階層型強化学習
RGoal アーキテクチャへの
再帰呼び出し用スタックの導入

2019-06-06 人工知能学会全国大会

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

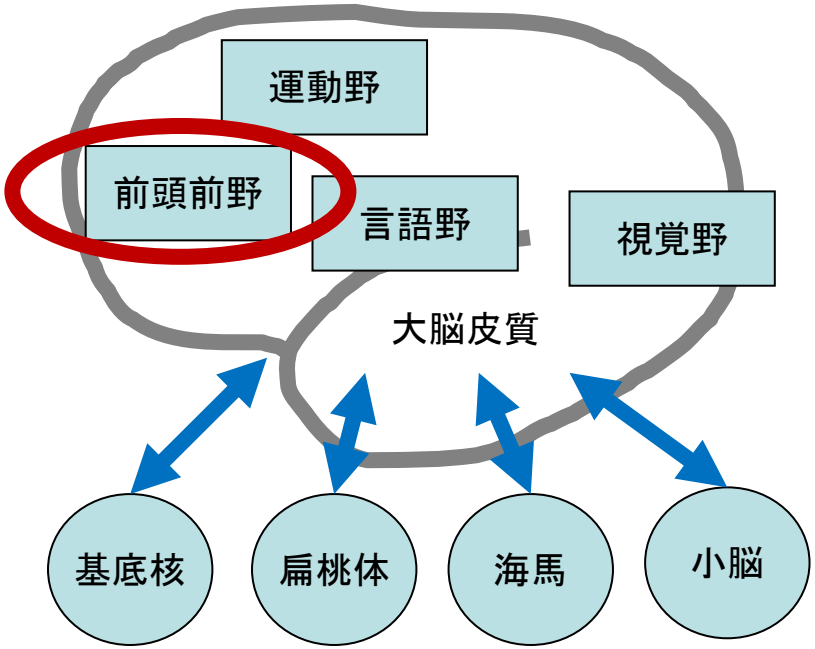
*1 産業技術総合研究所 人工知能研究センター
National Institute of Advanced Industrial Science and Technology (AIST), AIRC

*2 成蹊大学 理工学部 情報科学科
Department of Computer and Information Science, Faculty of Science and Technology, Seikei University

私の研究の長期的目標

- 脳を模倣して「人間のような知能を持つ機械」を作る。

本研究:
前頭前野による
行動計画の機能の
再現を目指す。



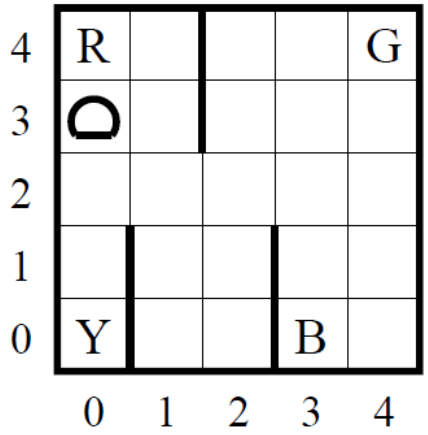
RGoal アーキテクチャ

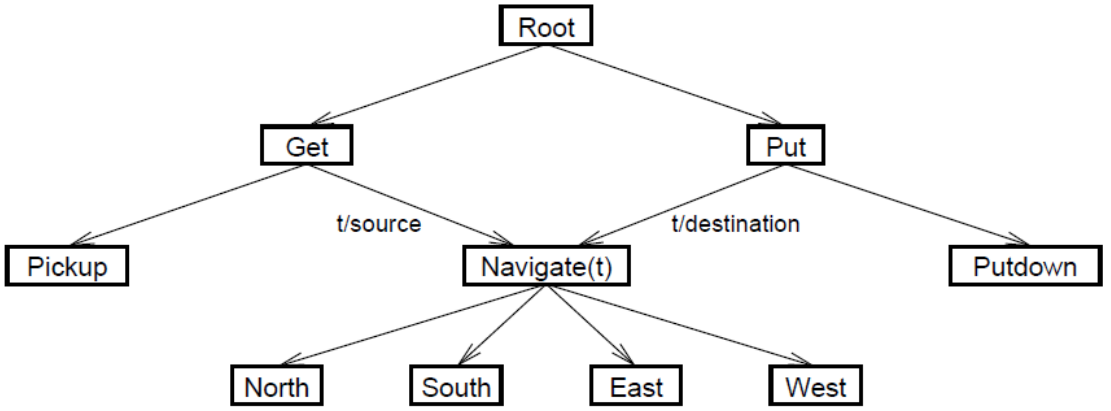
[一杉 et al. 2018 8月 汎用人工知能研究会]

- 特徴:
  - 階層型強化学習
  - **再帰的にサブゴールを設定可能**
    サブゴール設定＝サブルーチン呼び出し
    - プログラミング言語のサブルーチンに似たもの
    - マルチタスク環境での共有・再利用により学習を加速
  - **思考モード**で脳内シミュレーション
    - 未経験のタスクも過去に得た知識を組み合わせて解く
  - スタックなしで、シンプルなアルゴリズムで動作
    （→ **今回はスタックを導入**）

MAXQ [Dietterich 2000]

