فصل ششم

عاملهای منطقی

این فصل سرآغاز عامل معرفت گرا است. مفاهیمی که بحث می کنیم شامل بازنمایی معرفت و پردازش های استدلالی است که دانش را به عرصه وجود می آورد و برای کل حوزه هوش مصنوعی کلیدی است. عامل های معرفت گرا می توانند از دانشی که به شکل کلی بیان می شود استفاده نموده، و با ترکیب و باز ترکیب اطلاعات به اهداف گوناگون و مناسب برسند. البته، این پردازش می تواند بسیار فراتر از نیازهای کنونی باشد، همان طور که ریاضیدانان قضایا را ثابت می کنند و یا فضانوردان عمر زمین را محاسبه می کنند.

معرفت و استدلال نقش کلیدی در برخورد با محیط های نیمه مشاهده پذیر دارند. عامل معرفت گرا می تواند دانش عمومی خود را با ادراک کنونی ترکیب کند تا جنبه های پنهان وضعیت کنونی را قبل از انتخاب عملیات، کشف کند. برای مثال پزشک، بیماری فرد مریض را تشخیص می دهد- یعنی وضعیت بیماری را بدون مشاهده مستقیم نتیجه می گیرد- و این قبل از انتخاب درمان است. برخی معرفت ها که پزشک به کار می برد از لابه لای صفحات کتاب های درسی و گفتار استادان حاصل شده و برخی از طریق الگوهای شرکت پذیری که پزشک قادر به توصیف دقیق آن نیست ایجاد شده اند و این الگوها اگر در مغز پزشک نهفته، به عنوان دانش وی حساب می شود.

درک زبان طبیعی نیز نیاز به کشف وضعیت های نهانی دارد که به آن قصد گوینده می گویند. وقتی می شنویم "جان الماس را از پنجره دید و حسرت آن را خورد" می دانیم که "آن" به الماس بر می گردد و نه پنجره و این را با دانش ارزش سنجی خود می فهمیم. مشابهاً وقتی می شنویم "جان آجری به سمت پنجره پرت کرد و آن را شکست" می دانیم که "آن" به پنجره بر می گردد. استدلال به ما اجازه می دهد که با گستره نامتناهی مجازی بیانات، با استفاده از دانش محدود و ادراک عمومی خود، برخورد کنیم. عامل های

حل کننده مسئله با این گونه ابهامات مشکل دارند، چرا که بازنمایی مسائل احتمالی ذاتاً دارای پیچیدگی نمایی است.

آخرین استدلال ما برای علت مطالعه عامل های معرفت گرا، قابلیت انعطاف آنهاست. عاملهای معرفتگرا می توانند وظایف جدید را در قالب اهداف تشریح شده معین بپذیرند، می توانند به وسیله یاد گرفتن دانش جدید درباره محیط به سرعت به رقابت بپردازند، و آنها می توانند از طریق به هنگام سازی دانش مربوطه، خود را با تغییرات محیط هماهنگ سازند.

عامل های معرفت گرا

بخش مرکزی عامل **پایگاه دانش** (knowledge base) آن، یا KB است. به طور غیررسمی، پایگاه دانش، مجموعه ای از نمایش حقایق در مورد دنیا است. هر نمایش اختصاصی یک **جمله** (sentence) نامیده می شود. جملات در یک زبانی که زبان **بازنمایی دانش** (knowledge Representation) نامیده می شود، بیان می شوند.

باید راهی برای افزودن جملات جدید به پایگاه دانش وجود داشته باشد، و هم چنین راهی برای پرسش اینکه چه چیزهایی شناخته شده است، موجود باشد. اسامی استاندارد برای این عملیات ASK,TELL هستند. تقاضای اساسی که ما روی ASK,TELL اعمال می کنیم عبارتند از اینکه زمانی که سؤالی از پایگاه دانش ASK می شود، پاسخ باید از آن چیزی که قبلاً به پایگاه دانش گفته شده (TELLed) ناشی شود.

شکل ۶-۱ طرحی از یک برنامه عامل مبتنی بر دانش را نشان می دهد که مانند تمام عامل های قبلی، ادراکی را به عنوان ورودی دریافت می کند و عملی را بر می گرداند. عامل، پایگاه دانشی را نگهداری می کند یعنی همان KB که از ابتدا دارای **دانش پس زمینه ای** (Background Knowledge) است. هر زمان که برنامه دانش صدا زده می شود، دو عمل انجام می شود. ابتدا به پایگاه دانش گفته می شود (TELL) به این معنی که چیزی دریافت کرده است. دوم اینکه، از پایگاه دانش سؤال می شود (ASK) به این معنی که چه عملی باید انجام شود. در فرآیند پاسخ به این پرسش، استدلال منطقی برای اثبات اینکه کدام عمل بهتر

از بقیه است، استفاده می شود و دانسته های عامل و اهداف آن مشخص می شوند. سپس عامل عمل انتخاب شده را ارائه می دهد.

```
function KB-AGENT(percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

Figure 7.1 A generic knowledge-based agent.

شکل ۶-۱: عامل مبتنی بر دانش عمومی

جزئیات زبان بازنمایی داخل دو تابع که استنتاج را بین برنامه عامل "Tell" و هسته بازنمایی و سیستم استدلال پیاده سازی می کنند، پنهان شده اند. MAKE-PERCEPT-SENTENCE وظیفه گرفتن ادراک و زمان را دارد و جمله ای را بر می گرداند که این واقعیت را روشن می کند که عامل ادراک را در زمان داده شده دریافت کرده است. MAKE-ACTION-QUERY زمانی را به عنوان ورودی می گیرد و جمله ای بر می گرداند که برای درخواست عملی که باید در آن زمان انجام گیرد، مناسب باشد. جزئیات مکانیزم استنتاج داخل TELLو ASK پنهان شده است. قسمت های بعدی این جزئیات را فاش خواهند ساخت. بررسی دقیق شکل ۶-۱ نشان می دهد که کاملاً با طرح عامل هایی با وضعیت داخلی که در فصل ۲ شرح داده شده، مطابقت دارد. ما می توانیم یک عامل مبتنی بر دانش را در سه سطح تعریف کنیم:

- **سطح دانش** یا سطح epistemological که خلاصه ترین سطح است؛ می توانیم عامل را توسط گفتن اینکه عامل چه چیزهایی می داند، تعریف نماییم برای مثال، یک تاکسی اتوماتیک ممکن است آگاه باشد از اینکه پل گلدن گیت، سانفرانسیسکو را به مارین کانتی متصل می کند. اگر TELL و ASK به درستی کار کنند، بیشتر اوقات می توانیم در سطح دانش کار کنیم و نگرانی در مورد سطوح پایین تر نداشته باشیم.

