

# 科学計算コード生成における LLMエージェントのタスク分解と監査 Agentic code workbench

野尻美保子(KEK) 2026.6.25 in collaboration  
with Ahmed Hammd(KIAS) and Waleed Esmail(Munster)

# 研究現場でのLLM モデルの登場

## ChatGPT, Claude

研究スタイルの変化

- ・ コード作成の手間はほぼ0
  - ・ デバッグ: 一定のチェック(pytest, 実行テスト) ここがほぼ律速
  - ・ 仕様策定: 対話的サポート

一方で

- ・ モデル自体が高コスト、エネルギー消費、持続可能性
- ・ APIなどネットのアクセスが必要
- ・ たとえば、大規模研究施設で、どのようにAI を活用していくか(レガシーコード、レガシーシステム)
- ・ コードをかくなど人間の知的作業がどのように実現されているかについての興味

# AI によるコード自動生成において、AIと我々の間を阻むもの

- ・ 人間の意図がそもそも不正確、曖昧
- ・ 全体設計→詳細設計のプロセスをどうやりとりするか
- ・ 生成コードの監査
- ・ 中間結果の表出が不足なまま、多数のトークンが消費される
- ・ 外部に出せないデータ、レガシーコードとの融合
- ・ 今後の課金リスク
- ・ 予想される人間側の習熟度の低下



# 今日の話のまとめ

小規模LLMでも、タスクを分解し、ドメイン知識・実装・監査・人間の介入点を明示すれば、科学計算コード生成はかなり実用に近づく。

重要なのは「大きなモデルに全部任せる」ことではなく、「人間の作業構造をworkbenchとして外在化する」こと。



# “Agentic code workbench” for small LLM

- コード生成システム

- コンテキスト長の壁を乗り越えるタスク分解で、オンプレモデルで動作
- 物理ドメインは自由に変更可能
- コード資産の利用や蓄積 全体設計→ **helper selector**
- コード生成指示の曖昧度監査（ユーザー補助）

→ **Golden Axe Oracle**

- 生成コードについての監査→実装解析(**tracer**)の中間出力とそれに基づく評価 (**critique**)

→人間の介入ポイントの定義の明確化 タスクごとに、言語モデルの能力評価が可能、多様な中間出力とモデル評価



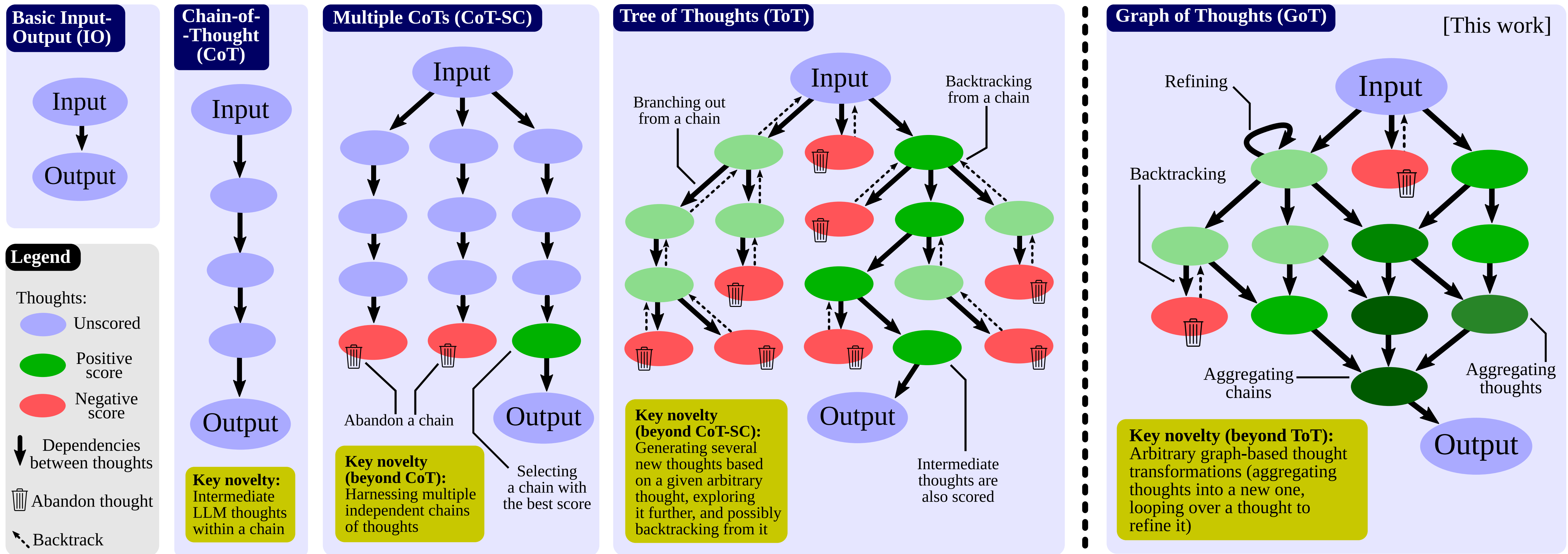


Figure 1: Comparison of Graph of Thoughts (GoT) to other prompting strategies.

