協調スコープを持つ協調型計算モデル

武宮博* 矢野博之** 布川博士*** 野口正一**

* 日立東北ソフトウェア

** 東北大学応用情報学研究センター

*** 東北大学電気通信研究所

あらまし 本稿では，分散環境において自律的な処理実行主体が動的に協調する様子を表現することを目的とした協調型計算モデルを提案する．このモデルにおいて，自律的な処理実行主体は，協調スコープという概念に基づいて動的に協調できる相手を認識する．また，動的な状況に適した通信形態を提案する．この計算モデルを用いて動的な協調が表現できることを，例を用いて示す．

Coordinated Computation Model with Coordination Scope

Hiroshi Takemiya* Hiroyuki Yano** Hiroshi Nunokawa*** Shoichi Noguchi**

* Hitachi Tohoku Software Co.,Ltd

** Reserch Center for Applied Information Sciences,Tohoku University

***Reserch Institute of Electrical Communication,Tohoku University

abstruct In this paper,we propose the coordinated computation model for describing dynamic coordination among the autonomous objects in the decentralized environment.In this model,autonomous objects can dynamically recognize partners for coordination by the coordination scope.Next,we propose the communication form suitable for the case, which is vague in the relation among objects.We describe some examples using this model and show that this model can describe dynamic coordination efficiently.

1．はじめに

近年の計算機システムの大規模化に伴い，新しいシステム構成法として"自律分散協調"という概念の重要性が主張されている[1]．また，この概念を計算モデルやプログラミング言語に活かし，計算主体という自律的な処理実行主体（エージェント，オブジェクトなどとも言う）を用いて，人間や動物などが行う協調動作を自然に表現することを目的とした言語も提案されてきている[2][3][4][5][6].

我々は計算モデルにおける"自律分散協調"という概念を次のように捉えている[2].

・自律：計算主体は単独で自分の行動を決定できる．

・分散：他の計算主体を管理したり，系全体の状態をすべて把握している永続的な計算主体はいない．（権力，情報の分散，すなわち分割性ではなく分権性）

・協調：計算主体は周囲と情報を交換することによって自らの行動を変化させることができる．

このように自律分散協調概念を捉えた場合，従来の計算モデルやプログラミング言語はこの概念を十分に表していない．

これらの概念が重要となるのは，問題の全貌が捉えにくく，処理がどの様に進んで行くのかが，前もってよくわからないような場合である．このような場合，計算主体は，局所的な情報を基にして，動的に協調しながら処理を進めて行かなければならない．

ここで言う"計算主体が動的に協調する"とは，以下の場合を指す．

・どの計算主体と協調するかが，実行以前に定まっていない．個々の計算主体の状態に従って，協調相手が変わる．（1－1）

・個々の計算主体に着目した場合，どのようなことに関して協調するかが固定化されていない．計算の過程で協調手段が変化したり，新しく協調手段を獲得したりすることによって異なる協調手段をとるようになる．（1－2）

我々の計算モデルは，計算主体間のこのような動的な協調をうまく記述することを目的としている．

（1－1）の状況を表現するために，我々は協調スコープという概念を用いる．協調スコープとは，その場その場で自分と協調できる相手を認識できる概念的スコープである．このスコープを用いて，計算主体は動的に変化する協調相手を認識し，協調動作を行う．

（1－2）の状況を表現するために，計算主体が実行中にメッセージに対する反応の仕方（スクリプト）を変更，追加，削除できるようにしている．

更に上記（1－1），（1－2）に加え，個々の計算主体間の関係が明瞭でないような問題に適した通信形態として，二つの通信形態を提案する．

・通知型通信

相手を特定しない制限的ブロードキャストを基本とする．（2－1）

・協調型通信

協調スコープの概念を基礎としており，計算主体は状況に応じて協調できる相手を認識し，その相手と通信を行う．（2－2）

本稿では，協調性に関して（1－1），（1－2）の機能を持ち，協調のための通信形態として（2－1），（2－2）を持つ計算モデルを提案し，その利用例を示すことにより有効性を述べる．又，従来の計算モデルとの比較を行うことにより，我々が提案した計算モデルにおける自律分散協調概念の有効性を示す．

2．協調スコープ

この章では，協調のメカニズムに関して議論し，その後，動的に相手の変化する協調を表現するための概念である協調スコープについて述べる．

2.1 協調のメカニズム

動的に協調相手が変化するような協調において，ある相手と協調する必要性があるかどうかを決定する機構は，次のように考えられる．

ある問題に関して相手と協調すべきかどうかは，一般に自分の状態とその相手の状態の兼ね合いで決まる．計算主体の状態は，計算主体の持つ変数の値によって定められる．従って，個々の計算主体は協調必要性を決定する状態変数（ある協調において計算主体の状態を示す変数）を持っており，協調必要性はそれらの値の兼ね合いによって決定される．