- 多層の階層型強化学習アーキテクチャ。
- 階層の深さは固定。設計者が与える。
- 行動価値関数の計算が複雑。
- 探索空間に制約があり近似解しか得られない。





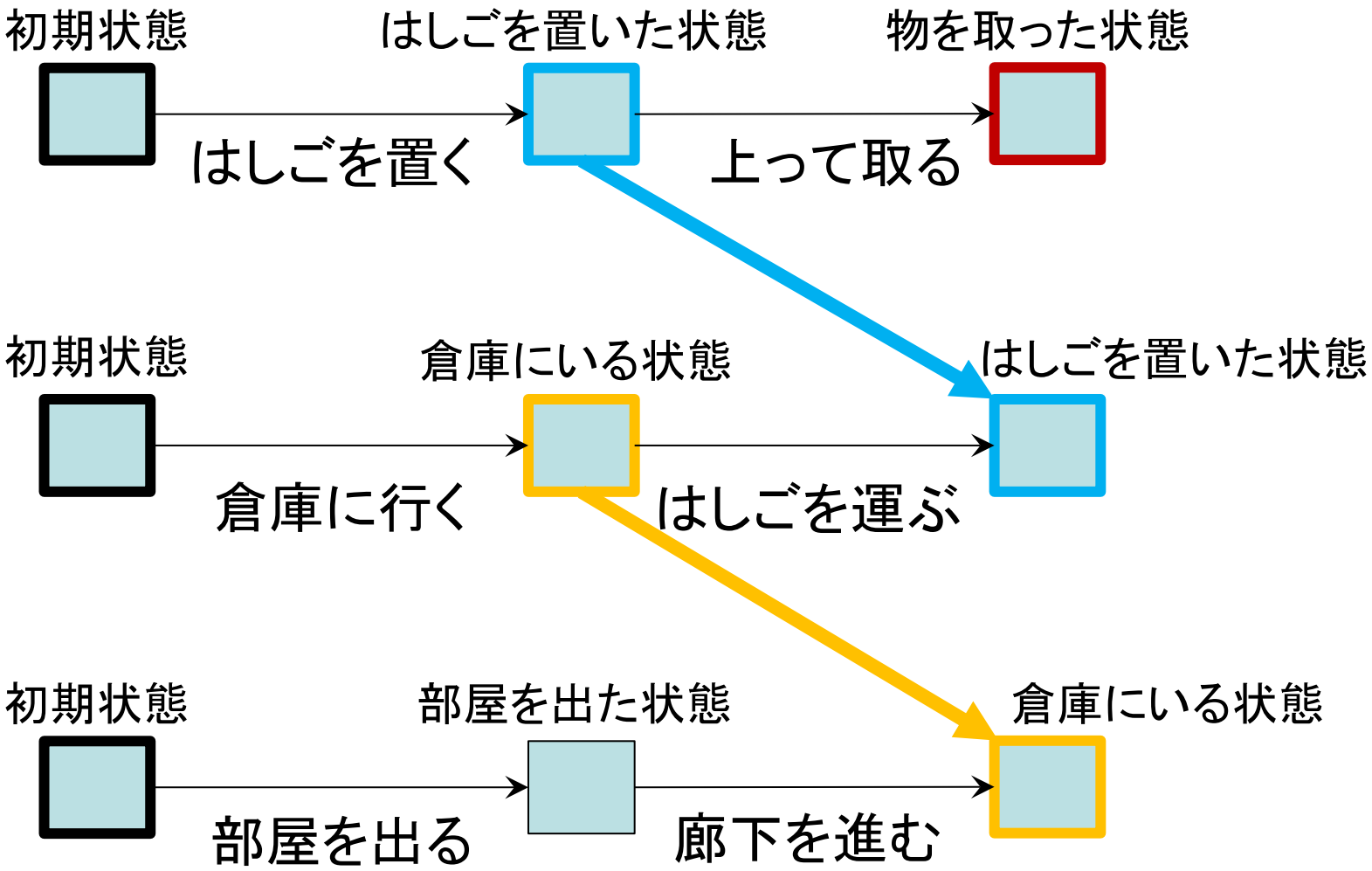
階層型強化学習の利点[Dietterich 2000]

1. **サブタスク共有**:
サブルーチンをタスク間で共有することで学習を加速。

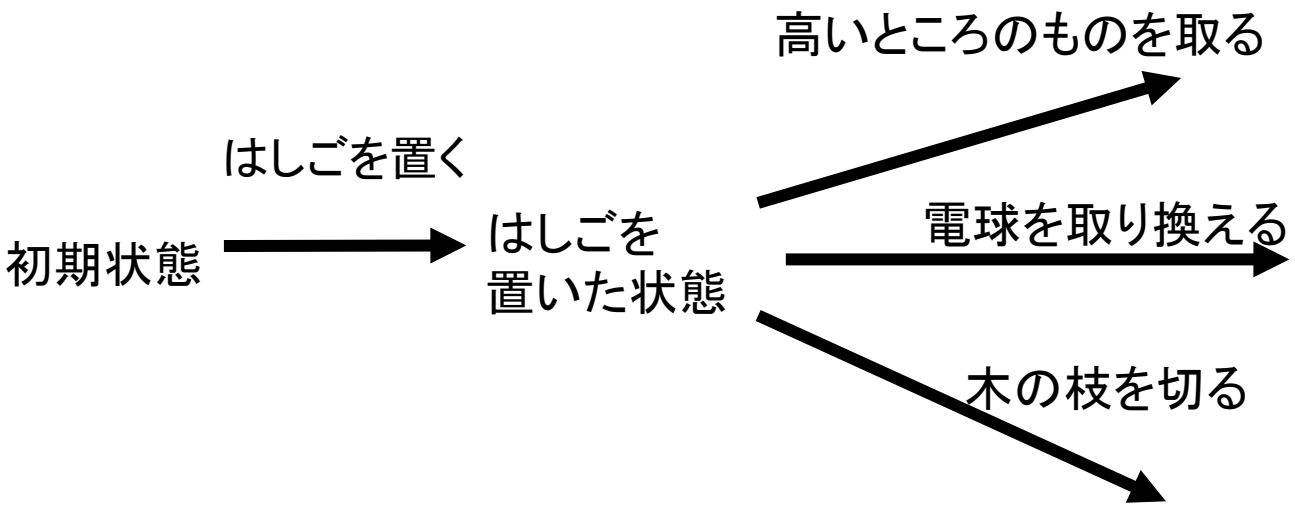
2. **時間抽象**:
行動系列をサブルーチンとして抽象化することで学習を加速。

