نظریه ها در ریاضی

استاد محترم :

جناب آقای گاراژیان

گردآورنده :

راحله دهسنگی

بهمن ۸۵

نظریه گراف

نظریه گراف دانشی است که درباره موجوداتی به نام گراف بحث می‌کند. به صورت مرئی گراف «چیزی» است شامل تعدادی رأس که با یالهایی به هم وصل شده‌اند. تعریف دقیق‌تر نظریهٔ گراف به این صورت است که گراف مجموعه‌ای از رأس‌ها است که توسط خانواده‌ای از زوج‌های مرتب که همان یال‌ها هستند به هم ربط داده شده‌اند.

آغاز نظریهٔ گراف به سدهٔ هجدهم بر می‌گردد. اویلر ریاضیدان بزرگ این نظریه را برای حل مسئله پل‌های کونیگزبرگ ابداع کرد اما رشد و پویایی اصلی این بخش بسیار زیبا از این نظریه تنها مربوط به نیم سدهٔ اخیر و با رشد علم داده‌ورزی (انفورماتیک) بوده است.

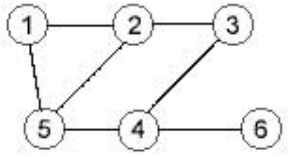
مهم‌ترین کاربرد گراف مدل‌سازی از پدیده‌های گوناگون و بررسی بر روی آنهاست. با گراف می‌توان به راحتی یک نقشه بسیار بزرگ یا شبکه‌ای عظیم را در درون یک ماتریس ذخیره کرد و یا الگوریتمهای مناسب را بر روی آن اعمال نمود.

یکی از قسمت‌های پرکاربرد نظریهٔ گراف، گراف‌های مسطح است که به بررسی گراف‌هایی می‌پردازد که می‌توان آن‌ها را به‌طوری روی صفحه کشید (با گذاشتن نقطه برای رأس‌ها و گذاشتن خم‌هایی که این نقاط را به هم وصل می‌کنند به جای یال‌ها) که این یال‌ها یکدیگر را قطع نکنند.

در ریاضی و علوم کامپیوتر، نظریه گرافعلمی است که به مطالعه گراف‌ها می‌پردازد.گراف مجموعه‌ای از راس‌هاست که بوسیله یال‌ها به هم وصل شده‌اند.به عبارت ساده‌تر به مجموعه‌ای از نقاط که بوسیله خطوط به هم وصل شده‌اند، گراف گویند. مفهوم گراف در سال ۱۷۳۶ توسط اویلر و با طرح راه‌حلی برای مساله پل konigsberg ارائه شد و به تدریج توسعه یافت.گراف‌ها امروزه کاربرد زیادی در علوم دارند. از گراف‌ها در شبکه‌ها،طراحی مدارهای الکتریکی, اصلاح هندسی خیابان‌ها برای حل مشکل ترافیک،و.... استفاده میشود.

تعریف

فرض کنید V یک مجموعه ناتهی و E زیرمجموعه‌ای از $V \times V$ باشد در این صورت زوج $G = (V, E)$ را یک گراف می نامند V.را مجموعه راس ها و E را مجموعه یال ها می گویند. اگر ترتیب قرار گرفتن راس ها در مجموعه E مهم باشد،گراف را گراف جهت‌دار می گویند و یال از راس v_1 به سمت راس v_2 را به صورت $e = (v_1, v_2)$ نشان می‌دهند.در غیر این صورت گراف را بدون جهت می‌نامند و یال بین راس های v_1 و v_2 با نماد $e = \{v_1, v_2\}$ نشان می‌دهند .



تعداد راس های یک گراف را **مرتبه** و تعداد یال های آن را **اندازه** گراف می نامیم .
در شکل روبرو گرافی را با شش راس و هفت یال مشاهده می کنیم

انواع گراف‌ها

گراف‌ها دارای انواع متعددی هستند که به برخی از آنها اشاره می‌کنیم :

- گراف همبند
- گراف ناهمبند
- گراف کامل
- گراف اویلری
- گراف همیلتونی
- گراف درختی
- گراف مسطح
- گراف دو بخشی
- گراف چندبخشی
- گراف-k مکعب
- گراف چرخ
- گراف ستاره‌ای
- گراف بازه‌ای
- گراف اشتراکی
- گراف منظم
- گراف جهت‌دار

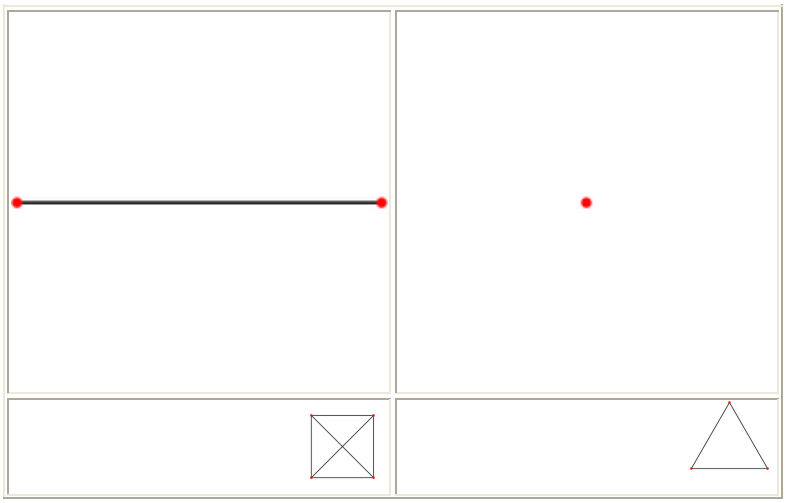
گراف کامل

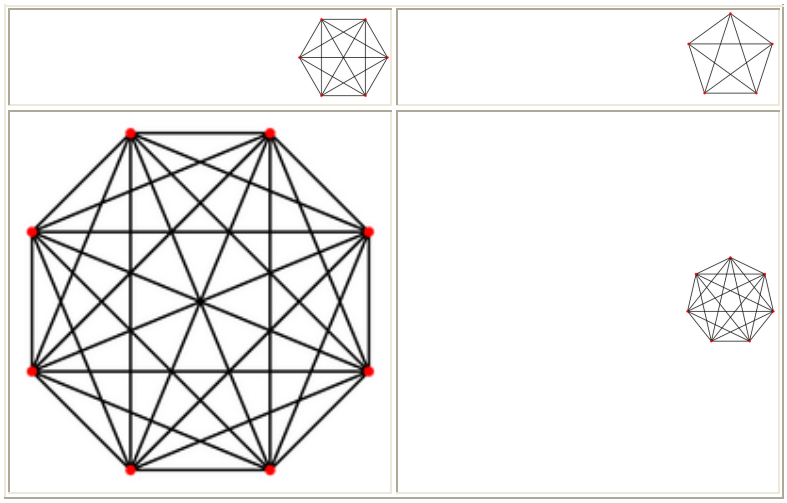
در نظریه گراف ،یک گراف کامل ،گرافی است که هر بین هر دو راس آن دقیقا یک یال وجود داشته باشد.

یک گراف کامل از مرتبه n،دارای n راس و $\frac{n(n-1)}{2}$ یال است و آن را با k_n نشان می‌دهند.
یک گراف کامل یک گراف منتظم از درجه n-۱ است.

مثال‌هایی از گراف کامل

در شکل زیر گراف‌های کامل از مرتبه یک تا مرتبه هشت نمایش داده شده است. از تعریف این نوع گراف معلوم است که گراف کامل از مرتبه اول ،هیچ یالی ندارد .





نظریه

اندازه در ریاضیات، به تابعی گفته می‌شود، که یک عدد یا مقدار را (به عنوان مثال اندازه، حجم یا احتمال) به هر زیرمجموعه از یک مجموعه خاص، نسبت می‌دهد. این نظریه به منظور، محاسبه انتگرالها بر روی مجموعه‌ها به جای برروی فاصله‌ها (یا همان بازه‌ها) که در معمول انجام می‌پذیرد، گسترش پیدا کرد و از این رو در آنالیز ریاضی و در نظریه احتمالات، بسیار دارای اهمیت می‌باشد.

نظریه اندازه، قسمت مهمی از آنالیز حقیقی است، به طوری که جبرهای سیگما، قیاس‌ها توابع قیاس‌پذیر و انتگرالها را مورد تحقیق، قرار می‌دهد و در نظریه احتمالات و آمار از اهمیت زیادی برخوردار است.
تعریف

برطبق تعریف، اندازه μ تابعی است (یا به عبارتی نگاشتی است)، که برروی جبر سیگمای Σ بر مجموعه X، تعریف می‌شود و مقادیر بین $[0, \infty[$ ، می‌پذیرد و دارای خصوصیت‌های زیر است:

$$\mu(\varnothing) = 0$$

$$\mu\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} \mu(E_i)$$

در اینجا $\varnothing$مجموعه تهی و $E_1, E_2, E_3 \cdots$ تعداد شمارایی از مجموعه‌هایی در Σ هستند، که اشتراک هرکدام از آنها با دیگری تهی است (مجموعه‌ها مجزا هستند). در این حالت به (Σ,μ,X) **فضای اندازه‌ای** و به اعضای Σ، **مجموعه‌های اندازه‌پذیر** گفته می‌شود.

دیدکلی
چرا اندازه گیری می‌کنیم؟
قوانین و نظریات فیزیک بصورت معادلات ریاضی بیان می‌شوند. حال ما از کجا بدانیم که هر معادله خاص ، رفتار چیزی را بیان می‌کند؟ باید این قاعده امتحان شود و به مرحله آزمون گذاشته شود. بنابراین ، اندازه گیری مهارتی است که میان نظریه علمی و دنیای واقعی رابطه ایجاد می‌کند. این رابطه دو طرفه می‌باشد. هر رویداد اندازه گیری شده‌ای که قبلا پیشگویی نشده باشد، باید نظریه جدید آنرا توجیه کند.

اشخاصی که کار تجربی انجام می‌دهند باید اطلاعات فنی جامعی از اصول اندازه گیری داشته باشند. نحوه اندازه گیری و محدودیتهای ناشی از وسایل اندازه گیری را بشناسد. هر دانشمندی فقط با دانستن اینکه چه

اندازه گیریهایی انجام شده است و نحوه اندازه گیریها چگونه بوده است، می‌تواند اثر و کشفیات دانشمندان دیگر را خوب بفهمد. بنابراین ، اندازه گیری هنری است که در حال حاضر تکنولوژی پیشرفته حامی آن است.

