

2019

## شبیه‌سازی منطق انسانی



حسین خاص



# شبیه‌سازی منطق انسانی

مؤلف:

حسین خاص

۱۳۹۸ مرداد

فهرست مطالب

۲-----**فهرست مطالب**

۶-----**مقدمه**

۷-----**فصل اول - نقطه آغاز و شناخت منطق انسانی**

۷-----**تناسب ذهن با جهان ما کروسکوپیک**

۷----- ذهن خام و درک «تفاوت»

۱۰----- اشیاء ذهنی

۱۱----- ویژگی‌ها

۱۳----- گزاره‌ها

۱۶----- دسته‌بندی، مجموعه و ویژگی

۱۸----- نقیض گزاره‌ها

۱۹----- سور

۲۱----- تعیین ارزش سور عمومی و فضای شیء

۲۲----- گزاره شرطی

۲۳----- ویژگی‌ها و تابع‌ها

۲۶----- ماهیت محض برای موجودیت‌های ذهنی

۲۷----- اعداد

۲۹-----**فصل دوم - یک مدل رایانه‌ای ساده از منطق ذهن انسان**

۳۰-----**شناخت اولیه مدل منسو**

۳۱----- ایجاد اشیاء و گزاره‌های ساده

۳۲----- ایجاد ویژگی و تابع

۳۵----- گزاره‌های درست و نادرست

۳۹----- عملگرهای ! و & و |



۴۱	گزارۀ شرطی و سورها
۴۲	دستور pick
۴۷	عملگرها
۴۸	تساوی
<b>۵۱</b>	<b>قوانين استدلال برای عملگرهاي منطقی</b>
۵۱	قوانين مربوط به اثبات یا رد گزارۀ پیوند عمومی
۵۳	قوانين مربوط به اثبات یا رد گزارۀ پیوند وجودی
۵۵	قوانين مربوط به اثبات یا رد if
۵۹	قوانين دیگر
۶۵	دربارۀ قوانین استدلالی
<b>۶۶</b>	<b>فضاهای و قوانین استدلال برای سورها</b>
۶۶	فضای فرض
۶۸	فضای شیء
۷۰	فضای انتخاب
۷۲	قوانين استدلالی مربوط به سورها
<b>۷۵</b>	<b>ریاضیات و مدل منسو</b>
۷۶	نمادهای ریاضی
۷۷	اشیاء ذهنی و مجموعه در ریاضی
<b>۸۲</b>	<b>فصل سوم - کاربردها و مباحث تکمیلی</b>
<b>۸۲</b>	<b>گام‌های بعدی</b>
۸۲	تبديل زبان منسو به یک نمود گرافیکی یا بازی رایانه‌ای
۸۳	شبیه‌سازی کامل ذهن
<b>۸۴</b>	<b>نتایج وجود مدل منسو</b>

---

۸۵	- بازی فکت‌ها
۸۸	- کلام و نظم افکار
۸۹	<b>منطق اجرای دستورات</b>
۸۹	- ارتباط بین دستور و زمان
۹۱	- مفهوم اجرای یک دستور
۹۲	- زبان خاص برای دستورات
۹۲	- شبیه‌سازی تفکر





## مقدمه

این پژوهه سعی دارد ذهن منطقی انسان را واکاوی و شبیه‌سازی کند. در این نوشه، مدل را معرفی خواهیم کرد که ذهن منطقی انسان را شبیه‌سازی می‌کند. این مدل را مدل منسو نامیده‌ایم. مدل منسو همراه با یک زبان برای نوشتمن کدهای منطقی و یک نرم‌افزار ارائه می‌گردد.

در فصل اول درباره مقدماتی که به مدل منسو منجر می‌شود، سخن خواهیم گفت. فصل دوم معرفی مدل اختصاص دارد و در فصل سوم به قدم‌های بعدی در پیشبرد این پژوهه و نتایج آن خواهیم پرداخت.

مدل منسو را می‌توان یکی از گام‌های اولیه و اساسی در شبیه‌سازی کامل ذهن باشد. نرم‌افزاری که برای این مدل طراحی کرده‌ایم، بخش اصلی این پژوهه است و طراحی آن حدود یکسال زمان برده است، وقتی بسیار بیشتر از تکمیل این نوشته. با این حال، این نرم‌افزار هنوز در وضعیت نسخه آلفاست به خصوص در مورد *GUI* و جای پیشرفت دارد تا به نرم‌افزاری مناسب‌تر برای کاربران تبدیل گردد.

- دو کاربرد اصلی نرم‌افزار منسو، عبارتند از
- ۱) کمک به توسعه کامل ذهن انسان و
  - ۲) بررسی خودکار اثبات‌های ریاضی توسط ذهنی مصنوعی.



# فصل اول – نقطهٔ آغاز و شناخت منطق انسانی

در این کتاب قرار است منطق انسانی، توسط قدرت منطقی یک انسان بررسی گردد. بی‌شک این کتاب توسط یک انسان نوشته شده است و توسط یک انسان خوانده می‌شود؛ یعنی ذهن به بررسی خود می‌پردازد. از این رو، در ابتدای کار، هیچ اصول و قاعده‌ای برای توضیح نخواهیم داشت و جز از طریق کلام و ارتباطات خام انسانی راهی برای بحث نخواهد بود.

## تناسب ذهن با جهان ماکروسکوپیک

چه مانند دانشمندان زیست‌شناسی تکامل تدریجی ذهن انسان در طی هزاران سال را به عنوان یک فرض قبول کرده باشیم و چه نظریات دیگر برای ما مطرح باشند، به ناچار باید بپذیریم که ذهن انسان جهان پیرامون خود را که از طریق حواس پنج‌گانه<sup>۱</sup> دریافت می‌کند می‌فهمد و می‌تواند با آن ارتباط برقرار کند. اما با پذیرش تکامل تدریجی، می‌توانیم دلیل هماهنگی ذهن انسان با جهان پیرامون را توضیح دهیم، و همین‌طور، توضیح دهیم که چرا ذهن انسان، گاه در شناخت جهان دچار خطاست؛ به عنوان مثال، این که چرا جهان کوانتمی فیزیک‌دانان را به راحتی درک نمی‌کنیم یا اتفاقاتی که در سرعت‌های بالا رخ می‌دهند (و نظریهٔ نسبیت خاص اینیشتین به آن می‌پردازد) برایمان دور از ذهن است. از این رو در این کتاب تکامل ذهن را به عنوان یک فرض خواهیم پذیرفت چه در واقعیت درست یا در واقعیت نادرست باشد.

## ذهن خام و درگ «تفاوت»

ذهن انسانی را تصور کنید که مانند ذهن نوزاد، از ابتدای رشد هیچ پیامی دریافت نکرده است. به چنین ذهنی، ذهن خام می‌گوییم. یک ذهن خام، مانند یک دستگاه رایانه است که هیچ سیستم عاملی بر آن نصب نشده است و در نتیجه، هیچ کاری نمی‌تواند انجام دهد.

<sup>۱</sup> در واقع تقسیم سنسورهای مغز به پنج حس بسیار ابتدایی است اما مشکلی در بحث ما ایجاد نمی‌کند.

می‌توان ذهنی ابتدایی‌تر را نیز تصور کرد که در ابتدای مسیر تکامل است و هنوز ساختار ذهن یک انسان را ندارد اما به حواس پنج‌گانه مجهز است. چنین ذهنی را ذهن خام اولیه می‌نامیم اما در اینجا زیاد با آن کاری نداریم.

فرض کنید به چشمان متصل به یک ذهن خام، تصویر یک پردهٔ کاملاً سیاه را (برای بار اول و به عنوان اولین چیزی که از ابتدای وجود تا به حال می‌بیند) نشان دهیم:



در این حالت، ذهن، پرده‌ای سیاه می‌بیند، اما احتمالاً متوجه بینایی نمی‌گردد، زیرا تنها چیزی را که می‌بیند یک سیاهی است. سپس، پردهٔ سیاه را با پرده‌ای سفید جایگزین می‌کنیم و آن را به ذهن خام نشان می‌دهیم:



به نظر می‌رسد که نخستین چیزی که ذهن خام، با دیدن این تصویر درک می‌کند، یک تفاوت است. چیزی متفاوت با آنچه قبلًا درک شده است. این تفاوت، تنها در صورتی برای ذهن خام قابل درک خواهد بود که این ذهن، خاطره‌ای از گذشته، در خود ذخیره کرده باشد. خاطره‌ای که با وضعیت فعلی مقایسه شده، درکی از یک تفاوت به وجود می‌آورد.

اگر پردهٔ سیاه را با عدد ۰ و پردهٔ سفید را با عدد ۱ نشان دهیم، می‌توان گفت ذهن خام، پس از مشاهدهٔ دو پرده، به وجود این دو عدد پی می‌برد و حسی خاص در مورد موجودیت مجزای آن‌ها در ذهن خام به وجود می‌آید که این حس خاص را، «تفاوت» می‌نامیم. به نظر می‌رسد که در مورد مفهوم «تفاوت»، هیچ چیز عمیق‌تری نمی‌توانیم بگوییم. تفاوت



یک حس است که در ذهن انسانی ما، به عنوان یک حس یا قابلیت خاص، وجود دارد و اگر تکامل تدریجی ذهن را بپذیریم، باید بگوییم که قوانین فیزیک، آن را در ذهن ایجاد کرده‌اند تا انسان بتواند، در جهان فیزیکی پیرامونش به حیات ادامه دهد.

اما آیا «تفاوت» یا وجود اعداد متمایز ° و ۱، واقعیتی فیزیکی در بیرون هستند؟ به نظر می‌رسد که ما نمی‌توانیم برای این پرسش پاسخی روشن و قطعی پیدا کنیم. برای یک نمونه از اشتباهات درک ذهنی از واقعیت، می‌توان به رنگ‌ها اشاره کرد. وجود رنگ‌های مختلف در جهان پیرامون ما و درک ما از رنگ‌ها کاملاً ساختگی (و برداشت ذهنی ما) است و در جهان بیرون رنگ، چیزی جز طول موج امواج الکترومغناطیسی نیست و در حقیقت، رنگی وجود ندارد. رنگ‌ها و زیبایی آن‌ها، حاصل پردازش‌های غیرمعمول ذهن از سنسور بینایی هستند. حال از کجا معلوم که «تفاوت» و «اعداد متمایز»، یک اغراق یا خطای ذهنی نباشند؟ آنچه مسلم است، این است که این برداشت ذهن از وجود «تفاوت» در جهان پیرامونی، برای ادامهٔ حیات انسان، و ایجاد درکی که او را برای ادامهٔ حیات یاری می‌رساند، مفید بوده و هست. اما آیا این درک، چه قدر حقيقی و اغراق‌نشده است، بر ما معلوم نیست و بعيد است که انسان بتواند میزان واقعی\_بودن این درک را دریابد.

بنابراین، «تفاوت»، یکی از واقعیات ذهن انسان است و همراه با واقعیت‌ها و قابلیت‌های دیگر ذهن، ابزار اصلی منطق انسانی است. در هر حال، هیچ دلیل محکمی بر اصالت این منطق و تطابق کامل آن بر جهان فیزیکی وجود ندارد. از شواهدی که ممکن است غیر\_واقعی\_بودن منطق انسانی را زیر سوال ببرد، آزمایش‌های فیزیک در مورد ذرات و نظریهٔ کوانتوم است که با منطق انسانی ناسازگار است. شاید اگر انسان، موجودی کوانتومی بود و ذهن او در فضای ذرات ریز تکامل یافته بود، منطق ذهن او متفاوت و سازگار با دنیای ذرات بود و جهان ماکروسکوپیک را عجیب و غیرمنطقی می‌دید.

با توجه به آنچه گفته شد، منطق انسان مخلوق قابلیت‌های ذهن انسان است و اصالت آن، نه به واقعیت‌های جهان فیزیکی، که تنها به ساختار ذهنی انسان‌ها برمی‌گردد؛ ساختاری که ممکن است خطاها‌یی اساسی داشته باشد، اما به طرز حیرت‌انگیزی، با جهان فیزیکی ماکروسکوپیک هماهنگی دارد.

## اشیاء ذهنی

وقتی به ذهن خام دو پردهٔ سیاه و سفید را نشان می‌دهیم، ذهن، متوجه یک تفاوت می‌شود. اما با چه فرایندی متوجه این تفاوت می‌گردد؟ شاید متخصصان و محققان مغز و اعصاب بتوانند فرایнд این «درک تفاوت» را توضیح دهند. این فرایند، هر چه باشد، به کار ما نمی‌آید. ما به مدلی ساده برای توضیح فرایندهای ذهنی نیاز داریم.<sup>۲</sup>

یکی از مدل‌هایی که می‌توانند این فرایند را توجیه کنند، به این صورت است: ذهن به ازای آنچه از چشم دریافت می‌کند، ما\_به\_ازایی ذهنی می‌سازد که به آن شیء ذهنی می‌گوییم: برای نمونه، پس از دیدن پرده‌های سیاه و سفید برای هریک از آن‌ها، شبیه ذهنی می‌سازد که نمایندهٔ آن‌ها در ذهن است. سپس، ذهن، درک می‌کند که دو شیء ذهنی متفاوت در خود دارد و از اینجا، تفاوت و تمایز بین آن‌ها را درک می‌کند. البته این درک تمایز، خود قابلیتی است که در فرایند تکامل «ذهن خام اولیه» به «ذهن خام» روی داده است.

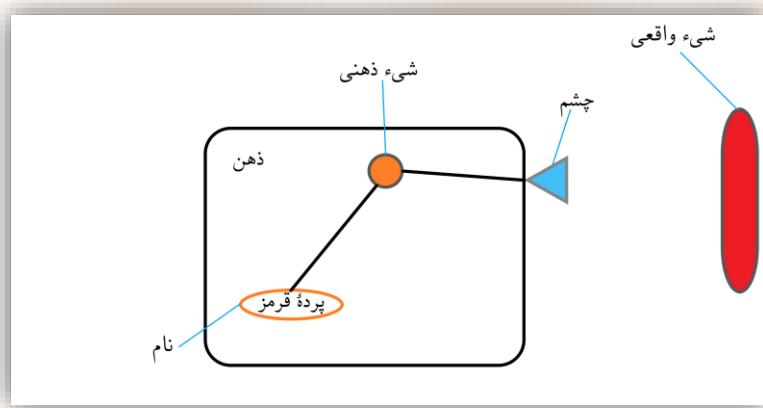
به این ترتیب، می‌توان گفت که ذهن، سیگنال‌های بینایی‌ای را که از دیدن پردهٔ سیاه دریافت کرده است، در جایی از ذهن، به عنوان شبیه ذهنی، ذخیره کرده، و سپس، پرده‌ای سفید را، به عنوان سیگنال بینایی دریافت کرده است. با دیدن پردهٔ سفید، ذهن، به طور خودکار، سیگنال‌های جدید مربوط به این پرده را، به صورت شیء ذهنی جدیدی ذخیره کرده است. در نهایت، ذهن متوجه وجود دو شیء ذهنی شده است که هم‌زمان در ذهن ذخیره شده‌اند. این متوجه\_شدن را «درک تفاوت» می‌نامیم.

اگر به ترتیب، در مقابل چشمان یک ذهن خام، سه پرده، با سه رنگ متمایز قرار دهیم، در این ذهن، سه شیء ذهنی ایجاد خواهد شد و اگر تعداد پرده‌ها را بیشتر کنیم، اشیاء ذهنی جدیدی در ذهنش ایجاد خواهند گردید. اما ظرفیت و حافظه ذهن محدود است و ممکن است پس از مدتی، اشیاء ذخیره شده از ذهن محو شوند؛ زیرا قابلیت نگهداشت تعداد بسیار زیادی از اشیاء ذهنی، کمکی به حیات انسان نمی‌کرده است که به وجود آید.

<sup>۲</sup> در برنامه‌نویسی، زبان ماشین به کار\_رفته در محتوای فایل اجرایی نهایی، که مثلاً از کامپایل\_شدن یک برنامه C++ حاصل شده است، چندان برای انسان قابل\_فهم نیست. آنچه قابل\_فهم است، کد C++ قبل از کامپایل\_شدن است. در اینجا نیز، آنچه واقعاً در مغز روی می‌دهد، برای بررسی‌های ما مناسب نیست، ما به مدلی ساده‌تر برای فهم فرایندهای ذهنی نیاز داریم.



ذهن انسان هر یک از اشیاء ذهنی را با سیگنال‌هایی که دریافت می‌کند می‌سازد، یعنی شیء ذهنی، همانند نمادی است که نمایندهٔ چیزی در بیرون است. اما از سوی دیگر، ذهن می‌تواند این اشیاء ذهنی را، به چیزهای دیگر و یا چیزهایی دیگر را به آن‌ها متصل کند. برای نمونه، می‌تواند به هر شیء ذهنی یک «نام» اختصاص دهد؛ مثلاً یک شیء ذهنی را «پردهٔ سیاه» و شیء ذهنی دیگر را «پردهٔ سفید» بنامد. تصویر سادهٔ زیر سعی دارد آنچه را گفتیم ارائه دهد:



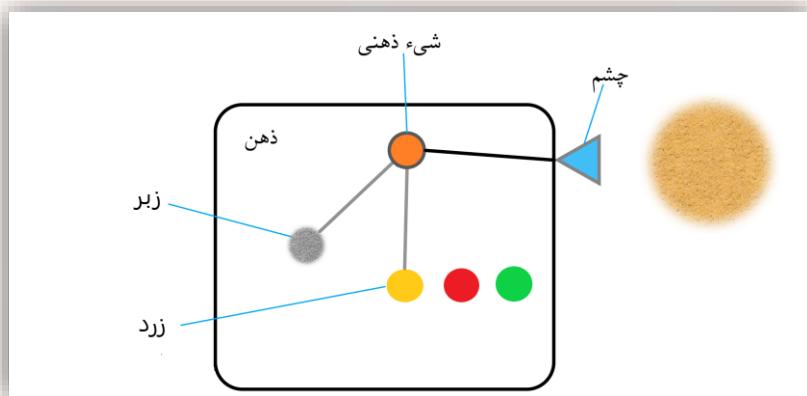
## ویژگی‌ها

فرض کنید دو بار یک تصویر واحد را به یک ذهن خام نشان دهیم. در این صورت، برای هر بار نشان\_دادن، شیء ذهنی جدیدی ایجاد نمی‌گردد؛ بلکه ذهن متوجه می‌شود که این تصویر جدید، همان تصویر قبلی است، و در نتیجه، آن را به شیء ذهنی قبلی ارتباط می‌دهد، بدون این که شیء ذهنی جدیدی ایجاد گردد. اما ذهن چگونه متوجه می‌شود که تصویر جدید، بر تصویر قبلی منطبق است؟ پاسخ این است که هر شیء ذهنی، ویژگی‌هایی هم دارد که توصیف\_کنندهٔ آن شیء ذهنی هستند.

یک شیء می‌تواند ویژگی‌هایی مانند: رنگ، سفتی، نرمی، زیری، صافی، طرح، شکل و... داشته باشد. هر یک از این ویژگی‌ها، تجارب خاصی هستند که ذهن، در برخورد با یک شیء آن‌ها را درک می‌کند. برای نمونه، درک می‌کند که یک شیء قرمزرنگ است و درک می‌کند که شیء دیگری سبزرنگ است. اما این که چرا این درک، ذهن وجود دارد،

بر می‌گردد به تکامل ذهن، از ذهن خام اولیه به ذهن خام. این تکامل، در مسری صورت گرفته است که تداوم حیات انسان را تضمین کند و این‌گونه درک‌ها از ویژگی‌ها، نیز، لاجرم از ضروریات تداوم حیات بوده‌اند.

در ذهن خام، به تدریج و با تجربه‌های مختلف، ویژگی‌های خاص ثبت می‌شوند. ویژگی‌هایی مانند رنگ، بو، زیری و... . پس از ثبت این ویژگی‌ها، ذهن، با دیدن یک شیء واقعی جدید، یا به طور کلی، حس کردن آن از طریق حواس پنج‌گانه، شیئی ذهنی ایجاد می‌کند و هنگام ایجاد این شیء ذهنی، ارتباطاتی بین آن شیء و ویژگی‌هایی که در ذهن ثبت شده‌اند به وجود می‌آورد.



شیء ذهنی برای مدتی (شاید سال‌ها) در حافظه ذهن ثبت می‌گردد.

فرض کنید شیئی با نام  $X$  در ذهن ذخیره شده است و این شیء دارای ویژگی‌های  $a$  و  $b$  است (مثلًاً زرد و زبر است). پس از مدتی، سیگنال‌های شیئی با ویژگی‌های  $a$  و  $b$  (زرد و زبر) به ذهن می‌رسند. ابتدا، ذهن سعی می‌کند شیء ذهنی جدیدی با همان ویژگی‌ها ایجاد کند و برای نمونه نام آن را  $y$  بگذارد، اما ناگهان، فرایندی ذهنی، به طور غیر ارادی اجرا می‌شود و می‌گوید که قبلًاً، شیئی با همین دو ویژگی در ذهن وجود دارد، و احتمالاً این شیء جدید همان شیء قبلی است. در این حالت، ذهن می‌تواند از ایجاد شیء  $y$  منصرف شود و همان شیء  $X$  را در نظر بگیرد.

فرض کنید شخصی را از پشت سر می‌بینید و نمی‌شناشید. در این حالت، شیء ذهنی جدیدی برای او ایجاد می‌کنید (با ویژگی‌هایی کم، در حد آشنازی در بار اول)، اما ناگهان آن شخص برمی‌گردد و می‌بینید که چهره‌اش آشناست و او از دوستان شماست و قبلًا شیء ذهنی‌ای با ویژگی‌های فراوان در ذهن خود از او دارید. در این حالت، شیء ذهنی جدید از ذهن شما محو می‌شود و شیء ذهنی قبلی یادآوری می‌شود (و آماده افرودن خاطرات (=ویژگی‌های) جدید می‌گردد).

## گزاره‌ها

گفتیم که در ذهن اشیاء ذهنی و ویژگی‌ها وجود دارند. شیء ذهنی، ویژگی یا هر چیز مشابه دیگر در ذهن را یک موجودیت ذهنی می‌نامیم. توجه کنید که یک شیء ذهنی یک موجودیت ذهنی هم هست اما یک موجودیت ذهنی ممکن است شیء ذهنی نباشد و مثلًا یک ویژگی باشد. چنان‌که در تصاویر پیشین دیدید، ممکن است ارتباطاتی بین موجودیت‌های ذهنی وجود داشته باشند.