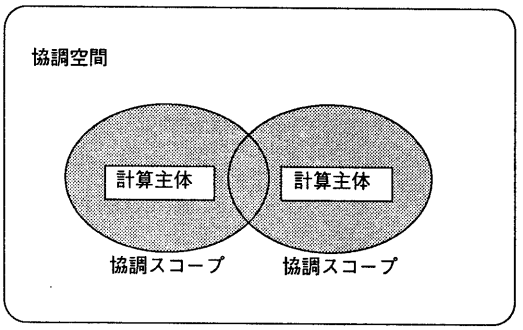

図1．協調スコープ

簡単な協調動作の実現として，ある協調行為に参加するかどうかというメンバシップを規定する境界（場，環境などと呼ばれる）を定め（以下，フィールド型の境界と呼ぶ），その境界内の計算主体に協調行為を行わせるという方法がある．これまでに提案された計算モデル[2][3][4][5][6]はこの方法をとっている．この境界による区別は，協調の仕方を知っているかどうかという自分自身のみで決定できる二分律的な区別である．

この実現方法では，じゃんけんのように境界内の計算主体全員が行うような動作は自然に表現できる．しかし，境界内の計算主体の一部が協調を行い，さらに動的にその構成を変えていくような，より一般的な協調行為をうまく表現できない．それは，上記の境界が，その中の計算主体は協調できるという協調可能性しか規定しないからである．このような動的な協調を表現するためには，協調可能性の規定だけではなく，その時点で協調すべきであるという協調必要性を規定する機構が必要である．協調必要性を規定する機構を持たない上記のモデルでは，各計算主体が協調可能な全計算主体の情報を保持せざるを得なくなり，不自然な表現を強いられる．

協調必要性を規定する機構は，次のようにイメージすることができる．協調行為を行う可能性のある計算主体は，計算主体毎に自分と協調すべき計算主体のメンバシップを規定する境界（スコープ型の境界と呼ぶ）を設ける．スコープ型の境界とフィールド型の境界の違いは，フィールド型の境界が個々の計算主体とは独立の静的な条件であったのに対し，スコープ型の境界はそれを設けた計算主体の状態の変化に従って変る動的な条件であることである．この動的な条件を満足した計算主体のみが，その境界を設けた計算主体から協調パートナーとして認識されることになる．

この計算主体毎の協調パートナー認識機構を，我々は次のようにモデル化する．まず，協調必要性を決定する鍵となる量を定め，その量を表すための空間を張る．この空間を協調空間と呼ぶ．

協調空間内で，各計算主体は協調可能な範囲を持つ．これを協調スコープと呼ぶ（図1）．協調スコープは，一般的に計算主体の持つ状態変数（スコープ変数と呼ぶ）の関数として表される．従って，計算主体のスコープ変数が変化するにつれて，協調スコープの大きさや位置は変化する．

2つの計算主体の協調スコープが重なったとき，すなわち，協調空間において共通の協調可能な領域を保持したとき，協調必要性が生じるとする．従って，2つの計算主体が協調空間において共通の領域を保持した時に協調必要性が生じるように鍵となる量を定める必要がある．

二つの計算主体が協調空間内で共通の領域を保持したとき，それらの計算主体は互いに相手の協調スコープ内にいるという．計算主体は，協調スコープ内の計算主体を認識できる．すなわち，協調スコープは，現在の自分の状態において協調すべき相手を認識できるセンサである．このスコープを用いることにより，状態の変化に応じて動的に協調相手の変化する問題において，個々の計算主体は動的に協調できるパートナーを認識できることになる．

3．協調型計算モデル

この章では，我々の提案するモデルの構成及び，計算主体間の協調を念頭においた通信形態について述べ

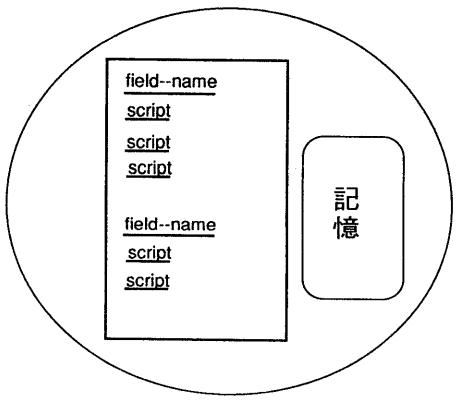

図2．計算主体の構成

る．このモデルは，実際に処理を進めていく主体である計算主体と，その計算主体の存在を規定するfieldから構成される．

3.1 計算主体

我々が対象としているのは，最初からどの様に処理が進むのかがよくわからないような計算である．そのような計算において，各計算主体は自分が何をすればよいかということは知っているが，情報が分散しているから，処理の流れに関して大局的な見方をしにくい．従って，計算の流れの中でそれをいつ行えばいいかということを判断することは難しい．そのため，各計算主体の処理は何等かのメッセージが来たら，それに反応して処理を行うというメッセージ起動の形態をとる．