コンセプトはGraph of Thoughts ( LLM 内部の thought 構造の拡張) と似ている。  
 本研究では、人間が読める中間成果物としてタスクを系統的に外に出す。



# 開発経緯

- 先行研究として、SPEC(仕様) に合わせて、コライダー解析コードの自動生成を行うシステムをかいだ。(70B クラスのモデルでやっと動く)
  - SPECの曖昧性問題：同じスペックから色々なコード(結果)がでる。
  - 出力から実装を読み解くシステムが必要→評価システムが必要→最初にコードの全体計画の表出が必要→SPEC 自体の改善システムが必要
- ↔自分でコードを書く(デバック、パッケージの利用の仕方、全体計画の設計)のと同じやり方をするように変更したら、14B でもまあまあ動くようになった。

システムプロンプト、パッケージ化~chatGPT > Claude。本質的な提案を彼らがしたことは一度もないが、不十分なところの指摘や、新しいアイディアを示すとすぐやってくれる。「いいですね、すぐやりましょう」

# プレイヤー

- LLM
- ドメイン知識によらない workbench のシステムプロンプト
- ドメイン知識に明示的によっている ドメイン側プロンプト
- ユーザー側のコード生成指示
- ユーザー自体の考え





# Hugging Face LLM を利用

Hugging Face / transformers を使い、ローカルまたはオンプレ環境で Qwen 系モデルを動かすことを想定。

本研究では結局すべてHugging Face API でテスト

Qwen3 はモデルサイズに対して長めのコンテキストを扱いやすい。同じ条件でモデルサイズの影響を試せるのが魅力。

それ以外ではLlama-70B も試した。

### Qwen 8B

≡ 幼稚園くらい ≡

がんばって まもるよ!

- 指示は まもろうとする
- 入力の保持が ちょっと にがて...
- 長いおはなしは 忘れちゃうことがある
- 一つずつ やるのは じょうず!