- **سطح منطقی** سطحی است که دانش به صورت جملات رمزگذاری می شود. برای مثال تاکسی ممکن است که جمله ای منطقی به صورت Links(GBridge, SF, Marin) در پایگاه دانش داشته باشد.
- **سطح پیاده سازی** سطحی است که در معماری عامل اجرا می شود و بازنمایی های فیزیکی از جملات سطح منطقی، در این سطح وجود دارد. جمله ای مانند Links(GBridge, SF, Marin) در پایگاه دانش توسط رشته "Links(GBridge, SF, Marin) " بازنمایی می شود.

دنیای هیولا Wumpus world

دنیای هیولا غاری است که شامل اتاق های مرتبط به هم است. هیولا جایی از غار مخفی شده و هر کس را که به اتاق او برسد می خورد. عامل می تواند هیولا را بزند ولی تنها یک تیر دارد. برخی اتاق ها دارای چاه های بی انتهایی هستند که هر کس را که داخل آن بیفتد به تله انداخته است. (به جز هیولا که آنقدر بزرگ است که در چاه نمی افتد) تنها چیز امیدوار کننده در این محیط احتمال یافتن کومه ای از طلا است. اگر چه دنیای هیولا بازی چندان جالبی نسبت به بازی های روز کامپیوتری نیست، ولی محیط آزمون بسیار مناسبی برای عامل های هوشمند است. اولین بار "میخائیل جنسرت" آن را پیشنهاد داد. دنیای ساده هیولا در شکل ۶-۲ نشان داده شده است. تعریف دقیق محیط بر پایه توصیف PEAS (همانند فصل ۱) به قرار زیر است:

- ملاک کارایی : ۱۰۰+ برای برداشتن طلا، ۱۰۰۰- برای افتادن در چاه یا خورده شدن توسط هیولا، ۱- برای هر عمل و ۱۰- برای استفاده از تیر.
- محیط: صفحه مشبک ۴×۴ از اتاق ها. عامل همواره از مربع [۱و۱] به سمت راست شروع می کند. مکان طلا و هیولا تصادفی انتخاب می شود که این عمل با توزیع یکنواخت و با حذف خانه شروع صورت می گیرد. به علاوه، هر مربع به جز مربع آغازین می تواند چاه با احتمال ۰/۲ باشد.
- عمل کننده ها: عامل می تواند مستقیم حرکت کند، ۹۰° به چپ بچرخد یا ۹۰° به راست بچرخد. عامل اگر به خانه هیولا یا چاه برسد خواهد مرد. (واضح است رفتن به خانه ای با هیولای مرده،

مطمئن است) حرکت به سمت جلو در حالی که دیوار است هیچ اثری نخواهد داشت. عمل برداشتن برای در دست گرفتن شیئی در همان مربعی به کار می رود که عامل در آن قرار دارد. عمل تیراندازی به معنی شلیک تیر در جهتی است که عامل به همان سمت است صورت می گیرد. تیر به جلو می رود تا به هیولا بخورد (بنابراین بکشد) یا به دیوار بخورد. عامل تنها یک تیر دارد. پس اولین شلیک اثر خواهد داشت.

پس عمل کننده ها شامل:

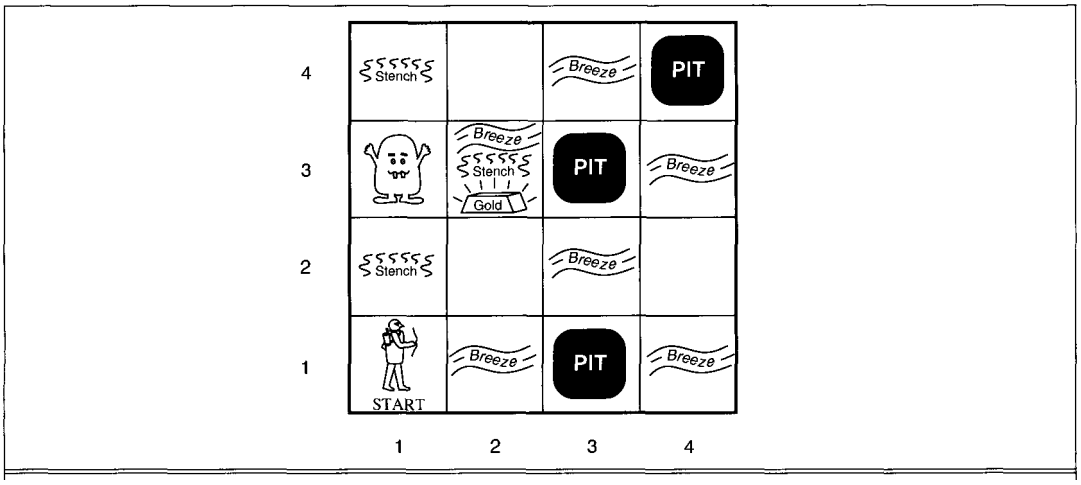
- Go Forward
- Turn Left
- Turn Right
- Grab (برداشتن طلا)
- Shoot (شلیک تیر)
- Climb(پریدن)

- حس گرها: عامل پنج حسگر دارد، هر کدام یک بخش از اطلاعات را می دهند:
  - Stench: در مربعی که هیولا در آن قرار دارد و مربع های مستقیم مجاور (نه قطری)، عامل بوی تعفنی را احساس می کند.
