Symmetry-Aware Transformer Training for Automated Planning

Markus Fritzsche Elliot Gestrin Jendrik Seipp

SoTA Limitations

Automated planning seeks action sequences transforming initial states into goals. While transformers excel at sequence tasks, their application to planning remains limited.

PlanGPT [Silver et al. 2024], current SoTA transformer, faces four challenges:

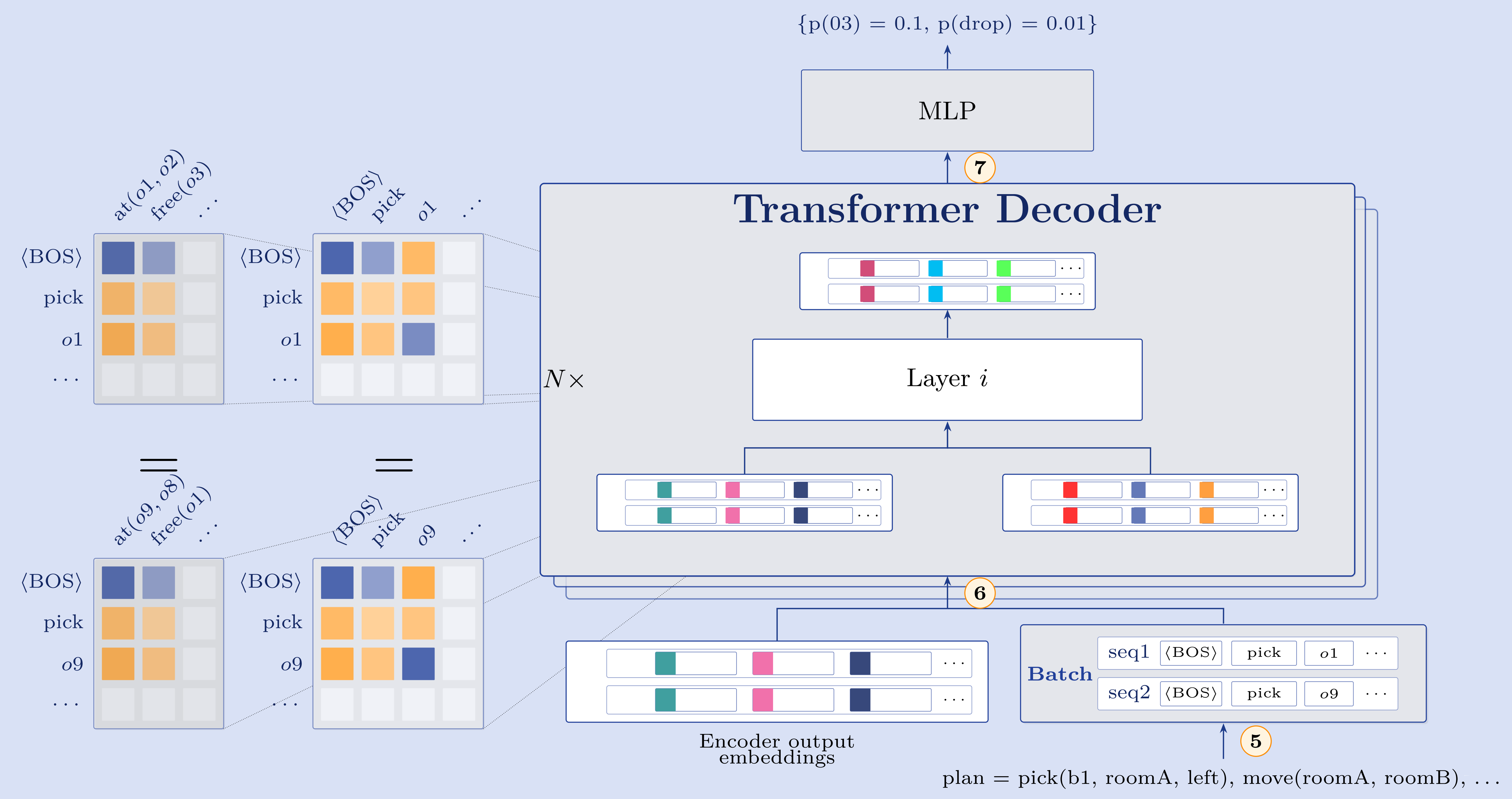
(1) Object Assignment Equivariance. Object names are arbitrary. With $|\mathcal{O}|$ objects and $|\mathcal{V}|$ vocabulary names, there are $\frac{|\mathcal{V}|!}{(|\mathcal{V}|-|\mathcal{O}|)!}$ equivalent assignments. For $|\mathcal{O}| = |\mathcal{V}| = 8$: 40,320 representations of the same task.

(2) Information Leakage. To mitigate (1), PlanGPT keeps semantic names (e.g., `loc-x1-y2`), allowing memorization over generalization.

(3) Atom Order Invariance. Atom ordering is arbitrary. With $|\mathcal{I}|$ initial atoms and $|\mathcal{G}|$ goal atoms, there are $|\mathcal{I}|! \cdot |\mathcal{G}|!$ equivalent orderings per assignment, compounding (1).

