知能システム学
第4回　状態空間法

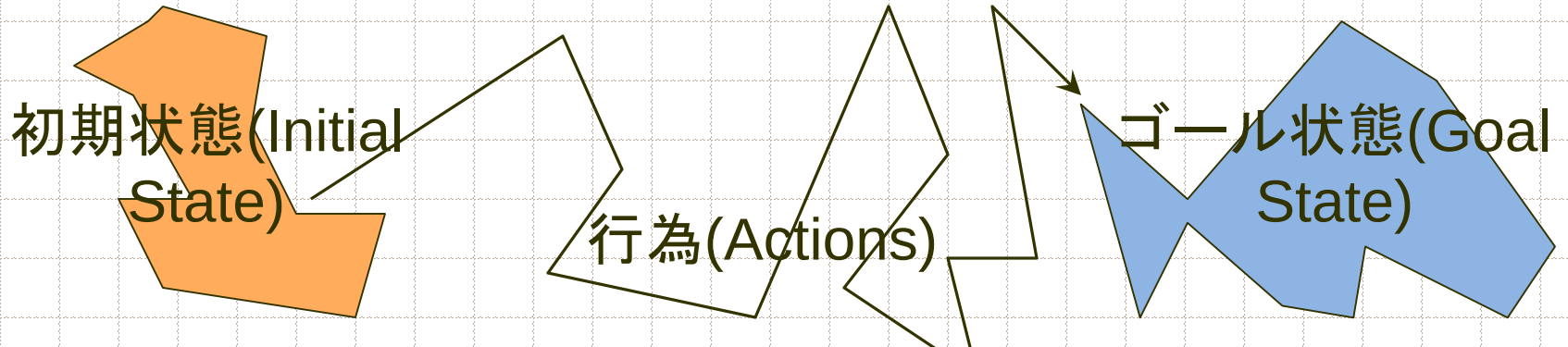
ソフトウェア情報学部
David Ramamonjisoa

目次

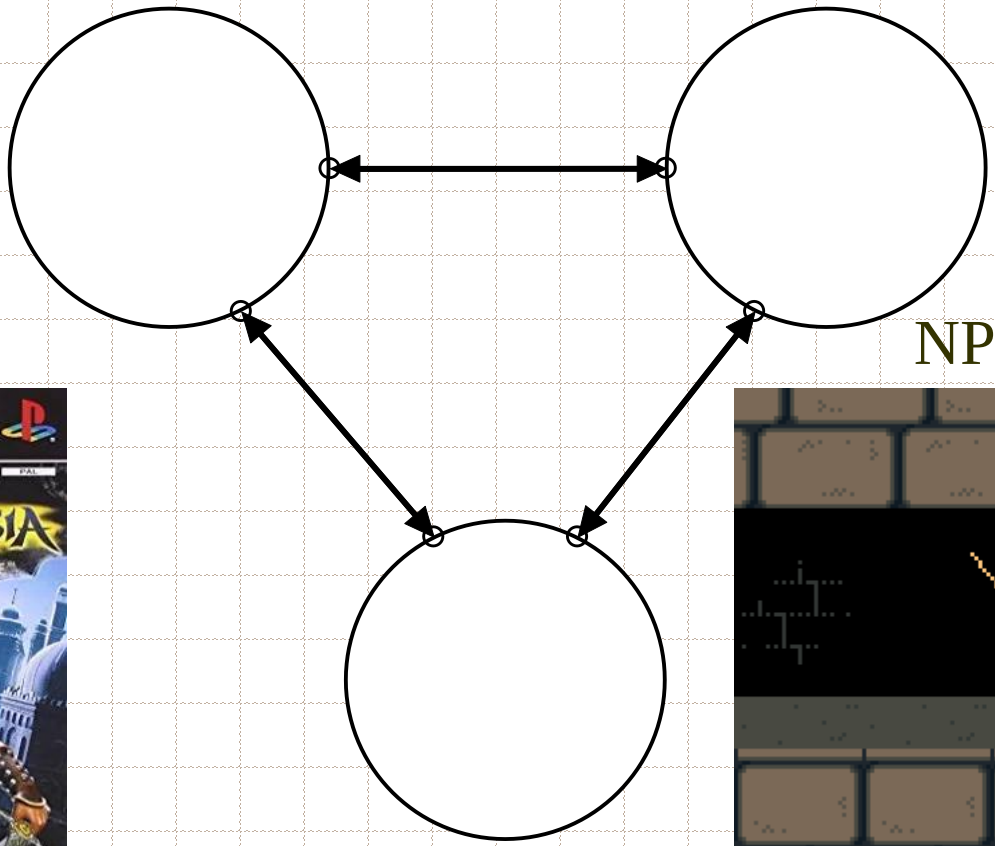
- 復習：状態、状態遷移（FSM）
- 状態空間法：問題を定式化すること
- 例題
- まとめ

ゴールを用いるエージェントの構築

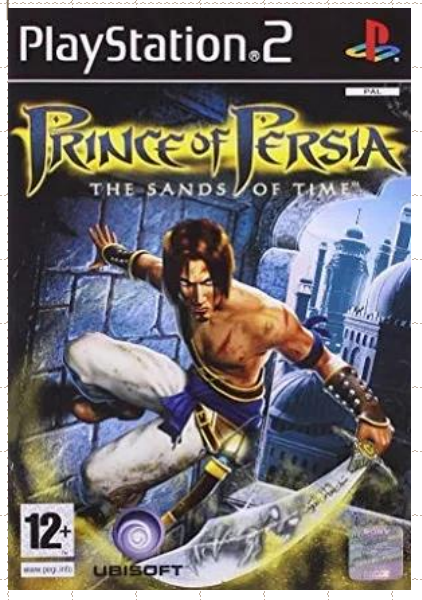
- エージェントの環境を状態で表現する(**state**)?
- エージェントのゴールは何か(**goal** to be achieved?)
- 可能な行為は何かある(What are the **actions**?)
- 問題を解決するためにどのような情報が状態と状態遷移に記述するべきか