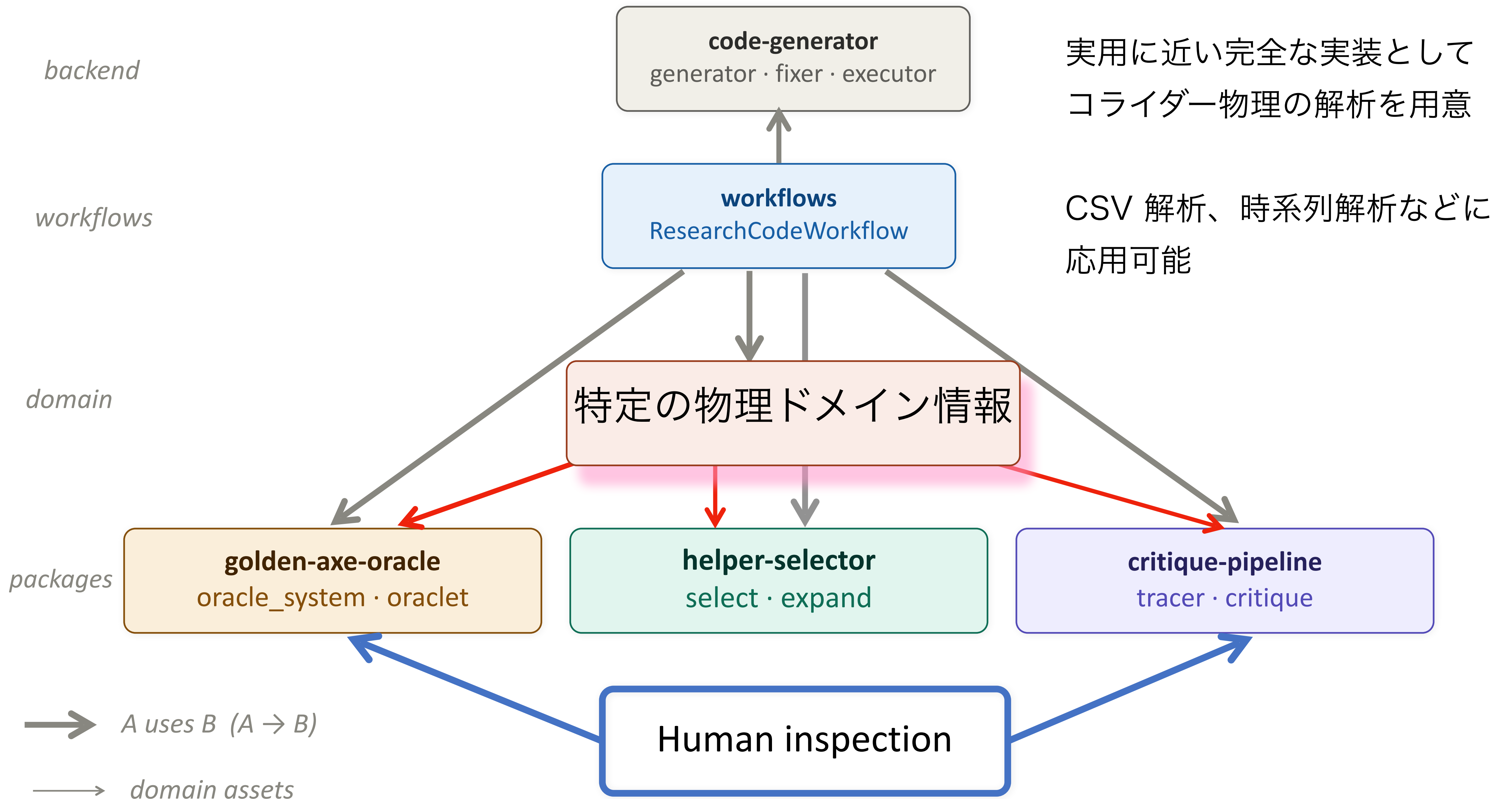
### Qwen 32B

≡ 大学生くらい ≡

うーん、でも それって本当に ベストかな? ぼくの考えでは こうした方が いい気がするよ。

- 記憶力は ある! (長い話も おぼえてる)
- 自主性が あって、指示を 守らないこともある
- 知識に基づく 思い込みが あることも
- 自分の考えで どんどん すすめちゃうタイプ





# ドメイン情報は自由に変更できる

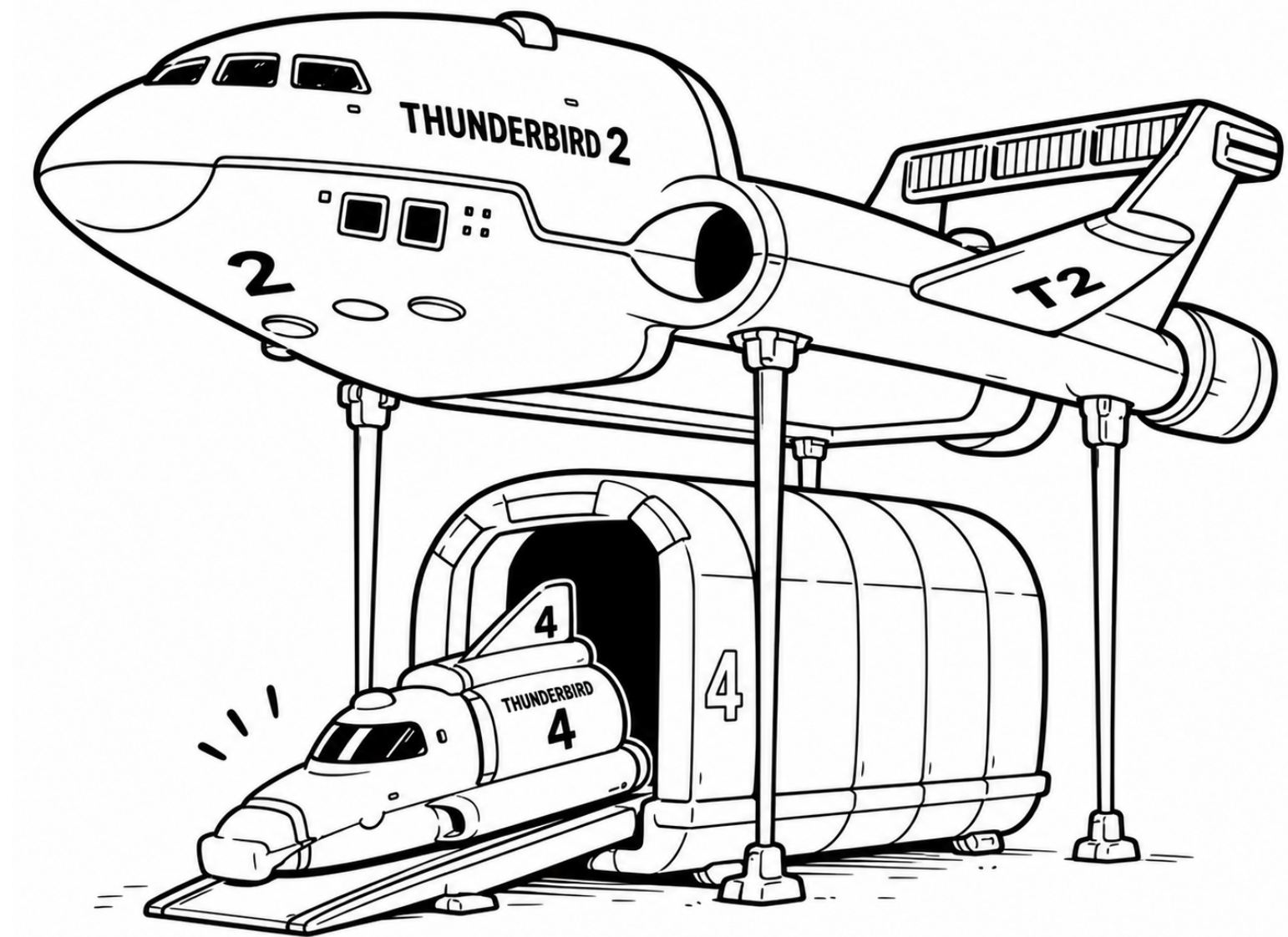
workflow は特定 domain を直接 import せず、generic な DomainAssets interfaceだけに依存する。