دقت در اندازه گیری

در اندازه گیریها جواب کامل نداریم، هر کسی که نتیجه اندازه گیری خود را گزارش می‌کند، همواره بهترین تخمین خود را از مقدار اصلی ، همراه با خطای اندازه گیری آن ، ارائه می‌دهد. یعنی اگر طول جسمی بصورت ۱۸۳±۵mm نوشته شود، منظور نویسنده این است که مقدار واقعی طول بین ۱۷۸ و ۱۸۸mm قرار دارد. صحت اندازه گیری از روی تطابق آن با واقعیت نتیجه می‌شود. خطای زیاد بیانگر عدم اعتماد آزمایشگر بر اندازه گیری است. اندازه گیری دقیق ، اندازه گیریی است که خطای آن ، در مقایسه با مقدار اندازه گیری شده بسیار کوچک باشد.

در مثال اخیر خطای نسبی اندازه گیری برابر است با: ۱۰۰% ±= ۲% . ۷٤ × (۱۸۳/۵±). دقت اندازه گیری به مهارت آزمایشگر در تخمین زنی ، مکانیزم عمل اندازه گیری ، حد تفکیک وسیله اندازه گیری ، حد تفکیک چشم و غیره بستگی دارد. البته درستی اندازه گیری به طبیعت جسمی که اندازه گیری می‌شود نیز وابسته است. بنابراین ، صحت تمامی اندازه گیریها ، به دلیل محدودیت در دقت (تکرار پذیری آزمایش) و خطای ناشی از طبیعت وسیله اندازه گیری و جسمی که اندازه گیری می‌شود، محدود است.

ارقام با معنی

پذیرش میزان خطا در اندازه گیری و نوع ریاضیاتی که در تخمین و محاسبات داده‌های آزمایش و نحوه قرائت آنها بستگی دارد. یک روش اصولی برای ارزیابی صحت اندازه گیری و پذیرش آن توجه به تعداد ارقام با معنی آن است. تعداد ارقام بامعنی ، درستی و دقت اندازه گیری را می‌رساند. به عبارتی هر چه اندازه گیریی دقیقتر باشد مقدار ارقام با معنی نتیجه اندازه گیری بیشتر خواهد بود. آخرین رقم با معنی در اندازه گیری همیشه تخمینی است. مثلا اگر در اثر اندازه گیری طول اتاقی ۷۲۰cm باشد، مفهوم این است که اندازه گیری با سه رقم معنی دار انجام شده است که رقم آخر آن صفر می‌باشد که ممکن است درست یا غلط باشد.

صفرهای موجود در عدد گزارش شده ممکن است با معنی باشند یا محل ممیز را نشان دهند. مثلا طول ۸۰۲mm که یک عدد دو رقمی است، بر حسب متر برابر ۰,۰۰۸۲ است، چون نتیجه تغییر نکرده پس این طول بر حسب متر هم یک عدد دو رقمی است. بنابراین قاعده کلی این است که: صفرهای سمت چپ هرگز معنی دار نیستند. صفرهای پایانی نیز ممکن است معنی دار باشند یا نباشند. اگر طول زمینی را ۲۳۰m اندازه بگیرید، در این اندازه گیری عدد گزارش شده دارای ٤ رقم با معنی است، البته بدون ممیز تشخیص معنی دارابودن یا نبودن رقم آخر با قطعیت مشخص نمی‌شود ، مگر اینکه از نحوه اندازه گیری اطلاعی داشته باشیم.

در مورد اندازه گیری مذکور بهتر است داشته باشیم ۲۳۰,۰ ، در چنین حالتی می‌گوییم دقت اندازه گیری تا ۰,۱ اعشار درست است. در جمع و تفریق اندازه گیریها انتشار خطا خواهیم داشت. مثلا خطای اندازه گیری با دقت ۰,۱ به اندازه گیری با دقت ۰,۰۰۱ سرایت می‌کند. البته در اندازه گیریها ، پردازش داده‌های اندازه گیری ، روش گرد کردن و محاسبه خطا (نسبی و مطلق) وجود دارد که میزان اعتبار و دقت اندازه گیری را بیان می‌نماید. معیار اصلی در گزارش اندازه گیری و مقادیر حاصل از آنها ، کاربرد دقیق تعداد ارقام با معنی است.

نمادگذاری علمی

اگر تمامی فواصل در متریک SI نوشته شود، هنگام نوشتن فاصله تا نزدیکترین ستاره (عدد بزرگ) یا هنگام نوشتن قطر هسته اتم (عدد کوچک) کار مشکل خواهد بود. در مورد ستاره ۱۵ صفر در پایان و در هسته ۱۵ صفر در ابتدای عدد وجود دارد. تنها تکلیف این صفرها مشخص نمودن محل ممیز می‌باشد. بهترین راه برای حل مشکل استفاده از نماد گذاری علمی است. در این روش در هر عدد ممیز را بعد از اولین رقم غیر صفر نوشته و سپس آنرا در توانی از ۱۰ ضرب می‌کنند تا محل ممیز را نشان دهند. مثلا عدد ۱٤۲۰۰۰ در نماد گذاری علمی بصورت زیر در می‌آید:

۱,٤۲ = ۱٤۲۰۰۰ × ۱,٤۲ = ۱۰۰۰۰۰×۱۰^۵

در واقع بهترین راه نوشتن اعداد بسیار بزرگ و کوچک همین است. البته در این روش تشخیص تعداد ارقام با معنی و محل ممیز راحت است. بخصص در مورد صفرها که کار بسیار راحت شده است. مزیت مهمی که نمادگذاری علمی دارد، این است که حساب در نماد گذاری علمی راحت صورت می‌گیرد. یعنی افزودن به توانهای ۱۰ راحتتر از شمردن صفرهاست. یعنی محاسبات اعشاری چه در اعداد کوچک و چه در اعداد بزرگ به محاسبات توانی تبدیل می‌شود که براحتی انجام می‌گیرد. البته در جمع و تفرق اعداد که توان برابر ندارند، ابتدا بایستی ممیز را در یکی از اعداد جابجا کرده و توان آنها را یکی نمود.

بعد اندازه گیری
هر اندازه گیری از دو قسمت عدد و نشان تشکیل شده است. مثلا اگر بگویید وزن من ٦۰ است، مخاطب چیزی از این عدد نمی‌فهمد. مگر اینکه بگویید قد من ٦۰ کیلوگرم است. برای کلیه اندازه گیریها باید یک شاخصی برای معرفی عدد در کنارش باشد تا به آن عدد ریاضی مفهوم واقعی دهد. برای کمیات مختلف یکاهای متعددی مطرح شده که در محاسبات و اندازه گیریها باید آنها را به یک یکای مشترک تبدیل کرد. به عبارت دیگر باید در یک متریک واحد اندازه گیریها را انجام داده و نتیجه را هم یا در آن متریک و یا با تبدیلات مربوطه در دستگاه دیگری بیان کرد. زیرا در اندازه گیریها و محاسبات فقط کمیاتی را که بعد یکسانی دارند، می‌توان با استفاده از یکاهای تبدیل باهم جمع یا از هم تفریق و یا باهم مقایسه کرد.
نظریه بازی در ریاضی

یک برداشت پیچیده از یک رابطه ساده

مسئله آشنایی است: n شئی متمایز داریم , می خواهیم r تا از آنها را کنار هم قرار دهیم . یعنی p(n,r) یک ترتیب.

$$P(n,r)=n!/(n-r)!$$

فرض کنید ۵ شئی متمایز داریم و r تای آنها را می خواهیم کنار هم قرار بدهیم (ترتیب مهم است) داریم:

$$P(۵،۳)=۵*۴*۳$$

تا اینجا نگرش تعداد انتخابهای ممکن بر اساس فاکتورهای محتمل از فضای قابل انتخاب برای هر سهم از r می باشد. در اینجا اگر بخواهیم مسئله فوق را به یک فضای تصمیم سازی گسترش بدهیم با برداشت زیر مواجه خواهیم شد که از کل گزینه های ممکن (!۵) تعداد حالاتی که توان تامین مطلوبیت ما را دارا هستند در ۳*۴*۵ گزینه خلاصه می شوند. در اینجا یک نکته وجود دارد :

آیا حالتی وجود دارد که در انتخابها نا دیده گرفته شده باشد. به عبارتی در ۱۲۰ حالت ممکن که ۶۰ حالت آن تامین کننده مطلوبیت است , ۶۰ حالت دیگر در نظر گرفته نشده باشد؟

قاعدتا بنا بر اصول تصمیم سازیهای استراتژیک , وقوع پیشامد فوق (ینی توجه به ۶۰ گزینه از بین ۱۲۰ گزینه ممکن) یک فاجعه است و درایت استراتژیست را به چالش می طلبد. پس ۶۰ انتخاب مستتر دیگر کجاست؟

می خواهم مسئله فوق را با نگرشی دیگر مورد بررسی قرار بدهم. به تعدا حالتهای ممکن توجه کنید:

۵	۴	۳	۲	۱

زمانیکه شما گزینه منتخب خود را تشکیل داده اید و ۳ عنصر (در بلوک خاکستری) از ۵ عنصر را بکار گرفته اید, نحوه چیدمان دو عنصر دیگر برای شما زیاد اهمیتی ندارد. زیرا انتخاب نشده اند و نحوه قرار گیری آنها در بلوکهای مجازی (بلوکهای سفید) مهم نیستند. اما واقعیت این است که اگر فقط بر اساس منطق صفر و یک آن را تحلیل نکنیم , برای هر انتخاب در این مثال دو حالت تبهگن داریم که از نظر تامین مطلوبیت دارای ارزش یکسانی هستند. همچون چیدمان فرضی زیر :

a	d	f	o	s
a	d	f	s	o

اما اگر در یک مسئله تصمیم سازی و بر اساس منطق فازی این دو حالت تبهگن را تحلیل کنیم , شاید به پتانسیلهایی بر خورد کنیم که جدول استراتژیک را دستخوش دگرگونی نمایند یا برای تامین مطلوبیت هزینه های متفاوتی را ایجاد کنند , شاید سرمایه گذاریی پنهان , یا امتیازی ویژه بوده و یا تمامی این حالات تبهگن تکرار های میسری از یک انتخاب در دوره های مختلف زمانی باشند . تنها چیزی که مهم است این که نگرش اول فقط به یک جواب می رسد , در حالیکه نگرش دوم تامین کننده حالات بسیار متنوع تبهگنی است که ویژگیهای متفاوت و خاصی از خود بروز می دهند و هر یک آبستن فرصتها و تهدیدهای محیط بیرونی بوده و با تحقق یک هدف استراتژیک به خط پایان نمی رسند.

اشغال در رد ریاضی

خلاصه

این مقاله به بررسی جنبه‌های مختلف و رو به رشد منطق محاسباتی می‌پردازد. تکنیکها و کاربردهای فعلی آن را مطالعه میکند و در نهایت به یک نتیجه‌گیری و ارایه پیشنهادهایی در مورد منطق محاسباتی می‌پردازد.

