تاسیس ۱۳۰۷

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده‌ی مهندسی نقشه‌برداری (ژئودزی و ژئوماتیک)

عامل‌های رایانه‌ای

نویسندگان:

دکتر علی اصغر آل شیخ

دکتر سعید بهزادی

پاییز ۱۳۹۲

سرشناسه : آل شیخ، علی‌اصغر، ۱۳۴۴ -
عنوان و نام پدیدآور : عامل‌های رایانه‌ای/ نویسندگان علی‌اصغر آل‌شیخ، سعید بهزادی.
مشخصات نشر : تهران: دانشگاه صنعتی خواجه نصیرالدین طوسی، انتشارات، ۱۳۹۲.
مشخصات ظاهری : د، ۲۲۸ ص.: مصور (بخشی رنگی)، جدول، نمودار.
شابک : ۹۰۰۰۰ ریال 978-600-6383-57-6:
وضعیت فهرست نویسی : فیپا
یادداشت : واژه نامه.
یادداشت : کتابنامه.
یادداشت : نمایه.
موضوع : عامل‌های هوشمند (نرم‌افزار کامپیوتر)
موضوع : طراحی منطقی
شناسه افزوده : دانشگاه صنعتی خواجه نصیر الدین طوسی
رده‌بندی کنگره : QA۷۶/۹۲۷ ۱۳۹۲
رده‌بندی دیویی : ۰۰۶/۳
شماره کتابشناسی ملی : ۳۳۴۱۵۷۹

نام کتاب: عامل های رایانه ای

نویسندگان : دکتر علی اصغر آل شیخ، عضو هیأت علمی دانشکده مهندسی نقشه برداری، دانشگاه صنعتی خواجه نصیرالدین طوسی، دکتر سعید بهزادی

ناشر: انتشارات دانشگاه صنعتی خواجه نصیرالدین طوسی

ویراستار ادبی: نسیم صفری

نوبت چاپ: اول

تاریخ چاپ: آذرماه ۱۳۹۲

تیراژ: ۵۰۰ نسخه

قیمت: **۱۰۵۰۰ تومان**

کد کتاب: ۳۴۳

شابک: ۹۷۸-۹۶۴-۸۷۰۳-۵۷-۶

ISBN : 978-964-8703-57-6

لیتوگرافی: هوررنگ

صحافی: گرنامی

آدرس و تلفن مرکز پخش و فروش: خیابان ولیعصر(عج)، بالاتر از میدان ونک، تقاطع میرداماد، روبروی ساختمان اسکان (۸۸۷۷۲۲۷۷-۰۲۱)

فهرست مطالب

عنوان — شماره صفحه

فهرست اشکال

فهرست جداول

مقدمه

۱-۱- مقدمه

هوش و خرد از الطاف و موهبت‌های الهی به انسان‌ها است و آدمیان، همین امر را دستمایهٔ برتری خود بر سایر موجودات کرهٔ خاکی قرار داده‌اند و با تکیه بر این قدرت بی-نظیر، سیادت و چیرگی خود را بر زمین استوار نموده‌اند. بشر برای رفع نیازهای خویش و بهره‌گیری بهتر از طبیعت، با استفاده از هوش و خلاقیت خود، دامنه عظیمی از ابزارها را اختراع کرد و به کار گرفت. گرچه مغز انسان دستگاه محاسباتی بی‌نظیر و خارق‌العاده‌ایست ولی از لحاظ حافظه و سرعت پردازش محدود می باشد. از این رو رایانه با سرعت و قدرت پردازش بی‌بدیل خود از جمله اختراعات سودمند بشری است که تحولی شگرف را در فعالیت‌های بشری پدیدار ساخته است. با فراهم شدن این ابزار توانا، انسان در صدد برآمد تا قوهٔ هوش خود را به ابزارهای پردازشی ساخته شده نیز انتقال دهد. از همین جا موضوع هوش مصنوعی که می‌توان از آن به عنوان دریچه‌ای جدید به علم و فناوری یاد کرد، پا به عرصهٔ وجود نهاد. گرچه عبارت هوش مصنوعی اولین بار در سال ۱۹۵۶ میلادی و پس از جنگ جهانی دوم به کار رفت، ولی این ایده تا زمان پیدایش و فراگیر شدن رایانه مجالی برای بروز و تبلور نیافت.

هم‌اکنون هوش مصنوعی گسترهٔ وسیعی از علوم را در برگرفته که محدوده آن از یادگیری و ادراک گرفته تا شطرنج، از اثبات نظریه‌ای ریاضی گرفته تا نوشتن شعر و از تشخیص بیماری‌ها تا جنگ‌های الکترونیک گسترش یافته است. هوش مصنوعی قادر به انجام امور به صورت خودکار و خودگردان است و نقشی مشابه فعالیت‌های هوشمند انسان بازی می کند. به همین سبب می‌توان دامنهٔ کاربرد هوش مصنوعی را جهانی دانست.

۱-۲- هوش مصنوعی

تعاریف متعددی از هوش مصنوعی ارائه شده است که تعدادی از آنها در جدول ۱-۱ آمده است. این تعاریف را می‌توان به چهار دسته‌ی اصلی تقسیم نمود. تعریف‌های موجود در نیمه بالایی جدول مربوط به فرآیند تفکر[1] و استدلال[2] است و نیمه پایین جدول تعاریف مرتبط با رفتار[3] را بیان می‌دارند. تعاریفی که در سمت چپ جدول ارائه شده‌اند مربوط به کارایی انسان[4] می‌باشد. این در حالیست که تعاریف سمت راست جدول برگرفته از مفاهیم عقلانیت[5] است؛ زمانی که یک سامانه درست‌ترین کار را انجام دهد آن‌را سامانه‌ای معقول می نامند.

جدول ۱-۱- تعاریف مختلف از هوش مصنوعی

سامانه ای که مثل انسان فکر می‌کند:	سامانه‌ای که عاقلانه فکر می‌کند:
• «تلاش برای ساخت رایانه‌هایی که فکر کنند...» «فعالیت‌هایی همانند تصمیم-گیری، حل مسئله، و یادگیری نمونه-هایی هستند که به فکر انسان مربوطند»	• «مطالعه استعدادهای ذهنی با استفاده از مدل‌های محاسباتی» • «مطالعه محاسباتی که از طریق آن بتوان ادراک، استدلال و عمل نمود»
سامانه‌ای که همانند هوش انسان عمل می‌کند:	**سامانه‌ای که به صورت عاقلانه عمل می‌کند:**
• « ایجاد فرآیندی که در زمان اجرا نیاز به هوش داشته باشد» • «مطالعه و ساخت رایانه‌هایی که کارهایی شبیه انسان انجام دهند.»	• «هوش محاسباتی، روش طراحی عامل-های هوشمند است» • «هوش مصنوعی با رفتارهای هوشمند در مصنوعات مرتبط است»

در ادامه به بیان جزئیات هر یک از گروه‌های فوق پرداخته می‌شود:

[1] Thought Processes
[2] Reasoning
[3] Behavior
[4] Human Performance
[5] Rationality

روش آزمون تورینگ[1]

آزمون تورینگ توسط آلن تورینگ[2] در نوشتاری به نام «محاسبات ماشینی و هوشمند» در سال ۱۹۵۰ برای سنجش میزان هوشمندی ماشین ارائه شد. در این آزمون شرایط به گونه‌ای فراهم می‌شود تا انسان با ماشین (رایانه) تعامل داشته باشد و پرسش‌هایی را برای بررسی هوشمندی ماشین طرح نماید. چنانچه در پایان آزمایش، رایانه نتواند تشخیص دهد که تعامل با انسان صورت گرفته یا با ماشینی دیگر، تست تورینگ با موفقیت انجام شده است. بررسی میزان هوشمندی یک سامانه می‌تواند به فرآیند شبیه-سازی اعمال انسان کمک نماید. برای اینکه رایانه بتواند به صورت هوشمند عمل کند، باید اجزاء ذیل را دارا باشد.

پردازش زبان طبیعی[3]: تا بتواند با انسان ارتباطی مناسب برقرار کند.

نمایش معلومات[4]: تا بتواند آنچه را که می‌داند و می‌شنود ذخیره نماید.

استدلال خودکار[5]: تا با استفاده از اطلاعات ذخیره شده، پاسخ به پرسش‌ها و گرفتن نتایج جدید بطور خودکار استدلال کند.

یادگیری ماشینی[6]: تا قابلیت پذیرش شرایط جدید و همچنین اکتشاف الگو را در محیط داشته باشد.

دید رایانه‌ای[7]: تا بتواند به کمک آن اشیاء را درک نماید.

روباتیک[8]: تا امکان طراحی و تولید آدم ماشینی را فراهم سازد.

هم اکنون بسیاری از محققان هوش مصنوعی تلاش‌های خود را صرف قبولی در آزمون تورینگ کرده‌اند. محققین این حوزه باور دارند که قبولی در این آزمون بسیار مهم‌تر از بررسی اصول هوش مصنوعی می‌باشد.

آزمون تورینگ فرض را بر آن می‌نهد که انسان‌ها می‌توانند با مقایسه‌ی میان رفتار ماشین و انسان به میزان هوشمندی آن پی ببرند. به دلیل این فرض و تعدادی پیش فرض

---

[1] Turing Test
[2] Alan Turing
[3] Natural Language Processing
[4] Knowledge Representation
[5] Automated Reasoning
[6] Machine Learning
[7] Computer Vision
[8] Robotics

دیگر، برخی از دانشمندان حوزه‌ی هوش مصنوعی، صحت آزمون تورینگ را مورد تردید قرار داده اند.

تفکر به شیوه انسانی

در صورتی می‌توان ادعا کرد یک برنامه رایانه‌ای شبیه انسان عمل می‌کند که بتوان به طریقی آن را ثابت نمود. برای اثبات این فرضیه، ابتدا باید مشخص کرد که در ذهن انسان چه می‌گذرد. برای بررسی این مسئله دو راه وجود دارد. راه نخست بر مبنای مدل‌سازی فرآیند فکر کردن انسان (درون‌گرایی) می‌باشد. راه دوم با ارزیابی تجربیات روان‌شناسی تحقق می‌یابد. پس از مشخص شدن رفتار ذهن انسان، می توان آن‌را با برنامه‌های رایانه‌ای شبیه‌سازی کرد. در صورتی که ورودی، خروجی و رفتار این برنامه، تقلیدی از رفتار انسان باشد، از برنامه رایانه‌ای به جای انسان استفاده خواهد شد. برای مثال آلن نیوول[1] و هلبرسیمن[2] تکنیک حل مسائل عمومی[3] یا همان GPS را ابداع کردند. آنها علاوه بر یافتن راه‌حل درست مسئله به دنبال مراحل استدلال رایانه و مقایسه آن با مراحل استدلال انسان بودند. علم شناخت[4] که علمی میان رشته‌ای می‌باشد مدل‌های رایانه‌ای را از هوش مصنوعی و تکنیک‌های تجربی را از روانشناسی گرفته و به کمک آنها تئوری‌های آزمون‌پذیر و دقیقی از کارکرد ذهنی انسان ارائه می‌کند.

تفکر به شیوه معقول و منطقی

ارسطو، فیلسوف یونانی، از اولین کسانی بود که تلاش کرد تفکر منطقی را تدوین نماید. قیاس یا قیاس منطقی[5]، استدلالی از کل به جزء است که اگر مقدمه‌های آن درست باشند نتیجه‌ی به دست آمده حتما درست است. برای مثال جمله‌ی «سقراط یک مرد است؛ همه مردها فنا شدنی هستند، بنابراین سقراط فنا شدنی است» نمونه‌ای از قیاس منطقی است. از قواعد تفکر معقولانه می‌توان برای ارزیابی عملیات ذهنی انسان استفاده کرد. مطالعه عمیق این قواعد، حوزه‌ای جدید به نام منطق[6] را به وجود آورده است.

علمای علم منطق از قرن ۱۹ میلادی بر آن شدند تا پدیده‌های مختلف را به صورت منطقی بیان نمایند. در سال ۱۹۶۵ دانشمندان موفق به ارائه برنامه‌هایی شدند که قادر بودند هر مسئله‌ای را که به زبان منطقی در آمده باشد، حل کنند. از این‌رو می‌توان امیدوار

---

[1] Alen Newell
[2] Helbert Siman
[3] General Problem Solver
[4] Cognitive Science
[5] Syllogism
[6] Logic

بود که با ترکیب منطق و هوش مصنوعی بتوان سامانه‌های هوشمند را طراحی کرد و به اجرا در آورد.

برای کمک به تحقق این اهداف دو مشکل اصلی وجود دارد: اول اینکه، در نمادگذاری منطقی به عبارات قاعده‌مند نیاز است. تبدیل اطلاعات بی‌قاعده به فرم‌های قاعده‌مند کار آسانی نیست. دوم اینکه، بین توانایی حل مسئله و انجام دادن آن تفاوت عمده‌ای وجود دارد.