ذهن، می‌تواند وجود یا عدم وجود ارتباط بین دو موجودیت را بررسی کند. برای مثال، با دیدن آب، شیئی ذهنی برای آن به وجود می‌آورد و آن را با ویژگی «شفافیت» مرتبط می‌کند. به این شکل، بین دو موجودیت ذهنی، یک ارتباط برقرار می‌کند. پس بین «شیء ذهنی آب» و «شفافیت» ارتباط برقرار می‌شود، اما بین «شیء ذهنی آب» و ویژگی «رنگ قرمز» ارتباطی برقرار نمی‌گردد.

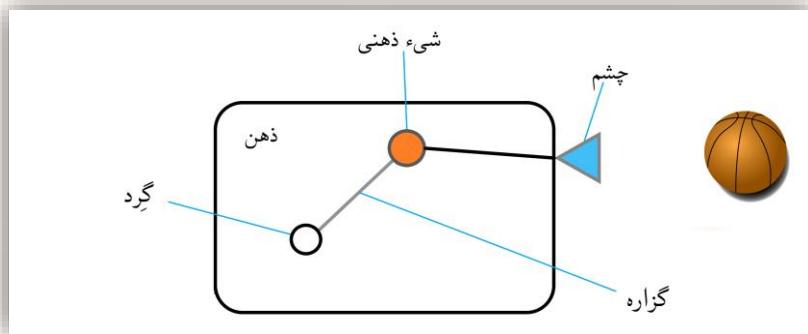
به این ترتیب، اگر  $x$  و  $y$  دو موجودیت ذهنی باشند، دو حالت وجود دارد: یا بین آن‌ها ارتباطی هست و یا نیست. زوج مرتب  $(x, y)$  را گزاره می‌نامیم. اگر این زوج را با  $p$  نشان دهیم،  $p$  می‌تواند دو وضعیت داشته باشد:

- ۱) وجود ارتباط بین  $x$  و  $y$ .
- ۲) عدم وجود ارتباط بین  $x$  و  $y$ .

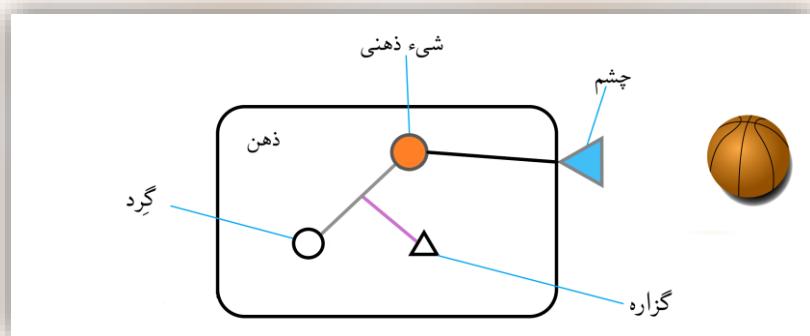
ذهن وجود یا عدم وجود ارتباط را با فرایندی ناشناخته، خود، تشخیص می‌دهد. اما مهم این است که برای گزاره  $p$ ، دو وضعیت وجود دارد. می‌توانیم وضعیت اول را «روشن» یا «درست» یا «صحیح» بنامیم و وضعیت دوم را «خاموش» یا «نادرست» یا «غلط». پس

گزاره، موجودیتی است که یکی از دو حالت درست یا نادرست را دارد. اما آیا گزاره موجودیتی ذهنی است و ذهن برای آن جایگاه و حافظه‌ای مجزا قائل است؟ یا این که تنها یک مفهوم است که برای توضیح کارکرد ذهن مناسب است؟

فرض کنید ذهن از طریق چشم یک توپ را می‌بیند:



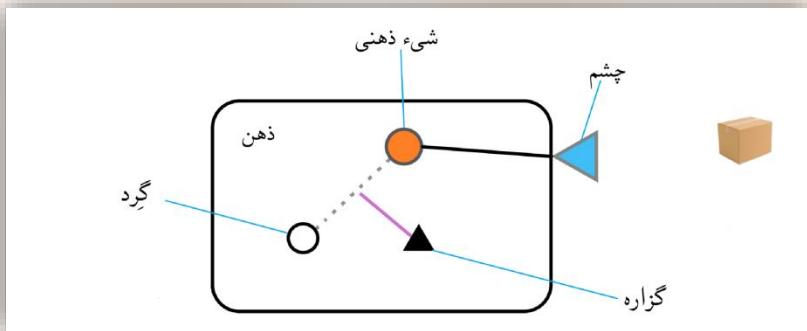
توپ، به شیئی ذهنی بدل می‌گردد و ارتباطی بین آن و ویژگی «گرد»—بودن برقرار می‌شود. این ارتباط، که در تصویر بالا، با خط خاکستری نشان داده شده است، در حقیقت، همان گزاره است؛ یعنی گزاره، در ذهن وجود دارد. اما برای این که این گزاره را بتوانیم موجودیتی ذهنی بنامیم، ذهن باید متوجه آن گردد و محلی جداگانه برای آن در نظر بگیرد:



در تصویر بالا، مثلث سفید، نماینده یک گزاره درست در ذهن است. وقتی می‌خواهیم گزاره تصویر بالا را به صورت یک جمله بگوییم، می‌گوییم «توپ گرد است». این جمله، در واقع،

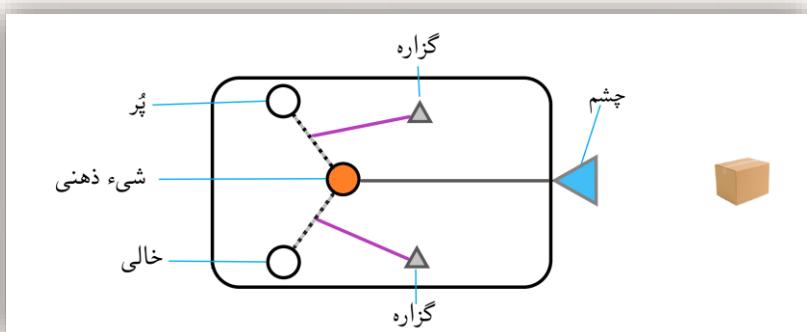


بیانگر محتوای گزاره است: ارتباط بین توب و ویژگی گردی. ذهن می‌تواند یک گزاره نادرست را هم به موجودیتی ذهنی تبدیل کند:



در تصویر فوق، ذهن، یک جعبه می‌بیند که شیء ذهنی ساخته شده برای آن، ارتباطی با ویژگی «گرد»-بودن ندارد؛ زیرا جعبه گرد نیست. این عدم ارتباط، که در تصویر فوق، با خط نقطه‌چین نشان داده شده است، یک گزاره نادرست است که ذهن می‌تواند به آن موجودیتی ذهنی دهد. در تصویر، این موجودیت ذهنی، با مثلث سیاه نشان داده شده است. مثلث سیاه، گزاره‌ای نادرست است. به این ترتیب، ذهن، هم گزاره‌های درست و هم گزاره‌های نادرست را می‌تواند به عنوان موجودیت‌هایی ذهنی در خود ذخیره کند.

ذهن، علاوه بر گزاره‌هایی که درستی یا نادرستی آنها را تشخیص داده است، می‌تواند گزاره‌هایی را به عنوان یک موجودیت ذهنی در خود جای دهد که هنوز درستی یا نادرستی‌شان را نمی‌داند:



در تصویر بالا، چشم، یک جعبه دربسته می‌بیند اما ذهن نمی‌داند درون جعبه پر است یا خالی. بنابراین، دو گزاره برای ذهن مطرح می‌شوند:

- (۱) جعبه پر است.
- (۲) جعبه خالی است.

در تصویر بالا، هر یک از این دو گزاره، با یک مثلث خاکستری نشان داده شده است.

به این ترتیب، گزاره یک موجودیت ذهنی است که می‌تواند دو حالت داشته باشد:

- (۱) بلا تکلیف.

(۲) ارزش‌گذاری شده. یک گزاره ارزش‌گذاری شده یکی از دو ویژگی «درست» یا «نادرست» را دارد.

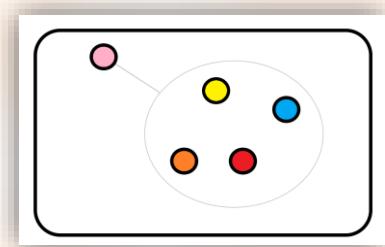
گزاره بلا تکلیف، گزاره‌ای است که ذهن درستی یا نادرستیش را نمی‌داند، اما عجالتاً آن به عنوان یک موجودیت ذهنی در نظر می‌گیرد تا بعداً، اگر شرایط مهیا شد، درستی یا نادرستیش تعیین گردد. ذهن بر این تصور است که گزاره بلا تکلیف، یکی از دو ویژگی «درستی» یا «نادرستی» را دارد و با انجام فرایندی مناسب، معلوم می‌شود که گزاره کدام ویژگی را دارد. به این ترتیب، «گزاره» نوعی «شیء ذهنی» است که حداقل یکی از دو ویژگی «درستی» یا «نادرستی» را دارد.

### دسته‌بندی، مجموعه و ویژگی

وقتی یک ساعت را می‌بینیم، یک شیء ذهنی، متاخر با آن ساعت، در ذهن ما ایجاد می‌گردد. این شیء ذهنی، تنها یکی است. حال فرض کنید پیچ و مهرهای آن ساعت را باز کنیم و قطعات مختلف آن را خارج کنیم و کنار هم روی میز قرار دهیم و دوباره آن‌ها را به هم متصل کنیم و ساعت را مثل ابتدای کار بیندیم. چه اتفاقی برای آن شیء ذهنی واحد اولیه می‌افتد؟

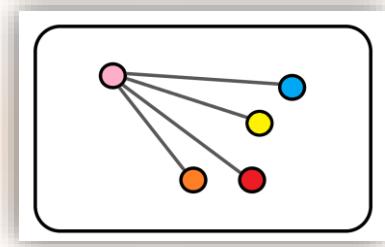
چنین مشاهداتی در مسیر تکامل ذهن، سبب شده است که ذهن توانایی دسته‌بندی پیدا کند. ذهن، برای تمام قطعاتی از ساعت که مشاهده کرده است، اشیائی ذهنی در نظر می‌گیرد و نیز آن‌ها را، در کنار هم، یک دسته واحد در نظر می‌گیرد و شیء ذهنی اولیه را نماینده این دسته می‌داند.

بنابراین، ذهن می‌تواند یک مجموعه از اشیاء ذهنی را به صورت شیئی ذهنی در بیاورد:



در تصویر بالا، موجودیت‌های ذهنی‌ای که با رنگ‌های زرد، آبی، قرمز و نارنجی نشان داده شده‌اند، به صورت یک مجموعه درآمده‌اند که موجودیت ذهنی صورتی‌رنگ، نمایندهٔ این مجموعه است.

ذهن، می‌تواند آزادانه هر دسته از موجودیت‌های ذهنی را به صورت یک مجموعه در بیاورد و آن را یک شیء ذهنی جداً‌گانه در نظر بگیرد. برای مثال، ذهن می‌تواند مجموعه‌ای از گزاره‌ها را در خود بسازد. در مدلی که از ذهن توصیف می‌کنیم، مجموعه را به صورت یک ویژگی محسوب خواهیم کرد؛ یک ویژگی ساختگی که عده‌ای از موجودیت‌های ذهنی حائز آن هستند؛ برای نمونه، در تصویر بالا، موجودیت‌های ذهنی زرد، آبی، قرمز و نارنجی دارای ویژگی‌ای هستند که با ویژگی صورتی‌رنگ نشان داده شده است. در واقع، تصویر فوق را، برای هماهنگی با تصاویری که پیش‌تر نشان داده شدند، می‌توان به این صورت تغییر داد:



ذهن، معمولاً<sup>اً</sup>، این توانایی را دارد که با درنظرگرفتن یک ویژگی، جستجو کند و ببینید که چه موجودیت‌های ذهنی‌ای، آن ویژگی را دارند. بنابراین، با داشتن ویژگی صورتی‌رنگ

در تصویر بالا، به راحتی تمام اشیاء ذهنی آبی، زرد، قرمز و نارنجی قابل\_پیدا\_شدن هستند. همچنین، ذهن، به راحتی می‌تواند ارتباط بین یک موجودیت و ویژگی را به عنوان یک گزاره بررسی کند. برای مثال، در تصویر بالا، «شیء قرمز، دارای ویژگی صورتی است» و این یک گزاره است.

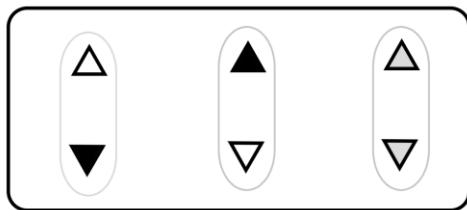
به این ترتیب، ویژگی‌های ذهنی، برای دسته‌بندی موجودیت‌های ذهنی، قابل\_استفاده‌اند و ذهن، می‌تواند، برای دسته‌بندی‌های خود، ویژگی‌هایی کاملاً انتزاعی به وجود آورد، یعنی ویژگی‌هایی که هیچ توصیفی گرفته‌شده از حواس پنج‌گانه در خود ندارند و کاملاً انتزاعی و مجرد هستند و تنها برای دسته‌بندی قابل\_استفاده‌اند.

### نقیض گزاره‌ها

ذهن دارای این قابلیت است که از روی گزاره‌های موجود در خود، گزاره‌هایی جدید خلق کند. این گزاره‌های جدید، لزوماً به معنی وجود ارتباط بین یک شیء ذهنی و یک ویژگی نیستند. در حقیقت، ذهن، به تدریج، این قابلیت را پیدا کرده است که گزاره‌ها را به عنوان موجودیت‌های ذهنی مستقلی به وجود بیاورد، بدون این که بیانگر ارتباط بین یک شیء ذهنی و یک ویژگی باشند. چندین روش کاملاً مشخص، برای ایجاد گزاره‌های جدید از روی گزاره‌های قبلی وجود دارد که برخی از آن‌ها، در این فصل معرفی خواهند شد. یک روش ایجاد گزاره جدید، ایجاد نقیض یک گزاره موجود است.

اگر گزاره‌ای در ذهن موجود باشد، ذهن می‌تواند نقیض آن گزاره را هم به عنوان موجودیتی ذهنی خلق و بررسی کند. اگر p یک گزاره باشد، در این کتاب، نقیض آن را با p ! نشان خواهیم داد. نقیض گزاره، گزاره‌ای است که ارزشش (یعنی درستی یا نادرستیش) طبق تعریف، بر عکس خود گزاره است. به عنوان نمونه، اگر p گزاره‌ای درست باشد، p ! گزاره‌ای نادرست خواهد بود و اگر p گزاره‌ای نادرست باشد، p ! گزاره‌ای درست خواهد بود. با این حال، ذهن، برای ایجاد نقیض یک گزاره، نیازی ندارد که بداند آن گزاره درست است یا غلط. یعنی ذهن می‌تواند برای گزاره‌های بلا تکلیف نیز نقیضی در نظر بگیرد. اما با معین\_شدن ارزش یک گزاره، ذهن، طبق قانون، می‌تواند ارزش نقیض آن را هم معین کند.





تصویر بالا، سه گزاره و نقیض‌هایشان را نشان می‌دهد. مثلث سیاه، یعنی گزاره نادرست، مثلث سفید یعنی گزاره درست و مثلث خاکستری یعنی گزاره بلا تکلیف. نقیض گزاره‌ای که با یک مثلث نشان داده شده، با مثلثی وارونه نمایش داده شده است.

ذهن، معمولاً، نقیض نقیض یک گزاره را منطبق بر خود گزاره می‌گیرد. برای مثال، فرض کنید  $p$ ، گزاره «توب گرد است» باشد. در این صورت،  $\neg p$  گزاره «توب گرد نیست» خواهد بود. نقیض  $\neg p$ ! که با  $\neg \neg p$ !! نشان داده می‌شود همان  $p$  است.

## سور

فرض کنید  $A$  مجموعه‌ای از گزاره‌ها باشد. ذهن، می‌تواند دو گزاره مهم از روی چنین مجموعه‌ای بسازد:

- (۱) همه اعضای  $A$  (یعنی همه گزاره‌هایی در ذهن که ویژگی  $A$  را دارند) گزاره‌هایی درستند.
- (۲) دست‌کم یکی از اعضای  $A$  درست است.

به این گزاره‌ها سور گفته می‌شود. به گزاره اول سور عمومی و به گزاره دوم سور وجودی گفته می‌شود. اما نامگذاری برای آن‌ها چندان اهمیت ندارد؛ آنچه مهم است دو مفهوم «همه» و «دست‌کم یکی» که ذهن با آن‌ها آشناست.

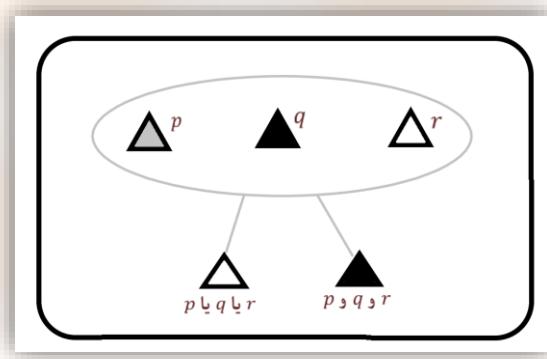
با تکامل زبان گفتاری نسل بشر، به جز کلمه «همه»، کلمه «و» با معنای مشابه، تکامل یافته است. اگر  $A$  متشکل از گزاره‌های  $p$ ,  $q$  و  $r$  باشد، یعنی اگر  $A = \{p, q, r\}$ ، به جای این که بگوییم همه اعضای  $A$  درستند، می‌گوییم « $p$  و  $q$  و  $r$ » درست است. به «و» پیوند عمومی خواهیم گفت.

همچنین، مطابق رسم ریاضیات، به جای این که بگوییم دستکم یکی از اعضای A درست است، می‌گوییم «p یا q یا r» درست است. به «یا» پیوند وجود خواهیم گفت.

به این ترتیب، «p و q و r» و نیز «p یا q یا r»، خود، گزاره‌هایی جدید هستند که ذهن می‌تواند آنها را به عنوان گزاره‌هایی جداگانه در نظر بگیرد.

ذهن برای تحقیق درستی یا نادرستی گزاره «p و q و r» درستی یا نادرستی تک‌تک گزاره‌های p، q و r را بررسی می‌کند، اگر «همه» درست بودند «p و q و r» را درست ارزش‌گذاری می‌کند، و در غیر این صورت، نادرست.

همچنین ذهن برای تحقیق درستی یا نادرستی گزاره «p یا q یا r» درستی یا نادرستی تک‌تک گزاره‌های p، q و r را بررسی می‌کند، اگر «دستکم یکی» از آنها درست بود، «p و q و r» را درست ارزش‌گذاری می‌کند، و در غیر این صورت، نادرست.



تصویر بالا نشان می‌دهد که ذهن از روی چند گزاره می‌تواند دو سور بسازد.

بسیاری از سورها به این صورت هستند که یک مجموعه از اشیاء ذهنی مانند X داریم و یک ویژگی مانند p داریم و سور، به صورت «همه اشیاء درون X دارای ویژگی p هستند» است؛ برای نمونه «همه لیوان‌های درون این سینی خالی هستند» یک سور است. به X، مجموعه سور می‌گوییم و به p ویژگی سور. توجه داشته باشید که این شکل از سور نیز، در واقع، بررسی دسته‌ای از گزاره‌های است، اما هر گزاره، وجود یا عدم وجود ارتباط بین یک



شیء از یک مجموعه و یک ویژگی است. این‌گونه سورها، فراوانی زیادی دارند و اگر با منطق آشنا‌تر باشید، می‌دانید که همه سورها به همین شکل قابل‌تبیین هستند.

### تعیین ارزش سور عمومی و فضای شیء

ذهن برای این که ارزش گزاره‌ها (یعنی درستی یا نادرستی آن‌ها) را بررسی کند فرایندهای مشخصی دارد که به تدریج و با کسب تجربیات تکامل یافته‌اند. شناسایی این فرایندها و تربیت آن‌ها نیازمند تمرین است. بسیاری از افراد حتی افرادی که در جایگاه‌های علمی خوبی است ممکن است در این فرایندهای ذهنی دچار ضعف باشند. این فرایندها، مانند قوانین شطروح هستند و باید با تمرین در آن‌ها مهارت پیدا کرد. اگرچه مهارت‌های منطقی، بر خلاف قوانین شطروح در ذهن ثبت شده‌اند، اما آن‌قدرها هم در ادامه حیات مؤثر نبوده‌اند که همه، در به‌کار‌گیری تمام و کمال آن‌ها ماهر باشند. در این بخش در مورد مهارت ذهنی‌ای است که توسط آن ارزش یک سور عمومی تعیین می‌گردد، سخن می‌گوییم:

برای این که ذهن ارزش یک سور عمومی را بررسی کند، همیشه نمی‌تواند به سراغ تمام گزاره‌هایی برود که سور درباره آن‌ها سخن می‌گوید. سور «همه لیوان‌های داخل این سینی خالی هستند» را در نظر بگیرید. ذهن برای بررسی این سور، باید تک‌تک لیوان‌ها را بررسی کند و اگر همه خالی بودند، سور را «درست» در نظر بگیرد. اما سور «همه سنگ‌ها در آب فرو می‌روند» چیزی نیست که ذهن بتواند درستی آن را با بررسی همه موارد تأیید کند.

مهم‌ترین روشی که برای تعیین ارزش یک سور عمومی به کار می‌رود استفاده از فرایندهای است که ما به آن «استفاده از فضای شیء» می‌گوییم. در این فرایند، شیء موقتی، مثلًاً با نام  $X$ ، در ذهن ساخته می‌شود که می‌تواند هر یکی از اعضای مجموعه سور باشد، اما هیچ عضو به خصوصی از آن نیست. تنها ویژگی شیء  $X$  این است که عضوی از مجموعه سور است؛ به عبارت درست‌تر، اگر  $X$  مجموعه سور باشد، تنها ویژگی  $X$ ، (ارتباط با)  $X$  است و هنگام ایجاد شیء موقت  $X$ ، ویژگی دیگری برای آن در نظر گرفته نمی‌شود. شیء  $X$ ، موقتاً موجود است و با استفاده از آن، فراینده استدلالی در ذهن انجام می‌پذیرد تا  $X$  به ویژگی دیگری مرتبط شود که همان ویژگی سور است.

