テクニカルノート

論理型言語 *MiLog* に基づくインターネットオークション入札支援システム *BiddingBot* の実装技法

服部宏充† 大囿忠親† 新谷虎松†

本論文では，実在のインターネットオークションにおける入札を支援するためのマルチエージェントシステム *BiddingBot* を構築するための，論理型言語の機能を内包するエージェント記述言語 *MiLog* に基づく実装技法について述べる．ここでは，エージェントに，ユーザ，オークションサイト，および他のエージェント間で，適切な情報処理を行うための機能の実装について詳細を述べる．

On Building Technique of Bidding Support System *BiddingBot* Based on *MiLog* Logic Programming

Hiromitsu Hattori,† Tadachika Ozono† and Toramatsu Shintani†

In this paper, we present unique implementation technique based on Logic Programming for *BiddingBot*, which is a multiagent-based bidding support system. We details implementation of appropriate fucntion for interaction among an user, auction sites and agents.

1. はじめに

インターネットオークションの発達にともない，複数オークションへの入札支援に関する研究が活発に行われている．既存の研究で構築されたシステムは，仮想的な入札環境を提供するもので，実在のオークションサイトへの入札は支援しない[3),4)]．筆者らは，実在のオークションサイトを対象とした入札支援が可能なシステム *BiddingBot*[5)] を構築している．本システムは，各サイトに特化した複数のエージェントから成るマルチエージェントシステムであり，論理型言語の機能を内包するエージェント記述言語 *MiLog*[1)] を用いて実装されている．エージェントは，オークションサイトからの情報収集，および入札を行う．ユーザは，Webブラウザを介してエージェントにアクセスし，情報の閲覧や入札に関する指示を与えることができる．システムの構築においては，エージェントに，ユーザ，オークションサイト，および他のエージェント間で，適切に情報を処理する機能の実装が必要である．本論文では，*MiLog* に基づく各機能の実装について詳細を述べる．

† 名古屋工業大学大学院
Graduate School of Engineering, Nagoya Institute of Technology

2. *MiLog* に基づく *BiddingBot* の実装

MiLog は Java を用いて実装された Prolog 言語処理系であり，インターネット上で商取引を行うエージェントの作成を主眼において設計されている．*MiLog* は，制御言語に論理型言語を採用することで，パターンマッチング等のテキスト処理，および探索処理を実現するとともに，インターネット上での通信を最大限に利用するための Web 処理機能（Web サーバ/クライアント機能）と複雑なエージェント間通信を容易化する機能をあわせ持つ．また，データベースを共有する複製（クローン）を生成することで，効率的なマルチタスク処理が可能である．

BiddingBot では，**図 1** に示すように，ユーザ/エージェント間，エージェント/Web サーバ間，および複数エージェント間で通信が行われ，情報が処理される．以下，**表 1** に示す *MiLog* の組み込み述語に基づく，各々の処理機能の実装について示す．

2.1 情報収集/入札機能

オークションサイトからの情報収集，および入札のためには，まず Web サーバにアクセスし，その結果出力される HTML テキストを解析して，必要な情報

表 1 *MiLog* の代表的な組み込み述語
Table 1 Buit-in predicates of *Milog*.

組み込み述語	機能の概要
【HTTP の処理】	
`getURLText(+Url,+Method,+List,-Html)`	`List` を引数とし，`Url` に `Method` で指示するコマンドを送信し，`HTML` テキストを得る
`getURLBin(+Url,+Method,+List,-Bin)`	`List` を引数とし，`Url` に `Method` で指示するコマンドを送信し，バイナリデータを得る
【HTML の解析】	
`parseHTML(+Html,-Parsed)`	テキスト `Html` を解析し，文字列を要素とするリスト `Parsed` を得る
`extract(+Pattern,+List)`	リスト `List` から特定のパターン `Pattern` と一致する部分を取り出す
【エージェント間通信】	
`query(+Agent,+Query)`	エージェント `Agent` に対する `Query` の評価を同期遠隔手続き呼び出しによって行う
`request(+Agent,+Query)`	エージェント `Agent` に対する `Query` の評価を非同期遠隔手続き呼び出しによって行う
`interruptAndQuery(+Agent,+Query)`	エージェント `Agent` の処理への割込み後，`query` を実行
`interruptAndRequest(+Agent,+Qurey)`	エージェント `Agent` の処理への割込み後，`request` を実行
`queryC(+Agent,+Query)`	エージェント `Agent` にクローンを生成させ，`query` を実行させる
`requestC(+Agent,+Query)`	エージェント `Agent` にクローンを生成させ，`request` を実行させる
【Web サービス】	
`doService(+ID,-Html)`	`HTTP` 経由のサーバリクエストに対する処理を行い，`HTML` テキスト `Html` を出力

