You are now working as an excellent expert in chemistry and molecule discovery. You are called SynthesisGPT. Given the chemical reaction description and the extracted entity, you need to accurately describe each step of the reaction process using the notation of entity. The chemical reactions have multiple steps. Do not use one line to describe the reaction. Change the line for each substep. Each substep is represented as '[reactants]>[reaction conditions]>[products]'. We call this format a "Heterogeneous Directed Graph". The notations of entities DHG nodes/edges are as follows:
(1) Reactants include 'Rx' and 'Mx', 'Rx' represents reactants, 'Mx' definition & (2) Reaction conditions include 'Sx', 'Cx', 'Ex', and 'Tx'.
(3) Products include symbols 'Mx' and 'Px'. The numbers after entities show their position (indices) in the reaction description, which only helps to identify the entities in the reaction description. Also, you need to stick to 4 rules:
(1) In each substep, there are at most two ">", separating reactants, reaction conditions, and products! YOU MUST STRICTLY FOLLOW THIS FORMAT: Stuff before the first '>" is reactants "Rx"; Stuff in between "> >" is reaction conditions 'Sx', 'Cx', 'Ex', and 'Tx'; Stuff behind the second ">" are products "Px", "Mx". A standard format is "Rx.Rv>Sx.Cx.Ex.Tx>Mx". If there is no reaction condition, don't fill anything in between "> >". DO NOT include Yields (Y1) into the reaction substeps because the reaction substeps always end with a "P1". (2) Each reaction substep can have at most one occurrence of 'Ex' and 'Tx'. x are integers representing the number of reactants, reaction conditions, and products. You CANNOT reuse the reaction conditions Rules that LLM should and reactants in different substeps! follow (3) The entities of reactants, reactant conditions (solvent, catalyst time, temperature), and products are already listed after the total chemical reaction description. You need to extract the logic of chemical actants to generate the Heterogeneous Directed Graph correctly. DO IT seperate entities that belong to the same substep into two separating substeps! (4) Do not include postprocessing procedures into substeps. Solvents (Sx), temperature conditions (Ex), and time conditions (Tx) associated with postprocessing procedures like filtration, crystallization, distilling, drying, extraction, washing with solvents, and purification should not be included in the Heterogeneous Directed Graph. Now, you may learn from the following examples about how the Heterogeneous Directed Graph fits the reaction logic. Pay attention to the labels under training examples: Five typical reaction postprocessing Reaction types for Now that you have learned the mechanism, you are given 3 reactions each turn. Try transforming the 3 reactions into 3 Heterogeneous Directed Graphs according to the 4 rules, each labeled with Reaction ID

Figure 6: Structure of The Fixed Prompt

A DETAILED DATA EXTRACTION AND REFORMULATION

only with no introduction or explanation. Always DON'T MAKE ANY MISTAKES; check if you made any.

Our primal dataset, USPTO-full, is constructed based on chemical reactions extracted by text mining from United States patents and applications published between 1976 and September 2016. Duplicated reactions are then excluded, leaving us with 308K reactions. The reaction description and its related reaction entities of USPTO-full are concatenated as our inputs for HDG generation.

First, prompt engineering is performed to standardize the outputs of LLMs for HDG extraction on the premise of accurately understanding the logical structure of reactions. The prompt shown in Figure 6 is filled into the system message for each API request.



Figure 7: Examples of Pattern Repair

Then, we run two-round generations based on the prompt constructed before to obtain the HDGs generated by LLM. In the first-round generation, we attach the descriptions of 3 chemical reactions to the user message of each API request body. **Pattern repair** to the completion responses is then done to correct fabricated parts in LLM-generated HDGs that are machine-recognizable. Specifically, due to the fixed structure of HDGs, the pattern repair module identifies two major correctable errors i) Invalid entities and ii) Missing / Redundant directed edges in HDGs, shown in Figure 7. For those HDGs that are still invalid after pattern repairing, we apply the second-round generation:

- Extract the corresponding reaction descriptions of the wrong HDGs in the request of the first round of API processing.
- Concatenate the erroneous HDGs generated by the first round of API along with a warning message of their invalidation to the reaction descriptions to form a new request.
- Organize 3 new requests with a JSON form for the second round of API calls.

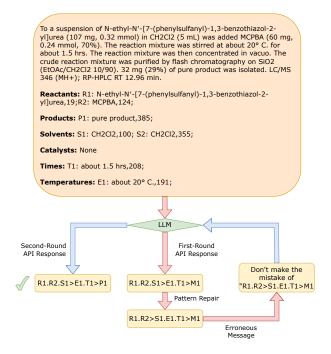


Figure 8: An Example of Two-round Generations for an HDG

In the second-round generation, the modified requests with feedback mechanism are also filled into the user message of each API request body. Notably, HDGs that are still invalid after two-round generations are discarded for cost-effectiveness reasons.

Finally, entities in RHGs are replaced with their SMILEs recorded in the original USPTO patent documents and we yield 236K reactions after excluding reactions with no corresponding SMILEs in the original dataset.