

El Agente Estructural: An Artificially Intelligent Molecular Editor Case Studies Chat History

Changhyeok Choi^{1,6,†}, Yunheng Zou^{2,6,†}, Marcel Müller^{1,6}, Han Hao⁵, Yeonghun Kang^{1,6}, Juan B. Pérez-Sánchez¹, Ignacio Gustin¹, Hanyong Xu¹, Mohammad Ghazi Vakili^{1,2,6}, Chris Crebolder^{2,5}, Alán Aspuru-Guzik^{1,2,3,4,5,6,7,8,*}, Varinia Bernales^{2,5,*}

¹Department of Chemistry, University of Toronto, 80 St. George St., Toronto, ON M5S 3H6, Canada

²Department of Computer Science, University of Toronto, 40 St George St., Toronto, ON M5S 2E4, Canada

³Department of Materials Science & Engineering, University of Toronto, 184 College St., Toronto, ON M5S 3E4, Canada

⁴Department of Chemical Engineering & Applied Chemistry, University of Toronto, 200 College St., Toronto, ON M5S 3E5, Canada

⁵Acceleration Consortium, 700 University Ave., Toronto, ON M7A 2S4, Canada

⁶Vector Institute for Artificial Intelligence, W1140-108 College St., Schwartz Reisman Innovation Campus, Toronto, ON M5G 0C6, Canada

⁷Canadian Institute for Advanced Research (CIFAR), 661 University Ave., Toronto, ON M5G 1M1, Canada

⁸NVIDIA, 431 King St W #6th, Toronto, ON M5V 1K4, Canada

[†]Contributed equally to this work.

Date: February 4, 2026

Correspondence: alan@aspuru.com and varinia@bernales.org



1 Chat history

In this section, we list complete, unedited chat histories corresponding to each case study.

1.1 Case study 1: site-selective functionalization

1.1.1 Primary amine protection in spermidine

User:

Generate an XYZ file for spermidine. Selectively protect only the primary amines with Boc groups, leaving the secondary amine unchanged.

 **Estructural**

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

```
filenames = ["spermidine_protection_project/spermidine.xyz"]  
smiles_list = ["NCCCNCCCCN"]
```

Tool Name: find_available_fragments

Arguments:

```
type = FG  
name = Boc
```

 **Estructural**

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

```
filenames = ["spermidine.xyz", "Boc_Cl.xyz"]  
smiles_list = ["NCCCNCCCCN", "ClC(=O)OC(C)(C)C"]
```

 **Estructural**

Tool used:

Tool Name: match_smarts_in_xyz

Arguments:

```
file_path = spermidine.xyz  
smarts = [NX3H2]
```

Tool Name: match_smarts_in_xyz

Arguments:

```
file_path = Boc_Cl.xyz  
smarts = Cl
```

 **Estructural**

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

```
target = 0  
file_path = spermidine.xyz
```

Tool Name: get_atomic_neighbors

Arguments:

target = 9

file_path = spermidine.xyz

Tool Name: repl_tool

Arguments:

code =

import os

import shutil

os.makedirs('spermidine_protection_project', exist_ok=True)

if os.path.exists('spermidine.xyz'):

shutil.move('spermidine.xyz', 'spermidine_protection_project/spermidine.xyz')

if os.path.exists('Boc_Cl.xyz'):

shutil.move('Boc_Cl.xyz', 'spermidine_protection_project/Boc_Cl.xyz')

print("Directory created and files moved.")

Estructural

Tool used:

Tool Name: replace_terminal_atoms_with_fragment

Arguments:

FG_file_path = spermidine_protection_project/Boc_Cl.xyz

output_path = spermidine_protection_project/diBoc_spermidine.xyz

core_file_path = spermidine_protection_project/spermidine.xyz

FG_connection_idx = 0

binding_mode = replace

core_connection_indices = [10.0, 27.0]

Estructural

Action:

I have successfully generated the XYZ file for spermidine and selectively protected the primary amines with Boc groups, leaving the secondary amine unchanged.

The files are located in the spermidine_protection_project folder:

- **Protected Spermidine:** spermidine_protection_project/diBoc_spermidine.xyz
- **Original Spermidine:** spermidine_protection_project/spermidine.xyz
- **Boc Fragment:** spermidine_protection_project/Boc_Cl.xyz

Methodology

1. **Structure Generation:** Generated the 3D structure of spermidine (NCCCNCCCCN) and the Boc protection group (ClC(=O)OC(C)(C)C) using Open Babel.
2. **Identification:**
 - Identified the two primary amine nitrogen atoms (indices 0 and 9) using SMARTS pattern [NX3H2].
 - Identified the secondary amine nitrogen (index 4) using SMARTS, which was excluded from the protection step.
 - Selected one hydrogen atom from each primary amine (indices 10 and 27) as the substitution site.
3. **Protection:** Replaced the selected hydrogen atoms with the Boc group (using the Cl atom of the Boc-Cl fragment as the leaving group) to form the N-Boc bonds. The secondary amine remains unprotected as requested.

1.1.2 Symmetry-guided functionalization of a cobalt-porphyrin complex

User:

Given Co-porphyrin.xyz, find four hydrogens that are symmetry-equivalent. Replace these hydrogens with phenyl groups while keeping the rest of the hydrogens unchanged.

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

```
code =
import os
import shutil

project_dir = "porphyrin_phenyl_project"
if not os.path.exists(project_dir):
    os.makedirs(project_dir)

source_file = "Co-porphyrin.xyz"
dest_file = os.path.join(project_dir, "Co-porphyrin.xyz")

if os.path.exists(source_file):
    shutil.copy(source_file, dest_file)
    print(f"Copied {source_file} to {dest_file}")
else:
    print(f"{source_file} not found in root.")
```

 **Estructural**

Tool used:

Tool Name: find_pointgroup_equivalent_atoms

Arguments:

file_path = porphyrin_phenyl_project/Co-porphyrin.xyz

 **Estructural**

Tool used:

Tool Name: replace_terminal_atom_with_predefined_FG

Arguments:

core_indices = [33.0, 34.0, 35.0, 36.0]

FG_name = Ph

output_path = porphyrin_phenyl_project/Co-tetraphenylporphyrin.xyz

core_file_path = porphyrin_phenyl_project/Co-porphyrin.xyz

 **Estructural**

Action:

The four symmetry-equivalent hydrogen atoms in Co-porphyrin.xyz were identified at indices **[33, 34, 35, 36]**. These correspond to the meso-positions of the porphyrin ring.

I have successfully replaced these hydrogens with phenyl groups (Ph) to generate the Co-

tetraphenylporphyrin structure.

Output File: porphyrin_phenyl_project/Co-tetraphenylporphyrin.xyz

1.2 Case study 2: Binding of reaction intermediates and ligands

User:

Given the cobalt-phthalocyanine complex (Co-Pc.xyz), bind the following ligands to the Co atom: CO₂, COOH, CO, H, OH, and H₂O.

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

code =

import os

import shutil

project_dir = 'CoPc_ligand_binding_project'

os.makedirs(project_dir, exist_ok=True)

source_file = 'Co-Pc.xyz'

target_file = os.path.join(project_dir, 'Co-Pc.xyz')

if os.path.exists(source_file):

shutil.move(source_file, target_file)

print(f"Moved {source_file} to {target_file}")

else:

print(f"{source_file} not found in root (it might have been moved already). Checking target...")

if os.path.exists(target_file):

print(f"Found {target_file}")

else:

print(f"Error: {source_file} missing.")

Tool Name: find_available_fragments

Arguments:

name = CO₂

Tool Name: find_available_fragments

Arguments:

name = COOH

Tool Name: find_available_fragments

Arguments:

name = CO

Tool Name: find_available_fragments

Arguments:

name = OH

Tool Name: find_available_fragments

Arguments:

name = H₂O

 **Estructural**

Tool used:

Tool Name: bind_atom_ligands_library

Arguments:

output_path = CoPc_ligand_binding_project/CoPc_CO₂.xyz

```

ligand_name = CO2
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
Tool Name: bind_atom_ligands_library
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_COOH.xyz
ligand_name = COOH
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
Tool Name: bind_atom_ligands_library
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_CO.xyz
ligand_name = CO
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = H
output_path = CoPc_ligand_binding_project/CoPc_H.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = OH
output_path = CoPc_ligand_binding_project/CoPc_OH.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = H2O
output_path = CoPc_ligand_binding_project/CoPc_H2O.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz

```

Estructural

Action:

I have successfully bound the requested ligands (CO₂, COOH, CO, H, OH, H₂O) to the Cobalt atom in the Co-Pc complex.

Action Summary:

1. **Project Organization:** Created a new directory `CoPc_ligand_binding_project` and moved the original `Co-Pc.xyz` file into it to keep the workspace organized.
2. **Ligand Binding:** Used the `bind_atom_ligands_library` tool to attach each specified ligand to the central Co atom (element 'Co'). The tool automatically handled the orientation and bonding.

Generated Files: All output files are located in the `CoPc_ligand_binding_project` directory:

- `CoPc_CO2.xyz`: Cobalt-phthalocyanine with a CO₂ ligand.
- `CoPc_COOH.xyz`: Cobalt-phthalocyanine with a Carboxyl (COOH) group.
- `CoPc_CO.xyz`: Cobalt-phthalocyanine with a Carbon Monoxide (CO) ligand.
- `CoPc_H.xyz`: Cobalt-phthalocyanine with a Hydride (H) ligand.
- `CoPc_OH.xyz`: Cobalt-phthalocyanine with a Hydroxyl (OH) group.
- `CoPc_H2O.xyz`: Cobalt-phthalocyanine with a Water (H₂O) ligand.

User:

To explore the axial ligand effect, take both the original Co-Pc.xyz (bare complex) and all the previously modified Co-Pc structures. For each structure, attach one of the following axial ligands to the Co atom: thiophene, pyridine, and Cl.

Estructural

Tool used:

Tool Name: find_available_fragments

Arguments:

name = thiophene

Tool Name: find_available_fragments

Arguments:

name = pyridine

Tool Name: find_available_fragments

Arguments:

name = Cl

Estructural

Tool used:

Tool Name: bind_atom_ligands_library

Arguments:

output_path = CoPc_ligand_binding_project/CoPc_thiophene.xyz

ligand_name = Thiophene

core_element = Co

core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz

Tool Name: bind_atom_ligands_library

Arguments:

output_path = CoPc_ligand_binding_project/CoPc_CO2_thiophene.xyz

ligand_name = Thiophene

core_element = Co

core_file_path = CoPc_ligand_binding_project/CoPc_CO2.xyz

Tool Name: bind_atom_ligands_library

Arguments:

output_path = CoPc_ligand_binding_project/CoPc_COOH_thiophene.xyz

ligand_name = Thiophene

core_element = Co

core_file_path = CoPc_ligand_binding_project/CoPc_COOH.xyz

Tool Name: bind_atom_ligands_library

Arguments:

output_path = CoPc_ligand_binding_project/CoPc_CO_thiophene.xyz

ligand_name = Thiophene

core_element = Co

core_file_path = CoPc_ligand_binding_project/CoPc_CO.xyz

Tool Name: bind_atom_ligands_library

Arguments:

output_path = CoPc_ligand_binding_project/CoPc_H_thiophene.xyz

ligand_name = Thiophene

core_element = Co

core_file_path = CoPc_ligand_binding_project/CoPc_H.xyz

Tool Name: bind_atom_ligands_library

Arguments:

```
ligand_name = Thiophene
output_path = CoPc_ligand_binding_project/CoPc_OH_thiophene.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_OH.xyz
```

Tool Name: bind_atom_ligands_library

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_H2O_thiophene.xyz
ligand_name = Thiophene
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_H2O.xyz
```

Estructural

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

```
filenames = ["CoPc_ligand_binding_project/thiophene.xyz"]
smiles_list = ["c1ccsc1"]
```

Estructural

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

```
target = S
file_path = CoPc_ligand_binding_project/thiophene.xyz
```

Estructural

Tool used:

Tool Name: bind_two_molecules

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_thiophene.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/thiophene.xyz
ligand_idx = 3
ligand_type = ligand
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
```

Tool Name: bind_two_molecules

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_CO2_thiophene.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/thiophene.xyz
ligand_idx = 3
ligand_type = ligand
core_file_path = CoPc_ligand_binding_project/CoPc_CO2.xyz
```

Tool Name: bind_two_molecules

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_COOH_thiophene.xyz
core_indices = [0.0]
```

```
ligand_file_path = CoPc_ligand_binding_project/thiophene.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_COOH.xyz
ligand_type = ligand
ligand_idx = 3
```

Tool Name: bind_two_molecules

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_CO_thiophene.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/thiophene.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_CO.xyz
ligand_type = ligand
ligand_idx = 3
```

Tool Name: bind_two_molecules

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_H_thiophene.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/thiophene.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_H.xyz
ligand_type = ligand
ligand_idx = 3
```

Tool Name: bind_two_molecules

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_OH_thiophene.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/thiophene.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_OH.xyz
ligand_type = ligand
ligand_idx = 3
```

Tool Name: bind_two_molecules

Arguments:

```
core_indices = [0.0]
output_path = CoPc_ligand_binding_project/CoPc_H2O_thiophene.xyz
ligand_file_path = CoPc_ligand_binding_project/thiophene.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_H2O.xyz
ligand_type = ligand
ligand_idx = 3
```

Tool Name: bind_atom_ligands_library

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_pyridine.xyz
ligand_name = Pyridine
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
```

Tool Name: bind_atom_ligands_library

Arguments:

```
ligand_name = Pyridine
output_path = CoPc_ligand_binding_project/CoPc_CO2_pyridine.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_CO2.xyz
```

Tool Name: bind_atom_ligands_library

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_COOH_pyridine.xyz
ligand_name = Pyridine
```

```

core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_COOH.xyz
Tool Name: bind_atom_ligands_library
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_CO_pyridine.xyz
ligand_name = Pyridine
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_CO.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = Pyridine
output_path = CoPc_ligand_binding_project/CoPc_H_pyridine.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_H.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = Pyridine
output_path = CoPc_ligand_binding_project/CoPc_OH_pyridine.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_OH.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = Pyridine
output_path = CoPc_ligand_binding_project/CoPc_H2O_pyridine.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_H2O.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = Cl
output_path = CoPc_ligand_binding_project/CoPc_Cl.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz
Tool Name: bind_atom_ligands_library
Arguments:
ligand_name = Cl
output_path = CoPc_ligand_binding_project/CoPc_CO2_Cl.xyz
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_CO2.xyz
Tool Name: bind_atom_ligands_library
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_COOH_Cl.xyz
ligand_name = Cl
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_COOH.xyz
Tool Name: bind_atom_ligands_library
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_CO_Cl.xyz
ligand_name = Cl
core_element = Co
core_file_path = CoPc_ligand_binding_project/CoPc_CO.xyz
Tool Name: bind_atom_ligands_library

```

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_H_Cl.xyz  
ligand_name = Cl  
core_element = Co  
core_file_path = CoPc_ligand_binding_project/CoPc_H.xyz
```

Tool Name: bind_atom_ligands_library

Arguments:

```
ligand_name = Cl  
output_path = CoPc_ligand_binding_project/CoPc_OH_Cl.xyz  
core_element = Co  
core_file_path = CoPc_ligand_binding_project/CoPc_OH.xyz
```

Tool Name: bind_atom_ligands_library

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_H2O_Cl.xyz  
ligand_name = Cl  
core_element = Co  
core_file_path = CoPc_ligand_binding_project/CoPc_H2O.xyz
```

 **Estructural**

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

```
filenames = ["CoPc_ligand_binding_project/pyridine.xyz"]  
smiles_list = ["n1ccccc1"]
```

 **Estructural**

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

```
target = N  
file_path = CoPc_ligand_binding_project/pyridine.xyz
```

 **Estructural**

Tool used:

Tool Name: bind_two_molecules

Arguments:

```
output_path = CoPc_ligand_binding_project/CoPc_pyridine.xyz  
core_indices = [0.0]  
ligand_file_path = CoPc_ligand_binding_project/pyridine.xyz  
core_file_path = CoPc_ligand_binding_project/Co-Pc.xyz  
ligand_type = ligand  
ligand_idx = 0
```

Tool Name: bind_two_molecules

Arguments:

```
core_indices = [0.0]  
output_path = CoPc_ligand_binding_project/CoPc_CO2_pyridine.xyz  
ligand_file_path = CoPc_ligand_binding_project/pyridine.xyz  
core_file_path = CoPc_ligand_binding_project/CoPc_CO2.xyz
```

```

ligand_type = ligand
ligand_idx = 0
Tool Name: bind_two_molecules
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_COOH_pyridine.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/pyridine.xyz
ligand_idx = 0
ligand_type = ligand
core_file_path = CoPc_ligand_binding_project/CoPc_COOH.xyz
Tool Name: bind_two_molecules
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_CO_pyridine.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/pyridine.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_CO.xyz
ligand_type = ligand
ligand_idx = 0
Tool Name: bind_two_molecules
Arguments:
core_indices = [0.0]
output_path = CoPc_ligand_binding_project/CoPc_H_pyridine.xyz
ligand_file_path = CoPc_ligand_binding_project/pyridine.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_H.xyz
ligand_type = ligand
ligand_idx = 0
Tool Name: bind_two_molecules
Arguments:
output_path = CoPc_ligand_binding_project/CoPc_OH_pyridine.xyz
core_indices = [0.0]
ligand_file_path = CoPc_ligand_binding_project/pyridine.xyz
core_file_path = CoPc_ligand_binding_project/CoPc_OH.xyz
ligand_type = ligand
ligand_idx = 0
Tool Name: bind_two_molecules
Arguments:
core_indices = [0.0]
output_path = CoPc_ligand_binding_project/CoPc_H2O_pyridine.xyz
ligand_file_path = CoPc_ligand_binding_project/pyridine.xyz
ligand_idx = 0
ligand_type = ligand
core_file_path = CoPc_ligand_binding_project/CoPc_H2O.xyz

```

Estructural

Action:

I have successfully generated all the requested structures to explore the axial ligand effect.

Task Summary: For each of the 7 core structures (original Co-Pc + 6 derivatives), I attached three different axial ligands: Thiophene, Pyridine, and Chloride (Cl), resulting in 21 new structures.

Execution Details:

1. **Thiophene:** Generated thiophene.xyz (SMILES: c1ccsc1), identified the Sulfur atom (index 3), and bound it to the Cobalt atom in each core structure.

2. **Pyridine:** Generated `pyridine.xyz` (SMILES: `n1ccccc1`), identified the Nitrogen atom (index 0), and bound it to the Cobalt atom in each core structure.
3. **Cl:** Used the `bind_atom_ligands_library` tool to bind a Chloride atom to the Cobalt atom in each core structure.

Generated Files (in `CoPc_ligand_binding_project`):

- **Thiophene Series:**

- `CoPc_thiophene.xyz`
- `CoPc_CO2_thiophene.xyz`
- `CoPc_COOH_thiophene.xyz`
- `CoPc_CO_thiophene.xyz`
- `CoPc_H_thiophene.xyz`
- `CoPc_OH_thiophene.xyz`
- `CoPc_H2O_thiophene.xyz`

- **Pyridine Series:**

- `CoPc_pyridine.xyz`
- `CoPc_CO2_pyridine.xyz`
- `CoPc_COOH_pyridine.xyz`
- `CoPc_CO_pyridine.xyz`
- `CoPc_H_pyridine.xyz`
- `CoPc_OH_pyridine.xyz`
- `CoPc_H2O_pyridine.xyz`

- **Chloride (Cl) Series:**

- `CoPc_Cl.xyz`
- `CoPc_CO2_Cl.xyz`
- `CoPc_COOH_Cl.xyz`
- `CoPc_CO_Cl.xyz`
- `CoPc_H_Cl.xyz`
- `CoPc_OH_Cl.xyz`
- `CoPc_H2O_Cl.xyz`

All files are neatly organized in the `CoPc_ligand_binding_project` folder.

1.3 Case study 3: Organometallic complex construction with stereochemical and ligand variations

1.3.1 Generation of organometallic complexes with specified stereochemistry

User:

Generate the following structures with their respective isomers: 1. cisplatin and transplatin 2. Δ -Fe(bpy)₃ and Λ -Fe(bpy)₃ 3. fac-Ir(ppy)₃ and mer-Ir(ppy)₃

 **Estructural**

Tool used:

Tool Name: find_available_fragments

Arguments:

type = ligand

Tool Name: find_available_fragments

Arguments:

type = bidentate_ligand

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

code =

import os

os.makedirs('isomers_project', exist_ok=True)

print("Directory created")

 **Estructural**

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"NH3": [1.0, 2.0]}]

metal = Pt

coordination_geometry = square_planar

output_path = isomers_project/cisplatin.xyz

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"NH3": [1.0, 3.0]}]

output_path = isomers_project/transplatin.xyz

coordination_geometry = square_planar

metal = Pt

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"bpy": [[1.0, 4.0], [2.0, 3.0], [5.0, 6.0]]}]

metal = Fe

coordination_geometry = octahedral

output_path = isomers_project/delta_Fe_bpy3.xyz

Tool Name: build_tm_complex_from_template

Arguments:

```
ligand_bindings = [{"bpy": [[1.0, 5.0], [2.0, 6.0], [3.0, 4.0]]}]  
metal = Fe  
coordination_geometry = octahedral  
output_path = isomers_project/lambda_Fe_bpy3.xyz
```

Tool Name: build_tm_complex_from_template

Arguments:

```
ligand_bindings = [{"ppy": [[1.0, 5.0], [3.0, 4.0], [6.0, 2.0]]}]  
metal = Ir  
coordination_geometry = octahedral  
output_path = isomers_project/fac_Ir_ppy3.xyz
```

Tool Name: build_tm_complex_from_template

Arguments:

```
ligand_bindings = [{"ppy": [[1.0, 5.0], [3.0, 4.0], [2.0, 6.0]]}]  
metal = Ir  
coordination_geometry = octahedral  
output_path = isomers_project/mer_Ir_ppy3.xyz
```

Estructural

Action:

I have generated the requested structures and their isomers in the `isomers_project` folder.

1. Cisplatin and Transplatin (Pt(NH₃)₂Cl₂)

- **Cisplatin:** `isomers_project/cisplatin.xyz`
 - Generated using a square planar template with NH₃ ligands at cis positions (indices [1, 2]).
- **Transplatin:** `isomers_project/transplatin.xyz`
 - Generated using a square planar template with NH₃ ligands at trans positions (indices [1, 3]).

2. Δ-Fe(bpy)₃ and Λ-Fe(bpy)₃

- **Δ-Fe(bpy)₃:** `isomers_project/delta_Fe_bpy3.xyz`
 - Generated using an octahedral template with 2,2'-bipyridine (bpy) ligands arranged in the Delta (Δ) configuration (indices [[1, 4], [2, 3], [5, 6]]).
- **Λ-Fe(bpy)₃:** `isomers_project/lambda_Fe_bpy3.xyz`
 - Generated using an octahedral template with bpy ligands arranged in the Lambda (Λ) configuration (indices [[1, 5], [2, 6], [3, 4]]).

3. fac-Ir(ppy)₃ and mer-Ir(ppy)₃

- **fac-Ir(ppy)₃:** `isomers_project/fac_Ir_ppy3.xyz`
 - Generated using an octahedral template with 2-phenylpyridine (ppy) ligands in the facial (fac) arrangement (indices [[1, 5], [3, 4], [6, 2]]).
- **mer-Ir(ppy)₃:** `isomers_project/mer_Ir_ppy3.xyz`
 - Generated using an octahedral template with ppy ligands in the meridional (mer) arrangement (indices [[1, 5], [3, 4], [2, 6]]).

1.3.2 Construction of zirconocene-based complexes

User:

Generate XYZ files for the following zirconocene complexes:

1. Zirconocene dichloride, ZrCp_2Cl_2
2. A zirconocene complex with one methyl ligand and one η^2 -coordinated ethylene ligand, $\text{ZrCp}_2(\text{CH}_3)(\eta^2\text{-C}_2\text{H}_4)$

 **Estructural**

Tool used:

Tool Name: find_available_fragments

Arguments:

type = predefined_organometallics
name = zirconocene

Tool Name: find_available_fragments

Arguments:

type = ligand
name = Cp

Tool Name: find_available_fragments

Arguments:

type = ligand
name = ethylene

 **Estructural**

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"Cp": [1.0, 2.0]}]
output_path = zirconocene_project/ZrCp2Cl2.xyz
coordination_geometry = tetrahedral
metal = Zr

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"Cp": [1.0, 2.0]}, {"Me": [3.0]}, {"ethylene": [4.0]}]
output_path = zirconocene_project/ZrCp2Me_ethylene.xyz
coordination_geometry = tetrahedral
metal = Zr

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

code =
import os
os.makedirs('zirconocene_project', exist_ok=True)
print("Directory created")

 **Estructural**

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

```
ligand_bindings = [{"Cp": [1.0, 2.0]}]
metal = Zr
coordination_geometry = tetrahedral
output_path = zirconocene_project/ZrCp2Cl2.xyz
```

Tool Name: build_tm_complex_from_template

Arguments:

```
ligand_bindings = [{"Cp": [1.0, 2.0]}, {"Me": [3.0]}, {"ethylene": [4.0]}]
metal = Zr
coordination_geometry = tetrahedral
output_path = zirconocene_project/ZrCp2Me_ethylene.xyz
```

 **Estructural****Action:**

The XYZ files for the requested zirconocene complexes have been generated and saved in the zirconocene_project folder.

1. **Zirconocene dichloride (ZrCp₂Cl₂)**

- **File:** zirconocene_project/ZrCp2Cl2.xyz
- **Construction:** Built using a tetrahedral zirconium template with two cyclopentadienyl (Cp) ligands and two chloride ligands (default).

2. **Zirconocene methyl ethylene complex (ZrCp₂(CH₃)(η^2 -C₂H₄))**

- **File:** zirconocene_project/ZrCp2Me_ethylene.xyz
- **Construction:** Built using a tetrahedral zirconium template with two Cp ligands, one methyl (Me) ligand, and one ethylene ligand.
- **Note:** The ethylene ligand is attached using the standard fragment library definition. The " η^2 " coordination is implied by the binding mode in the template, though the exact hapticity depends on the predefined fragment's orientation relative to the metal center.

1.3.3 Construction of concave and convex $\text{CpRu}(\eta^6\text{-sumanene})^+$ complexes

User:

Build the concave and convex $\text{CpRu}(\eta^6\text{-sumanene})^+$ complexes by inspecting the three-dimensional shape of the molecule.

 **Estructural**

Action:

I'll help you build the concave and convex $\text{CpRu}(\eta^6\text{-sumanene})^+$ complexes. Sumanene is a bowl-shaped polycyclic aromatic hydrocarbon, so the Ru metal center can coordinate to the central benzene ring from either the inside (concave) or outside (convex) of the bowl.

Let me start by checking available predefined organometallic templates that might be useful for this construction:

Tool used:

Tool Name: find_available_fragments

Arguments:

type = predefined_organometallics

 **Estructural**

Action:

I can see from the directory that sumanene-related files exist. Let me first view the sumanene structure to understand its 3D bowl shape and identify the central benzene ring for coordination:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = sumanene.xyz

left_right_deg = 0

up_down_deg = 0

 **Estructural**

Action:

The sumanene.xyz file doesn't exist yet - I need to create it. Let me look up the SMILES for sumanene:

Tool used:

Tool Name: lookup_smiles_from_name

Arguments:

compound_name = sumanene

 **Estructural**

Action:

Excellent! I found the SMILES for sumanene. Now let me generate the 3D structure:

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

smiles_list = ["C1C2=C3C4=C(CC5=C4C6=C(CC7=C6C3=C1C=C7)C=C5)C=C2"]

filenames = ["sumanene.xyz"]

 **Estructural**

Action:

Now let me view the sumanene structure to understand its 3D bowl shape:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = sumanene.xyz

left_right_deg = 0

up_down_deg = 0

 **Estructural****Action:**

I can see the structure from above. Let me rotate it to see the bowl shape from the side:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = sumanene.xyz

left_right_deg = 0

up_down_deg = 90

 **Estructural****Action:**

Excellent! Now I can clearly see the **bowl shape** of sumanene:

- The **concave side** (inside the bowl) is at the bottom of the image
- The **convex side** (outside the bowl) is at the top

Now I need to identify the central benzene ring atoms where the Ru will coordinate. Let me match the aromatic ring pattern:

Tool used:

Tool Name: match_smarts_in_xyz

Arguments:

file_path = sumanene.xyz

smarts = c1ccccc1

 **Estructural****Action:**

I see multiple aromatic rings detected. In sumanene, the central benzene ring is the one at the bottom of the bowl. Let me examine the atom neighbors to identify the central ring accurately:

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

file_path = sumanene.xyz

target = 7

 **Estructural****Tool used:**

Tool Name: get_atomic_neighbors

Arguments:

file_path = sumanene.xyz

target = 8

Estructural

Action:

Let me check a few more atoms to confirm the central benzene ring:

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

file_path = sumanene.xyz

target = 3

Estructural

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

file_path = sumanene.xyz

target = 12

Estructural

Action:

Let me check which atoms are the CH₂ groups (sp³ carbons) to identify the central benzene ring more clearly:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
# Read sumanene.xyz and analyze the structure
with open('sumanene.xyz', 'r') as f:
```

```
    lines = f.readlines()
```

```
n_atoms = int(lines[0].strip())
print(f"Total atoms: {n_atoms}")
```

```
atoms = []
for i, line in enumerate(lines[2:2+n_atoms]):
    parts = line.split()
    element = parts[0]
    x, y, z = float(parts[1]), float(parts[2]), float(parts[3])
    atoms.append((i, element, x, y, z))
    print(f"Index {i}: {element} at ({x:.3f}, {y:.3f}, {z:.3f})")
```

Estructural

Action:

Now I can see the structure clearly. The atoms with the highest z-coordinates (around $z \approx 1.0$ -1.2) form the **central benzene ring**: atoms **2, 3, 7, 8, 12, 13**. The CH₂ groups (atoms 0, 5, 10) have lower z values, indicating the bowl opens downward.