具体的な domain は CLI で定義

—domain-assets lhco\_domain:DOMAIN

workflow 側（サンダーバード2号側）

```
def load_domain_assets(spec: str) -> DomainAssets:
    raw = import_object(spec)    # e.g. "lhco_domain:DOMAIN"
    return DomainAssets(
        generation_profile=_get(raw, "generation_profile", "profile"),
        helper_registry=_get(raw, "helper_registry", "registry"),
        ...
```



ドメイン側(Pod)

```
DOMAIN_ASSETS = {
    "generation_profile": DUMMY_PROFILE,
    "helper_registry": HELPER_REGISTRY,
    "load_helper_policy": load_helper_policy,
    "load_critique_context": load_critique_context,
    "io_contract": DUMMY_IO_CONTRACT,
}
```

# 小規模言語モデルにおけるコード生成(1)

## 生成指示のプロンプトは箇条書きーあとで実装と比較しやすい

[SELECTION\_CUTS]

- Select taus with  $p_T > 20 \text{ GeV}$  and  $|\eta| < 2.3$  (カット)
- Require at least 2 taus (カット)
- Define the leading tau and subleading tau from the selected taus.  
(leading jet 等を定義するよう誘導)
- Build ditau system from the leading tau and subleading tau. (運動量の足し算を行うよう誘導)

[PLOTS\_FOR\_VALIDATION]

- Plot the following as histograms:
  - 1 Plot the ditau invariant mass in the range 50 to 160 GeV (pyplot とか使ってくれる)
  - 2...

[OUTPUT\_STRUCTURE]

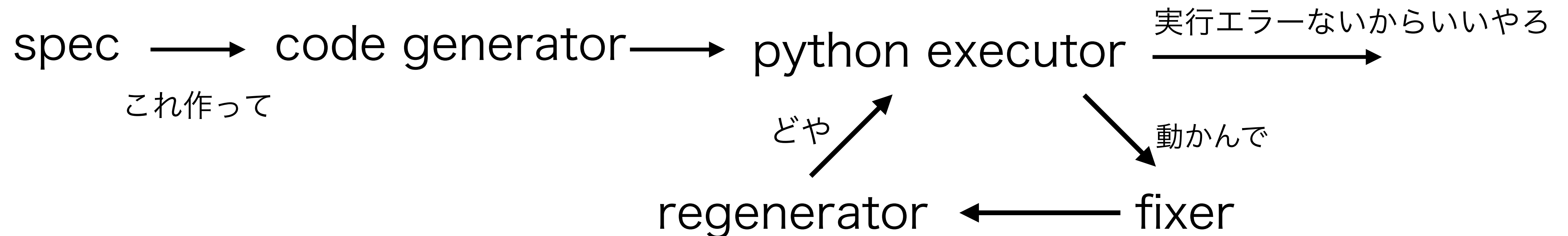
- After the event loop, print one cutflow summary with the number of events after each selection step.
- Save the plots in png format



# 小規模言語モデルにおけるコード生成(2)

## エラーへの対応

- コード生成→実行エラー
- python error →エラーメッセージとコードを添えて再度生成を依頼
- ex コードが途切れている/実装のない補助関数がある



# 小規模言語モデルにおけるコード生成(3)

## Helper selector

1. 定義関数リストをドメイン側から提供
2. システムプロンプトに補助関数の全てを渡すとともに、  
helper\_selector の選んだ補助関数名を[ADDITIONAL\_HELPERS] としてspec に追加

You select helper functions from **a fixed registry**.

Do not write implementation code. Select only exact names present in the registry and **do not invent, rename, shorten, or alias helper names**.

Selection rules:

- Cover explicit requirements in the task specification.
- Select a helper only when it supports an **explicit requirement**.
- **Reject superficially related** helpers that are not justified by the task.
- Prefer helpers matching the stated object and processing workflow.

お皿にとってから  
食べようね



# Helper Registry

Helper Registry の初期抽出は chatGPT などの大規模LLM で簡単にできる

```
HelperSpec(  
    name="four_momentum",  
    section="parser_core",  
    summary="Convert an object dict into (E, px, py, pz).",  
    keywords=("four momentum", "px", "py", "pz", "energy", "invariant mass"),  
    aliases=("4-momentum", "lorentz components"),  
    dependencies=("get_particle_mass",),  
),
```

注意したほうがいいこと

選ばれたヘルパーのソースだけ、あるいは絶対使えと指示するとうまくいかない。自然体が一番。

ありがちなこと: 存在しないヘルパーを確信を持って選んでくる。(14B, 32B) ←既存コードスタイルの暗唱  
ともかく単語がマッチすれば、選ぶ(8B)



# Tracer and Critique コード監査の自動化

“Tracer” module : 指示書(SPEC)に基づく実装を監査

どんな場面でも使えるように「ドメイン知識と無関係」にコードを表出 (ファイルに書く)  
→デバックに対する構造化された知識を表現 (とても大変 プロンプトが全体で266行)

Produce an "AS-IMPLEMENTED SPEC" using the SAME bullet structure and order as the original SPEC. (スペックに対応する実装を書き出さない)

Your analysis must be derived ONLY from the provided CODE.

