فهرست مطالب :

Shoham, “Agent-oriented Programming”

1—مقدمه

اکنون یک الگوی برنامه نویسی پیشرفته یک دیدگاه اجتماعی از کاربرد کامپیوتر،جایی

که "عامل" تاثیر می گذارد داریم.

1—1— عامل چیست؟

یک عامل هر موجودیتی است که حالت است به نظر می رسد به عنوان شامل اجزای

ذهنی (برای مثال گمانها،قابلیتها،انتخابها و الزام ها)

بنابراین کلاه خود عامل هست در مغز یک برنامه نویس

زمانی که هر چیزی میتواند به نظر برسد مثل داشتن حالت های ذهنی

آن همیشه مزیتی برای انجام ندارد.

1–2– مسئول استفاده از واژگان شبه ذهنی :

عناصر لازم برای نسبت دادن یک کیفیت معین به جزیی از ماشین.

*یک نظریه مفید راجعبه دسته ذهنی: معنا شناسی برای آنکه به طور واضح هنوز به استفاده متداول(واژه)نزدیک نشدیم.

*یک برهان که فقط از نظریه اطاعت میکند.

*یک برهان که نظریه قراردادی یک نقش غیر جزئی در آنالیزیا طراحی ماشین بازی میکند.

ارتباط نظریه نظریه قراردادی برای دریافت معمولی لازم نیست تحمیل شود.

1–3–AOPدر برابر OOP

استفاده میکنیم شکل دادن ذهنی را برای طراحی سیستم محاسباتی

دسته های ذهنی(روانی)در زبان برنامه نویسی ظاهر میشود.

معنای زبان برنامه نویسی ’معنای ساختار های ذهن را شرح می دهد.

نرم افزار برنامه نویسی عامل گرا(AOP) نرم افزار برنامه نویسی شئ گرا(AOP)را

اختصاصی کرده است. برای مثال در دریافت بازیگرهای هپوئیت:

میبینیم یک سیستم محاسباتی به عنوان ترکیب شده از لحاظ ارتباط ماژولها.

AOP ثابت می کندحالت ذهنی ماژول (عامل) را برای شامل شدن اجزاء مثل باورها- قابلیت ها و تصمیمات.

یک محاسبه ی شامل این عامل ها آگاه می کنند، درخواست میکنند، می پذیرند، رد می کنند، رقابت می کنند وبه دیگری کمک می کنند.

بر طبق گفته نظریه نمایش هر نوع از ارتباط عمل با پیش فرض های متفاوتی سروکار و اثرات متفاوتی دارد.

جدول 1 (AOPدر برابر OOP)

AOP	OOP	
عامل	شئ	واحد اصلی
گمانها، تعهدها، انتخاب ها و...	بدون محدودیت	حالت تعریف پارامترهای واحد اصلی
پیامها صادر می شوند و به متدها پاسخ می دهند.	پیامها صادر می شوند و به متدها پاسخ می دهند.	مرحله محاسبه
اطلاع دادن،درخواست دادن،پیشنهاد دادن،قول دادن	بدون محدودیت	نوع پیام ها

،کاستن		
پایداری/ درستی	ندارد	محدودیت روی روش ها (متدها)

2– دو سناریو:

اولین سناریو،پیچیده است– نوع رویایی برنامه ی کاربردی

دومین سناریو مثال اسباب بازی است که در خدمت سه هدف است:

- به تصویر کشیدن قطعی ایده های چندین AOP
- آن در زبان AGENT-0 قابل اجراء است که بعداً تعریف می شود.
- ان حقایق عامل ها را به تصویر می کشد. نیاز به عامل های رباتیک ندارد.

2–1–ساخت خودکارسازی (اتوماسیون)

عامل ها:

- آلفرد ماشین های منظم و معین را اداره می کند.
- براندا ماشین های خاص– منظم را اداره میکند.

- کالوین ربات جوشکاری است.
- راشیل یک برنامه هماهنگ کننده است که ماشین را کنترل می کند.

- (8:00) : آلفرد درخواست می کند که کالوین قول دهد که 10 بدنه را برای او، ان روز جوشکاری کند.

کالوین نیزموافق به انجام آن است.

- (8:30) : آلفرد درخواست می کند که کالوین قول دهد اولین بدنه را برساند ، کالوین قبول می کند و اولین بدنه می رسد.

کالوین شروع می کند به جوشکاری آن و قول میدهد به آلفرد برای اینکه وقتی بدنه بعدی آماده بود، او را آگاه کند.

- (8:45) : براندا درخواست می کند که کالوین روی ماشین خاص- منظم کار کند، زیرا که به فوریت نیاز دارد.

کالوین پاسخ می دهد که در آن هنگام نمی تواند درست کند.اما آن کار را وقتی خواهد کرد که کارش تمام شده باشد.

- (9:05) : کالوین جوشکاری اولین ماشین آلفرد را کامل می کند، آن را بیرون می فرستد، به براندا پیشنهاد جوشکاری او را می دهد.

براندا ماشین را می فرستدو کالوین شروع به جوشکاری می کند.

- (9:15) :آلفرد تحقیق می کند که چرا کالوین هنوز برای ماشین بعدی او(آلفرد) آماده نیست.
- (9:55) : کالوین جوشکاری ماشین برندا را کامل می کند و آن را بیرون می فرستد.

