سیستم‌های چندعاملی

درس ۲۴

برنامه‌نویسی عامل‌گرا

Agent-Oriented Programming (AOP)

کاظم فولادی قلعه
دانشکده مهندسی، دانشکدگان فارابی
دانشگاه تهران

http://courses.fouladi.ir/mas

برنامه‌نویسی عامل‌گرا

AGENT-ORIENTED PROGRAMMING (AOP)

برنامه‌نویسی عامل‌گرا:
یک «پارادایم برنامه‌نویسی» بر اساس نگاه اجتماعی به محاسبات

ابداع‌شده توسط *Shoham*

در AOP عامل‌ها مستقیماً بر اساس تصورات ذهنی
(مانند باور، مطلوب و قصد در مدل BDI)
برنامه‌نویسی می‌شوند.

همان‌طور که انسان‌ها از چنین مفاهیمی به عنوان یک مکانیزم **انتزاع** برای
بازنمایی خواص سیستم‌های پیچیده استفاده می‌کنند،
می‌توانیم از آنها برای برنامه‌نویسی ماشین‌ها نیز استفاده کنیم.

مقایسه‌ی عامل‌ها و اشیا

AGENTS VS. OBJECTS

شیئ *Object*	عامل *Agent*
روی رفتار (متد) خود کنترل نشان نمی‌دهد	روی رفتار خود کنترل نشان می‌دهد
تصمیم‌گیری به عهده‌ی شیئی است که متدی را درخواست کرده است	تصمیم‌گیری به عهده‌ی عاملی است که درخواست را دریافت کرده است
شیئ درخواست را به رایگان انجام می‌دهد	عامل درخواست را به ازای سود انجام می‌دهد
انعطاف‌پذیری پایین	انعطاف‌پذیری بالا
رشته‌ی کنترلی شیئ در دست سیستم است	رشته‌ی کنترل عامل در دست خود عامل است

مقایسه‌ی برنامه‌نویسی شیئ‌گرا و برنامه‌نویسی عامل‌گرا

OOP vs. AOP

	برنامه‌نویسی شیئ‌گرا *OOP*	برنامه‌نویسی عامل‌گرا *AOP*
واحد پایه	شیئ *(Object)*	عامل *(Agent)*
پارامترهای تعریف‌کننده‌ی واحد پایه	نامقید	باورها، تعهدها، گزینه‌ها، . . .
فرایند محاسبه	روش‌های گذر دادن پیام و پاسخ	روش‌های گذر دادن پیام و پاسخ
انواع پیام‌ها	نامقید	اطلاع‌دادن، درخواست‌دادن، پیشنهاد دادن، قول دادن، نپذیرفتن، . . .
قیدهای روی متُدها	هیچ	صداقت، سازگاری، . . .

چهارچوب سیستم برنامه‌نویسی عامل‌گرا

یک سیستم AOP کامل شامل سه مؤلفه‌ی اصلی است:

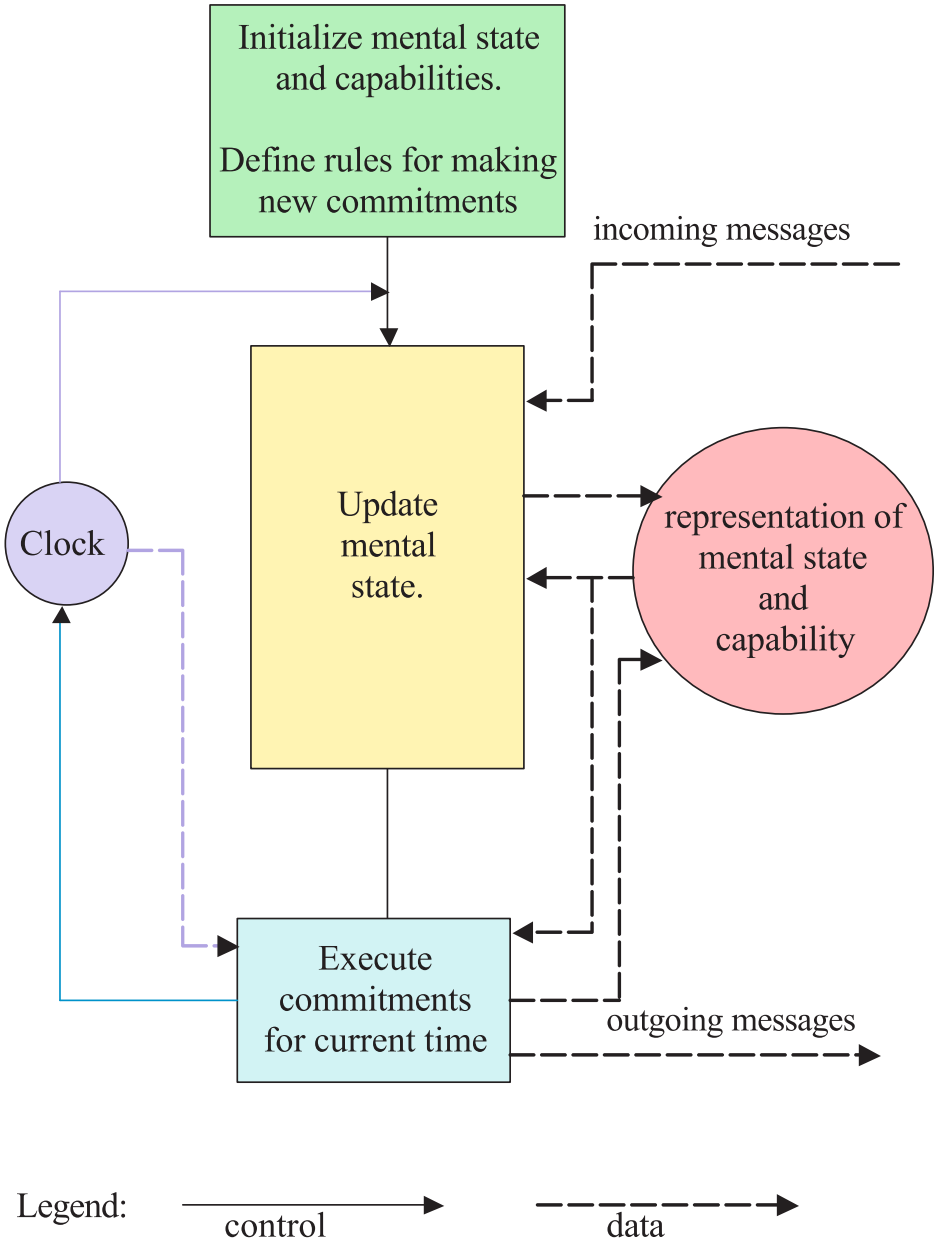
یک زبان صوری محدودشده *A Restricted Formal Language*	یک زبان برنامه‌نویسی مفسری *An Interpreted PL*	یک «عامل‌ساز» *An "Agentifier"*
شامل مدالیته‌های گوناگون (مانند باور و تعهد) با نحو و معناشناسی واضح برای توصیف حالت‌های ذهنی	برای تعریف و برنامه‌نویسی عامل‌ها، با فرمان‌های ابتدایی مانند: REQUEST, INFORM	برای تبدیل دستگاه‌های خنثی به عامل‌های قابل برنامه‌نویسی

مفسر عمومی عامل

A Generic Agent Interpreter

حلقه‌ی اصلی:

- خواندن پیام‌های جاری، و
  به‌هنگام‌سازی حالت ذهنی (شامل باورها و تعهدات)
- اجرای تعهدات زمان جاری،
  (ممکن است منجر به تغییرات بیشتر در باور شود)



کنش‌هایی که عامل می‌تواند به آنها متعهد شود:

۱) کنش‌های برقراری ارتباط مثل REQUEST, INFORM
۲) کنش‌های دلخواه «خصوصی»

زبان برنامه‌نویسی عامل‌گرا AGENT-0

AGENT-0 Agent-Oriented Programming Language

نخستین پیاده‌سازی از پارادایم AOP، زبان AGENT0 بود.