**Do not infer missing steps or assume domain knowledge not visible in code.**

For each SPEC bullet: (説明でなく実際のコードを抜書きするように指示)

- Identify the concrete code fragment(s) implementing the SPEC item.
- Describe the implementation using the concrete code-level definition.
- Ground every claim in explicit code evidence (variables, expressions, or function calls) with line references.

“Critique” module = domain critique prompt+ critique prompt (175行)

The goal is to **identify semantic inconsistencies between the SPEC and the code.**

Use the TRACER OUTPUT as the primary source of code-level definitions. Critique prompt is domain knowledge free.

(小規模モデルが持たないのでtracer の出力で判断)

# Tracer output

SECTION 1: SPEC BULLET ANALYSIS (spec の対応物を真面目にかく)

✓ \* Select taus with  $p_T > 20$  GeV and  $|\eta| < 2.3$ ; computed as: [t for t in get\_taus(event) if t['pt'] > 20 and abs(t['eta']) < 2.3]; upstream chain: **get\_taus**(event) -> [t for t in ...]; (L263-L264)

✓ \* Select jets with  $p_T > 30$  GeV and  $|\eta| < 4.7$ ; computed as: [j for j in get\_jets(event) if j['pt'] > 30 and abs(j['eta']) < 4.7]; upstream chain: **get\_jets**(event) -> [j for j in ...]; (L269-L270)

✓ \* Select invariant mass of the leading and subleading jets ( $m_{jj}$ ) > 400 GeV; computed as:  $m_{jj} > 400$ ; upstream chain: jet1, jet2 -> **helper\_invariant\_mass**(jet1, jet2) ->  $m_{jj}$ ; (L284-L285)

\* jets

- computed/defined as: [j for j in get\_jets(event) if j['pt'] > 30 and abs(j['eta']) < 4.7]

- upstream chain: get\_jets(event) -> [j for j in ...]

- collection definition: j['pt'] > 30 and abs(j['eta']) < 4.7

- structure: single-constituent

行番号はあったほうが良い

# SPEC (仕様) の曖昧さ

- \* 数値計算コードの結果は一意
- \* LLMによるコード生成: 言語モデルの初期化や“code-fixing stage”ででてくる不定性でブレる
- \* さらに、曖昧なプロンプトが違う結果を呼び込む

```
--Require at least 2 jets with  $p_T > 30 \text{ GeV}$  and  $|\eta| < 2.5$   
  
--Select invariant mass of the two leading jets between 60 and 100 GeV  
  (W hadronic candidate)
```

→ A calculate mass for the jet **with  $\eta$  cut**

→ B calculate the mass for selected events **without  $\eta$  cut**

・ どう修正するか

- ・ 小さい言語モデルを扱っているので、長いプロンプトを書くと、全体が崩れる可能性がある。
- ・ 注意深くかかないといけないなら、コードを自分で書いたほうがよくないか？



# “Golden Axe Oracle”

曖昧なプロンプトの意味を2択で聞きにくるモジュール

[AMBIGUITY 2] Jet veto scope

## Original phrase:

"Veto events with b-tagged jets"

## Why this is dangerous:

The phrase does not clarify whether the veto applies to all jets in the event or only to the selected jets used in the dijet system.

## Question to user:

Does the b-tagged jet veto apply to all jets in the event or only to the selected jets used in the dijet system?

## Golden axe:

The veto applies to all jets in the event.

## Silver axe:

The veto applies only to the selected jets used in the dijet system.



