Jason とプランナの結合によるロボット制御の改善

奈良女子大学　理学部情報科学科 4 年
新出研究室　小島侑子

概要

本研究では、主にシミュレーション上で扱われていた BDI エージェントを、市場に安価に広がり始めたロボットに実際に搭載し、その挙動及び現実とシミュレーションでの問題点を検証する実験を行った。その際、ロボットの行動の誤差による知覚の誤りによって、ロボットの信念が誤ったものとなり、目標が達成出来なくなる問題が発生した。本論文では、エージェントにプランナを結合することによって、エージェントの目標達成の行き詰まりを解決したことを示す。

1　はじめに

BDI エージェント [10] とは、信念（Belief）、願望（Desire）、意図（Intention）という 3 つの心的状態パラメータを用いて熟考を行い、自律的かつ合理的な行動を行うエージェントである。BDI エージェントの利点として考えられる特徴は、“意図” の持続性及びその一貫性にある。意図とはエージェントの願望に対しての、ある程度大まかな願望達成への方針である。これを保持することにより、エージェントは矛盾の無い一貫した行動をとることが可能となる。

一方、近年ロボットの開発が進み、比較的安価でその性能がある程度保証されているロボットが市場に広がってきている。しかし、ロボット開発が盛んになってきたのに対して、BDI エージェントは現状ではシミュレーション環境で使用されていることが多く、実際ロボットに搭載された例は少ない。関連研究として、Lemke ら [8] や Juul ら [7] の論文があるが、動作制御の点においてあまり現実的なロボット制御とは言い難い。従ってこのようなロボットを使用し、実世界における BDI エージェントの挙動及び現実とシミュレーションでの問題点を検証する実験を我々は行った。

実験に使用したロボットとしては、LEGO 社が販売している LEGO MINDSTORMS NXT[3] というロボットを採用した。

これまでに我々が行った実験 [12] では、4 × 5 のマス目空間の迷路を実際に作成し、ロボットにその迷路内に配置されているオブジェクトを探索しながら発見することを問題として与えた。

エージェントの設計としてはマス目単位で行動を行いながらその迷路内で前方知覚を行い、迷路内の壁を認識しながら行動を進め探索を行い、オブジェクトを発見すると元の位置までバックトラックして戻ってくるように作成した。

この結果、ロボットはシミュレータ上でのエージェントの挙動を忠実に従うのに対して、現実空間でのロボットは、床との摩擦における誤差、更にその誤差に伴って壁の無い位置で壁があると誤認してしまうということが起こった。これにより、オブジェクトを発見するという目標を達成できないという実験結果となった。

そこで今回我々は、ロボットの知覚による壁の誤認識という行動によってエージェントの信念が誤ったものとなり、最終的に行動が出来なくなるという問題点に着眼し、前回の実験における実装に再プランニングのメ

カニズムを取り入れ、エージェントの信念の信憑度を下げて正しい信念を得ることが出来るように知覚し直す戦略を取り入れて再実験を行った。

これを実現するにあたって、動的な環境に対応するプランナを別途エージェントに搭載することにより、エージェントの探索戦略を変更することを行い、エージェントに再行動を行わせるという方針を立てた。

エージェントは Jason というプラットフォーム上で構築されており、今回このエージェントにプランナを組み込む方法として、Jason とは別のプロセスにプランナのプロセスを起動させ、このプロセスとエージェントのプロセスを連携させるようにすることでプランナの取り付けを実現した。

このようにプランナを実装することにより、ロボットに再行動を取らせることが可能となり、壁でない部分を壁であると誤認してしまうという問題点を解決することができるようになった。

本論文では、このエージェントとプランナ間の連結部の設計及び実装について記す。

この連結部の設計としては、プロセス間での通信を行わせることにより、それぞれのプロセスが独立して動作するように設計を行った。従ってエージェントとプランナ間にプログラム上の依存関係はなく、独立に設計することが許される。

なお、本研究は奈良女子大学大学院博士後期課程 1 年藤田と、奈良女子大学 4 年片山との共同研究である。

2 研究背景

2.1 前回行った実験

前回、我々は BDI エージェントを構築する言語 AgentSpeak[5] 及びその処理系である Jason プラットフォームを用いて、現実世界のロボットを BDI エージェントにより制御することを目標とし、主にシミュレーション上で扱われてきていた概念を実世界のロボットに搭載することで、我々の設計の有用性及び問題点を検証した [12]。