3. **状態抽象**:
サブルーチンごとに実行に関係ない情報を無視することで学習を加速。

人間が再帰的にサブゴールを設定する例

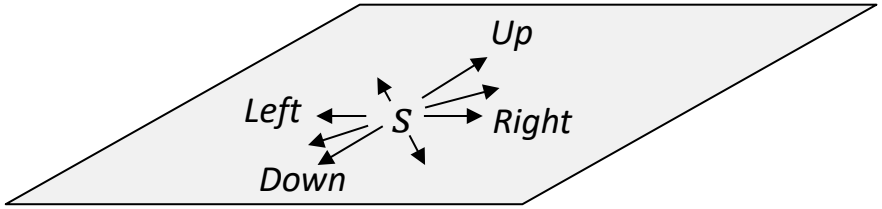


サブルーチン共有の例



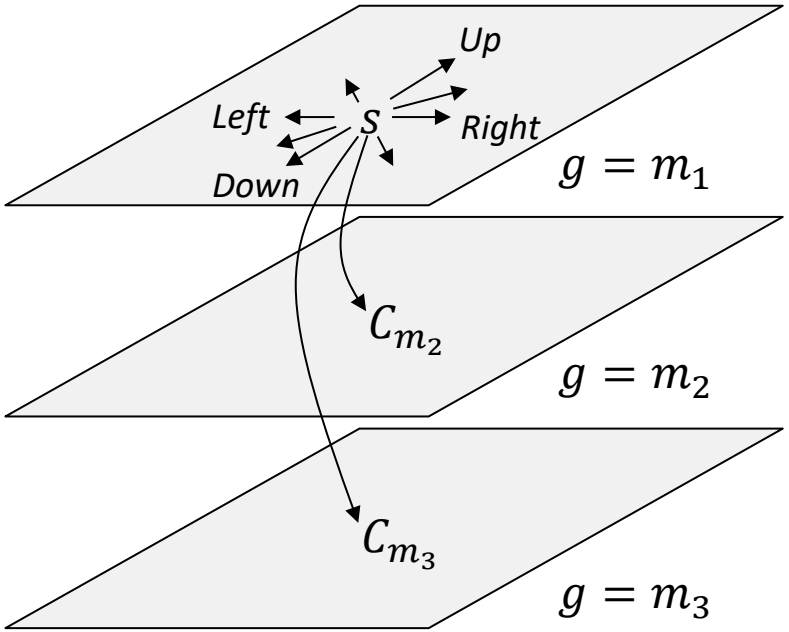
最終目的は違っていても、
「はしごを置く」というサブルーチンは共通

拡張状態行動空間 [Levy and Shimkin 2011]



$\mathcal{S} = \{(0,0), (0,1), \dots\}$
$\mathcal{A} = \{Up, Down, Right, Left, \dots\}$

拡張



もともとの状態 s とサブゴール g の組 $\tilde{s} = (s, g)$ を
拡張された状態と見なす。

$\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{M}$
$\tilde{\mathcal{A}} = \mathcal{A} \cup \mathcal{C}_{\mathcal{M}}$
$\mathcal{M} = \{m_1, m_2, \dots\} \subseteq \mathcal{S}$
$\mathcal{C}_{\mathcal{M}} = \{C_{m_1}, C_{m_2}, \dots\}$

数学的構造は通常のMDP と同じなので、MDP を前提とした様々な理論的帰結（例えば厳密解への収束性）や強化学習の高速化技術（例えば関数近似や適格度トレース）が利用可能。

価値関数分解[Singh 1992][Dietterich 2000]

- 関数 Q をサブゴール g の到着の前後で分解

到達前　　到達後

$$Q_G^{\pi}((s,g),a) = Q^{\pi}(s,g,a) + V_G^{\pi}(g)$$

ただし $V_G^{\pi}(g) = \Sigma_a \pi\big((g,G),a\big) Q^{\pi}(g,G,a)$

- **関数 Q(s,g,a) はもともとのゴール G に依存しないため、異なるタスク間で共有できる。**
  - 例:「はしごを置く」という行動のコスト(労力)は、はしごを使う目的に依存しない。
  - パラメタが減って、学習が速くなる。

Sarsa による学習

（予稿集の説明は不完全なので補足）

$$Q(\tilde{s},\tilde{a}) \leftarrow Q(\tilde{s},\tilde{a}) + \alpha(r + Q(\tilde{s}',\tilde{a}') - Q(\tilde{s},\tilde{a}))$$

サブゴールが g 、スタックの中身が g1,g2,...gn のとき、
エージェントの移動経路は s->g->g1->g2->...->gn である。
サブルーチン g' を呼び出すと移動経路は s->g'->g->g1->g2->...->gn に変化。
したがって、以下の式が成り立つ。

$$\begin{aligned}
&Q_{g_n}(\tilde{s}',\tilde{a}') - Q_{g_n}(\tilde{s},\tilde{a}) \\
&= \left(Q(s',g',\tilde{a}') + V_g(g') + V_{g_1}(g) + V_{g_2}(g_1) + \cdots + V_{g_n}(g_{n-1})\right) \\
&-\left(Q(s,g,\tilde{a}) + V_{g_1}(g) + V_{g_2}(g_1) + \cdots + V_{g_n}(g_{n-1})\right) \\
&= Q(s',g',\tilde{a}') - Q(s,g,\tilde{a}) + V_g(g')
\end{aligned}$$

