دانشگاه شیراز – دانشکده آموزشهای الکترونیکی

جزوه درسی

هوش مصنوعی

تهیه کننده : دکتر الهام پروین نیا

پیش نیاز : طراحی الگوریتم ها

تعداد واحد : ۳ واحد نظری

سر فصل های درس

- هوش مصنوعی چیست ؟ تاریخچه هوش مصنوعی و مرزهای دانش در هوش مصنوعی
- عاملها : ساختار و عملکرد ، عاملهای هوشمند ، محیط ها
- حل مسئله ، حل مسئله از طریق جستجو ، فرموله کردن مسائل ، چند مثال جستجو برای جواب
- روش های جستجوی غیر آگاهانه (جستجوی اول سطح ، جستجوی با هزینه ثابت ، جستجوی اول عمق ، جستجوی عمق محدود شده ، جستجوی اول عمق عمیق شونده تکراری ، جستجوی دو طرفه) .
- روشهای جستجو ی آگاهانه شامل جستجوی اول بهترین ، توابع جستجوی حافظه محدود ، جستجوی A* ، جستجوی اول- بهترین بازگشتی (RBFS) ، SMA^* ، توابع اکتشافی ، سایر روشهای جستجوی بهبود یافته ، الگوریتم های جستجوی محلی شامل جستجوی تپه نوردی، جستجوی بازپخت شبیه سازی شده، جستجوی پرتو محلی، الگوریتم ژنتیک.
- مسائل ارضاء محدودیت شامل جستجوی عقبگرد ، کنترل رو به جلو، انتشار محدودیت ، الگوریتم عقبگرد هوشمند، جستجوی محلی.
- جستجوی رقابتی (الگوریتم های بازی) بازی های دو نفره ، الگوریتم minimax و هرس آلفا – بتا ، بازیهای حاوی عنصر شانس (مانند تاس) ، بازیهای چند نفره ، و....
- عاملهای مبتنی بر دانش ، عاملهایی که منطقی استدلال می کنند ، نمایش منطق ، منطق گزاره ای ، استدلال ، چند روش استنتاج در منطق گزاره ها.
- منطق رتبه اول ، استنتاج در این منطق ، قوانین استنتاج ، استنتاج زنجیره ای به جلو و به عقب
- منبع :

Artificial Intelligence A modern Approach

by : S. Russel & P. Norvig

فصل اول

مفاهیم اولیه

حوزهٔ هوش مصنوعی (Artificial Intelligence) به اختصار AI سعی دارد تا نـه تنهـا، موجودیـت هـای هوشمند را درک کند، بلکه موجودیتهای هوشمند را بسـازد. از ایـن رو، یکـی از علـل مطالعـهٔ آن، یـادگیری بیشتر در مورد خودمان است.

در حال حاضر، AI زیر شاخه های وسیعی از موضوعات عمومی مانند ادراک و استدلال منطقـی تـا کارهـای خاص مانند بازی شطرنج، اثبات قضایای ریاضی، سرودن شعر و تشـخیص امـراض را شـامل مـی شـود. غالبـاً دانشمندان علوم دیگر زمینه ها، به تدریج به سوی هوش مصنوعی متمایل می گردند. تا جـایی کـه ابزارهـا و واژه هایی را می یابند تا از طریق آنها بتوانند وظایف هوشمندانهٔ خود را سیستماتیک و خودکار سازند. مشـابه به همین محققان AI می توانند از روش های خود برای هر زمینه ای از کوشش هوشمندانهٔ انسـان اسـتفاده کنند. از این منظر، این دانش حقیقتاً یک زمینهٔ واحد خواهد بود.

تعریف هوش مصنوعی: خودکارسازی فعالیت های مرتبط با تفکر انسان. مانند تصمیم گیری یا هنر سـاخت ماشین هائی که بتوانند کارهائی انجام بدهند که انسان برای انجام آن ها نیاز به تفکر دارد.

تعریف هوش مصنوعی بر دو جنبه پردازش فکری و استدلالی، و پردازش رفتـاری متمرکـز شـده اسـت. هـم چنـین ایـن تعریـف دو عامـل ماننـد انسـان بـودن و مفهـوم ایـده آل هوشـمندی کـه آن را عقلانـی بـودن

(rationality) می گویند، مدنظر دارد. که به طور خلاصه در جدول ۱-۱ و به طور مفصل تر در جدول ۱-۲ بیان شده است. در ادامه مطلب هر یک از این موارد را توضیح بیشتری می دهیم:

	مانند انسان	منطقی بودن (rationality) (عقلانی)
فکر کردن	۱- انسانی فکر کردن	۳- عقلانی (منطقی) فکر کردن
عمل کردن	۲- انسانی عمل کردن	۴- عقلانی (منطقی) عمل کردن

جدول ۱-۱

	ارائه انسانی	عقلانی بودن
پردازش فکری و استدلالی	«کشش جدید هیجان انگیز برای ساختن کامپیوترهایی که فکر کنند، ماشین هایی با قدرت تفکر، و به حس کامل» (هاوگلند، ۱۹۸۵) «عملیاتی که با اعمال تفکر انسان نظیر تصمیم گیری، حل مسئله، یادگیری مربوط می شوند» (بلمن "Bellman"، ۱۹۷۸)	«مطالعهٔ توانایی های ذهنی از طریق مدل های کامپیوتری» (چارنیاک و مک درموت، ۱۹۸۵) (Charniak MC Dermott) «مطالعهٔ محاسبات که آن را قادر سازد تا درک و استدلال و عمل کنند» (وینستون "Winston"، ۱۹۹۲)
پردازش رفتاری	«هنر خلق ماشین هایی که توانایی انجام عملیاتی داشته باشند که آن عملیات توسط انسان نیاز به هوشمندی داشته باشند» (کرزویل "Kurzweil" ۱۹۹۱) «مطالعه بر روی چگونگی ساخت کامپیوترهایی که کارها را در هر لحظه بهتر از انسان ها انجام دهند. "ریچ Rich" و "نایت Kinght"، ۱۹۹۱)	«حیطه ای از مطالعه که رفتار هوشمند را تحت عنوان فرآیندهای کامپیوتری شرح داده و مورد رقابت قرار دهد» (شالکوف "Schalkoff"، ۱۹۹۰) «شاخه ای از علوم کامپیوتر که با اتوماسیون رفتار هوشمند مربوط می شود» ("لوگر Luger" و استابلفید "Stubbfield"، ۱۹۹۳)

جدول ۱-۲

۱ - انسانی فکر کردن: رهیافت مدل سازی شناختی

اگر قادر به ایجاد تئوری دقیقی دربارهٔ ذهن باشیم آنگاه قادر خواهیم بود این تئوری را به برنامه کامپیوتری تبدیل کنیم. اگر ورودی و خروجی و زمان بندی با رفتار انسان تطبیق داشته باشد، گواهی بر آن دارد که برخی از مکانیزم های برنامهٔ ما در انسان هم عمل خواهد کرد.