実験で使用したロボットは LEGO MINDSTORMS NXT(図 1) である。このロボットは比較的操作しやすく安価であるため、ロボットを実際に扱って実験することには適したものであると言える [3]。

図 1　LEGO MINDSTORMS NXT

Jason によって実装されたエージェントはそのプラットフォーム上で行動決定を行い、その行動制御命令

は別の PC にテキスト文で伝達される。その伝達先の PC は Python によるロボット制御を行っている。PC はそのテキスト文を読み込み、Python プログラムの中で NXT の行動制御用の命令に解釈し、Bluetooth によってその命令の伝達を行っている。

NXT はこの手順の通りに制御され行動を行い、壁の知覚も行う。壁の知覚の際には、知覚した情報を Python 側の PC に転送し、Python プログラム内でその情報から壁を知覚したかどうかの判定を行い、エージェントにその判定を返す。

以上のことを繰り返し、目標達成への行動を行う。

2.2 関連研究との比較

本研究に関連している研究としては、Developing Multi-Agent Lego Robotics [8] や Agent Programming — Robot Traffic[7] などがある。[8] では NXT を NXT-G[1, 2] というグラフィック型のプログラミングソフトで制御しており、これは Windows にのみ対応して制御するように設計されている。よってこれでは OS に依存してしまうことになり、開発環境としては制限がかかってしまう。そこで我々は Windows のみに依存しない環境で制御を行うことにした。また [8] では、経路探索に A-star 探索アルゴリズム [6] を使用し、光センサを活用して格子状に張られたテープの色の度合を見て進んでいる。A-star 探索アルゴリズムとはロボットにマップの情報をすべて与えて、ゴールに一番近い方向の場所を高くして、出来るだけゴールに近い方向に動かすことにより、最短と思われる経路を探し出すことを目的としたアルゴリズムである。よってロボットは初めからマップが与えられている状態でゴールの位置も把握しており、ゴールに向かって最短経路で探索することが可能となっている。しかし我々の実験では、逐一迷路の壁を知覚してその場その場で進むべき方向を探索してゴールを導き出す手段を取ることを目標としている。この方法は現実的問題を考えると、他の研究よりも動的な環境に適応出来ていると思われる。

2.3 この実験による問題と考察

ロボットはエージェントのシミュレータ上での動きを完全に行うことができ、シミュレータ上のみでの行動は命令通りできるようになった。しかし、現実世界では床の質でロボットの行動に支障をきたすことが問題となった。段ボールのような凹凸のある床では回転に非常に誤差が出てしまい、まともな回転が出来なかったため、比較的凹凸の少ないプラスチックの床に変更して再度実験を試みたところ、ロボットはある程度正確に動くことが出来た。しかし何度も行動することが誤差の蓄積につながり、シミュレータ上での動きを床上で再現した時、シミュレータ上での位置とは大幅にずれた位置で行動を行うという結果になった。

これにより、ロボットの動作自体の誤差が一つの問題として浮き上がった。

また、シミュレータでのマス目の空間は現実空間から見ると離散的であるのに対し、現実空間の床は連続的な空間であるので、ロボットの行動に対してシミュレータ上でのマス目空間による空間のモデル化は適切であるかどうかは考慮すべき点であると思われる。

さらに、先程 1 章で述べたように、ロボットの行動誤差によってロボットは迷路内の壁でない場所に壁があると判断し、これが間違った信念としてエージェントに伝達され、エージェントは自身の信念に疑いを持つことなくその信念を信じて行動し、結果目標達成ができなくなることが観測された。

本研究ではこのうち最後の問題に焦点をあて、藤田が 2008 年に作成したプランナ [11] をエージェントと結合することで、エージェントに動的プランニングを行わせ、再知覚を行うプランに切り替えることにより問題

解決を図った。

3　基本知識と基本設計

3.1　Jason

Jason とは、BDI アーキテクチャを基礎とした BDI 記述用言語 AgentSpeak を拡張するためのインタプリタである。Jason は、エージェントを定義するエージェントプログラムに対し、そのエージェントが置かれる環境モデルを Java を用いて実装する。環境設定プログラムによって環境プログラムを指定する事により、その環境とエージェントの相互作用が可能となる。（図 2）

