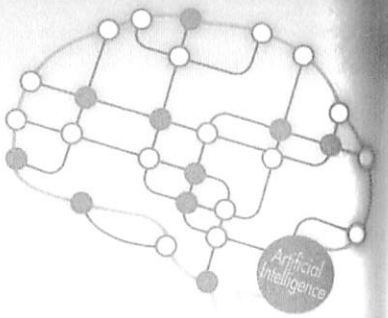


第5章 プランニング

人工知能あるいはロボティクスにおける**プランニング**（planning）とは，「エージェントに与えられた目標を達成するために必要な**行為**（action）の系列を自動生成すること」である．よって，プランニングでは，単なる推論あるいは探索ではなく，推論結果として得られる行為の系列である**プラン**が実世界で実行されることを考慮しなければならない．

本節では，まずSTRIPSに代表される基本的なプランニングについて述べ，次に世界の状態からなる探索空間ではなくプランの探索空間を用いた半順序プランニングについて触れる．そして，実世界でのプランの達成を目指した即応プランニングについて述べる．

5.1 STRIPSプランニング

人工知能におけるプランニングでは，エージェントが行為を実行すべき対象である**環境**（environment）が，計算機上の記号表現である**環境モデル**（environment model）を用いて記述される．実際には，環境を観測したエージェントが，環境の情報を環境モデルで記述すると考える（図5.1）．環境モデルで記述された環境の状態を，本章では，単に**状態**（state）と呼ぶ．なお，環境モデルとしては，一階述語論理が用いられる場合がほとんどである．本節で説明するプランニングの枠組は，スタンフォード大学で1970年代に開発されたプラナーであるSTRIPS[5]が基礎になっているので，本書ではそれを**STRIPSプランニング**と呼ぶことにする．

以下に，STRIPSプランニングの入出力，手続きをまとめて示す．

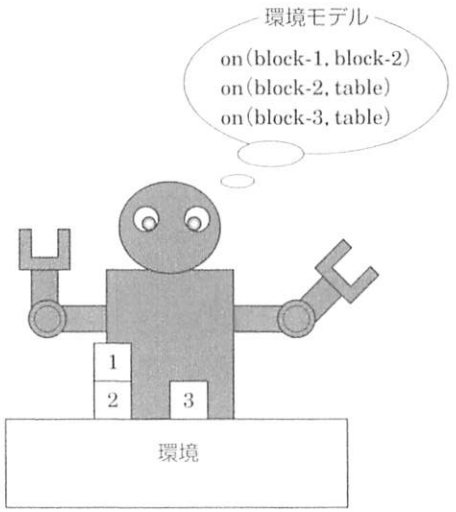


図5.1 環境モデル

□ **入力**
- **オペレータ（operator）**：環境モデルを変換する規則．環境への行為を記述したものである．
- **初期状態（initial state）**：現在の状態の環境モデル．
- **目標状態（goal state）**：目標である状態の環境モデル．

□ **出力**
- **プラン（plan）**：初期状態を目標状態に変換できるようなオペレータの系列．

□ **手続き**
- 与えられた初期状態を目標状態に変換できるようなプランを探索する．

上の枠組において，プランが得られると，後はそれに従って環境に対して行為を実行していけば，環境においても目標を実現できることになる．また，プラニングにおいて，目標は環境モデルの状態として記述される．

プランの自動生成は，状態をノードとして，状態間の遷移を表すアークを適用

されたオペレータとした探索空間における探索とみなせる．その探索空間を図示したものが図5.2である．この探索空間は，状態をノードとすることから状態探索空間と呼ばれる．

これに対し，プランを変形していくことにより，半順序プランを求めるプランニングもある．その探索空間はノードがプランであり，アークがプランを修正するオペレータであるプラン探索空間である．この種のプランニングについては，後述する5.2節「半順序プランニング」で詳しく述べる．

具体的なプランニングの例として，スタンフォード大学で1970年代に開発されたSTRIPS[5]を紹介する．STRIPSは，部屋の間の移動などを目的とする移動ロボットの行動のプランニングを行うために設計されたプログラムであり，その後今日に至るまでAIのプランニングの基本的枠組として広く研究されている．STRIPSの扱う環境とその環境モデルの例を図5.3に示す．プランニン