برای مثال، گزاره «همه سنگ‌ها در آب فرو می‌روند» را در نظر بگیرید. ذهن برای اثبات این گزاره، می‌تواند به این شکل استدلال کند:

(۱) ذهن فرض می‌کند  $x$  یک سنگ باشد ( $x$  یک شیء موقت است که تنها ویژگی آن سنگ‌بودن است).

(۲) ذهن می‌داند که چگالی هر سنگ از چگالی آب بیشتر است. پس چگالی  $x$  هم که یک سنگ است از آب بیشتر است.

(۳) ذهن می‌داند اگر که هر جسمی که چگالی آن از آب بیشتر است، در آب فرو می‌رود.  $x$  هم جسمی است که چگالی آن از آب بیشتر است. پس  $x$  در آب فرو می‌رود.

به این ترتیب، ذهن این گزاره را که « $x$  دارای ویژگی فرو\_رفتن\_در\_آب است»، گزاره‌ای درست تشخیص می‌دهد. حال، بر اساس فرآیند «استفاده از فضای شیء»، ذهن گزاره «همه سنگ‌ها در آب فرو می‌روند» را نیز درست محسوب می‌کند. پس از این تشخیص، شیء موقت  $x$  از ذهن محو می‌شود. شیء موقت  $x$ ، در برهه‌ای کوتاه از زمان موجود بود و به فرآیند اثبات کمک کرد. اما از آنجا که معمولاً این برهه از زمان، به صورت نوشتاری بر صفحه کاغذ در می‌آید، مدت زمان اعتبار آن، معادل با بخشی از نوشتار است که  $x$  در آن معتبر است. این بخش را «فضای شیء»  $x$  می‌گوییم؛ فضایی که در آن، این شیء قابل استفاده است.

## گزاره شرطی

گزاره شرطی، گزاره‌ای است در مورد ارتباط دو گزاره، پیش‌تر گفتیم که اگر یک گزاره درست باشد، نقیض آن، نادرست است و برعکس. در واقع، به بیان یک ارتباط بین یک گزاره و گزاره دیگر پرداختیم. اما همین ارتباط خود، یک گزاره است، یعنی با فرض این که  $p$  یک گزاره است، «اگر  $p$  درست باشد،  $p$  ! نادرست است»، خود یک گزاره است. به  $p$  چنین گزاره‌هایی گزاره شرطی می‌گوییم. گزاره شرطی به این صورت است: «اگر گزاره  $p$  درست باشد، آنگاه گزاره  $q$  نیز درست خواهد بود». این گزاره، در حقیقت، یک دستورالعمل برای ذهن است برای درست\_دانستن گزاره  $q$ ، وقتی که درستی  $p$  برای ذهن روشن شده است.





بسیاری از گزاره‌های شرطی، بر اساس تجربیات در ذهن به وجود می‌آیند. برای نمونه گزاره «اگر لیوان را رها کنی، می‌افتد»، گزاره‌ای است که با تجربیات فراوان انسان در ذهن او شکل گرفته است.

بنابراین، گزاره‌های شرطی، مانند سورها، گزاره‌های هستند که از روی گزاره‌های دیگر ساخته می‌شوند. معمولاً، گزاره شرطی «اگر  $p$  درست باشد آنگاه  $q$  درست است» را، در ریاضیات، با نماد  $q \rightarrow p$  نشان می‌دهند.  $q \rightarrow p$  گزاره‌ای است که از روی  $p$  و  $q$  ساخته شده است.

در ریاضیات یا بحث‌های منطقی، برای اثبات گزاره شرطی (یعنی تشخیص آن به عنوان گزاره‌ای درست)، معمولاً از فرایند فضای فرض استفاده می‌شود که مشابه فضای شیء است. در حقیقت، برای اثبات  $q \rightarrow p$ ، ذهن، موقتاً فرض می‌کند  $p$  گزاره‌ای درست باشد، سپس، با این فرض، سعی می‌کند به درستی  $q$  برسد. سپس، فرض اولیه را رها می‌کند. ذهن، با این فرایند به خود اجازه می‌دهد که  $q \rightarrow p$  را گزاره‌ای درست بداند.

برای مثال، برای اثبات «اگر بستنی در جای گرم قرار بگیرد آب می‌شود»، فرض می‌کنیم بستنی در جای گرم قرار گرفته است. چون هوا گرم است، هر چیز یخی آب می‌شود. چون بستی یخی است، پس بستنی هم آب می‌شود. فرض گرم بودن هوار را رها می‌کنیم و نتیجه می‌گیریم که «اگر بستنی در هوای گرم قرار بگیرد آب می‌شود».

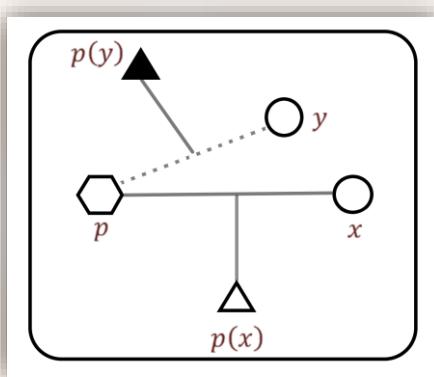
فرایند دیگری هم برای اثبات  $q \rightarrow p$  وجود دارد و آن اثبات  $p \rightarrow q$  است، که ذهن آموخته است اثبات  $p \rightarrow q$  با اثبات  $q \rightarrow p$  معادل است. فرایندهای فراوان دیگری هم هست که در فصل آینده تحت عنوان قوانین استدلال تبیین خواهند شد.

## ویژگی‌ها و تابع‌ها

می‌دانیم که گزاره‌ها درباره اتصال یک شیء ذهنی و یک ویژگی هستند:

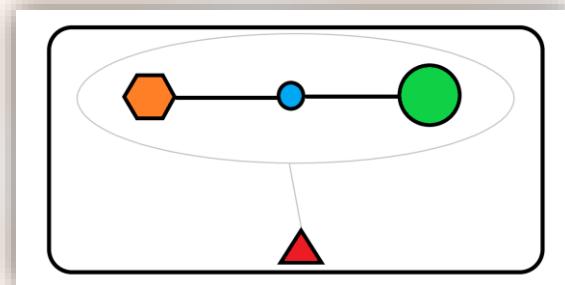


در تصویر بالا، شیء ذهنی  $x$ ، دارای ویژگی  $p$  است و شیء ذهنی  $y$ ، دارای ویژگی  $p$  نیست. چنان‌که پیش‌تر گفتیم، ارتباط بین  $x$  و  $p$  یک گزاره ایجاد می‌کند. گزاره‌ای درست که می‌گوید این ارتباط برقرار است. این گزاره را با  $p(x)$  نشان می‌دهیم. همین‌طور، عدم ارتباط بین  $y$  و  $p$  نیز، یک گزاره ایجاد می‌کند. گزاره‌ای نادرست که می‌گوید این ارتباط برقرار نیست. این گزاره را با  $p(y)$  نشان می‌دهیم:



به این ترتیب، ویژگی‌ها، موجودیت‌هایی هستند که ذهن با ترکیب آن و هر موجودیت ذهنی دیگر (یا برخی موجودیت‌های دیگر)، یک موجودیت ذهنی جدید می‌سازد. برای نمونه، در تصویر بالا، ذهن از روی  $p$  و  $y$ ، موجودیت  $(y)$  را، که یک گزاره است، ایجاد کرده است.

ویژگی‌هایی هم هستند که نیازمند دو شیئند تا گزاره‌ای ایجاد کنند. برای نمونه، ویژگی «اولی کوچک‌تر از دومی است»:

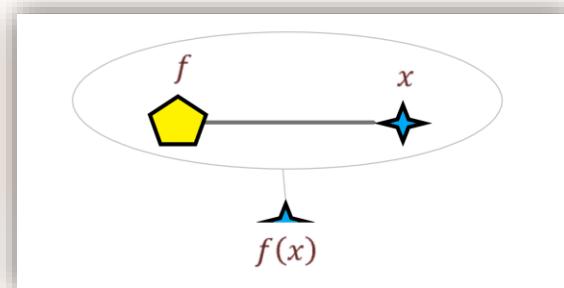




در تصویر بالا، ویژگی نارنجی رنگ، ویژگی «اولی کوچکتر از دومی است» است. «اولی» اولین چیزی است که ویژگی با آن ارتباط پیدا کرده است (در اینجا شیء آبی) و دومی، دومین چیزی است که ویژگی (از طریق شیء اول) به آن ارتباط پیدا کرده است (در اینجا شیء سبز). از مجموع این سه موجودیت، گزاره قرمزنگ حاصل شده است که گزاره «شیء آبی کوچکتر از شیء سبز است» است. اگر ویژگی نارنجی را با  $q$ ، شیء آبی را با  $a$  و شیء سبز را با  $b$  نشان دهیم، گزاره قرمز را با  $q(a, b)$  نشان خواهیم داد.  $q$ ، یک ویژگی است با دو آرگومان، یعنی تنها پس از ارتباط با دو آرگومان می‌توان از آن یک گزاره ساخت، و به علاوه، ترتیب این آرگومان‌ها نیز مهم است. به شکل مشابه، ویژگی‌هایی با تعداد بیشتر آرگومان نیز وجود دارند.

ویژگی‌ها نوعی تابع هستند. تابع موجودیتی ذهنی است که ذهن می‌توان با قراردادن آن در کنار یک یا چند موجودیت دیگر (که آرگومان نامیده می‌شوند)، موجودیت ذهنی جدیدی ایجاد کند. به عنوان مثال، در بالا، تابع  $q$  در کنار آرگومان‌های  $a$  و  $b$ ، موجودیت ذهنی  $q(a, b)$  را پدید می‌آورد. اما تابع‌ها، در کنار آرگومان‌ها، لزوماً، به گزاره منجر نمی‌شوند، بلکه می‌توانند به هر موجودیتی ذهنی دیگری هم منجر گردند. ویژگی‌ها، حالت خاصی از توابع هستند که محصولشان گزاره است (یعنی با آرگومان‌ها به گزاره منجر می‌شوند)، اما تابع، به طور کلی می‌تواند به هر موجودیت ذهنی دیگری هم منجر گردد.

برای نمونه، تابع «نیمه بالایی» را در نظر بگیرید. این تابع را با  $f$  نشان می‌دهیم. در این صورت  $(x, f(x))$  نیمه بالایی  $x$  خواهد بود:



در تصویر بالا،  $x$  شیئی ذهنی است که «نیمه بالایی»، برای آن معنی دارد و  $f(x)$  همان نیمه بالایی  $x$  است. توجه کنید که  $f(x)$  یک شیء ذهنی است نه یک گزاره. تابعها هم می‌توانند مانند ویژگی‌ها دو یا چند آرگومان داشته باشند.

علت این که  $f$ ، تابع نامیده می‌شود این است که موجودیت  $f(x)$  تابع موجودیت  $x$  است؛ یعنی اگر  $x$  با موجودیت دیگری جایگزین شود،  $f(x)$  (احتمالاً) موجودیت دیگری می‌گردد. در این فصل؛ قصد نداریم که مثال‌هایی ریاضی بیاوریم. اما مثال‌های ریاضی زیادی از تابعها وجود دارند. البته مفهوم تابع، یک مفهوم نیست که ساخته ریاضیدانان باشد، بلکه مثال‌های زیادی هستند که انسان‌ها در زندگی عادی با تابعها کار می‌کنند. به عنوان نمونه، روشن یا خاموش\_شدن لامپ، تابعی از وضعیت کلید است. انسان این مفهوم را می‌فهمد که می‌تواند از کلید استفاده کند. اگر ذهن، این قابلیت را نداشت، متوجه این عملکرد نمی‌شد. همین‌طور، انسان می‌فهمد که وضعیت تصویری که در آینه مشاهده می‌کند تابعی از وضعیت خود شخص است. این یک قابلیت در ذهن انسان است که می‌فهمد تصویر آینه، تابعی از خود او ( $=$  بدن او) است. بسیاری از حیوانات، به هیچ وجه، نمی‌توانند بین تصویر درون آینه و واقعیت، تمایزی قائل شوند و این، به وضوح نشانگر آن است که ذهن آن‌ها، ابتدایی‌تر از آن است که مفهوم «تابع» در آن شکل گرفته باشد (حتی با تمرین نیز متوجه چنین مفهومی نخواهند شد چرا که چنین قابلیتی در ذهن‌شان تکامل نیافته است). بنابراین، آنچه در مورد تابع یا مفاهیم دیگر در این فصل ارائه شد، تحلیل ذهن انسان است نه آموختن مفاهیمی در ریاضیات، فارغ از عملکرد ذهن. اما باید دقت داشت که آنچه گفته می‌شود، تنها مدلی ساده از ذهن است و این مدل را می‌توان غنی‌تر و کامل‌تر کرد، با این حال، در این کتاب، چندان به دنبال پیدا\_کردن کامل‌ترین مدل از ذهن نیستیم، تنها می‌خواهیم یک مدل ساده را با توجه به ساختار منطقی ذهن بیرون بیاوریم که با منطق انسانی سازگار است.

### ماهیت محض برای موجودیت‌های ذهنی

می‌توان همه موجودیت‌های ذهنی را یکسان دانست، به جز این که هر یک، ویژگی خاصی دارد. برای نمونه، می‌توانیم فرض کنیم گزاره یک موجودیت ذهنی است که ویژگی «گزاره بودن» دارد و شیء ذهنی، یک موجودیت ذهنی است که ویژگی «شیء ذهنی بودن» دارد،





همین‌طور، تابع یک موجودیت ذهنی است که ویژگی‌های «تابع بودن» (و نیز ویژگی‌هایی که تعداد پارامترهایش را مشخص می‌کنند) دارد.

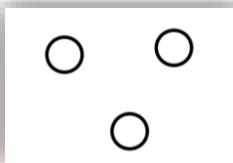
پس، می‌توان تصور کرد که هر موجودیت ذهنی، یک موجودیت محض است که به چندین ویژگی متصل است و یکی از ویژگی‌ها، نوع موجودیت را تعیین می‌کند مثل «گزاره بودن». حتی خود ویژگی‌ها را نیز، می‌توان موجودیت‌هایی محض دانست که ویژگی «ویژگی بودن» دارند. اما به نظر نمی‌رسد که این مدل، با ذهن سازگارتر از مدل وجود موجودیت‌های متفاوت باشد. به نظر نمی‌رسد که ذهن، یک گزاره را با شیء ذهنی‌ای که گزاره نیست، متفاوت می‌داند و یک تابع را، کاملاً موجودیتی متفاوت از شیء ذهنی می‌داند.

## اعداد

گفتیم که ذهن، به شکل غیر\_قابل\_توضیحی «تفاوت» را درک می‌کند و نیز گفتیم که ذهن قادر به دسته‌بندی است. ذهن، تفاوتی خاص و مشترک بین مجموعه‌های سه\_عوضی و مجموعه‌های دو\_عوضی را درک می‌کند. یعنی ذهن می‌فهمد که تفاوتی بین تصویر:

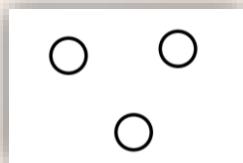


و تصویر:



وجود دارد. به نظر نمی‌رسد توضیح این که ذهن چه‌طور می‌تواند چنین تفاوتی را درک کند، برای ما ناممکن باشد. تنها می‌توانیم بگوییم ذهن خام اولیه، در طی تکامل تدریجی و مواجهه با فضای فیزیکی پیرامونش، چنان تکامل یافته است که چنین تفاوتی را احساس کند. پس، شاید احساس\_کردن این تفاوت، بازتابی از یک واقعیت فیزیکی باشد، هرچند که ممکن است این باتاب، ناقص باشد و حقیقتی دیگر در بین باشد.

به هر حال، ذهن تفاوتی را بین دو چیز و سه چیز درک می‌کند؛ همین‌طور بین یک چیز و دو چیز و همین‌طور بین سه چیز و چهار چیز. این تفاوت‌ها، سبب می‌شوند که اشیائی ذهنی، به صورت اعداد در ذهن ما وجود داشته باشند. در واقع، ذهن، با دیدن تصویر:



تابعی به نام «شمارش» را بر آن اعمال می‌کند و حاصل شیئی است که نمایندهٔ عدد ۳ است. تابع شمارش، قابلیتی است که ذهن بر اثر تکامل تدریجی آن را پیدا کرده است.

قابلیت‌های مشابه دیگری نیز در ذهن موجودند، مثل قابلیت تشخیص «بزرگی و کوچکی». اما شاید بتوان گفت قابلیت درک اعداد، مهم‌ترین قابلیت در ذهن است که توضیح روشنی ندارد. موضوع اعداد در ذهن، وقتی عجیب‌تر می‌شود که پراکنده‌ی اعداد اول را بررسی می‌کنیم. اعداد اول، در ریاضیات، هیچ نظم مشخصی ندارند و پرسش بزرگ این است که این شکل از بی‌نظمی، از کجا آمده است و چه طور اعداد اول انتخاب شده‌اند.

البته پرسش‌های مشابه دیگری نیز وجود دارند که پاسخ آسانی برایشان نیست. به عنوان نمونه، این که چرا عدد  $\pi$  حدود  $3/14$  است. تمام این ابهامات به منطق ابتدایی ذهن انسان و به خصوص درک اعداد بر می‌گردد که توضیح روشنی برای منشیان موجود نیست.



## فصل دوم – یک مدل رایانه‌ای ساده از منطق ذهن انسان

یکی از قابلیت‌های ذهن، استفاده از زبان و کلام برای توصیف و انتقال وضعیت منطقی خود است. بنابراین، آنچه در ذهن رخ می‌دهد، به صورت دنباله‌های از کلمات ارائه می‌شوند. تمام آنچه را در ذهن می‌گذرد، می‌توان به صورت رشته‌هایی از کلمات بیان کرد و این کلام را، به نوشتار تبدیل کرد. به همین سبب است که ریاضیات، منطق، برنامه‌نویسی و... همه به صورت رشته‌هایی از نمادها و حروف قابل‌ثبت هستند.

در این فصل، قرار است مدلی بسیار دقیق‌تر از آنچه در فصل قبل بیان شد، همراه با زبانی برای نوشتن آن ارائه کنیم. این مدل قابل‌بازسازی در رایانه است، و در نتیجه، آنچه در ذهن رخ می‌دهد تا درستی یک متن منطقی را بررسی کند، برای رایانه هم قابل‌شبیه‌سازی خواهد بود. پیاده‌سازی این مدل در رایانه، می‌تواند کمک بزرگی به ذهن پر-از-خطای انسان باشد. از آنجا که ذهن، حاصل یک تکامل، بر اساس تغییرات تصادفی، و با هدف فراهم‌کردن شرایط ادامه زیست و تکثیر نسل است، برای بررسی‌های منطقی پیچیده، چندان کارآمد نیست و احتمال خطای آن زیاد است. به علاوه، ذهن، قادر به انجام محاسبات و الگوریتم‌ها، با سرعتی که رایانه‌ها دارند، نیست. از این رو، اگر بتوان مدل درستی از منطق انسانی را در رایانه بازسازی کرد، می‌توان این مدل را، بدون نقص‌های موجود در ذهن، و با هدف بررسی‌های منطقی، نه تکثیر نسل و ادامه زیست، توسعه داد.

مدلی که در این فصل ارائه می‌دهیم، تنها بر اساس اشیاء ذهنی، گزاره‌ها و تابع‌هاست. نام مدل را برای اشاره‌های آینده، منسو<sup>۲</sup> گذاشتۀایم که در اسپرانتو به معنی «ذهن» است. زبانی را که مدل منسو بر اساس آن کار می‌کند زبان منسو خواهیم خواند. در مدل منسو، مجموعه‌ها و اعداد را شبیه‌سازی نکرده‌ایم، زیرا بدون آن‌ها هم می‌توان مدلی حداقلی از

<sup>۲</sup> Menso

منطق ارائه داد که کار می‌کند. البته برای شبیه‌سازی کاربردی، باید از حد مدل‌های حداقلی عبور کرد و مدل‌های کامل‌تری ارائه داد تا ارتباط بهتر و سریع‌تری بین آن‌ها و ذهن انسان ایجاد شود. اما ما در اینجا به دنبال ساختن مدل‌های کامل‌تر و نوشتن برنامه‌های کاربری رایانه‌ای نیستیم، بلکه به دنبال آشنایی خواننده با مدل منطقی ذهن هستیم. ما نرم‌افزاری با نام Edukadoj Menso Mind Logic Simulation اطراحی و ایجاد کرده‌ایم که همراه با این کتاب ارائه می‌شود و کدهای ارائه‌شده در این فصل را می‌توانید با این نرم‌افزار بیازمایید. این نرم‌افزار را، برای اختصار، نرم‌افزار منسو خواهیم خواند.

## شناخت اولیه مدل منسو

در مدل منسو، سه نوع موجودیت ذهنی قابل‌تعریف است:

- ۱) شیء، که منظور همان شیء ذهنی است و با کلمه کلیدی **object** معرفی می‌شود.
- ۲) گزاره، که با کلمه کلیدی **prop** معرفی می‌شود.
- ۳) تابع، که با ترکیبی خاص از کلمه‌های **object** و **prop** معرفی می‌شود.

به عنوان مثال، برای معرفی یک شیء جدید می‌توانیم در نرم‌افزار منسو بنویسیم:

```
new object a;
```

با این کد، شیء **a** ایجاد می‌شود. اما ایجاد\_شدن به چه معناست؟ ایجاد\_شدن به معنای اضافه\_شدن به لیست موجودیت‌های ذهنی است. در مدل منسو، سه لیست داریم:

- ۱) لیست موجودیت‌های ذهنی، که در نرم‌افزار منسو با نام **Things** نشان داده شده است. این لیست شامل اشیاء، گزاره‌ها و توابع (به جز عملگرها) است. این لیست، متشكل از آیتم‌هایی است که هریک، از نام یک موجودیت ذهنی به علاوه نوع آن موجودیت تشکیل شده است.
- ۲) لیست گزاره‌های درست که در نرم‌افزار منسو با نام **Truths** نشان داده شده است.
- ۳) لیست عملگرها، که در حقیقت، ادامه‌ای از لیست موجودیت‌های ذهنی است و فعلًاً می‌توانید آن را نادیده بگیرید.