براندا درخواست میکند کالوین دوباره قبول کند و تعدادی نقاشی انجام دهد. ولی کالوین نمی پذیرد و می گوید که چگونه نقاشی بکِشد.

سپس کالوین به آلفرد پیشنهاد می دهد که ماشین دیگرش را جوشکاری کند و برای یک مدتی به جوشکاری ماشین آلفرد مشغول می شود.

- (12:15) :برندا درخواست می کند که کالوین برای جوشکاری 4 ماشین او در ان روز وارد عمل شود.

کالوین پاسخ می دهد که نمی تواند ان کار را انجام دهد. زیرا از وقتی که به آلفرد قول داده، هنوز 6 ماشین را جوشکاری نکرده است.

برندا از آلفرد درخواست می کند که کالوین را از تعهدش نسبت به او(آلفرد) آزاد کند.ولی آلفرد نمی پذیرد.

برندا درخواست می کند که داشیل (داشیل را به یاد دارید؟)به کالوین برای پذیرفتن درخواست مهم او و لغو کردن تعهدش نسبت به آلفرد دستور دهد.

داشیل به کالوین دستور می دهد که 2 ماشین برندا را جوشکاری کند و سپس به کار آلفرد بپردازد.

- (3:30) : کالوین حس می کند که روغنش کم شده است و بدون قطع کردن کارش ازآلفرد میپرسد برای چه وقتی روغن کاری بعدی او برنامه ریزی شده است.

آلفرد می گوید که نمیداند، ولی داشیل باید بداند.

کالوین از داشیل می پرسد. داشیل میگوید که روغن کاری بعدی او برای ساعت 4:00 برنامه ریزی شده است.

با توجه به اینکه هنوز زود است، کالوین به جوشکاری ادامه میدهد.

2–2– سیستم رزو خطوط هوایی :

عامل ها:

- P : مسافر
- C : کارمند هواپیمایی (یک برنامه)
- S : ناظرC

به وسیله مطبقت یک رزو، سیستم هواپیمایی در یک تعهد برای صادر کردن یک گذرنامه برای یک مسافر در زمان مناسب وارد می کند.

MARCH

P to C : لطفاً به من بگویید که شما چه پروازی از سانفرانسیسکو به نیویورک در 18 آپریل دارید.

C to P : پرواز شماره 354 ذر ساعت 8:30 حرکت می کندو پرواز شماره 293 در ساعت 10:00 حرکت می کند و پرواز شماره 441 در ظهر ...

P to C : لطفاً برای من در شماره 354 رزو کنید.

C to P : از آن دیگر نداریم.(همه به فروش رفته است.)

P to C : لطفاً در شماره 293 رزو کنید.

C to P : تایید شد. شماره رزو شما 112358 است.

P to C : لطفاً برای من در پرواز شماره 441 نیز رزو کنید.

C to P : آن پرواز با شماره 293 تداخل می کند. من اجازه ندارم برای یک مسافر 2 رزو کنم.

P to C : لطفاً برای انجام این کار مجوز بگیرید.

C to S : من برای ادامه رزو مضاعف درخواست مجوز می کنم.

S to C : مجوز داده نمی شود.

C to P : متاسفم، من نمی توانم تایید بگیرم.

April 18 ؛ درفرودگاه:

P to C : اسم من P است. من یک رزو برای پرواز شماره 293 دارم.

C to P : این گذرنامه شما است، بفرمایید.

3– برداشت کلی از نرم افزار AOP :

یک سیستم AOP کامل شامل 3 جزء اولیه است:

1- یک زبان قراردادی انحصاری (که شامل چندین بعد، مثل گمان ها والزام ها است) با گرامر واضح و معنایی برای توصیف حات های ذهنی ؛

2- یک زبان برنامه نویسی تفسیری برای تعریف و برنامه نوشتن عامل ها با دستورات اولیه (برای مثال: REQUEST، INFORM)

3- یک "Agentifier"؛ دستگاه های بی اثر را به عامل های قابل برنامه نویسی تبدیل می کند.

- جزء 2 هدف اصلی این مقاله است.
- ان به جزء 1 تکیه می کند.

♦ جزء 3 باقیمانده نسبتاً مبهم است.

4– دسته های ذهنی و مشخصه های آنها:

آنجا نه انتخاب صحیح از دسته های ذهنی و نه یک تئوری صحیح از آنها است.

4–1– اجزاء حالت ذهنی:

انجام کار یک عامل به وسیله تصمیمات او یا انتخاب های او تعیین مشود.

تصمیمات به وسیله گمان های عامل محدود می شوند که بعداً اشاره خواهد شد.

- حالت جهان
- حالت های عامل های دیگر
- قابلیت های این عامل و عامل های دیگر

هم چنین تصمیمات به وسیله تصمیمات قبلی محدود می شوند.

بنابراین ما دو دسته ی ذهنی معرفی می کنیم ، گمان و تصمیم، و دسته ی سوم، قابلیت ها، (که نه به خودی خود ذهنی است) .

بهتر است یک تصمیم مثل پایه بگیریم. ما با الزام ها یا تعهدها شروع می کنیم و با تصمیمی مثل تعهد با خود او رفتار می کنیم.

