報酬最大化を目的とする 行動計画・実行・対話・推論の統一的制御機構

A unified control mechanism for action planning, execution, dialogue, and inference for the reward maximization

一杉裕志 *1 Yuuji Ichisugi　中田秀基 *1 Hidemoto Nakada　高橋直人 *1 Naoto Takahashi　竹内泉 *1 Izumi Takeuti　佐野崇 *2 Takashi Sano

*1産業技術総合研究所 人工知能研究センター
National Institute of Advanced Industrial Science and Technology (AIST), AIRC

*2東洋大学 情報連携学部情報連携学科
Faculty of Information Networking for Innovation And Design, Toyo University

We are developing an AI architecture that uses recursive reinforcement learning to control thought and behavior, in order to realize artificial general intelligence in the future. Agents will act on the environment, interact with others, and reason about the state of the environment under unified control in order to maximize rewards. In the future, we plan to implement a mechanism that allows agents to synthesize the control program based on their own experiences. In this paper, we describe the overall architecture and propose a mechanism for action planning that works on top of it. We implemented a prototype system of the proposed mechanism and verified its operation.

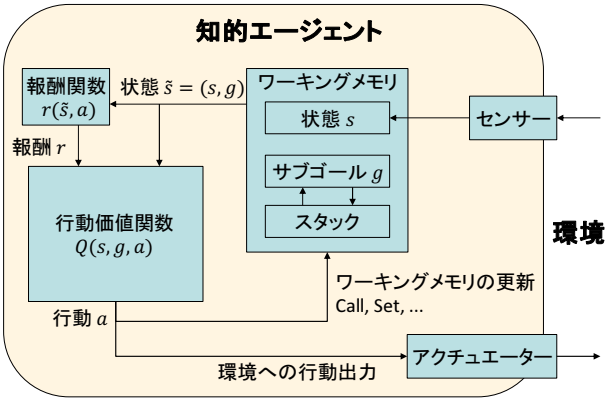

図 1: RGoal アーキテクチャ。行動は内部状態の更新または外部への運動出力である。

1. はじめに

我々は再帰的な階層型強化学習を用いて思考・行動を制御するＡＩアーキテクチャを開発中である [一杉 21]。エージェントは報酬最大化を目的として、環境への働きかけ・他者との対話・環境の状態に関する推論を、統一的に制御する。将来は、これらを制御するための知識を、エージェントが自分自身の経験にもとづいて自ら獲得する機構を実装する予定である。本稿ではこのアーキテクチャの上で動作する、行動計画の機構およびそのプロトタイプ実装について述べる。行動計画は、行動の選択肢とその効果についての宣言的知識を用いて、目標を達成するために必要な行動を推論することにより実現される。効率的な行動計画プログラムを、エージェントが自身自身の経験から獲得することが原理的に可能である。本機構は、ヒトの脳の前頭前野の計算論的モデルの候補でもある。

連絡先: 一杉裕志, E-mail: y-ichisugi@aist.go.jp

2. アーキテクチャの全体像

2.1 再帰的強化学習 RGoal

RGoal [Ichisugi 19] は再帰的なサブルーチン呼び出しを可能とする階層型強化学習アーキテクチャである（図 1）。

階層型強化学習はサブタスク共有・時間抽象・状態抽象 [Dietterich 00] の３つ恩恵により学習速度が向上する。RGoal ではサブルーチン呼び出しの自由度が高いため、タスクによってはこれらの恩恵をより大きく受けられる。

RGoal におけるサブルーチン g は、「任意の状態から状態 g に向かう方策」を意味する。この g はサブゴールとも呼ばれる。例えば「はしごを置く」という方策を定義するサブルーチンにおけるサブゴールは「目の前にはしごが置かれた状態」である。

2.2 合成対象言語 Pro5Lang

RGoal の行動価値関数 $Q(s,g,a)$ を圧縮表現したものは、エージェントの行動を制御するプログラムと見なすことができる [一杉 18]。我々はこのプログラムを表現するプログラミング言語を Pro5Lang *1 と名づけ [一杉 22a]、その言語仕様の設計・実装を進めている。行動価値関数の学習と圧縮を行うことは、Pro5Lang で書かれたプログラムをプログラム合成することに等しい。合成の性能を上げるためには、 Pro5Lang の言語仕様を適切に設計する必要がある。

