複雑な知能を生み出す
極限まで簡素化された環境の設計

第28回 人工知能学会 汎用人工知能研究会(SIG-AGI)

一杉裕志,高橋直人,竹内泉（産総研）,
佐野崇（東洋大学）,中田秀基(産総研,順天堂大学)

2024-12-20

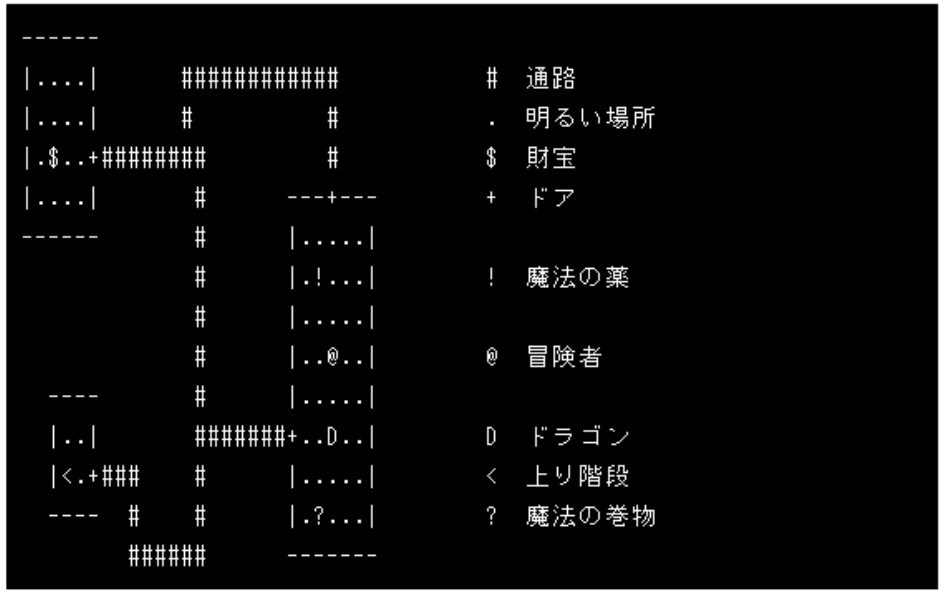
私の研究の中期的目標

[一杉+ 第18回 汎用人工知能研究会(SIG-AGI), 2021]

- ローグライクゲームのような、
  実世界を極力単純化した環境で
  脳型AGIのデモを動かす

今日の話

- 中核技術
  - プログラム合成対象言語 Pro5Lang
    [一杉+ 第20回 汎用人工知能研究会, 2022]
  - 再帰的強化学習 RGoal
    [Ichisugi+ 2019]
  - 大脳皮質モデル BESOM
    [Ichisugi 2007]



「ローグライクゲーム - Wikipedia」
https://ja.wikipedia.org/wiki/ローグライクゲーム

脳型AGIアーキテクチャの全体像については、
前回までの発表内容でほぼ固まった

現状の大規模言語モデル(LLM)の問題点

- **数学的推論能力が弱い** [Mirzadeh+ arXiv:2410.05229]
  - GPT-4o や o1-preview ですら、
    問題に無関係な情報を追加すると正解率低下
  - 論理的推論ではなく主にパターンマッチング
- **ハルシネーション、ジェイルブレイクの問題**
  - 何が「事実」なのか、何が「悪いこと」なのかを、
    正確に学習できていない・正確に推論できない
- パラメタサイズとデータを増やせば解決？ **→ すでに限界**
- 思考時間を増やせば解決？ **→ 不正確な推論を積み重ねても無意味**

- 脳型AGIの開発に取り組むことで原因と対策が見えてくる → 次ページ

私が考える現状の LLM の問題点

- **コンテキストウインドウが広い＝パラメタサイズが多い＝汎化性能悪化**
  - 文脈に依存した「それっぽい答え」を出すには向いているが、論理的な推論規則を獲得するには向かない
- **「内部状態」をテキスト表現の対話履歴の形でしか持てない**
  - テキスト表現はあいまい、知識の追加と検索が非効率的かつ不正確
  - トークン列で表現できない感覚イメージなどを直接表現できない
- **プログラムとデータの表現が全く違う**
  - （プログラム＝パラメタの値、 データ＝トークン列）
  - 「自己書き換えコード」のようなことがやりにくい
  - 対話・読書等を通じた継続的な知識獲得がやりにくい