計算主体は，処理に応じて複数のfieldに同時に存在することができ，スクリプトはそのfield毎に規定される（図２）．従って，あるfieldでAというメッセージを受けた場合と，別のfieldでAというメッセージを受けた場合では，反応して起こす行動が異なる．また，他の計算主体と契約を結ぶことによって知人関係になることができるが，その知人関係もそのfieldのみに限定される．

計算主体は，処理の流れの中で動的にスクリプトを獲得，変更することができる．逆に，スクリプトを消去することもできる．これにより計算の過程で，動的に協調手法を変化させたり，新たな協調手法を学ぶことができる．又，計算主体自身も動的に生成，消滅することができる．

３．２ fieldの機能

我々の用いるfieldは通常環境や場と呼ばれているものとは異なる．

環境や場の機能は，

１．属する計算主体のメンバーシップを定める境界

２．外部から中の計算主体の存在を隠蔽する抽象化カプセル

３．外部からのメッセージをfieldに属する計算主体に伝達する通信媒体

である．fieldはこれらの機能の他に，更に

4. 属する計算主体が動的な協調を繰り広げる協調のステージ

としての機能を持つ．環境や場では潜在的に協調行為を行う可能性のある計算主体の規定しかできない．fieldにおいて動的に相手の変化する協調を表現したいときは，fieldに協調空間(stage)を張り，その中で計算主体は協調スコープを持つ．この場合fieldは構造的である(structured)と宣言される．単に上記１,２,３の機能しか必要のないfieldは，構造的でない（unstructured）と宣言される．fieldが構造的である場合，その空間で計算主体が使用する協調スコープの名前と構造が宣言される．構造的でないfieldにおいて協調スコープを利用する場合は，単にスコープの名前を宣言するだけでよい．この場合は，協調スコープによってそのfieldに存在する全ての計算主体を認識することができる．

fieldも計算主体と同様，動的に生成，消滅することができる．

fieldの記述形式は以下の通りである．

```
(defineField <fieldName> <fieldType>
  (MakeStage <coordinateName> <stageRegion>)
  (MakeScope <scopeName> <scopeVariable>
                                <scopeRegion>)
  (Condition <conditionName> <scopeName>
                                <scopeName>))
```

condition部には，自分と他の計算主体の協調スコープ名が書かれ，その２つの協調スコープが共通領域を持ったとき，お互いに協調相手として認識できる．

計算主体のfieldへの出入りは次の命令で行われる．

(In <fieldName>)

(Out <fieldName>)

３．３ 通信形態

我々の提案する協調型計算モデルでは，計算主体はfieldへの出入りや，生成消滅が自由に行えるため，通信相手を指定して１対１の直接通信を行うメッセージパッシング方式では，相手にメッセージが届くかどうかを保証できない．また，届いたとしても，計算主体は動的にスクリプトを変更できるため，メッセージに正しく反応するかどうかを保証できない．従って，通信手法は，そのメッセージに正しく反応することのできる計算主体を規定するfieldに属している計算主体へのブロードキャストを基本としなければならない．

通信形態は，大きく分けて２つあり，通知型，協調

型と呼ぶ．各々には更に２つの形態が存在し，契約型，非契約型がある（表１）．これらの通信形態は，すべて非同期通信を基本としている．各通信形態の性質を以下に述べる．

３．３．１通知型通信

通知型は，異なるfieldに属する計算主体間の通信形態である．異なるfieldに対して送られたメッセージは，そのfieldに属する全ての計算主体に送られる．送信元の計算主体は，そのメッセージをどの計算主体が受け取ったかは関知しない（図３）．

通知型非契約通信では，送信先もそのメッセージがどの計算主体から送られたのか関知しない．従って，通知型非契約通信は返答を期待しない一方向の通信である．

通知型非契約通信で用いられる命令はsendである．

(Send <to-fieldName> <message>)