۱- مقدمه

منطق محاسباتی۲ بخشی از منطق است که به بررسی راهکارهای مختلف بررسی درستی احکام در دستگاه‌های مختلف منطقی میپردازد. این رشته به طور عمیقی با علوم کامپیوتر پیوند یافته است و به صورت

کلی رشد واقعی آن از وقتی شروع شد که توان محاسباتی کامپیوترها پیشرفت کرد و انجام محاسبات پیچیده بوسیله کامپیوترها با هزینه کم امکان پذیر شد. منطق محاسباتی به صورت کلی به منطق از دید محاسباتی آن مینگرد. این که در یک دستگاه منطقی انجام یک محاسبه (به طور مثال چک کردن درستی یک گزاره) امکان پذیر هست یا نه و اگر امکان پذیر است این کار چه هزینه ای دارد. از آنجا که حقایق علمی ما با منطق پیوند عمیقی دارند، برای بررسی این حقایق استفاده از زبان منطقی، یکی از بهترین راه های ممکن است.

امروزه بشر علاقه زیادی دارد که تمام کارها از جمله فکر کردن را به ماشین واگذار کند. اما واگذار کردن فکر کردن به یک ماشین کار ساده ای نیست. ما دید عمیقی درباره اینکه فکر کردن چیست و چگونه انجام میشود نداریم. ازینرو تلاشهای اولیه برای این کار با شکست مواجه شدند یا با سختی زیادی همراه بودند. اما اگر بخواهیم تنها قسمت منطقی فکر کردن را به ماشین واگذار کنیم کار ساده تر است چون برای این کار از منطق ریاضی استفاده میکنیم و منطق یک زیر شاخه قوی از ریاضی است که به سوالات زیادی در مورد آن جواب داده شده است. گرچه ما هنوز واقعا نمیدانیم که چه مقدار از روند تفکر ما منطقی است. به این مطلب در قسمت نتیجه گیری بیشتر خواهیم پرداخت.

امروزه منطق محاسباتی کاربرد گسترده ای در تکنولوژی پیدا کرده است. بدین ترتیب حجم کارهای انجام شده بر روی آن در حال افزایش است. این کارها نه تنها در زمینه ریاضی بلکه بر روی دیگر ابعاد مربوط به این قضیه نیز انجام میشود. عموما این کارها به سه دسته تقسیم میشوند. دسته اول کارهای مرتبط با پایه ریاضی منطق محاسباتی هستند. دسته دوم کارهای مرتبط با تکنیکهای هوش مصنوعی جهت ارتقای کارایی روشهای ریاضی ابداع شده و دسته سوم کارهای انجام شده جهت استفاده از منطق محاسباتی در مسایل واقعی مهندسی.

۲. پایه‌ی منطق محاسباتی

تمام موارد مرتبط با منطق محاسباتی احتیاج به پایه‌ای برای بنا کردن ساختارهایی معنا دار برای توصیف داده های مربوطه دارند. باید بتوانیم درباره درستی یک گزاره با توجه به دیگر گزاره ها اظهار نظر کنیم. بدین منظور میتوان از مراتب مختلف منطق استفاده کرد. سیستمهای بسیار ساده معمولا از منطق مرتبه صفر برای توصیف جهان خود استفاده میکنند. اما اکثر سیستمهای پیشنهادی از منطق مرتبه اول برای توصیف جهان خود استفاده میکنند. بعضی سیستمها هم از مراتب بالاتر منطق برای اهداف خود استفاده میکنند. هنوز نمیدانیم که ذهن انسان تحت چه مرتبه‌ای از منطق کار میکند، و حتی به درستی نمیدانیم آیا تمام جنبه های تفکر در ذهن انسان از اصول منطق تبعیت میکنند یا نه. به هر حال علم منطق روشی سمبولیک برای مدل کردن جهان در اختیار ما قرار میدهد.

چرچ در ۱۹۳۶ ثابت کرد که منطق مرتبه اول برای زبانی که فقط یک نماد رابطه‌ای دو موضعی داشته باشد تصمیم ناپذیر است. بنا بر قضیه چرچ روشی متناهی برای پاسخ به این سوال که آیا جمله A در منطق مرتبه اول معتبر است، به صورت "آری" یا "نه" نداریم، اما نیمه ای از پاسخ را میتوان مهیا کرد. به عبارت بهتر روشی متناهی وجود دارد که اگر A معتبر باشد، پاسخ روش "آری" است. به عبارت دیگر مجموعه جملات معتبر در منطق مرتبه اول لیست پذیر هستند. از طرف دیگر با توجه به قضیه تمامیت (در صورتی که در مورد دستگاه استنتاجی ما درست باشد) با استفاده از فرضها و اصول استنتاج میتوان جملات درست را لیست کرد. این قسمت در حقیقت قلب تپنده‌ی منطق محاسباتی است. در صورت پیدا شدن روشهای جدید و سریعتر برای چک کردن درستی یک جمله تحت چند فرض، شاهد تحول بزرگی در دیگر شاخه های مرتبط با این موضوع خواهیم بود.

تحقیقات در بخش پایه‌ی منطق محاسباتی به طور گسترده‌ای بر دیگر بخشهای این علم تاثیر دارند. این تحقیقات عموما به دو بخش تقسیم میشوند:

تحقیقات در زمینه‌های روشهای استنتاج از قبیل Resolution و ...
تحقیقات در زمینه‌ی پیدا کردن پایه۳ های مناسب ریاضی برای انجام به صرفه‌ی (از نظر زمانی و حافظه) محاسبات مربوط به منطق محاسباتی.

۲-۱ پایه‌های منطق محاسباتی

روش کلی برای فهمیدن درستی یک جمله این است که از فرضها شروع کرده و در هر مرحله یک جمله درست جدید را با توجه به جملات قبلی و استفاده از قواعد استنتاج تولید کنیم. (یعنی جملات درست را لیست میکنیم.) این کار ادامه پیدا میکند تا وقتی که به جمله مورد سوال یا نقیض آن برسیم.

قسمت دیگری که مورد توجه است، یکی سازی٤ است. به طور مثال دو جمله (x:f(x□ و (y:f(y□ را در نظر بگیرید. واضح است که درستی این دو جمله یکسان است. به طور کلی هر جمله را به طریقه های ظاهرا متفاوت بسیار زیادی میتوان نوشت که همگی یک معنای واحد داشته باشند. (در همین مثال به جای x از تمام متغیرها میتوان استفاده کرد. به صورت معمولی لااقل N۰ متغیر داریم.) بدین منظور تحقیقات زیادی بر روی روشهای کارا برای یکی سازی جملات منطقی انجام شده است.

برای تولید جملات جدید با توجه به قواعد استنتاج راههای زیادی پیشنهاد شده اند. یکی از محبوبترین راههای پیشنهاد شده به Resolution موسوم است. این روش برای منطق مرتبه اول کمی پیچیدگی دارد اما با بررسی آن برای منطق گزاره ها کلیت آن آشکار میشود.

Resolution Propositional

در این روش تمام جمله ها به صورت clausal form هستند. برای تبدیل یک جمله به این فرم ابتدا جمله را به صورت نرمال عطفی CNF تبدیل میکنیم.

(g □ (r → f)) ——CNFà (¬g □ r) □ (¬g □ ¬f)¬

و سپس نتیجه را به تعدادی مجموعه تبدیل میکنیم، مجموعه ای از مجموعه ها که هر عضو آن اعضای یکی از پرانتزهاست:

{g □ r) □ (¬g □ ¬f) ——Clausal Formà {¬g, r}, {¬g, ¬f¬)

این کار یک روش نسبتا خوب برای Unification است. برای انجام استنتاج بر اساس این قاعده عمل میکنیم:

میدانیم که

(p □ q) □ (¬p □ r) □ (q □ r)

میتوان نشان داد که استفاده از این رابطه به عنوان تنها قاعده استنتاج برای استنتاج کافی است. این رابطه در clausal form به این شکل تبدیل میشود:

{p, q}

{p, r¬}

_____

{q, r}

این تعریف برای مجموعه های بیش از دو عضو نیز قابل گسترش است. به موارد جالب زیر توجه کنید:

{p, q¬}

{p}

_____

{q}

(این معادل با قاعده Modus Ponens است.)

{p}

{p¬}

_____

{}

(تناقض!)

۲-۲ پایه‌ی ریاضی

دسته دیگری از کارهایی که به عنوان بخشهای پایه منطق محاسباتی شناخته میشوند، کار بر روی پایه های منطقی ریاضیات است. اگرچه دستگاه های کلاسیک شناخته شده ای برای توصیف ریاضی در بوسیله منطق وجود دارند اما کارهایی برای پیدا کردن دستگاه‌هایی که برای استنتاج خودکار در ریاضی عملکرد بهتری داشته باشند در حال انجام است. به عنوان یک مثال میتوانید به [Beli۰۱] مراجعه کنید.

۳ کاربردهای منطق محاسباتی

منطق محاسباتی امروزه به طور گسترده‌ای جهت حل مسایل مهندسی در حال استفاده است و این استفاده با سرعت زیادی در حال گسترش است. زمینه‌های مهمی که امروزه از منطق محاسباتی در آنها استفاده میشود عبارتند از:

Database Systems
با استفاده از سیستمهای مبتنی بر منطق محاسباتی میتوان Database ها را از محل ذخیره‌ی اطلاعات خام به پایگاههای هوشمند داده‌ها تبدیل نمود، به این ترتیب شاهد منابع هوشمندی از اطلاعات خواهیم بود که استفاده از آنها به مراتب ساده‌تر از موارد مشابه فعلی است. با توجه به اینکه جهان امروزی به شدت مبتنی بر استفاده از پایگاه‌های داده است، به نظر میرسد که در این قسمت سرمایه گذاری زیادی انجام شود و لذا پیشرفت در این زمینه بسیار سریع خواهد بود که این امر منجر به پیشرفت سریعتر دیگر موارد مربوط به منطق محاسباتی خواهد شد.

Software Analysis
با توجه به اینکه امروزه نرم افزارهای نوشته شده به سرعت در حال گسترش هستند و ما شاهد نرم افزارهایی هستیم که تنها یک قسمت از آنها میتواند شامل میلیونها خط از کد باشد، بنابراین به زودی شاهد بوجود آمدن مشکلات بزرگی هنگام تست کردن برنامه‌های عظیم نوشته شده خواهیم بود. تکنیک‌های فعلی مثل JUnit یا موارد مشابه، هیچکدام از هوشمندی لازم برخوردار نیستند و برای کاربردهای آینده مناسب نخواهند بود. با استفاده از منطق محاسباتی به طور دقیق و با سرعت کافی میتوان نقاط ضعف نرم افزارهای نوشته شده را پیدا کرده و حتی نسبت به رفع آنها اقدام نمود. ویرایشگرهای مدرن برنامه‌نویسی، امروزه، به طور گسترده‌ای ازمنطق محاسباتی برای کمک به برنامه‌نویس استفاده میکنند.

