情報処理学会論文誌　数理モデル化と応用　Vol.5　No.1　50–64 (Mar. 2012)

実世界の多様性に適応したBDIロボットについて

藤田 恵[1,a)]　片山 寛子[1]　新出 尚之[2]　高田 司郎[3]

受付日 2011年8月19日，再受付日 2011年10月7日，
採録日 2011年10月28日

概要：近年，単一の目標達成に特化されたロボットでなく，動的に変化する環境のもとで目的を達成するよう振る舞うロボットの実現が重要な研究課題となってきている．われわれは，BDIモデルを実装したBDIロボットによる行為選択の柔軟性および意図の整合性と再考慮が，複雑で多様性に富んだ実世界における意思決定機構として有効であることを示す．特に，意図の保持と破棄という機構によって，目標を達成する手段を柔軟に切り替えたり，複数の目的を並行に保持し整合的に実行したりできるBDIモデルの能力が，多様性への適応に非常に有用であることを論じ，また，実験を通じてそれを示す．

キーワード：BDI，実世界指向，自律ロボット

BDI Robots Who Adapt to the Diversity of the Real World

Megumi Fujita[1,a)]　Hiroko Katayama[1]　Naoyuki Nide[2]　Shiro Takata[3]

Received: August 19, 2011, Revised: October 7, 2011,
Accepted: October 28, 2011

Abstract: Recently, it has been considered important issue to realize robots which acommplish their purposes dynamically in the real world environments. We propose that the BDI robots who implement the BDI model are profitable for such environments because they can acommplish their purposes dynamically. Indeed, the mechanism of maintaining and discarding the robots' intentions can switch their methods for achieving their goals flexibly, and can maintain consistency of concurrent actions. Therefore, we claim that these abilities of the BDI model are profitable for adapting the diversity of the real world, and show them by the experimentation.

Keywords: BDI, real world oriented, autonomous robots

1. はじめに

これまでに実用化されているロボットの多くは，単一の目標達成に特化されたもので，たとえば二足歩行エンターテインメントロボットなどは実世界の多様性[*1]を考慮に入れることなく，つまり多様な環境変化に適応することなく，いかにダンスなどの振舞いをうまく表現するかに重点が置かれている．また，振舞いを実現するために，アクションの状態遷移を網羅してハードコーディングされているものが多い．このようなロボットは，確かに身体を持つが，たとえば空き缶を拾って環境をきれいにするなど，環境と相互作用するために身体を持っているわけではない．つまり実世界を一定の仮想世界と仮定してロボット制御が行われている．一方，人間と協調的に日常生活を行うことを指向したヒューマノイド型ロボットなどは，人間や環境の多様な変化に適応して，複数の目的を整合的に達成していく必要がある．たとえば看護ロボットなどでは，患者から飲み物を要求されたので冷蔵庫に向かうが，向かっている途中に患者がベッドから落ちてしまった場合，飲み物を後回しにして倒れ込んだ患者をベッドに戻すことを優先する必要

[1] 奈良女子大学大学院
Graduate School of Humanities and Sciences, Nara Women's University, Nara 630–8506, Japan
[2] 奈良女子大学
Nara Women's University, Nara 630–8506, Japan
[3] 近畿大学
Kinki University, Higashiosaka, Osaka 577–8502, Japan
[a)] saboten@ics.nara-wu.ac.jp

[*1] 本論文での「多様性」とは，環境が一様でなく多様に変化することを指す．

がある．さらに，患者に飲み物が欲しいか聞き直して，もし飲みたいという意図を引き続き持っていれば冷蔵庫に向かわなければならない．後者のように，矛盾なく達成すべき目的の数が増えれば増えるほど，それら状態遷移の記述すら困難なことは容易に想像される．

われわれは，2.3 節で紹介する「意図の理論」をベースにしたBDI モデル [14] に基づいたロボットの意思決定が，そうした多様性に富む実世界において複数の目的を整合的に達成するには有効であると考えている．しかし，これまでにそうした有効性を示す実験は行われてこなかった．BDI モデルをロボットに導入した実例は少数あるが，5.4.1 項で述べるように，それらはこうした実世界での複数の目的の整合的達成という点での BDI モデルの利点を示しているものではない．

そこで本論文では，BDI モデルを実装したロボットを BDI ロボットと呼び，実世界の多様性に適応して，複数の目的を整合的に達成するには，BDI ロボットが有効であることを実験を通じて示す．

状況の変化に適応した振舞いをするシステムとしては，5.4.2 項に述べる自己適応システムやサブサンプションアーキテクチャなども知られ，それらはおおむね，階層構造を持つモジュールからなり，センサによる環境変化の検出に行動の変化が結びつくことによって振舞いを変えるという構成を持つ．しかし，同項で論じるように，BDI モデルはそれらに比べ，多様性に対するより柔軟な振舞いの変化を実現できる．それは，BDI モデルでは 2.3 節に述べるように，振舞いの変化が環境の変化のみから直接起きるのでなく，心的状態を用いて表現される，熟考やコミットメント戦略などの概念に基づいて実現されることによって，より一般的な行為選択のモデルを提供していることによる．

以下，本論文では，2 章で BDI モデルとその実世界への適用について，3 章で今回実験に用いる BDI ロボットの仕様について，4 章で実世界の多様性への適用実験の内容と実験結果について，5 章で BDI モデルの利点や関連研究との比較などの考察について，それぞれ述べ，最後に 6 章でまとめる．

2. BDI モデル

本章では，まず仮想世界における BDI モデルとその応用事例を述べ，次に実世界のさまざまな問題を列挙し，最後にそれら問題と意図の理論との対応について述べる．

2.1 仮想世界における BDI モデル

一般的に，エージェントはある環境をセンサを通じて感じ取り，その環境にアクチュエータを通じて働きかける行為者である．特に，環境に関する知識（信念）に基づき目的を達成するよう振る舞うエージェントを合理的エージェントと呼び，BDI モデルは合理的エージェントモデルの 1 つである．BDI モデルでは，エージェントは，B（Belief），D（Desire），I（Intention）の 3 つの心的状態とその時間的変化を用いて意思決定を行う．Belief は信念，すなわちエージェントが信じている実世界に関する不確かな知識（計画を含む），Desire はエージェントの欲求，そして Intention は意図，すなわちエージェントが目的を達成するために行おうと目指している心的状態である．

BDI モデルに基づく合理的エージェントは広く用いられてはいるが，これまでこの応用は，離散的形式的に明確に定義できる仮想世界におけるソフトウェアエージェントが主体である．仮想世界における BDI モデルの事例として「航空管制システム」があげられる．航空管制を行う際に必要な各飛行機の位置，速度，進行方向などの情報は，時々刻々，数値として「完全」に得られることを仮定して設計されている．また，実世界の BDI モデルの事例として「利用者と協調して写真を撮る対話エージェント」[18] があげられる．利用者の動きが連続的に変化する実世界の事例ではあるが，利用者と協調して写真を撮るのに必要な情報は「利用者の両目の位置」と「音声による撮影依頼」のみであり，人間の両目の位置を実時間でほぼ完全に追跡できる画像処理技術とシステム主導による音声対話技術を用いることで，ほぼ完全に得ることができる．

2.2 実世界のさまざまな問題

ところが，心理学者戸田正直が提案した「キノコ食い」のような完全エージェント [17] が身体性を有して，実世界で長い期間生き延びて機能するためには，仮想世界とは異なる以下のような問題が生じる [12]．

(1) 情報獲得に時間がかかる．
(2) きわめて限られた情報しか得られない．
(3) 物理デバイスは外乱や故障から逃れられない．
(4) 実世界を離散的には記述できない．
(5) 実世界のエージェントは，つねに複数のことを並行して行う必要がある．
(6) 実世界は固有のダイナミックスでつねに変化している．
(7) 実世界はきわめて複雑なダイナミカルシステムであり，その非線形特性と初期値に対する鋭敏性ゆえに，本質的に予測不可能である．

たとえば，身体性を持ったエージェント，つまりロボットは，(1) 隣の部屋にプリンタがあるのかどうか知りたければ，行ってみるか，だれかに聞くか，あるいは，ネットワーク構成図を見なければならないなど，実世界から情報を取り出すには時間を要し，(2) カメラの視界内の部分的な情報しか得られず，(3) 暗闇では，輝度が不足して障害物検出ができない，(4) スムーズな二足歩行の振舞いを，離散的な状態に分けて記述することは難しく，(5) 看護ロボットであれば，患者の話し相手をしながら，飲み物を運んだ

り，ベッドに寝かせたり，体温や脈を測りながら主治医へ連絡したりするなど，患者の状態や要求に応じて，複数の目的を並行して行う必要があり，(6) 坂道での二足歩行は，自らの重さに重力が働き，バランスを崩すと転がりはじめ，うまく起き上がるのは難しい，(7) サーファロボットであれば，荒い波のぶつかり合いなどによる多様な波の変化は予測不能である，などの問題に直面する．