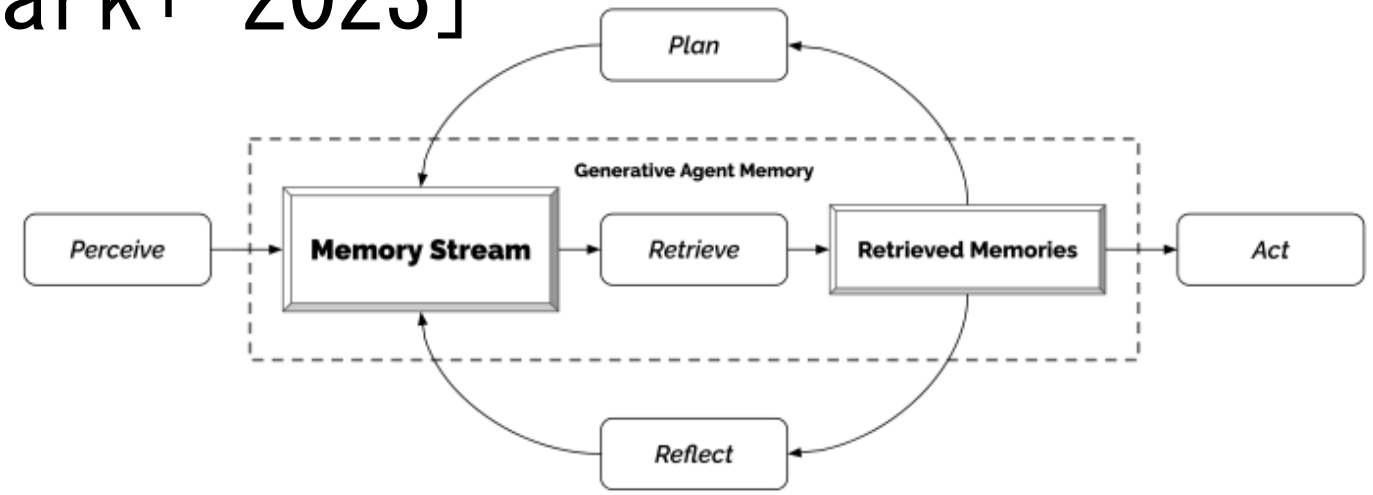
これらの問題が、ヒトレベルの知能への到達の障害になるのではないか？

開発中の脳型ＡＧＩアーキテクチャの特徴

- 計算論的神経科学の観点での特徴：
  - 前頭前野とその周辺の機能の計算論的モデルを目指す
  - 前頭前野は行動計画や推論などの、ヒトらしい知能の最高中枢
  - 主にシステム２（意識的な思考）
    ←→ システム１（無意識的な思考）
- 機械学習・AIの観点での特徴：
  - プログラム合成、階層型強化学習、生成モデル・確率的グラフィカルモデル
  - 記号AI、数理論理学
- デモの目標：
  - 環境の中で試行錯誤することで自律的に知識獲得
  - 複数のエージェントが生活する社会で法律や経済などを創発
- これまでの発表資料：
  https://staff.aist.go.jp/y-ichisugi/j-index.html

関連研究
Generative Agents [Park+ 2023]

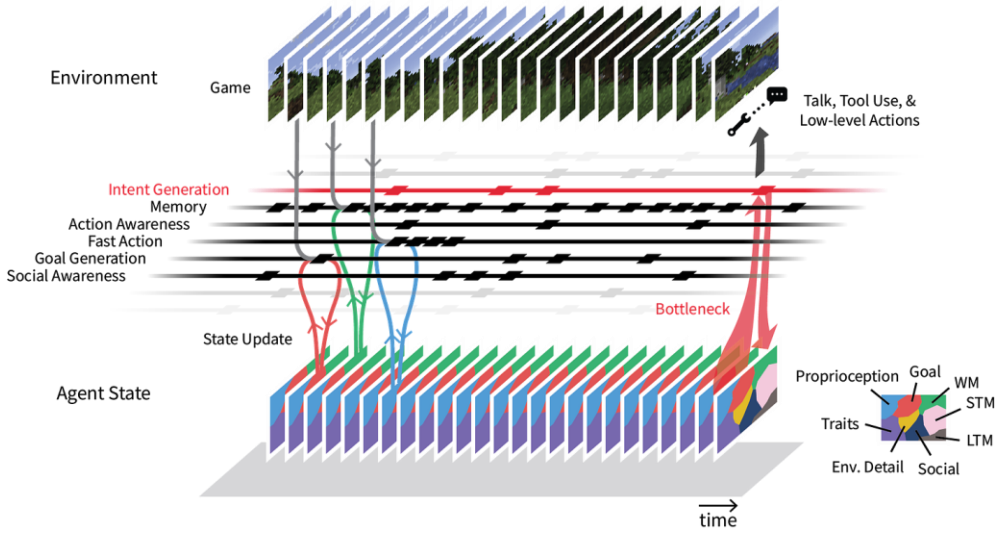
- ChatGPTを使って推論・行動するAIエージェント
- テキストベースの「エピソード記憶」や「ワーキングメモリ」を持つ
- エージェントどうしが町で対話などをしながら生活
- 「動機」を人間が与えて行動をシミュレーション
  - 「バレンタインパーティーを開きたい」
    → 数日後パーティーが開かれた



https://arxiv.org/pdf/2304.03442.pdf

関連研究
Project Sid （AI企業・Altera.AL）

- GPT-4o をベースとしたAIエージェント
- 研究としては初期段階
- 「私たちのエージェントは、空間的推論や物理的協調などの基本的な能力にまだ苦労しており、社会の進化を推進する人間の本質的な衝動(生存、好奇心、社会的絆)を欠いています。」