عمل کردن به صورت عاقلانه

واژه عامل[1] که از کلمه لاتین ager به معنای عمل کردن گرفته شده است به ماهیتی اطلاق می‌گردد که دارای خاصیت کنش باشد. عامل‌های کامپیوتری بایستی دارای ویژگی-های خاصی باشند تا بتوان آنها را از عامل‌های معمولی متمایز ساخت. عامل‌های منطقی[2] عامل‌هایی هستند که همواره دنبال یافتن بهترین جواب‌اند و در صورت برخورد با موارد غیر قطعی، این گونه عامل‌ها به دنبال بهترین نتیجه ممکن می‌باشند. لذا می‌توان گفت که عامل‌های منطقی به دنبال یافتن جواب‌های معقول می‌باشند.

در فرآیند منطقی اندیشی، تأکید اصلی بر روی استدلال‌های درست می‌باشد. آوردن یک استدلال درست، بخشی از ساختار عامل‌های معقول است. زیرا یک راه انجام اعمال معقول از طریق استدلال‌های منطقی است که بر اساس آن نتیجه عملی که قرار است انجام شود تجزیه و تحلیل می‌گردد. از طرف دیگر باید توجه نمود که صرف یک استدلال درست به تنهایی نشانه عقلانیت نمی‌باشد، زیرا این امکان وجود دارد که هیچ راه حل قطعی برای انجام یک عمل وجود نداشته باشد و حال آنکه، عامل مجبور است برای حل مسئله عملی را انجام دهد. همچنین عمل‌های معقول دیگری وجود دارند که از استدلال نشأت نگرفته‌اند. واکنش پس کشیدن دست بعد از تماس با یک شئ داغ به صورت سریع مثالی از این مطلب می‌باشد که معمولاً نتیجه آن بهتر از یک عکس‌العمل کند و پس از انجام مشورت است.

از طرف دیگر وجود تمام مهارت‌های مورد نیاز در آزمون تورینگ برای انجام اعمال معقول ضروریست. بنابراین نیاز است که تحلیل‌های مربوط به استدلال و دانش را در برگیرد که این امر سبب سهولت تصمیم‌گیری در محیط‌های مختلف می‌شود. بعلاوه باید بتوان جملات قابل فهم و به زبان طبیعی تولید نمود که به ادامه حیات در یک جهان پیچیده کمک کند. یادگیری از دیگر ضرورت‌های تدوین استراتژی مناسب است که در

[1] Agent
[2] Rational Agent

نهایت بر درک بصری مبتنی می‌باشد. نیاز به درک بصری نه تنها برای دیدن، بلکه برای بهتر ایده گرفتن از فرآیند است. برای مثال دیدن یک غذا سبب حرکت به سوی آن می‌شود.

با توجه به این دلایل، مطالعه هوش مصنوعی به عنوان عامل خردمند حداقل دارای دو منفعت است. اول اینکه این روش نسبت به روش قانون تفکر[1] جامع‌تر است. زیرا استدلال صحیح تنها یکی از چند مکانیزم مناسب برای دستیابی به عقلانیت می‌باشد. دوم اینکه این روش نسبت به روش‌هایی که بر اساس رفتار و فکر انسان هستند از نظر توسعه علمی مقبول‌تر ند. زیرا در آن استاندارد عقلانیت تعریف شده و قابل ارزیابی است. از طرف دیگر، رفتار انسان با یک محیط خاص سازگار است و تا حدودی محصول یک فرآیند تکاملی پیچیده است که [illegible] فاصله‌ی زیاد دارد.

تاریخچه محاسبات داده به وسیله پنج رویداد مهم نشان داده می‌شود:

- حاضر، در همه جا و در هر زمان[2]:

با استفاده از این رویه نیروی قدرتمندی به وجود خواهد آمد که قادر است در هر مکان و با هر وسیله‌ای اعمالی را که در گذشته غیر قابل تصور بوده است انجام دهد.

- ارتباط بینابین[3]:

در زمان‌های گذشته، سامانه‌های رایانه‌ای تنها با کاربر در ارتباط بودند، ولی امروزه رایانه‌ها به کمک شبکه با یکدیگر نیز در ارتباط هستند. یک نمونه از این شبکه‌ها، شبکه اینترنت است که امروزه به ندرت می‌توان رایانه‌ای یافت که به آن متصل نباشد. در زمان‌های گذشته رایانه‌ها به صورت مجزا از هم بودند و اتصال بین آن‌ها به نوعی غیر ممکن بود، در حالی که امروزه اتصال به شبکه امری عادی است.

- هوشمندی[4]:

در این حالت مجموعه‌ای از مسایل پیچیده توسط رایانه و به صورت خودکار حل می‌شود و به همراه آن عمل یادگیری نیز صورت می‌گیرد. این در حالی است که در گذشته انجام این کار توسط سامانه‌های رایانه‌ای غیر قابل تصور بود.

---

[1] Law of Thought
[2] Ubiquity
[3] Interconnection
[4] Intelligence
[5] Delegation

- قابلیت واگذاری[5]:

در این رویه کارهایی که حساس است و نیاز به دقت زیادی دارد به سامانه‌های رایانه‌ای واگذار می‌شود. در این‌گونه وظایف معمولاً فرآیند یکنواخت و ثابتی وجود دارد که تنها پیچیدگی موجود در این کارها دقت بالای آن‌ها می‌باشد. یک نمونه از این وظایف کار خلبانی است. البته باید توجه داشت که قابلیت اجرایی این سامانه‌ها برای انسان تایید شده است. در عمل واگذاری، کنترل کار به سامانه‌های رایانه‌ای سپرده می‌شود.

- انسان‌گرایی[1]:

در این رویه سعی بر آن است که سامانه‌ها از حالت ماشین‌گرایی خارج شده و به حالت انسان‌گرایی پیش روند. با انجام این کار درک سامانه‌ها از محیط بیرون شبیه درک انسان می‌شود. در سامانه‌های ماشین‌گرا اعمال داخلی برای اپراتور کاملاً مشخص است و در صورتی که سامانه‌ها، کار خود را درست انجام دهند نتیجه کار برای اپراتور مشخص می‌گردد.

به‌کارگیری این رویه‌ها سبب می‌شود که یک رشته جدید در علوم رایانه‌ای که سامانه چندعامله[2] نامیده می‌شود به وجود آید. ایده سامانه‌های چندعامله یک ایده ساده است. در این حالت عامل همانند یک سامانه رایانه‌ای در نظر گرفته می‌شود که قادر است یک سری کارها را به طور مستقل انجام دهد. به عبارت دیگر عامل به صورت یک وسیله در نظر گرفته می‌شود که می‌خواهد به اهدافش برسد. برای اینکه عامل به اهدافش برسد باید دید که به چه چیزهایی نیاز دارد. در سامانه‌های چندعامله تعدادی از عامل‌ها با هم در تعامل[3] هستند. تعامل بین عامل‌ها از طریق ارسال و دریافت پیام در داخل یک شبکه مشخص امکان‌پذیر است. عامل‌ها در سامانه‌های چندعامله بر اساس مجموعه‌ای از استدلال‌ها و اهداف مختلفی که دارند عمل می‌کنند. برای اینکه هر عامل بتواند به خوبی با عامل دیگر در ارتباط باشد، نیاز به همکاری[4]، هماهنگی[5] و گفتگو کردن[6] با دیگر عامل‌ها دارد. همان طور که مشاهده می‌شود این سه عمل در زندگی روزانه انسانها نیز وجود دارد.

با توجه به آنچه که در اینجا بیان شد، دو مسئله اساسی مطرح می‌گردد:

---

[1] Human-orientation
[2] multi-agent Systems
[3] Interact
[4] Cooperate
[5] Coordinate
[6] Negotiate

۱- قرار است که به یک عامل، کاری واگذار شود. چگونه می‌توان این چنین عاملی را ایجاد نمود که بتواند به صورت مستقل و با انجام عملی مستقل این کار را انجام دهد؟

۲- چگونه می‌توان عاملی را ایجاد کرد که قابلیت تعامل (همکاری، هماهنگی و گفتگو کردن) با دیگر عامل‌ها را داشته باشد و همچنین کاری که به او واگذار می‌شود را در حالتی که دیگر عامل‌ها از انجام دادن آن ناتوان هستند به خوبی انجام دهد؟

مسئله اول که در اینجا مطرح گردید مربوط به طراحی عامل‌ها[1] و مسئله دوم مربوط به طراحی نظام اجتماعی[2] است. این دو مسئله از هم جدا نبوده و با هم در ارتباط هستند. به عنوان مثال برای اینکه بتوان عامل‌های اجتماعی را طوری ساخت که در آن عامل‌ها، کارآمد باشند باید به ساختار عامل نیز توجه نمود.

دید[3] (تصور) عامل:

توجه به چگونگی انجام اعمال روزمره انسان‌ها کار بسیار دشواری است؛ اما اگر بدانیم که انگیزه آن‌ها از انجام این کارها چه بوده است، می‌توان متوجه شد که مردم می‌خواهند چه کاری را انجام دهند. در این بخش بعضی از انگیزه‌ها و دلایل انجام اعمال در اجتماع عامل‌ها بیان می‌شود. این دلایل و انگیزه‌ها در قالب تصورات دراز مدت عامل مطرح می‌شوند. بنابراین بایستی مشخص شود که تصور عامل از محیط پیرامونش در حال حاضر چه می‌باشد و از نگاه عامل این محیط در آینده چگونه ممکن است باشد. به این تصور عامل، به اصطلاح دید عامل گفته می‌شود.

کاربرد عامل در حوزه‌های مختلف

سامانه‌های چندعامله در علوم میان رشته‌ای قرار دارند و در رشته‌های مختلف علمی بکار می‌روند. از این‌گونه سامانه‌ها در زمینه‌های علمی متفاوت همچون اقتصاد، فلسفه، منطق، بوم‌شناسی و علوم اجتماعی استفاده می‌گردد. باید توجه داشت که هرکدام از این زمینه‌ها دیدگاه‌های مختلفی نسبت به سامانه‌های چندعامله دارند که در این بخش به بعضی از آن‌ها اشاره خواهد شد.

- عامل به عنوان یک الگو برای مهندسی نرم افزار

---

[1] Agent Design
[2] Society Design
[3] Vision

پیچیدگی برنامه از جمله مشخصاتی است که مهندسان نرم‌افزار، روزانه با آن مواجه‌اند. تعامل و برقراری ارتباط از مهم‌ترین مشخصات نرم‌افزارهای پیچیده است. موضوع معماری نرم‌افزار که شامل تعامل بین مؤلفه‌های مختلف و پروتکل‌های پیچیده است به طور قابل ملاحظه‌ای وقت مهندسین امر را می‌گیرد. متأسفانه بسیاری از کاربردها و فعالیت‌های جهان واقعی صراحتاً دارای چنین ویژگی‌هایی هستند. از این‌رو تحقیقاتی که طی دو دهه اخیر در علوم رایانه صورت گرفته است به توسعه ابزارها و تکنیک‌هایی پرداخته که می‌تواند چنین تعاملاتی را مدل‌سازی نموده و سپس آنها را قابل فهم و در انتها اجرا نماید. در واقع امروزه بسیاری از محققین بر این باورند که در آینده‌ای نزدیک، محاسبات[1] فرآیندی از تعامل قابل فهم خواهد بود. همان طور که در ابتدای این فصل اشاره شد، روند غالب محاسبات در بین سامانه‌های رایانه‌ای که با یکدیگر متصل بوده و در همه جا حضور دارند، خواهد بود.

- عامل به عنوان وسیله‌ای برای فهم جامعه بشری

اخیرا علمی نوین با نام psychohistory به وجود آمده است که ترکیبی از روان‌شناسی[2]، تاریخ و اقتصاد است. به کمک این شاخه علمی می‌توان رفتار انسان را تا صد سال آینده پیش بینی نمود. این علم قادر است تا سقوط قریب‌الوقوع انسان را در زمینه اجتماعی پیش‌بینی کند. این ایده تنها در حوزه‌های رمان‌های علمی و تخیلی وجود داشت. البته تا رسیدن به این مرحله که بتوان آینده اجتماع انسانی را دقیقاً پیش‌بینی کرد هنوز فاصله زیادی وجود دارد و در حال حاضر فقط می‌توان در این زمینه پیش‌بینی‌هایی نمود که تعداد انگشت شماری از آنها می‌توانند درست باشند. این واقعیت احتمالاً تا مدت‌ها تغییر نخواهد کرد. با وجود این، سامانه‌های چندعامله یک وسیله جدید و قابل توجه را برای شبیه‌سازی اجتماع انسان‌ها به وجود می‌آورند که می‌توانند ما را در پردازش‌های اجتماعی یاری رسانند.

معایب سامانه های چندعامله

در این بخش به بیان معایب سامانه‌های چندعامله می‌پردازیم و سپس به هرکدام از ایرادها پاسخی داده می‌شود.