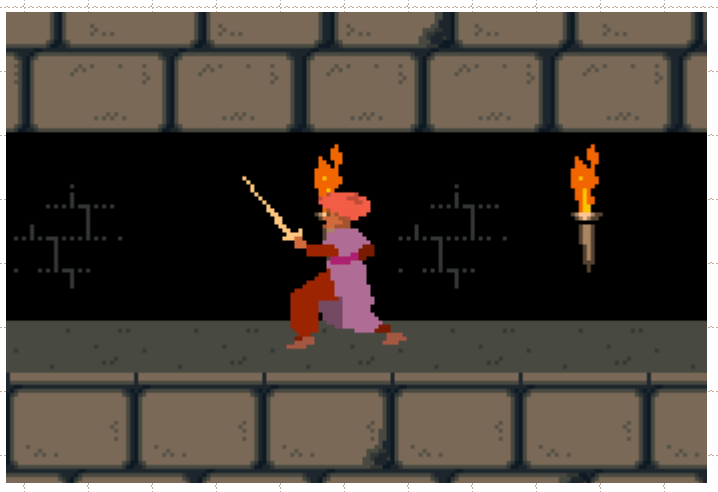


FSM(有限状態マシン): 3つの状態

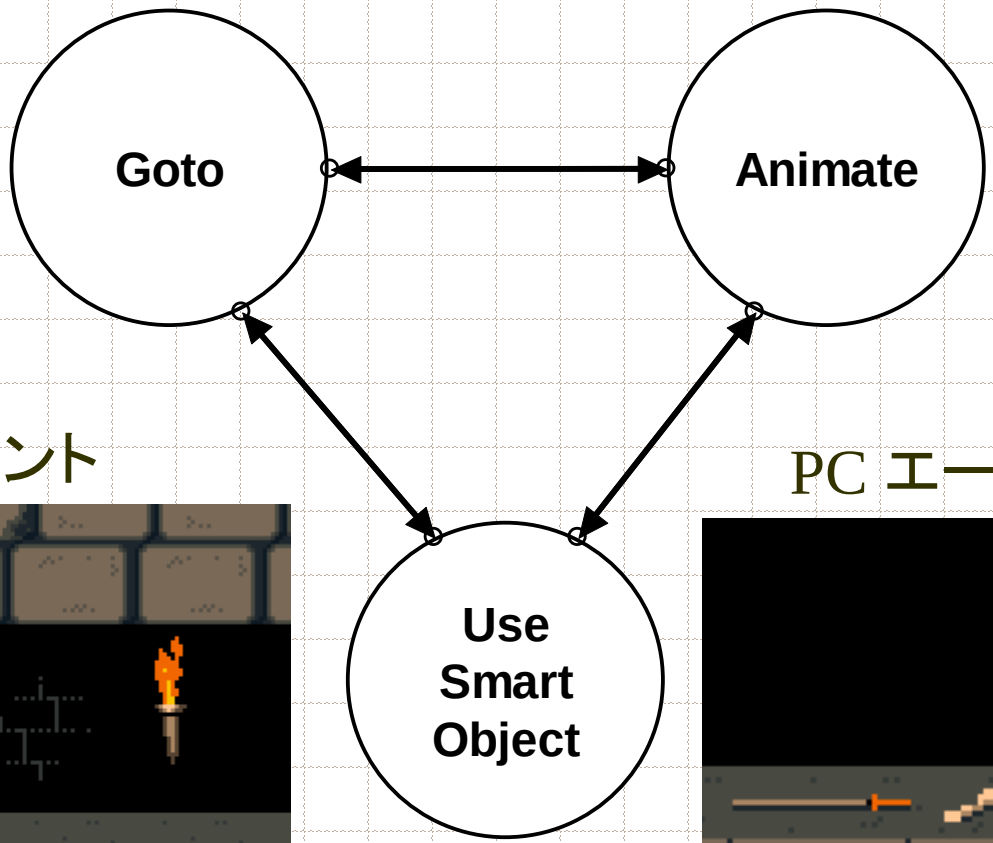


NPC エージェント

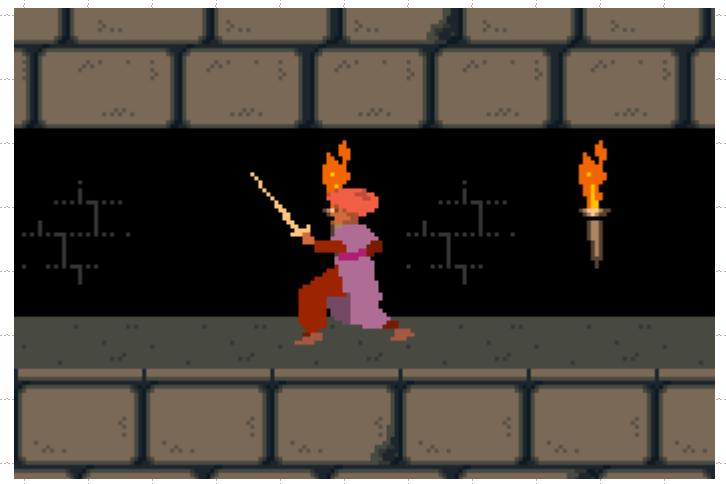




FSM(有限状態マシン):3つの状態

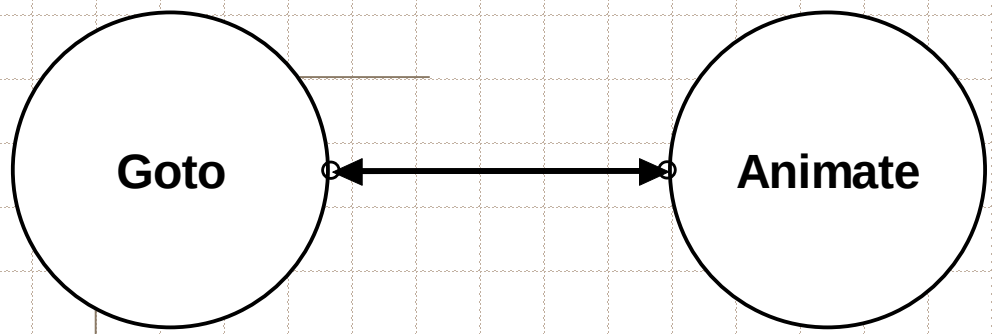


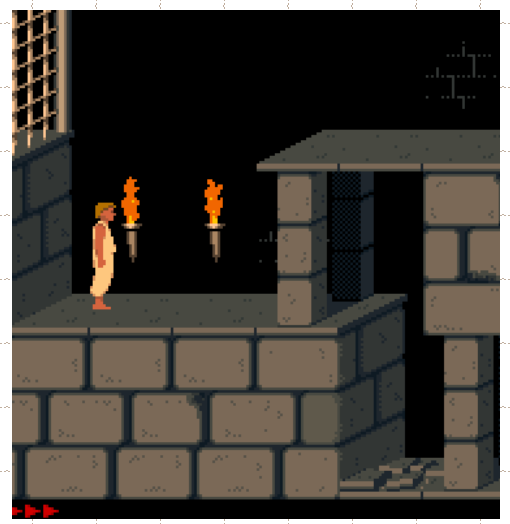
NPC エージェント

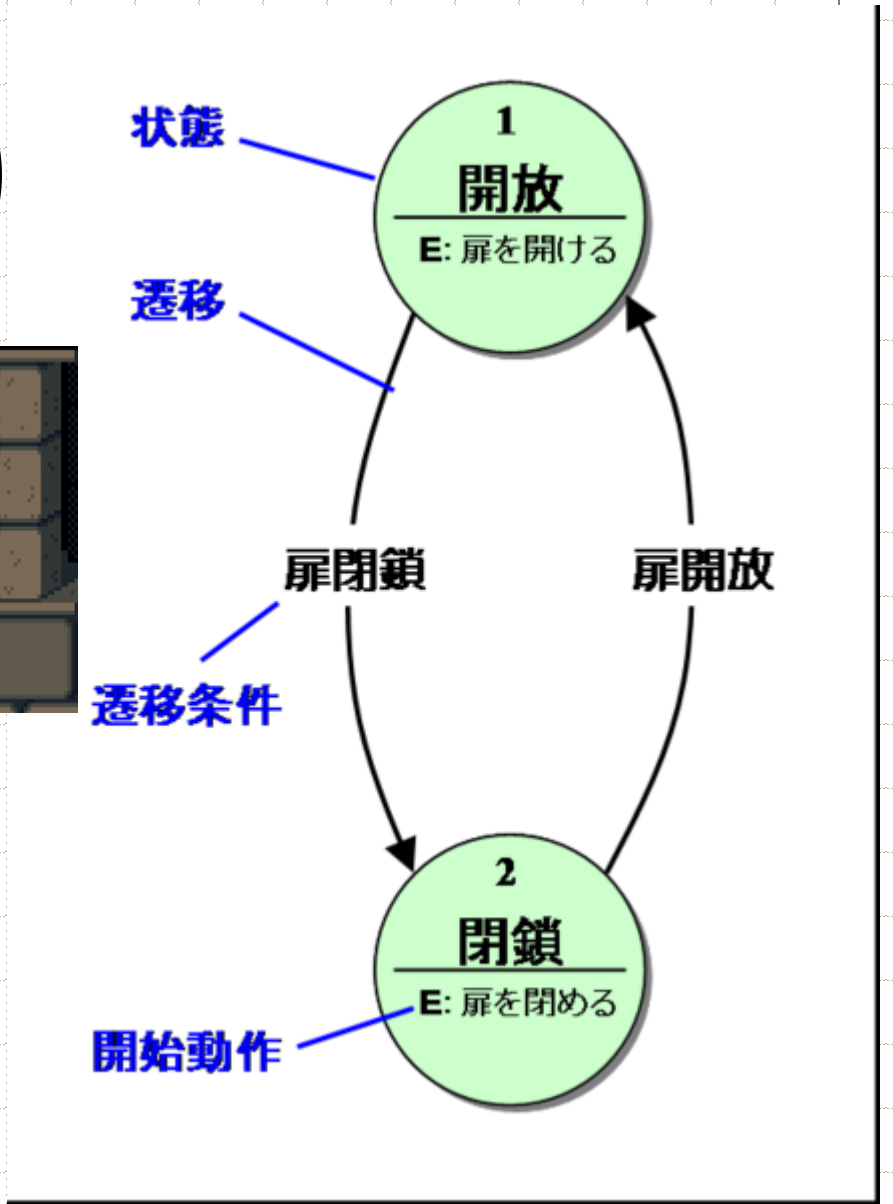


PC エージェント

FSM（有限状態マシン）: 2つの状態



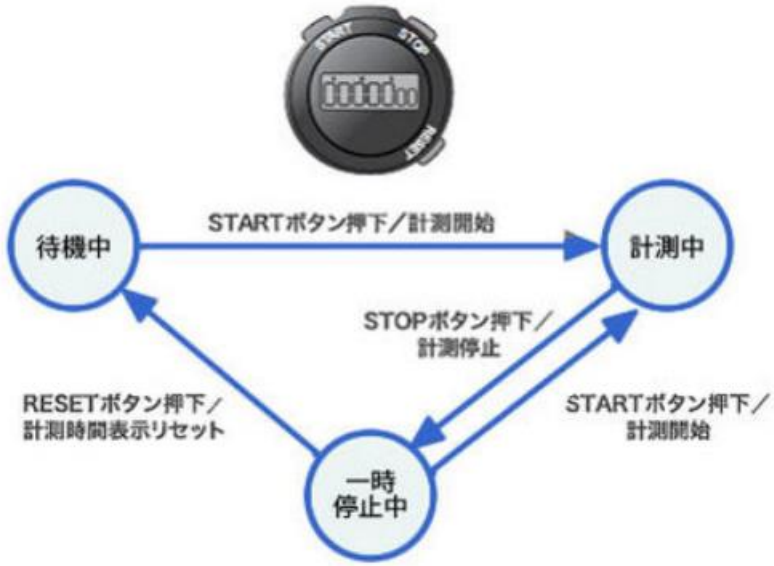




文章による仕様（ストップウォッチ）

- ストップウォッチには「スタート」「ストップ」「リセット」の3つのボタンがある
- 「スタート」ボタンを押下すると計測を開始する
- 「ストップ」ボタンを押下すると計測を一時停止し，計測結果時間を表示する
- 結果表示中に「スタート」ボタンを押下すると計測を再開する
- 結果表示中に「リセット」ボタンを押下すると計測結果時間をリセットする

ストップウォッチの状態遷移図