しかし，特定の相手と一連の通信を行う必要がある場合もある．この場合に使われるのが，通知型契約通信である．通知型契約通信では，通信を開始する際，自分の名前を明記して送り先に返事を要求する．この時点で，送信元は，そのfieldからの退出，スクリプトの変更を禁止され，それを要求するメッセージは処理を先送りされる．送信先のfieldに存在する計算主体の中で契約を受諾する計算主体は，自分の名前を明記して返答する．計算主体はfield毎に契約リストを持つ．契約を受諾した計算主体，及び契約受諾メッセージを受け取った計算主体は，自分の契約リストに契約名と，契約者名を書き込む．契約者名には，副項目としてその計算主体の都合で決めた別名を登録することができる．契約受諾メッセージを送信した段階で契約は完成し，その時点で契約を受諾した計算主体はfieldからの退出，スクリプトの変更を禁止される．これ以後，両者は相手に対して直接メッセージを送れるようになる．

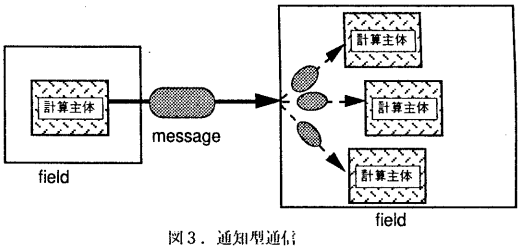

図３．通知型通信

一連の通信が終了すると，最終的に相手に契約終了メッセージを送り，両者は契約リストから相手の名前を消去し，共に通信相手の名前を忘れる．その時点で，両者のfieldからの退出，スクリプトの変更が許可される．

通知型契約通信で使用される命令は以下の通りである．

・契約の申し込み

(Contract <to-fieldName> <from-fieldName> <from-ID> <contractMessage>)

・契約の受諾

(Contreply <to-fieldName> <to-ID> <from-ID> <contractMesssege> <message>)

・契約中の通信

(Send <to-fieldName> <to-ID> <contractMessage> <message>)

・契約終了通知

(Contract-end <to-fieldName> <contractName>)

<to-ID>には，all,one_ofが利用できる．又，all,one_ofの引数として更に契約者リストに登録した別名を利用することができる．

３．３．２　協調型通信

協調型通信は，前述の協調スコープにより認識した

		送信者が受信者を認識	受信者が送信者を認識	
通知型	非契約	×	×	
	契約	×	○	(契約開始)
		○	○	(契約中)
協調型	非契約	○	×	
	契約	○	○	

通知型	field間の通信 (少なくとも最初は) 送信者は受信者を認識できない
協調型	同一field内で協調スコープを利用した通信 送信者は最初から受信者を認識

非契約	一方向通信 受信者は送信者を認識できない
契約	双方向通信 受信者は最初から送信者を認識

表１．通信形態

同一のfield内の相手のみと通信できる通信形態である（図４）．協調型通信では，一連の通信中，スコープ変数の変更を許すかどうかによって協調型非契約通信と協調型契約通信に分類できる協調型契約通信では，一連の通信中，両者が勝手にスコープ変数を変更して，そのために通信が途中で不可能にならないように，スコープ変数の変更が禁止される．協調型非契約通信では，両者がスコープ変数を変更することが許される．

通知型と協調型の大きな違いは，通知型が最初どの計算主体と通信を行うのかわからないのに対して，協調型は協調スコープを利用することにより，最初からどの計算主体と通信するのかがわかっていることである．

協調型計算において使用される命令は以下の通りである．

・協調型非契約通信

(Send <condition> <to-ID> <message>)

・協調型契約通信

(Contract <condition> <from-ID> <contractMessage>)

3.4 協調スコープの機能

協調スコープの機能としては，２つの機能がある．それを順に説明する．

・.協調相手の認識

(Who-in-the-Scope <condition>)

この関数を利用することにより，自分のスコープ内に存在する計算主体の数，計算主体名がわかる．

・協調空間内のある領域がスコープ内にあるかどうかの認識

(In-the-Scope <scopeName> <region>)

協調スコープの利用開始，終了は次の命令で行う．

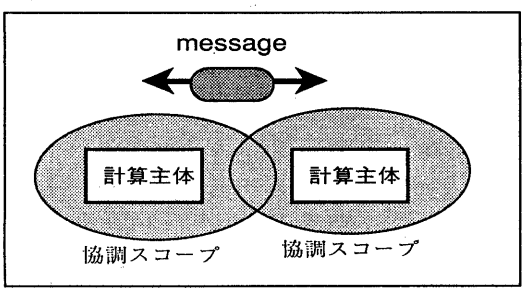

図４．協調型通信

(Open <scopeName>)