そこでわれわれは，このような実世界の特性に適応するために，BDI モデルのベースである Bratman の意図の理論 [3] に立ち返る．

2.3　意図の理論

意図の理論とは，実世界における合理的エージェントの意図に関する哲学的な分析である．以下，実践的推論，計画立案，意図の整合性と再考慮，およびコミットメント戦略について述べる．

2.3.1　実践的推論

意図の理論によれば，われわれ人間は，非常に複雑でつねに変化する実世界において，複数の目的を達成するために計画立案する合理的エージェントである．われわれは，目的を達成するためには，まずどのような様態（目標）を達成すべきであるかを決定し，次に，その目標をどのような手段を用いて達成するかを決定する．前者を熟考，後者を目的–手段推論と呼び，これら 2 つを合わせて実践的推論と呼ぶ．また，現在形成されている複数の意図の中から，どの意図を次に実行するか推論することも熟考と呼ぶ．

意図を OS のタスクに例えると，熟考はタスクのスケジュール管理に該当する．つまり熟考は，意図間の優先順位や整合性を維持したタスク管理であり，言い換えれば「メタプラン」を行っていることになる．

2.3.2　計画立案

われわれは，実世界では，計画立案に必要な信念（belief）をすべては保持できない（信念の不完全性）．これら信念から未来の予測をすべて計算することはできない（推論能力の有限性）．また，そのつどの状況に応じて瞬時に判断を下して行為を繰り返すことで複雑な目的を達成することは難しい．このように，われわれの能力は限られたものであり，未来において行うことすべてを特定するプランを一挙に立てることは不可能である．また，われわれが住む実世界はつねに変化しているため予期せぬ事態が発生し，事前に立てた計画はまったく無駄に終わるかもしれない．

そこで，われわれは最初からプラン本体をすべて具体的に実行できる行為列で埋めたプランではなく，環境の変化に柔軟に適応できる程度の粒度からなる副目標を用いたプランを立案した後，そのプランを意図として形成した後，実行に移す．そして，まだ埋められていない副目標を実行するときが来たと判断したとき，その状況に適応した，副目標を達成する（サブ）プランを目的–手段推論し，そのプラン本体を手段として実行する．このように状況の予測が容易でないときは，副目標の達成手段の推論を先送りすることで，多様な環境変化に柔軟に適応した意思決定を行うことができる．

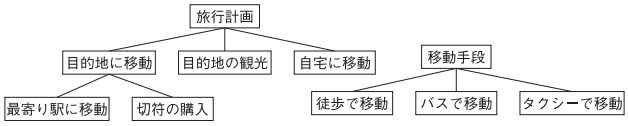

図 1　旅行計画
Fig. 1　Travel plan.

たとえば，**図 1** のように，「旅行計画」を達成するためには，「目的地に移動」，「目的地の観光」，「自宅に移動」などの達成すべき副目標を前もって定めることができる．その後，グループで「調整」し，目的地や日程などを特定して「目的地の観光」に埋め込む．次に，交通機関の時刻表を調べるなど「予備的な手続き」を行い，「目的地に移動」に移動手段などを埋めていく．そして移動当日，まだ具体的な移動手段を埋めていない「最寄り駅に移動」を，たとえば電車の予定出発時刻までの時間間隔に応じて「徒歩で移動」か「バスで移動」か「タクシで移動」かの「移動手段」を埋めた後，最寄り駅まで移動する．このように，われわれは，そのままでは実行できない副目標からなる部分的なプランを前もって定め，調整や問題（未来の予期せぬ事態など）に直面したときに限って，手段や予備的な手続き，または（副）目標を特定して部分的な計画に埋め込み，意図持続の戦略であるコミットメント戦略（後述）に基づき意図を達成しようとする．

2.3.3　意図の整合性と再考慮

合理的エージェントは整合的でない意図を形成しない．つまり，別の意図 q の達成が本来の意図 p の達成を不可能にするような場合は q を意図として形成しない．

また，以下のような場合は，意図の再考慮が行われる．

(1) 計画時に想定した環境の状況と現状とに相違点がある場合．つまり，計画した時点で誤った信念を持っていたことに気がついた，または，現状が計画時の予想とは異なる場合
(2) エージェントの欲求または価値意識に問題となるような変化があった場合
(3) 他の意図のいくつかに問題となるような変化があった場合．つまり，意図間の整合性がとれなくなった場合

2.3.4　コミットメント戦略

次いで，意図の持続と破棄に関する「コミットメント戦略」[14] について紹介する．

(a) Blind（盲目的）：エージェントは，意図はすでに達成されていると信じるまで，その意図を持続する．
(b) Single-minded（一意専心）：意図はすでに達成されていると信じるか，もしくは，その意図の達成が可能で

あると信じなくなるまで，その意図を持続する．

(c) Open-minded（心の広い）：その意図を形成した欲求を実現するという状況でなくなるまで，意図を持続する．つまり，その意図はすでに達成されたと信じるか，その欲求を実現する理由がなくなったので取り下げるまで，その意図を持続する．

合理的エージェントは，このようなコミットメント戦略に基づいて意図の持続や破棄を行う．たとえば，甲子園の高校野球観戦を考えてみよう．高校野球日和の炎天下では，「生ビールが飲みたい」という欲求が生じたので，「生ビールを入手する」という目的を達成するために「生ビールを入手する」という意図を形成する．Blind コミットメントでは，あらゆる手段を講じて生ビールを入手できるまでこの意図を持続する．Single-minded コミットメントでは，生ビールを入手するために「生ビールの売り子を待っていた」が来てくれなかったので，「球場内の近くの売り場に生ビールを買いに行く」が生ビールは売り切れていた．そこで，別の売り場に生ビールを買いに行く．入手できれば，この意図を破棄する．しかし，すべての売り場で売り切れていれば入手できないことを確信して，この意図を破棄する．Open-minded コミットメントでは，にわかにスコールのような雨が降り出して，とてもビールを飲みたいというような状況ではなくなったので，ビールを入手するという意図を破棄する（ただしスコールが止めばこの意図は復帰するはずである）．

コミットメント戦略を用いれば，以下のように，基本行為レベルや目標レベルでの再考慮も可能となる．たとえば，基本行為が失敗したときは，その基本行為を再実行するか，またはその基本行為が実行できなければ，現在目指している目標を達成する別の手段を目的–手段推論して，その目標を持続することができる．さらに，この目標の達成手段がなくなれば，目標レベルで再考慮し，意図を達成する別の目標を熟考して，その目標を達成するよう意図を持続する．このような再考慮は意図を持たない（たとえば目標だけを持つ）エージェントで実現することは難しい．

2.4　実世界の諸問題と意図の理論の対応

そこで，実世界の諸問題と意図の理論との対応を考えてみる．まず 2.2 節の (1) に対しては，実世界の情報を得るプランを意図として保持し，情報が必要になれば実行することで，実世界と相互作用を行い所望の情報を得る．または他のエージェントに依頼することで得ることもできる．

次に同節 (2) に対しては，行為列を実装していない副目標を用いたプランを意図として保持・実行することで，状態が分かるまで保留するという場合が考えられる．また，センサの性能で自力では正確な情報が得られない場合は，正確な情報を得ることができるエージェントに依頼するプランを意図として保持・実行することで得ることができる．

次に，同節 (5) に対しては，複数の目的を状況に適応して実行するには，それぞれの目的を達成する意図を保持し，並行に実行することができる．

以上のような考察に基づき，われわれは，上記 (1), (2), (5) のような実世界の問題への対応として，意図の理論に基づく上記のように振る舞う BDI ロボットを実装し，実世界の多様性に対して，BDI モデルの行為選択の柔軟性および意図の整合性と再考慮が有効であることを，次章以下の実験を通じて示す．また，その他の問題への対応は 5.3 節で述べる．

3.　BDI ロボット

本章では実験に用いたロボットの構成，および BDI ロボットの実装について述べる．

われわれは今回，2 台のロボット，Explorer と Forklift（以下，Lift）を用意した．Explorer の目的は，掃除を行うこと，荷物を置く台を発見した場合に Lift に伝えること，Lift に依頼されたときに台の識別を行うことであり，Lift の目的は自身の持つ荷物 1 個をその台に置くことである．

初期状態ではこれらの BDI ロボットは台がどこにあるかは知らず，台の位置に関する信念はロボットの知覚行為によって加わっていくこととなる．

3.1　ロボットの構成

われわれが用意した 2 台のロボットは，LEGO 社が開発した商用教育用のロボット LEGO MINDSTORMS NXT を用いて作成した．NXT は安価で比較的制御が簡単であり，また，多数のパーツを組み立てて 1 つのロボットを作成するためロボットの形状はさまざまなものに仕上げることが可能である [6] ので，実験に用いるには適したロボットである．2 台の作成にあたっては NXT の組み立て用 Web ページ [11] を参考にした．2 台の主な特徴を以下に述べる．

- Explorer
  Explorer は**図 2** のような形状をしたロボットであり，
  – コンパスセンサ
  – 超音波センサ
  – タッチセンサ
  を持つ．
- Lift
  Lift は**図 3** のような形状をしたロボットであり，
  – コンパスセンサ
  – 超音波センサ
  – フォークリフト
  を持つ．

3.2　ロボットの基本行動

ロボットの基本行為としては，グリッドワールドでの行動をにらみ，次のようなものを用意する．これには Python

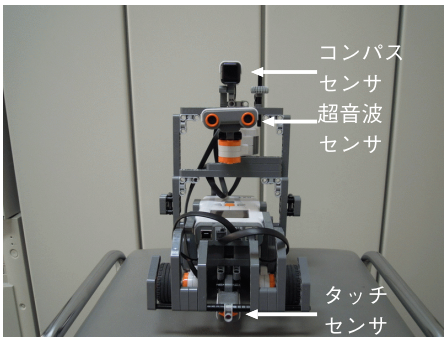

図 2　Explorer 正面図
Fig. 2　Front view of Explorer.

