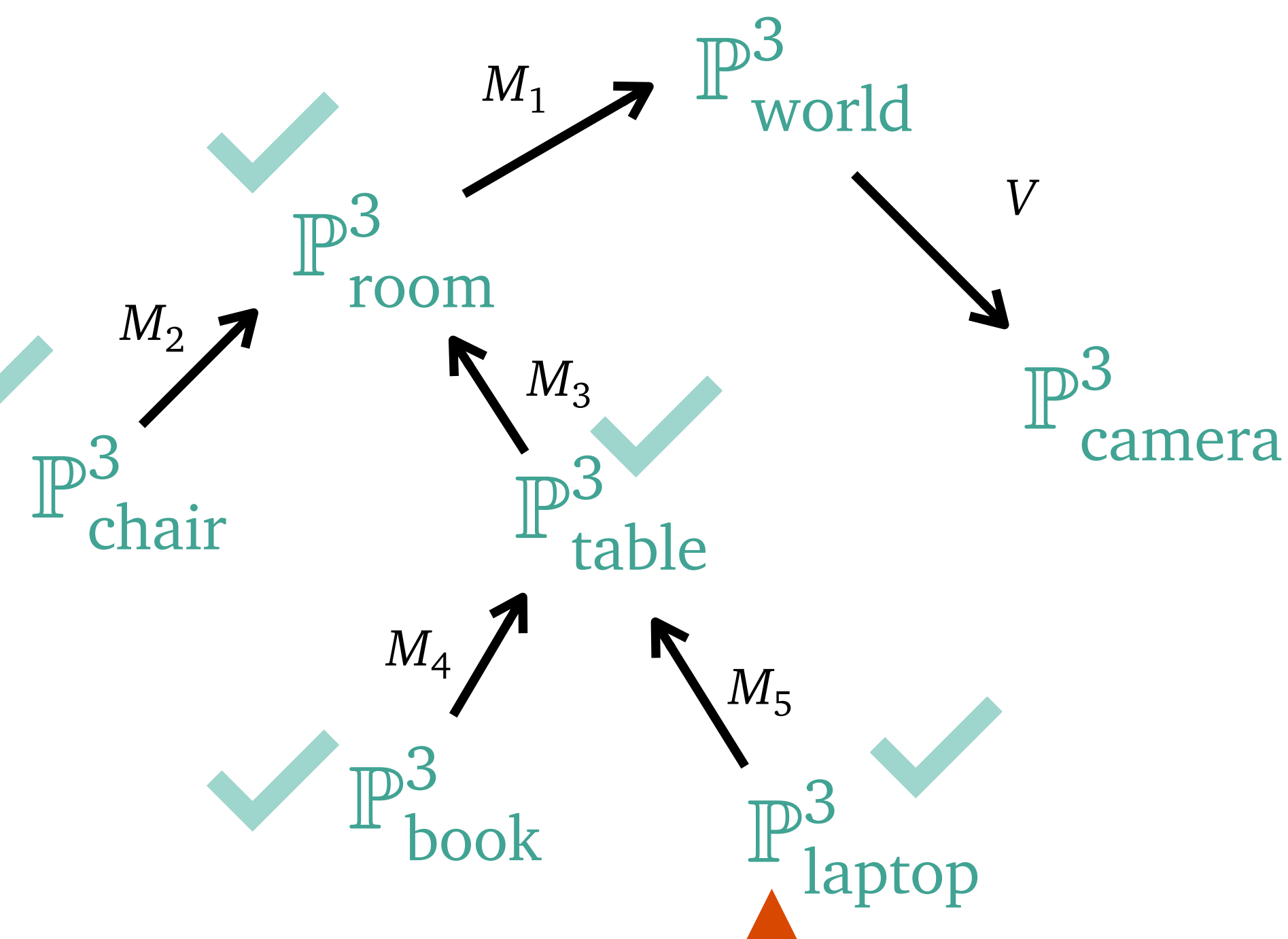
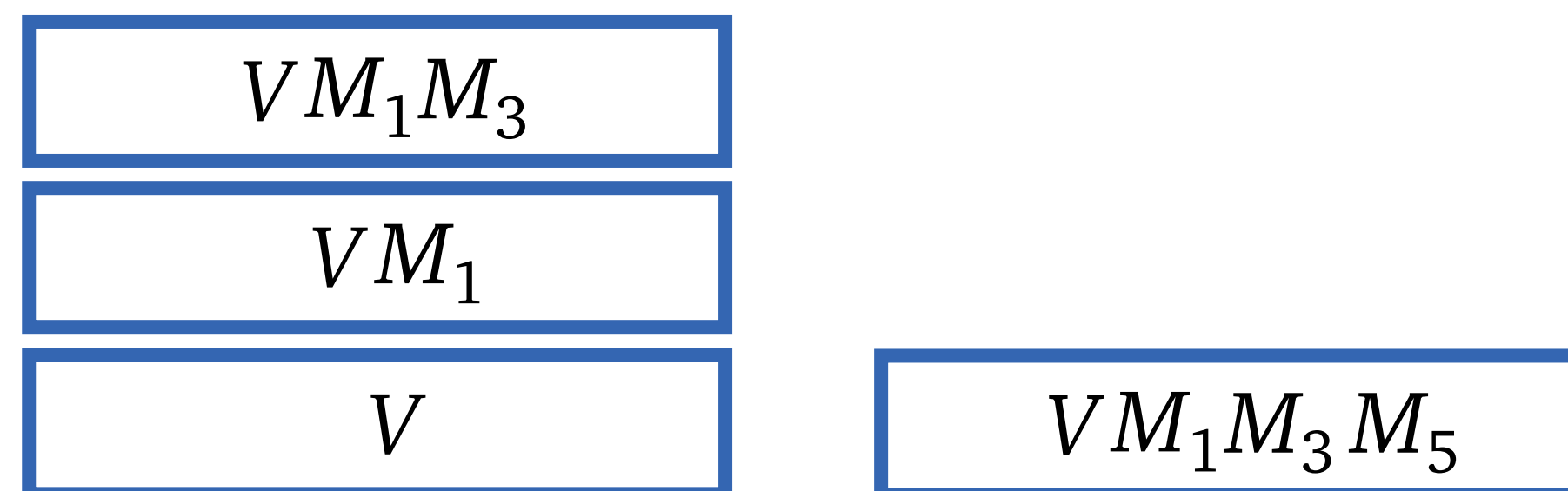
STACK

VM



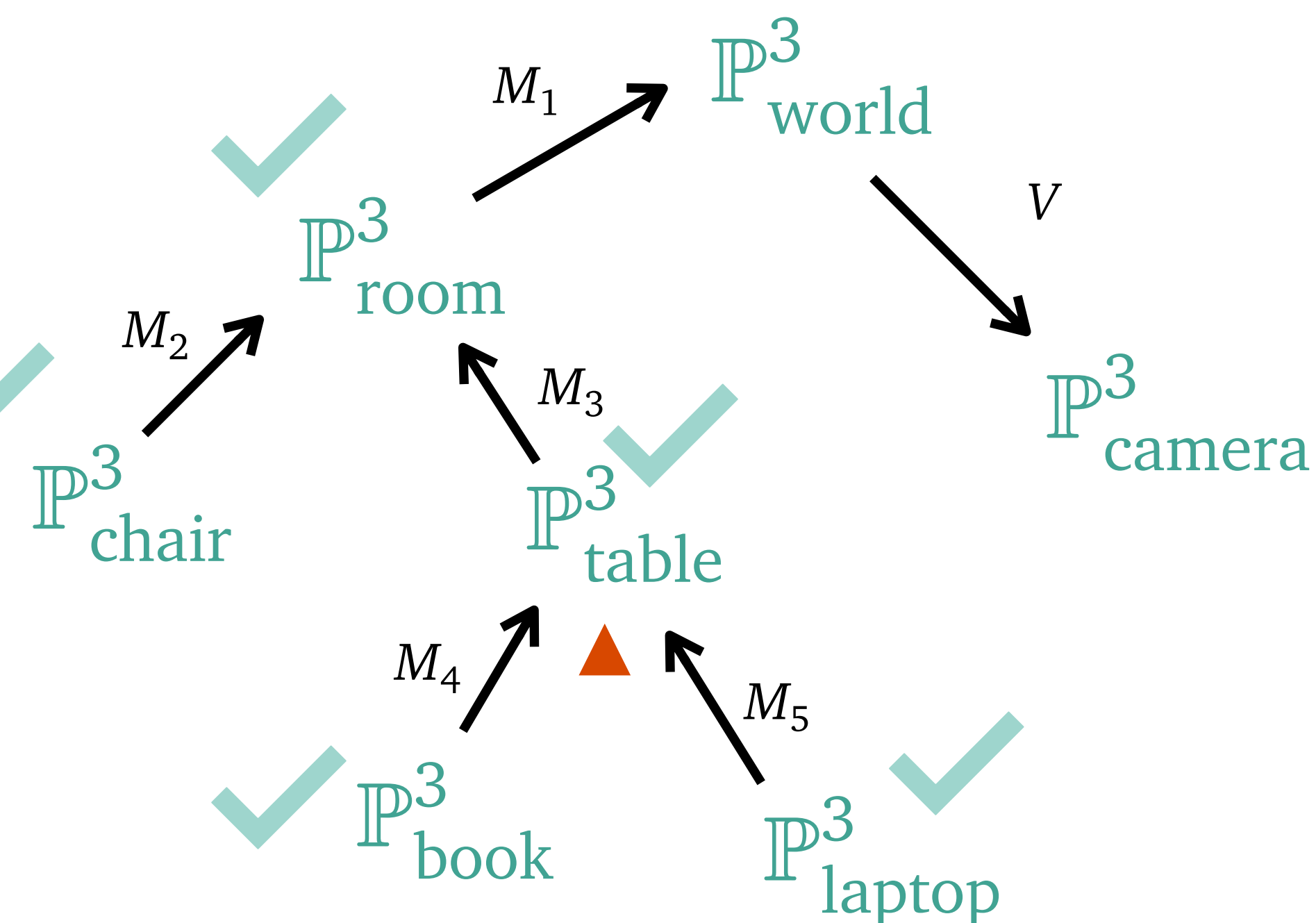
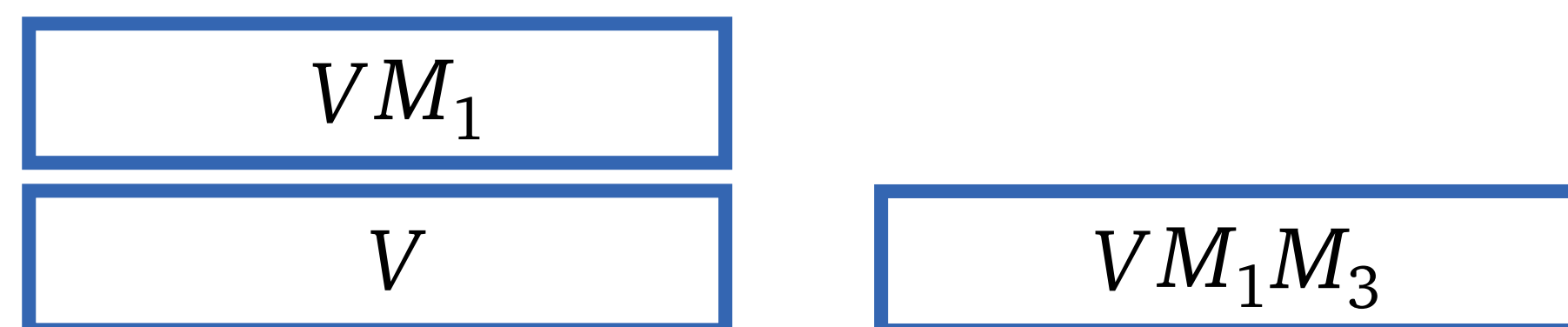
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_5$
- ▶ drawLaptop
- ▶ Pop $VM = STACK.POP$



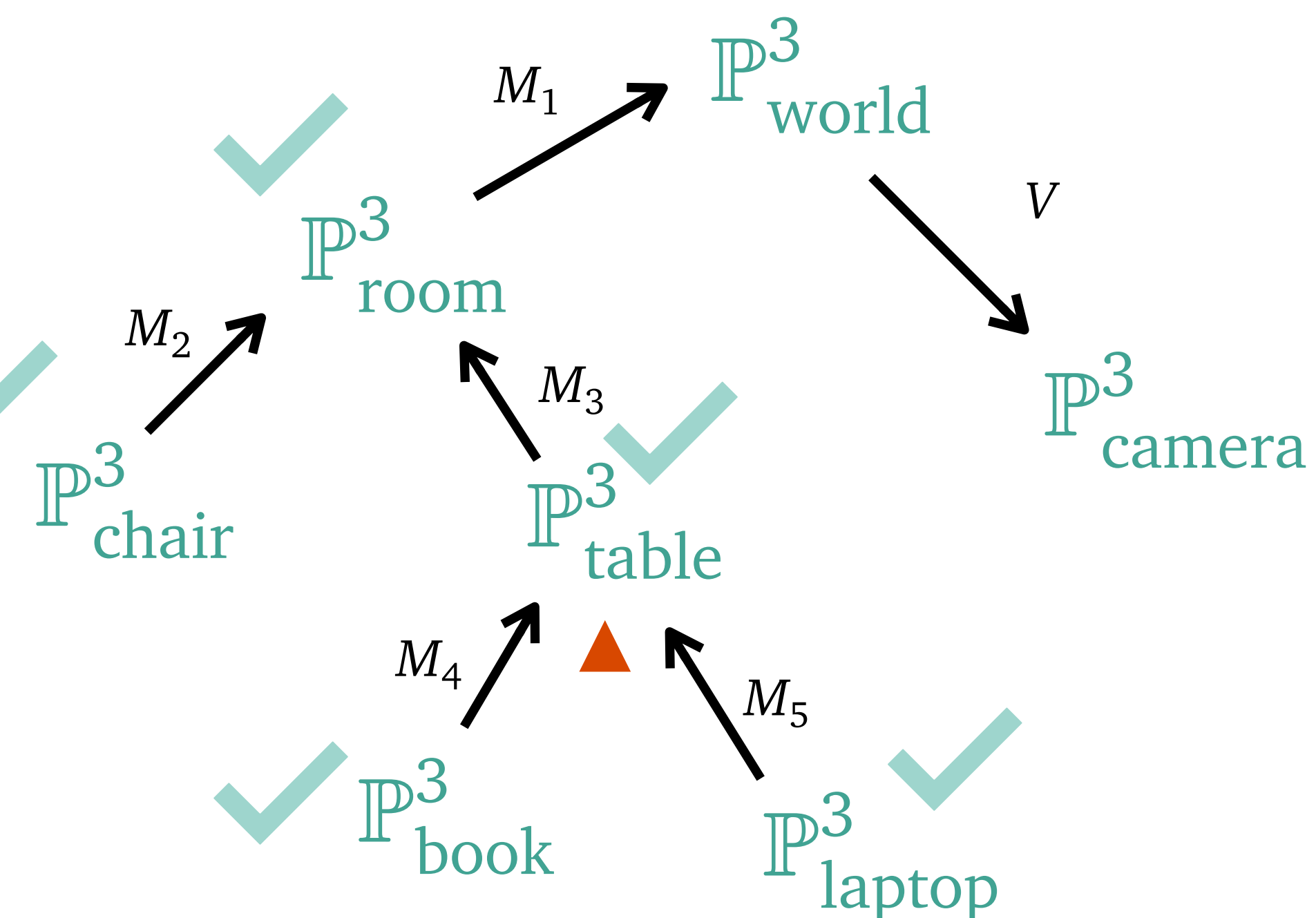
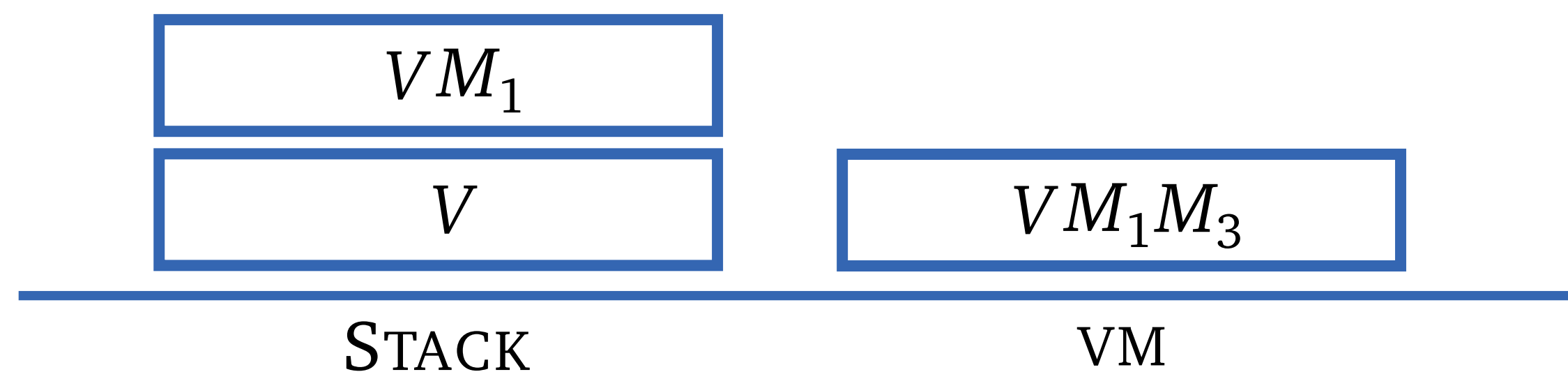
