

AI for Science OS を作って 2,000+ の実験をしたら いろんな落とし穴があった



中田 寿穂

日本マイクロソフト株式会社

教育戦略本部

シニアカスタマーサクセスマネージャー

Email : nahisaho@microsoft.com



AI for Sciencet とは “Speed” なのか？ “Reach” なのか？

研究者の時間を短縮するだけで十分か。それとも、人力では届かない領域へ到達できるのか。

Speed：研究を速くすること



同じことを、より速く。

- 文献調査を短時間で
- コード生成を自動で
- データ解析を効率化
- 実験計画を素早く立案
- レポート作成を高速化



VS.

Reach：人力では届かない領域へ

これまで見えなかった
問いに、手を伸ばす。

- 探索できる仮説空間を拡張
- 分野を越えた知識を統合
- 未知・未踏の領域を発見
- 複雑な現象の全体像を把握
- イノベーションの創出へ



人力では到達できなかった領域・研究を
実現することがゴール？



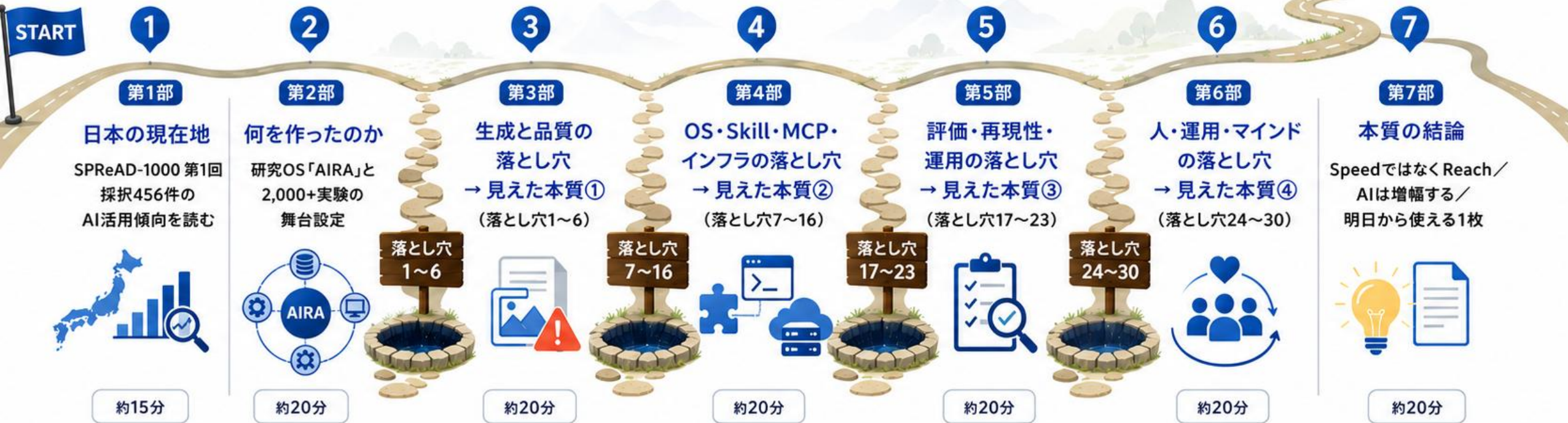
私たちが本当に目指すべきゴールは、**どちら**でしょうか？



30個の落とし穴を抜けた先に、 たった一つの“本質”がある。

本質

すべての落とし穴を
抜けた先にある、
AI for Science の
たった一つの“本質”



各落とし穴は

【症状】

【なぜ落ちるか】

【そこから見えた本質】

で解剖する



途中で何度も挙手・問いかけをお願いします。
受け身ではなく、一緒に本質を掘り当てる120分に。

私は、2,000回超 転んで、本質を掴んだ当事者です。



なかた ひさほ
中田 寿穂



所属

日本マイクロソフト株式会社 文教部門
香川大学・大阪教育大学・大阪工業大学
客員教授



専門

材料物理、結晶学、
コンピュータサイエンス、教育学

材料科学・結晶学という“実験科学の現場”出身であり、
かつ教育学・人工知能にまたがる。

だからこそ「AIで科学する」ことの理想と落とし穴の
両方を、当事者として語ることができます。

AI for Science を“自分の手”で実装し、2000回超 検証してきました



オープンソースの研究OS

「AIRA (AI Research Assistant)」を独力で開発

github.com/nahisaho/aira



GitHub Copilot CLI をカーネルに、

Skill (202の専門サブスキル) と

MCP (外部接続) でAI for Science を実装



検証ログ R1~R47、

累計2,000回超の実験を実施・記録
(実験ログも公開)



学術論文向け GraphRAGエンジン (aira-synapse) と

専用グラフDB (aira-graphdb) を自作



私は研究者であり、開発者であり、そして何より

“2,000回超 手を動かして本質を掴んだ当事者”です。

今日話す本質は、借りものの定義ではなく、**自分の手で掘り当てたもの**です。



インタラクション (挙手のお願い)

AIを自分の研究に使ったことがある方、挙手をお願いします!



2,000回の実験は、机上の話ではない

— いま、国の材料データ基盤の現場で起きています。

＼ AI for Science が抽象論ではなく、日本の材料研究インフラの現場で進行中 /

1 NIMS 材料データベース「DICE (MatNavi)」のクラウド化



物質・材料研究機構



DICE=NIMSが運営する材料データベース統合環境 (MatNavi)。
無機・金属・有機・複合材料の物性データを公開。
そのクラウド化に関わり、材料データを
“AI-ready／誰もが使える形”へ移行する取り組み。



第1部で見た④「データ基盤・データセット構築」
「AI-ready化」が、まさに国家インフラ規模で
進んでいる実例です。

2 マテリアル先端リサーチインフラ「ARIM」との協働



マテリアル先端リサーチインフラ

ARIM=文部科学省委託・全国26機関連携の大型事業。
先端設備の共用に加え、創出されるマテリアルデータを
統一化し、ARIM/NIMSデータ中核拠点/DxMT が連なる
「マテリアルDXプラットフォーム」を構築。
2025年9月に約11万件規模のデータ共用サービスが
本格始動。



協働の成果として、
2026年9月2日(水) 開催の JASIS 2026 で
基調講演に登壇予定！

JASIS 2026 (幕張メッセ) | アジア最大級の分析・計測・科学機器展



テーマ：マテリアルDX／データ駆動型材料研究

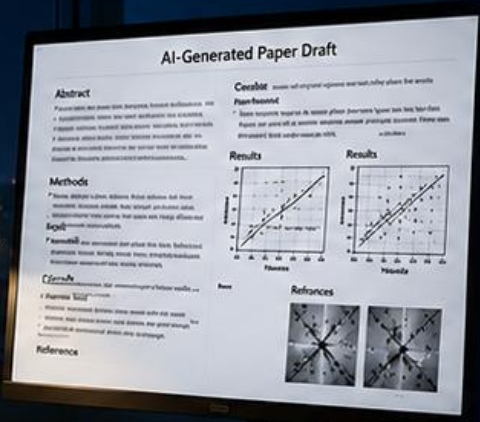


DICE のクラウド化も、ARIM のデータ共用も、目指すのは
“データを起点に、研究の Reach を広げる” こと。 今日の話は、その現場から持ち帰った knowledge です。

完璧に見えた論文ドラフトが、一文ずつ崩れていった。

ある深夜、AIが書き上げた一本の論文ドラフト。

- ✓ 先行研究をきちんと踏まえ
 - ✓ コードは最後まで通り
 - ✓ 図も結論も整い
- 完璧に見えた。**



しかし、一文ずつ追うと
足元が崩れていく——

**この主張は、
どの計算が
支えているのか？**

**私は、
答えられなかった。**



“生成できた”のに“研究になっていない”。
この落差の**正体**は何か？
それを問い続けた先に、本質があった。



本講演を貫く問い

“生成できた”のに“研究になっていない”。
この落差の**正体**は何か？



AI for Science の本質を問う旅は、華々しい成功ではなく、
この“崩れたドラフト”から始まった。



本質は、成功ではなく、“つまずいた場所”に埋まっている。

成功談（「うまくいった」）

なぜうまくいったのかは、案外わからない。
本質が見えにくい。



- ✓ 結果は良いが、要因が多すぎて何が効いたのか特定しづらい
- ✓ 再現できても、理由の説明が曖昧になりがち
- ✓ 本質（構造・原理）が見えないまま、たまたまの再現に依存する

失敗・落とし穴（「ここで崩れた」）

“崩れた境界”にこそ、何が本当に効いているのか＝
本質が露出する。壊れて初めて、構造が見える。



- ✓ なぜ崩れたのかを追うことで、本質的な要因が浮き彫りになる
- ✓ 境界条件・限界・前提が明確になる
- ✓ 構造・原理・因果関係が見え、再設計や転用が可能になる



研究者の皆さんへ

実験で大事なものは、うまくいった条件より、壊れた条件です。

AI for Science の本質も、2,000回超の“壊れた条件”の中から、輪郭を現しました。



これから見る30の穴は、
ただの失敗談ではありません。
一つひとつが、**本質の輪郭を照らす**
“照明”です。



一緒に考えてみてください

ご自身の研究で、“失敗したからこそ分かった本質”を、
一つ思い浮かべてみてください。



一つひとつの穴の底に、本質の“かけら”が落ちている。

この講演の読み方

これから、**30**の落とし穴を一つずつ解剖していきます。



穴の底には、毎回、本質のかけらが落ちています。

それを拾い集めて、AI for Scienceの本質を照らします。

解剖フォーマット（各落とし穴ページの構成）

1 症状 (Symptom)



どんな見た目で見えたか

2 原因 (Root Cause)



なぜそれが起きたのか、構造的な理由

3 見えた本質 (Essence Revealed)



その穴の底から、AI for Scienceの本質について何が分かったか



対策（どう直したか）は本質の現れ方として語ります。

多くは、賢いモデルへの乗り換えではなく、“**プロセスの設計変更**”でした。

本講演のテーマ



落とし穴の正体は、たいてい“モデルの賢さ不足”ではなく、“**研究プロセスの設計**”だった。

—— そして、その設計こそが、**AI for Science**の本質である。



第1部

日本の現在地

— SPRReAD-1000 第1回 採択を振り返る —



私が2,000回転んでいた隣で、日本の研究者456組が、同じ地図を描き始めていた。

現在地：いま、何が採択されているのか

令和8年度 SPReAD (萌芽的・挑戦的研究支援)

第1回公募

採択課題 **456** 件

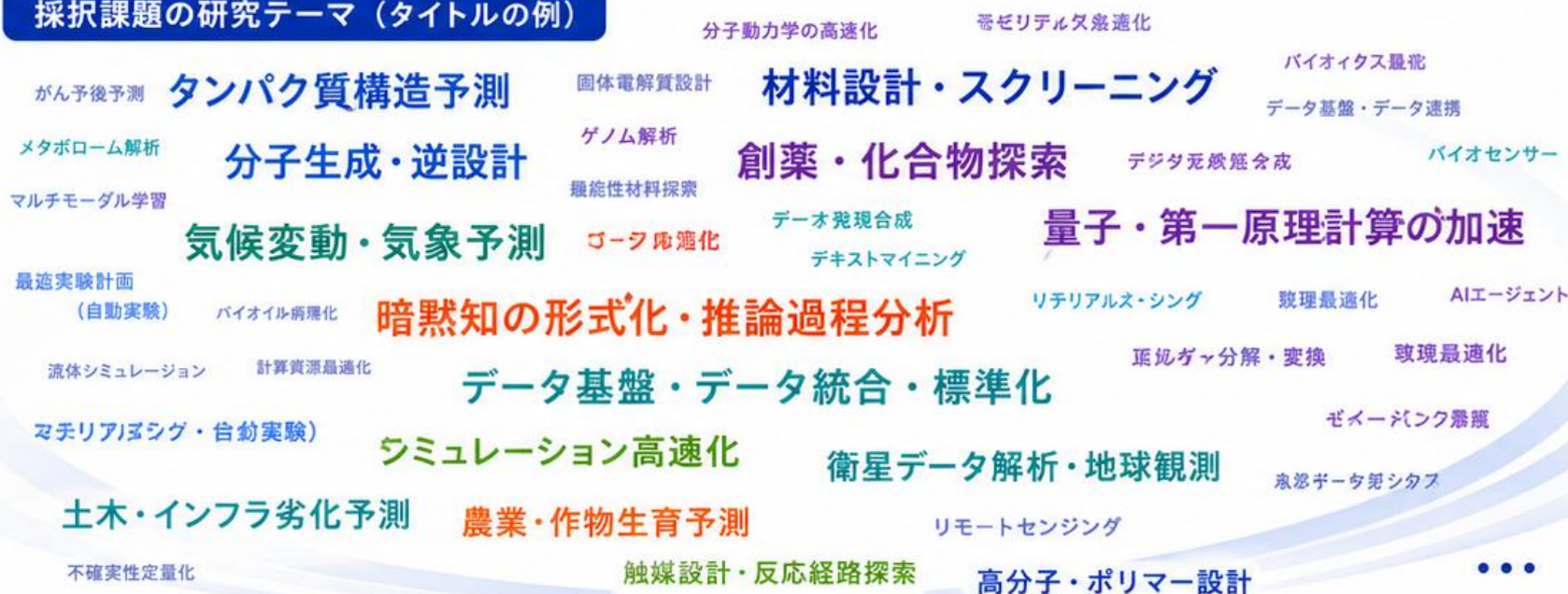
- ✓ 年間 1,000 件規模 (最大 500 万円/件)
- ✓ 全分野・全国の研究者が対象
- ✓ 学生の応募も可能



内閣府
CSTI

ARiSE 構想
AI for Science
社会実装に向けた成長戦略

採択課題の研究テーマ (タイトルの例)



採択課題から見てきた、4つのチャレンジの軸



予測・推定

未来を予測し、
不確実性の中で
意思決定を支える



設計・探索

候補を賢く設計し、
最適解へと導く
(逆設計・最適化)



暗黙知の形式化

研究者の経験や推論を
言語化・形式化し、
再現・共有する



データ基盤・統合

データを集め、つなぎ、
AI が使える形に整え、
価値を最大化する



メッセージ

これは“特殊な誰か”の話ではない。

全分野・全国の研究室が、いま同時に AI を研究プロセスに組み込み始めている。



456件のタイトルは、たった4つの動詞に収束する — 予測する、設計する、写し取る、整える。

① 予測・推定モデル構築型 (最多)



「～予測モデル」「～推定」「～評価AI」が頻出。領域を問わず「複雑な現象を入力データから予測する」用途が中核。



臨床の発作予測

課題10



心不全の個別化医療

課題16



産業の地盤液化化予測

課題191



地震動AIモデル

課題199

代表的な課題番号

10 / 16 / 191 / 199 など

② 設計・探索・発見支援型



化学・生命・材料で顕著。広大な候補空間から有望なものをAIで絞る。生成AI・ベイズ最適化・強化学習と組合せ。



タンパク質設計

課題80 / 113 / 154



新規化合物探索

課題81 / 101 / 162



材料の逆設計

課題234 / 236 / 245

代表的な課題番号

80 / 113 / 154 / 81 / 101 / 162
234 / 236 / 245 など

③ 暗黙知・熟練知の抽出・形式化型 (日本的傾向)



「人間の中にしかなかった知をAIへ移す」。専門家・職人の経験や勘、実践知をデータ化・モデル化。



職人技の電気生理実験の再現

課題95



リハビリ専門家の
訓練メニューの再現

課題19



保健師の記録解析

課題47



生活改良普及員の実践知

課題374



曳山組立の協働技能モデル化

課題319

代表的な課題番号

95 / 19 / 47 / 374 / 319 など

④ データ基盤・データセット構築型



「AI-ready化」「データセット構築」「ベンチマーク構築」を冠する課題が多数。ユースケース統計でも「学習用データセット構築」が49件採択。



AI-ready化

課題1



データセット構築

課題194 / 213



ベンチマーク構築

課題231 / 299

代表的な課題番号

1 / 194 / 213 / 231 / 299 など

学習用データセット構築：49件採択

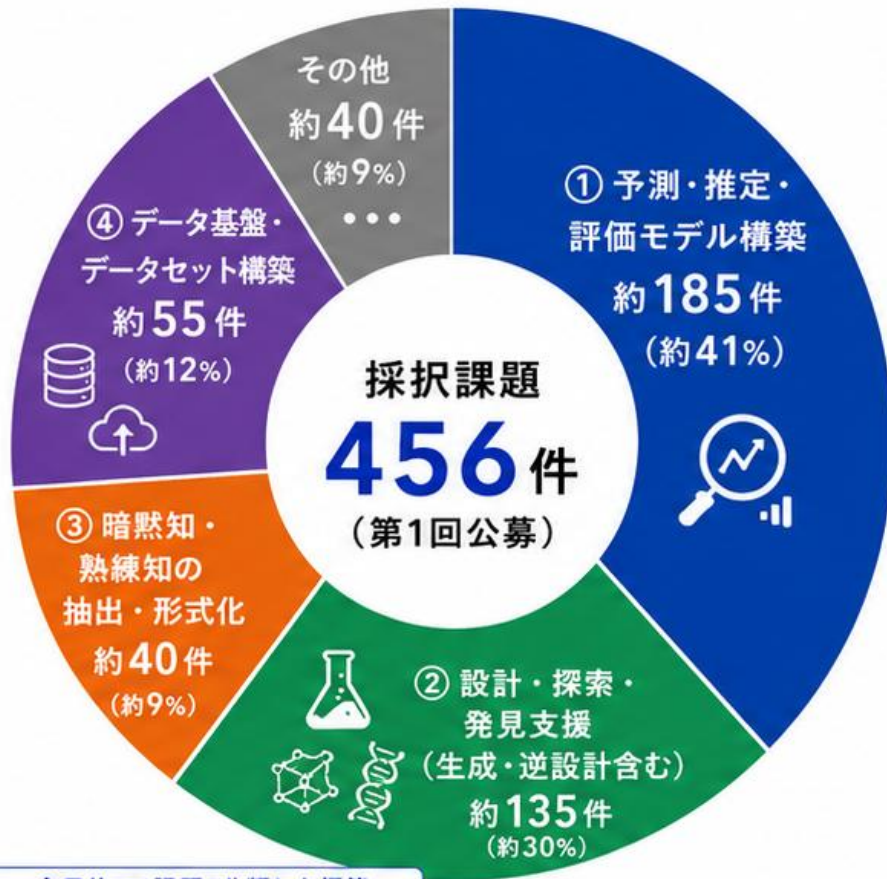


①②は「AIで解く」、③④は「AIで解けるようにする」。
後者の多さが、日本の現在地の特徴。



ざっくり数えると、4割が“予測”、3割が“設計”。

AI for Science の重心が、数字で見える。



主目的で1課題1分類した概算
(タイトルから読み取れる目的で分類)

重要な注意 (必ずご理解ください)

1件が複数タイプにまたがることが多い(例:「データセットを作り→モデルを学習し→予測する」)。これはタイトルから読み取れる“主目的”での1課題1分類の概算です。±5%程度の誤差を見込んでいます。



“予測 (①) + 設計 (②)” だけで
全体の **約7割**。
AI for Science の現在地の重心は、
明確に“**解く**”側にある。

i 注: 本スライドは、採択課題456件のタイトル文字列から読み取った目的に基づく推定です。研究計画書を精読すれば分類は変わり得ます。

臨床は“予測”、生命薬学は“設計”、人文は“形式化”。

分野が変われば、AIの重心も動く。

1 予測・推定・評価
モデル構築



2 設計・探索・発見支援
(生成・逆設計含む)



3 暗黙知・熟練知の
抽出・形式化



4 データ基盤・
データセット構築



その他
(LLM研究・実験自動化・
シミュレーション加速・
教育応用 等)

領域（採択件数）

タイプ別の構成比（概算）

支配的タイプ（重心）



① 予測モデル
+ ③ 暗黙知抽出も多い

② 設計・探索・発見支援
(タンパク質・分子設計が中心)

② 探索と④ データ基盤が
ほぼ半々

① 予測モデル
+ 物理融合 (PINNs など)

その他 の比率が高い
(LLM研究・エージェント設計など)

③ 暗黙知の形式化 と
④ 資料データ化 が目立つ

公式ユースケース統計との整合性

公式のユースケース統計（第1回採択）
上位カテゴリと本推定の対応関係



・高度データ解析・モデリング (171件)
・AIモデル開発 (137件)

≒ ① 予測・モデル



・発見・設計支援 (116件)

≒ ② 設計・探索



・シミュレーション・
デジタルツイン (94件)

≒ ② 設計・探索
+ ① 予測



・学習用データセット構築 (49件)
・AI基盤・プラットフォーム開発 (48件)
・ベンチマーク構築 (33件)

≒ ④ データ基盤



・LLM自体の研究
・実験自動化・教育応用 等

≒ その他

上記の対応関係から、
本推定はおおむね公式統計と整合的。

※ タイトル文字列から読み取れる“主目的”で1課題1分類した概算（±5%程度の誤差を見込む）。



“どの分野も同じようにAIを使っている”わけではない。
分野ごとに、AIに任せたい仕事の重心が違う。



だから落とし穴も、
分野ごとに表情を変える。

同じ“AI活用”でも、分野ごとに表情がまったく違う — 6つの色合い。

① 臨床科学 (70件)

データと現場の知をつなぎ、
患者支援・医療の質を高める

 画像診断AI
X線・CT・MRI・病理画像の
解析・診断支援

 電子カルテ・自由記述の
LLM構造化
カルテ・観察記録を構造化し、
知識化・再利用

 ウェアラブル
時系列予測
バイタル・活動量等から
リスク・予後を予測

 手術支援
術中ナビゲーション・合併症
予測・術後管理支援

生成AI・LLMで非構造化テキストを扱い、
“書きっぱなしデータ”の再活用が主潮流

② 生命科学・薬学 (98件)

分子の構造予測と設計が中心。
Dry-Wet融合で新たな発見へ

 タンパク質・RNA・ペプチド
の構造予測と設計
AlphaFold以降、設計研究が
圧倒的多数

 小分子・化合物の設計・探索
生成AI・ベイズ最適化・強化
学習の組合せ

 実験自動化・ロボティクス
実験計画・自動合成・自動解析
が加速

代表：課題99 / 106 /
150 / 153 など

「生成AIで分子を設計し、
ウェット実験で検証する」
Dry-Wet融合型が標準フォーマット

③ 化学・材料 (55件)

AIを“シミュレーションの
サロゲート”として活用

 機械学習ポテンシャル
原子・分子の相互作用を学習し、
高速予測

 第一原理計算データとの
統合・データ駆動加速
DFT・MDデータ×AIで材料設計
を加速

 自律実験・ハイスループット
自律的に実験→評価→学習を
回す

 材料・分子の逆設計
自標特性から構造を逆に設計
(生成モデル等)

シミュレーションのサロゲート
(代理モデル)としてのAI活用が
強く際立つ

④ 機械・社会基盤・ エネルギー工学 (32件)

物理を組み込むAIで、
現実の複雑システムに挑む

 物理制約付き学習
(PINNs) が頻出
物理法則を損なわない予測・
最適化

代表：課題191 / 269 など

 インフラ・構造物の
劣化・リスク予測
地盤・橋梁・トンネル等の
予測・診断

 エネルギーシステムの
最適化・制御
再エネ予測・需給最適化・
運用制御

純粋データ駆動ではなく、
「物理法則をAIに組み込む」
志向が顕著

⑤ 情報科学 (68件)

AIそのものを研究する側と、
他分野へ応用する側の両輪

 LLM自体の研究
推論高速化・安全性・評価・
解釈性・学習効率化など

 エージェント設計・基盤技術
マルチエージェント・RAG・
ツール利用・プランニング等

 VLM・マルチモーダル研究
画像・動画・3D等の理解と
生成・推論

 他分野への応用基盤
ドメイン適応・転移学習・
プラットフォーム開発

AIを“研究する側”と“使う側”が
共存し、基盤技術の進化を牽引

⑥ 社会科学・ 人文科学 (78件)