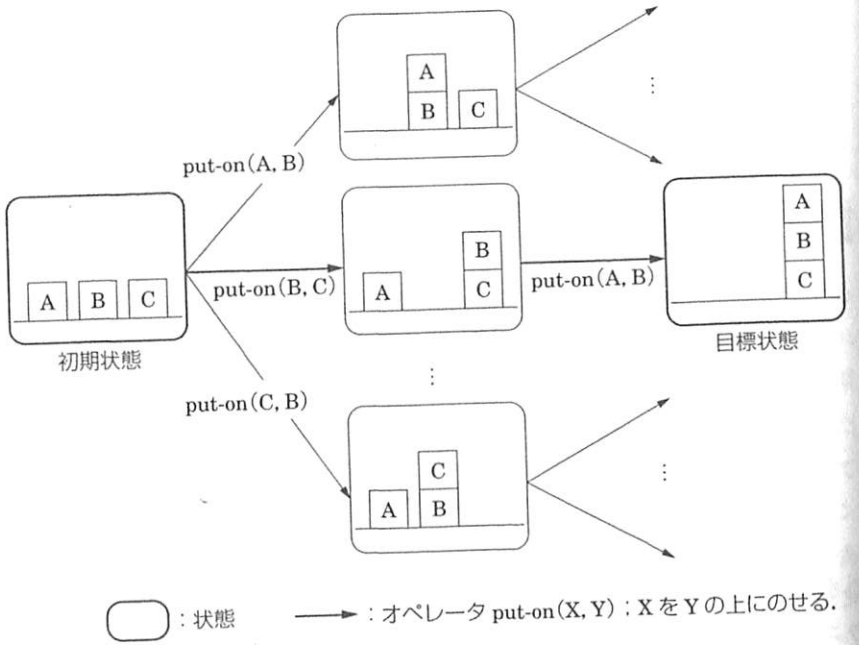


図5.2　プランニングの状態探索空間

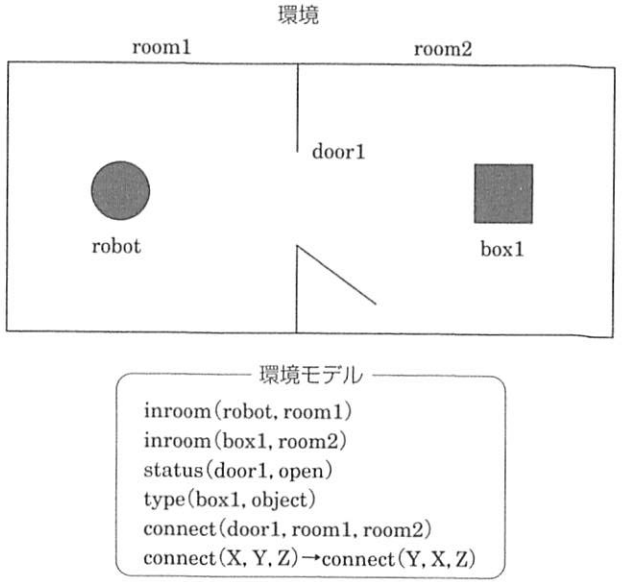


図5.3　STRIPSの環境と環境モデル

グは，あくまでも計算機上での記号操作なので，このような実環境を計算機の中に環境モデルとして取り込まなければならないが，STRIPSでは述語論理を用いて環境モデルを記述する．