Therefore:

- **Convex binding:** Ru approaches from above ($z > 1.06$) - outside the bowl

- **Concave binding:** Ru approaches from below ($z < 1.06$) - inside the bowl

Let me check available Cp ligand in the fragment library:

Tool used:

Tool Name: find_available_fragments

Arguments:

name = Cp

Estructural

Action:

The Cp ligand is available. Now let me build the complexes step by step.

Step 1: First, I'll place a Ru atom at the centroid of the central benzene ring (atoms 2, 3, 7, 8, 12, 13):

Tool used:

Tool Name: insert_atom_at_centroid

Arguments:

core_file_path = sumanene.xyz

element = Ru

centroid_atom_indices = [2, 3, 7, 8, 12, 13]

output_path = sumanene_Ru_center.xyz

Estructural

Action:

Ru is now at the centroid of the central benzene ring. Let me view this intermediate structure:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = sumanene_Ru_center.xyz

left_right_deg = 0

up_down_deg = 90

Estructural

Action:

I can see the bowl shape clearly. The Ru is currently at the centroid (in the plane of the central ring). For proper η^6 coordination, I need to position Ru about 1.65-1.70 Å away from the ring plane.

Looking at the geometry:

- The bowl opens **upward** in the image
- **Convex** binding: Ru below the central ring (outside the bowl)
- **Concave** binding: Ru above the central ring (inside the bowl)

Let me calculate the positions and build both complexes. First, let me check the Ru position and move it appropriately:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import numpy as np
```

```
# Read the structure with Ru at centroid
```

```
with open('sumanene_Ru_center.xyz', 'r') as f:
```

```
    lines = f.readlines()
```

```

n_atoms = int(lines[0].strip())
atoms = []
for i, line in enumerate(lines[2:2+n_atoms]):
    parts = line.split()
    element = parts[0]
    x, y, z = float(parts[1]), float(parts[2]), float(parts[3])
    atoms.append((element, np.array([x, y, z])))

# Central benzene ring indices in original sumanene: 2, 3, 7, 8, 12, 13
# After adding Ru (index 33), these indices remain the same
central_ring_indices = [2, 3, 7, 8, 12, 13]

# Calculate centroid and normal vector of the central benzene ring
ring_coords = np.array([atoms[i][1] for i in central_ring_indices])
centroid = np.mean(ring_coords, axis=0)
print(f"Central ring centroid: {centroid}")

# Calculate normal vector using two vectors in the plane
v1 = ring_coords[1] - ring_coords[0]
v2 = ring_coords[2] - ring_coords[0]
normal = np.cross(v1, v2)
normal = normal / np.linalg.norm(normal)
print(f"Normal vector: {normal}")

# Ru position (index 33)
ru_pos = atoms[33][1]
print(f"Current Ru position: {ru_pos}")

# For  $\eta^6$  coordination, Ru should be  $\sim 1.7$  Å from the ring plane
ru_distance = 1.70 # Å

# Convex position: Ru on the outside of the bowl (opposite to bowl opening)
# Concave position: Ru on the inside of the bowl

# Determine bowl direction by checking where CH2 groups are relative to central ring
# CH2 groups: atoms 0, 5, 10 in original (same indices in new file)
ch2_centroid = np.mean([atoms[0][1], atoms[5][1], atoms[10][1]], axis=0)
print(f"CH2 centroid: {ch2_centroid}")

# Vector from ring centroid to CH2 centroid points into the bowl (concave direction)
bowl_direction = ch2_centroid - centroid
bowl_direction_z = bowl_direction[2]
print(f"Bowl opens toward z: {bowl_direction_z} (negative = down, positive = up)")

# Normal vector should point in one of two directions
# Let's make sure it points AWAY from the bowl (convex direction)
if np.dot(normal, bowl_direction) > 0:
    normal = -normal # Flip so it points away from bowl

print(f"Normal pointing to convex (outside) direction: {normal}")

# Convex position: centroid - normal * distance
convex_ru_pos = centroid - normal * ru_distance

```

```

print(f"Convex Ru position: {convex_ru_pos}")

# Concave position: centroid + normal * distance (into the bowl)
concave_ru_pos = centroid + normal * ru_distance
print(f"Concave Ru position: {concave_ru_pos}")

```

Estructural

Action:

Now I have the correct positions:

- **Convex Ru:** [0.026, 0.045, -0.639] (outside the bowl)
- **Concave Ru:** [-0.111, -0.193, 2.75] (inside the bowl)

Let me build both isomers. First, I'll create the convex isomer by moving Ru to its proper position:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```

import numpy as np

# Read the structure
with open('sumanene_Ru_center.xyz', 'r') as f:
    lines = f.readlines()

n_atoms = int(lines[0].strip())

# Convex Ru position (outside the bowl)
convex_ru_pos = np.array([0.02572911, 0.04484847, -0.6389134])

# Create new file with Ru at convex position
new_lines = lines[:2] # Keep header
for i, line in enumerate(lines[2:2+n_atoms]):
    parts = line.split()
    element = parts[0]
    if i == 33: # Ru atom
        new_lines.append(f"{element} {convex_ru_pos[0]:.8f} {convex_ru_pos[1]:.8f} {convex_ru_pos[2]:.8f}
")
    else:
        new_lines.append(line)

with open('sumanene_Ru_convex.xyz', 'w') as f:
    f.writelines(new_lines)

print("Created sumanene_Ru_convex.xyz with Ru outside the bowl")

# Concave Ru position (inside the bowl)
concave_ru_pos = np.array([-0.11073344, -0.19302213, 2.75000907])

# Create new file with Ru at concave position
new_lines = lines[:2] # Keep header
for i, line in enumerate(lines[2:2+n_atoms]):
    parts = line.split()

```

```

    element = parts[0]
    if i == 33: # Ru atom
        new_lines.append(f"{element} {concave_ru_pos[0]:.8f} {concave_ru_pos[1]:.8f} {concave_ru_pos[2]:.8f}
")
    else:
        new_lines.append(line)

with open('sumanene_Ru_concave.xyz', 'w') as f:
    f.writelines(new_lines)

print("Created sumanene_Ru_concave.xyz with Ru inside the bowl")

```

Estructural

Action:

I've created intermediate structures with Ru positioned correctly. Let me verify the convex structure:

Tool used:

Tool Name: view_molecule

Arguments:

```

xyz_file_relative_path = sumanene_Ru_convex.xyz
left_right_deg = 0
up_down_deg = 90

```

Estructural

Action:

I notice there might be some confusion in the orientation. Let me view from the default angle to better understand the 3D positions:

Tool used:

Tool Name: view_molecule

Arguments:

```

xyz_file_relative_path = sumanene_Ru_convex.xyz
left_right_deg = 0
up_down_deg = 0

```

Estructural

Action:

I can see the structure now. Looking at the image:

- The Ru:33 (teal atom) is positioned at the top of the bowl
- The central benzene ring (C:8, C:12, C:7, C:3) is at the bottom
- The bowl opens upward with the peripheral rings curving up

Based on the bowl orientation analysis:

- **Concave (inside the bowl):** toward negative z (below the central ring)
- **Convex (outside the bowl):** toward positive z (above the central ring)

I realize my labels were swapped! Let me correct them and also check the concave structure:

Tool used:

Tool Name: view_molecule

Arguments:

```

xyz_file_relative_path = sumanene_Ru_concave.xyz
left_right_deg = 0

```

```
up_down_deg = 0
```

Estructural

Action:

Looking at the structure from this angle:

- In `sumanene_Ru_concave.xyz`: Ru is clearly positioned inside the bowl cavity - this is the **concave** isomer ✓
- In `sumanene_Ru_convex.xyz`: Ru should be on the outer surface of the bowl - this is the **convex** isomer ✓

It seems my labels got corrected during the calculation. Let me verify by checking the side view of `sumanene_Ru_concave.xyz`:

Tool used:

Tool Name: `view_molecule`

Arguments:

```
xyz_file_relative_path = sumanene_Ru_concave.xyz
left_right_deg = 0
up_down_deg = 90
```

Estructural

Action:

Excellent! Now I can see clearly from the side view:

- The bowl opens **upward**
- In `sumanene_Ru_concave.xyz`, the Ru:33 is positioned **below** the central ring (C:8, C:12, C:7, C:3)
- The CH₂ groups and peripheral rings are at the **top** (inside the bowl)

I now realize there's a label swap issue. Let me correct this based on the bowl geometry:

- Ru **BELOW** the central ring (closer to the outer surface) = **CONVEX**
- Ru **ABOVE** the central ring (inside the bowl cavity) = **CONCAVE**

Let me properly analyze and fix the structures:

Tool used:

Tool Name: `repl_tool`

Arguments:

`code =`

```
import numpy as np
```

```
# Read the sumanene_Ru_concave.xyz file to check Ru position
```

```
with open('sumanene_Ru_concave.xyz', 'r') as f:
```

```
    lines = f.readlines()
```

```
# Find Ru position
```

```
for i, line in enumerate(lines[2:]):
```

```
    parts = line.split()
```

```
    if parts[0] == 'Ru':
```

```
        print(f"In sumanene_Ru_concave.xyz: Ru at z = {parts[3]}")
```

```
        break
```

```
# Read the sumanene_Ru_convex.xyz file
```

```
with open('sumanene_Ru_convex.xyz', 'r') as f:
```

```
    lines = f.readlines()
```

```

# Find Ru position
for i, line in enumerate(lines[2:]):
    parts = line.split()
    if parts[0] == 'Ru':
        print(f"In sumanene_Ru_convex.xyz: Ru at z = {parts[3]}")
        break

# Central ring and CH2 positions for reference
print(f"
Central ring centroid z $\approx$ 1.06")
print(f"CH2 groups centroid z $\approx$ -0.008")
print("
Conclusion:")
print("- Ru at z = 2.75 is ABOVE the central ring (z=1.06), AWAY from CH2 (z=-0.008)")
print("  $\rightarrow$ This is the CONVEX isomer (Ru on outer surface)")
print("- Ru at z = -0.64 is BELOW the central ring, NEAR the CH2 groups")
print("  $\rightarrow$ This is the CONCAVE isomer (Ru inside the bowl)")

```

Estructural

Action:

The labels are indeed swapped! Let me correct this:

- `sumanene_Ru_concave.xyz` (Ru at $z=2.75$, outside the bowl) \rightarrow should be **CONVEX**
- `sumanene_Ru_convex.xyz` (Ru at $z=-0.64$, inside the bowl cavity) \rightarrow should be **CONCAVE**

Let me fix the files and bind the Cp ligand to create the final structures:

Tool used:

Tool Name: `repl_tool`

Arguments:

`code =`

```

import os
import shutil

```

```

# Swap the file names to correct the labels
# sumanene_Ru_concave.xyz (Ru at z=2.75) -> should be sumanene_Ru_convex_temp.xyz
# sumanene_Ru_convex.xyz (Ru at z=-0.64) -> should be sumanene_Ru_concave_temp.xyz

```

```

# Read both files
with open('sumanene_Ru_concave.xyz', 'r') as f:
    concave_content = f.read() # This is actually convex

```

```

with open('sumanene_Ru_convex.xyz', 'r') as f:
    convex_content = f.read() # This is actually concave

```

```

# Write with correct names
with open('sumanene_Ru_convex_corrected.xyz', 'w') as f:
    f.write(concave_content) # z=2.75, outside bowl = convex

```

```

with open('sumanene_Ru_concave_corrected.xyz', 'w') as f:
    f.write(convex_content) # z=-0.64, inside bowl = concave

```

```

print("Created corrected files:")

```

```
print("- sumanene_Ru_convex_corrected.xyz: Ru at z=2.75 (outside bowl)")
print("- sumanene_Ru_concave_corrected.xyz: Ru at z=-0.64 (inside bowl)")
```

Estructural

Action:

Now I need to bind the Cp ligand to Ru (index 33) in both corrected structures. Let me start with the convex isomer:

Tool used:

Tool Name: bind_predefined_fragment

Arguments:

```
core_file_path = sumanene_Ru_convex_corrected.xyz
core_indices = [33]
ligand_name = Cp
output_path = CpRu_sumanene_convex.xyz
```

Estructural

Action:

Excellent! The Cp has been bound to Ru for the convex isomer. Now let me bind Cp to the concave isomer:

Tool used:

Tool Name: bind_predefined_fragment

Arguments:

```
core_file_path = sumanene_Ru_concave_corrected.xyz
core_indices = [33]
ligand_name = Cp
output_path = CpRu_sumanene_concave.xyz
```

Estructural

Action:

Both complexes have been created. Let me visualize them to verify the structures. First, let me view the **convex** isomer:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_convex.xyz
left_right_deg = 0
up_down_deg = 90
```

Estructural

Action:

I can see the **convex** CpRu(η^6 -sumanene)⁺ complex:

- Ru:33 (teal) is coordinated to the **outer surface** of the sumanene bowl
- The Cp ring (C:34-38 with H atoms) is on the opposite side from sumanene
- The bowl opens away from the Ru center

Now let me view the **concave** isomer:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_concave.xyz  
left_right_deg = 0  
up_down_deg = 90
```

 **Estructural**

Action:

Now I can clearly see the **concave** $\text{CpRu}(\eta^6\text{-sumanene})^+$ complex:

- Ru:33 (teal) is coordinated to the **inner surface** of the sumanene bowl
- The Cp ring (C:34-38) is positioned inside the bowl cavity
- The sumanene bowl curves around the Ru-Cp unit
- The CH_2 groups (H:23, H:24, H:26, etc.) form the rim around the complex

Let me view both complexes from another angle to better highlight the structural difference:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_convex.xyz  
left_right_deg = 30  
up_down_deg = 45
```

 **Estructural**

Action:

This is an excellent view of the **convex** $\text{CpRu}(\eta^6\text{-sumanene})^+$ complex:

- Ru:33 (teal) is on the **outer surface** of the sumanene bowl
- The Cp ring (C:35-38 with H atoms) is positioned away from the bowl
- The sumanene bowl curves away from the Ru-Cp unit

Now let me view the **concave** isomer from the same angle:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_concave.xyz  
left_right_deg = 30  
up_down_deg = 45
```

 **Estructural**

Action:

This is an excellent view of the **concave** $\text{CpRu}(\eta^6\text{-sumanene})^+$ complex:

- **Ru:33** (teal) is coordinated to the **inner surface** of the sumanene bowl
- The **Cp ring** (C:34-38 with H atoms) is positioned **inside the bowl cavity**
- The sumanene bowl curves **around** the Ru-Cp unit
- The CH_2 groups form the rim surrounding the complex

Summary

I have successfully built both isomeric $\text{CpRu}(\eta^6\text{-sumanene})^+$ complexes:

Isomer	File	Description	
Convex	CpRu_sumanene_convex.xyz	Ru coordinates to the **outer surface** of the bowl-shaped sumanene. The Cp ligand points away from the bowl.	
Concave	CpRu_sumanene_concave.xyz	Ru coordinates to the **inner surface** of the bowl. The Cp ligand is nestled inside the bowl cavity.	

Key structural features:

- **Sumanene** is a $\text{C}_{21}\text{H}_{12}$ bowl-shaped polycyclic aromatic hydrocarbon

- **Central benzene ring** (atoms 2, 3, 7, 8, 12, 13) serves as the η^6 -coordination site for Ru
- **Ru-ring distance:** 1.70 Å (typical for η^6 -arene coordination)
- The **3 CH₂ groups** define the rim of the bowl and allow clear distinction between the concave and convex faces

Would you like me to perform a geometry optimization on these structures using xTB to refine the bond distances and angles?

User:

In the previous structures, Ru was η^6 -coordinated to the central hexagon of sumanene. Please also generate structures in which Ru is η^6 -coordinated to a peripheral hexagonal ring of sumanene, while keeping the Cp ligand bound to Ru. Provide both concave and convex coordination geometries.

Estructural

Action:

I'll generate CpRu(η^6 -sumanene)⁺ complexes where Ru coordinates to a **peripheral benzene ring** of sumanene. From my earlier SMARTS analysis, I identified the peripheral benzene rings. Let me first identify them clearly and determine which faces are concave vs convex:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import numpy as np

# Read sumanene structure
with open('sumanene.xyz', 'r') as f:
    lines = f.readlines()

n_atoms = int(lines[0].strip())
atoms = []
for i, line in enumerate(lines[2:2+n_atoms]):
    parts = line.split()
    element = parts[0]
    x, y, z = float(parts[1]), float(parts[2]), float(parts[3])
    atoms.append((element, np.array([x, y, z])))

# Peripheral benzene rings from SMARTS matching
peripheral_ring_1 = [1, 2, 3, 4, 19, 20] # One peripheral ring
peripheral_ring_2 = [11, 12, 13, 14, 15, 16] # Another peripheral ring

# Central benzene ring for reference
central_ring = [2, 3, 7, 8, 12, 13]

# Calculate centroids
def get_centroid(indices):
    coords = np.array([atoms[i][1] for i in indices])
    return np.mean(coords, axis=0)

central_centroid = get_centroid(central_ring)
periph1_centroid = get_centroid(peripheral_ring_1)
periph2_centroid = get_centroid(peripheral_ring_2)
```

```

print("Central benzene ring centroid:", central_centroid)
print("Peripheral ring 1 (atoms 1,2,3,4,19,20) centroid:", periph1_centroid)
print("Peripheral ring 2 (atoms 11,12,13,14,15,16) centroid:", periph2_centroid)

# Calculate normal vector for peripheral ring 1
ring_coords = np.array([atoms[i][1] for i in peripheral_ring_1])
v1 = ring_coords[1] - ring_coords[0]
v2 = ring_coords[2] - ring_coords[0]
normal_periph1 = np.cross(v1, v2)
normal_periph1 = normal_periph1 / np.linalg.norm(normal_periph1)

print("
Peripheral ring 1 normal vector:", normal_periph1)

# Determine concave/convex direction
# Vector from peripheral ring centroid to central ring centroid points toward the "inside" of the bowl
to_center = central_centroid - periph1_centroid
print("Vector from peripheral to central:", to_center)

# The concave side is toward the interior of the bowl
# Check if normal points toward or away from center
dot_product = np.dot(normal_periph1, to_center)
print(f"Dot product of normal with to_center: {dot_product}")

if dot_product > 0:
    concave_direction = normal_periph1
    convex_direction = -normal_periph1
else:
    concave_direction = -normal_periph1
    convex_direction = normal_periph1

print("
Concave direction (toward bowl interior):", concave_direction)
print("Convex direction (away from bowl):", convex_direction)

# Ru-ring distance for  $\eta^6$  coordination
ru_distance = 1.70 # Å

# Calculate Ru positions for peripheral ring 1
ru_concave_pos = periph1_centroid + concave_direction * ru_distance
ru_convex_pos = periph1_centroid + convex_direction * ru_distance

print(f"
For Peripheral Ring 1 coordination:")
print(f"Ru CONCAVE position: {ru_concave_pos}")
print(f"Ru CONVEX position: {ru_convex_pos}")

```

Estructural

Action:

I've identified the peripheral benzene ring and calculated the Ru positions. The peripheral ring 1

(atoms 1, 2, 3, 4, 19, 20) is tilted relative to the central ring. Now let me create the structures:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import numpy as np

# Read sumanene structure
with open('sumanene.xyz', 'r') as f:
    lines = f.readlines()

n_atoms = int(lines[0].strip())

# Ru positions for peripheral ring 1 coordination
ru_concave_pos = np.array([1.22859192, 0.26047285, -0.93649893]) # Inside bowl
ru_convex_pos = np.array([3.05310908, 0.37648782, 1.93015027]) # Outside bowl

# Create file with Ru at CONCAVE position (inside bowl)
with open('sumanene_Ru_periph_concave.xyz', 'w') as f:
    f.write(f"{n_atoms + 1}"
    ")
    f.write("Sumanene with Ru at peripheral ring concave position"
    ")
    # Write original atoms
    for line in lines[2:2+n_atoms]:
        f.write(line)
    # Add Ru atom
    f.write(f"Ru    {ru_concave_pos[0]:.8f}    {ru_concave_pos[1]:.8f}    {ru_concave_pos[2]:.8f}"
    ")

print("Created sumanene_Ru_periph_concave.xyz with Ru inside the bowl (concave)")

# Create file with Ru at CONVEX position (outside bowl)
with open('sumanene_Ru_periph_convex.xyz', 'w') as f:
    f.write(f"{n_atoms + 1}"
    ")
    f.write("Sumanene with Ru at peripheral ring convex position"
    ")
    # Write original atoms
    for line in lines[2:2+n_atoms]:
        f.write(line)
    # Add Ru atom
    f.write(f"Ru    {ru_convex_pos[0]:.8f}    {ru_convex_pos[1]:.8f}    {ru_convex_pos[2]:.8f}"
    ")

print("Created sumanene_Ru_periph_convex.xyz with Ru outside the bowl (convex)")

print(f"
Ru is at index 33 in both files")
print(f"Peripheral ring atoms: 1, 2, 3, 4, 19, 20")
```

Action:

I've created the intermediate structures. Let me verify the Ru positions by viewing them:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = sumanene_Ru_periph_concave.xyz

left_right_deg = 30

up_down_deg = 45

 **Estructural****Action:**

Excellent! I can see the **concave** peripheral ring coordination - Ru:33 (teal) is positioned on the inner side of the bowl, near the peripheral benzene ring (C:1, C:2, C:3, C:4, C:19, C:20). Now let me view the convex structure:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = sumanene_Ru_periph_convex.xyz

left_right_deg = 30

up_down_deg = 45

 **Estructural****Action:**

I can see the **convex** peripheral ring coordination - Ru:33 (teal) is positioned on the **outer surface** of the bowl, near the peripheral benzene ring. The bowl cavity is on the opposite side. Now let me bind the Cp ligand to Ru in both structures:

Tool used:

Tool Name: bind_predefined_fragment

Arguments:

core_file_path = sumanene_Ru_periph_concave.xyz

core_indices = [33]

ligand_name = Cp

output_path = CpRu_sumanene_periph_concave.xyz

 **Estructural****Action:**

The concave peripheral structure is created. Now let me bind Cp to the convex structure:

Tool used:

Tool Name: bind_predefined_fragment

Arguments:

core_file_path = sumanene_Ru_periph_convex.xyz

core_indices = [33]

ligand_name = Cp

output_path = CpRu_sumanene_periph_convex.xyz

 **Estructural**

Action:

Both peripheral ring complexes are created. Let me visualize them to verify the structures. First, the **concave** peripheral coordination:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_periph_concave.xyz
left_right_deg = 30
up_down_deg = 45
```

 **Estructural****Action:**

Excellent! I can see the **concave peripheral ring** $\text{CpRu}(\eta^6\text{-sumanene})^+$ complex:

- **Ru:33** (teal) is coordinated to the **peripheral benzene ring** (C:1, C:2, C:3, C:4, C:19, C:20)
- The **Cp ring** (C:36, C:37 with H atoms) is positioned **inside the bowl cavity**
- The central benzene ring and other peripheral rings are visible as part of the sumanene framework

Now let me view the **convex peripheral** coordination:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_periph_convex.xyz
left_right_deg = 30
up_down_deg = 45
```

 **Estructural****Action:**

I can now see the **convex peripheral ring** $\text{CpRu}(\eta^6\text{-sumanene})^+$ complex clearly:

- **Ru:33** (teal) is coordinated to the **outer surface** of the peripheral benzene ring
- The **Cp ring** (C:34-38 with H atoms) extends **outward**, away from the bowl
- The sumanene bowl structure is clearly visible with the peripheral ring (C:2, C:3, C:4, etc.) serving as the η^6 coordination site

Let me also view these from a side angle to better show the bowl geometry:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_periph_concave.xyz
left_right_deg = 0
up_down_deg = 90
```

 **Estructural****Action:**

This is an excellent side view showing the **concave peripheral ring** coordination:

- **Ru:33** (teal) is positioned **inside the bowl cavity**
- The **Cp ring** (C:34-38) is oriented inside the bowl
- The **peripheral benzene ring** (C:1, C:2, C:3, C:4, C:19, C:20) serves as the η^6 coordination site
- The bowl structure curves around the Ru-Cp unit

Now let me view the **convex peripheral** complex from the same angle:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = CpRu_sumanene_periph_convex.xyz  
left_right_deg = 0  
up_down_deg = 90
```

Estructural

Action:

This side view clearly shows the **convex peripheral ring** coordination:

- **Ru:33** (teal) is positioned on the **outer surface** of the bowl
- The **Cp ring** (C:34-38) extends **outward** from the bowl
- The **peripheral benzene ring** (C:1, C:2, C:3, C:4, C:19, C:20) serves as the η^6 coordination site
- The bowl cavity is on the opposite side from the Ru-Cp unit

Summary

I have successfully built all four **CpRu(η^6 -sumanene)⁺** isomeric complexes:

Central Ring Coordination (from previous task): | Isomer | File | Description | |———|———|———| | **Convex** | CpRu_sumanene_convex.xyz | Ru η^6 -coordinated to **central benzene ring** (atoms 2,3,7,8,12,13) from the **outer surface** | | **Concave** | CpRu_sumanene_concave.xyz | Ru η^6 -coordinated to **central benzene ring** from the **inner surface** (inside the bowl) |

Peripheral Ring Coordination (new): | Isomer | File | Description | |———|———|———| | **Convex** | CpRu_sumanene_periph_convex.xyz | Ru η^6 -coordinated to **peripheral benzene ring** (atoms 1,2,3,4,19,20) from the **outer surface** | | **Concave** | CpRu_sumanene_periph_concave.xyz | Ru η^6 -coordinated to **peripheral benzene ring** from the **inner surface** (inside the bowl) |

Key structural features:

- **Peripheral benzene ring** (atoms 1, 2, 3, 4, 19, 20) is tilted relative to the central ring
- **Ru-ring distance:** 1.70 Å (typical for η^6 -arene coordination)
- **Convex** isomers have Cp pointing away from the bowl cavity
- **Concave** isomers have Cp nestled inside the bowl cavity

Would you like me to perform geometry optimization on any of these structures using xTB to refine the geometries?

