モンテカルロ版 RGoal アルゴリズムの改良

第26回 人工知能学会 汎用人工知能研究会(SIG-AGI)
一杉裕志, 中田秀基, 高橋直人, 竹内泉（産総研）, 佐野崇（東洋大）
2024-03-08

発表の内容

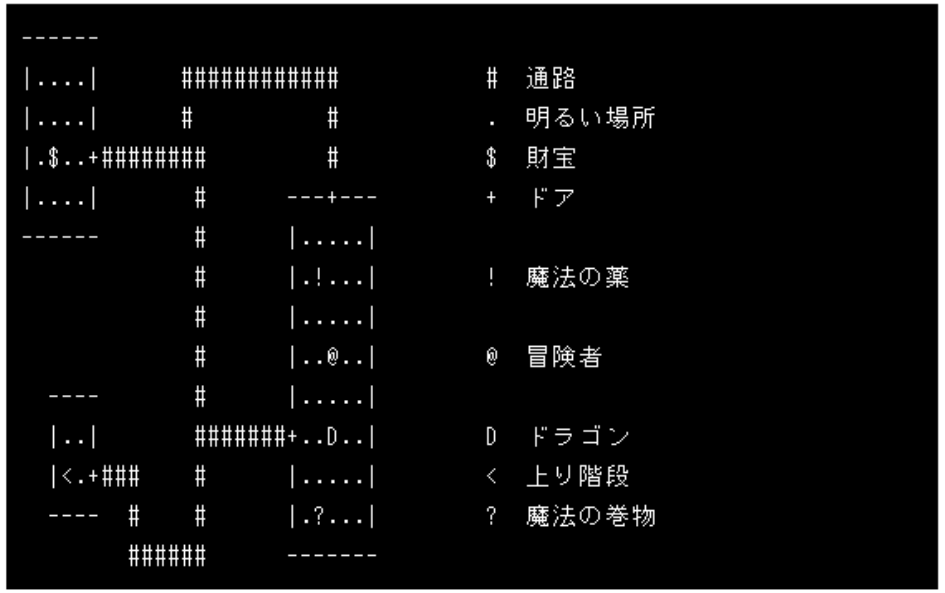
- 前回[一杉+ 第25回 汎用人工知能研究会, 2023] 提案した再帰的強化学習アルゴリズムにあった問題点を改良
- 提案アルゴリズムを行動計画タスク[一杉+ JSAI 2023] に適用

私の研究の中期的目標

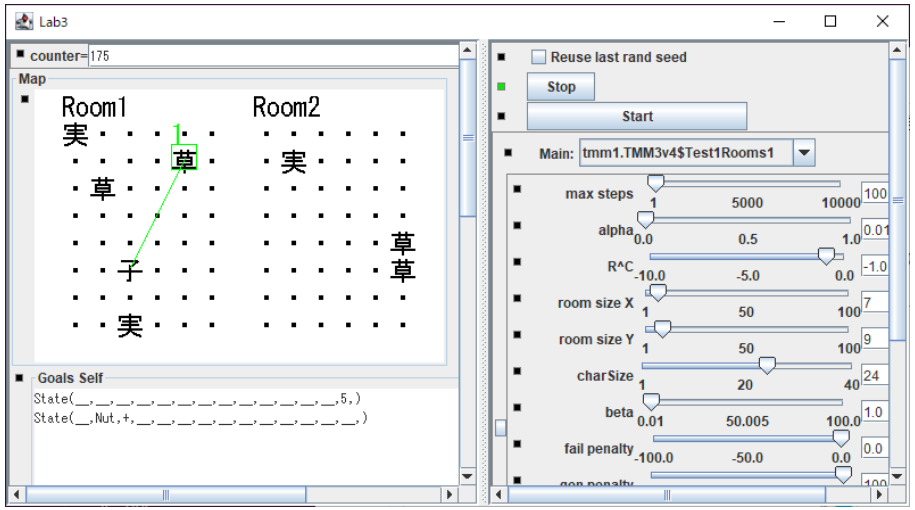
[一杉+ 第18回 汎用人工知能研究会(SIG-AGI), 2021]

- 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
- 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]

今日の話



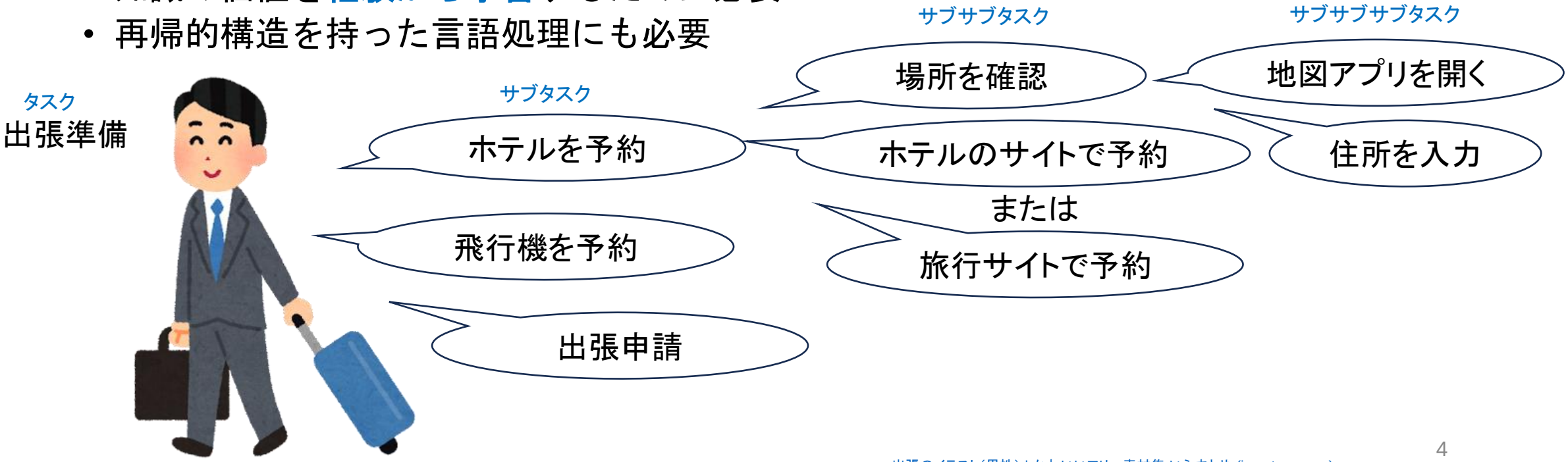
「ローグライクゲーム - Wikipedia」
https://ja.wikipedia.org/wiki/ローグライクゲーム