ここで，環境モデルのそれぞれの述語の意味を以下に示す．なお，大文字の引数は変数を，小文字の引数は値を表す．

inroom(A, R)：　物体Aが，部屋Rに存在する．

type(A, B)：　Aのタイプが，Bである．

connect(R1, R2, D)：　2つの部屋R1とR2がドアDを通してつながっている．また，図5.3中のconnect(X, Y, Z) → connect(Y, X, Z)は，第1引数と第2引数が入れ換え可能という意味．

status(D, S)：　ドアDの状態が，Sである．Sの値としては，open，closedなどをとる．

nextto(A, B)：　物体Aが物体Bの近くにある．

ところで，図5.3の見取図が表現している実際の部屋には，どれほどの情報があるのかを考えて欲しい．本来，図のような単純な部屋においても，部屋に存在する物体であるロボット，箱，壁そしてドアのそれぞれについて，その形状，重量，色，大きさ，位置，材質などの膨大な数の情報がある．実世界で行動するエージェントを考えると，エージェントはセンサを通して入力されるそのような膨大な情報から自分に必要なものを取捨選択しなければならない．このような問題は，**フレーム問題**（frame problem）[7)11)]と呼ばれ，一般に解くことは困難である．また，フレーム問題は，実世界とインタラクションを持ち，かつ環境をモデリングする必要のあるシステムには避けれない問題である．

実は，人間でもこのフレーム問題に悩まされることがある．外界からの情報のうち，どの部分に注意すれば正しい判断を下せるのかは，環境に対する知識があるからだが，我々がそのような知識を持っていない環境におかれたときにフレーム問題が生じる．たとえば，人が犬に接する場面を考えよう．このとき，犬のご機嫌の悪いときは，気を付けないと噛まれしまう．よって，犬という外界から得られる情報から，犬のご機嫌を推論したいわけであるが，尻尾を振っているときはご機嫌が良いというような犬についての知識がない人は，犬のどこを観ればご機嫌が推論できるかがわからない．これは，一種のフレーム問題に陥っていると考えられる．

プランニングの場合，図5.3の環境モデルから明らかなように，移動ロボットの目的にとって無関係と思われる膨大な情報は，最初から設計者により捨象されている．このように現在のところ，人工知能に限らず，すべての人工的システムの設計において，そのシステムにとって必要十分であろう情報の枠を人間である設計者が多大な労力をもって考え，システムに与えているのである．

次にSTRIPSで用いられるオペレータの一つである**gotod(DX)**を図5.4に示す．このオペレータは，「部屋RXにおいてドアDXに近付く」という行為を記述している．図中では，述語の右にその意味を書いてある．一つのオペレータは，条件リスト，削除リスト，そして追加リストの3つのリストからなり，それぞれのリストの要素は，環境モデルを記述している述語である．

プラナーは，オペレータを環境モデルに適用することにより，環境モデルの

オペレータ gotod(DX)

□ 条件リスト
- inroom(robot, RX)：ロボットが，部屋RXにいる．
- connect(DX, RX, RY)：ドアDXにより部屋RXと部屋RYがつながっている．

□ 削除リスト
- nextto(robot, A)：ロボットが，物体Aの近くにいる．

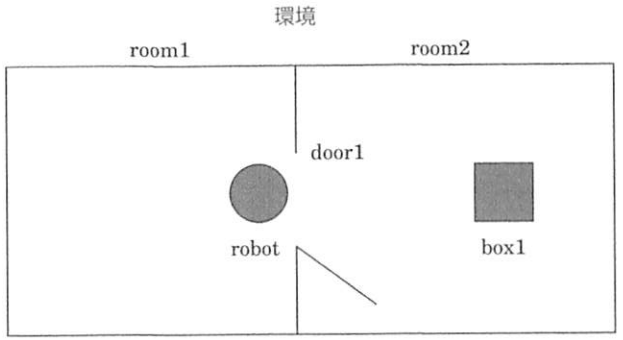
□ 追加リスト
- nextto(robot, DX)：ロボットが，ドアDXの近くにいる．

図5.4　STRIPSのオペレータ[5)]

状態を更新していく．STRIPSでのオペレータの適用方法は，以下のとおりである．これは，4.5.1節で説明したプロダクションシステムのルールの適用に似ている．

□ 条件リスト中のすべての述語が，環境モデルで成り立つか否かを調べる．
- もし成り立つなら，環境モデル中の述語のうち，削除リスト中の述語とマッチングするものを環境モデルから取り除き，追加リスト中の述語をすべて環境モデルに追加する．なお，このとき別のリスト中にあっても，同じ名前の変数は同じ値が代入される．
- もし成り立たないなら，何もしない．

