فصل سوم

جستجوی رقابتی

هدف از جستجوی رقابتی، یافتن پاسخ در یک گراف فضای حالت است، اما در این شرایط رقیب یا رقبائی نیز وجود دارند که می‌خواهند به پاسخ برسند در اینجا مهم نیست که آیا کوتاهترین راه‌حل برای رسیدن به پاسخ را یافته‌ایم یا خیر بلکه مهم آنست قبل از رقبا به هدف برسیم.

بدلیل این تغییر در صورت مسئله، راه‌حل اگر چه شباهت کلی با روش‌های جستجو در گراف فضای حالت دارد ولی تا حدودی متفاوت خواهد بود. قبل از آنکه بخواهیم به روش حل این دسته از مسائل بپردازیم البته لازم است تا نگاهی به انواع این دسته از مسائل و محدودیت‌های مطرح شده در راه‌حل پیشنهادی داشته باشیم.

بازی شطرنج نمونه بسیار خوبی برای بیان روش‌های جستجوی رقابتی سنتی است. در بازی شطرنج متوسط فاکتور انشعاب ۳۵ تخمین زده می‌شود و معمولاً یک بازی تا ۵۰ حرکت برای هر بازیکن ادامه خواهد داشت. پس اگر گره‌های موجود در گراف فضای حالت را بخواهیم تخمین بزنیم، در حدود $۳۵^{۱۰۰}$ گره یا وضعیت در بازی شطرنج می‌تواند وجود داشته باشد، اما بازی ساده‌ای همانند tic – tac – toc تنها !۹ حالت را شامل می‌شود. به همین دلیل شطرنج نیاز به تفکر عمیق دارد در حالی که بازی tic – tac – toe (یا o-x) بازی بسیار ساده‌ای است. یکی از کاربردهای عمده جستجوی رقابتی، استفاده از آن در بازیهای فکری است، اما آنچه که در این بخش بیان می‌شود شامل گروه محدودی از بازیها خواهد شد، ببینیم این محدودیت در نظر گرفته شده دربازی چه هستند؟

اولین نکته آنست که بازی ۲ نفره (در حالت خاص چند نفره) در نظر گرفته می‌شود، یعنی تیم‌خودی و رقیب در کار نخواهد بود. نوبت در بازی بر اساس قوانین بازی بین ۲ بازیکن خودی و رقیب جابجا خواهد شد.

بازیکن در هر لحظه قادر است تمام صفحه بازی را مشاهده کند و بعبارت دیگر هیچ اتفاقی در بازی برای دو بازیکن پوشیده نیست برد، باخت و مساوی در بازی طبق قوانین روشنی تعریف شده است. عامل تصادف در بازی (همانند، سکه، ورق، تاس) وجود ندارد، البته می‌توان این تئوری را بازی‌های تصادفی نیز توسعه داد که در انتها به آن پرداخته شده است.

آنچه گفته شد بخش مهمی از محدودیت‌های لازم در بازی مورد نظر است. بازی شطرنج مثال خوبی از انواع بازی است که تمام این محدودیت‌ها را رعایت می‌کند ولی دقیقاً بازی فوتبال در نقطه قابل مقابل قرار دارد و الگوریتم‌های مطرح شده در این بخش برای روبات فوتبالیست کاربردی ندارد.

بر خلاف گراف فضای حالت که برای مدل‌سازی مسائل جستجو بکار می‌رفت، در این مسئله خاص از درخت بازی (game tree) استفاده می‌شود. طبق تعریف، درخت بازی شامل تمامی حرکات ممکن از وضعیت آغازین برای بازیکن خودی و رقیب است. برگ‌های این درخت جائی است که بازی با برد، باخت یا مساوی به اتمام می‌رسد.

اگر برای یک بازی درخت آن رسم شود، می‌توان با تحلیل درخت مسیرهای منجر به باخت را شناسائی کرد و از آنها پرهیز نمود، متقابلاً با یافتن مسیرهای منجر به برد حرکت مورد نظر را برای رسیدن به آن وضعیت‌ها انتخاب کرد. از آنجا که بازی‌های مورد بحث دو نفره هستند، و معمولاً نوبت یک در میان بازیکن خودی و رقیب تغییری کند، درخت بازی شکل خاص به خود می‌گیرد که به درخت بازی MinMax شهرت دارد.

در این درخت، ریشه شامل وضعیت بازی از صفحه بازی است. پس تمامی سطوح فرد درخت شامل گره‌های مربوط به بازیکن اول و تمامی گره های موجود در سطوح زوج مربوطه به بازیکن دوم است. معمولاً بازیکن اول خودی و دوم رقیب می‌باشد.

اگر برای یک بازی درخت آن رسم شود، می‌توان با تحلیل درخت مسیرهای منجر به باخت را شناسائی کرد و از آنها پرهیز نمود، متقابلاً با یافتن مسیرهای منجر به برد حرکت مورد نظر را برای رسیدن به آن وضعیت‌ها انتخاب کرد. از آنجا که بازی شکل خاصی به خود می‌گیرد که به درخت بازی MiniMax شهرت دارد در این درخت. ریشه شامل وضعیت جاری از صفحه بازی است. پس تمامی سطوح فرد درخت شامل گره‌های مربوط به بازیکن اول و تمامی گره‌های موجود در سطوح زوج مربوط به بازیکن دوم است. معمولاً بازیکن اول خودی و دوم رقیب می‌باشد.

در این درخت، بازیکن اول (خودی) Max و بازیکن دوم (رقیب) Min نامیده می‌شود. برگ‌های این درخت شامل حالات برد، باخت یا مساوی است که به آنها اعدادی نسبت داده می‌شود. معمولاً برد را با $+\infty$، باخت با $-\infty$ و مساوی با صفر مشخص می‌شود. الگوریتم زیر براحتی قادراست با کمک مقادیر نسبت داده شده به برگ، تصمیم‌گیری کند که حرکت مناسب بعدی در ریشه چه خواهد بود.

```
search(node u)
{
if(u is a leaf)
return score of u;
elseif (u is a min node)
for(all childs of u : V1, V2,..., Vn)
return Min[search (V1),..., Search (V(n)];
else
for(all childs of u)
return Max[Search (V1),..., Search(Vn)];
}
```


این الگوریتم قادر خواهد بود تا معین کند کدامیک از فرزندان ریشه بالاترین امتیاز را دارد. لذا فرزندی بعنوان حرکت بعدی انتخاب خواهد شد که بیشترین امتیاز را داشته باشد

در بازی‌های پیچیده همانند شطرنج روش فوق اجرائی نیست. چرا که در این بازی‌ها تعداد گره‌های درخت بسیار زیاد بوده و زمان جستجو بشدت طولانی خواهد بود. دقت کنید زمان جستجو در الگوریتم مطرح شده، وابسته به تعداد گره های درخت بازی است و قبلاً دیدیم که این تعداد گره در درخت بازی شطرنج و یا دیگر بازی‌های مشابه بسیار زیاد است.

در عمل درخت بازی هیچگاه تا حصول برگ توسعه داده نمی‌شود، بلکه به فراخور زمان ممکن و امکانات سخت افزاری تا عمق معینی بسط داده خواهد شد. گره‌های برگی این درخت تحت چنین شرایطی برگ‌های واقعی درخت بازی نیستند، بلکه گره‌های بینابینی از وضعیت بازی خواهند بود. پس تابعی مورد نیاز است تا مشخص کند این گره‌های برگی (که واقع بینابینی هستند) چه وضعیتی دارند. بعبارت دیگر آیا در وضعیت مورد بحث شانس برد رقیب بیشتر است یا بازیکن خودی؟

تابعی بنام تابع سودمندی (utility) و یا در برخی مراجع بنام تابع امتیاز (payoff) تعریف شده که این تابع ارزیابی گره (وضعیت) از درخت بازی را بر عهده دارد. اما چگونه می‌توان این تابع را برای یک بازی معین نوشت. این تابع از رابطه زیر بدست می‌آید:

P(x)= شانس برد خودی - شانس برد رقیب

اگر x وضعیت جاری باشد، p(x) برایند شانس برد خودی نسبت به رقیب خواهند بود.

در چنین شرایطی اگر p(x) مقدار مثبتی باشد وضعیت برای بازیکن خودی مناسب بوده و اگر این مقدار منفی باشد، وضع به نفع رقیب است. مقدار صفر معرف وضعیت متوازن برای هر دو طرف خواهد بود برای مثال به بازی x − o وضعیت زیر دقت کنید:

O	X	

p(x) = ۶ − ۴ = +۲

p(x) = ۳ − ۱ = +۲

اگر بخواهیم شانس برد خودی را تخمین بزنیم می‌توان برای این بازی شانس اشغال یک سطر، ستون یا قطر برای x را در نظر گرفت (x خودی است). دراین مثال، دو قطر موجود و ستون دوم تنها ممکن است توسط x اشغال شوند و متقابلاً o تنها قادر است ستون اول را اشغال کند. البته سطر اول، سوم و ستون سوم نیز چون خالی است می‌توانند هم توسط x و هم o اشغال شود. پس محاسبه p(x) بصورت تعداد سطر، ستون یا قطرهائی که x می‌تواند اشغال کند منهای همین مقادیر برای O بدست می‌آید. دقت کنید در محاسبه p(x) این مثال می‌توان سطر، ستون یا قطرهای خالی را هم در محاسبه حساب کند (۶-۴) و یا اینکه از محاسبات حذف نمود (۱-۳) چون این مکان‌ها می‌توانند بوسیله هر دو مهره اشغال شوند و در حاصل تفاضل تأثیری نمی‌گذارد.

یک نکته مهم را نباید فراموش کرد، اگر هوشمندی x را در این حد اشاره شده فرض کنیم، **چه دلیلی وجود** دارد که رقیب (O) نیز به همین طریق فکر کند. در واقع ما می‌دانیم رقیب چگونه فکری کند و **شما می‌دانید** بازیکن خوب کسی است که بتواند حرکات رقیب خود را پیش‌بینی کند. این فرایند در **سطح الگوریتم‌های** هوشمند، به مدل‌سازی رقیب (opponenent modeling) شهرت دارد، اما در حد این نوشتار **فرض کنیم** هوشمندی رقیب همانند خودی است، یعنی از همان خلاقیت بازیکن خودی بهره گرفته است. **این فرض** برای ساده سازی است و در واقع رقیب احتمالاً به گونه‌ای دیگر فکر می‌کند. تحت شرایطی که **رقیب ابتکار** بهتری از خودی داشته باشد و یا برعکس بسیار ضعیف‌تر از خودی فکر کند، این پیش‌فرض **می‌تواند منجر** به تصمیمات غیر مناسب گردد.

الگوریتم جستجوی MinMax تحت این شرایط به قرار زیر است:

۱- درخت بازی تا جای ممکن توسعه پیدا کند.

۲- برای گره‌های پایانی در ان درخت بکمک تابع امتیاز (سودمندی)، مقادیر ارزش هر گره **محاسبه شود.**

۳- از برگ به سوی ریشه برگردیم، در هر سطح تشخیص داده شود آیا گره Min یا Max است. **برای** گره‌های Min بین فرزندان حداقل گیری و برای گره‌های Max بین فرزندان حداکثر گیری **صورت پذیرد.** این عمل تا ریشه درخت سطح به سطح تکرار شود.