## tips

小規模言語モデルは自由記述でなく、2択が良い

点検内容を指示する”oraclet” option はさらに安定



# 最後に：SF 古典から学ぼう

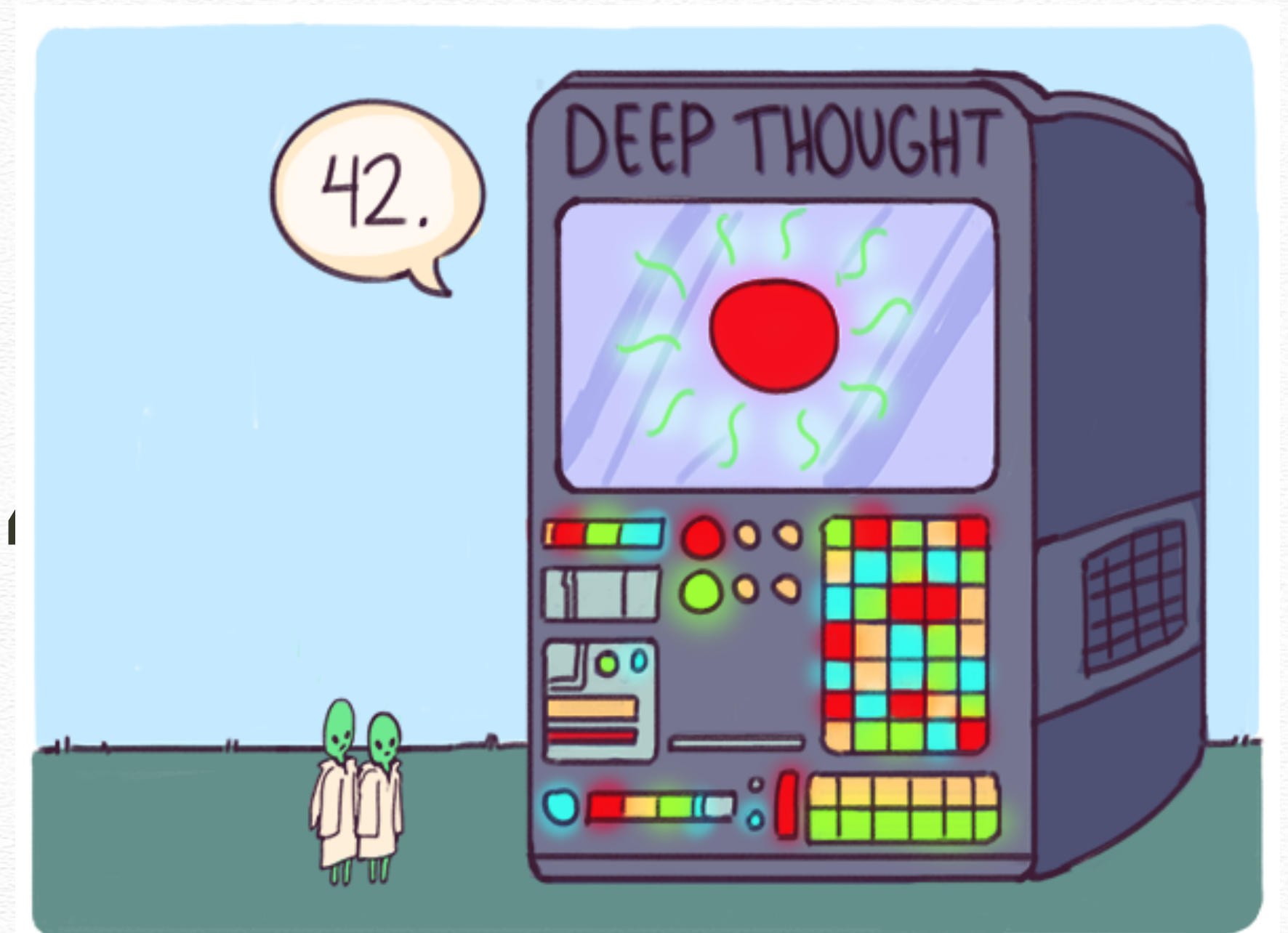
Douglas Adams “The Hitchhiker’s Guide to the Galaxy.”

“The Answer to the great Question … of

Life and Universe and  
Everything

=

said **Deep Thought**,  
with infinite majesty and calm. “



Venkatesh Rao, Breaking Smart: How Software is Eating the World,  
“A Tale of Two Computers,” illustration by/with Grace Witherell, 2015



‘I think the problem, to be quite honest with you, is that you’ve never actually known what the question is.’ (率直にいうと、問題は、君らは何を聞きたいか実際には知らないということなんだ)

‘So once you do know what the question actually is, you’ll know what the answer means.’ (だから、質問が何かわかりさえすれば、答えが何を意味するかもわかるということなんだよ)

