To answer your first question, Let's forget about the GAT equations in the paper and talk about the idea behind GAT.

- In traditional GNNs, a node n's embedding at l-th layer, h_n^l, is computed by **uniformly** aggregating(i.e. adding) messages from its neighboring node, which gives: $h_n^l = W_{self} + \sum MSG(h_N^{l-1})$.
- The traditional GATs use the attention mechanism to **determine the aggregation weights**, which gives: $h_n^l = W_{self} + \sum a_N MSG(h_N^{l-1})$.
- In these non-edge-attribute GATs, the attention weights are just computed by: $a_{dst}^l = W_{attn} \cdot (h_{src}^{l-1} || MSG(h_{dst}^{l-1}))$.
- Our GAT includes edge features to **control the attention weights**. We want to let the edge feature along with the destination node feature decide which part of neighboring information is prioritized in aggregation, which should give: $h_n^l = W_{self} + \sum a_N MSG(h_N^{l-1}),\ a_{dst}^l = W_{attn} \cdot (h_{src}^{l-1} || emb_e || MSG(h_{dst}^{l-1}))$. The message function is `fc` or `fc_dst` in our code.
- The paper is wrong about what is being aggregated. In common GAT implementations, its the neighboring **node embeddings** being aggregated. Additionally, our gat.py is directly modified from the DGL official implementation and is very similar to the GraphMAE implementation. It is very unlikely to be erroneous.

To answer your second question, please confirm that $a_{dst}^l = W_{attn} \cdot (h_{src}^{l-1} || emb_e || MSG(h_{dst}^{l-1}))$, which means $a_{dst}^l = W_{attn_{src}} \cdot h_{src}^{l-1} + W_{attn_{e|dst}}(emb_e || MSG(h_{dst}^{l-1}))$.