  - Breeze: در مربعی که مجاور چاه است، عامل نسیمی را احساس می کند.
  - Glitter: در مربعی که طلا وجود دارد، عامل درخششی را درک می کند.
  - Bump: زمانی که عامل به دیوار برخورد می کند، ضربه ای را احساس می کند.
  - Scream: زمانی که هیولا کشته می شود، فریادی سر می دهد که در تمامی غار شنیده می شود.

بدین طریق درک، شامل لیست پنج نمادی است: برای مثال اگر بوی تعفن و نسیم می آید، اما درخشش، ضربه یا جیغ نیست، درک عامل به صورت [هیچ، هیچ، هیچ ، بوی تعفن، نسیم] است.

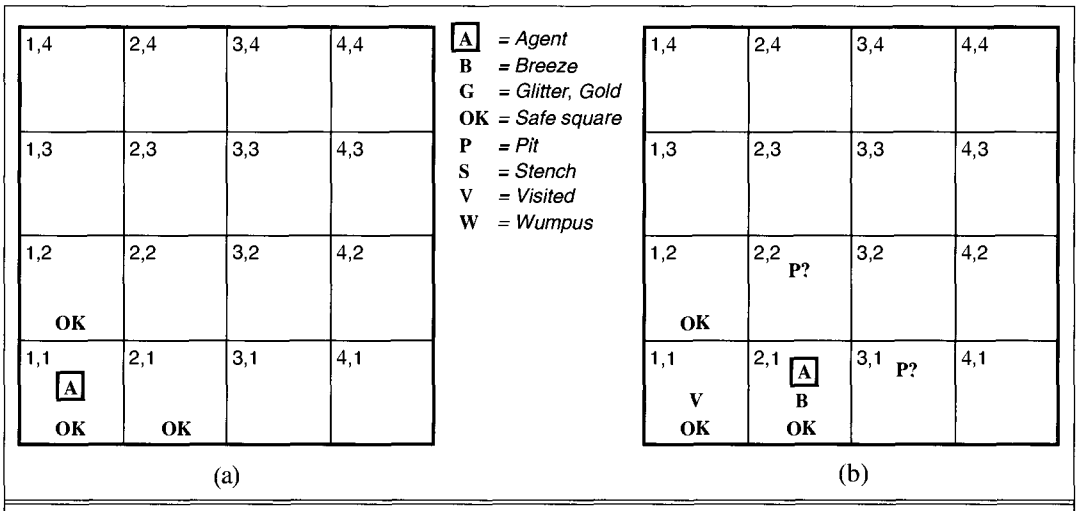
مشکل اصلی عامل عدم اطلاع از ترکیب بندی اولیه محیط است که برای فائق شدن بر آن نیاز به استدلال منطقی است. به ندرت، عامل باید بین دست خالی به خانه رفتن و ریسک مردن برای یافتن طلا انتخاب انجام دهد. در حدود ۰/۲۱ محیط ها عادلانه نیست، چرا که طلا در چاه است یا با چاه ها محاصره شده است.

بیایید نگاهی به عامل هیولای منطق گرا داشته باشیم که به کاوش در محیط شکل ۶-۲ می پردازد. پایگاه دانش اولیه عامل شامل قوانین محیط به صورتی که لیست شده است می باشد. به ویژه عامل می داند که در خانه [۱و۱] است و خانه [۱و۱] امن است. خواهیم دید که دانش وی چگونه با ادراکات و عملیات جدید رشد می کند.

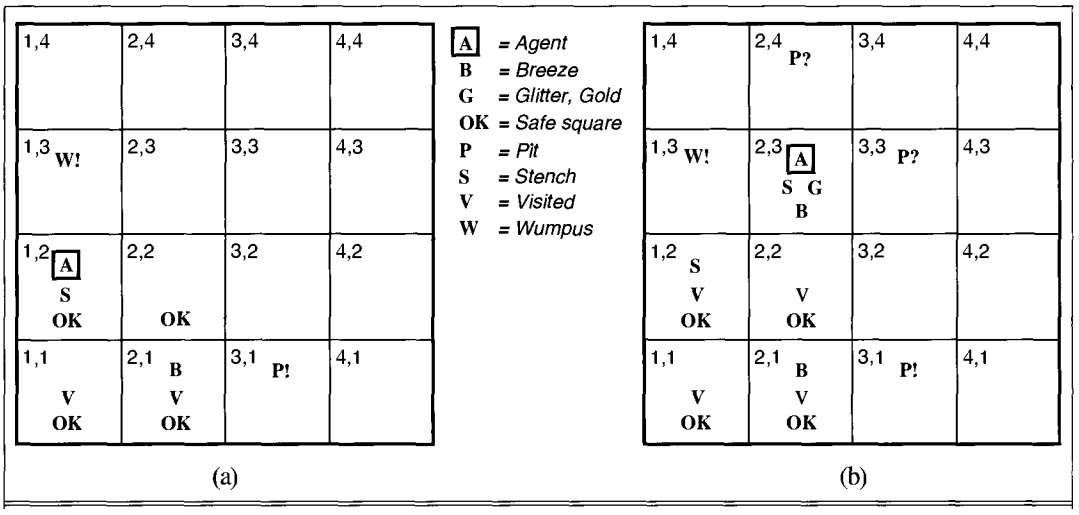


شکل ۶-۲: دنیای هیولای نمونه. عامل در گوشه پایین چپ است.

اولین ادراک [None, None, None, None, None] است که از آن عامل می تواند به این نتیجه برسد که خانه های همسایه امن هستند.



شکل ۶-۳: اولین قدم که توسط عامل در دنیای Wumpus برداشته شده است. (الف) شرایط اولیه، پس از ادراک [None, None, None, None, None]. (ب) پس از یک حرکت با ادراک.