この式は行動がサブルーチン呼び出しではなくプリミティブの時も成り立つ。
これから、下記の学習則が導かれる。

$$Q(s,g,a) \leftarrow Q(s,g,a) + \alpha(r + Q(s',g',a') - Q(s,g,a) + V_g(g'))$$

Softmax による行動選択

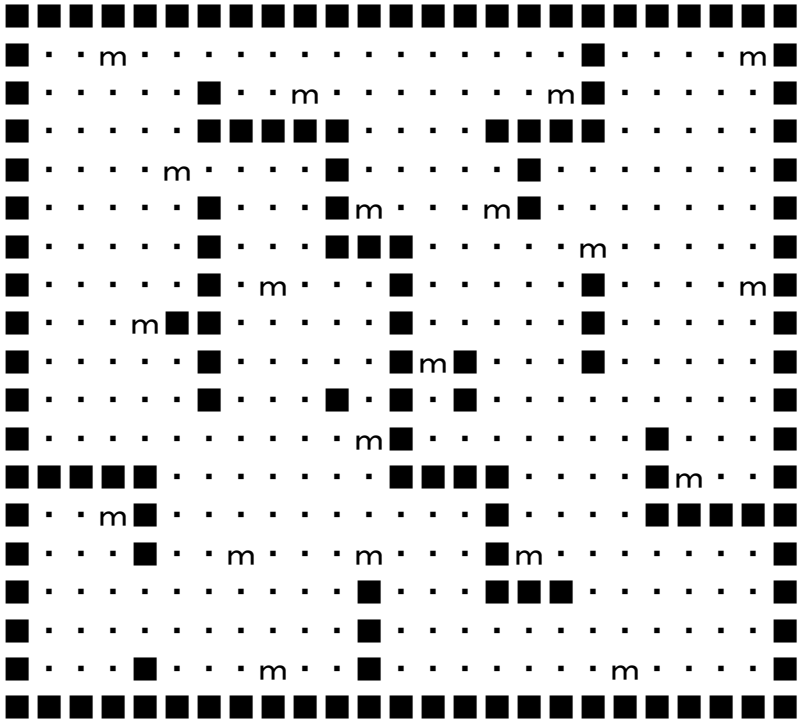
$$\pi(\tilde{s},\tilde{a}) = \frac{\exp\left(\beta Q(\tilde{s},\tilde{a})\right)}{\Sigma_{a'}\exp\left(\beta Q(\tilde{s},a')\right)}$$

$$= \frac{\exp\left(\beta\left(Q(s,g,a) + V_g(g')\right)\right)}{\Sigma_{a'}\exp\left(\beta\left(Q(s,g,a') + V_g(g')\right)\right)}$$

$$= \frac{\exp(\beta V_g(g'))\exp\left(\beta Q(s,g,a)\right)}{\exp(\beta V_g(g'))\,\Sigma_{a'}\exp\left(\beta Q(s,g,a')\right)}$$

$$= \frac{\exp\left(\beta Q(s,g,a)\right)}{\Sigma_{a'}\exp\left(\beta Q(s,g,a')\right)}$$