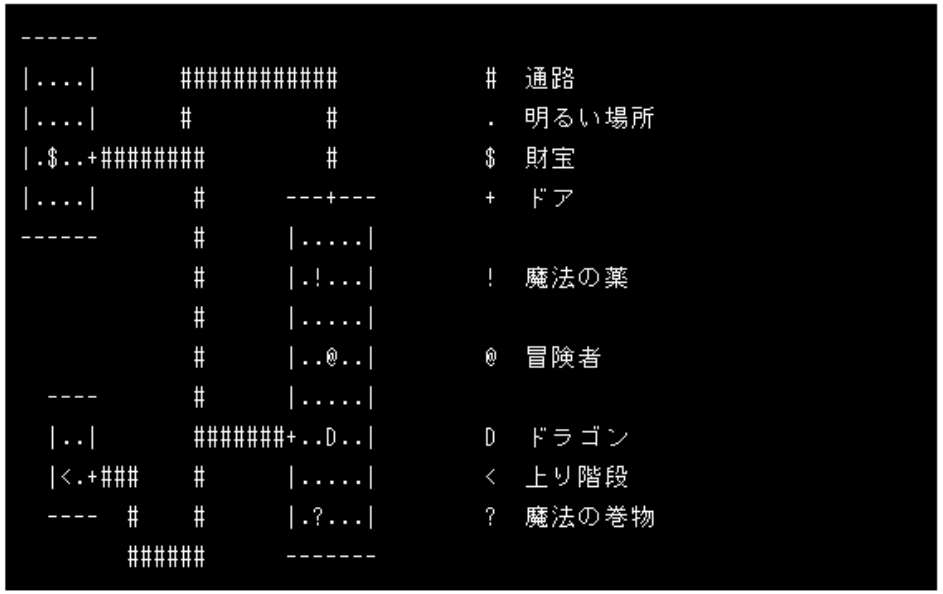
「Project Sid: Many-agent simulations toward AI civilization」Nov 02, 2024
https://digitalhumanity.substack.com/p/project-sid-many-agent-simulations

エージェントの環境の設計が重要

- **「豊かな環境では報酬を最大化するために様々な能力が要求されるため、豊かな知的能力が生み出される」**
  [Silver+ 2021]
- AGI の目標は複雑な実世界の環境で動作すること
- 一方で、いきなり実世界でＡＧＩ開発をするのは困難
  - 計算コスト、実験コスト、ノイズや実験の再現性、・・・
- **まずは実世界の本質的性質を保ったまま極力単純化した人工的な実験環境を作ることが重要**

ローグライクゲーム

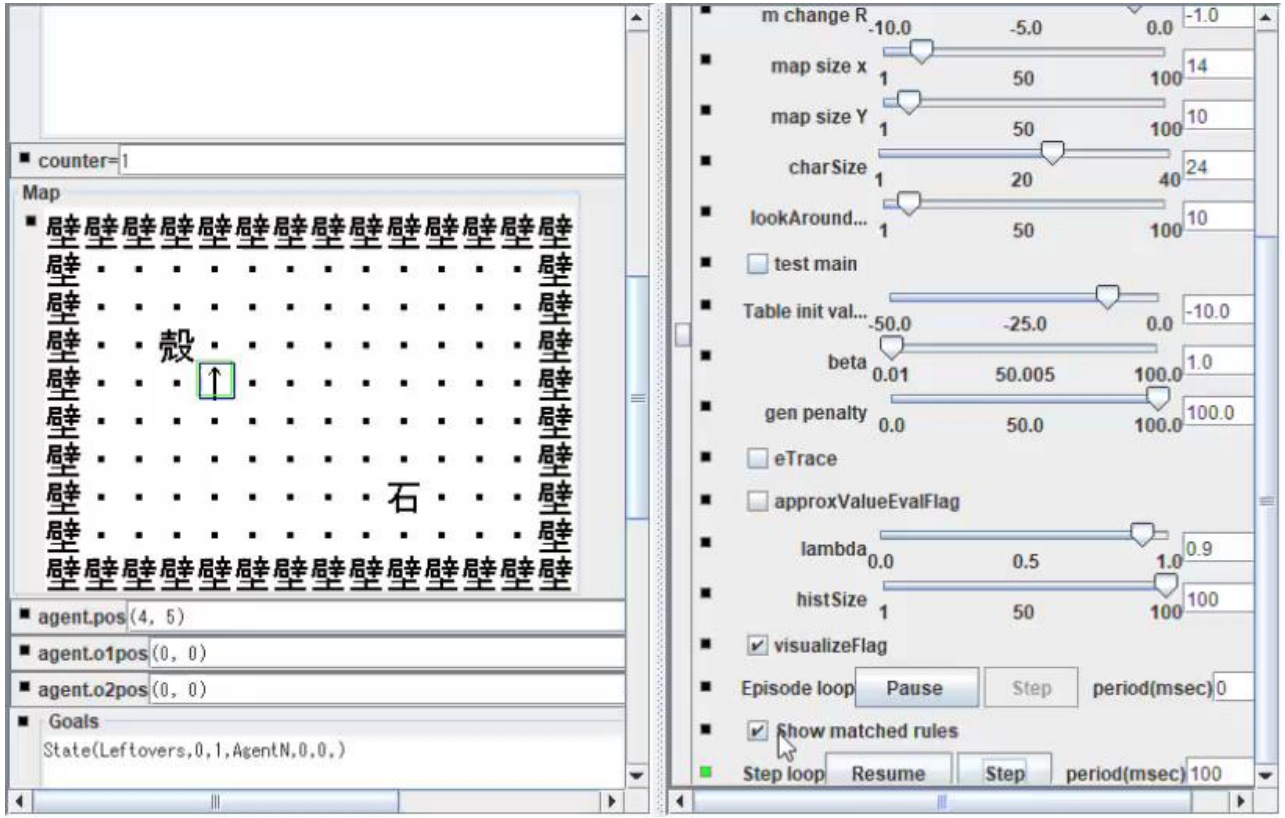
- 時間と空間が離散化
- マップ上のアイテムは記号化
- マップの大きさ、
  アイテムの数、
  物体同士の相互作用の数に制限がない
- 他者（敵）の動きが複雑
- プレイヤーは環境について経験から学習しなければならない