[None, breeze, None, None, None]



شکل ۶-۴: دو مرحله بعد از کارکرد عامل. (الف) پس از سومین حرکت با ادراک [Stench, None, None, None, None]. (ب) پس از پنجمین حرکت با [Stench, Breeze, Glitter, None, None].

با توجه به این حقیقت که بوی نامطبوع و باد ملایمی در نقطه [۱و۱] وجود ندارد، عامل نتیجه می گیرد که خانه [۲و۱] و [۱و۲] برای خطر کردن آزاد هستند. آنها با کمله OK برای تعیین این قضیه، علامت خورده اند. با توجه به اینکه عامل هنوز زنده است، نتیجه می گیریم که [۱و۱] OK است. یک عامل آگاه فقط به خانه ای حرکت می کند، که در شکل ۶-۳-b نشان داده شده است.

عامل نسیم ملایمی را در خانه [۱و۲] حس می کند، بنابراین باید چاله ای در خانه های همسایه (خانه [۱و۳] یا [۲و۲]) وجود داشته باشد. علامت P? امکان وجود یک چاله را نشان می دهد. چاله نمی تواند در [۱و۱] باشد، چون، عامل قبلاً آنجا بوده و در چاله ای نیفتاده است. از این جهت، فقط یک خانه شناخته شده وجود دارد که OK است و هنوز ملاقات نشده است. بنابراین عامل باید با تدبیر دور بزند و به خانه [۱و۱] برگردد، و سپس به [۲و۱] برود. وضعیت عامل در شکل ۶-۴-a نشان داده شده است.

عامل بوی بدی را در [۲و۱] حس می کند، یعنی در همان نزدیکی ها باید Wnmpus وجود داشته باشد. اما Wnmpus نمی تواند در [۱و۱] باشد (چون باید عامل را در شروع خورده باشد)، و نمی تواند در [۲و۲] باشد (چون باید عامل بوی بدی را زمانی که در [۱و۲] بوده حس کرده باشد). بنابراین، عامل می تواند این نتیجه را بگیرد که Wumpus در [۳و۱] است. علامت W! بیانگر این مطلب است. جالب تر اینکه نبودن باد ملایمی در [۲و۱] این معنی را می دهد که در [۲و۲] چاله ای وجود ندارد. اما قبلاً متوجه شدیم که چاله ای در [۲و۲] و یا [۱و۳] وجود دارد، بنابراین چاله حتماً در [۱و۳] است. این استنباط مشکلی است، زیرا دانش به دست آمده باید در زمان ها و مکان های مختلف با هم ترکیب شود و روی عدم وجود یک ادراک تکیه کند تا مرحله قطعی را ایجاد کند. استنباط، فراتر از قابلیت های فکری بیشتر حیوانات است. اما برای عاملی که به طور منطقی استدلال می کند، مفهوم رایجی است.

بعد از این استنتاج های مؤثر، فقط یک خانه OK ملاقات نشده [۲و۲] وجود دارد، بنابراین عامل باید به آنجا برود. ما حالت عامل از دانش را در [۲و۲] نشان نخواهیم داد؛ فقط فرض می کنیم که عامل می پیچید و به [۳و۲] می رود، و شکل ۶-۴-b حاصل می شود، در [۳و۲]، عامل درخششی را تشخیص می دهد، بنابراین باید طلا را برباید و به خانه رهسپار شود، و باید به این مسئله توجه داشته باشد که در راه برگشت از خانه هایی عبور کند که علامت Ok دارند.

در ادامه این فصل، توضیح می دهیم که چطور یک عامل منطقی بسازیم که بتواند عقایدی را مانند «وجود یک چاله در [۲و۲] یا [۱و۳]» و «عدم وجود Wumpus در [۲و۲]» و همه استنباط هایی را که در فوق ذکر شد بازنمایی کند. در هر موردی که عامل از اطلاعات موجود نتیجه ای می گیرد، اگر اطلاعات

موجود درست بودند، آن نتیجه قطعاً درست است. این یک خاصیت اساسی در استدلال منطقی است. در بقیه این فصل، تشریح می کنیم که چگونه عامل منطقی بسازیم که اطلاعات لازم را بازنمایی و استدلال نماید.

منطق (logic)

منطق از دو قسمت اصلی تشکیل شده است : نحو و معنی

نحو (Syntax) : نحوه نگارش برای نمایش جملات پایگاه معرفت است.

معنی (Semantic) : معنی با مسئله مفاهیم جملات سرو کار دارد و صحت هر جمله را نسبت به دنیای ممکن یا مدل تعریف می کند.

مدل : مدلها انتزاع ریاضی هستند و هر کدام از آنها درستی یا نادرستی هر جمله مربوط ممکن را مشخص می سازد. زبانهای ارائه دانش مثل prolog یک پیاده سازی از منطق و یا یک منطق مدل شده است.

استلزام (Entailment) : این ایده که یک جمله منطقاً بعد از جمله دیگر می آید و در بیان ریاضی می نویسیم: $\propto \models \beta$. تعریف رسمی استلزام آن است که جمله $\propto$ جمله β را مستلزم کرده است اگر و فقط اگر به ازای هر مدلی که در آن جمله $\propto$ درست است، جمله β می بایست درست باشد.

$$\propto \models \beta \equiv (\beta \; entails \; \propto) \equiv (\propto \Rightarrow \beta)$$

در هوش مصنوعی در بسیاری موارد تعریف استلزام می تواند منجر به نتیجه گیری شود در این صورت به استنتاج منطقی رسیده ایم. در اغلب موارد پایگاه معرفت به عنوان یک عبارت در نظر گرفته می شود و ما درباره پایگاه معرفتی که یک جمله را مستلزم می سازد گفتگو می کنیم یعنی:

$$kB \models \propto$$

منطق گزاره ها: Propositional Logic

نحو (Syntax) : نحو منطق گزاره ای جملات مجاز را نشان می دهد.