Pro5Lang は論理型言語（数理論理学を基礎にしたプログラミング言語）と機械語（コンピューターを構成する論理回路が直接解釈実行できる言語）の特徴を合わせ持ったプログラミング言語である。情報の記憶場所としてレジスタと連想記憶装置を有し、そこに証明済みの命題を記憶する。

Pro5Lang のプロトタイプシステムを、Java 言語のソースコード中に埋め込める DSL (Domain-Specific Language) として実装し、この DSL で様々なテストプログラムを手で書いて動かしてみることにより、処理系の動作検証と言語仕様の妥当性の検証を進めている。

*1 Probabilistic Proven-Proposition Processing Programming Language （確率的証明済み命題処理プログラミング言語）の略。

Pro5Lang のプログラムは**行動ルール**の集合として定義される。DSL では行動ルールは rule(s,g,a) という構文で定義される。エージェントは、現在のレジスタの状態 s およびサブゴールの値 g とマッチする s,g のパターンを持つ行動ルールを、その価値の高さに応じて確率的に１つ選択し、その行動 a を実行する。行動ルールの価値の学習が進めばエージェントは報酬を最大化するように合理的に振舞うようになる。

エージェントが取り得る行動のうち、call, set, recall 命令はそれぞれサブルーチン呼び出し、レジスタの値の更新、宣言的知識の想起を実行するプリミティブである。他に、環境の観測、物体や自分自身の移動、他者との対話などを行うプリミティブがある。

Pro5Lang の詳細については [一杉 22a] を参照されたい。

2.3 POMDP への対処

部分観測マルコフ決定過程 (POMDP) は、環境のモデルが与えられていれば、信念状態（エージェントが推定する環境の状態の確率分布）を状態とみなせば MDP になり、解けるようになる [Kaelbling 98]。POMDP を解いたエージェントは、必要な情報を得るために環境を能動的に観測したり、他者に聞くという行動をとるようになる。

Pro5Lang は、信念状態を近似的に表現することで POMDP を近似的に解く。信念状態は独立な変数に因子分解されて表現される。逆に言えば、環境の状態の確率分布は個々の変数が表現する確率の積として表現される。（この因子分解は何らかのオートエンコーダーを用いた教師なし学習で行うことを想定している。）個々の変数は unknown という値を取ることができる。これは信念状態においてその変数の確率分布が一様分布であることを意味している。Pro5Lang の行動ルールにおいては状態 s のパターンの表現に変数の値が unknown かどうかも指定することができる。これにより、「戸棚の中にチョコレートがあるかどうかわからなければ開けてみる」、「ハサミの場所がわからなければ兄に聞く」、「冷蔵庫に残っている食材を知るために最近の食事の記憶から推論する」といった POMDP に対処する行動ルールが、記号的に簡潔に表現できる。

2.4 Pro5Lang による推論

Pro5Lang のプログラムは、環境の隠れた状態などを推論することができる [一杉 20][一杉 22a]。推論規則は行動ルールによって表現される。例えば推論規則 P,Q ⊢ R は、Pro5Lang の疑似コードで書けば

rule({P,Q}, R, set(R))

となる。これの意味は「命題 P, Q が成り立っていて、かつ命題 R が証明のゴールの時、命題 R は証明されたと見なす」となる。このような行動ルール集合を保持しておけば、Prolog 言語のような後ろ向き連鎖 *2 による証明探索を実行し、様々な命題を推論することができる。

証明に用いる行動ルールの「正しさ」は、行動ルールの価値（ゴールに到達するまでのコスト）として、強化学習アルゴリズムを使って学習する [一杉 20]。価値の学習は、センサー情報と矛盾する命題を set する動作を検出すると推論を初期状態に戻すという機構のもとで行われる。この機構により、エージェントがいる環境において正しくない推論規則の価値は下がり、選択されなくなってゆく。すなわち、推論規則は環境に接地する。

推論のための行動ルールは、環境に働きかけるための行動ルールと本質的に区別なく扱われる。推論はエージェントにとって、 unknown な変数の値を知るための手段の１つにすぎず、自分で推論しないで直接環境を観測するとか、他者に聞くといった手段が他に考えられ、コストが低い手段ほど高い確率で選択される。また、複数の推論戦略があるときも、学習が進めば、状況に応じて効率的な戦略ほど高い確率で選択される。