実験では Jason を用いて、エージェントの置かれた環境を計算機上で可視化した。

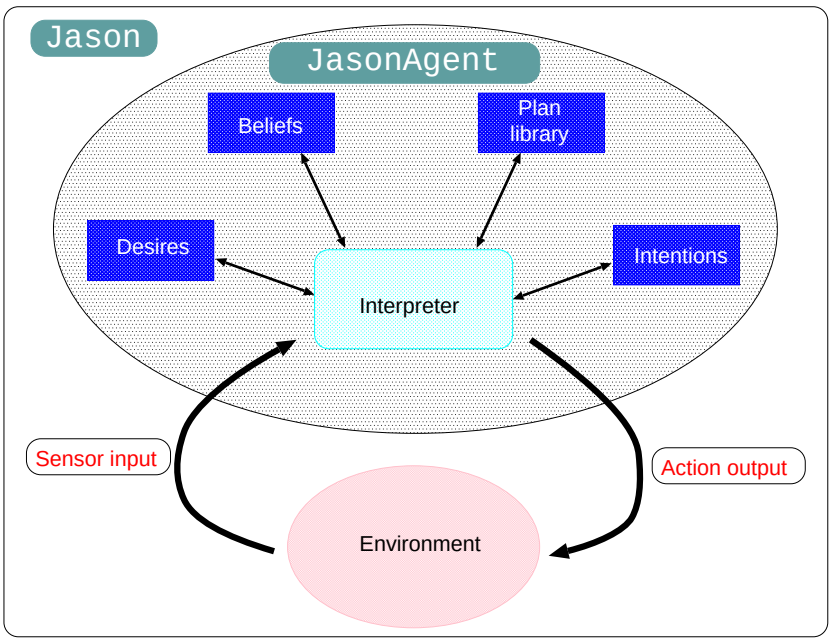

図 2　Jason と環境との相互作用

3.2　プランナ

プランナとは、エージェントの行動列生成プログラムのことである。エージェントは自身の初期状態と達成したい目標をプランナに与え、それをもとにプランナはその目標を達成する手段を組み立てる。この手段はエージェントにおける行動列（プラン）となる。

よく知られるプランナとして、TLPlan[4] や SHOP2[9] などがある。

これらは、初期状態と達成したい目標に値するゴールを与えられるとそのゴールを達成するための手段を最初から最後まで全て一つのプランとして生成する。従って、行為者が行う行動による効果のみが状態を変化させる手段であり、外部からの影響を一切考慮していない。つまり静的な状況における問題を解決するためのプ

ランニングプログラムである。

3.2.1 動的環境に適応するプランナ

今回の研究で用いるプランナは、藤田が作成した動的環境に適応するプランナ [11] である。このプランナは SHOP2 をプラン探索のエンジンとして使用している。SHOP2 とは階層的プランナであり、一階述語論理を扱えるプランナとして有名であるが、先程の 3.2 章で述べたように、動的環境には対応していない。

動的環境に対応するプランナの設計としては、

1. 初期状態からゴールまでのプランとして粒度の粗いプランを用意する（サブゴールと呼ぶ）
2. それぞれのサブゴールを、そのサブゴールが達成するための基本行為に分解する
3. この基本行為をこなすことにより、一つ一つのサブゴールを達成していくようにする

この動作によって、BDI エージェントにおける意図の選択と具体化を行っている。

また、プランナは問題依存しない設計である。従って、問題定義を記述しているドメインのファイルをこのプランナと共に起動時に読み込むことにより、自動的に粒度の粗いプランを行動列として作成することが可能である。これによりエージェントの意図の形成をすることが可能である。

そしてこのプランナのもう 1 つの利点は、計算量が通常のプランナよりも小さく見積もれることである。

通常のプランナは初期状態とゴール状態のみを考慮して、すべてのプランを基本行為から組み立てるよう設計されているが、このプランナでは、最初に作成された粒度の粗いプランはサブゴールとなり、副目標としてエージェントが新たに達成すべき目標に指定されるが、一つ一つのサブゴールはその目標自体を達成するべき時に初めて基本行為に分解される。よって、仮にあるサブゴールを達成すべく行動している間に、すでに次のサブゴールが達成されたと知覚された時、次のサブゴールは分解する必要がなく、その計算量は低く見積もることが可能である。