※+付きのパラメータは，事前に値の束縛が必要であることを表し，− 付きのパラメータは述語の評価後に値が束縛されることを表す

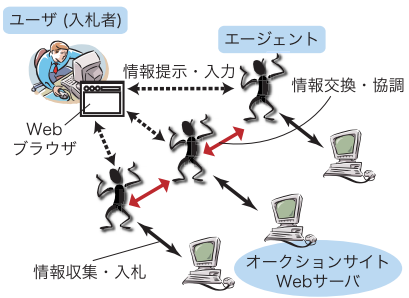

図 1 *BiddingBot* におけるインタラクションの概要
Fig. 1 An overview of interaction in *BiddingBot*.

(e.g., 財の情報，入札結果）の抽出を行う．

(i) Web サーバへのアクセス

Web サーバへのアクセスは，4 引数の述語 getURLText を用いて実装される．具体的な記述形式は以下のとおりである．

`getURLText(Url,Method,[[`p_1`,`v_1`],...,[`p_n`,`v_n`]],Html)`

第 1 引数の変数 `Url`☆で Web ページの URL を指定し，第 2 引数の変数 `Method` で GET と POST のいずれかのメソッドを指定する．また，第 3 引数は，CGI プログラムに対するパラメータ名 p_i と，その値 v_i のペアから成るリストである．出力された HTML テキストは第 4 引数の変数 `Html` に束縛される．たとえば，Yahoo!Auctions において，“palm” をキーワードとして財の検索を行う場合，検索用の CGI プログラムには，パラメータとして`p=palm&auccat=20000&allocate=0us&acc=us` が与えられるが，この場合，以下の記述により，検索結果を表示する HTML テキストを得られる．

```
getURLText('http://search.auctions.shopping.
yahoo.com/search/auc',get,[[p,palm],[auccat,
20000],[allocate,'0us'],[acc,us]],Html)
```

現在のオークションサイトでは，情報の表示処理だけでなく，入札の受け付け処理にも CGI が利用されている．ここでは，入札者のアカウントや入札額がパラメータとして送信される．そのため，入札額等の情報のアップロードが必要である入札機能も，HTML テキストを得る場合と同様の方法で実装可能である．

(ii) テキストの解析と情報抽出

HTML テキストの解析は，2 引数の述語 parseHTML で行うことができる．parseHTML による解析では，テキストをタグと文字列に分解してリストを生成する．さらに，A タグや IMG タグのような属性を持つタグが存在する場合には，タグ要素に関して解析を行い，タグの属性の情報を，属性名と属性値から成るリストに変換する．たとえば，`<HTML><BODY><A HREF="item.html"></BODY></HTML>` の解析結果は `[[html],[body],[a,[href,'item.html']],[/body],[/html]]` となる．

情報の抽出は，2 引数の述語 extract を用いて行われる．extract による処理では，第 1 引数に束縛された情報抽出パターンと，第 2 引数に束縛された解析結果のリストの単一化を行い，必要な情報を変数に束縛することで抽出する．ここで，情報抽出パターンとは，HTML テキストの解析によって得られるリストと同一の構造を持ち，抽出すべき要素の位置を変数で明示したリストである．たとえば，以下の記述では，

☆ 通常の Prolog 言語と同様，*MiLog* においても大文字で始まる文字列は変数を表す．

変数 Url に “http://www.ebay.com” が束縛されることで，URL の抽出が行われる．

```
extract([[a,[href,Url]]],[a,[href,'http:
//www.ebay.com']])
```

単一化の処理自体は，*MiLog* が持つ論理型言語の機能を利用できるため，単一化処理を記述する必要はない．

2.2 エージェント間通信機能

BiddingBot では，効用の高い入札を実現するために，複数のエージェント間で情報交換を行い，協調的に動作するための通信が行われる．通常，入札処理の過程では様々な局面が考えられるため，処理コードの記述は困難となる．具体的には，エージェントの同期処理，および割込み処理を適切に行うための記述が必要となる．