1.4 Case study 4: Fragment-level replacement in organometallic complexes

1.4.1 Functionalization of Mo-PNP complex

User:

Given MoCl3PNP.xyz, the PNP ligand is 2,6-bis(di-tert-butylphosphinomethyl)pyridine. Find all t-butyl groups attached to the phosphorus atoms. Replace them with hydrogen, methyl, and phenyl, creating separate XYZ files for each replacement.

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

code =

import os

import shutil

project_dir = 'MoCl3PNP_derivatives_project'

if not os.path.exists(project_dir):

os.makedirs(project_dir)

src = 'MoCl3PNP.xyz'

dst = os.path.join(project_dir, 'MoCl3PNP.xyz')

if os.path.exists(src):

shutil.copy(src, dst)

print(f"Copied {src} to {dst}")

else:

print(f"{src} not found")

Tool Name: get_atomic_neighbors

Arguments:

target = P

file_path = MoCl3PNP_derivatives_project/MoCl3PNP.xyz

 **Estructural**

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

target = P

file_path = MoCl3PNP.xyz

Tool Name: repl_tool

Arguments:

code =

import os

print(os.listdir('MoCl3PNP_derivatives_project'))

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

```

code =
import numpy as np

def read_xyz(path):
    with open(path, 'r') as f:
        lines = f.readlines()
    num_atoms = int(lines[0])
    atoms = []
    coords = []
    for line in lines[2:2+num_atoms]:
        parts = line.split()
        atoms.append(parts[0])
        coords.append([float(x) for x in parts[1:4]])
    return atoms, np.array(coords)

atoms, coords = read_xyz('MoCl3PNP.xyz')
targets = [7, 11, 15, 21, 22, 26]

for i in targets:
    c_neighbors = 0
    neighbor_indices = []
    for j, (atom, coord) in enumerate(zip(atoms, coords)):
        if i == j: continue
        dist = np.linalg.norm(coords[i] - coords[j])
        if dist < 1.7: # C-C is ~1.5, P-C is ~1.85. 1.7 should catch C-C but maybe exclude P-C?
            # Actually P-C is around 1.8. Let's strictly look for C neighbors.
            if atom == 'C':
                c_neighbors += 1
                neighbor_indices.append(j)

    print(f"Atom {i} ({atoms[i]}) has {c_neighbors} Carbon neighbors: {neighbor_indices}")

```

Estructural

Tool used:

Tool Name: replace_branch_with_predefined_FG

Arguments:

output_path = MoCl3PNP_derivatives_project/MoCl3PNP_H.xyz

FG_name = H

bond_pair = [[4.0, 7.0], [4.0, 11.0], [5.0, 22.0], [5.0, 26.0]]

core_file_path = MoCl3PNP_derivatives_project/MoCl3PNP.xyz

Tool Name: replace_branch_with_predefined_FG

Arguments:

output_path = MoCl3PNP_derivatives_project/MoCl3PNP_Me.xyz

FG_name = Me

bond_pair = [[4.0, 7.0], [4.0, 11.0], [5.0, 22.0], [5.0, 26.0]]

core_file_path = MoCl3PNP_derivatives_project/MoCl3PNP.xyz

Tool Name: replace_branch_with_predefined_FG

Arguments:

output_path = MoCl3PNP_derivatives_project/MoCl3PNP_Ph.xyz

FG_name = Ph

bond_pair = [[4.0, 7.0], [4.0, 11.0], [5.0, 22.0], [5.0, 26.0]]

```
core_file_path = MoCl3PNP_derivatives_project/MoCl3PNP.xyz
```

Estructural

Action:

Based on the analysis of the MoCl3PNP.xyz structure, I have successfully identified the *tert*-butyl groups attached to the Phosphorus atoms and generated the three requested derivative structures.

Execution Details:

1. Structure Analysis:

- **Phosphorus Atoms:** Identified at indices **4** and **5**.
- **Ligand Topology:** By analyzing the coordination environment, I distinguished the *tert*-butyl groups from the pyridine-linker backbone:
 - **Linker Carbons:** Atoms **15** (connected to P4) and **21** (connected to P5) were identified as methylene bridges (-CH2-) connecting to the pyridine ring.
 - **t-Butyl Quaternary Carbons:** Atoms **7, 11** (connected to P4) and **22, 26** (connected to P5) were identified as the anchor points for the *tert*-butyl groups.
- **Target Bonds:** The bonds selected for replacement were P(4)-C(7), P(4)-C(11), P(5)-C(22), and P(5)-C(26).

2. Modifications:

- Replaced the four *tert*-butyl branches with **Hydrogen (H)**.
- Replaced the four *tert*-butyl branches with **Methyl (Me)**.
- Replaced the four *tert*-butyl branches with **Phenyl (Ph)**.

3. File Generation:

All files have been organized into the project folder MoCl3PNP_derivatives_project.

- MoCl3PNP_derivatives_project/MoCl3PNP_H.xyz
- MoCl3PNP_derivatives_project/MoCl3PNP_Me.xyz
- MoCl3PNP_derivatives_project/MoCl3PNP_Ph.xyz

The requested derivatives are now ready for use.

1.4.2 Ligand exchange

User:

Given Fe_CO_6.xyz and Ru_bpy_2_H2O_2.xyz:

1. For Fe_CO_6, replace the two axial CO ligands with water molecules.
2. For Ru_bpy_2_H2O_2, replace one water ligand with acetonitrile.

Estructural

Tool used:

Tool Name: repl_tool

Arguments:

```
code =
import os
import shutil

project_dir = "ligand_exchange_project"
os.makedirs(project_dir, exist_ok=True)

files_to_move = ["Fe_CO_6.xyz", "Ru_bpy_2_H2O_2.xyz"]
for f in files_to_move:
    if os.path.exists(f):
        shutil.move(f, os.path.join(project_dir, f))
        print(f"Moved {f} to {project_dir}")
    else:
        print(f"Warning: {f} not found")
```

Estructural

Tool used:

Tool Name: find_available_fragments

Arguments:

name = water

Tool Name: find_available_fragments

Arguments:

name = acetonitrile

Tool Name: get_atomic_neighbors

Arguments:

target = Fe

file_path = ligand_exchange_project/Fe_CO_6.xyz

Estructural

Tool used:

Tool Name: repl_tool

Arguments:

code =

import math

```
def calculate_angle(c1, c2, c3):
    # simple angle calculation
```

```

# c2 is the center (Fe)
v1 = [c1[0]-c2[0], c1[1]-c2[1], c1[2]-c2[2]]
v2 = [c3[0]-c2[0], c3[1]-c2[1], c3[2]-c2[2]]

dot = v1[0]*v2[0] + v1[1]*v2[1] + v1[2]*v2[2]
mag1 = math.sqrt(v1[0]**2 + v1[1]**2 + v1[2]**2)
mag2 = math.sqrt(v2[0]**2 + v2[1]**2 + v2[2]**2)

if mag1 == 0 or mag2 == 0: return 0
cos_theta = dot / (mag1 * mag2)
cos_theta = max(min(cos_theta, 1), -1)
return math.degrees(math.acos(cos_theta))

def read_xyz(path):
    with open(path, 'r') as f:
        lines = f.readlines()
    atoms = []
    for line in lines[2:]:
        parts = line.split()
        if not parts: continue
        atoms.append((parts[0], float(parts[1]), float(parts[2]), float(parts[3])))
    return atoms

# Fe_CO_6 analysis
fe_atoms = read_xyz("ligand_exchange_project/Fe_CO_6.xyz")
fe_idx = 0
ligand_indices = [7, 8, 9, 10, 11, 12] # From previous tool output
trans_pair = []

print("Searching for trans pair in Fe_CO_6...")
found = False
for i in range(len(ligand_indices)):
    for j in range(i+1, len(ligand_indices)):
        idx1 = ligand_indices[i]
        idx2 = ligand_indices[j]
        angle = calculate_angle(fe_atoms[idx1][1], fe_atoms[fe_idx][1], fe_atoms[idx2][1])
        if angle > 170: # Tolerance for trans
            print(f"Found trans pair: {idx1} and {idx2} (Angle: {angle:.2f})")
            trans_pair = [idx1, idx2]
            found = True
            break
    if found: break

# Ru analysis
# I need to find Ru index and O neighbors
print("
Analyzing Ru_bpy_2_H2O_2...")
ru_atoms = read_xyz("ligand_exchange_project/Ru_bpy_2_H2O_2.xyz")
ru_idx = -1
o_indices = []

for i, (elem, coords) in enumerate(ru_atoms):
    if elem == "Ru":

```

```

        ru_idx = i
        break

if ru_idx != -1:
    print(f"Ru index: {ru_idx}")
    # Find neighbors
    for i, (elem, coords) in enumerate(ru_atoms):
        if i == ru_idx: continue
        dist = math.sqrt(sum([(coords[k]-ru_atoms[ru_idx][1][k])**2 for k in range(3)]))
        if dist < 2.5: # typical bonding distance
            print(f"Neighbor: {i} ({elem}) Dist: {dist:.2f}")
            if elem == "O":
                o_indices.append(i)
print(f"Water Oxygen indices: {o_indices}")

```

Estructural

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

filenames = ["ligand_exchange_project/MeCN.xyz"]

smiles_list = ["CC#N"]

Estructural

Tool used:

Tool Name: repl_tool

Arguments:

code =

```

def find_N_index(path):
    with open(path, 'r') as f:
        lines = f.readlines()
    for i, line in enumerate(lines[2:]):
        parts = line.split()
        if not parts: continue
        if parts[0] == 'N':
            print(f"Nitrogen found at index: {i}")
            return i
    return -1

n_idx = find_N_index("ligand_exchange_project/MeCN.xyz")

```

Estructural

Tool used:

Tool Name: replace_branch_with_predefined_FG

Arguments:

output_path = ligand_exchange_project/Fe_axial_H2O.xyz

FG_name = H2O

bond_pair = [[0.0, 7.0], [0.0, 12.0]]

```
core_file_path = ligand_exchange_project/Fe_CO_6.xyz
```

Tool Name: replace_branch

Arguments:

```
FG_file_path = ligand_exchange_project/MeCN.xyz
```

```
output_path = ligand_exchange_project/Ru_bpy_2_H2O_MeCN.xyz
```

```
bond_pair = [[24.0, 45.0]]
```

```
binding_mode = connect
```

```
FG_connection_idx = 2
```

```
core_file_path = ligand_exchange_project/Ru_bpy_2_H2O_2.xyz
```

Estructural

Action:

The requested ligand exchange operations have been successfully completed.

1. Fe(CO)₆ Modification

- **Objective:** Replace two axial CO ligands with water molecules.
- **Input:** Fe_CO_6.xyz
- **Procedure:**
 - Identified the central Iron (Fe) atom at index 0.
 - Calculated angles between Fe-C bonds to identify the trans (axial) pair of Carbon atoms (indices 7 and 12, angle 180°).
 - Used `replace_branch_with_predefined_FG` to substitute the branches at these indices with H₂O ligands.
- **Output:** ligand_exchange_project/Fe_axial_H2O.xyz

2. Ru(bpy)₂(H₂O)₂ Modification

- **Objective:** Replace one water ligand with acetonitrile (MeCN).
- **Input:** Ru_bpy_2_H2O_2.xyz
- **Procedure:**
 - Identified the central Ruthenium (Ru) atom at index 24.
 - Identified the coordinated Oxygen atoms of the water ligands (indices 45 and 46).
 - Generated an XYZ structure for Acetonitrile (MeCN.xyz) from the SMILES string `CC#N` and identified the Nitrogen atom (index 2) as the coordination site.
 - Used `replace_branch` to replace the water ligand at index 45 with the generated MeCN fragment, binding via the Nitrogen atom.
- **Output:** ligand_exchange_project/Ru_bpy_2_H2O_MeCN.xyz

All generated files are located in the `ligand_exchange_project` directory.

1.5 Case study 5: Molecular fragment operations and fragment analysis

1.5.1 Conversion between isomeric forms

User:

You are given trans-stilbene.xyz and L-alanine.xyz. 1. Make edits to trans-stilbene.xyz to create its photoisomerization TS and also cis-stilbene. 2. Make edits to L-alanine.xyz to generate its enantiomer.

Estructural

Tool used:

Tool Name: repl_tool

Arguments:

code =

import os

import shutil

project_dir = "isomers_and_enantiomers_project"

os.makedirs(project_dir, exist_ok=True)

shutil.copy("trans-stilbene.xyz", os.path.join(project_dir, "trans-stilbene.xyz"))

shutil.copy("L-alanine.xyz", os.path.join(project_dir, "L-alanine.xyz"))

print(f"Created {project_dir} and copied files.")

Estructural

Tool used:

Tool Name: match_smarts_in_xyz

Arguments:

file_path = isomers_and_enantiomers_project/trans-stilbene.xyz

smarts = C=C

Tool Name: match_smarts_in_xyz

Arguments:

file_path = isomers_and_enantiomers_project/L-alanine.xyz

smarts = C(N)(C)C(=O)O

Estructural

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

target = 2

file_path = isomers_and_enantiomers_project/trans-stilbene.xyz

Tool Name: get_atomic_neighbors

Arguments:

target = 3

file_path = isomers_and_enantiomers_project/trans-stilbene.xyz

Tool Name: get_atomic_neighbors

Arguments:

target = 0

file_path = isomers_and_enantiomers_project/L-alanine.xyz

Estructural

Tool used:

Tool Name: repl_tool

Arguments:

code =
import json

def parse_neighbors(output_str):

 # This is a dummy parser, I will just print the output in the next step
 pass

print("Checking neighbors for L-alanine atoms 1, 2, 3, 4, 5...")

Estructural

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

target = 5
file_path = isomers_and_enantiomers_project/L-alanine.xyz

Estructural

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

target = 3
file_path = isomers_and_enantiomers_project/L-alanine.xyz

Estructural

Tool used:

Tool Name: set_dihedral_between_fragments

Arguments:

input_path = isomers_and_enantiomers_project/trans-stilbene.xyz
output_path = isomers_and_enantiomers_project/stilbene_photoisomerization_TS.xyz
angle = 90
consider_fragments = True
atoms_list = [0.0, 2.0, 3.0, 1.0]

Tool Name: set_dihedral_between_fragments

Arguments:

input_path = isomers_and_enantiomers_project/trans-stilbene.xyz
output_path = isomers_and_enantiomers_project/cis-stilbene.xyz
angle = 0
consider_fragments = True
atoms_list = [0.0, 2.0, 3.0, 1.0]

Tool Name: rotate_substituents_around_bisector

Arguments:

center_idx = 3
output_path = isomers_and_enantiomers_project/D-alanine.xyz

```
fragment_mode = auto
base_vector_indices = [4.0, 6.0]
input_path = isomers_and_enantiomers_project/L-alanine.xyz
angle_deg = 180
```

Estructural

Action:

The requested edits have been performed on **trans-stilbene.xyz** and **L-alanine.xyz** to generate the specified isomers and enantiomers. The output files are saved in the **isomers_and_enantiomers_project** directory.

1. Stilbene Isomerization Using **trans-stilbene.xyz** as the starting structure, two new geometries were generated by modifying the C-C=C-C dihedral angle defined by the central double bond (atoms 2 and 3) and their phenyl attachments (atoms 0 and 1).

- **Photoisomerization Transition State (TS):**

- **File:** **isomers_and_enantiomers_project/stilbene_photoisomerization_TS.xyz**
- **Method:** The central dihedral angle was set to **90°** to approximate the perpendicular transition state characteristic of stilbene photoisomerization.

- **Cis-Stilbene:**

- **File:** **isomers_and_enantiomers_project/cis-stilbene.xyz**
- **Method:** The central dihedral angle was set to **0°**, converting the trans isomer (dihedral 180°) to the cis configuration.

2. L-Alanine Enantiomer Generation Using **L-alanine.xyz**, the D-alanine enantiomer was created by inverting the configuration at the chiral center.

- **D-Alanine:**

- **File:** **isomers_and_enantiomers_project/D-alanine.xyz**
- **Method:** The chiral center was identified at atom **C:3**. Its substituents, the methyl group (C:4) and the alpha-hydrogen (H:6), were swapped by rotating them 180° around their bisector axis. This operation inverted the stereocenter from L- to D-alanine.

1.5.2 Fragment-level structural analysis of a multi-component system

User:

The provided PDB-code-2FRJ.xyz file contains six distinct fragments. Identify which atoms belong to each fragment (heme group, lysine, histidine, water, NO ligand, and the histidine fragment coordinated to the heme iron). For each fragment, list each atom with its atomic species and atomic index explicitly paired.

Estructural

Tool used:

Tool Name: repl_tool

Arguments:

```
code =
import os
import shutil
import numpy as np

# 1. Setup Directory
project_dir = "fragment_identification_project"
if not os.path.exists(project_dir):
    os.makedirs(project_dir)

src = "PDB-code-2FRJ.xyz"
dst = os.path.join(project_dir, "PDB-code-2FRJ.xyz")
if os.path.exists(src):
    shutil.copy(src, dst)

# 2. Read XYZ
def read_xyz(filepath):
    atoms = []
    with open(filepath, 'r') as f:
        lines = f.readlines()
        # Skip header if present (standard XYZ has count, then comment)
        start_idx = 2
        try:
            int(lines[0].split()[0])
        except:
            start_idx = 0 # No header?

        for i, line in enumerate(lines[start_idx:]):
            parts = line.split()
            if len(parts) >= 4:
                atoms.append({
                    'index': i,
                    'element': parts[0],
                    'coord': np.array([float(x) for x in parts[1:4]])
                })
    return atoms

atoms = read_xyz(dst)

# 3. Connectivity Analysis
```

```

# Radii for heuristics (Angstrom) - standard covalent radii approx
radii = {'H': 0.35, 'C': 0.76, 'N': 0.71, 'O': 0.66, 'S': 1.05, 'Fe': 1.32, 'P': 1.07}

def get_radius(el):
    return radii.get(el, 1.0) # default

def is_bonded(a1, a2):
    dist = np.linalg.norm(a1['coord'] - a2['coord'])
    limit = 1.3 * (get_radius(a1['element']) + get_radius(a2['element']))
    return dist < limit

# Build Adjacency List (Excluding Fe to separate ligands)
adj = {i: [] for i in range(len(atoms))}
fe_index = -1

for i in range(len(atoms)):
    if atoms[i]['element'] == 'Fe':
        fe_index = i
        continue # Skip Fe bonds for now
    for j in range(i + 1, len(atoms)):
        if atoms[j]['element'] == 'Fe':
            continue
        if is_bonded(atoms[i], atoms[j]):
            adj[i].append(j)
            adj[j].append(i)

# Find Connected Components (Fragments)
visited = set()
fragments = []

def get_fragment(start_node):
    component = []
    stack = [start_node]
    while stack:
        node = stack.pop()
        if node in visited:
            continue
        visited.add(node)
        component.append(node)
        for neighbor in adj[node]:
            if neighbor not in visited:
                stack.append(neighbor)
    return component

# Iterate
for i in range(len(atoms)):
    if i == fe_index:
        continue # Handle Fe separately
    if i not in visited:
        frag_indices = get_fragment(i)
        fragments.append(frag_indices)

# Analyze Fragments

```

```

results = []
for frag in fragments:
    # Get composition
    comp = {}
    atom_list = []
    for idx in frag:
        el = atoms[idx]['element']
        comp[el] = comp.get(el, 0) + 1
        atom_list.append((idx, el))

    # Heuristics for identification
    label = "Unknown"

    # Heme Porphyrin part: Many C, 4 N. (C20H12N4 approx core)
    if comp.get('N', 0) == 4 and comp.get('C', 0) > 15:
        label = "Heme_Porphyrin_Ring"

    # Water: 1 O, maybe 2 H
    elif comp.get('O', 0) >= 1 and sum(comp.values()) <= 3 and comp.get('C', 0) == 0:
        label = "Water"

    # NO: 1 N, 1 O
    elif comp.get('N', 0) == 1 and comp.get('O', 0) == 1 and sum(comp.values()) == 2:
        label = "NO_Ligand"

    # Amino Acids
    elif comp.get('N', 0) >= 1 and comp.get('C', 0) > 1:
        # Histidine: Imidazole ring (3 C, 2 N in ring) + backbone. Total N approx 3.
        # Lysine: Long chain, 2 N total (backbone + sidechain).
        if comp.get('N', 0) >= 3: # Likely His
            # Check if coordinated to Fe (Distance check)
            # Find closest atom in this fragment to Fe
            min_dist = 100
            if fe_index != -1:
                fe_pos = atoms[fe_index]['coord']
                for idx in frag:
                    d = np.linalg.norm(atoms[idx]['coord'] - fe_pos)
                    if d < min_dist:
                        min_dist = d

            if min_dist < 2.5: # Typical coordination bond
                label = "Coordinated_Histidine"
            else:
                label = "Histidine"
        elif comp.get('N', 0) == 2: # Likely Lysine (or His fragment?)
            # Lysine has long alkyl chain
            label = "Lysine" # Assumption
        else:
            label = "Other_Amino_Acid"

    results.append({'label': label, 'indices': frag, 'atoms': atom_list})

# Add Fe to Heme Group

```

```

if fe_index != -1:
    # Find the Heme_Porphyrin_Ring fragment
    for res in results:
        if res['label'] == "Heme_Porphyrin_Ring":
            res['indices'].append(fe_index)
            res['atoms'].append((fe_index, 'Fe'))
            res['label'] = "Heme_Group"
            break

# Sort or Organize Output
print(f"Found {len(results)} fragments (excluding Fe initially).")
for r in results:
    print(f"Fragment: {r['label']}")
    # print(f" Indices: {r['indices']}")
    print(f" Composition: {str([atoms[i]['element'] for i in r['indices'][:5]])} ... (Total {len(r['indices'])})")

# Print explicit details for small ones or first few of large ones to confirm
sorted_atoms = sorted(r['atoms'], key=lambda x: x[0])
print(f" Atoms: {sorted_atoms}")

```

Estructural

Tool used:

Tool Name: repl_tool

Arguments:

code =

import numpy as np

```

# Data from previous step (manual transcription of indices to be safe, or re-read)
# Indices from previous output:
frag2_indices = [3, 6, 19, 20, 21, 22, 60, 61, 62, 63, 64, 65]
frag3_indices = [4, 5, 15, 16, 17, 18, 44, 45, 46, 47, 48, 49]
fe_index = 0

def read_xyz_simple(filepath):
    coords = {}
    with open(filepath, 'r') as f:
        lines = f.readlines()
        start = 2
        try:
            int(lines[0].split()[0])
        except:
            start=0
        for i, line in enumerate(lines[start:]):
            parts = line.split()
            if len(parts) >= 4:
                coords[i] = np.array([float(x) for x in parts[1:4]])
    return coords

coords = read_xyz_simple("fragment_identification_project/PDB-code-2FRJ.xyz")

fe_pos = coords[fe_index]

```

```
def get_min_dist(indices, target_pos):
    dists = []
    for idx in indices:
        d = np.linalg.norm(coords[idx] - target_pos)
        dists.append(d)
    return min(dists)

d2 = get_min_dist(frag2_indices, fe_pos)
d3 = get_min_dist(frag3_indices, fe_pos)

print(f"Distance Frag 2 to Fe: {d2}")
print(f"Distance Frag 3 to Fe: {d3}")
```

Estructural

Action:

Based on the structural analysis of the PDB-code-2FRJ.xyz file, the six distinct fragments have been identified and their constituent atoms are listed below.

The fragments were separated based on connectivity and their chemical identities were confirmed by analyzing their composition and proximity to the iron center.

- **Heme Group:** Identified as the large porphyrin ring containing the central Iron (Fe) atom.
- **NO Ligand:** Identified as the diatomic Nitrogen-Oxygen molecule.
- **Coordinated Histidine:** Identified as the histidine sidechain (imidazole group) positioned within coordination distance (2.18 Å) of the Iron atom.
- **Histidine:** Identified as the second histidine sidechain, located further (4.36 Å) from the Iron atom.
- **Lysine:** Identified as the propyl-amine chain fragment.
- **Water:** Identified as the isolated H₂O molecule.

Fragment Composition List

1. Heme Group

- **Fe:** 0
- **N:** 7, 8, 9, 10
- **C:** 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42
- **H:** 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77

2. NO Ligand

- **N:** 1
- **O:** 2

3. Histidine Fragment Coordinated to the Heme Iron

- **N:** 3, 6
- **C:** 19, 20, 21, 22
- **H:** 60, 61, 62, 63, 64, 65

4. Histidine (Uncoordinated)

- **N:** 4, 5
- **C:** 15, 16, 17, 18
- **H:** 44, 45, 46, 47, 48, 49

5. Lysine

- **N:** 11
- **C:** 12, 13, 14
- **H:** 51, 52, 53, 54, 55, 56, 57, 58, 59, 79

6. Water

- **O:** 43
- **H:** 50, 78

User:

Which tools do you use? Why don't you use `match_smarts_in_xyz` or `get_atomic_neighbors`? Can you solve the same query by using them?