جملات اتمی Atomic Sentence : عناصر نحوی غیرقابل تجزیه- شامل یک نماد گزاره تنها است. هر کدام از این نمادها که بر گزاره ها دلالت دارند، می توانند درست و یا نادرست باشند. ما از حروف بزرگ برای نمادها استفاده می کنیم: P، Q، R و غیره. نام ها اختیاری هستند ولی سعی می شود معانی خاصی را به خواننده تفهیم کند. برای مثال، از $W_{1,3}$ برای گزاره ای استفاده می کنیم که دلالت بر قرار گرفتن هیولا در خانه [۳و۱] دارد. دو نماد گزاره ای با معنی ثابت وجود دارد: True گزاره همیشه درست و False گزاره همیشه نادرست است.

لیترال : جمله اتمی (لیترال مثبت) و یا جمله اتمی نفی شده (لیترال منفی) است.

جملات ترکیبی : از ترکیب جملات ساده تر با استفاده از اتصالات منطقی تشکیل شده اند. پنج اتصال متداول استفاده می شود:

- **نقیض (not):** جمله ای همانند $\sim W_{1,3}$ نفی $W_{1,3}$ نام دارد
- **ترکیب عطفی (conjunct) (∧ AND)** : جمله ای که اتصال اصلی آن ∧ است، همانند $W_{1,3} \wedge P_{3,1}$.
- **ترکیب فصلی(Disyunct)(∨ OR)** : جمله ای که از ∨ استفاده می کند، همانند $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$
- **ترکیب شرطی ⇐ (نتیجه می دهد)** جمله ای همانند $(W_{1,3} \wedge P_{3,1}) \Rightarrow \sim W_{2,2}$ استنتاج (یا شرطی) نامیده می شود. استنتاج ها به عنوان قوانین یا جملات اگر- آنگاه نیز شناخته می شوند. نماد استنتاج گاهی به شکل ⊃ یا ← نوشته می شود.
- **ترکیب دو شرطی ⇔ (اگر و تنها اگر):** جمله $W_{1,3} \Leftrightarrow W_{2,2}$ شرط دو طرفه نام دارد.

شکل ۶-۵ دستور زبان رسمی منطق گزاره ها را می دهد.

$$
\begin{aligned}
\mathit{Sentence} &\rightarrow \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} &\rightarrow \mathbf{True} \mid \mathbf{False} \mid \mathit{Symbol} \\
\mathit{Symbol} &\rightarrow \mathbf{P} \mid \mathbf{Q} \mid \mathbf{R} \mid \ldots \\
\mathit{ComplexSentence} &\rightarrow \neg\ \mathit{Sentence} \\
&\mid\ (\ \mathit{Sentence} \wedge \mathit{Sentence}\ ) \\
&\mid\ (\ \mathit{Sentence} \vee \mathit{Sentence}\ ) \\
&\mid\ (\ \mathit{Sentence} \Rightarrow \mathit{Sentence}\ ) \\
&\mid\ (\ \mathit{Sentence} \Leftrightarrow \mathit{Sentence}\ )
\end{aligned}
$$

شکل ۶-۵: دستور زبان BNF جملات در منطق گزاره ها

توجه کنید که دستور زبان در مورد پرانتزها بسیار محکم است: هر جمله ای با اتصالات دو دویی می بایست در پرانتزی محصور شده باشد. این اطمینان می دهد که دستور زبان کاملاً غیر مبهم است. این به معنی آن نیز هست که به جای $A \wedge B \Rightarrow C$ بنویسیم$\big((A \wedge B) \Rightarrow C\big)$.

برای افزایش خوانایی، اغلب پرانتزها را حذف می کنیم، و به جای آن از مرتبه اولویت اتصالات استفاده می کنیم. مرتبه اولویت در منطق گزاره ها (از بالاترین تا پایین ترین): $\sim, \wedge, \vee, \Rightarrow, \Leftrightarrow$ است. بنابراین، جمله:

$\sim P \vee Q \wedge R \Rightarrow S$

هم ارز با جمله زیر است:

$\big((\sim P) \vee (Q \wedge R)\big) \Rightarrow \mathrm{S}$

اولویت، ابهام را در جملاتی همانند $A \wedge B \wedge C$ حل می کند که یا باید به صورت $\big((A \wedge B) \wedge C\big)$ و یا $\big(A \wedge (B \wedge C)\big)$ خوانده شوند. این دو نوع خواندن طبق معنی تشریح شده در بخش بعدی یک معنی می دهند، پس جمله ای همانند $A \wedge B \wedge C$ مجاز است. همچنین $A \vee B \vee C$ یا $A \Leftrightarrow B \Leftrightarrow C$ نیز مجاز هستند. جملاتی از قبیل $A \Rightarrow B \Rightarrow C$ مجاز نیستند چرا که دو نوع خواندن متفاوت معانی متفاوتی دارند و در این مورد بر پرانتزگذاری تأکید داریم. نهایتاً، گاهی از براکت های مربعی به جای پرانتزها برای شفافیت بیشتر استفاده می کنیم.

معانی (Semantic)

پس از مشخص نمودن نحو منطق گزاره ها، حال معنای آن را مشخص می سازیم. معانی قوانینی برای تعیین صحت جمله نسبت به یک مدل خاص را تعیین می کنند. در منطق گزاره ها، مدل مقادیر درستی (درست یا نادرست) را برای هر نماد گزاره ای مشخص می کند. برای مثال، اگر جمله ای در پایگاه معرفت از نمادهای گزاره ای $P_{3,1}, P_{2,2}, P_{1,2}$ استفاده کند، آنگاه یک مدل ممکن

$$m_1 = \{P_{1,2} = false, P_{2,2} = false, P_{3,1} = true\}$$

است. با این سه نماد گزاره، $2^3 = 8$ مدل ممکن وجود دارد.