開発中のAGIエージェント実行環境

なぜ再帰的強化学習が重要か？

- 再帰的強化学習：再帰的なサブルーチン呼び出しを許す階層型強化学習
- 脳型ＡＧＩの実現におそらく不可欠
  - 既存のサブルーチンを組み合わせて**新規タスクをゼロショットで解く**ために必要
  - 知識の価値を**経験から学習**するために必要
  - 再帰的構造を持った言語処理にも必要



出張のイラスト（男性）| かわいいフリー素材集 いらすとや (irasutoya.com)

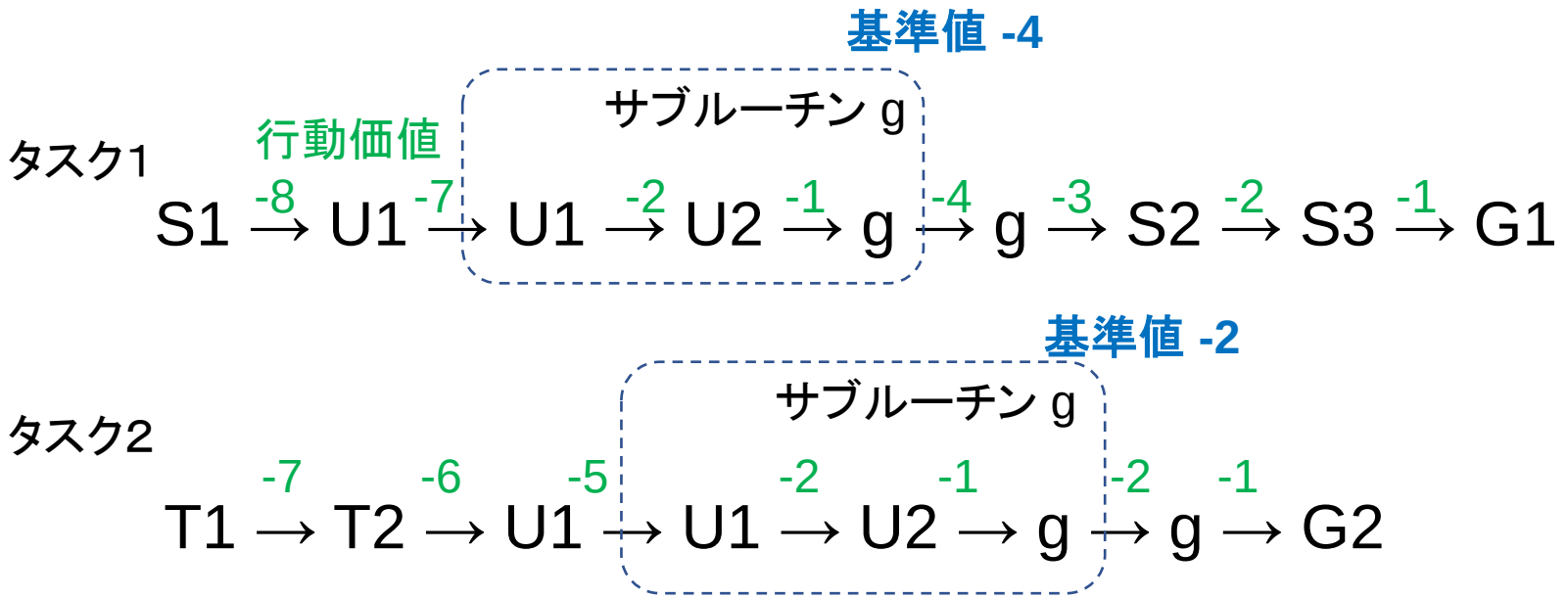
先行研究との関係

- MAXQ [Dietterich 2000]
  - 学習した方策の再利用が可能
  - サブルーチン外の報酬はサブルーチン内に伝搬しないので、**最適解が得られない**(recursively optimal)
- Options [Sutton+ 1999]
  - 学習結果の**再利用は設計の目標としていない**
  - （階層構造の制約の範囲内で）最適解が得られる(hierarchically optimal)
- 提案アルゴリズムの設計目標（AGI実現にどうしても必要）
  - 学習した方策を**再利用**
  - サブルーチン外の報酬を中に伝搬させることで**解の精度を上げる**

→ 解決策： MAXQ と似た考えを取り入れつつ、外の報酬を中に伝搬

アルゴリズムの設計方針

- サブルーチンを正常終了したときの状態の価値の**方策πにおける期待値**を**基準値**として、
  サブルーチン内の**相対価値**を学習する
- 基準値は、**異なる呼び出し文脈ごと**に学習する