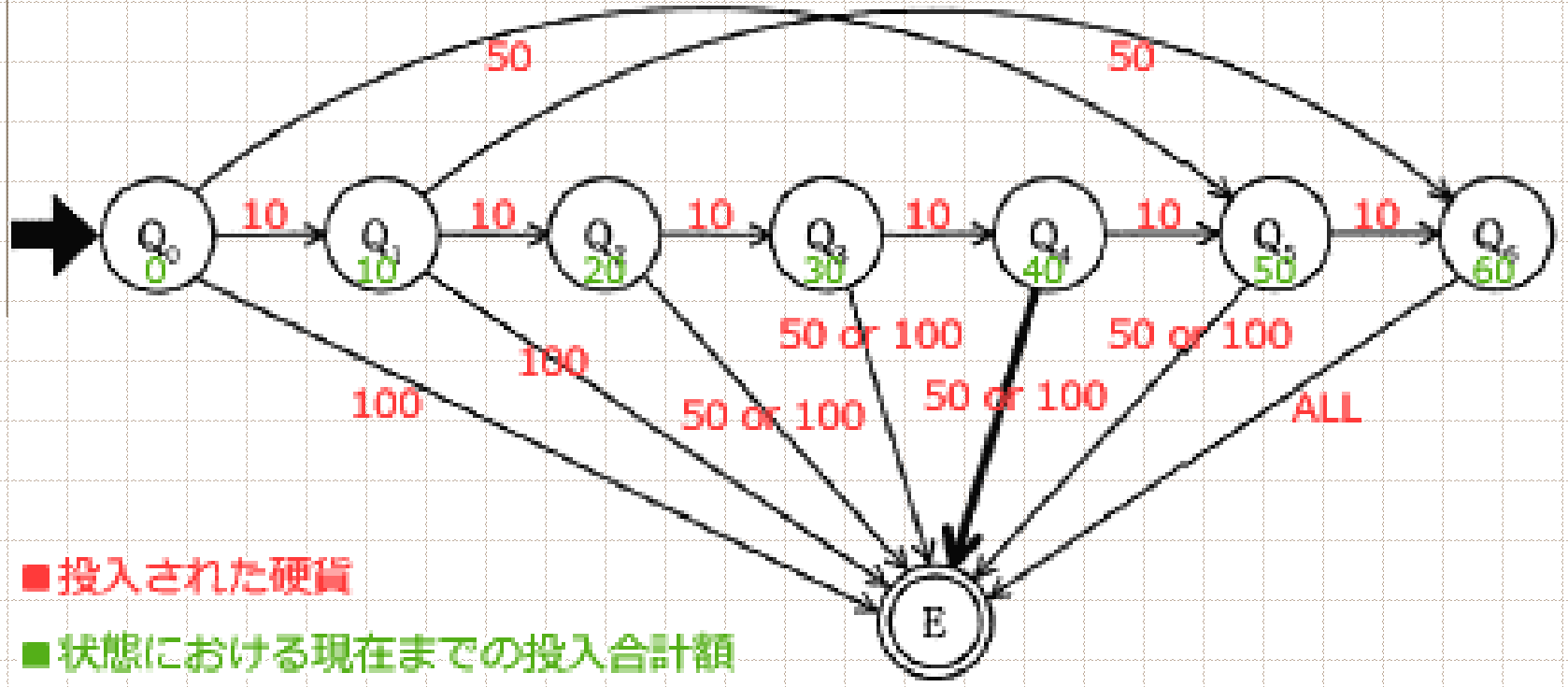


イベント

遷移前の状態	STARTボタン押下	STOPボタン押下	RESETボタン押下
待機中	計測中	—	—
計測中	—	一時停止中	—
一時停止中	計測中	—	待機中

遷移後の状態

FSM（有限状態マシン）:自動販売機のコイン処理の状態遷移



合理性(rationality)、合理的システム

- 与えられた知識のもとでシステムが正しいことをするということである。
- 合理的に考えるシステム
  - 「計算モデルを用いた心の機能の研究」
  - 「認識、推論、行為を可能にする計算の研究」
- 合理的に振る舞うシステム
  - 「知的を計算プロセスとして説明・模擬することを目的とする研究分野」
  - 「人工物の知的行動に関する研究」

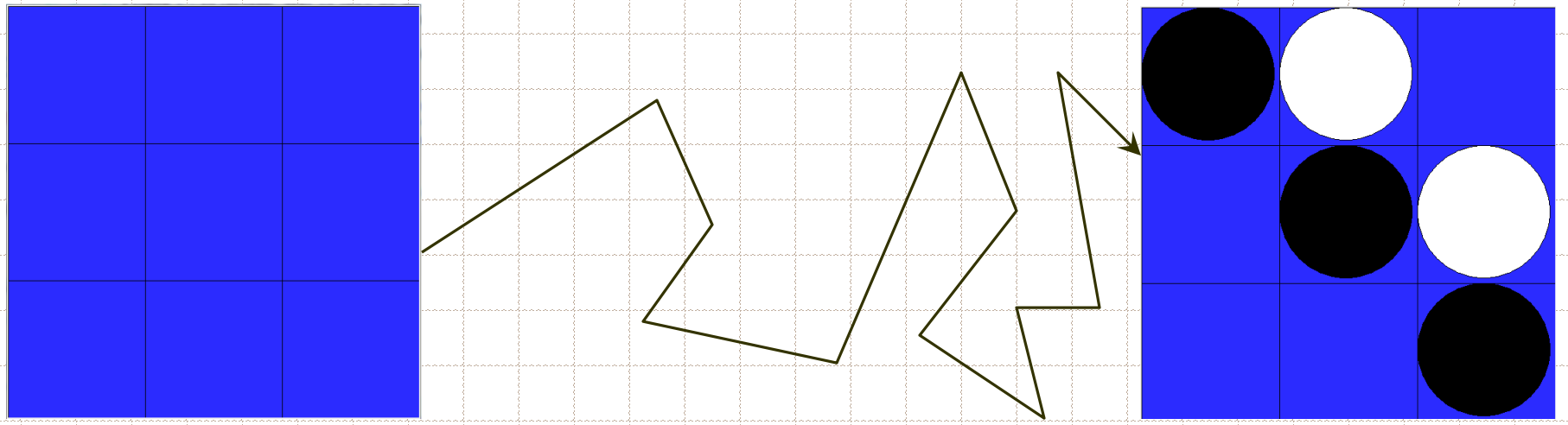
知性(intelligence)、知的システム

- 与えられた知識のもとでシステムが正しいことをするということである。
- 性能指標を最適化されたシステム。
- 合理的システムは全知全能システムではない
- 合理的システムは先見的システムではない
- 合理的システムはいつも成功するシステムではない

⇒知性は情報収集、探査、学習、自律性の性質を持つ。

三目並べ(さんもくならべ)(Tic Tac Toe)