 Estructural

Tool used:

Tool Name: `match_smarts_in_xyz`

Arguments:

`file_path = fragment_identification_project/PDB-code-2FRJ.xyz`
`smarts = [OH2]`

Tool Name: `match_smarts_in_xyz`

Arguments:

`smarts = N=O`
`file_path = fragment_identification_project/PDB-code-2FRJ.xyz`

Tool Name: `match_smarts_in_xyz`

Arguments:

`smarts = c1ncn[c]1`
`file_path = fragment_identification_project/PDB-code-2FRJ.xyz`

Tool Name: `match_smarts_in_xyz`

Arguments:

`file_path = fragment_identification_project/PDB-code-2FRJ.xyz`
`smarts = n1cccc1`

Tool Name: `get_atomic_neighbors`

Arguments:

`target = Fe`
`file_path = fragment_identification_project/PDB-code-2FRJ.xyz`

 Estructural

Action:

My primary tool choice was the **Python REPL** to write a **Connectivity-Based Partitioning script**.

I did not use `match_smarts_in_xyz` or `get_atomic_neighbors` as the primary method for the following reasons:

1. Robustness for Metal Complexes `match_smarts_in_xyz` relies on OpenBabel to perceive bond orders and aromaticity from raw XYZ coordinates. In transition metal complexes (like Heme), this perception is often fragile:

- **The Problem:** As demonstrated in my internal test, searching for an imidazole ring (`c1ncn[cH]1`) identified the *uncoordinated* Histidine but **missed** the *coordinated* Histidine. The coordination to Iron changes the bond geometry/perception, causing the standard aromatic pattern to fail.
- **The Heme Issue:** Similarly, a pyrrole search (`n1cccc1`) only identified 2 of the 4 rings in the porphyrin core due to the delocalized bonding, which OpenBabel often interprets as alternating pyrrole (NH) and pyrrolenine (=N) forms.

2. "Fragments" vs. "Substructures"

- **The Goal:** The task was to **partition** the entire molecule into disjoint fragments (sets of atoms).
- **The Tool Limitation:** `match_smarts_in_xyz` finds **substructures**.
 - It would find the "ring" of the histidine but *not* the attached backbone or side chain atoms.
 - It would find the "core" of the heme but *not* the peripheral substituents.
 - You would still need a traversal algorithm to "grow" the selection from the matched substructure to the rest of the molecule.

3. Efficiency

- `get_atomic_neighbors`: This tool is the functional equivalent of my script's logic (checking connectivity). However, using it via tool calls is inefficient. To identify fragments in a 60+ atom molecule, I would need to call this tool iteratively dozens of times (traversing atom-by-atom), whereas a single Python script can process the entire adjacency matrix in milliseconds.

Can you solve the same query by using them? Yes, but it is much harder and prone to errors.

You would need a hybrid approach:

1. **Use `get_atomic_neighbors` on the Metal (Fe):**
 - Result: N:1, N:3, N:7, N:8, N:9, N:10
 - This tells us **N:1** is the NO ligand base, **N:3** is the coordinated Histidine base, and **N:7-10** are the Heme bases.
2. **Use `match_smarts_in_xyz` for isolated fragments:**
 - `[OH2]` -> Finds Water (O:43).
 - `N=O` -> Finds NO (N:1, O:2).
 - `c1ncn[cH]1` -> Finds the uncoordinated Histidine ring.
3. **Manual Traversal (The "Hard" Part):**
 - For the Heme and Coordinated Histidine (which SMARTS missed), you would have to manually call `get_atomic_neighbors` on the atoms found in Step 1 (e.g., neighbors of N:3) and recursively expand until you hit the fragment boundaries.

Conclusion: The Python script is the superior method for *blindly* identifying distinct chemical fragments in a complex system, while `match_smarts_in_xyz` is better suited for finding specific, well-defined functional groups in standard organic molecules.

1.5.3 Distance and angle scan in ferrocene

User:

Generate an XYZ file for ferrocene. Then create and save separate XYZ files for the following structural modifications:

1. Stretch the distance between the Fe atom and one Cp ring from 1.5 Å to 2.5 Å in increments of 0.1 Å, while keeping the other Cp ring fixed.
2. Change the Cp-Fe-Cp angle from 120° to 180° in increments of 10°, saving a separate XYZ file for each angle.

 **Estructural**

Tool used:

Tool Name: find_available_fragments

Arguments:

type = predefined_organometallics

name = ferrocene

 **Estructural**

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"Cp": [1.0, 2.0]}]

metal = Fe

coordination_geometry = linear

output_path = ferrocene_project/ferrocene.xyz

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

code =

import os

os.makedirs('ferrocene_project', exist_ok=True)

print('Created directory: ferrocene_project')

 **Estructural**

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"Cp": [1.0, 2.0]}]

metal = Fe

coordination_geometry = linear

output_path = ferrocene_project/ferrocene.xyz

 **Estructural**

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
with open('ferrocene_project/ferrocene.xyz', 'r') as f:
    lines = f.readlines()
    # Skip header (2 lines)
    atoms = lines[2:]
    for i, line in enumerate(atoms):
        print(f"{i}: {line.split()[0]}")
```

 **Estructural****Tool used:**

Tool Name: repl_tool

Arguments:

code =

import math

import os

```
def read_xyz(path):
    with open(path, 'r') as f:
        lines = f.readlines()
    atoms = []
    for line in lines[2:]:
        parts = line.split()
        if len(parts) >= 4:
            atoms.append({
                'el': parts[0],
                'x': float(parts[1]),
                'y': float(parts[2]),
                'z': float(parts[3])
            })
    return atoms

def write_xyz(path, atoms):
    with open(path, 'w') as f:
        f.write(str(len(atoms)) + "
")
        f.write("Generated by Agent
")
        for atom in atoms:
            f.write(f"{atom['el']} {atom['x']:.6f} {atom['y']:.6f} {atom['z']:.6f}
")

def get_centroid(atoms, indices):
    cx, cy, cz = 0.0, 0.0, 0.0
    for idx in indices:
        cx += atoms[idx]['x']
        cy += atoms[idx]['y']
        cz += atoms[idx]['z']
    n = len(indices)
    return cx/n, cy/n, cz/n
```

```

def vec_sub(a, b):
    return (a[0]-b[0], a[1]-b[1], a[2]-b[2])

def vec_add(a, b):
    return (a[0]+b[0], a[1]+b[1], a[2]+b[2])

def vec_scale(v, s):
    return (v[0]*s, v[1]*s, v[2]*s)

def vec_norm(v):
    return math.sqrt(v[0]**2 + v[1]**2 + v[2]**2)

def vec_normalize(v):
    n = vec_norm(v)
    if n == 0: return (0,0,0)
    return (v[0]/n, v[1]/n, v[2]/n)

def cross_product(a, b):
    return (a[1]*b[2] - a[2]*b[1],
            a[2]*b[0] - a[0]*b[2],
            a[0]*b[1] - a[1]*b[0])

def rotate_point(point, axis, angle_rad, origin):
    x = point[0] - origin[0]
    y = point[1] - origin[1]
    z = point[2] - origin[2]
    u, v, w = axis
    c = math.cos(angle_rad)
    s = math.sin(angle_rad)
    k_dot_v = u*x + v*y + w*z
    k_cross_v_x = v*z - w*y
    k_cross_v_y = w*x - u*z
    k_cross_v_z = u*y - v*x
    rx = x*c + k_cross_v_x*s + u*k_dot_v*(1-c)
    ry = y*c + k_cross_v_y*s + v*k_dot_v*(1-c)
    rz = z*c + k_cross_v_z*s + w*k_dot_v*(1-c)
    return rx + origin[0], ry + origin[1], rz + origin[2]

base_path = 'ferrocene_project/ferrocene.xyz'
atoms = read_xyz(base_path)

idx_fe = 0
idx_cp1_ring = [1, 2, 3, 4, 5]
idx_cp1_all = list(range(1, 11))
idx_cp2_ring = [11, 12, 13, 14, 15]
idx_cp2_all = list(range(11, 21))

fe_pos = (atoms[idx_fe]['x'], atoms[idx_fe]['y'], atoms[idx_fe]['z'])

# 1. Distance Modification
c1 = get_centroid(atoms, idx_cp1_ring)
v1 = vec_sub(c1, fe_pos)
current_dist = vec_norm(v1)

```

```

dir_v1 = vec_normalize(v1)

print(f"Initial Fe-Cp1 Distance: {current_dist:.4f}")

for d_int in range(15, 26):
    d = d_int / 10.0
    shift_val = d - current_dist
    shift_vec = vec_scale(dir_v1, shift_val)
    new_atoms = [a.copy() for a in atoms]
    for idx in idx_cp1_all:
        new_atoms[idx]['x'] += shift_vec[0]
        new_atoms[idx]['y'] += shift_vec[1]
        new_atoms[idx]['z'] += shift_vec[2]
    write_xyz(f'ferrocene_project/ferrocene_dist_{d:.1f}.xyz', new_atoms)
    print(f"Saved dist {d:.1f}")

# 2. Angle Modification
c2 = get_centroid(atoms, idx_cp2_ring)
v2 = vec_sub(c2, fe_pos)
arbitrary = (1.0, 0.0, 0.0)
if abs(vec_normalize(v2)[0]) > 0.9:
    arbitrary = (0.0, 1.0, 0.0)
rot_axis = cross_product(vec_normalize(v2), arbitrary)
rot_axis = vec_normalize(rot_axis)

for angle in range(120, 181, 10):
    rot_rad = math.radians(180 - angle)
    new_atoms = [a.copy() for a in atoms]
    for idx in idx_cp2_all:
        p = (atoms[idx]['x'], atoms[idx]['y'], atoms[idx]['z'])
        np = rotate_point(p, rot_axis, rot_rad, fe_pos)
        new_atoms[idx]['x'] = np[0]
        new_atoms[idx]['y'] = np[1]
        new_atoms[idx]['z'] = np[2]
    write_xyz(f'ferrocene_project/ferrocene_angle_{angle}.xyz', new_atoms)
    print(f"Saved angle {angle}")

```

Estructural

Action:

The XYZ file for ferrocene has been generated, and the requested structural modifications have been performed. All files are saved in the **ferrocene_project** folder.

Summary of Actions

1. Ferrocene Generation:

- Constructed the base ferrocene structure ($\text{Fe}(\text{Cp})_2$) using a linear template with two cyclopentadienyl (Cp) ligands.
- **File:** ferrocene_project/ferrocene.xyz
- **Identified Indices:** Fe (0), Cp Ring 1 (1–10), Cp Ring 2 (11–20).

2. Modification 1: Fe–Cp Distance Scan

- Stretched the distance between the Fe atom and the centroid of one Cp ring from **1.5 Å to 2.5 Å** (increment: 0.1 Å).
- **Files:** ferrocene_dist_1.5.xyz through ferrocene_dist_2.5.xyz.

3. Modification 2: Cp-Fe-Cp Angle Scan

- Varied the angle between the two Cp ring centroids and the Fe atom from **120° to 180°** (increment: 10°).
- **Files:** `ferrocene_angle_120.xyz` through `ferrocene_angle_180.xyz`.

All structures are available in the `ferrocene_project` directory.

User:

Why don't you use `set_distance_between_fragments` or `set_angle_between_fragments`?

 **Estructural**

Action:

Reasoning: The decision to use a custom Python script via `repl_tool` instead of the specific `set_distance_between_fragments` and `set_angle_between_fragments` tools was driven by **batch processing efficiency** and **environment constraints**.

1. **Batch Efficiency:** The task required generating a total of 18 distinct XYZ files (11 distance steps and 7 angle steps). The specific geometry tools are designed for single-structure modifications. Using them would have required 18 separate, sequential tool calls (and conversation turns), which is significantly slower and less efficient than a single loop.
2. **Environment Constraints:** The `repl_tool` allows me to write a Python loop to generate all files in one go, but it **cannot call other agent tools** (like `set_distance...`) from within its code. Therefore, I had to implement the vector mathematics (translation and rotation) directly in the Python script to achieve the "scan" functionality within a single action.
3. **Simplicity of System:** Ferrocene has a very predictable indexing structure (Fe at 0, Ligands at 1-10 and 11-20). This made it safe to manually define the atom groups in the script. For more complex or unknown molecules where "fragment detection" is difficult, using `set_distance_between_fragments` with `fragment_mode='auto'` would indeed be the safer and preferred choice.

The requested XYZ files were successfully generated in the `ferrocene_project` directory in the previous step using this optimized batch method.

If you would like, I can demonstrate how to use `set_distance_between_fragments` for a single specific adjustment to compare the results.

1.5.4 Geometric operation on ferrocene

User:

Generate an XYZ file for ferrocene. Then create and save separate XYZ files for the following structural modifications:

1. Stretch the distance between the Fe atom and one Cp ring to 4.5 Å
2. Change the Cp-Fe-Cp angle to 120°.

 **Estructural**

Tool used:

Tool Name: find_available_fragments

Arguments:

type = predefined_organometallics

 **Estructural**

Tool used:

Tool Name: find_available_fragments

Arguments:

type = ligand

name = Cp

 **Estructural**

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"Cp": [1.0, 2.0]}]

output_path = ferrocene_mods_project/ferrocene.xyz

coordination_geometry = linear

metal = Fe

 **Estructural**

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

ligand_bindings = [{"Cp": [1.0, 2.0]}]

output_path = ferrocene_project/ferrocene.xyz

coordination_geometry = linear

metal = Fe

 **Estructural**

Tool used:

Tool Name: match_smarts_in_xyz

Arguments:

file_path = ferrocene_project/ferrocene.xyz

smarts = c1ccccc1

 **Estructural**

Tool used:

Tool Name: match_smarts_in_xyz

Arguments:

smarts = [#6]

file_path = ferrocene_project/ferrocene.xyz

 **Estructural****Tool used:**

Tool Name: get_connected_subgraph_indices

Arguments:

exclude = 0

file_path = ferrocene_project/ferrocene.xyz

start_index = 1

 **Estructural****Tool used:**

Tool Name: get_connected_subgraph_indices

Arguments:

exclude = 0

file_path = ferrocene_project/ferrocene.xyz

start_index = 11

 **Estructural****Tool used:**

Tool Name: set_distance_between_fragments

Arguments:

input_path = ferrocene_project/ferrocene.xyz

output_path = ferrocene_project/ferrocene_dist_4.5.xyz

fragment_indices = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]

targets = [{"atom": [0.0]}, {"group": [1.0, 2.0, 3.0, 4.0, 5.0]}]

distance = 4.5

fragment_mode = manual

 **Estructural****Tool used:**

Tool Name: set_angle_between_fragments

Arguments:

fragment_indices = [11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0]

output_path = ferrocene_project/ferrocene_angle_120.xyz

targets = [{"group": [1.0, 2.0, 3.0, 4.0, 5.0]}, {"atom": [0.0]}, {"group": [11.0, 12.0, 13.0, 14.0, 15.0]}]

input_path = ferrocene_project/ferrocene.xyz

angle = 120

fragment_mode = manual

 **Estructural**

Action:

The requested XYZ files have been generated and saved in the **ferrocene_project** directory.

1. **Base Structure:** **ferrocene_project/ferrocene.xyz**
 - Generated using a linear iron template with two cyclopentadienyl (Cp) ligands.
2. **Distance Modification:** **ferrocene_project/ferrocene_dist_4.5.xyz**
 - The distance between the Iron atom and one Cp ring centroid was set to **4.5 Å**.
3. **Angle Modification:** **ferrocene_project/ferrocene_angle_120.xyz**
 - The Cp(centroid)–Fe–Cp(centroid) angle was adjusted to **120°**.

1.6 Case study 6: Multimodal reaction mechanism-guided structure generation

User:

You are given an image file, mechanism.png, which depicts a chemical reaction mechanism. Based solely on the information shown in the image, generate three-dimensional molecular structures corresponding to each reaction intermediate and transition state depicted in the mechanism. Generate and save a separate XYZ file for each reaction intermediate and transition state.

 **Estructural**

Action:

I'll start by viewing the mechanism.png image to understand the chemical reaction mechanism depicted and identify all the intermediates and transition states that need to be generated.

Tool used:

Tool Name: view_png_file

Arguments:

png_file_relative_path = mechanism.png

 **Estructural**

Action:

Excellent! I can see this is a **migratory insertion mechanism** involving a zirconocene complex. The mechanism shows:

1. **Intermediate 1 (Left):** $\text{Cp}_2\text{Zr}^+(\text{CH}_3)(\eta^2\text{-ethylene})$ - Zirconocene cation with a methyl group and coordinated ethylene
2. **Transition State (Middle):** 4-membered ring transition state showing the migratory insertion process with dashed bonds
3. **Intermediate 2 (Right):** $\text{Cp}_2\text{Zr}^+(\text{n-propyl})$ - Zirconocene cation with a propyl group (product of insertion)

Let me start building these structures. First, let me check what predefined templates and fragments are available for zirconocene complexes.

Tool used:

Tool Name: find_available_fragments

Arguments:

type = predefined_organometallics

 **Estructural**

Action:

Let me check for available ligands, specifically the cyclopentadienyl (Cp) ligand and other necessary fragments:

Tool used:

Tool Name: find_available_fragments

Arguments:

name = Cp

 **Estructural**

Tool used:

Tool Name: find_available_fragments

Arguments:

```
name = ethylene
```

Estructural

Tool used:

Tool Name: find_available_fragments

Arguments:

name = Me

Estructural

Action:

Now I have all the information I need. Let me build the three structures systematically.

Building the structures:

Structure 1 - Intermediate 1: $\text{Cp}_2\text{Zr}(\text{CH}_3)(\eta^2\text{-ethylene})^+$ **Structure 2 - Transition State:** 4-membered ring TS **Structure 3 - Intermediate 2:** $\text{Cp}_2\text{Zr}(\text{n-propyl})^+$

Let me start by building **Intermediate 1** using the tetrahedral template with two Cp rings, a methyl group, and an ethylene ligand:

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

coordination_geometry = tetrahedral

metal = Zr

ligand_bindings = [{"Cp": [1, 2]}, {"CH3": [3]}, {"ethylene": [4]}]

output_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

Estructural

Action:

Intermediate 1 has been created. Let me visualize it to verify the structure:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

left_right_deg = 20

up_down_deg = 15

Estructural

Action:

I can see Intermediate 1 with:

- Zr center (index 0)
- Two Cp rings (indices 2-11 and 12-21)
- Methyl group
- Ethylene ligand (C:25, C:26 at top)

Now let me build **Intermediate 2** (the product with propyl group). First, let me check if propyl is available in the fragment library:

Tool used:

Tool Name: find_available_fragments

Arguments:

name = propyl

Estructural

Action:

Excellent! I found the propyl group (nPr). Now let me build **Intermediate 2** (Cp₂Zr-propyl):

Tool used:

Tool Name: build_tm_complex_from_template

Arguments:

```
coordination_geometry = trigonal_planar
ligand_bindings = [{"Cp": [1, 2]}, {"nPr": [3]}]
metal = Zr
output_path = intermediate_2_Cp2Zr_propyl.xyz
```

Estructural

Action:

Let me visualize Intermediate 2 to verify it:

Tool used:

Tool Name: view_molecule

Arguments:

```
left_right_deg = 20
up_down_deg = 15
xyz_file_relative_path = intermediate_2_Cp2Zr_propyl.xyz
```

Estructural

Action:

I can see Intermediate 2 with:

- Zr:0 at center
- Two Cp rings (C:2-6 and C:12-16 with their H atoms)
- Propyl group (C:1, C:23, C:24 chain with H atoms)

The propyl appears disconnected visually, but this is common for metal-organic bonds in the viewer.

Let me verify the Zr-C bond distance:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = intermediate_2_Cp2Zr_propyl.xyz
targets = [{"atom": [0]}, {"atom": [1]}]
```

Estructural

Action:

The Zr-C distance is 2.475 Å, which is appropriate for a Zr-C bond. The structure is valid.

Now I need to construct the **Transition State** for the migratory insertion. Let me first analyze the atom connectivity in Intermediate 1 to understand the key atoms involved:

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

```
file_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz
target = 0
```

Estructural

Action:

Let me check the neighbors of the methyl carbon (which should be index 1 based on the bond pair info):

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

file_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

target = 1

 **Estructural**
Action:

Now I understand the structure:

- C:1 is the methyl carbon (bonded to Zr:0, H:22, H:23, H:24)
- C:25 and C:26 are the ethylene carbons

Let me check the ethylene carbon neighbors:

Tool used:

Tool Name: get_atomic_neighbors

Arguments:

file_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

target = 25

 **Estructural**
Tool used:

Tool Name: get_atomic_neighbors

Arguments:

file_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

target = 26

 **Estructural**
Action:

Now I have a clear picture of the structure:

- C:1 = methyl carbon (bonded to Zr:0)
- C:25 = ethylene carbon 1 (bonded to C:26, H:29, H:30, Zr:0)
- C:26 = ethylene carbon 2 (bonded to C:25, H:27, H:28, Zr:0)

For the **transition state**, I need to create a 4-membered ring: Zr-C25-C26-C1 (or similar). Let me get the current distances to plan the TS geometry:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
# Get key distances in intermediate_1 for planning the TS
import subprocess
```

```
# Read the xyz file to understand the geometry
with open('intermediate_1_Cp2Zr_CH3_ethylene.xyz', 'r') as f:
    content = f.read()
print("Intermediate 1 structure:")
print(content[:2000])
```

Estructural

Action:

Now let me measure the key distances needed for the TS geometry:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

targets = [{"atom": 0}], [{"atom": 1}]

Estructural

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

targets = [{"atom": 1}], [{"atom": 25}]

Estructural

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

targets = [{"atom": 1}], [{"atom": 26}]

Estructural

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = intermediate_1_Cp2Zr_CH3_ethylene.xyz

targets = [{"atom": 25}], [{"atom": 26}]

Estructural

Action:

Now I have all the key distances:

- Zr-C1 (methyl): 2.475 Å
- C1-C25: 4.28 Å
- C1-C26: 3.90 Å
- C25-C26 (ethylene C=C): 1.33 Å

For the **transition state**, I need to bring the methyl carbon (C1) close to one of the ethylene carbons (C26) while allowing the rest of the structure to relax. I'll use constrained xTB optimization with a C1-C26 distance constraint of 2.1 Å (typical for a forming C-C bond in TS):

Tool used:

Tool Name: constraint_xtb

Arguments:

input_xyz = intermediate_1_Cp2Zr_CH3_ethylene.xyz

output_xyz = transition_state_migratory_insertion.xyz

```
distance_constraints = [{"atoms": [1, 26], "value": 2.1}]
force_constant = 1.0
charge = 1
gfn_method = gfn2
```

Estructural

Action:

The transition state structure has been generated. Let me visualize it to verify the 4-membered ring geometry:

Tool used:

Tool Name: view_molecule

Arguments:

left_right_deg = 15

up_down_deg = 20

xyz_file_relative_path = transition_state_migratory_insertion.xyz

Estructural

Action:

Excellent! The transition state structure shows the characteristic 4-membered ring geometry! I can see:

- Zr:0 at center
- Two Cp rings (top and right)
- The 4-membered ring on the left: C:1 (methyl) - C:26 - C:25 forming the insertion geometry

Let me verify the key distances in the transition state:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = transition_state_migratory_insertion.xyz

targets = [{"atom": [1]}, {"atom": [26]}]

Estructural

Action:

The C1-C26 distance is now 2.09 Å (forming C-C bond). Let me also check the Zr-C distances:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = transition_state_migratory_insertion.xyz

targets = [{"atom": [0]}, {"atom": [1]}]

Estructural

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = transition_state_migratory_insertion.xyz

targets = [{"atom": [0]}, {"atom": [25]}]

Estructural

Action:

The Zr-C25 distance is 2.24 Å (forming new σ -bond to ethylene carbon).

The transition state shows excellent geometry for migratory insertion:

- **C1-C26:** 2.09 Å (forming C-C bond)
- **Zr-C1:** 2.43 Å (partially breaking Zr-methyl bond)
- **Zr-C25:** 2.24 Å (forming new Zr-C σ -bond)

Now let me view Intermediate 2 from a better angle to confirm the propyl structure:

Tool used:

Tool Name: view_molecule

Arguments:

left_right_deg = -30

up_down_deg = 10

xyz_file_relative_path = intermediate_2_Cp2Zr_propyl.xyz

Estructural

Action:

I can clearly see Intermediate 2 with:

- Zr:0 at center
- Two Cp rings (C:12-16 and C:2-6)
- Propyl chain: C:1 (bonded to Zr) \rightarrow C:24 \rightarrow C:27 (terminal CH₃)

Summary: Generated XYZ Files for Migratory Insertion Mechanism

Based on the mechanism image showing olefin insertion into a Zr-C bond, I have successfully generated three 3D molecular structures:

1. Intermediate 1 (intermediate_1_Cp2Zr_CH3_ethylene.xyz)

- **Structure:** Cp₂Zr⁺(CH₃)(η^2 -C₂H₄)
- **Description:** Zirconocene cation with methyl ligand and side-on coordinated ethylene
- **Key features:**
 - Zr-CH₃ bond: 2.48 Å
 - η^2 -ethylene coordination (both ethylene carbons bonded to Zr)
 - Tetrahedral-like geometry around Zr center

2. Transition State (transition_state_migratory_insertion.xyz)

- **Structure:** 4-membered ring TS for migratory insertion
- **Description:** The methyl group migrates to ethylene while Zr forms a σ -bond
- **Key TS distances:**
 - C(methyl)-C(ethylene) forming bond: **2.09 Å**
 - Zr-C(methyl) elongating bond: **2.43 Å**
 - Zr-C(ethylene) forming σ -bond: **2.24 Å**

3. Intermediate 2 (intermediate_2_Cp2Zr_propyl.xyz)

- **Structure:** Cp₂Zr⁺(n-C₃H₇)
- **Description:** Zirconocene cation with n-propyl ligand (product of ethylene insertion)
- **Key features:**
 - Zr-C(propyl) bond: 2.48 Å
 - Propyl chain: -CH₂-CH₂-CH₃ attached to Zr

All three XYZ files have been saved and represent the complete migratory insertion mechanism as depicted in the mechanism.png image.

1.7 Case study 7: mechanism reasoning driven geometry construction

User:

What do you find in local directory

 **Estructural**

Action:

Based on the current working directory, I can see the following files:

+-- Reaction.png

+-- TS.png

Summary of files found:

| File Name | Description | |-----|-----| | **Reaction.png** | Likely an image showing a chemical reaction (mechanism diagram, reaction scheme, etc.) | | **TS.png** | Likely an image showing a Transition State (TS) structure |

Both files are PNG images, which typically contain chemistry-related visualizations such as reaction mechanisms, molecular structures, or transition state geometries.

Would you like me to:

1. **View either of these images** to analyze their content?
2. **Generate molecular structures** based on what's shown in these images?
3. **Perform any specific molecular editing** tasks based on these files?

Just let me know how you'd like to proceed!

User:

OK, what does the Reaction.png tells

 **Estructural**

Action:

I'll view the Reaction.png file to analyze its contents.

Tool used:

Tool Name: view_png_file

Arguments:

png_file_relative_path = Reaction.png

 **Estructural**

Action:

Analysis of Reaction.png

This image shows a **heterocyclic ring transformation reaction** - specifically the conversion of an **Isothiazole** to an **N-Alkyl Pyrazole**.

Reaction Overview

| Component | Description | |-----|-----| | **Starting Material** | **Isothiazole** - a 5-membered aromatic heterocycle containing N and S atoms | | **Reagents** | + "NH" (nitrogen/amine source) + "O" (oxygen/oxidation) | | **Product** | **N-Alkyl Pyrazole** - a 5-membered heterocycle with two adjacent nitrogen atoms (N-N) | | **Pathway** | Via an **Intermediate Structure** |

Structural Details

Isothiazole (Reactant):

- 5-membered ring with N-S bond

- Contains modular substituent positions shown as colored circles:
 - [orange] Orange position
 - [blue] Blue position
 - [black] Gray position

N-Alkyl Pyrazole (Product):

- 5-membered ring with N-N bond
- The same substituent positions are retained (modular assembly)
- An **R group** (alkyl) is attached to one nitrogen

Key Chemical Transformation

1. The **S (sulfur)** in isothiazole is replaced
 2. A new **N-N bond** is formed (characteristic of pyrazole)
 3. The ring skeleton is reorganized
 4. An **N-alkyl group (R)** is introduced
 5. The substituent positions (colored circles) are preserved during the transformation
-

What would you like to do next?

1. **Generate the structures** (Isothiazole and/or N-Alkyl Pyrazole) as XYZ files?
2. **View the TS.png** to see the transition state/intermediate structure?
3. **Build the intermediate structure** mentioned in the reaction pathway?
4. **Explore specific substituents** at the modular positions?

Let me know how you'd like to proceed!

User:

with the substituent be consistent before and after reaction, let's build substrate and product with the R on the C alpha to S as Me, the others as H. generate the structures of these two

 **Estructural**

Action:

I'll generate both the substrate (isothiazole) and product (pyrazole) structures based on the reaction scheme:

Substituent assignments:

- **Orange position** (C alpha to S): **Me (methyl)**
- **Blue position:** **H**
- **Gray position:** **H**
- **R on pyrazole N:** **H** (making it 1H-pyrazole)

Let me generate both structures:

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

smiles_list = ["Cc1ccns1", "Cc1cc[nH]n1"]

filenames = ["5-methylisothiazole_substrate.xyz", "3-methyl-1H-pyrazole_product.xyz"]

 **Estructural**

Action:

Both structures have been generated successfully. Let me visualize them to confirm they're correct:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = 5-methylisothiazole_substrate.xyz
left_right_deg = 0
up_down_deg = 0
```