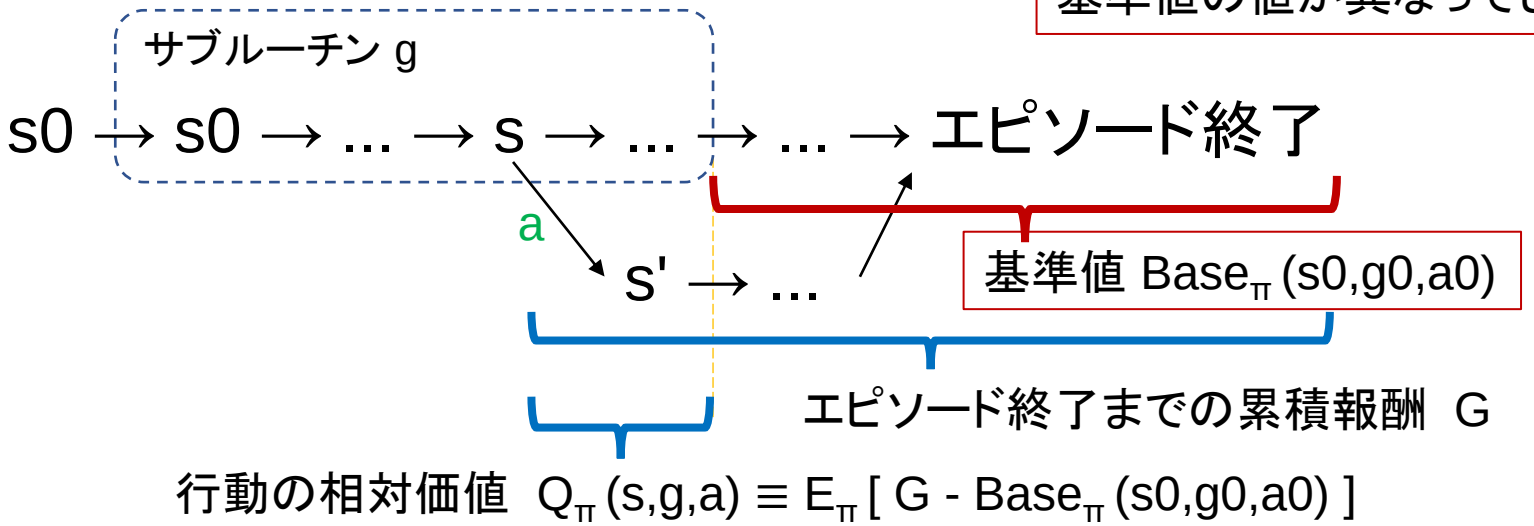


前回発表のモンテカルロ版 RGoal 基準値の問題点

[一杉+ 第25回 汎用人工知能研究会, 2023]

