……

·AlphaFold 3·····MSA·····JSON·····G
PU···

·····MSA·····AlphaF
old 3·····
·····

·····

·····
·JSON···MSA·····JSON···chain ID·····JS
ON·····AlphaFold 3····JSON·····ID·····

··· | ····· | ····

·· | ····· | ·····MSA/··

·· | modifications····· | ·····

MSA··· | ····· | ·····

··· | ·····null | ·····

·····

·····JSON·····MSA····JSON····CPU·····
·····MSA···JSON·····

```
#!/usr/bin/env bash
set -euo pipefail
af3=/opt/alphafold3
model=/data/af3/models
db=/data/af3/databases
root="$PWD/af3-fixed-chains-001"
test -s "$af3/run_alphafold.py"
test -d "$model"
test -d "$db"
test ! -e "$root" || { echo "output exists: $root" >&2; exit 2; }
mkdir -p "$root/data_A" "$root/data_B"
python "$af3/run_alphafold.py" --json_path="$PWD/monomers/A.json"   --output_dir="$root/data_A" --model_dir="$model" --db_dir="$db"   --run_inference=false > "$r
oot/A.log" 2>&1
python "$af3/run_alphafold.py" --json_path="$PWD/monomers/B.json"   --output_dir="$root/data_B" --model_dir="$model" --db_dir="$db"   --run_inference=false > "$r
oot/B.log" 2>&1
find "$root/data_A" "$root/data_B" -type f -name '*_data.json' -print
```

·····JSON·····A·B·····data JSON·····TSV·····ID·····data JSON
·····JSON·····TSV·····chain ID·

·····

··Python·····JSON·····dialect·version·····Path
·····unpairedMsa·pairedMsa·templates·····
·····JSON··SHA256·····

```
from pathlib import Path
import copy, csv, hashlib, json, sys
base = json.loads(Path(sys.argv[1]).read_text())
def no_paths(x):
    if isinstance(x, dict):
        assert not any(k.endswith("Path") for k in x)
        for v in x.values(): no_paths(v)
    elif isinstance(x, list):
        for v in x: no_paths(v)
no_paths(base)
chains = {}
for entity in base["sequences"]:
    assert set(entity) == {"protein"}
    p = entity["protein"]
    assert p["id"] not in chains
    chains[p["id"]] = p
with open(sys.argv[2], newline="") as f:
    rows = list(csv.reader(f, delimiter=" "))
assert rows and all(len(r) == 2 for r in rows)
assert len({r[0] for r in rows}) == len(rows)
audit = []
for chain_id, filename in rows:
    assert chain_id in chains
    path = Path(filename); assert path.is_absolute()
    donor = json.loads(path.read_text()); no_paths(donor)
    assert donor["dialect"] == base["dialect"]
    assert donor["version"] == base["version"]
    assert len(donor["sequences"]) == 1
    src = donor["sequences"][0]["protein"]; dst = chains[chain_id]
    assert src["sequence"] == dst["sequence"]
    assert src.get("modifications", []) == dst.get("modifications", [])
    assert isinstance(src.get("unpairedMsa"), str)
    assert isinstance(src.get("pairedMsa"), str)
    assert isinstance(src.get("templates"), list)
    for key in ("unpairedMsa", "pairedMsa", "templates"):
        assert dst.get(key) is None
        dst[key] = copy.deepcopy(src[key])
    audit.append({"chain": chain_id, "source": str(path),
                  "sha256": hashlib.sha256(path.read_bytes()).hexdigest()})
out = Path(sys.argv[3]); assert not out.exists(); out.mkdir()
(out/"complex.json").write_text(json.dumps(base, indent=2))
(out/"sources.json").write_text(json.dumps(audit, indent=2))
```

·····

··JSON·····MSA/·····null·····--run_data_pipeline=false·····JSON·
·····paired MSA·····unpaired MSA·····paired MSA···
·····

·····MSA·····
·····GPU···GPU·····
·····

·····I/O····CPU/RAM·····GPU
·····

····

AlphaFold 3·····MSA····· https://github.com/google-deepmind/alphafold3/blob/main/docs/performance.md

AlphaFold 3·····MSA···· https://github.com/google-deepmind/alphafold3/blob/main/docs/input.md

AlphaFold 3····· https://github.com/google-deepmind/alphafold3/blob/main/run_alphafold.py