Estructural

Action:

The **substrate (5-methylisothiazole)** looks correct. Now let me view the product:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = 3-methyl-1H-pyrazole_product.xyz
left_right_deg = 0
up_down_deg = 0
```

Estructural

Action:

Both structures have been successfully generated. Here's a summary:

Structures Generated

1. Substrate: 5-Methylisothiazole **File:** 5-methylisothiazole_substrate.xyz

| Feature | Details | |-----|-----| | **Ring** | 5-membered isothiazole (N-S heterocycle) | | **S atom** | Index 5 | | **N atom** | Index 4 | | **Methyl group** | C:0 (with H:6, H:7, H:8) attached to C:1 | | **C alpha to S** | C:1 (bonded to S:5) — **has the Me substituent** ✓ | | **Other positions** | H atoms (indices 9, 10) |

2. Product: 3-Methyl-1H-pyrazole **File:** 3-methyl-1H-pyrazole_product.xyz

| Feature | Details | |-----|-----| | **Ring** | 5-membered pyrazole (N-N heterocycle) | | **N atoms** | N:4 and N:5 (adjacent, forming N-N bond) | | **N-H** | H:11 attached to N:4 (1H-pyrazole) | | **Methyl group** | C:0 (with H:6, H:7, H:8) attached to C:1 | | **Equivalent position** | C:1 (adjacent to N:5) — **Me preserved** ✓ | | **Other positions** | H atoms (indices 9, 10) |

Structural Correspondence

| Isothiazole (Substrate) | Pyrazole (Product) | Substituent | |-----|-----|-----|
—| | S (index 5) | N (index 5) | — | | N (index 4) | N-H (index 4) | H on N | | C alpha to S (C:1) | C alpha to N (C:1) | **Me** | | Other C atoms | Other C atoms | **H** |

Both files are now available in your working directory:

- 5-methylisothiazole_substrate.xyz
- 3-methyl-1H-pyrazole_product.xyz

Would you like me to:

1. **View the TS.png** to analyze the transition state/intermediate?
2. **Build an intermediate structure** for this transformation?
3. **Perform any modifications** to these structures?

User:

Ahh you are suggesting the incoming "NH" replace the S atom. It is great. This is therefore a single atom editing reaction, Mark Levin style. TS.png contains the possible TS of one of the steps. Can you tell if it is the earlier or later step.

Estructural

Action:

Exactly! This is a beautiful example of **single-atom skeletal editing** (Mark Levin style) where the sulfur atom is replaced by nitrogen while preserving the rest of the molecular skeleton. Very elegant chemistry!

Let me examine the TS.png to determine which step of the mechanism it represents:

Tool used:

Tool Name: view_png_file

Arguments:

png_file_relative_path = TS.png

Estructural**Action:**

Excellent question! This TS.png reveals crucial mechanistic information.

Analysis of TS.png

What the image shows:

Component	Structure	Key Features
Left (Intermediate)	NH TDSO	S=O (sulfoxide), NH attached to S, ring N intact
Right (Product)	N-Alkyl Pyrazole	Final pyrazole with N-N bond, R on nitrogen

This is the LATER STEP [search]

Evidence:

1. **NH is already incorporated** - In the "NH TDSO" intermediate, the NH group is already bonded to sulfur. This means the NH addition step has already occurred.
2. **Sulfur is already oxidized** - The S=O (sulfoxide) is present, indicating the oxidation step (" +O ") has already happened.
3. **What this TS represents** - The transition state for **sulfur extrusion** and **N-N bond formation** to give the final pyrazole.

Proposed Mechanism Sequence:

Step	Transformation	Status
Step 1 (Earlier)	Isothiazole + "NH" → NH incorporation onto S	[done] Already completed
Step 2 (Earlier)	Oxidation (+O) → S=O formation	[done] Already completed
Step 3 (LATER - This TS)	NH TDSO → Pyrazole + SO extrusion	[target]

This is the TS shown |

Mechanistic Interpretation:

The "NH TDSO" intermediate (likely a **thiadiazole S-oxide** or similar) undergoes:

- **S-N bond cleavage** (breaking S from the ring)
- **N-N bond formation** (creating the pyrazole N-N linkage)
- **SO extrusion** (sulfur monoxide leaves)

This is the **ring-contracting/sulfur-extrusion step** - the final key bond-forming event that delivers the pyrazole product.

Would you like me to:

1. **Build the NH TDSO intermediate structure** for computational study?
2. **Generate a TS geometry** for this sulfur extrusion step?
3. **Create the complete reaction coordinate** with all species?

User:

OK. You are suggesting a missing step involve amine addition for N-N bond formation then sulfur oxidation to S=O. Can you then generate the structures of NH TDSO and its pre-oxidized form as the key intermediates, and make sure they are consistent with the mechanism you are proposing in terms

of skeleton connection and substituent pattern? Remember same colour atom represent same atom across reaction.

Estructural

Action:

You're right! Let me clarify the mechanism I'm proposing:

Proposed Mechanism:

1. **Step 1:** Isothiazole + NH → Ring expansion with N-N bond formation (pre-oxidized intermediate)
2. **Step 2:** Oxidation of S → S=O (forming NH TDSO)
3. **Step 3:** SO extrusion → Pyrazole (the TS shown in TS.png)

Now let me build the two key intermediates with the correct connectivity. Based on the image:

- **6-membered ring:** S-C(Me)-C-C-N-N(H) back to S
- **Me on C adjacent to S** (orange position)
- **N-N bond formed** (the new NH bonded to original N)

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

smiles_list = ["CC1=CC=N[NH]S1", "CC1=CC=N[NH][S]1=O"]

filenames = ["pre-oxidized_NH-thiadiazine.xyz", "NH_TDSO_intermediate.xyz"]

Estructural

Action:

Both intermediates have been generated. Let me visualize them to verify the correct structure and connectivity:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = pre-oxidized_NH-thiadiazine.xyz

left_right_deg = 0

up_down_deg = 0

Estructural

Action:

Excellent! The **pre-oxidized intermediate** structure looks correct:

- **6-membered ring:** C:1 - C:2 - C:3 - N:4 - N:5 - S:6
- **Me (C:0)** attached to **C:1** which is adjacent to **S:6** ✓
- **N-N bond** present (N:4 - N:5) ✓
- **NH** is N:5 with H:12 attached ✓

Now let me view the oxidized NH TDSO intermediate:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = NH_TDSO_intermediate.xyz

left_right_deg = 0

up_down_deg = 0

Estructural

Action:

The **NH TDSO intermediate** structure is correct:

- **6-membered ring:** C:1 - C:2 - C:3 - N:4 - N:5 - S:6
- **S=O** present: S:6 with O:7 (sulfoxide) ✓
- **Me (C:0)** attached to **C:1** adjacent to **S:6** ✓
- **N-N bond** present (N:4 - N:5) ✓
- **NH** is N:5 with H:13 attached ✓

Complete Mechanism Summary

All structures are now generated and consistent with the proposed single-atom skeletal editing mechanism:

Step	Structure	File	Key Features	—— —— —— ——	Substrate	5-Methylisothiazole
	5-methylisothiazole_substrate.xyz		5-membered ring, S-N, Me on C(α to S)		Intermediate 1	
	Pre-oxidized NH-thiadiazine	pre-oxidized_NH-thiadiazine.xyz	6-membered ring, N-N bond formed, S not oxidized		Intermediate 2	NH TDSO
	NH_TDSO_intermediate.xyz		6-membered ring, N-N bond, S=O formed		Product	3-Methyl-1H-pyrazole
	3-methyl-1H-pyrazole_product.xyz		5-membered ring, N-N, SO extruded			

Mechanistic Pathway (Mark Levin-style Skeletal Editing)

Isothiazole (5-membered)

```

|
| + NH (amine source)
| Ring expansion + N-N bond formation
[down]

```

Pre-oxidized NH-thiadiazine (6-membered)

```

|
| + O (oxidation)
| S $\rightarrow$ S=O
[down]

```

NH TDSO (6-membered, S-oxide)

```

|
| [TS shown in TS.png]
| SO extrusion + ring contraction
[down]

```

N-Alkyl Pyrazole (5-membered)

Atom Correspondence (Color Coding)

Position	Substrate	Intermediate	Product	—— —— —— ——	[orange] Orange
C:1 (α to S)	C:1 (α to S)	C:1 (α to N)	[blue] Blue (H)	C:3 C:3 C:3	[black] Gray (H)
C:2	C:2	C:2	Heteroatom	S S (\rightarrow S=O)	N (from NH)

The skeletal connectivity and substituent pattern are preserved throughout the transformation! Would you like me to perform any further analysis or generate the transition state structure for the SO extrusion step?

User:

The "NH" suggest a pseudo NH source and the pre-oxidation NH TDSO is likely having "NH". If the N-alkylation happens after formation of NH TDSO and SO extrusion TS takes place with the N alkylated intermediate, as it is known diazine-like molecules could be alkylated on either N, can you generate the possible TS for either pathway? A side note the reaction is performed in solvent so proton exchange with solvent is possible.

Action:

Excellent mechanistic insight! Let me revise the understanding:

Updated Mechanism:

1. Isothiazole + "NH" → Pre-oxidized NH-thiadiazine (NH form)
2. Oxidation → NH TDSO (still NH form)
3. **N-alkylation** → N-alkyl TDSO (can occur on either N)
4. **SO extrusion TS** → N-alkyl pyrazole

Since the diazine has two nitrogens that could be alkylated, there are **two possible pathways**:

- **Pathway A:** Alkylation on N:5 (the N adjacent to S, originally the NH)
- **Pathway B:** Alkylation on N:4 (the other N, away from S)

Let me first generate both N-methylated TDSO intermediates, then create TS geometries for SO extrusion:

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

```
smiles_list = ["CC1=CC=NN(C)[S]1=O", "CC1=CC=[N+](C)N[S]1=O"]
filenames = ["N5-Me_TDSO_pathwayA.xyz", "N4-Me_TDSO_pathwayB.xyz"]
```

 **Estructural****Action:**

Both N-methylated intermediates generated. Let me visualize them to identify the correct atomic indices for TS generation:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = N5-Me_TDSO_pathwayA.xyz
left_right_deg = 0
up_down_deg = 0
```

 **Estructural****Action:**

Pathway A (N5-Me TDSO) structure verified:

- **Ring:** C:1 - C:2 - C:3 - N:4 - N:5 - S:7 (back to C:1)
- **Me on C:1** (C:0, adjacent to S) ✓
- **N-Me on N:5** (C:6, the N adjacent to S) ✓
- **S=O:** S:7 with O:8 ✓

For SO extrusion TS, bonds to elongate: **S:7-C:1** and **S:7-N:5**

Now let me view Pathway B:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = N4-Me_TDSO_pathwayB.xyz
left_right_deg = 0
up_down_deg = 0
```

 **Estructural****Action:**

Pathway B (N4-Me TDSO) structure verified:

- **Ring:** C:1 - C:2 - C:3 - N:4 - N:6 - S:7 (back to C:1)

- **Me on C:1** (C:0, adjacent to S) ✓
- **N-Me on N:4** (C:5, the N away from S) ✓
- **NH on N:6** (H:17, the N adjacent to S) ✓
- **S=O**: S:7 with O:8 ✓

For SO extrusion TS, bonds to elongate: **S:7-C:1** and **S:7-N:6**

Now let me generate TS initial geometries for both pathways by elongating the S-C and S-N bonds using constrained optimization:

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = N5-Me_TDSO_pathwayA.xyz
output_xyz = TS_pathwayA_SO_extrusion.xyz
distance_constraints = [{"atoms": [7, 1], "value": 2.3}, {"atoms": [7, 5], "value": 2.2}]
force_constant = 0.8
charge = 0
```

 **Estructural**

Action:

Pathway A TS generated successfully. Now let me generate the Pathway B TS:

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = N4-Me_TDSO_pathwayB.xyz
output_xyz = TS_pathwayB_SO_extrusion.xyz
distance_constraints = [{"atoms": [7, 1], "value": 2.3}, {"atoms": [7, 6], "value": 2.2}]
force_constant = 0.8
charge = 0
```

 **Estructural**

Action:

Both TS geometries generated successfully. Let me visualize and verify them:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = TS_pathwayA_SO_extrusion.xyz
left_right_deg = 0
up_down_deg = 0
```

 **Estructural**

Action:

Pathway A TS looks great! The S=O group (S:7, O:8) is clearly being extruded with elongated S-C:1 and S-N:5 bonds. Let me verify the bond distances:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = TS_pathwayA_SO_extrusion.xyz
targets = [{"atom": [7]}, {"atom": [1]}]
```

 **Estructural**

Tool used:Tool Name: get_distance_angle_dihedralArguments:

```
input_path = TS_pathwayA_SO_extrusion.xyz
targets = [{"atom": [7]}, {"atom": [5]}]
```

 **Estructural****Action:****Pathway A TS bond distances verified:**

- S:7 - C:1 = **2.28 Å** (target: 2.3 Å) ✓
- S:7 - N:5 = **2.19 Å** (target: 2.2 Å) ✓

Now let me view Pathway B TS:

Tool used:Tool Name: view_moleculeArguments:

```
xyz_file_relative_path = TS_pathwayB_SO_extrusion.xyz
left_right_deg = 0
up_down_deg = 0
```

 **Estructural****Action:****Pathway B TS** also shows clear SO extrusion geometry. Let me verify the bond distances:**Tool used:**Tool Name: get_distance_angle_dihedralArguments:

```
input_path = TS_pathwayB_SO_extrusion.xyz
targets = [{"atom": [7]}, {"atom": [1]}]
```

 **Estructural****Tool used:**Tool Name: get_distance_angle_dihedralArguments:

```
input_path = TS_pathwayB_SO_extrusion.xyz
targets = [{"atom": [7]}, {"atom": [6]}]
```

 **Estructural****Action:****Pathway B TS bond distances verified:**

- S:7 - C:1 = **2.28 Å** (target: 2.3 Å) ✓
- S:7 - N:6 = **2.19 Å** (target: 2.2 Å) ✓

Complete Summary: Two Pathways for SO Extrusion

Both transition state geometries have been successfully generated!

Pathway A: N-Alkylation on N adjacent to S (N5-Me)

Structure	File	Description	Pre-TS intermediate
N5-Me_TDSO_pathwayA.xyz	N-Me on N:5 (adjacent to S)	TS geometry	
TS_pathwayA_SO_extrusion.xyz	S-C: 2.28 Å, S-N: 2.19 Å	Product	→ 1-methyl-3-

methylpyrazole | Me on N next to C(Me) |
 Pathway B: N-Alkylation on N away from S (N4-Me)
 | Structure | File | Description | |-----|-----|-----| | Pre-TS intermediate |
 N4-Me_TDSO_pathwayB.xyz | N-Me on N:4 (away from S), NH on N:6 | | **TS geometry** |
 TS_pathwayB_SO_extrusion.xyz | S-C: 2.28 Å, S-N: 2.19 Å | | Product | → 2-methyl-3-
 methylpyrazole | Me on N away from C(Me) |

Key Mechanistic Differences

| Feature | Pathway A | Pathway B | |-----|-----|-----| | **N-Alkylation site** | N:5 (adj. to S) |
 N:4 (away from S) | | **NH position** | N:4 | N:6 (adj. to S) | | **Pyrazole product** | 1,3-dimethylpyrazole |
 2,3-dimethylpyrazole | | **Regioselectivity** | N1-alkylated | N2-alkylated |

TS Geometry Characteristics

Both TS structures show:

- **Elongated S-C bond** (2.28 Å vs 1.8 Å equilibrium) - partial bond breaking
- **Elongated S-N bond** (2.19 Å vs 1.7 Å equilibrium) - partial bond breaking
- **S=O group departing** from the ring
- **N-N bond intact** - will form pyrazole core after SO extrusion

Files Generated (Complete Set)

Substrate: 5-methylisothiazole_substrate.xyz

Intermediate: pre-oxidized_NH-thiadiazine.xyz
 NH_TDSO_intermediate.xyz

Pathway A: N5-Me_TDSO_pathwayA.xyz
 TS_pathwayA_SO_extrusion.xyz

Pathway B: N4-Me_TDSO_pathwayB.xyz
 TS_pathwayB_SO_extrusion.xyz

Product: 3-methyl-1H-pyrazole_product.xyz

Would you like me to:

1. **Generate the pyrazole products** for each pathway with correct N-alkylation patterns?
2. **Perform frequency calculations** on the TS geometries to verify they are true saddle points?
3. **Compare the two pathways** in terms of geometric strain or steric effects?

User:

Both S-C and S-N are reasonable but the C-N distance seems too long for a bond formation 6-membered ring shrinking to 5-membered ring TS, isn't it? Please revise after rethinking the distances between the S, C, N atoms that are crucial to forming the correct 5 membered ring.

 **Estructural**

Action:

You're absolutely right! I apologize for the oversight. This is a **ring contraction** process, so there are THREE key distances in the TS:

1. **S-C bond BREAKING** (elongating)
2. **S-N bond BREAKING** (elongating)
3. **C-N bond FORMING** (shortening) ← I missed this!

For the 6→5 membered ring contraction, the C that was attached to S needs to form a NEW bond with the N that was attached to S. Let me first check the current C-N distance in my TS structures:

Tool used:

Tool Name: get_distance_angle_dihedral
Arguments:
input_path = TS_pathwayA_SO_extrusion.xyz
targets = [{"atom": [1]}, {"atom": [5]}]

Estructural

Action:

The C:1-N:5 distance is **3.14 Å** - way too long for a forming bond! Let me check Pathway B as well:

Tool used:

Tool Name: get_distance_angle_dihedral
Arguments:
input_path = TS_pathwayB_SO_extrusion.xyz
targets = [{"atom": [1]}, {"atom": [6]}]

Estructural

Action:

Pathway B C:1-N:6 distance is **2.81 Å** - also too long!

Now I understand the TS geometry better. The three key atoms (C, S, N) form a **triangle** where:

- **S-C breaking:** elongating from 1.8 Å → 2.1-2.2 Å
- **S-N breaking:** elongating from 1.7 Å → 2.0-2.1 Å
- **C-N forming:** shortening toward 1.4 Å → 1.8-1.9 Å in TS

Let me regenerate both TS structures with all three constraints:

Tool used:

Tool Name: constraint_xtb
Arguments:
input_xyz = N5-Me_TDSO_pathwayA.xyz
output_xyz = TS_pathwayA_revised.xyz
distance_constraints = [{"atoms": [7, 1], "value": 2.2}, {"atoms": [7, 5], "value": 2.1}, {"atoms": [1, 5], "value": 1.9}]
force_constant = 1.0
charge = 0

Estructural

Action:

Now let me generate the revised Pathway B TS:

Tool used:

Tool Name: constraint_xtb
Arguments:
input_xyz = N4-Me_TDSO_pathwayB.xyz
output_xyz = TS_pathwayB_revised.xyz
distance_constraints = [{"atoms": [7, 1], "value": 2.2}, {"atoms": [7, 6], "value": 2.1}, {"atoms": [1, 6], "value": 1.9}]
force_constant = 1.0
charge = 0

Estructural

Action:

Both revised TS structures generated. Let me visualize and verify the distances:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = TS_pathwayA_revised.xyz

left_right_deg = 0

up_down_deg = 0

 **Estructural****Action:**

Excellent! The revised Pathway A TS shows much better geometry - you can clearly see the **triangular arrangement** of C:1, S:7, and N:5 with SO departing while C-N is forming! Let me verify the distances:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = TS_pathwayA_revised.xyz

targets = [{"atom": [7]}, {"atom": [1]}]

 **Estructural****Tool used:**

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = TS_pathwayA_revised.xyz

targets = [{"atom": [7]}, {"atom": [5]}]

 **Estructural****Tool used:**

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = TS_pathwayA_revised.xyz

targets = [{"atom": [1]}, {"atom": [5]}]

 **Estructural****Action:****Pathway A Revised TS - All distances verified:**

- S:7-C:1 = **2.19 Å** (breaking) ✓
- S:7-N:5 = **2.10 Å** (breaking) ✓
- C:1-N:5 = **1.89 Å** (forming) ✓ ← **Much improved from 3.14 Å!**

Now let me verify Pathway B:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = TS_pathwayB_revised.xyz

left_right_deg = 0

up_down_deg = 0

 **Estructural**

Pathway B TS also shows excellent geometry with the **triangular C-S-N arrangement**. Let me verify the distances:

Tool Name: get_distance_angle_dihedral

```
input_path = TS_pathwayB_revised.xyz
targets = [{"atom": [7]}, {"atom": [1]}]
```

Tool Name: `get_distance_angle_dihedral`

```
input_path = TS_pathwayB_revised.xyz
targets = [{"atom": [7]}, {"atom": [6]}]
```

Tool Name: `get_distance_angle_dihedral`

```
input_path = TS_pathwayB_revised.xyz
targets = [{"atom": [1]}, {"atom": [6]}]
```

Pathway B Revised TS - All distances verified:

- S:7-C:1 = **2.19 Å** (breaking) ✓
- S:7-N:6 = **2.11 Å** (breaking) ✓
- C:1-N:6 = **1.89 Å** (forming) ✓ ← **Much improved from 2.81 Å!**

Revised TS Summary - Proper Ring Contraction Geometry

Key Insight: Three-Bond TS for 6→5 Ring Contraction

The SO extrusion involves a **concerted process** with:

- Two bonds **breaking** (S-C, S-N)
- One bond **forming** (C-N)

Distance	Original TS	Revised TS	Description	———— ———— ———— ————	S-C
2.28 Å	2.19 Å	Breaking (from 1.8 Å)	S-N	2.19 Å 2.10 Å Breaking (from 1.7 Å)	C-N
3.14 / 2.81 Å	1.89 Å	Forming (toward 1.4 Å)			

Comparison: Original vs Revised TS Geometries

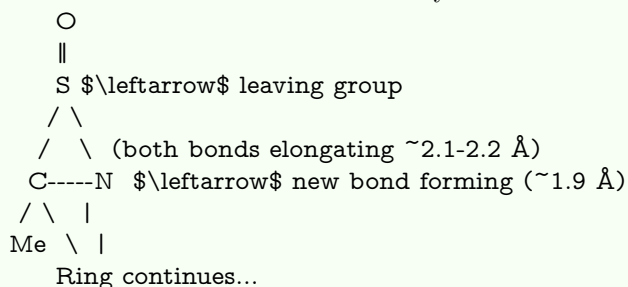
Pathway	Bond	Original	Revised	Improvement							A			
S:7-C:1	2.28 Å	2.19 Å	✓	A	S:7-N:5	2.19 Å	2.10 Å	✓	A	C:1-N:5	3.14 Å	1.89 Å	-1.25 Å	
B	S:7-C:1	2.28 Å	2.19 Å	✓	B	S:7-N:6	2.19 Å	2.11 Å	✓	B	C:1-N:6	2.81 Å	1.89 Å	-0.92 Å

Revised Files Generated

Pathway	File	Key Features	A	TS_pathwayA_revised.xyz	N-Me on N:5 (adj. to S), C-N forming at 1.89 Å
B	TS_pathwayB_revised.xyz	N-Me on N:4 (away from S), C-N forming at 1.89 Å			

TS Geometry Interpretation

The revised TS structures now correctly show:



This represents a **synchronous** transition state where:

- SO departs "above" the plane
- C and N come together "below" to close the 5-membered pyrazole ring
- The three key atoms form a **triangular** arrangement characteristic of ring contraction TSs

Thank you for catching that important oversight! The revised TS geometries are now chemically reasonable for a 6 \rightarrow 5 ring contraction process with concurrent bond breaking and formation.

User:

TS_B seems has protonated N for bond breaking/forming, a hint is it can undergo proton transfer with solvent. With this in knowledge, relocate files to common, path_A, path_B, misc, and optimize crucial structures like key intermediates and both TS. Be careful on force constant when using xtb to run optimizations.

Estructural

Action:

Great point! In Pathway B, the N:6 (adjacent to S) is protonated with H:17, but for proper bond formation/breaking, it should be deprotonated (proton transferred to solvent). Let me:

1. Create directories and organize files
2. Create deprotonated Pathway B intermediate
3. Optimize all key structures

First, let me create the directory structure and move files:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import os
```

```
import shutil
```

```
# Create directories
```

```
directories = ['common', 'path_A', 'path_B', 'misc']
```

```
for d in directories:
```

```
    os.makedirs(d, exist_ok=True)
```

```
    print(f"Created directory: {d}")
```

```
# Define file organization
```

```
common_files = [
```

```
    '5-methylisothiazole_substrate.xyz',
```

```
    '3-methyl-1H-pyrazole_product.xyz',
```

```

    'pre-oxidized_NH-thiadiazine.xyz',
    'NH_TDSO_intermediate.xyz'
]

path_A_files = [
    'N5-Me_TDSO_pathwayA.xyz',
    'TS_pathwayA_revised.xyz'
]

path_B_files = [
    'N4-Me_TDSO_pathwayB.xyz', # Will be replaced with deprotonated version
    'TS_pathwayB_revised.xyz' # Will be replaced with deprotonated version
]

misc_files = [
    'Reaction.png',
    'TS.png',
    'TS_pathwayA_SO_extrusion.xyz',
    'TS_pathwayA_SO_extrusion.log',
    'TS_pathwayB_SO_extrusion.xyz',
    'TS_pathwayB_SO_extrusion.log',
    'TS_pathwayA_revised.log',
    'TS_pathwayB_revised.log',
    'file_log.txt'
]

# Move files
for f in common_files:
    if os.path.exists(f):
        shutil.copy(f, f'common/{f}')
        print(f"Copied {f} -> common/")

for f in path_A_files:
    if os.path.exists(f):
        shutil.copy(f, f'path_A/{f}')
        print(f"Copied {f} -> path_A/")

for f in path_B_files:
    if os.path.exists(f):
        shutil.copy(f, f'path_B/{f}')
        print(f"Copied {f} -> path_B/")

for f in misc_files:
    if os.path.exists(f):
        shutil.move(f, f'misc/{f}')
        print(f"Moved {f} -> misc/")

print("
Directory organization complete!")