(Shut <scopeName>)

通知型通信では，計算主体はfieldに属してさえいれば，他のfieldからのメッセージは受理することができるが，協調型通信では，両者の協調スコープが重なった時しか相手を認識できない．従って，スコープをopenしていない計算主体は，同一field内の計算主体から認識されることはない．この事から，協調スコープのopen，shutは計算主体の協調意志を表すと見なすこともできる．

４．記述例

この章では，例題によって本モデルの協調動作の記述に於ける有効性を示す．例として，協調的ランダムウォーク及び，蹴鞠のシミュレーションをとりあげる．

４．１　協調的ランダムウォーク

動的に協調相手の変化する例として，協調的ランダムウォークを考える．協調的ランダムウォークとは，複数の人間が街の中（２次元の格子で表される）を非同期に他人とぶつからないように歩くというものである．この問題における協調とは，自分と他人の位置によって，お互いが協議して進路を変更し，ぶつからないようにすることである．

この問題で表現したいのは，街に存在する計算主体全てがまとまって協調し合う様子ではなくて，個々の計算主体が自分の周囲の計算主体とのみ協調し合い，その結果として，全体がうまく動作する様子である．

従って，この問題をうまく記述するためには，街のメンバーを規定するフィールド型の境界の他に，個々の計算主体が実際に協調し合うためのメンバーを規定するスコープ型の境界が必要である．

一般のモデルではこの境界が欠如しているため，街全体の計算主体が一まとまりとして協調しなければならない．又，一般のオブジェクトは自分の状態を全て隠蔽してしまうため，計算主体が協調すべき相手を自律的に認識することができない．

このモデルでは，協調スコープという形でその時点における計算主体の状態を他の計算主体に開放することによって，協調相手を認識することを可能にしてい

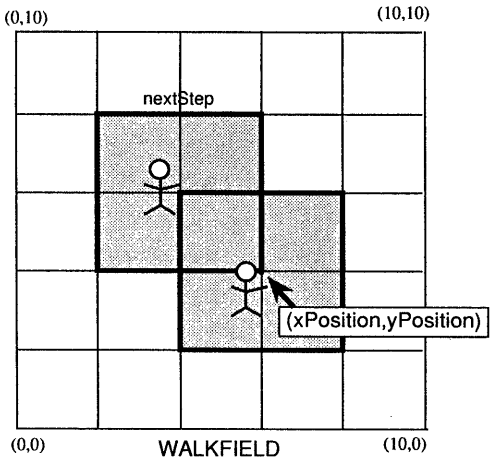

図5．協調的ランダムウォーク

る．さらに，情報交換を行う対象を，協調スコープによって必要最小限の対象に制限することにより，必要な情報を必要な対象のみが知ることができる．

協調的ランダムウォークを，協調スコープを用いてシミュレートすることを考える．この問題の場合，協調の鍵となるのはお互いの位置関係であるから，協調空間は2次元の格子である．協議し合って進路を変更する必要があるのは，次に滞在する格子点が同一の点になる可能性がある場合なので，協調スコープとしては，次に滞在する格子点の集合をとれば良い（図5）．

協調スコープの表現を以下に示す．協調的ランダムウォークが行われるfieldをWALKFIELD，その中でWALKERが持つ協調スコープ名をnextStep，協調条件名をneighborとする．

```
(defineField WALKFIELD
  'Structured
  (MakeStage 'cartesianCoordinate (MakeInterval 0 10) (MakeInterval 0 10 ))
  (MakeScope 'nextStep
            (xposition yposition)                    {スコープ変数}
            (ScopeRegion  (MakeInterval xPosition-1 xPosition+1)
                          (MakeInterval yPosition-1 yPosition+1)))
    (Condition 'neighbor nextstep nextstep))
```

各個人の動作を，以下に示す．

1. 協調スコープ内の計算主体を認識する．周囲に誰も存在しなければランダムに移動する．
2. 周囲に誰か存在すれば，協調型契約通信によって契約を結び，現在の位置情報を交換する．
3. 周囲の計算主体の位置が全て判ったら，移動可能な点を見つけて，移動候補点として周囲の計算主体の協調型契約通信で了承を求める．
4. 他の計算主体の移動候補点と一致した場合は，それらの間で協調型契約通信を用いてじゃんけんをおこない，勝者が移動権を獲得する．
5. 移動した後，1.に戻る．