(4) Learned Positional Encodings. PlanGPT uses learned positional encodings, preventing generalization to unseen positions in longer plans or larger problems.

These limitations result in poor extrapolation performance and challenge the use of transformers for automated planning.



Symmetry-Aware Training

Architectural changes alone are insufficient to overcome object name symmetries. We introduce contrastive training on pairs of symmetric planning problems.

Core Idea. For each problem, generate two input sequences X and X' with identical structure but different object names. The model should process them equivalently.

Attention Loss. If the model encodes the same algorithm regardless of names, attention patterns must match:

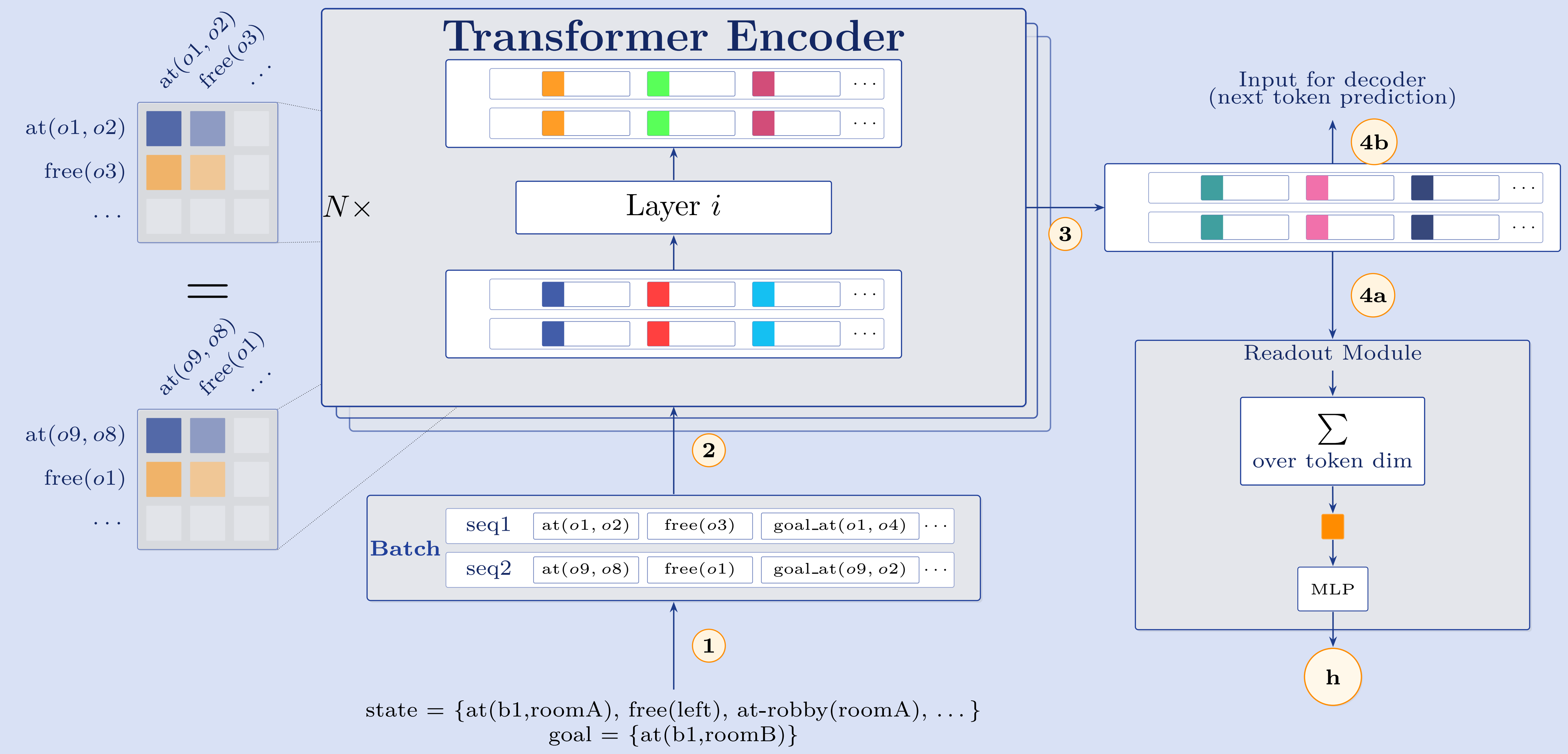
$$L_{\text{att}} = \frac{1}{B} \sum_{\alpha \in Att} \sum_{i=1}^{\#\text{rows}(\alpha)} \sum_{j=1}^{\#\text{cols}(\alpha)} (\alpha_{i,j} - \alpha'_{i,j})^2$$

Hidden State Loss. Token representations should encode name-independent problem structure:

$$L_{\text{hid}} = \frac{1}{B} \sum_{H \in \{X,Y\}} \sum_{l=1}^{\#\text{layers}(H)} (H^{(l)}_{[:d_k]} - H'^{(l)}_{[:d_k]})^2$$

Here, B is batch size, Att is all attention modules, and prime denotes the same input with randomized object names.