- NeurIPS 2021 におけるコンペティションの題材にもなっている

「ローグライクゲーム - Wikipedia」
https://ja.wikipedia.org/wiki/ローグライクゲーム

最初に作ったプロトタイプ



殻に石をぶつけて
実を食べるタスク

・注意の機構を導入、サブルーチン実行に無関係な情報を無視
・身体中心座標と環境中心座標の分離し行動ルールの汎用性を高める

行動ルール集合

（エージェントが経験から獲得しなければならないプログラムを私が手で書いてみたもの）

```
{   // o1 に近づくサブルーチン。
    // 物体２への注意をリセット。
    q(s(o1,x1,y1,o2,x2,y2), s(o1,0,1,A,0,0), Action.Reset2);
    // 前方の o1 に近づく。
    q(s(o1,x1,1,A,0,0), s(o1,0,1,A,0,0), Action.MoveForward);
    q(s(o1,x1,2,A,0,0), s(o1,0,1,A,0,0), Action.MoveForward);
    q(s(o1,x1,3,A,0,0), s(o1,0,1,A,0,0), Action.MoveForward);
    q(s(o1,x1,4,A,0,0), s(o1,0,1,A,0,0), Action.MoveForward);
    q(s(o1,x1,5,A,0,0), s(o1,0,1,A,0,0), Action.MoveForward);
    // 後方の o1 に向かって向きを変える。
    q(s(o1,x1,-1,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,x1,-2,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,x1,-3,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,x1,-4,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,x1,-5,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    // 右の o1 に向きを変える。
    q(s(o1,1,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnRight);
    q(s(o1,2,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnRight);
    q(s(o1,3,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnRight);
    q(s(o1,4,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnRight);
    q(s(o1,5,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnRight);
    // 左の o1 に向きを変える。
    q(s(o1,-1,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,-2,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,-3,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,-4,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
    q(s(o1,-5,0,A,0,0), s(o1,0,1,A,0,0), Action.TurnLeft);
}
{   // o1 を o2 の１つ手前の場所に移動するサブルーチン。
    StateN g = s(o1,0,1,o2,0,2);
    // 前方の o2 に近づく。
    q(s(o1,0,1,o2,x2,2), g, Action.MoveForward);
    q(s(o1,0,1,o2,x2,3), g, Action.MoveForward);
    q(s(o1,0,1,o2,x2,4), g, Action.MoveForward);
    q(s(o1,0,1,o2,x2,5), g, Action.MoveForward);
    q(s(o1,0,1,o2,x2,6), g, Action.MoveForward);
    // 後方の o2 に近づく。
    q(s(o1,0,1,o2,x2,0), g, Action.Pull);
    q(s(o1,0,1,o2,x2,-1), g, Action.Pull);
    q(s(o1,0,1,o2,x2,-2), g, Action.Pull);
    q(s(o1,0,1,o2,x2,-3), g, Action.Pull);
    q(s(o1,0,1,o2,x2,-4), g, Action.Pull);
    q(s(o1,0,1,o2,x2,-5), g, Action.Pull);
    // o1←o2 となった場合。回り込んで　→o1 o2　とする。
    q(s(o1,0,1,o2,0,-1), g, Action.TurnRight);
    q(s(o1,-1,0,o2,1,0), g, Action.MoveForward);
    q(s(o1,-1,-1,o2,1,-1), g, Action.TurnLeft);
    q(s(o1,-1,1,o2,-1,-1), g, Action.MoveForward);
    q(s(o1,-1,0,o2,-1,-2), g, Action.MoveForward);
    q(s(o1,-1,-1,o2,-1,-3), g, Action.TurnLeft);
    q(s(o1,-1,1,o2,-3,1), g, Action.MoveForward);
    q(s(o1,-1,0,o2,-3,0), g, Action.TurnLeft);
    // o1　 o2　となった場合は左側に回り込んで →o1　 o2 とする。
    // ↑
    q(s(o1,0,1,o2,x1,1), g, Action.TurnLeft);
    q(s(o1,1,0,o2,1,y2), g, Action.MoveForward);
    q(s(o1,1,-1,o2,1,y2), g, Action.TurnRight);
    q(s(o1,1,1,o2,x1,1), g, Action.MoveForward);
    q(s(o1,1,0,o2,x1,0), g, Action.TurnRight);
}
```

```
{   // Nut を食べるサブルーチン。
    StateN g = s(Leftovers,0,1,A,0,0);
    // 物体２を無視
    q(s(__,__,__,__,__,__), s(Leftovers,0,1,A,0,0), Reset2);
    // Nut を探す。
    q(s(__,__,__,A,0,0), g, c(Nut,x1,y1,A,0,0));
    {   // Nut が見つかった場合。
        // Nut のところに移動する。
        q(s(Nut,x1,y1,A,0,0), g, c(Nut,0,1,A,0,0));
    }
    {   // Nut が見つからない場合。
        // Stone と Shell を探す。
        q(s(A,0,0,A,0,0), g, c(Stone,x1,y1,A,0,0));
        q(s(Stone,x1,y1,A,0,0), g, c(Stone,x1,y1,Shell,x2,y2));
        // Shell を Stone の手前に移動。
        q(s(Stone,x1,y1,Shell,x2,y2), g, c(Stone,0,1,Shell,0,2));
        // 押す。（目の前に Nut が現れるはず。）
        q(s(Stone,0,1,Shell,0,2), g, MoveForward);
        // →石実　という状態なのでいったん右に出る。
        q(s(Stone,0,1,Nut,0,2), g, TurnRight);
        q(s(Stone,-1,0,Nut,-2,0), g, MoveForward);
        // あとは普通に　Nut を食べに行く。
        q(s(Stone,-1,-1,Nut,-2,-1), g, c(Nut,-2,-1,Nut,-2,-1));
        q(s(Nut,-2,-1,Nut,-2,-1), g, Reset2);
    }
    // 目の前に Nut がある場合は前に進む。
    q(s(Nut,0,1,__,__,__), g, MoveForward);// これは選ばれない。
    q(s(Nut,0,1,A,0,0), g, MoveForward);
}
{   // Stone を Shell の手前に移動するサブルーチン。
    StateN g = s(Stone,0,1,Shell,0,2);
    // いったん Shell を無視。
    q(s(Stone,x1,y1,Shell,x2,y2), g, Reset2);
    // Stone のところに移動
    q(s(Stone,x1,y1,A,0,0), g, c(Stone,0,1,A,0,0));
    // Stone が目の前にある時に Shell を再度探す。
    q(s(Stone,0,1,A,0,0), g, c(Stone,0,1,Shell,x2,y2));
    // o1 を o2 の１つ手前の場所に移動。
    //q(s(Stone,0,1,Shell,x2,y2), g, c(Stone,0,1,Shell,0,2));
}
{   // 物体１を探すサブルーチン。
    StateN g = s(o1,__,__,o2,__,__);
    q(s(__,__,__,o2,__,__), g, Find1);
}
{   // 物体２を探すサブルーチン。
    StateN g = s(o1,__,__,__,o2,__);
    q(s(o1,__,__,__,__,__), g, Find2);
}
```