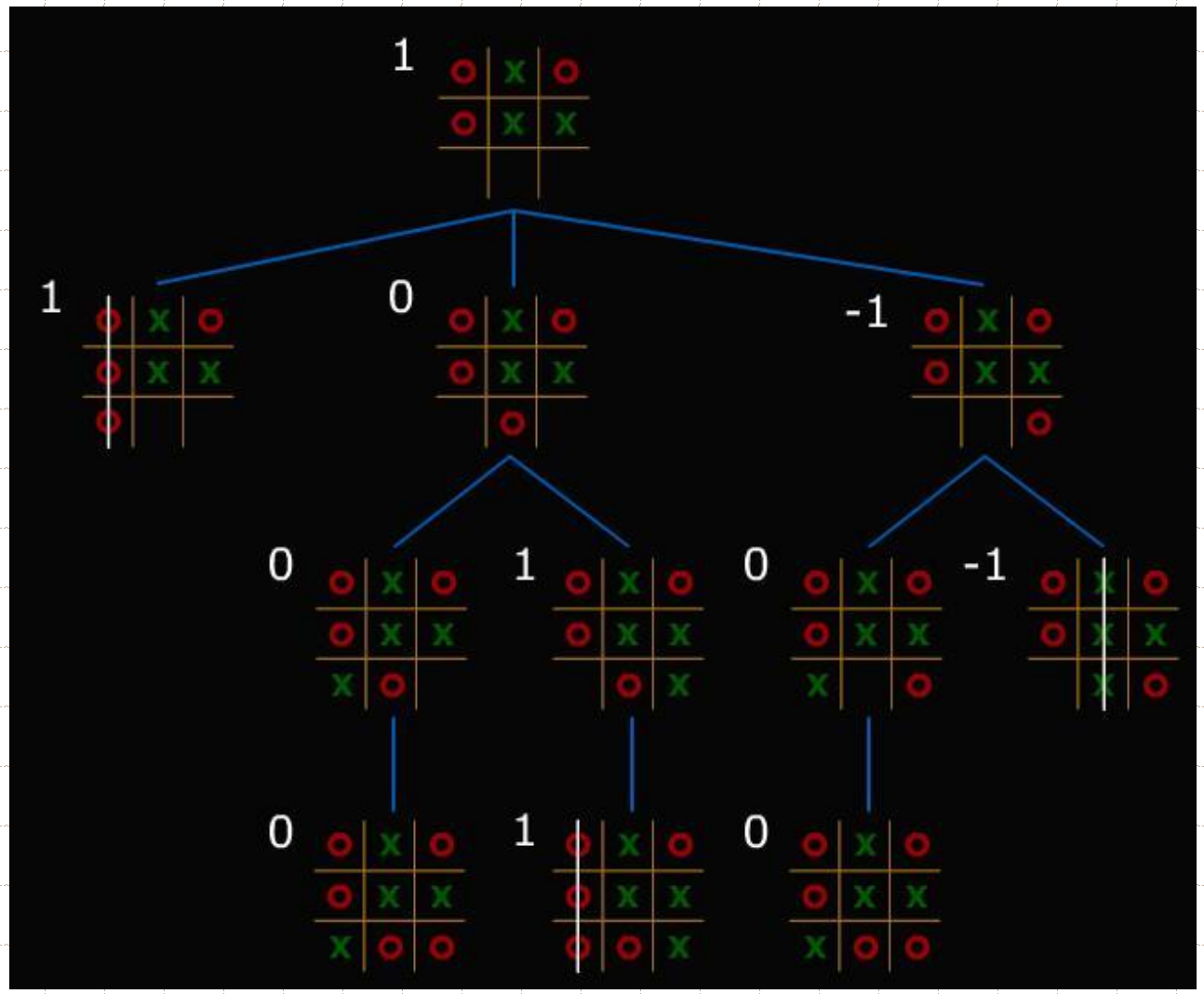
- 三目並べ(さんもくならべ)、まるばつとは、3×3 の格子を用意し、二人が交互に「○」と「×」を書き込んでいき3つ並べるゲームである。まるかけ、まるぺけ、○●ゲームとも呼ばれる。



スタート状態

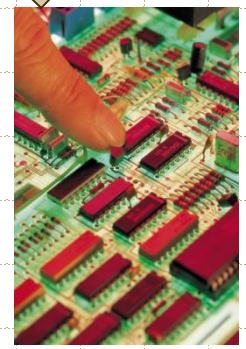
終了状態、●勝

三目並べ(Tic Tac Toe)の状態空間



知的なエージェント：Deep Blue (チェスプレーヤーの世界チャンピョン)

ゲーム：



チェスプレーヤーチャンピョン　　ディップブルー（機械）
カスパロフ氏

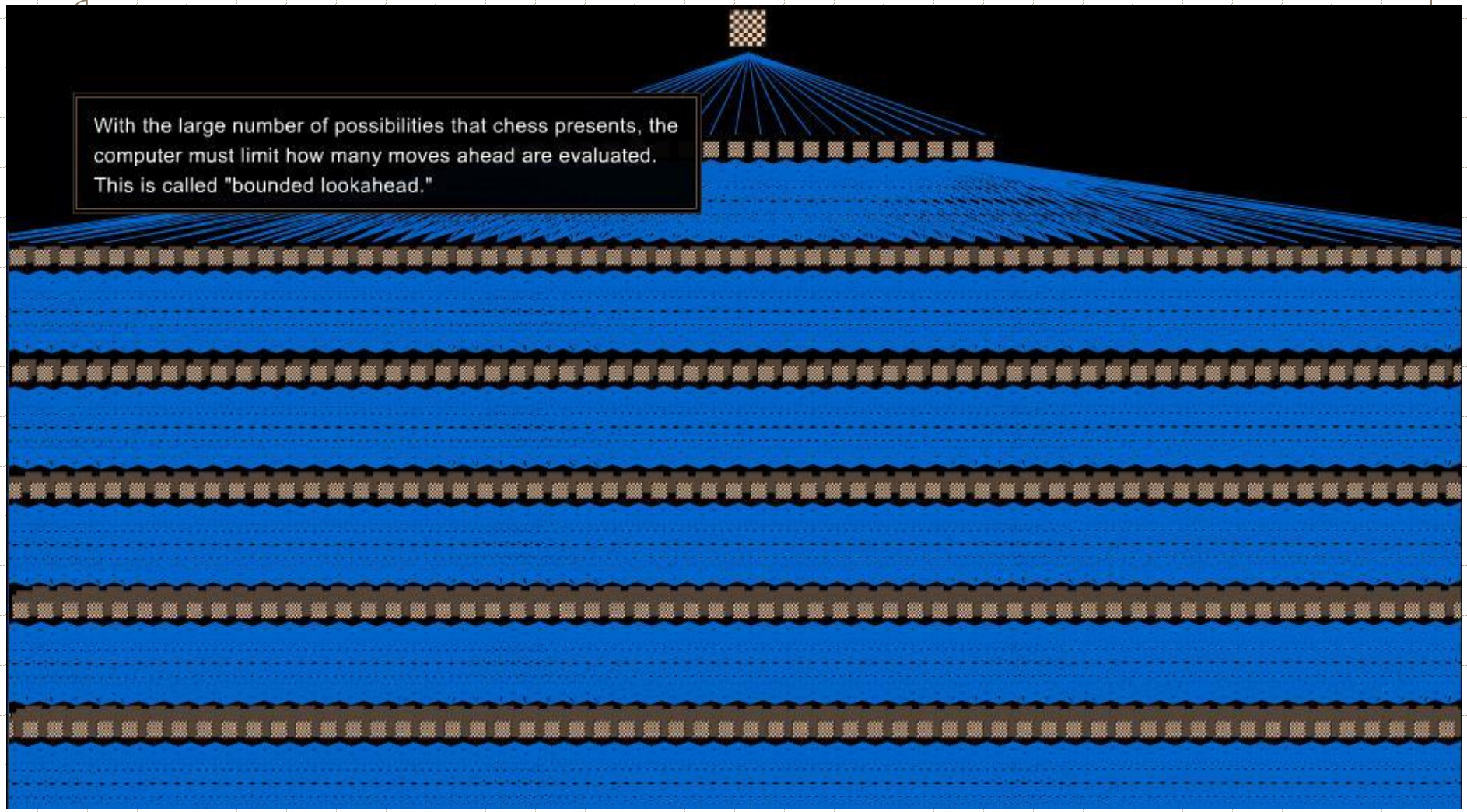
1997年、賞金：100万ドル/1億2千万円
1対2：人間は破れた

チェスの初期状態

In chess there are many more possibilities than in tic-tac-toe. At the start of the game, White has 20 possible moves as does Black. Between them, in their combined first moves there are 400 different possibilities.