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_5$
- ▶ drawLaptop
- ▶ Pop $VM = STACK.POP$



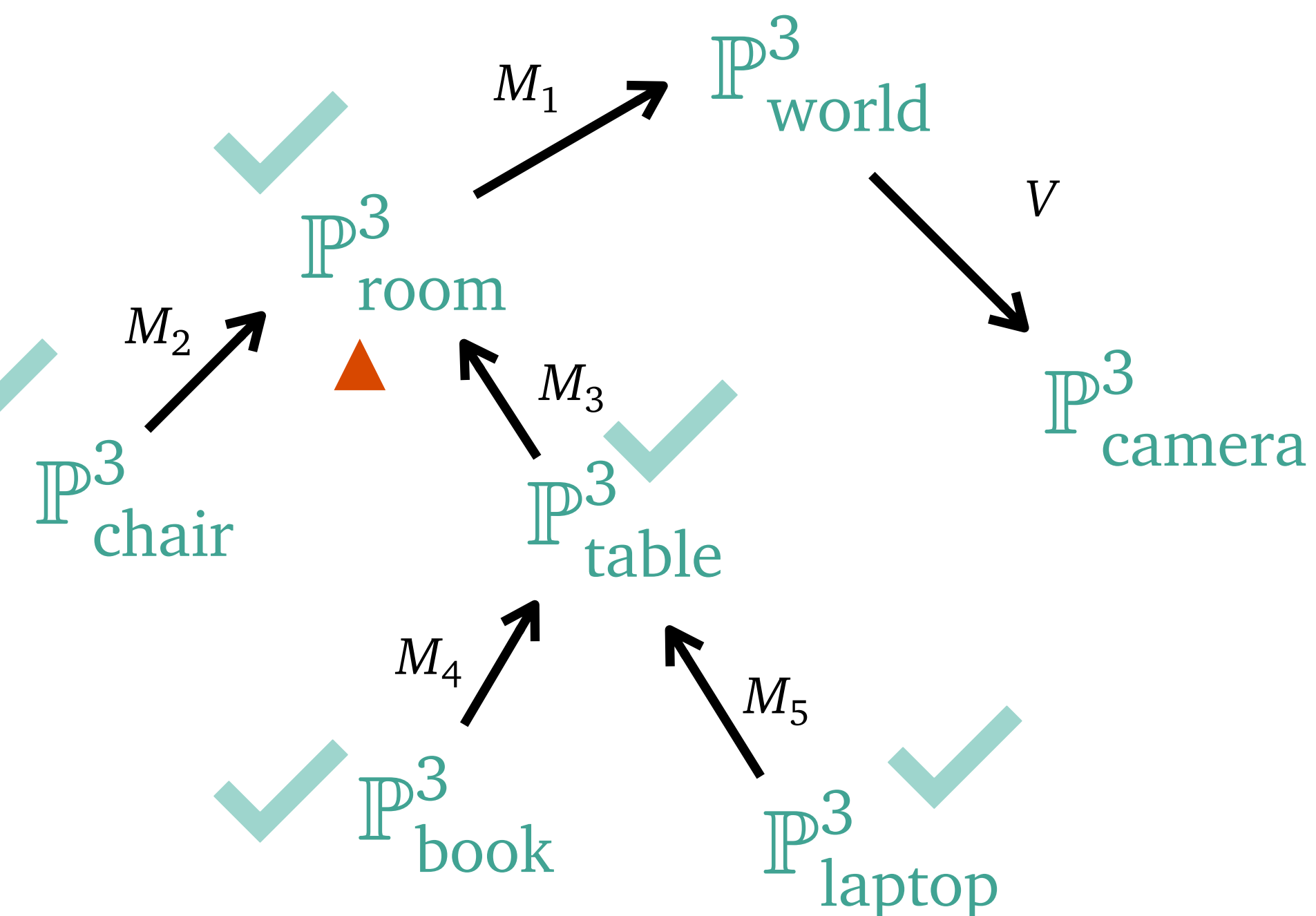
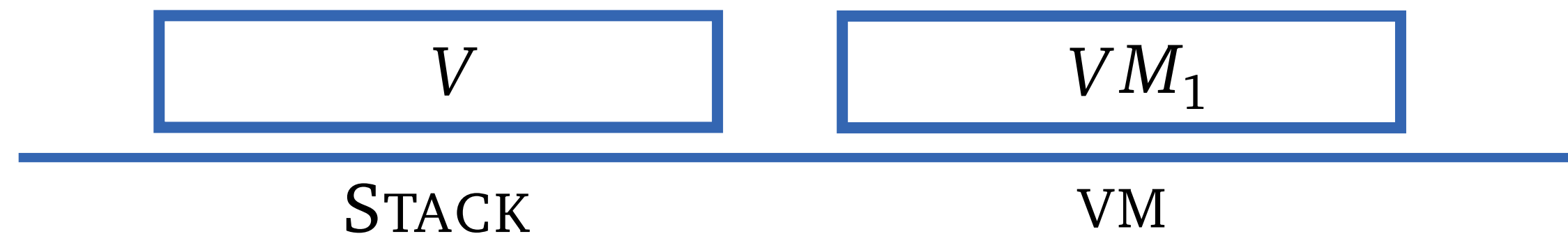
Matrix Stack

- ▶ Initially $VM = V$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_1$
- ▶ drawRoom
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_2$
- ▶ drawChair