4-2- یک زمان برای گمان، تعهد و قابلیت زمان:

ما گمان می کنیم اشیاء هردو مورد،حدود زمان های متفاوت و در زمان های متفاوت، هستند.

به علاوه برای ابعاد دیگر.

ما یک نقطه- زبان موقتی به عنوان پایه در نظر می گیریم.به عنوان مثال:

```
holding(robot,cup)t
```

یعنی : " روبات، فنجان را در زمان t نگه داشته است."

عمل

ما بین عمل ها و حقایق تمایزی قائل نمی شویم .

رویداد یک عمل به وسیله حقیقت مشابه "نگه داشتن" نمایش داده می شود.

به عنوان مثال به جای گفتن اینکه بازوی بالا برنده را در زمان می گیرد، میگوییم درست که عبارت `raise-arm(robot)t` است.

(برای نگه داشتن وسیله بعد از عمل، یک تصمیم معرفی می کنیم.)

نظر به اینکه عمل ها یک حقیقت و لحظه ای می باشند.

گمان

استفاده از عملگر نمایی B

$B_a^t \varphi \Rightarrow$ " در زمان t عامل a،φ را گمان می کند. "

جایی که φ یک جمله است. (تعریف شده بازگشتی)

برای مثال : $B_a^3 B_b^{10} like(a,b)^7$

یعنی: "در زمان 3 عامل a گمان می کند که در زمان 10 عامل b گمان خواهد کرد که در زمان 7 a liked b ."

تعهد

$OBL_{a,b}^t \varphi$

یعنی:"در زمان tعامل a متعهد است به عامل b درباره یφ"

تصمیم(انتخاب)

تصمیم برای تعهد به خودش تعریف شده است.

$DEC_a^t \varphi =_{def} OBL_{a,a}^t \varphi$

توانایی

$$CAN_a^t \varphi$$

یعنی : "در زمان t عامل a توانا به φ است"

برای مثال ،

$$CAN_{robot}^{5} open(door)^{8}$$

یعنی: "در زمان 5 روبات می تواند مطمئن شود که در، در زمان 8 باز است."

ABLE مدل بی درنگ از CAN است.

زمان درنگ φ دورترین زمان اتفاق افتادن در جمله φ است،

$$ABLE_a \varphi \quad =_{def} \quad CAN_a^{time(\varphi)} \varphi$$

4-3- مشخصات اجزاء مختلف

پایداری داخلی

فرض می کنیم که هر دوی گمان ها و تعهد ها به طور داخلی پایدارند:

- For any t, a: $\{\varphi : B_a^t \varphi\}$
- For any t, a: $\{\varphi : OBL_{a,b}^t \varphi \text{ for some } b\}$

هر دو مورد فوق پایدارند.

عقیده ی خوب

عامل ها به کار برده می شوند فقط برای آنچه آنها گمان می کنند خود توانا هستند و فقط اگر آن را معنی می کنند.

- For any t, a, b, φ: $OBL^t_{a,b}\varphi \rightarrow B^t_a((ABLE_a\varphi) \wedge \varphi)$

درون نگری

عامل ها از تعهدشان آگاه هستند.

- For any t, a, b, φ: $OBL^t_{a,b}\varphi \Leftrightarrow B^t_a OBL^t_{a,b}\varphi.$
- For any t, a, b, φ: $\neg OBL^t_{a,b}\varphi \Leftrightarrow B^t_a \neg OBL^t_{a,b}\varphi.$

ولی فرض نمی کنیم که عامل ها نیاز دارند از تعهدات ساخته شده در برابر آنها نیز آگاه باشند.

ماندگاری حالت ذهنی

ملاحظه به اینکه چگونه حالت های ذهنی ایستادگی یا تغییر می کنند.

گمان ها ایستادگی به وسیله پیش فرض : عامل ها حافظه ی کامل از گمان هایشان دارند؛یک گمان افتاده است فقط زمانی که یک حقیقت متناقض فهمیده است.

همچنین فقدان گمان ایستادگی کردن به وسیله پیش فرض است.

تعهد ها

ایستادگی کردن فرضی، اما آنجا شرایطی است که تحت آنها لغو شده اند.

برای مثال:

- آزاد کردن صریح عامل توسط گروه، برای آنکه آنها متعهد هستند.
- ادراک به وسیله عامل که در آن به مدت طولانی نمی تواند تعهد را انجام دهد.
- نظر به اینکه تصمیم در شرایط تعهد تعریف شده است، آن جایگزین پایداری پیش فرض میشود.

مادامی که یک عامل نمی توان تعهد به دیگران را به طور یکجانبه لغو کند،او می تواند تعهدات به آن را رد کند – شامل تصمیم ها

- فرض می کنیم که توانایی ها ثابت شده اند.

نوع وابسته ی دستورات نمایی

پرش

4–4– یک انحراف کوتاه : مقایسه بین کوهن و لوسکیو

پرش

5– مفسر عامل عمومی:

نقش یک برنامه عامل، کنترل تحول یک حالت ذهنی است.

عمل رخ می دهد. مثل اثرات جانبی عاملِ تعهد داده شده به یک عمل چه زمانی می آید.

حلقه اصلی

هر عامل دنباله از مراحل را در فاصله ی منظم تکرار می کند.

1) پیام جاری را بخوان و حالت ذهنی را به روز کن، شامل گمان ها و تعهدات

(برنامه عامل برای به روز رسانی بسیار بحرانی است.)