۴- در نهایت ریشه مقداری به خود می‌گیرد که باید با یکی از فرزندان برابر باشد. تحت این **شرایط، حرکت** پیشنهادی بعدی، به سوی آن فرزند خواهد بود.

اگر عمق درخت بازی را با m و فاکتور انشعاب b باشد، زمان جستجو از $O(b^m)$ خواهد بود. **اگر چه** بنظر می‌رسد این روش جستجوی MinMax سطحی است ولی برای بهینگی فضای معرفی، **آنرا بر مبنای** DFS پیاده‌سازی می‌کنند تا فضای مصرفی به $O(bm)$ تقلیل یابد.

الگوریتم MinMax اگر چه باید به جستجو در درخت بازی بپردازد ولی در عمل کاربردی **ندارد. این** الگوریتم گره‌های بسیاری را توسعه می‌دهد که نیازی به بسط آنها نبوده است. در عمل **می‌توان برخی از** گره‌های مورد بحث که توسط MinMax توسعه داده می‌شوند را هرس نمود. **الگوریتمی که به هرس** نمودن گره‌های درخت بازی حین جستجو می‌پردازد $\alpha - \beta$ prunning نام دارد.

ایده اولیه الگوریتم هرس آلفا - بتا ساده است: بجای جستجوی کلیه گره‌ها تا عمق m، در حین **جستجو بر** روی درخت بازی بر روش depth-limited و تا عمق m، مقادیر موقتی به گره‌های max **نسبت داده** می‌شود که آلفا و مقادیر نسبت داده شده به گره‌های Min ، بتا نام دارد. دو اصل زیر در **مورد این مقادیر** برقرار است:

اصل ۱- در حین جستجو، مقادیر منتسب به گره‌های Max (آلفا)، هیچگاه کاهش پیدا نمی‌کنند.

اصل ۲- در حین جستجو ، مقادیر منتسب به گره‌های Min (بتا)، هیچگاه افزایش پیدا نمی‌کنند.

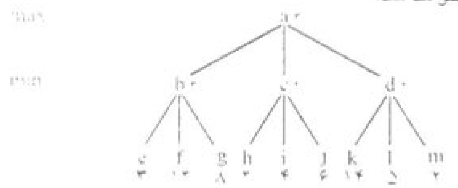
برای شروع جستجو آلفا - بتا، ابتدا تا عمق m بصورت DFS در مسیر معینی **(معمولاً دست چپ‌ترین** مسیر ممکن) حرکت کرده تا به گره عمق m برسیم. با استفاده از تابع امتیاز به این گره **مقدار مناسب را** نسبت داده و سپس برای تمامی گره‌های خواهر- برادر وی نیز این کار انجام خواهد شد. **فرض کنید این**

گره‌ها همگی در سطح Min می‌باشند. مقدار حداکثر بین این مقادیر به والد نسبت داده خواهد شد (بدیهی است این گره Max است)، سپس همین مقدار بصورت موقت به پدر بزرگ نیز نسبت داده می‌شود تا ... برش بررسی گردد.

حال الگوریتم از زیر شاخه بعدی به سطح نودها خواهد رسید و اگر در این زیر شاخه نوده‌ای ... بتا بزرگتر و یا مساوی پدر بزرگ خود داشته باشد، جستجو نخواهد شد. همانند همین ... گره‌های Max با مقادیر آلفا نیز روی می‌دهد. اگر بخواهیم فرایند تشریح شده فوق را بصورت ... کنیم، دو قانون زیر قابل حصول است:

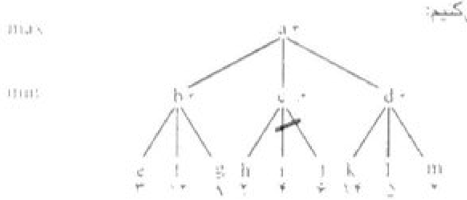
قانون ۱ - عمل جستجو تحت گره Min ای که مقادیر بتا آن کوچکتر و یا مساوی مقادیر آلفا هر گره ... اجداد آن است، متوقف خواهد شد.

قانون ۲ - عمل جستجو تحت گره Max ای که مقادیر آلفا آن بزرگتر یا مساوی مقادیر بتا هر گره ... اجداد آن است، متوقف خواهد شد.



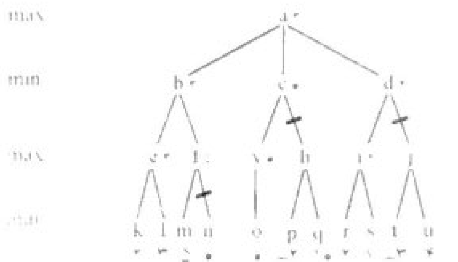
مثال:

درخت بازی با مقادیر تابع امتیاز در برگ‌ها که به روش MinMax حل شده، حال همین مسئله را به روش هرس آلفا - بتا حل می‌کنیم:



در گره c و بعداز بازگشت از گره h، مقدار موقت $\beta = ۲$ را به خود می‌گیرد که گره a دارای مقدار ... $\alpha = ۳$ بوده است. پس گره‌های i و j هرس خواهد شد.

مثال:



چند نکته در مورد هرس آلفا - بتا را نباید فراموش کرد:

۱- احتمال وقوع برش در درخت بازی بسته به مقدار بازگشتی توابع امتیاز است که بر روی گره‌ها اعمال می‌شود. در هر صورت وقوع برش تصادفی بوده و ممکن است در یک مثال اصولاً برشی رخ ندهد.

۲- انتخاب فرزند بعدی در روش DFS که بنیان این روش نیز است تصادفی است. معمولاً در مثال‌ها فرزندان بترتیب از چپ به راست نیز مورد آزمون قرار می‌گیرند، ولی در برخی از مراجع فرزندان بترتیب از راست به چپ نیز مورد آزمون قرار گرفته‌اند. جهت توسعه فرزندان می‌تواند در محل برش موثر باشد.

۳- الگوریتم هرس آلفا- بتا تنها روشی برای بهبود کارائی الگوریتم MinMax است و با فرض شرایط برابر، می‌توان ثابت کرد که این الگوریتم برای یک مسئله مشخص همواره یک راه‌حل را پیشنهاد خواهند کرد، اما بدیهی است که هرس آلفا- بتا احتمالاً زودتر به پاسخ خواهد رسید.

۴- هرس آلفا- بتا چقدر در زمان محاسبه بهبود خواهد بخشید؟ برای محاسبه این امر ابتدا باید تعریفی را مطرح کنیم. فاکتور موثر انشعاب متوسط درجه خروجی گره‌های درخت بازی پس از برش‌های انجام شده با عمق d است. همانطور که گفته شود توالی مقادیر حاصل از تابع امتیاز در میزان برش‌ها موثر خواهد بود و در کل نمی‌توان همواره ضمانتی برای برش داد. اما در حالت ایده‌آل فاکتور انشعاب از b به $\sqrt{b}$ کاهش پیدا می‌کند و مرتبه جستجو از $O(b^{m})$ به $O(b^{\frac{m}{2}})$ کاهش پیدا می‌کند.

۵- بهتر است برای افزایش احتمال قطع (برش) در درخت بازی، برای هر گره Max فرزندان را بترتیب نزولی و گره‌های Min را بترتیب صعودی بر مبنای مقدار تابع امتیاز برای آن گره ملاقات کرد. این عمل معمولاً منجر به افزایش حجم برش صورت گرفته در درخت بازی خواهد شد.

پس از مطرح الگوریتم هرس آلفا- بتا، به مرور تغییراتی جهت بهبود بر روی الگوریتم MinMax مطرح شد که برخی از آنها به قرار زیر است:

۱- **انتظار برای سکون (Quiescence):** گاهی بعلت ماهیت بازی ممکن است گره‌ای در سطح معینی دارای تابع امتیازی باشد، که اگر آن گره را تا چند سطح دیگر بسط دهیم، به این نتیجه خواهیم رسید که تابع امتیاز در مورد آن گره بشدت اشتباه کرده است. برای مثال اگر گره b وضعیتی در ابتدای حرکت جابجائی مهره‌ها در یک بازی (مثل شطرنج) باشد، ارزش بازگشتی تابع امتیاز می‌تواند برای مثال ۴- باشد، اما ادامه این مسیر و بسط گره b (برای مثال تا دو حرکت بعد) نشان می‌دهد که مقدار گره b ‏+۶ خواهد شد. این تغییر ناگهانی می‌تواند حرکت بد را به حرکت خوب و یا برعکس تبدیل کند.

در مقابله با این پدیده، می‌بایست بسط درخت بازی را تا گره‌هائی ادامه دهیم که اصطلاحاً سکون یافته‌اند یعنی چنین تغییرات ناگهانی در آنها روی نخواهد داد.

۲- **جستجوی ثانویه:** از آنجا که در مسئله سکون نیز دریافتیم که اگر درخت بازی تا عمق معینی توسعه یابد، این احتمال وجود دارد که میزان امتیاز گره‌های برگی دقت کافی را نداشته باشند. برخی پیشنهاد داده‌اند توسعه درخت ادامه یابد، این کار برای تمامی گره‌ها به دلایلی که گفته شد امکان پذیر نیست، اما

می‌توان درخت را تا عمق معینی بسط داد و پس از انتخاب گره مناسب، زیر شاخه منشعب شده از آن گره را (ونه دیگر گره‌ها) تا چند حرکت دیگر بسط داد و از عدم وجود حالاتی همانند عدم سکون اطمینان یافت این جستجو پس از انتخاب پاسخ و برای اطمینان از انتخاب، جستجوی ثانویه نام دارد.

۳- **هرس رو به جلو:** برای کاهش اندازه درخت و ایجاد روشی ضعیف از جستجوی انتخابی، روش‌هائی برای حذف برخی زیر شاخه‌ها مطرح شده است. برای مثال پیشنهاد شده برای هر گره N بهترین حرکت بعدی انتخاب شود و تنها همین فرزندان توسعه داده شوند، میزان N معمولاً با فاصله گرفتن از ریشه کاهش پیدا می‌کند. در ایده‌ای مجزا (الگوریتم گاما و هرس روی به جلوی حاشیه‌ای) پیشنهاد شده حرکاتی حذف شوند که مقدار امتیاز آنها بدتر از بهترین مقادیر گره‌های جستجو شده باشد، چرا که تصور بر این است که حرکت رقیب تنها در جهت خراب شدن اوضاع برای خودی خواهد بود. هر کدام از این روش‌ها با این خطر روبرو هستند که با حذف یک زیر شاخه یک گره مناسب برای حرکت بعدی را از دست بدهند

۴- **عمیق شونده تکراری:** در برخی از مسابقات که بصورت تورنمنت برگزار می‌شود، معمولاً زمان بازی محدود است و بازیکن تنها حق فکر کردن در یک بازه محدود زمانی را دارد. تحت چنین شرایطی می‌توان از ایده جستجوی عمیق‌شونده تکراری استفاده کرد که قبلاً مفصلاً در مورد آن صحبت شده است. می‌توان درخت بازی را بر اساس این دیدگاه، سطح به سطح توسعه داد و هر بار تا عمق یعنی جلو رفت. یعنی سری‌های یک درخت را تا سطح معینی بسط داده و اگر فرصتی باقی مانده باشد تا عمق بیشتری نیز جلو رود نکته مهم آنست که هر بار اعمال الگوریتم منجر به تولید جوابی خواهد شد که اگر فرصت برای توسعه درخت تا عمق بیشتر وجود نداشت، حداقل یک پیشنهاد برای حرکت بعدی قبلاً تولید شده است

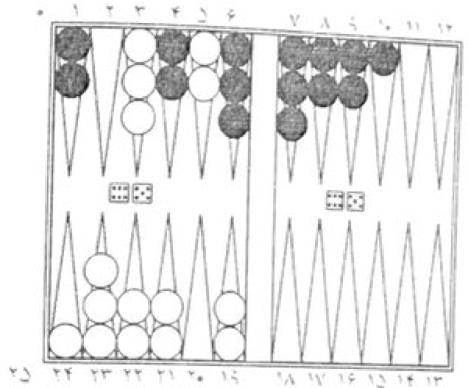
با توجه به تمام نکات مطرح شده و انواع توسعه‌های اشاره شده به این روش حل مسائل در جستجوی رقابتی، روش بنیادی MinMax با مشکلاتی اساسی روبرو است که از جمله این مسائل می‌توان به پدیده افق (Horizon effect) اشاره کرد. در این پدیده وقوع یک رویداد بد در بازی بوسیله حرکات جنبی مخفی شده، بگونه‌ای وقتی MinMax درخت بازی را تا عمق معینی توسعه می‌دهد، متوجه این رویداد بد نخواهد شد. با استفاده از یافتن وضعیت‌های سکون و ادامه زیر شاخه تا چنین وضعیت‌هائی می‌توان تا حدودی جلوی این پدیده را گرفت ولی واقعیت آنست که برای مقابله با چنین پدیده‌ای روش مطمئنی پیدا نشده و جزو مسائل حل نشده باقی مانده است.