در زبان AGENT0 هر عامل بر حسب موارد زیر مشخص می‌شود:

مجموعه‌ای از **قابلیت‌ها** *Capabilities*	مجموعه‌ای از **باورهای** اولیه *Beliefs*	مجموعه‌ای از **تعهدات** اولیه *Commitments*	مجموعه‌ای از **قواعد تعهد** *Commitment Rules*
آنچه عامل می‌تواند انجام دهد	در نقش باورهای *BDI*	در نقش قصدهای *BDI*	در قالب شرط ـ کنش

- یک قاعده‌ی تعهد، شامل یک **شرط پیام**، **شرط ذهنی** و یک **کنش** است.
- یک قاعده «فایر» می‌شود، هرگاه
  - ـ شرط پیام با پیام دریافت شده توسط عامل مطابقت کند و
  - ـ شرط ذهنی با باورهای عامل مطابقت کند.
- وقتی یک قاعده‌ی تعهد «فایر» می‌شود، عامل متعهد به انجام کنش می‌شود.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

کنش‌ها و پیام‌ها

کنش‌ها *Actions*	
خصوصی *Private*	**برقرارکننده‌ی ارتباط** *Communicative*
روال‌هایی که درون عامل اجرا می‌شود	ارسال پیام‌ها

پیام‌ها (به عنوان کنش‌های گفتاری) *Messages (as Speech Acts)*	
درخواست *Request / Unrequest*	**اطلاع دادن** *Inform*
برای انجام کنش‌ها یا منع کنش‌ها: (معمولاً منجر به تغییر تعهدات عامل‌ها می‌شود)	گذر دادن اطلاعات، (معمولاً منجر به تغییر باورهای عامل‌ها می‌شود)

AGENT-0 زبان برنامه‌نویسی عامل‌گرا

قاعده‌ی تعهد

If

I receive a message from `agent`
requesting me to do `action`
at `time`,
and I believe that

- `agent` is currently a friend;
- I can do `action`;
- at `time`, I am not committed to doing any other action,

Then

commit to doing `action` at `time`.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

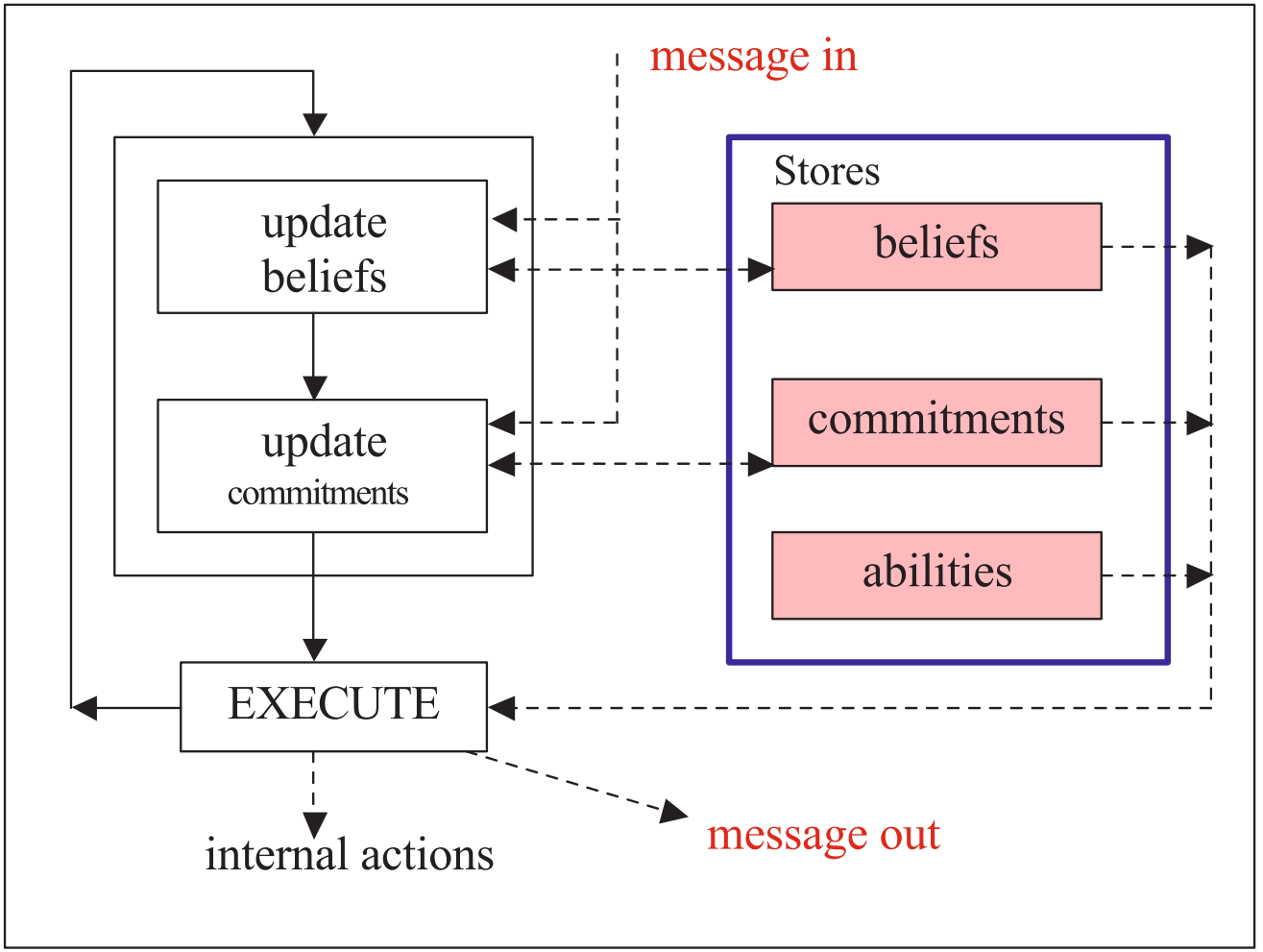
مثال: قاعده‌ی تعهد

```
COMMIT(
  ( agent, REQUEST, DO(time, action)
  ), ;;; msg condition
  ( B,
    [now, Friend agent] AND
    CAN(self, action) AND
    NOT [time, CMT(self, anyaction)]
  ), ;;; mental condition
  self,
  DO(time, action)
)
```