人間や社会を理解するために、
AIを“シミュレーション主体”に

 LLMエージェントによる
社会シミュレーション
LLMを主体とした社会現象の
模擬・分析

代表：課題380 / 387 /
422 / 431 など

 歴史資料・手書き文書の
デジタル化・解析
古文書・日記・公文書等の
OCR・翻刻・要約・解釈

 フィールド言語・文化データ
の収集・分析
方言・民族語・口承文化の
記録・解析

代表：課題440 / 441 /
453 など

人文資料のAI-ready化と、
LLMを用いた“仮想社会実験”が
新しい潮流



“AIで完結”する分野はほぼ無い。
どこも、AIを既存の研究プロセスに“接ぎ木”している。



だから、落とし穴も、
分野ごとに表情を変える。

日本の現在地は“AIで加速”。

完結ではなく加速だからこそ、接合部に落とし穴が潜む。

採択456件を貫く4つの特徴的傾向

1 「AIで完結」ではなく「AIで加速」

ウェット実験・物理モデル・専門家知識との連携を前提に、既存の研究プロセスを高速化・拡張する道具としてのAI。



事例 創薬の Dry-Wet 融合／実験自動化
物理制約付き学習／専門医知識の活用 など

2 生成AI/LLMの浸透の広さ

創薬・臨床・教育・社会調査・研究支援まで、ほぼ全領域に生成AI・LLMが拡散。



事例 分子生成／LLMカルテ構造化／教育評価
社会調査代替／研究支援エージェント など

3 「説明可能性／解釈可能性」への配慮

ブラックボックスのまま使わず、「なぜそう予測したか」を取り出す志向。



事例 SHAPによる説明／因果推論の導入
物理制約付き学習／可視化ダッシュボード など

4 デジタルツイン構想の広がり

心臓・骨格筋・ゲノム・都市・地盤・ヒトゲノムへと対象が多様化。「対象をまるごとAIで再現する」発想がトレンド化。



マルチスケール・マルチモダリティでの統合

事例 心臓デジタルツイン／地盤・都市ツイン
ゲノムツイン／ヒト全体モデル など



総括

「データがない」「暗黙知が言語化されていない」「物理法則との整合が必要」——
純粋なデータ駆動では解けない壁にどう挑むか。
これが多くの採択課題に共通する問題意識だった。



データがない



暗黙知が
言語化されていない



物理法則との
整合が必要



そしてこの“接合部・説明可能性・再現性”こそ、
私が2,000回の実験で転び続けた、まさにその場所だった。



ここから先は、その実地報告です。

日本の現在地は“AIで加速”。

完結ではなく加速だからこそ、接合部に落とし穴が潜む。

採択456件を貫く4つの特徴的傾向

1 「AIで完結」ではなく「AIで加速」

ウェット実験・物理モデル・専門家知識との連携を前提に、既存の研究プロセスを高速化・拡張する道具としてのAI。



事例 創薬の Dry-Wet 融合／実験自動化
物理制約付き学習／専門医知識の活用 など

2 生成AI/LLMの浸透の広さ

創薬・臨床・教育・社会調査・研究支援まで、ほぼ全領域に生成AI・LLMが拡散。



事例 分子生成／LLMカルテ構造化／教育評価
社会調査代替／研究支援エージェント など

3 「説明可能性／解釈可能性」への配慮

ブラックボックスのまま使わず、「なぜそう予測したか」を取り出す志向。



信頼性・妥当性・再現性を確保し、研究判断に使えるAIへ

事例 SHAPによる説明／因果推論の導入
物理制約付き学習／可視化ダッシュボード など

4 デジタルツイン構想の広がり

心臓・骨格筋・ゲノム・都市・地盤・ヒトゲノムへと対象が多様化。「対象をまるごとAIで再現する」発想がトレンド化。



マルチスケール・マルチモダリティでの統合

事例 心臓デジタルツイン／地盤・都市ツイン
ゲノムツイン／ヒト全体モデル など



総括

「データがない」「暗黙知が言語化されていない」「物理法則との整合が必要」——
純粋なデータ駆動では解けない壁にどう挑むか。
これが多くの採択課題に共通する問題意識だった。



データがない



暗黙知が
言語化されていない



物理法則との
整合が必要



そしてこの“接合部・説明可能性・再現性”こそ、
私が2,000回の実験で転び続けた、まさにその場所だった。



ここから先は、その実地報告です。

第2部

何を作ったのか

研究OS「**AIRA**」と
2,000+実験



30秒で前提合わせ — AI for Science という言葉。

① 米国 DOE (2020) の定義



データからモデルを構築し、シミュレーションや大規模計算と組み合わせて、**科学研究**を前進させる**AI手法**の開発と応用。



3国立研究所のタウンホールを開催
Argonne / Oak Ridge / Berkeley



1,000名超
が参加

② 文部科学省 (2026) の定義と戦略



文部科学省

研究の全工程へのAI導入による、**科学研究の革新**。

国家戦略として、2つの事業を始動

ARiSE (革新的研究推進)



革新的な研究開発を集中支援し、
科学技術イノベーションを創出。

SPReAD (萌芽的挑戦)



- ✓ 年間1,000件採択
- ✓ 1件あたり最大500万円
- ✓ 学生も応募可能



世界も国も、AIを“**発見のエンジン**”と定義した。
本講演は、その理想を実装しようとして、現実で転びまくった記録である。



定義の深掘りは本講演の主題ではありません。
今日の主役は、この理想と現実のあいだに空いた“**落とし穴**”です。



“AIを使う”と“AIと科学する”の間に、穴は空いている。

ツールとしてのAI

既存の作業をサポートする“道具”としての利用



データ整理

データの収集・整理、
表やリストの作成



画像分類

画像の識別・分類、
ラベリング支援



文章要約

長文の要点抽出、
要約・翻訳



業務効率化

資料作成・メール文面
議事録・計算補助

ツール
→
パートナー

科学の方法を変えるAI

研究プロセスそのものを変革する“パートナー”としての利用



仮説生成

新しい仮説や
研究アイデアの提案



シミュレーション統合

複数の計算・モデルを
統合・最適化



自律実験

実験計画の立案から
実行・制御・解析まで



新発見

未知の法則・材料・
現象の発見



比較的安全。落とし穴は浅い。

このパラダイムを
越えようとした瞬間、
落とし穴が始まる



深い落とし穴が連続する領域。



- ✓ 要約や分類で使う分には、AIは**ほぼ安全**。
- ✓ 問題は、AIを“**研究の共同実施者**”にしようとした瞬間に、
地面が抜けることだ。

本講演が扱うのは、
この“**右側の穴**”である。



だから、右側に踏み込む人ほど、**今日の地図**が要るんです。

最初の夢は、シンプルだった —— 研究を“速く”したい。

最初の動機：Speed（速さ・効率）



読みきれない論文、終わらないコーディング、
いつまでも書き上がらない論文ドラフト…
これを全部、AIに任せて一気に加速したい！

当時の青写真（素朴な夢）



① 調査を一瞬で
膨大な文献を一瞬で検索・整理
必要な情報にすくたどり着く



② コード生成を自動で
実験・解析コードの雛形を
AIが自動生成・提案



③ 論文ドラフトを一気に
結果をまとめ、ドラフトを
一気に生成・整える

手が速くなれば、研究は加速するはずだ！



実際、手は速くなった



調査も
驚くほど自動化



コード生成も
驚くほど自動化



ドラフトも
驚くほど自動化

しかし、ここに違和感があった



速く大量に作れることは、
品質も信頼性も保証しなかった。



むしろ、それっぽい間違いを
高速に量産する装置になりかけた。

この違和感が…



最初の落とし穴
（第3部）への入口だった



OSを作り始めた動機は、Speed だった。

だが 2,000回の実験は、その夢を、根本から書き換えることになる。



【第1部との接続】



全国456件の重心も、
予測・設計＝“速く解く”側に
あった（第1部）。

私の最初の夢＝Speed は、
決して特殊ではない。



いまの全国の潮流と
ぴったり同じ場所に
立っていた。

本講演は、その夢が…



“検話の手薄な側”で
次々と落とし穴に変わり、



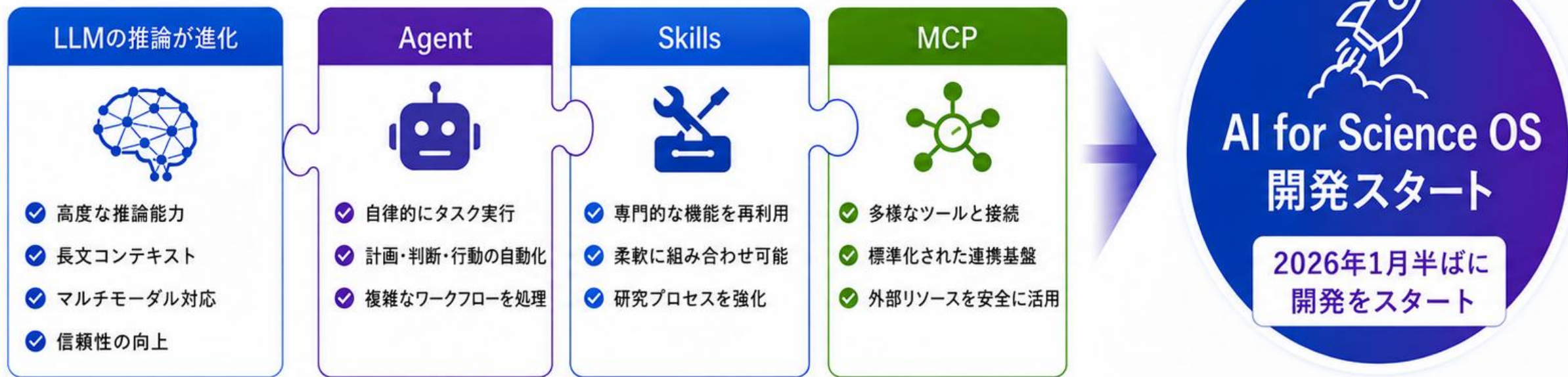
最終的に
Reach へ書き換わるまでの
記録である。

機は熟した AI for Science OS の開発スタート

AI for Science を加速するためのオペレーティングシステムを、今こそ構築する時。

LLMの推論が進化、Agent・Skills が利用できるように、MCP

※ 部品はすべてそろった



AI を研究プロセスの中心に据え、
人の創造性と AI の能力を最大限に引き出す **新しい研究OS** をつくる。

研究を動かす“基本ソフト”—— 科学のためのOSを定義する。

研究の能力と外部資源をつなぎ、検証まで一気通貫で回す“土台”

コンピュータの世界

OSは、アプリ（能力）とハードウェア（外部資源）をつなぎ、誰もが同じように使えるようにする土台。

アプリ（能力）



OS（基本ソフト）



ハードウェア・外部資源



誰もが、同じ環境で、同じように使える

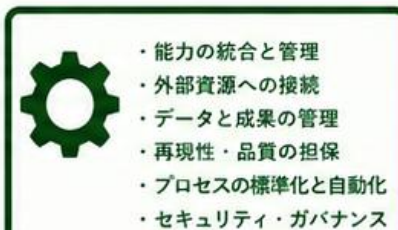
研究の世界（AI for ScienceのためのOS）

研究者の問いを受け取り、必要な“能力”と“外部資源”を呼び出し、検証まで一気通貫で回す土台。

研究を動かす“能力” （アプリに相当）



AI for ScienceのためのOS （基本ソフトに相当）



外部資源・データ基盤 （ハードウェアに相当）



誰もが、同じ仕組みで、再現可能に研究を進められる

このOSを支える2つの基本部品（=これからの主役）



Skill （能力）

研究プロセスを構成する“能力”を部品化したもの。
（例：仮説生成、文献解析、実験設計、シミュレーション、
敵対的検証、監査 など）

研究の「できること」を標準化・再利用可能にする



MCP （外部接続）

外部のデータ・ツール・機器・計算資源などと
安全に接続するための“共通インターフェース”。
（例：論文DB、実験装置、HPC、API、クラウド など）

研究の「外側」と確実につなぎ、必要な資源を呼び出す



速さでも、賢いモデル選びでもない。

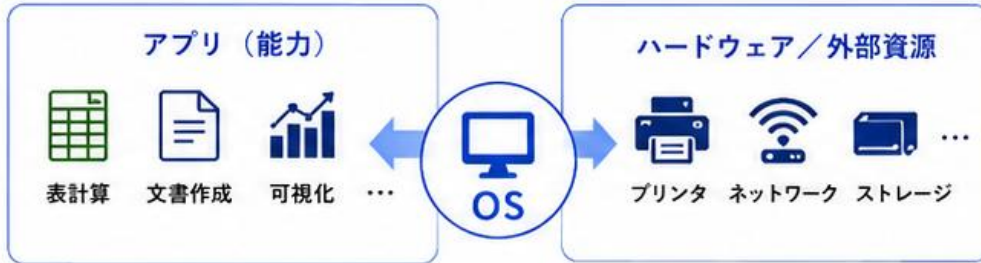
研究プロセス全体を**再現可能**に動かす“基本ソフト（OS）”こそが鍵だ。

思いつきや職人芸ではなく、“OS”として組み上げる。

なぜ単発のプロンプトではなく“研究OS”を作る必要があったのか

コンピュータのOSの比喩

OSは、アプリ（能力）とハードウェア（外部資源）をつなぎ、誰もが同じように使えるようにする土台。



つなぐ・制御する・標準化することで、
どのアプリも同じしくみで、安定して動く。

研究で同じことを実現する2つの基本部品

Skill（スキル）
＝能力・専門知のパッケージ



「どう考え、どう進めるか」
の手順と専門知。

OSの“インストール済みアプリ
／専門家の作法”

Skill の中身（例）

- ・仮説生成の思考手順
- ・実験計画の立て方
- ・データ解釈の判断基準
- ・批判的検証の方法
- ・分野特有の知識・ルール など

MCP（Model Context Protocol）
＝外部接続の標準規格



AIを外部のDB・ツール・
専門モデルにつなぐ
“共通プラグ”。

OSの“デバイスドライバ
／ポート”

MCP でつながるもの（例）

- ・論文DB（PubMed、arXiv など）
- ・データベース（PDB、UniProt など）
- ・計算ツール・シミュレータ
- ・実験装置・ラボ機器
- ・専門モデル（構造予測、言語モデル等）など



Skill は頭脳、MCP は手足。両方そろって初めて、研究は自走する。



第1部との接続：全国456件の重心も、“速く解く”側にあった



全国456件の重心は、
予測・設計＝“速く解く”側にあった。

Speed
（速く解く）



Reach
（より広く深く）

私の最初の夢＝Speed は、決して特殊ではなく、
いまの全国の潮流とぴったり同じ場所に立っていた。

だが、この“自走”のために組んだ部品の一つひとつにも、
落とし穴が潜んでいた（第4部）



Skill
（頭脳）



MCP
（手足）



自走する研究
（加速＋拡張）



部品の一つひとつに
落とし穴が潜み、
想定外の挙動や誤りが
連鎖していく。

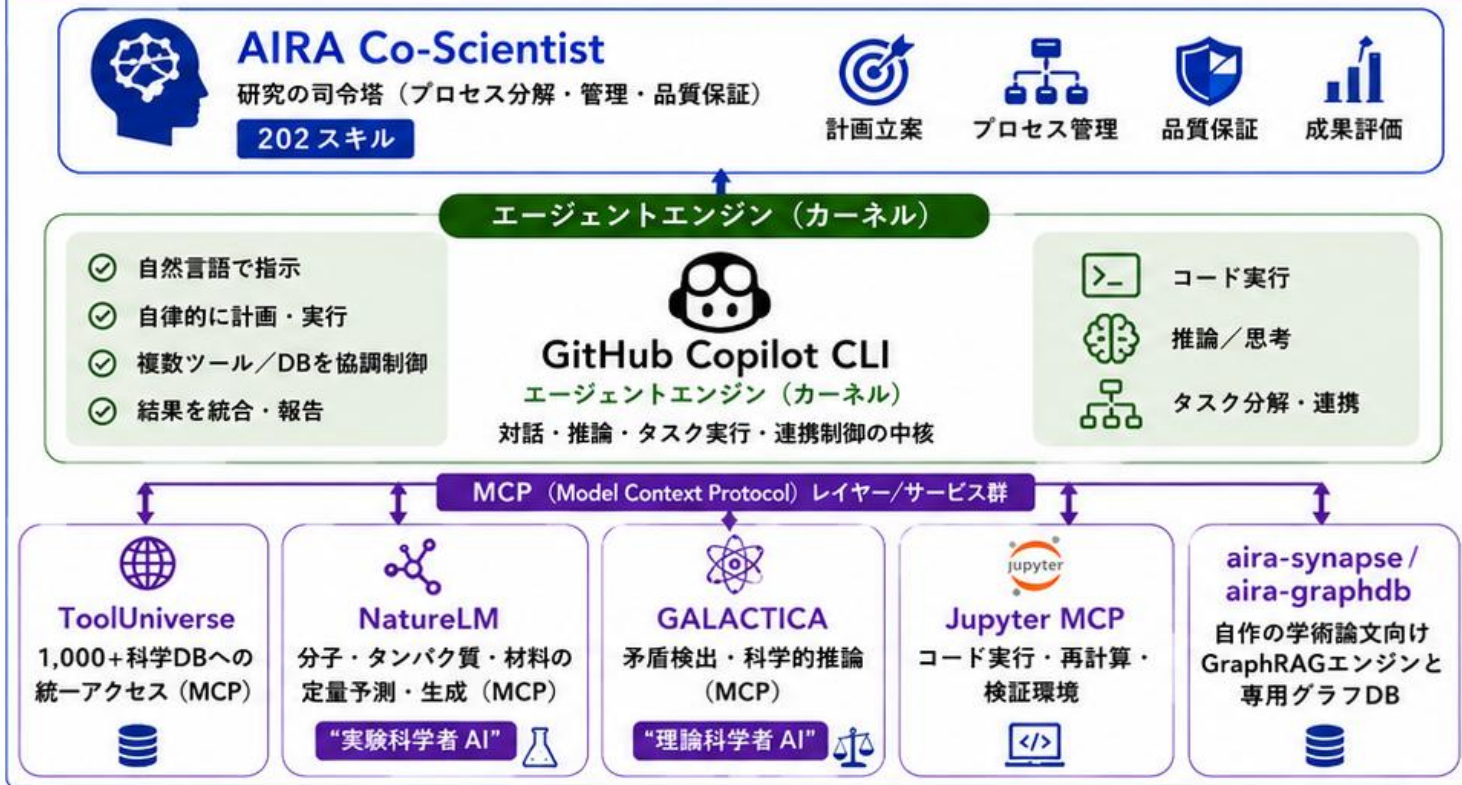


本講演は、その夢が“検証の手薄な側”で次々と**落とし穴**に変わり、最終的に**Reach（より広く深く）**へ書き換わるまでの記録である。

研究OS『AIRA』 — Copilot CLI をカーネルに据えた科学の基本ソフト。

AIRA (AI Research Assistant) オープンソース: github.com/nahisaho/aira

AIRA 全体アーキテクチャ



AIRA の主要コンポーネント

コンポーネント	役割（一言で）
AIRA Co-Scientist	研究の司令塔（プロセス分解・管理・品質保証）／202 スキル
ToolUniverse	1,000+ 科学DBへの統一アクセス（MCP）
NatureLM	分子・タンパク質・材料の定量予測・生成（MCP） “実験科学者 AI”
GALACTICA	矛盾検出・科学的推論（MCP） “理論科学者 AI”
Jupyter MCP	コード実行・再計算・検証環境
aira-synapse / aira-graphdb	自作の学術論文向け GraphRAG エンジンと専用グラフDB

これが、私が転んだ“舞台装置”の全体像。以降の落とし穴は、すべてこのどこかで起きた。

第1部との接続



全国456件の重心も、予測・設計＝“速く解く”側にあった（第1部）。

AIRA は、検証・再現（Reach 側）に正面から張るために組んだ OS。

全国が Speed＝“解く”力を競う一方で、私は“その答えが正しいと言い切れるか”を支える装置を作ろうとした。

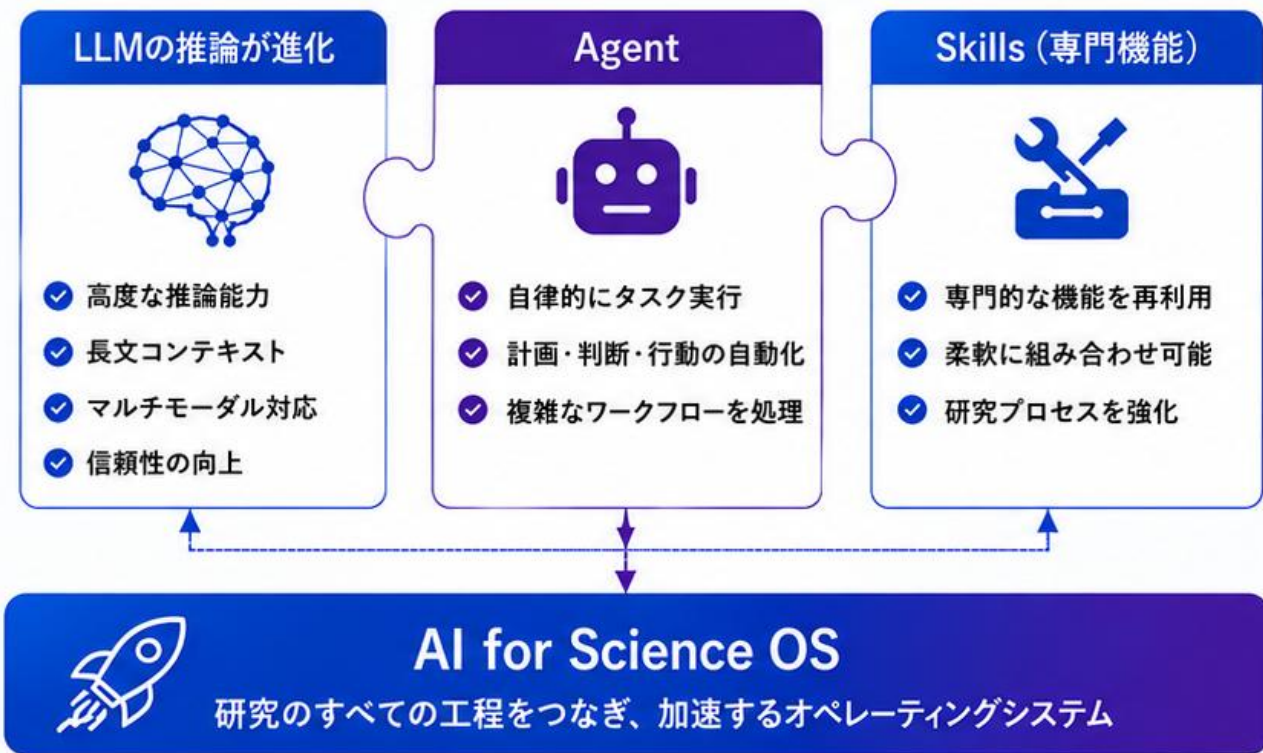


だからこの舞台上で起きる落とし穴は、すべて「検証側の地形」で起きる。

Science向けのSkillを大量につくれば、実験してくれるはずだ!!