Hardware Engineering
امروزه در طراحی سخت‌افزار هم به مانند طراحی نرم‌افزار پیشرفتهای عمده‌ای به وجود آمده است، قانون مور بیان میکند که تعداد ترانزیستورهای یک تراشه در هر سال دو برابر خواهد شد (از نظر تکنولوژی ساخت). این قانون تاکنون نقض نشده است و اگر این روند ادامه پیدا کند در طی مدتی کوتاه شاهد تراشه‌های کامپیوتری‌ای خواهیم بود که حاوی صدها میلیار ترانزیستور هستند، وضوحا تست کردن درستی عملکرد این تراشه‌های عظیم از طریق روشهای کلاسیک موجود امکان‌پذیر نخواهد بود و سیستمهای مبتنی بر منطق محاسباتی به طور عمده‌ای برای این کار استفاده خواهند شد.

Automated Theorem Proving
از منطق محاسباتی میتوان به صورت گسترده‌ای جهت اثبات خودکار قضایای ریاضی استفتده کرد.

۱-۳ STRIPS

یکی از مهمترین مسایل در علم روباتیک برنامه ریزی حرکت روبات۵ است. روباتی را در نظر بگیرید که دارای توانایی‌های متعارف حرکتی است و قادر است با دستهای خود اقدام به جابجا کردن اشیا نماید، همچنین به وسیله چشم الکترونیکی خود میتواند اطلاعات تصویری از جهان پیرامون خود دریافت کند و آنها را تجزیه و تحلیل کند. این ویژگیها برای یک روبات همگی ویژگیهایی مکانیکی محسوب میشوند. روباتی با این ویژگیها مهمترین کاری که باید بتواند انجام دهد این است که بتواند به طریقه‌ای مفید از قابلیتهای مکانیکی خود استفاده کند.

به این ترتیب با توجه به پیشرفت قابل توجه علم روباتیک، مسایل مکانیکی روباتیک تقریبا حل شده به نظر میرسند و مساله‌ی مهمتر هوشمند ساختن روباتها می باشد. مدلهای مختلفی برای هوشمند ساختن روباتها پیشنهاد شده‌اند که هرکدام از این مدلها مزایا و معایب خاص خود را دارند اما مشکل بیشتر این مدلها این است که تنها قادر به حل یک رده از مسایل خاص هستند و نمیتوان آنها را برای حل مسایل جدید تغییر داد، همچنین این مدلها اکثرا از هوشمندی لازم برای یادگیری برخوردار نیستند. اما مدلهای مبتنی بر منطق محاسباتی در حد خوبی این مشکلات را ندارند، در این جا به بررسی یکی از این مدلها میپردازیم.

STRIPS عضو یک رده از حل‌کننده‌های مسایل۶ است که در فضای مدلهای جهان۷ برای یافتن مدلی که در آن یک هدف داده شده یافت شود، جستجو میکنند. برای هر مدل جهان فرض میکنیم یک مجموعه از عملگرهای قابل اعمال به جهان وجود دارند. هر عملگر مدل جهان را عوض میکند، به این ترتیب با استفاده از عملگرهای مختلف میتوانیم از یک مدل به مدلهای مختلفی برویم. وظیفه‌ی حل‌کننده مساله این است که دنباله‌ای از عملگرها را پیدا کند که توسط آنها بتوان از یک مدل ابتدایی به مدلی انتهایی رسید که هدف داده شده را ارضا کند.

این چارچوب برای حل مسایل در مورد بسیاری از مسایل هوش‌مصنوعی قابل اعمال است، اما هدف ما در اینجا حل مسایلی که یک ربات با آنها مواجه میشود است. مدل جهان این گونه از مسایل بسیار گسترده و پیچیده است. برای مقایسه مسایل مرتبط با حل پازل را در نظر بگیرید. مدل یک پازل را میتوان به سادگی در یک لیست یا ماتریس نگه داشت. اما در مورد یک ربات مدل ما از جهان شامل حجم بزرگی از اطلاعات مثل مکان ربات، مکان و خصوصیات اشیا، فضاهای باز و دیوارها و ... میباشد. بدین ترتیب یکی از بزرگترین

مسایل دراین مورد نحوه نگه داری مدل جهان است. در STRIPS مدل جهان در یک مجموعه از فرمولهای خوش شکل ریخت((wff۸ در زبان منطق مرتبه اول، نگه داری میشود.

عملگر۹ها عناصر اصلی برای پیدا کردن جواب هستند. هر عملگر متناظر با یک روال کاری۱۰ است. فرق این دو این است که عملگرها هنگام برنامه ریزی استفاده میشوند ولی روالهای کاری وقتی استفاده میشوند که ربات میخواهد جواب پیدا شده را به صورت فیزیکی انجام دهد. (یعنی یک عملگر در جهان منطقی معادل با یک روال کلری در جهان واقعی است.)

سوال بعدی این است که یک عملگر مدل جهان را چگونه تغییر میدهد؟ در STRIPS این کار از طریق یک لیست اضافه۱۱ و یک لیست حذف ۱۲ انجام میشود. همچنین هر عملگر تعدادی شرط مقدماتی ۱۳ برای انجام شدن دارد.

مثلا عملگر push(k,m,n) را برای هل دادن شی k از مکان m به n به صورت زیر تعریف میشود:

(push(k, m, n

:precondition

((m١٤ATR

(At(k, m

:delete list

(ATR(m

(AT(k, m

:add list

(ATR(n

(AT(k, n

پس برای بدست آوردن جهان جدید از جهان قبلی تحت تاثیر یک عملگر تنها کافی است که ابتدا با چک کردن شرطهای مقدماتی بفهمیم که آیا عملگر قابل اعمال به جهان هست یا نه؟ اگر بود، جملات لیست حذف را از جهان حذف کرده و جملات مربوط به لیست اضافه را به جهان اضافه کنیم.

مثال

مثال زیر را در نظر بگیرید:

فرض کنید که این شکل نقشه‌ی یک ساختمان است که روبات در آن قرار دارد و همچنین تعداد جعبه و ... در این ساختمان است.

جهان و عملگرها به این شکل تعریف میشوند:

Initial World Model

(CONNECTS(x,y,z)=~ C C N N E CTS(x,z,y

(٥, ROOM١, ROOM١CONNECTS(DOOR

(٥, ROOM٢, ROOM٢CONNECTS(DOOR

(٥, ROOM٣, ROOM٣CONNECTS(DOOR

(٥,ROOM٤,ROOM٤CONNECTS(DOOR

(٤LOCINROOM(f, ROOM

(١AT(BOXl,a) INROOM(BOXI,ROOM
(١,ROOM٢,b) INROOM(BOX٢AT(BOX
(١,ROOM٣,c) INROOM(BOX٣AT(BOX
(١AT(LIGHTSWlTCH l,d) INROOM(ROBOT, ROOM
(١,ROOM١ATROBOT(e) INROOM(LIGHTSWITCH
(TYPE(BOXI,BOX) PUSHABLE(BOXI
(٢,BOX) PUSHABLE(BOX٢TYPE(BOX
(٣,BOX) PUSHABLE(BOX٣TYPE(BOX
TYPE(IM,DOOR) ONFLOOR
(, OFF١,DOOR) STATUS(LIGHTSWITCH٣TYPE(D
(, LIGHTSWITCH١,DOOR) TYPE(LIGHTSWITCH٢TYPE(D
(TYPE(DI,DOOR

Operators

.(m): Robot goes to coordinate location m١goto

:Preconditions

[(ONFLOOR) ^ (□x)[INROOM(ROBOT,x) □ LOCINROOM(m,x)

($,Delete list: ATROBOT($),NEXTTO(ROBOT

(Add list: ATROBOT(m

.(m): Robot goes next to item m٢goto

:Preconditions

(ONFLOOR) □ {(□x)[INROOM(ROBOT,x) □ INROOM(m,x)] □ (□x)(□y)

{[(INROOM(ROBOT,x) □ CONNECTS(m,x,y]

($,Delete list: ATROBOT($),NEXTTO(ROBOT

(Add list: NEXTTO(ROBOT, m

pushto(m,n): robot pushes object m next[to item n

:Precondition

(PUSHABLE(m) □ ONFLOOR □ NEXTTO(ROBOT,m) □ {(□x)[INROOM(m,x

{[(INROOM(n,x)] □ (□x□y)[INROOM(m,x) □ CONNECTS(n,x,y □

:Delete list

($,AT ROBOT ($), NEXTO (ROBOT, $), NEXTI'O ($,m), AT(m, $), NEXTTO(m

(Add list: NEXTTO(m,n), NEXTTO(n,m), NEXTTO(ROBOT, m

.turnonlight(m): robot turns on lightswitch m

:Precondition

{n)[TYPE(n,BOX) □ ON(ROBOT, n) □ NEXTO(n,m)I□)}

(TYPE(m, LIGHTSWITCH □

(Delete list: STATUS(m,OFF

(Add list: STATUS(m,ON

.climbonbox(m): Robot climbs up on box m

:Preconditions

(ONFLOOR □ TYPE(m,BOX) □ NEXTTO(ROBOT, m

Delete list: ATROBOT($),ONFLOOR

(Add list: ON(ROBOT,m

.climboffbox(m): Robot climbs off box m

(Preconditions: TYPE(m,BOX) □ ON(ROBOT, m

(Delete list: ON(ROBOT, m

Add list: ONFLOOR

.gothrudoor (k, l, m): Robot goes through door k from room l into room m

:Preconditions

NEXTTO(ROBOT, k) □ CONNECTS(k, l, m) □ INROOM(ROBOT, l) □ ONFLOOR

:Delete list

($,ATROBOT($), NEXTTO(ROBOT,$), INROOM(ROBOT

(Add list: INROOM(ROBOT,m

با استفاده از STRIPS این راه حلها برای انجام کارهای زیر پیشنهاد شده است:

۱. Turn on the lightswitch

(Goal wff: STATUS (LIGHTSWITCHI, ON

:STRIPS solution

,((BOXl), climbonbox (BOXl), climboffbox (BOXl ۲goto}

,(pushto (BOXI, LIGHTSWlTCHl), climbonbox (BOXl

{(turnonlight (LIGHTSWITCHI

۲. Push three boxes together

(۳, BOX۲) □ NEXTTO (BOX۲Goal wff: NEXTTO (BOXI, BOX

:STRIPS solution

), pushto۳(BOX ۲, BOXl), goto۲), pushto (BOX۲(BOX ۲goto}

{(۲, BOX۳BOX)

۳. Go to a location in another room

(Goal wff: ATROBOT (f

:STRIPS solution

,(, ROOMS۱, ROOM۱(DOORl), gothrudoor (DOOR ۲goto}

,(٤, ROOMS, ROOM٤), gothrudoor (DOOR٤(DOOR ۲goto

{(gotol (f

بنابراین روش پیشنهاد شده به صورت موثری قادر به مدل کردن جهان مساله و بررسی آن است. این یکی از مثالهای خوب استفاده عملی از منطق مرتبه اول برای مدل کردن جهان است. اما هنوز دوسوال بدون جواب مانده‌اند:

چگونه میتوان در مدل یک جهان خاص درستی یک جمله را چک کرد؟
چگونه میتوان با شروع از مدل مبدا، مدل مقصد جهان را یافت؟

جواب سوال دوم یک مساله‌ی کلاسیک هوش مصنوعی است. در حقیقت مدلهای مختلف ما از جهان تشکیل فضای مساله را میدهند، که با استفاده از عملگرها میتوان از یک مدل به مدلی دیگر رفت. به عبارت دیگر میتوان مدلهای مختلف جهان را به عنوان راسهای یک گراف جهت‌دار در نظر گرفت که یالهای آن عملگرهای قابل اعمال به هر مدل هستند. میخواهیم در این گراف با شروع از یک راس و جستجو در گراف از طریق پیمودن یالهای آن به یک راس مقصد با ویژگیهای خاص برسیم. ساده‌ترین راه برای این کار انجام یک جستجوی اول عمق۱۵ بر روی گراف با شروع از راس آغازین است، جستجو در گراف تا وقتی ادامه پیدا میکند که یک راس پیدا شود که هدف مورد نظر را ارضا کند. اما این روش نمیتواند به عنوان یک روش کاربردی مورد استفاده قرار گیرد، چون حتی در ساده‌ترین مسایل اندازه‌ی گراف به قدری بزرگ است که نمیتوان امیدوار بود در زمان معقولی به جواب رسید. پس به طریقی باید بین راههایی که برای پیمایش داریم اولویت بندی کنیم. این اولویت بندی معمولا به کمک یک تابع هیوریستیک۱٦ انجام میشود. این تابع به هر راس گراف عددی نسبت میدهد که نشان دهنده‌ی میزان تقریبی فاصله آن راس تا راس مقصد است. برای پیمودن گراف در هر مرحله راسی را انتخاب میکنیم که فاصله تقریبی کمتری تا مقصد داشته باشد. پس مساله تبدیل به یافتن یک تابع هیوریستیک خوب میشود. اگر این تابع به درستی انتخاب شود، ما را مستقیما به راس هدف میبرد و اگر درست انتخاب نشود ممکن است باعث شود که هیچوقت راس هدف را پیدا نکنیم. با تمام این اوصاف به نظر میرسد که انتخاب هوشمندانه یک تابع هیوریستیک مساله را کاملا حل میکند اما قضیه‌ای به نام no-fee-lunch وجود دارد که میگوید هیچ شیوه‌ای برای جستجو در یک گراف وجود ندارد که در تمام موارد بهتر از جستجوی اول عمق (به نمایندگی از یک جستجوی کلی و بدون هوش) کار کند. این مطلب در شکل زیر نشان داده شده است:

شکل به نقل از ویکی‌پدیا۱۷

با توجه به این شکل انتخاب یک تابع هیوریستیک خوب در بیشتر موارد به ما در یافتن جواب کمک میکند اما در مواردی هم استفاده از این تابع به ضرر ما است.

برای جستجو در گراف مربوط به مدلها یک تابع هیوریستیک در[۷۱STR] معرفی شده است که به نظر میرسد برای انجام این کار تابع قابل قبولی است. برای بدست آوردن اطلاعات بیشتر در این مورد به [۷۱STR] مراجعه کنید.

اما جواب به سوال اول همان مساله کلاسیک منطق محاسباتی است، در انجام کارهای سطح بالا (مثلا برنامه‌ریزی حرکت روبات) معمولا در این قسمت کار زیادی نمیشود و ترجیحا از یک برنامه کامپیوتری (مانند Otter) که این کار را به صورت عمومی انجام میدهد استفاده میکنند.

نکته‌ی قابل توجهی که در این مورد وجود دارد این است که ایده‌ی استفاده از مدلهای منطقی جهان در هوش مصنوعی برای اولین بار در سال ۷۱ در [۷۱STR] مطرح شد. در این مقاله یک سیستم به نام STRIPS برای طراحی حرکت روباتها پیشنهاد شده است و همچنین اشاره شده است که کلمه‌ی STRIP به اختصار از STanford Research Institute Problem Solver گرفته شده، اما هم اکنون بعد از گذشت بیش از ۳۰ سال STRIPS به رده‌ای کلی از الگوریتمها اطلاق میشود که با استفاده از همین ابزار جهان را مدل میکنند. این ایده بعدها توسط دیگر محققان پیگیری شد، به طور مثال در سال ۹۷ یک نسخه به شدت بهبود یافته از الگوریتم به نام GraphPlan ارایه شد. اطلاعات بیشتر را میتوانید در [GraphPlan] بیابید.

یک مساله قابل توجه در مورد STRIPS قابلیت یادگیری خودکار۱۸ در آن است. یادگیری یکی از مهمترین مسایلی است که در هوش مصنوعی با آن مواجهیم. یادگیری به این معنا که وقتی ماشین بوسیله تفکر موفق به حل مساله ای شد، بار بعد با دیدن آن مساله دوباره اقدام به حل آن نکند و از حل قبلی خود استفاده کند، همچنین بتواند از راه حل یک مساله برای حل مسایل بزرگتر استفاده کند. یک جمله مشهور در این مورد وجود دارد که میگوید: "یک ماشین هوشمند واقعی همیشه میتواند مسایلی را که کمی سخت‌تر از مسایل قبلی حل شده هستند را حل کند." روش پیشنهاد شده برای یادگیری در STRIPS به این شکل است که با حل کردن یک مساله، یک عملگر جدید به مجموعه عملگرها اضافه شود که نشان‌دهنده‌ی آن مساله و راه حل آن باشد. این روش بسیار ساده اما به شدت کارا است. تنها مشکلی که وجود دارد این است که تعداد عملگرها بعد از مدتی بیش از اندازه زیاد خواهد شد و این باعث کند شدن سیستم میشود. برای رفع این مشکل باید اقدام به حذف عملگرهای اضافی و همچنین انتخاب عملگرهای مناسب هنگام جستجو نمود که این کار خود یک مساله هوش‌مصنوعی است.

٤- آینده منطق محاسباتی

در صورت دست یافتن به کارایی بالا در قسمتهای پایه‌ی منطق محاسباتی قادر به انجام کارهایی خواهیم بود که تا امروز امکان ناپذیر بوده اند.

مهمترین این کارها اثبات خودکار قضایای ریاضی و نوشتن خودکار برنامه های کامپیوتری هستند. گرچه امروزه هم تحقیقاتی در این مورد انجام شده است اما موفقیت چندانی حاصل نشده است.

در مورد اثبات خودکار قضایای ریاضی سه دسته از برنامه ها وجود دارند. دسته اول برنامه هایی هستند که اثبات یک قضیه را گرفته و درستی آن را چک میکنند. در مورد ساختن برنامه ای برای این کار مشکل چندانی وجود ندارد. دسته دوم برنامه هایی هستند که کل عملیات اثبات را به صورت خودکار انجام میدهند. در مورد این دسته با مشکلات بسیاری مواجه هستیم. ورودی برنامه ای از این دست، تعدادی گزاره به عنوان فرض و یک گزاره به عنوان حکم است و میخواهیم که برنامه درستی یا نادرستی حکم را (به همراه اثبات آن) به ما ارایه کند. به سادگی میتوان نشان داد که این مساله معادل با مساله‌ی SAT است. بنابراین ما در اینجا با یک مساله NP-Complete و PSpace مواجه هستیم. پس یه صورت تئوری تا وقتی که مساله P=NP حل نشده است امید چندانی به حل این مساله نمیتوانیم داشته باشیم. اما همانند دیگر مسایل مشابه امروزه تکنیکهای هوش مصنوعی بسیاری برای حل موضعی این مشکل در حال شکل گیری هستند. دسته سوم برنامه هایی هستند که کار اثبات را با راهنمایی یک انسان انجام میدهند. مشکلات در مورد این دسته بسیار کمتر هستند اما

چون برقراری ارتباط بین یک انسان و کامپیوتر در مورد این مساله مشکل است استقبال کمی از این دسته از برنامه ها شده است.

٤- نتیجه‌گیری

در این مقاله بعضی جنبه های منطق محاسباتی مورد بررسی قرار گرفت. به نظر میرسد که در این مورد باید منتظر یک جهش عظیم باشیم چون الگوریتمهای فعلی از کارایی خوبی برخوردار نیستند. به صورت کلی از آنجایی که این یکک مساله NP-Complete است نمیتوانیم به حل کامل آن امیدوار باشیم اما نشانه های خوبی نیز دیده میشود. مهمترین نشانه ای که باید به آن توجه کنیم مغز انسان است. مغز انسان با کارایی نسبتا خوبی قادر به حل مسایل است. گرچه حجم داده ها و درجه‌ی پردازش موازی در مغز بسیار زیاد است، اما به نظر نمیرسد که تنها این دو عامل باعث این کارایی خوب مغز باشند. اگرچه حجم حافظه طولانی مدت مغز بسیار زیاد است، اما کافی است دقت کنیم که حافظه کوتاه مدت انسان قادر به نگه داری ٥ تا ٧ مورد است. همچنین به نظر نمیرسد که مغز برای حل مسایل خود واقعا از یک موازیگری١٩ بینهایت استفاده کند. این مطلب و موارد مشابه محققان را علاقه مند به درک و کشف راهکارهای مغز برای حل مسایل خود نموده است.

یکی از ملزومات یک سیستم اثبات گر خودکار قضیه((٢٠ATP این است که دارای حافظه ای باشد که قضیه هایی را که تاکنون اثبات کرده یاد بگیرد. اگر چنین نباشد باید هر قضیه جدید را با شروع از مجموعه اصول و از صفر اثبات کند و این در مورد قضیه های سطح بالا باعث میشود که هیچ وقت موفق به اثبات آنها نشود.

اما چرا این اتفاق می افتد؟ چرا بعضی قضیه ها سطح بالا و بعضی سطح پایین هستند؟ چرا این سطح‌بندی در همه شاخه هایی که به نوعی با تفکر انسان مرتب هستند، اتفاق می افتد؟

جواب این سوال دقیقا با ساختار داخلی مغز برای تفکر مرتبط است. مغز اقدام به نگه داری اشیا به طریقه سلسله مراتبی می کند. مغز به شدت علاقه دارد که چیزهای مختلف را طبقه بندی کرده و هر چیز را در طبقه مخصوص به آن قرار دهد. مغز اطلاعات را به صورت خام نگه نمیدارد. یادگیری در مغز به شدت با یادگیری در کامپیوتر فرق دارد. در کامپیوتر شما هر داده ای را میتوانید با کد کردن در حافظه ذخیره کنید. اما هر داده ای را نمیتوان به سادگی در مغز ذخیره کرد. مغز بیشتر علا قه دارد که چیزها را از روی شباهتشان یاد بگیرد.