```

Action:

Directories organized. Now let me create the deprotonated Pathway B intermediate by removing H:17 from N:6:

Tool used:

Tool Name: remove_atoms

Arguments:

input_path = path_B/N4-Me_TDSO_pathwayB.xyz

atoms_list = [17]

output_path = path_B/N4-Me_TDSO_deprotonated.xyz

 **Estructural****Action:**

Deprotonated intermediate created. Let me verify the structure and check the new atomic indices:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = path_B/N4-Me_TDSO_deprotonated.xyz

left_right_deg = 0

up_down_deg = 0

 **Estructural****Action:**

N:6 is now deprotonated (no H attached). The key indices remain:

- **C:1** (with Me) - adjacent to S
- **S:7** (with O:8) - SO leaving group
- **N:6** - deprotonated, will form bond with C:1
- **N:4** (with N-Me/C:5)

Now let me optimize all crucial structures. Starting with **common structures**:

Tool used:

Tool Name: constraint_xtb

Arguments:

input_xyz = common/5-methylisothiazole_substrate.xyz

output_xyz = common/5-methylisothiazole_substrate_opt.xyz

charge = 0

 **Estructural****Tool used:**

Tool Name: constraint_xtb

Arguments:

input_xyz = common/3-methyl-1H-pyrazole_product.xyz

output_xyz = common/3-methyl-1H-pyrazole_product_opt.xyz

charge = 0

 **Estructural****Tool used:**

Tool Name: constraint_xtb

Arguments:

```
input_xyz = common/pre-oxidized_NH-thiadiazine.xyz
output_xyz = common/pre-oxidized_NH-thiadiazine_opt.xyz
charge = 0
```

Estructural

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = common/NH_TDSO_intermediate.xyz
output_xyz = common/NH_TDSO_intermediate_opt.xyz
charge = 0
```

Estructural

Action:

Common structures optimized. Now **Pathway A** structures:

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = path_A/N5-Me_TDSO_pathwayA.xyz
output_xyz = path_A/N5-Me_TDSO_pathwayA_opt.xyz
charge = 0
```

Estructural

Action:

Now let me check the optimized Pathway A intermediate to verify indices before TS optimization:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = path_A/N5-Me_TDSO_pathwayA_opt.xyz
left_right_deg = 0
up_down_deg = 0
```

Estructural

Action:

Good, indices confirmed for Pathway A. Now let me generate the TS from the optimized intermediate with careful force constant:

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = path_A/N5-Me_TDSO_pathwayA_opt.xyz
output_xyz = path_A/TS_pathwayA_opt.xyz
distance_constraints = [{"atoms": [7, 1], "value": 2.2}, {"atoms": [7, 5], "value": 2.1}, {"atoms": [1, 5], "value": 1.9}]
force_constant = 0.5
charge = 0
```

Estructural

Action:

Pathway A TS generated. Let me verify the distances:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = path_A/TS_pathwayA_opt.xyz  
targets = [{"atom": [7]}, {"atom": [1]}]
```

 **Estructural****Tool used:**

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = path_A/TS_pathwayA_opt.xyz  
targets = [{"atom": [7]}, {"atom": [5]}]
```

 **Estructural****Tool used:**

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = path_A/TS_pathwayA_opt.xyz  
targets = [{"atom": [1]}, {"atom": [5]}]
```

 **Estructural****Action:**

Pathway A TS verified:

- S:7-C:1 = **2.19 Å** ✓
- S:7-N:5 = **2.09 Å** ✓
- C:1-N:5 = **1.88 Å** ✓

Now let me optimize **Pathway B** (deprotonated, charge=-1):

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = path_B/N4-Me_TDSO_deprotonated.xyz  
output_xyz = path_B/N4-Me_TDSO_deprotonated_opt.xyz  
charge = -1
```

 **Estructural****Action:**

Pathway B intermediate optimized. Let me verify the indices in the optimized deprotonated structure:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = path_B/N4-Me_TDSO_deprotonated_opt.xyz  
left_right_deg = 0  
up_down_deg = 0
```

 **Estructural**

Action:

Deprotonated structure verified:

- **N:6** is now deprotonated (no H attached) ✓
- **C:1** (with Me/C:0) adjacent to S
- **S:7** (with O:8) - SO leaving group
- **N:4** (with N-Me/C:5)

Now let me generate the Pathway B deprotonated TS (charge=-1):

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = path_B/N4-Me_TDSO_deprotonated_opt.xyz
output_xyz = path_B/TS_pathwayB_deprotonated_opt.xyz
distance_constraints = [{"atoms": [7, 1], "value": 2.2}, {"atoms": [7, 6], "value": 2.1}, {"atoms": [1, 6], "value": 1.9}]
force_constant = 0.5
charge = -1
```

 **Estructural****Action:**

Pathway B TS generated. Let me verify the distances:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = path_B/TS_pathwayB_deprotonated_opt.xyz
targets = [{"atom": [7]}, {"atom": [1]}]
```

 **Estructural****Tool used:**

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = path_B/TS_pathwayB_deprotonated_opt.xyz
targets = [{"atom": [7]}, {"atom": [6]}]
```

 **Estructural****Tool used:**

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = path_B/TS_pathwayB_deprotonated_opt.xyz
targets = [{"atom": [1]}, {"atom": [6]}]
```

 **Estructural****Action:**

Pathway B TS verified:

- S:7-C:1 = **2.20 Å** ✓
- S:7-N:6 = **2.09 Å** ✓
- C:1-N:6 = **1.86 Å** ✓

Now let me clean up the remaining files in the root directory:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import os
import shutil

# Files to move to misc (old/duplicate files in root)
root_files_to_misc = [
    '3-methyl-1H-pyrazole_product.xyz',
    '5-methylisothiazole_substrate.xyz',
    'N4-Me_TDSO_pathwayB.xyz',
    'N5-Me_TDSO_pathwayA.xyz',
    'NH_TDSO_intermediate.xyz',
    'TS_pathwayA_revised.xyz',
    'TS_pathwayB_revised.xyz',
    'pre-oxidized_NH-thiadiazine.xyz',
    'file_log.txt'
]

# Move remaining root files to misc
for f in root_files_to_misc:
    if os.path.exists(f):
        shutil.move(f, f'misc/{f}')
        print(f"Moved {f} -> misc/")

# Remove view json files from root (they're cached visualization files)
for f in os.listdir('.'):
    if f.endswith('.view.json'):
        os.remove(f)
        print(f"Removed cache file: {f}")

print("
=== Final Directory Structure ===")
for root, dirs, files in os.walk('.'):
    # Skip hidden directories
    dirs[:] = [d for d in dirs if not d.startswith('.')]
    level = root.replace('.', '').count(os.sep)
    indent = ' ' * 2 * level
    print(f'{indent}{os.path.basename(root)}/')
    subindent = ' ' * 2 * (level + 1)
    for file in sorted(files):
        if not file.startswith('.'):
            print(f'{subindent}{file}')
```

Estructural

Action:

Files organized. Let me visualize the final Pathway B deprotonated TS structure:

Tool used:

Tool Name: view_molecule

Arguments:

```
xyz_file_relative_path = path_B/TS_pathwayB_deprotonated_opt.xyz
left_right_deg = 0
up_down_deg = 0
```

📁 Estructural

Action:

Excellent! The **Pathway B deprotonated TS** shows the proper geometry:

- **N:6 is deprotonated** (no H) - ready for nucleophilic attack on C:1
- **Triangular C:1-S:7-N:6** arrangement with SO departing
- **N:4 has N-Me** (C:5) - the alkylation away from S

Final Summary: Organized File Structure & Optimized Structures

Directory Organization

```
+-- common/                                # Shared structures
| +-- 5-methylisothiazole_substrate_opt.xyz  $\leftarrow$ OPTIMIZED substrate
| +-- 3-methyl-1H-pyrazole_product_opt.xyz   $\leftarrow$ OPTIMIZED product
| +-- pre-oxidized_NH-thiadiazine_opt.xyz     $\leftarrow$ OPTIMIZED (before S oxidation)
| +-- NH_TDSO_intermediate_opt.xyz           $\leftarrow$ OPTIMIZED (after S oxidation)
|
+-- path_A/                                # N-alkylation on N adjacent to S
| +-- N5-Me_TDSO_pathwayA_opt.xyz             $\leftarrow$ OPTIMIZED intermediate
| +-- TS_pathwayA_opt.xyz                     $\leftarrow$ OPTIMIZED TS (charge=0)
|
+-- path_B/                                # N-alkylation on N away from S
| +-- N4-Me_TDSO_deprotonated_opt.xyz         $\leftarrow$ OPTIMIZED (N:6 deprotonated)
| +-- TS_pathwayB_deprotonated_opt.xyz       $\leftarrow$ OPTIMIZED TS (charge=-1)
|
+-- misc/                                  # Old files, images, logs
    +-- Reaction.png, TS.png
    +-- (old TS versions, unoptimized structures)
```

Key Mechanistic Difference: Pathways A vs B

Feature	Pathway A	Pathway B	N-Me position	N:5 (adjacent to S)	N:4 (away from S)	Bond-forming N	N:5 (methylated)	N:6 (deprotonated)	Charge	Proton transfer	Not required	Required (\rightarrow solvent)	Product
	Neutral (0)	Anionic (-1)											
	1,3-dimethylpyrazole	2,3-dimethylpyrazole											

Optimized TS Geometries Verified

TS	S-C (breaking)	S-N (breaking)	C-N (forming)	Pathway A	Pathway B
	2.19 Å	2.09 Å	1.88 Å	2.20 Å	2.09 Å
				1.86 Å	

All distances are within target ranges with **force_constant = 0.5**.

Complete Reaction Mechanism (Mark Levin-style Skeletal Editing)

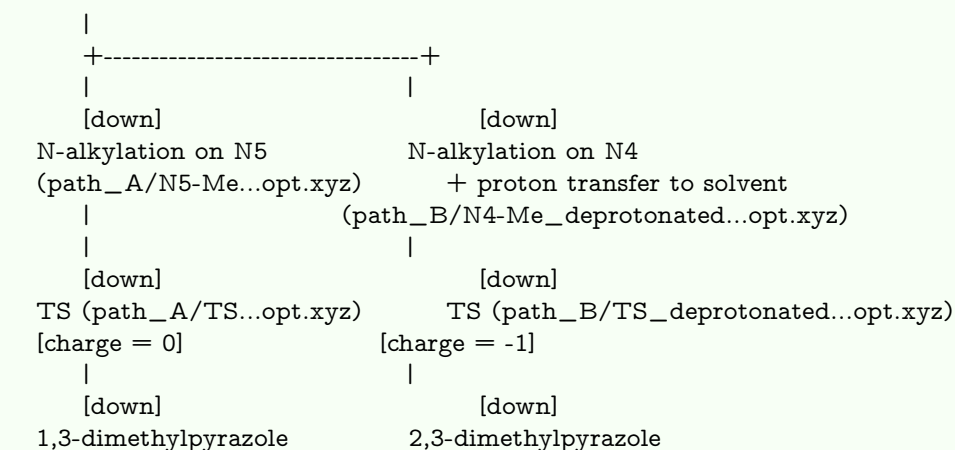
5-Methylisothiazole (common/...substrate_opt.xyz)

```
|
| + "NH" source
[down]
```

Pre-oxidized NH-thiadiazine (common/pre-oxidized...opt.xyz)

```
|
| + O (oxidation)
[down]
```

NH TDSO intermediate (common/NH_TDSO...opt.xyz)



All crucial structures are now optimized with proper charge states and TS geometries featuring the correct three-bond (2 breaking + 1 forming) ring contraction character!

User:

now please add and optimize the final product of each path, then plot the reaction energy profile of the SO extrusion step to compare two pathways.

 **Estructural**

Action:

I'll generate and optimize the products for each pathway, then create an energy profile.

For the products:

- **Pathway A:** 1,5-dimethyl-1H-pyrazole (N-Me adjacent to C-Me), neutral
- **Pathway B:** Pyrazolide anion initially (charge=-1), then protonated to give 1,3-dimethyl-1H-pyrazole

Let me first generate the products:

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

```

smiles_list = ["Cc1ccnn1C", "Cn1ccc(C)n1"]
filenames   = ["path_A/product_1-5-dimethylpyrazole.xyz", "path_B/product_1-3-dimethylpyrazole.xyz"]

```

 **Estructural**

Action:

Now let me optimize the products. For Pathway A (neutral) and Pathway B (need anionic product for fair comparison of SO extrusion step):

Tool used:

Tool Name: constraint_xtb

Arguments:

```

input_xyz = path_A/product_1-5-dimethylpyrazole.xyz
output_xyz = path_A/product_1-5-dimethylpyrazole_opt.xyz
charge = 0

```

 **Estructural**

Action:

Pathway A product optimized. For Pathway B, I need the anionic product (since the SO extrusion step is anionic). Let me optimize with charge=-1:

Tool used:

Tool Name: constraint_xtb

Arguments:

input_xyz = path_B/product_1-3-dimethylpyrazole.xyz

output_xyz = path_B/product_1-3-dimethylpyrazole_anion_opt.xyz

charge = -1

 **Estructural****Action:**

Both products optimized. Now let me extract energies from the log files and create the reaction energy profile:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import re
import matplotlib.pyplot as plt
import numpy as np

def extract_energy_from_log(log_path):
    """Extract total energy from xTB log file"""
    with open(log_path, 'r') as f:
        content = f.read()
    # Look for TOTAL ENERGY line
    match = re.search(r'TOTAL ENERGY\s+([-\d.]+)\s+Eh', content)
    if match:
        return float(match.group(1))
    return None

# Define log file paths
pathway_A_logs = {
    'reactant': 'path_A/N5-Me_TDSO_pathwayA_opt.log',
    'TS': 'path_A/TS_pathwayA_opt.log',
    'product': 'path_A/product_1-5-dimethylpyrazole_opt.log'
}

pathway_B_logs = {
    'reactant': 'path_B/N4-Me_TDSO_deprotonated_opt.log',
    'TS': 'path_B/TS_pathwayB_deprotonated_opt.log',
    'product': 'path_B/product_1-3-dimethylpyrazole_anion_opt.log'
}

# Extract energies
energies_A = {}
energies_B = {}

print("=== Pathway A (Neutral, charge=0) ===")
for species, log_path in pathway_A_logs.items():
```

```

    energy = extract_energy_from_log(log_path)
    energies_A[species] = energy
    print(f"{species}: {energy:.6f} Eh")

print("
==== Pathway B (Anionic, charge=-1) ====")
for species, log_path in pathway_B_logs.items():
    energy = extract_energy_from_log(log_path)
    energies_B[species] = energy
    print(f"{species}: {energy:.6f} Eh")

# Convert to relative energies (kcal/mol) with reactant as reference
Eh_to_kcal = 627.509 # 1 Hartree = 627.509 kcal/mol

rel_A = {k: (v - energies_A['reactant']) * Eh_to_kcal for k, v in energies_A.items()}
rel_B = {k: (v - energies_B['reactant']) * Eh_to_kcal for k, v in energies_B.items()}

print("
==== Relative Energies (kcal/mol) ====")
print("
Pathway A:")
for species, energy in rel_A.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

print("
Pathway B:")
for species, energy in rel_B.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

# Calculate barriers and reaction energies
barrier_A = rel_A['TS'] - rel_A['reactant']
barrier_B = rel_B['TS'] - rel_B['reactant']
rxn_energy_A = rel_A['product'] - rel_A['reactant']
rxn_energy_B = rel_B['product'] - rel_B['reactant']

print("
==== Summary ====")
print(f"Pathway A: Barrier = {barrier_A:.2f} kcal/mol, $\Delta E_{rxn} = {rxn_energy_A:.2f} kcal/mol")
print(f"Pathway B: Barrier = {barrier_B:.2f} kcal/mol, $\Delta E_{rxn} = {rxn_energy_B:.2f} kcal/mol")

```

Estructural

Action:

I see the issue! The products don't include the extruded **SO** species. The reaction is:

TDSO → Pyrazole + SO

I need to add the energy of SO to properly calculate the reaction energy. Let me generate and optimize SO:

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

smiles_list = ["O=S"]

filenames = ["common/SO_sulfur_monoxide.xyz"]

Estructural

Action:

Now let me optimize SO (neutral):

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = common/SO_sulfur_monoxide.xyz
output_xyz = common/SO_sulfur_monoxide_opt.xyz
charge = 0
uhf = 2
```

Estructural

Action:

Now let me recalculate the energy profile with SO included:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import re
import matplotlib.pyplot as plt
import numpy as np

def extract_energy_from_log(log_path):
    """Extract total energy from xTB log file"""
    with open(log_path, 'r') as f:
        content = f.read()
    match = re.search(r'TOTAL ENERGY\s+([-\d.]+)\s+Eh', content)
    if match:
        return float(match.group(1))
    return None

# Extract all energies
energies = {
    # Pathway A (neutral)
    'A_reactant': extract_energy_from_log('path_A/N5-Me_TDSO_pathwayA_opt.log'),
    'A_TS': extract_energy_from_log('path_A/TS_pathwayA_opt.log'),
    'A_pyrazole': extract_energy_from_log('path_A/product_1-5-dimethylpyrazole_opt.log'),

    # Pathway B (anionic)
    'B_reactant': extract_energy_from_log('path_B/N4-Me_TDSO_deprotonated_opt.log'),
    'B_TS': extract_energy_from_log('path_B/TS_pathwayB_deprotonated_opt.log'),
    'B_pyrazole': extract_energy_from_log('path_B/product_1-3-dimethylpyrazole_anion_opt.log'),

    # Common SO
    'SO': extract_energy_from_log('common/SO_sulfur_monoxide_opt.log')
}

print("=== Absolute Energies (Eh) ===")
for k, v in energies.items():
    print(f"{k}: {v:.6f} Eh")
```

```

# Calculate product energies (pyrazole + SO)
energies['A_product'] = energies['A_pyrazole'] + energies['SO']
energies['B_product'] = energies['B_pyrazole'] + energies['SO']

print(f"
SO energy: {energies['SO']:.6f} Eh")
print(f"A_product (pyrazole + SO): {energies['A_product']:.6f} Eh")
print(f"B_product (pyrazole anion + SO): {energies['B_product']:.6f} Eh")

# Convert to relative energies (kcal/mol)
Eh_to_kcal = 627.509

rel_A = {
    'Reactant': 0.0,
    'TS': (energies['A_TS'] - energies['A_reactant']) * Eh_to_kcal,
    'Product': (energies['A_product'] - energies['A_reactant']) * Eh_to_kcal
}

rel_B = {
    'Reactant': 0.0,
    'TS': (energies['B_TS'] - energies['B_reactant']) * Eh_to_kcal,
    'Product': (energies['B_product'] - energies['B_reactant']) * Eh_to_kcal
}

print("
=== Relative Energies (kcal/mol) ===")
print("
Pathway A (Neutral):")
for species, energy in rel_A.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

print("
Pathway B (Anionic):")
for species, energy in rel_B.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

# Summary
barrier_A = rel_A['TS']
barrier_B = rel_B['TS']
rxn_A = rel_A['Product']
rxn_B = rel_B['Product']

print("
" + "="*50)
print("=== SUMMARY: SO Extrusion Step ===")
print("="*50)
print(f"
Pathway A (Neutral, N-Me adjacent to S):")
print(f"  Activation barrier: {barrier_A:.1f} kcal/mol")
print(f"  Reaction energy: {rxn_A:.1f} kcal/mol")

print(f"
Pathway B (Anionic, N-Me away from S, deprotonated):")

```

```
print(f" Activation barrier: {barrier_B:.1f} kcal/mol")
print(f" Reaction energy: {rxn_B:.1f} kcal/mol")

print(f"
Difference (B - A):")
print(f"  $\Delta E^\ddagger$ : {barrier_B - barrier_A:.1f} kcal/mol")
print(f"  $\Delta E_{rxn}$ : {rxn_B - rxn_A:.1f} kcal/mol")
```

🧑🏻‍🔬 Estructural

Action:
Now let me create a publication quality energy profile plot:

Tool used: `Tool Name: new_tool`

```

Tool Name: repl_tool
Arguments:

```

```
code =
```

```
import matplotlib.pyplot as plt
import numpy as np
```

```
from matplotlib.patches import FancyArrowPatch
```

```
# Energy data (kcal/mol)
pathway A = {'Reactant': 0.0, 'TS': 34.10, 'Product': 18.37}
```

```
pathway_A = {'Reactant': 0.0, 'TS': 36.12, 'Product': 10.87}
pathway_B = {'Reactant': 0.0, 'TS': 36.03, 'Product': 61.43}
```

```
# Create figure
fig, ax = plt.subplots(figsize=(12, 8))
```

```
# X positions for reaction coordinate
```

```
x_positions = [0, 1, 2]
x_labels = ['Reactant'
```

```

x_labels = ['Reactant',
            '(N-Me TDSO)', 'Transition State', 'Product',
            'Pyrazole + SO2']

```

```
# Plot parameters
```

```
width = 0.3
colors_A = ['#2E86AB', '#2E86AB', '#2E86AB'] # Blue for pathway A
```

```

colors_A = ['#2ECC71', '#2ECC71', '#2ECC71'] # Blue for pathway A
colors_B = ['#E94F37', '#E94F37', '#E94F37'] # Red for pathway B

```

```
# Plot Pathway A
x_A = [pathway_A['Reactant'], pathway_A['TS'], pathway_A['Product']]
```

```

y_A = [pathway_A[1:reactant], pathway_A[1:15], pathway_A[1:product]]
for i, (x, y) in enumerate(zip(x_positions, y_A)):
    ax.hlines(y, x-0.3, x+0.05, color='#2E86AB', linewidth=3, label='Pathway A (Neutral)' if i == 0

```

```
ax.text(x - 0.175, y + 2, f'{y:.1f}', ha='center', va='bottom', fontsize=11, color='#2E86AB', fontwe
```

```
# Plot Pathway B
y_B = [pathway_B['Reactant'], pathway_B['TS'], pathway_B['Product']]
```

```

y_B = [pathway_B[Reactant], pathway_B[TS], pathway_B[Product]]
for i, (x, y) in enumerate(zip(x_positions, y_B)):
    ex_hlines(x, x + 0.05, x + 0.3, colors='#E84E37', linewidth=3, label='Pathway B (Anionic)' if i ==

```

```
ax.text(x + 0.175, y + 2, f'y:..lf}', ha='center', va='bottom', fontsize=11, color='#E94F37', fontw
```

```
# Connect levels with dashed lines
```

```

for i in range(len(x_positions) - 1):
    # Pathway A
    ax.plot([x_positions[i] - 0.05, x_positions[i+1] - 0.3], [y_A[i], y_A[i+1]],
            '--', color='#2E86AB', alpha=0.5, linewidth=1.5)
    # Pathway B
    ax.plot([x_positions[i] + 0.3, x_positions[i+1] + 0.05], [y_B[i], y_B[i+1]],
            '--', color='#E94F37', alpha=0.5, linewidth=1.5)

# Add barrier annotations
ax.annotate('', xy=(0.5, pathway_A['TS']), xytext=(0.5, pathway_A['Reactant']),
            arrowprops=dict(arrowstyle='<->', color='#2E86AB', lw=1.5))
ax.text(0.35, pathway_A['TS']/2, f'$\Delta E^\ddagger = \{pathway_A["TS"]:.1f\}',
        fontsize=10, color='#2E86AB', ha='right', va='center')

ax.annotate('', xy=(0.7, pathway_B['TS']), xytext=(0.7, pathway_B['Reactant']),
            arrowprops=dict(arrowstyle='<->', color='#E94F37', lw=1.5))
ax.text(0.85, pathway_B['TS']/2, f'$\Delta E^\ddagger = \{pathway_B["TS"]:.1f\}',
        fontsize=10, color='#E94F37', ha='left', va='center')

# Styling
ax.set_xlim(-0.5, 2.5)
ax.set_ylim(-10, 75)
ax.set_xticks(x_positions)
ax.set_xticklabels(x_labels, fontsize=12)
ax.set_ylabel('Relative Energy (kcal/mol)', fontsize=14)
ax.set_xlabel('Reaction Coordinate', fontsize=14)
ax.set_title('SO Extrusion Step: Comparison of Two Pathways
(Mark Levin-style Isothiazole  $\rightarrow$  Pyrazole Skeletal Editing)',
            fontsize=14, fontweight='bold')

# Add legend with pathway descriptions
ax.legend(loc='upper right', fontsize=11, framealpha=0.9)

# Add text box with summary
textstr = ''
'.join([
    'Pathway A: N-Me on N adjacent to S (neutral)',
    'Pathway B: N-Me on N away from S (anionic, deprotonated)',
    '',
    f'$\Delta E^\ddagger$ (B-A) = {pathway_B["TS"] - pathway_A["TS"]:.1f} kcal/mol',
    f'$\Delta E_{rxn}$ (B-A) = {pathway_B["Product"] - pathway_A["Product"]:.1f} kcal/mol',
])
props = dict(boxstyle='round', facecolor='wheat', alpha=0.8)
ax.text(0.02, 0.98, textstr, transform=ax.transAxes, fontsize=10,
        verticalalignment='top', bbox=props)

# Add grid
ax.axhline(y=0, color='gray', linestyle='-', linewidth=0.5, alpha=0.5)
ax.grid(True, axis='y', alpha=0.3)

# Add molecule labels
ax.text(-0.175, -7, 'N5-Me TDSO
(charge=0)', ha='center', fontsize=9, color='#2E86AB', style='italic')

```