آیا سامانه‌های چندعامله همان سامانه‌های همزمان/توزیع یافته هستند؟

انجمن سامانه‌های چندعامله برای چندین دهه سامانه‌هایی که دارای مؤلفه‌های تعامل‌پذیر چندگانه بودند را مورد بررسی قرار داده و علاوه بر آن، ابزار، زبان‌های برنامه-

[1] Computations [2] Psychology

نویسی و تئوری‌هایی را برای تشریح و مدل‌سازی آن‌ها ارائه نموده است. به نظر می رسد که سامانه‌های چندعامله کلاسی از سامانه‌های همزمان باشند. در انجمن سامانه‌های توزیع‌یافته نیز افرادی وجود دارند که می‌پرسند: آیا سامانه‌های چندعامله به اندازه کافی متفاوت از سامانه‌های همزمان/توزیع یافته هستند که به عنوان یک موضوع جداگانه مورد بررسی قرار گیرند؟ پاسخ این سوال زمانی اهمیت می‌یابد که بخواهیم یک سامانه چندعامله را طراحی و اجرا کنیم. در چنین مراحلی بسیار مهم و ضروری است که دانش چنین سامانه‌هایی به تجربه سامانه‌های همزمان/توزیع یافته متصل گردد.

سامانه‌های چندعامله و سامانه‌های همزمان در دو مورد با هم متفاوتند:

۱- عامل‌ها به عنوان موجودات خودکاری در نظر گرفته می‌شوند که قادرند درباره آنچه که می‌خواهند انجام دهند به طور مستقل تصمیمات بگیرند. عموماً در چنین سامانه‌هایی، همزمانی و هماهنگی به عنوان بخشی از سامانه در نظر گرفته نمی‌شود، در حالی که این بخش در سامانه‌های همزمان/توزیع یافته الزامی است. با این وجود مکانیزمی نیاز است که این اجازه را به عامل بدهد تا فعالیتش را در زمان اجرا، هماهنگ و همزمان نماید.

۲- عناصر محاسباتی در سامانه‌های چندعامله بر اساس منفعت شخصی طرح‌ریزی شده است در حالی که در سامانه‌های همزمان/توزیع‌یافته، عناصر محاسباتی به گونه‌ای فرض می‌شوند که همگی یک هدف مشترک داشته باشند. به این معنی که کل سامانه به درستی کار کند و به هدف مشخصی برسد. در عوض در سامانه‌های چندعامله این‌گونه فرض می‌شود که هر عامل در اصل، نگران رفاه و آسایش خود می‌باشد. با این وجود در راستای اهداف عامل‌های دیگر نیز عمل می نماید.

بر اساس دلایل گفته شده، موضوعاتی که در انجمن سامانه‌های چندعامله مورد مطالعه قرار می‌گیرد با موضوعات مورد مطالعه در انجمن سامانه‌های همزمان/توزیع‌یافته متفاوت است. در این بین آنچه که اهمیت دارد این است که عامل‌ها چگونه از طریق گفتگو بر موضوعات مورد علاقه خودشان به سازش رسیده و همچنین چگونه فعالیتشان را با دیگر عامل‌ها هماهنگ می‌کنند؛ در حالی که انگیزه و اهداف متفاوتی نسبت به هم دارند.

آیا سامانه‌های چندعامله همان هوش مصنوعی هستند؟

حوزه سامانه‌های چندعامله ارتباط بسیار نزدیکی با حوزه هوش مصنوعی دارد. در واقع تا مدت‌ها سامانه‌های چندعامله به عنوان زیر مجموعه‌ای از هوش مصنوعی در نظر

گرفته می‌شد، این در حالی است که محققین سامانه‌های چندعامله معتقدند هوش مصنوعی زیر مجموعه ای از سامانه های چندعامله است. در اینجا چندین نکته مهم وجود دارد که باید به آن اشاره شود:

- اولاً، مفاهیم هوش مصنوعی به عناصر هوش بستگی دارد. هوش را می‌توان قابلیت یادگیری، طراحی و فهمیدن تصاویر دانست. در مقابل، حوزه عامل با موجودیت‌هایی در ارتباط است که ترکیبی از عناصر تشکیل‌دهنده هوش بوده و هدف آن ایجاد ماشینی است که بتواند تصمیمات مستقلی بگیرد. برای ساختن یک عامل نیاز است که تمام مشکلات در ایجاد عناصر هوش مصنوعی حل گردد؛ زیرا عامل نیاز به یادگیری و طراحی دارد. به عبارت دیگر می‌توان گفت که عامل‌های هوشمند، نود و نه درصد علم رایانه و یک درصد هوش مصنوعی را شامل می‌شود. زمانی که عامل ساخته می‌شود تا کاری را در یک محیط انجام دهد لازم است انواع مختلفی از اجزای هوش مصنوعی به کار گرفته شوند. به طور خلاصه، تکنیک‌های هوش مصنوعی در جهت ساخت عامل‌ها به کار گرفته می‌شوند و نیازی نیست که تمام مشکلات مربوط به هوش مصنوعی را برای ساختن عامل حل نمود.
- ثانیاً، هوش مصنوعی کلاسیک جنبه اجتماعی بودن عامل را نادیده می‌گیرد. در هوش مصنوعی قابلیت یادگیری و حل مسئله وجود دارد ولی قابلیت همکاری، ارتباط و رسیدن به توافق در آن نادیده گرفته شده است. این قابلیت‌های اجتماعی به همان اندازه که در رفتارهای هوشمند اهمیت دارند در مؤلفه‌های هوش همانند طراحی و یادگیری نیز دارای اهمیت می‌باشند، بنابراین معمولاً در هوش مصنوعی مورد مطالعه قرار نمی‌گیرند.

آیا سامانه‌های چندعامله همان نظریه اقتصادی یا نظریه بازی‌ها هستند؟

نظریه بازی‌ها یک نظریه ریاضی است که در آن تعاملات بین عامل‌های منفعت‌طلب مورد بررسی قرار می‌گیرد. نظریه بازی‌ها تا اندازه زیادی به کمک اقتصاددانانی آمده است که علاقه‌مند هستند با مطالعه این نظریه تعامل بین موجودیت‌های اقتصادی را در جهان واقعی مورد مطالعه قرار دهند. اخیراً تکنیک‌های مربوط به نظریه بازی‌ها، در تحقیقات سامانه‌های چندعامله محاسباتی، به خصوص آن‌هایی که مسئله گفتگو را مطرح می‌کنند به کار گرفته شده‌اند. امروزه از نظریه بازی‌ها به عنوان ابزار نظری غالب در سامانه‌های چندعامله استفاده می‌شود. سؤال دیگری که در اینجا مطرح می‌شود این است که آیا

سامانه‌های چندعامله به عنوان نظریه اقتصادی یا نظریه بازی‌ها در نظر گرفته می‌شوند یا خیر. در اینجا باید به دو نکته توجه نمود:

- اولاً، بسیاری از مفاهیم مربوط به راه حل مسئله که در نظریه بازی‌ها توسعه داده شده‌اند، بدون در نظر گرفتن توان محاسباتی ایجاد گشته‌اند. این قضیه منجر به توصیف مفاهیمی شده است که در آن ویژگی‌های راه‌حل بهینه جایگزین چگونگی حل مسئله گشته است. تحقیقات سامانه‌های چندعامله به چگونگی محاسبه می پردازد، آنگاه مسایل به کمک ابزارهای علوم رایانه-ای حل می‌شوند.
- ثانیاً، بعضی از محققین برای اینکه به نتیجه دلخواه خود برسند فرضیه‌هایی از نظریه بازی‌ها را به چالش می‌کشند. مخصوصاً این مسئله در انجمن سامانه‌های چندعامله مورد بحث قرار می‌گیرد که آیا تصور و اندیشه یک عامل خردگرا[1] که برای فهم اجتماع عامل‌های مصنوعی یا انسان معتبر است بایستی در نظریه بازی‌ها، الگو قرار داده شود یا خیر؟

آیا سامانه‌های چندعامله همان علوم اجتماعی هستند؟

موضوع علوم اجتماعی اساساً مرتبط با اجتماع انسان‌ها می‌باشد. بعضی از محققین علوم اجتماعی، علاقه‌مند به سامانه‌های چندعامله هستند تا توسط برخی از ابزارهای این سامانه‌ها بتوانند اجتماع انسان‌ها را مدل‌سازی نمایند. از طرف دیگر یک راه ساده برای طراحی سامانه‌های چندعامله مشاهده رفتار انسان در جامعه بشری و شبیه‌سازی آن می-باشد. با این مقدمه، آیا سامانه‌های چندعامله زیرمجموعه‌ای از علوم اجتماعی هستند؟

اگرچه از اجتماع انسان‌ها در طراحی سامانه‌های چندعامله استفاده می‌شود اما باید توجه نمود که این الگو برداری به طور کامل ممکن نیست. الگو گرفتن واضح و کامل از رفتار اجتماعی انسان‌ها بسیار سخت است، زیرا این رفتار وابستگی زیادی به پارامترهای مختلف دارد. اگرچه طراحی سامانه‌های چندعامله بر اساس مقایسه و تشابه با اجتماع انسان‌ها کاری کاملاً منطقی است، اما این بهترین کار برای طراحی سامانه‌های چندعامله نیست. روش‌های دیگری نیز وجود دارند که می‌توان به کمک آن این طراحی‌ها را انجام داد. یکی از این روش‌ها نظریه بازی‌هاست که در بخش قبل توضیح داده شد.

[1] rational agent

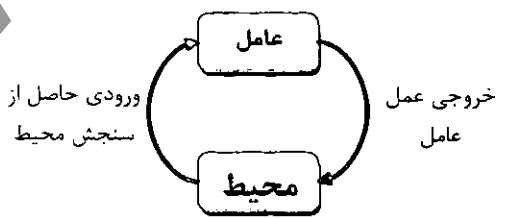
سامانه های چندعامله ابزارهایی توانمند و نوینی برای مدل مدلسازی و درک اجتماع هستند؛ در حالی که علوم اجتماعی منبعی غنی از مفاهیم برای درک و ایجاد سامانه‌های چندعامله بوده و یک رشته علمی مجزا به حساب نمی‌آید.

۱-۳- عامل هوشمند

در تعریف مفهوم عامل دیدگاه‌های متفاوتی وجود دارد ولی همه آن‌ها بر این باورند که عامل، خودمختار[1] است. یکی از دلایل وجود نظرات متفاوت در باره عامل، کاربرد آن در زمینه‌های مختلف می‌باشد. به طور مثال در بعضی کاربردها، یادگیری[2] بسیار اهمیت دارد و در بعضی دیگر چندان مهم نیست. با این وجود جنبه‌های مختلف، تعریف عامل اهمیت زیادی دارد. به طور کلی می‌توان عامل را به صورت زیر تعریف نمود:

«عامل را می‌توان یک سامانه رایانه‌ای دانست که قادر است در داخل یک محیط مشخص قرار گیرد. این سامانه توانایی آن‌را دارد که بر اساس اهدافی که برایش مشخص شده، اعمالی را به طور مستقل در داخل این محیط انجام دهد (ولدریج و جنینگز ۱۹۹۵).»

شکل ۱-۱ یک دید کلی از عامل را نشان می‌دهد. همان طور که در این شکل مشاهده می‌شود عامل یک عمل را انجام می‌دهد که نتیجه آن بر روی محیط اثر می‌گذارد. در بسیاری از مواقع این امکان وجود دارد که عامل به طور کامل بر محیط احاطه نداشته باشد، در این حالت کنترل عامل بر روی محیط، جزیی[3] است. از دیدگاه عامل، این امکان نیز وجود دارد که دو عمل یکسان در دو محیطی که به هم شبیه نیستند، نتایج متفاوتی ارائه دهند. بنابراین این امکان وجود دارد که عامل در رسیدن به هدفش ناکام باشد. پس تمام عامل‌ها باید احتمالی را برای شکست در نظر بگیرند. در چنین مواقعی می‌توان گفت که محیط تعیین‌کننده نیست.



شکل ۱-۱- یک عامل در داخل یک محیط

[1] Autonomy
[2] Learning
[3] Partial

عامل‌ها دارای مجموعه‌ای از اعمال هستند که قابلیت‌های عامل را نمایش داده و می‌توانند محیط را تحت تأثیر قرار دهند. باید توجه نمود که عامل‌ها همواره قادر به انجام همه عمل‌ها نیستند. مثلاً بالا بردن میز زمانی امکان‌پذیر است که وزن آن برای عامل قابل تحمل باشد. بنابراین هر عمل یک سری پیش شرط‌هایی دارد که امکان‌پذیر بودن آن‌را بیان می‌کند.

نکته اصلی که باید به آن توجه نمود این است که عامل کدام عمل را باید انجام دهد بهترین نتیجه حاصل شود. انتخاب بهترین عمل به معماری عامل[1] بستگی دارد. در قسمت زیر مثالی از عامل‌ها و نحوه عملکرد آن‌ها بیان می‌گردد:

مثال: سامانه کنترل حرارتی

هر سامانه کنترلی را می‌توان یک عامل دانست. مثلاً سامانه کنترل حرارتی ترموستات است. ترموستات دارای یک سری سنسور می‌باشد که می‌تواند دمای اتاق را اندازه‌گیری نماید. این سنسورها داخل محیط اتاق قرار گرفته و اطلاعاتی در مورد دمای اتاق دریافت می‌کنند. خروجی ترموستات می‌تواند به یکی از دو حالت زیر بستگی داشته باشد:

۱- وقتی که دما بسیار کم باشد.
۲- وقتی که دما مناسب باشد.

عملی که می‌تواند از این ترموستات سر بزند یکی از دو حالت زیر می‌باشد:

۱- روشن شدن وسیله گرمازا[2] که سبب می‌شود دمای اتاق افزایش یابد.
۲- خاموش شدن وسیله گرمازا[3] که سبب می‌شود دمای اتاق کاهش یابد.