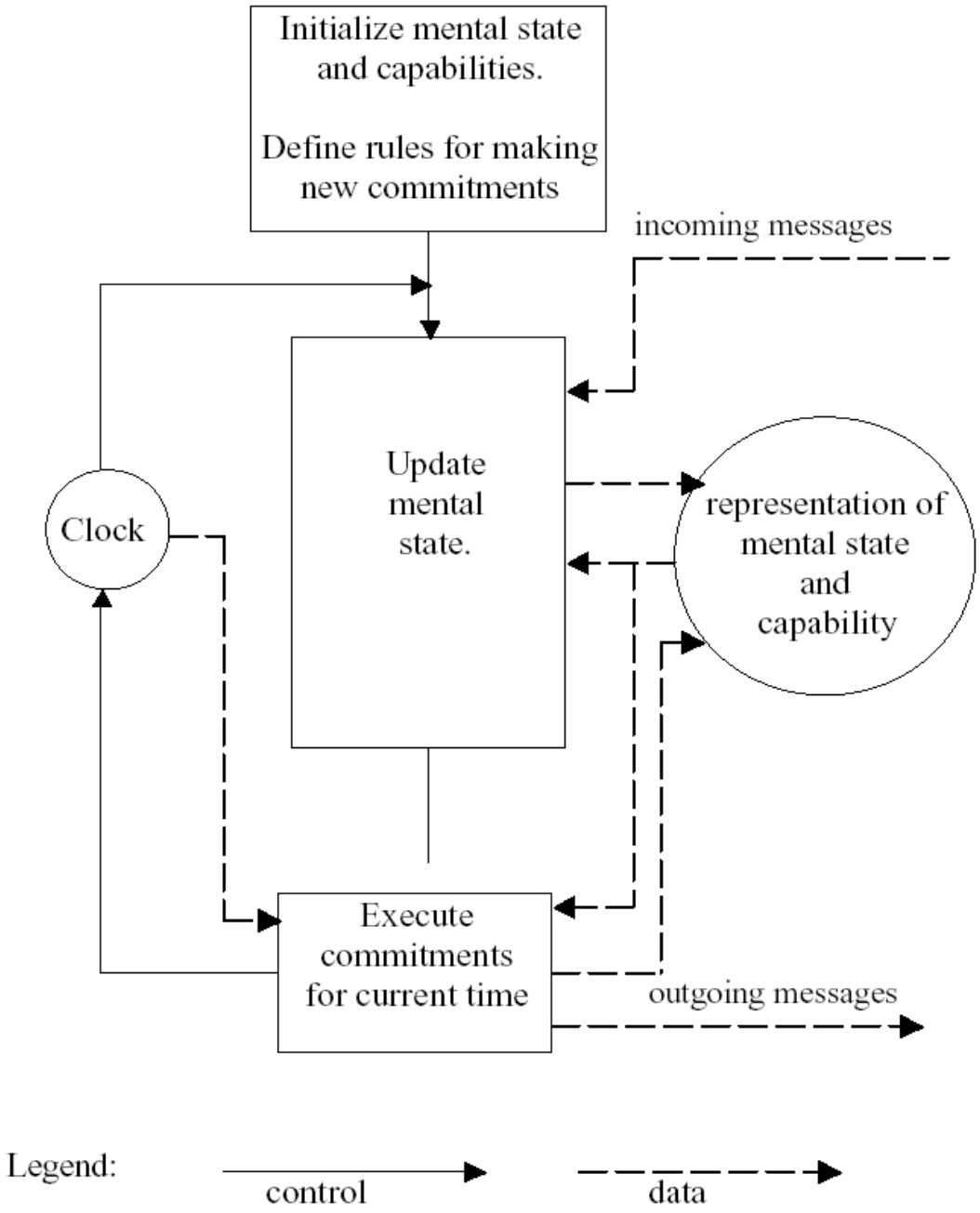
2) تعهدها برای زمان جاری را اجرا کن، شاید در تغییر کمان بیشتر نتیجه می گیرد.

(این کار مستقل از برنامه عامل است)

عمل هایی که برای آنکه عامل ها بتوانند متعهد شوند عبارتند از:

- عمل های گویا (برای مثال: آگاهی دادن و درخواست کردن)
- عمل های "اختصاصی" اختیاری

شکل را ببینید:



فرض در مورد انتقال پیام

فرض می کنیم که پایگاه می تواند پیام ها را به عامل های دیگر بفرستد.آدرس پذیر

توسط نام.

مفسر وقتی پیام ها فرستاده می شود تصمیم می گیرد.

فرض در مورد clock

Clock از سر گیری 2 مرحله در فاصله های منظم را راه اندازی می کند.

طول این فاصله به وسیله ی متغیر پایدار(time grain) معین شده است.

فرض می کنیم یک متغیر اکنون که متغیر نصب شده به وسیله کلاک به زمان جاری .

فرض می کنیم که تکرارها تنها میان حلقه کمتر از خرده زمان.(یک فرض بسیار قوی)

هم زمان سازی برای کار صحیح یک جامعه از عامل ها بحرانی است.

6– AGENT-0، یک زبان ساده؛ و پیاده سازی آن

6–1 گرامر AGENT-0

زبان خودش فقط شرایط را برای ساختن تعهدها تعیین می کند.