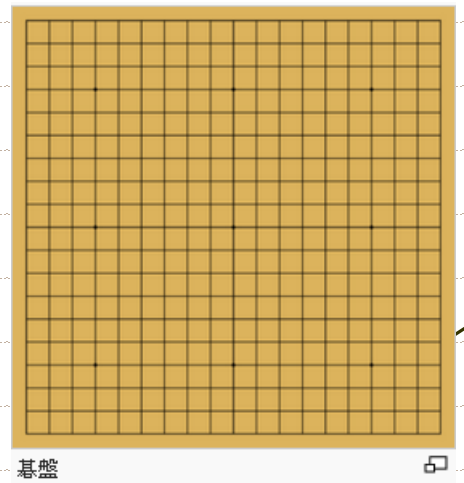
This number grows quickly…
In the first three moves it is 8,902. By four moves it is almost 200,000 and by five it is over 4.8 million.

チェスゲームの状態

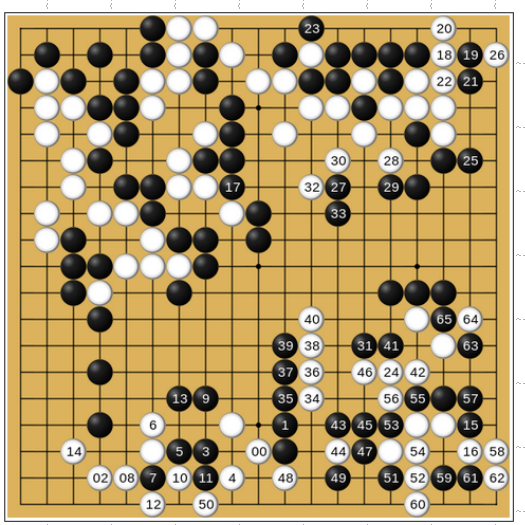

With the large number of possibilities that chess presents, the computer must limit how many moves ahead are evaluated. This is called "bounded lookahead."