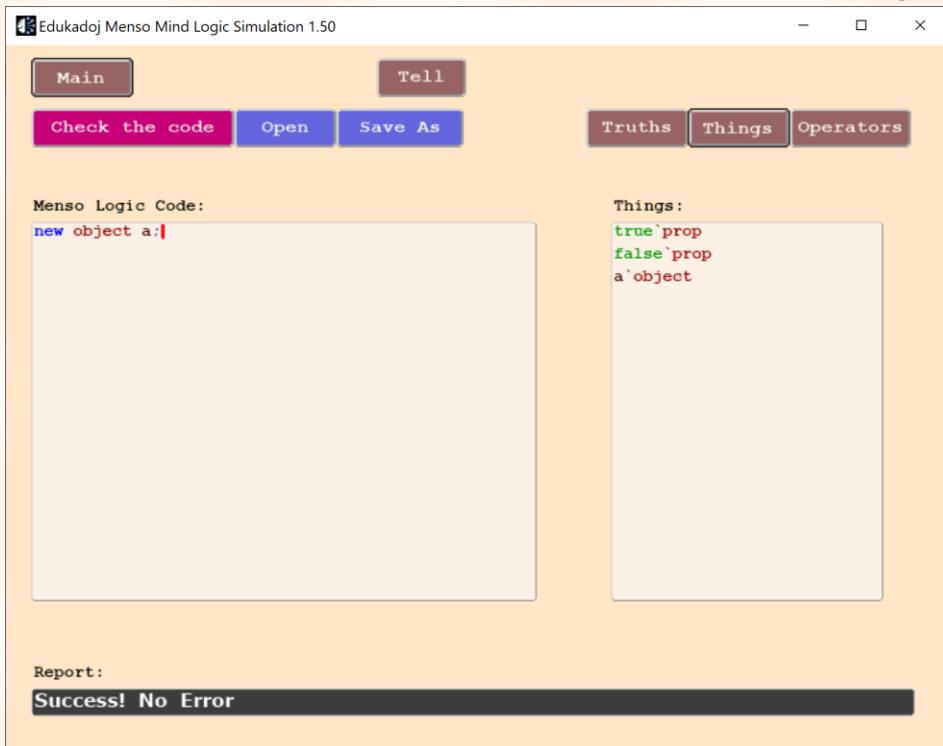


## ایجاد اشیاء و گزاره‌های ساده

کد:

```
new object a;
```

را در قسمت نوشتاری Meno Logic Code در نرم‌افزار منسو، وارد کنید و کلید Check the code را فشار دهید. سپس، کلید Things را فشار دهید تا لیست موجودیت‌های ذهنی نمایان شود:



می‌بینید که سه موجودیت ذهنی در این لیست قابل مشاهده است که آخرین آن، موجودیتی با نام a، و نوع object است. دو موجودیت true و false که نوع prop دارند (یعنی گزاره هستند)، به صورت پیش‌فرض، در لیست موجودیت‌های ذهنی قرار دارند و برای راحت‌تر شدن کدنویسی مورد استفاده‌اند.

حال، اگر به کد قبلی، خطی دیگر بیفزاییم و یک گزاره جدید معرفی کنیم:

```
new object a;
new prop p;
```

لیست موجودیت‌ها، به این صورت در می‌آید:

```
true`prop
false`prop
a`object
p`prop
```

توجه کنید که هر موجودیت، یک نام و یک نوع دارد. برای نمونه، در لیست اخیر، موجودیتی با نام `p` و نوع `prop` وجود دارد. در لیست‌هایی که در نرم‌افزار منسو ارائه می‌گردند، هر آیتم در خطی جداگانه قرار دارد.

توجه کنید که گزاره‌هایی که به لیست موجودیت‌ها اضافه می‌شوند، نه درستند و نه نادرست، بلکه بلاتکیفند تا زمانی که درستی یا نادرستیشان معلوم گردد. درستی یک گزاره وقتی معلوم می‌گردد که خودش در لیست گزاره‌های درست قرار بگیرد (یا بدیهی باشد) و نادرستی آن وقتی معلوم می‌گردد که نقیضش در لیست گزاره‌های درست قرار بگیرد (یا نادرستیش بدیهی باشد). در آینده، در این باره بیشتر سخن خواهیم گفت.

## ایجاد ویژگی و تابع

هرچند در نسخه‌ای از مدل منسو که در اینجا ارائه می‌گردد، مجموعه، به صورت خاص، معادلی ندارد، اما، تابع و در نتیجه، ویژگی قابل‌تبیین است، ویژگی هم با توجه به آنچه در فصل قبل گفته‌یم، همان مجموعه است. به هر حال، چون تمهید خاصی برای مجموعه در مدل منسو نداریم، بازآفرینی آن کمی دردرس دارد.

می‌دانیم که ویژگی، یک موجودیت است که نیازمند آرگومان‌هایی به آن داده شد، می‌توان از طریق ویژگی، به یک گزاره رسید. برای نمونه، اگر `p` یک ویژگی باشد که به یک آرگومان که شیء ذهنی است، نیاز دارد، و اگر `x` یک شیء ذهنی باشد، آنگاه `(x)p` یک گزاره خواهد بود. ویژگی `p`، آرگومان `x` را می‌گیرد و گزاره `(x)p` را، به عنوان خروجی به ما می‌دهد. اگر بخواهیم در مدل منسو، ویژگی `p` را تعریف کنیم، می‌توانیم بنویسیم:



```
new prop p(object);
```

در این کد، `object`، نوع آرگومانی را که ویژگی `p` می‌پذیرد، مشخص می‌کند. همچنین، `prop` نوع خروجی `p` را به عنوان یک تابع تعیین می‌کند؛ خروجی ویژگی‌ها همواره `prop` است. خروجی به معنی محصول نهایی (مثلًا `(x)p`) پس از دادن آرگومان است. کد زیر را در نظر بگیرید:

```
new object x;
new prop p(object);
```

با این کد، `p` یک گزارهٔ خواهد بود، زیرا `x`، یک شیء و `p` یک ویژگی با تنها یک آرگومان است که نوع این آرگومان هم شیء است؛ پس اگر آرگومان مناسب `x` را به `p` بدهیم، گزارهٔ `(x)p`، به عنوان «خروجی»، حاصل خواهد شد.

می‌توان ویژگی‌هایی با دو آرگومان تعریف کرد:

```
new object x;
new object y;
new prop p(object, object);
```

در اینجا، `p` یک ویژگی با دو آرگومان است. با کد بالا، `(x,y)p` یک گزارهٔ است.

به طور کلی، تابع‌ها به همین شکل تعریف می‌شوند، اما لزوماً خروجی آن‌ها، `object` نیست. برای مثال، می‌توانیم یک تابع را به این صورت تعریف کنیم:

```
new object f(object);
```

با این تعریف، `f`، یک تابع خواهد بود که یک آرگومان با نوع `object` می‌گیرد و یک `object` دیگر به وجود می‌آورد. به عنوان نمونه، اگر `x` نوع `object` داشته باشد، `(x)f` نیز یک `object` خواهد بود. اگر کد بالا را در نرم‌افزار منسو وارد کنید و کلید مربوط به بررسی (Check the code) را فشار دهید، لیست موجودیت‌ها به این صورت خواهد شد:

```
true`prop
false`prop
f`object*(object)
```

که در اینجا، نوع `f`، به صورت `object*(object) object` ثبت شده است. پس، نوع `f`، نه `prop` و نه `object` است، بلکه نوعی تابعی با آرایش مشخص است. به جای:

```
new object f(object);
```

می‌توان نوشت:

```
new object*(object) f;
```

تا نوع `f`، به طور کامل قبل از آن آورده شود.

تابع‌ها می‌توانند یک یا بیشتر آرگومان داشته باشد. آرگومان یا خروجی هر تابع، می‌تواند هر یک از نوع‌های `object` یا `prop` را داشته باشد. برای مثال، کد زیر را در نظر بگیرید:

```
new object x;
new prop p;
new object h(object, prop);
```

با این کد، `(x,p)` یک `object` خواهد بود. توجه کنید که آرگومان دوم `h`، باید نوع `prop` داشته باشد.

نوع آرگومان یک تابع می‌تواند نوعی تابعی باشد. کد زیر را در نظر بگیرید:

```
new object f(object);
new object g(object*(object));
```

با این کد، `g` یک تابع است که یک آرگومان دارد. اما نوع این آرگومان، باید `object*(object)` باشد، یعنی این آرگومان باید خود تابع باشد. با کد فوق، `(f,g)` یک `object` خواهد بود.

حتی خروجی تابع‌ها هم می‌تواند یک نوع تابعی داشته باشد:

```
new object*(object) f(prop);
```

با این تعریف، اگر `p` نوع `prop` داشته باشد، `(p)` موجودیتی خواهد بود که خود یک تابع است با نوع `object*(object)`. پس اگر `x` موجودیتی با نوع `object` باشد، `(x,p)` یک `object`



موجودیتی با نوع `object`\*(`object`)\*(`prop`) خواهد بود. نوع `f`، `object`\*(`object`)\*(`prop`) است و به جای:

```
new object*(object) f(prop);
```

می‌توان نوشت:

```
new object*(object)*(prop) f;
```

روشن است که موجودیت‌هایی که نوع `object` یا `prop` دارند، معادل‌هایی طبیعی در ذهن انسان دارند، حتی تابع‌ها هم می‌توانند معادل‌هایی طبیعی باشند. اما وقتی نوع خروجی یا ورودی، تابعی باشد، به سختی می‌توان گفت ذهن انسان، به طور طبیعی با چنین موجودیت‌های ذهنی‌ای آشناست. در حقیقت، قرار نیست، مدل‌های رایانه‌ای ذهن، لزوماً همان محدودیت‌های ذهن انسان را داشته باشند، بلکه با توجه به حافظه و سرعت بیشتر، می‌توان توسعه‌هایی در آن‌ها پدید آورد تا یک قاعده، به دیوارهای محدودیت‌هایی شبیه به محدودیت‌های ذهن انسان بر نخورد. با وجود این، آنچه در عمل به کار می‌آید، بیشتر بخش‌هایی از مدل است که با ذهن ما سازگارتر است. برای مثال، در عمل، هیچ‌گاه نیاز نداریم با موجودیتی که نوع:

```
object*(object)*(prop)*(object, object, object*(object))* (prop)
```

دارد کار کنیم، هرچند که نرم‌افزار منسو، با چنین نوعی هیچ مشکلی ندارد. از سوی دیگر، اگر مدل ما، سازگار با جهان فیزیکی پیرامون ما باشد، ممکن است نادیده‌گرفتن محدودیت‌های ذهن، ما را در شناخت طبیعت قوی‌تر کند.

## گزاره‌های درست و نادرست

در مدل منسو گزاره‌ها در ابتدا بلا تکلیفند تا آن که با قوانین مشخص منطقی، خودشان یا نقیضشان به «لیست گزاره‌های درست» اضافه شود. در نرم‌افزار منسو، باید با دستورات خاص و از طریق به\_کار\_بردن ابزارهای منطقی موجود، گزاره‌های بلا تکلیف یا نقیضشان را به لیست گزاره‌های درست اضافه کرد.

اگر یک گزاره در لیست گزاره‌های درست قرار بگیرد، آن گزاره، در مدل منسو، «درست» محسوب خواهد شد. در مدل منسو، یک گزاره، نادرست محسوب می‌شود تنها اگر نقیضش درست باشد، یعنی در لیست گزاره‌های درست قرار داشته باشد.

در مدل منسو یک گزاره با نام `true` داریم که `خود به_خود` در لیست گزاره‌های درست قرار دارد. وجود این گزاره تنها برای ساده\_شدن برخی کدهاست. همچنین، گزاره‌ای با نام `false` داریم که نقیضش در لیست گزاره‌های درست قرار دارد. در مدل منسو، نقیض گزاره `p` با `!p` نشان داده می‌شود. اگر دگمه `Truths` را در نرمافزار منسو فشار دهید، لیست گزاره‌های درست نمایان خواهد شد و گزاره‌های `true` و `false` را در این لیست مشاهده خواهید کرد.

ذهن انسان تنها با تجربیات مکرر از طریق حواس پنجگانه، و بدون استدلالی درونی، تصمیم می‌گیرد گزاره‌هایی را درست بداند. برای نمونه، گزاره «خورشید نورانی است» یا «شب تاریک است» یا «زمین جاذبه دارد». این گزاره‌ها، هیچ مبنای استدلالی‌ای ندارند و تنها بر اساس مشاهدات درست تشخیص داده می‌شوند. اما در مدل منسو، تنها بخش استدلالی ذهن شبیه‌سازی می‌شود و در این مدل، حواس پنجگانه‌ای وجود ندارد که از طریق آن حواس، `خود به_خود`، تصمیم به درست\_دانستن برخی گزاره‌ها بگیرد و علاوه بر آن، در مدل منسو، تمام موجودیت‌ها محض هستند و ارتباطی بین آن‌ها و آنچه از طریق حواس پنجگانه دریافت می‌گردد، برقرار نیست. برای مثال، اگر بنویسیم:

```
new object Sun;
```

آنگاه `Sun`، به عنوان یک شیء ذهنی، به ذهن شبیه‌سازی شده نرمافزار اضافه خواهد شد؛ اما این شیء، `ما به_ازای خارجی` (مثلاً خورشید) ندارد و تنها یک موجودیت محض است. اشیاء ذهنی انسان، اغلب `ما به_ازای خارجی` دارند؛ هرچند که ذهن می‌تواند اشیاء محض را نیز به وجود آورد و برای ایجاد شیء لزوماً نیازمند حرک خارجی نیست. بنابراین، نیاز داریم گزاره‌هایی را به صورت مصنوعی و بدون استدلال به لیست گزاره‌های درست اضافه کنیم، همانگونه که ذهن انسان، بر اساس مشاهدات یا تکرارها، گزاره‌هایی را، بدون استدلال، درست محسوب می‌کند. برای اضافه\_کردن مصنوعی و



مستقیم گزاره‌ها در نرم‌افزار منسو از دستور `add` استفاده می‌کنیم. به عنوان نمونه، اگر کد زیر را اجرا کنید:

```
new prop p;
add p;
```

و سپس به لیست Truths در نرم‌افزار منسو نگاه کنید، خواهید دید که `p` به این لیست اضافه شده است. همچنین کد:

```
new prop p(object);
new object x;
add p(x);
```

گزاره (`x`) `p` را به لیست گزاره‌های درست اضافه می‌کند.

به این ترتیب، با `add` گزاره‌های اولیه به لیست گزاره‌های درست اضافه می‌شوند و گزاره‌های ثانویه، باید با عملیات منطقی، که در ادامه تبیین خواهند شد، به لیست اضافه گردند. عملیات منطقی، از گزاره‌های اولیه استفاده می‌کند و طبق قانون، گزاره‌های ثانویه را به لیست گزاره‌های درست اضافه می‌کند.

`add`، جایگزین راههای غیراستدلالی اثبات در ذهن انسان است. اما از آنجا که قرار نیست همواره بتوان هر گزاره‌ای را با استفاده از آن اضافه کرد و تنها برای اضافه\_کردن گزاره‌های اولیه، باید از `add` استفاده کرد، در نرم‌افزار منسو، دستور `noadds` قاب\_استفاده است که پس از آن، دیگر نمی‌توان `add` را برای اضافه\_کردن گزاره‌ها به لیست Truths به کار برد:

```
new prop p(object);
new object x;
add p(x);
noadds;
```

در واقع، مفهوم `noadds` آن است که از این به بعد، همه گزاره‌ها تنها با عملیات و استدلال منطقی ثابت می‌شوند؛ ثابت\_شدن به معنی اضافه\_شدن به لیست گزاره‌های درست از طریق استدلال‌های منطقی است.

برای این که بدانیم یک گزاره در مدل منسو درست است یا نه، می‌توانیم به لیست Truths نگاه کنیم و ببینیم، گزاره در لیست وجود دارید یا خیر. راه بهتر استفاده از دستور check است:

```
new prop p;
add p;
check p;
```

اگر گزاره‌ای که مقابله کلیدی check قرار گرفته است، درست نباشد، نرم‌افزار، یک پیام خطا صادر می‌کند و از بررسی ادامه دستورات صرف نظر می‌کند. check برای بررسی درست\_بودن، لیست گزاره‌های درست را می‌گردد تا ببینید آیا گزاره مورد نظر در این لیست وجود دارید یا خیر. در مدل منسو، درست\_بودن دقیقاً به معنی موجود\_بودن در لیست گزاره‌های درست نیست، بلکه، برخی گزاره‌های بدیهی نیز، هرچند در لیست Truths نباشند، درست انگاشته می‌شوند؛ برای نمونه، گزاره  $x = x$ ، آنقدر بدیهی است که نرم‌افزار منسو حتی اگر آن را در لیست گزاره‌های درست پیدا نکند، باز هم آن را درست می‌داند. یکی از ویژگی‌های دستور check این است که حتماً گزاره مقابله را، اگر درست بود، اما در لیست نبود، به لیست اضافه کند. برای نمونه، اگر کد:

```
new object x;
check x=x;
```

را بررسی کنید، خواهید دید که  $x = x$  به لیست گزاره‌های درست اضافه می‌گردد. جالب است بدانید که بررسی کد:

```
new object x;
add x = x;
```

منجر به خطا می‌گردد، زیرا add، گزاره‌ایی را که از قبل درستی آنها مورد تأیید است، به لیست Truths اضافه نمی‌کند. به این ترتیب، check، تنها برای بررسی گزاره‌ها نیست، بلکه برای اضافه\_کردن صریح آنها در لیست گزاره‌های درست نیز هست. وجود یک گزاره در لیست Truths می‌تواند برای قوانین استدلالی و عملیات خودکار نرم‌افزار مفید باشد. ذهن انسان نیز می‌داند که هر چیز برابر با خودش است، اما می‌تواند یک تساوی



خاص را مورد توجه خاص قرار دهد. قرار\_گرفتن در لیست Truths، مانند همین توجه خاص است.

ممکن است این پرسش مطرح شود که چرا نرمافزار مسنون، همه گزاره‌های درست را چه بدبیهی و چه غیر\_بدبیهی به لیست Truths اضافه نمی‌کند؟ پاسخ، محدودیت‌های سرعت و حافظه رایانه است. اگر قرار باشد نرمافزار، خود بگردد و تمام گزاره‌های درست را به طور خودکار پیدا و اضافه کند، درگیر حلقه‌های نامحدود اجرای دستورات می‌گردد. اگر هم قرار باشد مانند ذهن انسان، به طور تصادفی برخی گزاره‌ها را پیدا و اضافه کند، نتیجه چندان جالب نخواهد بود. به این ترتیب، بهتر است ما به عنوان کاربر نرمافزار، در مورد اضافه\_شدن برخی گزاره‌ها به Truths تصمیم بگیریم.

## عملگرهای ! و & و |

در مدل منسو، اگر `p` یک گزاره باشد، `p!` نیز یک گزاره است که نقیض گزاره `p` محسوب می‌گردد. همچنین، اگر `p` و `q` گزاره باشند، `q & p` و `p & q` نیز گزاره هستند. & شبیه‌سازی «و» (پیوند عمومی) است و | شبیه‌سازی «یا» (پیوند وجودی). در واقع، گزاره `q & p` گزاره‌ای است که وقتی درست محسوب می‌شود که «`p` و `q`» درست باشند و `q | p` گزاره‌ای است که وقتی درست محسوب می‌شود که حداقل یکی از «`p` یا `q`» درست باشد.

**مثال:** کد زیر را در نظر بگیرید:

```
new prop p;
add p;
check !!p;
```

بررسی این کد، خطایی در بر ندارد. در کد بالا، `p` به عنوان گزاره‌ای درست معرفی شده است. سپس درستی `p!!` بررسی شده است و نرمافزار منسو درستی آن را تشخیص داده است. زیرا، `p!!`، نقیض نقیض `p` است و ارزش نقیض نقیض یک گزاره با خود آن گزاره یکسان است. به طور کلی، نرمافزار منسو `p!!` را به صورت `p` می‌بینید.



**مثال:** کد زیر را در نظر بگیرید:

```
new prop p;  
new prop q;  
add p;  
add q;  
check p & q;
```

این کد هم بدون خطاست. در این کد، `p` و `q`، به عنوان دو گزاره درست معرفی می‌شوند. سپس، درستی `q & p` بررسی می‌گردد. چون هر دوی `p` و `q` درست هستند، مدل منسو، `p & q` را درست تشخیص می‌دهد.



**مثال:** کد زیر را در نظر بگیرید:

```
new prop p;  
new prop q;  
add p;  
check p | q;
```

این کد هم بدون خطاست. در این کد، `p` و `q` به عنوان دو گزاره معرفی شده‌اند که یکی از آن‌ها درست است. سپس، درستی `q | p` بررسی می‌شود. چون حداقل یکی از `p` یا `q` درست است، مدل منسو، `q | p` را درست تشخیص می‌دهد.



عملگرهای `&` و `|` برای هر تعداد محدود از گزاره‌ها قابل‌استفاده‌اند. برای نمونه برای سه گزاره:

```
new prop p;  
new prop q;  
new prop r;  
add p | q | r;
```

اما نرم‌افزار منسو `r | q | p` را معادل با `(p | q) | r` می‌گیرد و معنای مستقلی برای آن در نظر نمی‌گیرد. به همین سبب، در حالی که کد:

```
new prop p;  
new prop q;
```





```
new prop r;
add p | q;
check p | q | r;
```

خطای ندارد و توسط نرم‌افزار به سرعت بررسی و تأیید می‌شود، بررسی کد:

```
new prop p;
new prop q;
new prop r;
add q | r;
check p | q | r;
```

وقت و محاسبات بسیار بیشتری را به نرم‌افزار تحمیل می‌کند. اما باید توجه داشت که هرچه استدلال‌های نرم‌افزار منسو بیشتر به صورت خودکار انجام شوند، عملیات آن زمان‌گیرتر خواهد شد و در کدهای طولانی ممکن است، نرم‌افزار بلاستفاده گردد. از این رو ممکن است گاه نیاز به کاهش هوشمندی نرم‌افزار باشد.

## گزاره شرطی و سورها

گزاره شرطی «اگر  $p$  آنگاه  $q$ » در زبان منسو، به صورت  $q \text{ if } p$  نوشته می‌شود. همچنین، گزاره «برای هر شیء  $x$ ، گزاره  $p(x)$  درست است»، به صورت  $\text{object } x \text{ } p(x)$  و گزاره «برای حداقل یک شیء  $x$ ، گزاره  $p(x)$  درست است»، به صورت  $\text{some object } x \text{ } p(x)$  نوشته می‌شود.  
مثال: به کار بردن گزاره شرطی:

```
new prop p;
new prop q;
add (if p)q;
add p;
check q;
```

در کد بالا، گزاره  $q \text{ if } p$  به لیست **Truths** (یعنی لیست گزاره‌های درست) اضافه شده است. همچنین، گزاره  $p$  نیز، به این لیست اضافه شده است. چون هر دوی این گزاره‌ها در لیست هستند، گزاره  $q$ ، گزاره‌ای درست در نظر گرفته شده است.  $q \text{ if } p$  گزاره‌ای است که درست\_بودنش به مدل اجازه می‌دهد که در صورت درست\_بودن  $p$ ،  $q$  را درست به

حساب بیاورد. ذهن انسان نیز، با گزاره‌های شرطی، رفتار مشابهی دارد و مدل منسو، تنها تلاش می‌کند که عملکردی مشابه ذهن انسان داشته باشد.