3.2.2 今回の研究における動的環境に適応するプランナの使用法

今回の研究での使用法として、エージェントに与えられた目標達成へのサブゴールの列は次のようになる。

まずエージェントの目標は、‘迷路の中を探索し、オブジェクトを発見し、元の道をたどり迷路を抜ける’ ことである。これによりサブゴールは、‘迷路を探索し、オブジェクトを発見する’ →‘バックトラックする’、という 2 つの目標から成る列になる。これらのサブゴールをこなすための基本行為を、今回は Jason におけるエージェントプログラムに組み込まれた行動制御を使って行っている。従って、実質サブゴールのみを信念の変更に伴って、もし達成条件が合えば達成されたものとみなし、次のサブゴールを達成するべくエージェントは行動を行う。

今回の迷路探索では、2 章で述べた前回の実験の結果のように、ロボットの行動の誤差によって壁でない場所を壁と認識する誤認が確認された。行動誤差の問題は一概にエージェントの問題として考える事は適切ではないが、少なくとも、間違った信念を訂正し再行動を行わせる手段としてプランナを構築することは、1 つの解決手段と言える。従って、最初のサブゴールでの迷路を探索する部分で、もし開拓していない場所が無くなり、新しく構築することが出来なくなったと判断した時点で、エージェントはプランナに次の行動を問い合わせる。そしてこのプランナは開拓していない場所を再確認するためのプランをエージェントに渡し、エージェントは現在保持している信念の信憑度を下げて、さらに探索の精度を上げて全方位知覚を行い、壁でない部分を発見した場合、その位置に壁がなかったことを正しい認識として、再び探索を行うようにしている。

3.2.3　プランナの動作の流れ

実際にプランナがどのように動作しているかという一連の流れは以下の通りである。

- エージェントプログラム起動と同時にプランナが起動する
- エージェントは空の初期状態をプランナに渡す
- プランナは、探索開始のサブゴールであり信念の価値観を絶対的なものとみなす “search” という戦略を実行するというプランを生成して返す
- 探索に失敗したとき、エージェントは探索経路を誤認したと見なしてプランナに問い合わせ、プランナは自身の信念を疑いながら探索を行う “research” という戦略を実行するプランをエージェントに返し、エージェントは現在の信念の信憑度を下げて全方位探索を行う
- オブジェクトを発見したらプランナにその情報を渡し、次のサブゴールである ‘バックトラックをして元の場所に帰る’ というゴールを返してもらう