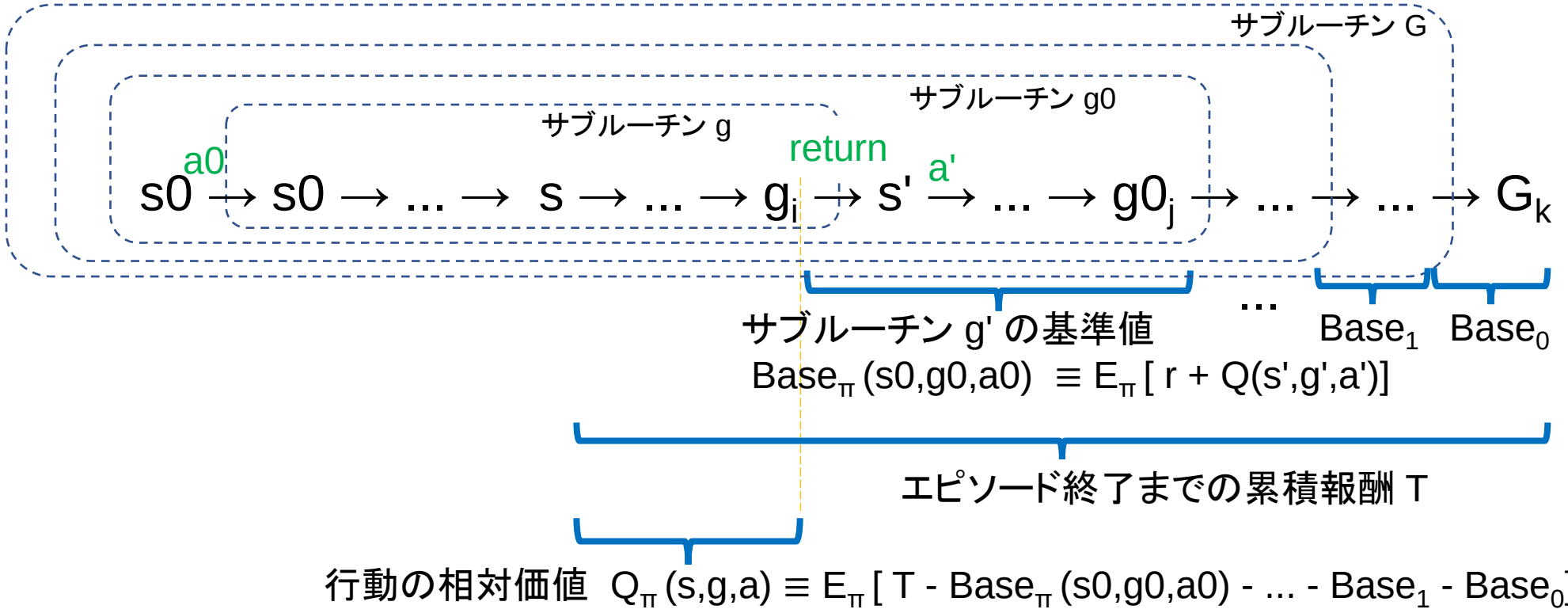
基準値はエピソード終了までの累積報酬

→ 別のタスクでサブルーチンを再利用すると
基準値の値が異なってしまうという問題がある



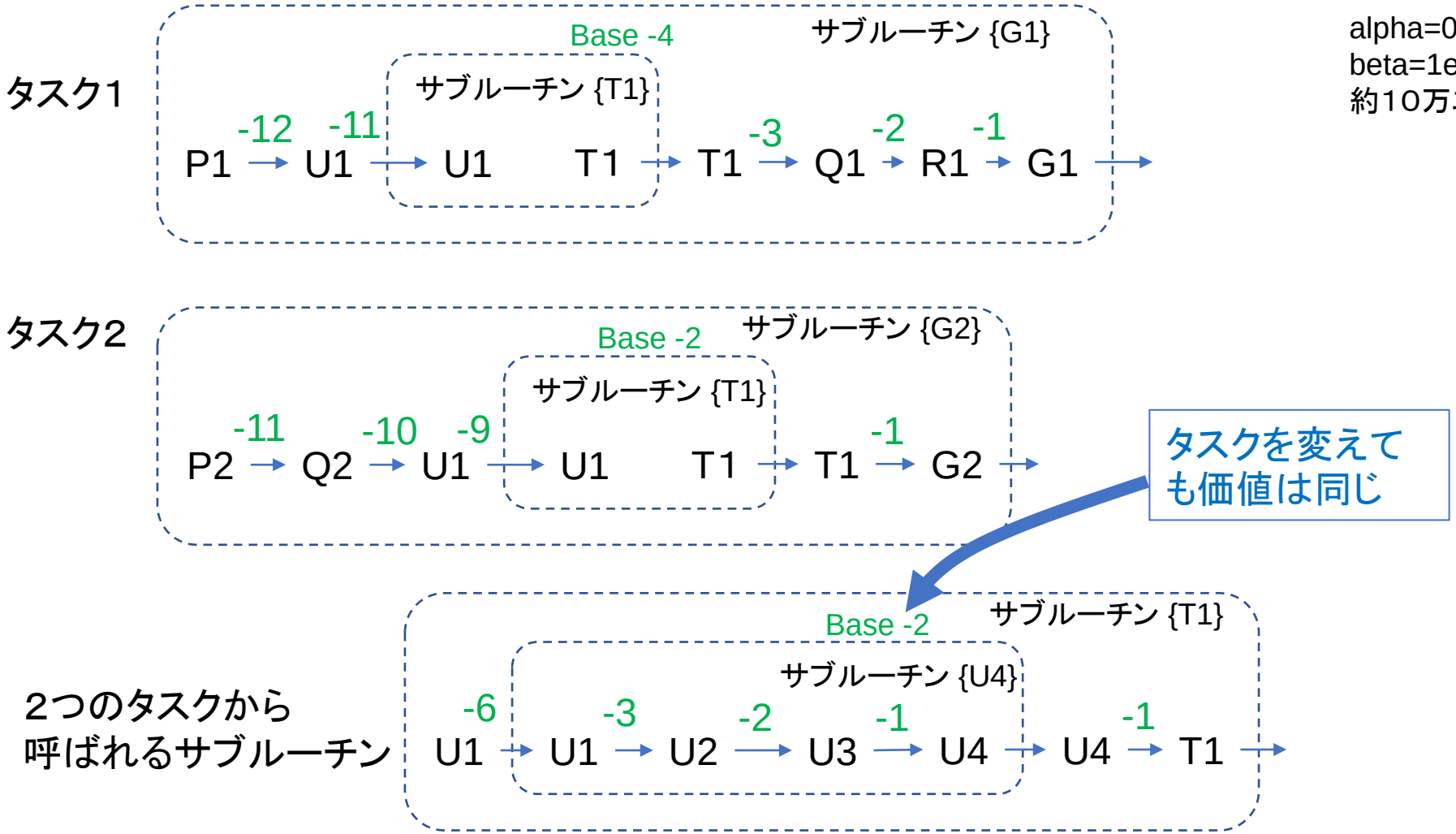
行動の相対価値 $Q_\pi(s,g,a) \equiv E_\pi[\, G - Base_\pi(s0,g0,a0) \,]$

今回実装したアルゴリズムにおける基準値と相対価値の定義



実装方法:
- 実行履歴には1ステップごとにスタックのコピーを保存
- 相対価値の学習は、スタック内のすべての呼び出しの基準値を累積報酬 T から引いた値を目標値にする

テスト用タスク 学習結果



alpha=0.001
beta=1e-5
約10万エピソード

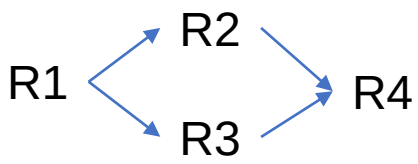
行動計画（プランニング）してから実行するテスト用タスク

- エージェントが持っている宣言的知識・手続き的知識を組み合わせて、**未知のタスクをゼロショット**で解く
- Pro5Lang [一杉+ 第20回 汎用人工知能研究会, 2022] で書かれた行動計画プログラム [一杉+ JSAI 2023] を提案アルゴリズム用に書き換えて動作確認

- Pro5Lang は自律的プログラム合成の対象となるプログラミング言語
  - Cf. プロダクションシステム(Soar, ACT-R)
  - 将来はプログラムを経験から自律的に獲得（現在は手で与える）

行動計画のテスト用タスク

タスク: R1 から R4 に移動
（ゼロショットで解く）



環境の地図

エージェントは
「R1 から R2 にいく」と「R2 から R4 にいく」
または
「R1 から R3 にいく」と「R3 から R4 にいく」
のどちらかの手順で目標を達成