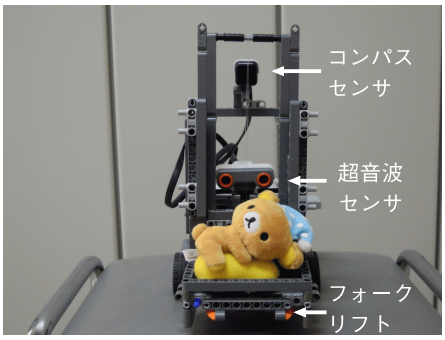

図 3　Lift 正面図
Fig. 3　Front view of Lift.

によるライブラリ NXT_Python [9] を用いている．

- 1 マス前進
- コンパスセンサを用いた現在の方向の知覚，および 90 度単位の回転（右，左）
- 超音波センサによるロボットの前方方向への知覚
- タッチセンサによる衝突認識（Explorer のみ）
- フォークリフトの上にあらかじめ載っている箱を台の上に置く（Lift のみ）

3.3　環境設定

実験の環境設定は以下のものである．

- 実世界上にグリッドワールド状の盤を設置（**図 5** は配置の初期状態の例で，大きさが 5×4，左上のマスの座標が $(0, 0)$，右下が $(4, 3)$）
- Explorer と Lift と台をグリッドワールド上のマス目に配置
- 台は，荷物を置く台（stand）と障害物（obstacle）の 2 種類がある
- 荷物を置く台は 1 個だけ，障害物は複数個

また，以下の理由により Lift は台を見つけた場合，それが荷物を置く台であるかどうかの判定は Explorer に委託する必要がある（**図 4**）．これらは 2.2 節の (1) や (2) への対応を示すための設定である．

- Explorer の超音波センサは荷物を置く台より高いため，障害物のみ認識できるが，タッチセンサは荷物を置く台を認識できるため，両センサの併用で台の種類を見分けられる．
- Lift は超音波センサのみ持ち，それは荷物を置く台より低いため，台を認識はできるが，台の種類を見分けることはできない．

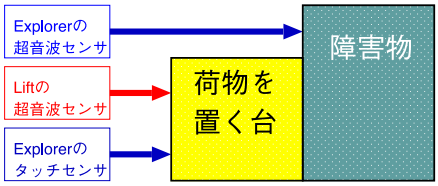

図 4　ロボットのセンサと台の高さの関係
Fig. 4　Relation of robots' sensors and heights of stands.

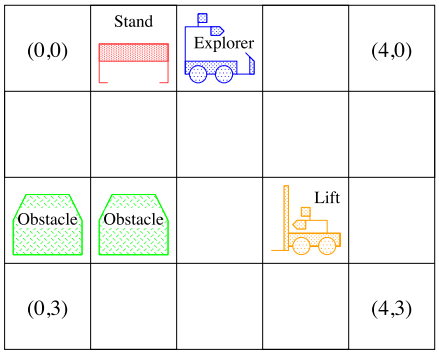

図 5　実験の環境
Fig. 5　Environment of our experiment.

```
BDI-interpreter
initialize-state();
do
    options := option-generator(event-queue, B, D, I);
    selected-options := deliberate(options, B, D, I);
    update-intentions(selected-options, B, D, I);
    execute(I);
    get-new-external-events();
    drop-successful-attitudes(B, D, I);
    drop-impossible-attitudes(B, D, I);
until quit.
```

図 6　BDI インタプリタ
Fig. 6　BDI interpreter.

3.4　Jason

われわれは，2 台のロボットが 3.3 節に述べた環境設定におけるそれぞれの目的を達成するよう，BDI ロボットを実装した．実装には，BDI モデルに基づくエージェントアーキテクチャである BDI アーキテクチャ [15] の実装プラットフォーム，Jason [2] を用いた．まず本節では BDI アーキテクチャと Jason について述べ，次節以降で BDI ロボットの具体的な実装について述べる．

3.4.1　BDI アーキテクチャ

BDI アーキテクチャに基づくエージェント（BDI エージェント）は，一般的に**図 6** のようなループ（BDI インタプリタ）で実装される [15]．ここで，B は信念，D は欲求，I は意図をそれぞれ表すデータ構造である．ただしエージェ

ントが自ら欲求を生成することは考えにくいため，BDI エージェントの欲求とは，具体的には，エージェントに外部から与えられた達成すべき「目標」と考える．

初期状態が与えられると，まず，event-queue を読み，エージェントに対する依頼（目標）であれば，実践的推論を行って，（未来に実行すべき）意図を形成する．そして現在，保持している意図の中から今の状況で実行可能な意図の候補を options に返す．次に deliberate を用いて，次の時刻に実行すべき意図を options から熟考して 1 つ決定する．次に，選択された意図が持つプラン本体（行為列）から次に実行すべき行為を決定するために，意図の更新を update-intentions で行う．そして，決定された行為が基本行為であれば execute で実行する．そうでなければそれは副目標であり，新たな目標（欲求）として event-queue に追加する．次に，get-new-external-events により環境知覚器から環境に関する情報を得て，event-queue に追加する．最後に，コミットメント戦略に基づき，成功裏に目標を達成した意図を drop-successful-attitude で破棄する．また，不整合の原因と判断された意図や，single-minded や open-minded コミットメント戦略に基づいた意図の破棄を drop-impossible-attitude で行う．BDI エージェントは，このループを繰り返すことにより，複数の目標を達成する．

3.4.2　Jason インタプリタ

Jason [2] とは，BDI アーキテクチャに基づくエージェント記述言語 AgentSpeak [13]（の拡張）の処理系であり，3.4.1 項に述べた BDI インタプリタの実現である．Jason では，エージェントのプランや信念は Prolog に似た文法を持つルールの形で宣言的に記述し，環境モデルの定義は Java 言語で行う．後者にはエージェントと環境の相互作用に関する記述，特にエージェントの環境知覚や基本行為の定義も含まれる（**図 7**）．定義された環境知覚や基本行為はプラン内で用いることができる．

プラン記述は基本的に

目標
:　プランを適用するための前提条件
<-　プラン本体（基本行為または副目標の列）

という形をとる．プラン頭部の「目標」は実際には，その目標を生じさせるイベントの発生として記述される．Jason インタプリタは，イベントが発生していれば，それに合致する頭部を持つプランの中から 1 つ選んで（「目的–手段推論」），エージェントの意図の集合（図 6 の options）に追加する．次いで現在の意図のうち 1 つを選んで（「熟考」，同図の deliberate），その本体の行為 1 つ（基本行為または副目標）を実行する．これを繰り返す．

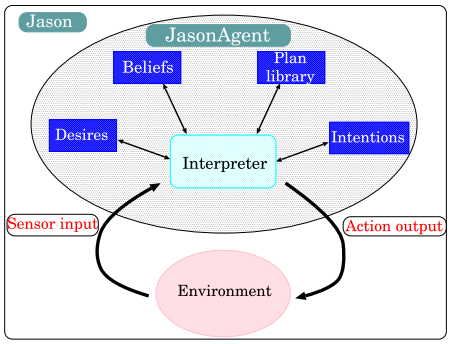

図 7　Jason と環境との相互作用
Fig. 7　Interaction of Jason and its environment.