- ▶ Pop $VM = STACK.POP$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_3$
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_4$
- ▶ drawBook
- ▶ Pop $VM = STACK.POP$
- ▶ drawTable
- ▶ Push $STACK.PUSH(VM)$
- ▶ Update $VM = VM * M_5$
- ▶ drawLaptop
- ▶ Pop $VM = STACK.POP$
- ▶ Pop $VM = STACK.POP$



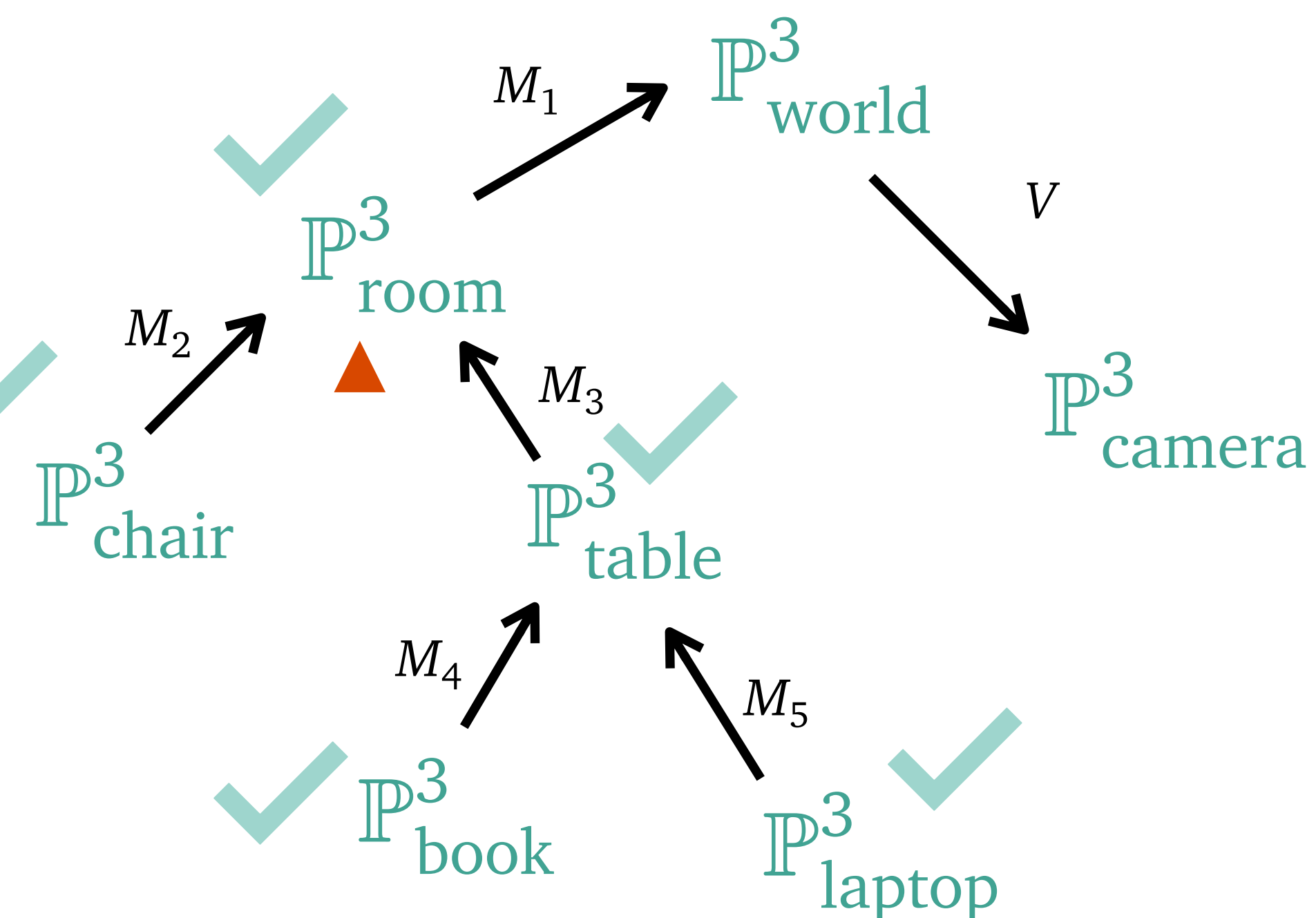
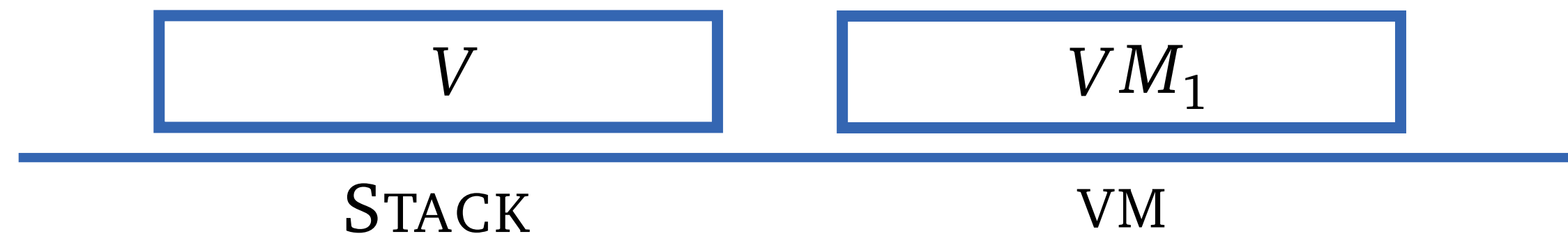