۲- انسان گونه عمل کردن : رهیافت آزمون تورینگ (Turing)

تست تورینگ که توسط آلن تورینگ (۱۹۵۰) پیشنهاد شد، تعریف عملی، رضایت بخشی از هوش ایجاد کرده است. تورینگ رفتار هوشمند را به عنوان توانایی رسیدن به سطح ارائهٔ انسانی در تمامی وظایف ادراکی تعریف کرد که حتی قادر به فریفتن یک محقق نیز باشد. آزمونی که او پیشنهاد کرد، آن بود که کامپیوتر می بایست توسط فردی که از طریق تله تایپ (teletype) به آن دسترسی دارد مورد تحقیق قرار گیرد و زمانی در آزمون موفق می شود که محقق نتواند دریابد، در آن طرف انسان قرار دارد یا کامپیوتر. برنامه ریزی کامپیوتری که بتواند این تست را انجام دهد کار زیادی می برد. کامپیوتر مذکور باید قابلیت های زیر را داشته باشد:

- پردازش زبان طبیعی (natural language processing) تا قادر به محاوره به زبان انگلیسی (یا زبان انسانی دیگر) گردد.
- بازنمایی دانش (knowledge representation) تا اطلاعات تولید شدهٔ قبل یا در حین آزمون را ذخیره سازد.
- استدلال خودکار (automated reasoning) تا از اطلاعات ذخیره شده برای پاسخ به پرسش ها استفاده کرده و نتایج جدیدی را استخراج کند.

- یادگیری ماشینی (machine learning) تا خود را با شرایط تازه وفق دهد و الگوها را کشف و برون ریزی کند.

تست تورینگ اندیشمندانه از ارتباط فیزیکی مستقیم بین کامپیوتر و آزمون شونده اجتناب می کند، زیرا شبیه سازی فیزیکی فرد، برای هوشمندی ضروری نیست.

۳- منطقی فکر کردن: رهیافت قوانین تفکر

منطق دستور زبان دقیقی برای جملاتی در مورد تمامی انواع اشیاء در جهان و رابطهٔ بین آنها ایجاد می کند. دو مشکل عمده در این راه وجود داشت. اول اینکه دریافت دانش غیر رسمی (informal) و تبدیل آن به شکل رسمی توسط علائم منطقی ساده نیست، مخصوصاً زمانی که دانش ما از درجهٔ اطمینان کمتر از ۱۰۰٪ برخوردار باشد. دوم اینکه تفاوت عمده ای بین قادر به حل مسئله بودن در «اصول» و انجام آن در عمل وجود دارد.

۴- منطقی عمل کردن: رهیافت عامل عقلانی

- رفتار منطقی بدین معناست که با داشتن عقیدهٔ واحد به هدف واحدی برسیم.
- یک عامل در اصل چیزی است که ابتدا درک می کند و سپس عمل می کند.
- هوش مصنوعی به عنوان مبنای عاملهای منطقی به کار برده می شود.

عامل و محیط ها

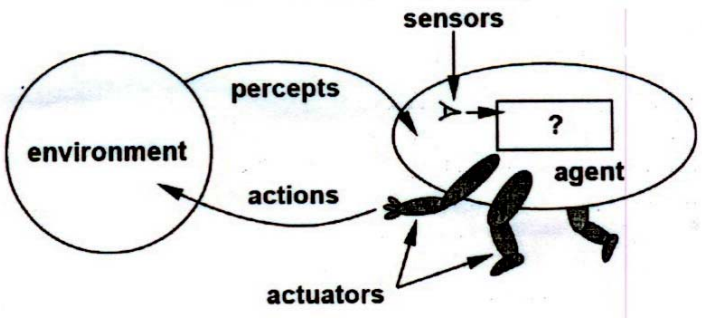
عامل هر چیزی است که قادر به نگرش **محیط**[1] از طریق حسگرها و اثرگذاری بر روی محیط توسط **عمل کننده ها**[2] باشد. این ایدهٔ ساده در شکل زیر نشان داده شده است. عامل انسانی چشم، گوش و دیگر

[1] . environment
[2] . actuators

ارگان های حسی و دست، پا، بینی و دیگر اعضای بدن برای عمل کردن دارد. عامل روباتیک ممکن است دوربین یا یابنده های مادون قرمز برای حسگر و انواع موتورها برای عمل کنندگی داشته باشد. عامل نرم افزاری ضربات کلید، محتویات فایل و بسته های شبکه را به عنوان ورودی حسی می گیرد و بر روی محیط توسط نمایش روی صفحه، نوشتن فایل ها و ارسال بسته های شبکه اثری می گذارد. ما فرضی عمومی می کنیم که هر عاملی می تواند اعمال خود را درک کند. (اما همیشه اثرهای آن را درک نمی کند).

ما از واژه **درک**[1] برای ارجاع به ورودی ادارکی عامل در هر وضعیت داده شده ای، استفاده می کنیم.

دنبالهٔ ادراکی عامل تاریخچهٔ کامل هر چیزی است که عامل درک کرده است. به طور عمومی، شانس عمل عامل در هر وضعیت داده شده ای می تواند وابسته به کل دنبالهٔ ادراکی مشاهده شده تا به امروز باشد.



شکل ۱-۱

تعریف: عامل عقلانی (معقول) rational agent

عاملی که کارها (action)ها را درست انجام می دهد. منظور از عمل درست چیزی است که باعث می شود عامل کاملاً موفق شود.

سؤال: موفقیت چطور اندازه گیری می شود؟

با توصیف محیط و حسگرها و عملگرهای عامل، می توان انجام وظیفه عامل را توصیف نمود.

معیاری به عنوان اندازه گیری کارآیی (performance measure) تعریف می شود که ملاک موفقیت رفتار یک عامل است. و برای ارزیابی و داوری عملکرد عامل استفاده می شود.

[1] . percept

زمانی که عاملی در یک محیط فعال می شود، بر طبق آنچه که از حسگرهای خود دریافت مـی کنـد، دنبالـه ای از عملیات را تولید کند. این دنباله منجر به رفتن محیط به دنباله ای از وضعیت ها خواهد شد. اگر دنبالـه مطلوب باشد، بدین معنی است که عامل به خوبی عمل کرده است. گاهی عامل ها طوری طراحی مـی شـوند که خودشان در مورد عملکردشان قضاوت می کنند. یعنی کارآیی را خودشـان تعریـف و ارزیـابی مـی کننـد. بنابراین می توانیم از عامل سؤال کنیم که چقدر از عملکرد خود راضی است. می توان گفت این نـوع ارزیـابی نمی تواند کاملاً بی طرفانه باشد و برخی عاملها خودشان را فریب می دهند.

مثلاً اگر عامل یک جارو برقی باشد، می توانیم اندازه کارایی را بر حسب میزان خاک تمیز شده در یک ساعت در نظر بگیریم. عامل منطقی می تواند اندازهٔ کارایی خود را با تمیز کردن خاک و دوباره ریختن آن روی کـف اتاق و دوباره تمیز کردن آن و... حداکثر سازد.

البته این ملاک کارایی را می توان با یک جریمه به خاطر مصرف برق و ایجاد سر و صدا یا در نظر گرفتن یک جایزه برای تمیز کردن خانه در واحد زمان اصلاح نمود.

شاید بهتر آن باشد که برای اندازه گیری و تعریف ملاک کارایی از یک داور یا یک عامل دیگر استفاده شود.

تفاوت بین عامل معقول و یک عامل همه چیزدان (دانای کل) (omniscience)

عامل دانای کل، نتایج خروجی اعمال خود را می داند و می تواند بر طبق آن عمل کند. او همه دنیای اطـراف را به خوبی sense می کند و همهٔ actionها را به طور صحیح انجام می دهد. حتی انسـان هـم یـک عامـل همه چیزدان نیست. اما یک عامل معقول سعی می کند تا حد امکان action صحیح را انتخاب کند.