برای مثال این دو شکل را در نظر بگیرید:

هرکسی با دیدن این عکس بی درنگ خواهد گفت دو مثلث داریم که یکی را چرخانده ایم. اما در واقع فهمیدن اینکه این دو مثلث مثل هم هستند و یکی از آنها چرخیده است اصلا کار ساده ای نیست. در واقع نمیتوانیم به آسانی برنامه ای بنویسیم که این را تشخیص دهد. در موزد تصویر باز وضعیت بهتری داریم چون میتوانیم آنرا بوسیله یک شبکه از پیکسلها در کامپیوتر به خوبی نمایش دهیم. این وضعیت در مورد اطلاعات دیگر مثلا صدا، یا حتی علم! بغرنج تر میشود. توجه کنید که سر منشا علم ریاضی مجرد٢١ کردن جهان بوده است. اما این مجرد کردن یعنی چه؟ اینها همه سوالاتی است که برای تولید ماشینی علم باید به آنها جواب داده شود.

درک روشهای مغز برای تفکر از دو جهت حایز اهمیت است. اول اینکه الگوریتمهای جدیدی برای تفکر در اختیار ما قرار میدهد و دوم اینکه علم به عنوان موضوعی که بوسیله مغز به عنوان یک ماشین تولید شده است ساختار پذیرفته است. یعنی اینکه مغز به عنوان یک ماشین یک اسکلت بندی خاص را به علم به عنوان خروجی ماشین تحمیل میکند.

به عنوان مثال آیا تا حالا فکر کرده اید که چرا علم امروزه به شکل یک درخت است؟ چرا طور دیگری نیست؟ آیا دقت کرده اید که چرا بیشتر قضیه هایی که اثبات آنها را میبینیم طول کوتاهی حد اکثر در حدود ٥-٧ قسمت دارند؟ آیا دقت کرده اید که بعضی از قضایا را به عنوان قضایای اساسی میشناسیم و بعضی از قضایا را به عنوان قضایای فرعی؟

دلیل تمام این حرفها ساختار مغز برای تفکر است. این نشان میدهد که مغز همه آدمها دارای زیرساختهای مشترکی برای فکر کردن هستند. حافظه کاری انسان، حافظه کوتاه مدت اوست. میدانیم که حجم این حافظه بین ۵ تا ۷ مورد است. این به این معنا است که وقتی داریم درمورد چیزی فکر میکنیم نمیتوانیم همزمان بیش از ۷ مورد مختلف را در این حافظه نگه داری کنیم. بدین ترتیب ارایه کردن اثبات های با طول زیاد بسیار مشکل است. اگر نتوانیم چیزی را در چند مرحله کوتاه اثبات کنیم، عملا ممکن است هیچ وقت نتوانیم آنرا اثبات کنیم. مغز این مشکل را با استفاده از تعریف مفاهیم جدید حل میکند. مغز به شدت علاقه دارد که چند چیز را کنار هم بگذارد و به این طریف از آنها یک مفهوم جدید بسازد. به این ترتیب همه آن چیزها فقط یک جا را اشغال میکنند.

چرا بعضی از قضایا اساسی محسوب میشوند؟ یکی از دلایل آن میتواند این باشد که این قضیه ها قوی هستند. قضیه‌ی قوی یعنی چه؟ آیا هر قضیه قوی، اساسی است؟ مطمئنا خیر. قضیه ای اساسی است که زیاد از آن استفاده شده باشد. به این ترتیب یکی دیگر از خصوصیتهای مغز آشکار میشود. مغز علاقه دارد که اگر از چیزی زیاد استفاده شد، باز هم از همان استفاده کند. به این ترتیب گرچه حافظه بلند مدت ما حجم نامحدودی دارد اما ما عملا از تمام آن استفاده نمیکنیم. ما علاقه داریم که با قسمتهای آشناتر و اساسی تر آنها کار کنیم.

اما تعریف اساسی بودن به این سادگی ها نیست. به طور مثال وقتی در زمینه گراف یک مساله را حل میکنیم، ابتدا به قضایای اساسی مرتبط با نظریه گراف مراجعه میکنیم. بسیار بعید است که به یک قضیه اساسی در مورد نظریه اعداد مراجعه کنیم. پس مغز همه چیز، حتی قضایای اساسی را هم دسته بندی میکند. این دسته بندی چگونه امکان پذیر است. چگونه میتوانیم الگوریتمی برای طبقه بندی قضایا به عنوان تعدادی جمله منطقی پیدا کنیم؟

مغز این کار را از طریق کشف شباهتها میان اشیا انجام میدهد. به این معنی که با گرفتن یک شی جدید، آنرا به صورت مجرد (Abstract) در نظر میگیرد و بررسی میکند که در این حالت این شی با کدامیک از مفاهیم مجرد قبلی همخوانی دارد. و البته این کار در حرف ساده است و در عمل ....

این مقاله دو ضمیمه دارد. در ضمیمه اول به مفهوم Abstraction و کارهایی که در مورد آن انجام پذیرفته است میپردازیم و در ضمیمه دوم به بررسی Connectionism به عنوان یکی از بهترین روشهای علمی مطالعه مغز میپردازیم.

ضمیمه ۱- Abstraction

abstraction به عنوان یک روش قوی برای اثبات قضایا پیشنهاد شده است. به طور معمول از یک ساختار abstract برای اثبات کمک گرفته میشود تا اثبات واقعی ساخته شود.

با دقت در نحوه فکر کردن انسان پی میبریم که انسان هم هنگام اثبات کردن به نحو خوبی از abstraction کمک میگیرد. نحوه اثبات کردن انسان به این شکل است که ابتدا یک ساختار(و یا روش) کلی را در نظر میگیرد (مثلا استقرا) سپس سعی میکند که مساله را با توجه به این ساختار حل کند. او سپس مساله را ریزتر بررسی میکند و در صورتی که ساختار درستی انتخاب کرده باشد، میتواند به یک نمای کلی از اثبات دست یابد. در این مرحله است که حس میکند به شهودی کلی از اثبات مورد نظر رسیده است. و در مرحله آخر، حفره‌های کوچک باقیمانده در اثبات را پر میکند. حال مطلب فوق را به صورت دقیقتر بررسی میکنیم. هر اثبات معادل با یک درخت اثبات است. به این شک که برگهای درخت فرضهای مساله هستند و هر راس داخلی با استفاده از فرزندانش و استفاده از یکی از قواعد استنتاج بدست آمده است. ریشه‌ی درخت حکم مساله است. در نظر گرفتن یک روش کلی برای اثبات در حقیقت به معنای تحمیل یک ساختار به درخت است. به مثال زیر توجه کنید:

این درخت اثبات مربوط به استقرا است:

درخت استقرا

پس اگر استقرا را به عنوان روش اثبات انتخاب کنیم، ساختار درخت اثبات چنین شکلی خواهد داشت:

یک درخت اثبات کامل با استقرا

در مرحله بعد این درخت اثبات کاملتر میشود و یا پس از مقداری تلاش به این نتیجه میرسیم که استفاده از این روش کلی برای اثبات مناسب نیست و روش دیگری را استفاده میکنیم.

نظر برخی از افراد این است که abstraction (به معنای کلی و عمومی آن) مهمترین روش مغز برای حل مسایل است. دقت کنید که هنگام حل یک مساله ما با یک مساله p-space و NP-Complete مواجهیم. از طرف دیگر حجم داده‌های پیش فرض مغز در حد غیر قابل تصوری بزرگ است. دقت کنید که هیچ چیزی از حافظه بلند مدت انسان پاک نمیشود. هنگامی که یک انسان در حال حل مساله است، تمام مسایل و قضایایی که او قبلا دیده است شانس این را دارند که در حل این مساله هم شرکت کنند. بدین ترتیب برخی افراد عقیده دارند که موازی بودن پردازش در مغز به هیچوجه توجیه کننده سرعت بالای مغز نسبت به کامپیوتر نیست. این افراد اعتقاد دارند که در مغز راهکارها و الگوریتمهای خاصی برای سازماندهی و استفاده به موقع از این حجم بزرگ اطلاعات وجود دارد که باید نسبت به کشف آنها اقدام شود. مثلا هنگامی که فردی در حال حل یک مساله در زمینه‌ی نظریه گراف است، بیشتر مطالب مربوط به گراف به ذهن او میرسند تا مطالب مربوط به نظریه اعداد. این یک مثال خیلی جامع از انتخاب مغز است.

منظور از abstraction در معنای کلی کشف مشابه بودن چیزها و روالهای مختلف و توانایی استخراج این مشابهتها به عنوان یک شی مستقل است. به طور مثال یک انسان عادی بعد از اینکه چند مساله که راه حل مشابهی دارند را دید، میتواند آن راه حل را به صورت عمومی استخراج کرده و در حل مسایل بعدی خود از آن راه حل استفاده کند. به این ترتیب مغز مساله را در دو قسمت حل میکند. در یک قسمت سعی میکند مسایل مشابهی را که قبلا حل کرده به خاطر بیاورد و بدین ترتیب راه حل کلی را حدس بزند. در این مرحله قسمتی از درخت اثبات تشکیل شده است. سپس سعی در پر کردن این درخت میکند. در این مرحله تمام قضایایی که در حافظه ذخیره شده‌اند شانس شرکت در اثبات را دارند اما مغز اولویت را به قضایایی میدهد که ارتباط بیشتری با موضوع داشته باشند، این ارتباط به شیوه های مختلف مثل آموزش، استفاده در مسایل مشابه و ... در مغز ذخیره میشود.

علی‌رغم اهمیت abstraction در کارهای مغز، یک جستجوی ساده در اینترنت، نشان میدهد که تا کنون آنطور که باید و شاید در مورد این مساله پژوهش صورت نگرفته است. در حقیقت پیدا کردن مطالب علوم کامپیوتری مرتبط با این موضوع کار ساده‌ای نیست و در نگاه اول به نظر میرسد که هیچ گونه کاری در این زمینه صورت نگرفته است. یکی از دلایل این امر این است که abstraction موضوع خوش تعریفی نیست و ما نمیدانیم abstraction دقیقا چیست؟ از طرف دیگر abstraction به خودی خود کاربردی ندارد، بلکه کاربرد آن در دیگر زمینه‌هاست، به همین علت خیلی کم به این مساله به صورت کلی آن پرداخته شده است.

در بین معدود کارهای انجام شده [a،GW۹۰]b۹۰ و [GW۹۰c] به ارایه روش قابل قبولی برای استفاده از abstraction در حل مسایل میپردازند. همچنین [CT۹۹] با استفاده از تئوری مطرح شده در سه منبع اول به ارایه روشی برای اثبات قضایا میپردازد.

