マルチエージェントシステム開発フレームワーク Yaccai
Yaccai: A multiagent systems development framework

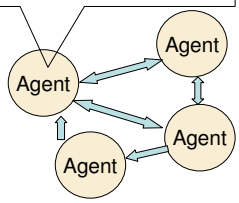
グェン トアン ドゥク　　竹内郁雄　　(東京大学)

背景

- 計算問題、資源が膨大
  - Grid, cluster, ... が普及
  - 集中制御不可能
- ソフトウェア エージェント
  - 自律的に推論、行動するプログラム
- マルチエージェント システム
  - 複数のエージェントが協調 (通信、同期、知識交換) → うまく分散制御

- 問題解決ための協調が重要
- しかし、協調記述が難しい

An agent in environment



マルチエージェントシステム

サンプル プログラム

```
1  agentclass HelloAgent {
2    public m_comm;
3    public function HelloAgent() {
4      m_comm = Environment.GetCommunicator(
5                 "World", MsgListener );
6    }
7    public plan MsgListener( msg ) {
8      id = -1;
9      match msg.Value with {
10       "Hello from", @{id}, "at", @{addr} -> {
11         belief fact new {rank=id, host=addr};
12       }
13     }
14   }
15   public plan act() {
16     myRank = Environment.GetRank();
17     m_comm.Bcast( "Hello from " + myRank +
18                   " at " + Environment.Hostname() );
19   }
20 }
21
22 class SimpleMultiAgentSystem {
23   public static function Main() {
24     a1 = create HelloAgent() at "localhost";
25     a2 = create HelloAgent() at "somehost.com";
26   }
27 }
```

communicator “World” に参加

match で受信したメッセージを解析

知識ベースに格納

推論サイクルのmain (自動に実行される)

集団操作: 自分のID (rank) と hostname をbroadcast

エージェントを fork する

目的

- 協調記述を容易にする
- 簡単にエージェント指向プログラムを構成できる

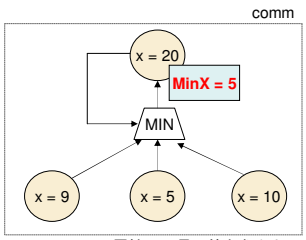
エージェント指向プログラミング (AOP) 言語

集団操作の応用例

- Global knowledge derivation
  = マルチエージェントシステム全体の知識

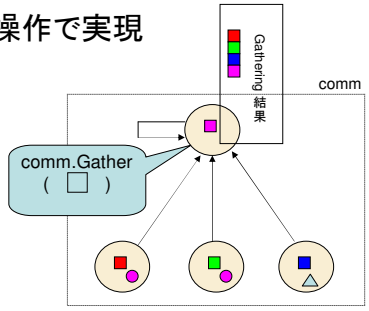
  Ex) 属性の最小値、総和 など

  →集団操作で実現



Reduce で属性 x の最小値を求める:

min_x = comm.Reduce(MIN, @{x});



Gather でデータを収集:

result = comm.Gather(data);

- エージェントの同期→Barrierで実現

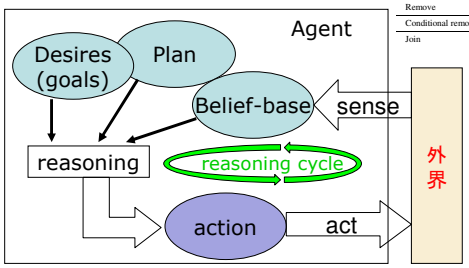
提案手法

- エージェントの構成手法
  - Belief-baseの組み込み
  - Reasoning cycle自動構成

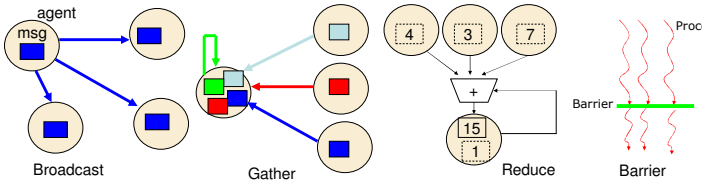
Belief-base operations

Operation	Example
Add fact	belief fact new { x = 1, y = 2 };
Query	belief query {x, y};
Projection	belief query {x, y} as p{ x };
Conditional query	belief query {x, y} as p where p.x == 1;
Remove	belief remove {x, y};
Conditional remove	belief remove {x, y} as p where p.x == 1;
Join	(belief query {x, y} as p) join (belief query {y, z} as q) on p.y == q.y;

cf. LINQ

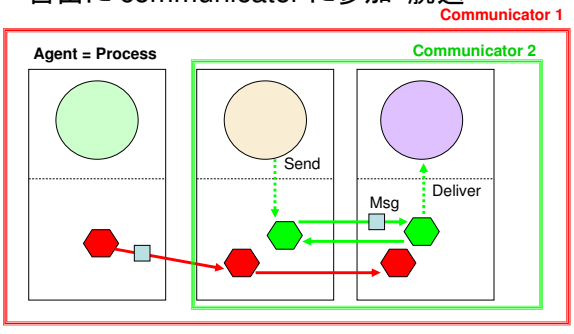


- 協調動作を集団操作で抽象化 cf. MPI
  - 集団操作とは同時に実行される操作



実行と通信モデル

- Reasoning と message passing の分離
- 自由に communicator に参加・脱退



実行と通信モデル

今後の課題

- 集団操作の有効性の評価
- Message passing の最適化
- 実際のアプリケーションの開発に応用

http://www.nue.ci.i.u-tokyo.ac.jp/~duc/yaccai/