تعهدها ساخته می شوند. و بعداً در زمان مناسب انجام می دهد.

تعهدات فقط برای عمل های ابتدایی هستند. (به طور مستقیم توسط عامل اجرا می شوند.)

بنابراین یک عامل نمی تواند متعهد به انجام شدن هر شرطی که برنامه ریزی را درخواست می کند، شود.

دستورات حقیقی

دستورات حقیقی برای تعیین کردن هر دو محتوای عامل ها و شرایط برای اجرای آنها استفاده می شوند.

جمله های هدف atomic زبان موقتی توصیف شده در بالا:

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

دستور عمل های گویا و خصوصی

عمل ها ممکن است خصوصی یا گویا باشند.

گرامر برای یک عمل خصوصی (برای مثال به وسیله یک عامل پایگاه داده یا یک روبات)

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

جایی که t است یک نقطه ی زمان و p-action یک نام عمل خصوصی است.

عمل های خصوصی ممکن است IO شوند.

عمل های گویا همیشه برای IO و معمولاً برای همه ی عامل ها درگیر می شوند.

AGENT-0 فقط شامل 3 نوع عمل گویا است:

❖ گرامر **اطلاع کردن** است:

```
(INFORM t a fact)
```

جایی که t یک نقطه زمان است ، a یک نام عامل و fact یک دستور حقیقی است.

❖ گرامر **درخواست کردن** است:

```
(REQUEST t a action)
```

جایی که عمل یک دستورعمل است، تعریف شده ی بازگشتی می باشد.

برای مثال :

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

❖ گرامر **حذف** یک درخواست است:

```
(UNREQUEST t a action)
```

هم چنین می توانیم از تعهد به یک عمل به خصوص جلوگیری کنیم.

```
(REFRAIN action)
```

دستورات عمل شرطی :

ما وجه تمایز قائل می شویم بین :

- تعهدات برای عمل های شرطی، که شامل شروط تست شده فقط قبل عمل کردن
- شروط برای وارد کردن به تعهدات در مرتبه ی اول (در زیر می بینید)

یک عمل شرطی به یکئ شرط ذهنی تکیه می کند، که اشاره به حالت ذهنی عامل می کند.

هنگامی که زمان برای اجرای عمل وارد کار می شود،حات ذهنی در آن زمان آزمایش می کند تا ببیند آیا شرط ذهنی راضی است.

بنابراین عامل و اجزاء زمان حالت ذهنی در چپ به طور ضمنی هستند.

شرط ذهنی،بنابراین،هر ترکیب دستورات نمایی در زبان موقتی نمایی،با عامل اولیه و دستورات حذف شده می باشد.

به طور ویژه ،یک شرط ذهنی یک ترکیب منطقی از الگوی ذهنی نرم است.

```
(B fact) or ((CMT a) action)
```

CMT به همان معنی OBL است.

برای مثال :

```
(B (t (employee smith acme)))
```

گرامر یک عمل شرطی به صورت زیر است:

```
(IF mntlcond action)
```

برای مثال:

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

شروط ذهنی امکان دارد شامل یک رابط AND یا NOT باشد.

برای مثال، دنباله ی3 دستور ، تشکیل یک پرسش و پاسخ درباره ی چه حقیقتی است. (b

پرس وجو شده و سئوال می شود برای آگاه کردن 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))))))
```

متغیرها

رویه ها در یک مدل الگوی مستقیم لغو می شوند.

قوانین تعهد به وسیله الگوهای معین (به طور کلی با متغیرها سروکار دارد) در پیام های

آمده و حالت ذهنی فعال می شوند.(در پایین می توانید مشاهده کنید):

یک متغیر به وسیله ی "؟" می آید.

ممکن است آن روی نام های عامل، دستورات حقیقی یا دستورات شرطی

تغییر دهد.

برای مثال :

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

متغیر ها در دستور های شرطی به عنوان واجد شرایط وجودی تفسیر شده اند.

دامنه کمیت سنج برای دامنه اولین NOT رو به بالا است یا (اگر متغیر در دامنه یک

NOT نباشد.) آن دستور کامل نیست.(آخرین مثال را ببینید.)

یک متغیر کممیت عمومی با "!؟" می آید، دامنه ی آن دستور العمل کامل است.

برای مثال :

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

قوانین تعهد

بیشتر دستورات شرط در زمان برنامه نویسی، نا شناخته هستند.

آنها به وسیله ی عامل های ذیگر مرتبط می شوند.

برنامه فقط شامل شروط برای عامل، برای وارد کردن به تعهدات جدید

است.

بیشتر تعهدات برای پاسخ به پیام ها هستند.

شروط برای تعهدات هر دو شرط ذهنی (بالا را ببینید) و شروط پیام (اشاره

می کند به پیام های آمده ی جاری) است.

یک شرط پیام ، الگوی پیام یک شرط است.