迷路課題



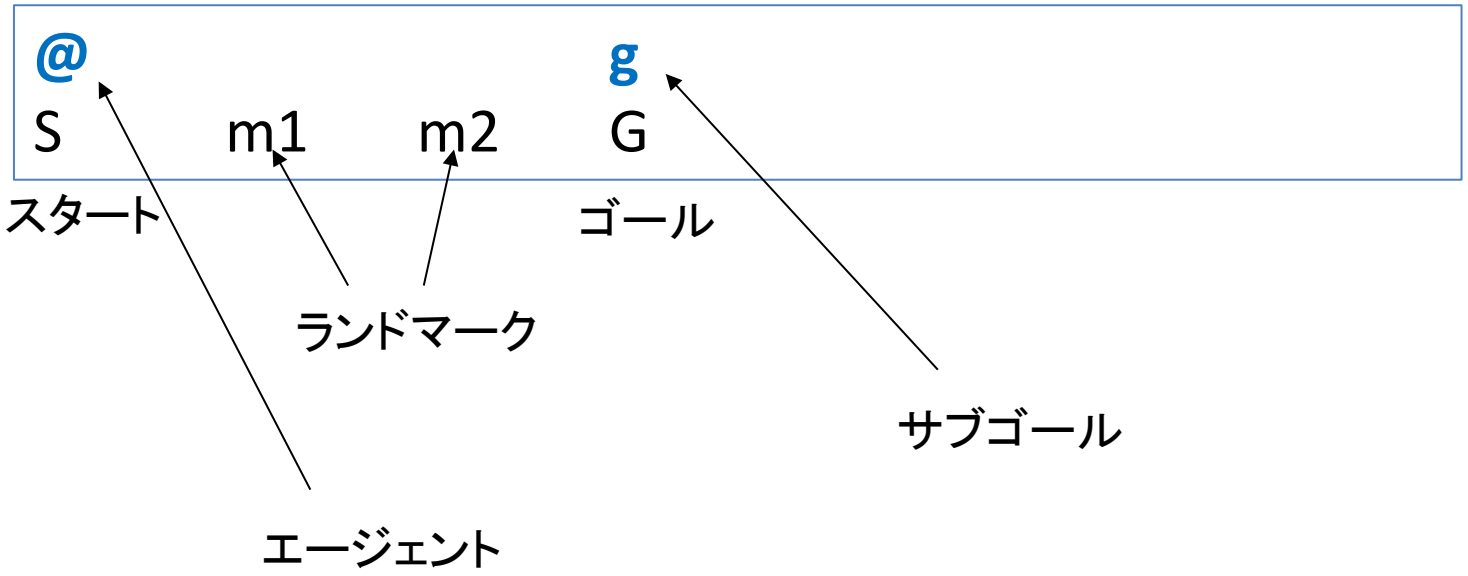
m はランドマーク

マップとランドマークの集合は固定。

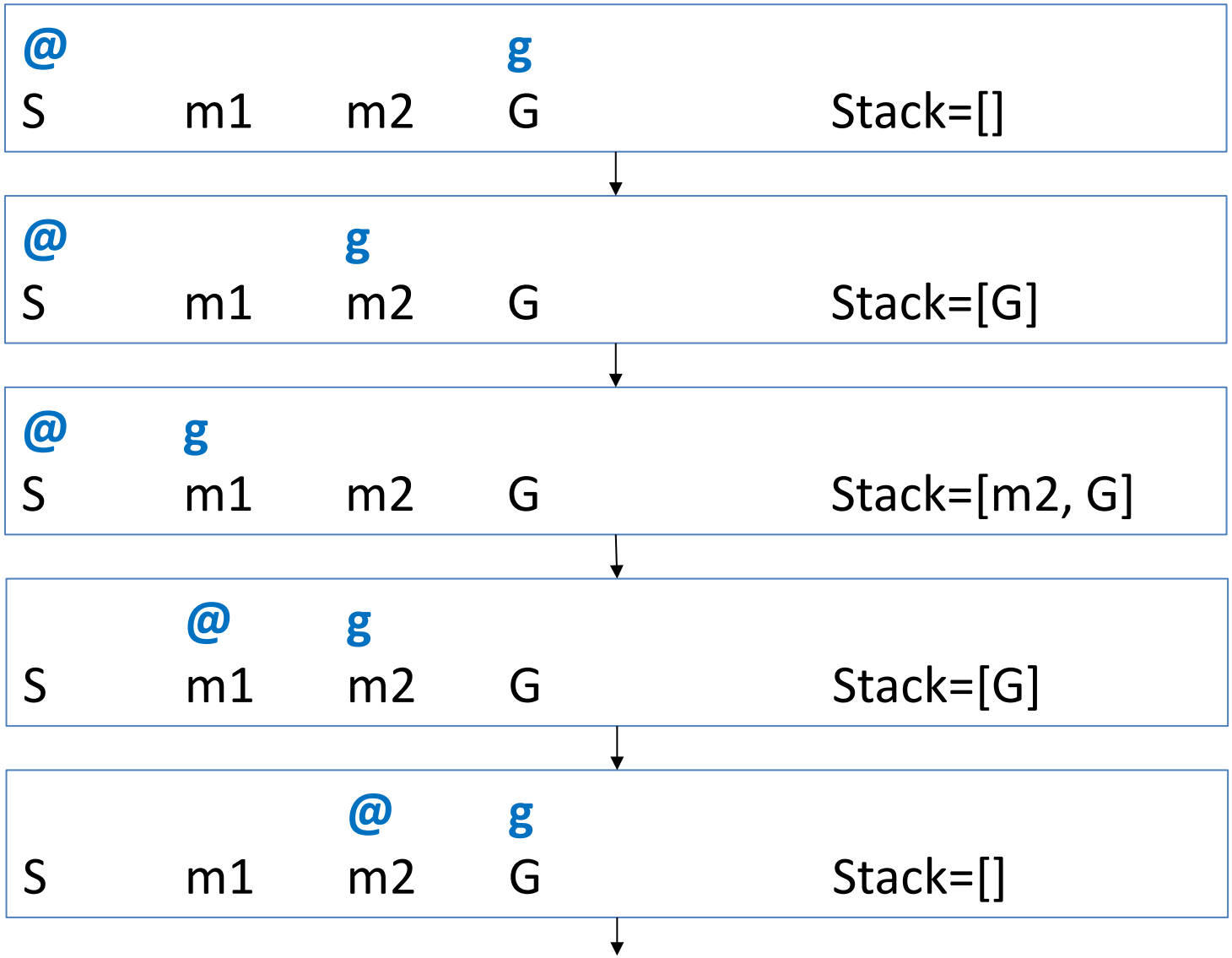
ランドマークは、ゴールやサブゴールになり得る場所。

スタート S とゴール G をエピソードごとにランダムに設定する。

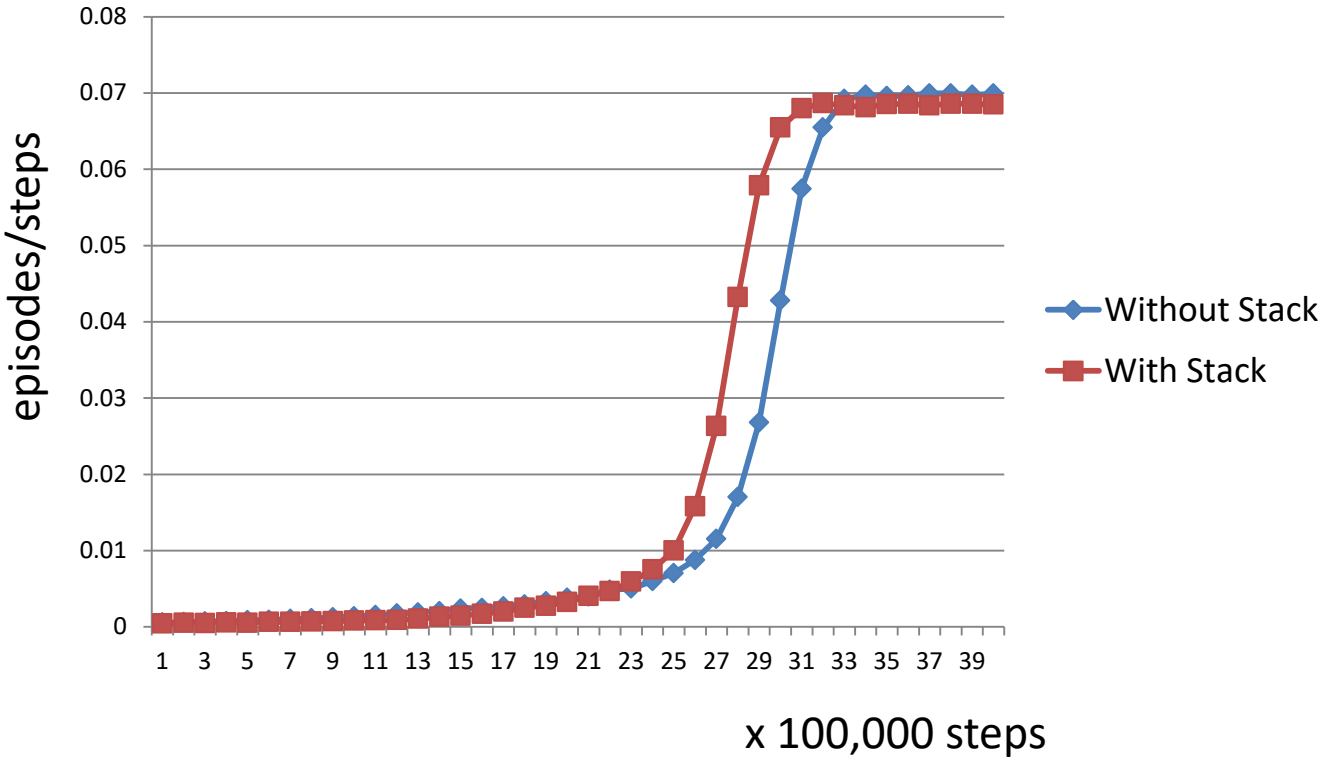