همانطور که قبلاً نیز اشاره شد، یکی از فرضیات مهم در روش MinMax استفاده از این فرض است که رقیب بر مبنای تابع امتیاز مطرح شده، حرکت بهینه را انجام خواهد داد. اگر بازیکن خودی در وضعیت برد باشد، این سیاست منطقی بنظر می‌رسد و مسلماً منجر به برد خواهد شد ولی اگر بازیکن خودی در وضعیت باخت باشد نیاز به ریسک است تا اگر رقیب اشتباهی کرد، از فرصت بدست آمده استفاده شود. بدیهی است برای انجام ریسک مناسب باید مدل‌سازی از رقیب صورت گرفته باشد و همانطور که قبلاً نیز اشاره شد مدل‌سازی رقیب یکی از مباحث یادگیری ماشینی است و از بحث این نوشتار خارج می‌شود

بازی‌هایی که شامل عنصر شانس هستند

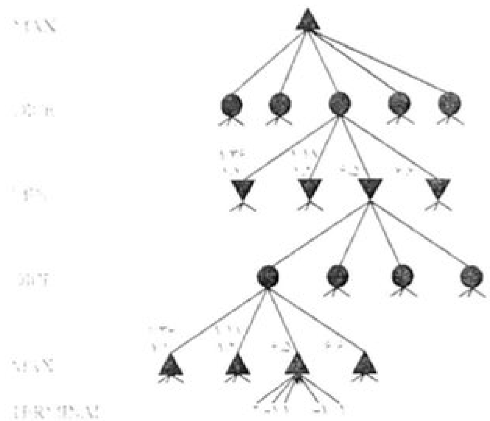
بازی‌هایی که شامل عنصر شانس هستند در زندگی واقعی، بر خلاف شطرنج، حوادث غیر قابل پیش‌بینی زیادی وجود دارند که ما را در شرایط غافلگیرانه قرار می‌دهند. بازیهای زیادی این غیرقابل پیش‌بینی بودن را توسط یک عنصر تصادفی مانند پرتاب تاس، نشان می‌دهند. از این طریق، آنها ما را یک قدم به واقعیت نزدیکتر می‌کنند، و ارزش این را دارد که ببینیم چطور این مطلب بر روی فرآیند تصمیم‌گیری اثر می‌گذارد.

تخته نرد یک بازی عمومی است که شانس و مهارت را با هم ترکیب می‌کند. تاسها در ابتدای شروع بازی توسط بازیکنی که نوبتش است ریخته می‌شوند تا مجموعه‌ای از حرکات که قابل انجام هستند، تعیین شود. در موقعیت تخته نرد شکل ۲۴، سفید ۶ و ۵ آورده است و ۴ حرکت امکان‌پذیر را می‌تواند انجام دهد.



شکل ۲۴- نمایش از بازی تخته نرد

اگر چه سفید از حرکات قانونی‌اش مطلع است، اما از عملکرد سیاه در مرحله بعدی خبر ندارد، و از این رو از حرکات سیاه شناختی ندارد. این بدان معنی است که سفید نمی‌تواند درخت بازی کاملی از نوعی که برای شطرنج و Tic-Tac-Te دیدیم، داشته باشد. درخت بازی در تخته نرد باید شامل گره‌های شانس برای گره‌های MIN و Max باشد. گره‌های شانس مانند دوایری که در شکل ۲۵ نشان داده شده‌اند. شاخه‌ها از هر گره شانسی تاسهای ممکن را مشخص می‌کنند، و هر کدام با شانسی که دارند، برچسب خورده‌اند. ۳۶ راه برای پرتاب تاس وجود دارد، اما از آنجایی که ۵ و ۶ با ۶ و ۵ برابر است در مجموع ۲۱ روش مختلف می‌شد. شش جفت (از ۱-۱ تا ۶-۶) ۱/۳۶ شانس دارند در صورتی که بقیه ۱۵ روش، شانسی معادل ۱/۱۸ دارند.



شکل ۲۵- نمونه‌ای از درخت بازی تخته نرد

مرحله بعدی فهم چگونگی ساخت تصمیمات صحیح است. روشن است که ما هنوز می‌خواهیم حرکتی از $A_1,\ldots,A_n$ انتخاب کنیم که به بهترین موقعیت منجر شود. بهرحال، هر کدام از مواقع ممکن مقدار minimax قطعی ندارد (که در بازیهای قطعی سودمندی برگی بود که توسط بهترین بازی به آن می‌رسیدیم). در عوض، فقط می‌توانیم میانگین یا مقدار انتظاری را محاسبه کنیم که میانگین بر اساس تمام اعداد تاس که پدید می‌آیند، گرفته می‌شود.

محاسبه مقادیر انتظاری گره‌ها، صریح است. برای گره‌های پایانی، از تابع سودمندی مانند بازیهای قطعی استفاده می‌کنیم. با یک مرحله پیش‌رفتن در درخت جستجو، به یک گره شانس برخورد می‌کنیم. در شکل ۲۵ گره‌های شانسی دوایر هستند و آنهایی در نظر گرفته می‌شوند که با C برچسب خورده‌اند. فرض کنید که d_i یک پرتاب تاس و $P(d_i)$ شانس و یا احتمال رسیدن به حرکت تعیین شده سپس سودمندیهایی که توسط شانسی که حرکت ویژه تاس بدست آورده و وزن‌دار شده، اضافه می‌کنیم. اگر ما فرض کنیم که $S(C,d_i)$ مجموعه موقعیتهای تولید شده توسط اعمال حرکات قانونی برای پرتاب تاس $P(d_i)$ در موقعیت C باشد، می‌توان مقدار expectimax از C را با استفاده از فرمول

$$\text{expectimax}(c) = \sum_i P(d_i) \max_{s} S(c,d_i)(\text{utility}(s))$$

محاسبه نمود.

این فرمول، سودمندی مورد انتظار در موقعیت c را با فرض بهترین بازی ارائه می‌دهد. با یک سطح بالا رفتن تا گره‌های ΔMIN در شکل ۲۵، اکنون می‌توانیم فرمول مقدار minimax نرمال را اعمال کنیم زیرا مقادیر سودمندی را به تمام گره‌های شانس منتسب کردیم. سپس به سمت گره شانس B حرکت می‌کنیم جایی که می‌توان مقدار expectimin را با استفاده از فرمولی که قابل مقایسه با expectimax است محاسبه نمود.

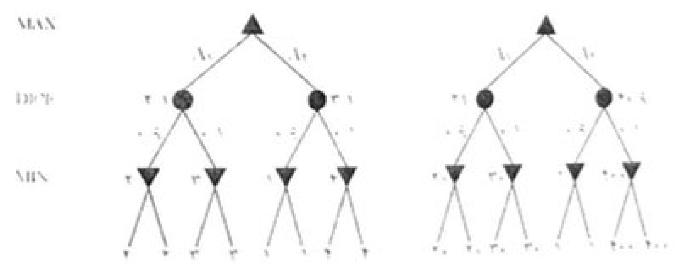
این پردازش به صورت بازگشتی در تمام طول درخت به جز در سطح بالایی جایی که **پرتاب تاس از قبل** شناخته شده است، اعمال می‌شود. برای محاسبه بهترین حرکت، به سادگی **MINMAX-VALUE را** توسط EXPECTIMINMAX-VALUE جایگزین می‌کنیم.

ارزیابی موقعیت در بازیها با گره‌های شانس

expectiminmax با آمدن کنار برای واضح تخمین .minmax مانند **قطع جستجو در چند نقطه و اعمال تابع** ارزیابی بروی برگها است. ممکن است کسی فکر کند که **توابع ارزیابی برای بازیها مانند تخته نرد متفاوت** نیستند. در اصل، از توابع ارزیابی برای شطرنج، باید آنها **امتیازات بالاتری را به وضعیت‌های بهتری** بدهند.

در حقیقت، **حضور گره‌های شانس بدین معناست که باید در مورد آنچه که به معنای مقادیر ارزیابی است.** دقیق بود. به خاطر داشته باشید که برای minmax، **هر انتقال با حفظ مرتبه از مقادیر برگی تأثیری روی** حرکت ندارد. از این رو، می‌توان از **مقادیر ۱ و ۲و ۳و ۴ یا مقادیر ۴۰۰ و ۳۰ و ۲۰ و ۱ استفاده نمود و** تصمیم مشابهی را گرفت. این مطالب آزادی بسیاری را در **طراحی تابع ارزیابی در اختیار ما می‌گذارد:** مادامی که وضعیت‌های با ارزیابیهای بالاتر منجر به برد می‌شوند، **خوب کار می‌کند.**

با گره‌های شانسی، این آزادی را از دست می‌دهیم. **شکل ۲۶ اینکه چه اتفاقی می‌افتد را نشان می‌دهد: با** مقادیر برگی ۱ و ۲ و ۳و ۴، حرکت A_1 بهترین است. **با مقادیر برگی ۴۰۰ و ۳۰ و ۲۰ و ۱، حرکت A_2** بهترین است. از این رو اگر ما تغییری را در **مقیاس مقادیر ارزیابی ایجاد کنیم، برنامه در مجموع به طور** متفاوت رفتار می‌کند. روشن است که **برای اجتناب از این حساسیت تابع ارزیابی فقط می‌تواند یک انتقال** خطی مثبت از احتمال برد در یک موقعیت (یا به طور کلی‌تر، **از سودمندترین مورد انتظار موقعیت) باشد. این** حالتی از خواص کلی و مهم شرایطی است که **درگیر عدم قطعیت است، و ما در مورد آن بعداً بحث می‌کنیم.**



پیچیدگی expectiminimax

اگر برنامه به طور پیشرفته از تمام پرتابهای تاس که در ادامه بازی اتفاق افتاده، مطلع باشد، **حل بازی‌هایی** که با تاس همراه هستند مشابه حل بازیهای بدون تاس است که minmax در آنها زمان **$O(b^m)$ دارد.** بدلیل اینکه expectiminimax تمام دنباله‌های پرتاب تاس را در نظر می‌گیرد، زمانی معادل **$O(b^m n^m)$** می‌برد، که n تعداد پرتابهای محدود است.