以上の動作を行い、エージェントの行動制御に携わっている。

本研究でのロボット制御の全体の概要は図 3 のようになる。

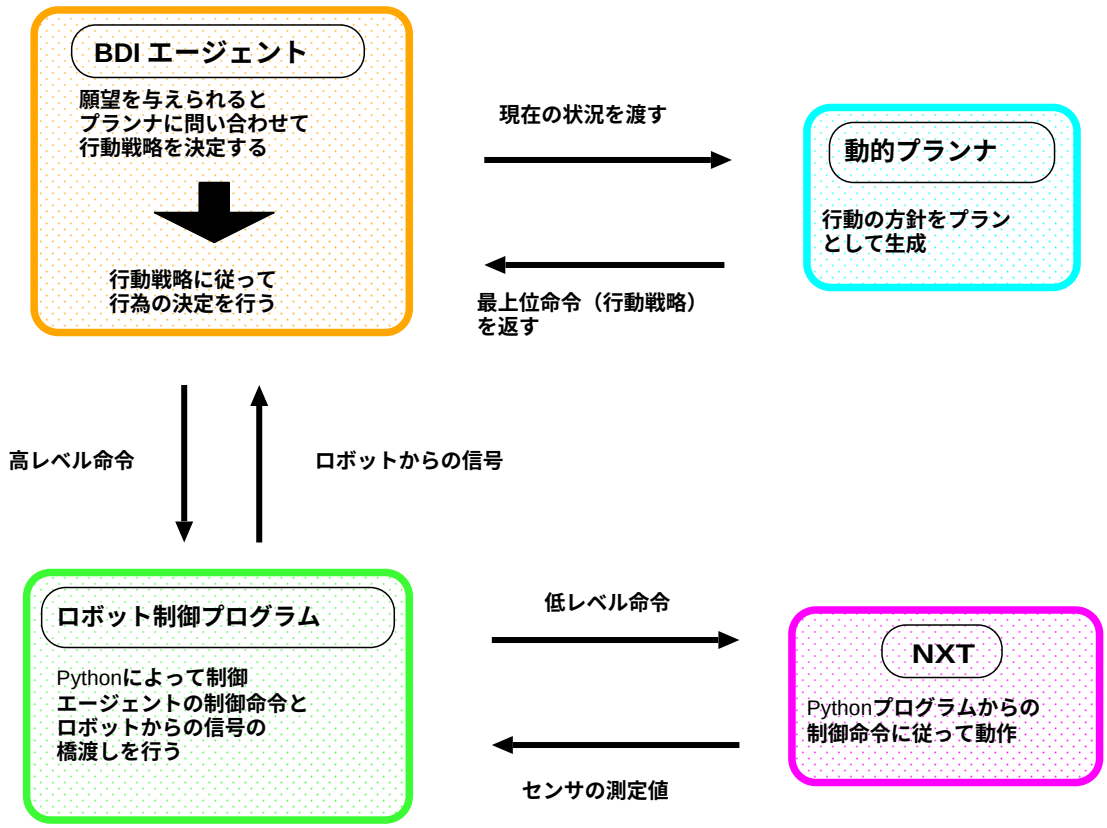

図 3　ロボット制御の全体の概要

4 Jason とプランナの結合

4.1 設計方針

本節では、本研究のうち著者が担当した部分である、Jason(BDI エージェントが置かれている環境モデルを Java を用いて実装したもの) とプランナの通信部分の設計について述べる。Jason の環境ファイルは Java で、プランナは Common Lisp で実装されている。実装するプログラミング言語が違うため、プランナを外部プロセスとして Jason と通信するように設計を行った。設計方針としては、Jason と外部プロセスとの入出力を考え、Jason が出力する文字列を外部プロセスに渡し、外部プロセスから返ってくる結果を Jason に渡すという Java クラスを作成する。この Java クラスが Jason に提供する機能は以下の 2 つである。

1. 外部プロセスに文字列を渡す機能
2. 外部プロセスから出力結果を受け取る機能

これとは別に Jason 側では、1. の機能を呼んで文字列を渡し、2. の機能を呼んでプランを受け取るようにするため、以下 3 つのメソッドを持つ Java クラスを作成した。

1. Jason 側から渡された文字列を外部プロセスに渡すメソッド
2. 外部プロセスから返ってくる出力結果を受け取るメソッド
3. 2. の関数からプランの部分だけを抽出して Jason 側に渡すメソッド

1. と 2. はプランナに依存せず、3. はプランを抽出するためプランナに依存している。2. と 3. で分離して設計することにより、機能拡張に汎用性を持たせた。

4.2 実装方法

図 4 のように、外部プロセスを生成しプロセスへの入出力を行う process クラス、そして process クラスと Jason を連結する役割を持つ message クラス、2 つの Java クラスを用意した。前者は 4.1 章の前半で述べたもの、後者は後半で述べたものにあたる。

process クラスでは、外部プロセスを生成し、プロセスのストリームに対して文字を書き込む write メソッド、プロセスのストリームからプランナの出力結果を 1 行ずつ読み出す readline メソッドを定義している。

message クラスでは、Jason 側から外部プロセス側へ文字を渡すスレッド (スレッド 1)、外部プロセス側から Jason 側へ文字を渡すスレッド (スレッド 2) が定義されている。そしてキューという、先に入力したデータが先に出力される (FIFO) という特徴をもつデータ構造が 2 つ定義されており、queue1 は Jason 側から渡された文字列を格納するキュー、queue2 は外部プロセス側からの出力結果を格納するキューである。なお、この 2 つのキューは Java の LinkedBlockingQueue というクラスで定義している。これは要素の取得時にキューが空で無くなるまで待機したり、要素の格納時に容量の制限が無いなどの機能があるためである。このことによって、外部プロセス側と同期せずに Java 側は任意のタイミングで入力を渡したり出力を要求したり出来るようになっている。

message クラスではこの他に、queue2 から取得した文字列からプランの部分を抽出するメソッドを作成した。プランは三重括弧で囲まれて出力されるので、正規表現で ‘(((’ で始まり ‘)’ で終わる部分を探し出し、その部分だけを Jason に渡すように設計した。