問題点：アクション、座標の抽象化が不十分

前頭前野の機能の実験・開発に集中したい

- 前頭前野：行動計画・推論などを実行する部位

- **前頭前野以外の部位が担当する機能を徹底的に排除して簡素化**
- 座標・距離の概念を排除
  - 頭頂葉・前運動野・前頭眼野
- 抽象度の高いプリミティブ行動
  - 前運動野（視覚情報などを用いた習熟した運動の実行に関与する部位）
- 他者との意思疎通では論理式を直接やり取り
  - 言語野

今回提案する環境

- 環境は、複数の部屋から構成される
- 各部屋内に０個以上の物体がある
- 部屋内・部屋間の距離や座標は扱わない
- 同じ部屋内の他のAIエージェントと論理式を使って意思疎通が可能

例：Room1 に何もなく、Room2 にアリスとカギ、Room3 に宝箱がある状態：

```
Room1:
Room2: Alice Key
Room3: Box
```

おそらく十分に複雑な知能を生み出せる

Room1:
Room2: Alice Key
Room3: Box

- 知能を持った他者の存在
  - 模倣による知識獲得
  - 対話により他者の心的状態の推論・操作が可能
- 環境の状態の自律的変化
  - 定期的に食物が発生する農場
  - 時間の経過による食物の腐敗
  - 自然災害
- この環境内で報酬最大化するために必要なこと：
  - 効率的な収穫戦略
  - 将来に備えた備蓄
  - 他の地域のエージェントとの間の経済活動

