	R R R	R G B	R R B
A_1	$\frac{1}{3}(s_1+s_2+s_3)$	s_1	$\frac{1}{2}(s_1+s_2)$
A_2	$\frac{1}{3}(s_1+s_2+s_3)$	s_2	$\frac{1}{2}(s_1+s_2)$
A_3	$\frac{1}{3}(s_1+s_2+s_3)$	s_3	s_3

	R R R	R G B	R R B
A_1	$\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$	(1,0,0)	$\left(\frac{1}{2},\frac{1}{2},0\right)$
A_2	$\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$	(0,1,0)	$\left(\frac{1}{2},\frac{1}{2},0\right)$
A_3	$\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$	(0,0,1)	(0,0,1)

Figure 1: *Illustration of how relational cross attention works on a simple task. The input objects (x_1, x_2, x_3) are depicted in the top row. The output is $(A_1, A_2, A_3) = RelationalCrossAttn(X, S)$. We suppose that the left and right encoders are learned such that $\langle \phi_1(x_i), \phi_2(x_j) \rangle$ is large if the attribute (color or shape) is the same. The abstract states represent the relations between the input objects. The right side of the figure shows the abstract states when the symbols are chosen to be the canonical basis vectors, $s_i = e_i$.*

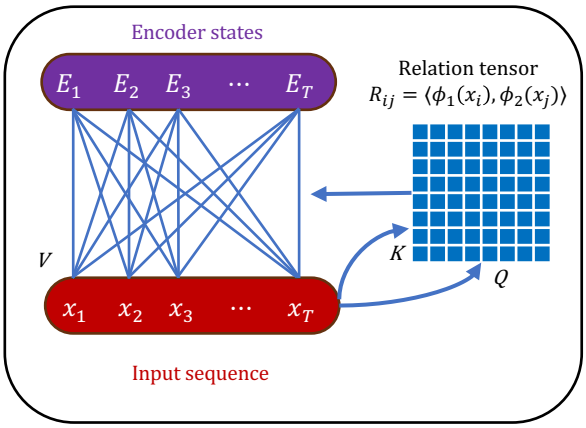

Fig2a: $E = \text{SelfAttn}(X)$

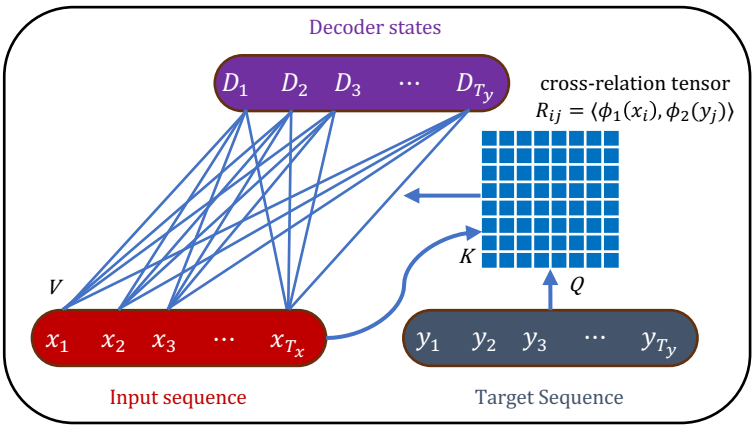

Fig2b: $D = \text{CrossAttn}(X, Y)$

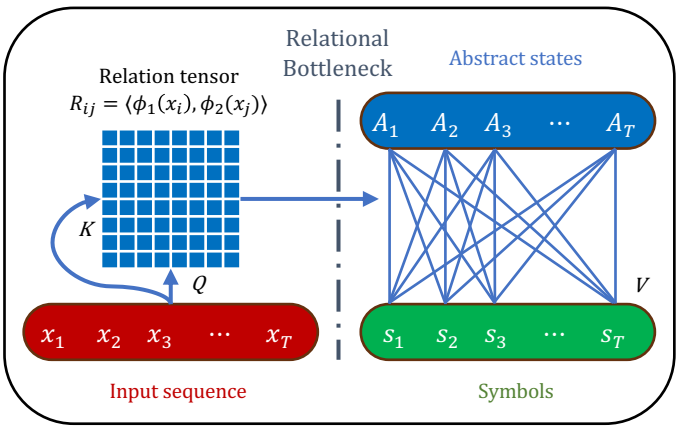

Fig2c: $A = \text{RelationalCrossAttn}(X, S)$

Figure 2: *An illustration of the different attention mechanisms in Transformers and Abstractors. The transformer uses **(a)** self-attention and **(b)** cross-attention. The Abstractor framework introduces **(c)** relational cross-attention as a new attention mechanism. K, Q, and V denote keys, queries, and values. R denotes relation tensor. In self-attention and cross-attention, relational information and value information of individual objects are mixed in message-passing-like operations. In relational cross-attention, since the 'messages' are input-independent symbols, the abstract states represent only relational information—this is the relational bottleneck.*

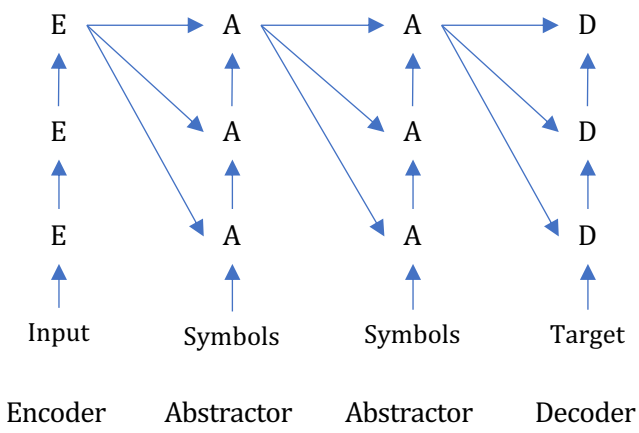

Figure 3: *A depiction of composing several Abstractors together. The abstract states at the output of one Abstractor is the input to the next Abstractor. Hence, the composition of Abstractors computes relations on relations.*