زبان برنامه‌نویسی عامل‌گرا AGENT-0

جریان کنترل

The flow of control in AGENT0



زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0

THE SYNTAX OF AGENT0

تعهدها تنها کنش‌های ابتدایی هستند
(که مستقیماً توسط عامل اجرا می‌شوند.)

زبان AGENT0 شرط‌ها را فقط برای اتخاذ تعهدها مشخص می‌کند.

تعهدها واقعاً اتخاذ می‌شوند، و سپس در زمان‌های مناسب بعدی به‌طور خودکار انجام می‌شوند.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: جمله‌های واقعیت

FACT STATEMENTS

جمله‌های واقعیت *Fact Statements*	جمله‌هایی که برای مشخص‌سازی محتوای کنش‌ها و شرط‌های لازم برای اجرای آنها استفاده می‌شوند.

مثال:

```
(t (employee smith acme))
(NOT (t (employee jones acme)))
```

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: جمله‌های کنش خصوصی و برقراری ارتباط

PRIVATE AND COMMUNICATIVE ACTION STATEMENTS

جمله‌های کنش
Action Statements

کنش‌ها می‌توانند خصوصی / برقراری ارتباط یا شرطی / غیرشرطی باشند.

The syntax for a **private action**

```
(DO t p-action)
```

`t` نقطه‌ی زمانی
`p-action` نام کنش خصوصی

کنش خصوصی می‌تواند حاوی IO باشد یا خیر.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: جمله‌های کنش خصوصی و برقراری ارتباط

PRIVATE AND COMMUNICATIVE ACTION STATEMENTS

جمله‌های کنش
Action Statements

کنش‌ها می‌توانند خصوصی / برقراری ارتباط یا شرطی / غیرشرطی باشند.

AGENT0 فقط سه نوع کنش برقراری ارتباط دارد:

The syntax of **informing**	The syntax of **requesting**	The syntax of **canceling** a request
`(INFORM t a fact)`	`(REQUEST t a action)`	`(UNREQUEST t a action)`

`t` نقطه‌ی زمانی
`a` نام عامل
`fact` یک جمله‌ی واقعیت
`action` یک جمله‌ی کنش (تعریف بازگشتی)

```
(REQUEST 1 a (DO 10 update-database))
(REQUEST 1 a (REQUEST 5 b
                    (INFORM 10 a fact)))
```

کنش برقراری ارتباط همیشه حاوی IO است و برای همه‌ی عامل‌ها مشترک است.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: جمله‌های کنش شرطی

CONDITIONAL ACTION STATEMENTS

جمله‌های کنش شرطی
Conditional Action Statements

- تعهدها برای کنش‌های شرطی،
  که شامل شرط‌هایی است که باید دقیقاً پیش از کنش کردن آزمایش شود.
- شرط‌هایی که برای ورود به تعهدها در موقعیت نخست لازم است.

یک کنش شرطی متکی بر **شرط ذهنی** است (که به حالت ذهنی عامل مراجعه می‌کند).