```
(From Type Content)
```

جایی که

- `Form` اسم فرستنده است.
- `Type`؛ `INFORM` ، `REQUEST` یا `UNREQUEST` است.
- `Content` یک دستور واقعی است یا یک دستورالعمل می باشد که بستگی به `Type` دارد.

دیگر اطلاعات با یک پیام (مقصد و زمان ورود) شرکت داده می شوند.

به عنوان مثال:

```
(a INFORM fact)
```

یعنی: یکی از پیام های جدید عامل Fact را به a اطلاع می دهد.

و

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

یعنی: پیام از a درخواست می کند برای راه رفتن عامل و نه درخواست جدید از هر کسی که برچسب عامل دارد.

گرامر یک قانون تعهد :

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

(دستور عمل ممکن است شامل شرط ذهنی خودش شود.)

برای مثال :

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

برنامه ترتیبی از قونین تعهد است، جلوتر به وسیله ی یک تعریف از تونایی های عامل و گمان اولیه و ثابت شدن خرده زمان.

در دنباله گرامر AGENT-0 برای BNF داریم .

```
<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>
```

6–2–مفسر AGENT-0

نظر به اینکه مفسر AGEN-0 یک نمونه از مفسر عمومی است، جایگزین 2 مرحله

ی طراحی حلقه می شود.

در AGENT-0 ، حالت ذهنی مرکب از 3 جزء است.

یک- توانایی ها- ثابت شده اند.

بنابراین اولین مرحله در حلقه ممکن است به عنوان های زیر اختصاصی شود.

(1a) به روز رسانی گمان ها

(2a) به روز رسانی تعهدات

در AGENT-0، گمان ها، تعهدات و توانایی های عامل هر کدام به وسیله پایگاه

داده به نمایش در می آیند.

به روز رسانی گمان ها

پایگاه داده گمان هیچ کدام را به روز رسانی نمی کند.

الف) به عنوان یک نتیجه آگاه می کند.

ب) به عنوان یک نتیجه، یک عمل خصوصی می گیرد.

1. یک عامل پایگاه داده به گمان کردن یک حقیقت بعد از انجام یک عمل کردف می آید.
2. یک عامل روبات به گمان کردن تعدادی بعد از انجام یک رویه بصری می آید.

ما در a بهره می بریم.

یکسان کردن یک حقیقت جدید φ در یک پایگاه داده ی موجود Γ، ملاحظه می کنیم.

چک کردن پایداری برای تئوری های بدون محدودیت (پایگاه داده) هیچ یک سخت (در حالت گذاره ای) یا تصمیم نا پذیر (در حالت اول- منظم) نیست.

اگرφ با Γ متناقض باشد، بیشتر تئوری های یکسان سازی نیاز دارند که Γ برای بازیابی پایداری اندکی اصلاح شوند- این یک زوج سخت است.

2 هدف زیر برای گرفتن اطراف پیچیدگی وجود دارند:

1. نیازمندی ها را تخفیف دهیم. یک الگوریتم یکسان سازی ابتکاری بگیریم که بین درستی یا تمامیت مصالحه کند.

2. [در AGENT-0 گرفته شده] جملات را محدود می کند، بنابراین مشکل مهار شدنی می شود.

AGENT-0 ارتباط متفوت خنثی سازی را رد می کند- پایداری بیشتر خطی ها در اندازه ی پایگاه داده را چک می کند.

(این به عملگر های نمایی رد شده اضافه می کند، به گمان های تودرتو نیاز داریم.)

در انجا باقیمانده ی سوال که چگونه اطلاعات جدید را تشخیص می دهیم سرانجام به یک تئوری نفوذ نیاز داریم.

عامل های AGENT-0 هر حقیقتی را که شما می گویید ترکیب می کند. گمان های atomic متناقض را اگر شما نگه داشته باشید، جمع میکند.

به روز رسانی تعهدات

آیتم ها در پایگاه داده ی تعهدات، جفت هستند.

عاملی که به آن متعهد شده و محتوای تعهد (agent action)

آیتم ها در پایگاه داده ی توانایی ها جفت هستند.

```
(Privateaction mntlcond).
```

بخش شرط ذهنی ار تعهد به عمل های ناسازگار جلوگیری می کند.

برای مثال:

```
((?!time (rotate wheelbase ?degrees))
 (NOT ((CMT ?x) ?!time (service wheelbase))))
```

تعهدات موجود هیچ یک را حذف نمی گنند.

آ) به عنوان یک نتیجه از پیام های UNREQUEST

عامل آیتم های مترادف از پایگاه داده ی تعهد را اگر موجود باشد، حذف می کند. در غیر این صورت کاری نمی کند.

یا

ب) به عنوان یک نتیجه از تغییر گمان.

تغییر گمان ممکن است روی توانایی ها اثر بگذارد. نظر به اینکه توانای هر عمل به خصوص به پیش شرط های ذهنی بستگی دارد.

او ان پیش شرط ها را در پایگاه داده ی توانایی که تخلف کرده است، حذف میکند.

او باید یک تعهد اضافه کند تا به عاملی که به آن متعهد شده است،اطلاع دهد.

(ولی AGENT-0 این کار را نمی کند.)

حذف تعهدات موجود به برنامه بستگی ندارد، ولی اضافه کردن آنها به برنامه بستگی دارد.

الگوریتم اضافه کردن تعهدات برای هر دستور تعهد برنامه

(COMMIT msgcond mntlcond (ai actioni)*)

اگر

- msgcond، پیام های آمده ی جدید را نگه می دارد.

- ❖ Mntlcond، حلت ذهنی جاری را نگه می دارد.
- ❖ برای همه ی i ها،عامل به طور جاری به $action_i$ توانا است.
- ❖ برای همه ی i ها
  - عامل ها متعهد به برگردان $action_i$ نیستند.
  - اگر $action_i$ خودش برگردان $action_i$ REFRAIN باشد عامل به $action_i$ متعهد نیست.

سپس برای همه ی i ها ، متعهد به a_i برای برگردان $action_i$

شروط برای یک عامل برای توانا شدن آن به انجام عمل است.

- ❖ یک عامل می تواند هر چیزی را از هر کسی درخواست بدهد یا ندهد.
- ❖ یک عامل می تواند هر کسی را برای یک حقیقتی که گمان می کند، آگاه کند.

یک عامل ابداً نمی تواند برای هر حقیقت خودش را آگاه کند (مورد استفاده برای استدلال پیاده سازی در عامل).

- ❖ یک عامل قادر است هر عمل خصوصی در پایگاه داده ی توانایی را فراهم کند شرط ذهنی شرکت پذیر با ان عمل خصوصی را به وسیله پایگاه داده ای که در آن زمان را نگه می دارد.
- ❖ یک عامل می تواند از هر عملی ککه برای تعهد به آن آماده نیست، برگردان کند.
- ❖ یک عامل می تواند عملی شرطی را انجام دهد

(IF mntlcond action)

اگر او بتواند action را تحت شرط mntlcond انجام دهد.

انجام تعهدات

هر تعهد در پایگاه داده ی تعهد یک زمان شرکت پذیر دارد.

در این مرحله دوم، مفسر همه عمل هایی را که زمان در وقفه است، اجرا کند.

(now - timegrain, now].

معنی "execute" به نوع عمل بستگی دارد.

- Inform،REQUEST ، UNREQUEST: پیام اختصاص دادن را می فرستد.
- REFRAIN: هیچ اثری روی اجزاء ندارد.
- DO: به گمان و پایگاه داده های تعهد همفکری می کند، شرط ذهنی شرکت با شرط اولیه در پایگاه داده توانایی را چک می کند. اگر آن را نگه داشته باشد،سپس (به طور بازگشتی) عمل را اجرا می کند.

6–3یک برنامه ی ساده و تفسیر آن

اکنون یک پیاده سازی برنامه ی یک مدل ساده شده از هواپیمایی که در بخش 2 بود.

ایده ی پشت برنامه این است:

- فعالیت مناسب روی بخشی از هواپیمایی که گذرنامه را برای مسافر صادر می کند.
- تایید یک رزو یک تعهد به صورت یک گذرنامه در زمان مناسب است.

ما اول چند ما تعریف می کنیم

```
(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)))
```

- **توضیح:** صدور یک گذر نامه به طور دقیق 1 ساعت قبل از پرواز است.

```
physical_issue_bp
```

یک عمل خصوصی شامل تعدادی رویداد خارجی است.

```
(query_which t asker askee q) ⇒
  (REQUEST t askee
           (IF (B q) (INFORM (+ t 1) asker q)))
```

- **توضیح:** این درخواست ها فقط یک پاسخ مثبت هستند.

اگر q یک متغیر کمیت عمومی باشد، سپس query-which در خواست می کند از همه ی نمونه های پاسخ به پرس و جوی q آگاه شود.

```
(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))))
```

- **توضیح:** query-whether انتظار هیچ یک از یک تایید یا عدم تایید یک حقیقت را ندارد.

حالا، برای تعریف عامل هواپیمایی، ما یک گمان های اولیه، توانایی ها و قوانین تعهد را تعریف می کنیم.

گمان های اولیه:

درباره ی زمانبند پرواز:

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

و شماره ی صندلی های قابل دسترس :

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

توانایی ها

توانای ها گذر نامه را صادر می کنند و شماره صندلی های قابل دسترس پرواز را به می کنند.

به علاوه پایگه داده توانایی شامل 2 آیتم است:

```
((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.

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)))
```

جدول 2 را ببینید:یک تبادل ساده بین یک مسافر وعامل هواپیمایی .

پیام ها ی مسافر به وسیله ی خودش مشخص شده است.

هواپیمایی به وسیله ی عامل مفسر به آنها (پیام ها ی مسافر) واکنش می دهد.

عامل **عمل**