ضمیمه ۲- Connectionism۲۲

Connectionism نام حرکتی در علوم شناختی ۲۳ است که میخواهد تواناییهایی هوشی انسان را از طریق شبکه عصبی۲٤ توضیح دهد. شبکه های عصبی مدلهای ساده شده ای از مغز انسان هستند. آنها از تعداد

بسیار زیادی نورون۲۵ (به عنوان کوچکترین واحد پردازش در مغز) و همچنین اتصالهای بین این نورونها که برای رد و بدل کردن داده استفاده میشود تشکیل شده اند ۲۶. وزن اتصالها نشاندهنده قدرت ارتباط بین دو نورون است. آزمایشها بر روی این مدل از شبکه عصبی نشان داده اند که این مدل توانایی انجام کارهایی از قبیل تشخیص چهره۲۷، خواندن متون و تشخیص نکات ساده گرامری را دارد.

درباره‌ی شبکه‌های عصبی

یک شبکه عصبی شامل تعداد زیادی از نورونهاست که بر اساس یک الگوی اتصال (connection) به هم پیوند یافته اند. نورونها معمولا به سه دسته تقسیم میشوند: نورونهای ورودی، که ورودی را برای پردازش دریافت میکنند. نورونهای خروجی که حاوی نتیجه پردازش هستند و نورونهای میانی که به نام نورونهای پنهان۲۸ نامیده میشوند.

این یک مثال ساده از شبکه عصبی است:

یک شبکه عصبی ساده

نورونهای ورودی از طریق منابع خارجی فعال میشوند. هر نورون ورودی پس از فعال شدن مقدار فعال سازی خود را به نورونهای پنهانی که به آنها متصل است میفرستد. هرکدام از نورونهای پنهان با توجه به مقادیری که از نورونهای همسایه خود دریافت میکند، فعال بودن خود را تعیین میکند و اگر فعال بشود مقدار فعال سازی خود را به نورونهای خروجی یا به لایه دیگری از نورونهای پنهان میفرستد و به طور پیوسته و همزمان تمام نورونها به کار خود ادامه میدهد و به این ترتیب سیگنالهای تولید شده توسط نورونهای ورودی در تمام شبکه پخش میشوند تا وضعیت فعال بودن نورونهای خروجی مشخص شود.

الگوی فعال شدن یک شبکه توسط وزن بین نورونهای آن مشخص میشود. وزن میتواند مثبت یا منفی باشد. یک وزن منفی نشاندهنده عدم تمایل نورون مقصد نسبت به فعال شدن در صورت فعال بودن نورون مبدا است. فعال بودن هر نورون توسط یک تابع ساده فعال سازی۲۹ انجام میشود. توابع فعال سازی مختلفی وجود دارند اما همگی بر یک پایه بنا شده اند. تابع تمام ورودیهای نورون (اگر نورون مبدا فعال باشد اندازه ورودی آن به نورون مقصد به اندازه وزن بین آن دو است.) را با هم جمع میکند. حاصلجمع معمولا کمی تغییر می یابد. مثلا به عددی بین ۰ و ۱ تبدیل میشود و سپس اگر این مجموع از عددی خاص بیشتر بود، نورون فعال میشود.

connectionistها ادعا میکنند که عملکرد ذهنی انسان توسط یک شبکه عصبی با مشخصات بالا قابل توصیف است. از جایی که در مدل بالا تمام نورونها به یک شکل عمل میکنند، عملکرد کلی شبکه توسط وزن ها و الگوی اتصال نورونها تعیین میشود.

شبکه‌ای که در بالا معرفی شد معروف به شبکه feed forward است. در این شبکه ورودی از لایه‌ی ورودی دریافت شده و سپس از یک یا چند لایه پنهان میگذرد تا به لایه خروجی برسد. هیچ یالی درون یک لایه یا از لایه جلوتر به لایه عقب تر وجود ندارد. مدلهای واقعی تر از مغز بسیار پیچیده تر هستند. آنها شامل تعداد بسیار زیادی از لایه های پنهان، و اتصالهای برگرداننده۳۰ هستند که سیگنال را از لایه های جلویی به لایه های عقبی میفرستد. این برگرداندن برای توصیف کردن خصوصیاتی مثل حافظه کوتاه مدت۳۱ ضروری هستند. connectionistها معمولا سعی میکنند از اتصالهای برگرداننده پرهیز کنند چون تحلیل و تربیت این گونه شبکه ها بسیلر مشکل است.

شبکه عصبی و ماشین تورینگ

اینکه شبکه‌های عصبی از چه قدرت محاسباتی برخوردار هستند جای بحث بسیار دارد. با یک استدلال ساده میتوان نشان داد که شبکه عصبی حداقل قدرتی برابر با ماشین تورینگ دارد.

شبکه عصبی زیر را در نظر بگیرید.

شبکه عصبی معادل با XOR

در این شبکه هر نورون به شرطی فعال میشود که مجموع ورودی های آن بیشتر از ۰,۵ باشند. با کمی محاسبه میتوان نشان داد که این شبگه معادل با عملگر منطقی XOR است. از آنجا که عملگر XOR کامل است، (تمام عملگرهای منطقی را میتوان با ترکیب تعدادی XOR ساخت)، پس با استفاده از این شبکه هر مدار الکترونیکی که با عملگرهای منطقی ساخته شده باشد را میتوان شبیه سازی کرد. از طرف دیگر میدانیم که هر ماشین تورینگ قابل شبیه سازی بوسیله مدارهای الکترونیکی است. (مدارهایی از قبیل کامپیوترهای امروزی) بنابراین هر ماشین تورینگ را میتوان توسط یک شبکه عصبی شبیه سازی کرد. پس شبکه های عصبی قدرتی بزرگتر یا مساوی ماشینهای تورینگ دارند.

در حالت کلی میتوان ثابت کرد که ماشین تورینگ و شبکه عصبی معادلند (به صورت بازگشتی). بنابراین اگر ادعای connectionistها درست باشد مغز را میتوان معادل با یک ماشین تورینگ یا یک تابع محاسبه پذیر (با تغییرات جزیی) در نظر گرفت و این خود سرآغاز مسایل بزرگ فلسفی مثل جبر و اختیار و ... است. دقت کنید که اگر چنین حرفی به صورت مطلق درست باشد از یک منظر انسان موجودی جبری خواهد بود. این بحث بسیار گسترده است و در اینجا بیش از این به آن نمیپردازیم.

نمونه‌هایی از شبکه عصبی

Connectionistها فعالیت زیادی برای نشان دادن قدرت شبکه عصبی در انجام فعالیتهای ذهنی کرده اند. آنها به نمونه های موفقی دست یافته اند که فرض اولیه آنها را به شدت تقویت میکند.

در ۱۹۸۷، Senjowski و Rosenberg اقدام به طراحی شبکه ای به نام NETTalk کردند که میتوانست از روی متون انگلیسی بخواند. (به صورت صوتی). NETTalk توسط حجم بزرگی از داده ها که شامل متن و فونتیک مربوط به آن متن بودند آموزش داده شد. صدای خروجی NETTalk در مراحل مختلف آموزش آن جالب است. در ابتدا خروجی یک نویز تصادفی است بعد از مدتی صداها کمی مفهوم تر میشوند. بعد از آن مثل این است که کسی به صورت مقطع در حال تلفظ کلمات است و در آخرین مرحله صدای خروجی شفاف میشود و NETTalk کار تلفظ متن داده شده (حتی اگر متن برایش جدید باشد) را به خوبی انجام میدهد. برای مطالعه بیشتر به [Sejno۸۷] مراجعه کنید.

یک نمونه موفق دیگر از شبکه عصبی در ۱۹۸٦ توسط Rumelhart و McClelland برای یادگرفتن و حدس زدن زمان گذشته افعال در زبان انگلیسی ساخته شد. این شبکه به این دلیل جالب است که تقریبا همه افعال در انگلیسی با افزودن ed به آخر فعل به زمان گذشته برده میشوند ولی خیلی از افعال متداول بی قاعده هستند(is, was; come, came; go, went) شبکه ابتدا توسط یک مجموعه بزرگ از افعال بی قاعده تربیت شد و سپس توسط ٤٦۰ فعل باقاعده. شبکه زمان گذشته آن ٤٦۰ فعل ا پس از ۲۰۰ مرتبه تربیت شدن یاد گرفت و به خوبی یاد گرفت فعلهایی را که تاکنون ندیده بود به زمان گذشته تبدیل کند. این شبکه همچنین قواعدی که درافعای بی قاعده وجود دارند را نیز کشف کرد(send, sent; build, built; blow, blew; fly, flew). اشتباهاتی که شبکه در طول میکرد نیز قابل توجه است. به طور مثال شبکه در طول یادگیری زمان گذشته break را broked حدس زد. این مشکل با تربیت بیشتر حل شد. این نکته جالب است که کودکان هم هنگام یادگیری زمان گذشته افعال این اشتباهات را میکنند.

ابعاد فلسفی Connectionism

connectionism توجه فلاسفه را هم به خود جلب کرده است، چون connectionism یک تئوری جدید در برابر تئوری کلاسیک فلسفه ذهن است: دیدگاه سابقه داری که میگوید ذهن چیزی معادل کامپیوترهای امروزی است که کار آن پردازش یک زبان سمبولیک ۳۲ است. فلاسفه کلاسیک ادعا میکنند که اطلاعات در مغز به صورت رشته ای و سمبولیک نگه داری میشوند، در مقابل connectionistادعا میکنند که اطلاعات در مغز به صورت خیر سمبولیک و در وزنهای بین نورونها ذخیره میشوند. فلاسفه کلاسیک ادعا میکنند که مغز یک ابزار پردازش سمبولیک است که کار آن پردازش رشته هاست، در مقابل connectionistها ادعا میکنند که کار پردازش در مغز توسط شبکه ی پیچیده ای از نورونها انجام میشود.

به صورت کلی در مورد connectionism دو جهت گیری وجود دارد. جهت گیری اول قسمتی از کارهای مغز را فرامادی میداند و جهت گیری دیگر اعتقاد دارد که مغز از ساختاری غیر از این برخوردار است. به هر حال connectionism کم کم در حال پیدا کردن مکان خود در روانشناسی است. connectionistها موفق به ارایه ایده های جدیدی درمورد مسایل کهنه فلسفی شده اند. برای اطلاعات بیشتر در این زمینه به [StaEncy] مراجعه کنید. همچنین میتوانید از منبع جالب اما کمی قدیمی [Bra۸٤] استفاده کنید. برای مطالعه دقیق در مورد ارتباط منطق محاسباتی با connectionism به [Ste۰۳] فصل آخر مراجعه کنید.