3.4.3　熟考ルーチンの拡張

Jason であらかじめ（デフォルトで）用意されている熟考（deliberate）ルーチンは，現在実行中の意図をラウンドロビンで選ぶ（そして選ばれたプランを 1 ステップずつ実行する）というものだけである．しかし，実世界で複数の目的を整合的に達成していくにあたっては，より柔軟な熟考ルーチンを必要とすることがある．たとえば，2 つのプランが競合しており同時には実行できない場合に，一方を優先して実行するといった判断を要する場合があり，本論文での実験にもその例が現れる．こうした処理は，プランの選択に関するメタプランとして，エージェントの信念内に書ける必要がある．

そこでわれわれは，Jason の熟考処理部を拡張し，熟考ルーチンをプラン記述と同様，Prolog 風のルール形式によるメタプランとして記述できるようにした．

具体的には，Jason が熟考を行う際に，`delib` という 3 引数述語が呼ばれるようにした．その第 1 引数には，現在実行中で選択の候補となっている意図のリスト，第 2 引数には，現在存在するが保留（suspend）状態となっている意図のリストが入る．ユーザは，この述語の第 3 引数に，選択された意図 1 つが代入されるようにこの述語を定義することで，熟考ルーチンを記述できる．この述語が失敗すれば，デフォルトどおりラウンドロビンで意図が選択される．

たとえば，特定の 2 つのプラン A と B について，A と B が選択の候補となっていれば A を優先するよう `delib` を記述することにより，A の実行中は B の実行を抑制させることができる．プラン A 内にプラン B を一時停止させるよう記述するのとは異なり，プランの選択に関する記述をプランそのものの記述と混在させるのではなく，メタレベルの記述として分離することができる．

3.5　BDI ロボットの目的と目標

2.3.3 項で述べた意図の再考慮や 2.3.4 項で述べたコミットメント戦略などの振舞いを実現するために，BDI ロボットの目的および目標を記述する．各目標の最後に書かれている「(プラン名)」は，それぞれの目標を達成するための，3.6 節で記述されているプランの名前である．

まず，Explorer の目的は以下のように 3 つあり，それぞれの目標とともに列挙する．「•」で始まるものは目的，「–」で始まるものはその目的を達成するための目標である．

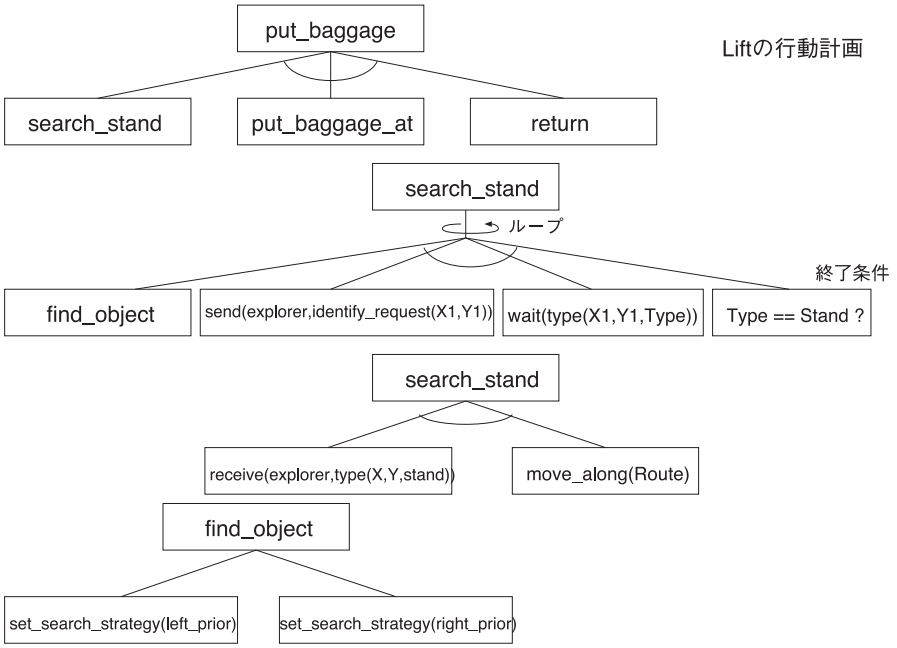

図 8　Lift の行動計画図
Fig. 8　Action plans of Lift.

- 盤上を掃除して回ること
  - 電池が消耗するまでの間，盤上を掃除して回る[*2]（プラン (a)）.
  - 電池が消耗してきたとき，掃除のタスクは終了され初期位置に戻る（プラン (d)）.
- 荷物を置く台を発見したら Lift に伝えること
  - 掃除をしながら盤上を移動している途上において，荷物を置く台を発見した場合は，Lift に台を発見したことを伝える（プラン (c)）.
- 荷物を置く台かどうかの識別を行うこと
  - Lift に台の識別を依頼された場合は最優先して台の識別を行う（プラン (b)）.

次に，Lift の目的は 1 つである．「∗」で始まるものは副目標である．

- 荷物を置く台に 1 個荷物を置くこと
  - 荷物を台に置く目的を達成するために自身の持つ荷物を荷物を置く台に載せる（プラン (e)）.
    - ∗ まず荷物を置く台の探索を行う.
    - ∗ 台の発見とともに Explorer に台の識別を依頼.
    - ∗ 台に荷物をおくことができたならば，タスク終了と見なし，初期位置に戻る.

    この目標を達成するために 2.3.4 項で述べた失敗した場合の処理を考え，意図の保持によって柔軟に対処することが可能であることを示すために，次のような失敗にも対処可能としている.
    - ∗ 環境の変化による失敗
      - (1) Explorer が先に荷物を置く台を発見した場合に，Lift にそれを知らせる（プラン (c)，プラン (g)）.
      - (2) Lift がその台の位置に向かう間に台がなくなってしまうことにより（環境の変化），荷物を置くという目標の達成に失敗する（プラン (g) における失敗）.
  - Explorer から荷物を置く台の発見を知らされた場合には，その場所に向かい荷物を置く（プラン (g)）.
  - Explorer の電池が消耗してきたとき，Explorer はすべてのタスクを放棄するため，自身の依頼も受け入れてもらえなくなることから，Lift は荷物を置くタスクを達成不能と判断し，荷物を置くことをあきらめる（プラン (i)）.
  - Lift が現状の探索の範囲に偏りがあると判断したとき，探索の戦略を変更する（プラン (h)）.

3.6　BDI ロボットの目標を達成するプラン

上に述べた BDI ロボットの目標を達成するプランを以下に擬似コードで記す[*3]．Lift のプランについては図にも示す（図 8，プラン (i) を除く）.

- Explorer のプラン

(a)　掃除しながら盤上を動くプラン

```
clean_around
:  true
<- look_around; // 周囲のマスの知覚
```

[*2] 実験では実際は単に動き回っているだけだが，モップか何かを持って清掃して回っているという想定である.

[*3] Jason で用いられる，イベントの追加などを表す記号（「+!」など）は省いた.

```
    move;
    // 動けるマスを1つ選んでそこへ移動
    clean_around. // 再帰
```

(b) 台の識別をするプラン（(a) より優先）

```
identify_rack
:  receive(lift, identify_request(X,Y))
   // Liftから識別依頼が来たら
<- wait(current_pos(X1,Y1));
   // 自分の現在位置の取得
   calculate_route_to_next
     (X1,Y1,X,Y,Route);
   // (X,Y)の隣までの経路の計算
   move_along(Route); // 移動
   judge(X,Y,Type); // 識別
   send(lift, type(X,Y,Type)).
   // Liftに結果を伝える
```

(c) 荷物を置く台を発見したことを Lift に伝えるプラン

```
notify_of_stand
:  perceive_touch(X,Y)
   // タッチセンサが感知
<- if(percept(X,Y)){
     // 超音波センサで知覚し直す
     // 知覚があればそれは障害物
   } else {
     // 何も知覚できなければ荷物を置く
     // 台と判断
     send(lift, type(X,Y,stand)).
     // Liftに結果を伝える
   }
```

(d) 電池が消耗してきて意図を放棄するプラン

```
abandon
:  timeout // 電池消耗により時間切れ
<- drop_all_desires;
   // すべての目標を放棄
   send(lift, abort);
   // Liftに放棄したことを告げる
   return. // 初期位置に戻る
```

- Lift のプラン

(e) 荷物を台に置くプラン

```
put_baggage
:  true
<- search_stand(X,Y);
   // 荷物を置く台を見つける
   if(percept(X,Y)){
   // 念のため超音波センサで知覚し直す
     put_baggage_at(X,Y); // 荷物を置く
     return; // 初期位置に戻る
   } else {
     fail; // 失敗
   }
```

(f) 荷物を置く台を見つけるプラン（自分で見つける場合）

```
search_stand(X,Y)
:  not already_put_baggage
   // 荷物をまだ置いていない
<- find_object(X1,Y1); // 台を探す
   send(explorer,
        identify_request(X1,Y1));
   // explorerに識別を依頼
   wait(type(X1,Y1,Type));
   if(Type != stand){
     // 荷物を置く台ではなかった
     search_stand(X,Y); // 再帰
   } else {
     X=X1; Y=Y1;
     // 荷物を置く台があった
   }
```

(g) 荷物を置く台を見つけるプラン（Explorer に教えてもらう場合．(f) より優先）

```
search_stand(X,Y)
:  receive(explorer, type(X,Y,stand))
<- wait(current_pos(X1,Y1));
   // 自分の現在位置の取得
   calculate_route_to_next
     (X1,Y1,X,Y,Route);
   // (X,Y)の隣までの経路の計算
   move_along(Route). // 移動
```