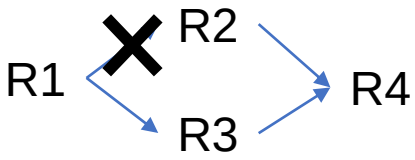
```
// 1つ隣の場所に移動するための行動ルール
1: rule(R1 から R2 にいく, GoTo(R2));
2: rule(R2 から R4 にいく, GoTo(R4));
3: rule(R1 から R3 にいく, GoTo(R3));
4: rule(R2 から R4 にいく, GoTo(R4));
// 行動計画に使われる知識
5: rule(w(), w(a(A(+,+))), exit(a(Failed)));
6: rule(R1 から R2 に移動可能);
7: rule(R2 から R4 に移動可能));
8: rule(R1 から R3 に移動可能));
9: rule(R3 から R4 に移動可能));
```

手続き的知識

宣言的知識

エージェントが持つ
地図に関する知識の
疑似コード

R1->R2 が通行不可の場合の学習結果



通行不可の通路を通ろうとしたら
R1 に戻されるものとする

```
// 1つ隣の場所に移動するための行動ルール
1: rule(R1 から R2 にいく, GoTo(R2));
2: rule(R2 から R4 にいく, GoTo(R4));
3: rule(R1 から R3 にいく, GoTo(R3));
4: rule(R2 から R4 にいく, GoTo(R4));
// 行動計画に使われる知識
5: rule(w(), w(a(A(+,+))), exit(a(Failed)));
6: rule(R1 から R2 に移動可能);
7: rule(R2 から R4 に移動可能));
8: rule(R1 から R3 に移動可能));
9: rule(R3 から R4 に移動可能));
```

「R1 から R2 に移動できる」などの
役に立たない知識（≒間違った知識）の価値が下がる

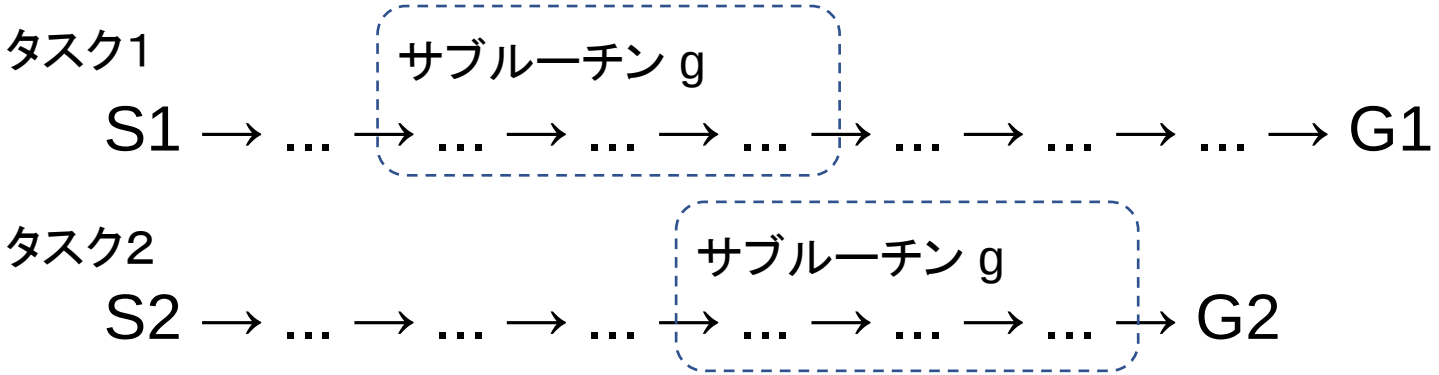
まとめと今後

- 再帰的強化学習アルゴリズム RGoal のモンテカルロ版を改良
- 行動計画の機構を新しい RGoal にあわせて再実装し、簡単な動作確認
  → 自律的な知識獲得に向けて一歩前進
- 今後：
  - 学習アルゴリズムの改良
  - より現実的なタスクでのテスト
  - 神経科学的妥当性の検証