وقتی زمان اجرای کنش رسید، حالت ذهنی در آن زمان بررسی می‌شود
تا ببینید که آیا شرط ذهنی ارضا شده است یا خیر.

یک شرط ذهنی، یک ترکیب منطقی از الگوهای ذهنی به‌صورت‌های زیر است:

```
(B fact)
```

```
((CMT a) action)
```

E.g.: `(B (t (employee smith acme)))`

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: جمله‌های کنش شرطی

CONDITIONAL ACTION STATEMENTS

The syntax of a **conditional action**

```
(IF mntlcond action)
```

جمله‌های کنش شرطی
Conditional Action Statements

E.g.:

```
(IF (B (t' (employee smith acme)))
    (INFORM t a
            (t' (employee smith acme))))
```

شرط‌های ذهنی می‌توانند حاوی رابطهای منطقی AND، OR و NOT باشند.

E.g.: the following three actions constitute a query about whether fact is true (b is being queried and is asked to inform a):

```
(REQUEST t b (IF (B fact)
                 (INFORM t+1 a fact)))
(REQUEST t b (IF (B (NOT fact))
        (INFORM t+1 a (NOT fact))))
(REQUEST t b
        (IF (NOT (BW fact))
            (INFORM t+1 a
              (NOT (t+1 (BW a fact))))))
```

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: متغیرها

VARIABLES

روال‌ها مطابق مدل هدایت‌شده با الگو به‌کار گرفته می‌شوند:

قواعد تعهد توسط الگوهای خاصی در پیام‌های وارده و حالت ذهنی جاری فعال می‌شوند (عموماً حاوی متغیر).

هر متغیر با ? شروع می‌شود.
دامنه‌ی متغیر: نام عامل/ جمله‌های واقعیت/ جمله‌های کنش

E.g.:

```
(IF (NOT ((CMT ?x) (REFRAIN sing))) sing)
```

فرض می‌شود که متغیرها در جمله‌های کنش، دارای **سور وجودی** هستند.
متغیر دارای **سور عمومی** با !? شروع می‌شود و حوزه‌ی دید آن کل فرمول است.

E.g.:

```
(IF (B (t (emp ?!x acme)))
    (INFORM t' a (t (emp ?!x acme))))
```

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: قواعد تعهد

COMMITMENT RULES

بسیاری از جمله‌های کنش در زمان برنامه‌نویسی نامعلوم هستند:
توسط سایر عامل‌ها با برقراری ارتباط تعیین می‌شوند.

برنامه خودش حاوی شرایطی برای عامل است تا آن را وارد تعهدات جدید کند.

* بیشتر تعهدها در پاسخ به پیام ها شکل می‌گیرند.

شرایط تعهدها هم شامل شرایط ذهنی و هم شرایط پیام می‌شوند.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: شرط پیام

MESSAGE CONDITION

یک شرط پیام، یک **الگوی پیام** به‌صورت زیر است:

```
(From Type Content)
```

From نام فرستنده

Type نوع پیام INFORM, REQUEST, CONTENT

Content یک جمله‌ی واقعیت یا جمله‌ی کنش وابسته به Type

سایر اطلاعات مرتبط با یک پیام به‌طور ضمنی باقی می‌مانند.

E.g.:

```
(a INFORM fact)
```

یعنی: یکی از پیام‌های جدید از عامل a به‌عامل از fact اطلاع می‌دهد.

E.g.:

```
(AND (a REQUEST (DO t walk))
     (NOT (?x REQUEST (DO t chew-gum))))
```

یعنی: پیامی از عامل a وجود دارد که از عامل می‌خواهد قدم بزند ولی درخواست جدیدی از هیچ‌کس وجود ندارد که عامل آدامس بجود!

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: قواعد تعهد

COMMITMENT RULES

قواعد تعهد
Commitment Rules

The syntax of a **commitment rule**

```
(COMMIT msgcond mntlcond (agent action)*)
```

+ جمله‌ی کنش خودش می‌تواند حاوی شرط ذهنی خودش باشد.