たとえば，図5.4のオペレータgotodを図5.3に適用してみると，gotobの条件リスト中の述語は，すべて図5.3の環境モデルで成り立っているので，gotobは適用可能である．このとき，変数RX，RY，DXには，それぞれroom1，room2，door1が代入されている．次に，環境モデル中の述語で削除リスト中の述語nextto(robot, A)とマッチする述語を取り除こうとするが，図5.3の環境モデルではマッチするものがないので，何も削除されない．そして，追加リスト中のnextto(robot, door1)が環境モデルに追加される．その結果が図5.5であり，環境モデルとそれに対応する環境が変化していることがわかる．この場合，環



環境モデル

nextto(robot, door1)
inroom(robot, room1)
inroom(box1, room2)
status(door1, open)
type(box1, object)
connect(door1, room1, room2)
connect(R1, R2, D)→connect(R2, R1, D)

図 5.5　gotod が適用された結果

境モデルは，図 5.3 に nextto(robot, door1) が追加されただけである．

注意して欲しいのは，プランニングの段階では，実環境に行為が行われないので，ここで変化するのは環境モデルであり実環境ではないことである．よって，最終的に得られたプランがアクチュエータにより実行されて，始めて実環境が変化することになる．なお，条件リスト中の述語が，環境モデルで成り立っているか否かのチェックは，定理証明のアルゴリズムである**導出法**（resolution）[9] を用いて，計算機により自動的に行うことが可能である．この導出法による自動定理証明が実現したことが，環境を述語で記述する STRIPS の研究の動機付けになっている．

STRIPS の基本になっている考え方は，**GPS**（General Problem Solver）[8][9] という一般問題解決システムで用いられた**手段目標解析**（means-ends analysis）である．既に，2.3.3 節でも触れたように，手段目標解析では，まず最初に初期

状態と目標状態の差異（difference）を取り出す．そして，その差異を減少させるようなオペレータを選択する．そして，今度はそのオペレータを適用すること，つまり，そのオペレータの適用条件†を満たすことを次の**副目標**（sub-goal）にして，また差異の検出とオペレータ選択を繰り返していく．その結果，差異がなくなったときに，初期状態から目標状態に至るようなオペレータの系列，つまりプランが生成されるわけである．これは，探索方式として**後向き推論**（2.3.1 節を参照）をしていることになる††．ただし，その STRIPS の手続きは，完全性を持たないことが知られている．プラニングにおける**完全性**（completeness）とは，プランが探索空間上にあれば，必ず見つけるという性質を意味する．また，**健全性**（soundness）とは，プラナーによって生成されたプランを実行することにより，必ず目標状態が満たされる性質を意味する．

例として，初期状態が前述の図 5.3 で，目標状態が図 5.6 であるような問題を考えよう．図 5.6 から明らかなように，目標状態もまた，初期状態と同じく環境モデルで与えられる．このとき，初期状態と目標状態の差異は，初期状態で成り立っていない目標状態の述語であるから，nextto(robot, box1) である．STRIPS は，この差異を減少させるオペレータ，つまりその追加リスト中に nextto(robot, box1) を含むオペレータを探す．そして，今度は，そのオペレータの条件リストを副目標として，初期状態との差異の検出，さらにその差異を解消するオペレータの選択を繰り返していき，差異がなくなったときにプランが完成する．オペレータの詳細な記述は割愛するが，たとえば，この問題については，図 5.7 のようなプランが得られる．

プランニングは，環境の観測を頻繁に行わない．よって，プランニングは，一度観測した状態を初期状態として，それに既与のオペレータをいかに適用していけば目標状態に至ることができるかというエージェント内部の問題空間における探索問題に帰着される．しかし，プランの探索は，非常に多くの計算量を

† STRIPS では，条件リストに対応する．

†† 厳密には，STRIPS では，初期状態に適応可能なオペレータがプラン中に現れたときは，適用して状態を更新するという手続きが含まれる．この手続きにより，前向き推論の要素も含むことになる．