معنی در منطق گزاره ها باید مشخص کند که چگونه مقدار درستی هر جمله ای را با مدل داده شده محاسبه کنیم. این عمل بصورت خود بازگشتی صورت می گیرد. تمامی جملات از جملات اتمی و پنج اتصال ساخته می شوند، بنابراین نیازمند آنیم که چگونگی محاسبه درستی جملات اتمی و چگونگی محاسبه درستی جملات تشکیل شده از پنج اتصال را، مشخص سازیم.

قبلاً گفتیم که یک پایگاه معرفت متشکل از مجموعه جملات است. حال می توانیم ببینیم که یک پایگاه معرفت متشکل از ترکیب عطفی از جملات است. یعنی، اگر با KB تهی آغاز کنیم و $Tell(KB, S_1) \dots Tell(KB, S_n)$ را انجام دهیم آنگاه خواهیم داشت: $KB = S_1 \wedge \dots \wedge S_n$. جداول درستی برای "و"، "یا" و "نفی" شبیه مفاهیم همین واژه ها در زبان است.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

شکل ۶-۶: جداول درستی برای چند اتصال منطقی.برای استفاده جهت محاسبه، به عنوان نمونه، مقدار PVQ زمانی که P درست و Q نادرست باشد، از طریق سطری که P درست است و Q نادرست است (سطر سوم) محاسبه می شود. سپس به ستون مربوط به PVQ در این سطر نگاه کنید و نتیجه درست را خواهید دید. راه دیگر آن است که به هر سطر به عنوان مدلی نگاه کنیم و محتویات هر ستون در یک سطر به ما می گوید که آیا جمله تحت آن مدل صحیح است یا خیر.

یک پایگاه معرفت ساده

تا به حال معنی منطق گزاره ها را تعریف نمودیم، پس می توانیم پایگاه معرفتی برای دنیای هیولا بسازیم. برای سادگی، تنها به چاه پرداخته ایم.

اول، نیازمند انتخاب نمادهای واژگان خود هستیم، برای هر i و j:

- $P_{i,j}$ صحیح است اگر چاهی در [i,j] باشد.
- $B_{i,j}$ صحیح است اگر نسیمی در [i,j] باشد.

پایگاه معرفت شامل جملات ذیل است که هر کدام برای روشنی برچسب خورده اند:

- چاهی در [1,1] نیست.

$R_1: \sim P_{1,1}$

- خانه ای نسیم دارد اگر و تنها اگر در همسایگی آن چاهی باشد، این برای هر خانه ای صادق است، الان تنها خانه های مربوط را شامل می کنیم:

$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

- جملات قبلی برای تمامی دنیاهای هیولا صحت دارند. حال درک نسیم برای دو خانه ملاقات شده اول در دنیای خاص عامل آن را به موقعیت شکل ۶-۳-b می رساند:

$R_4: \sim B_{1,1}$
$R_5: B_{2,1}$

پایگاه معرفت هم اکنون شامل جملات R_1 تا R_5 است. می توان آنها را با یک جمله واحد $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$ نشان داد زیرا ادعا می کند تمامی جملات مجزا صحیح هستند.

استنتاج (از طریق پیاده سازی مستقیم تعریف استلزام)

به یاد آورید که هدف استنتاج منطقی تصمیم گیری درباره $KB| = \propto$ برای جمله ای همانند $\propto$ است. برای مثال، آیا $P_{2,2}$ مستلزم است؟ الگوریتم ما برای استنتاج، پیاده سازی مستقیم تعریف استلزام بدین صورت است: مدل ها را بررسی کن و کنترل کن $\propto$ در هر مدلی که KB در آن درست است، درست یا نه.

برای منطق گزاره ها، مدل ها جایگذاری های درست یا نادرست روی هر نماد گزاره ای هستند. اگر به مثال دنیای هیولا برگردیم، نمادهای گزاره ای مربوط $P_{3,1}, P_{2,2}, P_{2,1}, P_{1,2}, P_{1,1}, B_{2,1}, B_{1,1}$ هستند. با هفت نماد، 2^7=128 مدل ممکن داریم که در سه تای آنها KB صحیح است (شکل ۶-۷). در این سه مدل، $\sim P_{1,2}$ صحیح است، پس چاهی در [۲و۱] نیست. از سوی دیگر، $P_{2,2}$ نیست.

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

شکل ۶-۷: جدول درستی ساخته شده برای پایگاه معرفت در متن. KB درست است اگر R_1 تا R_5 درست باشند که تنها در ۳ سطر از ۱۲۸ سطر روی داده است. در تمامی ۳ سطر، $P_{1,2}$ نادرست است، بنابراین چاهی در [۲و۱] نیست. از سوی دیگر ممکن است (یا ممکن نیست) چاهی در [۲و۲] باشد.

پس می توانیم به استلزام زیر دست یابیم:

$$KB \mid =\sim P_{1,2}$$

هم ارزی، اعتبار و ارضاء پذیری

قبل از آنکه به جزئیات الگوریتم های استنتاج منطقی بپردازیم، نیاز به مفاهیم جدید مربوط به استلزام داریم. این مفاهیم قابل اعمال به تمامی انواع منطق است. اما به ویژه در منطق گزاره ها نمود بهتری دارند.