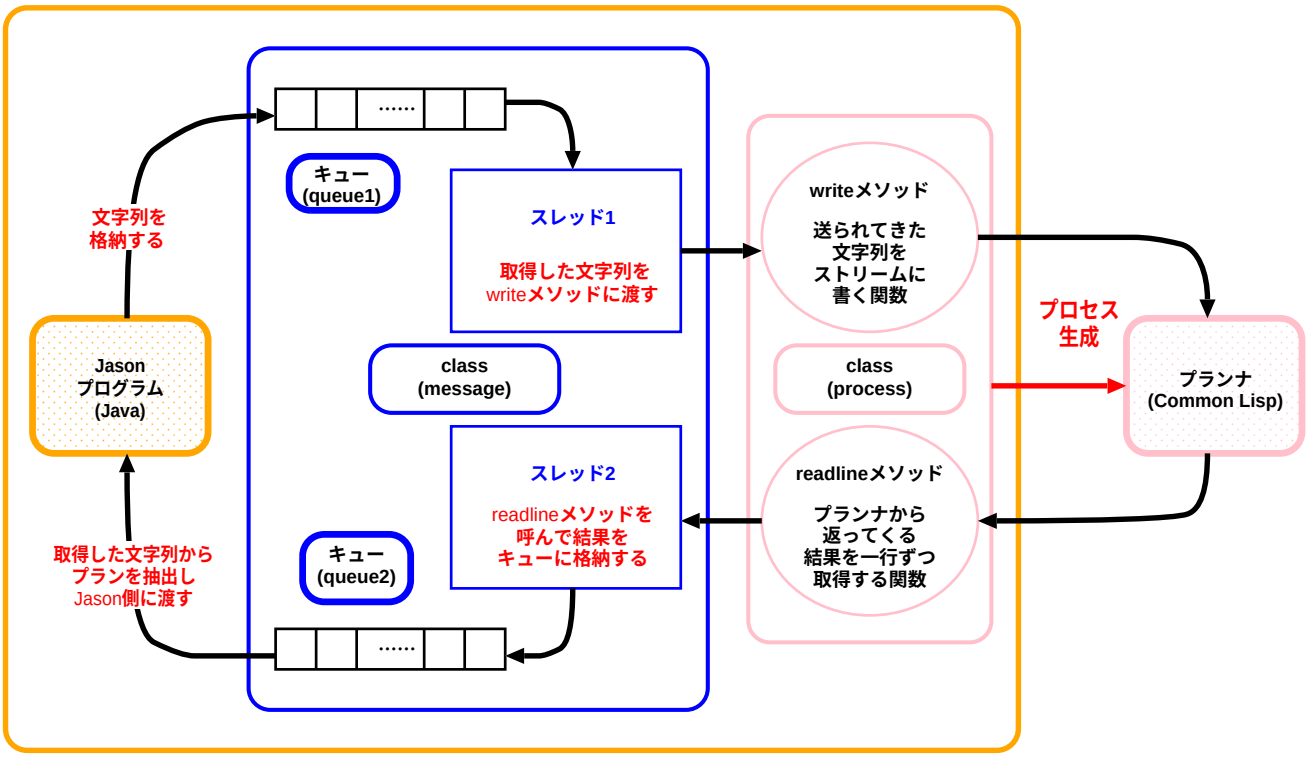

図 4　Jason とプランナの通信機構

通信の流れは以下の通りである。

1. message クラス、process クラスのインスタンスが生成されプロセスを生成、プランナが起動する
2. Jason 側から渡された、現在の状況である文字列を queue1 に格納する
3. スレッド 1 が queue1 から文字列を取得し、process クラスの write メソッドに渡す
4. write メソッドが外部プロセスのストリームに文字列を書き込む
5. スレッド 2 が process クラスの readline メソッドを呼び出し、外部プロセスの出力結果を 1 行読み出す
6. 読み出した 1 行を queue2 に格納する
7. queue2 から取得した文字列から、プランの部分を抽出して Jason 側に渡す

以上のように設計をし、通信機構を実装した。

5　実験

4 章で説明した設計に基づき通信機構を取り付けることで Jason とプランナ間での通信を行うことが可能となり、前回の実験と同じく迷路を探索するという題材で実験を行った。