**مثال:** به کار بردن سورها:

```
new prop p(object);
add (all object x)p(x);
check (all object y)p(y);
```

در کد بالا، `p`، یک ویژگی است. `(all object x)p(x)` در مدل منسو، معادل گزاره «برای هر شیء `x`، گزاره `p(x)` درست است» در ذهن انسان است. این گزاره، به اضافه شده است. در `(x)p(x)`، `x` را، که متغیر سور نام دارد، می‌توان با هر نام دیگری جایگزین کرد. اگر به لیست `Truths` نگاه کنید خواهید دید که به جای آن `(all object y)p(y)` همچو `(all object x)p(x)` با `(all object y)p(y)` تفاوتی ندارد و کد بالا این را تأیید می‌کند.



**مثال:** متغیر سور می‌تواند هر نوعی داشته باشد:

```
new prop p(object*(object));
add (some object f(object))(p(f));
```

در سور `(p(f))`، `object*(object)`، متغیر سور نوع `(some object f(object))` دارد.



حول گزاره پایانی، در گزاره‌های شرطی و سورها می‌توان پرانتر گذاشت و یا نگذاشت. اما پرانتر\_گذاشتن، سبب می‌شود که محدوده گزاره سور مشخص گردد. به عنوان مثال، گزاره `(if p)(q) & r` با `(if p)(q) & r` مقاوت است؛ گزاره اول، معادل است با `((if p)(q) & r)`.

## pick دستور

کد زیر را در نظر بگیرید:



```
new prop p(object);
new object x;
add (all object t)p(t);
```

در اینجا، چون معادل `(all object t)p(t)` در ذهن انسان گزاره «برای هر شیء `t`، `p(t)` درست است» است و مدل منسو شبیه‌سازی‌ای از ذهن انسان است، انتظار داریم، این مدل، پس از کد فوق، گزاره `(x)p` را درست بداند، چون `x` یک شیء است و قاعده‌ای `(x)p` باید درست باشد. اما اگر پس از کد فوق، در نرم‌افزار منسو، بنویسیم:

```
check p(x);
```

و کد را بررسی کنیم، نرم‌افزار خطای می‌گیرد؛ زیرا بررسی خودکار سورها، می‌تواند زمان‌گیر باشد و نسخه فعلی نرم‌افزار قادر به این بررسی نیست و باید برای بررسی مناسب، نرم‌افزار را با استفاده از دستور `pick` راهنمایی کرد. کد زیر مشکل را برطرف کرده است:

```
new prop p(object);
new object x;
add (all object t)p(t);
pick x in (all object t)p(t);
check p(x);
```

خط:

```
pick x in (all object t)p(t);
```

به نرم‌افزار می‌گوید که «`x` را به جای `t` در `(all object t)p(t)` قرار بده تا `p(x)` به دست آید». نرم‌افزار، با خواندن این خط، این کار را منطقی تشخیص می‌دهد و `p(x)` را به عنوان گزاره‌ای درست به لیست گزاره‌های درست می‌افزاید. توجه کنید که نرم‌افزار، تنها کدهایی را تأیید می‌کند که با منطق تعریف شده در مدل منسو سازگار باشند.

**مثال:** استفاده از `pick` برای سور عمومی:

```
new prop p(prop);
new prop q;
add (all prop e)p(e);
pick p(q) in (all prop e)p(e);
```

```
check p(p(q));
```

در کد بالا، چون `(all prop e)p(e)` یک سور عمومی درست است و `p(q)` نیز یک `prop` است، باید `(p(p(q))p(p(q))` درست باشد. با دستور `pick` به نرمافزار کمک کرده ایم که به درستی `p(p(q))` پی ببرد.



کد زیر را در نظر بگیرید:

```
new prop p(object);
new object a;
add p(a);
```

پس از این کد، `p(a)` گزاره‌ای درست است. بنابراین، با منطق انسانی می‌توان گفت حداقل یک شیء `x` وجود دارد که `(x)p(x)` درست باشد، و این شیئی که وجود دارد، همان `a` است. پس به زبان منسو، `(some object x)p(x)` باید گزاره‌ای درست باشد، اما نرمافزار منسو از تأیید:

```
check (some object x)p(x);
```

عاجز است مگر آن که با دستور `pick` آن را یاری کنیم. برای این کار کافی است بنویسیم:

```
pick a in (some object x)p(x);
```

که یعنی «در `(some object x)p(x)`، `x` را همان `a` بگیر تا به درستی سور پی ببری». بنابراین، کد زیر، بدون اشکال مورد تأیید نرمافزار منسو است:

```
new prop p(object);
new object a;
add p(a);
pick a in (some object x)p(x);
check (some object x)p(x);
```

به این ترتیب، `pick` دو عمل متفاوت (تقریباً برعکس یکدیگر)، برای سورهای عمومی و وجودی انجام می‌دهد.





دستور `pick` را می‌توان برای گزاره‌های شرطی هم به کار برد هرچند لزومی ندارد، زیرا نرمافزار منسو برای گزاره‌های شرطی نیازی به راهنمایی با استفاده از `pick` ندارد. با این حال، امکان این راهنمایی هست:

```
new prop p;
new prop q;
add (if p)q;
add p;
pick p in (if p)q;
check q;
```

در کد بالا، خط:

```
pick p in (if p)q;
```

به نرمافزار منسو می‌گوید: «با توجه به این که `p` و `q` هر دو گزاره‌هایی درستند، `q` را به لیست گزاره‌های درست اضافه کن». اما نرمافزار منسو نیازی به این راهنمایی ندارد و اگر خط فوق حذف شود، کد بدون اشکال تأیید می‌گردد.

دستور `pick` را می‌توان برای گزاره‌هایی که با `&` درست شده‌اند (گزاره‌های پیوند عمومی) و گزاره‌هایی که با `|` درست شده‌اند (گزاره‌های پیوند وجودی) نیز به کار برد و عین حال، به کار بردن آن ضرورتی ندارد. کد زیر را در نظر بگیرید:

```
new prop p;
new prop q;
add p & q;
pick p in p & q;
check p;
```

در این کد `pick` برای یک گزاره پیوند عمومی به کار رفته است. خط:

```
pick p in p & q;
```

به نرمافزار منسو می‌گوید: «گزاره `p`، برابر با یکی از دو گزاره اطراف `&` در `q` است و به علاوه، `q` & `p` گزاره‌ای صحیح است، پس `p` را به لیست گزاره‌های درست اضافه کن».

به این ترتیب به نرم‌افزار منسو راهنمایی می‌شود برای اضافه\_کردن p به Truths. اما نرم‌افزار منسو نیازی به این راهنمایی ندارد و خود می‌تواند درستی p را نتیجه بگیرد.

کد زیر نیز حاوی به\_کار\_بردن pick برای یک گزاره پیوند وجودی است:

```
new prop p;
new prop q;
add p | q;
add !q;
pick !q in p | q;
check p;
```

در کد بالا، خط:

```
pick !q in p | q;
```

به نرم‌افزار منسو می‌گوید: «حال که q | p درست است اما q درست نیست، حتماً درست است و بنابراین p را به لیست گزاره‌های درست اضافه کن». این تنها یک راهنمایی برای نرم‌افزار است که البته نرم‌افزار به آن احتیاجی ندارد و بدون این راهنمایی هم می‌تواند درستی p را تأیید کند.

به این ترتیب، pick یک دستور راهنمایی است برای ذهن مصنوعی نرم‌افزار منسو، اما گاه این ذهن، آن قدر باهوش هست که نیازی به این راهنمایی نداشته باشد. اما باید توجه کرد که باهوشتر\_کردن این ذهن مصنوعی، اغلب همراه با کندتر\_شدن آن است و گاه نیاز است که از هوشمندی آن کم کنیم تا بتواند در زمان معقولی متن‌های منسو را بررسی کند. باهوش\_بودن نرم‌افزار به این معنی است که دسته‌ای از راهنمایی‌های مختلف را در پشتپرده می‌آزماید تا ببینید آیا می‌تواند به نتیجه مطلوب دست یابد یا خیر: در این فرآیند، گاه، ذهن نرم‌افزار منسو قوی‌تر از ذهن انسان عمل می‌کند، و گاه، به دلیل محدودیت‌های رایانه‌ای و ضعف طراحی برنامه، ضعیفتر.

نرم‌افزارهای رایانه‌ای هم معمولاً دارای روند تکامل تدریجی هستند و به تدریج تکامل می‌یابند و خطاهایشان کم‌تر و سرعتشان بیشتر می‌گردد. می‌توان گفت نرم‌افزار منسو در این مسیر در ابتدای راه است و وجود ضعف‌ها در آن طبیعی است.



## عملگرها

پیش‌تر با چند عملگر آشنا شدیم. عملگرهای `=`, `&`, `!` و `=`. عملگرهای، در حقیقت، همان توابع هستند، اما شکل به\_کار\_گیری آن‌ها متفاوت است. عملگرها در زبان منسو، یا یک\_آرگومانی‌ند مثل عملگر `! یا دو_آرگومانی` مثل `&.` می‌توان عملگرهای جدیدی تعریف و از آن‌ها استفاده کرد. شکل تعریف بسیار مشابه شکل تعریف تابع‌هاست:

```
new operator object +(object, object);
new prop p(object);
new object x;
add p(x + x);
```

در کد بالا، با:

```
new operator object +(object, object);
```

عملگر `+` معرفی شده است که درست مانند یک تابع است که دو آرگومان با نوع `object` می‌گیرد و نوع خروجی آن نیز `object` است. در واقع `+` یک تابع است اما به جای این که بنویسیم `(x, x) +` باید بنویسیم `x + x.` تفاوت دیگری وجود ندارد.

اگر عملگر دو\_آرگومانی `+` را مثل کد فوق تعریف کنیم، نوشتن `x + x` اشکالی ندارد. اما نرم‌افزار منسو آن را معادل با `x + (x + x)` خواهد گرفت.

عملگر یک\_آرگومانی نیز به شکل مشابه تعریف می‌شود و طرز استفاده از آن، مانند طرز استفاده از «`!`» است:

```
new operator prop @ (object);
new object x;
add @x;
```

نوع و آرگومان‌ها یا خروجی، در تعریف یک عملگر اهمیتی ندارد، تنها محدودیت‌ها این است که تعداد آرگومان‌ها باید یک یا دو باشد و نماد انتخابی، تنها یکی از نمادهای `~`, `%`, `@`, `$`, `+`, `-`, `^`, `<` و `>` باشد. به علاوه، باید توجه داشت در عملگرهای دو\_آرگومانی، ترتیب آرگومان‌ها اهمیت دارد.

آیا عملگر در زبان منسو معادلی در ذهن انسان دارد؟ واقعیت این است که نوشتار، قراردادی است که طی تکامل، در ذهن انسان به وجود نیامده است و تنها عارضه‌ای قابل استفاده از آن است و بنابراین، تغییرات نوشتاری از تابع به عملگر و مشابه اینها، تنها قراردادی هستند و مبنای محکمی در ذهن ندارند. ذهن انسان توانایی آموختن این قوانین نوشتاری ساده را دارد. اما می‌توان گفت عملگرها و طرز استفاده از آنها با «زبان» سازگارترند تا توابع و طرز استفاده از آنها. بنابراین، ذهن انسان، می‌تواند احساس راحتی بیشتری با عملگرها داشته باشد.

### تساوی

تساوی یک از قابلیت‌های ذهن انسان است. اما تساوی مطلق هیچ کاربردی ندارد و از این رو، ذهن قادر به بررسی تساوی‌هایی را با شدت ضعیفتر است. برای مثال، دو لیوان یک شکل را مساوی می‌گیرد، هرچند لااقل، این دو لیوان، از نظر محل استقرار متفاوند. در ریاضیات، معمولاً برای تساوی‌های ضعیف، از لفظ «هم‌ارزی» استفاده می‌کنند، اما در آنجا نیز تساوی مطلقی در کار نیست و آنچه هست، یک قرارداد است. ذهن‌های انسانی ممکن در تنظیم شدّت تساوی چهار مشکل و تناقض باشند. به عنوان نمونه، ممکن است شخصی، همه «کارگرها» را یکسان و مساوی بداند، اما شخص دیگری آن‌ها را به دو دسته «ماهر» و «مبتدی» تقسیم کند و همه را مساوی نداند. ممکن است ذهن برخی افراد در انتخاب شدّت مناسب برای یک تساوی، چهار نوعی خامی باشد و نتواند شدّت مناسبی برای تساوی انتخاب کند. برخی ممکن است مدعی شوند که فلان کتاب آنقدر دقیق است که اگر در جایی از آن واژه «نادرست» نوشته شده باشد و در جای دیگر از واژه «غلط» استفاده شده باشد، این دو واژه با هم یکسان نیستند و بار معایی متفاوتی دارند و برخی دیگر بگوند آن کتاب آنقدر دقیق نیست که بین این دو واژه بتوان تفاوتی قائل شد.

حال، چگونه باید تساوی را بین موجودیت‌های ذهنی در نظر گرفت و با چه شدّتی؟ در پاسخ باید گفت، ظاهراً دلیل ایجاد مفهوم تساوی در ذهن انسان، امکان جایگزینی است. وقتی در جایی، یک چیز با چیز دیگر قابل‌جایگزین‌شدن باشد، آن دو با هم مساوی در نظر گرفته می‌شود، زیرا هر یک از آن‌ها که مورد استفاده باشد، تفاوتی ایجاد نمی‌گردد. در مدل منسو نیز تساوی بر همین اساس تعریف شده است.



در مدل منسو، تساوی یک ویژگی با دو آرگومان است. در واقع، تساوی یک عملگر دو آرگومانی با خروجی `prop` است. در این مدل، تساوی تنها یک شدت دارد و این شدت به حدی است که ظاهر نوشتاری موجودیت‌ها و محل آن‌ها در متن، هیچ اهمیّتی در این تساوی ندارند. برای مثال، دو شیء با نام یا ظاهر متفاوت، می‌توانند مساوی باشند. علت عدم اهمیّت، ظاهراً این است که نوشتار، تنها راهی برای ثبت اطلاعات و انتقال است و تا حد امکان باید از معیار بودن آن پرهیز کرد.

در مدل منسو، گزاره‌های درست، همه با هم مساوی در نظر گرفته می‌شوند. همین‌طور گزاره‌های نادرست نیز، همه با هم مساوی در نظر گرفته می‌شوند. برای نمونه، کد زیر خطای ندارد:

```
new prop p;
add p;
check p = true;
check !p = false
```

یکی از ویژگی‌های تساوی این است که اگر دو موجودیت با یک موجودیت واحد مساوی باشند، خود آن دو موجودیت نیز مساوی هستند:

```
new object x;
new object y;
new object z;
add x = z;
add y = z;
check x = y;
```

کد بالا خطای ندارد. در این کد، چون `x` و `y` هر دو مساوی با `z` هستند، خود آنها نیز مساویند و نرم‌افزار منسو، معمولاً به اندازه‌ای هوشمند هست که این را، خود به‌خود تشخیص دهد.

در مدل منسو دو چیز تنها وقتی می‌توانند مساوی باشند که نوع یکسانی داشته باشند. برای مثال، تساوی بین یک `object` و یک `prop` تعریف شده نیست.

همان‌گونه که گفته شد، تفاوت تساوی با هر ویژگی دو آرگومانی، ایجاد امکان جایگزینی است. در مدل منسو، وقتی دو موجودیت با هم مساوی باشد، می‌توان در گزاره‌های درست



یکی را برداشت و دیگری را جایگزین آن کرد. برای این کار، در زبان مسنو، از دستور `replace` استفاده می‌شود. این کد را در نظر بگیرد.

```
new object x;
new object f(object);
new prop p(object);
add p(x);
add x = f(x);
```

در کد فوق `(x)` `p`، گزاره‌ای درست است. `x` نیز با `f(x)` مساوی است. بنابراین، می‌توان در گزاره `(x)` `p`، `x` را برداشت و به جای آن `f(x)` گذاشت بدون این که ارزش گزاره تغییری بکند. پس باید `(x)` `p` گزاره‌ای درست باشد. اما نرم‌افزار منسو، خود\_به\_خود، قادر به تشخیص درستی آن نیست. به همین سبب، می‌توان با دستور `replace` آن را برای این تشخیص راهنمایی کرد و نوشت:

```
replace x with f(x) in p(x);
```

این کد به نرم‌افزار می‌گوید: «`(x)` `p` گزاره‌ای درست است و `x` هم با `f(x)` مساوی است پس در `(x)` `p`، به جای `x`، `f(x)` بگذار تا `(x)` `p` به دست آید و `(x)` `p` را به لیست گزاره‌های درست اضافه کن». بنابراین، کد زیر بدون خطاست:

```
new object x;
new object f(object);
new prop p(object);
add p(x);
add x = f(x);
replace x with f(x) in p(x);
check p(f(x));
```

**مثال:** استفاده از `replace`

```
new object f(object);
add (all object x)f(x)=x;
new object a;
pick a in (all object x)f(x)=x;
check f(a) = a;
replace a with f(a) in f(a) = a;
check f(f(a)) = f(a);
```





```
check f(f(a)) = a;
```

در کد بالا، با `replace` ثابت می‌شود که  $f(f(a)) = f(a)$  و با توجه به این که  $f(a) = a$  نیز در لیست گزاره‌های درست است، نرم‌افزار منسو به این نتیجه می‌رسد که  $f(f(a)) = a$ .



## قوانين استدلال برای عملگرهاي منطقی

در قسمت قبل، ضمن معرفی اجزای زبان در مدل منسو، به معرفی تعدادی از قوانین استدلال هم پرداختیم. در این قسمت، تمرکز ما بر معرفی این قوانین است. پس از آن که گزاره‌های درست اولیه، با استفاده از `add` به `Truths` اضافه شدند، می‌توان با استفاده از این گزاره‌های درست اولیه و قوانین استدلال، گزاره‌های دیگری را به لیست `Truths` اضافه کرد. اما باید توجه کرد که مدل و نرم‌افزار منسو تنها از راه این قوانین اجازه می‌دهند که گزاره‌های جدیدی به لیست `Truths` اضافه کنیم. این قوانین، همه، برگرفته از ذهن منطقی انسان هستند.

### قوانين مربوط به اثبات يا رد گزاره پيوند عمومي

به عملگر & پيوند عمومی می‌گوییم. در مورد این عملگر چند قانون وجود دارد:

(۱) در هر شرایط `q` & `p` و `p` & `q` برابر و یکسانند.

(۲) اگر `p` و `q`، هر دو، گزاره‌ای درست باشند، مدل منسو اجازه دارد که `q` & `p` را گزاره‌ای درست محسوب کند (و آن را به لیست گزاره‌های درست اضافه کند).

(۳) اگر حداقلی یکی از `p` یا `q` نادرست باشد، مدل منسو اجازه دارد که `q` & `p` را نادرست محسوب کند.

(۴) اگر `q` ! `p` درست باشد، مدل منسو اجازه دارد که `q` & `p` را نادرست محسوب کند.

(۵) اگر `q` ! `p` (یا `p` ! `q`) درست باشد، مدل منسو اجازه دارد که `q` & `p` را نادرست محسوب کند.



مدل منسو می‌تواند از این قوانین استدلالی استفاده کند تا گزاره‌های پیوند عمومی یا نقیضشان را به لیست گزاره‌های درست بیفزاید. نرم‌افزار منسو، همه قواعد فوق را بدون راهنمایی اجرا می‌کند.

**مثال:** استفاده از قانون استدلالی (۱):

```
new prop p;  
new prop q;  
  
check p & q = q & p;
```



**مثال:** استفاده از قانون استدلالی (۲):

```
new prop p;  
new prop q;  
add p;  
add q;  
noadds;  
  
check p & q;
```



**مثال:** استفاده از قانون استدلالی (۳):

```
new prop p;  
new prop q;  
add !p;  
noadds;  
  
check !(p & q);
```



**مثال:** استفاده از قانون استدلالی (۴):

```
new prop p;  
new prop q;  
add !p | !q;
```





```
noadds;  
check !(p & q);
```



**مثال:** استفاده از قانون استدلالی (۵):

```
new prop p;  
new prop q;  
add (if p)!q;  
noadds;  
  
check !(p & q);
```



### قوانين مربوط به اثبات یا رد گزارهٔ پیوند وجودی

به عملگر «|» پیوند عمومی می‌گوییم. در مورد این عملگر، چند قانون وجود دارد:  
 (۶) در هر شرایط  $q \mid p$  و  $p \mid q$  برابر و یکسانند.

(۷) اگر حداقل یکی از  $p$  یا  $q$  گزاره‌ای درست باشد، مدل منسو اجازه دارد که  $q \mid p$  را گزاره‌ای درست محسوب کند.

(۸) اگر  $q \mid p$  یا  $if \neg q(p)$  درست باشد (یعنی لاقل یکی از آنها درست باشد)، مدل منسو اجازه دارد  $q \mid p$  را درست محسوب کند.

(۹) اگر هر دوی  $p$  و  $q$  نادرست باشند، مدل منسو اجازه دارد که  $q \mid p$  را نادرست محسوب کند.

(۱۰) اگر  $\neg q \mid p$  درست باشد، مدل منسو اجازه دارد که  $q \mid p$  را نادرست محسوب کند.

(۱۱) اگر  $q \mid p$  یا  $if \neg q(p)$  نادرست باشد، مدل منسو اجازه دارد  $q \mid p$  را نادرست محسوب کند.