E.g.:

```
(COMMIT (?a REQUEST ?action)
        (B (now (myfriend ?a)))
        (?a ?action))
```

زبان برنامه‌نویسی عامل‌گرا AGENT-0

نحو AGENT-0: برنامه

PROGRAM

برنامه *Program*	یک برنامه، دنباله‌ای از قواعد تعهد است، که به‌دنبال تعریف قابلیت‌ها و باورهای اولیه‌ی عامل می‌آید.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

گرامر BNF برای نحو زبان AGENT-0

BNF FOR THE AGENT-0 SYNTAX

```
<program> ::=
   timegrain := <time>
   CAPABILITIES := (<action> <mntlcond>)*
   INITIAL BELIEFS := <fact>*
   COMMITMENT RULES := <commitrule>*
<commitrule> ::=
   (COMMIT <msgcond> <mntlcond>
           (<agent> <action>)*)
<msgcond> ::=
    <msgconj> | (OR <msgconj>*)
<msgconj> ::=
    <msgpattern> | (AND <msgpattern>*)
<msgpattern> ::=
    (<agent> INFORM <fact>)   |
    (<agent> REQUEST <action> |
    (NOT <msgpattern>)
<mntlcond> ::=
    <mntlconj> | (OR <mntlconj>*)
<mntlconj> ::=
    <mntlpattern> | (AND <mntlpattern>*)
<mntlpattern> ::=
    (B <fact>)                |
    ((CMT <agent>) <action>) |
    (NOT <mntlpattem>)
```

```
<action> ::=
   (DO <time> <privateaction>)          |
   (INFORM <time> <agent> <fact>)       |
   (REQUEST <time> <agent> <action>)    |
   (UNREQUEST <time> <agent> <action>) |
   (REFRAIN <action>)                   |
   (IF <mntlcond> <action>)
<fact> ::=
   (<time> (<predicate> <arg>*))
<time> ::=
   <integer> | now | <time-constant>     |
   (+ <time> <time>) | (- <time> <time>) |
   (× <iD.teger> <time>)
      ; Time may be a <variable> when
      ; it appears in a commitment rule
<time-constant> ::=
   m | h | d | y
      ; m (minute) =60, h (hour) 3600, etc.
<agent> ::=
   <alphanumeric_string> | <variable>
<predicate> ::= <alphanumeric_string>
<arg> : :=
   <alphanumeric_string> | <variable>
<variable> ::=
   ?<alphanumeric_string> |
   ?!<alphanumeric_string>
```

AGENT-0 زبان برنامه‌نویسی عامل‌گرا

مثال (۱ از ۱۰)

Airline Reservation

Agents:

- P is a passenger.
- C is an airline clerk, a program.
- S is C's supervisor.

By confirming a reservation, the airline enters into a commitment to issue a boarding pass to the passenger at the appropriate time.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۲ از ۱۰)

March

- P to C: Please inform me what flights you have
  from San Francisco to New York on April 18.
- C to P: Flight #354 departs at 08:30,
  flight #293 departs at 10:00,
  flight #441 departs at noon, …
- P to C: Please book me on #354.
- C to P: That is sold out.
- P to C: Please book me on #293.
- C to P: That is confirmed: your reservation number is 112358.
- P to C: Please book me also on #441.
- C to P: That conflicts with #293:
  I am not allowed to double book a passenger.
- P to C: Please get permission to do so.
- C to S: I request permission for the fallowing double booking: …
- S to C: Permission denied.
- C to P: Sorry, I cannot get approval.

April 18, at the airport

- P to C: My name is P; I have a reservation for flight #293.
- C to P: Here is your boarding pass.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۳ از ۱۰)

Present a program implementing
a simplified version of the airline representative.

The ideas behind the program are that

- the relevant activity on the part of the airline is issuing a boarding pass to the passenger, and
- confirming a reservation is a commitment to issue a boarding pass at the appropriate time.

AGENT-0 زبان برنامه‌نویسی عامل‌گرا

مثال (۴ از ۱۰)

We first define some macros.

```
-----------------------------------------------
(issue_bp pass flightnum time) =>
  (IF (AND (B ((- time h) (present pass)))
           (B (time
                (flight ?from ?to flightnum))))
      (DO time - h
          (physical_issue_bp
                  pass flightnum time)))
```

Explanation:

Issue the boarding pass precisely 1 hr. (h) before the flight.

`physical_issue_bp` is a private action involving some external events.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۵ از ۱۰)

We first define some macros.

```
-----------------------------------------
(query_which t asker askee q) =>
  (REQUEST t askee
           (IF (B q) (INFORM (+ t 1) asker q)))
```

Explanation:

This requests only a positive answer.