تعادل نش به عنوان قرارداد اجتماعی
این متن به عنوان بخشی از امتحان درس خرد ۲ (دکتر فرهاد نیلی) نوشته شده است. (فایل pdf)

درآمد
پدید آمدن قواعد و قراردادهای اجتماعی، به معنی آنچه "عرف" را تشکیل می دهد، در چارچوب تعامل رفتاری افراد یک اجتماع قابل بررسی است. چنانچه به این رفتارهای تعاملی به عنوان یک "بازی" نگاه کنیم، بررسی قراردادهای اجتماعی از دیدگاه نظریه بازی، مورد توجه قرار می گیرد.
در یک رفتار تکراری اجتماعی، افراد اجتماع، یا همان بازیکن ها، در طی دوره های مختلف با مشاهده نتایج اعمال خود، به شناسایی بازی مورد نظر می پردازند. در واقع در بسیاری موارد، ممکن است خود سیستم(بازی) در ابتدا توسط بازیکن ها شناخته شده نباشد. به مرور و در طی دوره های مختلفی که بازی تکرار می شود، شناسایی بازی توسط بازیکن ها انجام می شود. به بیان ریاضی، ماتریس پرداخت ها، چه از نظر ابعاد(ماهیت و تعداد استراتژی های هر فرد) و چه از نظر درایه ها(نتیجه بازی در استراتژی های مختلف) در طی دوره های مختلف شناسایی می شود.
در برخی موارد، شناسایی بازی در طی زمان منجر به تکرار یک یا چند استراتژی مشخص از میان همه استراتژی های قابل انتخاب می شود. تکرار این نتیجه، و پایداری آن، بستگی به شرایط آن نقطه مشخص دارد.

تعاریف
تعریف ۱. تعادل نش در یک بازی، یک مجموعه استراتژی برای هر یک از بازیکن ها را تعادل نش می نامیم در صورتی که تغییر استراتژی توسط هر یک از بازیکن ها در حالتی که بقیه همان استراتژی قبل را دنبال کنند، باعث افزایش پرداخت به آن بازیکن نشود.
در واقع در یک تعادل نش، تغییر استراتژی هر بازیکن به تنهایی، نمی تواند وضع او را بهتر کند.
تعریف ۲. قانون اجتماعی محدود کردن بازیکنان به یک زیر مجموعه از استراتژی ها در یک بازی، که باعث به وجود آمدن یک زیر بازی می شود، یک قانون اجتماعی نامیده می شود. [۲]
تعریف ۳. قرارداد اجتماعی قانون اجتماعی ای که تنها یک استراتژی را در اختیار بازیکن قرار دهد، یک قرارداد (یا عرف) اجتماعی نامیده می شود. [۲]

پدید آمدن یک قرارداد اجتماعی
بررسی قوانین و قراردادهای اجتماعی، جدا از تعریف، مستلزم ملاحظات دیگری، مانند نحوه دستیابی به قرارداد، پایداری یا ناپایداری، و عقلانی یا غیر عقلانی بودن آن هم هست؛ برای روشن شدن موضوع، ابتدا بهتر است به نحوه پدید آمدن یک قرارداد اجتماعی نگاه دقیق تری داشته باشیم.
یک بازی دو نفره را تصور کنید که در آن، هر یک از بازیکن ها، از یک گروه(اجتماع) به طور تصادفی انتخاب می شوند. همچنین فرض کنید این فرآیند به طور مداوم تکرار می شود. در این صورت در طی زمان و به مرور، ممکن است به انتخاب یک مجموعه استراتژی ثابت توسط بازیکن ها منجر شود.

چنین مجموعه استراتژی ای، دارای ویژگی های مشخصی خواهد بود. اگر این نقطه را یک تعادل اجتماعی بنامیم، طبق تعریف، یک قرارداد اجتماعی نیز خواهد بود. اما تفاوت این تعادل با یک قرارداد اجتماعی، در حالت کلی، در نحوه ایجاد محدودیت است.
در یک تعادل اجتماعی، تکرار بازی و خود بازی، محدودیت موجود را القا می کند، در حالی که یک قرارداد اجتماعی، در حالت کلی ممکن است از طریق دیگری، مثلا یک قدرت مرکزی(Central Authority)، این محدودیت را اعمال کند. در این حالت ، ممکن است هر یک از بازیکن ها، قادر باشد با انتخاب یکی از استراتژی های خارج از قرارداد اجتماعی، به وضعیت بهتری دست یابد، که البته می تواند جریمه ای، خارج از تعریف بازی (ماتریس پرداخت ها) به دنبال داشته باشد.

تعادل نش به عنوان قرارداد اجتماعی
در حالتی که بازیکن ها در چارچوب بازی به یک نقطه تعادل رسیده باشند، شاهد نتایج متفاوتی نسبت به حالت کلی هستیم؛ در این حالت، با فرض اینکه بازیکن ها درباره نتایج دوره های قبل دارای حافظه باشند، در صورتی که بعد از تکرار چندین دوره، به یک نقطه تعادل برسیم، این تعادل باید یک تعادل نش باشد. در غیر این صورت، حداقل یکی از بازیکنان استراتژی ای را انتخاب کرده است که بهترین نتیجه را با توجه به انتخابهای بقیه به او نمی دهد. در واقع اگر بازیکن ها در ابتدا به صورت تصادفی هم استراتژی را انتخاب کنند، فقط در صورتی که نتایج دوره های قبل را در نظر بگیرند و در دوره های بعد با توجه به آن(و احتمالا باز هم با چاشنی تصادف در مورد استراتژی ها و نتایج آزموده نشده) انتخاب استراتژی بکنند، در صورتی که بازی بعد از چند دوره به تعادل برسد، حتما این تعادل یک تعادل نش خواهد بود. در غیر این صورت یکی از بازیکن ها، بهترین واکنش (Best Response) را در مقابل استراتژی بقیه انتخاب نکرده است.

ویژگی های تعادل نش به عنوان قرارداد اجتماعی
۱. خود اصلاحی
در یک تعادل نش که به عنوان قرارداد اجتماعی پذیرفته شده است، در صورتی که هر یک از بازیکن ها، به تنهایی از تعادل منحرف شود، با کاهش پرداخت مواجه می شود. به این ترتیب، سیستم می تواند به طور پایدار در این تعادل باقی بماند.
نکته قابل توجه در مقایسه با حالتی که یک قرارداد اجتماعی توسط یک قدرت مرکزی وضع می شود این است که قراردادی که در یک تعادل نش شکل می گیرد، خاصیت خود اصلاحی دارد، یعنی در صورتی که هر یک از بازیکن ها از تعادل منحرف شود، شرایط بازی(ماتریس پرداخت ها) او را جریمه خواهد کرد.
با این ملاحظه، در صورتی که انتخاب و وضع یک قرارداد اجتماعی توسط یک قدرت مرکزی با در نظر گرفتن این ویژگی انجام نشود، یعنی قرارداد در یک تعادل نش وضع نشده باشد، ماهیت بازی، بازیکن ها را به انحراف از قرارداد سوق می دهد.
۲. شرایط تغییر قرارداد
از سوی دیگر، ویژگی دیگر تعادل نش این است که استراتژی هر بازیکن، با مفروض بودن استراتژی بقیه، بهترین واکنش است. اما در صورتی که همه بازیکن ها به طور همزمان تغییر استراتژی بدهند، جامعه می تواند به یک تعادل نش دیگر(در صورت وجود) منتقل شود. بنا بر این، تغییر قرارداد تنها در صورتی امکانپذیر است که یک توافق جمعی بین بازیکن ها در مورد جابجایی به یک تعادل دیگر، یا همان تغییر قرارداد، صورت گیرد. تغییر یکجانبه و عدم رعایت قرارداد توسط هر فرد به تنهایی، موجب زیان خود بازیکن خواهد شد.

نتیجه گیری
به طور خلاصه، می توان نتیجه گرفت که به تعادل رسیدن رفتارهای یک گروه بازیکن ها در یک بازی اجتماعی، در صورتی به طور خود به خودی و در مکانیزم خود بازی انجام پذیر است که نقطه تعادل، یک تعادل نش برای بازی مورد نظر باشد.
با توجه به ویژگی ها موجود در این تعادل، می توان نتیجه گرفت که در صورتی که یک قدرت مرکزی مایل به وضع یک قانون اجتماعی باشد، می تواند با دستکاری ماتریس پرداخت ها در نقطه مورد نظر خود، یک تعادل نش ایجاد کند. از این طریق، مکانیزم رفتاری بازیکن ها، به طور خودکار اجتماع را در آن تعادل نش پایدار خواهد ساخت.

مراجع

[۱] Andreu Mas-Colell and Michael D. Whinston. Microeconomic Theory. Oxford University Press, ۱۹۹۵.
[۲] Yoav Shoham and Moshe Tennenholtz. On the Emergence of Social Conventions: modeling, analysis, and simulations. In Artificial Intelligence, ۹٤(۱):۱۳۹–۱٦٦, ۱۹۹۷.
این مطلب در تاریخ ۱۵ مرداد ۱۳۸٤،
مراجع

محمد اردشیر، ۱۳۸۳، منطق ریاضی، انتشارات نوبهار
[Beli۰۱] Johan Gijsbertus Frederik Belifante, ۲۰۰۱] Computer proofs about finite and regular sets: the unifying concept of subvariance,
[BioNNWik] Biological neural network, www.Wikipedia.org]
[Bra۸٤] Brand and Hamish, ۱۹۸٤] The Logical Geography of Computational Approaches: A View from the East Pole, University of Arizona Press ,
[CogWik] Cognitivism (psychology), www.Wikipedia.org]
[CT۹۹] C. Walther, T. Kolbe, ۱۹۹۹] Proving theorems by reuse, Artificial Intelligence ۱۱۵) ۲۰۰۰ (۱۷-٦٦) ,
Gestalt psychology, www.Wikipedia.org
[GraphPlan۱۹۹۷] Avrim L. Blum, Merrick L. Furst, Fast Planning Through Planning Graph Analysis, Artificial Intelligence, ۲۸۱:۹۰-۳۰۰,
[GW۹۰] F. Giunchiglia, T. Walsh, ۱۹۹۰a] Using Abstraction ,
[GW۹۰] A. Bundy, F. Giunchiglia, T. Walsh, ۱۹۹۰b] Building abstractions ,
[GW۹۰] F. Giunchiglia, T. Walsh, ۱۹۹۰c] Abstract Theorem Proving: Mapping Back ,
[Senjo۸۷] T. Senjowski, C. Rosenberg, ۱۹۸۷] Parallel Networks that Learn to Pronounce English Text ,
[StaEncy۲۰۰۵] Connectionism article on Stanford Encyclopedia of Philosofy,
[Ste۰۳] Steffen Höldobler, ۲۰۰۳] Computational Logic -Working Material-, Dresden ,
[Str۷۱] Richard E. Fikes, Nils J. Nilson, ۱۹۷۱] STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Presented at the ۲nd IJCAI, Imperial College, London, England, September ۱-۳

http://daneshnameh.roshd.ir/mavara/mavara-world.com/show۳۰http://forum.p