**مثال:** استفاده از قانون استدلالی (۶):

```
new prop p;  
new prop q;
```

```
check p | q = q | p;
```



مثال: استفاده از قانون استدلالی (۷):

```
new prop p;  
new prop q;  
add q;  
noadds;  
  
check p | q;
```



مثال: استفاده از قانون استدلالی (۸):

```
new prop p;  
new prop q;  
add (if !p)q;  
noadds;  
  
check p | q;
```



مثال: استفاده از قانون استدلالی (۹):

```
new prop p;  
new prop q;  
add !p;  
add !q;  
noadds;  
  
check !(p | q);
```



مثال: استفاده از قانون استدلالی (۱۰):

```
new prop p;  
new prop q;  
add !p & !q;
```



```
noadds;  
check !(p | q);
```



مثال: استفاده از قانون استدلالی (۱۱):

```
new prop p;  
new prop q;  
add !(if !p)q;  
noadds;  
  
check !(p | q);
```



### قوانين مربوط به اثبات یا رد if

در مورد گزارهایی که با **if** ساخته می‌شوند چند قانون وجود دارد:

- (۱۲) اگر **p** نادرست باشد، مدل منسو مجاز است که **q** (**if p q**) را درست بداند.
- (۱۳) اگر **q** درست باشد، مدل منسو مجاز است که **p** (**if q p**) را درست محسوب کند.
- (۱۴) اگر **!q & p** نادرست باشد، مدل منسو اجازه دارد **q** (**if p q**) را درست به حساب آورد.
- (۱۵) اگر **q | !p** درست باشد، مدل منسو می‌تواند **q** (**if p**) را درست به حساب آورد.
- (۱۶) اگر **!p !q** (**if !q !p**) درست باشد، مدل منسو می‌تواند **q** (**if p**) را درست بداند.
- (۱۷) اگر **if p q1 & if p q2** درست باشند، مدل منسو می‌تواند **if p (q1 & q2)** را درست محسوب کند.
- (۱۸) اگر **if p (q1 & q2)** درست باشد، مدل منسو می‌تواند **if p (q1 & if p q2)** را درست محسوب کند.
- (۱۹) اگر **q | !p** نادرست باشد مدل منسو می‌تواند **q** (**if p**) را نادرست به حساب آورد.

(۲۰) اگر  $p \& q$  درست باشد، مدل منسو اجازه دارد  $p \text{ if } q$  را نادرست به حساب آورد.

(۲۱) اگر  $p \& q$  درست باشند، مدل منسو اجازه دارد  $p \text{ if } q$  را نادرست به حساب آورد.

(۲۲) اگر  $p \text{ if } !q$  نادرست باشد، مدل منسو می‌تواند  $p \text{ if } q$  را نادرست بداند.

(۲۳) اگر  $!(p \text{ if } q)$  درست باشد، مدل منسو می‌تواند  $!q \& p$  را درست بداند.  
(جديد)

مثال: استفاده از قانون استدلالی (۱۲):

```
new prop p;
new prop q;
add !p;
noadds;

check (if p)q;
```



مثال: استفاده از قانون استدلالی (۱۳):

```
new prop p;
new prop q;
add q;
noadds;

check (if p)q;
```



مثال: استفاده از قانون استدلالی (۱۴):

```
new prop p;
new prop q;
add !(p & !q);
noadds;

check (if p)q;
```



مثال: استفاده از قانون استدلالی (۱۵):

```
new prop p;
new prop q;
add !p | q;
noadds;

check (if p)q;
```



مثال: استفاده از قانون استدلالی (۱۶):

```
new prop p;
new prop q;
add (if !q)!p;
noadds;

check (if p)q;
```



مثال: استفاده از قانون استدلالی (۱۷):

```
new prop p;
new prop q1;
new prop q2;
add (if p)q1;
add (if p)q2;
noadds;

check (if p)q1 & q2;
```



مثال: استفاده از قانون استدلالی (۱۸):

```
new prop p;
new prop q1;
new prop q2;
add (if p)(q1) & (if p)q2;
```



```
noadds;  
  
check (if p)q1 & q2;
```



مثال: استفاده از قانون استدلالی (۱۹):

```
new prop p;  
new prop q;  
add !(p | q);  
noadds;  
  
check !(if p)q;
```



مثال: استفاده از قانون استدلالی (۲۰):

```
new prop p;  
new prop q;  
add p & !q;  
noadds;  
  
check !(if p)q;
```



مثال: استفاده از قانون استدلالی (۲۱):

```
new prop p;  
new prop q;  
add p;  
add !q;  
noadds;  
  
check !(if p)q;
```



مثال: استفاده از قانون استدلالی (۲۲):

```
new prop p;  
new prop q;
```



```
add !(if !q)!p;
noadds;

check !(if p)q;
```



## قوانين دیگر

قوانين دیگری نیز وجود دارند که نمی‌توان آن‌ها مستقیماً به گزاره‌های پیوندی و شرطی منتسب دانست:

(۲۴)  $p!!$  در هر شرایط با  $p$  برابر و یکسان است. یعنی اساساً مدل منسو  $p!!$  را به

صورت  $p$  می‌خواند و ثبت می‌کند، نه به صورت  $p!!$ . پس اگر در متن دو نقیص کنار هم باشد، نرم‌افزار منسو آن دو را نمی‌بیند. (در به\_کار\_بردن دستور `replace` باید مراقب این قانون بود)

(۲۵) اگر  $p \& q$  گزاره‌ای درست باشد، مدل منسو اجازه دارد که گزاره‌های  $p$  و  $q$  را درست بداند.

(۲۶) اگر  $p$  و  $q(p \& q)$  درست باشند، مدل منسو می‌تواند  $q$  را درست بداند.

(۲۷) اگر  $q(p \& q)$  درست باشد، مدل منسو می‌تواند  $q$  را درست بداند.

(۲۸) اگر  $p$  نادرست و  $q(p \mid q)$  درست باشد، مدل منسو می‌تواند  $q$  را درست بداند.

(۲۹) اگر  $(p \mid q) \& p!$  درست باشد، مدل منسو می‌تواند  $q$  را درست بداند.

(۳۰) اگر  $p = true$ ، مدل منسو می‌تواند  $p$  را درست به حساب آورد.

(۳۱) اگر  $p$  درست باشد، مدل منسو می‌تواند  $p$  را مساوی با  $true$  بداند.

(۳۲) اگر  $p = false$ ، مدل منسو می‌تواند  $p$  را نادرست به حساب آورد.

(۳۳) اگر  $p$  نادرست باشد، مدل منسو می‌تواند  $p$  را مساوی با  $false$  بداند.

(۳۴) اگر دو موجودیت با موجودیت سومی مساوی باشند، مدل منسو می‌تواند هر سه موجودیت را مساوی بداند.

(۳۵) اگر  $q(p \& q)$  و  $p(q(p \& q))$  درست باشند، مدل منسو می‌تواند  $p$  و  $q$  را مساوی بداند.

(۳۶) اگر  $p(q(p \& q))$  درست باشد، مدل منسو می‌تواند  $p$  و  $q$  را مساوی بداند.

(۳۷) اگر گزاره  $p$  مساوی با گزاره  $q$  باشد، مدل منسو می‌تواند  $q$  (if  $p$ ) را درست بداند.

(۳۸) اگر گزاره  $p$  مساوی با گزاره  $q$  باشد، مدل منسو می‌تواند  $q$  !  $p$  را درست بداند.

(۳۹) اگر گزاره  $p$  مساوی با گزاره  $q$  باشد، مدل منسو می‌تواند  $q$  &  $p$  را نادرست بداند.

(۴۰) اگر با فرض درست\_بودن گزاره  $p$ ، از قوانین دیگر، مدل منسو مجاز باشد که گزاره  $q$  را درست بداند، آنگاه مدل منسو مجاز است که  $q$  (if  $p$ ) را درست بداند.

(۴۱) اگر گزاره درست  $p$  در لیست گزاره‌های درست موجود باشد که  $p$  ! نیز درست باشد، مدل منسو می‌تواند  $false$  را درست بداند. (هنگامی که درست باشد، دیگر اجرای هیچ دستوری مجاز نیست و نوعی پایان است)

مثال: استفاده از قانون استدلالی (۲۴):

```
new prop p;
check p = !!p;
```



مثال: استفاده از قانون استدلالی (۲۵):

```
new prop p;
new prop q;
add p & q;
noadds;

check p;
check q;
```



مثال: استفاده از قانون استدلالی (۲۶):

```
new prop p;
new prop q;
add (if p) q;
```



```
add p;  
noadds;  
  
check q;
```



مثال: استفاده از قانون استدلالی (۲۷):

```
new prop p;  
new prop q;  
add p & (if p) q;  
noadds;  
  
check q;
```



مثال: استفاده از قانون استدلالی (۲۸):

```
new prop p;  
new prop q;  
add !p ;  
add (p | q);  
noadds;  
  
check q;
```



مثال: استفاده از قانون استدلالی (۲۹):

```
new prop p;  
new prop q;  
add !p & (p | q);  
noadds;  
  
check q;
```



مثال: استفاده از قانون استدلالی (۳۰):

```
new prop p;  
add p = true;  
noadds;  
  
check p;
```



مثال: استفاده از قانون استدلالی (۳۱):

```
new prop p;  
add p;  
noadds;  
  
check p = true;
```



مثال: استفاده از قانون استدلالی (۳۲):

```
new prop p;  
add p = false;  
noadds;  
  
check !p;
```



مثال: استفاده از قانون استدلالی (۳۳):

```
new prop p;  
add !p;  
noadds;  
  
check p=false;
```



مثال: استفاده از قانون استدلالی (۳۴):

```
new object x1(prop);  
new object x2(prop);
```



```
new object x3(prop);  
new object x4(prop);  
add x1 = x2;  
add x2 = x3;  
add x3 = x4;  
noadds;  
  
check x1 = x4;
```



مثال: استفاده از قانون استدلالی (۳۵):

```
new prop p;  
new prop q;  
add (if p)q;  
add (if q)p;  
noadds;  
  
check p = q;
```



مثال: استفاده از قانون استدلالی (۳۶):

```
new prop p;  
new prop q;  
add (if p)(q) & (if q)p;  
noadds;  
  
check p = q;
```



مثال: استفاده از قانون استدلالی (۳۷):

```
new prop p;  
new prop q;  
add p = q;  
noadds;  
  
check (if p)q;
```



مثال: استفاده از قانون استدلالی (۳۸):

```
new prop p;  
new prop q;  
add p = q;  
noadds;  
  
check p | !q;
```



مثال: استفاده از قانون استدلالی (۳۹):

```
new prop p;  
new prop q;  
add p = q;  
noadds;  
  
check !(p & !q);
```



مثال: استفاده از قانون استدلالی (۴۰):

```
new prop p;  
new prop q;  
  
check (if p & q) p;
```



مثال: استفاده از قانون استدلالی (۴۱):

```
new prop p;  
new prop q;  
add p = !q;  
add p;  
add q;  
noadds;
```



`check false;`



## درباره قوانین استدلالی

- از قوانین استدلالی‌ای که ذکر شد، برخی به ذهن انسان بسیار نزدیکند، مثلاً:
- اگر  $p \wedge q$  درست باشد، مدل منسو می‌تواند  $p$  را درست بداند.
  - و برخی از ذهن دور هستند:
  - اگر  $p \wedge \neg p$  درست باشد، مدل منسو اجازه دارد  $\neg p$  را نادرست به حساب بیاورد.

معمولًا اگر قوانین دور از ذهن استدلال بسته نمی‌شود، زیرا این قوانین دور را، می‌توان با قوانین دیگر ثابت کرد. اما چرا در مدل منسو، این قوانین دور از ذهن هم به عنوان قانون اضافه شده‌اند، در حالی که قرار است مدل منسو، ذهن منطقی انسان را شبیه‌سازی کند؟ پاسخ این است که وجود این قوانین، مدل منسو را از شبیه‌سازی ذهن باز نمی‌دارد، بلکه علاوه بر آن، مجهر به قابلیت‌هایی می‌کند که می‌توانند به ذهن انسان کمک کنند و نقص‌های آن را پوشانند.

منطق موجود در ذهن، در حدی تکامل یافته است که بتواند جواب‌گوی حداقل‌های لازم برای ادامه حیاتش باشد، اما وقتی قرار باشد ذهن، برای عملیات منطقی پیچیده، مانند ریاضیات، به کار رود، خطاهای زیادی دارد که البته با تمرین، این خطاهای کاسته می‌شوند. مدل منسو می‌تواند به عنوان دستیاری برای ذهن انسان مورد استفاده قرار گیرد و بهتر است در برابر ضعف‌هایی که نسبت به ذهن انسان دارد، نقاط قوتی نیز داشته باشد.

در فهرستی از قوانین استدلالی که ارائه شد، برخی از قوانین، نتیجه مستقیمی از قوانین دیگرند؛ برای مثال:

- اگر  $p$  گزاره  $p$  مساوی با گزاره  $q$  باشد، مدل منسو می‌تواند  $\neg p \vee q$  را درست بداند.
- نتیجه مستقیمی:
- اگر  $\neg p \wedge p$  نادرست باشد، مدل منسو، اجازه دارد  $\neg p \vee q$  را درست به حساب آورد.



در واقع بیشتر قوانین ذکر شده قابل نتیجه‌گیری از چند قانون نزدیک به ذهن هستند.

## فضاها و قوانین استدلال برای سورها

برای اثبات گزاره‌های شرطی و سورها، می‌توان از فضاهای فرض، شیء و انتخاب استفاده کرد. در این قسمت، این فضاها و قوانین مربوط به استدلال سورها را مورد بررسی قرار می‌دهیم.

### فضای فرض

فضای فرض فضایی از متن بین دو آکلاد است که در آن فضا، موقتاً یک گزاره درست فرض می‌شود:

```
new prop p;  
suppose p  
{  
    check p;  
}
```

در کد بالا قسمت:

```
suppose p  
{  
}
```

نشان‌دهنده یک فضای فرض است. در این فضا، موقتاً گزاره بلاتکلیف  $p$ ، درست فرض می‌شود. پس از فضا، گزاره  $p$  دیگر درست فرض نمی‌شود و دوباره بلاتکلیف می‌گردد. بنابراین، اگر لیست گزاره‌های درست را پس از بررسی کد نگاه کنید، اثری از  $p$  نخواهد یافت. به فضای فرض بالا، فضای فرض  $p$  می‌گوییم، زیرا در آن، فرض شده است که  $p$  درست است. همچنین به خود  $p$ ، فرض فضای فرض گفته می‌شود.

فضای فرض دو کاربرد دارد. کاربرد اول آن، اثبات گزاره‌های شرطی، یعنی گزارهایی به صورت  $q \text{ if } p$  است: اگر گزاره  $q$ ، درون فضای فرض  $p$  ثابت شود، پس از فضای فرض، مدل منسو  $q \text{ if } p$  را درست خواهد دانست.



**مثال:** استفاده از فضای فرض برای اثبات گزاره شرطی:

```
new prop p(object);
new object a;

suppose p(a)
{
    pick a in (some object x)p(x);
    check (some object x)p(x);
}

check (if p(a))(some object x)p(x);
```

در کد بالا گزاره شرطی:

```
check (if p(a))(some object x)p(x);
```

ثابت شده است. برای این کار، در فضای فرض، `p(a)` با دستور `pick` از گزاره:

```
(some object x)p(x)
```

ثابت شده است. هنگام خروج از فضای فرض، این گزاره، خود به خود به صورت:

```
(if p(a))(some object x)p(x)
```

در آمده است.



کاربرد دیگر فضای فرض، اثبات نادرستی یک گزاره است. اگر درون فضای فرض `p`، درستی گزاره `false` ثابت شود، آنگاه، مدل منسو مجاز است که گزاره `p` را نادرست بداند. در بحث‌های منطق و ریاضی، معمولاً، به `p` که نادرستیش با فضای فرض ثابت می‌گردد، فرض خلف می‌گویند.

**مثال:** فضای فرض و فرض خلف:

```
new prop p(object);
new object a;
add !p(a);
```



```

suppose (all object x)p(x)
{
  pick a in (all object x)p(x);
  check p(a);
  check !p(a);
  check false;
}

check !(all object x)p(x);

```

در کد بالا، به قصد اثبات  $(\forall x)p(x)!$ ، یک فضای فرض با نقیض آن ایجاد شده است. درون فضای فرض، `false` اثبات شده است (یعنی با قوانین استدلال به لیست گزاره‌های درست اضافه شده است). در این حالت، با اتمام فضای فرض، نقیض فرض فضای فرض، یعنی  $(\forall x)p(x)!$ ، به لیست گزاره‌های درست اضافه شده است.



توجه کنید که تمام گزاره‌های درست قبل از فضای فرض، از جمله  $(a)p$ ، درون و پس از فضای فرض، همچنان معتبر باقی می‌مانند و همه گزاره‌هایی که درون فضای فرض ثابت می‌شوند، با اضافه\_شدن یک `if`، پس از فضای فرض باقی می‌مانند، مگر آن که درون فضای فرض، `false` ثابت شود؛ اگر درون یک فضای فرض `false` ثابت شود، پس از فضا تنها نقیض فرض فضا، به لیست گزاره‌های درست اضافه می‌شود و تمام گزاره‌هایی که درون فضا ثابت شده‌اند حذف می‌گردند.

### فضای شیء

فضای شیء مشابه فضای فرض است با دو تفاوت: اول این که به جای گزاره یک شیء جدید، موقتاً درون فضای شیء معتبر است و دوم این که همه گزاره‌هایی که درون فضای شیء ثابت می‌شوند، پس از فضای شیء به صورت سور عمومی در می‌آیند. برای مثال، کد زیر را در نظر بگیرید:

```

new prop p(object);
new prop q(object);
add (all object x)(if p(x))q(x);
add (all object x)p(x);

```



```
noadds;

set object a
{
    pick a in (all object x)(if p(x))q(x);
    pick a in (all object x)p(x);
    check q(a);
}

check (all object a) q(a);
```

بخش:

```
set object a
{
}
```

فضای شیء a است. به a، شیء فضای شیء بالا می‌گوییم. بین دو آکلاد، شیء a قابل استفاده است و پس از آن، از لیست اشیاء حذف می‌گردد. در فضای شیء کد فوق، گزاره `q(a)` ثابت شده است. پس از اتمام فضای شیء، این گزاره به صورت `(all object a) q(a)` در می‌آید و خط:

```
check (all object a) q(a);
```

در کد بالا، این را تأیید می‌کند. همه گزاره‌های ثابت شده در یک فضای شیء، پس از فضا، به صورت یک سور عمومی در می‌آیند. فضای شیء، ابزاری برای اثبات سورهای عمومی است. همه گزاره‌های پیش از یک فضای شیء، درون فضای شیء معتبرند و پس از فضای شیء گزاره‌هایی که به شیء فضای شیء وابسته‌اند، به صورت سور عمومی در می‌آیند.

**مثال:** شیء فضای شیء می‌تواند هر نوعی داشته باشد:

```
new object x;
new object y;
add (all prop p(object))(if p(x))p(y);
add (all prop p(object))p(x);
noadds;
```



```
set prop q(object)
{
    pick q in (all prop p(object))(if p(x))p(y);
    pick q in (all prop p(object))p(x);
    check q(y);
}

check (all prop q(object)) q(y);
```



## فضای انتخاب

فضای شیء برای اثبات سورهای عمومی به کار می‌رود. فضای انتخاب، مشابه فضای شیء است، اما کاربرد آن، اثبات سورهای وجودی است. برای شروع یک فضای انتخاب، به یک سور وجودی درست نیاز داریم. برای مثال، کد زیر را در نظر بگیرید:

```
new prop p(object);
new prop q(object);
add (some object x)p(x);
add (all object x)(if p(x))q(x);
noadds;

select x in (some object x)p(x)
{
    check p(x);
    pick x in (all object y)(if p(y))q(y);
    check (if p(x))q(x);
    check q(x);
}

check (some object x)q(x);
```

در کد بالا:

```
select x in (some object x)p(x)
{
```





یک فضای انتخاب است.  $x$ ، شیء فضای انتخاب و  $(x)p$  (some object) سور فضای انتخاب نامیده می‌شود. نوع شیء فضای انتخاب، همان نوع متغیر سور است که در اینجا **object** است. گزاره سور، برای شیء فضای انتخاب درست است، یعنی در اینجا  $(x)p$  (درون فضای انتخاب) درست است. در کد بالا خط:

```
check p(x);
```

این مطلب را تأیید می‌کند. البته نیازی نیست شیء فضای انتخاب، همنام با متغیر سور باشد.

در کد بالا گزاره  $(x)p$  (some object) درست است، و اگر درست نبود، نمی‌شد با آن یک فضای انتخاب ایجاد کرد. این گزاره را، برای فهم ذهن انسان، می‌توان به صورت «حداقل یک  $x$  وجود دارد که  $(x)p$  درست باشد» بیان کرد. برای فهم ذهن انسان دستتر:

```
select x in (some object x)p(x)
```

را می‌توان به این صورت خواند: «فرض کن  $x$  یکی از  $x$ هایی باشد که برای آنها  $(x)p$  درست است». در کد بالا، در درون فضای انتخاب،  $(x)q$  ثابت شده است. پس، به زبان انسانی، برای حداقل یک  $x$ ،  $(x)q$  صحیح است. به همین سبب است که پس از فضای انتخاب، گزاره  $(x)p$  (some object) درست در نظر گرفته می‌شود.

به این ترتیب، فضای انتخاب برای این است که از روی یک سور وجودی، سور وجودی دیگری را ثابت کنیم. ذهن انسان نیز برای اثبات گزاره‌های مشابه، از روش مشابهی استفاده می‌کند. پس، آنچه در مدل و نرم‌افزار منسو روی می‌دهد، غیر\_عادی نیست و با کمی دقت قابل\_درک است.

گفتیم که  $(x)p$  (some object) یه معنی «حداقل یک  $x$  وجود دارد که  $(x)p$  درست باشد» است. اما ممکن است در مواردی، دقیقاً یک  $x$  وجود داشته باشد که  $(x)p$  درست باشد. در این موارد، علاوه بر گزاره  $(x)p$  (some object) گزاره:

```
(all object x)(all object y)(if p(x) & p(y))x = y
```

نیز صحیح است که به معنی منحصر به فرد بودن  $x$  است که برای آن  $p(x)$  درست است. در چنین مواردی، نیازی نیست که فضای انتخاب و شیئی موقت ایجاد گردد و می‌توان با دستور:

```
select x in (some object x)p(x);
```

آن شیء منحصر به فرد را به دست آورده:

```
new prop p(object);
new prop q(object);
add (some object x)p(x);
add (all object x)(if p(x))q(x);
add (all object x)(all object y)(if p(x) & p(y))x = y;
noadds;

select unique in (some object x)p(x);

check p(unique);
pick unique in (all object y)(if p(y))q(y);
check (if p(unique))q(unique);
check q(unique);
```

در کد بالا، شیء منحصر به فردی را که برای آن  $p(x)$  درست است با دستور:

```
select unique in (some object x)p(x);
```

استخراج کرده‌ایم و آن را `unique` نامیده‌ایم.