میزان معقول بودن یک عامل معقول به فاکتورهای زیر بستگی دارد:

- اندازهٔ کارایی که میزان موفقیت را مشخص می کند.
- دانش اولیهٔ عامل از محیط.

- عملی که عامل می تواند ترتیب دهد.
- دنبالهٔ ادراکی عامل تاکنون.

این ۴ فاکتور ما را به تعریف عامل عقلانی ایده آل رهنمون می کند.

یک عامل عقلانی ایده آل برای هر دنباله ادراکی ممکن باید عملی را انتخاب کند که به نظر می رسد اندازهٔ کارایی را حداکثر سازد و این کار را با استفاده از دنباله ادراکی و هر نوع دانش درونی که عامل دارد انجام می دهد.

البته باید توجه داشته باشیم که هیچ کس یک عامل را به خاطر مشاهداتی که حسگرهایش قدرت دریافت آن را نداشته اند و یا به خاطر اعمالی که عمل کننده هایش قادر به انجام آن نبوده اند سرزنش نمی کند. سپس همیشه اندازه کارایی در حیطه توانائی یک عامل تعریف می شود و نه بیشتر از آن.

گفتیم که عاملی که برای هر دنباله ادراک ممکن، عملی را انتخاب می کند که موفقیت آن حداکثر باشد یک عامل معقول است. عامل معقول برای انتخاب عمل به چند چیز باید دقت کند:

- دانش داخلی: یعنی دانش اولیه و مختصری که توسط طراح به عامل داده می شود و خود عامل در ایجاد آن نقشی نداشته است.
- جمع آوری اطلاعات که همان اطلاعاتی هستند که حسگرها به عامل می دهند و به این عمل پویش یا (exploration) می گویند.
- یادگیری هر چه بیشتر از ادراک خود

مثلاً یک ساعت ساده را در نظر بگیرید. او از محیط اطراف خود چیزی نمی گیرد. پس معیار تصمیم گیری او براساس دانش داخلی اش است. یعنی عمل او براساس دانش اولیه ای است که طراح به او داده است. این عامل در حیطه توانایی خودش یک عامل معقول است. حال اگر این ساعت را به کشور دیگری ببریم ساعت اشتباه را نشان می دهد اما باز هم معقول است چون در حیطه توانایی اش کار می کند. اما اگر این ساعت را به سیستم GPS مجهز کنیم در آن صورت از محیط توسط حسگرهایش اطلاعاتی کسب می کند و خود را به ساعت جهانی هماهنگ می کند و در کشور دیگر هم زمان را درست نشان می دهد.

خود مختاری (استقلال) autonomy

اگر همه اعمال یک عامل براساس دانش داخلی اش باشد دارای درجه خود مختاری صفر است.

عامل عقلانی باید خود مختار باشد یعنی باید بیاموزد که چگونه می تواند با دانش اولیه اشتباه یا ناقص خود برخورد کند. هر چه یک عامل از دانش داخلی خود کمتر استفاده کند و بیشتر متکی به اطلاعات جمع آوری شده توسط حسگرهایش باشد، خود مختارتر و مستقل تر است و بنابراین میزان هوشمندی اش بیشتر است.

زمانی که عامل هیچ تجربه ای ندارد تصادفی عمل می کند مگر آنکه طراح اطلاعات اولیه ای به او داده باشد. بنابراین منطقی است که عامل هوشمند مصنوعی را به دانش اولیه مجهز کنیم. بعد از جمع آوری اطلاعات از محیط، رفتار عامل عقلانی به طور موثری مستقل از دانش اولیه آن خواهد شد.

مشخص سازی محیط کار

در بحث عقلانیت نیاز به تعریف اندازه کارایی (P)- محیط (E)- عملگرها (A) و حسگرها (S) داریم که همه اینها را تحت عنوان محیط (PEAS) کار ذکر می کنیم. در طراحی عامل، اولین قدم همواره باید مشخص سازی محیط کاری تا حداکثر ممکن باشد.

مثال: راننده تاکسی خودکار.

۱- اندازهٔ کارایی که ما مایل به کسب آن برای رانندهٔ خودکار هستیم، چیست؟ کیفیتهای مناسب شامل رسیدن به مقصد صحیح، حداقل سازی مصرف سوخت و استهلاک، حداقل سازی زمان سفر یا هزینه، حداقل سازی نقض قوانین ترافیکی و برخورد با دیگر رانندگان، حداکثرسازی امنیت و اطمینان مسافر، حداکثرسازی سود هستند. آشکارا، برخی از این اهداف متناقضند ولی باید بین آنها به تعادلی رسید.

۲- محیط رانندگی تاکسی چیست؟ هر رانندهٔ تاکسی باید با گستره ای از مسیرها برخورد کند، از خیابان ها و کوچه های باریک گرفته تا آزاد راه های ۱۲ خطی مسیرها شامل ترافیک، عابر پیاده، حیوانات

وحشی، کارگران ساختمانی، ماشین های پلیس و چاله ها هستند. راننده می بایست در تعادل با مسافران بالقوه و بالفعل باشد. همچنین برخی انتخابات اختیاری نیز وجود دارد. تاکسی ممکن است در جنوب کالیفرنیا رانندگی کند، که برف بندرت می بارد یا در آلاسکا که بندرت نمی بارد. گاهی ممکن است سمت چپ رانندگی کند مثل زمان رانندگی در بریتانیا یا ژاپن. آشکارا، محدودسازی بیشتر محیط، طراحی را ساده تر می کند.

۳- عملگرهای موجود برای تاکسی خودکار همانند چیزهای موجود برای راننده انسانی است، کنترل بر موتور از طریق پدال گاز و کنترل بر ترمز، به علاوه نیاز به یک صفحهٔ نمایش یا سازندهٔ صوت برای گفتگو با مسافر و یا حتی برای برقراری ارتباط با دیگر خودروهاست.

۴- برای حصول اهداف در محیط رانندگی، تاکسی می بایست بداند کجاست، چه کسان دیگری در مسیر هستند و با چه سرعتی حرکت می کنند. حسگرهای اولیه شامل یک یا چند دوربین تلویزیونی قابل کنترل، سرعت سنج و ادومتر است. برای کنترل مناسب خودرو، به ویژه در مسیرهای منحنی، نیاز به شتاب سنج، دانستن وضعیت مکانیکی خودرو است و نیاز به حسگرهای آرایه ای سیستم الکتریکی و موتور است.

ممکن است ابزارهای دقیقی نیاز باشد که برای رانندهٔ انسانی متوسط موجود نیست. سیستم موقعیت یاب عمومی ماهواره (GPS) برای ساختن موقعیت صحیح نسبت به نقشهٔ الکترونیکی، و حسگرهای مادون قرمز یا سونار برای سنجیدن فاصله تا دیگر ماشین ها و موانع. در نهایت به صفحهٔ کلید یا میکروفونی برای مسافر نیاز است تا درخواست مقصد را بدهد.