با وجود اجرای این اعمال به هیچ وجه نمی‌توان صحت نتیجه را تضمین نمود. به عنوان نمونه، در صورتی که در اتاق باز باشد تغییر دما توسط ترموستات هیچ تأثیری بر دمای اتاق نخواهد داشت.

تصمیمی را که توسط ترموستات انجام می‌شود می‌توان به صورت زیر بیان نمود:

۱- اگر دمای اتاق کم باشد ← وسیله گرمازا روشن می‌شود.
۲- اگر دمای اتاق مناسب باشد ← وسیله گرمازا خاموش می‌شود.

[1] Agent Architectures
[2] Heating on
[3] Heating off

در صورتی که محیط پیچیده شود عمل تصمیم‌گیری نیز پیچیده می‌گردد.

عامل می‌تواند در داخل محیط‌های فیزیکی (جهان واقعی) یا در محیط‌های مجازی (محیط نرم افزاری) قرار بگیرد. سپس به کمک سنسورها محیط اطراف را درک می‌کند که این سنسورها بسته به محیط می‌تواند فیزیکی یا نرم افزاری باشد. سپس با توجه به اطلاعات دریافتی اعمالی را روی محیط انجام می‌دهند.

۱-۴- محیط

راسل و نورویج ویژگی‌های محیط را به شرح ذیل تقسیم‌بندی می‌نمایند:

قابل دسترس و غیر قابل دسترس: در محیط‌های قابل دسترس عامل می‌تواند اطلاعات کامل، به روز و دقیق را از حالات محیط بدست آورد. هرچه محیط واقعی‌تر باشد غیر قابل دسترس بودن آن بیشتر است و هرچه محیط قابل دسترس باشد، کارایی عامل در آن بالاتر است. یک عامل خوب عاملی است که بتواند تصمیم درستی را برای محیط بگیرد و این تصمیم به اطلاعاتی که عامل از محیط دریافت می‌کند وابسته است. در صورتی که اطلاعات از محیط به صورت غیردقیق و نادرست باشد در این صورت تصمیم عامل ضعیف خواهد بود.

قطعی بودن و غیر قطعی بودن محیط: در صورتی که محیط به صورت قطعی باشد نتیجه عمل اجرا شده بر روی محیط به صورت یک نتیجه مشخص است. در این حالت هیچ‌گونه تردیدی در مورد جواب مسئله وجود ندارد.

غیر قطعی بودن، جنبه‌های مهمی از خصوصیات محیط را در بر می گیرد، که بخشی از آن عبارتند از :

- عامل بر بخشی از محیط احاطه دارد نه بر کل آن: بنابراین کل محیط را درک نمی‌کند و کنترل آن به صورت جزئی است.
- عمل عامل می‌تواند نتیجه مطلوب و مورد نظر را نداشته باشد.

در عمل همه محیط‌های واقعی به صورت غیر قطعی در نظر گرفته می‌شوند. غیرقطعی بودن تا حدود زیادی به پویا بودن سیستم بستگی دارد.

ایستا و پویا بودن: محیط ایستا محیطی است که بدون تغییر می‌ماند مگر اینکه عامل عملی را برای تغییر محیط انجام دهد. در مقابل، محیط پویا محیطی است که در حال

تغییر می‌باشد، و این تغییرات لزوماً توسط عامل صورت نمی‌پذیرند. جهان واقعی را می‌توان به عنوان یک محیط پویا در نظر گرفت.

محیط‌های پویا حداقل دارای دو ویژگی مهم هستند:

۱- در صورتی که عامل بین بازه t_0 و t_1 عملی را انجام ندهد، نمی‌توان تضمین کرد که محیط بین این دو بازه تغییر نکرده است. در این حالت برای اینکه عامل بخواهد یک تصمیم درست در لحظه t_1 بگیرد باید دوباره به جمع‌آوری اطلاعات از محیط بپردازد. در صورتی که در محیط‌های استاتیک نیازی به جمع‌آوری دوباره اطلاعات نیست.

۲- فرآیندهای دیگر در محیط می‌تواند با عمل عامل تداخل داشته باشد.

گسسته و پیوسته بودن: محیطی گسسته است که مجموعه‌ای محدود و مشخص از ادراک[1] و عمل[2] در آن اتفاق بیفتد، درحالی‌که محیط‌های پیوسته مجموعه‌ای نامحدود از ادراکات و اعمال می‌باشند. به طور مثال بازی شطرنج با توجه به محدودیت حالات محتمل بازی (با آنکه تعداد حالات این بازی زیاد می‌باشد) یک محیط گسسته است ولی رانندگی تاکسی با مجموعه‌ای نامحدود از حالات محتمل، مثالی از یک محیط پیوسته می‌باشد.

محیط‌های گسسته برای طراحی عامل‌ها ساده‌ترند. محیط‌های رایانه ای به صورت گسسته می‌باشند. این سامانه‌ها می‌توانند محیط‌های گسسته را با هر مقدار دقت به محیط‌های گسسته تبدیل کنند. البته باید توجه نمود که در این حالت، مقداری اطلاعات از دست می‌رود.

به طور خلاصه پیچیده‌ترین حالت برای محیط محیطی است که غیرقابل دسترس، غیرقطعی، پویا و پیوسته باشد. محیط‌هایی که این ویژگی‌ها را داشته باشند محیط‌های باز[3] نامیده می‌شوند.

در طراحی عامل دو معیار اهمیت زیادی دارد: معیار اول ویژگی‌هایی محیط است که در بالا به آن اشاره شد و ویژگی دوم ارتباط[4] بین عامل و محیط می‌باشد.

سامانه های تابعی[5]: در این سامانه‌ها ابتدا ورودی گرفته می‌شود، سپس توابعی بر روی آن اعمال می‌شود و در نهایت خروجی تولید می‌شود. در چنین سامانه‌هایی تابع f بر روی ورودی I عملی را انجام می‌دهد تا خروجی O بدست آید.

---

[1] Percepts
[2] Actions
[3] Open
[4] Interaction
[5] Functional System

$$(f : I \rightarrow O)$$

یک نمونه از این سامانه‌ها، کامپایلر می‌باشد.

یکی از ویژگی‌های مهم سامانه‌های تابعی خاتمه‌پذیر بودن[1] آن‌هاست. به‌این معنا که برای اجرای آن‌ها نیاز به یک سری پیش شرط[2] و پس شرط[3] می‌باشد. در این حالت ابتدا باید پیش شرط‌ها برقرار باشد تا عملیات مورد نظر اجرا گردد؛ لذا این پیش‌شرط‌ها هستند که مشخص می‌کنند پس از اجرای عمل و خاتمه یافتن آن چه خروجی‌هایی درست خواهد بود.

بسیاری از سامانه‌های رایانه‌ای به صورت تابعی نبوده و از نوع واکنشی[4] می‌باشند که دارای ویژگی‌های زیر هستند:

- سامانه‌های واکنشی به اندازه کافی از دیدگاه رابطه‌ای[5] و تابعی[6] قابل توصیف نیستند. دیدگاه رابطه‌ای، برنامه‌ها را توابعی از حالت آغازین تا حالت پایانی در نظر می‌گیرد. نقش اصلی سامانه‌های واکنشی، برقراری تعامل با محیط بوده و ناباراین بایستی به عنوان رفتارهای در حال پیشرفت[7] تعریف گردند.
- سامانه واکنشی سامانه‌ای است که قابلیت پاسخگویی سریع به تغییرات محیط را دارا است. در این حالت Reactive بودن مترادف باResponsive می‌باشد.
- سامانه‌های واکنشی سامانه‌هایی هستند که به صورت مستقیم به جهان پاسخ می‌دهند.

طراحی سامانه‌های واکنشی سخت‌تر از سامانه‌های تابعی می‌باشد. دلیل اصلی آن اینست که عامل در ارتباط پایان ناپذیر با محیط بوده و باید به طور پیوسته تصمیم‌های داخلی گرفته تا از این تصمیم‌ها نتایج کلی حاصل شود.

---

[1] Terminate
[2] Preconditions
[3] Postconditions
[4] Reactive
[5] Relational
[6] Fuctional
[7] On-going Behavior

در محیط‌های رخدادی[1]، کارایی عامل به تعداد حادثه[2] های جدا از هم است که در آن توجهی به کارایی عامل در حوادث مختلف نمی‌شود. یک نمونه از محیط‌های حادثه ای سامانه منظم کردن پستی می‌باشد.

جنبه دیگر ارتباط بین عامل و محیط، مسئله آنی‌بودن[3] می‌باشد. در این حالت زمان در ارزیابی میزان کارایی عامل نقش اساسی دارد. در این قسمت چند نمونه از تعاملات زمانمند بیان می‌گردد:

۱- تصمیم‌گیری برای انجام دادن عمل در یک محدوده زمانی مشخص انجام می‌شود.
۲- وقوع امر در حداقل مدت زمان ممکن انجام شود.
۳- تکرار تعدادی عمل به دفعات امکان پذیر است.

در صورتی که مسئله زمان مطرح نباشد، عامل می‌تواند مدت زمان زیادی را صرف نماید تا بهترین جواب مسئله بدست آید. انتخاب بهترین جواب در صورتی امکان پذیر است که عامل تمام حالت‌هایی را که منجر به انجام عمل می‌شود بررسی نموده و در نهایت بهترین عمل را برای اجراشدن انتخاب نماید. در این حالت مدت زمان انتخاب بهترین جواب به تعداد حالاتی بستگی دارد که عامل می‌تواند انجام دهد. باید توجه داشت که در محیط‌های واقعی انجام این چنین بررسی امکان پذیر نیست، بنابراین در محیط‌های واقعی مسئله زمانمند بودن باید در نظر گرفته شود.

۱-۵- قابلیت‌های عامل‌های هوشمند[4]

عامل هوشمند چیست؟ برای پاسخ به این سؤال باید قابلیت‌های که از یک عامل هوشمند می‌توان انتظار داشت را بیان نمود. این قابلیت‌ها توسط ولدریج و جنینگز ۱۹۹۵ به شرح ذیل پیشنهاد شده‌اند:

خاصیت واکنش‌پذیری: عامل هوشمند عاملی است که می‌تواند محیط پیرامون خود را درک کند و قادر است تا به موقع به محیط پاسخ دهد. هدف از این پاسخ تغییر محیط و رسیدن به اهداف خود عامل می‌باشد.

خاصیت هدفمند بودن: عامل هوشمند عاملی است که می‌تواند رفتارهای هدف‌گرا را به صورت ابتکاری به منظور دستیابی به اهداف از پیش طراحی شده خود به نمایش

[1] Episodic Environment
[2] Episodic
[3] Real Time
[4] Intelligent Agents

گذارد. به طور مثال متدهایی که در زبان جاوا[1] نوشته می‌شوند یک نمونه از این موضوع هستند. نوشتن چنین فرآیندی، نیازمند یک سری فرضیه‌ها است که به آن پیش شرط گفته می‌شود. در صورتی که این پیش شرط‌ها صادق باشد، نتیجه‌ای به وجود می‌آید که به آن پس شرط گفته می‌شود. به نتایج بدست آمده هدف[2] گفته می‌شود که این هدف همان چیزی است که نویسنده برنامه به دنبال آن می‌باشد. در صورتی که پیش شرط‌ها صادق باشد، عمل به درستی اجرا می‌شود و نتیجه آن رسیدن به پس شرط‌هایی است که خود نیز درستند. در این حالت می‌توان گفت که عامل به هدف مورد نظرش رسیده است.

طراحی برنامه‌های هدفمند برای سامانه‌های تابعی روش مناسبی می‌باشد ولی در سامانه‌های غیر تابعی این روش پذیرفتنی نیست، زیرا سبب محدود شدن فرضیه‌های مسئله می‌گردد. در این حالت فرض اساسی بر این است که محیط در حین اجرای عمل بدون تغییر باقی می‌ماند. در صورتی که محیط تغییر کند و فرض‌های اساسی در حین اجرای عمل نادرست درآید، رفتاری که از این فرآیند سر می‌زند ممکن است نامعین و نامشخص باشد.

در بسیاری از محیط‌ها، هیچ‌یک از این فرض‌ها صادق نیستند. در این صورت، این امکان وجود دارد که محیطی که قرار است توسط عامل مشاهده شود یک محیط پیچیده بوده و عامل‌های دیگر نیز در آن محیط وجود داشته باشند. در این حالت محیط به صورت غیر قطعی است. در چنین محیط‌هایی اجرای کورکورانه یک فرآیند بدون در نظر گرفتن اینکه آیا فرضیه‌های اساسی صادق است یا نه، یک استراتژی ضعیف تلقی می‌شود. در محیط‌های پویا، عامل باید به صورت واکنشی عمل نماید.

ساختن سامانه‌هایی که به طور کامل هدفمند و یا به صورت واکنشی باشند کار مشکلی نیست. اما طراحی سامانه‌هایی که ترکیبی از رفتارهای هدف‌گرا و واکنشی هستند امری دشوار است. ترکیب رفتارهای هدف‌گرا و واکنشی یکی از مشکلاتی‌ست که عامل با آن مواجه است.