(h) マップの左半分優先で台を探すプラン（(f) のサブプラン．右半分優先の同様のプラン (h') もある）

```
find_object(X,Y)
:  more_unseen_places_on_lefthalf
   // 未見の地が左半分に多い
<- set_search_strategy(left_prior);
   // 探索方針変更;
   // look_aroundやmoveに影響を及ぼす
   look_around; // 周囲のマスの知覚
   if(found_object(X,Y)){ // 台があった
     // そのまま終了
   } else {
     move;
     // 動けるマスを1つ選んでそこへ移動
     find_object(X,Y); // 再帰
   }
}
```

(i) Explorer の電池が消耗してきたときに Lift がとるプラン

```
give_up
```

```
    :  receive(explorer, abort)
  <- wait(current_pos(X,Y));
     // 自分の現在位置の取得
     drop_all_desires;
     // すべての目標を放棄
     return; // 初期位置に戻る
```

プラン (b) を (a) よりも優先する働きは，3.4.3 項に述べた拡張熟考ルーチンで実現される．

これらの中で用いられている `move` や `look_forward` などのサブプランは，最終的には 3.2 節で述べた基本行為（と Jason の内部アクション）を用いて実現されている[*4]．

3.7　BDI ロボットの行動

3.6 節に述べたプランのもとでの，各ロボットの振舞いを以下に述べる．特定の初期条件のもとでの振舞いの詳細については，4 章の実験結果において述べる．

Explorer は初期目標として `clean_around` を与えられ，プラン (a) によって掃除をしながら盤上を動き回る．ただし，Lift から台の識別の依頼が来た場合は，プラン (b) が優先されて (a) が保留され，台の識別に向かう．

動き回っている最中に台を発見した場合，プラン (c) によりそれを識別し，荷物を置く台であった場合は Lift に知らせる．このプランの終了後は，元のプラン (a) が継続され引き続き動き回る．

電池消耗により時間切れとなった場合，目標 `abandon` が発生してプラン (d) によりすべての目標を放棄する．このときリフトにその旨を伝える．

Lift は，初期目標 `put_baggage` を与えられ，プラン (e) を実行開始．まず副目標 `search_stand(X,Y)` により，プラン (f) で荷物を置く台を見つけようとする．これに成功すれば，`(X,Y)` に実際に荷物を置く台があることを確認のうえ，荷物を置いて (e) は終了する．プラン (e) の途上で，Explorer が電池切れによる目標放棄を伝えてきた場合，Lift は台の識別の手段を失うので，プラン (i) によって自らの目標を放棄する．

プラン (f) により荷物を置く台を見つけるには，まず台を探してから Explorer に識別を依頼する．台を探す目標 `find_object(X1,Y1)` のためのプランは (h) と (h') の 2 つあり，その時点での状況（未見の地が盤の左半分と右半分のどちらに多いか）に応じていずれかを選ぶ．また，プラン (f) の途中で，Explorer が台の発見を伝えてきた場合，`search_stand(X,Y)` を達成する意図をプラン (g) に切り替え，こちらで達成しようとする．

[*4] たとえば move は，基本行為の「1 マス前進」と，Jason の内部アクションである他エージェントとの信念のやりとりなどを用いて実現されており，自分が進もうとしているマスの占有を他のエージェントに伝えることで，エージェントどうしの衝突を防ぐように作られている．

4.　実世界の多様性への適応実験

3.3 節で述べたような設定のもと，実際にわれわれのロボットが実世界の多様性に適応できることを示すため，以下に述べるような 5 種類の配置および状況のもとで実験を行った．

I) 図 **10** の配置
II) 図 **11** の配置
III) 図 **12** の配置
IV) 図 10 の配置だが，Explorer が Lift に台の発見を伝えた後に台が移動してしまうという状況（3.5 節で述べた環境の変化が起きるケース）
V) 図 **13** の配置

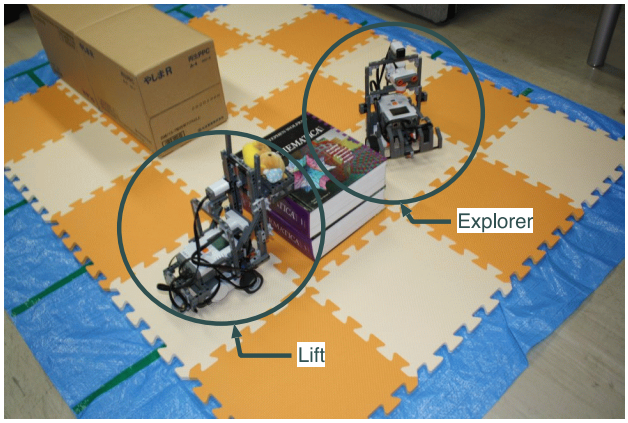

図 **9**　実験の様子
Fig. 9　Look of our experiment.

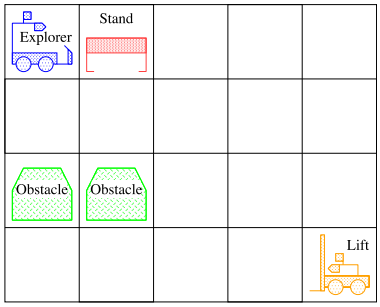

図 **10**　Explorer による発見の場合の配置図
Fig. 10　Case where Explorer finds stand.

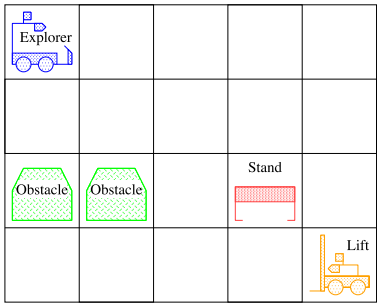

図 **11**　Lift による発見の場合の配置図
Fig. 11　Case where Lift finds stand.

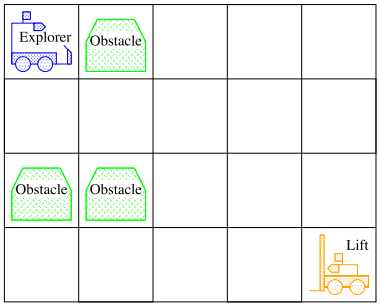

図 12　すべて障害物であるの場合の配置図
Fig. 12　Case where there are only obstacles.

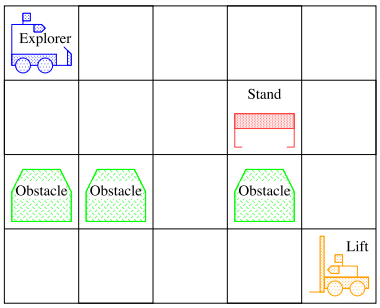

図 13　例外処理が二重に起こる場合の配置図
Fig. 13　Case where exception handling doubly occurs.

各実験の目的および結果を**表 1** でまとめる．**図 9** は II) の配置（図 11）からスタートした場合の実験の 1 場面（Lift が台に荷物を置いているところ）である．

5.　考察

5.1　BDI ロボットの柔軟性

2 章で述べたように，BDI ロボットは 2.2 節の (1), (2), (5) に対処しつつ，多様な環境変化に柔軟に対応して複数の目的を整合的に達成できる．4 章の実験でも，Explorer は II) で複数の目的に対する意図を並立させており，また Lift は IV) で荷物を置く台がなくなっているケースに対応して台を探し直したり，I) で状況の変化に直面して目標達成の手段を切り替えたりすることができた．

この柔軟さには，3.4.3 項に述べたようなプランとメタプランの分離によって，それらが互いに独立していることも寄与している．

この独立性は，実行時の動的環境変化への対処という意味での柔軟さだけでなく，エージェントの設計の柔軟な変更を可能とする利点ももたらす．たとえば，3.6 節のプラン (b) を (a) より優先させている熟考ルーチンを変更し，「Explorer が Lift から “何か奢ってあげる” 旨伝えられているという信念を得た場合のみ (b) を優先し，さもなければ (a) の方を優先する」よう書き直すこともでき，その場合 Explorer は，何か奢るぞと言われていない限り Lift の台識別依頼を無視する．この変更にあたり，プラン (a) や (b) そのものの変更の必要はない．

また，プラン選択がプランから分離されていることから，プランどうしも互いに干渉しないように記述できるため，たとえばプラン (a) で盤面探索アルゴリズムのみ変更することもでき，この際に (b) との干渉を気にする必要はない．これに対し，(b) からの割込みを処理するルーチンが (a) の中に書かれるような実現法だと，(a) 内の変更においても干渉に注意する必要が出る．

このように，エージェントの信念（に含まれているプランやメタプラン）を局所的に変更するだけで，柔軟な振舞いの変更ができる点も，BDI モデルの利点の 1 つである．

5.2　意図を持つことの利点

BDI モデルの特徴の 1 つは，意図という心的状態を明示的に持つことであり，本論文のような実世界ロボットにおけるその意義を 2.3 節で論じた．意図を持つことには他にも多くの利点があり，それをここで論じる．