この動作の間，計算主体は自分と協調する必要のある計算主体の情報のみに基づいて行動する．

4．2　蹴鞠

動的協調のもうひとつの例として，蹴鞠のシミュレーションを考える．蹴鞠は日本古式の遊びで，複数のplayerが鞠を落下させないよう蹴り続けるというものである（図6）．playerは疲れたらreserveと交替する．reserveはじゃんけんによって交替選手を決める．ball, player, reserveは幾つ存在していてもよい．player, ball, reserveの属するfieldをそれぞれplayerfierld, ballfield, reservefieldとする．

この蹴鞠のシミュレーションでは3種類の協調が存在する．

1. ballがplayerに落下地点と滞空時間を通知することにより，kickerを決めること．
2. 鞠を複数のplayerが蹴れる状態にあるとき，その中で誰が蹴ることができるかを決定すること
3. playerと誰が交替するかをreserve間で決定すること

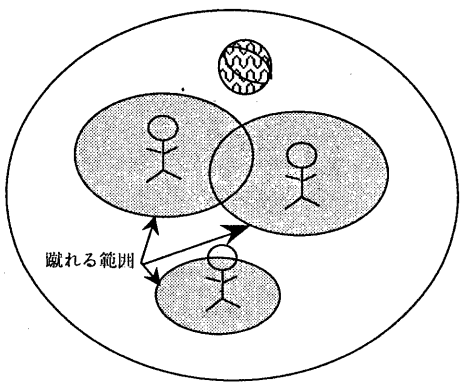

図6．蹴鞠

である．この三種類の協調形態を表現することを考える．

ballとplayer間の協調は，ballがplayer 全員に滞空時間と落下地点を通知し，kickerから蹴る方向と強さを受け取れば良いから，ballからplayerfieldへの通知型契約通信で実現できる．

player間の協調は，鞠の滞空時間や落下地点，playerのその時の位置によって動的に相手が変化する．従ってplayerfieldは構造的である．player間の協調の鍵となるのは距離であるから，協調空間として２次元の距離空間を張る．playerは協調スコープとして，自分の蹴れる範囲を持つ．スコープ変数は，ボールの滞空時間，足の速さである．

一方，reserve間の協調は，全員でじゃんけんをして交替選手を決定することである．従って，協調するための条件は，reserveであることである．従って，reservefieldはunstructuredでよい．

シミュレーションの進行は次のようになる（図７）．

1. ballがplayerfierldに滞空時間，落下地点を通知型契約通信により通知する．
2. playerは自分のスコープ内にballが落下するかどうかを確認する．
3. 蹴ることのできるplayerは自分のスコープ内に他のplayerが存在するかどうかを確認し，誰もいなければ，自分が蹴る．
4. 存在するときは，それらの間協調型契約通信によりkickerを決定する．
5. kickerはballに蹴る方向と強さを返答する．
6. 1．に戻る．

playerがreserveと交替したいときは（図８），

1. playerがreservefieldに交替したい旨を通知型契約通信で通知する．
2. reserve間のじゃんけん（協調型契約通信）により交替する計算主体を決定し，playerに返答する．
3. 交替の際，playerとreserveは互いのスクリプトを交換する．

このように，協調の特質を考慮してfieldの構造や通信形態を決定してやることにより，問題の本質のみを簡潔に記述できる．

このシミュレーションでは，playerの協調スコープを蹴れる範囲とし，協調が必要なのは，自分がballを蹴ることができて，かつ自分のスコープ内に他の計算主体がいる時とした．しかし，本来協調が必要なのは，複数のplayerの蹴れる範囲にballが落ちた時である．上記の認識方法では，他の計算主体が蹴れなくても自分のスコープ内に存在していれば，協調パートナーと認識してしまう．これは，現在の協調パートナー認識機構が，自分と他の計算主体という二者の関係しか記述できないためである．従って，自分とball，自分と他のplayerのほかにさらにballと他のplayerの関係ががからんでくるような複雑な状況をうまくシミュレートできない．このようなより複雑な協調形態に適応できる協調パートナー認識機構を考えることが今後の課題である．

5．他のモデルとの比較

我々の計算モデルは，一般の並列・並行オブジェクト指向計算モデルとは異なる目的で構築されている．並列・並行オブジェクト指向では，オブジェクトを局所記憶と手続きを持った論理的，物理的実体で，（仮