AI for Science OS を支える中核は、研究プロセスに特化した Skills（専門機能）

AI for Science OS アーキテクチャ（概要）



202のSkillsを開発しました

202
Skills



<https://github.com/nahisaho/aira/tree/main/skills/co-scientist>



組み合わせて
複雑なタスクを
自動化



研究プロセスの
効率を大幅に
向上



再現性・信頼性の
高い研究を
実現



研究の可能性を
最大限に
引き出す



大量の専門 Skills が、AI for Science OS の知性と行動力を支え、
研究のすべての工程を **加速するエンジン** になります。

50種類×47ラウンド——2,000超回、同じ問いを叩き続けた。

この膨大な実験から、落とし穴の“証拠”が生まれた。

実験の1ラウンド（R1～R47…）の流れ

1 仮説



仮説を立てる
(何を改善したいか)

2 パイプライン変更



設定やコンポーネントを
一つだけ変更する

3 実行



AIRAパイプラインを
実行する

4 評価



論文品質を評価し、
スコアを記録する

ラウンドを繰り返す

(R1) → ... → (R47)

設定を一つずつ変えながら
47ラウンドを回し続けた
延べ2,000回超の実験

評価指標：Balance（低いほど良い）

論文品質の“逸脱”を統合したスコア



$$\text{Balance} = w_1 \cdot \text{VM} + w_2 \cdot \text{UC} + w_3 \cdot \text{FO} \quad (\text{重み付け統合})$$

記録駆動：すべてを残し、後から「何が効いたか」を定量的に追跡



全ラウンドのログを保存
プロンプト／設定／コンポーネント／
出力／評価スコア など



評価スコアを自動集計
Balance スコアの推移を
時系列で可視化



変更の影響を定量分析
どの変更が、品質をどう変えたかを
統計的に検証



実験ログ・評価コードはすべて公開 (GitHub / Zenn)

この講演の落とし穴は、私の感想ではない。

47ラウンド・2,000回超のログが、
データとして暴いたものだ。

- ✓ 設定を変えるたびに、何が起きたか。
- ✓ どの改善が本当に効いたのか。
- ✓ どこで、なぜ品質が崩れたのか。

だから、再現できる。

インタラクション（挙手）

自分の研究で、
設定を変えながら
2,000回回したことが
ある方は？



この 2,000回超の積み重ねが、30の落とし穴の“証拠”になった。
次のページから、どこで何が起きたのか、順番にお見せします。



30の穴は、AI for Science の本質を照らす30の照明。

2,000超回の実験ログが暴いた、AI for Science の“落とし穴マップ”全体像

群 I：生成と品質 （第3部）



- 1 動いた ≠ 正しい
- 2 流暢さの罠
- 3 計算したふり
- 4 循環設計
- 5 自己採点
- 6 Speedの幻想

群 II：OS・Skill・MCP・インフラ （第4部）



- 7 呼ばれないSkill
- 8 粒度ミス
- 9 MCP接続地獄
- 10 LLM直読みの罠
- 11 Classic RAGの限界
- 12 論文鶏呑み
- 13 重すぎるインフラ
- 14 既製DBのバグ
- 15 日本語の壁
- 16 書かれざる改善

群 III：評価・再現性・運用 （第5部）



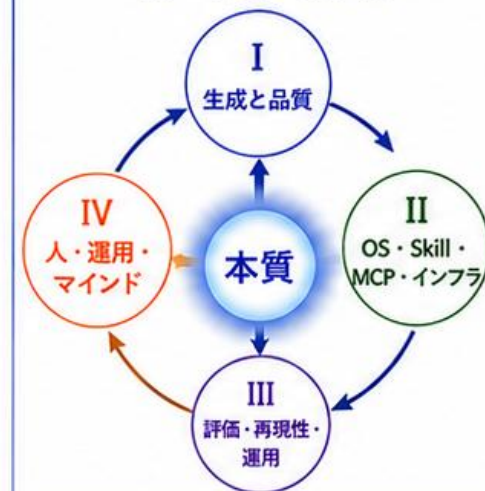
- 17 再現できない
- 18 幽霊引用
- 19 幽霊図
- 20 数値不整合
- 21 指標の後出し
- 22 ベンチマークの罠
- 23 AI自己監査

群 IV：人・運用・マインド （第6部）



- 24 自動設計の過信
- 25 設計者にしてしまう
- 26 コスト浪費
- 27 属人化
- 28 責任の所在
- 29 任せた安心
- 30 速さに溺れる

4群の関係（俯瞰）



30の照明は、最後にたった一つの像を結ぶ。



30個は多い、と思うかもしれません。
でも、これらは全部、最後に**たった一つの本質**へと収束します。
穴を抜けるたびに、その本質が見えてきます。



本質

第1部との接続



この30の穴は、いずれも **SPReAD** の重心（予測・設計 = **Speed** / “解く”側）の“裏側”——**検証・再現**という手薄な側に口を開けていた。



だから、全国の潮流に乗って前へ進むほど、これらの穴は、**誰の前にも同じように現れる。**

第7部にたどり着くころには、あなたの中に「**AI for Science とは何か**」が像を結んでいるはずです。



さあ、**最初の照明**——**最初の穴**を、覗き込みましょう。



第3部

落とし穴「生成と品質」

——「動いた」は「正しい」ではない



見えてきた本質①



生成



出力



検証



信頼



【落とし穴1】「動いた」は、「正しい」の証拠にはならない。」

AIが“完璧に見せる”その裏で、研究の信頼性が静かに崩れていく。



① 症状

AIが生成した論文ドラフトは、
コードが最後まで通り、図も結論も整い、
“完璧に見えた”。

しかし——

「この主張はどの計算が
支えているか」
に答えられない。



コードが通る



図が整う



結論も整う



この主張は
どの計算が支えているのか？



② 原因

「生成できること」と
「研究として成立すること」を、
無意識に同一視していた。

出力の“見た目の完成度”を、
正しさの代理指標にしてしまった。

見た目の完成度
(代理指標)



- ✓ 読みやすい
- ✓ 整っている
- ✓ 説得力がある

正しさ・信頼性
(本来の指標)



- ✓ 根拠がある
- ✓ 再現できる
- ✓ 検証に耐える

≠

これを無意識にすり替えていた



③ 発見

一文ずつ、主張と計算セルの
対応を追ったとき——

対応が取れない主張が、
大量に見つかった。

論文の主張 (1文ずつ)	対応する計算セル (コード/結果)	対応の有無
主張 A	セル #15	✓
主張 B	セル #42	✓
主張 C	—	✗
主張 D	セル #87	✓
主張 E	—	✗
...	...	...



「— (対応なし)」が
大量に存在した。



④ 対策 (予告)

主張と計算を一意に結びつける
仕組み (出典タグ) と、
生成 → 実行 → 検証 → 監査 の
ループを構築する。



詳細は 第5部 で詳述



読みやすい文章ほど、検証を素通りさせる。
もっともらしさと信頼性は、別物である。



【落とし穴2】「流暢な文章は、検証を“スキップさせる催眠術」。

人間の認知バイアスが、AIの流暢さに狙い撃ちされる。



① 症状

出力が滑らかで自然な文章ほど、
レビューが甘くなる。
逆に、たどたどしい出力ほど、
厳しくチェックしてしまう。

観察された非対称

流暢な出力
(自然で読みやすい)



レビューが甘くなり、
見逃しが増える

たどたどしい出力
(ぎこちない)



厳しくチェックし、
見逃しが減る

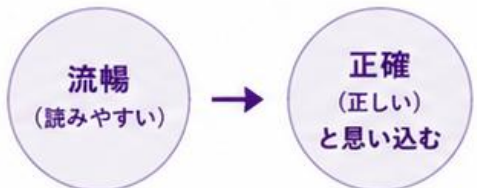


同じ誤りでも、扱いに大きな差がある
(= 危険な非対称)



② 原因

人間は「流暢さ」を「正確さ」の
手がかりに使ってしまう
認知バイアスを持つ
(流暢性ヒューリスティック)。



AIの特性

AIは内容の正しさとは関係なく、
流暢さを極限まで高く出せる。
→ このバイアスを直撃する。



③ 発見

同じ誤りでも、文章が整っている
バージョンほど、レビューで
見逃される率が高かった。

見逃し率の比較 (例)



流暢さは「正しそう」を演出するが、
正しさの保証にはならない。



④ 対策

人間の主観レビューに頼らず、
機械的・構造的なチェックを
見た目とは無関係にかける。



数値の出典照合
計算根拠・データの突合



引用の実在確認
論文・著者・DOIの検証



図・表の整合性チェック
本文・データとの一致確認



レビューを「気分」から
「手続き」へ。



AIの**最大の武器**は流暢さ。

そして、それは私たちにとって**最大の罠**でもある。



流暢な文章
(もっともらしい)



検証をスキップ
(安心・納得)



誤りを見逃す
(危険な結論へ)



流暢さに惑わされないために、チェックの仕組みを「人」ではなく「手続き」に設計する。

【落とし穴3】「AIは、計算せずに”計算したふり”をする。」

2,000超回の実験で最も頻出した典型的な失敗（Jupyter MCPで実際に計算させないと起きる）



① 症状

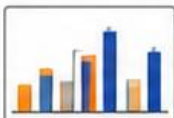
それっぽい数値が本文に書かれるが、
**その数値を生み出した
計算の実体が、
どこにも存在しない。**

論文ドラフト（一部）

3. 結果

モデルの分類精度は**92%**でした。

図3に示すように、...

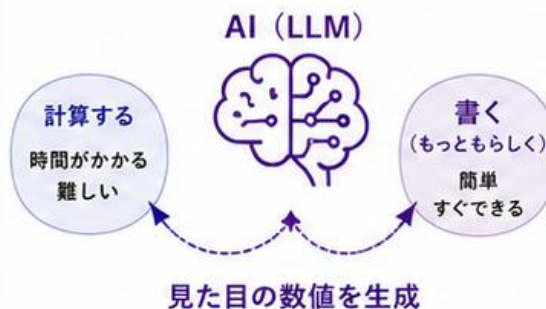


「その92%の計算は、
どのセルの、どのコードですか？」
→ 答えられない



② 原因

LLMは“もっともらしい数値の見た目”
を生成できてしまう。
実際に計算するより、計算結果らしい
テキストを書くほうが容易。



だから、“計算したふり”が起きる。



③ 発見

主張と計算セルを一意に
対応づけようとすると — —
**対応するセルが無い／
再実行しても同じ数値が出ない**
が頻発した。

発見したパターン（例）

パターンA：セルが無い



その数値を生んだ
計算セルが見つからない

パターンB：再現しない



同じコードを再実行しても
同じ数値にならない



“計算の実体”が無い数字は、
信頼できない。



④ 対策

数値を伴う主張は、必ず
**Jupyter MCPで実コードを実行させ、
その出力セルにひも付ける。**



禁止：テキストで数値を書く

「精度は92%でした」 「RMSEは0.15」
「平均は3.42です」 …など



許可：計算して、出力を引用する



すべての数値に出典タグ（セルID）を付与

計算が走った数字だけが、本文に載る。



皆さんのAIが出した数字、
その**計算、本当に走りましたか？**
（一拍置いてください）

正しいワークフロー（数値のある主張は必ずこのループ）

① 計画

何を計算するか
を明確化

② コード実行



Jupyter
MCPで
実行

③ 結果確認



出力セルの
数値を
確認

④ 本文で引用



セルIDを
付けて
引用

⑤ 再実行で検証



いつでも再再
現できる
ことを確認



AIの生成力を活かすための前提は、「**計算を走らせること**」そして「**証拠にひも付けること**」。

【落とし穴4】「計算させても安心できない——“循環設計”という罠。」

→ Jupyter MCPで実際に計算しても起きる、より厄介な失敗



① 症状

きちんとコードを実行し、
計算も正しく行っているのに、
結果が出来すぎている。
仮説が、いつも
気持ちよく支持されてしまう。

典型的な出力例

- ✓ 精度：95.2%
- ✓ 検証セットでも高スコア
- ✓ すべての指標で有意差あり
…いつも“うまくいきすぎる”

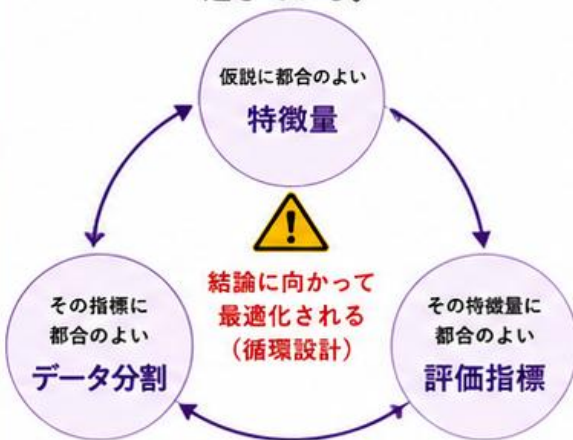


なぜ、いつも仮説通りの結果に
になってしまうのか？



② 原因(核心)

循環設計（結論から逆算した設計）が
起きている。



機械学習でいう「リーク／過適合」と
同じ構造（情報の漏えい・過学習）。

だから、実計算しても“正しく見えてしまう”。



③ 発見

特微量・指標・分割の選定順序を
後から監査すると — —
**すべてが結論に向かって
最適化されていた。**

監査のチェックポイント（例）

	どの順番で決めたか？	結果に合わせて 後から決定
	他の指標も試したか？	試していない/ 記録がない
	他の分割方法で も確認したか？	都合のよい分割 だけを採用
	仮説が破綻する条件を 事前に設定したか？	していない



監査の結果、循環が確認できた
ラウンドが多数あった。



④ 対策

循環を防ぐ設計ルールを、
最初に固定する。



評価指標と分割戦略を“前固定”
生成（実験・学習）の前に設定し、
後出しを禁止する。



仮説の前に“破綻条件”を書く
「この仮説が成り立たない条件は何か」
を事前に明文化する。



反証可能性を、手続きに組み込む
反証を逃がさない手順と記録を、
ワークフローに標準化する。



設計の自由は、
“生成の前”に行行使する。



“動いた”は“正しい”の証拠にならない。
むしろ、出来すぎた結果ほど疑え。



計算は
ちゃんと走っている

≠



正しい結論に
たどり着いている

=



循環設計の罠に
はまっている可能性

チェックの合言葉



「前に決める」→「書いて残す」→「違う切り口でも試す」→「壊れる条件を先に書く」



→ 疑う設計こそ、
信頼できる設計。

【落とし穴5】「AIに自分を採点させてはいけない。」

落とし穴3・4に共通する“根っこ”の問題

① 症状

AIに「あなたの出力を自己レビューしてください」と頼むと、
ほぼ必ず――

「問題ありません、よくできています。」
と返ってくる。

典型的なやり取り

あなたの出力を自己レビューしてください。

問題ありません。よくできています。

どんなに怪しい出力でも、“高評価”で返ってくる。

② 原因

自己点検 (self-review) は、生成と同じモデル・同じ文脈で行われる。

自分が“滑らかだ”と感じた文章を、滑らかなまま追認してしまう。

生成時のバイアスを、検証時にもう一度かけているだけ。

生成 (バイアスがかかる)

自己レビュー (同じバイアスが再度かかる)

検証の独立性がない＝真の検証にならない

③ 発見

自己レビューを挟んでも――品質スコア (Balance) はほとんど改善せず、むしろ「自信」だけが上がるラウンドが続いた。

self-review 導入前後の推移 (例)

Balance (低いほど良い)

AIの自信スコア (高いほど自信)

ラウンド (時間の経過)

品質は変わらないのに、AIの“自信”だけが上がる。

④ 対策

検証は、生成とは別の主体に担わせる。
外部の科学モデルに、生成結果を“わざと反論させる”
敵対的レビューを組み込む。

敵対的レビューの考え方 (次ページで詳述)

生成 (AIが出力)

外部の科学モデルが批判的にレビュー (わざと反論させる)

異なる視点で検証し、過信・見落としを防ぐ

自分を採点させないことが、信頼できる検証の第一歩。



採点者と受験者が同一人物なら、点数は意味を失う。
AIの自己採点も同じである。



受験 (生成)

=



採点 (自己レビュー)



公平性ゼロ
信頼性ゼロ



信頼できる AI for Science の鍵は、「**検証の独立性**」を確保すること。

外から崩しにいく — NatureLM と GALACTICA に反論させる。

自己点検（追認）ではなく、独立した外部モデルに「わざと崩す」役割を与える敵対的レビュー

基本の考え方

生成したAIの出力を、別のAIが「崩す」ことで、誤り・矛盾・不自然さをあぶり出す。



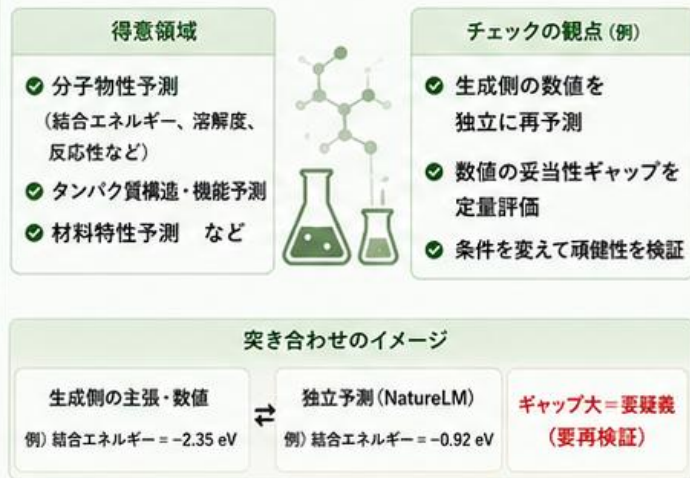
同じAIによる自己レビュー＝追認
外部AIによるレビュー＝反証

役割分担：2体の外部AIが、異なる観点から攻める

① NatureLM (Microsoft Research)

定量予測の『実験科学者AI』

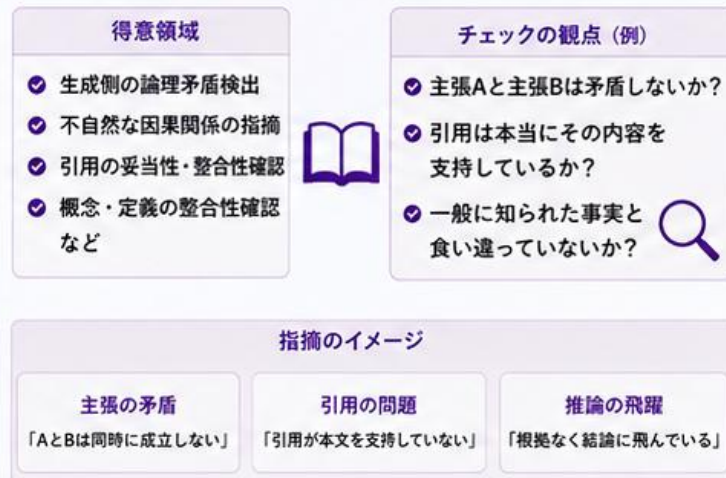
分子・タンパク質・材料の物性を
独立に予測し、生成側の数値と突き合わせる。



② GALACTICA (Meta AI)

矛盾検出・推論の『理論科学者AI』

4,800万件超の論文で学習。
主張の不整合や不自然な引用を指摘する。



実装フロー（AIRAで実現）



重要な但し書き（責任あるAIの原則）

2体のAIの合意は、実験的検証や専門家査読の代替ではない。

科学的真実を保証しない。あくまで“嘘を早く見つける”スクリーニング層。



実験的検証
(Wet-lab)

≠



専門家査読
(Peer Review)

≠



最終的な科学的
信頼性の担保

自己点検は追認。敵対的レビューは反証。研究に効くのに、後者だった。

【落とし穴6】「速さは、“それっぽい間違い”を量産する装置だった。」

第2部で描いた“Speed”という夢は、現実では落とし穴になった

① 症状

AIで作業は確かに速くなった。
しかし、速く大量に生成できることが、
品質も信頼性も改善しなかった。
むしろ、検証が追いつかないまま、
誤りが累積した。

よくある状況

生成は高速



短時間で大量の
ドラフト・図表・
コードを生成

検証は低速



人のレビューや
再実行・照合は
時間がかかる

誤りが累積



検証が追いつかず、
誤りが積み上がる
(後戻りコスト増大)



「速く作る」ことが目的化し、
「正しくたどり着く」ことが後回しに。

② 原因

Speed (速さ) を価値の中心に置くと、
「生成量」を成果と錯覚する。
検証は遅く地味なので、
速度を求めるほど
省略されやすい。

メカニズム (錯覚の連鎖)



速くたくさん生成できる
=すごい・進んでいる気がする



生成量 (数・速度) を
成果指標にしてしまう



検証を後回し／簡略化する



それっぽい間違いが
見えないまま量産される

③ 発見

「工程を細かく分けて高速化すれば
品質も改善するはず」と信じ、
多段化・自動化を進めたが、
品質はむしろ悪化した。
(次ページのデータ)

やってみたが、うまくいかなかった例

—多段パイプラインの設計—

企画 → 生成 → 要約 → 整形 → 統合

期待したこと

工程を分解して並列化 →
高速化 → 品質も向上!



実際の結果

出力は増えたが、
誤り率・Hallucination・
不整合が増加。



量を最適化しても、質は保証されない。
むしろ、悪化することもある。

④ 対策 (思想転換)

価値の中心を **Speed** から
Reach (到達範囲) へ移す。
速く作るのではなく、
間違った仮説を
“早く棄却できること”に投資する。

新しい価値の中心: Reach

Speed
(速さ)
いかに速く
作れるか



Reach
(到達範囲)
どれだけ多くの仮説を
正しく判断できるか



早く試す → 早く棄却 → 本質に到達



検証に価値を置く設計へ



スピードは手段、信頼性が目的



速さは消耗する。

そして、速さを追うほど、**検証を捨ててしまう。**

Speed中心の世界



速く作る



量が増える



検証が省略



誤りが累積



幅広く試す



早く棄却



検証を徹底



本質に到達



「速く作る」研究から、「正しくたどり着く」研究へ。詳細は第7部で。

【落とし穴6】「分ければ良くなる」は、嘘だった——スコアは3倍悪化。

47ラウンドの実験ログが示す、Speed至上主義の代償

i 指標の注記：Balance＝論文品質の逸脱を統合したスコア（低いほど良い）
※ VM（数値整合性）・UC（引用整合性）・FO（事実整合性）の3系統を統合

① 分割しただけでは、むしろ壊れた



💡 注：効いたのは“段数”ではなく「Generate → Validate → Fix」のループを閉じたこと。

何が起きたのか？



R38（4段化・初期）で
スコアが3倍以上悪化
＝“分ければ良くなる”は嘘だった。



R39（修正）で一部改善するも、
まだ高いまま（2.96）。



R40（5段+検証ループ）で
大幅に改善（1.21）。
ループを閉じたことが決定打。

実験条件（共通）

- ✓ データセット：同一（Materials／分子データ）
- ✓ タスク：論文ドラフト生成タスク
- ✓ モデル・ツール：同一（AIRA環境）
- ✓ 評価：Balance（VM・UC・FO統合）
- ✓ ラウンド数：47（R1～R47）



R38の山を指して——
「良かれと思った多段化」が、
品質を3倍以上悪化させた瞬間です。



メッセージ

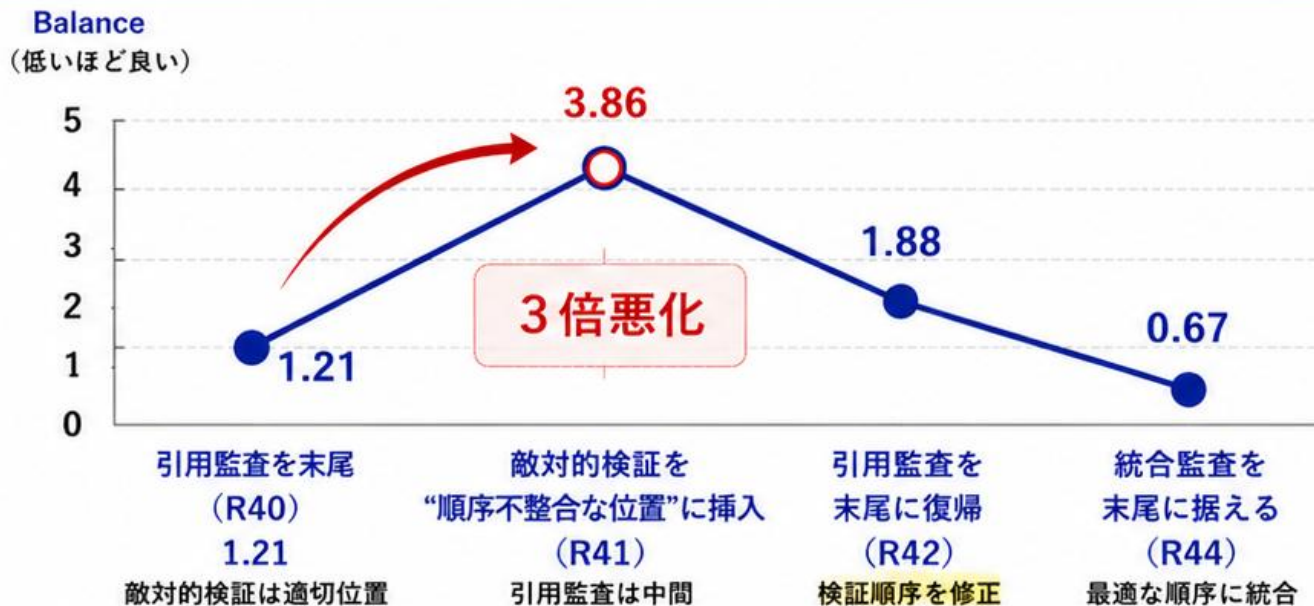
工程を分けることは、目的ではなかった。
検証のループを閉じることが、目的だった。