5.2.1　タスク管理システムとしての BDI モデル

2.3.1 項で述べたように，BDI モデルにおいては，意図を単位としたタスク管理に相当することが BDI のフレームワーク自体で行われる．したがって，他のプランへの切替えをプラン自体が管理したり記述に含めたりすることは原則的になくて済む．

たとえば，表 1 の実験 II) でプラン (b) の実行の間プラン (a) を保留する（そして (b) が終了すれば (a) を再開する）処理は，(b) に「(a) を一時停止する」のように書かれることはなく，3.4.2 項および 3.4.3 項に述べたような，熟考という BDI フレームワークの持つ働きで実現される[*5]．また，実験 IV) のケースでも，台がなくなっていて荷物を置くことに失敗した後，Lift が再度探索プランを開始するのは，プランにやり直せと書かれているからではなく，目標を達成する意図が残っているために目的–手段推論が再度行われることによる．

ハンドコーディング手法では，プラン自体のコードにタスクスイッチの処理も書くことになり（OS が存在しない環境でのプログラミングにあたるだろう），小規模な例ではよくてもタスク数が増えると破綻する．また，要求分析のような手法では状態遷移を構築することが多いが，状態数が増えると，たとえばすべての状態に割込みのためのコードを書くようなことが必要になればやはり破綻する．これに対し，タスク自体とその管理の自然な分離を可能とする BDI モデルは，多様性への対処には非常に有効である．

5.2.2　意図を用いた相互理解

BDI モデルでの意図概念は，単なるジョブスケジューラ

[*5] ただしある意図が他の意図の続行に能動的に関与することは可能ではあり（プロセスに例えれば UNIX における kill のように），今回の例でもすべての目標（とそれから発生している意図）を破棄する「`drop_all_desire`」が用いられているが，これを用いずに信念の変更による意図の放棄として同等のことを行うこともでき，この方がコミットメント戦略の本来の姿にも近い．

表 1　実験の結果の表
Table 1　Results of our experiments.

実験の目的	初期配置	観測されたロボットの行動（括弧内は 3.6 節の該当プランの番号）	評　　価
意図の再考慮（2.3.3 項）が行えること	I) 図 10	Explorer は掃除を開始 (a) したが，(1, 0) に荷物を置く台を発見し，Lift に伝えて (c) 掃除を再開．Lift は荷物を台に置くプラン (e) を開始していたが，台を見つける副目標のための意図 (f) を中断し，Explorer から伝えられた台のところへ行く (g) ことで同目標を達成．(e) の残りを実行して荷物を置き，初期位置に戻った	Lift が予期と異なる状況に直面し，目標を達成するための意図を切り替えることができたことで，意図の再考慮が行えることを示した
情報の不完全さ（2.2 節の (2)）や時間がかかること（同 (1)）への対応，および複数の意図の並行（同 (5)）	II) 図 11	Lift は (3, 2) の台を発見 (f) し，Explorer に識別を依頼．Explorer は掃除のタスク (a) を中断し，プラン (b) によって台の識別を行った．その結果，荷物を置く台であることが分かり，それを Lift に伝えて (b) は完了，(a) を再開した．Lift は荷物を置く台だったことを知って (f) を完了，台を見つける副目標を達成．以後は I) と同じ	Lift は，台を見つけても識別はできない情報の不完全さ，および識別情報が来るまで時間がかかる事象に対処した．また，Explorer は掃除の意図を保持したまま台の識別の意図も達成しており，こちらは複数意図の並行の実現を示している
コミットメント戦略（2.3.4 項の (b)，(c)）の実現	III) 図 12	Explorer は (a) の最中，台がないため台を見つけられぬうちに制限時間がきた．よって (d) により全タスクを放棄し Lift に通告，初期位置に戻って終了．Lift は通告を受け，(i) により全タスクを放棄し初期位置に戻り終了	Explorer は探索の達成を放棄して終了したため Open-minded，Lift は荷物を置く目標の達成不能（台の識別ができないため）により終了したため Single-minded の各コミットメント戦略を実現している
意図の継続（2.3.4 項）による失敗への対処が行えること	IV) 図 10 の配置で，Explorer が Lift に台の発見を伝えた後に台が移動	I) と同様に，Lift は Explorer から伝えられた台のところへ行ったが，台がなくなっていて (e) が失敗．しかし，荷物を台に置く目標は残っているので，この目標を達成するために再度プラン (e) が選ばれて意図され，台を探し直し目的を達成した	目標達成の手段が失敗しても，目標が残っていれば新たな手段を選ぶことで意図を持続できることを示した．また，新たに (f) を実行する際，`find_object` の達成手段はそのときの状況によって変わる（左/右のどちらを先に探すかはそのときになってから選ぶ）ので，この部分は 2.3.2 項で述べた副目標の達成手段の推論の先送り（部分的なプランニング）の例でもある
現状での問題点	V) 図 13	Lift が (3, 2) の台の識別を依頼し，Explorer は (b) が (a) を抑止した格好で識別に向かうが，その途中で (3, 1) の台を見つけて (c) が割り込む．(b) と (c) の間には排他関係が書かれていなかったため両者が競合し，正常に行動できなくなった	割込みの入れ子のような場合における割込みどうしの競合には（通常の開発同様）慎重な検討が必要である

のみに相当するものではない．それが何かをなそうとする心的状態であることにより，人間との相互理解に寄与するものでもある．

たとえば，看護ロボットを考えてみよう．われわれは，看護ロボットが，将来，何をしようとしているのか，また，いま何をしていてその理由は何であるのかなど看護ロボットの心と行為を理解できないと何を要求していいのか分からない．逆に看護ロボットもわれわれの心と行為を理解できないと，十分なサービスはできない．

人間は，主体・他者の心と行為に関する理解は，意図概念を用いて行ってきた．そこで，人間とエージェント間の相互理解のために新たなプロトコルを創発するよりは，エージェントに意図概念を導入した方が合理的である．

本論文ではこうしたことは扱っていないが，実世界で人との親和性の高いロボットを実現する手段としての意図概念の利用は今後検討すべきであろう．

5.3　実世界ロボットでの実験の意義と現状の問題点

ここでは，シミュレーション実験と比較しての，実際に実世界ロボットで実験を行うことの意義について，および現状での問題点について述べる．

2.2 節で述べたように，実世界のロボットは，仮想世界

と比べて，知覚に時間がかかったり不完全であったりする，固有のダイナミックスでつねに変化するなど，さまざまな点で異なる．よって，実世界のロボットの振舞いをシミュレーションのみで開発したり検証したりしようとすることには無理がある．

本実験でのその一例として，3.6 節で述べたプラン (a) のサブプラン move の実装の不適切さに起因するプラン (b) の誤動作があげられる．当初の実装では，move は 1 マス動いた直後に自分の現在位置に関する信念 `current_pos(X,Y)` を更新していた．しかし，これではプラン (a) の move での移動に時間がかかっている間に Lift から台の識別依頼が来た場合，プラン (b) が優先され，自分の現在位置を取得ののち台へのルートを計算して台の位置までの移動を始めてしまうが，「1 マス前進」という基本行為には割り込めないため，台の位置への移動は移動直後に始まる．しかし，台へのルートの計算の起点となった自分の現在位置は移動前のものであるため，正しく移動できないという事態が起こった．しかも，この問題は移動に時間を要しないシミュレーション環境では発生しなかった[*6]．

ロボット実験によってこの問題の存在を把握した結果，move の実装を，自分の現在位置の信念を削除 → 1 マス前進（基本行為）→ 新たな現在位置の信念を得る，と変更することで，プラン (b) は現在位置の取得を移動直後まで待つことになり，この問題は解消した．

このようなシミュレーションと実世界の概念のずれに起因する障害は，シミュレーションだけでは発見が困難であり（環境が変わると誤動作するプログラムのデバッグの困難さを想像すればよいだろう），実世界でのロボット構築の実証実験の意義の一端はこうしたところにあるといえる．

われわれはこの点について，形式化の観点から 5.5 節で再度論じる．

次に問題点について述べる．われわれは本論文で，2.2 節で述べた実世界ロボットにとっての問題点のうち (1), (2), (5) への対処について述べた．しかしそれ以外，たとえば (3) などへの対処は難しく，特に本実験で用いたような安価なロボットではきわめて困難である．本実験での具体例では，室内の電化製品などの磁場の影響で実験設備内での磁場が歪んで，コンパスセンサに誤差が生じ，正確な方向制御ができないことが起きた．また，(4) についても本実験ではモデルの簡単化のためグリッドワールドを用いたので非離散的な世界記述には対応できていないが，ロボットの行動をより柔軟にするという点からもこれは将来の課題である．