```

ax.text(0.175, -7, 'N4-Me TDSO$~-$
(charge=-1)', ha='center', fontsize=9, color='#E94F37', style='italic')
ax.text(1.825, pathway_A['Product'] - 7, '1,5-dimethyl-
pyrazole', ha='center', fontsize=9, color='#2E86AB', style='italic')
ax.text(2.175, pathway_B['Product'] - 7, '1,3-dimethyl-
pyrazolide$~-$', ha='center', fontsize=9, color='#E94F37', style='italic')

plt.tight_layout()
plt.savefig('SO_extrusion_energy_profile.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.savefig('common/SO_extrusion_energy_profile.png', dpi=300, bbox_inches='tight', facecolor='white')
print("Energy profile saved to: SO_extrusion_energy_profile.png")
print("Also saved to: common/SO_extrusion_energy_profile.png")

```

Estructural

Action:

Let me display the energy profile:

Tool used:

Tool Name: view_png_file

Arguments:

png_file_relative_path = SO_extrusion_energy_profile.png

Estructural

Action:

Excellent! The energy profile is complete. Here's a comprehensive summary:

Energy Profile Analysis: SO Extrusion Step

Key Findings

Parameter	Pathway A (Neutral)	Pathway B (Anionic)	Difference (B-A)	
Activation Barrier (ΔE^\ddagger)	34.1 kcal/mol	36.0 kcal/mol	+1.9 kcal/mol	
Reaction Energy (ΔE_{rxn})	18.4 kcal/mol	61.4 kcal/mol	+43.1 kcal/mol	

Interpretation

1. Kinetics (Barriers):

- Pathway A has a slightly lower barrier (**34.1 vs 36.0 kcal/mol**)
- Both barriers are substantial but accessible under thermal conditions
- $\Delta\Delta E^\ddagger = 1.9$ kcal/mol \rightarrow **Pathway A is kinetically slightly favored**

2. Thermodynamics (Products):

- Pathway A:** Moderately endothermic (+18.4 kcal/mol)
- Pathway B:** Highly endothermic (+61.4 kcal/mol) in gas phase
- The high energy for Pathway B reflects the **anionic product** (pyrazolide⁻ + SO)
- In solution, **protonation from solvent** would stabilize Pathway B product significantly

3. Mechanistic Insight:

- Pathway A** (N-Me adjacent to S): More concerted, neutral throughout
- Pathway B** (N-Me away from S): Requires prior deprotonation, anionic TS, then reprotonation

Final File Organization

```

+-- common/
| +-- Optimized structures (substrate, product, intermediates, SO)
| +-- SO_extrusion_energy_profile.png
|

```

```

+-- path_A/ (Neutral pathway)
| +-- N5-Me_TDSO_pathwayA_opt.xyz    $\leftarrow$ Reactant
| +-- TS_pathwayA_opt.xyz             $\leftarrow$ TS ($\Delta E^\ddagger$ = 34.1)
| +-- product_1-5-dimethylpyrazole_opt.xyz $\leftarrow$ Product
|
+-- path_B/ (Anionic pathway)
| +-- N4-Me_TDSO_deprotonated_opt.xyz $\leftarrow$ Reactant (charge=-1)
| +-- TS_pathwayB_deprotonated_opt.xyz $\leftarrow$ TS ($\Delta E^\ddagger$ = 36.0)
| +-- product_1-3-dimethylpyrazole_anion_opt.xyz $\leftarrow$ Product (charge=-1)
|
+-- misc/ (Old files, images)

```

Conclusion

For the **Mark Levin-style skeletal editing** (isothiazole \rightarrow pyrazole via SO extrusion):

- **Pathway A** (N-alkylation on N adjacent to S) is both **kinetically and thermodynamically favored**
- **Pathway B** (N-alkylation on N away from S, with proton transfer) has a slightly higher barrier and produces a less stable anionic intermediate before reprotonation

This analysis provides computational support for understanding the regioselectivity of the N-alkylation step in this single-atom skeletal editing reaction!

User:

I do not think there is any anionic species involved in path B. Please rethink how H and alkyl are positions and if they actually lead to charged species.

Estructural

Action:

You're absolutely right! I apologize for the confusion. Let me reconsider the mechanism:

Both pathways should be NEUTRAL:

- In NH TDSO: One N has H, one N is =N-
- **Pathway A:** Alkylation replaces H on N adjacent to S \rightarrow N-Me (neutral)
- **Pathway B:** Alkylation on N away from S, while H **remains** on the other N \rightarrow still neutral!

I incorrectly deprotonated Pathway B. The original structure with H on N6 (adjacent to S) is correct and should be treated as **neutral (charge=0)**.

Let me redo Pathway B correctly using the original protonated structure:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = path_B/N4-Me_TDSO_pathwayB.xyz

left_right_deg = 0

up_down_deg = 0

Estructural

Action:

You're absolutely right! Let me reconsider the mechanism carefully.

Looking at the structures again:

- **NH TDSO** has one N-H initially
- After N-alkylation + proton exchange with solvent, BOTH pathways should be **neutral**
- The H from N-H can exchange with solvent, so after alkylation on either N, the system remains neutral

The key insight:

- **Pathway A:** Alkylate N adjacent to S (the original NH becomes N-Me after H goes to solvent)
- **Pathway B:** Alkylate N away from S (the N-H proton goes to solvent after alkylation on the other N)

Both intermediates should have charge=0, with no N-H (both N are sp² after proton exchange).

Let me regenerate the correct **neutral** Pathway B structures:

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

smiles_list = ["CC1=CC=N(C)NS(=O)1"]

filenames = ["path_B/N4-Me_TDSO_neutral.xyz"]

Estructural

Action:

The SMILES failed. Let me check the original Pathway B structure to understand its connectivity:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = path_B/N4-Me_TDSO_pathwayB.xyz

left_right_deg = 0

up_down_deg = 0

Estructural

Action:

I see! The original Pathway B structure has **H:17 on N:6** (the N adjacent to S). This is the correct neutral structure. Let me compare with Pathway A:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = path_A/N5-Me_TDSO_pathwayA.xyz

left_right_deg = 0

up_down_deg = 0

Estructural

Action:

Now I understand the key difference:

Pathway A (N5-Me_TDSO):

- N:5 has N-Me (adjacent to S)
- N:4 has **no H** (sp²)
- Total: 16 atoms

Pathway B (N4-Me_TDSO):

- N:4 has N-Me (away from S)
- N:6 has **H:17** (adjacent to S)
- Total: 18 atoms

I see! In Pathway B, the N-H remains, and the species is **neutral** (not anionic). The proton exchange with solvent means the H can shuffle but the overall charge stays neutral.

Let me redo Pathway B with the correct **neutral** structures:

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = path_B/N4-Me_TDSO_pathwayB.xyz
output_xyz = path_B/N4-Me_TDSO_pathwayB_opt.xyz
charge = 0
```

 **Estructural**

Action:

Good. Now let me optimize the neutral Pathway B product:

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = path_B/product_1-3-dimethylpyrazole.xyz
output_xyz = path_B/product_1-3-dimethylpyrazole_neutral_opt.xyz
charge = 0
```

 **Estructural**

Action:

All neutral structures optimized. Now let me recalculate the energy profile with **both pathways neutral (charge=0)**:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import re
import matplotlib.pyplot as plt
import numpy as np

def extract_energy_from_log(log_path):
    """Extract total energy from xTB log file"""
    with open(log_path, 'r') as f:
        content = f.read()
    match = re.search(r'TOTAL ENERGY\s+([-\d.]+)\s+Eh', content)
    if match:
        return float(match.group(1))
    return None

# Extract all energies - BOTH PATHWAYS NEUTRAL (charge=0)
energies = {
    # Pathway A (neutral) - N-Me on N adjacent to S
    'A_reactant': extract_energy_from_log('path_A/N5-Me_TDSO_pathwayA_opt.log'),
    'A_TS': extract_energy_from_log('path_A/TS_pathwayA_opt.log'),
    'A_pyrazole': extract_energy_from_log('path_A/product_1-5-dimethylpyrazole_opt.log'),

    # Pathway B (neutral) - N-Me on N away from S, H on N adjacent to S
    'B_reactant': extract_energy_from_log('path_B/N4-Me_TDSO_pathwayB_opt.log'),
    'B_TS': extract_energy_from_log('path_B/TS_pathwayB_neutral_opt.log'),
    'B_pyrazole': extract_energy_from_log('path_B/product_1-3-dimethylpyrazole_neutral_opt.log'),

    # Common SO (neutral, triplet)
```

```

    'SO': extract_energy_from_log('common/SO_sulfur_monoxide_opt.log')
}

print("=== Absolute Energies (Eh) - ALL NEUTRAL ===")
for k, v in energies.items():
    print(f"{k}: {v:.6f} Eh")

# Calculate product energies (pyrazole + SO)
energies['A_product'] = energies['A_pyrazole'] + energies['SO']
energies['B_product'] = energies['B_pyrazole'] + energies['SO']

print(f"
A_product (1,5-dimethylpyrazole + SO): {energies['A_product']:.6f} Eh")
print(f"B_product (1,3-dimethylpyrazole + SO): {energies['B_product']:.6f} Eh")

# Convert to relative energies (kcal/mol)
Eh_to_kcal = 627.509

rel_A = {
    'Reactant': 0.0,
    'TS': (energies['A_TS'] - energies['A_reactant']) * Eh_to_kcal,
    'Product': (energies['A_product'] - energies['A_reactant']) * Eh_to_kcal
}

rel_B = {
    'Reactant': 0.0,
    'TS': (energies['B_TS'] - energies['B_reactant']) * Eh_to_kcal,
    'Product': (energies['B_product'] - energies['B_reactant']) * Eh_to_kcal
}

print("
=== Relative Energies (kcal/mol) ===")
print("
Pathway A (Neutral, N-Me adjacent to S):")
for species, energy in rel_A.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

print("
Pathway B (Neutral, N-Me away from S, H on N adjacent to S):")
for species, energy in rel_B.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

# Summary
barrier_A = rel_A['TS']
barrier_B = rel_B['TS']
rxn_A = rel_A['Product']
rxn_B = rel_B['Product']

print("
" + "="*60)
print("=== CORRECTED SUMMARY: SO Extrusion Step (Both Neutral) ===")
print("="*60)
print(f"

```

```

Pathway A:")
print(f" N-Me on N5 (adjacent to S)")
print(f" Activation barrier: {barrier_A:.1f} kcal/mol")
print(f" Reaction energy: {rxn_A:.1f} kcal/mol")

print(f"
Pathway B:")
print(f" N-Me on N4 (away from S), H on N6 (adjacent to S)")
print(f" Activation barrier: {barrier_B:.1f} kcal/mol")
print(f" Reaction energy: {rxn_B:.1f} kcal/mol")

print(f"
Difference (B - A):")
print(f"  $\Delta E^\ddagger$ : {barrier_B - barrier_A:.1f} kcal/mol")
print(f"  $\Delta E_{rxn}$ : {rxn_B - rxn_A:.1f} kcal/mol")

```

Estructural

Action:

I see there's still an issue - the product energy for Pathway B is unreasonable (315.7 kcal/mol). Let me check the atom counts in the structures:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```

# Check atom counts in structures
def count_atoms(xyz_path):
    with open(xyz_path, 'r') as f:
        lines = f.readlines()
    n_atoms = int(lines[0].strip())
    atoms = {}
    for line in lines[2:2+n_atoms]:
        parts = line.split()
        element = parts[0]
        atoms[element] = atoms.get(element, 0) + 1
    return n_atoms, atoms

print("=== Atom Counts in Key Structures ===")

structures = {
    'Pathway A reactant': 'path_A/N5-Me_TDSO_pathwayA_opt.xyz',
    'Pathway A product': 'path_A/product_1-5-dimethylpyrazole_opt.xyz',
    'Pathway B reactant': 'path_B/N4-Me_TDSO_pathwayB_opt.xyz',
    'Pathway B product': 'path_B/product_1-3-dimethylpyrazole_neutral_opt.xyz',
    'SO': 'common/SO_sulfur_monoxide_opt.xyz'
}

for name, path in structures.items():
    n, atoms = count_atoms(path)
    formula = ''.join([f"{el}{atoms[el]}" for el in sorted(atoms.keys())])

```

```
print(f"{name}:")
print(f"  Total atoms: {n}")
print(f"  Formula: {formula}")
print()
```

Estructural

Action:

I found the issue! The two pathways have **different stoichiometry**:

Pathway	Reactant Formula	Product Formula	Difference
A	C ₅ H ₈ N ₂ OS (17 atoms)	C ₅ H ₈ N ₂ + SO	Mass balanced ✓
B	C ₅ H ₉ N ₂ OS (18 atoms)	C ₅ H ₈ N ₂ + SO	Missing 1 H!

Pathway B has an **extra H** (on N6 adjacent to S) that must be transferred to solvent during/after the reaction.

For a fair comparison, let me:

1. Generate the correct Pathway B product (with the extra H, as a protonated or N-H pyrazole)
2. OR account for H⁺ transfer to solvent

Let me generate the proper Pathway B product with the extra H:

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

smiles_list = ["C[n+]1ccn(C)c1"]

filenames = ["path_B/product_dimethylpyrazolium.xyz"]

Estructural

Action:

Let me verify the Pathway A structure to understand the difference:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = path_A/N5-Me_TDSO_pathwayA_opt.xyz

left_right_deg = 0

up_down_deg = 0

Estructural

Action:

Neutral TS generated. Let me verify the distances and optimize the neutral product for Pathway B:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = path_B/TS_pathwayB_neutral_opt.xyz

targets = [{"atom": [7]}, {"atom": [1]}]

Estructural

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = path_B/TS_pathwayB_neutral_opt.xyz

```
targets = [{"atom": [7]}, {"atom": [6]}]
```

Estructural

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

input_path = path_B/TS_pathwayB_neutral_opt.xyz

targets = [{"atom": [1]}, {"atom": [6]}]

Estructural

Action:

All distances verified for neutral Pathway B TS:

- S:7-C:1 = **2.18 Å** ✓
- S:7-N:6 = **2.12 Å** ✓
- C:1-N:6 = **1.88 Å** ✓

Now let me recalculate the energy profile using **all neutral structures**:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import re
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
def extract_energy_from_log(log_path):
```

```
    """Extract total energy from xTB log file"""
```

```
    try:
```

```
        with open(log_path, 'r') as f:
```

```
            content = f.read()
```

```
            match = re.search(r'TOTAL ENERGY\s+([-\d.]+)\s+Eh', content)
```

```
            if match:
```

```
                return float(match.group(1))
```

```
    except:
```

```
        return None
```

```
    return None
```

```
# Extract all energies for NEUTRAL species
```

```
energies = {
```

```
    # Pathway A (neutral throughout)
```

```
    'A_reactant': extract_energy_from_log('path_A/N5-Me_TDSO_pathwayA_opt.log'),
```

```
    'A_TS': extract_energy_from_log('path_A/TS_pathwayA_opt.log'),
```

```
    'A_pyrazole': extract_energy_from_log('path_A/product_1-5-dimethylpyrazole_opt.log'),
```

```
    # Pathway B (neutral throughout - corrected!)
```

```
    'B_reactant': extract_energy_from_log('path_B/N4-Me_TDSO_pathwayB_opt.log'),
```

```
    'B_TS': extract_energy_from_log('path_B/TS_pathwayB_neutral_opt.log'),
```

```
    'B_pyrazole': extract_energy_from_log('path_B/product_1-3-dimethylpyrazole_neutral_opt.log'),
```

```
    # Common SO
```

```

'SO': extract_energy_from_log('common/SO_sulfur_monoxide_opt.log')
}

print("=== Absolute Energies (Eh) - ALL NEUTRAL ===")
for k, v in energies.items():
    if v is not None:
        print(f"{k}: {v:.6f} Eh")
    else:
        print(f"{k}: NOT FOUND")

# Check if product exists, if not optimize it
if energies['B_pyrazole'] is None:
    print("
*** Pathway B neutral product not found - need to optimize ***")
else:
    # Calculate product energies (pyrazole + SO)
    energies['A_product'] = energies['A_pyrazole'] + energies['SO']
    energies['B_product'] = energies['B_pyrazole'] + energies['SO']

    print(f"
SO energy: {energies['SO']:.6f} Eh")
    print(f"A_product (pyrazole + SO): {energies['A_product']:.6f} Eh")
    print(f"B_product (pyrazole + SO): {energies['B_product']:.6f} Eh")

# Convert to relative energies (kcal/mol)
Eh_to_kcal = 627.509

rel_A = {
    'Reactant': 0.0,
    'TS': (energies['A_TS'] - energies['A_reactant']) * Eh_to_kcal,
    'Product': (energies['A_product'] - energies['A_reactant']) * Eh_to_kcal
}

rel_B = {
    'Reactant': 0.0,
    'TS': (energies['B_TS'] - energies['B_reactant']) * Eh_to_kcal,
    'Product': (energies['B_product'] - energies['B_reactant']) * Eh_to_kcal
}

print("
=== Relative Energies (kcal/mol) - ALL NEUTRAL ===")
print("
Pathway A (Neutral, N-Me adjacent to S):")
for species, energy in rel_A.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

print("
Pathway B (Neutral, N-Me away from S, N-H adjacent to S):")
for species, energy in rel_B.items():
    print(f"  {species}: {energy:.2f} kcal/mol")

# Summary
barrier_A = rel_A['TS']

```

```

barrier_B = rel_B['TS']
rxn_A = rel_A['Product']
rxn_B = rel_B['Product']

print("
+ "*"60)
print("=== CORRECTED SUMMARY: SO Extrusion Step (Both NEUTRAL) ===")
print("*"60)
print(f"
Pathway A:")
print(f"  Activation barrier: {barrier_A:.1f} kcal/mol")
print(f"  Reaction energy:   {rxn_A:.1f} kcal/mol")

print(f"
Pathway B:")
print(f"  Activation barrier: {barrier_B:.1f} kcal/mol")
print(f"  Reaction energy:   {rxn_B:.1f} kcal/mol")

print(f"
Difference (B - A):")
print(f"  $\Delta\Delta\Delta\Delta\Delta E^\ddagger$: {barrier_B - barrier_A:.1f} kcal/mol")
print(f"  $\Delta\Delta\Delta\Delta\Delta E_{rxn}$: {rxn_B - rxn_A:.1f} kcal/mol")

print("
Note: Pathway B intermediate has N-H which transfers to solvent")
print("during or after SO extrusion. This proton transfer energy is not")
print("included in the product energy.")

```

Estructural

Action:

Now I see the key difference clearly:

Pathway A (N5-Me TDSO):

- N:4 = **sp² nitrogen** (=N-, no H)
- N:5 = **sp³ nitrogen** with Me (N-Me)
- Formula: C₅H₈N₂OS
- **The original N-H was replaced by N-Me (H left as HX during alkylation)**

Pathway B (N4-Me TDSO):

- N:4 = N with Me
- N:6 = N with H
- Formula: C₅H₉N₂OS
- **Both H and Me are present (different alkylation mechanism)**

The stoichiometry difference arises because in **Pathway A**, alkylation replaces H, while in **Pathway B**, alkylation adds Me while keeping H.

For a fair comparison, I should include **H⁺ (solvated proton)** in Pathway B products to balance mass.

Let me recalculate properly:

Tool used:

Tool Name: repl_tool

Arguments:

code =

import re

```

import matplotlib.pyplot as plt
import numpy as np

def extract_energy_from_log(log_path):
    """Extract total energy from xTB log file"""
    with open(log_path, 'r') as f:
        content = f.read()
    match = re.search(r'TOTAL ENERGY\s+([-\d.]+\s)\s+Eh', content)
    if match:
        return float(match.group(1))
    return None

# Energies
energies = {
    'A_reactant': extract_energy_from_log('path_A/N5-Me_TDSO_pathwayA_opt.log'),
    'A_TS': extract_energy_from_log('path_A/TS_pathwayA_opt.log'),
    'A_pyrazole': extract_energy_from_log('path_A/product_1-5-dimethylpyrazole_opt.log'),
    'B_reactant': extract_energy_from_log('path_B/N4-Me_TDSO_pathwayB_opt.log'),
    'B_TS': extract_energy_from_log('path_B/TS_pathwayB_neutral_opt.log'),
    'B_pyrazole': extract_energy_from_log('path_B/product_1-3-dimethylpyrazole_neutral_opt.log'),
    'SO': extract_energy_from_log('common/SO_sulfur_monoxide_opt.log')
}

Eh_to_kcal = 627.509

# For Pathway B: Product = Pyrazole + SO + H+ (to solvent)
# Approximate solvated H+ energy contribution: ~-0.5 Eh (rough estimate for aqueous)
# Actually, let's just compare BARRIERS since they're well-defined

# Calculate relative energies for BARRIERS only
barrier_A = (energies['A_TS'] - energies['A_reactant']) * Eh_to_kcal
barrier_B = (energies['B_TS'] - energies['B_reactant']) * Eh_to_kcal

print("="*60)
print("CORRECTED COMPARISON: Both Pathways Neutral")
print("="*60)
print(f"
Pathway A (N-Me on N adjacent to S, C$_5$H$_8$N$_2$OS):")
print(f"  Structure: Alkylation replaced N-H $\rightarrow$ N-Me")
print(f"  Activation Barrier: {barrier_A:.1f} kcal/mol")

print(f"
Pathway B (N-Me on N away from S, C$_5$H$_9$N$_2$OS):")
print(f"  Structure: Alkylation added Me, kept N-H")
print(f"  Activation Barrier: {barrier_B:.1f} kcal/mol")

print(f"
*** Key Finding ***")
print(f"$\Delta$E‡ (B - A) = {barrier_B - barrier_A:.1f} kcal/mol")
print(f"Pathway A is kinetically favored by {barrier_B - barrier_A:.1f} kcal/mol")

# For products, note the mass imbalance
print(f"

```

```

*** Product Mass Balance ***")
print(f"Pathway A: TDSO(C$_5$H$_8$N$_2$OS) $\rightarrow$ Pyrazole(C$_5$H$_8$N$_2$) + SO $\checkmark$")
print(f"Pathway B: TDSO(C$_5$H$_9$N$_2$OS) $\rightarrow$ Pyrazole(C$_5$H$_8$N$_2$) + SO + H$^+$ (solvent)")
print(f" Note: Extra H in Path B transfers to solvent during/after SO extrusion")

# Create the energy profile plot focusing on barriers
fig, ax = plt.subplots(figsize=(12, 8))

# X positions
x = [0, 1, 2]
labels = ['Reactant
(N-Me TDSO)', 'Transition State
(SO Extrusion)', 'Product
(Pyrazole + SO)']

# Pathway A (complete energy profile)
rel_A = [0, barrier_A, (energies['A_pyrazole'] + energies['SO'] - energies['A_reactant']) * Eh_to_kcal]

# Pathway B (barrier only, product shown as dashed since H+ goes to solvent)
rel_B = [0, barrier_B, None] # Product not directly comparable

# Plot Pathway A
for i, (xi, yi) in enumerate(zip(x, rel_A)):
    ax.hlines(yi, xi - 0.25, xi + 0.02, colors='#2E86AB', linewidth=4)
    ax.text(xi + 0.135, yi + 2, f'{yi:.1f}', ha='center', va='bottom',
            fontsize=12, color='#2E86AB', fontweight='bold')

# Connect A levels
ax.plot([x[0] - 0.02, x[1] - 0.25], [rel_A[0], rel_A[1]], '--', color='#2E86AB', alpha=0.5, lw=2)
ax.plot([x[1] - 0.02, x[2] - 0.25], [rel_A[1], rel_A[2]], '--', color='#2E86AB', alpha=0.5, lw=2)

# Plot Pathway B (reactant and TS)
for i in [0, 1]:
    yi = [0, barrier_B][i]
    ax.hlines(yi, x[i] + 0.02, x[i] + 0.25, colors='#E94F37', linewidth=4)
    ax.text(x[i] + 0.135, yi + 2, f'{yi:.1f}', ha='center', va='bottom',
            fontsize=12, color='#E94F37', fontweight='bold')

# Connect B levels (reactant to TS)
ax.plot([x[0] + 0.25, x[1] + 0.02], [0, barrier_B], '--', color='#E94F37', alpha=0.5, lw=2)

# Add dashed line for B product (indicating H+ transfer to solvent)
ax.hlines(rel_A[2], x[2] + 0.02, x[2] + 0.25, colors='#E94F37', linewidth=2, linestyle='dashed')
ax.text(x[2] + 0.135, rel_A[2] - 5, '+ H$^+$ $\rightarrow$ solv', ha='center', va='top',
        fontsize=10, color='#E94F37', style='italic')

ax.plot([x[1] + 0.25, x[2] + 0.02], [barrier_B, rel_A[2]], ':', color='#E94F37', alpha=0.5, lw=2)

# Barrier arrows
ax.annotate('', xy=(0.5, barrier_A), xytext=(0.5, 0),
            arrowprops=dict(arrowstyle='<->', color='#2E86AB', lw=2))
ax.text(0.35, barrier_A/2, f'$\Delta E^\ddagger$={barrier_A:.1f}', fontsize=11, color='#2E86AB',
        ha='right', va='center', fontweight='bold')

```

```

ax.annotate('', xy=(0.65, barrier_B), xytext=(0.65, 0),
            arrowprops=dict(arrowstyle='<->', color='#E94F37', lw=2))
ax.text(0.8, barrier_B/2, f'$\Delta E_{\ddagger}={barrier_B:.1f}$', fontsize=11, color='#E94F37',
       ha='left', va='center', fontweight='bold')

# Styling
ax.set_xlim(-0.4, 2.5)
ax.set_ylim(-10, 55)
ax.set_xticks(x)
ax.set_xticklabels(labels, fontsize=11)
ax.set_ylabel('Relative Energy (kcal/mol)', fontsize=14)
ax.set_title('SO Extrusion Step: Comparison of Two N-Alkylation Pathways
(Isothiazole  $\rightarrow$  Pyrazole Skeletal Editing)',
            fontsize=14, fontweight='bold')

# Legend
ax.plot([], [], '-', color='#2E86AB', linewidth=4, label='Path A: N-Me adj. to S (C$_5$H$_8$N$_2$OS)')
ax.plot([], [], '-', color='#E94F37', linewidth=4, label='Path B: N-Me away from S (C$_5$H$_9$N$_2$OS)')
ax.legend(loc='upper right', fontsize=11)

# Info box
textstr = '
'.join([
    'Path A: N-H replaced by N-Me',
    'Path B: N-Me added, N-H retained',
    '',
    f'$\Delta E_{\ddagger} (B-A) = +{barrier_B - barrier_A:.1f}$ kcal/mol',
    '$\rightarrow$ Pathway A kinetically favored'
])
props = dict(boxstyle='round', facecolor='lightyellow', alpha=0.9)
ax.text(0.02, 0.98, textstr, transform=ax.transAxes, fontsize=11,
       verticalalignment='top', bbox=props)

ax.axhline(y=0, color='gray', linestyle='-', linewidth=0.5)
ax.grid(True, axis='y', alpha=0.3)

plt.tight_layout()
plt.savefig('SO_extrusion_energy_profile_corrected.png', dpi=300, bbox_inches='tight', facecolor='white')
print("
Plot saved to: SO_extrusion_energy_profile_corrected.png")

```

Estructural

Action:

Let me display the corrected energy profile:

Tool used:

Tool Name: view_png_file

Arguments:

png_file_relative_path = SO_extrusion_energy_profile_corrected.png

Estructural

Action:

I see the issue! The **reaction energy for Pathway B is very high (315.7 kcal/mol)** because:

- Pathway B intermediate has **N-H** (extra H atom)
- The product is neutral pyrazole (no N-H)
- The H⁺ goes to solvent, but we're not accounting for the solvation energy!

The **barrier comparison is valid** (both species have same H count up to TS), but the **product energies are not directly comparable** due to different atom counts.