حتی اگر عمق درخت به عمق کوچک وثابت d محدود شود، هزینه ویژه‌ای که با minmax مقایسه می‌شود، آن را برای پیش بینی خیلی دور در بازیهای ماند تخته نرد غیر ممکن می‌سازد. زمانی که n برابر ۲۱ و b حدود ۲۰ باشد. ولی در بعضی از مواقع b به ۴۰۰۰ نیز می‌رسد.

راه دیگر در مورد این مسئله به قرار زیر است: مزیت آلفا- بتا، با داشتن بهترین بازی نادیده گرفتن پیشرفتها در آینده است که احتمال وقوعشان کم است. از این رو، بروی حوادث محتمل تمرکز می‌کند در بازیهای به همراه تاس، دنباله‌های محتملی از حرکات وجود ندارد، چون برای آن حرکاتی که باید انجام بگیرند. ابتدا تأس باید به روش درستی پرتاب شود تا آن حرکات منطقی شوند. این یک مسئله کلی است زمانی که عدم قطعیت وجود دارد، احتمالات به طور وسیع چند برابر می‌شوند و صورت دادن برنامه‌های جزیی از عملیات بی‌مورد است چون دنیای احتمال قادر به ادامه بازی نیست

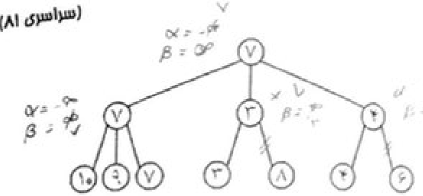
شکی نیست که اکنون بر خواننده مسلم گشته که شاید اعمال هرس آلفا بتا روی درختهای ساری با گره‌های شانسی چاره ساز باشد. واضح است که شدنی است. گره شانس c درشکل ۲۵ را در نظر بگیرید چه اتفاقی برای مقدارش خواهد افتاد زمانی که بچه‌هایش را مورد آزمایش و ارزیابی قرار می‌دهیم. پرسش اینجاست: آیا یافتن حد بالایی برای مقدار c قبل از اینکه به فرزندانش بپردازیم، امکان پذیر است؟ (یادآوری می‌شود که این همان چیزی است که آلفا – بتا نیاز دارد تا گره و زیر درخشش را هرس کند.)

در نظر اول، به نظر غیر ممکن می‌آید، زیرا مقدار c میانگین مقادیر فرزندانش است. و تا وقتی که به تمام پرتابهای تاس نظر می‌اندازیم، این میانگین هر چیزی می‌تواند باشد. زیرا فرزندان آزمایش نشده ممکن است هر مقداری داشته باشند. اما اگر ما حدهایی روی مقادیر ممکن تابع سودمندی قرار دهیم، سپس می‌توانیم به حدهایی برای میانگین برسیم. برای مثال، اگر بگوییم که تمام مقادیر سودمندی بین مقدار گره شانسی بدون توجه به فرزندانش قرار دهیم.

تست‌های فصل سوم

۱- در درخت بازی زیر (i,j) نمایش ارزش عنصر j ام از چپ به راست در سطح i ام است. مثلاً (۲و۳) نمایش عنصر دوم از سطح سوم بـا ارزش ۹ مـی‌باشـد. چـه عناصـری شـامل α – B pruning می‌شوند؟

(سراسری ۸۱)



۱) (۳ و ۳)

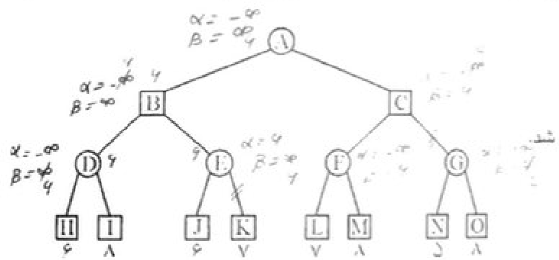
۲) (۳و۵) و (۳و۷)

۳) (۳و۳) و (۲و۲)

۴) (۳و۶) و (۳و۴) و (۲و۳) و (۱و۳)

۲- درخت مقابل در اثر جستجوی minmax ایجاد شده است. گره‌های دایره،گره min و گره‌های مربع، گره max هستند. اعداد زیر گره‌های برگ ارزش آن‌ها را نشان می‌دهد. در صورت استفاده از روش هرس آلفا و بتا کدام یک از گره‌های این درخت جستجو نخواهد شد؟

(سراسری ۸۲)



۱) گره های M,k

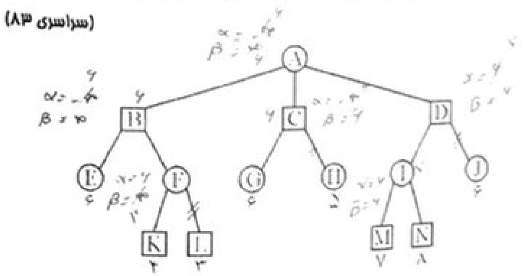
۲) گره‌های O,M,K

۳) گره‌های M,O,G,K

۴) کلیه گره‌ها جستجو خواهد شد.

۳- اگر با استفاده از روش جستجوی minmax درخت جستجوی مقابل پیمایش شود، با استفاده از روش هرس آلفا- بتا کدام یک از گره‌های این درخت ملاقات نخواهد شد؟ (دوایر معرف گره‌ی min و مربع‌ها معرف گره‌ی max و اعداد کنارگره‌های برگ، معرف ارزش این گره‌ها است)

(سراسری ۸۳)



۱) {L, H, N}

۲) {L, N, J}

۳) {L, H, J}

۴) {K, N, J}

۴- اگر در روش هرس آلفا – بتا که بطور عادی از روش جستجوی عمق اول بهره می‌گیـرد، از روش جستجوی سطح اول استفاده نمائیم، کدامیک از گزینه‌های زیر صحیح خواهد بود؟

(فن‌آوری اطلاعات ۸۳)

۱) جستجو پاسخ بهینه مسئله را می‌یابد.

۲) تغییری در عملکرد روش صورت نمی‌گیرد.

۳) کارائی جستجو در حد روش minmax کاهش می‌یابد.

۴) کارائی روش افزایش می‌یابد، اما حافظ بیشتری نیز مصرف می‌گردد.

۵- بیشترین تأثیر افزودن عنصر شانس (مثل ریختن تاس) در بازی‌ها، بر روی درخـت جـستجوی تولید شده چیست؟

(سراسری ۸۴)

۱) هرس کردن شاخه‌ها مشکل تر می‌شود

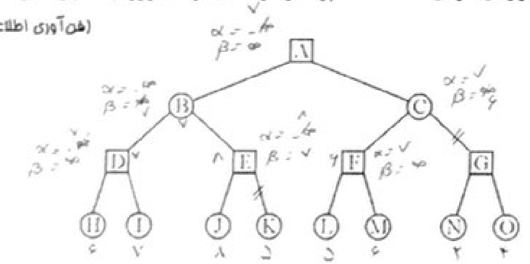
۲) روش‌های مثل minmax و آلفا – بتا نمی‌توانند با عنصر شانس کار کنند.

۳) در محاسبه تابع ارزیابی باید لبه‌های مرزی که بازتاب عنصر شانس هستند را اعمال نمود.

۴) برای هر حرکت بازیکن، سطح دیگری از گره‌ها تولید می‌شوند که احتمالات معرفی شـده بوسیله عنصر شانس را در بر می‌گیرند.

۶- در صورت استفاده از روش هرس α – B کدامیک از گره‌های درخت زیر جستجو نخواهد شد؟

(فن‌آوری اطلاعات ۸۴)



۱) K,M

۲) K,M,O

۳) K,G,N,O

۴) E,J,K,G,N,O

۷- در فضای بازی‌ها مقدار تابع ارزیابی (evaluation function) نشانگر چیست؟

(فن‌آوری اطلاعات ۸۴)

۱) تعیین کننده خاتمه بازی

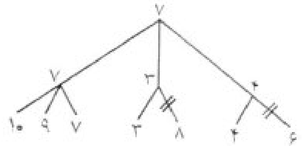
۲) مقدار عددی خروجی بازی در گره‌های پایانی

۳) درجه‌ی از عدم قطعیت که به دلیل حضور حریف و یا عنصر شانس ایجاد می‌شود.

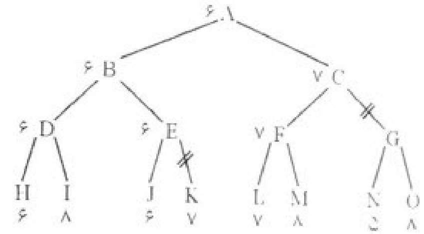
۴) تخمینی که از میزان موفقیت (مناسب بودن) مورد انتظار گروه‌های یک پیکربندی خاص از بازی

پاسخ تست‌های فصل سوم

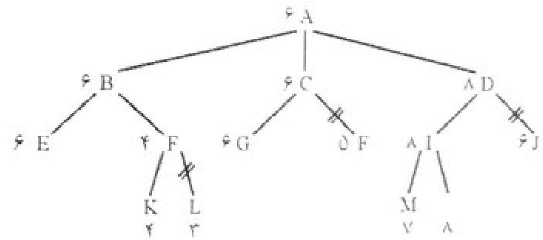
۱- گزینه «۲» صحیح است.



۲- گزینه «۳» صحیح است.



۳- گزینه «۳» صحیح است.



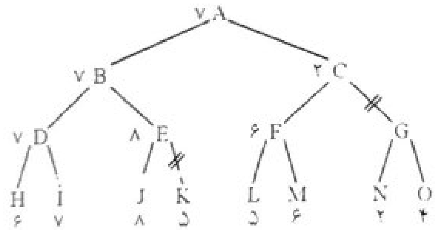
۴- گزینه «۳» صحیح است.

در صورت استفاده از استراتژی BFS، امکان یافتن مقادیر موقت آلفا و بتا وجود **نخواهد داشت** چرا که حرکت بر روی درخت بازی، بجای عمق در سطح خواهد بود. **به همین علت مقادیر آلفا و بتا** نهائی خواهد بود. مسلم است تحت این شرایط مسئله قطع (Out off) مطرح **نبوده و روش به همان** MinMax تقلیل می‌یابد.

۵- گزینه «۴» صحیح است.

بدیهی است، به متن درس مراجعه کنید.

۶- گزینه «۳» صحیح است.



۷- گزینه «۴» صحیح است.

منظور از تابع ارزیابی همان تابع امتیاز (سودمندی است). این تابع شانس موفقیت یا عدم موفقیت در یک وضعیت را معین کند.

فصل چهارم

مبانی منطق ریاضی

۴-۱- مقدمه

چگونه می‌توان در زندگی روزمره استدلال کرد؟ ممکن است ما برخی واقعیات را بیان کرده و بر مبنای آنها یک نتیجه جدید را مطرح کنیم. برای مثال، واژه‌هائی همانند بنابراین، پس، در نتیجه و غیره مبین نتیجه‌گیری در تشریح استدلال انسان‌ها هستند. زمانیکه نتیجه‌ای از یک نوع بیان می‌شود، در واقع از یک قانون منطقی که به آن قانون استنتاج می‌گویم استفاده شده است. مشهورترین قانون استنتاجی modes ponens نام دارد و طرز کار آن بدین ترتیب است. فرض کنید A و B دو جمله هستند و فرض می‌کنیم A و اگر A آنگاه B درست هستند. می‌توان نتیجه گرفت که B نیز درست است. برای نمونه به مثال زیر بر مبنای این قانون استنتاجی توجه کنید:

اگر هوا بارانی است، ابرها در آسمان هستند

هوا بارانی است

بنابراین ، ابرها در آسمان هستند.