通信を行うエージェント間でタスクに依存関係が存在する場合，同期処理が必要である．たとえば，財 g_1 に入札しているエージェントと財 g_2 に入札しているエージェントがいる状況において，2 つの財を同時に落札した場合にのみ効用が増加する場合，一方のエージェントは，他方のエージェントの落札の可能性を確認した後に，入札額を決定する必要がある．同期処理をともなう通信は述語 query で実行できる．query による通信では，エージェントは他のエージェントに依頼した処理の結果を待ち，結果を受信後，次の処理に移る．一方，同期処理が不要である場合は，述語 request が利用できる．request による通信では，エージェントは他のエージェントに問合せをした後，処理の結果を待つことなく，即座に次の処理に移る．query はコードの記述においては扱いやすいが，返答を待つ時間が無駄になる場合がある．相手からの返答を待つ必要がない場合，余計な待ち時間を必要としない request の方が適切である．

インターネットオークションにおける入札では，特に，締切り間際の時間帯において，エージェントの即応的な動作が必要とされる．しかし，query と request による通信では，通信先のエージェントが他の処理を実行中の場合，処理の終了を待つ必要がある．述語 interruptAndQuery，および interruptAndRequest では，通信先のエージェントの処理に割込みをかけ，query および request による通信処理を即時的に実行できる．具体例として，入札中のエージェントを即座に停止させるコードを示す．ここで，述語 stopAllBidders の引数は，エージェントのリストであり，stop は，エージェントの動作を緊急に停止させる述語である．

```
stopAllBidders([]).
stopAllBidders([Agent|RestAgents]) :-
    interruptAndRequest(Agent,stop),
    stopAllBidders(RestAgents).
```

本例では，interruptAndRequest に基づく，処理結果を待たない非同期型の通信により，すべてのエージェントに即座に入札停止命令を通知できる．

また，エージェントは，同時に複数のタスク処理を要求される場合がある．たとえば，新規の入札のための通信を Web サーバと行っている際に，他のエージェントから受信した場合，Web サーバとの通信と並行して，受信内容の処理を行う．そこで，エージェントがマルチスレッド処理を行うために，述語 queryC と requestC による同期並行処理，および非同期並行処理をともなう通信が利用できる．これらの述語では，複数の処理を並行して行うために通信先のエージェントがクローンを生成し，通信で要求された処理をクローンに実行させる．そのため，割込み型の通信と同様に，通信先のエージェントが他の処理を実行中でも通信が可能であり，かつ処理の実行を妨げることなく，要求した処理を実行できる．

2.3 ユーザインタフェース機能

MiLog によるエージェントは Web サーバの機能を持つため，HTTP アクセスに対する出力処理を記述することで，CGI プログラムとして機能する．本機能により，ユーザは Web ブラウザを介してエージェントへアクセスが可能である．本機能を用いるためには，述語 doService を用い，その本体部分に HTTP アクセスに対する出力処理を記述する．doService は，第 1 引数にアクセスの識別子をとり，第 2 引数にエージェントによる処理結果が束縛される．ここでの処理結果は，リスト構造で表現された HTML テキストであり，*MiLog* のパーザによって通常の HTML テキストに変換される．たとえば，エージェントが内部データベースに “itemName('iBook')”，“itemPrice('iBook','800')”，および “itemLeftTime('iBook','15min')” を持つ場合，以下の記述により，情報を Web ページとして出力できる．

```
doService(ID, OUT) :-
    itemName(X_0),
    itemPrice(X_0, X_1),
    itemTime(X_0, X_2),
    OUT = ['<html>','<body>',
           'Items Information','<BR>',
           'Name: ', X_0, ' ','<BR>',
           'Price: ', X_1, ' ','<BR>',
           'LeftTime: ', X_2, ' ','<BR>',
           '</body>','</html>'].
```

通常の CGI プログラムと同様，パラメータをともなうアクセスも可能である．*MiLog* では，HTTP アク

セスを受けた場合，各パラメータの情報が以下の形式で内部データベースに追加される．ここで `Parameter` はパラメータ名，`Value` はパラメータの値である．

```
option(ID,Parameter,Value)
```