خاصیت اجتماعی بودن[3] : عامل هوشمند قادر است که به منظور رسیدن به اهداف از پیش طراحی شده با دیگر عامل‌ها (و همچنین با انسان) در ارتباط باشد.

هر روزه میلیون‌ها رایانه در سراسر دنیا، اطلاعاتی را با انسان و دیگر رایانه‌ها مبادله می‌کنند، اما مبادله این اطلاعات به معنای اجتماعی بودن آن‌ها نیست. در دنیای انسان‌ها

[1] JAVA
2 Goal
[3] Social Ability

تنها تعداد معدودی از کارها بدون همکاری با دیگر انسان‌ها قابل اجراست. در چنین دنیایی انسان‌ها برای رسیدن به اهداف باید با یکدیگر گفتگو داشته باشند و همچنین برای انجام کار باید با همدیگر همکاری کنند. در چنین ساختاری انسان‌ها باید از اهداف دیگر افراد و دلایل آن آگاهی داشته باشند و راه رسیدن به اهداف خود را با توجه به رفتارها و اهداف دیگران مشخص نمایند؛ لذا قابلیت اجتماعی بودن بسیار پیچیده‌تر از تبادل ساده اطلاعات با یکدیگر است.

۱-۶- عامل و اشیاء[1]

اشیا مجموعه‌ای از حالات[2] و متدهایی[3] برای نیل به این حالات است. این اشیا از طریق رد و بدل کردن پیام باهم در ارتباط هستند. علیرغم اینکه بین عامل و شئ شباهت‌های زیادی وجود دارد، تفاوت‌های قابل ملاحظه‌ای بین این دو سلسله وجود دارد. اولین اختلاف مربوط به درجه خود مختاری عامل و شئ می‌باشد. اشیا می‌توانند بر حالت‌های درونی خود کنترل داشته باشند. می‌توان گفت که این حالات قادرند به صورت خصوصی[4] و عمومی[5] باشند. وقتی که این حالات به صورت خصوصی باشند تنها توسط خود شئ قابل کنترل است و در حالت عمومی این امکان وجود دارد که حالات توسط اشیای دیگر قابل دسترسی باشد؛ این عمل با ایجاد قابلیت دسترسی توسط متدها صورت می‌پذیرد. بنابراین در صورتی که در یک شئ حالتی عمومی وجود داشته باشد، کنترل کاملی روی این حالت وجود نخواهد داشت زیرا اینجا نیز اشیا توسط متدهای دیگر تحت کنترل قرار گرفته‌اند.

البته باید توجه داشت که وقتی یک سامانه به کمک اشیاء ایجاد می‌گردد، کل سامانه به دنبال انجام یک هدف مشترک است. در حالی که در سیستم‌های چندعامله این حالت هیچگاه فرض نمی‌شود. در سامانه‌های چندعامله، عامل i عمل α (متد) را به منظور درخواست عامل j در رخداد عمل α انجام نمی‌دهد. در صورتی که عامل j از عامل i درخواست کند که عمل α انجام پذیرد عامل i ممکن است این عمل را انجام دهد و یا انجام ندهد. بنابراین می‌توان نتیجه گرفت که تصمیم‌گیری برای انجام یک عمل در سامانه‌های شئ گرا و چندعامله با هم متفاوت می‌باشد. در سامانه‌های چندعامله تصمیم گیری بر اساس ارائه درخواست صورت می‌گیرد، درحالی‌که در سامانه‌های شئ گرا تصمیم گیری بر مبنای فراخوانی متد صورت می‌گیرد. این تفاوت را می‌توان به کمک جمله زیر بیان نمود:

---

[1] Object
[2] State
[3] Method
[4] Private
[5] Public

«اشیاء عمل را به صورت آزادانه انجام می‌دهند؛ درحالی‌که عامل، عمل را انجام می‌دهد زیرا می‌خواهد آن را انجام دهد.»

تفاوت دوم که بین عامل و شئ وجود دارد مربوط به رفتارهای خودمختار وانعطاف-پذیر آن است. همان طور که توضیح داده شد این رفتار شامل واکنش پذیری، هدفمند بودن و اجتماعی بودن می‌باشد. یک شئ استاندارد به هیچ وجه نمی‌تواند سامانه‌ای مرکب از این سه رفتار را شکل دهد در حالی که عامل می‌تواند سامانه‌ای بر مبنای ترکیب این سه رفتار فراهم آورد.

سومین تفاوت مهمی که بین عامل و شئ وجود دارد این است که سامانه‌های چندعامله به طور ذاتی چند رشته ای[1] می‌باشند؛ یعنی اینکه هر عامل حداقل دارای یک رشته کنترل سیستم است، در حالی که در شئ گرایی تنها یک رشته کنترل در سیستم وجود دارد.

۱-۷- عامل‌ها و سامانه‌های خبره[2]

سامانه‌های خبره مهم‌ترین تکنولوژی هوش مصنوعی در دهه ۸۰ میلادی می‌باشند. سامانه خبره سامانه‌ای است که توانایی حل مسئله و یا توانایی مشورت دادن در زمینه‌های مورد نظر بر مبنای دانش را دارا باشد. یک نمونه ساده از سامانه‌های خبره، سامانه MYCIN می‌باشد که به پزشک در زمینه بیماری‌های خونی انسان کمک می‌نماید. این سامانه در تعامل با انسان عمل می‌کند، در نتیجه مجموعه‌ای از حقایق به عنوان ورودی برای سامانه ایجاد می‌گردد، سپس سامانه بر اساس این حقایق به نتیجه‌گیری می‌پردازد. این سامانه بیشتر شبیه به یک مشاور عمل می‌کند و به طور مستقیم بر روی افراد عملی را انجام نمی‌دهد یا به عبارت ساده‌تر این سامانه به طور مستقیم با محیط در ارتباط نیست. بنابراین تفاوت اصلی که بین عامل و سامانه‌های خبره وجود دارد این است که سامانه‌های خبره به طور ذاتی مجزا از جسم[3] هستند. این بدان معناست که سامانه‌های خبره به صورت مستقیم با محیط در ارتباط نیستند، در نتیجه اطلاعات را از طریق سنسور دریافت نمی‌کنند و دریافت اطلاعات به کمک یک واسطه امکان‌پذیر می‌باشد. به همین ترتیب این سامانه به طور مستقیم روی محیط عمل نمی‌کند بلکه این عمل به کمک وسایل واسطه صورت می‌گیرد. به علاوه سامانه‌های خبره عموماً توانایی همکاری با دیگر عامل‌ها را ندارند. به طور خلاصه می‌توان تفاوت بین سامانه‌های خبره و عامل را به صورت زیر بیان نمود:

---

[1] Multi-threaded
[2] Expert Systems
[3] Disembodied

۱- سامانه‌های خبره به طور ذاتی مجزا از جسم هستند. این سامانه‌ها با هیچ محیطی در ارتباط نیستند و بر روی آن‌ها به صورت مستقیم عمل نمی‌کنند (عمل بر روی محیط از طریق یک واسط صورت می‌گیرد).
۲- سامانه‌های خبره عموماً قابلیت انجام رفتارهای واکنشی و هدفمند را ندارند.
۳- سامانه‌های خبره قابلیت برقراری روابط اجتماعی در زمینه گفتگو کردن، هماهنگی و همکاری را ندارند.

در کنار این تفاوت‌ها، بسیاری از سامانه های خبره که کارهای کنترل آنی را انجام می‌دهند تا حدود بسیار زیادی شبیه به عامل‌ها عمل می‌کنند.

۱-۸- عامل به عنوان سامانه‌ی ارادی[1]

عامل را می توان به عنوان یک سامانه‌ی ارادی در نظر گرفت. با این تعریف می‌توان اصطلاحاتی نظیر حالات روحی[2]، باور[3]، میل داشتن[4]، آرزو[5]، امید[6] و ... را به عامل ربط داد. منطق این روش به صورت زیر بیان می‌شود. زمانی که یک فعالیت مرتبط با انسان توضیح داده می‌شود، اغلب بهتر است که جملاتی به صورت زیر بیان گردد:

«علی چترش را برداشت زیرا او باور داشت که باران می‌بارد.»
«علی به سختی کار می‌کند زیرا او می‌خواهد که این کتاب را تمام کند.»

این‌گونه جملات بر اساس استفاده از روانشناسی قومی[7] ایجاد می‌گردند که به کمک آن رفتار انسان از قبیل اعتقاد داشتن، خواستن و ... پیش‌بینی و توضیح داده می‌شود. این روانشناسی قومی به خوبی مسلم و برقرار می‌باشد. بسیاری از مردم که این جملات را می‌خوانند، به طور کاملاً واضح معنی آن را متوجه می‌شوند.

وضعیت و حالاتی را که در تعاریف روانشناسی قومی وجود دارد اندیشه ارادی[8] می‌نامند. عنوان سامانه‌های ارادی برای اولین بار توسط فیلسوف دانیل دنت مطرح گردید و به کمک آن رفتار انسان از قبیل باور، میل و تیزهوشی عاقلانه قابل پیش‌بینی شد. دنت سطوح مختلف سامانه های ارادی را به صورت زیر بیان می‌کند:

سامانه‌های ارادی مرتبه اول: این سامانه‌ها دارای باور، میل و ... می‌باشند اما هیچ‌گونه اعتقاد و میلی نسبت به باور، میل و ... ندارند.

---

[1] Intentional Systems
[2] Mental states
[3] Belief
[4] Desire
[5] Wish
[6] Hope
[7] Folk Psychology
[8] Intentional Notions

سامانه‌های ارادی مرتبه دوم: این سامانه‌ها پیچیده‌تر می‌باشند و دارای باورها و امیالی (و بدون شک دیگر حالات ارادی) نسبت به باورها و امیال دیگر (حالات ارادی) هستند.

می‌توان اصطلاحاتی از قبیل اعتقاد، میل و ... را برای برنامه‌های رایانه‌ای نیز تعریف نمود. در صورتی که باور، اختیار، اراده، هوشیاری، شایستگی یا خواستن به ماشین نسبت داده شود، (این امر قانونی[1] است) نتیجه آن همان برداشتی می‌شود که از انسان انتظار می رود. این قضیه زمانی سودمند[2] است که این اسناد به ما کمک نماید ساختار ماشین، رفتار پیشین و آینده و چگونگی توسعه آن را متوجه شویم. کلمات باور، دانش و خواستن برای ماشین به صورت ساده‌تری نسبت به انسان ایجاد شده‌اند. نسبت دادن تعاریف ذهنی به ماشین‌هایی که ساختار مشخصی دارند بسیار سر راست[3] است. اما به کار گیری این تعاریف برای موجودیت‌هایی که ساختارشان به طور ناقص شناخته شده بسیار مناسب می‌باشد.

چه اشیایی می‌توانند با حالت ارادی توصیف شوند؟ در پاسخ باید گفت هر وسیله خودکار می‌تواند دارای چنین حالتی باشد. برای مثال، کاملاً منطقی است که با کلید چراغ همانند یک عامل برخورد شود که این عامل قابلیت دارد جریان را به طور دلخواه انتقال دهد. زمانی که کلید برق باور دارد که ما می‌خواهیم جریان انتقال یابد، کلید به طور ثابت جریان را انتقال می‌دهد و برعکس. ضربه زدن به کلید یک راه مخابره کردن میل ما در جهت قطع جریان است.

هرچه انسان بیشتر راجع به یک سامانه بداند، کمتر به روش شکل گیری آن سامانه و رفتارهایش احتیاج دارد. به طور مثال کودکان ابتدا از اطلاعات مربوط به جان گرفتن و ایجاد یک شئ استفاده می‌کنند و این عمل را تا موقعی که مفاهیم تکنیکی آن را بدست آورند ادامه می‌دهند. باید توجه داشت که نحوه تکامل دانش بشر نیز به همین صورت است.

در اینجا یک سؤال پیش می‌آید و آن این است که در صورتی که راه ساده‌تری برای تعریف یک سامانه وجود دارد چرا از حالت ارادی برای توصیف استفاده می‌شود؟ یک راه مشخص کردن و تعریف نمودن سامانه های پیچیده استفاده از حالت فیزیکی می‌باشد. ایده استفاده از حالت فیزیکی بر اساس شکل‌بندی اصلی سامانه بوده و در نتیجه با استفاده از قوانین فیزیک چگونگی رفتار سامانه قابل پیش‌بینی می‌باشد.

---

[1] Legitimate
[2] Useful
[3] Most Straightforward

به طور مثال زمانی که پیش‌بینی می‌شود یک سنگ در صورتی که از بلندی رها شود سقوط خواهد کرد، در این صورت برای توصیف این سامانه از قوانین فیزیک استفاده شده است. در اینجا نیازی به استفاده از باور و میل برای این عمل سنگ نیست. برای پیش‌بینی عمل سنگ تنها کافی است که از جرم و قانون جاذبه استفاده نمود.

روش دیگر توصیف استفاده از حالت طراحی[1] می‌باشد. در این حالت از دانشی استفاده می‌شود که به کمک آن مشخص می‌گردد سامانه قصد دارد چه اهدافی را انجام دهد. در این صورت می‌توان چگونگی رفتار سامانه را پیش‌بینی کرد.