再帰的強化学習は脳型AGIの実現におそらく不可欠、
多くの研究者の参入を期待

以上

階層型強化学習の利点の１つ：サブタスク共有

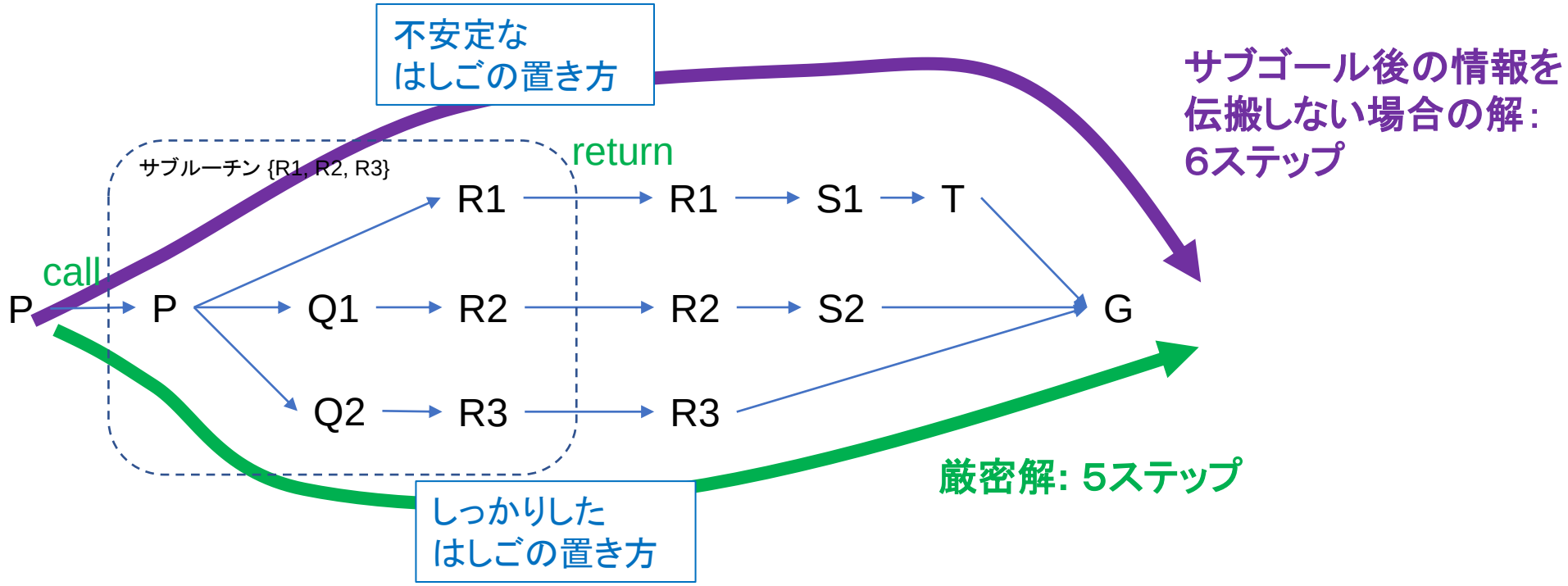


サブルーチン呼び出し文脈が異なっていても、
サブルーチン内の最適方策はほぼ同じはず

例：はしごを使って枝を切る、はしごを使って電球を替える、・・・

いらすとや
https://www.irasutoya.com/2017/04/blog-post_516.html

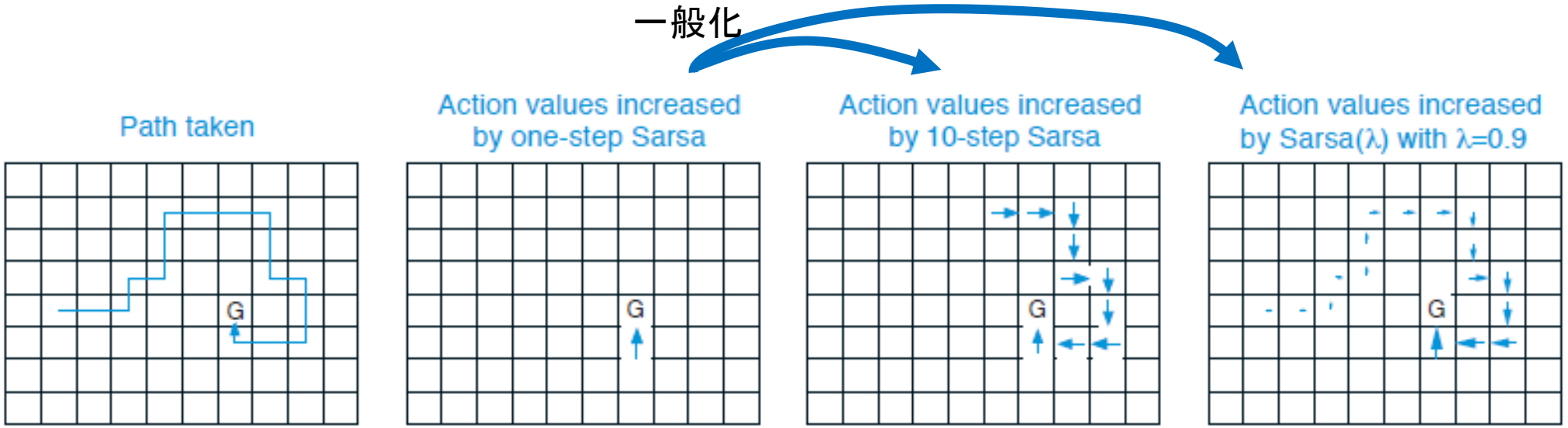
もう１つの設計目標：
終了後の報酬をサブルーチン内部に伝搬させたい



・のちのち報酬が得られる行動をサブルーチン内部でとるようになる
・再利用性の高い行動や知識を経験を通じて環境から獲得するために不可欠であろう

モンテカルロ法，n-step Sarsa，Sarsa(λ)