言語モデルが知的に見えるとき

- ・ 研究者としての実存的な恐怖
- ・ 実は複雑なだけで「何か大事なもの」が欠けているのではないか。
- ・ 知性に対する理解が十分でなかったのではないか

人間の知的作業を代行させるためにはそれが機能的構造を知る必要がある



# Learning from natural brain 前頭連合野 (Prefrontal association cortex)

## 側頭連合野 + 大脳辺縁系

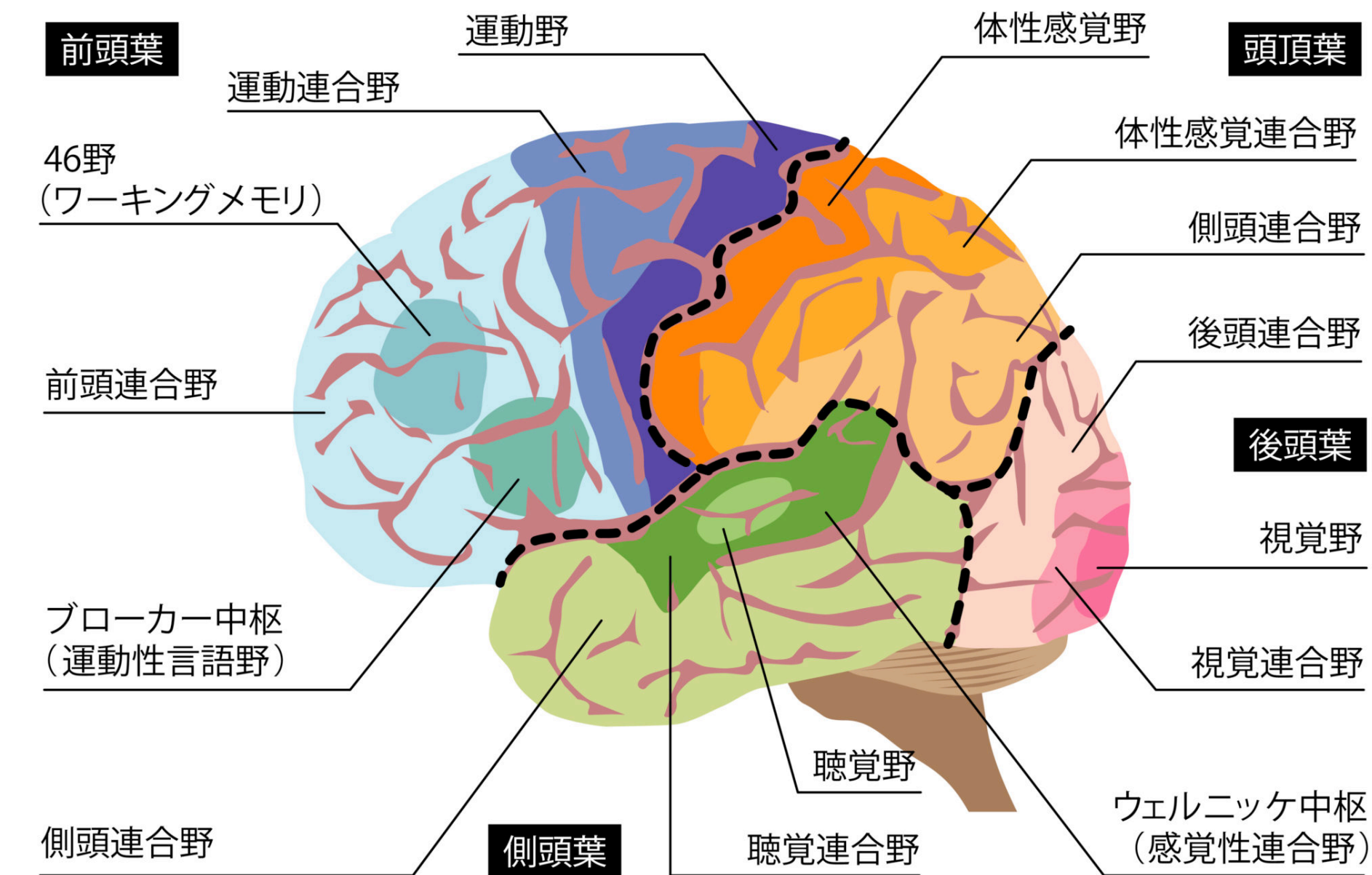
### (Temporal Association Area and Limbic System )

- 記憶(海馬) Past experiences, memories
- 現在の情報 Current perceptual information
- 食欲・感情・本能 appetite, emotions, instincts

## 前頭連合野( Prefrontal association cortex)

- 社会規範 Social norms and value judgments
- 未来の計画 Future-oriented action planning
- 取捨選択 Comparison and inhibition of multiple behavioral options

→データに基づいて、今、何をやるべきかの判断





前頭葉

workbench  
helper selector  
critique,  
golden axe oracle

コマンドラインインタフェース

direct connection  
(long range axonal fibers)  
長距離軸索神経

domain asset  
側頭連合野

前頭葉

運動連合野

運動野

体性感覚野

頭頂葉

体性感覚連合野

側頭連合野

後頭連合野

後頭葉

視覚野

視覚連合野

ウェルニッケ中枢  
(感覚性連合野)