به طور مثال وقتی کسی به ما ساعت شماطه‌دار می‌دهد، برای فهمیدن رفتار آن نیازی به استفاده از قوانین فیزیک نیست. می‌توان از این حقیقت استفاده نمود که ساعت شماطه‌دار برای این طراحی شده که مردم را در ساعتی که می‌خواهند بیدار کند. برای فهمیدن این مطلب هیچ گونه نیازی به داشتن دانش از مکانیک ساعت نیست زیرا همه ما می‌دانیم که ساعت شماطه‌دار این‌چنین کار می‌کند.

برای سامانه‌های پیچیده علیرغم این‌که گاهی تصویری کامل از معماری و نحوه کار سامانه در اختیار است با این وجود حالت‌های طراحی و فیزیکی نیز ممکن است نتوانند رفتار سامانه را توصیف کنند. نمونه آن رایانه است؛ اگرچه توصیف تخصصی و کامل از رایانه در اختیار داریم، ولی عملاً توصیف بعضی از کارهایی که در رایانه اتفاق می‌افتد مشکل است. به طور مثال چرا در صورت کلیک کردن روی یک آیکن فهرست منو باز می‌شود؟ در چنین حالتی اگر توصیف حالات ارادی سازگارتر باشد، بهتر است از این توصیف استفاده نمود.

توجه شود که در علوم رایانه از حالت ارادی به عنوان ابزار تجرد یا انتزاع[2] یاد می‌شود که به کمک آن می‌توان بدون توجه به اینکه سامانه چگونه کار می‌کند رفتارهای آن را پیش بینی نموده و توضیح داد. هم اکنون بسیاری از علوم رایانه‌ای به دنبال یافتن مکانیزم تجردی هستند که به کمک آن بتوانند به روش بسیار ساده‌ای پیچیدگی‌ها را مدیریت نمایند.

۱-۹- معماری انتزاعی برای عامل هوشمند

به راحتی می‌توان دید انتزاعی را برای عامل‌هایی که تاکنون معرفی شده‌اند بیان نمود. در ابتدا محیط به صورت مجموعه E در نظر گرفته می‌شود که این مجموعه شامل حالت‌های گسسته و لحظه‌ای فرض شده است و به صورت زیر نشان داده می‌شود:

[1] Design Stance
[2] Abstraction Tool

$$E = \{e, e', \ldots\}$$

باید توجه داشت که مدل فرض شده، یک مدل گسسته است. فرض گسسته بودن محیط برای اهداف ما خیلی مهم نیست زیرا می‌توان هر محیط پیوسته را با مقدار معینی دقت به یک مدل گسسته تبدیل نمود. عامل‌ها به گونه‌ای فرض می‌شوند که دارای مجموعه‌ای از عمل[1]های ممکن باشند و قابلیت تغییر حالت‌های محیط را داشته باشند. مجموعه‌ای از عمل‌ها را می‌توان به صورت زیر نمایش داد:

$$Ac = \{\alpha, \alpha', \ldots\}$$

مجموعه‌ای که در اینجا بیان گردید مجموعه‌ای است که به صورت کران‌دار در نظر گرفته می‌شود.

مدل پایه عامل که با محیط در تعامل است به صورت زیر تعریف می‌شود:

محیط، کار خود را با یک حالت شروع می‌کند و عامل با انتخاب یک عمل بر روی محیط کاری را انجام می‌دهد. نتیجه عمل این است که محیط این عمل را با یکی از حالاتی[2] که دارد پاسخ می‌دهد. باید توجه داشت که عامل از قبل نمی‌داند محیط چه پاسخی خواهد داد. بر اساس حالت دوم محیط، عامل دوباره یک عمل را برای اجرا انتخاب می‌کند و محیط دوباره با یکی از حالاتی که دارد پاسخ می‌دهد و این عمل دائماً ادامه پیدا می‌کند. در فرآیند r که در زیر نشان داده شده است، عامل در یک محیط قرار گرفته است که در این فرآیند مجموعه‌ای از حالات و اعمال وجود دارد.

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$

و در آن:

R: مجموعه محدود مراحل ممکن می‌باشد (هم برای E و هم برای Ac).

R^{Ac}: زیرمجموعه‌ای از مراحل R است که به یک عمل ختم می‌شود.

R^{E}: زیرمجموعه‌ای از مراحل R است که به یک حالت محیط ختم می‌شود.

در اینجا r، r' و ... اعضای مجموعه R می‌باشد.

به منظور نشان دادن نتیجه عمل عامل بر روی محیط، یک تابع تبدیل حالت[3] معرفی می‌گردد که این تابع به صورت زیر تعریف می شود:

$$\tau : \Re^{Ac} \rightarrow \wp(E)$$

[1] Action
[2] State
[3] State Transformer Function

تابع تبدیل، مرحله ختم شده به عامل را به مجموعه حالات (آنهایی که می‌تواند نتیجه اجرای عمل عامل باشد) ممکن در محیط تبدیل می‌کند.

درباره این تعریف باید به دو نکته مهم توجه نمود:

۱- در اینجا محیط را وابسته به پیشینه[1] می‌دانیم. به عبارت دیگر حالت بعدی محیط تنها به وسیله عمل عامل و حالت فعلی محیط به وجود نمی‌آید. عملی که زودتر توسط عامل ایجاد گردد، نقشی را در تعیین حالت فعلی محیط بازی می‌کند.

۲- این تعریف برای حالت غیر قطعی[2] در محیط هم صادق است. بنابراین حالت عدم قطعیت درباره نتایج اجرای عمل در تعدادی از حالات وجود دارد.

در صورتی که داشته باشیم:

$$\tau(r) = \phi$$

جایی که r مرحله ختم شده به یک عامل باشد و هیچ حالت جانشین ممکن برای r وجود ندارد. در این صورت می‌گوییم عمل اجرا پایان یافته است. البته ما چنین فرض می‌کنیم که تمام اجراها پایان پذیر است.

می‌توان محیط Env را به صورت سه تایی نمایش داد:

$$Env = < E, e_0, \tau >$$

در اینجا E مجموعه حالت‌های محیط می‌باشد و همچنین ($e_0 \in E$) حالت آغازین و T تابع تبدیل حالت است. حال نیاز به معرفی مدلی از عامل‌ها است که در سامانه عمل می‌کنند. عامل به عنوان یک تابع در نظر گرفته می‌شود که مرحله اجرا (مرحله‌ای که به یک حالت محیط ختم می‌شود) را به عمل تبدیل می‌کند.

$$Ag : \Re^E \rightarrow Ac$$

بنابراین عامل تصمیم می‌گیرد چه عملی را بر اساس تاریخچه سامانه اجرا کند که بهترین عمل باشد. در صورتی که محیط غیر قطعی باشد، عامل‌ها به صورت قطعی فرض شده و بنابراین AG مجموعه همه عامل‌ها در نظر گرفته می‌شود.

سامانه را یک مجموعه دوتایی می‌نامیم که شامل عامل و محیط می‌باشد. سامانه را می‌توان به صورت زیر تعریف نمود:

[1] History Dependent
[2] Non-determinism

R (Ag, Env)

در این رابطه R مجموعه‌ای از اجراها می‌باشد که این اجراها مربوط به عامل Ag در داخل محیط Env می‌باشد. برای سادگی کار، فرض می‌شود که R مجموعه‌ای از اجراهای پایان‌پذیر است؛ در این صورت در مجموعه r ای وجود دارد که سبب می‌شود: $\tau(r) = \phi$

مجموعه:

$$(e_0, \alpha_0, e_1, \alpha_1, e_2, \ldots)$$

مجموعه‌ای از اجراها می‌باشند که عامل Ag آن‌ها را در محیط $Env = < E, e_0, \tau >$ عمل می‌کند در صورتی که:

۱- e_0 حالت اولیه محیط Env باشد.

۲- α_0 عملی است که عامل بر روی محیط وقتی در حالت e_0 قرار دارد انجام می‌دهد ($\alpha_0 = Ag(e_0)$)

۳- برای هر $u > 0$ داریم:

$$e_u \in \tau((e_0, \alpha_0 \ldots, \alpha_{u-1}))$$

که:

$$\alpha_u = Ag((e_0, \alpha_0, \ldots, e_u))$$

دو عامل Ag_1 و Ag_2 نسبت به محیط Env دارای رفتاری یکسان هستند اگر و تنها اگر:

$$R(Ag_1, Env) = R(Ag_2, Env)$$

عامل‌های کاملاً واکنشی[1]:

نوع معینی از عامل‌ها وجود دارند که بدون توجه به گذشته خودشان کاری را انجام می‌دهند. این‌گونه عامل‌ها برای گرفتن هرگونه تصمیمی تنها به زمان حال توجه می‌کنند و هیچ‌گونه توجهی به کارهایی که در گذشته انجام داده‌اند ندارند. چنین عامل‌هایی، عامل‌های کاملاً واکنشی نامیده می‌شوند، از این‌رو اینگونه عامل‌ها به طور مستقیم به محیط پاسخ می‌دهند. نحوه پاسخگویی اینگونه عامل‌ها همانند پاسخگویی حیوان و گیاه به محیط اطراف می‌باشد.

رفتار عامل‌های کاملاً واکنشی را می‌توان به صورت تابع زیر نشان داد:

[1] Purely Reactive Agents

$$Ag : E \rightarrow Ac$$

می‌توان گفت برای هر عامل کاملاً واکنشی، یک عامل استاندارد بیان می‌شود.

عامل ترموستات، یک نمونه از عامل‌های کاملاً واکنشی می‌باشد. می‌توان فرض کرد محیطی که عامل در آن قرار دارد یکی از دو حالت «خیلی سرد» یا «معتدل» را داشته باشد. در این صورت عامل ترموستات را می‌توان به صورت زیر تعریف نمود:

$$Ag(e) = \begin{cases} heater\ off & if\ e = temperature\ OK \\ heater\ on & otherwise \end{cases}$$

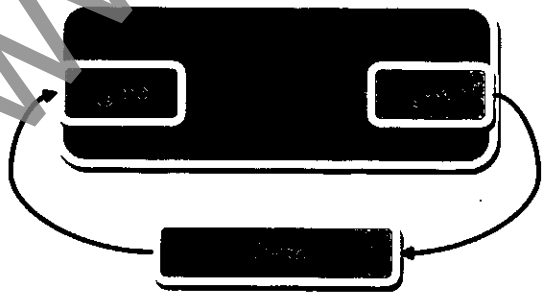
ادراک[1]:

به منظور روشن‌تر شدن مدل انتزاعی عامل، می‌توان از ایده‌ای که در مهندسی نرم‌افزار وجود دارد استفاده نمود. بر این اساس مدل انتزاعی عامل را می‌توان به زیر سامانه‌هایی تقسیم نمود. چنین زیر سامانه‌هایی سبب می‌شوند که یک عامل به وجود آید. در اصل معماری عامل، ترسیمی از محتوای داخلی عامل است که شامل: ساختار داده؛ اپراتورهایی که ممکن است بر روی این ساختار عمل کنند؛ و روندهای کنترلی بین این ساختار داده‌ها می‌باشند.

تابع تصمیم گیری عامل را می‌توان به دو زیر سامانه تقسیم نمود که در شکل ۱-۲ نشان داده شده است:

۱- زیر سامانه ادراک[2]

۲- زیر سامانه عمل[3]



شکل ۱-۲- زیر سامانه‌های ادراک و عمل یک عامل

[1] Perception
[2] Perception Subsystems
[3] Action Subsystems

این ایده بدین صورت است که تابع See از طریق مشاهده کردن عمل جمع‌آوری اطلاعات از محیط را انجام می‌دهد، درحالی‌که تابع Action فرآیند تصمیم‌گیری عامل را نشان می‌دهد. چنانچه عامل در جهان فیزیکی واقع شود، تابع See می‌تواند به صورت سخت‌افزاری اجرا گردد؛ برای مثال این تابع می‌تواند دوربین‌های ویدیویی، سنسورهای مادون قرمز و یا روبات متحرک باشد. برای عامل‌هایی که به صورت نرم‌افزاری ایجاد شده‌اند، این سنسورها می‌توانند دستورهای سامانه باشند که به جمع‌آوری اطلاعات از محیط‌های نرم‌افزاری می‌پردازند. خروجی تابع See، دریافت[1] نامیده می‌شود. این تابع را می‌توان به صورت زیر نمایش داد:

$$See: E \rightarrow Per$$

ورودی این تابع، محیط و خروجی آن همان دریافت می‌باشد که از آن به عنوان ورودی عامل استفاده می‌شود. تابع عمل را نیز می‌توان به صورت زیر تعریف نمود:

$$Action: Per^* \rightarrow Ac$$

که در آن یک رشته از دریافت‌ها را به عمل تبدیل می‌کند. در اینجا عامل Ag به صورت یک زوج مرتب از تابع دیدن و عمل در نظر گرفته می‌شود و به صورت زیر نمایش داده می‌شود.

$$Ag = < see, action >$$

این تعاریف ساده سبب می‌شود که خصوصیات جالبی از عامل و دریافت‌های آن بدست آید.