2.5 対話

Pro5Lang は、環境内の他のエージェントと対話するためのプリミティブも持つ [一杉 22b]。

対話は言語理解と発話から成る。言語理解は話者の意図推定（話者が何を伝えようとしているのかを推定する）であり、これは環境の隠れた状態の推論の一種である。また、発話は他者への知識伝達や行動の依頼などを目的としており、これは環境の状態を変化させる行動の一種である。この解釈の下では、報酬を最大化するような合目的的な言語理解と発話は、通常の推論や行動と特別な違いはなく、Pro5Lang で書かれたプログラムにより実現可能である。そして、そのようなプログラムは原理的に、エージェントが自分自身の経験から獲得可能である。

2.6 テスト環境

以上のような特徴を持ったＡＩアーキテクチャのテストを行う環境の開発を、Pro5Lang の開発と並行して進めている。エージェントが動作するテスト環境の設計においては、実世界の本質的特徴を残しつつ極力単純化することを重視している [一杉 21]。このテスト環境の中で、エージェントは他者と会話をしながら動き回り、身体に作り込まれた報酬関数が定める報酬を最大化するように学習を進める。

2.7 脳との対応

アーキテクチャの設計の際には、脳との対応付けを強く意識している。手続き的知識（Pro5Lang の行動ルール集合）と連想記憶は、それぞれ前頭前皮質と海馬の長期記憶に対応すると考えている。また、レジスタの値は前頭極や感覚連合野における短期記憶に対応すると考えている。

3. 行動計画

3.1 提案手法

これまで述べたように、推論、言語理解、発話はいずれもエージェントの「行動」の一形態として解釈でき、 Pro5Lang で表現される行動ルール集合によって制御される。この節では、行動計画もまた推論の一形態であり、行動ルール集合で制御可能であることを示す *3。

ここでは行動計画を「初期状態と終了状態が与えられたとき、終了状態に到達するまでの行動系列を求める問題」と定義する。例えば、「電球を取り替えたい」という目標を持って行動計画を立てると、「電気屋で電球を買ったあと、納屋に行って脚立をとってきて上って取り換えよう」というような行動系列が得られる。環境のモデル（ある行動をとると環境がどう変化するか）はすでに獲得済みであるものとする。

行動計画には様々な実装方法があり得るが、本稿では一例として、以下のシンプルな方針で実装したものを説明する。「状態 s のときサブゴール g' を設定すればいつかゴール g を達成できる」という命題を $Achieves(s, g', g)$ と表記する。この形の宣言的知識をあらかじめ蓄えておき、与えられたゴールに到達するために取るべき行動を、以下の推論規則を用いて後ろ向き連鎖を用いて推論する。

$$Achieves(g', x, g), Achieves(g'', y, g') \vdash Achieves(g'', g', g)$$

*2 ゴールから後ろ向きに推論規則を適用していき証明を完成させる手法。

*3 実装にあたって従来の Pro5Lang の機能をほとんど変えていないが、レジスタが保持する節の数を２つから３つに増やしている。

```
// 行動計画を開始
1: rule(s, g, call(Achieves(s, +, g)))
// 行動計画の結果を使って最初の１ステップを実行
2: rule(Achieves(s, g', g), g, call(g'))

// 行動計画サブルーチン
gp = Achieves(s, +, g) // Goal of Planning
// 行動計画の最初の１ステップ
3: rule(_ , gp, recall(Achieves(+, + ,g)))
// 行動計画の途中のループ
4: rule(Achieves(g', x, g), gp,
      recall(Achieves(+, +, g')))
5: rule({Achieves(g', x, g), Achieves(g", y, g')}, gp,
      set(Achieves(g", g', g)))
```

図 2: 行動計画プログラムの一例の疑似コード

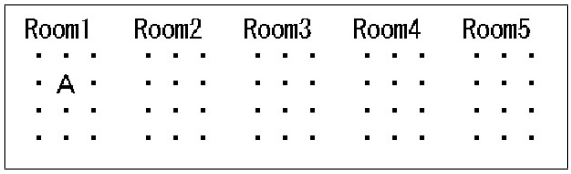

図 3: Room1 から Room5 に移動するタスク

この推論規則は、「状態 g' から g を達成可能かつ g'' から g' を達成可能ならば g'' から g' を経由して g を達成可能である」ということを意味している。この推論によって、最初に実行すべきサブルーチン g' が得られる。これを実行してサブゴール g' に到達したらあらためて行動計画を立て、次に実行すべきサブルーチンを推論する。これをゴールに到達するまで繰り返す。