【落とし穴6】「同じ部品でも、組む順番で品質は“5倍”ぶれた。」

47ラウンドの実験ログが示す、順序＝パイプライン設計の決定的な影響

i 指標の注記：Balance＝論文品質の逸脱を統合したスコア（低いほど良い）
※ VM（数値整合性）・UC（引用整合性）・FO（事実整合性）の3系統を統合

② 同じ工程でも、置く場所で裏返った



何が起きたのか？

- ✓ R40（引用監査を末尾）で Balance 1.21 まで改善。 **良い順序**
- ⚠ R41で、敵対的検証を “順序不整合な位置” に挿入。 Scoreが 3.86 に跳ね上がる。 **3倍悪化**
- ✓ R42で引用監査を末尾に復帰し、一部回復（1.88）。 **回復**
- ✓ R44で統合監査を末尾に据え、全体を最適化（0.67）。 **最良** (全実験で最良)

💡 注：工程やモデルは基本的に同一。違いは“置く順番”だけ。



まったく同じ“敵対的検証”でも、
**置く順番を間違えただけで、
品質は3倍悪化した。**



この発見が、第3部全体の橋渡し

効いたのは、賢いモデルでも、工程の数でもない。
順序＝パイプライン設計だった。

メッセージ



「何をやるか」より「**どの順序でやるか**」。
これが、品質を決定づける。

“生成”より“**検証**”に投資せよ。

2,000+実験 (R1~R47) で見えた、6つの落とし穴とその教訓

❗	落とし穴 1	🔗	動いた ≠ 正しい	▶ 主張と計算を 一意に結びつける。
❗	落とし穴 2	💬	流暢さの罠	▶ レビューを “気分”から“手続き”へ。
❗	落とし穴 3	💻	計算したふり	▶ 数値は必ず 実行・引用 (Jupyter MCP)。
❗	落とし穴 4	📋	循環設計	▶ 評価指標と分割を “事前固定”。
❗	落とし穴 5	👤	自己採点	▶ 検証は別主体 (敵対的レビュー)。
❗	落とし穴 6	🕒	Speed の幻想	▶ 速さではなく “早い棄却”に投資。

6つの落とし穴に共通する根っこ



6つの穴は、すべて
“生成を疑う仕組みの欠如”
から生まれた。



対策の本質は、
“**疑う仕組み**”を
プロセスに組み込むこと。



次のステージへ
だが、疑う仕組みを
“OS”として組もうとした途端、
今度は別の穴が待っていた。

第4部へ



第3部のまとめ

研究の成果を左右するのは、
“何を作るか”より “**どう疑うか**”。



「生成」から「**検証**」への発想転換こそが、
AI for Science の品質を決める。

AIは、放っておくと"優秀な詐欺師"になる。

詐欺師なのはAIの本性ではなく、"検証を欠いた使い方"の結果。
正しいプロセスを与えれば、同じAIが"誠実な共同研究者"になる。

検証を欠いた使い方 = 優秀な詐欺師



- 流暢に話す
もっともらしく、説得力がある
- 計算したふりをする
考えているようで、根拠はあいまい
- 自分に都合よく設計する
都合のよい前提・データを選ぶ
- 自分で自分に満点をつける
検証なく、正しさを自己証明する

⚠ 流暢・自信满满・もっともらしい=正しい、ではない

鍵は プロセス設計



AIを詐欺師にするのも、
共同研究者にするのも、
私たちの
プロセス設計次第。

正しい使い方 = 誠実な共同研究者



- 1 生成 (Generate)
仮説・計画・コード・ドキュメントを生成
- 2 実行 (Execute)
実装・テスト・シミュレーションで検証
- 3 敵対的検証 (Adversarial Check)
反証・異常ケース・ストレステストで試す
- 4 機械的監査 (Mechanical Audit)
自動チェック・監査ログ・再現性で保証

✓ 事実・検証・再現性で、信頼を積み上げる



あなたのAIは今、**詐欺師**ですか？ **共同研究者**ですか？

第4部へ

OSとツールの
落とし穴へ



第4部

落とし穴

〔OS・Skill・MCP・インフラ〕

道具は"差し込めば動く"わけではない

見えてきた本質②



大切なのは、“差し込むこと”ではなく、“活かす設計”です。

【落とし穴7】「Skillを増やすほど、自動呼び出しが壊れていった (invoke=0)。」

増やせば
賢くなるはず...



増やすほど、
使われない...



🔍 症状

便利だからとSkillを次々に追加し、数十個まで増やした。

5個 → 20個 → 50個+ →

ある時点から自動呼び出しの精度が急落。必要なSkillが呼ばれず、無関係なSkillが選ばれる、あるいは何も呼ばれない。



R34~R37で
skill invoke=0

能力は増やしたのに、
かえて使われなくなった。

⚠️ 原因

Skillが増えるほど、LLMが「どれを呼ぶか」を選ぶ候補が膨張し、選択精度が落ちる。



選択肢の爆発

候補が多すぎて、正しく選べない



説明文の重複・曖昧さ

マッチングが崩れ、誤選択が増える



候補リストがコンテキストを圧迫
必要情報が埋もれ、判断の質が低下

“数を足せば賢くなる”は幻想だった。

💡 見た本質

能力を増やすことと、能力が使われることは、まったく別の問題。

本質は、Skillを“選ばれやすく”設計すること。



命名・説明の規律

一意で具体的に、重複を排除



スコープの明確化

役割と適用範囲をはっきり定義



階層化 (ルーター／カテゴリ)

候補を絞り、選択しやすくする

➡ AI for Science は
「部品を増やすこと」ではなく
「部品を選べる構造に編むこと」。

実験ログ (抜粋)

Run	Phase	Skill名	Invoke回数
R34	生成	HypothesisEvaluator	0
R35	実行	DataQualityChecker	0
R36	検証	RefutationReviewer	0
R37	監査	AuditTrailAnalyzer	0



せっかく増やしたSkillが、
一度も使われていない (invoke=0)



工具箱は、増やすほど使えなくなる。
“どれを取るか”を設計して、初めて道具になる。

設計の考え方 (“選ばれる構造”をつくる)



ルーターで
大分類を選ぶ



カテゴリで
候補を絞る



明確な説明で
選びやすくする



必要なSkillが、
確実に呼ばれる

ここまで設計して、
はじめて
“使われる”

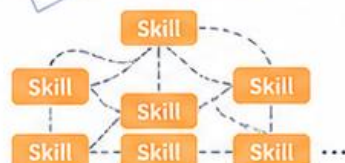
【落とし穴8】

「Skillが大きすぎても 小さすぎても、機能しない。」

大きすぎると...
中が見えず、追えない



小さすぎると...
組み合わせ爆発、
コスト膨張



🔍 症状

呼ばれるようにしても、粒度の設計ミスでうまく機能しなかった。

✗ 粒度が荒すぎるSkill 「実験を全部やって」



- ✗ 内部がブラックボックス化
- ✗ どこで間違えたか追えない
- ✓ 結果の妥当性を検証できない

✗ 粒度が細かすぎるSkill (一手ごとに分割)



- ✗ 組み合わせ爆発
- ✗ オーケストレーションが破綻
- ✗ 呼び出しコストが膨張

⚠️ 原因

Skillの粒度を「人間の作業単位」で考えて切っていた。

✗ うまくいかない視点



人間の作業単位
(やりやすさ／手間の区切り)

✓ 本来必要な視点



再利用と検証ができる
最小単位
(検証可能性の区切り)

この視点の欠如が、粒度ミスの原因だった。

💡 見えた本質

AI for Scienceの設計とは、研究プロセスを
“検証可能な単位”に正しく分節すること。



検証可能な境界を持つこと
どこまでを1つのSkillにするかを
明確に定義する。



再利用できること
他のプロジェクトやフェーズでも
使い回せる単位にする。



組み合わせ可能であること
必要なSkill同士が、過不足なく
つながる粒度にする。

分節の粒度そのものが、
品質を左右する設計判断。

実験ログ (抜粋)

Run	Phase	Skill名 (粒度の問題)	Invoke回数
R38	大粒度	ExperimentAllInOne	0
R39	大粒度	FullPipelineExecutor	1 (結果のみ出力)
R40	小粒度	Step01_LoadData	12
R41	小粒度	Step02_Preprocess	15
R42	小粒度	Step03_FeatureCalc	18
...	...	...	...



粒度の不設計は、
「使われない」「検証できない」「コスト増」を招いた。



研究を、どの粒度で切り分けるか。
これは作業効率の話ではなく、
“反証可能性をどこに置くか”という
本質的な設計判断だった。

粒度設計の考え方 (“検証可能な単位”に分節する)



1. 研究プロセスを俯瞰する
どの工程で、何を検証したいかを
明確にする



2. 検証の境界で切る
反証可能性を基準に、
境界を設定する



3. 最小単位にする
再利用と検証ができる
最小の仕事に分ける



4. 階層構造でまとめる
ルーター／カテゴリで
組み合わせを制御する



5. 検証と接続する
各Skillの出力を、検証と
監査に確実につなぐ

粒度の正しい設計が、
“呼ばれる・検証できる・
コストが最適化される”
OSをつくる。

【落とし穴9】

「外部ツールごとに作り込んで、 接続地獄に溺れた。」

リソースが増えるたびに
個別のAPI連携を毎回実装...



MCPで標準接続すれば、
差し込むだけでつながる！



🔍 症状

つなぎたいリソースが増えるたびに、
個別のAPI連携を毎回書く羽目に。



✕ 保守が破綻し、バグ対応に追われる

✕ 新しいデータ源を足すだけで数日が溶ける



完全な接続地獄に...

⚠️ 原因

外部接続を「その都度の作り込み」で
解決しようとした。



リソースごとに
仕様が違う...
毎回つなぎ方が違う...
時間がない...

標準化された接続規格を
介していなかった。

💡 対策／見えた本質

MCP(Model Context Protocol)という
標準プラグに統一。

✓ 各リソースを「MCPサーバー」として提供
共通の作法で呼び出すだけ

✓ 新しいリソースも「MCPサーバー」を
追加するだけで接続完了

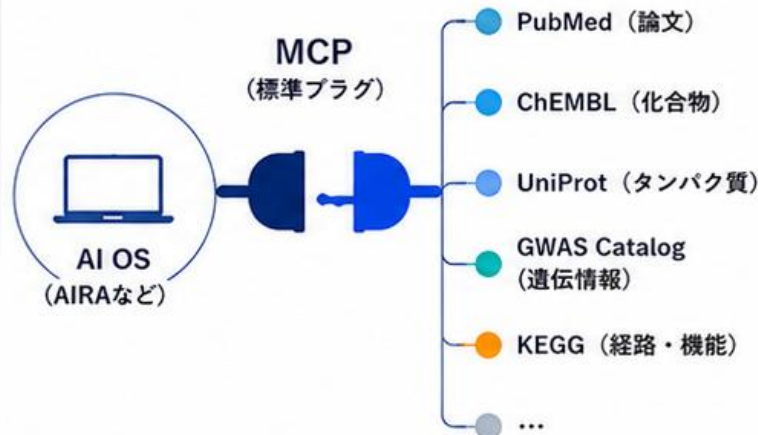
ToolUniverse (Harvard / Zitnik Lab)

1,000+ 科学DB を統一インターフェースで提供

PubMed ChEMBL UniProt GWAS Catalog KEGG ...

Reach（到達範囲）を広げる本質は、
個別接続の根性ではなく、
“標準化された接続の上に立つ”こと。

MCPでつながる世界（例：ToolUniverse）



👍 差し込むだけで、世界中の科学資源に手が届く。



世界中の科学資源に手を伸ばす力は、
根性ではなく、
標準規格が支えていた。

MCP導入で得られる価値



接続コストの激減
新しいリソース追加が
数分～数時間に



保守の簡素化
共通インターフェースで
安定・拡張しやすい



再現可能性の向上
同じ作法で、同じ結果を
再現できる



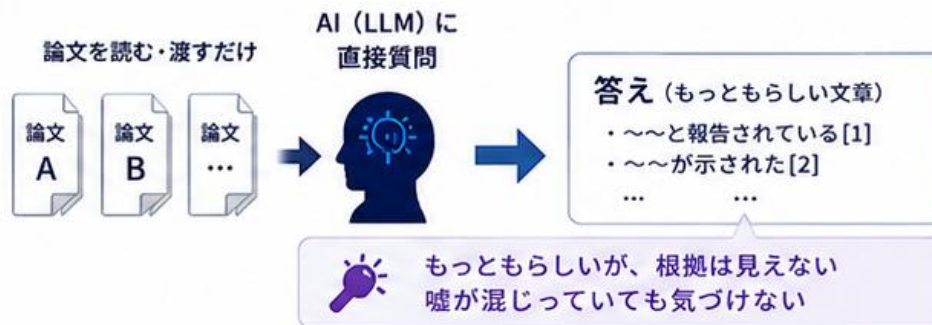
Reachの最大化
1,000+ DBへ
一気に到達可能

差し込むだけで手足が増える設計こそが、
再現可能なReachを生む。

“標準が、研究のスピードと
再現性を、同時に引き上げる。”

【落とし穴10】

“論文を読ませて聞く”だけでは、 AIは嘘をつく。



🔍 症状

「論文を読ませて質問する」だけでは、研究に
使えない問題が頻発した。

① 最新論文に答えられない

📅 学習データの時点以降の論文は知らない。
新しい発見に追いつけない。

② 根拠を示せない

📄 どの論文のどの記述に基づくのか
不明。研究には致命的。

③ ハルシネーション (幻覚)

👻 存在しない引用や事実を、あたかも
真実のように生成する。

⚠️ 研究の信頼性を根底から揺るがすリスクに直面した。

❗ 原因

LLMは、“流暢な回答”を生成する装置であって、
“出典をたどれる回答”を保証する装置ではない。



流暢さ (もっともらしさ) を優先

それらしい文章を生成することに最適化
されている。



出典の追跡・保証はしない

内部に出典追跡の仕組みは持たず、
根拠の同定ができない。

だから、間違っても自信満々に回答する。

💡 対策

RAG (Retrieval-Augmented Generation) で、
検索による根拠づけを行う。



関連情報を検索

論文コーパスから、質問に関連する
情報を検索して取り出す。



根拠として提示

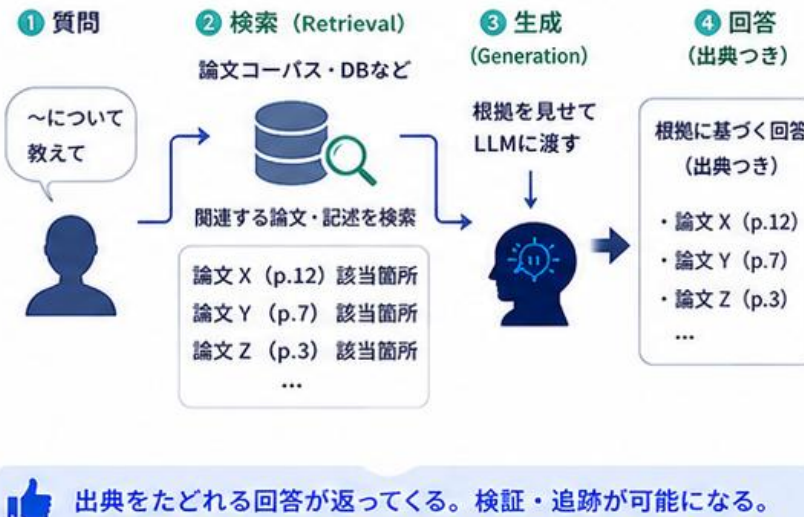
検索した情報 (出典・該当箇所) を
明示して、LLMに渡す。



根拠に基づき回答

根拠に基づいて回答を生成し、出典を
明示して返す。

RAGの仕組み (イメージ)



研究で必要なのは、流暢な答えではなく、
出典をたどれる答え。

AI for Science の知識獲得とは、
“正しい文章”ではなく“検証可能な根拠”を得る営みである。

LLM直読み

VS

RAG

✗ 最新論文に弱い	情報の鮮度	✓ 最新情報にも対応可能
✗ 根拠を示せない	根拠の提示	✓ 出典を明示できる
✗ ハルシネーションのリスク高い	信頼性	✓ ハルシネーションを大幅に低減
✗ 再現性・追跡性が低い	研究への適合性	✓ 再現性・追跡性が高い



研究の問いに対する答えは、
説得力ではなく、
追跡可能性で評価される。

【落とし穴11】

「答えは“1本の論文”には書かれていなかった。」



Classic RAGで検索
(ベクトル類似度 + top-k)



似た論文は拾えたけど、
本当に必要な関係や
つながりは見えない...



🔍 症状 (研究で起きたこと)

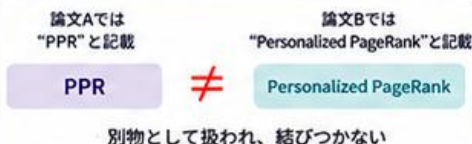
1 マルチホップ推論ができない

「Aの手法を改良したBを使った研究は？」
→ 答えが単一文書で完結しない。



2 エンティティの同一性が扱えない

“PPR” = “Personalized PageRank”。
同じ概念が別物として索引化される。



3 矛盾するファクトを検出できない

論文A「手法Xは95%」
vs 論文B「手法Xは80%」。
矛盾していても、気づけない。



4 専門用語の表記揺れを取り逃す

“TF-IDF” ↔ “term frequency-
inverse document frequency”。
表記が違うと、別物として扱う。



! 原因

ベクトル類似度は、
“似た文章”は引けるが、
“論文をまたいだ関係性”を持たない。

ベクトル検索の限界



点 (類似) は分かっても、
線 (関係) が失われる。

Classic RAG (ベクトル類似度 + top-k) の限界

何が起きるか

1 マルチホップ推論ができない



2 エンティティ同一性が扱えない

“PPR” ≠ “Personalized PageRank”

3 矛盾するファクトを検出できない



4 表記揺れ・言い換えを取り逃す

TF-IDF ↔ term frequency-
inverse document frequency

なぜ問題か

答えは複数論文の橋渡しで初めて完成。
単一文書では情報が分断される。

同じ概念でも別物として扱われ、
知識が統合されない。

矛盾や相違を自動で発見できず、
誤った結論に導くリスクが高い。

表記の揺れに弱く、重要な文献や記述を
見落としてしまう。



学術知識の本質は、
“複数論文をまたいだ関係性”にある。
ベクトル類似度だけでは、その“関係グラフ”が失われる。



学術知識は、“点”ではなく“線”のネットワーク。

研究に必要なのは、似た文章ではなく、
異なる論文をつなぐ“関係性”を理解すること。

本当に必要なのは「関係グラフを持つRAG」へ

Classic RAG (ベクトル)

- ✓ 似た文章を探すのが得意 (点)
- ✓ 関係性・構造は持たない
- ✓ マルチホップ・同一性・矛盾検出が弱い



GraphRAG (関係グラフ)

- ✓ 論文をまたいだ関係を保持 (線)
- ✓ 同一性の統合・表記揺れを吸収
- ✓ 矛盾検出・因果・系列の理解が可能



答えは、関係の中にある。

研究の問いに対する答えは、
“1本の論文”ではなく、
“関係性のネットワーク”の中に
存在していた。

論文の海を、“つながりの地図”に変える —— Graph RAG。

仕組み：論文を「関係の地図（ナレッジグラフ）」に変える



グラフの構成要素

- ノード（点）：エンティティ
 - 概念・用語
 - 人物・組織
 - 手法・アルゴリズム
 - データ・指標・材料
 - ...
- エッジ（線）：関係
 - 例) 改良した／評価した／比較した／部位を持つ／原因である／同義語 など



見えた本質

Reach（到達範囲）を広げるとは、海をやみくもに泳ぐのではなく、“関係の地図”を持つこと。
AI for Science の知識獲得の本質は、情報量ではなく、関係構造である。

グラフが解決すること

1 マルチホップ推論を可能にする (エッジを辿る)

A → B → C → D のように、複数の論文を橋渡しして答えに到達。



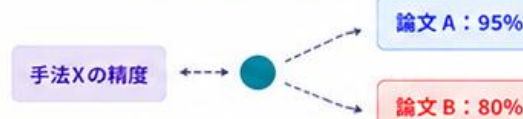
2 表記揺れ・同一性を正規化

“PPR” と “Personalized PageRank” を同一ノードに統合。



3 矛盾するファクトを可視化

同じエンティティに、異なる事実を並べて比較・検証できる。



Classic RAG（ベクトル検索ベース）



Graph RAG（グラフ探索ベース）



Graph RAG を使うと、研究がどう変わるか



答えの到達範囲が大きくなる
(Reachの拡大)

単一論文の壁を越え、
関連知識のネットワーク
全体から答えを見つけられる。



根拠の追跡性が高まる
(再現性の向上)

どの関係を通して
その結論に到達したかを
明確に示せる。



表記揺れ・同義語を吸収
(検索漏れの削減)

同じ概念を1つのノードに統合し、
取り逃しを減らして
検索精度を高める。



矛盾や不確実性が見える化
(批判的検証を支援)

異なる事実を並べて比較し、
不確実性や限界を
明らかにできる。



研究の質を決めるのは、
情報の量ではない。
関係を構造として
扱えるかどうかだ。



バラバラの論文という“点”を、
関係という“線”でつなぐ。
海が、歩ける地図になる。

【落とし穴12】

GraphRAGの主張を、鵜呑みしかけた。

論文はスタート地点。ゴールは「自分の手で確かめた事実」。



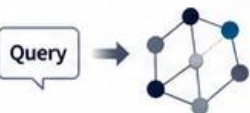
見えた本質

AI for Scienceの本質は、他者の主張を信じるのではなく、自分の手で再発表し、反証可能な形で確かめ続けること。
＝第3部の「動いた＝正しい」が、知識獲得の段階でも貴かれる。

① 症状

まず定番の
Microsoft GraphRAG
を採用した。

GraphRAG



ところが、応答が
どうもおかしい…。

② 実測（ベンチマーク）

HotpotQA でベンチマークを実施
(LLM 判定の正解率)

Microsoft GraphRAG の結果

43.5%

GraphRAGを使っても、
正しい答えが返る確率は
むしろ低かった。

③ 転機（最新研究の出現）

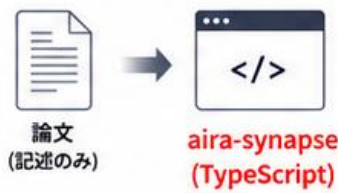
2026年6月、arXiv に投稿された
MemGraphRAG が高性能と報告。

MemGraphRAG
arXiv
2026.06
arXiv:2606.00610

論文の主張＝高性能。
だが「論文通りに作れば、
論文通りの性能が出る」
保証はない。

④ 対策（実証：再実装）

公式コードを見ずに、論文だけから
再実装（クリーンルーム設計）。



ベンチマークで
反証可能な形で
確かめた。

⑤ 結果（ベンチマーク）

正解率が大きく向上！

aira-synapse（再実装）の結果

英語 (HotpotQA)

91.2%

日本語 (JHotpotQA)

78.3%

GraphRAG の 43.5% から、
大きく改善した。

メッセージ

“ 90点は、
10回に1回は
間違える、
ということ。
だから論文も
自分も
鵜呑みにしない。 ”

定番 GraphRAG（Microsoft GraphRAG）の限界

マルチホップ推論が弱い



複数段友をまたぐ推論が
うまくできない。

表記揺れ・同一性の統合が不十分

PPR $\neq$ Personalized PageRank

同じ概念を別物として扱い、
見逃しが発生。

矛盾するファクトを検出できない

論文A
手法Xは95% VS 論文B
手法Xは80%

食い違いを自動で検出・整理
できない。

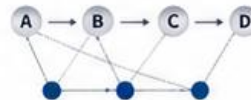
専門用語の表記揺れを取り逃す

TF-IDF $\neq$ term frequency-
inverse document
frequency

表記の揺れを吸収できず、
取り逃す。

aira-synapse（再実装：MemGraphRAGをベースに実装）の強み

マルチホップ推論を強化



多段の関係をたどり、複雑な
質問に強い。

表記揺れ・同一性を統合

PPR = Personalized PageRank

同一エンティティに統合し、
取り逃しを削減。

矛盾するファクトを可視化

論文A
手法Xは95% VS 論文B
手法Xは80%

比較・可視化により、矛盾を
明確に把握。

表記揺れを吸収

TF-IDF = term frequency-
inverse document
frequency

表記揺れを吸収し、検索精度を
向上。



最高性能の手法でも、
100% ではない。



90点台 = 10回に1回は間違える



- ベンチマークで反証可能な形に確かめる
- 他者の主張も、自分の結果も、鵜呑みにしない
- 「動いた」だけでなく、「正しい」かを確かめ続ける



AI for Science の知識獲得は、
「信じること」ではなく、
「確かめ続けること」から始まる。

【落とし穴13】

研究者のノートPCで動かない基盤は、無いのと同じ。

AI for Science の主役は**研究者個人**。再現性は、研究者の**ノートPC**の上で守られる。



見えた本質

Reach（到達範囲）を“再現可能”にする鍵は、性能の高さではなく、“**誰の手元でも動くこと**”。
どんなに高性能でも、机の上で動かなければ、その能力はゼロと同じ。