فرض شود دو حالت e_1 و e_2 از محیط وجود دارد: $e_1 \in E$ و $e_2 \in E$ به طوری که $e_1 \neq e_2$، اما see (e_1) = see (e_2). در این حالت عامل از دو حالت متفاوت در محیط، اطلاعات یکسانی را دریافت می‌کند. در این صورت e_1 و e_2 غیر قابل تشخیص هستند. برای اینکه این مفهوم قابل درک شود دوباره مثال ترموستات بیان می‌گردد. جمله (الف) و (ب) دو حقیقت درباره محیط اطراف می‌باشد:

الف) دمای اتاق مناسب است.

ب) علی نخست وزیر است.

در این صورت مجموعه حالاتی که ممکن است در محیط اتفاق بیفتد شامل چهار عضو می‌باشد:

[1] See

$$E=\left\{\underbrace{\{\neg x,\neg y\}}_{e_1},\underbrace{\{\neg x,y\}}_{e_2},\underbrace{\{x,\neg y\}}_{e_3},\underbrace{\{x,y\}}_{e_4}\right\}$$

در حالت e_1 دمای اتاق مناسب نمی‌باشد و همچنین علی نخست وزیر نیست. در حالت e_2 دمای اتاق مناسب نبوده ولی علی نخست وزیر است. ترموستات تنها به دمای اتاق حساس می‌باشد و دمای اتاق هیچ ارتباطی به این ندارد که علی نخست وزیر است یا نه. بنابراین، حالتی که علی نخست وزیر است یا خیر اساساً برای ترموستات غیر قابل تشخیص می‌باشد. تابع see مربوط به ترموستات دو دریافت p_1 و p_2 خواهد داشت که نشان می‌دهد دمای اتاق یا خیلی سرد است یا مناسب می‌باشد. تابع see مربوط به ترموستات به صورت زیر تعریف می‌شود:

$$see(e)=\begin{cases}p_1 & if\ e=e_1\ or\ e=e_2\\ p_2 & if\ e=e_3\ or\ e=e_4\end{cases}$$

در این مثال برای دو حالت e_1 و e_2 داریم: see (e_1) = see (e_2).

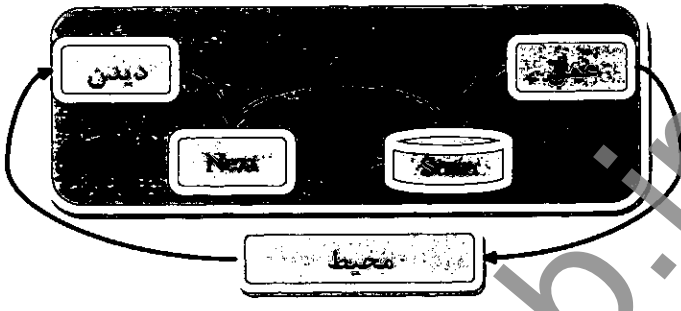
در این مثال برای حالت $e\in E$ و $e'\in E$، اگر رابطه see (e) = see (e') برقرار باشد در این صورت داریم ($e \sim e'$). در اینجا علامت (~) به عنوان علامت هم ارزی در نظر گرفته می‌شود. در این حالت E دارای مجموعه هایی غیر قابل تشخیص از حالات است. در صورتی که تعداد دریافت‌های مجزا برابر با تعداد حالت‌های مختلف محیط باشد ($|\sim|=|E|$)، عامل می‌تواند به طور کامل محیط را درک کند. در این صورت می‌توان گفت که عامل همه‌چیزدان[1] است. در صورتی که تعداد دریافت‌های مجزا برابر با ۱ باشد ($|\sim|=1$)، قابلیت ادراکی عامل ناموجود[2] می‌باشد، در این صورت می‌توان گفت که عامل هیچ حالت مختلفی از محیط را نمی‌تواند تشخیص دهد. در اینجا تمام حالت‌های محیط برای عامل یکسان می‌باشد.

عامل‌ها به همراه حالت[3]:

در قسمت قبل، تابع تصمیم گیرنده عامل به صورت یک رشته از حالات محیط یا دریافت‌هایی برای انجام عمل الگو واقع شد. در این قسمت عامل‌ها به گونه ای تصمیم می‌گیرند که تحت تأثیر کارهای قبلی خود قرار گیرند. در این وضعیت عامل‌ها دارای مخزنی از حالت‌ها هستند. این عامل‌ها دارای ساختار داده داخلی می‌باشند که سبب

[1] Omniscience
[2] Non-existent
[3] Agents with State

می‌شود اطلاعاتی راجع به حالت‌های محیط و سابقه[1] را در خود نگهداری نماید. شکل ۱-۳ نمونه‌ای از این حالات را نشان می‌دهد.



شکل ۱-۳- عامل نگه دارنده حالات

در اینجا I مجموعه‌ای از حالات درونی عامل در نظر گرفته می‌شود. فرآیند تصمیم-گیری عامل در این حالت بر اساس این اطلاعات می‌باشد. تابع ادراک See برای این‌گونه عامل‌ها همانند حالت قبلی است و هیچ تغییری نمی‌کند و در این تابع حالات محیط به دریافت‌هایی تبدیل می‌شود:

$$See: E \rightarrow Per$$

تابع عمل نیز به صورت زیر تعریف می‌گردد:

$$Action: I \rightarrow Ac$$

این تابع نگاشتی از حالات داخلی به عمل است. در این گونه عامل‌ها تابع جدیدی به نام تابع Next اضافه می‌گردد که نگاشتی از حالات درونی و دریافت‌ها به خود حالات درونی می‌باشد. این تابع به صورت زیر تعریف می‌گردد:

$$Next: I \times Per \rightarrow I$$

رفتار این‌گونه عامل‌ها را می‌توان به طور خلاصه به صورت زیر بیان نمود. عامل کار خود را با حالت‌های اولیه داخلی (i_0) آغاز می‌کند، سپس حالت e محیط را مشاهده می‌کند و دریافتی از محیط بدست خواهد آورد. این کار با عمل see انجام می‌پذیرد ((see (e)). سپس حالت درونی عامل به کمک تابع Next به روز می‌گردد ((Next (i_0, see (e))). پس از

[1] History

آن عمل عامل بر روی محیط اجرا می‌گردد (Action (next (i_0 , see(e))). پس از آنکه عمل توسط عامل اجرا گردید، دوباره عامل تمام مراحل را تکرار می‌کند.

باید توجه داشت که این‌گونه عامل‌ها هیچ‌گونه توانایی بالاتری نسبت به عامل‌های استاندارد ندارند. در حقیقت این‌گونه عامل‌ها می‌توانند به عامل‌های استاندارد تبدیل شوند به طوری که از نظر رفتاری با هم یکسان باشند.

۱-۱۰- نحوه برقراری ارتباط با عامل برای انجام کار

معمولاً عامل برای این ساخته می‌شود که برای ما کار مشخصی را انجام دهد. برای اینکه عامل بتواند کاری انجام دهد، باید با آن ارتباط برقرار کرد تا بتوان کاری را که باید انجام شود برایش مشخص نمود. به این ترتیب باید به طریقی این کار را برای عامل مشخص نمود. سؤال بدیهی که در اینجا به وجود می‌آید این است که چگونه می‌توان این کار را انجام داد. چگونه می‌توان به عامل گفت چه کاری را انجام دهد. یک راه ساده برای بیان این کارها، برنامه‌نویسی برای عامل است تا بتواند کارهای خواسته شده را انجام دهد. مزیتی که در این روش وجود دارد این است که هیچ عدم قطعیتی برای آنچه که عامل می‌خواهد انجام دهد وجود ندارد. در این حالت عامل دقیقاً همان کاری را خواهد کرد که به او گفته شود. عیب این روش این است که در این حالت باید برنامه‌نویس فکر کند چه اتفاقی برای عامل پیش خواهد آمد و سپس بر اساس آن برنامه‌نویسی را انجام دهد. در این حالت اگر یک حالت پیش بینی نشده برای عامل اتفاق بیافتد، عامل قادر به پاسخگویی نخواهد بود. «می‌خواهیم به عامل بگوییم چه کاری را انجام دهد بدون آنکه به او بگوییم چگونه این کار را انجام دهد». یک راه برای انجام این کار، تعریف کار به طور غیر مستقیم از طریق سنجش کارایی[1] می‌باشد. روش‌های مختلفی برای تعریف سنجش کارایی وجود دارد. یکی از این روشها استفاده از سودمندی[2] می‌باشد.

توابع سودمندی[3]:

در اینجا برای هر حالت یک عدد در نظر گرفته می‌شود. هر چه عدد بالاتر باشد، آن حالت مناسب‌تر است. عمل عامل به صورتی‌ست که حالتی را بدست آورد که بالاترین سودمندی را داشته باشد. باید توجه داشت که در اینجا به عامل گفته نمی‌شود که چگونه عمل را انجام دهد. تابع سودمندی به صورت زیر تعریف می‌شود:

[1] Performance Measure
[2] Utility
[3] Utility Functions

$$u: E \rightarrow R$$

در این تابع به هر حالت محیط یک عدد حقیقی نسبت داده می‌شود. طریقه استفاده از تابع سودمندی برای یک حالت مشخص از محیط می‌تواند به روش‌های مختلفی صورت گیرد. در حالت بدبینانه تابع سودمندی که برای عامل تعریف می‌گردد بر اساس سودی است که از مواجه شدن با بدترین حالت بدست می‌آید. حالت دیگر تعریف تابع سودمندی استفاده از میانگین سودی است که از مواجه شدن با تمام حالات بدست می‌آید. باید توجه داشت هیچ روش درست و غلطی برای تعریف تابع سودمندی وجود ندارد. نحوه اندازه‌گیری تابع سودمندی بسته به کاری دارد که عامل می‌خواهد انجام دهد.

مشکل این روش این است که در این آن برای تعریف تابع سودمندی بایستی به حالت‌های داخلی توجه شود، در نتیجه مشخص کردن چشم اندازی بلندمدت با استفاده از حالت‌های انفرادی مشکل است. برای رفع این مشکل می‌توان به جای اینکه تابع سودمندی را به حالت‌های انفرادی اختصاص داد، آن را به خود اجرا نسبت دهیم.

$$u: \mathcal{H} \rightarrow R$$

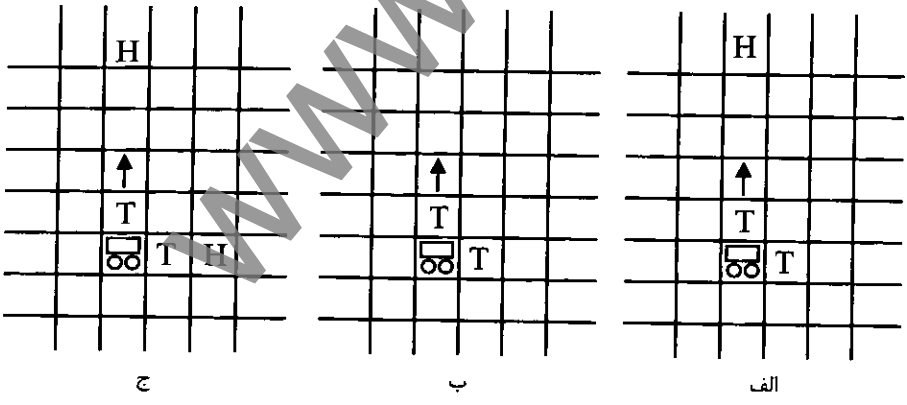
در صورتی که عامل بخواهد به طور مستقل و به مدت زیادی بر روی محیط عمل کند، استفاده از این‌گونه تابع سودمندی مفید می‌باشد. یک نمونه از کارهایی که در آن از این‌گونه تابع سودمندی استفاده می‌شود، مثال «دنیای کاشی‌ها[1]» می‌باشد. از این روش در ابتدا برای ارزیابی معماری عامل استفاده می‌شد. در این مثال محیط کار به صورت یک فضای شبکه رستری می‌باشد که در داخل آن مجموعه‌ای از عامل‌ها، موانع، چاله (H) و موزاییک‌هایی (T) وجود دارد. عامل می‌تواند در چهار جهت حرکت کند (بالا، پایین، چپ و راست) و در کنار موزائیک قرار بگیرد یا می‌تواند در داخل آن جای گیرد. موانع به صورت مجموعه خانه‌هایی است که عامل نمی‌تواند در آنها قرار گیرد؛ در نتیجه عامل نمی‌تواند از موانع عبور کند. چاله‌ها باید به کمک عامل پر گردند. هدف عامل از قرار گرفتن در این محیط پر کردن خانه‌های چاله می‌باشد. برای انجام این کار عامل باید به سراغ خانه‌هایی برود که دارای چاله (H) باشند. این محیط یک نمونه از محیط‌های پویاست به این دلیل که این محیط بدون دخالت عامل تغییر می‌کند. محیط کار خود را با تعداد تصادفی از موزاییک و چاله آغاز می‌کند و در هر مرحله تعداد و محل چاله‌ها تغییر می‌کند. کارایی عامل بر اساس شمارش تعداد چاله‌هایی است که عامل در مدت زمان مشخص پر می‌کند. کارایی عامل از رابطه زیر بدست می‌آید:

[1] Tileworld

$$u(r) \triangleq \frac{\text{تعداد چاله‌های پر شده در اجرا}}{\text{تعداد کل چاله‌های در اجرا}}$$

برد تابع $u(r)$ بین عدد صفر و یک می‌باشد. در صورتی که مقدار این تابع صفر باشد، در مدت اجرا، عامل نتوانسته است چاله‌ای را پر کند و در صورتی که مقدار بدست آمده از این تابع برابر با یک باشد، آنگاه عامل تمام چاله‌های ظاهر شده را پر کرده است.