انسان‌ها بدون درکی از قانون استنتاجی modes ponens آنرا بکار می‌برند. قانون دیگری از این دسته که شاید باز هم بدون درک آن مورد استفاده قرار می‌گیرد modes tollens نام دارد. فرض کنید A و B دو جمله هستند. اگر جمله "اگر A آنگاه B" درست و جمله B اشتباه باشد، آنگاه می‌توان نتیجه گرفت که جمله A نیز نادرست است.

ریاضیات (calculus) چیست؟ ریاضیات زبان بیان برخی انواع عبارات است که از طریق قوانین متناهی برای شکل دادن به عبارت روشن می‌گردد. مقادیر یا معانی با عبارت در ارتباط قرار دارند و قوانین متناهی برای تبدیل یک عبارت به دیگری به همراه مقادیری وجود دارد.

برای مثال زبان فارسی چیزی شبیه ریاضیات است که عبارات آن جملات فارسی هستند که از طرز طریق دستور زبان مشخص می‌شوند. بدیهی است جملات فارسی معنی خاص خود را دارند، اما قوانینی برای تبدیل جملات وجود ندارد و از این رو ریاضیات ما برای زبان فارسی کامل نیست.

معمولاً در ریاضیات محاسبه بر روی اعداد حقیقی صورت می‌گیرد. اما هدف از این بخش منطق ریاضی است. در منطق ریاضی، عبارات بوسیله قوانینی بیان می‌شوند. مقادیر با ارزش‌های درست و یا نادرست مشخص می‌شوند و قوانین معینی برای تبدیل عبارات وجود دارند.

۴-۲- منطق گزاره‌ها

در منطق گزاره‌ها، هر جمله (یا گزاره) دارای ارزش درست یا نادرست است. معمولاً برای بیان گزاره‌ها از حروف انگلیسی همانند Q،P یا R استفاده می‌شود. دستور زبان در منطق گزاره‌ها بصورت زیر است

$sentence \rightarrow atomic - sentence \mid complex - sentence$

$atomic - sentences \rightarrow True \mid false \mid P \mid Q \mid R \mid ...$

$complex - sentence \rightarrow (sentence) \mid sentence\ connective\ sentence \mid \sim sentence$

$connective \rightarrow \wedge \mid \vee \mid \Leftrightarrow \mid \Rightarrow$

عبارتی که از دستور زبان فوق تبعیت کند، فرمول خوش (ترکیب) (well - formed formula) یا به اختصار wff نامیده می‌شود.

اولویت بین متصل کننده‌ها (connective) اهمیت خاص دارد که طبق جدول زیر می‌باشد

عملگر	اولویت
$\sim$	1
$\wedge$	2
$\vee$	3
$\Rightarrow$	4
$\Leftrightarrow$	5

همچنین نباید فراموش کرد که به جز نقیض (~) تمام عملگرها شرکت پذیر از چپ هستند. برای نمونه به پرانتز گذاری کامل عبارت زیر توجه کنید:

$\sim P \Rightarrow P \vee Q V R \qquad [(\sim P) \Rightarrow ((P \wedge Q) V R)]$

یک wff گزاره همیشه درست (tautology) نامیده می‌شود اگر برای تمامی مقادیر جدول ارزش گذاری مقدار آن درست باشد. برای نمونه، گزاره‌های $P \rightarrow P$ و $P \vee \sim P$ گزاره‌های همیشه درست هستند اما اگر تمامی مقادیر جدول ارزش نادرست باشد گزاره همیشه نادرست (contradiction) نامیده می‌شود که برای نمونه می‌توان به $P \wedge \sim P$ اشاره کرد. اگر wff دارای جدول ارزشی با ارزش‌های نادرست باشد، یک احتمال (contingency) نام دارد.

دو گزاره هم ارز (equivalence) نامیده می‌شوند (همانند A و B) اگر جدول ارزش دقیقاً همانند یکدیگر داشته باشند که معمولاً بصورت $A \equiv B$ نمایش داده می‌شود

از جمله مشهورترین هم ارزی‌ها، در منطق گزاره‌ها می‌توان به موارد زیر اشاره کرد:

$\sim \sim A \equiv A$

$A \vee true \equiv A \qquad A \wedge true \equiv A \qquad A \rightarrow true \equiv true$

فصل چهارم

مبانی منطق ریاضی

۱-۴- مقدمه

چگونه می‌توان در زندگی روزمره استدلال کرد؟ ممکن است ما برخی واقعیات را بیان کرده و بر مبنای آنها یک نتیجه جدید را مطرح کنیم. برای مثال، واژه‌هائی همانند بنابراین، پس، در نتیجه و غیره مبین نتیجه‌گیری در تشریح استدالال انسان‌ها هستند. زمانیکه نتیجه‌ای از یک نوع بیان می‌شود، در واقع از یک قانون منطقی که به آن قانون استنتاج می‌گویم استفاده شده است. مشهورترین قانون استنتاجی modes ponens نام دارد و طرز کار آن بدین ترتیب است. فرض کنید A و B دو جمله هستند و فرض می‌کنیم A و اگر A آنگاه B درست هستند. می‌توان نتیجه گرفت که B نیز درست است. برای نمونه به مثال زیر بر مبنای این قانون استنتاجی توجه کنید:

اگر هوا بارانی است، ابرها در آسمان هستند

هوا بارانی است.

بنابراین ، ابرها در آسمان هستند.

انسان‌ها بدون درکی از قانون استنتاجی modes ponens آنرا بکار می‌برند. قانون دیگری از این دست که شاید باز هم بدون درک آن مورد استفاده قرار می‌گیرد modes tollens نام دارد. فرض کنید A و B دو جمله هستند، اگر جمله " اگر A آنگاه B " درست و جمله B اشتباه باشد، آنگاه می‌توان نتیجه گرفت که جمله A نیز نادرست است.

ریاضیات (calculus) چیست؟ ریاضیات زبان بیان برخی انواع عبارات است که از طریق قوانین متناهی برای شکل دادن به عبارت روشن می‌گردد. مقادیر یا معانی با عبارت در ارتباط قرار دارند و قوانین متناهی برای تبدیل یک عبارت به دیگری به همراه مقادیری وجود دارد.

برای مثال زبان فارسی چیزی شبیه ریاضیات است که عبارات آن جملات فارسی هستند که از طرز طریق دستور زبان مشخص می‌شوند. بدیهی است جملات فارسی معنی خاص خود را دارند، اما قوانینی برای تبدیل جملات وجود ندارد و از این رو ریاضیات ما برای زبان فارسی کامل نیست.

معمولاً در ریاضیات محاسبه بر روی اعداد حقیقی صورت می‌گیرد، اما هدف از این بخش منطق ریاضی است. در منطق ریاضی، عبارات بوسیله قوانینی بیان می‌شوند، مقادیر با ارزش‌های درست و یا نادرست مشخص می‌شوند و قوانین معینی برای تبدیل عبارات وجود دارند.

۲-۴- منطق گزاره‌ها

در منطق گزاره‌ها، هر جمله (یا گزاره) دارای ارزش درست یا نادرست است. معمولاً برای بیان گزاره‌ها از حروف انگلیسی همانند Q،P یا R استفاده می‌شود. دستور زبان در منطق گزاره‌ها بصورت زیراست:

$$\text{sentence} \rightarrow \text{atomic-sentence} \mid \text{complex-sentence}$$

$$\text{atomic-sentences} \rightarrow \text{True} \mid \text{false} \mid P \mid Q \mid R \mid \ldots$$

$$\text{complex-sentence} \rightarrow (\text{sentence}) \mid \text{sentence connective sentence} \mid \sim \text{sentence}$$

$$\text{connective} \rightarrow \wedge \mid \vee \mid \Leftrightarrow \mid \Rightarrow$$

عبارتی که از دستور زبان فوق تبعیت کند، فرمول خوش (ترکیب) (well - formed formula) یا به اختصار wff نامیده می‌شود.

اولویت بین متصل کننده‌ها (connective) اهمیت خاص دارد که طبق جدول زیر می‌باشد.

عملگر	اولویت
$\sim$	1
$\wedge$	2
$\vee$	3
$\Rightarrow$	4
$\Leftrightarrow$	5

همچنین نباید فراموش کرد که به جز نقیض ($\sim$) تمام عملگرها شرکت پذیر از چپ هستند. برای نمونه به پرانتز گذاری کامل عبارت زیر توجه کنید:

$$\sim P \Rightarrow P \vee QVR \qquad [(\sim P) \Rightarrow ((P \wedge Q)VR)]$$

یک wff گزاره همیشه درست (tautology) نامیده می‌شود اگر برای تمامی مقادیر جدول ارزش گذاری مقدار آن درست باشد. برای نمونه، گزاره‌های $P \rightarrow P$ و $P \vee \sim P$ گزاره‌های همیشه درست هستند. اما اگر تمامی مقادیر جدول ارزش نادرست باشد گزاره همیشه نادرست (contradiction) نامیده می‌شود که برای نمونه می‌توان به $P \wedge \sim P$ اشاره کرد. اگر wff دارای جدول ارزشی با ارزش‌های نادرست باشد. یک احتمال (contingency) نام دارد.

دو گزاره هم ارز (equivalence) نامیده می‌شوند (همانند A و B) اگر جدول ارزش دقیقاً همانند یکدیگر داشته باشند که معمولاً بصورت $A \equiv B$ نمایش داده می‌شود.

از جمله مشهورترین هم ارزی‌ها، در منطق گزاره‌ها می‌توان به موارد زیر اشاره کرد:

$$\sim\sim A \equiv A$$

$$A \vee \text{true} \equiv A \qquad A \wedge \text{true} \equiv A \qquad A \rightarrow \text{true} \equiv \text{true}$$

$A \vee false \equiv A$ $\quad A \wedge false \equiv false$ $\quad A \rightarrow false \equiv \sim A$

$A \vee A \equiv A$ $\quad A \wedge A \equiv A$ $\quad true \Rightarrow A \equiv A$

$A \vee A \sim\equiv true$ $\quad A \wedge \sim A \equiv false$ $\quad false \Rightarrow A \equiv true$

$A \Rightarrow A \equiv true$

$A \Rightarrow B \equiv \sim A \vee B$

$\sim(A \Rightarrow B) \equiv A \wedge \sim B$

$A \Rightarrow B \equiv A \wedge \sim B \Rightarrow false$

$A \wedge (A \vee B) \equiv A$ $\quad A \vee (A \wedge B) \equiv A$

$A \wedge (\sim A \vee B) \equiv A \wedge B$ $\quad A \vee (\sim A \wedge B) \equiv A \vee B$

$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$

$A \vee (B \wedge C) \equiv (A \wedge B) \wedge (A \vee C)$

$\sim(A \wedge B) \equiv \sim A \vee \sim B$

$\sim(A \vee B) \equiv \sim A \wedge \sim B$

فرم نرمال فصلی

اگر لیترال (literal) را یک جمله اتمی یا نفی آن در منطق گزاره‌ها تعریف کنیم، **عطف پایه لیترال و یا عطف** دو تا چند لیترال نامیده می‌شود. برای مثال P یا $P \wedge \sim Q$ عطف‌های پایه هستند. **فرم نرمال فصلی** (DNF) یا یک عطف پایه است و یا از دو یا چند عطف پایه تشکیل شده است. برای مثال، **نمونه‌های زیر** DNF هستند.