### قوانین استدلالی مربوط به سورها

هرچند بسیاری از قوانین مربوط به سورها را می‌تواند با فضاهای فرض، شیء و انتخاب ثابت کرد یا به دست آورد، در مدل منسو، چند قانون به صورت خودکار بررسی می‌شود: (۴۲) اگر  $p$  یک گزاره سور باشد که متغیر سور در بخشی از آن به کار نرفته است، مدل منسو می‌تواند به جای سور،  $p$  بگذارد.

(۴۳) اگر  $(\text{all type } x)p(x)$  و  $(\text{all type } x)q(x)$  درست باشند، مدل منسو می‌تواند  $(\text{all type } x) p(x) \& q(x)$  درست بداند.

(۴۴) اگر  $(\text{some } \text{type } x) ! p(x)$  درست باشد، مدل منسو مجاز است گزاره  $(\text{all } \text{type } x) ! p(x)$  را درست محسوب کند.

(۴۵) اگر  $(\text{some } \text{type } x) ! p(x)$  درست باشد، مدل منسو مجاز است گزاره  $(\text{all } \text{type } x) ! p(x)$  را درست محسوب کند.

(۴۶) اگر  $(\text{all } \text{type1 } x) (\text{all } \text{type2 } y) p(x, y)$  درست باشد، مدل منسو می‌تواند  $(\text{all } \text{type2 } y) (\text{all } \text{type1 } x) p(x, y)$  را درست به حساب آورد. و  $\text{type1}$  و  $\text{type2}$  را با هر نوعی می‌توان جایگزین کرد.

(۴۷) اگر  $(\text{some } \text{type1 } x) (\text{some } \text{type2 } y) p(x, y)$  درست باشد، مدل منسو می‌تواند  $(\text{some } \text{type2 } y) (\text{some } \text{type1 } x) p(x, y)$  را درست به حساب بیاورد.

(۴۸) اگر  $(\text{all } \text{type1 } x) (\text{some } \text{type2 } y) p(x, y)$  درست باشد، مدل منسو می‌تواند گزاره  $(\text{some } \text{type2 } f(\text{type1})) (\text{all } \text{type1 } x) p(x, f(x))$  را درست به حساب آورد.

(۴۹) اگر با فرض اضافه\_کردن  $x$  به لیست موجودیت‌ها،  $(x) p$  گزاره‌ای درست باشد، مدل منسو مجاز است که  $(\text{all } \text{type } x) p(x)$  و  $(\text{some } \text{type } x) p(x)$  را گزاره‌هایی درست بداند.

(۵۰) اگر  $(\text{some } \text{type } x) p(x)$  درست باشد، مدل منسو می‌تواند  $x) p(x)$  را درست بداند.

می‌توان قوانین دیگری را نیز به لیست فوق اضافه کرد اما با وجود فضاهای انتخاب، شیء و فرض نیاز زیادی به قوانین دیگر نیست. هر چند قانون فوق را هم می‌توان با استفاده از فضاهای اثبات کرد.

**مثال:** استفاده از قانون استدلالی (۴۲):

```
new prop p;
add (some object x)p;
noadds;

check p;
```



مثال: استفاده از قانون استدلالی (۴۳):

```
new prop p(object);
new prop q(object);
add (all object x)p(x);
add (all object x)q(x);
noadds;

check (all object x) p(x) & p(x);
```



مثال: استفاده از قانون استدلالی (۴۴):

```
new prop p(object);
new prop q(object);
add !(all object x)!p(x);
noadds;

check (some object x)p(x);
```



مثال: استفاده از قانون استدلالی (۴۵):

```
new prop p(object);
new prop q(object);
add !(some object x)!p(x);
noadds;

check (all object x)p(x);
```



مثال: استفاده از قانون استدلالی (۴۶):

```
new prop p(object, object);
add (all object x)(all object y)p(x,y);
check (all object y)(all object x)p(x,y);
```





**مثال: استفاده از قانون استدلالی (۴۷):**

```
new prop p(object, object);
add (some object x)(some object y)p(x,y);
check (some object y)(some object x)p(x,y);
```



**مثال: استفاده از قانون استدلالی (۴۸) که می‌توان آن را معادلی از اصل انتخاب در ریاضیات دانست:**

```
new prop p(object, object);
add (all object x)(some object y)p(x,y);
check (some object f(object))(all object x)p(x,f(x));
```



**مثال: استفاده از قانون استدلالی (۴۹):**

```
check (some object x)(x = x);
check (all object x)(x = x);
```



**مثال: استفاده از قانون استدلالی (۵۰):**

```
new prop p(object);
add (all object x)p(x);
check (some object x)p(x);
```



## ریاضیات و مدل منسو

شاید ریاضیات محض مهم‌ترین بخش از علوم باشد که در آن ذهن منطقی انسان به کار گرفته می‌شود. ریاضیدانان، سال‌ها تمرین می‌کنند تا قوانین منطق را، که پیش‌تر بسیاری از آن‌ها را در قالب مدل منسو ذکر کردیم، بدون خطا به کار گیرند. با این حال، بسیاری از آنان به قوانین منطقی‌ای که به کار می‌گیرند، آگاه نیستند، همان‌گونه که صحبت‌کنندگان به یک زبان خاص، ممکن است از دستور زبان آن زبان، بی‌خبر باشند. در حقیقت،

همان‌طور که ما از عملکرد سیستم بینایی خبر نداریم، اما از این سیستم به خوبی استفاده می‌کنیم، بسیاری از افراد، ممکن از جزئیات منطق تکامل یافته انسانی بی‌خبر باشند، اما از آن به‌خوبی استفاده کنند.

همان‌گونه که سیستم بینایی نوزاد به تدریج و با تمرین (ثبت خاطرات بینایی) کامل و قوی می‌گردد، سیستم منطقی ذهن انسان نیز، با تمرین می‌تواند قوی و قابل‌استفاده شود. اما یک ریاضیدان که منطقی قوی و کامل دارد، لزوماً از جزئیات این منطق و قوانین حاکم بر آن با خبر نیست.

ریاضیدانان سعی کرده‌اند با همان قالب ریاضی‌ای که در آن بسیاری از نظریات خود را تبیین می‌کنند، به بررسی منطق ریاضی بپردازند و آن را در قالب یک نظریه ریاضی (مانند نظریه گروه‌ها یا نظریه رسته‌ها) ارائه کنند. اما در این نظریات، خبری از فضای فرض، فضای شیء یا فضای انتخاب و لیست گزاره‌های درست، لیست اشیاء معرفی شده و... نیست. بنابراین، نمی‌توان مدل آن‌ها را بررسی ذهن منطقی انسان دانست.

در این بخش به ارتباط بین مدل منسو و آنچه در متون ریاضی رایج است می‌پردازیم.

### نمادهای ریاضی

در زبان منسو، برای گزاره‌ها، نوشتار خاصی مورد استفاده است؛ به عنوان نمونه، برای سور وجودی، از نوشتاری مانند:

`(some object x)p(x)`

استفاده می‌کنیم. اما در ریاضیات از نمادهای دیگری استفاده می‌شود.

هرچند نمادها در ریاضیات استاندارد واحدی ندارد و در هر کتاب یا شاخه ممکن است نمادهای خاصی به کار رود، در اینجا سعی می‌کنیم رایج‌ترین نمادها را که با زبان منسو و دقیق آن سازگار باشد، ارائه دهیم.

۱) برای سور عمومی مانند  $(\text{object } x)p(x)$  از نمادی مانند:  
 $(\forall x \in A)(p(x))$

استفاده می‌شود که  $A$  یک مجموعه است که در مورد آن در آینده صحبت خواهیم کرد.

(۲) برای سور وجودی مانند  $(\exists x \in A)(p(x))$  از نماد:

$$(\exists x \in A)(p(x))$$

استفاده می‌شود که در آن  $A$  یک مجموعه است.

(۳) برای گزارۀ شرطی  $p \rightarrow q$  از نماد:

$$p \rightarrow q$$

استفاده می‌شود.

(۴) برای گزارۀ پیوند عمومی  $q \& p$  نماد:

$$p \wedge q$$

به کار می‌رود.

(۵) برای گزارۀ پیوند وجودی  $q \mid p$  نماد:

$$p \vee q$$

به کار می‌رود.

(۶)  $p$ ! که نقیض گزارۀ  $p$  است، در ریاضیات، به صورت  $\neg p$  نشان داده می‌شود.

(۷) آنچه در مدل منسو، تساوی دو گزاره نامیده می‌شود، در ریاضیات، معادل\_بودن یا همارزی گزاره‌ها نامیده می‌شود. همچنین در مورد گزاره‌ها، به جای  $q = p$ ، از نماد  $q \leftrightarrow p$  استفاده می‌شود.

## اشیاء ذهنی و مجموعه در ریاضی

در مدل منسو، به جز `prop` که مخصوص گزاره‌های است، و معنای مشخصی در ریاضیات دارد، موجودیت‌هایی با نوع `object` داریم. اما معادل `object` در ریاضیات چیست؟ در پاسخ باید گفت که `object`‌ها را می‌توان نمایندهٔ چیزهای مختلفی در ریاضیات گرفت. برای مثال، می‌توان بحث و منطق را به یک مجموعه منحصر کرد و `object`‌ها را اعضای آن مجموعه گرفت. در جای دیگر می‌توان هر `object` را یک مجموعه از ردۀ همهٔ مجموعه‌ها گرفت. در هر حال، باید توجه داشت که مدل منسو، در نسخهٔ فعلی، مخصوص ریاضیات و متن‌های ریاضی طراحی نشده است. بلکه هدف اصلی آن، شبیه‌سازی ذهن منطقی انسان است و قرار نیست تابع قراردادها و سنت‌های ریاضیات باشد.

**مثال:** در ریاضیات برای تعریف اعداد صحیح نامنفی (یعنی اعداد  $0, 1, 2, 3, \dots$ ،) اصولی به نام اصول پئانو وجود دارد. در مدل منسو، می‌توان هر `object` را یک عدد طبیعی در نظر گرفت. کد زیر، سعی دارد اصول پئانو را با زبان منسو ارائه دهد.

```
// Peano axioms:
new object 0;
new object next(object);
add (all object n)(all object m)((m=n) = (next(n)=next(m)));
add (all object n)!(next(n)=0);
add (all prop p(object))(if p(0) & (all object n)(if
p(n))p(next(n)))(all object n)p(n);

// definition of +:
new operator object+(object, object);
add (all object n) n + 0 = n;
add (all object m)(all object n)next(n + m)= n + next(m);

define 1 = next(0);
define 2 = next(1);
define 3 = next(2);
```

در کد بالا سعی شده است که اصول پئانو برای اعداد صحیح نامنفی تبیین گردد. در این کد، هر `object`، نماینده یک عدد صحیح نامنفی است. برای مثال، در:

```
add (all prop p(object))(if p(0) & (all object n)(if
p(n))p(next(n)))(all object n)p(n);
```

اصل استقرای اعداد صحیح تبیین شده است که منظور از `next(n)` همان  $n + 1$  است و تالی  $n$  نامیده می‌شود. در کد بالا، از دستور `define` استفاده شده است که پیشتر معرفی نشد: اگر یک موجودیت باشد، مثلاً با نوع `object`، کد:

```
define x = y;
```

معادل است با:

```
new object x;
add x = y;
```

اما از `define` پس از `noadds` هم می‌توان استفاده کرد.

پس از کد بالا، می‌توان بخش بزرگی از قضایای مربوط به اعداد و نظریهٔ اعداد را، با زبان منسو بازنویسی کرد و مزیت این است که این بار، منطق نوشه‌ها توسط یک ذهن رایانه‌ای قابل‌بررسی است نه ذهن پر از خطای انسانی.



**مثال:** در ریاضیات محض، ساختاری به نام «گروه» تعریف می‌شود که شامل یک مجموعه و یک عمل است. عمل، در زبان منسو، همان عملگر است. در گروه، گزاره‌هایی برای این عملگر برقرارند. می‌توان یک گروه مانند  $G$  در نظر گرفت و فرض کرد `object`‌ها در زبان منسو اعضای این گروه هستند:

```

new operator object .(object,object);
add (all object x)(all object y)(all object z)
(
    (x.y).z = x.(y.z)
);
add (some object 1)
(
    (all object x)((x.1 = x) & (1.x = x)) &
    (all object x)(some object y)((x.y = 1) & (y.x = 1))
);
noadds;

set object 1
{
    set object I
    {
        suppose
        ((all object x)((x.1 = x) & (1.x = x)) &
        (all object x)(some object y)((x.y = 1) & (y.x = 1))) &
        ((all object x)((x.I = x) & (I.x = x)) &
        (all object x)(some object y)((x.y = I) & (y.x = I)))
        {
            check (all object x)((x.1 = x) & (1.x = x)) &
                (all object x)(some object y)((x.y = 1) & (y.x =
1));
        }
    }
}

```

```

pick I in (all object x)((x.1 = x) & (1.x = x));
check I.1 = I;

check (all object x)((x.I = x) & (I.x = x)) &
      (all object x)(some object y)((x.y = I) & (y.x =
I));

pick 1 in (all object x)((x.I = x) & (I.x = x));
check I.1 = 1;
check 1 = I;
}

}

select 1 in (some object 1)
(
  (all object x)((x.1 = x) & (1.x = x)) &
  (all object x)(some object y)((x.y = 1) & (y.x = 1))
);

check (all object x)((x.1 = x) & (1.x = x));
check (all object x)(some object y)((x.y = 1) & (y.x = 1));

```

در کد فوق، منحصر به فرد بودن عنصر همانی گروه که با ۱ یا I نشان داده شده است، ثابت می‌گردد و با دستور `select`، این عنصر همانی، از قید سورها در می‌آید و به عنوان شیئی دائمی معرفی می‌گردد.



به این ترتیب، `object`، در ریاضیات، می‌تواند نماینده یک عضو از یک مجموعه خاص یا یک رده خاص باشد. می‌توان مدل منسو را توسعه داد تا بتوان اعضاًی از چند مجموعه را هم‌زمان در دسترس داشت. اما در مدل منسو، در شکل فعلی، تنها دو نوع اصلی `prop` و `object` موجودند. اگر قرار به نوشتن متن‌های ریاضی با مدل منسو باشد، می‌توان امکانات خاصی را مخصوص ریاضیات به آن افزود تا نوشتن متن‌های ریاضی با آن ساده‌تر گردد.





# فصل سوم – کاربردها و مباحث تکمیلی

در این فصل در مورد کاربردهایی که شبیه‌سازی ذهن منطقی انسان دارد و ادامه این شبیه‌سازی سخن خواهیم گفت.

## گام‌های بعدی

### تبديل زبان منسو به يك نمود گرافيكي يا بازي راياني

زبان منسو که در فصل گذشته معرفی شد یا شکل تکامل یافته‌اش را می‌توان به صورت گرافیکی ارائه داد؛ چیزی شبیه به تصاویری که در فصل اول برای ذهن انسان نشان داده شد. تصور کنید که گزاره‌های مختلف به صورت اشکالی گرافیکی بر پنجره یک نرم‌افزار ظاهر شود و بتوان قوانین منطقی را به صورت گرفیکی به نمایش گذاشت. برای مثال، این کد را در نظر بگیرید:

```
new prop p(object);
add (all object x)p(x);
new object a;
pick a in (all object x)p(x);

check p(a);
```

نرم‌افزاری را تصور کنید که (all object x)p(x) و a را به صورت تصاویری نشان می‌دهد و کاربر نرم‌افزار، می‌تواند با کلیک بر روی تصویر مربوط به a، آن را بگیرد و بکشد روی تصویر مربوط به (all object x)p(x)؛ و با این کار، ناگهان p(a) به صورت تصویری جدید ظاهر گردد. این عمل گرفتن و کشیدن می‌تواند معادلی برای:

```
pick a in (all object x)p(x);
```

باشد.



اگر چنین نرم‌افزاری ساخته شود، که البته نیازمند طراحی و هزینه فراوان است، می‌توان منطق ذهن انسان را به صورت نوعی بازی رایانه‌ای در آورد. چنین بازی‌ای می‌تواند آنقدر پیشرفت‌ه باشد که بتواند اثبات قضایای مربوط به ریاضات را به صورت تصویری به نمایش بگذارد. مزیت این کار، این خواهد بود که افراد برای یادگیری منطق و ریاضیات نیازی به تمرین‌های خشک و طاقت‌فرسا نخواهند داشت و تنها نیازمند آموختن یک بازی فکری رایانه‌ای خواهند بود. با چنین پیش‌زمینه‌ای، وارد\_شدن به مباحث درسی مربوط به منطق و ریاضیات بسیار ساده‌تر خواهد شد.

از سوی دیگر، با ادامه توسعه، ممکن است روزی چنین نرم‌افزاری چنان پیشرفت‌ه شود که بتوان با آن، قضایای اثبات‌نشده ریاضیات را، به صورت یک معما درون یک بازی رایانه‌ای ادامه داد و از طیف بزرگی از مخاطبان خواست تا آن معما را از طریق بازی حل کنند. مخاطبانی که ممکن است هیچ آشنایی‌ای با ریاضیات نداشته باشند، اما با منطق ساده‌بازی به خودی ارتباط برقرار کنند. به این ترتیب، ممکن است از بین این مخاطبان، مخاطبانی بتوانند آن قضیه را در قالب بازی ثابت کنند و روش آن را برای طراح ارسال نمایند تا طراح، روش بازی را به یک متن استاندارد ریاضی برگرداند. به این شکل، ریاضیدانانی پیدا خواهند شد که ریاضی نمی‌دانند.

### شبیه‌سازی کامل ذهن

برای شبیه‌سازی کامل ذهن، بی‌شک قوهٔ تفکر یک مرکز نقل است. تفکر هم چیزی شبیه به کاری است که نرم‌افزار منسو انجام می‌دهد. تفکر یعنی یک دسته از جملات معنی‌دار در ذهن بررسی شوند. اگر نتیجهٔ بررسی به خطأ منجر شد، آن جملات اصلاح می‌شوند و دوباره بررسی می‌شوند و یا دستهٔ دیگری از جملات معنی‌دار بررسی می‌گردند. به این ترتیب، می‌توان گفت تفکر معادل با اصلاح و بررسی یک کد به زبان منسو است.

اما این اصلاح به چه صورت انجام می‌گیرد؟ به نظر می‌رسد فرآیند اصلاح در مغز، خود تفکر بر متن‌های شبه‌منسوبی است که با تجربیات مختلف و با تصادف به وجود آمده‌اند.

آیا می‌توان فرآیند اصلاحی را که ذهن انجام می‌دهد شبیه‌سازی کرد؟ پاسخ مثبت است اما به نظر می‌رسد این کار، تنها با تکمیل شبیه‌سازی ذهن ممکن باشد. ذهن منطقی انسان

بخشی از ذهن است. یکی از ویژگی‌های ذهن هدف‌محوری است. یک هدف در ذهن انسان به دلایلی، اغلب بر مبنای تصادف در ساختار تکامل‌سافتة ذهن، اهمیت می‌یابد و توان ذهن را به سوی خود می‌کشد به طوری که ذهن شروع به تست متن‌های شبه‌منسوی فراوان و انجام دستوراتی برای رسیدن به آن هدف می‌کند. برای تکمیل شبیه‌سازی ذهن انسان و افروختن قابلیت تصحیح خودکار متن‌ها توسط یک نرم‌افزار رایانه‌ای و بدون دخالت انسان، لازم است هدف‌محوری ذهن شبیه‌سازی گردد. برای شبیه‌سازی هدف‌محوری، نیاز داریم به بررسی‌های فراوان در مورد ذهن تا بتوان آن را به صورت کد ارائه داد.

در هر حال، مدل منسو یکی از اولین گام‌ها برای شبیه‌سازی کامل ذهن انسان است. اما این کار زمان‌بر، زمانی ممکن می‌شود که قابلیت‌های بیشماری که در طی هزاران سال در ذهن انسان شکل گرفته‌اند، شناسایی و بازطراحی شوند. در این کتاب سعی کردیم در مورد بازطراحی منطق انسانی سخن بگوییم.

## نتایج وجود مدل منسو

مدل منسو، قوانین دقیقی را تبیین می‌کند که برگرفته از ذهن انسان هستند. بنابراین، ذهن هر انسان، بر اساس همین قوانین فکر و نتیجه‌گیری می‌کند. هرچند ممکن است در مراحل و به کار-گیری این قوانین دچار خطأ گردد. در حقیقت، انجام مراحل مختلف نتیجه‌گیری، یک بازی فکری مفید است که افراد، باید قوانین این بازی را بشناسند. بسیاری از افراد، در این بازی، خوب عمل نمی‌کنند و بازیکن مناسب این بازی نیستند و به صورت رسمی، وارد حوزه‌های تخصصی مربوط به بازی منطق، مثل ریاضیات، برنامه‌نویسی، فلسفه و..., نمی‌گردند. از سوی دیگر، بسیاری از افراد که مدعی مهارت در بازی‌های تفکرند، وارد حوزه‌های تخصصی مربوط به آن می‌شوند، اما در عین حال، این بازی را خوب نیاموخته‌اند. علت اختلافات عمیق بین فلاسفه، در مواردی، عدم توانایی آن‌ها در حل این اختلافات، عدم شناخت آن‌ها از بازی منطق است. ریاضیدانان، معمولاً در هیچ چیز مربوط به ریاضیات اختلافی ندارند. زیرا در ریاضیات، مهارت در استفاده از قوانین منطق حرف اول را می‌زند.





## بازی فکت‌ها

در مدل منسو، گزاره‌های اولیه که با **add** به لیست گزاره‌های درست اضافه می‌شوند، مبنایی برای نتیجه‌گیری‌های بعدی هستند. با مشخص\_بودن گزاره‌های اولیه، نتایجی که از طریق کدهای منسو گرفته می‌شوند (یعنی گزاره‌های ثانویه‌ای که با استفاده از قوانین استدلالی به لیست گزاره‌های درست اضافه می‌شوند)، مورد تأیید ذهن انسان است و فرآیند استدلال و قوانین آن، آنقدر دقیق است که جایی برای اختلاف باقی نمی‌گذارد.