囲碁

- 囲碁（いご）とは、2人で行うボードゲームの一種。交互に盤上に石を置いていき、自分の石で囲んだ領域の広さを争う。単に碁（ご）とも呼ばれる。

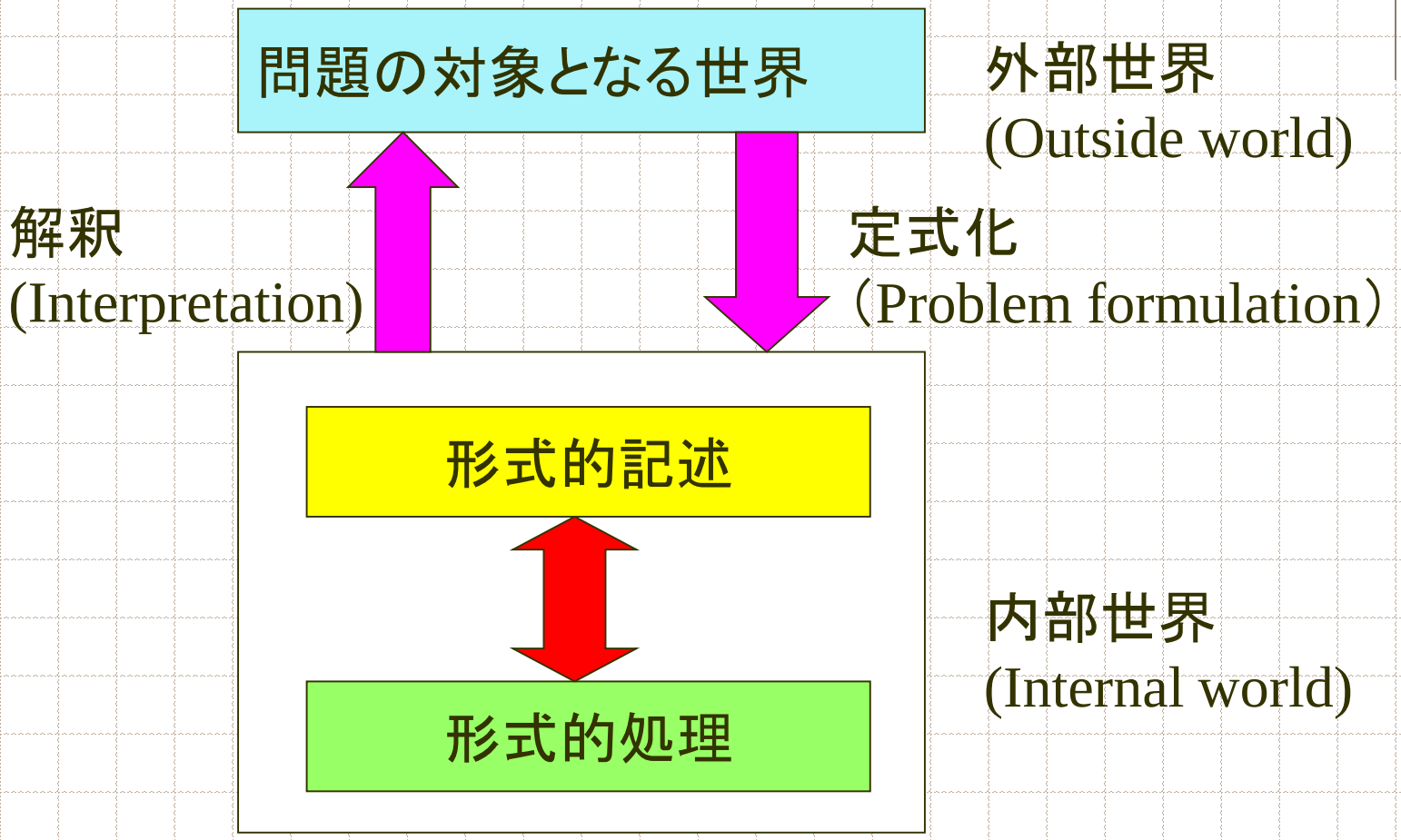


スタート状態

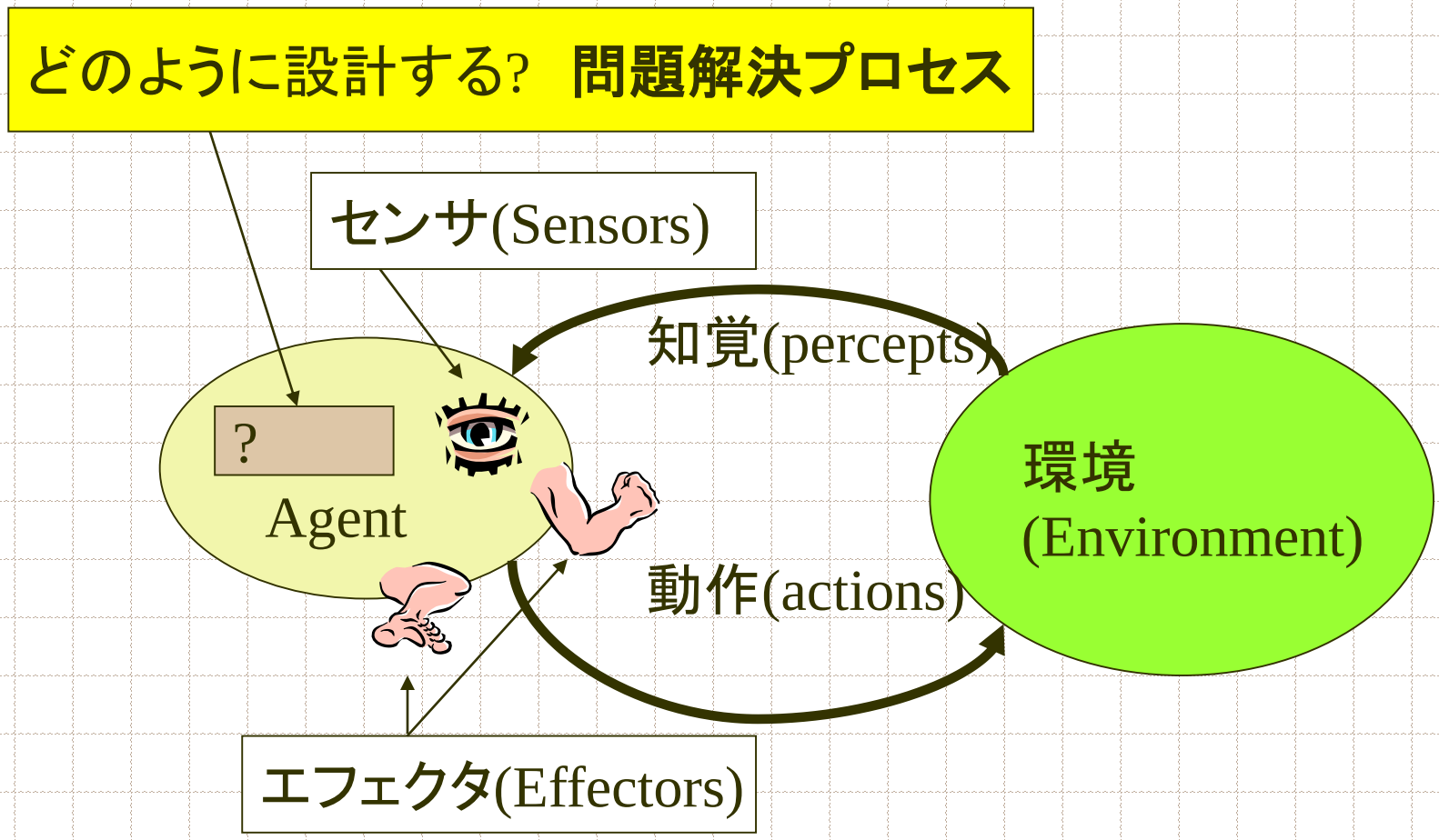


アルファゴ、終了状態、●勝

問題解決プロセス（The Problem Solving Process）



エージェント（agent）



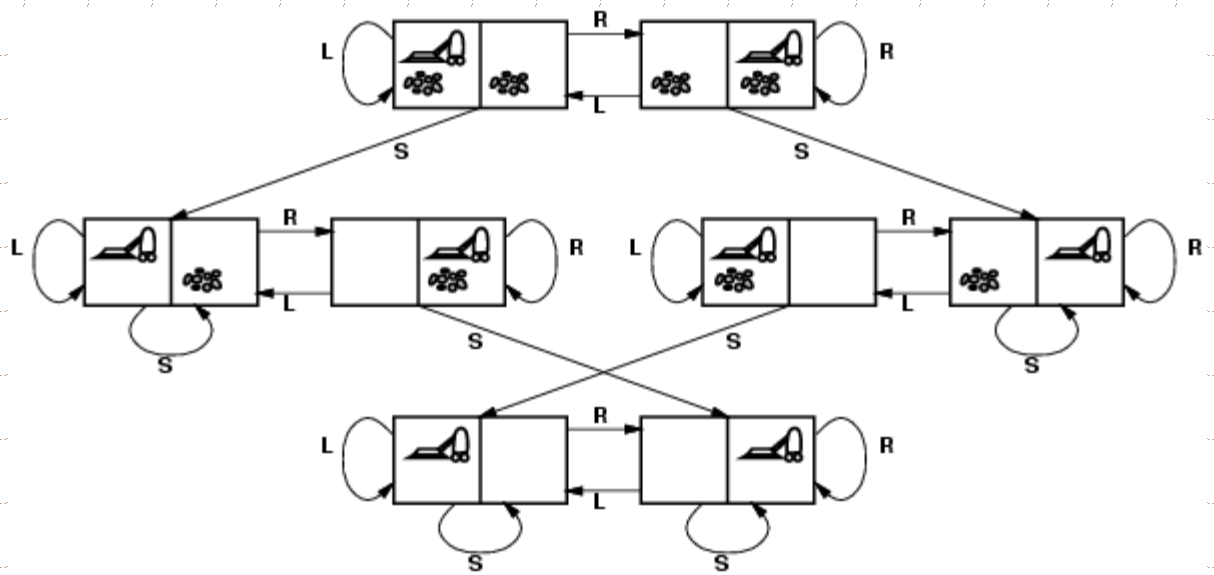
問題を定式化すること

- 問題の構成要素
  - **初期状態**：エージェントが認識している最初の状態
  - **オペレータ**：エージェントが利用できる可能な行為
  - **ゴール検査**：エージェントが到達した状態がゴール状態であるかを決定する検査.
  - **経路コスト**：経路をたどるのに必要なコスト．経路コストは，経路に沿った各々の行為のコストの総和.
- 問題の環境
  - 状態空間：初期状態から行為の任意の系列によって到達可能なすべての状態の集合
  - 経路：1つの状態を他の状態に導く行為列

問題解決エージェントの例2 (掃除機の世界：vacuum world)

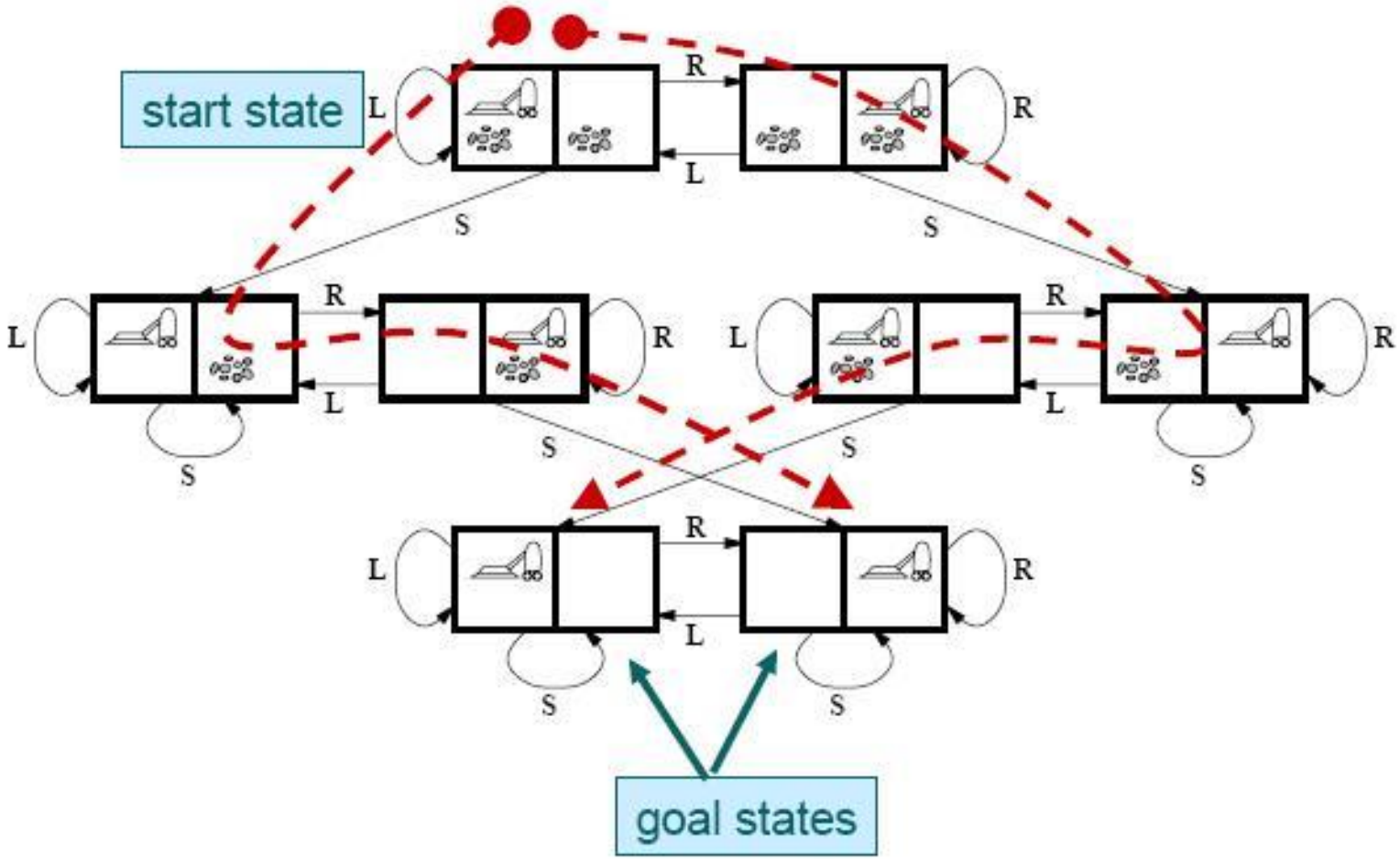
- 掃除機（エージェント）の世界は2つの場所だけを含むようにしよう．エージェントは、1つの場所か他の場所にいる可能性がある。8つの可能な正解状態がある。
  - **ゴールの定式化**：
    - 「**すべての埃をきれいに掃除することである**」を，ゴールとして設定すべきである．
  - **問題の定式化**：
    - 「**左へ移動（Left)**」あるいは「**右へ移動（Right)**」、「**吸込み（Suck)**」は，行為としてある．
  - **解の探索**：
    - 場所間の隣接情報から，可能な場所に行くし，**掃除する**．
  - **初期状態**：
    - どんな状態でも**OK**

掃除機の世界：状態空間のグラフ (Vacuum world state space)