If `q` contains a universally-quantified variable, then `query_which` requests to be informed of all instances of the answer to the query `q`.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۶ از ۱۰)

We first define some macros.

```
-----------------------------------------
(query_whether t asker askee q) =>
   (REQUEST t askee
      (IF (B q)
          (INFORM (+ t 1) asker q)))
   (REQUEST t askee
      (IF (B (NOT q))
          (INFORM (+ t 1) asker (NOT q))))
```

Explanation:

`query_whether` expects either a confirmation or a disconfirmation of a fact.

زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۷ از ۱۰)

Now, to define the airline agent,
we define its (1) initial beliefs, (2) capabilities, and (3) commitment rules.

Initial Beliefs:

Concerning the flight schedule:

```
(time (flight from to number))
```

And the number of seats available:

```
(time (remaining_seats time1 flight_number seats)
```

زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۸ از ۱۰)

Now, to define the airline agent,
we define its (1) initial beliefs, (2) capabilities, and (3) commitment rules.

Capabilities:

These are issuing boarding passes and updating the count of the available seats on flights.
So the capability database contains two items:

```
((issue_bp ?a ?flight ?time) true)

((DO ?time
     (update_remaining_seats ?time1 ?flight_number ?additional_seats))
 (AND
    (B (?time
          (remaining_seats ?time1
                ?flight_number ?current_seats)))
    (?current_seats >= |?additional_seats|)))
; update_remaining_seats is a private action
; changing the belief about remaining_seats.
```

زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۹ از ۱۰)

Now, to define the airline agent,
we define its (1) initial beliefs, (2) capabilities, and (3) commitment rules.

Commitment Rules:

```
(COMMIT
    (?pass REQUEST
           (IF (B ?p) (INFORM ?t ?pass ?p)))
    true
    ?pass
    (IF (B ?p) (INFORM ?t ?pass ?p)))
```

```
(COMMIT
   (?cust REQUEST
      (issue_bp ?pass ?flight ?time))
   (AND
      (B (?time (remaining_seats ?flight ?n)
      (?n > 0)
      (NOT ((CMT ?anyone)
           (issue_bp ?pass ?anyflight ?time))))
(myself
   (DO (+ now 1)
       (update_remaining_seats
       ?time ?flight -1)))
   (?cust (issue_bp ?pass ?flight ?time)))
```