حال که مدل منسو، محلی برای اختلاف به وجود نمی‌آورد و مدل منسو یک شبیه‌سازی از ذهن انسان است، چرا انسان‌ها در موضوعات مختلف به شدت دچار اختلاف هستند؟

منشأ این اختلافات می‌تواند یک یا چند مورد از موارد زیر باشد:

۱) ضعف در شناخت قوانین منطق یا تعریف مغرضانه آنها. این مورد بین انسان‌هایی رخ می‌دهد که معمولاً تمرین‌های تفکری کمی داشته‌اند. برای مثال، فردی که تحصیلات کمی دارد و برای حیات نیازمند تفکر نبوده است یا به طور مادرزادی ذهن او دارای ضعف منطقی است.

۲) گفت\_و\_گو با اهداف پرآکنده. در مناظره‌ها و گفت\_و\_گوها، گزارهٔ خاص و ثابتی مورد بحث نیست و مدام اشخاص از موضوعی به موضوع دیگری وارد می‌شوند و اهدافشان تغییر می‌یابند. اثبات یک گزاره، با استفاده از گزاره‌های اولیه، یک فرآیند دقیق و زمان‌گیر است و مناظرات و گفت\_و\_گوها بسیار از این فرآیند دور هستند. اختلافات و مشاجرات بر سر بحث‌های متفرق، ممکن است، به غلط، اختلاف بر سر یک موضوع خاص تلقی شوند.

۳) گزاره‌های اولیهٔ غیر\_همسان و متغیر. اگر اشکالات منطقی ذهن انسان‌ها و اغراض و اهداف پرآکنده آن‌ها را نادیده بگیریم، منشأ اصلی اختلافات بر سر **add** است؛ یعنی گزاره‌هایی که بدون بررسی کافی یا عدم\_امکان بررسی کافی در ذهن پذیرفته شده‌اند. برای مثال، ممکن است کسی این گزاره را که «آدم‌ها حقه‌بازند مگر این که خلافش ثابت شود» را در ذهنش **add** کرده باشد و شخص دیگری «آدم‌ها رو راستند مگر این که خلافش ثابت شود» را در ذهنش **add** کرده باشد. و همین‌طور گزاره‌های فراوان دیگر که منشأ اختلاف باشند. این تفاوت در گزاره‌های اولیه، می‌تواند به اختلاف در نتایج منجر گردد.

در اینجا کاری به دو مورد اول نداریم، زیرا خارج از بحث این کتاب هستند. اما برای مورد سوم، که در آن دو فرد که از نظر منطقی ماهر، در برابر هم هستند و اغراض مختلف ندارند، راهی برای حل اختلاف وجود دارد. در ادامه؛ به فردی که از نظر منطقی ماهر است و اغراض مختلف ندارد، سخنگوی دانا خواهیم گفت.

برای بررسی دلیل و راه حل اختلافات بین سخنگویان دانا، بهتر است ابتدا ببینیم ریاضیدانان، که سخنگوی دانا محسوب می‌شوند، در مواردی که گزاره‌های اولیه آن‌ها، که در ریاضیات اصل<sup>۴</sup> نامیده می‌شوند، با هم فرق دارند، چه می‌کنند. در ریاضیات، این اختلافات، در واقع اختلاف نیستند، زیرا تنها ادعای ریاضیدان این است که «اگر این اصول را بپذیریم»، این نتایج برقرارند و ریاضیدانان، دیگر نه تنها با این ادعا مشکلی ندارند حتی با آن موافقند؛ تنها موضوع مطرح این است که برخی از ریاضیدانان کارهای خود را بر اساس پذیرش برخی اصول پیش می‌برند و برخی بر اساس پذیرش اصول دیگر. بنابراین، می‌توان گفت در ریاضیات اختلافی وجود ندارد. با رویه مشابه، در موضوعات دیگر، به جز ریاضیات نیز می‌توان گفت که نباید بین دو سخنگوی دانا، اختلافی وجود داشته باشد و سخنگویان دانا، می‌توانند، مانند ریاضیدانان عمل کنند.

اما مشکل بر سر انتخاب‌هایی است که بر تصمیمات و زندگی دیگران اثر می‌گذارند. در ریاضیات، ریاضیدان، هیچ کاری نمی‌کند جز نوشتن مقالات یا کتاب‌ها و ارائه آن‌ها و نوشتن کتاب و مقاله هیچ محدودیتی برای افراد دیگر ایجاد نمی‌کند. هر کس می‌تواند بر اساس add های خود، مقالات و کتاب‌های خود را بنویسد و کاری به کار بقیه نداشته باشد. معمولاً در مورد کارهای دانشمندان دیگر نیز، مانند دانشمندان فیزیک، زیست‌شناسی و...، وضعیت مشابهی وجود دارد. اما همه چیز، علم و نوشتن مقاله نیست و همه سرنشیان یک اتومبیل نمی‌توانند راننده باشند و اتومبیل را بر اساس نتایج و گزاره‌های اولیه خود به آنجا که می‌خواهند هدایت کنند. اختلاف وقتی پیش می‌آید که باید راننده انتخاب شود و همه تابع تصمیم او گرددند.

<sup>۴</sup> axiom

پس، پرسش اصلی این است که وقتی دو سخنگوی دانا بر سر انتخابی مشترک دچار اختلاف هستند، چگونه می‌توانند با گفت‌و‌گو به تفاهم برسند؟ این اختلاف هر چه باشد، بر\_می‌گردد به گزاره‌های اولیه‌ای که در ذهن هر یک از سخنگویان **add** شده‌اند.

سخنگویانان دانا باید در گام نخست این گزاره‌های ابتدایی را شناسایی کنند. برای شناسایی، هر سخنگو، می‌باشد، نخست، انتخاب خود را تعیین کند و سپس توضیح دهد که چگونه به این انتخاب رسیده است. این توضیح باید در حد یک متن ریاضی یا کد منسو دقیق و کامل باشد تا از این توضیح گزاره‌های اولیه مشخص شوند. پس از مشخص\_شدن گزاره‌های اولیه هر سخنگو، مرحله بعد، بحث بر سر گزاره‌های اولیه مورد اختلاف است. بی‌شک برای **add**\_شدن یک گزاره اولیه در ذهن یک سخنگو، دلایل وجود دارد. دلایلی مثل تجربیات یا احساسات. به هر\_یک از این دلایل، یک فکت<sup>۵</sup> می‌گوییم.

مرحله بعد، می‌تواند ارزش‌گذاری فکتها باشد. هر\_یک از سخنگویان می‌تواند به هر فکت، امتیازی بدهد. از آنجا که سخنگوی دانا مغرض نیست، به هر فکت امتیازی منطقی خواهد داد. سپس؛ از بین فکتها مورد اختلاف؛ آنها که امتیاز بیشتری دارند از سوی هر دو طرف انتخاب می‌شوند؛ یعنی بر سر آنها تفاهم می‌شود به شکلی که هر دو طرف؛ فکتها واحدهای پیدا می‌کنند. در نهایت؛ آنان می‌توانند بر اساس فکتها جدید، به نتیجه و انتخاب واحدی برسند؛ زیرا بر اساس مدل منسو گزاره‌های اولیه یکسان به نتایجی مورد توافقی منجر می‌گردند.

به الگوریتمی که برای حل اختلاف بین سخنگویان دانا تبیین شد، بازی فکتها می‌گوییم. به این ترتیب، مذاکره و مناظره، به صورت یک بازی منطقی در می‌آید که قطعاً به تفاهم منجر خواهد گردید. اما آیا بازی فکتها پیشنهادی مناسب برای حل اختلافات در بین انسان‌هاست. بی‌شک اگر این بازی در سطح وسیع آموزش داده شود و به عنوان یک رسم یا فرهنگ پذیرفته شود، می‌تواند روند رسیدن به توافقات بین انسان‌ها را تسهیل کند، اما واقعیت این است که سخنگویان دانا بسیار کمیابند و این بازی، مخصوص سخنگویانان داناست. پس تنها مورد کاربرد آن می‌تواند در برخی مباحث علمی باشد و برای اختلافات دیگر مانند اختلافات سیاسی و مذهبی و فرهنگی در عمل، کاربرد امیدوارکننده‌ای نخواهد

<sup>۵</sup> fact

داشت، مگر آن که مسیر تکامل تدریجی فراوانی سخنگویان دانا را در آینده دور بیشتر سازد.

## کلام و نظم افکار

آنچه می‌نویسیم و می‌گوییم، اغلب، گزاره‌هایی از ذهن منطقی ما هستند. بنابراین، «کلام» برای انسان معادل با «متن نوشته شده به زبان منسو» در مدل منسو است. اما کلام که حاصل تکامل تدریجی است، بسیار نامنظم‌تر است. با این حال، اگر دقیق شویم، می‌توانیم عناصر تشکیل‌دهنده زبان منسو را در کلام و افکار خود بیابیم:

با واکاوی عناصر در افکار و گفتار خود در طی زمان، می‌توانیم به «گزاره‌های اولیه» و آنچه در بخش قبل، «فکت» نامیدیم برسیم. اگر فکتهاي افکار خود را پیدا کنیم، می‌توانیم به اصالت آنها فکر کنیم و ببینیم که چرا این فکتها برای ما اهمیت پیدا کرده‌اند و آنها را در ذهن [add](#) کرده‌ایم. این فرآیند، می‌تواند ما را به اصول فکر اصیل تری برساند؛ قواعدی مشخص که حیات و افکار ما بر اساس آن‌هاست.

از سوی دیگر، بررسی کلام و مقایسه آن با زبان منسو، می‌تواند ما را در تفکر ماهرتر سازد. اما یک تفاوت مهم بین افکار انسان و مدل منسو این است که ذهن انسان با گزاره‌های صد\_در\_صد درست یا صد\_در\_صد نادرست کار ندارد؛ بلکه بیشتر، با گزاره‌هایی بلاتکلیف سر\_و\_کار دارد. ذهن مجبور است برش خی از آنها را بر اساس «احتمال»، درست یا نادرست فرض کند. از این رو، یکی از مباحث مهم در افکار انسان‌ها، (مهارت در) تعیین احتمال است. پس، علاوه بر شناخت منطق انسانی، به روش‌هایی علمی برای تعیین احتمالات هم نیازمندیم. تعیین احتمالات در ذهن افراد ممکن است بر اثر اختلال‌هایی مثل وسواس، یا تجربیات یا شخصیت افراد به شکلی غیرمنطقی انجام گیرد. به همین دلیل، علاوه بر مدل منسو، به الگوریتم‌ها و مدل‌هایی برای تعیین احتمال و انتخاب مناسب گزاره‌های اولیه نیز نیازمندیم. اما در هر حال، تعیین چنین الگوریتمی، خود، از طریق منطق انسانی انجام می‌گیرد.

بی‌شك حیات مناسب انسانی در سایه انتخاب‌های منطقی و نزدیک\_شدن انسان به چیزی است که در بخش قبل، «سخنگوی دانا» نامیدیم. برای رسیدن به این هدف، لازم

است ذهن منطقی انسان شناخته و تقویت شود، و نیز، برای افکار و تصمیم‌گیری‌ها به درستی مورد استفاده قرار گیرد.

## منطق اجرای دستورات

در متن منسو، دستوراتی مانند `add`, `pick` و... به کار رفته‌اند. همین‌طور، زبان‌های برنامه‌نویسی، مملو از دستورات گوناگون است. بنابراین، خوب است بدانیم دستور و منطق حاکم بر دستور چیست. در گذشته، فرض ما این بود که خواننده، به صورت غریزی با استفاده و معرفی دستورها مشکلی ندارد، اما وارد بحث ماهیت آن نشدیم.

### ارتباط بین دستور و زمان

کار یک دستور<sup>۶</sup> ایجاد یک تفاوت است. برای نمونه، وضعیت لیست گزاره‌های درست قبل و پس از:

`add p;`

متفاوت است.

پس، دستور به معنی «ایجاد\_کننده تفاوت» است. این ایجاد\_کننده تفاوت، یک «زمان اجرا» دارد. قبل از شروع زمان اجرا، تفاوتی به وجود نیامده است و پس از زمان اجرا، تفاوت به وجود آمده است. بنابراین، اجرای دستورات، در ارتباط تنگاتنگ با زمان است.

همان‌گونه در فصل اول گفتیم، مفهوم «تفاوت»، ممکن است یک عارضهٔ تکاملی ذهن و واقعیت خارجی چیز دیگری باشد؛ در مورد «زمان» نیز می‌توانیم همین را بگوییم. ما تنها می‌توانیم زمان را مفهومی در ذهن انسان بدانیم و هرگز با اطمینان نمی‌توانیم از واقعیت‌های فیزیکی سخن بگوییم.

ذهن، می‌تواند متوجه تفاوت گردد. فرض کنید در ذهن یک شیء ذهنی ایجاد شده است به نام «پدر». با گذشت سال‌ها ذهن متوجه می‌شود که با گذشت هر سال، پدری که در ذهن اوست با واقعیتی که از چشم دریافت می‌کند «کمی» متفاوت شده است (پیرتر شده

<sup>6</sup> command

است). بنابراین، ذهن، متوجه تفاوت‌های «اندک» در شیء ذهنی می‌گردد. تفاوت در شیء ذهنی به معنی اتصال به یک ویژگی جدید یا گستین آن از یک ویژگی قبلی است. «تفاوت اندک» به این معنی است که ویژگی‌های تغییرکرده آنقدر نیستند که لازم به ایجاد شیء ذهنی جدیدی باشد. اگر «پدر» وارد اتفاقی شود در را ببندد و وقتی بیرون می‌آید، شخص کاملاً متفاوتی با ظاهری کاملاً متفاوت باشد، ذهن قبول نمی‌کند این شخص همان «پدر» است و شیء ذهنی جدیدی برای آن به وجود می‌آورد. اما در «تغییرات تدریجی» ذهن موجودیت جدیدی به وجود نمی‌آورد و موجودیت قبلی را تصحیح می‌کند.

به این ترتیب، ذهن تفاوتی بین «تفاوت اندک» و «تفاوت زیاد» قائل است. ذهن با ادامه حیات متوجه تفاوت‌های اندک و مرحله‌به\_مرحله، تقریباً در همه چیز می‌گردد. ذهن در طی تکامل تدریجی، مجّهـز این قابلیت شده است که با توجه به خاطراتش، از مراحلی که (از گذشته) مشاهده و در خود ثبت کرده است، انتظاری از ایجاد مراحلی داشته باشد که هنوز در ذهنش ثبت نشده‌اند (یعنی تصویری از مراحل آینده داشته باشد). یعنی ذهن این‌گونه تکامل یافته است که قوانینی برای یک آینده ذهنی داشته باشد.

در حقیقت، یک «تغییر» برای ذهن، یک موجودیت ذهنی است و ذهن، دنباله‌ای از تغییرات را در خود ذخیره کرده است. ذهن، برای «تغییر»، نوعی فاصله در نظر می‌گیرد: یک تغییر را در گذشته دور و تغییر دیگر را در گذشته نزدیک می‌داند. مفهوم فاصله زمانی نیز در طی تکامل تدریجی ذهن شکل گرفته است. ذهن، در طی همین تکامل، به این توانایی رسیده است که می‌تواند تغییرات محضی را در نظر بگیرد که (هنوز) روی نداده‌اند اما طبق قوانینی مشخص (شبیه و شامل قوانین منطقی استدلال)، انتظار دارد که در آینده روی دهد که این آینده، می‌تواند دور یا نزدیک باشد. به این ترتیب، مفهوم فاصله زمانی و زمان، یک مفهوم ذهنی است که به\_وجود\_آمدن آن، نتیجه تکامل تدریجی و دلیل آن ادامه زیست بوده است. مانند قوانین منطق، این مفهوم ممکن است به واقعیتی فیزیکی بسیار نزدیک باشد و یا مانند «رنگ» قابلیتی غیر\_اصیل و ساختگی در ذهن باشد.

فیزیک‌دانان، با بررسی پدیده‌ها، متوجه شده‌اند که می‌توانند قوانین منطقی ساده‌ای را، با «زمان» پیوند بزنند تا طبق آن قوانین، بتوانند آینده را، با دقت زیادی پیش‌بینی کنند.





«آینده»، مراحلی از تغییرات محض ذهنی است که انتظار داریم روی دهنده و از حالت محض به حالتی عینی در آیند.

حال، بر می‌گردیم به مفهوم «دستور». اجرای یک دستور، به معنی استفاده از یک قانون برای ایجاد تغییرات مشخصی در آینده است. وقتی یک تغییر مطابق با یک دستور (= قانونی خاص) عینیت می‌یابد، می‌گوییم آن دستور (یا قانون) «اجرا» شده است.

### مفهوم اجرای یک دستور

در مورد اجرای قوانین فیزیک معمولاً نمی‌توانیم دلیلی پیدا کنیم. برای مثال، نمی‌دانیم چرا برخی ذرات مثل فوتون با ذرات بوزن هیگر ارتباطی برقرار نمی‌کنند و جرم ندارند و برخی ذرات دیگر ارتباط برقرار می‌کنند و جرم دارند. حتی اگر در این موارد، دلیلی پیدا شود و دلایل واکاوی شوند، احتمالاً به دلایلی می‌رسیم که همچون گزاره‌های اولیه، خود، فاقد دلیل هستند (یا دنباله‌ای بی‌پایان از سلسله دلایل موجود است<sup>7</sup> و از واکاوی باز می‌مانیم).

قوانین فیزیک، مدام در حال اجرا\_شدن هستند، بدون این که بدانیم چرا و شاید این دلیل\_خواستن، تنها یک عارضه تکاملی ذهن باشد. در هر حال، قوانین فیزیک اجرا می‌شوند. معمولاً اگر در اجرای یک قانون فیزیک یا دسته‌ای از قوانین فیزیک خود یا دیگری را دخیل بدانیم، به آن قانون «دستور» می‌گوییم. ما یا دیگران دستورات را انجام می‌دهیم یا اجرا می‌کنیم؛ این یعنی خود را در انجام آن دخیل می‌بینیم و احساسی نیست به آن داریم که آن را «اختیار» نامیده‌ایم. هرچند می‌توان احساس اختیار را تنها عارضه‌ای از تکامل دانست؛ اما در اینجا می‌توانیم آن را واقعی بدانیم.

پس گاه در اجرای دستورات تأثیری نداریم؛ مثلاً در افتادن یک قطره باران از ابر ما هیچ دخالتی نداریم و همه چیز خارج از اراده ما روی می‌دهد و گاه در مورد اجرای برخی قوانین فیزیک حس می‌کنیم که اراده و اختیار داریم، مثل حرکت\_دادن دست. به دسته‌ای از

<sup>7</sup> برخلاف ادعاهای فلاسفه قدیمی سلسله بی‌نهایت از دلایل که آن را تسلسل به راحتی قابل\_رد نیست. اما باید توجه کنید که مدل منسو بر اساس سلسله‌ای متناهی از دلایل است و هر کد به زبان منسو متناهی است. با این حال، این ممکن است ناشی از محدودیت ذهن انسان در در\_نظر\_گرفتن سلسله بی‌نهایت از دلایل باشد و معلوم نیست واقعیات خارجی از این محدودیت پیروی کنند.



قوانين که کاری ساده را تحت اختیار و خواست ما انجام می‌دهند، «دستور» می‌گوییم. به دسته‌ای از دستورات کنار هم که کار بزرگ‌تری را انجام می‌دهند و ما را به هدفی خاص می‌رسانند «الگوریتم» می‌گوییم. اما باید دانست که اجرای یک دستور یا الگوریتم چیزی نیست جز اجرای قوانین فیزیک که مدام در حال اجرا هستند و ما دلیلی برای این روند اجرا نمی‌یابیم.

### زبان خاص برای دستورات

همان‌گونه که گزاره‌ها دسته‌بندی می‌شوند، در رایانه یا در هر جای دیگری که مربوط به کارهای انسانی ماست، دستورها را نیز می‌توان دسته‌بندی کرد و برای هر دسته یک کلمهٔ خاص نوشتاری ایجاد کرد؛ مثل دستور `add`. دستور `add`، در حقیقت، دسته‌ای از قوانین فیزیکی برای ایجاد یک تغییر است. همین‌طور هر دستور دیگر.

در برنامه‌نویسی، دستورات شرطی و حلقه‌ها نیز وجود دارند. دستور شرطی، دستوری است که قبل از اجرا درستی یک گزاره را بررسی می‌کند و اگر درست بود تغییری را به وجود می‌آورد. در واقع، دستور شرطی خود یک دستور بررسی درستی گزاره را نیز در بر دارد.

در این کتاب قصد نداریم منطق حاکم بر دستورات را به صورت کامل تبیین کنیم. اما باید بدانیم که اجرای هر دستور ممکن است تغییری در مجموعهٔ گزاره‌های درست یا نادرست به وجود آورد. از نظر منطقی می‌توان یک دستور را «ابزاری برای تبدیل یک گزارهٔ بلا تکلف به گزاره‌ای درست یا نادرست» دانست.

### شبیه‌سازی تفکر

گفتیم که تفکر می‌تواند چیزی مشابه ساختن و بررسی متن‌های منسو باشد. اگر دستورات مربوط به تفکر و الگوریتم‌های تفکر به صورت دقیق تبیین شوند می‌توان مدل منسو را به مدلی توسعه داد که خود برای اثبات گزاره‌هایی خاص متن‌هایی می‌نویسد. برای مثال؛ می‌توان الگوریتم‌هایی برای یک ریاضیدان نرم‌افزاری نوشت که طبق آنها به پیدا\_کردن اثبات‌ها می‌پردازد. فرآیند تفکر در ذهن انسان؛ اجرای دستوراتی شبیه به دستورات زبان منسو است که مدام و بی‌اراده اجرا می‌شوند. شبیه‌سازی این دستورات به سادگی امکان‌پذیر نیست؛ زیرا قدرت و روش تفکر در افراد مختلف متفاوت است.



در شبیه‌سازی سیستم شنیداری انسان، که انسان با آن کلمات را تشخیص می‌دهد، دانشمندان دست به دامان الگوریتم‌های تکاملی شده‌اند. الگوریتم تکاملی، الگوریتمی است که مانند تکامل تدریجی، خود به خود، بر اساس آزمایش و خطا تکامل می‌یابد. به این شکل حتی طراح نرم‌افزار تشخیص صدا، خود، نمی‌داند نرم‌افزار دقیقاً چگونه کلمات را تشخیص می‌دهد و او تنها زمینه را برای ایجاد یک دیتابیس مناسب برای تشخیص صدا ایجاد کرده است. ممکن است برای ایجاد نرم‌افزاری برای تفکر نیازمند چنین الگوریتم‌هایی شویم. اما اگر از این راه استفاده نشود، دانش بیشتری در شبیه‌سازی خواهیم داشت