```
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))
```

6–4– پیاده سازی

پرش

Agentification –7

آزاد کردن سازنده از نیازمندی به پشتیبانی یک حالت ذهنی یک شکاف بین آنها می سازد.

- سطح ارادی برنامه عامل
- ارائه پروسری ماشین نگری یک دستگاه معوم

قانون agetifier (شناسایی عامل) پلی برای این شکاف است.

ما نظریه ماشین های جایگزین kaebling در این تجزیه ارادی و سطوح ماشین.

وجود دارد:

- یم زبان سطح پایین برای توصیف دستگاه
- یک زبان سطح بالا برای طراح تا درباره ی دستگاه استدلال کند.

کامپایلر یک زبان در سطح بالا را می گیرد و یک توصیف از یک دستگاه در زبان سطح پایین تولید می کند.

برای زبان پردازش سطح پایین ما نیازمندیم :

- ارائه ی زمان پردازش، شامل مدت زمان های ارزش واقعی
- پردازش های هم زمان
- سطوح چندگانه از تجرید

او مدل پردازش خود را ارتقاء داده است،نظریه ماشین های موقتی (منابع را ببینید) در agentification

- ورودی به مرتجم شامل یک توصیف از ماشین در زبان پردازش است.
- خرجی یک برنامه ارادی است.

بنابراین کامپایل کردن در نظریه ماشین های جایگزین de- agentification است.

8- کاربرد مربوط

پرش

9- بحث: سیری که در آن نرم افزار را توسعه می دهند.

❖ دسته های ذهنی:

کامل کردن زبان حالت های ذهنی بیشتر شامل مفاهیم پیچیده است.(برای مثال

خواسته، منظورو نقشه)

این اجازه خواهد داشت:

- دستورات ارتباطی قوی تر
- ساخت یافتگی بیشتر روی رفتار عامل

❖ بنای رسته های ذهن:

یک بخشی از محاسبات توزیع شده به تئوری قراردادی از دانش، بنای سخت معنایی

است:

مجوعه ی دنیاهای قابل دسترس، مجموعه حالت های عمومی قایل دسترس از یک

مجموعه پردازش های حالت محدود می شود، یک پروتکل می گیرد.

با agntification ، ما باید به طور یکسان قادر به نگه داشتن گمان تعهد باشیم.

❖ احتمال و سودمندی:

باید بیشتر روی دانش کار کنیم و گمان مفاهیم خشک رفتارهای ذهنی را می پذیرد – هیچ ارثه ای از گمان درجه بندی شده یا تعهد.

این مغایرت با تئوری بازی روی اثر متقابل پذیرفتن عامل های میانی در علم اقتصاد کار می کند و AI جایی که به طور نامعلوم و سودمند نقش کلیدی بازی می کند.

❖ ارث بری و گروه ها:

یک آنالوگ برای ارث بری در دنیای oop باید در "عامل های گروه" در Aop باشد، یک گروه از عامل های تشکیل شده از یک عامل.

اگر ما تعریف کنیم

- گمان های عامل مرکب به عنوان گمان های مشترک عامل های اختصاصی
- تعهدات عامل مثبت به عنوان تعهدات مشترک عامل های اختصاصی

سپس وضعیت های ذهنی گروه نیاز دارند به وسیله عامل های اختصاصی به ارث برده شوند.

❖ ماندگاری حالت های ذهنی:

مکاتبات رسمی با حالت های ذهنی معمولاً سخت تر از مکاتبه با مسئله ی الب بندی شده است.

برای مثال اگر گمان کنیم که شما x را گمان نمی کنید، انجام من گمان می کنم شما x را گمان نمی کنید، به زودی ممکن است؟

❖ محدودیت های منابع:

در تعریف مفسر فرض شده که گمان و تعهد در یک زمان بی اهمیت به روز می شوند.

ولی در بسیاری از برنامه های کاربردی Real-time (بدون درنگ) این فرض نقض می شود.

بنابراین مفسر باید یک تصمیم عاقلانه در میان عملکرد های ذهنی اتخاب کند.

مصالحه در حل مسئله بدون درنگ هوشمند وجود دارد، شامل سبک و سنگین کردن بین کیفیت و مناسب بودن

❖ مدل گمان و به روز رسانی:

AGENT-0 همه ی اطلاعات جدید را می پذیرد.

اما ما باید ملاحظه کنیم که چه خط مشی معقول ، به روز رسانی گمان را تشکیل می دهد.

هر دو معنا و پرسش های الگوریتمی است.

❖ نقشه های گمان موقتی :

AGENT-0 نمی تواند گان های عامل ها درباره ی گمان ها یا تعهدات دیکر عامل ها را نمایش دهد.

AGENT-0 لبه ی آن گمان ها را به وسیله ی نقشه ی زمان و نگهداری نقطه های انتقال و پیش فرض پایداری آنها را حفظ می کند.

AGENT-1 اجازه ی نمایی های تو در تو را در پایگاه داده ی گمان می دهد، از نقشه های زمان اندازه-بالا که نقشه های زمان ذهنی نامیده می شوند ، استفاده می کند.

نقشه های گمان موقتی حالت به خصوصی هستند.

❖ جامعه ها:

ما فقط به عامل های در حال کار به صورت خودکار نگاه کردیم.

اما جامعه ی عامل موفق به نیاز به محدودیت های عمومی دارد. برای مثال :

- جامعه ی قوانین
- جامعه ی نقش ها

هر دوی این ها نیاز حل مسئله ی کاهش می دهند. به وسیله ی عامل ها و سربار ارتباط را.

بدنه ی غنی از مطبوعات در جامعه های کامپیوتر

برای مثال:

- جامعه ی اطلاعاتی Minsky شبیه مغز
- بررسی های winograd از نقش های اجتماعی (انسان و ماشین)
- مباحث Moses و Tnnenhols در مورد مزیت های محاسباتی

قوانین اجتماعی

- کار Doyles روی ارتباط بین AI، روانشناسی منطقی و علم اقتصاد.

Shoham اخیراً رسیدگی کرده به طراحی off-line قانون های اجتماعی که یک تعادل خوب را بین موارد زیر نقض می کنند:

- دادن اجازه ی آزادی مناسب به عامل های اختصاصی.
- به عامل های اختصاصی اجازه ی آزادی مناسب بدهیم

به طور جاری رسیدگی به یادگیری on-line خودکار این قبیل قانون ها وجود دارد.