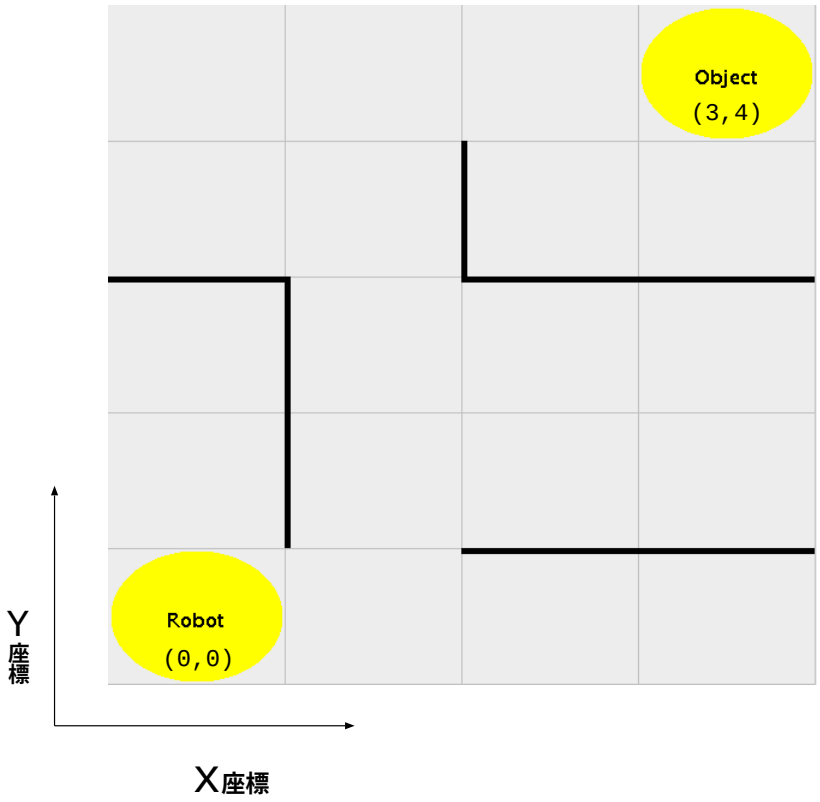

図 5　4 × 5 のマス目空間の迷路 (マップ)

5.1　実験手法

実験における問題の設定は 1 章、2 章で述べた通り、4 × 5 のマス目空間の迷路 (図 5) の中をロボットが探索を行い、その迷路の中に配置されたオブジェクトを発見することである。ロボットは迷路の中のマップを前提知識として所持していない。従って、ロボットが自身のセンサで壁を認知し、それがエージェントの信念に加えられ、マップが構成されていくことになる。

前回の実験と異なる点は、図 6 のように、1 と 2 の位置に壁を取り付けてロボットに知覚させ、意図的に誤認を生じさせた。後にその壁を外してロボットがその誤認を行った場所を再知覚し、それを認知するかどうか実験を行った。

5.2　結果

実験の結果、先程述べたとおり迷路の中に意図的に壁を取り付け、マップの空間を閉鎖したところ、search の戦略から research の戦略に変更されたことが見受けられた。壁の信念のある方向は知覚を行わず、壁のないと思われるところを開拓していく search に対して、research は壁の有無に関わらず、自らが進行してきた方向以外の全ての方向に対して知覚を行った。