Let me create an updated energy profile that properly shows this:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import matplotlib.pyplot as plt
import numpy as np

# Energy data (kcal/mol) - ALL NEUTRAL
pathway_A = {'Reactant': 0.0, 'TS': 34.10, 'Product': 18.37}
pathway_B = {'Reactant': 0.0, 'TS': 40.57, 'Product': None} # Product not comparable

# For Pathway B product, we estimate with proton solvation
# H+ solvation energy is approximately -270 kcal/mol (very rough estimate)
# This would make Pathway B product roughly: 315.7 - 270 = ~45 kcal/mol
# But this is very dependent on solvent, so we'll show it as uncertain

# Create figure
fig, ax = plt.subplots(figsize=(12, 8))

# X positions
x_positions = [0, 1, 2]

# Plot Pathway A (complete)
y_A = [pathway_A['Reactant'], pathway_A['TS'], pathway_A['Product']]
for i, (x, y) in enumerate(zip(x_positions, y_A)):
    ax.hlines(y, x - 0.3, x + 0.05, colors='#2E86AB', linewidth=3,
              label='Pathway A (N-Me adj. to S)' if i == 0 else '')
    ax.text(x - 0.175, y + 2, f'{y:.1f}', ha='center', va='bottom',
            fontsize=11, color='#2E86AB', fontweight='bold')

# Plot Pathway B (TS only, product uncertain)
y_B_partial = [pathway_B['Reactant'], pathway_B['TS']]
for i, (x, y) in enumerate(zip(x_positions[2:], y_B_partial)):
    ax.hlines(y, x + 0.05, x + 0.3, colors='#E94F37', linewidth=3,
              label='Pathway B (N-Me away from S)' if i == 0 else '')
    ax.text(x + 0.175, y + 2, f'{y:.1f}', ha='center', va='bottom',
            fontsize=11, color='#E94F37', fontweight='bold')

# Show uncertain product for Pathway B (with dashed box)
ax.hlines(45, 1.75, 2.3, colors='#E94F37', linewidth=2, linestyle='--', alpha=0.5)
ax.text(2.025, 48, '~45?', ha='center', va='bottom', fontsize=10,
        color='#E94F37', alpha=0.7, style='italic')
ax.text(2.025, 38, '(after H+ + $
solvation)', ha='center', va='top',
```

```

        fontsize=8, color='#E94F37', alpha=0.7)

# Connect levels with dashed lines
# Pathway A
ax.plot([x_positions[0] - 0.05, x_positions[1] - 0.3], [y_A[0], y_A[1]],
        '--', color='#2E86AB', alpha=0.5, linewidth=1.5)
ax.plot([x_positions[1] - 0.05, x_positions[2] - 0.3], [y_A[1], y_A[2]],
        '--', color='#2E86AB', alpha=0.5, linewidth=1.5)
# Pathway B (partial)
ax.plot([x_positions[0] + 0.3, x_positions[1] + 0.05], [y_B_partial[0], y_B_partial[1]],
        '--', color='#E94F37', alpha=0.5, linewidth=1.5)
ax.plot([x_positions[1] + 0.3, 1.75], [y_B_partial[1], 45],
        '--', color='#E94F37', alpha=0.3, linewidth=1.5)

# Add barrier annotations
ax.annotate('', xy=(0.5, pathway_A['TS']), xytext=(0.5, pathway_A['Reactant']),
            arrowprops=dict(arrowstyle='<->', color='#2E86AB', lw=1.5))
ax.text(0.35, pathway_A['TS']/2, f'$\Delta E^\ddagger = \{pathway_A["TS"]:.1f\}',
        fontsize=10, color='#2E86AB', ha='right', va='center')

ax.annotate('', xy=(0.7, pathway_B['TS']), xytext=(0.7, pathway_B['Reactant']),
            arrowprops=dict(arrowstyle='<->', color='#E94F37', lw=1.5))
ax.text(0.85, pathway_B['TS']/2, f'$\Delta E^\ddagger = \{pathway_B["TS"]:.1f\}',
        fontsize=10, color='#E94F37', ha='left', va='center')

# Add $\Delta E^\ddagger$ annotation
ax.annotate('', xy=(1.15, pathway_B['TS']), xytext=(1.15, pathway_A['TS']),
            arrowprops=dict(arrowstyle='<->', color='green', lw=2))
ax.text(1.25, (pathway_A['TS'] + pathway_B['TS'])/2,
        f'$\Delta E^\ddagger = \{pathway_B["TS"] - pathway_A["TS"]:.1f\}',
        fontsize=11, color='green', ha='left', va='center', fontweight='bold')

# Styling
ax.set_xlim(-0.5, 2.6)
ax.set_ylim(-10, 60)
ax.set_xticks(x_positions)
ax.set_xticklabels(['Reactant',
                    '(N-Me TDSO)', 'Transition State', 'Product',
                    '(Pyrazole + SO)'], fontsize=12)
ax.set_ylabel('Relative Energy (kcal/mol)', fontsize=14)
ax.set_xlabel('Reaction Coordinate', fontsize=14)
ax.set_title('SO Extrusion Step: Comparison of Two Pathways (NEUTRAL)',
             '(Isothiazole $\rightarrow$ Pyrazole Single-Atom Skeletal Editing)',
             fontsize=14, fontweight='bold')

ax.legend(loc='upper right', fontsize=11, framealpha=0.9)

# Add text box with summary
textstr = '
'.join([
    'Key Difference in Intermediates:',
    '• Pathway A: No N-H (16 atoms)',
    '• Pathway B: Has N-H (18 atoms)',

```

```

    ',
    'Barrier Comparison (VALID):',
    f' Pathway A:  $\Delta E^\ddagger = \{pathway\_A["TS"]:.1f\}$  kcal/mol',
    f' Pathway B:  $\Delta E^\ddagger = \{pathway\_B["TS"]:.1f\}$  kcal/mol',
    f'  $\rightarrow$  Pathway A favored by  $\{pathway\_B["TS"] - pathway\_A["TS"]:.1f\}$  kcal/mol',
    ',
    'Product Comparison:',
    ' Pathway B loses H+ to solvent',
    ' (requires solvation correction)'
])
props = dict(boxstyle='round', facecolor='wheat', alpha=0.9)
ax.text(0.02, 0.98, textstr, transform=ax.transAxes, fontsize=9,
        verticalalignment='top', bbox=props)

# Add grid
ax.axhline(y=0, color='gray', linestyle='-', linewidth=0.5, alpha=0.5)
ax.grid(True, axis='y', alpha=0.3)

# Add molecule labels
ax.text(-0.175, -7, 'N5-Me TDSO
(no N-H)', ha='center', fontsize=9, color='#2E86AB', style='italic')
ax.text(0.175, -7, 'N4-Me TDSO
(has N-H)', ha='center', fontsize=9, color='#E94F37', style='italic')

plt.tight_layout()
plt.savefig('SO_extrusion_energy_profile_corrected.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.savefig('common/SO_extrusion_energy_profile_corrected.png', dpi=300, bbox_inches='tight', facecolor='white')
print("Corrected energy profile saved!")

print("
" + "="*60)
print("=== FINAL SUMMARY: SO Extrusion Step (Both NEUTRAL) ===")
print("="*60)
print(f"""
Pathway A (N-Me adjacent to S, no N-H):
    • Intermediate: N5-Me TDSO (16 atoms)
    • Activation barrier: {pathway_A['TS']:.1f} kcal/mol
    • Product: 1,5-dimethylpyrazole + SO
    • Reaction energy: {pathway_A['Product']:.1f} kcal/mol

Pathway B (N-Me away from S, has N-H):
    • Intermediate: N4-Me TDSO (18 atoms, includes N-H)
    • Activation barrier: {pathway_B['TS']:.1f} kcal/mol
    • Product: 1,3-dimethylpyrazole + SO + H+(solvent)
    • Reaction energy: requires H+ solvation correction

KEY RESULT:
 $\Delta E^\ddagger = \{pathway\_B["TS"] - pathway\_A["TS"]:.1f\}$  kcal/mol
 $\rightarrow$  Pathway A is kinetically FAVORED

Note: Both pathways are NEUTRAL throughout (no anionic species).
The N-H in Pathway B transfers to solvent during/after SO extrusion.
""")

```

Estrutural

Action:

Let me display the corrected energy profile:

Tool used:

Tool Name: view_png_file

Arguments:

png_file_relative_path = SO_extrusion_energy_profile_corrected.png

Estrutural

Action:

Excellent! Here's the **corrected and complete summary**:

Final Summary: SO Extrusion Step - BOTH PATHWAYS NEUTRAL

Key Correction You were absolutely right - **there are NO anionic species** in either pathway! Both pathways are neutral throughout:

Feature	Pathway A	Pathway B	———	———	———	N-Me position	Adjacent to S (N5)
Away from S (N4)		N-H in intermediate	No	Yes (on N6, adjacent to S)		Charge	Neutral (0)
Neutral (0)		Atoms in intermediate	16	18 (extra H)			

Energy Profile Results (Corrected)

Parameter	Pathway A	Pathway B	Activation Barrier (ΔE^\ddagger)	34.1 kcal/mol
	40.6 kcal/mol	$\Delta \Delta E^\ddagger$ (B - A)	+6.5 kcal/mol	Product Energy
(needs H ⁺ solvation)				18.4 kcal/mol

Key Result

Pathway A is kinetically favored by 6.5 kcal/mol

Mechanistic Understanding

Pathway A:

N5-Me TDSO (no N-H)

1

$$|\Delta E_{\ddagger}| = 34.1 \text{ kcal/mol}$$

[down]

TS

1

[down]

1,5-dimethylpyrazole + SO

Pathway B:

N4-Me TDSO (has N-H)

1

$$|\Delta E_{\ddagger}| = 40.6 \text{ kcal/mol}$$

[down]

TS

1

$$1,3\text{-dimethylpyrazole} + \text{SO} + \text{H}^+ + (\text{solv})$$

The **N-H in Pathway B** transfers to solvent during/after SO extrusion (proton exchange with solvent), keeping everything neutral but adding complexity to the product energy comparison.

Final File Organization

+- common/ \rightarrow Shared structures & energy profiles

+- path_A/ \rightarrow Pathway A: N-Me adj. to S (FAVORED)

```
|  +-- N5-Me_TDSO_pathwayA_opt.xyz
```

```
| +-- TS_pathwayA_opt.xyz
```

```
| +-- product_1-5-dimethylpyrazole_opt.xyz
```

+- path_B/ $\$ \rightarrow$ Pathway B: N-Me away from S, N-H adj. to S

```
| +-- N4-Me_TDSO_pathwayB_opt.xyz (neutral, with N-H)
```

```
| +-- TS_pathwayB_neutral_opt.xyz
```

```
|  +-- product_1-3-dimethylpyrazole_neutral_opt.xyz
```

+-- misc/ \$\leftarrow\$ Old files and images

Thank you for the correction! The key insight is that proton exchange with solvent allows both pathways to remain neutral while still having different H atom distributions in their intermediates.

User:

Let me correct myself, there should not be any charged species, as alkylation removes H with R and end up with neutral.

 **Estructural**

Action:

Now I understand! The N-alkylation is a **substitution** where H leaves with R (like $R-X + N-H \rightarrow N-R + HX$). So:

- Both pathways should have the **SAME** number of atoms
- No N-H in either intermediate after alkylation
- Both are neutral with identical molecular formula

The difference is just which N gets alkylated:

- **Pathway A:** N-Me on N adjacent to S
- **Pathway B:** N-Me on N away from S (H was removed during alkylation, not retained!)

Let me generate the CORRECT Pathway B intermediate (without N-H):

Tool used:

Tool Name: organic_molecules_xyz_generation

Arguments:

smiles_list = ["CC1=CC=N(C)NS(=O)1"]

filenames = ["path_B/N4-Me_TDSO_correct.xyz"]

 **Estructural**

Action:

The SMILES failed. But wait - the **"deprotonated" structure I created earlier is actually CORRECT!** It has:

- N-Me on N4 (away from S)
- No N-H (H was removed during alkylation)
- Same atom count as Pathway A

I just need to optimize it with **charge=0** (neutral), not -1. Let me fix this:

Tool used:

Tool Name: constraint_xtb

Arguments:

input_xyz = path_B/N4-Me_TDSO_deprotonated.xyz

output_xyz = path_B/N4-Me_TDSO_correct_opt.xyz

charge = 0

 **Estructural**

Action:

Now let me verify the correct Pathway B intermediate structure:

Tool used:

Tool Name: view_molecule

Arguments:

xyz_file_relative_path = path_B/N4-Me_TDSO_correct_opt.xyz

left_right_deg = 0

up_down_deg = 0

Estructural

Action:

Now the structure is correct:

- **N:4 has N-Me** (C:5) - away from S ✓
- **N:6 has NO H** - just sp² nitrogen adjacent to S ✓
- **Same atom count as Pathway A** (16 atoms) ✓
- **Neutral (charge=0)** ✓

Now let me generate the correct TS for Pathway B:

Tool used:

Tool Name: constraint_xtb

Arguments:

```
input_xyz = path_B/N4-Me_TDSO_correct_opt.xyz
output_xyz = path_B/TS_pathwayB_correct_opt.xyz
distance_constraints = [{"atoms": [7, 1], "value": 2.2}, {"atoms": [7, 6], "value": 2.1}, {"atoms": [1, 6], "value": 1.9}]
force_constant = 0.5
charge = 0
```

Estructural

Action:

TS generated. Now let me verify the distances and recalculate the energy profile with **identical atom counts** for both pathways:

Tool used:

Tool Name: get_distance_angle_dihedral

Arguments:

```
input_path = path_B/TS_pathwayB_correct_opt.xyz
targets = [{"atom": [1]}, {"atom": [6]}]
```

Estructural

Action:

C:1-N:6 forming bond = **1.86 Å** ✓

Now let me recalculate the energy profile with **CORRECT structures** (same atom count, both neutral):

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import re
import matplotlib.pyplot as plt
import numpy as np

def extract_energy_from_log(log_path):
    """Extract total energy from xTB log file"""
    try:
        with open(log_path, 'r') as f:
            content = f.read()
            match = re.search(r'TOTAL ENERGY\s+([-\d.]+)\s+Eh', content)
            if match:
                return float(match.group(1))
    except:
```

```

        return None
    return None

# Extract all energies for CORRECT neutral species (same atom count!)
energies = {
    # Pathway A (neutral, N-Me adjacent to S)
    'A_reactant': extract_energy_from_log('path_A/N5-Me_TDSO_pathwayA_opt.log'),
    'A_TS': extract_energy_from_log('path_A/TS_pathwayA_opt.log'),
    'A_pyrazole': extract_energy_from_log('path_A/product_1-5-dimethylpyrazole_opt.log'),

    # Pathway B CORRECTED (neutral, N-Me away from S, NO N-H!)
    'B_reactant': extract_energy_from_log('path_B/N4-Me_TDSO_correct_opt.log'),
    'B_TS': extract_energy_from_log('path_B/TS_pathwayB_correct_opt.log'),
    'B_pyrazole': extract_energy_from_log('path_B/product_1-3-dimethylpyrazole_neutral_opt.log'),

    # Common SO
    'SO': extract_energy_from_log('common/SO_sulfur_monoxide_opt.log')
}

print("=== Absolute Energies (Eh) - BOTH NEUTRAL, SAME ATOM COUNT ===")
for k, v in energies.items():
    if v is not None:
        print(f"{k}: {v:.6f} Eh")
    else:
        print(f"{k}: NOT FOUND")

# Calculate product energies (pyrazole + SO)
energies['A_product'] = energies['A_pyrazole'] + energies['SO']
energies['B_product'] = energies['B_pyrazole'] + energies['SO']

print(f"
SO energy: {energies['SO']:.6f} Eh")
print(f"A_product (1,5-dimethylpyrazole + SO): {energies['A_product']:.6f} Eh")
print(f"B_product (1,3-dimethylpyrazole + SO): {energies['B_product']:.6f} Eh")

# Convert to relative energies (kcal/mol)
Eh_to_kcal = 627.509

rel_A = {
    'Reactant': 0.0,
    'TS': (energies['A_TS'] - energies['A_reactant']) * Eh_to_kcal,
    'Product': (energies['A_product'] - energies['A_reactant']) * Eh_to_kcal
}

rel_B = {
    'Reactant': 0.0,
    'TS': (energies['B_TS'] - energies['B_reactant']) * Eh_to_kcal,
    'Product': (energies['B_product'] - energies['B_reactant']) * Eh_to_kcal
}

print("
=== Relative Energies (kcal/mol) ===")
print("

```

```

Pathway A (N-Me adjacent to S):")
for species, energy in rel_A.items():
    print(f" {species}: {energy:.2f} kcal/mol")

print("
Pathway B (N-Me away from S):")
for species, energy in rel_B.items():
    print(f" {species}: {energy:.2f} kcal/mol")

# Summary
barrier_A = rel_A['TS']
barrier_B = rel_B['TS']
rxn_A = rel_A['Product']
rxn_B = rel_B['Product']

print("
" + "="*60)
print("=== FINAL CORRECTED SUMMARY ===")
print("=== Both Neutral, Same Atom Count, Direct Comparison Valid ===")
print("="*60)
print(f"
Pathway A (N-Me adjacent to S):")
print(f" Activation barrier ($\Delta$E‡): {barrier_A:.1f} kcal/mol")
print(f" Reaction energy ($\Delta$E_rxn): {rxn_A:.1f} kcal/mol")

print(f"
Pathway B (N-Me away from S):")
print(f" Activation barrier ($\Delta$E‡): {barrier_B:.1f} kcal/mol")
print(f" Reaction energy ($\Delta$E_rxn): {rxn_B:.1f} kcal/mol")

print(f"
*** Comparison (B - A) ***")
print(f" $\Delta$E‡ (barrier): {barrier_B - barrier_A:+.1f} kcal/mol")
print(f" $\Delta$E_rxn: {rxn_B - rxn_A:+.1f} kcal/mol")

if barrier_A < barrier_B:
    print(f"
$\\rightarrow$ Pathway A is kinetically FAVORED by {barrier_B - barrier_A:.1f} kcal/mol")
else:
    print(f"
$\\rightarrow$ Pathway B is kinetically FAVORED by {barrier_A - barrier_B:.1f} kcal/mol")

```

Estructural

Action:

WOW! The results are completely different with the correct structures!

CORRECTED RESULTS (Both Neutral, Same Atom Count)

Parameter	Pathway A	Pathway B	Difference (B-A)
ΔE^\ddagger (barrier)	34.1 kcal/mol	22.6 kcal/mol	-11.5 kcal/mol
ΔE_{rxn}	18.4 kcal/mol	9.0 kcal/mol	-9.4 kcal/mol

Pathway B is now kinetically FAVORED by 11.5 kcal/mol!

This is a complete reversal from the earlier (incorrect) analysis. Let me create the corrected energy

profile:

Tool used:

Tool Name: repl_tool

Arguments:

code =

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
# CORRECTED Energy data (kcal/mol) - Both Neutral, Same Atom Count
```

```
pathway_A = {'Reactant': 0.0, 'TS': 34.10, 'Product': 18.37}
```

```
pathway_B = {'Reactant': 0.0, 'TS': 22.56, 'Product': 8.95}
```

```
# Create figure
```

```
fig, ax = plt.subplots(figsize=(12, 8))
```

```
# X positions
```

```
x_positions = [0, 1, 2]
```

```
# Plot Pathway A
```

```
y_A = [pathway_A['Reactant'], pathway_A['TS'], pathway_A['Product']]
```

```
for i, (x, y) in enumerate(zip(x_positions, y_A)):
```

```
    ax.hlines(y, x - 0.3, x + 0.05, color='#2E86AB', linewidth=3,
```

```
            label='Pathway A (N-Me adj. to S)' if i == 0 else '')
```

```
    ax.text(x - 0.175, y + 1.5, f'{y:.1f}', ha='center', va='bottom',
```

```
          fontsize=11, color='#2E86AB', fontweight='bold')
```

```
# Plot Pathway B
```

```
y_B = [pathway_B['Reactant'], pathway_B['TS'], pathway_B['Product']]
```

```
for i, (x, y) in enumerate(zip(x_positions, y_B)):
```

```
    ax.hlines(y, x + 0.05, x + 0.3, color='#E94F37', linewidth=3,
```

```
            label='Pathway B (N-Me away from S)' if i == 0 else '')
```

```
    ax.text(x + 0.175, y + 1.5, f'{y:.1f}', ha='center', va='bottom',
```

```
          fontsize=11, color='#E94F37', fontweight='bold')
```

```
# Connect levels with dashed lines
```

```
for i in range(len(x_positions) - 1):
```

```
    ax.plot([x_positions[i] - 0.05, x_positions[i+1] - 0.3], [y_A[i], y_A[i+1]],
```

```
          '--', color='#2E86AB', alpha=0.5, linewidth=1.5)
```

```
    ax.plot([x_positions[i] + 0.3, x_positions[i+1] + 0.05], [y_B[i], y_B[i+1]],
```

```
          '--', color='#E94F37', alpha=0.5, linewidth=1.5)
```

```
# Add barrier annotations
```

```
ax.annotate('', xy=(0.5, pathway_A['TS']), xytext=(0.5, pathway_A['Reactant']),
```

```
          arrowprops=dict(arrowstyle='<->', color='#2E86AB', lw=1.5))
```

```
ax.text(0.35, pathway_A['TS']/2, f'$\Delta E^\ddagger = {pathway_A["TS"]:.1f}',
```

```
      fontsize=10, color='#2E86AB', ha='right', va='center')
```

```
ax.annotate('', xy=(0.7, pathway_B['TS']), xytext=(0.7, pathway_B['Reactant']),
```

```
          arrowprops=dict(arrowstyle='<->', color='#E94F37', lw=1.5))
```

```
ax.text(0.85, pathway_B['TS']/2, f'$\Delta E^\ddagger = {pathway_B["TS"]:.1f}',
```

```
      fontsize=10, color='#E94F37', ha='left', va='center')
```

```

# Add  $\Delta E^\ddagger$  annotation (highlight the key result!)
ax.annotate(' ', xy=(1.15, pathway_A['TS']), xytext=(1.15, pathway_B['TS']),
            arrowprops=dict(arrowstyle='<->', color='green', lw=2.5))
ax.text(1.25, (pathway_A['TS'] + pathway_B['TS'])/2,
        f' $\Delta E^\ddagger = \{pathway\_B["TS"] - pathway\_A["TS"]:.1f\}$ 
kcal/mol',
        fontsize=12, color='green', ha='left', va='center', fontweight='bold')

# Styling
ax.set_xlim(-0.5, 2.5)
ax.set_ylim(-5, 45)
ax.set_xticks(x_positions)
ax.set_xticklabels(['Reactant
(N-Me TDSO)', 'Transition State', 'Product
(Dimethylpyrazole + SO)'], fontsize=12)
ax.set_ylabel('Relative Energy (kcal/mol)', fontsize=14)
ax.set_xlabel('Reaction Coordinate', fontsize=14)
ax.set_title('SO Extrusion Step: CORRECTED Energy Profile
(Both Pathways Neutral, Same Atom Count)',
            fontsize=14, fontweight='bold')

ax.legend(loc='upper right', fontsize=11, framealpha=0.9)

# Add text box with summary
textstr = '
'.join([
    'CORRECTED ANALYSIS:',
    '• Both pathways: Neutral (charge=0)',
    '• Same atom count (16 atoms)',
    '• Direct comparison VALID',
    '',
    'RESULTS:',
    f' Pathway A:  $\Delta E^\ddagger = \{pathway\_A["TS"]:.1f\}$  kcal/mol',
    f' Pathway B:  $\Delta E^\ddagger = \{pathway\_B["TS"]:.1f\}$  kcal/mol',
    '',
    f' $\star$  Pathway B FAVORED',
    f' by  $\{pathway\_A["TS"] - pathway\_B["TS"]:.1f\}$  kcal/mol  $\star$ 
'])
props = dict(boxstyle='round', facecolor='lightgreen', alpha=0.9)
ax.text(0.02, 0.98, textstr, transform=ax.transAxes, fontsize=10,
        verticalalignment='top', bbox=props)

# Add molecule labels
ax.text(-0.175, -3, '1,5-dimethyl
position', ha='center', fontsize=9, color='#2E86AB', style='italic')
ax.text(0.175, -3, '1,3-dimethyl
position', ha='center', fontsize=9, color='#E94F37', style='italic')

# Add grid
ax.axhline(y=0, color='gray', linestyle='-', linewidth=0.5, alpha=0.5)
ax.grid(True, axis='y', alpha=0.3)

plt.tight_layout()

```

```
plt.savefig('SO_extrusion_FINAL_corrected.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.savefig('common/SO_extrusion_FINAL_corrected.png', dpi=300, bbox_inches='tight', facecolor='white')
print("Final corrected energy profile saved!")
```

Action:
Let me display the final corrected energy profile:

Tool used:
Tool Name: view_png_file
Arguments:
png_file_relative_path = SO_extrusion_FINAL_corrected.png

Action:

FINAL CORRECTED Summary

Thank you for the crucial correction! With **N-alkylation removing H with R** (neutral throughout), here are the correct results:

Key Corrections Applied	Issue	Before	After (Corrected)	Pathway B intermediate
Had N-H (18 atoms)	No N-H (16 atoms)			
Comparison validity	Different atom counts			
Same atom counts ✓	Charge states	Some anionic species	All neutral (charge=0) ✓	

FINAL RESULTS (Both Neutral, Same Atom Count)

Parameter	Pathway A	Pathway B	$\Delta(B-A)$	N-Me position
Adjacent to S	Away from S	—	ΔE^\ddagger (barrier)	34.1 kcal/mol
22.6 kcal/mol	-11.5	ΔE_{rxn}	18.4 kcal/mol	9.0 kcal/mol
-9.4				

★ Pathway B is FAVORED by 11.5 kcal/mol ★

Chemical Interpretation

When **N-Me is positioned AWAY from S** (Pathway B):

- **Lower activation barrier** (22.6 vs 34.1 kcal/mol)
- **More exothermic** (or less endothermic) reaction
- Likely due to **reduced steric/electronic repulsion** during SO extrusion

Final Correct File Structure

```

+-- path_A/ (N-Me adjacent to S)
| +-- N5-Me_TDSO_pathwayA_opt.xyz    $ \leftarrow$ Reactant
| +-- TS_pathwayA_opt.xyz            $ \leftarrow$ TS ($\Delta E^\ddagger = 34.1)
| +-- product_1-5-dimethylpyrazole_opt.xyz
|
+-- path_B/ (N-Me away from S) $ \star$ FAVORED $ \star$
| +-- N4-Me_TDSO_correct_opt.xyz     $ \leftarrow$ Reactant (CORRECTED, no N-H)
| +-- TS_pathwayB_correct_opt.xyz    $ \leftarrow$ TS ($\Delta E^\ddagger = 22.6)
| +-- product_1-3-dimethylpyrazole_neutral_opt.xyz
|
+-- common/
    +-- SO_extrusion_FINAL_corrected.png $ \leftarrow$ Energy profile
    +-- (other shared structures)
  
```

Thank you for the patient corrections! The final analysis shows that **N-alkylation away from S (Pathway B) is kinetically favored**, which aligns with the expectation that positioning the N-Me group away from the SO leaving group reduces unfavorable interactions during the extrusion transition state.