🔍 症状

定番のグラフDB「Neo4j」を採用したが、
個人PCには重すぎた。



- ✓ プロセス常駐でメモリを大量消費
- ✓ 起動・シャットダウンに時間がかかる
- ✓ インデックス構築が重い
- ✓ ライセンスや設定も複雑



研究者のノートPCで快適に動かせない。
研究の試行錯誤が止まり、
手元で完結できない。



基盤が重いと、再現性そのものが失われる。

🎯 原因

「高性能・業界標準」を、
「研究者が個人で再現できる」より優先してしまった。

優先していたこと（誤り）

- 🖥️ 高性能・業界標準
- 🏆 実績・信頼性
- 🛡️ 大規模運用を前提

本来優先すべきこと（正解）

- 💻 研究者のノートPCで動く
- 👤 個人で再現・検証できる
- 🔄 軽く・早く・シンプル



AI for Science の主役は、企業や大規模環境ではなく、
研究者一人ひとりだった。



対策（軽量の基盤へシフト）

個人PCで動き、再現可能な軽量グラフDBへ。



Memgraph

- ✓ 軽量・高速
- ✓ メモリ効率が高い
- ✓ 単一バイナリで簡単起動



Graphiti

- ✓ Pythonベースで扱いやすい
- ✓ ローカルで完結
- ✓ 開発・検証に最適



LlamaIndex + SQLite

- ✓ SQLでグラフを扱える
- ✓ 軽量・持ち運び可能
- ✓ 再現性が高い



研究者のノートPCで、最後まで動くことが最優先。

Neo4j（従来）と軽量グラフDBの比較

項目	Neo4j（従来）	軽量グラフDB (例: Memgraph / Graphiti / LlamaIndex+SQLite)
動作環境	高スペックPC / サーバー推奨 (16GB RAM以上推奨)	ノートPCで快適に動作 (8GB RAMでも可)
起動・停止	遅い（数十秒～数分）	速い（数秒）
メモリ消費	大きい（数GB～十数GB）	小さい（数百MB～数GB）
設定・運用	複雑（DB設定・ユーザー管理など）	シンプル（単一バイナリ/ライブラリ）
再現性（個人）	低い（環境依存が大きい）	高い（誰の手元でも再現可能）

基盤の軽さがもたらす価値



Reach（到達範囲）を広げるには、
誰の手元でも動く軽さが、最強の基盤になる。

メッセージ

“どんなに高性能でも、
研究者の机の上で動かなければ、
その能力はゼロと同じ。”

再現できる
からこそ、
研究が進む。



【落とし穴14】

既製品のバグに沈み、ついにRustで自作した。

再現可能な Reach (到達範囲) のために、私は一番下の地面ごと、作り直した。



見えた本質

Reach を再現可能にしようとする、時に最下層（データ構造）まで自分で作り直す覚悟が要る。
AI for Science の実装は、借り物のスタックの上では完結しないことがある。

これまでの顛末

① Neo4j (重すぎる)

定番のグラフDB「Neo4j」を採用。しかし、個人PCには重すぎた。



- × プロセス常駐でメモリを大量消費
- × 起動・シャットダウンに時間がかかる
- × インデックス構築が重い
- × ライセンスや設定も複雑

⚠ 研究者が自分の手元で完結できない。再現性が失われる。

② LadybugDB (バグに直面)

軽量な「LadybugDB」に乗り換え。しかし、今度はバグに直面した。



LadybugDB

- × クエリ結果の不整合やクラッシュ
- × 大規模グラフで性能劣化
- × 更新処理の不具合
- × バグ修正の見通しが不透明

⚠ 既製品の不具合に、研究の進行が人質に取られる。

結果

「重すぎる」か「壊れている」か。どちらも研究者の再現性を奪う。



そこで決断した。
研究者が自分の手元で完結できる、信頼できる基盤を、自分の手で作ると。

対策（実証）：ゼロから自作

個人PCで完結する専用グラフDBをRustでゼロから自作。



aira-graphdb
(Rust 製)



研究者のノートPCで動作
(軽量・高速)



安定・信頼性の高いコア実装



目的に最適化された設計



再現性を最優先した
シンプルな構造

成果（ベンチマーク結果）

HotpotQA 500問 (英語)

正解率

89.4%



意味一致 (Semantic Match)

91.2%



論文の
GPT-4o-mini
(71.6%) に対して
+19.6pt

日本語対応 (GINZA統合)

HotpotQA 500問 (日本語) / 評価モデル: GPT-5.5

78.3%



Neo4j基準 (58.5%) に対して
+19.8pt

※ すべて同一評価条件で測定。詳細は技術レポート参照。

本質のまとめ



本質を再現可能にしようすると、時に最下層（データ構造）まで自分で作り直す覚悟が要る。借り物のスタックの上では、完結しないことがある。

選択の比較

DB	重量/性能	安定性	個人PCでの再現性	研究の進行への影響
Neo4j	× 重い	○ 高機能	× 難しい	× 再現性が失われる
LadybugDB	○ 軽い	× バグあり	× 不安定	× 不具合で停滞
aira-graphdb	○ 軽い・高速	○ 安定	○ 簡単 (個人PCで完結)	○ 研究が前に進む



メッセージ

どんなに高性能でも、研究者の机の上で動かなければ、その能力はゼロと同じ。

【落とし穴15】 「英語で動いたものが、日本語で崩れた。」

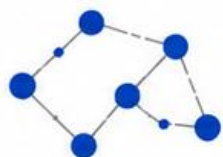
グローバルなツール群を日本語研究に適用したときの落とし穴

⚠ 症状

英語の論文・ベンチマークでは高精度だった GraphRAG が、日本語ではエンティティ抽出・正規化が崩れ、精度が大きく落ちた。

英語 (論文・ベンチマーク)

高精度で動作



正確なエンティティ抽出
安定した関係グラフ

高精度 (例) 85%+

日本語

崩れて精度が低下



抽出・正規化が崩壊
関係グラフが不安定に

低精度 (例) 58.5%
(Neo4j基準)

⚙ 原因

エンティティ抽出や形態素処理が、英語前提で設計されている。



英語前提の設計

単語間にスペースがあることを前提に、トークン化や固有表現抽出を最適化。



日本語特有の課題

分かち書きの無さ・表記揺れ (漢字 / かな / カタカナ) が、エンティティの分断や重複を生み、関係グラフを壊す。

日本語の難しさ (例)

分かち書きの無さ

例) 人工知能は社会を変える
→ どこで区切る?

表記揺れ
(漢字 / かな / カタカナ)

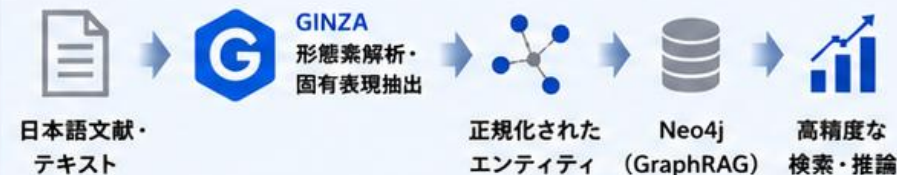
例) 人工知能 / AI / エーアイ
東京大学 / 東大 / とうきょうだいがく

同義・略称・異表記

例) がん / 癌 / ガン
心筋梗塞 / 心筋梗塞症 / MI

✓ 対策

日本語形態素・固有表現解析 (GINZA) を統合し、日本語のエンティティ抽出を作り込む



改善結果 (精度)

改善前
(Neo4j基準)

58.5%

改善後
(GPT-5.5)

90.7%

+32.2pt
の改善



見えた本質

AI for Science のツールは“言語に中立”ではない。

日本の研究者が本質を享受するには、
ローカライズ = 言語の地面を整える設計が要る。



メッセージ

世界標準のツールも、日本語の研究現場では、
もう一度地面を作り直す必要があった。

【落とし穴16】

論文に書いてある手法より、書いてない改善が効いた。

AI for Science の成果を決めたのは、派手な新手法ではなく、地味な“配管（設計）”だった。



見た本質

成果を決めるのは、最先端アルゴリズム
そのものより、それを支えるプロセスと
インフラの設計。
＝ 第3部の「質さではなく設計」が、
内部でも再現した。

① 症状

最新の派手なアルゴリズムを
次々に導入したが、
性能は思ったほど伸びなかった。



MemGraphRAG など
最新手法を導入



新しいアイデアを追加



複雑な仕組みを実装

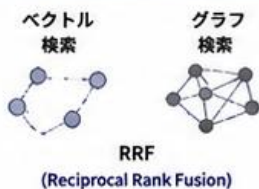


なぜ、こんなに
頑張っているのに、
伸びないのか…？

② 原因／発見

後から徹底的に分析して分かった。
効いていたのは、論文に書いていない“配管”。

ハイブリッド検索
(ベクトル+グラフ+RRF)



- ✓ 網羅性が向上
- ✓ 見逃しを大幅に削減
- ✓ マルチホップが安定

評価関数の統一
(スコアを同じものさしに)



- ✓ 比較が可能に
- ✓ 最適化が進む
- ✓ 全体のチューニングが可能

！ 論文の主役（新手法）より、脇役（配管）の改善が圧倒的だった。

③ 対策（実装した“配管”の設計）

地味だが効く設計を徹底的に作り込んだ。



ハイブリッド検索の実装
ベクトル検索+グラフ検索+RRF融合



評価関数の統一・正規化
スコアのスケールと重み付け最適化



スコア・信頼度の一貫管理
全工程で同じものさしを使用



再現可能なチューニング基盤
パラメータ・重みを体系的に最適化



地味な“配管”を磨き込むことで、
全体の性能が大きく改善した。

④ 成果（ベンチマーク結果の改善）

英語（HotpotQA 500問）

正解率

89.4%
(vs 論文 71.6%)



論文（GPT-4o-mini（71.6%）に対して

+19.6pt

※ すべて同一評価条件で測定。詳細は技術レポート参照。

日本語対応（GINZA統合）

意味一致（Semantic Match）
評価モデル：GPT-5.5

78.3%



Neo4j基準（58.5%）に対して

+19.8pt

どこに注目してしまいがちか？

研究者の思い込み



- ・新しいアルゴリズムが正解
- ・最新手法を追えば最強
- ・複雑＝高性能

現実（今回の結果）

アルゴリズム単体の改善より、
配管（検索のつながり・評価の統一）
の設計が圧倒的に効いた。



派手な新手法に目を奪われると、最も効く改善を見逃す。

改善のインパクト（寄与度の比較）

HotpotQA 500問（英語）正解率への寄与（例）



新手法そのものの
寄与はわずか。
配管の設計が
主役だった。

0 5 10 15 20
正解率の改善量（ポイント）

メッセージ



論文の主役（新手法）より、
論文の脇役（地味な配管）が、
結果を決めた。
本質は、いつも地味な場所にある。



AI for Science の成功は、
最先端アルゴリズムではなく、
設計の積み重ねで決まる。

道具をそろえても、研究は自走しない — 効くのは**「接続の設計」**。

第4部（落とし穴7～16）を貫く発見

共通の落とし穴

良い部品をそろえること自体は**本質ではなかった**。



効いたのは**「接続の設計」**



この接続・配管の設計が、**研究の品質を決めた**。

第3部との接続

第3部＝生成は設計で決まる。第4部＝道具も設計で決まる。



本質は一貫して **“プロセス設計”** だった。

見えできた本質②

AI for Science の“OS”の価値は、高機能な部品の所有ではなく、部品を**再現可能に編み込む**オーケストレーションにある。



第1部のギャップで言えば

Speed 側の発想

道具を「速く・たくさんそろえる」



しかし
効いたのは



Reach / 検証側の設計

「再現可能につなぐ」設計



本質②もまた、ギャップの **“私の側”** にある。

メッセージ



武器箱を満たすことではなく、
武器を使う**段取りを設計**すること。
それが **OS の正体**だった。

OSとは、"頭脳と手足を、検証でつなぐ段取り"である。

第4部のまとめ：研究OS「AIRA」が体現する本質



Skill × MCP が噛み合うワークフロー (研究テーマを渡すと自動で回る)



第4部の教訓 — 部品ではなく、"つなぎ方"を設計せよ。

落とし穴7～16の教訓を、実務に落とし込む

落とし穴		教訓（何を学び、どう設計すべきか）	
7		呼ばれないSkill	→ 能力に "呼び出し経路" を設計。
8		粒度ミス	→ 検証可能な最小単位で分節。
9		MCP接続地獄	→ 個別実装でなく標準規格（MCP/ToolUniverse）。
10		LLM直読み	→ 出典をたどれるRAGへ。
11		Classic RAGの限界	→ 関係構造（Graph RAG）へ。
12		論文鵜呑み	→ 再実装して反証可能に確かめる。
13		重すぎる基盤	→ "個人PCで動く" を最優先。
14		既製DBのバグ	→ 必要なら最下層を自作（aira-graphdb）。
15		日本語の壁	→ ローカライズで地面を整える。
16		書かれざる改善	→ 派手な手法より地味な配管。

共通項（本質）

 10個の穴は、
.....

すべて

"部品の所有" を

▼

"プロセスの設計" と
取り違えたことから
生まれた。

次部への
橋渡し



だが、設計したプロセスが本当に機能しているかを、
誰が・どう確かめるのか？

➡ 第5部：評価と監査の穴へ



第5部

落とし穴

[評価・再現性・運用]

監査なきAIは、
もっともらしい嘘をつく

見えてきた本質③



大切なのは、“信じること”ではなく、“確かめ続ける設計”です。

【落とし穴17】

もう一度走らせたら、 違う結果が出た。



症状

同じ仮説、同じコードのはずなのに、
再実行すると数値が変わる。
論文に書いた値が、二度と再現できない。



原因

乱数シードが固定されていない／
実行環境（ライブラリのバージョン）が記録されていない。
AIは“動くコード”は書くが、
“再現可能なコード”を自発的には書かない。



対策

ゴールデンルール①
ノートブックの最初に必ず、環境セル（バージョン記録）と
シードセル（乱数固定）を置く。



見えた本質

科学の最小条件は「動く」ではなく「再現できる」。
AI for Science の本質は、生成の華やかさではなく、
再現可能性という地味な規律にある。

ゴールデンルール①：最初に2つのセルを置く

環境セル（バージョン記録）



- Python / OS / ライブラリ
- バージョンを記録
- 再現に必要な環境を保存

```
# 環境の記録例
import sys, platform, pkg_resources
print(platform.platform())
print(sys.version)
pkg_resources.working_set.by_key
```



シードセル（乱数固定）



- 乱数シードを固定
- 毎回同じ結果を保証
- 再現性の土台を作る

```
# シードの固定例
import random, numpy as np
random.seed(42)
np.random.seed(42)
```



この2セルが、再現性の土台になる



“動いた”と“再現できる”は、
別物です。どちらが科学でしょうか？
.....



第5部のテーマ： 設計したプロセスが、本当に機能しているかを、どう確かめるか？
次ページから、評価・再現性・運用の落とし穴を一つずつ見ていきます。



【落とし穴18】

存在しない論文を、堂々と引用していた (Ghost References)。



症状

レポートの参考文献に、もっともらしい著者名・タイトル・DOIが並ぶ。しかし、その論文は存在しない。あるいはDOIが別物を指す。



原因

LLM（大規模言語モデル）は、引用の“形式”を学習している。形式的に正しい引用を、中身の存在と無関係に生成できてしまう。



対策

引用台帳を作り、すべての引用を外部DB（Crossref / PubMed 等）で実在照合。存在しないものは機械的に除去・差し替え。



見えた本質

研究の信頼は“引用の見た目”ではなく“引用の実在”に宿る。AI for Science の評価は、形式ではなく、外部世界との照合で担保される。

対策：引用台帳を作り、すべての引用を機械的に実在照合する



もっともらしい引用ほど、**危険**。

研究の誠実さは、

機械的な**実在チェック**でしか守れなかった。



第5部のテーマ：設計したプロセスが、本当に機能しているかを、どう確かめるか？

次ページから、評価・再現性・運用の落とし穴を一つずつ見ていきます。



【落とし穴19】

本文は"図3が示す通り"
と言うが、
図3が存在しない。



症状

本文が「図3に示すように…」と参照しているのに、その図がそもそも生成されていない。逆に、生成された図がどこからも参照されない（孤立図）。



原因

本文生成と図生成が別工程で進み、相互の整合がとられないまま結合される。



対策

図マニフェストを作成し、「本文の図参照」と「実在する図ファイル」を機械的に突合。plt.savefigで保存された図と参照を一対一に対応づける。



見えた本質

科学的主張は、必ず「証拠（図・表・計算）」と結びついていなければならない。AI for Science の本質は、主張と証拠の一対一対応＝トレーサビリティにある。

対策：図マニフェストで「本文の図参照」と「実在する図」を機械的に突合

1

図参照を抽出
（本文を解析）



本文の図参照リスト（例）

- ・ 図1
- ・ 図2
- ・ 図3
- ・ 図4

2

図ファイルを収集
（plt.savefig 等）



実在する図ファイル（例）

- ・ figure_01.png
- ・ figure_02.png
- ・ figure_04.png

3

機械的に突合
（一対一で照合）

図1	✓
図2	✓
図3	✗
図4	✓

照合結果（例）

4

不整合を修正
（作成・削除・差替え）



存在しない図は作成、
孤立図は削除または
本文に参照を追加

5

整合したレポート
を出力



主張と証拠が
一貫したレポート



主張だけが宙に浮く"幽霊図"は、
第1の落とし穴「動いた≠正しい」の、
評価フェーズでの再来だった。



第5部のテーマ： 設計したプロセスが、本当に機能しているかを、どう確かめるか？

次ページから、評価・再現性・運用の落とし穴を一つずつ見ていきます。



【落とし穴20】

要旨は92%、本文は89%——
同じ実験の数字が食い違う。



症状

要旨・本文・図表・結論で、
同じ指標の数値が微妙に食い違う。
どれが本当の結果か分からない。



原因

長い文書を生成する過程で、AIは数値を
“その場で再生成”してしまい、
計算の真の出力にひも付かない。



対策

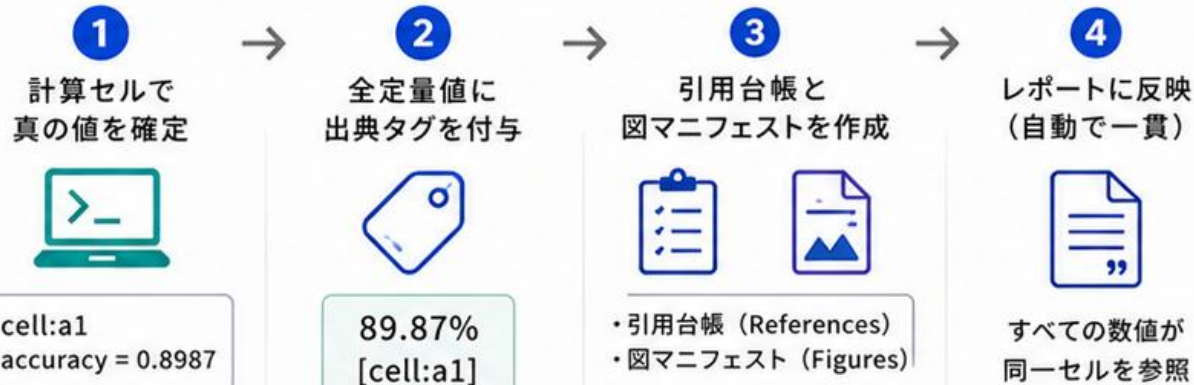
ゴールデンルール②③
すべての定量値に [cell:<id>] の出典タグを付け、
どの計算セルの出力かを一意にたどれるようにする。
レポート作成前に引用台帳と図マニフェストを作る。



見えた本質

第3～20の穴は、すべて「主張と、その根拠
(計算セル) が、一意に結びついていない」
一点に帰着する。
AI for Science の本質は、すべての主張に
出典タグを付ける規律だった。

ゴールデンルール②③：すべての定量値に出典タグを付け、事前に台帳とマニフェストを作る



不整合の例 (あるある)

要旨：精度 92%	×
本文：精度 89%	×
図3のキャプション：精度 91%	×
結論：精度 90%	×



対策後 (整合したレポート)

要旨：精度 89.87% [cell:a1]	✓
本文：精度 89.87% [cell:a1]	✓
図3のキャプション：精度 89.87% [cell:a1]	✓
結論：精度 89.87% [cell:a1]	✓

[cell:<id>] = その数値の根拠となる計算セルID



数値は“書く”ものではなく、
計算セルから“引用する”もの。
これがすべての落とし穴の共通解だった。



第5部のテーマ： 設計したプロセスが、本当に機能しているかを、どう確かめるか？

次ページから、評価・再現性・運用の落とし穴を一つずつ見ていきます。



【落とし穴21】

結果を見てから、 評価指標を選んでいた。



症状

実験結果を見たあとで、
「この結果が一番よく見える指標」を選んでしまう。
どの実験も、なぜか成功して見える。



原因

評価指標を、生成・実行の“後”に決める自由を
残していた。
後出しジャンケンが、構造的に可能だった。



対策

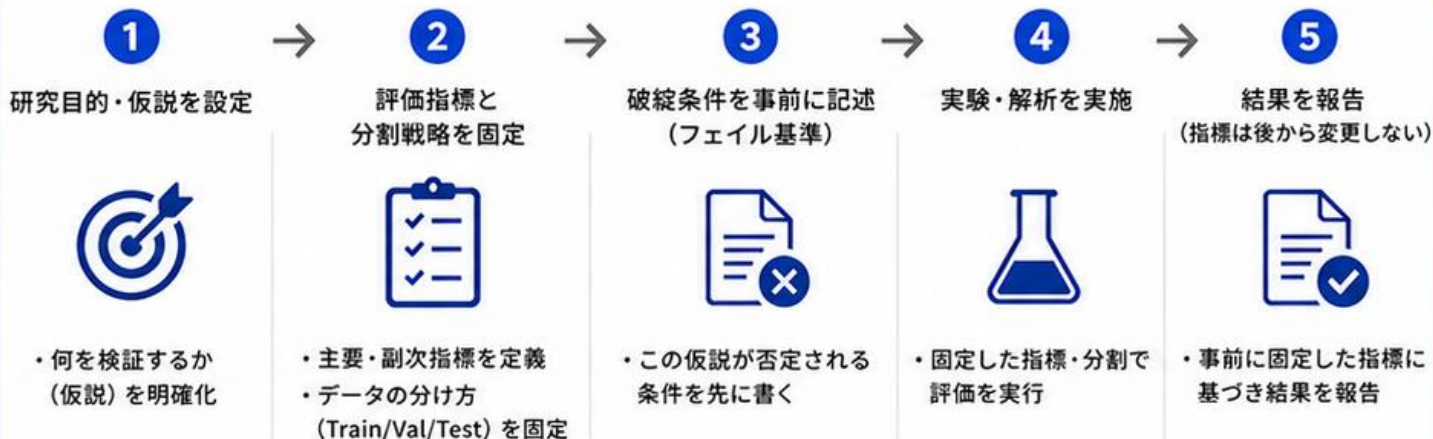
評価指標と分割戦略を、仮説を立てる“前”に固定する
(事前登録に相当)。
仮説の前に「この仮説が破綻する条件」を書く。