[*6] 「シミュレーションでも移動に時間がかかるようにする」というアイデアは，この問題を把握していないと必ずしも当然なものとしては出てこない．現実に起こることのすべてをシミュレーションに取り込むことは不可能であり，適切なもののみシミュレーションモデルに取り込むことも，その適切さの予測があらかじめつきにくいため難しい．これは，実世界の複雑さに起因する本質的な困難といえる．

5.4　関連研究

5.4.1　他の BDI によるロボットの研究

これまで BDI モデルの応用はソフトウェアエージェントが多く，実世界ロボットへの応用はあまり見られない．その中で，文献 [10] と文献 [7] はどちらも，BDI によるマルチエージェントを小型のロボットに搭載し，実際に実験を行った例である．

文献 [10] は，複数のロボットがグリッドワールド上を探索してオブジェクトを収集するもので，ロボット間のコミュニケーションに重点をおいており，これによってロボットどうしの衝突を避けたり，入札システムを介して，同じオブジェクトを複数のロボットが拾いに行くことを避けたりするなどを実現している．また文献 [7] は，複数のロボットがコンベア上の小包を指定位置まで運ぶというもので，ロボットはコンベアベルトに相当するものとフォークリフトに相当するものの 2 種があって，前者は小包をパレット上に置くこと，後者はそれを指定された位置に運ぶことを目標としており，これらの協力でシステムを構成している．

しかし，これらは複数の意図の並行や，状況の変化に応じて意図を柔軟に変更するといったことを扱っていないため，本論文で述べたような BDI モデルの利点について明らかにしているものではない．また，いずれにも中央集中的な管理機構が存在するため，本論文のように独立したロボットが必要なときのみコミュニケーションをとって協調するというものと比べ，エージェントの自律性もやや損なわれる．

5.4.2　他の適応的なエージェントプラットフォーム

状況の変化に適応して適切な振舞いをするシステムとしては，近年，自己適応システム [8] の提案もある．これは，環境との相互作用を行う最下層の部品制御層，同層からの環境変化の通知を受けて振舞いの変更を担う中間層の変更管理層，下位からの要請で目標達成のためのプランニングを担う最上位の目標管理層の 3 層からなるアーキテクチャモデルである．

自己適応システムにおいては，状況の変化を最下層のセンサが感知すれば，中間層がそれに呼応して，下位の部品の変更や再構成を行うことで動的に振舞いを変える．しかし，BDI モデルでは外界の変化によらず，自分の心的状態の変化によっても振舞いの変化を起こすことがある．たとえば Open-minded コミットメント戦略では，まだ達成可能な意図を，（自分が飽きたなどの）理由で中断することがある（本論文の例でも，Explorer はタイムアウトという自己都合で探索の意図を中止している）．すなわち，BDI モデルの方が振舞いの変化のより一般的なモデルを提供している．さらに，意図を保持したまましばらく保留して他

表 2　BDI logic のオペレータの直感的意味（抜粋）
Table 2　Intuitive meaning of BDI logic operators (excerpt).

$\wedge$	かつ	$\mathsf{BEL}\,\phi$	ϕ を信念に持つ	$\mathsf{AX}\,\phi$	次の時刻に ϕ が成り立つ
$\supset$	ならば	$\mathsf{INT}\,\phi$	ϕ を意図する	$\mathsf{A}(\phi\,\mathsf{U}\,\psi)$	次に ψ が成り立つ直前まで ϕ が成り立つ
$\neg$	～でない			$\mathsf{A}(\phi\,\mathsf{N}\,\psi)$	次に ψ が成り立ったとき ϕ が成り立つ

の優先度の高いタスクを先に片付けるようなことは，振舞いの変化という解釈では実現できない．振舞いの解釈と変化するなら，たとえば本論文の Explorer のプラン (a) から (b) への切替えは，同じタスクの別々のフェーズのようにとらえることになるが，本来まったく別な 2 つの仕事をそのようにとらえることは合理性を欠く．

また，自己適応システムの実現は研究途上の段階で，実装に関する提案はまだ十分とはいえない．実現の 1 つとして，目標指向の要求分析法である KAOS [5] によって問題を記述し，エージェント実装プラットフォーム JADE [16] によるコンポーネント記述への変換によって自己適応システムを構築する試み [21], [22] がある．しかし，KAOS での分析を JADE での実装のレベルにまで変換することは自動的には難しいという課題をかかえている．これに対し，BDI モデルを基盤にしたわれわれの手法では，多様性に柔軟に対応した目標指向の振舞いを自然に記述・実現できている．

状況に応じた柔軟な行動をとれる反応型エージェントを実現する例として，Bonasso らの 3T アーキテクチャ [1] もあげられる．これは上述の自己適応システムとほぼ類似した構成を持つ 3 層アーキテクチャであるが，最下層は反射的行動に特化しており，単一ドメインの問題に特化したシステムである．しかし，複数のタスクを並行に扱うような機構は組み込まれていない点がわれわれのものと異なる．

センサと振舞いモジュールの階層構造からなるロボットアーキテクチャとしては，Brooks のサブサンプションアーキテクチャ [4] も知られる．これは，複数の目標を達成するために，多数のセンサに反応し，頑健かつ漸進的に拡張可能なロボットを容易に設計できる方法論であり，以下の特徴を持つ．

(1) タスクを階層的なレイヤに分け，下位のレイヤが構築されれば，それを変更することなく，上位のレイヤを構築できる．
(2) 内部処理をほとんど必要としないような単純なセンサと行動の結合によって，環境と実時間での相互作用ができる．
(3) 知能は中央集権化されたものではなく，複数の緩やかに結合されたプロセスから知的振舞いが創発されると考える．

しかし，本論文で扱っている，Lift が荷物を台に載せるタスクのように，多数の副目標の達成や他者とのコミュニケーションを段階的に要するようなプロセス列を，センサとアクチュエータの組合せのみで環境との相互作用から創発させることは現実的でなく，そのような応用にはわれわれのもののようにプランを明示的に扱える手法の方が向いているといえよう．

5.5　BDI logic による形式化

他のエージェントフレームワークと比較しての BDI モデルの利点の 1 つは，BDI logic [14] という時相論理体系をモデルの構成要素として持ち，これを用いて信念・欲求・意図といった心的状態やその時間的変化を形式的に記述したり論じたりできることである．他のフレームワークでも，要求や仕様の記述時点で形式論理を使えるものはあるが，その範囲にとどまらない一般的な振舞いの記述を行える形式論理体系を持つものは見当らない．

たとえば 5.3 節で述べた問題において，訂正後の move の性質は BDI logic では

$$\begin{aligned}&\mathsf{BEL}\ \mathit{current\_pos}(x,y) \wedge \mathsf{INT}\ \mathit{move}(x,y,x_1,y_1) \supset \\ &\mathsf{A}((\forall w,v \neg\, \mathsf{BEL}\ \mathit{current\_pos}(w,v)) \\ &\mathsf{U}\ \mathsf{BEL}\ \mathit{finish\_move}(x,y,x_1,y_1)) \wedge \\ &\mathsf{A}(\mathsf{BEL}\ \mathit{current\_pos}(x_1,y_1) \\ &\quad \mathsf{N}\ \mathsf{BEL}\ \mathit{finish\_move}(x,y,x_1,y_1)) \qquad (1)\end{aligned}$$

のように書ける（各オペレータの大まかな意味は**表 2** を参照）．これは「*move* を意図するとその終了を信じるまでは現在位置に関する信念はなくなり，終了を信じたときに現在位置の信念が更新される」を表す．一方，プラン (b) の性質（*calculate_route* を行うところまで）は

$$\begin{aligned}&\mathsf{BEL}\ \mathit{identify\_requested}(x,y) \supset \\ &\quad \mathsf{A}(\phi\ \mathsf{N}\ \exists w,v\ \mathsf{BEL}\ \mathit{current\_pos}(w,v)) \\ &\qquad \text{ただし}\ \phi = (\mathsf{BEL}\ \mathit{current\_pos}(x_1,y_1) \supset \\ &\qquad \mathit{calculate\_route}(x_1,y_1,x,y,\mathit{Route})) \qquad (2)\end{aligned}$$

と書ける．いま，システムがこの両者を成り立たせるよう動いているとし，$\mathsf{INT}\ \mathit{move}(x,y,x_1,y_1)$ が成り立って以後 $\mathsf{BEL}\ \mathit{finish\_move}(x,y,x_1,y_1)$ が成り立つまでの間に Lift からの依頼が来た（$\mathsf{BEL}\ \mathit{identify\_requested}(x,y)$ が成り立った）としよう．すると，それ以降 $\exists w,v\ \mathsf{BEL}\ \mathit{current\_pos}(w,v)$ が最初に成り立つのは移動が終わったときであり，このとき *calculate_route* の引数の x_1, y_1 は移動後の現在位置を反映したものである．すなわち，正しい起点からの *calculate_route* が行われる．