エージェントの動作



エージェントの動作

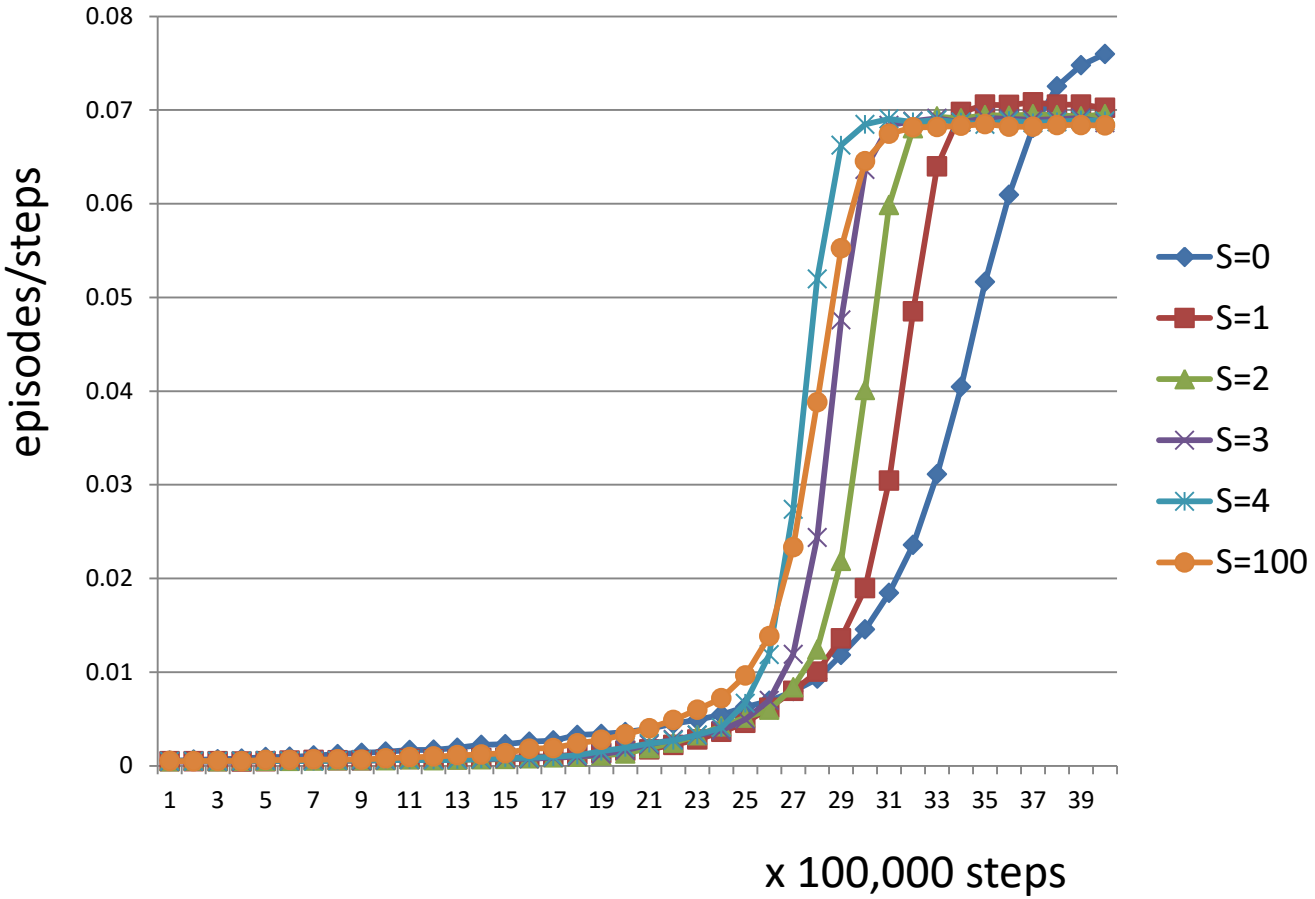

@ g
S m1 m2 G Stack=[]
@ g
S m1 m2 G Stack=[G]
@ g
S m1 m2 G Stack=[m2, G]
@ g
S m1 m2 G Stack=[G]
@ g
S m1 m2 G Stack=[]