Combined Objective: $L = w_1 \cdot L_{\text{pred}} + w_2 \cdot L_{\text{att}} + w_3 \cdot L_{\text{hid}}$
where L_{pred} is the standard prediction loss (cross-entropy for plans, MSE for heuristics).


Transformer Encoder
Layer i
$N\times$
at(o1, o2) free(o3) ...
at(o9, o8) free(o1) ...
=
Input for decoder (next token prediction)
4b
3
4a
Readout Module
∑ over token dim
MLP
h
2
Batch
seq1 at(o1, o2) free(o3) goal_at(o1, o4) ...
seq2 at(o9, o8) free(o1) goal_at(o9, o2) ...
1
state = {at(b1,roomA), free(left), at-robby(roomA), ...}
goal = {at(b1,roomB)}

Symmetry-Aware Architectures

We make four architectural changes to address the identified limitations:

Encoders (Limitation 3). Use encoders without positional embeddings, guaranteeing permutation-equivariance. Yields $|\mathcal{I}|! \cdot |\mathcal{G}|!$ reduction of input space.

Joint Atom Embeddings. Encode each atom as a single embedding to avoid massive vocabulary. Concatenate predicate and object embeddings, then pass through linear layer:

$$T_{p(o_1,\ldots,o_n)} = \mathbf{W}(E[p]|E[o_1]|\ldots|E[o_n]|\text{pad}) + \mathbf{b}$$

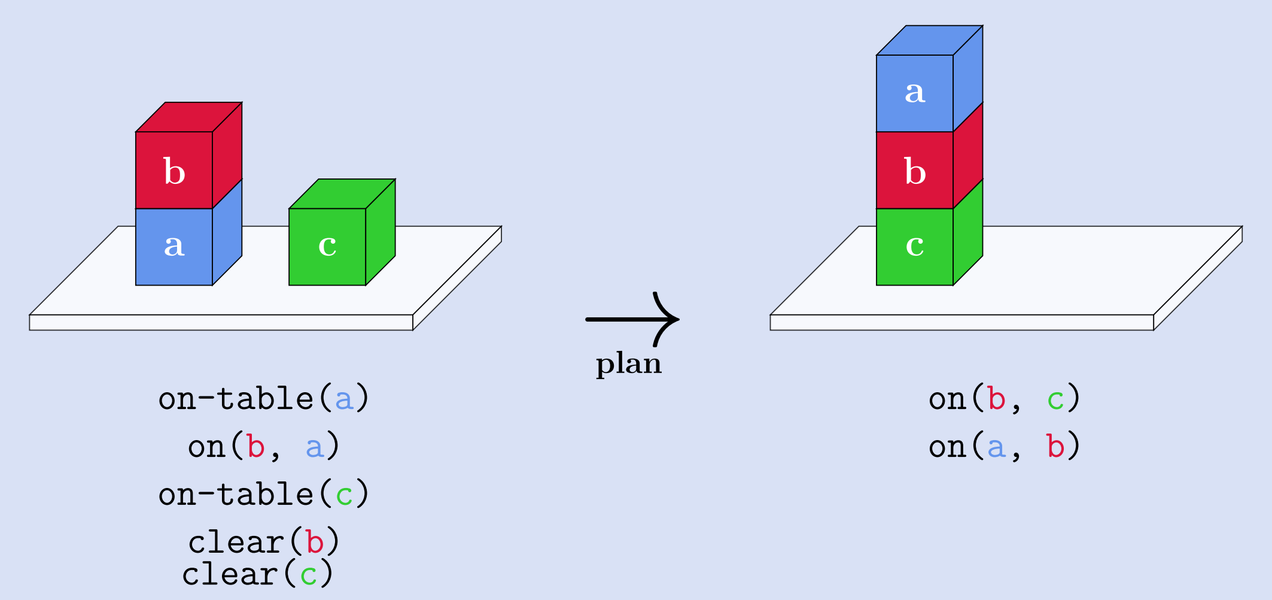
NoPE (Limitation 4). Omit positional encodings in decoder, learning position implicitly from masking. Enables length generalization.

Shared Weights. Motivated by GNNs, share weights across layers to reduce parameters (7M-16M vs PlanGPT's 117M).

Two architectures of *Sym*metry-Aware *T*ransformers:

- **SymT$^{\text{E}}$**: Encoder-only for heuristic prediction.
- **SymT$^{\text{ED}}$**: Encoder-decoder for plan generation.

The Task



Results

Normalized coverage scores $[(\mu \pm \sigma)]$. Our models significantly outperform PlanGPT, though extrapolation remains challenging.

	PlanGPT - Decoder (baseline)			SymT$^{\text{E}}$ (ours)	SymT$^{\text{ED}}$ (ours)		
	greedy	applicable	regrounding	greedy	greedy	applicable	regrounding
validation	.00±.00	.06±.10	.00±.00	**.75±.43**	.38±.40	.73±.42	.50±.50
interpolation	.17±.24	.53±.18	.15±.21	.75±.38	.69±.33	**.81±.35**	.81±.37
extrapolation	.00±.00	.00±.01	.00±.00	.12±.19	.02±.03	.11±.08	**.36±.37**