

Behavior Based Modelling

ERIKA

Takeshi Matsumura

TM: Waseda University

Abstract. *The agents of my team are constructed of two layers; Reactive and Behavior Layer. Reactive Layer produces basic functions as an agent, that is, a world model, a perception and an actuator. Behavior Layer makes it easy for a designer to make a rational agent program. This paper especially focuses on the Behavior Layer.*

1 Introduction

RoboCup99 is the second world cup for Team Erika. At the previous one in Paris, Erika was beaten for all teams due to its slowness. These two layers, Reactive and Behavior, are introduced to make an agent faster.

Reactive Layer is based on BDI-architecture which has the features of pre-planning and the world-model mixed information from the perception and predicted information based on them [2][3]. This architecture is used by AT-Humbolt 98, too[1].

Behavior Layer produces the structure called behavior which is similar to the normal function object. The designer can keep agent's decision process in short time by using behavior. This Layer also produces simple synchronization with another agent.

2 Behavior layer

Behavior is a couple of a small goal (desire) to be achieved by oneself and a planner (intention) which generates a plan that is a sequence of actions.

Executing behavior means that an agent evaluates its intention repeatedly to achieve its desire (See Fig.1). The fundamentals of this layer are the method to define a behavior, a composition of two behaviors and a behavior execution engine.

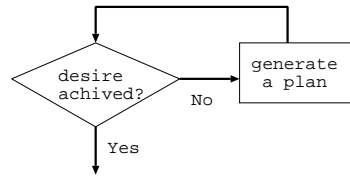


Figure 1: flowchart of executing behavior

2.1 Behavior Modelling

For example, goto-region behavior whose desire, short goal, is that an agent gets near to the given position (x,y) is modelled as follows:

```

goto-region ( x , y )
  desire      : (1) if he is near the position (x,y)
                ⇒ achieved
                (2) Another agent being on his way
                ⇒ unachievable
  intention   : dash to the direction for position ( x , y )
  
```

The desire part will return the symbol 'achieved', 'unachievable' or return nothing. The intention part will be executed repeatedly until the desire part returns either of the two symbols.

2.2 Behavior composition

Behavior can call another behavior in order to achieve its desire. Consider the last example, assuming agent A is executing goto-region behavior, if another agent B is on A's way to the target position (x,y) , A cannot achieve the behavior.

However, if he can run away the agent B, he will achieve the behavior. So we can rewrite the second part of the desire as follows :

```

desire      : (2) Another agent being on his way
                ⇒ Executing avoid-object behavior.
                If he couldn't achieve it
                ⇒ unachievable
  
```

where the avoid-object behavior is defined like this :

```

avoid-object ( obj, x , y )
  desire      : Another agent 'obj' is out of his view corn
                and he can see the target position (x, y)
                ⇒ achieved
  intention   : dash to the tangent direction of the 'obj'
                to avoid him.
  
```

While avoiding another agent if he checks only the desire of the avoid-object behavior, he may not find out that he would achieve the desire of

goto-region behavior. Therefore, he must check the composition of the two desires, that is:

(desire of goto-region) or (desire of avoid-object)

Behavior Layer makes this composition automatically.

2.3 Synchronization with another agent by evaluating his desire

Behavior Layer can produce a simple synchronization with another agent less communication. Assuming agent A knows that agent B is executing the goto-region behavior. The agent A would be able to find out for B to finish his goto-region behavior by evaluating the desire of the behavior in B's context.

It is equal to the following behavior:

```
observe-goto-region (B, x, y)
  desire      : (1) if B is near position (x,y)
                ⇒ achieved
                (2) Another agent being on B's way
                ⇒ unachievable
  intention   : observe the B's movement
```

Behavior Layer produces this translation by checking whether the agent himself executes it or another agent does.

2.4 Example of cooperation using behavior

Here is an example of cooperation between two agents using behavior synchronization. Receive-ball behavior makes the agent get a ball which is coming to him. It can be written like this:

```
receive-ball ( pl )
  desire      : Player pl has a ball ⇒ achieved
  intention   : if distance(pl,ball) < 5m then dash-to-ball
```

And we are thinking about triangle-passing, that is, at first agent A kick the ball to agent B, then B kick to another point (x,y) which A catches the ball. This passing's aim is to prevent the ball from opponent's interception. The table.1 is the sketch of activities both A and B. See that in the step 3, both A and B use receive-ball behavior, for B to receive ball actually, on the other hand for A to predict when the B receives the ball. Another important thing is that no auditory message passing are needed between step 3 to 4.

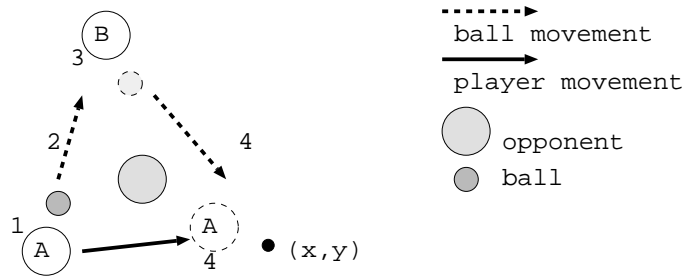


Figure 2: triangle pass

	A's activity	B's activity
1	Tell B that they will make triangle passing. Promise the ball receive point (x,y) from B.	
2	Kick ball to B	Catch A's message and Turn to the ball
3	OBSERVE receive-ball behavior	EXECUTE receive-ball behavior
4	Dash to the promised point (x,y)	Kick ball to the point (x,y)

Table 1: sketch of triangle passing

3 Conclusions

My agent is constructed with Reactive and Behavior Layer. Reactive Layer is based on BDI Architecture. Behavior which produces desire composition and simple synchronization with another agent less auditory communications makes it easy to design of rational agent.

I've modelled many basic actions as behavior and designed a triangle-passing cooperation using a behavior. At present I'm implementing them in Common Lisp language.

References

- [1] The robocup 97 team of the humbolt university. http://www.ki.informatik.hu-berlin.de/RocoCup/RocoCup97/index_e.html.
- [2] Anand S.Rao and Michael P.Geoff. Modelling rational agents within a bdi-architecture. *Australian Artificial Intelligence*, February 1991.
- [3] Anand S.Rao and Michael P.Geoff. Bdi agents:from theory to practive. *Australian Artificial Intelligence*, April 1995.