実験1：スタックなし版 RGoal [一杉 et al. 2018] との比較



実験条件にもよるが、同程度の早さで収束している。

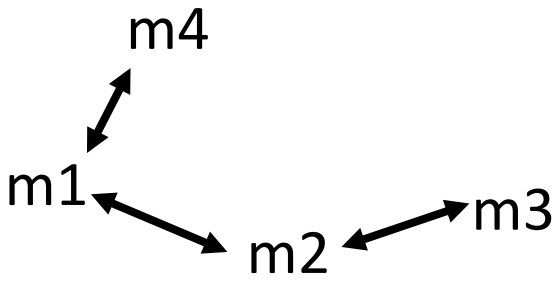
実験2：スタックの深さの上限sと性能の関係



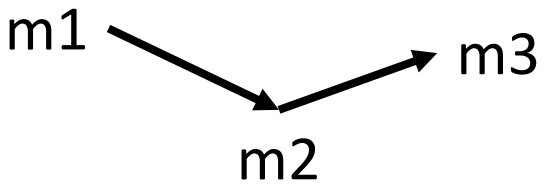
スタックの上限 S が大きいほど収束は早くなる傾向を示している。

思考モード

- 隣接するランドマーク間の最適移動経路は学習済みだとする。



- 離れたランドマーク間の最適移動経路の近似解は、隣接するランドマークをつなげば得られるはず。



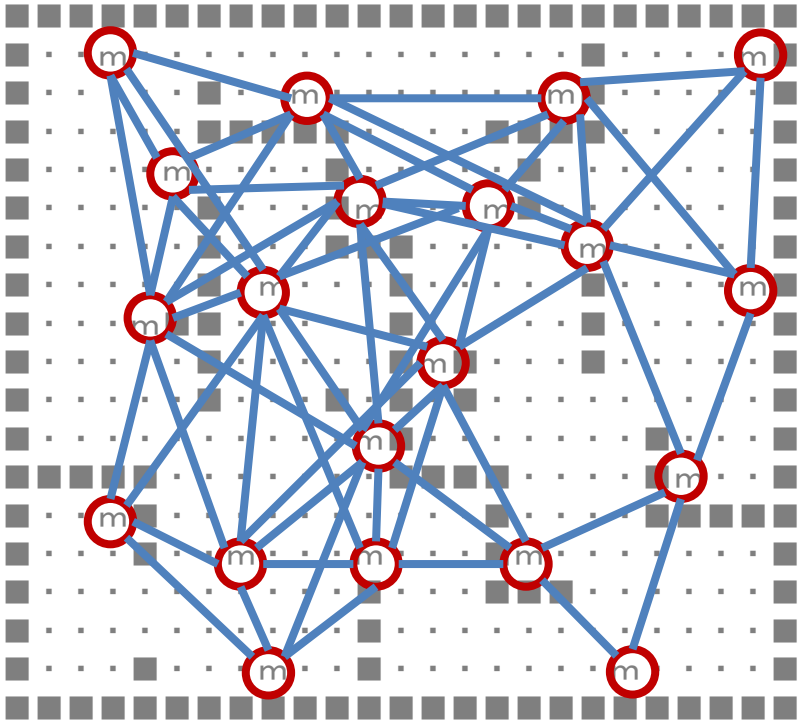
- **この近似解は実際に行動しなくても「脳内シミュレーション」だけで見つけられる。**[Singh 1992]
  - 一種のモデルベース強化学習
  - 演繹推論

思考モードの実装

```
1:  procedure EPISODE(S, G, think-flag)
2:      s ← S; g ← G
3:      stack ← empty
4:      Choose a from s, g using policy derived from Q
5:      while s ≠ G do
6:          # Take action.
7:          if a = RET then
8:              s' ← s; g' ← stack.pop(); r ← 0
9:          else if a is C_m then
10:             stack.push(g)
11:             s' ← s; g' ← m; r ← R^C
12:         else
13:             if think-flag then
14:                 s' ← g; g' ← g; r ← Q(s, g, a)
15:             else
16:                 Take action a, observe r, s'
17:                 g' ← g
18:         # Choose action.
19:         if s' = g' then
20:             a' ← RET
21:         else
22:             Choose a' from s', g'
23:                 using policy derived from Q
24:         # Update
25:         if s = g or (think-flag and a is not C_m) then
26:             # Do nothing.
27:         else
28:             Q(s, g, a) ← Q(s, g, a)
29:               +α(r + Q(s', g', a') − Q(s, g, a) + V_g(g'))
30:         s ← s'; g ← g'; a ← a'
```