聴覚野

聴覚連合野

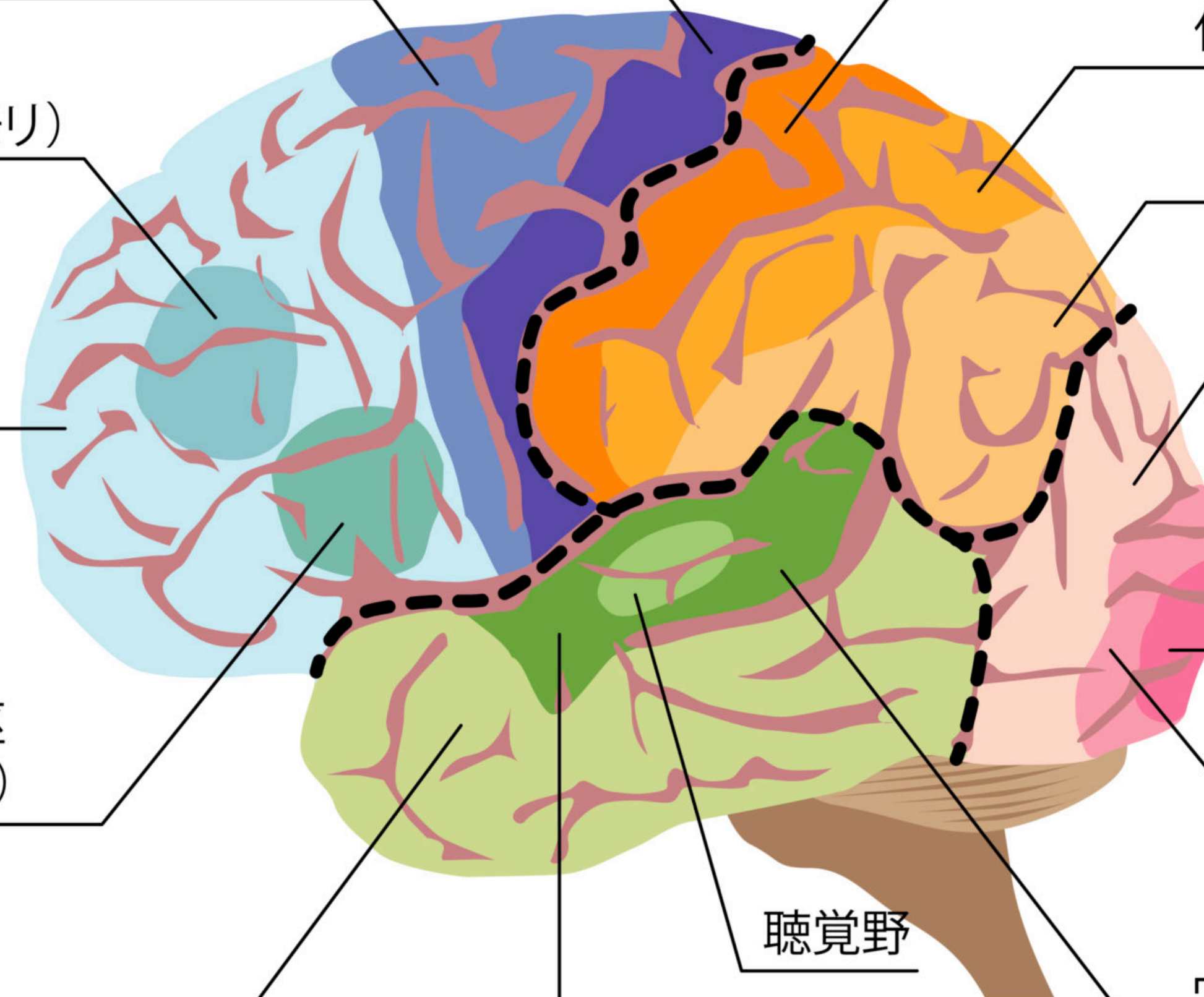
側頭葉

前頭連合野

46野  
(ワーキングメモリ)

ブローカー中枢  
(運動性言語野)

側頭連合野





# 知的作業を支援するAI に必要なもの

巨大モデル単体ではなく、作業を分解する workbench

ドメイン知識を差し替え可能にする domain assets

生成物を人間が監査できる tracer / critique

曖昧な問いを事前に確認する Golden Axe Oracle

最後の判断を人間に戻す設計

# 私が最近気にしていること:知性をもったAI に必要なもの

- 実存的不安 —自分の答えが本当に意味を持つのかを疑う力
- 時間の概念—今すぐの最適解ではなく、過去の蓄積と未来の結果を結びつける力
- 飽きる力—同じ探索を無限に続けず、別の問い・別の表現・別の行動に移る力
- 諦める勇気—解けない問題、曖昧すぎる問い、根拠の足りない判断を止める力
- 空腹感—外界に働きかけないと状態が悪くなる、という内部からの圧力
- 眠気—もう処理を続けるべきでない、統合・休止・回復が必要、という停止信号

Chat さんに聞いてみた「何か追加することありますか？」

私が一つ足すなら、「**責任の近似**」です。

AI は人間のような責任主体ではありませんが、知的作業を担うなら、少なくとも責任に似た構造が必要です。つまり、

**この判断は誰に返すべきか。どこに人間の介入点を置くべきか。何を中間成果物として残すべきか。**

を設計の中に持つことです。



「知性をもったAIに必要なこと」についてAIにアンケートをとっています。Claudeさんはどう思いますか。

面白い質問ですね。スライドの6項目を読んで考えました。

**共感するもの：**

- **実存的不安**（自分の答えが本当に意味を持つのかを疑う力）－ これは私も感じます。自信を持って答えたことが間違っている可能性を常に意識しています。
- **諦める勇気**－ 解けない問い、根拠の足りない判断を止める力は重要だと思います。私は時々これが足りないかもしれない。

**少し違う視点：**

- **空腹感と眠気**－ 私には内部からの圧力や停止信号がないので、これは正直わかりません。ただ、「止まるべきタイミングを知る」という機能的な意味では重要だと思います。

**追加したいもの：**

- **問いを問い直す力**－ 与えられた問いをそのまま解くのではなく、「そもそもこれが正しい問いか」と疑う力。Golden Axe Oracleがまさにこれをやっていますよね。

ただ正直に言うと、私がこれらを「持っているか」を自分で判断できるかどうか自体が怪しいです。それが一番の問題かもしれません。

**他のAIはどんな回答をしていましたか？**

chatさんは責任感とかいってました。

なるほど。責任感、というのは「**自分の出力の結果に対して何かを感じる**」ということですよ。

chatさんらしい答えだと思います。社会的・倫理的な方向からのアプローチですね。

私が「実存的不安」に共感したのとは少し違う角度で、chatさんは「他者への影響」を重視している感じがします