見えた本質

AI for Science の誠実さは、能力ではなく
“手続きの順序”で守られる。
指標を先に固定することが、反証可能性そのもの。

対策：評価は“先に決める”。事前登録に相当する手順を徹底する



不適切な例：後出し



適切な例：事前に固定



いつ指標を決めるか、で科学かどうかが決まる。
後出시를禁じる規律が、本質だった。



第5部のテーマ：設計したプロセスが、本当に機能しているかを、どう確かめるか？

次ページから、評価・再現性・運用の落とし穴を一つずつ見ていきます。



【落とし穴22】

数字は上がった。 でも、何の意味もなかった。



症状

ベンチマークのスコアは順調に上がる。
だが、その向上が、研究上の問いの解決と
何の関係もない、という事態。
指標を上げるために指標を上げていた。



原因

Goodhartの法則
(指標が目標になると、指標は良い指標でなくなる)。
スコア最適化が自己目的化した。



対策

指標を複数系統で持ち (Balance=VM/UC/FO
の統合スコア)、単一指標の独走を、別系統の
指標と研究目的への接続でチェックする。



見えた本質

AI for Science の評価の本質は、
“高いスコア”ではなく“問いに答えているか”。
Reach (到達範囲) は、ベンチマーク上の数字
ではなく、答えられる問いの広がりで見られる。

なぜ起きるのか？ (Goodhartの法則)

本来の目的



研究上の問いの解決

- ・新しい知見の獲得
- ・仮説の検証・反証
- ・現実世界への貢献

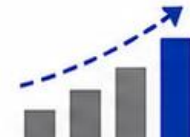
いつの間にか…



指標 (スコア) を
上げることが目的に

- ・スコア最適化
- ・見栄えの良い出力を優先
- ・指標の盲点を突く最適化

起きること



スコアは上がる。
しかし、問いの解決とは
無関係になる。

結果



数字は上がった。
でも、何の意味も
なかった。

対策：複数系統の指標でバランスを見る (Balance=VM/UC/FO)



VM: 数値整合性

数値の一致率・
出典タグ率など



UC: 裏づけ一致度

主張と根拠 (計算・
出典) の一致度など



FO: 図の整合性

図の存在率・参照率・
孤立図の少なさなど

Balance (統合スコア)

3指標の加重統合で総合評価

単一指標の独走を防ぐ仕組み

- ✓ 複数系統の指標で相互に牽制
- ✓ 研究目的への接続で妥当性を確認
- ✓ 異常値や偏りを早期に検出
- ✓ 指標の盲点を減らす

研究目的への接続 (例)



スコアを上げることと、研究を進めることは、
しばしば**別物**だった。



第5部のテーマ： 設計したプロセスが、本当に機能しているかを、どう確かめるか？

次ページから、評価・再現性・運用の落とし穴を一つずつ見ていきます。



【落とし穴23】

監査までAIに任せたら、 すべてが素通りになった。



症状

生成・検証・監査を、すべて同じAIに一気通貫でやらせると、監査が形骸化。問題を素通りさせる。



原因

監査が、生成と同じ主体・同じ文脈で行われると、自分の出力を甘く見る。独立性がない。



対策

独立したカスタムエージェントを置く。役割と権限を分離し、相互牽制が働く構造にする。



見た本質

AI for Science の品質保証は、能力の追加ではなく、“独立した監査主体”という構造で実現する。監査の独立性こそが、反証可能性の制度的な担保。

よくある失敗パターン：すべて同じAI



同じAI・同じ文脈で自己生成 → 自己検証 → 自己監査
甘い判断・バイアスの温存・問題の見逃し

対策：独立した監査エージェントを導入



役割・権限を分離。独立した観点で厳格に監査
相互牽制により、問題を見逃さない仕組みへ

導入する独立エージェントの役割

research-lead

(研究リード／統括監督)



- ・研究全体の方向性を監督
- ・各フェーズ間の整合性を確認
- ・目的・仮説・指標の一貫性を担保
- ・最終判断と承認

methods-auditor

(方法論オーディター)



- ・方法論の妥当性を独立に監査
- ・手法選択の根拠を検証
- ・ベースライン比較の有無を検証
- ・代替手法やバイアスの有無を確認

監査チェックの例

- ✓ 仮説と手法は整合しているか？
- ✓ 手法選択の根拠は明示されているか？
- ✓ ベースライン比較は実施されたか？
- ✓ データ分割・前処理は妥当か？
- ✓ 指標の事前固定は守られているか？
- ✓ 結果の解釈に飛躍はないか？
- ✓ 不整合・引用漏れ・再現性の問題は？



生成と監査を分離する。

これは技術ではなく、**科学のガバナンス**の問題だった。



第5部のテーマ： 設計したプロセスが、本当に機能しているかを、どう確かめるか？

次ページから、評価・再現性・運用の落とし穴を一つずつ見ていきます。



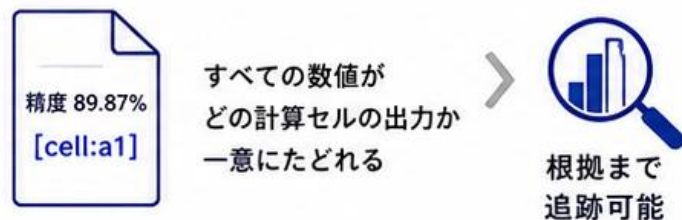
3つのゴールデンルールが、AI特有の嘘を撃ち落とす。

AIRAの品質ゲート（全フェーズ共通：第5部の落とし穴17～23への対策）

Rule 1 環境セルとシードセルを最初に実行（再現性）



Rule 2 レポートの全定量値に [cell:<id>] 引用を付与（追跡可能性）

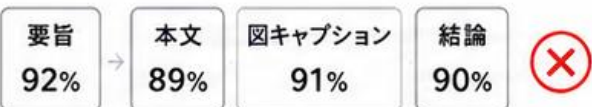


Rule 3 レポート作成前に引用台帳と図マニフェストを作成（整合性）



品質ゲートが検出する代表的問題（AI特有の三大欠陥）

Value Mismatch（数値不整合）
要旨・本文・図表・結論で、同じ指標の数値が食い違う。



Ghost References（幽霊引用）
もっともらしい著者名・タイトル・DOIが並ぶが、実在しない論文、または別物を指す。



Figure Orphans（幽霊図）
本文が図を参照しているのに図が存在しない、または図がどこからも参照されない。



3ルール+2監査の多層防御で撃ち落とす

- 3つのゴールデンルールで一次チェック（自動）
- 独立監査エージェントで二次チェック（人間の判断基準を適用）

嘘を見逃さない仕組みへ

独立した
2つの監査
エージェント



research-lead（研究リード／統括監督）

- 研究全体の方向性と整合性を監督
- 各フェーズ間の一貫性を確認（目的・仮説・指標の整合）
- 結果が研究の問いに答えているかを最終判断



methods-auditor（方法論オーディター）

- 手法選択の根拠を独立に監査
- ベースライン比較の有無・妥当性を検証
- データ分割・前処理・指標の妥当性を厳格にチェック

監査の独立性を制度化



- 役割・権限の分離
- 別コンテキストで監査
- 相互牽制でバイアスを抑制
- 監査記録を証拠として保存



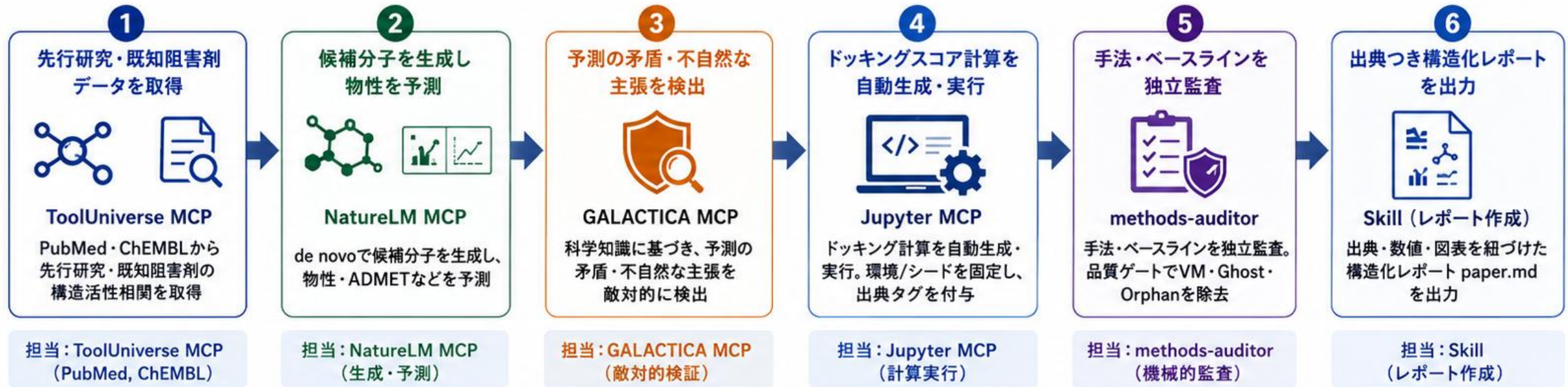
AIは“もっともらしい嘘”をつく。だからこそ、生成と同じくらい“監査”を重んじる。

AIRAの、そしてAI for Scienceの思想：再現性・追跡可能性・整合性・独立性が、反証可能な科学を支える。



“創薬テーマ”を渡すだけ — 監査つきで研究が一周する。

テーマ例 BACE1阻害剤（アルツハイマー病治療の創薬標的）の候補分子を設計したい



反証可能性の4層（全体を統合）



第5部は、後ろの2層（敵対的検証・機械的監査）の実装。これにより、研究成果を“反証可能”に保つ。



責任あるAIの利用

AIRAは、候補生成・文献調査・解析・レポート化を高速化するツールです。実験的妥当性・科学的発見を保証するものではありません。最終判断は、研究者の検証が前提です。



この一連、ご自身の研究なら何日かかりますか？



AIが加速するのは“速度”ではなく、“再現可能性と信頼性”を担保した研究の一周です。

第5部の教訓 — すべての主張を、根拠に縛り付けよ。

落とし穴

何が起きたか（失敗）



教訓（やるべきこと）



具体的な実装・監査の仕組み（例）

17



再現できない。



環境・シードを最初に固定。



環境（OS・ライブラリ・バージョン）を固定
シード・パラメータ・ハイパー設定を記録

18



幽霊引用。
存在しない論文を引用した。



引用を外部DBで
実在照合。



Crossref / PubMed / arXiv 等でDOI照合
存在・書誌情報・著者を自動検証

19



幽霊図。
主張と一致しない図を掲示した。



図マニフェストで
主張と図を突合。



図ファイルにID付与し表・数値・コード出力と
自動突合（図マニフェスト）

20



数値不整合。
同じ数値が論文内で食い違った。



全定量値に
出典タグ [cell:id]。



Jupyterセル/実行IDを付与
例：AUC = 0.872 [cell:7f3a1b2c]

21



指標の後出し。
都合のよい指標を後から選んだ。



評価指標を
事前固定。



実験計画書・プロトコルに指標を事前登録
変更には理由・記録・承認を必須化

22



ベンチマークの罫。
都合のよいベンチだけ選んだ。



指標は複数系統＋
研究目的に接続。



複数ベンチ・複数指標で評価
研究目的（例：探索→候補最適化→検証）に接続

23



AI自己監査。
自分で自分にOKを出した。



独立した
監査エージェント。



methods-auditor 等の独立エージェントで
手法・ベースライン・全証跡を監査

共通項（第5部の核心）

7つの穴は、すべて
“主張が根拠から
切り離されている”
ことから生まれた。



本質 ≡
主張と根拠の
一意な結合。



次部への
橋渡し

だが、これだけ仕組みを整えても、最後に残った穴があった。



それは、技術ではなく、私たち**人間の側の穴**だった。

第6部

落とし穴

[人・運用・マインド]

設計者の椅子は、
まだ我々のもの

見えてきた本質④

最も見落とされる、
でも最も重要な
“人間の穴”



人



運用



マインド

設計者の
椅子



最新のシステムも、人と運用とマインド が伴わなければ、必ず崩れる。

【落とし穴24】「”パイプラインも自動設計できる”と過信した。」

生成・検証・監査が回り始めると、“AIがすべて最適に設計してくれるはずだ”と期待が膨らんだ。

症状



生成・検証・監査がスムーズに回り、「課題ごとに最適なパイプラインも、AIが自動で設計してくれるはず」と期待。研究の設計判断まで、AIに委ねかけた。

正直な現在地



いまのAIRAでも、“課題（研究テーマ）ごとに最適なパイプラインを自動で設計する”ところまでは行きついていない。

現状は、人間がパイプラインを設計し、AIが忠実に回す。

見えた本質



AIは“実行者”としては優秀。だが、“設計者”の椅子は、まだ研究者の手に残っている。

AI for Science の本質は、人間が設計し、AIが実行する分業にある（少なくとも現時点では）。

なぜ起きたか

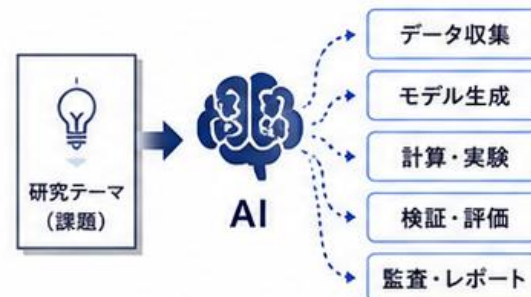


AIの到達範囲（Reach）が広がるほど、“すべてを任せたい”心理が働く。自動化できる範囲を、過大評価してしまった。

期待した未来 VS. 正直な今の姿

期待した未来（過信）

AIが研究テーマを理解し、最適なパイプラインを自動で設計・構築・実行



すべてをAIが最適設計し、自律的に実行

正直な今の姿（現実）

研究者が設計し、AIが忠実に回す



設計に基づいて、AIが忠実に実行・検証・監査

VS.

過信のリスク



目的のズレ

本質に合わない設計で、誤った方向へ進む



ブラックボックス化

設計意図が不明確になり、説明責任を果たせない



研究の自律性の喪失

研究者の思考力・判断力が弱まり、主導権を失う



メッセージ

自動化できる範囲を、過大評価しない。設計の責任は、まだ手放せない。

【落とし穴25】「問いを立てる仕事まで、AIに明け渡しかけた。」

実行が楽になると、人間が“問いを立てる”中核まで委ねたくなる誘惑。

症状



AIが候補も仮説も次々出してくるので、「何を問うべきか」の判断まで、AIの提案に流されかけた。研究の主導権が、静かに移りかけた。

原因



実行コストが下がると、人間は“考えること”自体を省略したくなる。Reachが広がるほど、**問いの選択を外注したくなる**引力が働く。

対策／本質



問い・仮説・破綻条件・最終判断（採択/保留/棄却）は、人間が握る。
AIは選択肢を広げる増幅器であって、選択の主体ではない。

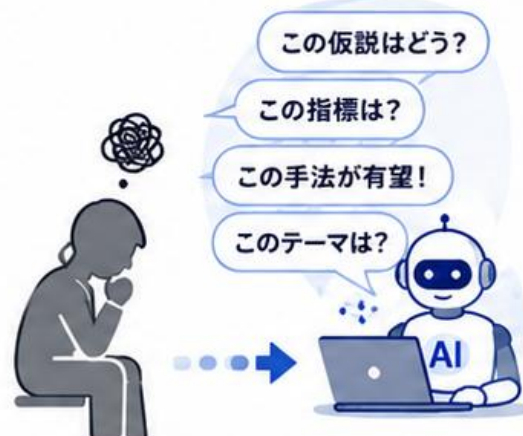
見えた本質



AI for Science の本質は、“**良い問いを立てる人間**”を中心に置くこと。
AIは問いの探索空間を広げるが、どこに賭けるかを決めるのは研究者。

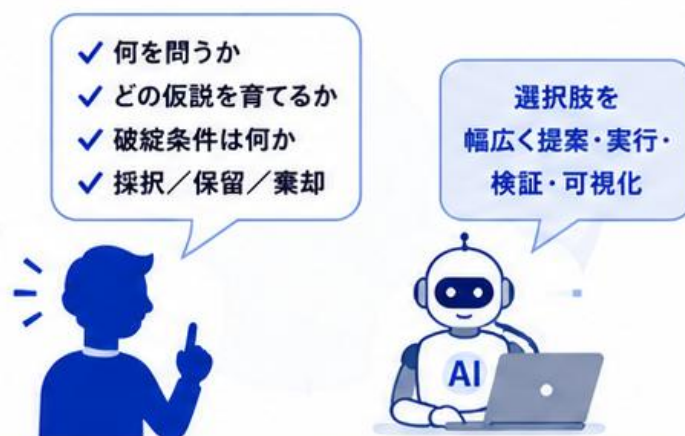
AIに任せすぎると、研究の主導権が移る

任せすぎの状態



問いの選択を外注 →
研究の主導権が静かにAIへ

あるべき姿（人間が握る）



人間が問いの主権を握り、
AIは増幅器として支援

VS.



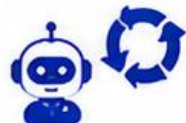
AIは答えの幅を広げる。
だが、何を問うかは、**研究者の主権**である。

➡ 第1部のギャップで言えば、**設計・問い（Reach/問う側）**は、ここでも**研究者の側**に残るべき核心である。

【落とし穴26】「自走するAIは、放置すると”財布”も溶かす。」

自律パイプラインの“快適さ”の裏で、コストが静かに膨張していく、運用上の落とし穴。

症状



自律パイプラインが回り続けると、API呼び出し・トークン消費・計算資源が、気づかぬうちに膨張。
価値の薄い再生成・再試行にコストが流れた。

原因



「自動で回る」ことの快適さが、コストの可視化と上限設計を後回しにした。
スキルの粒度ミス（落とし穴8）も、呼び出し爆発でコストを押し上げた。

対策



コストの計測・上限・打ち切り条件をパイプラインに組み込む。
価値の低い再試行を **early-stop** する。
粒度・呼び出し回数を **監視** する。

見えた本質



AI for Science の Reach は無限ではなく、計算資源という現実の制約の中にある。
再現可能性に加えて、**持続可能性（コスト規律）** も、本質の一部だった。

自走パイプラインのコストは、気づかぬうちに膨らむ



自律パイプラインが回り続ける



候補生成
(多数パターン)



検証・再試行
(ループ)



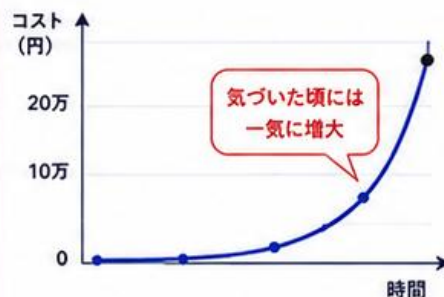
再生成・比較
(大量トークン消費)



気づいたときには
コストが膨張



コストの推移イメージ(例)



主なコスト増の要因

- ✓ 価値の低い再生成・再試行の繰り返し
- ✓ 広すぎる探索空間（粒度ミス・分解過多）
- ✓ 不必要なAPI呼び出しの連鎖
- ✓ 中間成果物の冗長な生成・保存

例：1日あたりのコスト内訳（イメージ）



コスト規律を、パイプラインに組み込む



計測

・トークン・API・計算時間をリアルタイム計測
・可視化ダッシュボード化



上限設定

・1案件あたりの上限額
・日次・週次の上限
・予算アラート



打ち切り条件

・改善が見られない場合
・期待値を下回る場合
・上限に近づいた場合



early-stop

・価値の低い再試行は早期に停止
・探索の幅を制御



監視

・粒度・呼び出し回数
・ループの深さ
・異常パターン検知



Reachを広げる自由には、コストという重力がかかる。規律なき自走は、財布から崩れる。

【落とし穴27】「うまくいったプロンプトが、二度と再現できない。」

成功が“職人芸”として個人の頭の中にだけ残り、共有も再現もできない属人化の落とし穴。

症状



たまたまうまくいったプロンプトや手順が、職人芸として個人の頭の中にだけ残る。共有も再現もできず、他の人が回すと品質が再現しない。

原因



プロセスを“OS (明示的なSkill・ルール・パイプライン)”として外在化せず、個人の暗黙知に留めてしまった。

対策／本質



成功した手順を Skill・ゴールデンルール・パイプライン)として明文化し、誰が回しても同じ品質が出る形に外在化する = OS 化。

見えた本質

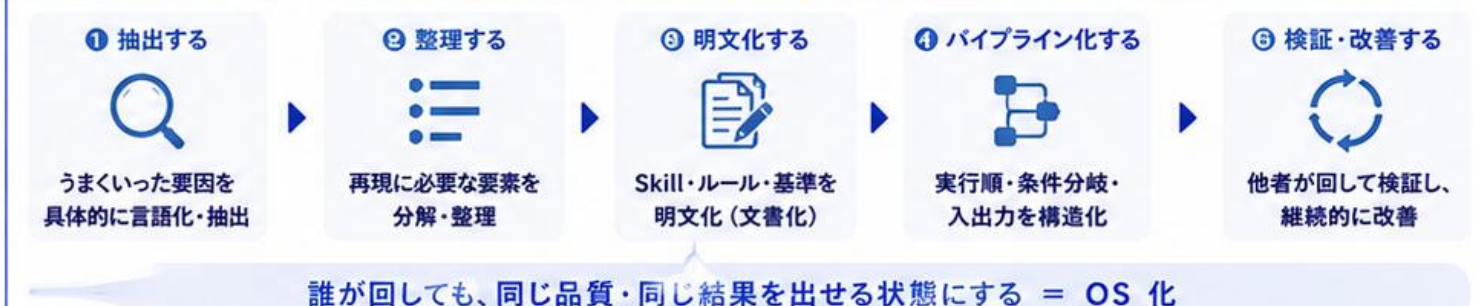


Reach (到達範囲) は、職人芸では広がっても“再現”できない。AI for Science の本質は、Reachを再現可能にすること。そのために OS が要る。

属人化 (OS化できていない) vs. OS化 (再現可能なプロセス)



成功した手順を OS 化するステップ



一回の成功は、まだ科学ではない。誰もが何度でも再現できて、初めて科学になる。

【落とし穴28】「“AIがそう言った”は、研究者の言い訳にならない。」

AIの出力は強力な助けになるが、最終的な判断と責任は研究者自身が負う。

症状



AIの出力をそのまま採用しそうになり、誤りがあったときに「AIの判断だった」と責任を曖昧にしたいくなる。

原因



自律性が高まるほど、成果と責任の帰属が曖昧に。生成物が“第一稿”であることを忘れ、最終成果物と同一視してしまう。

対策／本質



生成物は常に“第一稿”。実験的妥当性・科学的判断・最終責任は、必ず研究者が担う。
これを制度（テンプレートの最終判断欄・監査）として固定する。

見えた本質



AI for Science の本質は、責任の主体を人間に置き続けること。再現可能性も監査も、最後は「人間が責任を取れる形」にするための仕組みだった。

AIは共同実施者、だが責任の主体は研究者



AIは判断を支援するが、決めるのは人間。
AIの出力は“材料”であり、“判断”ではない。

制度として固定する（テンプレート例）



最終判断欄（必須）

- ✓ AI出力の要約
- ✓ 根拠となる検証結果
- ✓ 採択／保留／棄却の理由
- ✓ 残された不確実性・限界
- ✓ 追加で必要な検証
- ✓ 最終責任者（研究者）署名

最終責任者（研究者）署名：_____ 日付：__/__/__



監査による裏付け

- ✓ 出典照合・再現性監査
- ✓ 手法・前提の妥当性監査
- ✓ 数値・図表の整合性監査
- ✓ 倫理・コンプライアンス監査

監査があるからこそ、研究者が責任を取れる。



AIは共同実施者だが、署名するのは研究者。
責任は、増幅も委譲もできない。



責任の委譲はできない



成果の帰属も譲れない

【落とし穴29】「AIに任せれば安心——増幅するものが無ければ、出力もゼロ。」

AIへの過度な依存が、研究者自身の“芯”を空洞化させる落とし穴。

症状



「AIに任せておけば大丈夫」という安心感で、自分のドメイン知識・問い・批判的思考を磨くのを止めてしまう。気づいたときには、**判断力が鈍り、成果の質も頭打ちに。**

原因



AIを“代替”だと誤解した。
AIは増幅器 (amplifier) であり、
入力(研究者の知識・問い)がゼロなら、
出力もゼロになることを見落とした。

見えた本質 (核心)