Matrix Stack

- | | |
|--------------------------|--------------------------|
| ▶ Initially $VM = V$ | ▶ Update $VM = VM * M_3$ |
| ▶ Push $STACK.PUSH(VM)$ | ▶ Push $STACK.PUSH(VM)$ |
| ▶ Update $VM = VM * M_1$ | ▶ Update $VM = VM * M_4$ |
| ▶ drawRoom | ▶ drawBook |
| ▶ Push $STACK.PUSH(VM)$ | ▶ Pop $VM = STACK.POP$ |
| ▶ Update $VM = VM * M_2$ | ▶ drawTable |
| ▶ drawChair | ▶ Push $STACK.PUSH(VM)$ |
| ▶ Pop $VM = STACK.POP$ | ▶ Update $VM = VM * M_5$ |
| ▶ Push $STACK.PUSH(VM)$ | ▶ drawLaptop |
| | ▶ Pop $VM = STACK.POP$ |
| | ▶ Pop $VM = STACK.POP$ |



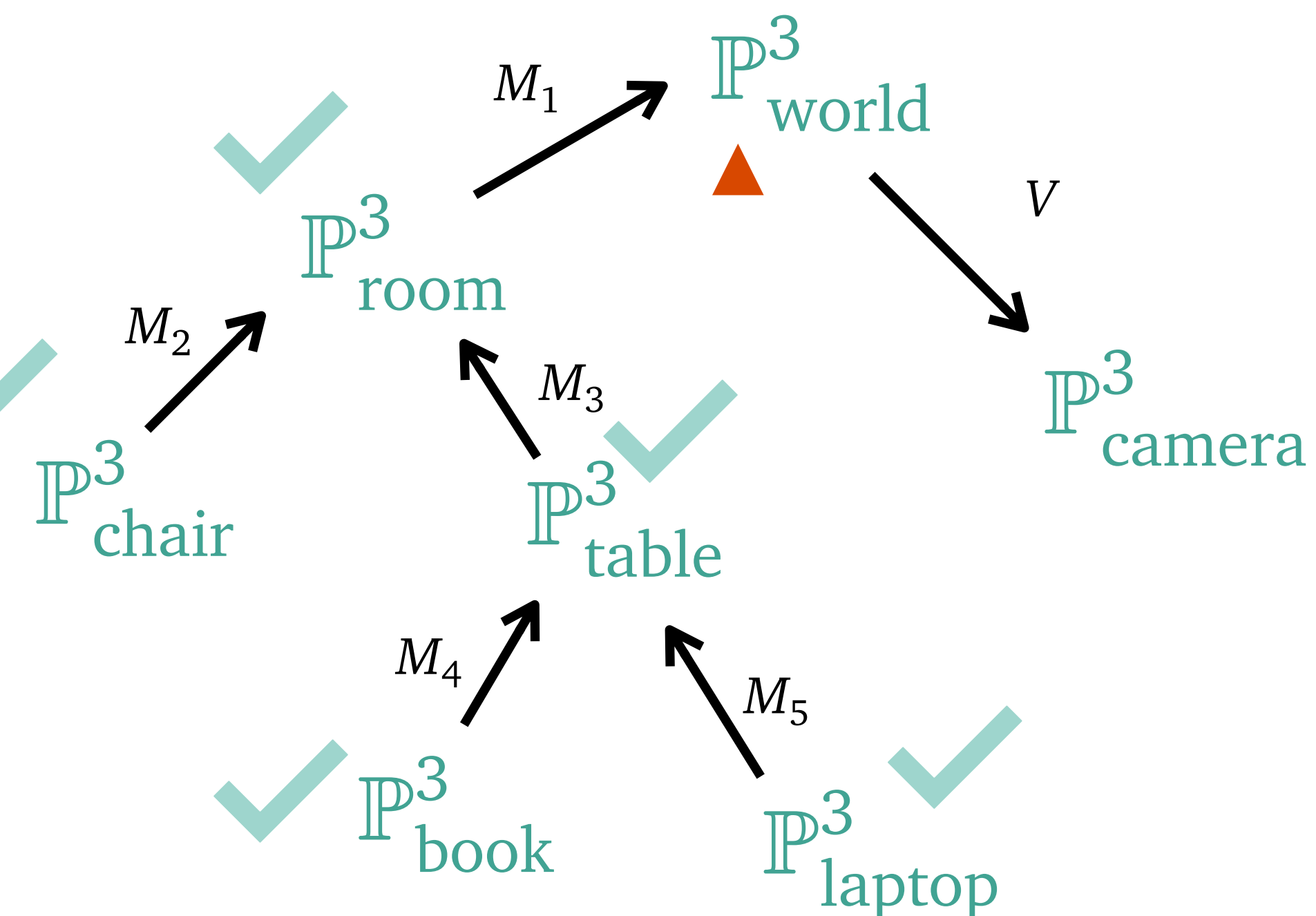
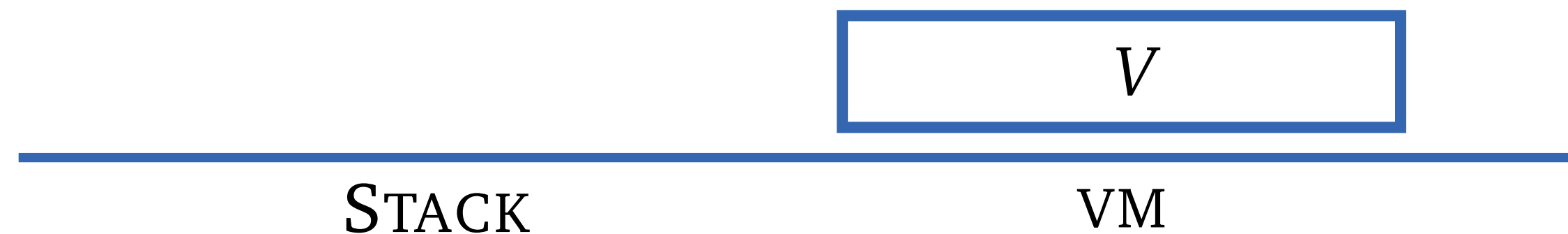
Matrix Stack

- | | |
|--------------------------|--------------------------|