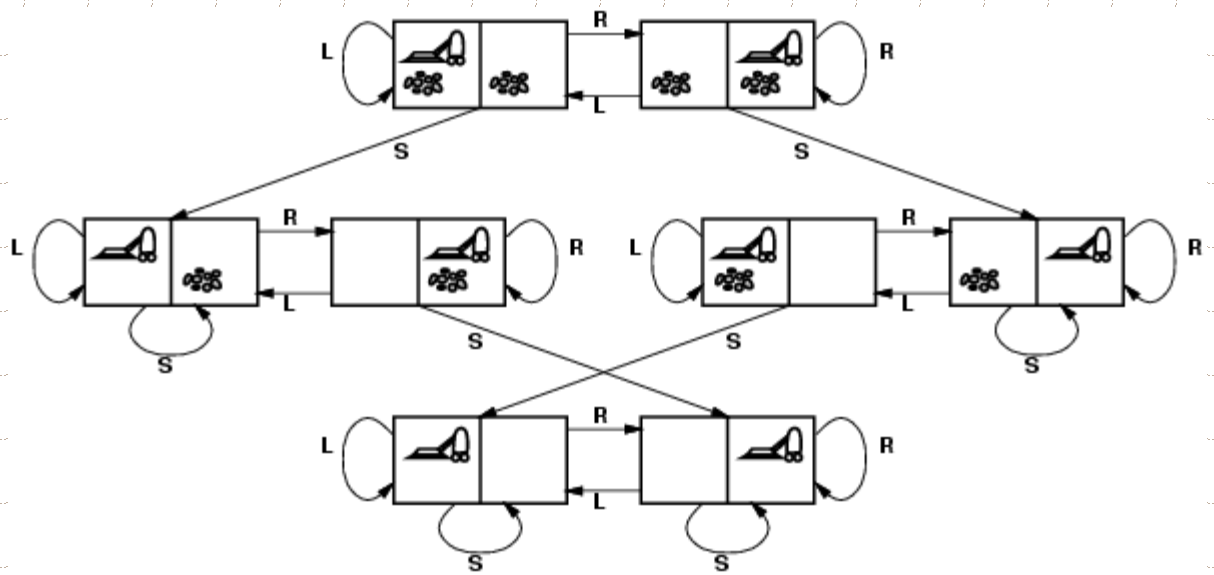


- 状態(states)?
- 行為(actions)?
- ゴール検査(goal test)?
- 経路コスト(path cost)?

掃除機の世界:状態空間のグラフ(Vacuum world state space)



掃除機の世界：状態空間のグラフ (Vacuum world state space)

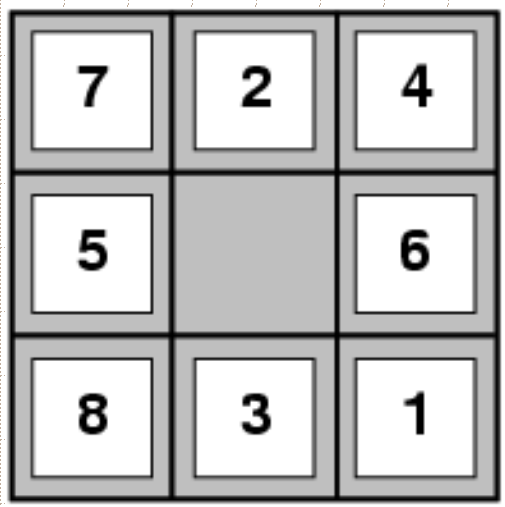


- 状態(states)? 状態1-8の部分集合(integer dirt and robot location)
- 行為(actions)? L:左に動く、R:右に動く、S:吸い込む(*Left*, *Right*, *Suck)*
- ゴール検査(goal test)?埃がない(no dirt at all locations)
- 経路コスト(path cost)? 1各々の行為は、コスト1を要する(1 per action)

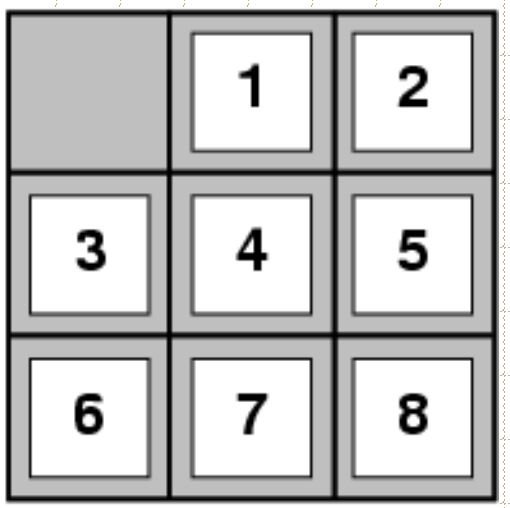
例題1.　8パズル

- 問題：滑りブロックパズルの族に属する。

8つのタイルの各々が9個の区画のどの位置にあるかによって決まる.

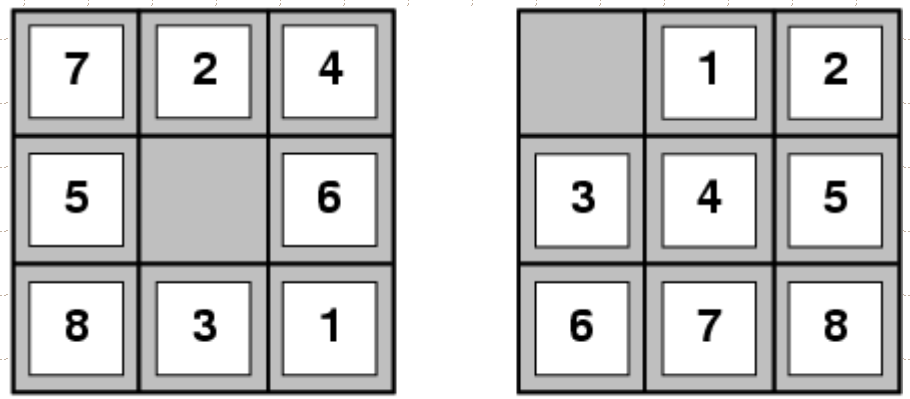


Start State



Goal State

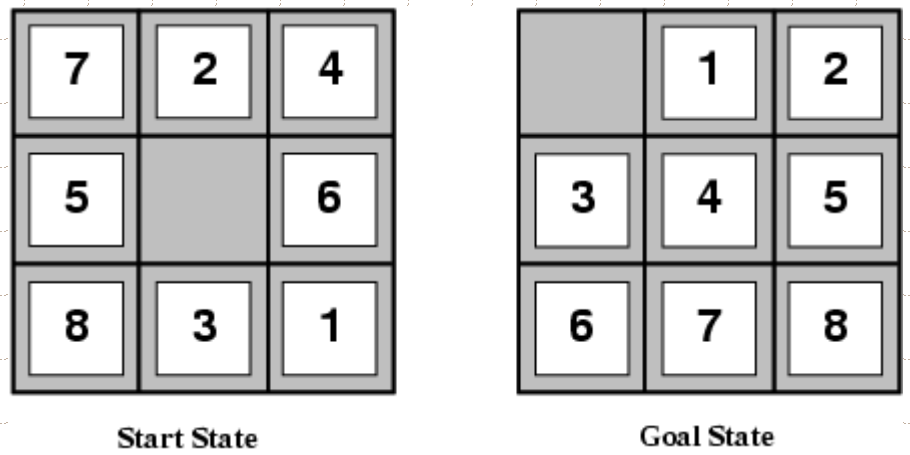
例題1.　8パズル



Start State　　Goal State

- 状態(states)?
- 行為(actions)?
- ゴール検査(goal test)?
- 経路コスト(path cost)?