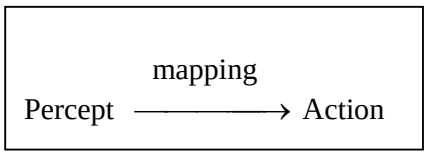
Agent Type	**Performance Measure**	**Environment**	**Actuators**	**Sensors**
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roaders, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer. GPS, odometer, accelerometer,

engine sensors, keyboard				

جدول ۱-۳: تشریح PEAS محیط کار برای ی تاکسی خودکار

عامل نرم افزاری (software agents)

عامل نرم افزاری در گسترهٔ نامحدود و غنی قرار دارند. ممکن است که عامل نرم افزاری یک نگاشت از مجموعهٔ اطلاعات جمع آوری شده به مجموعهٔ عمل ها را انجام می دهد. یعنی به ازای اطلاعات حس شده بگوید چه عملی باید انجام شود.



مثل این است که عالم دارای یک جدول است که در یک ستون آن percept و در ستون دیگر آن action نوشته شده و کار عامل پیدا کردن عمل مناسب یک اتفاق خاص موجود در جدول است.

این جدول در صورتی که توسط طراح پر شود عامل ما اصلاً استقلال ندارد.

برخی محیط های "واقعی" عملاً بسیار ساده هستند. برای مثال، روباتی که برای آن طراحی شده تا قطعات روی نوار نقاله را کنترل کند، می تواند از فرضیات ساده کننده استفاده کند: نور همواره ثابت است، قطعات روی نوار نقاله یکی از انواع شناخته شده است و تنها دو عمل وجود دارد (قبول یا رد).

در نقطهٔ مقابل، برخی عامل های نرم افزاری (یا روبات های نرم افزاری یا softbots) در دامنه های نامحدود و غنی قرار دارند. روبات نرم افزاری را در نظر بگیرید که در محیط شبیه سازی پرواز برای هواپیمایی تجاری بزرگ طراحی شده است. شبیه سازی جزئیات بیشماری دارد، محیط پیچیده شامل دیگر هواپیما ها و عملیات زمینی، و عامل نرم افزاری باید از طیف گستردهٔ عملیات در وضعیت بلادرنگ تصمیم گیری کند. یا روبات نرم افزاری را تصور کنید که طراحی شده تا منابع خبری اینترنت را جستجو کرده و موارد مورد توجه را به مشتری نشان دهد. برای خوب عمل کردن، باید قادر به پردازش زبان طبیعی باشد. در این صورت نیاز به یادگیری سلیقه مشتری دارد، و نیازمند تغییر پویای برنامه ریزی است مانند زمانی که اتصال یک منبع

خبری قطع شد، یا زمانی که دیگری **بر خط** می شود. اینترنت محیطی است که پیچیدگی آن بیشتر از محیط های فیزیکی است و اهالی آن عاملهای مصنوعی متعددی هستند.

محیط

اولین چیزی که در طراحی یک عامل باید در نظر گرفت محیطی است که عامل در آن فعالیت می کند. محیط ها می توانند با هم خیلی تفاوت داشته باشند. انواع آنها عبارتند از:

- **کاملاً قابل مشاهده در مقابل نیمهٔ قابل مشاهده accessible- nonaccessible**

اگر حسگرهای عاملی دسترسی به حالت کامل محیط در هر مقطع زمانی بدهند، می گوییم محیط کار کاملاً قابل مشاهده است. یک محیط کار به طور مؤثر کاملاً قابل مشاهده است، اگر حسگرها تمامی زمینه های مربوط به انتخاب عمل را آشکار سازند. محیط های کاملاً قابل مشاهده ساده هستند، چون نیازی به نگهداری حالت درونی برای حفظ وضعیت دنیا ندارند.

محیط نیمه قابل مشاهده خواهد بود، اگر حسگرها نادرست، یا با خطا باشند، یا بخشی از حالات به سادگی از داده حسگر حذف شده باشند. برای مثال عامل جارو برقی تنها با یک حسگر کثیف محلی نمی تواند بگوید که در دیگر خانه ها کثیفی وجود دارد و یا رانندهٔ تاکسی خودکار نمی تواند بفهمد دیگر رانندگان چه فکری در سر دارند.

قطعی در مقابل تصادفی Deterministic- non Deterministic

اگر حالت بعدی محیط کاملاً توسط حالت جاری و عمل اجرا شده توسط عامل، قابل تعیین باشد، می گوییم محیط قطعی است و در غیر این صورت تصادفی خواهد بود.

مثال غیر قطعی: عمل ترمز کردن در رانندگی و انجام شدن پیش بینی ما که متوقف شدن ماشین است.

مثال قطعی: عمل قرار دادن کتاب روی میز که بعد از آن کتاب روی میز است.

به محیط هائی مانند صفحه شطرنج محیط های strategic می گویند. وجود حریف در محیط ایجاد عدم قطعیت می کند. حرکت های ما در شطرنج قطعی هستند ولی حرکات حریف قابل پیش بینی نیستند و ایجاد عدم قطعیت می کنند.

اپیزودیک در مقابل دنباله ای (تقسیم پذیر در مقابل ترتیبی)

در محیط کار اپیزودیک، تجربهٔ عامل به اپیزودهای اتمی تقسیم می شود. هر اپیزود مشتمل بر ادراک عامل و انجام عمل واحدی پس از آن است. اپیزود بعدی معمولاً وابسته به اعمال صورت گرفته در اپیزود قبلی نیست. در محیط های اپیزودیک، انتخاب عمل در هر اپیزود وابسته به خود اپیزود است. بسیاری از کارهای دسته بندی اپیزودیک هستند. برای مثال، عاملی که نقاط اتصال قطعات در خط تولید را کنترل می کند، هر تصمیم خود را بر پایهٔ قطعهٔ جاری می گیرد، بدون توجه به تصمیمات قبلی خود و به علاوه تصمیم جاری اثری بر تعیین خرابی قطعهٔ بعدی ندارد. در محیط های دنباله ای، تصمیم جاری بر تمامی تصمیمات بعدی مؤثر خواهد بود. شطرنج و رانندگی تاکسی دنباله ای هستند، چرا که در هر دو مورد اعمال کوتاه مدت می تواند اثرات دراز مدت به همراه داشته باشند. محیط های اپیزودیک بسیار ساده تر از محیط های دنباله ای هستند، چرا که عامل نیازی به تفکر دربارهٔ آینده ندارد.

ایستا در مقابل پویا

اگر محیط در حین تفکر عامل تغییر کند، می گوییم محیط برای آن عامل پویاست و در غیر این صورت ایستا خواهد بود. محیط های ایستا برای کار ساده ترند، زیرا عامل نیازی به نگاه کردن به محیط حین تصمیم گیری ندارد و یا نیازی به نگرانی در مورد گذشت زمان نیست. محیط های پویا، به طور مداوم از عامل

در مـــــــــــــــــــــــــــــــــورد آنچـــه می خواهند انجام دهند سؤال می کنند، اگر هنوز تصمیمی نگرفته اند یعنی تصمیم گرفته اند کاری نکنند.

اگر خود محیط در گذر زمان تغییر نکند ولی امتیاز کارایی عامل تغییر کنـد، مـی گـوییم محـیط نیمـه پویـا است. رانندگی تاکسی به وضوح پویا است، دیگر ماشین ها و تاکسی هـا در حـالی کـه الگـوریتم راننـدگی در حال تصمیم گیری است، حرکت می کنند. شطرنج در حین بازی با ساعت، نیمه پویـا اسـت. جـدول کلمـات ایستا است.