$P \vee (\sim P \wedge Q)$

$(P \wedge Q) \vee (\sim P \wedge R)$

قضیه: هر فرمول خوش ترتیب (wff) معادل DNF دارد.

یکی از راه‌های تولید DNF استفاده از روابط هم ارزی اشاره شده دربخش‌های قبلی و **تبدیل wff اولیه به** DNF معادل آن است. معمولاً **سه فرایند اصلی در راه ساخت DNF از wff اولیه صورت می‌گیرد:**

۱- حذف اگر - آنگاه ($\Rightarrow$)

۲- کاهش دامنه نفی (با استفاده از قوانین دمورگان)

۳- استفاده از قوانین توزیع‌پذیری

فرم نرمال عطفی

مشابه با DNF می‌توان به تعریف CNF پرداخت. اگرفصل پایه مشابه عطف پایه تعریف شود، **CNF را** می‌توان یک عطف پایه و یا فصل دو یا چند عطف پایه تعریف نمود. برای مثال:

$(P \vee Q) \wedge (\sim Q \vee R)$

$(P \vee Q \vee R) \wedge (\sim P \vee Q \vee R)$

قضیه: هر فرمول خوش ترتیب (wff) معادل CNF دارد.

برای ساخت CNF از wff اولیه، مراحلی مشابه ساخت DNF طی می‌شود.

مثال: wff زیر را به CNF معادل آن تبدیل کنید:

$(A \wedge B) \vee (C \wedge D) \vee (E \Rightarrow F)$

$(A \vee C \vee \sim E \vee F) \wedge (B \vee C \sim E \vee F) \wedge (A \vee D \vee \sim E \vee F) \wedge (B \vee D \vee \sim E \vee F)$

قوانین استنتاجی در منطق گزاره‌ها

برای استنتاج در یک سیستم استدلالی، **به قوانین استنتاجی نیاز است تا بر پایه آن بتوان نتیجه جدیدی را** تولید نمود. قبلاً با mods ponens و modes tallens آشنا شدیم. برای بیان قوانین استنتاجی از نگارش خاصی استفاده می‌شود که در زیر این نگارش برای modes ponens نشان داده شده است.

$\dfrac{A \Rightarrow B, A}{B}$ modes ponens

قوانین مشهور استنتاجی در منطق گزاره‌ها به قرار زیر است.

modes tollens $\dfrac{A \Rightarrow B, B}{\sim A}$

conjunction $\dfrac{A, B}{A \wedge B}$

simplification $\dfrac{A \wedge B}{A}$

addition $\dfrac{A}{A \vee B}$

Disjunctive syllogism $\dfrac{A \vee B, \sim A}{B}$

Hypothetical syllogism $\dfrac{A \Rightarrow B, B \Rightarrow C}{A \Rightarrow C}$

Constructive Diallema $\dfrac{A \vee B, A \Rightarrow C, B \Rightarrow D}{C \vee D}$

Destructive Diallema $\dfrac{\sim C \vee \sim D, A \Rightarrow C, B \Rightarrow D}{\sim A \vee \sim B}$

اصل موضوع (Axiom) به wff **ای گفته می‌شود که می‌خواهیم با استفاده از آن بعنوان پایه** استدلال خود را مطرح نمائیم. معمولاً **اصل موضوع بعنوان نقطه شروع اثبات در نظر گرفته شده** و به همین علت با دانش اولیه قابل اثبات بوده است.

سیستم استدلال رسمی به سه چیز نیاز دارد. مجموعه wff، مجموعه اصل موضوعات و مجموعه **قوانین** استنتاجی اثبات (proof) دنباله متناهی از wff هاست چنانچه هر wff در دنباله **یا یک اصل موضوع باشد و** یا اینکه از یک wff قبلی در دنباله بکمک قوانین استنتاجی تولید شده باشد. آخرین wff **دنباله، قضیه** (theorem) نام دارد.

از طریق منطق گزاره‌ها می‌توان دنیای پیرامون خود را مدل کنیم و برخی استنتاج‌ها را **نیز بر روی آن** انجام داد، اما این منطق با مشکلات مهمی روبروست. مهمترین مشکل آنست که **برای بیان دنیای پیرامون** نیازمند به تعداد بسیار زیادی قانون هستیم. برای مثال فرض کنید دریک صفحه **چهار خانه، عاملی بر** مبنای منطق گزاره‌ها می‌خواهد خانه‌های مجاور خود را از جهت وجود مانع کنترل کند. **در این حالت اگر** حرکات عامل را بصورت چهار جهت اصلی در نظر گیریم، درک این موضوع نیازمند **به نگارش ۴ قانون** مجزا دارد (برای هر خانه مجاور یک قانون) بدیهی است برای یک صفحه بزرگ **تعداد این قوانین به هزاران** خواهد رسید. وجود این تعداد قانون اولاً کار تهیه آنها را بسیار دشوار کند و **ثانیاً موتور استنتاجی** خودکار برای ارائه یک اثبات بر مبنای این تعداد قانون با مشکل **کندی سرعت روبرو خواهد شد.**

از سوی دیگر، روبرو شدن با تغییرات در محیط نیز دشواری به بار خواهد آورد. **اگر شرایط محیطی در** حال تغییر باشد، با حرکت عامل از مکان نخستین و طی زمان، ممکن است **قوانین موجود ارزش خود را از** دست دهند. بعبارت دیگر شاید لازم باشد عامل حین حرکت از حافظه‌ای برخوردار باشد **تا بیاد آورد در** گذشته چه اتفاقاتی روی داده است. این مشکل در منطق مرتبه اول نیز به نوعی **مطرح خواهد شد و در** آنجا به بحث جدی‌تری در مورد آن خواهیم پرداخت.

۳-۴- منطق مرتبه اول

تاکنون با منطق گزاره‌ها آشنا شدیم و دریافتیم در این منطق چگونه مدل‌سازی **دنیا صورت گرفته و** استنتاج چگونه برای تولید استدلال ایجاد می‌شود. منطق گزاره‌ای محدوده بسیار **کوچکی از دنیاهای واقعی** را می‌تواند مدل کند و علت این امر عدم امکان تعریف سور (quantifier) در آن است.

منطق مرتبه اول، دنیا را بصورت اشیائی که در روابطی قرار دارد نگاه می‌کند. روابط **مختلفی می‌تواند بین** اشیاء این منطق برقرار گردد که از جمله آن می‌توان به توابع اشاره کرد. عناصر **اصلی جملات در این** منطق از مولفه‌های زیر تشکیل شده است.

نمادهای ثابت: هر نماد ثابت به یک جسم خاص اشاره می‌کند. تمام اشیاء نیازی به داشتن **نام ندارند و** برخی می‌تواند چندین نام داشته باشند.

نمادهای گزاره: نماد گزاره ارتباط بین اشیاء را بیان می‌کند، که همانند فعل در جملات **زبان‌های طبیعی** است.

نمادهای تابع: برخی از روابط در قالب تابع معین می‌گردند. بدین معنی که هر شیئی **دقیقاً به شیئی دیگری** توسط رابطه رجوع می‌کند (همانند اینکه هر فردی فقط یک پدر دارد)

دستور زبان در منطق رتبه اول بصورت زیر است:

sentence → atomic − sentence |
    sentence connective setenee |
    Quantifier variable,...sentence |
    ~ sentence |
    (sentence)

atomic − sentence → predicate(term,...) | term = term

term → function(term,...) | constant | variable

connective → ⇒ | ∧ | ∨ | ⇔

quantifire → ∀ | ∃

constant → A | B | C | ...

variable → x | y | z | ...

predicate → before | has color | ...

Function → Mother | Lef − log− of | ...

ترم (term) عبارتی منطقی است که به یک شیئی رجوع می‌کند. همانطور که در دستور زبان نیز مشخص شد، این مقدار می‌تواند ثابت، متغیر و یا یک تابع باشد.

چند مثال:

brother(richard. john)

این جلد مقدار ثابت جان ریچارد را در رابطه برادر بودن با هم قرار می‌دهد. فراموش نکنید برای سیستم منطقی هیچکدام از این واژه ها (برخلاف متفکر انسانی) معنی ندارد.

marrid (father − of (john), mother − of (richard))

این جمله بیان می‌کند که پدر ریچارد با مادر جان ازدواج کرده است. نماد گزاره همان married است و father-of و mother-of توابع مورد استفاده می‌باشند.

$older(john,۳۰) \vee younger(john,۳۰)$

جان ممکن است بیشتر و یا کمتر از ۳۰ سال سن داشته باشد.

$\exists x Sister(x, sport) \wedge cat(x)$

اسپات خواهری دارد که گربه است. دقت کنید در این مثال از سور وجودی (existential) استفاده شده است.

$\forall x\ cat(x) \wedge mamnal(x)$

تمامی گربه‌ها پستاندار هستند. در این جمله نیز از سور عمومی (universal) استفاده شده است.

جدول اولویت عملگرها در منطق مرتبه اول به قرار زیر است:

عملگر	اولویت
$\sim, \exists x, \forall x$	۱
$\wedge$	۲
$\vee$	۳
$\Rightarrow$	۴
$\Leftrightarrow$	۵

اگر سور یا نفی در کنار هم آمده باشد، سمت راست‌ترین نماد با کوچکترین wff از سمت راست هم‌گروه خواهد شد. به چند مثال زیر توجه کنید.

فرم پرانتزی کامل	فرم فاقد پرانتز
$\forall x(\sim(\exists y(\forall z\, p(x,y,z)))$	$\forall x \sim \exists y \forall z\, p(x,y,z)$
$(\exists x p(x)) \vee q(x)$	$\exists x p(x) \vee q(x)$
$(\forall x p(x)) \Rightarrow q(x)$	$\forall x p(x) \Rightarrow q(x)$
$(\exists x(\sim p(x,y))) \Rightarrow (q(x) \wedge r(y))$	$\exists x \sim p(x,y) \Rightarrow q(x) \wedge r(y)$
$(\exists x(p(x)) \Rightarrow ((\forall x q(x)) \vee (p(x) \wedge r(x)))$	$\exists x p(x) \Rightarrow \forall x q(x) \vee p(x) \wedge r(x)$

حوزه (scope) یک سور، بخشی از یک wff است که در آن سور مربوط اثر دارد. برای سور عمومی و وجودی، بسته به پرانتز گذاری، حوزه سور قابل تعیین است.

وقوع متغیری همانند x در یک wff محدود شده (bound) نامیده می‌شود اگر در حوزه سور عمومی و یا وجودی همانند $\exists x$ یا $\forall x$ قرار داشته باشد. در غیر اینصورت x متغیر آزاد (free) نام دارد. برای مثال $\exists x\, p(x,y) \Rightarrow q(x)$

متغیر x در نماد گزاره $p(p(x,y))$ محدود شده ولی در $q(x)$ آزاد است. (به پرانتز گذاری این عبارت دقت کنید)

در برخی موارد ممکن است حوزه سورهای هم نام (از نظر نام متغیر) با هم تداخل پیدا می‌کنند. به مثال زیر توجه کنید.