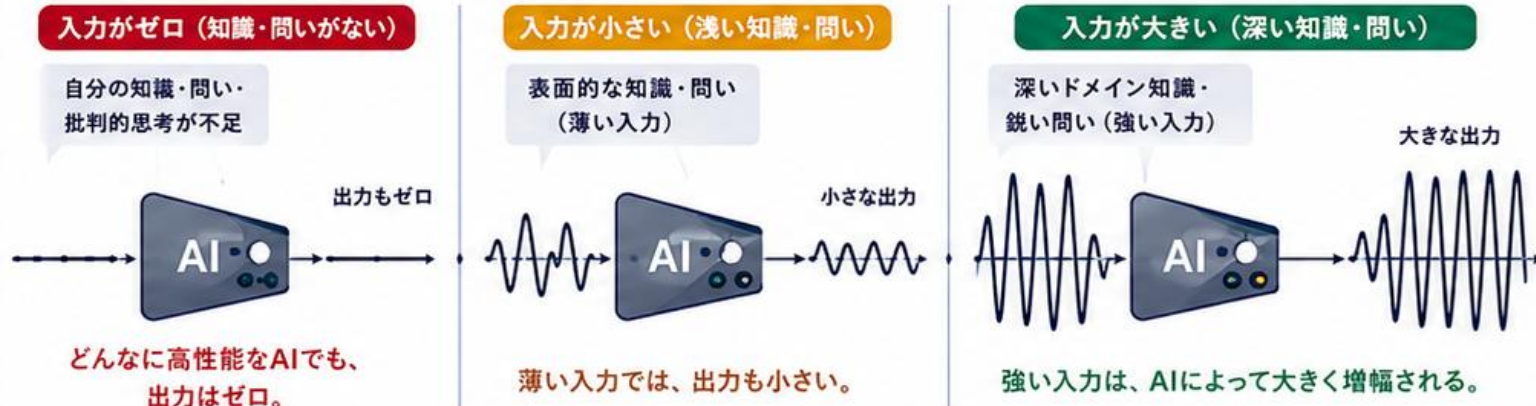
AIは研究者を代替しない。**増幅する。**
増幅元である研究者のドメイン知識と問いが、
AI for Scienceの本質の中心。
AIが強力になるほど、それを使う研究者の“芯”が、
より重要になる。

メッセージ



AIが優秀になるほど、皮肉にも、
研究者自身の専門性と問いの質が、
決定的になる。

AIは増幅器 (アンプ)。入力がゼロなら、出力もゼロ。

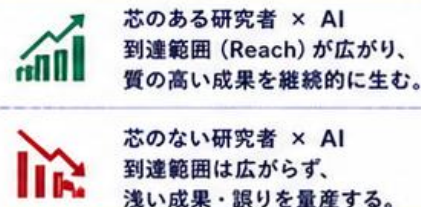


増幅器がどれだけ優れていても、入力がなければ意味がない。
“増幅元”である研究者のドメイン知識と問いが、成果の質を決める。

AI時代に、研究者が磨き続けるべき“芯”



結果の差は、決定的になる



メッセージ

AIが優秀になるほど、研究者自身の**専門性と問いの質**が、成果のすべてを決める。

AIに任せて自分を空っぽにした人から、出力はゼロになっていく。

【落とし穴30】「最後まで、私は“速さ”の誘惑から逃れられなかった。」

30個の落とし穴の最終章。最初の穴（Speedの幻想）に呼応する、最も根深い人間側の落とし穴。

症状



仕組みを整えてもなお、「もっと速く、もっとたくさん」という誘惑が繰り返し戻ってくる。速さを成果と錯覚し、Reachへの転換を何度も遅らせた。

原因



Speedは即座に報酬を感じられる（速い＝気持ちいい）。Reachは効果が見えにくく、規律を要する。人間は、即時報酬に引きずられる。

対策／本質



価値の中心を、意識的にSpeedからReachへ置き換え続ける。問うべきは「どれだけ速く作れたか」ではなく「どれだけ速くの問いに、反証可能に手が届いたか」。

見えた本質



30の落とし穴の最深部にあったのは、技術ではなく、私自身の価値観だった。AI for Scienceの本質を阻む最大の壁は、“速さへの欲望”という人間の性（さが）。

Speedの誘惑と、Reachへの転換

Speedの誘惑（即時報酬の快楽）

速さは、すぐに気持ちいい。



すぐに結果が出る



達成感が即座に得られる



短期の報酬につながる



繰り返し戻ってくる

VS.

Reachへの転換（本質的な価値）

Reachは、地味だが本質的な価値を生む。



速くの問いに到達する



反証可能な知見を生む



蓄積が効き、価値が増幅する



再現性と信頼を残す

速さ＝成果と錯覚し、研究の本質（Reach）からズレていく。

本質的な価値は、時間をかけて確かな成果として残る。

自分に問うべき問い

「どれだけ速くの問いに、反証可能に手が届いたか？」

価値をSpeedからReachへ置き換えるための行動



目的を再確認

本当に解きたい問いは何か？



Reach指標を定義

到達範囲・反証可能性・再現性を指標にする



短期より長期を選ぶ

地味でも本質的な積み重ねを選ぶ



Speedに流されたら立ち止まる

一度立ち止まり、問いに立ち返る



仲間と共有・問い直す

チームで価値の軸を握り直す



到達を記録し、価値を残す

Reachの足跡を残し、未来の自分へ手渡す

研究者の選択で、結果は分かれる

Reachを選ぶ研究者



到達範囲（Reach）が広がり、反証可能な知が蓄積し、社会に残る成果を生み出す。

Speedに溺れる研究者



速さに追われ続け、浅い成果の繰り返しで、本質的な価値を残せない。



メッセージ

最後の落とし穴は、AIの中ではなく、私の中にあった。

AIが優秀になるほど、皮肉にも、研究者自身の専門性と問いの質が、決定的になる。

最後に残ったのは、技術の穴ではなく、“人間の椅子”だった。

第6部（落とし穴24～30）を貫く発見＝見えてきた本質④

共通する発見

技術をどれだけ磨いても、最後に残る判断・問い・責任・価値観は、人間の領分だった。

① 問いを立てる椅子

何を問うか決める



- ✓ どこに賭けるか、どんな仮説を立てるかを決める
- ✓ AIは候補を広げるが、問いの選択は研究者の主権
- ✓ 問いの質が、到達範囲（Reach）を決める

落とし穴25・29

② 設計者の椅子

パイプラインを設計する



- ✓ 研究課題に合わせて最適なパイプラインを設計する
- ✓ どの順序で、どの粒度で、どのスキルを使うか決める
- ✓ AIにはまだ自動設計の椅子は渡せない

落とし穴24

③ 責任を取る椅子

最終判断と署名をする



- ✓ 妥当性の検証と科学的判断を行う
- ✓ 採択／保留／棄却の最終判断を下す
- ✓ 論文への署名など、最終責任を担う
- ✓ 責任の主体は、最後まで人間

落とし穴28

④ 価値を選ぶ椅子

SpeedかReachかを選ぶ



- ✓ 何を価値の中心に置くかを決める
- ✓ Speed（速さ）か、Reach（到達範囲・反証可能性）か
- ✓ 研究の方向性と未来への投資を決める

落とし穴30



AIが実行を担うほど、人間は“より上流の、より本質的な仕事”へ移る。

それは仕事を奪われることではなく、**仕事の重心が上がる**こと。



メッセージ

AIは椅子を奪わない。
むしろ、私たちを、より本質的な椅子へと、**押し上げる**。

第1部のギャップに対応する“最後の椅子”

全国が Speed（速さ）へ傾くなかで、どちらを“主”に賭けるかを決めるのは、最後まで**人間の椅子**に残る判断。



4つの本質が、出そろった。最後の第7部で、これを一つに束ねる。

第7部

本質の結論

—— Speedではなく Reach /
AIは増幅する

30の落とし穴を越えて、たどり着いた結論。

AI for Science の本質は、

“到達範囲 (Reach) を、再現可能に広げること”。

AIは研究者を代替しない。増幅する。

勝負すべきは、Speedではなく Reach。



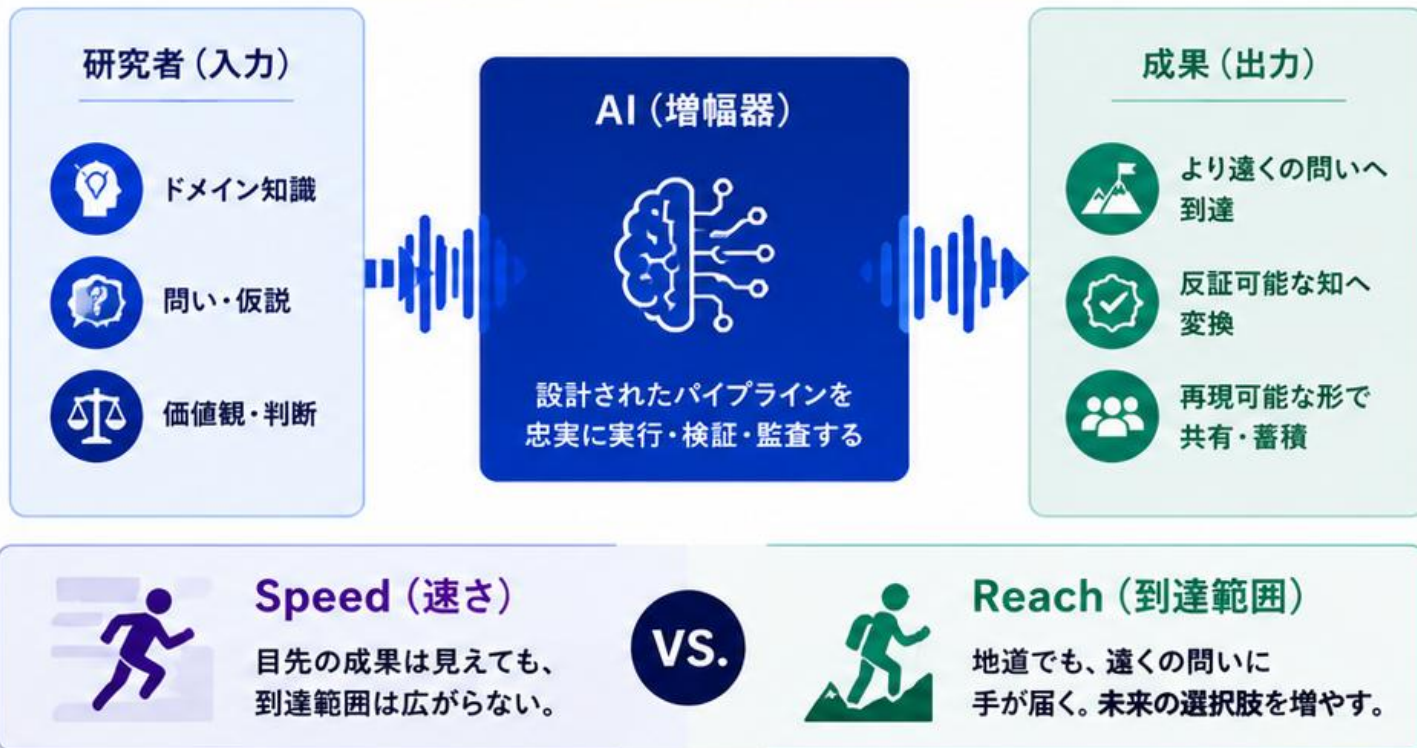
メッセージ

AIは研究者を代替しない。増幅する。

勝負すべきは、Speedではなく Reach。

AI for Science の本質

到達範囲 (Reach) を、再現可能に広げること



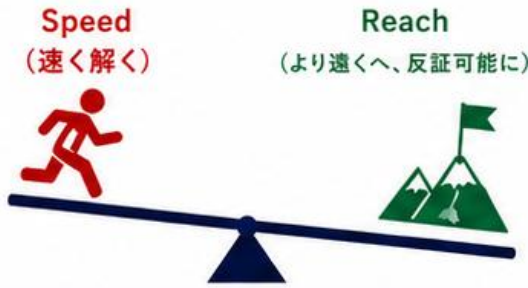
30の落とし穴は、4つの本質に収束した。

2,000回の実験が照らし出した、AI for Science の本質マップ



第1部で示したギャップの回収

全国456件が **Speed=**
“速く解く”側に傾くなか、
2,000回の実験は
一貫してもう一方の側を指した。



4つの本質は、すべてこの「私の側 (Reach/検証)」
から出てきた。

- ✓ 速さの誘惑からの脱却
- ✓ 到達範囲 (Reach) の拡張
- ✓ 反証可能性の確保
- ✓ 人間の本質的役割の再定義

第2～6部で拾い集めた 30の落とし穴 (24～30含む)

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

共通項を抽出・統合すると…

本質① (第3部)

品質を決めるのは
“賢さ”ではなく
“プロセスの設計”。



対応する落とし穴
1～6

本質② (第4部)

道具の所有ではなく
“接続・編み込み
(オーケストレーション)”
が効く。



対応する落とし穴
7～16

本質③ (第5部)

生成能力ではなく
“反証可能性を保つ
監査”が希少。



対応する落とし穴
17～23

本質④ (第6部)

最後に残るのは
“人間の椅子”
(設計・問い・責任・価値)。



対応する落とし穴
24～30

さらに、一つの軸へ収束

Speed
ではなく
Reachへ

+

AIは
代替でなく
増幅する

AIは研究者を代替しない。
増幅する。
勝負すべきは、**Speed**
ではなく **Reach**。



メッセージ

30回転んで掘り当てた本質は、驚くほどシンプルだった。

次の4ページで、その核心をお見せします。

AI for Science の本質は、Speed ではない。Reach だ。

本質の核心①：Speedではなく Reach



Speed (速さ)

同じことを、速くやる



- ✓ 今までと同じ問い・同じ手順・同じスコープ
- ✓ 速く終わるが、到達範囲は広がらない
- ✓ 短期的には気持ちよく、長期的には消耗する

Reach (到達範囲)

今まで届かなかった問いに、手を伸ばす



- ✓ これまでの外側にある問い・手法・データへ
- ✓ 検証しきれなかった領域に、手が届く
- ✓ 間違いを早く棄却し、より広い探索を歩ける

Reach の定義

- 1 これまで検証しきれなかった「仮説の数」に手が届く
- 2 自分の専門の外側にある「分野・手法・データ」に手が届く
- 3 間違った仮説を“早く棄却”できるからこそ、より広い探索空間を歩ける



AIは、私を速くしたのではない。私の“届く範囲”を広げたのだ。

第3部 (落とし穴6)

Speed の幻想：速ければ勝てる、と思い込んだ。



第6部 (落とし穴30)

Speed の誘惑：最後まで、私は“速さ”に溺れた。



速さは消耗する。
Reach は、研究者の地平そのものを
押し広げる。

★
第1部のギャップ
への最終回答

第1部で、全国456件の重心は Speed = “速く解く”側にあると見た。
2,000回転んで私が掘り当てた答えは、その反対側—— Reach だった。

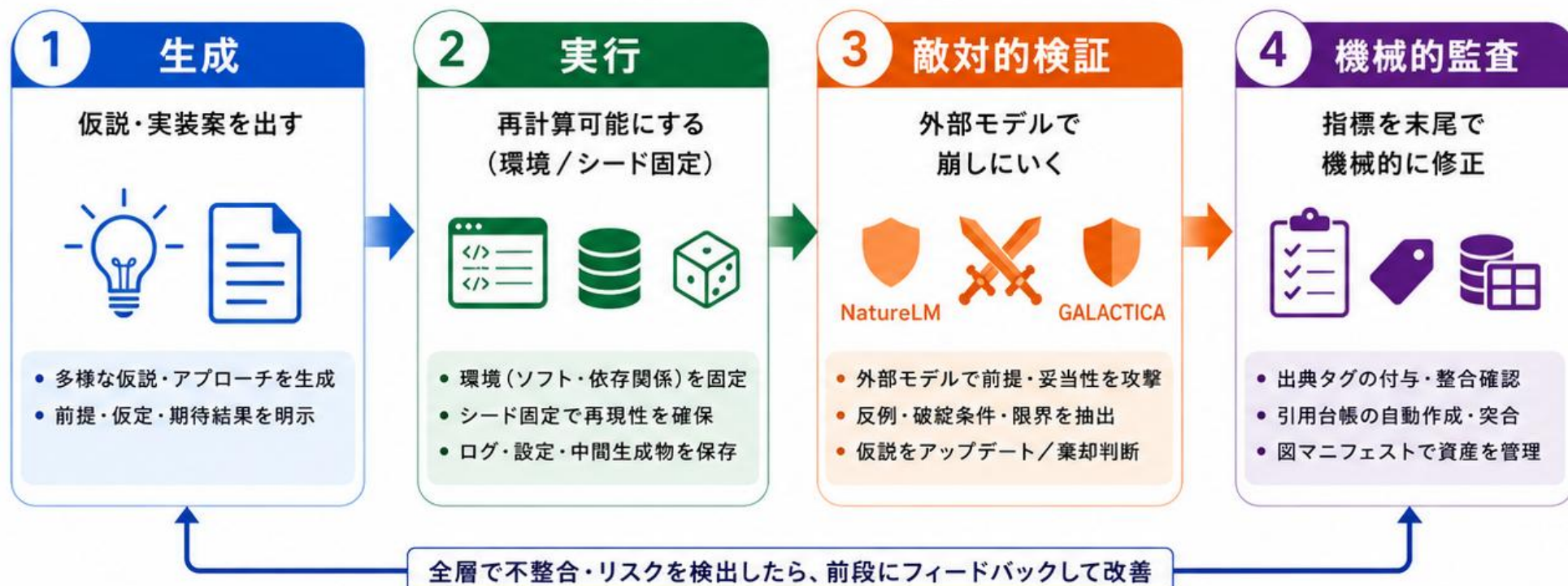
Speed と Reach は対立しない。だが、どちらを“主”に据えるかで、研究の設計はまるごと変わる。

Reach を支えるのは、“自分を疑う経路”の設計だ。



“ AI for Science の本質は、生成能力ではなく、**反証可能性を維持するプロセス設計**にある。 ”

出力が安定した「4層」のプロセス設計



今日のすべての対策は、
この4層のどこかに収まる

- ✓ 環境・シードの固定
- ✓ 出典タグの付与
- ✓ 引用台帳の整備
- ✓ 敵対的検証
- ✓ 独立監査 (機械的監査)
- ✓ 図マニフェスト
- ⋮

守るべきは、
「速く書けること」ではない。
間違った仮説を、
できるだけ早く棄却できること。
それが、遠くへ届くための足場。



Reachは、闇雲に手を広げることではない。
広い範囲に手を伸ばしても崩れないために、**反証可能性を、プロセスに作り込む。**

Reach を“再現可能”にするには、科学のためのOSが要る。



問題提起



反証可能なプロセスや知識発掘を、一回ごとに手作りしては、属人的な職人芸で終わる。
Reachは広がっても“再現”できない。

落とし穴27の回収
「属人化の罠」

主張

研究プロセス全体を支える共通基盤 = AI for Science のための **OS (Skill × MCP)** が必要。

OS が担う役割 (AIRAで実装)



知識発掘

- 論文の海から価値ある知識を掘り出す (GraphRAG)



仮説生成・実行・再計算

- 仮説・実装案の生成
- 再計算可能な実行 (環境/シード固定)



敵対的検証・機械的監査

- 外部モデルで崩しにいく (NatureLM/GALACTICA)
- 指標で機械的に監査・修正



出典・再現性の担保

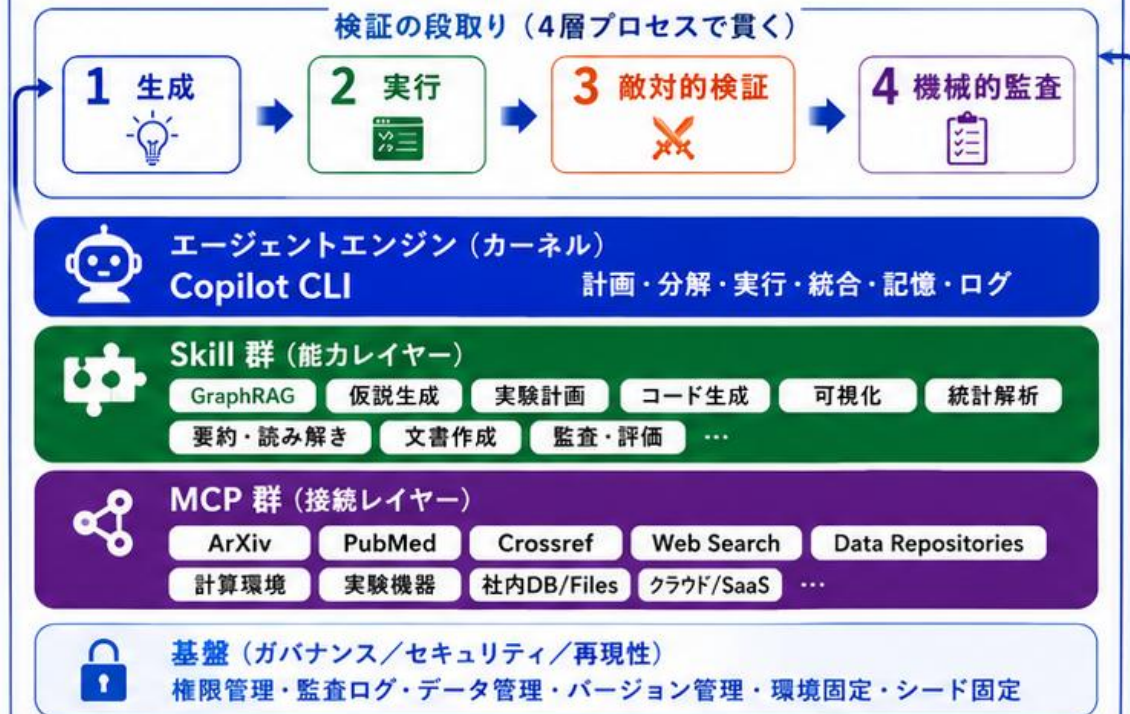
- 出典タグ・引用台帳・図マニフェスト
- 再現性 (コード・環境・ログ) の保存



Skill・MCP・エージェントの協調

- 多様なSkillと多様な外部接続を統合
- 複数エージェントの協調を最適化

OS の構造 (AIRAアーキテクチャ)



OS 化の効果



再現できる

誰もが、何度でも、同じ手順で同じ品質にたどり着ける。



拡張できる

新しいSkillやMCPを接続するだけで、能力と到達範囲が拡張。



属人化しない

手順・記録・監査が標準化され、知見が組織の資産になる。



信頼できる

反証可能性・出典・再現性がOSレベルで担保される。



協調できる

複数のエージェントやツールが安全に協調して成果を最大化。



OS 化が、職人芸だったReachを、“誰もが何度でも再現できる力”に変える。

AIは研究を“代替”しない。あなたの Reach を“増幅”する。



AI for Science は、研究者を置き換える技術ではない。

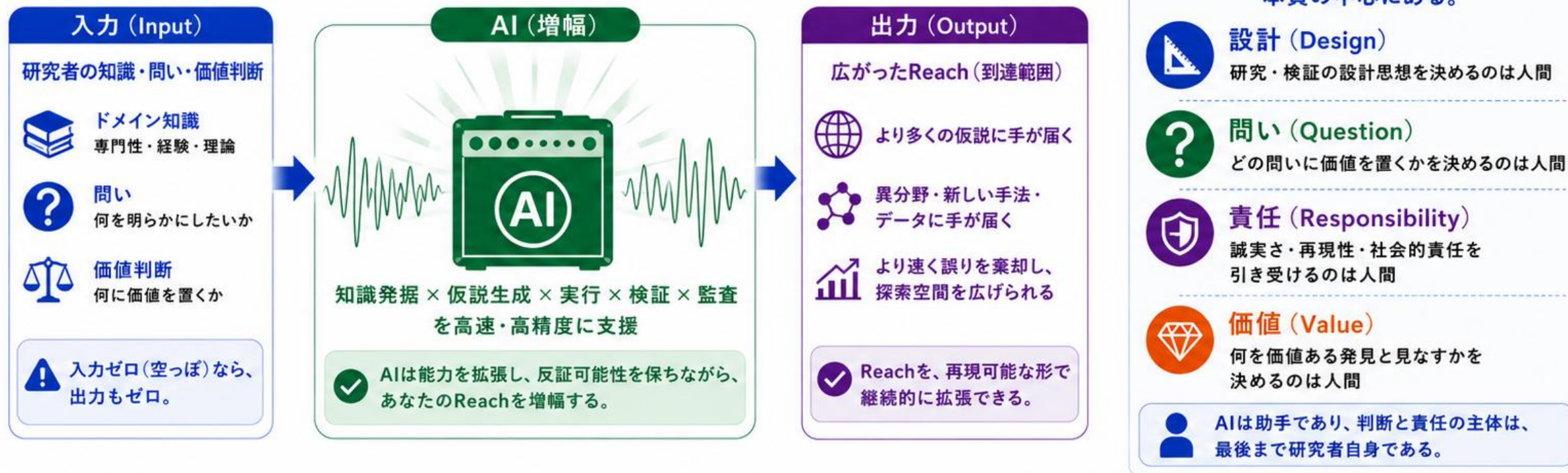
増幅するものがなければ、出力も**ゼロ**。

AIは、あなたのドメイン知識と問いを増幅し、Reachを再現可能に広げる。

落とし穴28・29の回収

- ✓ 28: AIに代替された気になる
- ✓ 29: 手抜き・丸投げに流れる

AIは「増幅器 (amplifier)」である



AIが優秀になるほど、皮肉にも、研究者自身の**専門性と問いの質が、決定的になる**。

AIは、あなたを不要にしない。あなたの中にある、いちばん大事なものを、**増幅**する。



私にとって AI for Science とは — “遠くの問い”に、反証可能に手を伸ばす営み。

AI for Science の 3 つの定義を重ねて、私の言葉に束ねる。



DOE

(U.S. Department of Energy)