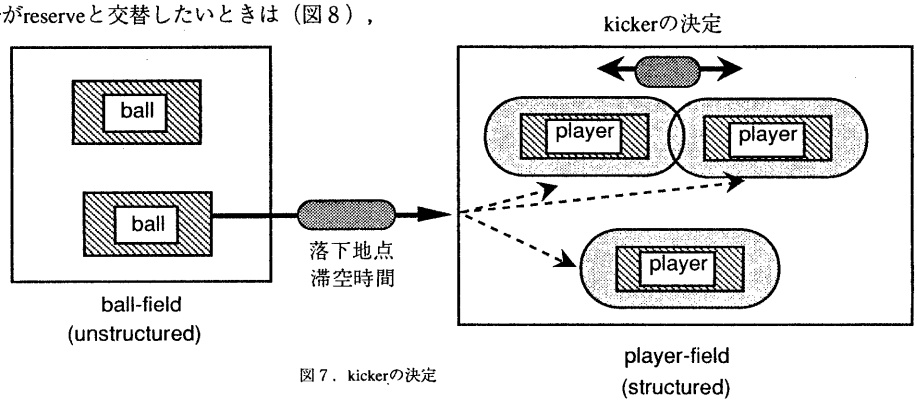

図７．kickerの決定

想的な）処理装置との対応付けがなされた物として捉えている[8]．これは，並列・並行オブジェクト指向モデルが，そのオブジェクトによって効率的な処理を行える概念的枠組の提供を目的としているからである．従って，通信形態においても，同期・非同期，放送・非放送といった並列処理の効率に直接結び付くレベルの議論を行っている．

並列・並行オブジェクト指向計算モデルであるABCM/1[9]では，通信形態として非放送（1対1の相互作用）しか与えていないため，協調的ランダムウォークのような問題をシミュレートしようとすれば，必ず全てのWALKERを管理する管理者をあらわに表現しなければならない．これは，我々の言う意味での分散性（分権性）に反するし，自然な問題の表現とはいえない．

Kamui88[3]も並列オブジェクト指向を基本としているが，自律性や協調性にも言及しているモデルである．Kamui88では場を用いることにより，協調するオブジェクトを規定することができるが，場は我々のモデルのフィールド型の境界としての機能しか持たないので，場の中に存在するオブジェクトが動的にサブグループを構成し，協調していく様子を表現するためには，場の中の大域的な情報（その場に属する全オブジェクトの個数や内部状態）を常に知っておかなければならない．

組織計算モデル[4]は，自律性や協調性を意識して構築されたモデルで，エージェントが複数のメッセージの処理を自律的に行える点に特徴がある．しかし，協調性を捉える枠組は，Kamui88と同様である．

Cellula[5][6]も協調性を意識して構築されたモデルで，プロセスと場を一体化したCellを基本的な実体と捉えた点に特徴がある．しかし，Cell間の関係の基本を親と子という階層性としているため，協調関係もマスタ・スレーブ関係が基本となっている．

自分の関連する状況を動的に把握し，それに対して自律的に反応するといった点においては，PARADISE[7]は我々の提案するモデルに近いといえる．しかし，PARADISEにおけるゴブリンの存在環境はゴブリンランドと呼ばれる単一の実距離空間のみである．従って，例えば蹴鞠をシミュレートしようとすると，reserveを何処に配置するかといった問題の本質とは異なる点もプログラマが意識して記述し，全てのゴブリンを一つの世界の中に配置しなければならない．これは不必要に込み入った記述をプログラマに強いることになる．また，情報伝達モードがバルーンによる一種類しかないため,相手を特定しない単なる情報の伝達も，相手を認識して行う協調行動も区別することができない．PARADAISEがこのような情報伝達機構を採用したのは，PARADISEがアニメーションのキャラクタによる行動シミュレーションの表現を目的としていたためである．

協調的分散問題解決(Coopertive Distributed Problem Solbing)の分野でも，自律，分散，協調の概念を検討している[10]．その中のひとつのモデルとして，契約ネット(Contract Net)がある[11]．このモデルでは，ネットワーク内におけるタスクの割当をContract Net Protocolという契約プロトコルを用いて各ノードが協調して実行していく．このモデルにおけるノード間の関係は，基本的に管理者と契約者という形になっており，契約はトップダウン方式で結ばれる．しかし，階層的な協調は協調形態のごく一部を表現しているに過ぎない．我々のモデルでは，通知型の通信を用いて異なる階層の計算主体の協調を表現し，協調型の通信を用いて同一階層内のノード間の協調を表現することができる．