- **思考モードはアルゴリズムのわずかな修正だけで実現可能**
- Q(s,g,a) を環境のモデルと見なして、時間を抽象化した脳内シミュレーション

あらかじめ簡単なタスクを事前学習

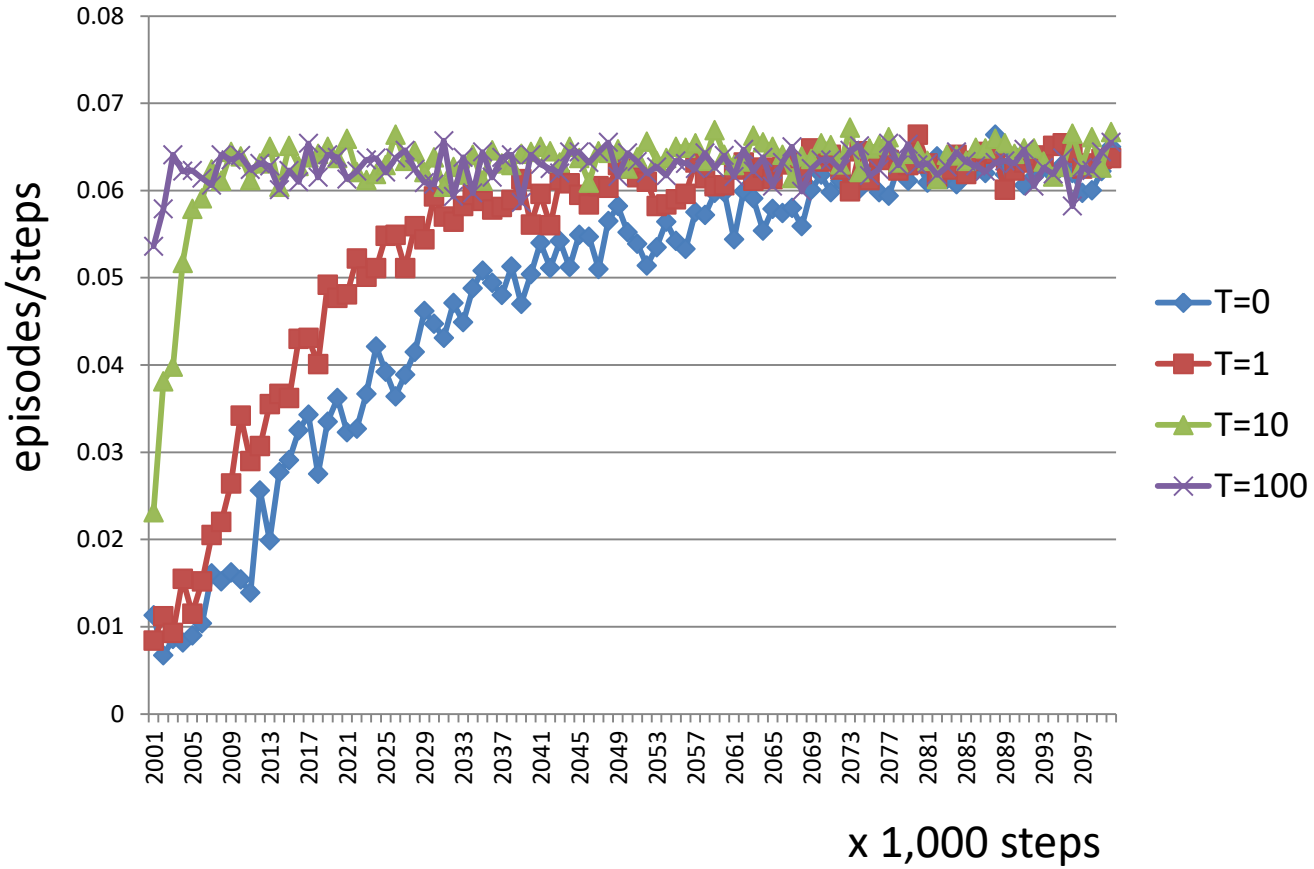


ユークリッド距離が8以内にあるランドマークのペアの間の経路を事前学習

20x19=380個のペアのうちの60個

m はランドマーク

実験3：思考フェーズの長さTと性能の関係



各エピソードの実行直前に、思考モードでエピソードをT回実行

思考フェーズ（脳内シミュレーション）が十分長ければ、未経験のタスクであっても、近似解がゼロショットで得られている。

汎用人工知能につながる RGoal アーキテクチャの有望な性質

- 報酬最大化のために最適なサブゴールを選択する。→ **自律性・目的指向性**
- 帰納論理プログラミングと組み合わせると、サブルーチンの汎用性が増し、知識が増えるたびに解ける問題が組み合わせ爆発的に増える。→ **汎用性**
- 学習結果はサブゴールとスタックを内部状態に持つオートマトンと見なせる。
  → **万能性**

まとめと今後

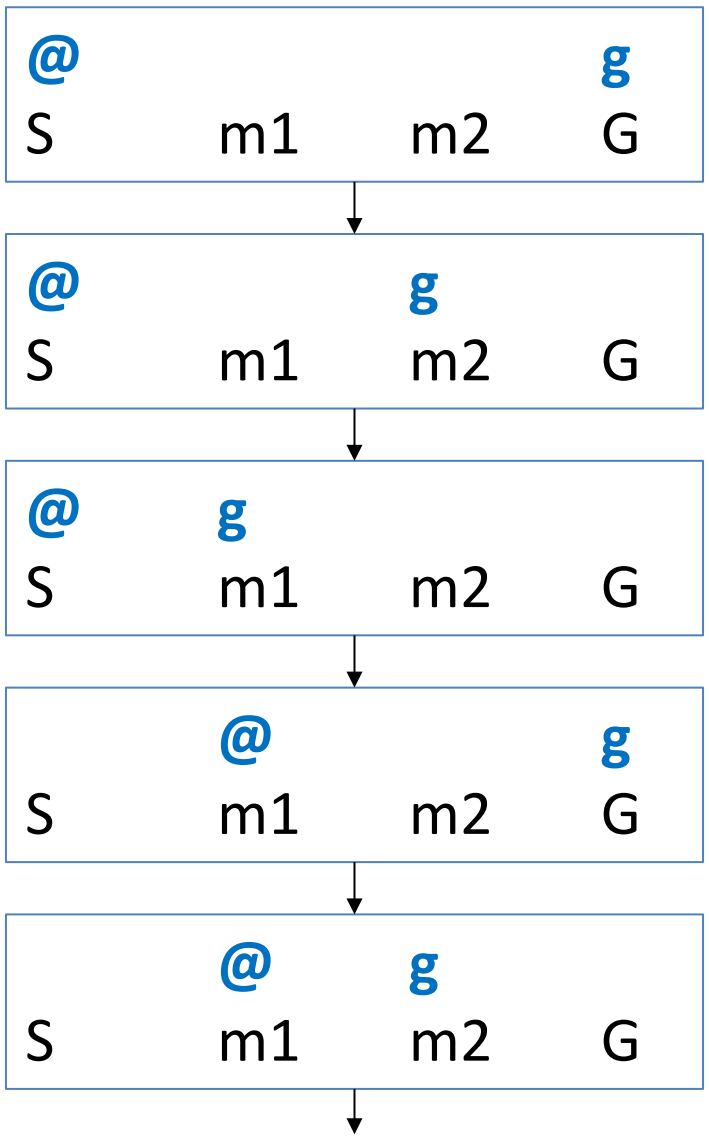
- RGoal アーキテクチャ（スタックあり）を提案
  人間の知能の2つの重要な特性を再現
  - 再帰的サブゴール設定
  - 時間を抽象化した高速なプランニング
- 今後：
  - 記号推論（帰納論理プログラミング）と強化学習の統合
  - 脳型汎用人工知能の中核技術を目指す

予備スライド

先行研究との関係

- 提案アーキテクチャは、階層型強化学習のいくつかのアイデアをシンプルな1つのアーキテクチャに統合
  - 可変時間分解能モデル（時間抽象） [Singh 1992]
  - 再帰的なサブゴール設定 [Kaelbling 1993]
  - 価値関数分解 [Singh 1992][Dietterich 2000]
  - 拡張状態行動空間 [Levy et al. 2011]

スタックなし版での動作



ここまではスタックあり版と同じ

おおもとのゴール G を
サブゴールに再設定