اولین مفهوم هم ارزی Logical equivalence منطقی است: دو جمله α و β هم ارز منطقی هستند اگر در همان مجموعه مدل ها درست باشند که می نویسیم $\alpha \Leftrightarrow \beta$. برای مثال، به سادگی نشان داده می شود (جدول درستی) که $P \wedge Q$ و $Q \wedge P$ منطقاً هم ارزند. دیگر هم ارزی ها در شکل ۶-۸ نشان داده شده اند. تعریف متفاوت دیگر هم ارزی به صورت زیر است:

برای هر دو جمله $\propto$ وβ:

$$\propto \equiv \beta \quad iff \quad \propto \models \beta \ , \ \beta \models \propto$$

دومین مفهوم مورد نیاز **اعتبار** (Validity) است. جمله معتبر است اگر و تنها اگر در تمامی مدل ها درست باشد. برای مثال، جمله $P \vee \sim P$ معتبر است. جملات معتبر اغلب **تاتولوژی** (tautology) نام دارند، چرا که ضرورتاً درست هستند. چون جمله True در تمامی مدل ها درست است، پس هر جمله معتبری هم ارز منطقی True است. بنابر تعریف استلزام می توانیم قضیه قیاس را بدست آوریم:

برای هر $\propto$ و β ، $\propto \models \beta$ اگر و تنها اگر جمله $\propto \Rightarrow \beta$ معتبر باشد.

$$\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}$$

شکل ۶-۸: هم ارزی های استاندارد منطقی. نمادهای $\gamma, B, \propto$ بر جملات اختیاری منطق گزاره ها دلالت دارند.

مفهوم نهایی مورد لزوم **ارضاء پذیری** (Satisfiability) است. جمله ارضاء شدنی است اگر در برخی مدل ها درست باشد. برای مثال، پایگاه معرفت داده شده قبلی $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$ ارضاء شدنی است، زیرا سه مدل در آن درست است که در شکل ۶-۷ نشان داده شده است. اگر جمله ای همانند $\propto$ در یک مدل m درست باشد، می گوییم m ، $\propto$ را ارضاء می کند یا آن m یک مدل $\propto$ است. ارضاء پذیری از طریق امتحان کردن مدل های ممکن تا یافتن یک جمله ارضاء شده کنترل می شود.

استنتاج از طریق تعریف ارضا پذیری بصورت زیر انجام می شود:

$\propto \models \beta$ اگر و تنها اگر جمله $(\propto \wedge \sim\beta)$ ارضاء ناپذیر باشد.

اثبات β از $\propto$ با کنترل ارضاء پذیری $(\propto \wedge \sim\beta)$ وابسته به یک روش اثبات ریاضی استاندارد است که "برهان خلف" نام دارد. می توان جمله β را نادرست گرفت و نشان داد این به تناقض با اصل موضوع $\propto$ می رسد. این تناقض دقیقاً مثل پرسش درباره ارضاء ناپذیری جمله $(\propto \wedge \sim\beta)$ است.

استنتاج از طریق الگوهای استدلال در منطق گزاره ها

این بخش الگوهای استاندارد استنتاج که قابل اعمال به زنجیره استنتاج منجر به هدف مطلوب هستند را پوشش می دهد. این الگوهای استنتاج قوانین استنتاجی نام دارند.

- قانون انتظار (Modus ponens):

$$\frac{\propto \Rightarrow \beta \quad , \quad \propto}{\beta}$$

برای مثال، اگر

$$\frac{(\text{Wumpus Abead} \wedge \text{Wumpus Alive}) \Rightarrow \text{shoot} \quad , \quad (\text{Wumpus Abead} \wedge \text{Wumpus Alive})}{\text{shoot}}$$

- قانون حذف AND (AND Elimination)

$$\frac{\propto \wedge \beta}{\propto}$$

با در نظر گرفتن جدول درستی α, β می توان صحت Modus-Ponens و حذف AND را به سادگی اثبات کرد. این قوانین می توانند در هر مورد ویژه ای که اعمال شوند، نتایج درستی بدون نیاز به شمارش مدل ها تولید کنند.

- قانون هم ارزی: تمامی هم ارزی های منطقی شکل ۶-۸ می توانند به عنوان قوانین استنتاجی استفاده شوند. برای مثال، هم ارزی حذف دو شرطی دو قانون استنتاجی زیر را می دهد:

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}, \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

- قانون OR introduction

اگر در رابطه زیر یک α_i درست باشه پس $\alpha_1 \vee \alpha_2 \vee \alpha_3 \ldots \vee \alpha_n$ نیز درست است.

- قانون $\sim\sim\alpha$

$$\frac{\sim\sim\alpha}{\alpha}$$

- قانون Unit Resolution

$$\frac{\alpha \vee \beta \quad \sim\beta}{\alpha}$$

فرم کلی قانون را به صورت زیر می توان بیان کرد:

$$\frac{l_1 \vee l_2 \ldots . \vee l_k \quad \sim l_i}{l_1 v l_2 \ldots . v l_{i-1} v l_{i+1} v \ldots l_k}$$

- قانون Resolution

$$\frac{\alpha \vee \beta \quad \sim\beta \vee \gamma}{\alpha \vee \gamma}$$

فرم کلی قانون به صورت زیر است. :

$$\frac{l_1 \vee l_2 \ldots . \vee l_k \quad m_1 \vee m_2 \ldots . \vee m_k}{l_1 \vee l_2 \ldots . \vee l_{i-1} \vee l_{i+1} \ldots \vee l_k \vee m_1 \vee m_2 \ldots . \vee m_{j-1} \vee m_{j+1} \ldots m_n}$$

m_j ,l_i مکمل هم یا نقیض هم هستند . این قانون برای حذف کردن و ساده سازی و کوچک کردن جملات است.

اجازه دهید ببینیم این قوانین استنتاجی و هم ارزی چگونه می توانند در دنیای هیولا به کار روند. با نیست یعنی : [1,2] ∽ یعنی چاهی در $P_{1,2}$ شروع می کنیم و ثابت کنیم که R_5 تا R_1دانش اولین

$KB \mid =\sim P_{1,2}$

ابتدا حذف دو شرطی را روی R_2 اعمال می کنیم.

$$R_6: \left(\beta_{1,1} \Longrightarrow \left(P_{1,2} \vee P_{2,1}\right)\right) \wedge \left(P_{1,2} \vee P_{2,1} \Longrightarrow B_{1,1}\right)$$

سپس حذف AND را روی R6 اعمال می کنیم.