گسسته در مقابل پیوسته (Discrete- continuous)

تمایز گسسته/ پیوسته می تواند به حالت محیط، به نحوۀ برخورد با زبان و به ادراکات و اعمال عامـل مربـوط شود. برای مثال یک محیط گسسته همانند بازی شطرنج دارای تعداد متناوبی حالات مجـزا اسـت. بـه عـلاوه شطرنج مجموعۀ گسسته ای از ادراکات و اعمال دارد.

رانندگی تاکسی مسئله زمان و حالت پیوسته دارد: سرعت و مکان تاکسی و دیگـر خودروهـا در بـازۀ مقـادیر پیوسته تغییر می کنند و در زمان نرم نرمک انجام مـی شـوند. همچنـین اعمـال راننـدگی تاکسـی پیوسـته هستند. تصاویر دریافتی از دوربین های دیجیتال گسسته است، اما به صـورت تغییـرات پیوسـتۀ تـراکم هـا و مکان ها به آن برخورد می شود.

تک عامله در مقابل چند عامله single Agent- Multi Agent

تمایز بین محیط های تک عامل و چند عامل به نظر به اندازۀ کافی ساده مـی آیـد. بـرای مثـال، عـاملی کـه جدول کلمات متقاطع را حل می کند خودش در یک محیط تک- عامـل قـرار دارد. حـال آنکـه عامـل بـازی شطرنج در محیط دو- عامل قرار دارد.

سخت ترین محیط ها دارای خصوصیات زیر هستند:

- نیمه قابل مشاهده

- غیر قطعی
- غیر اپیزودیک
- پویا
- پیوسته
- چند عامله

ساختار عامل ها

وظیفه هوش مصنوعی طراحی برنامهٔ عامل است. در واقع هوش مصنوعی تابع نگاشت ادراکات حسگرها به عمل ها را پیاده سازی می کند. فرض می کنیم که این برنامه بر روی برخی انواع ابزارهای محاسبه گر با حسگرها و عمل کننده ها اجرا خواهد شد که آن را معماری می نامیم. بنابراین می توان گفت: **عامل = معماری + برنامه**

هر عامل از دو قسمت سخت افزاری و نرم افزاری تشکیل شده. معماری قسمت سخت افزاری عامل و برنامه قسمت نرم افزاری عامل است. قسمت سخت افزاری می تواند شامل شکل ظاهری مثلاً روبات یا کامپیوتر باشد. حسگرها و عمل کننده ها در قسمت سخت افزاری هستند. وظیفه قسمت سخت افزاری گرفتن ادراکات، اجرای توابع و انجام عمل های مناسب توسط عمل کننده هاست. قسمت نرم افزاری یک برنامه است که می تواند به هر زبانی پیاده سازی شود که وظیفه آن نگاشت ادراکات به عمل ها است.

```
function  f( percept )
        return action
```

برنامه عامل: ادراک کنونی را به عنوان ورودی دریافت می کند. (بیشتر از این از محیط قابل دریافت نیست)

تابع عامل: کل تاریخچه ادراک را دریافت می کند.

اگر اعمال عامل وابسته به کل دنبالهٔ ادراکی باشد، عامل باید بتواند ادراکات قبلی را به یاد آورد. بـرای مثـال، شکل زیر برنامهٔ عامل ساده ای را نمایش داده که دنبالهٔ ادراکی را حفظ می کند و سپس از آن برای شـاخص گذاری جدول اعمال، برای تصمیم گیری استفاده می کند. به عنوان طراح مـی بایسـت بـرای سـاخت عامـل منطقی، جدولی بسازیم که شامل عمل مناسب برای هر دنبالهٔ ادراکی ممکن باشد.

```
Function TABLE-DRIVEN-AGENT (percept) returns an action
    static: percepts, a sequence, initially empty
            table, a table of actions, indexed by percept sequences, initially fully specified
    append percept to the end of percepts
    action ← LOOKUP (percepts, table)
    return action
```

شکل ۱-۲: برنامهٔ Table-Driven-Agent برای هر ادراک جدید فراخوانی شده و هر زمانی عملی را بر می گرداند. همچنین با استفاده از ساختار داده های خصوصی خود قادر به حفظ دنبالهٔ ادراکی است.

رهیافت جدول گرا برای ساخت عامل به وضوح شکست می خورد. تاکسی خودکار را در نظـر بگیریـد: ورودی بینایی از یک دوربین با نرخ ۲۷ مگا بایت در ثانیه (۳۰ فریم در ثانیه، ٤٨٠×٦٤٠ بیت اطلاعات رنـگ) اسـت. این جدول مراجعه ای به اندازهٔ بیش از $10^{250000000000}$ خانه برای یک ساعت رانـندگی مـی خواهـد. (تعـداد اتمهای هستی کمتر از 10^{80} است) اندازهٔ این جدول ها به معنی آن است که:

- هیچ عامل فیزیکی در این جهان حافظه برای نگهداری این جدول ندارد.
- طراحی زمان برای خلق جدول ندارد.
- هیچ عاملی فرصت یاد گرفتن صحیح خانه های جدول از تجربه را ندارد.
- حتی اگر محیط به اندازهٔ کافی ساده باشد تا به جدول قابل قبولی برسیم، طراح هنوز راهنمایی برای چگونگی پر کردن خانه های جدول ندارد.

به رغم تمام این نکات، Table-Driven-Agent تابع عامل مطلوب را پیاده سازی می کند. چالش کلیدی AI یافتن چگونگی نوشتن برنامه ای است که رفتار منطقی را از طریق حجم کمی کد، به جای جداول بسیار بزرگ، تولید کند. مثال هایی خواهیم زد که نشان می دهند این امر با موفقیت در دیگر زمینه ها شدنی است. برای مثال، جدول بزرگ ریشهٔ دوم توسط مهندسان یا دانش آموزان قبل از دههٔ هفتاد میلادی استفاده می شد که با برنامهٔ پنج خطی روش نیوتون روی ماشین حساب ها تعویض شد. پرسش اینست، آیا AI می تواند برای رفتار هوشمند در کل کاری بکند که نیوتون برای ریشه دوم کرد؟ باور ما آن است که پاسخ این پرسش مثبت است.

برای نوشتن یک برنامه عامل باید بدانیم که:

۱- هدف عامل چیست؟

۲- عمل های عامل چه هستند؟

۳- عامل در چه محیطی است؟

۴- ادراکات چگونه هستند؟

۵- اندازه گیری کارایی چگونه تعریف می شود؟

مثال: عامل: طراحی یک پزشک

۱- هدف: بهبودی بیمار

۲- عمل ها: معاینه، پرسیدن سؤال، تجویز دارو، آزمایش و...

۳- محیط: بیمارستان

۴- ادراکات: علائم بیمار، پاسخ های بیمار به سؤالات پزشک و...