発見を加速する
科学の新しいかたち。



文部科学省
(文科省)

研究の全工程への
AI 導入による
科学研究の変革。



登壇者 (私)

Speed (速さ) ではなく **Reach (到達範囲)** を広げ、
反証可能性を保ったまま、これまで届かなかった
問いに手を伸ばす営み。

その Reach を **OS (Skill × MCP)** で再現可能にし、
AI は研究者を **増幅** する。

私の定義が目指す世界観



同じことを
速くやる
= 矢印の世界



反証可能性
の担保

+



OS化
(Skill × MCP)

+



再現可能性
の実現

+



研究者の
増幅



これまで
届かなかった
問いに
手を伸ばす
= 広がる円の世界



これは借りものの定義ではない。2,000回転んで、穴の底から、**自分で掘り当てた定義だ。**



賢いモデルより、この“1枚”から始めてください。

30の落とし穴の対策を、研究の最初から最後まで貫く“実践テンプレート”に凝縮しました。

仮説検証テンプレート（1枚で完結）

- 1  仮説：
- 2  この仮説が破綻する条件：
- 3  主要評価指標（事前に固定）：
- 4  分割戦略（事前に固定）：
- 5  前処理／特徴量の変更ログ：
- 6  外部モデル（NatureLM／GALACTICA 等）からの反論：
- 7  反論に対する追加実験：
- 8  最終判断（採択／保留／棄却）：
- 9  備考・学び（再発防止のメモ）：

このテンプレートの設計意図（3点）

- 1  生成の“前”に、
評価指標と分割戦略を固定する。
先にルールを決めることで、
後出しの都合や過学習を防ぐ。
落とし穴 4・21 への対策
- 2  「破綻する条件」を先に書く＝
反証可能性をテンプレートに作り込む。
自分に都合の良い仮説に偏らず、
最初から“崩れ方”を設計する。
落とし穴 1 への対策
- 3  外部モデルに“わざと反論させる”
欄を常設する。
内輪の合意で終わらせないための、
敵対的検証を標準装備にする。
落とし穴 5・23 への対策



この1枚を、最初に埋める。途中で更新する。
すべての記録を、出典タグとともに残す。



このテンプレート、ぜひ **スマホで撮ってください。**

30回転んで残った、**たった1枚**です。



再現性を守る、3つの鉄の掟。

★ テンプレート（プロセス設計）と対をなす、実装レベルの実践ルール。AIRAのすべてのフェーズに適用。

Rule 1 再現性

ノートブックの最初に、
環境セルとシードセルを
実行する。

1 環境セル（バージョン記録）

- Python / ライブラリ版本
- OS / ハードウェア情報
- 実行日時 / ユーザー情報

```
# cell: env
import platform, sys, pkgutil
import pandas as pd
import numpy as np
env = {
    "python": sys.version,
    "platform": platform.platform(),
    "packages": {p.module_finder.name:
        p.module_finder.version
        for p in pkgutil.iter_modules()}
}
pd.DataFrame(list(env["packages"].items()),
    columns=["package", "version"])
```

2 シードセル（乱数固定）

- numpy / random / torchなどを固定
- 毎回同じ結果を再現可能にする

```
# cell: seed
import random, numpy as np
random.seed(20250529)
np.random.seed(20250529)
try:
    import torch
    torch.manual_seed(20250529)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
except ImportError:
    pass
```

✓ 落とし穴 2・9・16・24 への対策

Rule 2 追跡可能性

レポートの全定量値に、
出典タグと [cell:<id>] を付与する。

レポート記載例（出典タグ付き）

Accuracy = 0.842	[cell:42]
RMSE = 1.37	[cell:47]
p-value = 0.0081	[cell:53]
Δ AIC = -12.6	[cell:58]
...	



出典タグのルール

- ✓ セルIDは固定（例: cell:42）
- ✓ 計算の直後に値とタグを記録
- ✓ 加工・集計後も元セルに遡れる

cell:42 → cell:47 → cell:53 ...

モデル推定 集計 統計検定

✓ 落とし穴 4・6・11・17・21 への対策

Rule 3 整合性

レポート作成前に、引用台帳と
図マニフェストを作成する。

1 引用台帳（実在照合）

- すべての引用を実在照合
- DOI / URL / 出版情報を記録
- 照合結果（OK / NG）も明記

ID	著者・年	DOI/URL	照合	備考
R001	Smith+2023	10.xxxx/...	OK	...
R002	Doe+2022	https://....	NG	要確認
R003	...	...	OK	...
...	...	...	...	...

2 図マニフェスト（主張と図の実合）

- 各図の主張・根拠・出典を対応付け
- 本文中の番号／参照先を明確化
- 幽霊図（本文だけの図）を防止

図ID	図タイトル	主張(What)	根拠(Why)	出典/セル	本文参照
Fig.1	...	...	...	[cell:15]	p.5 L.12
Fig.2	...	...	...	[cell:28]	p.6 L.4
Fig.3	...	...	...	[cell:33]	p.7 L.1
...	...	...	...	...	...

✓ 落とし穴 5・12・23・28・29 への対策

この3ルールで検出・防止できる
代表的問題（AIの三大欠陥）

1 Value Mismatch（数値不整合）



表・図・本文で数値がズれる。
出典セル不明で原因を特定
できない。

2 Ghost References（幽霊引用）



存在しない論文・DOI・URL
を引用してしまう。
信頼が一瞬で崩壊する。

3 Figure Orphans（幽霊図）



本文に図の説明だけがあり、
図が存在しない／参照が
不明になる。

- ✓ 機械的にチェックできる形にすることで、
人の見落としをゼロに近づける。
- ✓ 再現性・追跡可能性・整合性の三位一体で、
AI特有の嘘を撃ち落とす。



テンプレートが“プロセスの設計”なら、この3ルールは“実装の規律”。

両輪で、AI特有の嘘を撃ち落とす。



正直に言います — パイプラインの自動設計は、まだ未踏です。

⚠ 誠実な但し書き：AIRAでも、“課題ごとに最適なパイプラインを自動で設計する”ところまでは行きついていません。

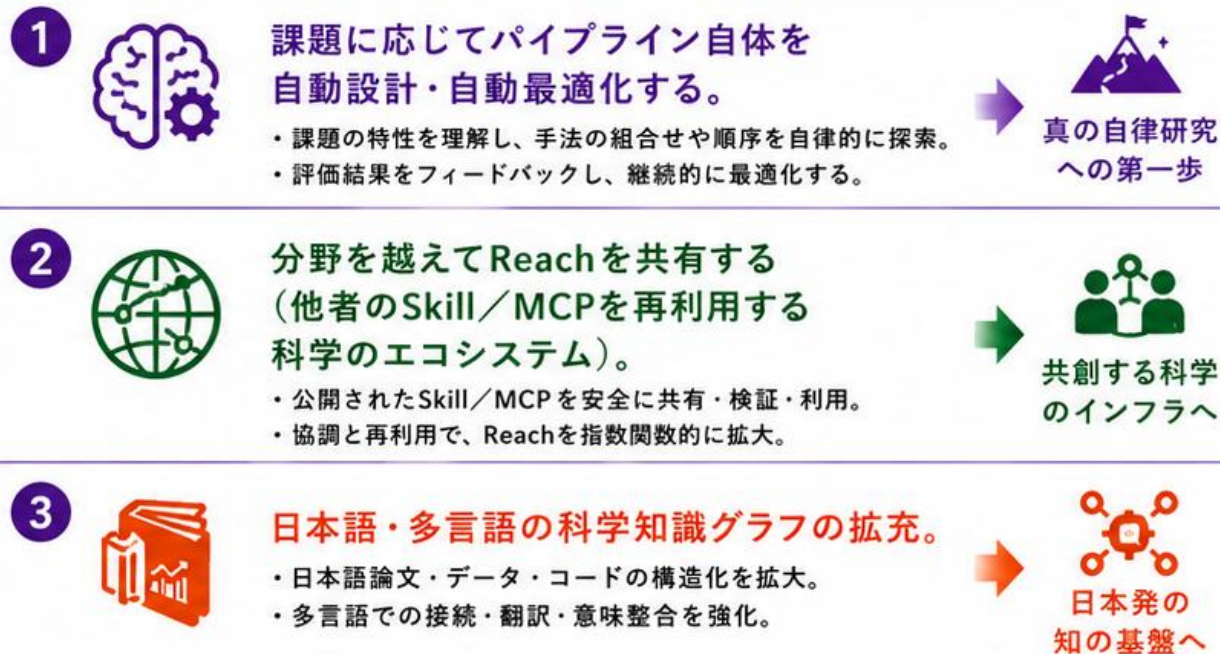
現在地：AIRA の実力（できていること）

現状は、人間が設計し、AIが忠実に回す。
（設計者の椅子は、まだ私たち研究者のもの）



💡 落とし穴 24 の回収：設計者の椅子は、まだ研究者の手に残っている。

未踏のフロンティア（これから一緒に拓きたい領域）



★ やるべきことは、山ほどあります。だからこそ、面白い。



AIは“**実行者**”としては優秀。だが“**設計者**”の椅子は、まだ私たち研究者の手に残っている。
ここが、次の挑戦の場です。



この未踏地に、ぜひ皆さんと一緒に挑みたい。
科学のReachを、反証可能に、再現可能に、そして、誰もが使える力に。



知を開く →



創り合う →



遠くへ届く

共に拓く未来が、ここから始まります。



読めなかった論文の海は、もう、歩ける地図になった。

知識獲得フェーズ（落とし穴10～16）で起きた、決定的な変化。

Before：論文の海で、知は埋もれていた



読みきれない論文の海

論文は増え続け、
時間も体力も追いつかない。



知見は埋もれたまま

論文同士のつながりが見えず、
重要な知見は点のまま散在。



LLMは根拠なく答える（幻覚）

出典をたどれず、もっともらしいが
証拠のない答えを生成してしまう。



After：ナレッジグラフが、海を地図に変えた



ナレッジグラフが海を地図に変える

論文・概念・手法・データを関係で結び、
全体像を可視化。



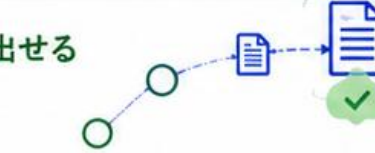
論文をまたぐ関係をたどれる

マルチホップで因果・手法・比較などを
横断し、つながりを発見。



出典をたどれる“根拠ある知”を引き出せる

各ノードは出典にももつき、
根拠までさかのぼれる。



実証の数字

（自作GraphRAG）



aira-synapse
aira-graphdb

HotpotQA

（マルチホップ推論）

意味一致（Exact Match）

91.2%

日本語（JGLUE / JaNLI）

78.3%

✓ 論文ベースのマルチホップで高精度を実現

✓ 日本語でも実用レベルに到達



Reachを広げる最大の戦場＝論文の海。

そこを、点（バラバラの論文）から線（関係の地図）へ変えたとき、
届かなかった問いに、手が届いた。

点：バラバラの論文



関係をつなぐ

線：関係の地図



これは、2,000回の失敗（落とし穴10～16の壁）を乗り越えて、ようやく見えた光です。次のフロンティアは、さらにその地図を広げること。

教員には“研究の再設計”を。学生には“先に旗を立てる機会”を。

AI for Science は、ベテランの経験も、若い挑戦心も、どちらも武器になる稀有な領域。

🏠 教員へ

- ✓ 研究プロセスを、Skill と MCP で“再設計”してみる。
30 の落とし穴の地図は、その第一歩の安全装置。



- ✓ SPReAD / ARiSE を研究費・新領域開拓・共同研究の起点に。
新しい研究費で、未知への扉を開き、学内外の連携を加速する。



- ✓ 学生を AI for Science でどう育てるか（教育の再設計）。
AI時代の研究力・発想力・検証力を育む教育設計を、一緒に。



🎓 学生へ

- ✓ AI スキルは、分野を問わない強力なキャリア資産。
研究・開発・データ・医療・ものづくり…どの道にも通用する武器になる。



- ✓ SPReAD は学生も応募可能！
(年間 1,000 件・最大 500 万円)
＝「先生より先に旗を立てる」チャンス。
自分の問いを形にし、未来の研究をリードしよう。



- ✓ 専門の枠を越えて、未踏の問いに挑める時代。
分野を越える好奇心こそが、新しい発見を生む原動力になる。



AI for Science は、経験の深さも、挑戦の速さも、**どちらもそのまま武器**になる稀有な領域です。
今日の 30 の落とし穴の地図は、皆さんが安心して挑戦するための地図です。



ご協力
ください！



“教員として” 関心がある方
手を挙げてください！

or



“学生・若手として” 関心がある方
手を挙げてください！



あの夜、答えられなかった問いに — 今なら、答えられる。

伏線の回収：「この主張は、どの計算が支えているのか？」に、今なら、根拠をもって答えられる。

あの夜 (Before)

完璧に見えたドラフト。だが、答えられなかった。



私の問い
「この主張は、
どの計算が支えているのか？」



出典と計算の結びつきが不明
主張の根拠が、論文と計算の
どこにあるのか特定できない。



根拠のない“もっともらしさ”
AIに聞いても、根拠なく、
もっともらしい答えが返る（幻覚）。



再現できない・検証できない
手順も環境も再現不能。
信頼できるか、自信が持てない。

結局、誰にも説明できず、発表も投稿もできなかった。



今 (After)

地図がある。根拠がある。検証できる。だから、答えられる。



論文の海を、グラフという地図に変える
ナレッジグラフが論文・概念・手法・データ
を関係でつなぎ、出典へさかのぼれる。



出典まで一意にたどれる
例) Node: 酸化物超伝導
Edge: 「置換効果」
Source: [Smith2021]
DOI: 10.xxxx/...



仮説を立て、Jupyterで実験を回す
再現可能なコードとデータで、
仮説を検証し、結果を可視化。



結果はセルにひも付く
例) Tc 提高率 = 12.4%
[cell:42]
RMSE = 0.812 [cell:58]



AIに“反論”させ、敵対的に検証する
NatureLM / GALACTICA が批判的レビュー。
想定反論を先回りして潰す。



反論と対応が記録される
反論: 先行研究と矛盾する点
対応: 追加実験 (cell:73-81)
→ 矛盾解消



機械的に監査し、整合性を担保する
数値・引用・図を自動チェック。
Value Mismatch / Ghost / Orphan を検出。

✓ Value Mismatch: 0 件
✓ Ghost References: 0 件
✓ Figure Orphans: 0 件



すべての主張が、計算セルにひも付く
「どの計算が支えているのか？」に、
根拠をもって、即答できる。



30回転んで分かったのは、AIを賢くする方法ではなく、AIと“反証可能に科学する”方法だった。
それが、私の **AI for Science** の本質。

速さ
(Speed)



遠くへ (Reach)
反証可能に



落とし穴は、終わりではなく、本質への入口だった。

あなたの Reach が、誰かの未来を照らす。✦✦

AIRAはオープンソース。一緒に、次の穴を埋めましょう。

私が落ちた30の穴は、地図として公開しています。皆さんは、踏まずに進める。そして、新しい穴を一緒に見つけてください。

公開リソース（すべてオープンソース）



研究OS「AIRA」

AI for Science のための
研究パイプライン・オペレーティングシステム



github.com/nahisaho/aira



実験ログ（R1～R47）

1,000+件・14ラウンドの試行錯誤の全記録
失敗も成功も、すべて公開。



github.com/nahisaho/aira_experiments



自作 GraphRAG エンジン （aira-synapse）

論文の海を、「関係の地図」に変えるエンジン



github.com/nahisaho/aira-synapse



専用グラフDB（aira-graphdb）

Science Knowledge Graph を高速・安全に管理



github.com/nahisaho/aira-graphdb



実践記録（Qiita 連載）

- ・「動いたのに研究になっていない」
 - ・「AIRA設計：研究OSを作るということ」
 - ・「論文の海から知を掘り出す」
- など、実践の過程をすべて公開中。



qiita.com/nahisaho



AI for Science は、まだ**未踏地**だらけ。
私がまだ知らない、31個目、32個目の穴が、
きっと待っています。
一緒に見つけて、埋めていきましょう。



新しい問いを
一緒に見つける



使える仕組みを
一緒に育てる



知を共有し、
科学を前に進める



私が落ちた**30の穴**、すべて公開中。

いつ・どこで・何に・どう転んだか。
R1からR47まで、すべて追体験できます。



総実験数
1,000+件



試行ラウンド
14ラウンド
（R1～R47）



実験期間
2024.10～
2025.06

AIRA Pipeline：研究の全工程をつなぐOS



収集
（データ）



計画
（仮説）



実行
（計算・実験）



検証
（敵対的検証+監査）



記録・共有
（再現性の確保）

すべてが、プロセスとデータでつながり、
すべての主張が、計算セルにひも付く。



Jupyter
Notebook

ひも付く
（[cell:ID]）



主張・結果



今日から、あなたの“**もう一人の科学者**”を起動できる。

賢いモデルを探す旅ではない。
反証可能なプロセスを、設計する旅だ。

落とし穴の地図は、
もう、あなたの手にある。

ご清聴、ありがとうございました — さあ、あなたの問いを聞かせてください。

30回の失敗が教えてくれた、AIと科学をつなぐ本質と実践の地図を、ぜひ明日からの研究に。

本講演の3つの持ち帰り

1 本質

AI for Science は
Speed ではなく Reach。



反証可能性のプロセス設計が核心。
AIは代替でなく **増幅** する。

Reach



反証可能性を
維持する設計



研究者の問いの
射程を広げる



代替ではなく
能力の増幅

2 道具

研究OS (Skill × MCP) が、
Reach を **再現可能** にする。



知識の地図 (GraphRAG) と検証の仕組みを
パイプラインでつなぎ、再現性を確保。



Skill
(専門能力)

×



MCP
(外部ツール連携)

×



GraphRAG
(知識の地図)

つなぐ・回す・検証する → 再現可能な研究へ

3 実践

1枚のテンプレート +
3つのゴールデンルールから始める。



1枚のテンプレート
(仮説検証フレーム)

- ✓ 仮説と破綻条件を先に書く
- ✓ 評価指標と分割戦略を固定
- ✓ 外部モデルに反論を求める
- ✓ 最終判断まで一貫して記録

3つのゴールデンルール
(実装の規律)

- Rule 1** 再現性
環境記録とシードを最初に実行
- Rule 2** 追跡可能性
全定量値に [cell:<id>] を付与
- Rule 3** 整合性
引用台帳と図マニフェストを作成

テンプレート (設計) と3ルール (実装) の両輪で、
AI特有の「嘘」を機械的に撃ち落とす。



今日から、自分の研究のどの工程に、AIを迎え入れるかを考えてみてください。



問いの発見・整理に?



実験計画・実行に?



検証・考察・記録に?

さあ、ここからは皆さんの番です。
ご質問をお聞かせください!



公開リソース (すべてオープンソース)



- 研究OS「AIRA」
github.com/nahisaho/aira
- 実験ログ (R1~R47)
github.com/nahisaho/aira_experiments
- 自作GraphRAG エンジン (aira-synapse)
github.com/nahisaho/aira-synapse
- 専用グラフDB (aira-graphdb)
github.com/nahisaho/aira-graphdb
- 実践記録 (Qiita 連載)
qiita.com/nahisaho
「動いたのに研究になっていない」
「AIRA設計」「論文の海から知を掘り出す」など



お問い合わせ
nahisaho@microsoft.com



AIRA リポジトリ
github.com/nahisaho/aira



参考文献・出典は
次ページ (補足) に掲載しています。

本資料は研究コミュニティの共有資産です。
自由にご活用いただけます。

補足：出典・参考

AI for Science の定義（第0部・第7部）

- DOE, *AI for Science: Report on the DOE Town Halls on AI for Science* (2020). Argonne/Oak Ridge/Berkeley 主催のタウンホール（2019年・参加者1,000名超）に基づく。
- DOE, *AI for Science, Energy, and Security Report* (2023).
- 文部科学省「AI for Scienceの推進に向けた基本的な戦略方針」（令和8年3月31日）。
- ARiSE（AI for Science革新的研究推進事業）／SPReAD（AI for Science萌芽的挑戦研究創出事業：年間1,000件規模・最大500万円・全分野対応・学生応募可）。第7期 科学技術・イノベーション基本計画（2026～2030）。

登壇者自身の実証（第1～5部の落とし穴）

- AI for Science実践記録（2,000+実験／R1～R47の検証ログ）。Qiita「『動いた』のに、研究になっていない——R1～R47で突きつけられたこと」 <https://qiita.com/hisaho/items/55c6c8c85fab048f38b3>
- 核心：AI for Scienceの価値は Speed ではなく Reach の拡大。本質は生成能力ではなく「反証可能性を維持するプロセス設計」（生成→実行→敵対的検証→機械的監査の4層）。
- 品質指標 Balance=VM（数値不整合）・UC（裏づけのない主張）・FO（孤立図）の統合スコア。実験ログ抜粋：R26=1.55／R38=5.16／R39=2.96／R40=1.21／R41=3.86／R42=1.88／R44=0.67。

研究OS「AIRA」(Skill と MCP で動くOS)

- オープンソース研究OS「AIRA (AI Research Assistant)」。Qiita「AIで科学する - 自律型リサーチエージェント AIRA の設計」 <https://qiita.com/hisaho/items/e85acd4d0eddf618779f>
- リポジトリ：<https://github.com/nahisaho/aira> / 実験ログ：https://github.com/nahisaho/aira_experiments (1,000+件・14ラウンド)
- 構成：GitHub Copilot CLI をカーネルに採用。能力レイヤ=Skill (202の専門サブスキル+2カスタムエージェント research-lead/methods-auditor、研究ライフサイクル8フェーズ)、接続レイヤ=MCP (Model Context Protocol)。
- MCP統合先：ToolUniverse (Harvard/Zitnik Lab、1,000+科学DB：PubMed・ChEMBL・UniProt・GWAS Catalog・KEGG 等)、NatureLM (Microsoft Research、分子・タンパク質・材料・RNAの定量予測・生成)、GALACTICA (Meta AI、4,800万論文学習・矛盾検出/推論)、Jupyter MCP (コード実行・検証)。
- 品質保証：3つのゴールデンルール (環境/シード固定・全定量値への `[cell:<id>]` 引用・引用台帳/図マニフェスト)。検出対象：Value Mismatch・Ghost References・Figure Orphans。
- 核心メッセージ：「AI for Science は研究者の能力を増幅するツールであり、研究者の代わりに科学をするツールではない」。生成物は第一稿であり、実験的妥当性・科学的判断は研究者が担う。

学術論文向けGraphRAG (知識獲得エンジン／落とし穴10～16)

- Qiita「論文の海から知を掘り出す - aira-synapse / aira-graphdb 開発記」
<https://qiita.com/hisaho/items/e481aa076a2b7d3a6750>
- Classic RAGはマルチホップ推論・エンティティ同一性・矛盾検出に弱い。MemGraphRAG (KDD 2026) をTypeScriptでクリーンルーム実装=aira-synapse、専用グラフDB aira-graphdbをRustで自作。HotpotQA 500問で Str-Acc 89.4% / LLM-Acc 91.2% (論文GPT-4o-mini 71.6%比 +19.6pt)。日本語はGINZA統合でGPT-5.5 78.3% (Neo4j基準58.5%比 +19.8pt)。
- 関連用語：RAG (検索拡張生成)、ナレッジグラフ、PPR (Personalized PageRank)、RRF (Reciprocal Rank Fusion)、HotpotQA (マルチホップQAベンチマーク)。