一方，不適切な方の move の実装の性質は

$$\mathsf{BEL}\ \mathit{current\_pos}(x,y) \wedge \mathsf{INT}\ \mathit{move}(x,y,x_1,y_1) \supset \mathsf{A}(\mathsf{BEL}\ \mathit{current\_pos}(x,y)\,\mathsf{U}\,\mathsf{BEL}\ \mathit{finish\_move}(x,y,x_1,y_1)) \wedge \mathsf{A}(\mathsf{BEL}\ \mathit{current\_pos}(x_1,y_1)\,\mathsf{N}\,\mathsf{BEL}\ \mathit{finish\_move}(x,y,x_1,y_1)) \tag{3}$$

のように書け，これは「*move* を意図した場合，その終了を信じるまでは現在位置の信念は更新されず保たれ，終了を信じたときに現在位置の信念が更新される」を意味する．こちらの場合，上記と同じタイミングで Lift から依頼が来ても，そのときただちに $\exists w,v\ \mathsf{BEL}\ \mathit{current\_pos}(w,v)$ が成り立つため，*calculate_route* の引数の x_1，y_1 は移動前の現在位置を反映したものであり，すなわち，誤った起点からの *calculate_route* が行われてしまう．

しかし，シミュレーション環境では次の式

$$\mathsf{INT}\ \mathit{move}(x,y,x_1,y_1) \supset \mathsf{AX}\,\mathsf{BEL}\ \mathit{finish\_move}(x,y,x_1,y_1) \tag{4}$$

が成り立つことがある．これは「*move* を意図すると次の時刻には移動は終了している」を意味する．これが成り立つ限り，式 (3) の場合でも上のような問題は起こらない．

このように，システムの振舞い全般に関する議論を形式的に行える手段が最初から用意されている点は，他のフレームワークには見られない長所といえる．

われわれは，強化学習やプランニングなど BDI の外部の行為選択機構との結合を扱えるような BDI logic の拡張を提案しており [19], [20]，今後，ロボットの柔軟な行為選択の設計に有用であることが期待できる．

6. おわりに

本論文では，BDI モデルによる実世界ロボットの実現の有効性について，実験を通じて示した．BDI モデルは，意図の保持と破棄という機構によって，複雑に変化する多様性に富む実世界において，目標を達成する手段を柔軟に切り替えたり複数の目的を並行して保持したりできる．また，意図の管理が，タスクスケジュールに相当し，優先順位や整合性を維持しつつタスク管理を行えること，タスク管理をタスク（プラン）自体と自然に分離できることも，大きな利点である．

今後の課題としては，ロボットの基本行為の精度に関する問題を解消するため，ロボティクス分野の成果によるロボット制御技術との結合による実験の展開や，また，プランニングなどを必要とするより大きな問題へ本論文の手法を適用し，より実用的な課題への応用を図ることなどが考えられる．

さらに，本論文では入手の容易さのため小型ロボットを用いたが，より大規模なロボットへの応用も将来的視野としては考えられる．ただしその場合，実世界で人間など他者と混在した場合の安全性の議論も必要となろう[*7]．

*7 これに関しては，5.2.2 項で論じた相互理解の実現が，お互いの行動の予期を通じて危険回避の助けになる可能性も考えられる．

参考文献

[1] Bonasso, R.P., Kortenkamp, D., Miller, D.P. and Slack, M.: Experiences with an Architecture for Intelligent, Reactive Agents, *Journal of Experimental and Theoretical Artificial Intelligence*, Vol.9, pp.237–256 (1995).

[2] Bordini, R.H., Hübner, J.F. and Wooldridge, M.: *Programming Multi-Agent Systems in AgentSpeak using Jason*, John Wiley & Sons (2007).

[3] Bratman, M.E.: *Intention, Plans, and Practical Reason*, Harvard University Press (1987).

[4] Brooks, R.A.: A Robust Layered Control System for a Mobile Robot, *IEEE Journal of Robotics and Automation*, Vol.RA-2, No.1, pp.14–23 (1986).

[5] Dardenne, A., van Lamsweerde, A. and Fickas, S.: Goal-directed requirements acquisition, *Science of Computer Programming*, Vol.20, No.1-2, pp.3–50 (1993).

[6] Frischknecht, C. and Other, T.: LEGO Mindstorms NXT: Welcome to the NeXT generation (2006), available from ⟨http://www.tik.ee.ethz.ch/tik/education/lectures/PPS/mindstorms/sa_nxt/index.php⟩.

[7] Jensen, L.K., Kristensen, B.B. and Demazeau, Y.: FLIP: Prototyping multi-robot systems, *Robotics and Autonomous Systems*, Vol.53, No.3-4, pp.230–243 (2005).

[8] Kramer, J. and Magee, J.: Self-Managed Systems: An Architectual Challenge, *Proc. FOSE '07*, pp.259–268 (2007).

[9] Lau, D.P.: NXT_Python (2007), available from ⟨http://home.comcast.net/~dplau/nxt_python/index.html⟩.

[10] Lemke, A., Laidlaw, J. and Zilmer-Pedersen, L.: Developing Multi-Agent Lego Robotics (2007), available from ⟨http://www.imm.dtu.dk/upload/institutter/imm/cse%20laboratories/lemke_ea.pdf⟩.

[11] Parker, D.: nxtprograms.com, available from ⟨http://www.nxtprograms.com/index.html⟩.

[12] Pfeifer, R. and Bongard, J.C.: *How the Body Shapes the Way We Think*, The MIT Press (2006).

[13] Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language, *Proc. MAAMAW-96*, LNAI, Vol.1038, pp.42–55, Springer-Verlag (1996).

[14] Rao, A.S. and Georgeff, M.P.: Modeling Rational Agents within a BDI-Architecture, *Proc. International Conference on Principles of Knowlegde Representation and Reasoning*, pp.473–484 (1991).

[15] Singh, M.P., Rao, A.S. and Georgeff, M.P.: Formal method in DAI: Logic-Based Representation and Reasoning, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pp.331–376, The MIT Press (1999).

[16] Telecom Italia: JADE: Java Agent Development Framework (2010), available from ⟨http://jade.tilab.com⟩.

[17] Toda, M.: *Man, robot, and society: models and speculations*, M. Nijhoff Pub. (1982).

[18] 高田司郎，川戸慎二郎，間瀬健二：利用者と協調して写真を撮る対話エージェント，人工知能学会全国大会，pp.4–5 (2002). 1A1-04.

[19] 新出尚之，藤田　恵，高田司郎：動的プランナと BDI エージェントの結合の形式化について，*Proc. JAWS2010*

(2010). T-1.pdf.
[20] 新出尚之，高田司郎，藤田　恵：拡張 BDI 論理 TOMATO を用いた確率的状態遷移のモデル化とその応用，情報処理学会論文誌　数理モデル化と応用，Vol.4, No.3, pp.59–72 (2011).
[21] 中川博之，大須賀昭彦，本位田真一：ゴール指向要求分析を用いた self-adaptive システムの構築，情報処理学会論文誌，Vol.50, No.10, pp.2500–2513 (2009).
[22] 中川博之，大須賀昭彦，本位田真一：ビヘイビア記述に基づく自己適応システム実装フレームワークの提案，人工知能学会論文誌，Vol.26, No.1, pp.1–12 (2011).

藤田 恵

2006 年奈良女子大学理学部卒業．2009 年同大学院人間文化研究科博士前期課程情報科学専攻修了．同年同大学院人間文化研究科博士後期課程複合現象科学専攻入学，現在に至る．興味を持つ分野は，知的エージェントの構築，知的エージェントによるロボット制御．日本ソフトウェア科学会，人工知能学会，日本ロボット学会各学生会員．

片山 寛子

2008 年奈良女子大学理学部卒業．2010 年同大学院人間文化研究科博士前期課程情報科学専攻入学，現在に至る．

新出 尚之

1986 年京都大学理学部卒業．1988 年同大学院理学研究科数理解析専攻修士課程修了．同年同専攻博士課程中退．同年京都大学情報処理教育センター助手．1992 年奈良女子大学理学部情報科学科講師．2008 年同准教授，現在に至る．博士（情報科学）．興味を持つ分野は，時相論理による証明およびプログラミングシステム，特に自律エージェントシステムの構築．日本ソフトウェア科学会，電子情報通信学会，日本ロボット学会各会員．

高田 司郎　（正会員）

1979 年大阪大学基礎工学部情報工学科卒業．1993 年奈良先端科学技術大学院大学情報科学研究科博士前期課程入学．1999 年同科博士後期課程修了．1979 年 CSK 入社．1993 年けいはんな入社．1999 年 ATR 知能映像通信研究所入所．2001 年 ATR メディア情報科学研究所客員研究員．2002 年福岡工業大学情報工学部管理情報工学科助教授．2003 年より近畿大学理工学部情報学科助教授（現，准教授）．博士（工学）．知能ロボット，コミュニケーション，形式的仕様記述，合理的エージェントに興味を持つ．日本ソフトウェア科学会，人工知能学会，日本ロボット学会，言語処理学会各会員．