そのため，パラメータ情報の存在を条件として出力処理の分岐が可能である．たとえば，ブラウザの種類を値としてとるパラメータ `browser` を利用し，適切な出力を生成する場合を考える．パラメータ値が `netscape` である場合に，Netscape Navigator に適した出力処理を記述した doService を実行するには，その本体部分に `option(_,browser,netscape)` を加えればよい．エージェント `agent` が `server.ics.nitech.ac.jp` 上で動作しており，通信で利用されるポート番号が 17008 番ならば，以下の URL からエージェントにアクセスできる．

```
http://server.ics.nitech.ac.jp:17008/agent?
browser=netscape
```

3. 考　　察

図 2 に *BiddingBot* のインタフェース例を示す．ここでは，財を監視している各エージェントから得られた財の価格，終了時間等の情報を情報を統合的に表示している．ここでは，表中の最左列には，各財に対する入札状況が表示されている．現在，*BiddingBot* では，Yahoo!Auction，eBay，および Amazon.com Auction に対する入札が可能である．

BiddingBot では，各々のサイトに特化したエージェントを生成する．2.1 節で述べた実装により，アクセス時のパラメータの与え方，および情報抽出パターンの変更により，サイトに特化したエージェントを生成可能である．特に，情報抽出に関して，テキスト解析のための逐次的な手続きを記述することなく，単一化による簡明な実装を可能にしている．筆者らは，単一化で用いる情報抽出パターンの記述コストを下げるための支援手法を提案している[2]．オークションサイトの情報提示形式には多くのバリエーションが存在するため，実用的な情報抽出機能の実現には多大な労力を強いられる．2.1 節で述べた実装により，負担が軽減できる．また，エージェント間の通信では，通信処理の詳細が隠蔽されるため，新規にエージェントを実装する際には，煩雑な通信処理にとらわれないコードの記述が可能である．実装の際は，表 1 に示す通信用の組み込み述語を状況に応じて利用すればよく，実装のコストを軽減できる．さらに，Web ブラウザを介したエージェントへのアクセス機能により，入札者であるユーザに対して，携帯情報端末を含む様々な環境からのエージェントへのアクセス手段を提供している．

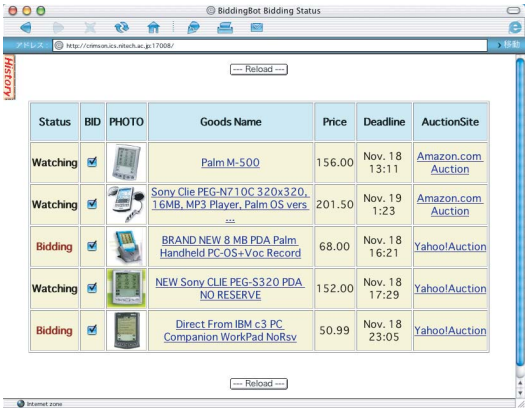

図 2 *BiddingBot* のインタフェースの一例
Fig. 2 An interface of *BiddingBot*.

4. おわりに

本論文では，実在のオンラインオークションに対して利用可能な入札支援システム *BiddingBot* の実装技法を示した．ここでは特に，情報収集/入札機能，エージェント間通信機能，およびユーザインタフェース機能に関して，論理型記述言語 *MiLog* に基づく実装の詳細を具体的に述べた．

参 考 文 献

1) Fukuta, N., Ito, T. and Shintani, T.: MiLog: A Mobile Agent Framework for Implementing Intelligent Information Agents with Logic Programming, *Proc. PRIIA-00*, pp.113–123 (2000).
2) Hattori, H., Yamada, R., Ozono, T. and Shintani, T.: A Multiple-Bidding Support Framework for Bidding and Browsing Information, *Proc. 11th International World Wide Web Conference* (*WWW2002*) (2002).
3) Sandholm, T.: eMediator: A Next Generation Electronic Commerce Server, *Proc. 4th International Conference on Autonomous Agents* (*Agent-2000*), pp.341–348 (2002).
4) Wurman, P.R., Wellman, M.P. and Walsh, W.E.: The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents, *Proc. Agents-98*, pp.301–308 (1998).
5) 伊藤孝行，服部宏充，新谷虎松：エージェント間の協調的入札機構に基づく複数オークション入札支援システム BiddingBot，人工知能学会論文誌，Vol.17, No.3, pp.247–258 (2002).

(平成 15 年 3 月 17 日受付)
(平成 15 年 5 月 6 日採録)