さらにその取り付けた壁を外して、その部分の知覚をロボットが行った後、ロボットは壁を取り外した方向

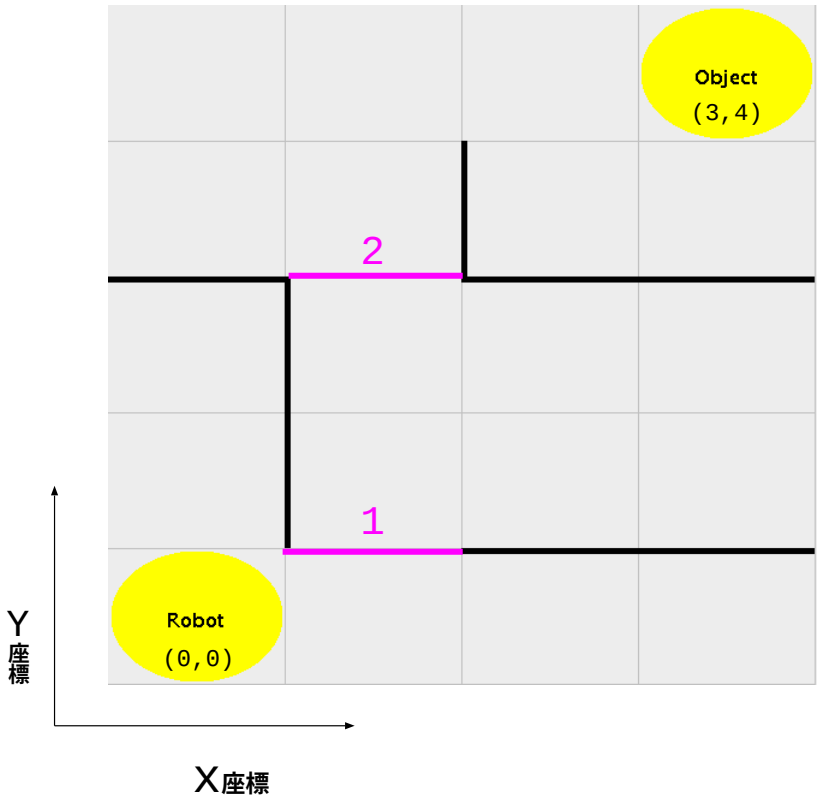

図 6　誤認を生じさせたときの迷路

に進み、search の戦略に戻って未開拓な領域を探索し始めた。

以上により、今回の誤認識による目標が達成できなくなる事態を避けることが可能となった。また、再度別の場所に壁を取り付けて誤認識を行わせた場合においても、毎回戦略を変更することにより進路が阻まれることは起こらなくなった。

6　考察

“search” と “research” の戦略を取ることで行動戦略を変更することにより、壁の誤認識によって起こる弊害は改善された。

また、4.2 章で述べた 2 つの Java クラスは、OS に依存しないので移植性があり、結合させるプログラムにも依存しないのでプランナ以外のものを取り付けることも可能である。更に、Jason 側は message クラスのインスタンスを生成して文字列を渡すメソッド、プランを受け取るメソッドを呼び出すだけで良いので、独立性の高い設計を行うことが出来た。

7　まとめ

今回、実世界における BDI エージェントの挙動と、現実とシミュレーションでの問題点を検証する実験を行い、ロボットの制御における弊害についての知見を得、これを元に問題点の克服を図った。特に、Jason に

ない機能を新たに別プロセスで実行し通信を行うことで Jason の機能を拡張し、これによりロボットの行動がより確かなものとなった。

今後の方針としては、作成した通信機構の設計及び実装により汎用性を持たせ、複雑な問題にも対応出来るよう改良していきたい。

8 謝辞

本研究を遂行するにあたり、丁寧にご指導して下さった指導教官の新出尚之准教授に深く感謝し、厚く御礼申し上げます。また普段から丁寧な助言を与えてくださっている藤田恵先輩、そして鴨浩靖准教授、更に新出、鴨研究室の皆様に感謝の意を表します。ありがとうございました。

参考文献

[1] Lego mindstorms nxt. http://www.nebomusic.net/NXT-G-DriveASquare.html.

[2] Nxt tutorial. http://www.ortop.org/NXT_Tutorial/index.html.

[3] Lego mindstorms nxt: Welcome to the next generation. http://www.tik.ee.ethz.ch/tik/education/lectures/PPS/mindstorms/sa_nxt/index.php?page=print, 2006.

[4] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, Vol. 116, pp. 123–191, 2000.

[5] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason.* WILEY, 2007.

[6] Hart P. E, Nilsson N. J, and Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, pp. 100–107, 1968.

[7] Soren Andreas Juul, Kaspar Henrik Moss Lyngsie, Michael Vandborg, Kim Fiedler Vestergaard, Torsteinn Sævar Hjartarson, and René Bach Gustafson. Agent Programming — Robot Traffic. *Aalborg University*, 2008.

[8] Anders Lemke, Johan Laidlaw, and Lars Zilmer-Pedersen. Developing Multi-Agent Lego Robotics. *Technical University Of Denmark*, 2007.

[9] Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. Shop2: An HTN planning system. *Journal of Artificial Intelligence Research*, Vol. 20, pp. 379–404, 2003.

[10] Munindar P. Singh, Anand S. Rao, and Michael P. Georgeff. Formal method in DAI. *Multiagent systems: a modern approach to distributed artificial intelligence*, 1999.

[11] 藤田恵, 新出尚之. 動的な環境に対応する BDI エージェントのためのプランナの設計と実装. *Proc. of Joint Agent Workshops and Symposium 2008*, 2008.

[12] 藤田恵, 小島侑子, 片山寛子, 新出尚之. 実世界での BDI ベースのロボットの柔軟な行為決定の実現に向けて. *Proc. of Joint Agent Workshops and Symposium 2009*, pp. 354–361, 2009.