$\forall x[cat(x) \vee (\exists x\, brother(richard, x))]$

در اینجا x در گزاره brother هم در حوزه $\exists x$ و هم در حوزه $\forall x$ قرار دارد حال آنکه این شکل برای $Cat(x)$ وجود ندارد.

در چنین مواردی قانون آنست که متغیر به داخلی‌ترین سور که به حوزه آن تعلق دارد، بر می‌گردد. در این مثال x موجودی در گزاره brother به سور $\exists x$ باز می‌گردد. به این حالت سورهای لانه‌ای (nested) می‌گویند.

تفسیری برای یک wff همانند w مدل نامیده می‌شود اگر w نسبت به تفسیر درست باشد. در غیر اینصورت تفسیر ضد مدل (counter model) نام دارد. به زبان دیگر مدل، دنیائی است که در آن جمله‌ای تحت تفسیر خاصی، درست باشد.

یک wff معتبر (valid) نامیده می‌شود، اگر برای تمام تفاسیر ممکن درست باشد. در غیر اینصورت، wff نامعتبر (invalid) نام دارد. Wff ارضاء نشدنی (unsatisfiable) نام دارد اگر تحت تمامی تفاسیر ممکن نادرست باشد. یعنی تمامی تفاسیر آن ضد مدل هستند. در غیر اینصورت ارضا شدنی نام دارد. پس هر wff ممکن است یکی از حالات زیر را داشته باشد.

۱- معتبر و ارضاشدنی (گزاره همیشه درست)

۲- ارضا شدنی و نامعتبر (گزاره احتمالی)

۳- ارضا نشدنی و غیر معتبر (گزاره همیشه نادرست)

قبلاً به این مفهوم در منطق گزاره‌ها نیز برخورد کرده بودیم.

چند شرط مشهور معتبر به قرار زیر است.

$\forall x A(x) \Rightarrow \exists x A(x)$

$\exists x(A(x) \wedge B(x)) \Rightarrow \exists x A(x) \wedge \exists x B(x)$

$\forall x A(x) \vee \forall x B(x) \Rightarrow \forall x(A(x) \vee B(x))$

$\forall x(A(x) \Rightarrow B(x)) \Rightarrow (\forall x A(x) \Rightarrow \forall x B(x))$

$\exists x \forall x P(x,y) \Rightarrow \forall x \exists y P(x,y)$

بستارها

به دوتبدیل جالب زیر دقت کنید

- $P(x) \wedge \sim P(y)$ ارضا شدنی است ولی $\forall x \forall y(p(x) \wedge p(y))$ ارضا شدنی نیست.
- $p(x) \Rightarrow p(y)$ نامعتبر است ولی $\exists x \exists y(p(x) \Rightarrow p(y))$ معتبر است.

نکته قابل تأمل در این دو مثال آنست که اعتبار زمانی حفظ می‌شود که بر روی متغیرهای آزاد سور وجودی زده شود.

فرض کنید w یک wff با متغیرهای آزاد $x_1, x_2, \ldots, x_n$ باشد. بستار عمومی w یک wff بصورت زیر است.

$\forall x_1, \ldots, \forall x_n W$

به همین ترتیب بستار وجودی به صورت زیر است.

$\exists x_1 \ldots \exists x_n W$

اما خواص بستار به قرار زیر است.

۱- wff معتبر است اگر تنها اگر بستار عمومی آن معتبر باشد.

۲- wff ارضا نشدنی است، اگر و تنها اگر بستار وجودی آن ارضا شدنی نباشد.

هم ارزی را قبلاً در منطق گزاره‌ها تعریف کردیم. همان تعریف در منطق مرتبه اول نیز وجود دارد. در مورد سورها روابط هم‌ارزی زیر حاکم است.

$\sim(\forall x\, w) \equiv \exists x \sim w$

$\sim(\exists x\, w) \equiv \forall x \sim w$

$$\forall x \forall y W \equiv \forall y \forall x W$$
$$\exists x \exists y W \equiv \exists y \exists x W$$
$$\exists x(p(x) \Rightarrow q(x)) \equiv \forall x p(x) \Rightarrow \exists x q(x)$$
$$\exists x(p(x) \vee q(x)) \equiv \exists x p(x) \vee \exists x q(x)$$
$$\forall x(p(x) \wedge q(x)) \equiv \forall x p(x) \wedge \forall x q(x)$$

هم‌ارزی‌های محدود شده زیر با شرط آنکه x بصورت متغیر آزاد در wff بنام C ظاهر نشده باشد، **برقرار** هستند.

$$\forall x C \equiv C$$
$$\forall x(C \vee A(x)) \equiv C \vee \forall x A(x)$$
$$\exists x(C \vee A(x)) \equiv C \vee \exists x A(x)$$
$$\forall x(C \wedge A(x)) \equiv C \wedge \forall x A(x)$$
$$\exists x(C \wedge A(x)) \equiv C \wedge \exists x A(x)$$
$$\forall x(C \Rightarrow A(x)) \equiv C \Rightarrow \forall x A(x)$$
$$\exists x(C \Rightarrow A(x)) \equiv C \Rightarrow \exists x A(x)$$
$$\forall x(A(x) \Rightarrow C)) \equiv \exists x A(x) \Rightarrow C$$
$$\exists x(A(x) \Rightarrow C)) \equiv \forall x A(x) \Rightarrow C$$

بیان جملات زبان‌های طبیعی در منطق مرتبه اول

برای تبدیل جملات زبان طبیعی (همانند فارسی) به منطق مرتبه اول، روش مکانیزه و مدرنی **که رسمی** باشد وجود ندارد. برای فهم بهتر این فرایند ابتدا مثالی می‌زنیم.فرض کنید $p(x)$ معرف **سیاستمدار بودن** x و $q(x)$ معرف نادرست بودن x باشد حال به جملات فارسی زیر و معادل آنها در **منطق مرتبه اول** توجه کنید.

برخی سیاستمداران نادرست هستند.	$\exists x(p(x) \wedge q(x))$
هیچ سیاستمداری نادرست نیست.	$\forall x(p(x) \Rightarrow \sim q(x))$
تمامی سیاستمداران نادرست هستند.	$\forall x(p(x) \Rightarrow q(x))$
تمام سیاستمداران نادرست نیستند.	$\exists x(p(x) \wedge \sim q(x))$

از مثال‌های فوق می‌توان دو قانون زیر را حدس زد.

۱- سور عمومی $(\forall x)$ برشط اگر - آنگاه اعمال می‌شود.

۲- سور وجودی $(\exists x)$ بر عطف $(\wedge)$ اعمال می‌شود.

شاید شما اعتراض کنید و مدعی شوید برای مثال در جمله برخی سیاستمدارن نادرست **هستند را می‌توان** بصورت $\exists x(p(x) \wedge q(x))$ نیز نوشته شود. این شکل اشتباه است چون در این حالت **اگر هیچ** سیاستمداری نادرست نباشد نیز wff مطرح شده درست خواهد بود ولی در شکل مطرح شده در **مثال دوم**

دربالا (هیچ سیاستمداری نادرست نیست) ممکن است جمله زیر پیشنهاد شود $\sim \exists x(p(x) \wedge q(x))$ می‌توان ثابت نمود که این دو جمله هم ارز هستند.

$$\begin{aligned} \sim \exists x(p(x) \wedge q(x) &\equiv \forall x \sim (p(x) \wedge q(x)) \\ &\equiv \forall x(p(x) \vee \sim q(x)) \\ &\equiv \forall x(p(x) \Rightarrow q(x)) \end{aligned}$$

منطق‌های مرتبه بالاتر

در منطق مرتبه اول، **تنها می‌توان بر روی متغیرها سور زد و آرگومانهای گزاره تنها** می‌تواند ترم‌ها باشند. عدم رعایت **این محدودیت‌ها منجر به تولید منطق مرتبه بالاتر خواهد شد.**

منطق مرتبه بالاتر نام دارد **اگر بتوان بر روی مجموعه‌ها سور زد و یا مجموعه‌ها بتوانند** عناصر دیگر مجموعه‌ها شوند.

Wff ای که **حاوی سور بر روی مجموعه یا دارای آرگومانی از یک مجموعه باشد که خود مجموعه باشد**، wff مرتبه بالاتر نام دارد.

مثال:

$$\exists S \; S(x)$$
$$S(x) \wedge T(S)$$

حساب موقعیت

باز نمائی تغییرات در دنیا، **توسط منطق مرتبه اول به دشواری صورت می‌گیرد.** در واقع منطق مرتبه اول برای برخورد **با تغییر در شرایط محیطی توانائی قابل ملاحظه‌ای ندارد.** اما در دنیای واقعی بررسی تغییرات اتفاق افتاده **امری بدیهی است که نیاز به آن کاملاً محسوس است.**

ساده‌ترین را **برخورد با تغییرات، تغییر پایگاه معرفت است.** این سیاست می‌تواند پاسخ‌گوئی آخرین وضعیت باشد **ولی باعث نابودی تمام دانش جمع شده در گذشته خواهد شد که این امر تصمیم‌گیری در مورد آینده را دشواری می‌سازد.**

راه‌حل دیگر بیان دانش در **پایگاه‌های معرفت جداگانه به ازاء تغییرات است.** بعبارت ساده‌تر هر وضعیت توسط **یک پایگاه معرفت مستقل و جداگانه بیان شود، و با تغییرات وضعیت پایگاه معرفت** نیز تغییر یابد و پایگاه جدیدی ایجاد شود. **این روش نیز با دشواری هائی روبرو است** اگر چه بر مبنای این روش می‌توان پاسخ یک پرسش را داد ولی **تصمیم‌گیری در مورد چند موقعیت همزمان بدلیل قرارگیری در پایگاه‌های معرفت متعدد، امکان پذیر نخواهد بود.**

از دیدگاه نظری، بازنمائی **موقعیت‌ها و اعمال چیزی متفاوت از بازنمائی اشیاء واقعی یا روابط** واقعی نیست. پس می‌توان **برای آنها نیز قوانین و اصل موضوع تعریف کرد.**

راهکاری که برای برخورد به تغیرات در منطق مرتبه اول مطرح شده حساب موفقیت نام دارد. در این راه‌حل، دنیا بصورت دنباله‌ای از موقعیت‌ها در نظر گرفته می‌شود که هر کدام یک عکس (snapshot) از وضعیت دنیا است.

هر رابطه یا خاصیت که در طول زمان تغییر می‌کند از طریق یک آرگومان اضافی به گزاره بیان می‌شود. برای مثال گزاره (At(Agent, Location بیان کننده موقعیت عامل در وضعیت مشخصی است که می‌تواند بصورت At(Cat, So) یا $At(Cat, S_1)$ نشان داده شود.

در حساب وضعیت برای بیان تغییرات دنیا از تابع (Result(action, situation استفاده می‌گردد، که این تابع بیان کننده وضعیت حاصل از اعمال یک عمل بر یک وضعیت آغازین است، برای مثال $Result(Formard, S_0) = S_1$ و غیره

اصل موضوعاتی که در این رابطه مطرح می‌شوند یا موثر (effect) هستند (یعنی اصل موضوعی که تغییر را بیان می‌کند) و یا اینکه Frame نامیده می‌شود که فریم ثبات در دنیا را بیان می‌کند. مجموع اصل موضوعات موثر و فریم دنیا را مدل می‌کنند.