علیرغم اینکه این محیط یک محیط بسیار ساده می‌باشد، سبب می‌شود که بسیاری از قابلیت‌های عامل را مورد ارزیابی قرار دهد. مهم‌ترین قابلیت‌هایی که در اینجا مورد بررسی قرار می‌گیرد، قابلیت واکنشی نسبت به تغییرات محیط و همچنین استفاده از فرصت‌هاست. برای مثال فرض کنید که عامل به اندازه یک سرامیک به سمت چاله می‌رود (شکل ۱-۴-الف). در مرحله بعد چاله ناپدید می‌شود (شکل ۱-۴-ب). در این مرحله پس از ناپدید شدن چاله، دنبال کردن مسیر قبلی کار بیهوده‌ای است. بنابراین، اگر عامل می‌خواهد همواره بهترین عمل را انجام دهد باید این تغییرات را شناسایی کند. برای اینکه بتوان مفهوم استفاده از فرصت‌ها را بیان نمود، فرض کنید در همان حال چاله‌ای در سمت راست عامل ظاهر شود (شکل ۱-۴-ج)، در صورتی که عامل فرصت طلب باشد مسیر خود را تغییر می‌دهد و به سمت راست می‌رود؛ یک دلیل برای انجام این کار این است که چاله موجود در سمت راست به عامل نزدیک‌تر باشد.



ج ب الف

شکل ۱-۴- نمایشی از دنیای کاشی‌ها

فرض کنید که تابع سودمندی دارای کران بالا باشد؛ در این صورت:

$$\exists k \in R \mid \forall\, r \in R, u(r) \le k$$

در این حالت می‌توان عامل بهینه را تعریف نمود. عامل بهینه عاملی است که بیشترین سودمندی را بدست آورد.

در این صورت می‌توان گفت که $P(r \mid Ag, Env)$ احتمال وقوع اجرا r می‌باشد زمانی که عامل Ag در محیط Env قرار می‌گیرد. واضح است که:

$$\sum_{r \in R(Ag,Env)} P(r \mid Ag, Env) = 1$$

عامل بهینه Ag_{opt} در محیط Env عاملی است که بیشترین سودمندی مورد انتظار را بدست می‌آورد. این عامل را می‌توان به صورت زیر تعریف نمود:

$$Ag_{opt} = \arg \max_{Ag \in AG} \sum_{r \in R(Ag,Env)} u(r) P(r \mid Ag, Env)$$

رابطه فوق ویژگی عامل بهینه را بیان می‌کند ولی این رابطه نمی‌گوید که عامل باید چگونه عمل کند تا بتواند بهینه شود. حالت بدتر این است که بعضی از عامل‌ها نمی‌توانند در ماشین‌های واقعی اجرا شوند. باید اشاره نمود که تمام عامل‌هایی که تا اینجا مورد بررسی قرار گرفتند به عنوان توابع ریاضی انتزاعی بودند که به صورت زیر تعریف می‌شوند:

$$Ag : R^E \rightarrow Ac$$

در این تعریف هیچ صحبتی نمی‌شود که عامل به چه مقدار حافظه نیاز دارد و همچنین پیچیدگی توابع آن به چه صورت است. نکته دیگری که باید به آن توجه نمود این است که تعریف توابع پیچیده از نظر ریاضی امکان‌پذیر است ولی پیاده‌سازی آنها در دنیای واقعی پیچیده و یا غیر ممکن می‌باشد. باید توجه نمود که هر کاری را نمی‌توان به کمک سامانه انجام داد و توانایی‌های سامانه نیز دارای حد و مرز مشخصی است. این قضیه نیز برای عامل‌ها معتبر است و حد و مرزی برای کارایی عامل مشخص می‌شود. در این صورت زیر مجموعه معینی از مجموعه همه عامل‌های AG قادرند که یک کار مشخص را انجام دهند؛ بنابراین عامل AG_m یک عامل از زیرمجموعه AG می‌باشد که می‌تواند کار m را انجام دهد. این عامل را می‌توان به صورت زیر تعریف نمود:

$$AG_m = \{Ag \mid Ag \in AG \text{ and } Ag \text{ can be implemented on } m\}$$

تابع سودمندی که برای عامل تعریف شد معایبی دارد. مهم‌ترین عیبی که این تابع دارد این است که اغلب تعریف این تابع و مشخص کردن آن مشکل می‌باشد. مشکل دیگری

که می‌توان درباره تابع سودمندی بیان نمود این است که بهتر است بجای میزان سودمندی راجع به اهدافی[1] که قرار است عامل به آن‌ها برسد صحبت شود.

خصوصیات کارهای مستند شده[2]:

این حالت همان تابع سودمندی می‌باشد با این تفاوت که خروجی آن { ۰ یا ۱ } است. برای تمایز این حالت با حالت قبلی از تابع Ψ استفاده می‌کنیم و آن را می‌توان به صورت زیر تعریف نمود:

خروجی تابع $\Psi(r)$ می‌تواند یکی از دو مقدار «درست» یا «غلط» باشد. در صورتی که مقدار تابع $u(r)$ برابر با یک باشد در این صورت مقدار تابع $\Psi(r)$ برابر با «درست» می‌باشد. در غیر این صورت به ازای همه مقادیر بین صفر و یک که تابع $u(r)$ گرفته است مقدار تابع $\Psi(r)$ برابر با «غلط» است.

محیط‌های کار[3]:

محیط‌های کار به صورت یک زوج مرتب $< Env , \Psi >$ تعریف می‌شوند که در آن Env محیط و تابع Ψ به صورت زیر تعریف می‌شود:

$$\Psi : R \rightarrow \{0, 1\}$$

محیط کار دو چیز را معلوم می‌کند:

۱- خصوصیات سامانه‌ای که عامل در آن کار می‌کند.

۲- معیارهایی که عامل برای «موفق» شدن یا «رد» شدن در کار در نظر می‌گیرد.

دو نمونه رایج از «کار»، کارهای دستاورد[4] و کارهای نگهداری[5] است که در ادامه به آن پرداخته می‌شود:

کارهای دستاورد: کار دستاورد به کمک مجموعه‌ای از حالات هدف تعریف می‌شود. هدف عامل رسیدن به یکی از حالت‌های مجموعه اهداف می‌باشد. اینکه این هدف از کدام نوع باشد مهم نیست. زیرا در این زمینه تمام هدف‌ها هم سطح در نظر گرفته می‌شوند. کارهای دستاورد رایج‌ترین نوع کار در هوش مصنوعی می‌باشند. بسیاری از مسائل مربوط به هوش مصنوعی از نوع این کارها هستند. برای فهم مطلب باید اشاره کرد که این قضیه همانند کار عامل با محیط می‌باشد. در این حالت عامل بر روی محیط عملی انجام می‌دهد

[1] Goal
[2] Predicate Task Specifications
[3] Task Environments
[4] Achievement Tasks
[5] Maintenance Tasks

و محیط با یک حالت به این عمل پاسخ می‌دهد، این عمل آنقدر ادامه پیدا می‌کند که در نهایت عامل به یکی از حالت‌های هدف برسد.

کارهای نگهداری: در این حالت برای عامل محدوده‌ای وجود دارد که عامل باید از آن اجتناب نماید. این محدوده B نامیده می‌شود. در صورتی که عامل از این محدوده دوری کند برنده خواهد شد.

حالتی دیگر نیز وجود دارد که نسبت به دو حالت قبلی پیچیده‌تر می‌باشد. این وضعیت ترکیب دو حالت کارهای دستاورد و کارهای نگهداری می‌باشد. در این حالت عامل به دنبال این است که یکی از حالت‌های کارهای دستاورد را بدست آورد و از حالت کارهای نگهداری دوری گزیند. البته حالت‌های خیلی پیچیده‌تر از این سه حالتی که توضیح داده شد نیز وجود دارد.

۱-۱۱- ترکیب عامل‌ها

دانستن اینکه تنها یک عامل وجود دارد که در محیط کار می‌تواند موفق باشد بسیار مفید خواهد بود، اما سودمندتر آن است که بدانیم ایجاد چنین عاملی به تنهایی بسیار مشکل می‌باشد. چگونه می‌توان چنین عاملی را بدست آورد؟ پاسخ صریحی که می‌توان به این سؤال داد این است که بر اساس خصوصیات موجود می‌توان به طور دستی چنین عاملی را ایجاد نمود. با این وجود حداقل دو راه دیگر نیز وجود دارد که در ادامه بحث خواهد شد:

۱- توسعه دادن الگوریتمی که به طور اتوماتیک ترکیبی از عامل‌ها را برای انجام یک کار مشخص کند.
۲- توسعه دادن الگوریتمی که به طور مستقیم بتواند عاملی طراحی کند که هدف آن این باشد که رفتار مشخصی انجام دهد.

اساساً ترکیب عامل‌ها بر اساس برنامه‌نویسی اتوماتیک صورت می‌پذیرد که هدف آن این است که محیط کاری را به عنوان ورودی دریافت کند و بر اساس این محیط کار عامل تولید شود که بتواند در این محیط با موفقیت عمل کند. این الگوریتم را می‌توان به صورت یک تابع نمایش داد که این تابع به صورت زیر تعریف می‌شود:

$$syn: TE \rightarrow (AG \cup \{\bot\})$$

توجه داشته باشید که خروجی تابع *syn* می‌تواند یک عامل یا (⊥)که همان null است باشد. این الگوریتم می‌تواند به دو صورت باشد:

سالم[1]: در این حالت، عامل می‌تواند با موفقیت بر روی محیط ورودی تابع syn خود عمل کند. در این حالت ممکن است که الگوریتم هیچ عاملی را معرفی نکند حتی اگر آن عامل درست کار کند. بنابراین باید توجه داشت تمام عامل‌های تولیدی توسط این الگوریتم با موفقیت کار خود را در محیط انجام می‌دهند. در صورتی که احتمال شکست عامل در محیط وجود داشته باشد، آن عامل هرگز تولید نخواهد شد.

کامل[2]: در این حالت موقعی که عامل به عنوان خروجی ارائه می‌شود، در صورتی می‌تواند بر روی محیط ورودی تابع syn کار خود را با موفقیت انجام دهد که این عامل توسط الگوریتم تضمین گردد. بنابراین احتمال این وجود دارد که عامل‌هایی تولید شوند که نتوانند بر روی محیط با موفقیت عمل نمایند. نکته‌ای که باید در اینجا به آن توجه نمود این است که خروجی این الگوریتم حتماً یک عامل خواهد بود حتی اگر احتمال شکست آن وجود داشته باشد. بنابراین در این الگوریتم هیچگاه $\perp$ به عنوان خروجی نمی‌باشد.

حالت ایده‌آل این است که الگوریتم‌های تولیدکننده عامل ترکیب الگوریتم سالم و کامل باشند. البته باید توجه نمود که از میان این دو نوع الگوریتم، الگوریتم سالم از اهمیت بیشتری برخوردار است. به این دلیل ممکن است که الگوریتم ترکیب عامل معیوب تولید نماید.

[1] Sound [2] Complete

۱-۱۲- مراجع و منابع

1. Haugeland, J. (Ed.). (1985). Artificial Intelligence: The Very Idea. MIT Press, Cambridge, Massachusetts.
2. Charniak, E. and McDermott, D. (1985). Introduction to Artificial Intelligence. Addison-Wesley, Reading, Massachusetts.
3. Bellman, R. E. (1978). An Introduction to Artificial Intelligence: Can Computers Think? Boyd & Fraser Publishing Company, San Francisco.
4. Winston, P. H. (1992). Artificial Intelligence (Third edition). Addison-Wesley, Reading, Massachusetts.
5. Kurzweil, R. (1990). The Age of Intelligent Machines. MIT Press, Cambridge, Massachusetts.
6. Poole, D., Mackworth, A. K., and Goebel, R. (1998). Computational intelligence: A logical approach. Oxford University Press, Oxford, UK.
7. Rich, E. and Knight, K. (1991). Artificial Intelligence (second edition). McGraw-Hill, New York.
8. Nilsson, N. J. (1991). Logic and artificial intelligence. Artificial Intelligence, 47(1-3), 31-56.
9. Russell, S., and Norvig, P. (1995). Artificial Intelligence: a Modern Approach: Prentice-Hall, Englewood Cliffs, NJ.
10. Russell, S. J., and Norvig, P. (2003). Artificial Intelligence.
11. Weiss, G. (1999). Multiagent systems: a modern approach to distributed artificial intelligence: The MIT press.
12. Wooldridge, M. and Rao, A. (Eds.). (1999). Foundations of rational agency. Kluwer, Dordrecht, Netherlands.
13. Wooldridge, M., and Jennings, N. R. (1995). Intelligent agents: theory and practice. The Knowledge Engineering Review, 10(2), 115-152.
14. Wooldridge, M. J. (2002). An introduction to multiagent systems: Wiley.