一般化



one-step Sarsa
＝ 普通の Sarsa

n-step Sarsa

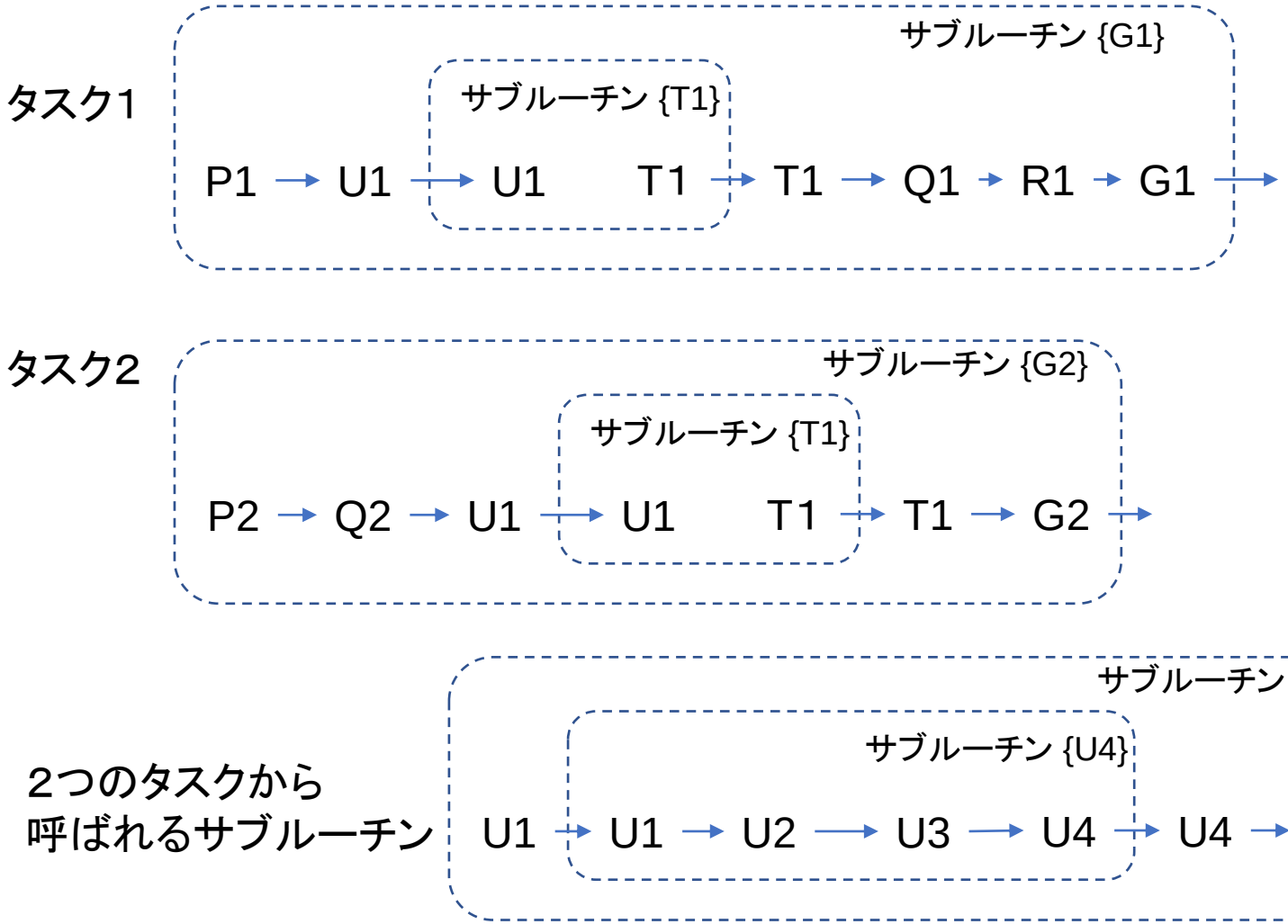
λは適格度トレースにおける適格度の減衰係数

Sarsa(0)
＝ 普通の Sarsa

Sarsa(1)
≒∞-step Sarsa
≒モンテカルロ法

Reinforcement Learning: An Introduction, second edition
Richard S. Sutton and Andrew G. Barto

テスト用タスク




```
StateN g1 = w(a(1,G));
StateN g2 = w(a(2,G));
StateN g3 = w(a(1,T));
StateN g4 = w(a(4,U));

setRulesName("g1");
rule(w(a(1,P)), g1, set(a(1,U))); // 1
rule(w(a(1,U)), g1, call(g3));    // 2
rule(w(a(1,T)), g1, set(a(1,Q))); // 3
rule(w(a(1,Q)), g1, set(a(1,R))); // 4
rule(w(a(1,R)), g1, set(a(1,G))); // 5

setRulesName("g2");
rule(w(a(2,P)), g2, set(a(2,Q))); // 1
rule(w(a(2,Q)), g2, set(a(1,U))); // 2
rule(w(a(1,U)), g2, call(g3));    // 3
rule(w(a(1,T)), g2, set(a(2,G))); // 4

setRulesName("g3");
rule(w(a(1,U)), g3, call(g4)); // 1
rule(w(a(4,U)), g3, set(a(1,T))); // 2

setRulesName("g4");
rule(w(a(1,U)), g4, set(a(2,U))); // 1
rule(w(a(2,U)), g4, set(a(3,U))); // 2
rule(w(a(3,U)), g4, set(a(4,U))); // 3
```

プログラム言語 Pro5Lang ［一杉+ 第20回 汎用人工知能研究会，2022］

- Pro5Lang = 論理型言語＋機械語
  - **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
  - **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
- 「ヒトの前頭前野にあるプログラム」を定義するための言語
- エージェントが自律的獲得しやすい言語仕様になる必要がある

```
1: k(e(Yesterday, Home, Chocolate, Exists, 0,  0));  // きのうチョコレートがあった
2: k(e(Yesterday, Home, Snack, Exists, 0,  0));      // きのうスナックがあった
3: k(e(Today, Home, Brother, NotEat, Chocolate, 0)); // 兄はチョコレートを食べてない

4: g = w(a(Today, Home, PLS, Exists, 0, 0));         // サブゴール:「きょう家に PLS がある」
5: rule(w(), g, recall(e(Yesterday, Home, PLS, Exists, 0,  0)));
6: rule(w(e(Yesterday, Home, x, Exists, 0,  0)),
        g, set(a(Yesterday, Home, x, Exists, 0,  0)));
7: rule(w(a(Yesterday, Home, x, Exists, 0,  0)),
        g, recall(e(Today, Home, Brother, NotEat, x, 0)));
8: rule(w(a(Yesterday, Home, x, Exists, 0,  0),      // きのう x があり、
          e(Today, Home, Brother, NotEat, x, 0)),    // 兄が x を食べていないならば、
        g, set(a(Today, Home, x, Exists, 0, 0)));    // x がある
```

推論を行うプログラムの例

現在はプログラムを手で与えて
言語仕様の妥当性を検証、
将来は経験から自律的に獲得

プログラム＝行動価値関数Q(s, g, a)を圧縮したもの

[一杉+ 第10回 汎用人工知能研究会, 2018]

X \ Y	0	1	2	3	4
0	2.0	1.0	1.0	3.0	1.0
1	1.0	2.0	1.0	3.0	1.0
2	1.0	1.0	2.0	3.0	1.0
3	1.0	1.0	1.0	4.0	1.0
4	1.0	1.0	1.0	3.0	2.0

パターン	値
(3,3)	4.0
(X,3)	3.0
(X,X)	2.0
(X,Y)	1.0

サイズ 5 x 5 = 25 のテーブルが4個のルールに圧縮

変数を使ってテーブルを圧縮
→ 記号AIと同様の高い汎化能力

機械学習の手法で圧縮可能:
2層ベイジアンネットを使った圧縮: [一杉+ 第15回 汎用人工知能研究会(SIG-AGI), 2020]
k-means 法に似た方法で圧縮: [一杉+ 第22回 汎用人工知能研究会(SIG-AGI), 2022]