テスト用タスクのシナリオ：
Key を Box に適用して Gold を得るタスク

Room1: Alice
Room2: Key
Room3: Box
↓ **Key の場所に移動**
Room1:
Room2: Key Alice
Room3: Box
↓ **Key を Box の場所に持っていく**
Room1:
Room2:
Room3: Key Box Alice
↓ **Key を Box に適用して Gold を得る**
Room1:
Room2:
Room3: Gold Alice

「宝箱・宝物のイラスト | かわいいフリー素材集 いらすとや」
https://www.irasutoya.com/2013/05/blog-post_2133.html

テスト用タスクの前提

- Alice は Key と Box がどこにあるかを推論できる
- Alice は以下の対外アクションが実行できる
  - Room 内にある物体を探す: **FindItem**
  - 他の Room に移動する: **GoToRoom**
  - Room 内にある物体を他の Room に持っていく: **Bring**
  - Room 内のある物体をもう１つの物体に適用する: **Apply**

Room1: Alice
Room2: Key
Room3: Box
↓ **Key の場所に移動**
Room1:
Room2: Key Alice
Room3: Box
↓ **Key を Box の場所に持っていく**
Room1:
Room2:
Room3: Key Box Alice
↓ **Key を Box に適用して Gold を得る**
Room1:
Room2:
Room3: Gold Alice

Pro5Lang プログラムがエージェントの行動を制御

- 機械語と論理型言語の特徴を合わせ持った手続き型言語
- 行動ルール rule(s, g, a) の集合
  s ： 現在の状態
  g ： 現在のサブゴール
  a ： 実行すべきアクション
- 現在の s, g にマッチする行動ルールを
  その価値の高さに応じた確率で選択し、
  行動 a を実行
- 行動ルールの価値は強化学習で学習

```
 1: k(e(0, 0, Socrates, Is, AMan, 0));      // ソクラテスは人間である
 2: k(e(If, c(0, 0, x, Is, AMan, 0), c(0, 0, x, Is, Mortal, 0))); // 人間は死ぬ

 3: eifxy = e(If, c(x1,x2,x3,x4,x5,x6), c(y1,y2,y3,y4,y5,y6));
 4: eif0y = e(If, c(__,__,__,__,__,__), c(y1,y2,y3,y4,y5,y6));
 5: ax = a(x1,x2,x3,x4,x5,x6);
 6: ay = a(y1,y2,y3,y4,y5,y6);
 7: rule(w(),       w(ay), recall(eif0y));  // y を証明するために x → y を想起
 8: rule(w(eifxy),  w(ay), call(ax));       // x → y なら y を証明するために x を証明
 9: rule(w(ax),     w(ay), recall(eifxy));  // x なら y を証明するために x → y を想起
10: rule(w(ax, eifxy), w(ay), set(ay));     // x, x → y なら y が成り立つ

11: ex = e(x1,x2,x3,x4,x5,x6);
12: rule(w(),   w(ax), recall(ex));
13: rule(w(ex), w(ax), set(ax));
```