例題1.　8パズル



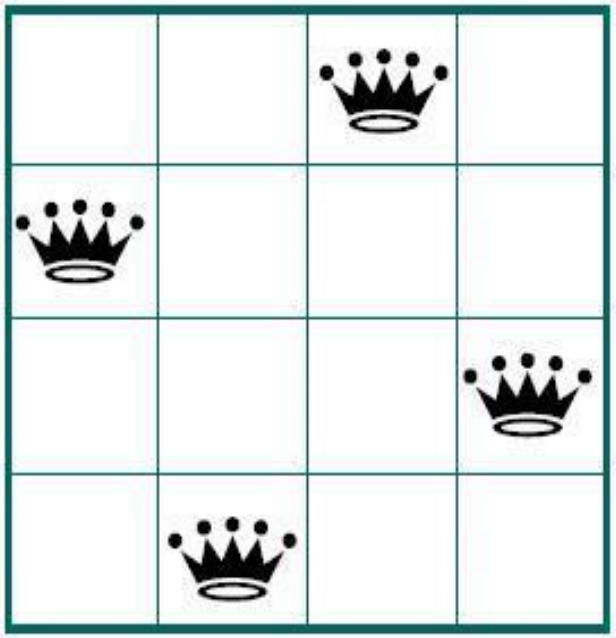
Start State　　Goal State

- 状態(states)? タイルの位置(locations of tiles)
- 行為(actions)?空所を上下左右に動かす(move blank left, right, up, down)
- ゴール検査(goal test)?状態上図のゴール配置に一致する
- 経路コスト(path cost)? 各々の行為は、コストは1を要する(1 per move)

[備考: 最適の解のクラスはNP-完全であること(optimal solution of *n*-Puzzle family is NP-hard)]

Examples: toy problems (cont'd)

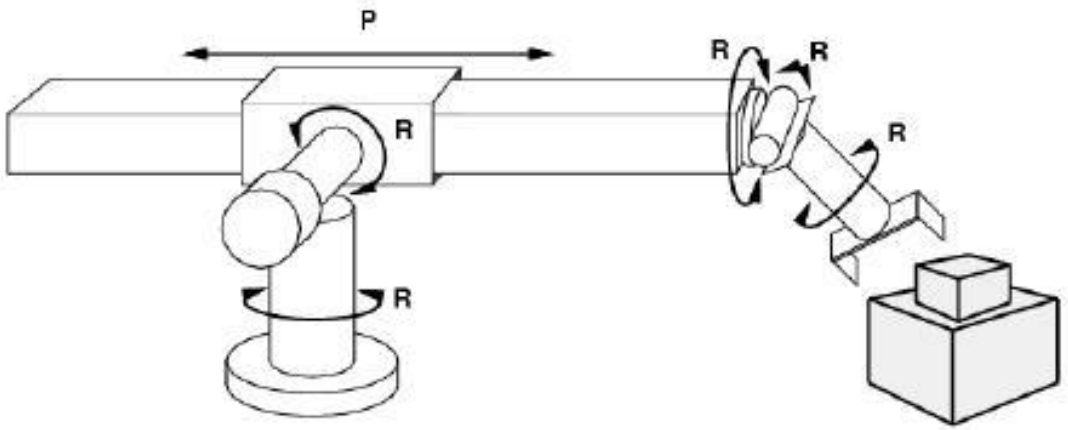
- 8-queens or N-queens
  - Arrange 8 queens on an 8x8 board so that none attack each other
- At least two possible formulations
- Incremental:
  - start state? empty board
  - actions? add a queen to an empty square so that there's no attack
  - Backtrack if there is
- Iterative:
  - start state? Add all queens at once
  - actions? Move attacking ones to minimize attacks
-



4-queens

path cost irrelevant!
(just solve the problem)

Real-world example



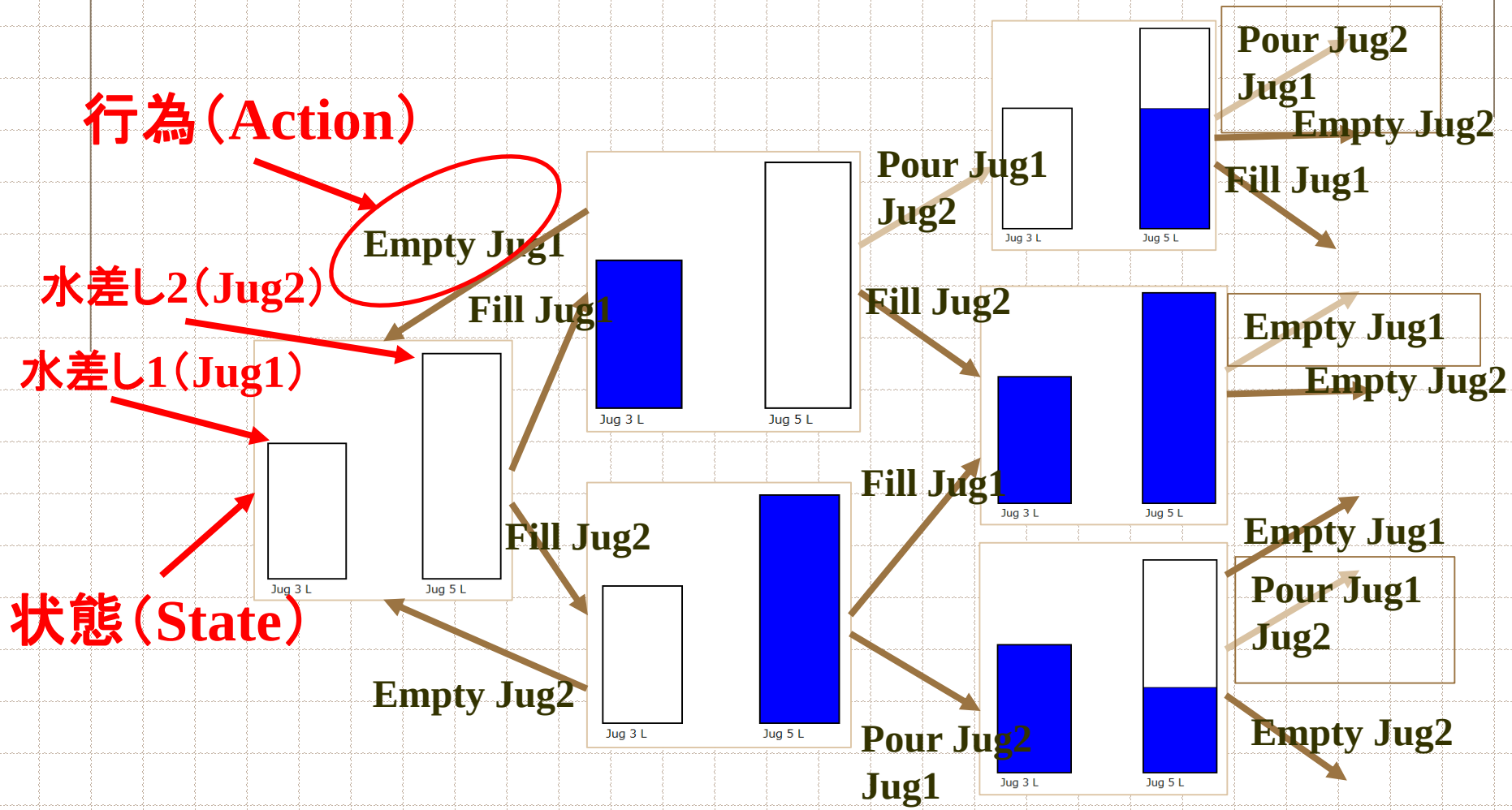
- states? real-valued coordinates of robot joint angles parts of the object to be assembled
- actions? continuous motions of robot joints
- goal test? complete assembly
- path cost? time to execute

水差し問題(Water Jug Problem)

<具体例1>　水差し問題

4ガロンと3ガロンの二つの水差しと，水を入れるためのポンプが与えられています．4ガロンの水差しに正確に2ガロンの水を入れるにはどうしたらよいでしょうか．もちろん，水差しに目盛は付いていません．

水差し問題(Water Jug Problem)の状態空間：



状態遷移関数(状態)→オペレータ

水差し1＝4ガロン、水差し2＝3ガロンの例では、以下のルールである。

R1：4ガロンの水差しが満杯でないなら，それを満杯にする．

R2：3ガロンの水差しが満杯でないなら，それを満杯にする．

R3：4ガロンの水差しに少しでも水が入っているならそれを空にする．

R4：3ガロンの水差しに少しでも水が入っているならそれを空にする．

R5：3ガロンの水差しの水を移して4ガロンの水差しを満杯にする．

R6：4ガロンの水差しの水を移して3ガロンの水差しを満杯にする．

R7：3ガロンの水差しの水を全て4ガロンの水差しに移す．

R8：4ガロンの水差しの水を全て3ガロンの水差しに移す．

オペレータ（行為）

- Fill Jug 1: 水差し1を満杯する
- Fill Jug 2: 水差し2を満杯する
- Empty Jug 1: 水差し1を空にする
- Empty Jug 2: 水差し2を空にする
- Pour Jug1 to Jug2: 水差し1の水を水差し2に満杯にする
- Pour Jug2 to Jug1: 水差し2の水を水差し1に満杯にする

まとめ

- FSM
- 状態空間
- 状態空間法は問題定式化の一つの手法.
- 状態空間法:
  - **初期状態**
  - **オペレータ**
  - **目標状態**
  - **状態遷移関数**
- 例題:掃除機、パズル、など