۵- اندازه گیری کارایی: میزان بهبودی بیمار

در ادامهٔ این بخش، انواع اولیهٔ برنامه های عامل را که در بیشتر سیستم های هوشمند به عنوان پایه قرار می گیرند، به اختصار توضیح می دهیم:

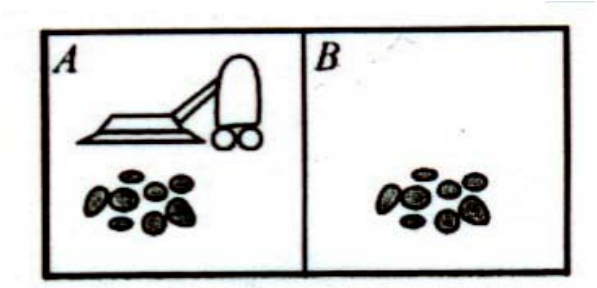
- **عامل های واکنشی ساده Simple Reflex Agent**

ساده ترین نوع، عامل واکنشی ساده است. این عامل ها اعمال خود را بر طبق ادراک جاری انتخاب می کنند، بدون توجه به اینکه تاریخچهٔ ادراکات قبلی چیست.

مثال: عامل مکش: دنیای جارو برقی تنها با دو مکان را ایجاد می کند.

- این جارو برقی یا در مکان A یا مکان B است.
- عملهای آن رفتن به چپ، رفتن به راست، و مکش است.
- حسگرهای آن دو وضعیت تمیز و کثیف را تشخیص می دهند.

عامل مکش یک عامل واکنشی ساده است. زیرا تصمیمات را بر مبنای حالت جاری و وجود کثیفی در آن می گیرد. تابع عامل در شکل ۱-۴ نشان داده شده است.



Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean] , [A, Clean]	Right
[A, Dirty] , [A, Clean]	Suck
.	.
.	..
[A, Clean] , [A, Clean] , [A, Clean]	Right
[A, Dirty] , [A, Clean] , [A, Clean]	Suck
.	.
.	.

شکل ۱-۳ : دنیای جارو برقی تنها در دو مکان

function REFLEX- VACUUM- AGENT (*location*, *status*) **returns** an action

if status = Dirty **then return** Suck

```
else if location = A then return Right
else if location = B then return Left
```

شکل ۱-۴: برنامهٔ عامل برای عامل واکنش ساده در عامل مکش. این برنامه تابع عامل را به صورت جدول موجود در شکل ۱-۳ پیاده سازی می کند.

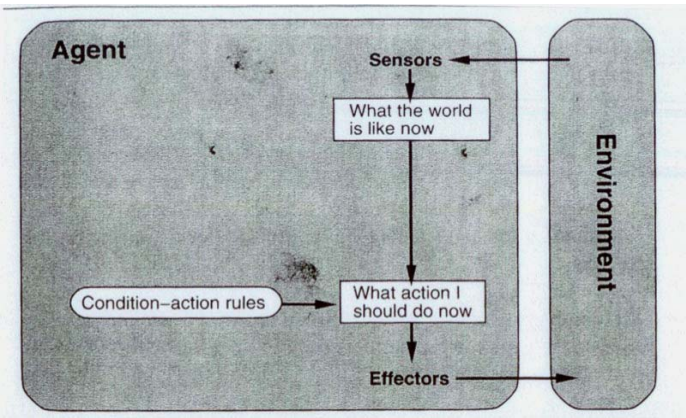
توجه کنید، برنامه عامل مکش در مقایسه با جدول وابسته به آن بسیار کوچک است. بیشترین کاهش از عدم توجه به تاریخچهٔ عامل می آید که مقدار احتمالات را از 4^T فقط به ۴ کاهش می دهد. دلیل کوچک بودن از آنجا نشأت می گیرد که زمانی که خانهٔ جاری کثیف باشد، عمل وابسته به محل نخواهد بود.

تصور کنید که در تاکسی راننده خودکار هستید. اگر ماشین جلویی ترمز کند، و چراغ خطرهای آن روشن شود، عامل متوجه می شود و بلافاصله ترمز می کند. به عبارت دیگر، برخی پردازش ها در ورودی بصری انجام می شود تا بفهمیم "ماشین جلویی ترمز کرد". سپس در برنامهٔ عامل، عواملی فعال شده و عمل" ترمز کردن" را انجام می دهند. چنین اتصالی را **قانون شرط- عمل** (Condition- action rule)می نامیم و می نویسیم:

اگر ماشین جلویی ترمز کرد آنگاه ترمز کن.

انسان ها اتصالات متعددی از این نوع دارند. برخی از آنها پاسخ های یاد گرفته شده اند (مثل رانندگی) و برخی بازتاب های غیر ارادی هستند (مثل چشمک زدن حین نزدیک شدن جسمی به چشم).

این برنامهٔ عامل مکش برای یک محیط مکش خاص است. رهیافت عمومی تر و انعطاف پذیرتر آن است که ابتدا مفسری همه منظوره برای قوانین شرط- عمل بسازیم و سپس مجموعهٔ قوانین را برای محیط کاری خاص ایجاد کنیم. سپس قوانین جدیدی که ایجاد می شوند را در هر مرحله به این جدول اضافه کنیم. شکل زیر ساختار چنین برنامهٔ عاملی را به طور ترسیمی داده است که در آن چگونگی قابلیت قوانین شرط- عمل جهت ساختن ارتباط بین ادراک و عمل نشان داده شده است. از مربع مستطیل ها برای نمایش حالت درونی جاری فرآیند تصمیم گیری عامل و از بیضی ها برای نمایش اطلاعات پس زمینهٔ مورد استفاده در فرآیند استفاده شده است.



شکل ۱-۵: نمودار ترسیمی یک عامل واکنش ساده

برنامهٔ عامل، در شکل زیر نشان داده شده است. تابع INTERPRET- INPUT توصیف انتزاعی حالت جاری از ادراک را تولید می کند و تابع RULE- MATCH اولین قانون در مجموعهٔ قوانین را که با تشریح حالت جاری تطبیق دارد را بر می گرداند. RULE-ACTION عمل مناسب در مقابل این قانون را پیدا کرده و بر می گرداند.

```
function SIMPLE- REFLEX- AGENT (percept) return an action
    static: rules, a set of condition- action rules
    state ← INTERPRET- INPUT (percept)
    rule ← RULE- MATCH (state, rules)
    action ← RULE- ACTION [rule]
    return action
```

شکل ۱-۶: یک عامل واکنشی ساده بر طبق قانونی که شرط آن بر حالت جاری تطبیق دارد عمل می کند.

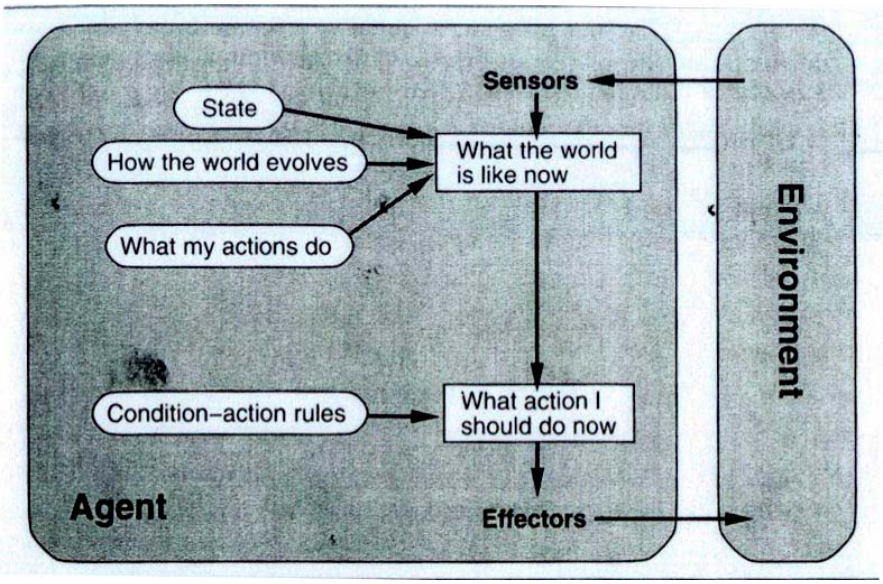
عامل های واکنشی ساده، ساختاری بسیار ساده دارند ولی آشکار خواهد شد که هوشمندی بسیار محدودی هم دارند. برنامه عامل شکل بالا تنها زمانی کار می کند که تصمیم جاری بر مبنای تنها حالت جاری گرفته شده باشد. یعنی تنها زمانی میسر است که محیط کاملاً قابل مشاهده باشد. حتی کمی از عدم مشاهدهٔ محیط می تواند مشکلات جدی به وجود آورد. برای مثال، قانون ترمز قبلی فرضی کرد که ترمز کردن ماشین جلویی می تواند از طریق حالت جاری تعیین شود و دوربین به گونه ای طراحی شده که فرض می کند چراغ