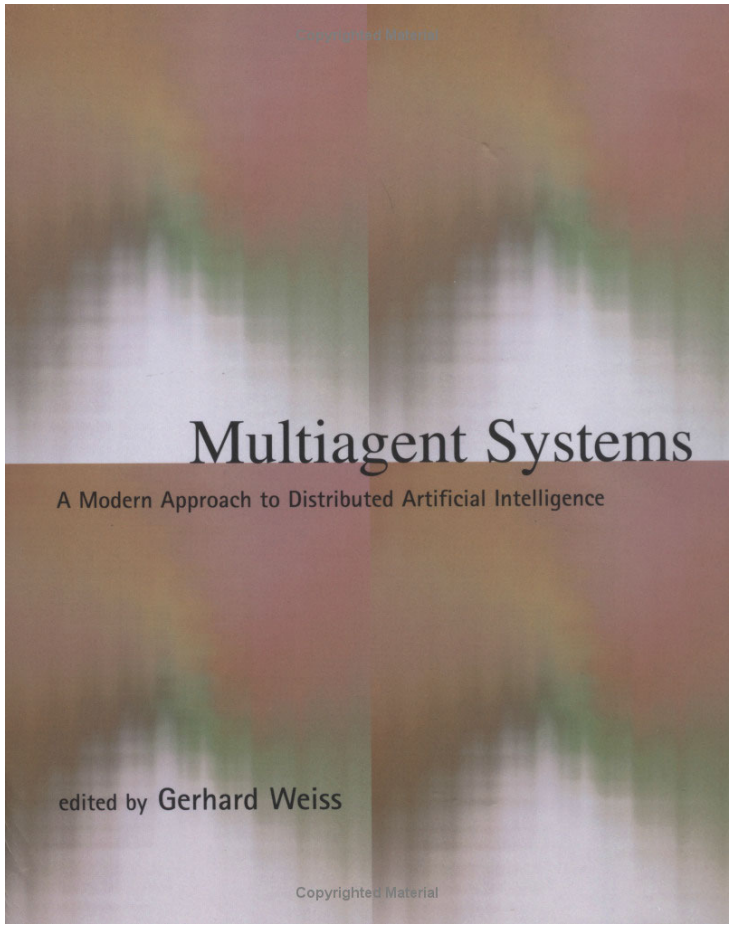
زبان برنامه‌نویسی عامل‌گرا AGENT-0

مثال (۱۰ از ۱۰): نمونه‌ی مبادلات بین دو عامل مسافر و آژانس هواپیمایی

```
agent     action
-------------------------------------------------------------
smith     (query_which lmarch/1:00 smith airline
             (18april/?!time (flight sf ny ?!num)))
airline   (INFORM lmarch/2:00 smith
             (18april/8:30 (flight sf ny #354)))
airline   (INFORM lmarch/2:00 smith
             (18april/10:00 (flight sf ny #293)))
airline   (INFORM lmarch/2:00 smith
             (18april/...)
smith     (REQUEST lmarch/3:00 airline
             (issue_bp smith #354 18april/8:30))
smith     (query_whether lmarch/4:00 smith airline
             ((CMT smith)
              (issue-bp smith #354 18april/8:30)))
airline   (INFORM lmarch/5:00 smith
             (NOT ((CMT smith)
                   (issue-bp smith #354 18april/8:30))))
smith     (REQUEST lmarch/6:00 airline
             (issue-bp smith #293 18april/10,.00))
Smith     (query-whether lmarch/7:00 smith airline
             ((CMT smith)
              (issue-bp smith #293 18april/10:00)))
airline   (INFORM lmarch/8:00 smith
             ((CMT smith)
              (issue-bp smith #293 18april/10:00)))
...
smith     (INFORM 18april/9:00 airline (present smith))
airline   (DO 18april/9:00 (issue-bp smith #293 18april/10:00))
```

منابع

منبع اصلی



Gerhard Weiss (ed.),
Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence,
MIT Press, 1999.
Chapter 1 (1.5)

1 Intelligent Agents

Michael Wooldridge

1.1 Introduction

Computers are not very good at knowing what to do: every action a computer performs must be explicitly anticipated, planned for, and coded by a programmer. If a computer program ever encounters a situation that its designer did not anticipate, then the result is not usually pretty—a system crash at best, multiple loss of life at worst. This mundane fact is at the heart of our relationship with computers. It is so self-evident to the computer literate that it is rarely mentioned. And yet it comes as a complete surprise to those encountering computers for the first time.

For the most part, we are happy to accept computers as obedient, literal, unimaginative servants. For many applications (such as payroll processing), it is entirely acceptable. However, for an increasingly large number of applications, we require systems that can *decide for themselves* what they need to do in order to satisfy their design objectives. Such computer systems are known as *agents*. Agents that must operate robustly in rapidly changing, unpredictable, or open environments, where there is a significant possibility that actions can *fail* are known as *intelligent agents*, or sometimes *autonomous agents*. Here are examples of recent application areas for intelligent agents:

- When a space probe makes its long flight from Earth to the outer planets, a ground crew is usually required to continually track its progress, and decide how to deal with unexpected eventualities. This is costly and, if decisions are required *quickly*, it is simply not practicable. For these reasons, organisations like NASA are seriously investigating the possibility of making probes more autonomous—giving them richer decision making capabilities and responsibilities.
- Searching the Internet for the answer to a specific query can be a long and tedious process. So, why not allow a computer program—an agent—do searches for us? The agent would typically be given a query that would require synthesising pieces of information from various different Internet information sources. Failure would occur when a particular resource was unavailable, (perhaps due to network failure), or where results could not be obtained.

This chapter is about intelligent agents. Specifically, it aims to give you a thorough

Shoham, "Agent-oriented Programming"

§1. Introduction

Present a programming paradigm promoting a social view of computing, where "agents" interact.

§1.1. What is an Agent?

An agent is any entity whose state is viewed as consisting of mental components (e.g., beliefs, capabilities, choices, and commitments).

So agenthood is in the mind of the programmer.

While anything can be viewed as having mental states, it's not always advantageous to do so.

§1.2. On the Responsible Use of Pseudo-mental Terminology

Elements required to ascribe a given quality to a component of a machine:

- a precise theory regarding the mental category: a semantics that's clear yet close to the ordinary use of the term;
- a demonstration that the component obeys the theory; and
- a demonstration that the formal theory plays a nontrivial role in analyzing or designing the machine.

The correspondence of the formal theory to common sense needn't be exact.