プロトタイプ版 Pro5Lang のプログラム例

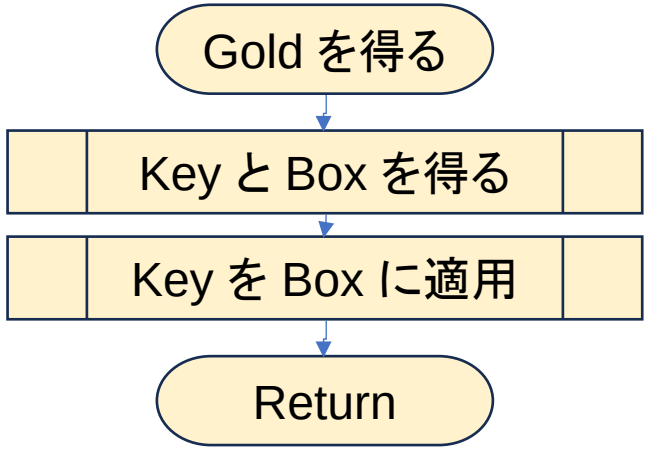
環境とのインタラクション：「プリミティブ行動」

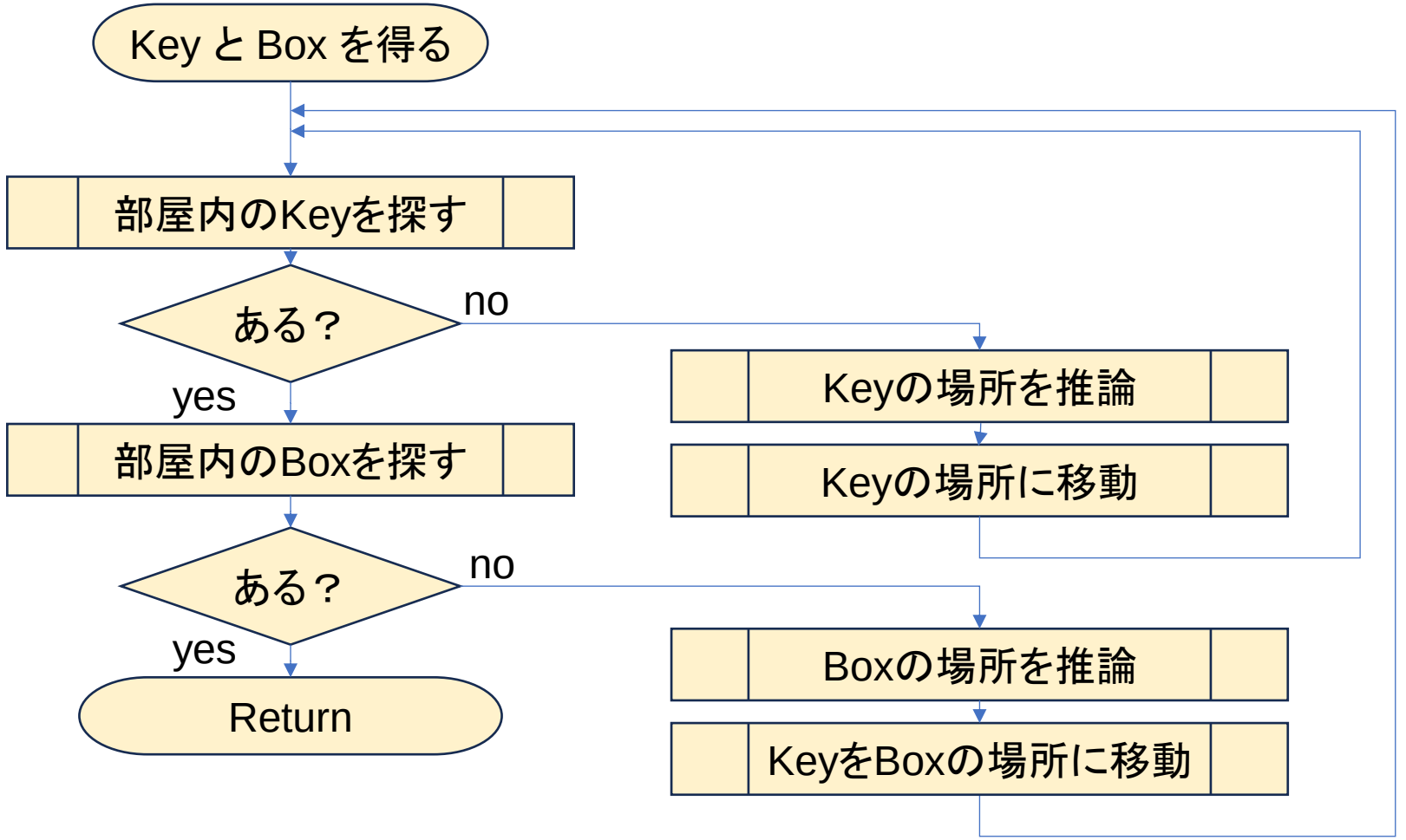
- プリミティブのパラメタはサブゴールを通じて渡す
- 実行結果は a レジスタにセットされる
- プリミティブ行動は前運動野が実行すると想定

プリミティブを呼び出す行動ルールの例

1: rule(_,
「***Alice*** は **PLS** で **PLS** に注視している」, ***FindItem***)
2: rule(「***Alice*** は **r** で **x** に注視している」,
「***Alice*** は **r** で **x** と **PLS** に注視している」, ***FindSecondItem***)
3: rule(「***Alice*** は **r** で **x** と **y** に注視している」,
「***Alice*** は **r** で **x** を **y** に適用した」, ***Apply***)
4: rule(_,
「***Alice*** は **PLS** から **PLS** に移動した」, ***GoToRoom***)
5: rule(「***Alice*** は **r1** で **x** に注視している」,
「***Alice*** は **x** を **r1** から **r2** に持ってきた」, ***Bring***)

テスト用タスクを解くフローチャート

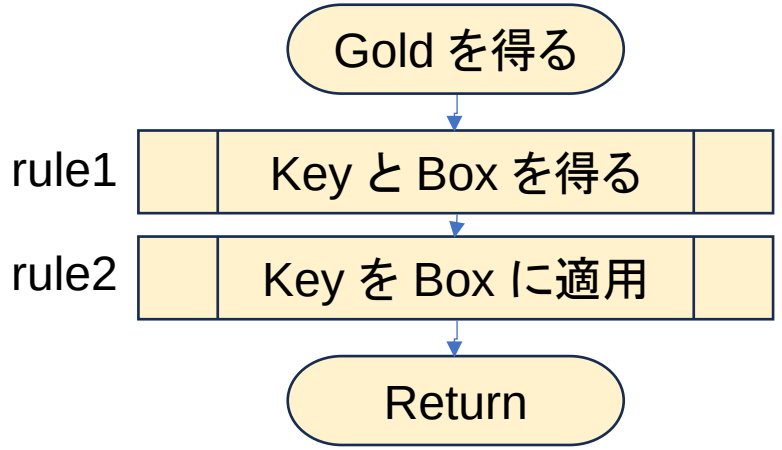




テスト用タスクを解くプログラムのメインルーチン

疑似コード

```
   g = 「Alice は PLS で Gold に注視している」
1: rule(_, g,
       call(「Alice は PLS で Key と Box に注視している」));
2: rule(「Alice は r1 で Key と Box に注視している」, g,
       call(「Alice は r1 で Key を Box に適用した」));
```