خطر در مرکز ماشین قرار داده شده است. متأسفانه ماشین های مدل قدیمی چیدمان متفاوتی از نوع چراغ های عقب، رنگ چراغ خطر و تغییر رنگ چراغ دارند و نمی توان از یک تصویر واحد فهمید ترمز کرده اند. عامل واکنشی ساده در حال رانندگی که پشت چنین ماشینی است ممکن است دائماً و بدون دلیل ترمز کند و یا بدتر از آن اصلاً ترمز نکند.

- **عامل های واکنشی مدل گرا Model- Based Agent**

بهترین راه مدیریت نیمه قابل مشاهده ، برای عامل آن است که بخشی از دنیایی را که نمی تواند الان ببیند در حافظه نگه دارد. یعنی عامل باید نوعی **حالت داخلی** وابسته به تاریخچه ادراک داشته باشد و از طریق آن حداقل برخی جنبه های غیر قابل مشاهدهٔ حالت جاری را باز نماید. برای مسئله ترمز کردن، حالت داخلی زیاد هزینه بر نیست. فقط آخرین تصویر دوربین لازم است تا عامل کنترل کند دو نور ترمز در کناره های خودرو به طور مداوم خاموش و روشن می شوند یا خیر. برای دیگر وظایف رانندگی همانند تغییر خط رانندگی، عامل نیاز به ذخیره کردن وضعیت دیگر ماشین ها در زمان رویت آنها را دارد. این حالات داخلی مرتباً باید refresh شوند. چون زمان در حال سپری شدن است و از آن جائی که حافظه ی آن محدود است مرتباً باید زمان های قبلی حذف شوند.

عاملی که از چنین مدلی استفاده کند، **عامل مدل گرا** نامیده می شود. شکل ۱-۷ ساختار عامل واکنشی با حالت داخلی و برنامه عامل را می دهد. این شکل نمایانگر چگونگی ترکیب ادراک کنونی با حالت داخلی قدیمی جهت تولید توصیف به هنگام شدهٔ حالت جاری است. بخش قابل توجه تابع update- State است که مسئول ایجاد توصیف حالت داخلی جدید است. به همان میزان که ادراک جدید را تفسیر می کند، از اطلاعات دربارهٔ چگونگی تغییرات دنیا جهت حفظ بخش های دیده نشدنی دنیا نیز استفاده می کند و اغلب باید دربارهٔ عمل های عامل برای حالت دنیا نیز اطلاعات کسب کند.



شکل ۱-۷: عامل واکنشی مدل گرا

function REFLEX- AGENT- WITH- STATE (*percept*) **return** an action

static: *state*, a description of the current world state

rules, a set of condition- action rules

action, the most recent action, initially none

state ← UPDATE- STATE (*state*, *action*, *percept*)

rule ← RULE- MATCH (*state*, *rules*)

action ← RULE- ACTION [*rule*]

return action

شکل ۱-۸: یک عامل واکنشی مدل گرا. حالت جاری دنیا را به کمک یک مدل داخلی حفظ می کند. سپس یک عمل را همانند روش عامل واکنشی انتخاب می کند.

جدول look up آن به صورت زیر است:

State	Action

$S_{t-n},...,S_{t-1},S_t$	action 1

دو نوع عامل قبلی که از جدول look-up استفاده می کنند اشکالات زیر را دارند:

- حجم زیاد جدول
- استقلال agent صفر است.
- بعضی از قوانین باید به روز شوند. مثلاً در مثال رانندهٔ تاکسی: قوانین که مربوط به مسیر حرکت باشد با سوار شدن مسافرین جدید باید به روز شوند.

- **عامل های هدف گرا Goal- Based Agent**

دانستن دربارهٔ وضعیت کنونی محیط برای تصمیم گیری عمل نمی تواند کافی باشد. برای مثال، در یک چهارراه، تاکسی می تواند به چپ، راست یا مستقیم تغییر مسیر دهد. تصمیم بستگی به مقصد تاکسی دارد. به عبارت دیگر، به همان گونه که عامل نیازمند شرح وضعیت جاری است، به نوعی نیازمند اطلاعات **هدف** است که توضیح موقعیت مطلوب است. برای مثال، می توان به مقصد مسافر اشاره نمود. برنامه عامل می تواند این اطلاعات را با اطلاعاتی دربارهٔ نتایج اعمال ممکن (همانند اطلاعاتی که در عامل واکنش برای به هنگام سازی وضعیت داخلی استفاده شد) ترکیب کند تا اعمال مناسب را برای دسترسی به هدف انتخاب نماید.

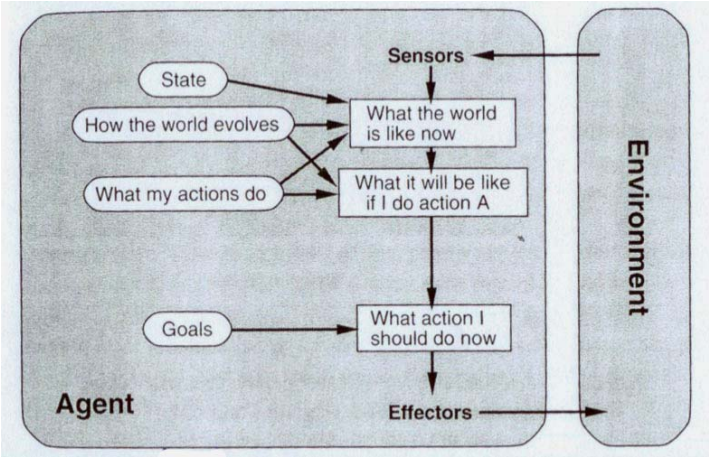
به طور کلی دو نوع مسئله مطرح است:

- Single- Action- Problem : مسائلی که عامل با انجام دادن یک عمل از بین عملهای ممکن بتواند به هدف برسد.

- Multiple- Action- Problem : مسائلی که در آن ها عامل برای رسیدن به هدف نیاز به انجام چند action پی در پی داشته باشد که این مسائل سخت تر هستند.