行動計画プログラムの再実装と動作確認

- JSAI 2023 で発表した Pro5Lang による行動計画のプログラムをいろいろ書き直した。
  - RGoal の exit 命令を使った書き方に変更した。（fail の機構は廃止。）
  - 後ろ向き連鎖ではなく前向き連鎖で推論するようにした。
  - 「A から B に到達可能」という知識を宣言的知識ではなく手続き的知識で表現することにした。
- 動作確認したこと：
  - 実際の環境と一致しない推論規則の価値が下がることを確認。環境への接地。
    → ただし、間違っていない推論規則の価値もテスト環境によっては下がる。（個々の知識の使われ方が独立でない場合。）
  - 複数の可能性の中から実行可能な行動系列を探索する動作を確認。

行動計画　前向き連鎖

```
1: fp = w(a(F(s,+,g)));
2: rule(w(a(Failed)), fp, exit(a(Failed)));
// Start planning.
3: rule(w(), fp, call(a(A(s,+))));
4: rule(w(a(A(s,g’))), fp, set(a(F(s,g’,g’))));
// Loop.
5: rule(w(a(F(s,+,g’))), fp, call(a(A(g’,+))));
6: rule(w(a(A(g’,g”))), fp, recall(e(F(s,+,g’))));
7: rule(w(a(A(g’,g”)), e(F(s,+,g’))),
fp, set(a(F(s,g’,g”))));
```

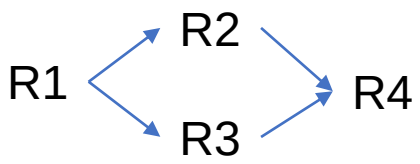
A(s, g) :「s から g に到達できる」
F(s, g', g) :「s から g' 経由で g に到達できる」

推論規則　　A(g', g''), F(s, +, g') ├ F(s, g', g'')

行動計画：知識 A(s,g) を組み合わせて、与えられた S, G に対して F(S, g', G) を満たす g' を見つけること

行動計画のテスト用タスク

タスク: R1 から R4 に移動



環境の地図

エージェントは
「R1 から R2 にいく」と「R2 から R4 にいく」
または
「R1 から R3 にいく」と「R3 から R4 にいく」
のどちらかの手順で目標を達成

```
// 1つ隣の場所に移動するための行動ルール
1: rule(w(a(R1)), w(a(R2)), GoTo(R2));
2: rule(w(a(R2)), w(a(R4)), GoTo(R4));
3: rule(w(a(R1)), w(a(R3)), GoTo(R3));
4: rule(w(a(R3)), w(a(R4)), GoTo(R4));
// 行動計画に使われる知識
5: rule(w(), w(a(A(+,+))), exit(a(Failed)));
6: rule(w(), w(a(A(R1,+))), set(a(A(R1,R2))));
7: rule(w(), w(a(A(R2,+))), set(a(A(R2,R4))));
8: rule(w(), w(a(A(R1,+))), set(a(A(R1,R3))));
9: rule(w(), w(a(A(R3,+))), set(a(A(R3,R4))));
```

手続き的知識

宣言的知識

エージェントが持つ
地図に関する知識

MAXQ 価値関数分解[Dietterich 2000]との関係

- MAXQ 価値関数分解と提案アルゴリズムの相対価値表現はかなりよく似ている
  - completion function C(i,s,a) が提案アルゴリズムの基準値にほぼ相当
  - MAXQ はサブルーチンの外の報酬が中に伝搬しない
  - 「基準値」の考え方の方が設計の自由度が高い
- 学習アルゴリズムはかなり違う

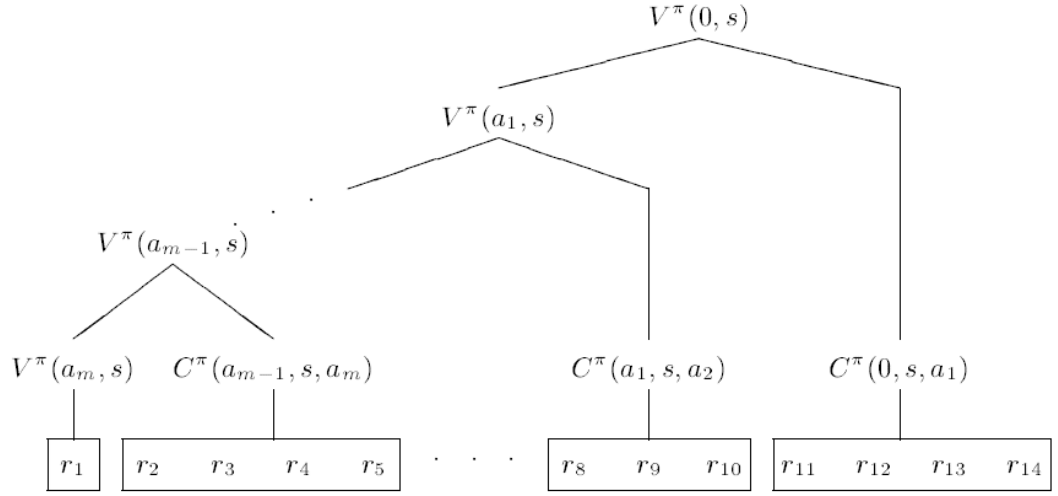


Figure 5: The MAXQ decomposition; $r_1, \ldots, r_{14}$ denote the sequence of rewards received from primitive actions at times $1, \ldots, 14$.

C(i, s, a) はサブタスク a 終了後、
親タスク i 終了までの間の報酬