実際のコード

```
StateN g = w(a(c(LookingAt,Alice,Gold,0,PLS,Now,true)));
rule(w(), g,
       call(a(c(LookingAt2,Alice,Key,Box,PLS,Now,true))));
rule(w(a(c(LookingAt2,Alice,Key,Box,r1,Now,true))), g,
       call(a(c(AppliedO1ToO2,Alice,Key,Box,r1,Now,true))));
```

テスト用タスクを解くプログラムの疑似コード
サブルーチン「x と y を得る」

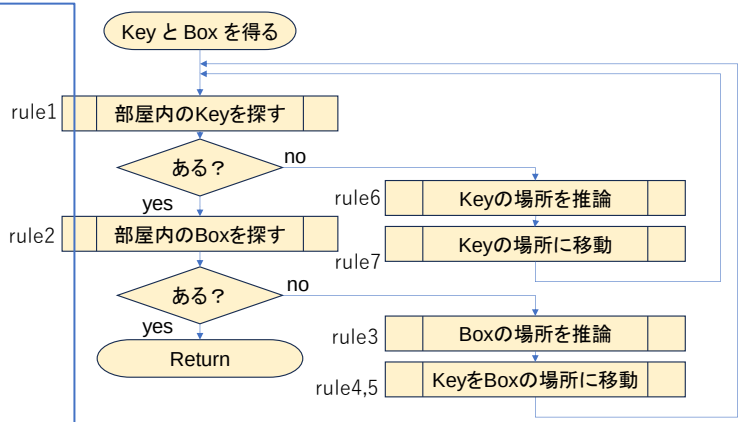
Start

l2g = 「***Alice*** は **PLS** で **x** と **y** に注視している」

1: rule(_, l2g,
  call(「***Alice*** は **PLS** で **x** に注視している」)
2: rule(「***Alice*** は **r1** で **x** に注視している」, l2g,
  call(「***Alice*** は **r1** で **x** と **y** に注視している」))
3: rule(「***Alice*** は **r1** で **x** と **y** に注視していない」, l2g,
  call(「**y** が **PLS** にある」))
4: rule(「**y** が **r2** にある」, l2g,
  recall(「***Alice*** は **PLS** で **x** に注視している」))
5: rule({「**y** が **r2** にある」,「***Alice*** は **r1** で **x** に注視している」}, l2g,
  call(「***Alice*** は **x** を **r1** から **r2** に持ってきた」))
6: rule(「***Alice*** は **r1** で **x** に注視していない」, l2g,
  call(「**x** が **PLS** にある」))
7: rule(「**x** が **r1** にある」, l2g,
  call(「***Alice*** は _ から **r1** に移動した」))

no
yes
no
yes

Return



実際のコード
サブルーチン「x と y を得る」

```
        StateN l2g = w(a(c(LookingAt2,Alice,x,y,PLS,Now,true)));
1:      rule(w(), l2g,
              call(a(c(LookingAt,Alice,x,0,PLS,Now,PLS))));
2:      rule(w(a(c(LookingAt,Alice,x,0,r1,Now,true))), l2g,
              call(a(c(LookingAt2,Alice,x,y,r1,Now,PLS))));
3:      rule(w(a(c(LookingAt2,Alice,x,y,r1,Now,false))), l2g,
              call(a(c(Exists,y,0,0,PLS,Now,true))));
4:      rule(w(a(c(Exists,y,0,0,r2,Now,true))), l2g,
              recall(e(c(LookingAt,Alice,x,0,PLS,Now,true))));
5:      rule(w(a(c(Exists,y,0,0,r2,Now,true)), e(c(LookingAt,Alice,x,0,r1,Now,true))), l2g,
              call(a(c(BroughtO1FromO2,Alice,x,r1,r2,Now,true))));
6:      rule(w(a(c(LookingAt,Alice,x,0,r1,Now,false))), l2g,
              call(a(c(Exists,x,0,0,PLS,Now,true))));
7:      rule(w(a(c(Exists,x,0,0,r1,Now,true))), l2g,
              call(a(c(CameFrom,Alice,__,0,r1,Now,true))));
```

まとめと今後

- ヒトのような知能を生む極力簡素化したエージェント環境を提案
  - 前頭前野の計算モデルの実験・開発に特化
    - 時間・空間は離散化、ただしサイズに制限なし
    - 部屋内・部屋間の座標や距離の概念を排除
  - 他者の行動を観測可能（未実装）
  - 他者と意思疎通が可能（未実装）
  - 極めて単純でありながら、おそらく複雑な知能が獲得可能な環境
- 今後
  - この研究アプローチの面白さをアピールするデモ
    - ヒトらしい知能とは何かを理解できるはず
    - 大規模言語モデルの限界を突破するアイデアが得られるはず
- これまでの発表資料：
  https://staff.aist.go.jp/y-ichisugi/j-index.html
  質問・コメント等を歓迎いたします

以上