مثلاً در عامل راننده تاکسی: قبل از شروع حرکت، عامل یک درخت از مسیرها می سازد و سپس مسیری را که مسافر را از مقصد به هدف می رساند انتخاب می کند. به این تکنیک جستجو می گوئیم که در فصلهای بعدی توضیح داده خواهد شد و توسط عامل های هدف گرا استفاده می شود. این عامل حتماً مسیر بهینه را انتخاب نمی کند. روش دیگر برای رسیدن به هدف از طریق بررسی مسیرهاست که planning نام دارد.

اگر چه عامل هدف گرا ناکارا نشان داده می شود، بسیار انعطاف پذیر است. اگر باران شروع شود، عامل میتواند دانش خود را در مورد چگونگی کیفیت ترمز به هنگام سازد. این فرآیند به طور خودکار باعث می گردد که تمامی رفتارهای مربوط برای وفق دادن با شرایط جدید تغییر پیدا کنند. برای عامل واکنشی، ما مجبور به دوباره نویسی تعداد زیادی قوانین شرط- عمل خواهیم بود.

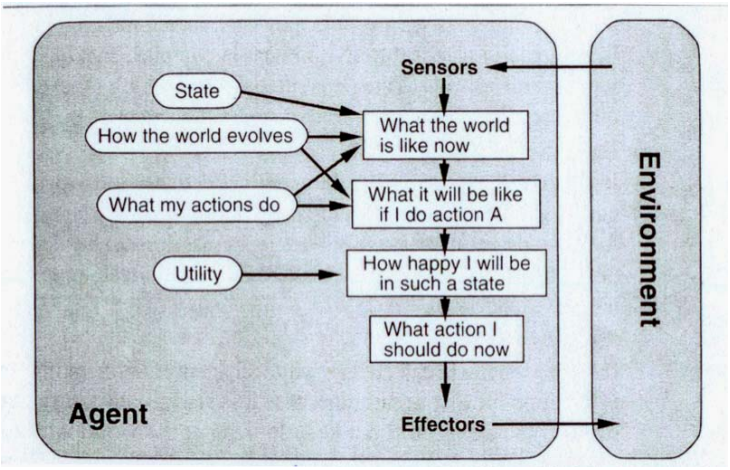


شکل ۱-۹: عامل هدف گرا و مدل گرا. حالت دنیا را همانند مجموعهٔ اهداف که قصد رسیدن به آنها را دارد حفظ می کند و عملی را انتخاب می کند که اهدافش را نزدیکتر سازد.

- **عامل های سودمند Ability- Based Agent**

اهداف به تنهایی برای تولید رفتار با کیفیت بالا کافی نیستند. برای مثال، دنباله های عملیات زیادی وجود دارند که تاکسی را به مقصد می رساند، اما برخی سریع تر، امن تر، مطمئن یا ارزان تر از دیگران هستند. تنها

تفاوت این روش با روش قبلی در این است که این روش از بین راه های ممکن بهترین مسیر را که کم هزینه تر باشد را انتخاب می کند. یعنی مسیری با حداکثر سودمندی و حداقل طول مسیر. این نوع عامل تابعی بنام سودمندی دارد که به هر مسیر اولویت می دهد. بنابراین سودمندی تابعی است که یک وضعیت را به عدد حقیقی نگاشت می دهد، که درجهٔ رضایت مربوط را تشریح می کند. مشخصات کامل تابع سودمندی امکان تصمیم گیری های منطقی را برای دو نوع حالتی که هدف مشکل دارد، مجاز می سازد. اول، زمانی که اهداف متناقض وجود دارند، تنها برخی از آن قابل حصول خواهد بود (برای مثال، سرعت و امنیت) و تابع سودمندی مصالحهٔ مناسب را معین می کند. دوم، زمانی که چندین هدف وجود دارد که عامل می تواند آنها را هدف قرار دهد و هیچکدام از آنها با قطعیت قابل حصول نیست. سودمندی راهی را ایجاد می کند که در آن احتمال موفقیت در مقابل اهمیت هدف باید مقایسه شود. ساختار کلی عامل سودمند در شکل زیر نشان داده شده است.



شکل ۱ – ۱۰: یک عامل سودمندگرا، مدل گرا، از مدل دنیا به همراه تابع سودمندی استفاده می کند که ترجیحات را در بین حالات دنیا اندازه گیری می کند. سپس عملی را که منجر به بهترین سودمندی مورد نظر می شود انتخاب خواهد کرد . سودمندی مورد انتظار از طریق متوسط گیری روی تمامی حالات خروجی ممکن با وزن احتمال خروجی محاسبه می شود.

عامل یادگیری

یادگیری می تواند به عامل اجازه دهد تا در محیط ابتدا ناشناخته عمل کرده و نسبت به دانش اولیه خود پیشرفت کند. ما به دنبال طراحی عاملهایی هستیم که قابلیت یادگیری داشته باشند. در این حالت باید برای

انجام عمل ها از محیط یک باز خورد بگیریم که نتیجه آن می تواند تشویق (reward) و یا تنبیه (punishment) باشد.

به جدول مثال زیر توجه کنید:

دانش اولیه

percept	action
A	
B	
C	
D	
...	

جدول ۱-۴

اگر طراح می خواست عامل را طراحی کند باید قسمت عمل را برای هر کدام از ادراک های موجود A تا D خودش پر می کرد. مثلاً اگر در موقعیت A باشیم و سه عمل چپ- راست- جلو را داشته باشیم یکی از عملها را به صورت random انتخاب می کند. سپس بازخورد محیطی می گیرد و اگر حاصل تنبیه بود به سراغ عمل بعدی می رود و تا زمانی که به حالت تشویق برسد، عمل را عوض می کند. بنابراین با جایگزین کردن عملها یاد می گیرد که در مقابل هر ادراکی چه عملی مناسب است. شکل زیر ساختار عامل یادگیری را نشان می دهد. در این ساختار ۴ مؤلفه مفهومی وجود دارد.

- **عنصر کارائی**: مسئول انتخاب اعمال است. این عنصر چیزی است که در توضیح عاملهای قبلی به عنوان کل عامل در نظر گرفته می شد. (ادراک را دریافت کرده و عمل را تصمیم گیری می کند).
- **عنصر نقد**: عملکرد عامل را نقد می کند.
- **عنصر یادگیری**: با نقد عملکرد عامل از بازخوردها، تغییرات در جهت بهتر کردن عنصر کارایی در آینده را تعیین می کند.

- **عنصر تولید کننده مسئله**: مسئول پیشنهاد عمل هائی است که به تجربیات جدید و مفید منجر می شود.

در مثال تاکسی خودکار: تاکسی بر مبنای عنصر کارایی در خیابان ها می راند و عنصر کارایی شامل مجموعه دانش و روال های تاکسی برای انتخاب عمل ها است. اما عنصر کارایی مرتباً به اطلاعاتی که از عنصر یادگیری می رسد و عملهای پیشنهادی عنصر تولید کننده مسئله توجه می کند. مثلاً اگر تاکسی یک انحراف به چپ سریع داشته باشد، عنصر نقد انتقاد دیگر رانندگان را مشاهده می کند و به عنوان بازخورد به عنصر یادگیری می دهد. از این تجربه، عنصر یادگیری قانون جدید وضع می کند و تحویل عنصر کارایی می دهد. همچنین تولید کننده مسئله ممکن است عمل هائی را که نیاز به توسعه دارد را تعیین کرده و تجربیاتی را پیشنهاد کند. مثلاً آزمون ترمز کردن در شرایط متفاوت.