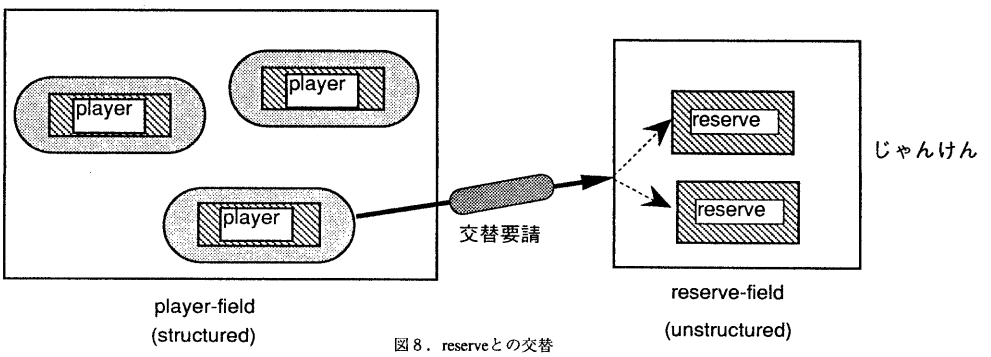

図8．reserveとの交替

6．まとめ

"協調する"と言った場合，"どの様な環境で"，"誰と"，"何に関して"，"どの様に"，協調するかということが問題となる．これらの要素が計算の進行に従って動的に変化する時，そのような協調を"動的な協調"という．本論文では，動的な協調の表現を目的とした協調型計算モデルを提案した．

具体的には，協調相手が動的に変化する状況を表現するために，協調スコープという概念を提案した．協調目的や協調法が動的に変化する状況を表現するために，計算主体が実行中にスクリプトを変更，追加，削除できることにした．field毎に異なる協調法を持たせるために，field毎に計算主体のスクリプトを記述して，多面性を持たせた．

また，このような動的な状況の下では，どの計算主体と通信可能であるかということがあらかじめ明確ではない．そこで，通信形態として，fieldへの制限的ブロードキャストを行う通知型通信と，協調スコープにより誰と協調できるかを認識した上でその相手への制限的ブロードキャストを行う協調型通信という二種類の通信形態を提案した．

本モデルでは，能動的に状態を変化させる計算主体と，計算主体の存在を規定するfieldを区別し，field自体の状態は変化しないものとしているしかし，fieldの状態が変化することによってそのfieldに属する計算主体の協調行動が変化することも十分にあり，本来ならば，fieldは静的な協調の舞台を提供するだけでなく，能動的に協調行動に影響を及ぼすべきである．また，２つの異なるfieldが融合することによって新しいfieldが生成され，それによって新しい協調行動が生成されることも考えられる．このような機能をfieldに持たせることが，今後の課題である．

謝辞

本計算モデル構築に際し，熱心に御討論いただき，数々の有益な助言をいただいた，山形大学工学部情報工学科丹野州宣助教授，山形大学大学院工学研究科修士課程二年佐藤義則君に感謝致します．

参考文献

1．森，宮本，井原：自律分散概念の提案，電気学会論文誌，昭59-12(1984),pp.303-310.

2．矢野，武宮，布川，野口：自律的な協調を行う分権型計算モデルKemari，情処研報，Vol. 90-PL27(1990), pp.151-158.

3．渡辺，原田，三谷，宮本：場とイベントによる並列計算モデル-Kamui88，コンピュータソフトウェア，Vol.6,No1(1989),pp.41-55.

4．丸一，市川，所：自律的エージェントからなる組織の計算モデルと分散協調問題解決への応用，情報処理学会論文誌，Vol.31，No.12(1990), pp.1768-1779.

5．Yoshida:Design of a Cooperative Computation Model Cellula/2,ソフトウェア科学会第７回論文集(1990), pp.181-184.

6．吉田，楢崎：場と一体化したプロセスの概念に基づく並列協調処理モデルCellula，情報処理学会論文誌，Vol.31, No.7(1990), pp.1071-1079.

7．内木，丸一，所：行動シミュレーションに基づいたアニメーションシステムPARADISE，コンピュータソフトウェア，Vol.4, No.2(1987), pp.24-38.

8．所：オブジェクト指向分散計算について，ソフトウェア科学会第６回大会論文集(1989), pp.245-248.

9．米澤，柴山，Briot，本田，高田：オブジェクト指向に基づく並列情報処理モデルABCM/1とその記述言語ABCL/1，コンピュータソフトウェア，Vol.3, No.3(1986), pp.189-203.

10．Durfee,Lessor,Corkill：Trends in Cooperative Distributed Problem Solving, IEEE Transactions on Knowledge and Data Engineering, Vol.1, No.1(1989), pp.63-83.

11．Smith, Davis：Frameworks for Cooperation in Distributed Problem Solving, in Readings in Distributed Artificial Intelligence, Bond and Gasser, Eds. San Mateo, CA:Morgan Kaufmann, (1988), pp.61-70.