3.2 テストプログラム例

図 2 は行動計画を行う Pro5Lang の行動ルール集合の一例である。ここで “_” は任意の値、 “+” は任意の unknown でない値にマッチする記号である。

このプログラムの動作は以下のようになる。状態 s とゴール g が与えられるとルール１が選択され、ルール３， ４， ５で定義される行動計画サブルーチン gp が呼び出される。サブルーチン内ではまずルール３でゴールから１ステップさかのぼれる宣言的知識を想起する。そのあと、ルール４， ５を繰り返し、 $g'' = s$ となったときにサブゴール gp が達成されるのでサブルーチンから戻る。そのときのレジスタの状態は $Achieves(s, g', g)$ であるのでルール２が実行され、推論で得られた最初のサブゴール g' が呼び出される。（この行動計画プログラムは行動系列を一度にすべて推論せずに、最初の１ステップのみを推論し、それを実行する。）ルール３および４における recall 命令は、宣言的知識の中から引数で指定したクエリにマッチするものを非決定論的に１つ想起する。マッチするものが１つもない場合（すなわちそれ以上ゴールからさかのぼるための知識が存在しない場合）は、Pro5Lang の機能により、実行コンテキストがリセットされ最初から実行をやり直す。これを繰り返し、もし解が見つかれば、そこで行動計画が終了する。（解が必ず見つかるとは限らない。）

```
宣言的知識：
Achieves(「Room1 にいる」, 「Room2 にいる」, 「Room2 にいる」)
Achieves(「Room2 にいる」, 「Room3 にいる」, 「Room3 にいる」)
Achieves(「Room3 にいる」, 「Room4 にいる」, 「Room4 にいる」)
Achieves(「Room4 にいる」, 「Room5 にいる」, 「Room5 にいる」)

手続き的知識：
rule(「Room1 にいる」, 「Room2 にいる」, 東に進む)
rule(「Room2 にいる」, 「Room3 にいる」, 東に進む)
rule(「Room3 にいる」, 「Room4 にいる」, 東に進む)
rule(「Room4 にいる」, 「Room5 にいる」, 東に進む)
```

図 4: 与える宣言的知識と手続き的知識の疑似コード

図 3 のマップにおいて、エージェント A が Room1 から Room5 に移動するタスクを用いて実際の動作の一例を説明する。

図 4 は与える知識の例である。エージェントは１つ東側の部屋に移動する方法は知っているが、より離れた部屋に移動する方法はまだ知らないものとする。この知識の下で s =「Room1 にいる」、g = 「Room5 にいる」としてプログラムの実行を開始するとまず図 2 のルール１によりサブルーチン $Achieves$(「Room1 にいる」, g',「Room5 にいる」) が呼び出され、その結果 g' =「Room2 にいる」 が推論される。次にエージェントはルール２を実行しサブゴール「Room2 にいる」を設定する。このサブゴールはエージェントが持つ手続き的知識（「Room1 にいる」ときに「Room2 にいる」状態になるためには東に進む）を実行することにより達成される。次にルール１により $Achieves$(「Room2 にいる」, g',「Room5 にいる」) が呼び出され、以下同様に行動と行動計画が繰り返され、最終的にゴール「Room5 にいる」が達成される。

3.3 宣言的知識の獲得方法

前節では行動計画に必要な宣言的知識と手続き的知識はあらかじめ手で与えたが、将来的にはエージェントが経験から知識獲得するようにする必要がある。

知識獲得の手段として、自分自身の行動のエピソード記憶から推論、他者の行動の観察から推論、他者の発話から推論、他の宣言的知識からの推論、などを想定している。

3.4 他の行動計画プログラムの可能性

図 2 のプログラム以外にも、様々な行動計画プログラムの実現方法が考えらえれる。ヒトはおそらく複数の行動計画プログラムを保持しており、状況に応じて使い分けているのだろう。具体的には、ゴールまでの行動系列を一度に推論する方法や前向き連鎖を使う方法などが考えられる。

本稿で述べた方法ではサブルーチンの実行コストを考慮していない。宣言的知識にコスト c を含めて $Achieves(s, g', g, c)$ とすれば行動計画と同時にゴール達成コストを計算することができ、得られた複数の解を比較してコストが低いものを選択することが可能になるだろう。