مسئله فریم (frame problem) شامل نیاز غیرقابل اجتناب از تعریف تعداد زیادی اصل موضوع فریم است که برای تشریح اعمال غیر کارا و غیر ضروری ساخته می‌شوند.

تست‌های فصل چهارم

۱- کدامیک از گزینه‌های زیر نتیجه منطقی جملات مقابل است؟ **(سراسری - ۸۲)**

$$\exists x\, cat(x) \wedge Owns(Hamid, x)$$
$$\forall x (\exists y\, cat(y) \wedge Owns(x, y)) \Rightarrow Animal-lover(x))$$
$$\forall x \forall y (Animal-lover(x) \wedge Animal(y) \Rightarrow \sim kill(x, y))$$
$$kills(Hamid, Pupu) \vee kills(Behzad, Pupu)$$
$$Fish(Pupu)$$
$$\forall x (Fish(x) \Rightarrow Animal(x))$$
$$\sim kill(Hamid, Behzad)$$

۱) بهزاد قاتل ماهی است.

۲) حمید دوستدار گربه است.

۳) بهزاد قاتل پوپا است یا حمید قاتل بهزاد است.

۴) حمید قاتل پوپا است یا گربه قاتل پوپا است.

۲- فرض کنید مجموعه گزاره $\{q \Rightarrow false, q \Leftrightarrow (p_2 \vee p_3)\}$ درست (true) است. آنگاه کدام یک از عبارت‌های زیر را می‌توان از مجموعه گزاره بالا نتیجه گرفت؟ (نماد false نشانه ارزش نادرست است.) **(سراسری - ۸۳)**

۱) q ۲) $\sim p_2, \sim p_3$

۳) $(p_2 \vee p_3)$ ۴) $\sim q, (p_2 \vee P_3)$

۳- حساب وضعیت‌ها (situation calculus)، برای حل چه مشکلی در منطق ایجاد شده است؟ **(فن‌آوری اطلاعات - ۸۴)**

۱) بازنمائی تغییرات   ۲) بازنمائی توابع و سورها

۳) بازنمائی گزاره‌های متغیر دار   ۴) بازنمائی فضای حالت مسائل جستجو

۴- با این فرض که متغیر x در Q به صورت آزاد (free) ظاهر شده است، مقدار کدام‌یک از عبارات زیر در منطق مسندات (predicate logic) نادرست (false) است؟

۱) $(\exists x (p(x) \rightarrow Q)) \rightarrow (\forall x p(x) \rightarrow Q)$

۲) $(\forall x p(x) \rightarrow Q) \rightarrow (\forall x (p(x) \rightarrow Q))$

۳) $(\exists x p(x) \rightarrow Q) \rightarrow \forall x (p(x) \rightarrow Q))$

۴) $(\exists x p(x) \rightarrow Q) \rightarrow (\exists x (p(x) \rightarrow Q))$

۵- در صورتی که پایگاه دانش مقابل را داشته باشیم و از الگوریتم زنجیره‌سازی به جلو (forward – chaining) استفاده نمائیم چه نتایجی قابل دستیابی می‌باشد؟ **(مکاترونیک ۸۴)**

$\forall x\, shinny(x) \rightarrow nice\ weather(x)$

$\forall x \forall y\, healty(x) \wedge nice\ weather(y) \rightarrow gotoswim(x, y)$

$\forall x\, gotoswim(x, Friday) \rightarrow healty(x)$

shinny(Saturday)

healty(A min)

gotoswim(Ali, Friday)

۱) healty(Ali) ۲) shinny(Friday)

۳) gotoswim(A min, Friday) ۴) nice weather(Friday)

۶- کدام یک از جملات زیر به صورت هرن (Horn) نوشته شده است؟

۱) $P_1 \wedge P_2 \wedge P_3 \wedge Q_1$ ۲) $P_1 \wedge P_2 \wedge P_3 \Rightarrow Q_1$

۳) $P_1 \wedge P_2 \wedge P_3 \Rightarrow Q_1 \vee Q_2$ ۴) $P_1 \vee P_2 \vee P_3 \Rightarrow Q_1 \wedge Q_2$

۷- یک مدل model در منطق چیست؟ **(فن‌آوری اطلاعات - ۸۴)**

۱) مجموعه‌ای از قواعد استنتاجی Sound

۲) دنباله‌ای از اعمال روال‌های استنتاجی برای اثبات یک جمله

۳) جهانی که در آن یک جمله تحت تفسیر خاص معتبر است.

۴) مجموعه جملاتی که از روی آنها می‌توان قابل نتیجه گیری بودن یک جمله خاص را اثبات نمود.

پاسخ تست‌های فصل چهارم

۱- گزینه «۳» صحیح است.

گزینه اول معادل گزاره Kill (Behzad,Fish) می‌باشد. این گزاره با منطق ارائه شده تطبیق ندارد، چون در منطق مرتبه اول نمی‌توان یک گزاره (Fish) را بعنوان ترم نیز در نظر گرفت

گزینه دوم معادل Cat-lover(Hamid) است که اصولاً ارتباطی با منطق فوق ندارد چرا که چنین گزاره‌ای در آن تعریف نشده است.

گزینه سوم صحیح است چرا که معادل $kill(Hamid, Behzad) \vee Kill(Behzad, Pupu)$ خواهد بود. بخش دوم گزاره با جمله آخر منطق متضاد است ولی تحت عبارت فصلی است پس اگر Kill(Behzad, pupu) درست باشد، کل گزاره نیز درست خواهد بود. برای اثبات این جمله می‌توان گفت:

$$\left.\begin{array}{l} Fish(pupu) \\ \forall x(fish(x) \Rightarrow Animal(x)) \end{array}\right\} \Rightarrow \text{۱} - Animal(Pupu)$$

$$\left.\begin{array}{l} \exists x Cat(x) \wedge Owns(Hamid, x) \\ \forall x(\exists y(Cat(y) \wedge Owns(x, y)) \Rightarrow Animal-lover(x, y)) \end{array}\right\} \Rightarrow \text{۲} - Animal-lover(Hamid)$$

$$\text{۱،۲.}\ \forall x \forall y(Animal-lover(x) \wedge Animal(y) \Rightarrow \sim kill(x, y)) \Rightarrow \text{۳} - \sim kill(Hamid, Pupu)$$

$$\text{۳.}\ kill(Hamid, pupu) \vee kill(Behzad, Pupu) \Rightarrow \text{حکم}$$

۲- گزینه «۲» صحیح است.

گزاره $q \Leftrightarrow (P_2 \vee P_3)$ زمانی صحیح است که هر دو طرف گزاره یا درست و یا نادرست باشند. چون $q = false$، پس ارزش q باید نادرست باشد، پس ارزش $P_2 \vee P_3$ نیز نادرست خواهد بود. پس نقیض آن یعنی $\sim p_2$ و $\sim p_3$ صحیح می‌باشد.

۳- گزینه «۱» صحیح است.

بدیهی است، طبق تعریف به متن درس مراجعه شود.

۴- گزینه «۲» صحیح است.

گزینه ۱و۳ بسادگی ثابت می‌شود که صحیح هستند. کافیست با روابط هم ارزی اشاره شده در متن درس $\forall$ و $\exists$ را به داخل پرانتز برد و دو طرف اگر آنگاه اصلی یکی خواهد شد.

اما گزینه ۲ تبدیل به: $(\forall x P(x) \rightarrow Q) \rightarrow (\exists x P(x) \rightarrow Q)$

خواهد شد که می‌توان با ساده نمودن این گزاره نادرستی آنرا ثابت نمود.

۵- گزینه «۱» صحیح است.

برای مفهوم زنجیره‌سازی رو به جلو به فصل ۶ کتاب مراجعه کنید. اعمال سه واقعیت انتهائی به سه قانون اول (سمت چپ) منجر به تولید healty (Ali) , nice weather (Saturday) خواهد شد.

(قوانین ۱ و ۳) با اعمال مجدد تنها قانون ۲ فعال شده و gotoswim (Ali. Saturday) و gotosmim (Amin. Saturday) تولید می‌شود پس تنها گزینه ۱ تولید شده است .

۶- گزینه «۲» صحیح است.
بدیهی است به متن درس مراجعه کنید.

۷- گزینه «۳» صحیح است.
بدیهی است،به متن درس مراجعه شود.

فصل پنجم

استدلال خودکار

یکی از اهداف هوش مصنوعی، ساخت سیستم‌هائی بوده که قادر باشند همانند انسان استدلال کنند شکل خاص این مسئله اثبات خودکار قضایا نام دارد، یعنی ماشین قادر باشد اثباتی برای قضیه ورودی مطرح کند. اگر بخواهیم به زبان منطق این مطلب را عنوان کنیم، به دنبال اثبات ارضا نشدنی (unsatisfiable) بودن یک wff هستیم. برای اثبات درستی wff همانند w می‌توان ثابت کرد که $\sim W$ ارضا نشدنی نیست.

برای اینکار ماشین از قانون استنتاجی رزولوشن (resolution) استفاده می‌کند. امکان استفاده از رزولوشن وجود ندارد مگر آنکه فرمول‌های خوش ترکیب (wff) به شکل خاصی که شکل کلاز (clause) نامیده می‌شود، نوشته شوند. پس در اولین قدم باید با مفهوم کلاز آشنا شویم.

۱-۵- کلازها و شکل کلازی

لیترال (literal) یا جمله‌ای اتمی و یا نقیض یک جمله اتمی است. برای مثال $p(x)$ و یا $\sim q(x,b)$ دو لیترال هستند. کلاز فصل (disjunction) صفر و یا چند لیترال است. برای مثال عبارت زیر کلاز است.

$$p(x) \vee \sim q(y) \vee p(z)$$

فرم کلازی بستار (closure) جامع فصل کلازها است. به زبان دیگر، فرم کلازی فرم نرمال فصلی پرنیکس (prenex) است که تمام سورهای آن عمومی است و هیچ متغیر آزادی در آن وجود نداشته باشد. برای راحتی نگارش، معمولاً فرم کلازی را با مجموعه‌ای ازکلازها نمایش می‌دهیم. برای مثال wff

$$(p(a) \vee p(b)) \wedge q(a,b)$$

(که در آن a و b مقادیر ثابت هستند) به صورت $\{p(a) \vee p(b) \wedge q(a,b)\}$ نمایش داده می‌شود.

اگر گزاره ورودی دارای سور وجودی باشد، برای تبدیل آن به کلازهای معادل باید ابتدا سورهای وجودی در آن گزاره را حذف کرد. برای مثال: $\forall x \exists y p(x,y)$ گزاره‌ای است که بدون حذف سور وجودی آن قابل تبدیل به کلاز معادل نیست. برای حذف سور وجودی فردی بنام اسکولم (skolem) روشی را ابداع کرده است. اگر بخواهیم در همین گزاره $\exists y$ را حذف کنیم، باید از طریق تعریف تابعی وابستگی بین سور وجودی و سور عمومی در حوزه این گزاره را ایجاد کنیم. اگر فرض کنیم معنی این گزاره آنست که برای هر فرزندی همانند x والدی مانند y وجود دارد، پس وابستگی بین x و y را می‌توان از طریق آن وابستگی y به x بیان شود و خود y حذف خواهد شد. پس حاصل به صورت $\forall x p(x, F(x))$ تبدیل خواهد شد. تابع F در این مثال تابع اسکولم نامیده می‌شود. قانون اسکولم بصورت زیر تعریف می‌شود.