| ▶ Initially $VM = V$ | ▶ Update $VM = VM * M_3$ |
| ▶ Push $STACK.PUSH(VM)$ | ▶ Push $STACK.PUSH(VM)$ |
| ▶ Update $VM = VM * M_1$ | ▶ Update $VM = VM * M_4$ |
| ▶ drawRoom | ▶ drawBook |
| ▶ Push $STACK.PUSH(VM)$ | ▶ Pop $VM = STACK.POP$ |
| ▶ Update $VM = VM * M_2$ | ▶ drawTable |
| ▶ drawChair | ▶ Push $STACK.PUSH(VM)$ |
| ▶ Pop $VM = STACK.POP$ | ▶ Update $VM = VM * M_5$ |
| ▶ Push $STACK.PUSH(VM)$ | ▶ drawLaptop |
| | ▶ Pop $VM = STACK.POP$ |
| | ▶ Pop $VM = STACK.POP$ |
| | ▶ Pop $VM = STACK.POP$ |



Matrix Stack

- | | |
|--------------------------|--------------------------|
| ▶ Initially $VM = V$ | ▶ Update $VM = VM * M_3$ |
| ▶ Push $STACK.PUSH(VM)$ | ▶ Push $STACK.PUSH(VM)$ |
| ▶ Update $VM = VM * M_1$ | ▶ Update $VM = VM * M_4$ |
| ▶ drawRoom | ▶ drawBook |
| ▶ Push $STACK.PUSH(VM)$ | ▶ Pop $VM = STACK.POP$ |
| ▶ Update $VM = VM * M_2$ | ▶ drawTable |
| ▶ drawChair | ▶ Push $STACK.PUSH(VM)$ |
| ▶ Pop $VM = STACK.POP$ | ▶ Update $VM = VM * M_5$ |
| ▶ Push $STACK.PUSH(VM)$ | ▶ drawLaptop |
| | ▶ Pop $VM = STACK.POP$ |
| | ▶ Pop $VM = STACK.POP$ |
| | ▶ Pop $VM = STACK.POP$ |



Next time

- Lighting