4. 関連研究

RMDP [Hahn 22] は、RGoal と同様に、深さに制限のない再帰呼び出しを許す階層型強化学習の１つの定式化である。RMDP は RGoal と異なり、１つのサブルーチンが複数の出口を持つ場合も扱っており、そのため学習アルゴリズムが若干複雑になる。ただし、全てのサブルーチンの出口が１つしかない特殊ケースにおいては、学習アルゴリズムは RGoal と本質的に同じになる。

Dyna-Q のようなモデルベース強化学習 [Sutton 18] は、行動計画と目的は似ている。モデルベース強化学習はエージェントがいわば脳内でシミュレーションを行って未経験の行動の価値を学習する手法だが、本稿で述べた行動計画の機構は行動価

値関数は変えずに、レジスタおよび連想記憶の内容だけを変化させる点が異なっている。

筆者らの以前の研究では思考モード [Ichisugi 19] と呼ぶ一種のモデルベース強化学習の機構を提案していたが、神経科学的な妥当性が不明確な上、思考と行動を合目的的に切り替える機構が別途必要になるという問題もあるため、本稿ではそのような問題のない機構を提案した。

5. まとめ

本稿ではまず、行動・推論・発話・言語理解を再帰的強化学習の枠組みの中で統一的に実行できるアーキテクチャの概要について説明した。そのうえで、行動計画もまた一種の推論として統一的に扱えることを示した。

行動計画の機構により、未知の状況でも既知の経験を組み合わせて対処できる。また、行動計画に必要な情報をエージェント自身が推論・対話・観測行動を通じて得るようにすることも容易であろう。提案手法のこれらの特徴は汎用人工知能の実現に役立つだろう。

今回は非常にシンプルな行動計画プログラムを例にとって説明したが、3.4 節で述べた他の方法の実装も検討している。実装した行動計画の機構を用いて、エージェントどうしが対話を通じて合目的的な意思疎通を行うデモの実現を目指している。

謝辞

本研究は JSPS 科研費 JP22K12188 の助成を受けたものです。

参考文献

[Dietterich 00] Dietterich, T. G.: Hierarchical reinforcement learning with the MAXQ value function decomposition, *Journal of artificial intelligence research*, Vol. 13, pp. 227–303 (2000)

[Hahn 22] Hahn, E. M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., and Wojtczak, D.: Recursive Reinforcement Learning, in Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. eds., *Advances in Neural Information Processing Systems* (2022)

[Ichisugi 19] Ichisugi, Y., Takahashi, N., Nakada, H., and Sano, T.: Hierarchical Reinforcement Learning with Unlimited Recursive Subroutine Calls, in *Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning*, pp. 103–114, Cham (2019)

[Kaelbling 98] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R.: Planning and acting in partially observable stochastic domains, *Artificial Intelligence*, Vol. 101, pp. 99–134 (1998)

[Sutton 18] Sutton, R. S. and Barto, A. G.: *Reinforcement learning: An introduction*, MIT press (2018)

[一杉 18] 一杉裕志, 高橋直人, 中田秀基, 佐野崇：単一化の機構を利用した階層型強化学習のテーブル圧縮手法の検討, 第 10 回人工知能学会 汎用人工知能研究会 (SIG-AGI) (2018)

[一杉 20] 一杉裕志, 中田秀基, 高橋直人, 佐野崇：推論規則の価値を階層型強化学習 RGoal を用いて学習する手法の提案, 第 14 回 人工知能学会 汎用人工知能研究会 (SIG-AGI) (2020)

[一杉 21] 一杉裕志：報酬最大化原理にもとづく脳型ＡＧＩアーキテクチャの構想, 第 18 回人工知能学会 汎用人工知能研究会 (SIG-AGI) (2021)

[一杉 22a] 一杉裕志, 中田秀基, 高橋直人, 竹内泉, 佐野崇：汎用人工知能のためのプログラム合成対象言語 Pro5Lang のエピソード記憶機構, 第 20 回 人工知能学会 汎用人工知能研究会 (SIG-AGI) (2022)

[一杉 22b] 一杉裕志, 中田秀基, 高橋直人, 竹内泉, 佐野崇：報酬最大化ＡＧＩのための意思疎通機構の設計とプロトタイプ実装, 第 21 回人工知能学会 汎用人工知能研究会 (SIG-AGI) (2022)