$$R_7: \left(P_{1,2} \vee P_{2,1} \Longrightarrow B_{1,1}\right)$$

هم ارزی منطقی برای ضد مثبت می دهد:

$$R_8: \sim B_{1,1} \Longrightarrow \sim\left(P_{1,2} \vee P_{2,1}\right)$$

حال Modus Ponens را روی R_8 , R_4 اعمال کرده تا:

$$R_9: \sim\left(P_{1,2} \vee P_{2,1}\right)$$

در نهایت قانون دمورگان را اعمال می کنیم:

$$R_{10}: \sim P_{1,2} \wedge \sim P_{2,1}$$

$$\sim P_{1,2}$$

یعنی نه در [1,2] و نه در [2,1] چاهی نیست. این عملیات اثبات نام دارد.

مثال دوم:

شکل ۶-۴-a را در نظر بگیرید. عامل از [2,1] به [1,1] برگشته و سپس به [1,2] می رود که بوی تعفن احساس می کند و نسیمی در کار نیست. این واقعیات را به پایگاه معرفت اضافه می کنیم.

$$R_{11}: \sim B_{1,2}$$

$$R_{12}: B_{1,2} \Leftrightarrow \left(P_{1,1} \vee P_{2,2} \vee P_{1,3}\right)$$

ثابت کنید : $\sim P_{2,2}$, $\sim P_{1,3}$

$$R_{11}: \sim B_{1,2}$$

قانون ترکیب دو شرطی :

$$R_{12}: [B_{1,2} \Longrightarrow (\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{1,3})] \wedge [(\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{1,3}) \Longrightarrow \mathrm{B}_{1,2}]$$

هم ارزی منطقی برای ضد مثبت:

$$\sim B_{1,2}$$
$$\sim \mathrm{B}_{1,2} \Rightarrow \sim (\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{1,3})$$

طبق قانون Modus ponens نتیجه زیر گرفته می شود:

$$\sim (\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{1,3})$$

دمورگان:

$$\sim \mathrm{P}_{1,1} \wedge \sim \mathrm{P}_{2,2} \wedge \sim \mathrm{P}_{1,3}$$

حذف AND:

$$R_{13}: \sim P_{2,2}$$
$$R_{14}: \sim P_{1,3}$$

به عنوان مثال دیگری ثابت کنید : $\mathrm{P}_{3,1}$

با اعمال حذف شرط دو طرفی به R3 به همراه modus ponens به R5 به این واقعیت می رسیم که چاهی در [1,1] یا [۲و۲] یا [۱و۳] وجود دارد:

$$R_3: \mathrm{B}_{2,1} \Leftrightarrow (\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{3,1})$$
$$R_5: \mathrm{B}_{2,1}$$

قانون ترکیب دو شرطی :

$$[B_{2,1} \Longrightarrow (\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{3,1})] \wedge [(\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{3,1}) \Longrightarrow \mathrm{B}_{2,1}]$$

حذف AND و Modus Ponens:

$$\frac{B_{2,1} \Longrightarrow (\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{3,1}) \qquad \mathrm{B}_{2,1}}{\mathrm{P}_{1,1} \vee \mathrm{P}_{2,2} \vee \mathrm{P}_{3,1}}$$

حال اولین کاربرد قانون رزولوشن پیش می آید: لیترال $\sim \mathrm{P}_{2,2}$ در R_{13} با لیترال $\mathrm{P}_{2,2}$ در R_{15} حل شده و می دهد.

$$\frac{R_{15}: (P_{1,1} \vee P_{2,2} \vee P_{3,1}) \quad R_{13}: \sim P_{2,2}}{P_{1,1} \vee P_{3,1}}$$

$R_{16}: P_{1,1} \vee P_{3,1}$

به زبان فارسی اگر چاهی در یکی از خانه های [۱و۱]، [۲و۲] و[۱و۳] باشد و در [۲و۲] نباشد پس در حل شده تا داشته باشیم: R_{16} در $P_{1,1}$ بالیترال $\sim R_{1,1}$: R_1[۱و۱] یا [۱و۳] داشت. مشابهاً

$$\frac{R_{16}: P_{1,1} \vee P_{3,1} \quad R_1: \sim P_{1,1}}{P_{3,1}}$$

$P_{3,1}$: R17 به زبان فارسی اگر چاه در [۱و۱] یا [۱و۳] باشد و در [۱و۱] نباشد، پس در [۱و۳] است. دو مرحله استنتاجی آخر مثال هایی از رزولوشن واحد قوانین استنتاجی هستند.

فرم نرمال عطفی Conjunctive Normal Form (CNF)

قانون رزولوشن تنها قابل اعمال به ترکیب فصلی لیترالهاست و هر جمله منطق گزاره ها نیز هم ارز منطقی با یک ترکیب عطفی از ترکیبات فصلی لیترالهاست.

جمله ای که به صورت ترکیب عطفی از ترکیبات فصلی لیترالها بیان شوم فرم CNF دارد.

$(l_{1,1} \vee \dots \vee l_{1,k}) \wedge \dots . \wedge (l_{n,1} \vee \dots \vee l_{n,k})$

به هر ترکیب فصلی از لیترالها یک کلاز (clause) می گویند. هر قانون را می توان به صورت CNF نوشت. قانون R_2 به فرم CNF به صورت زیر است :

$B_{11} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

۱. حذف $\Leftrightarrow$: جایگزینی $\alpha \Leftrightarrow \beta$ با $(\beta \Rightarrow \alpha) \wedge (\alpha \Rightarrow \beta)$

$$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

۲. حذف $\Rightarrow$: جایگزینی $\alpha \Rightarrow \beta$ با $\sim \alpha \vee \beta$

$$(\sim B_{11} \vee P_{1,2} \vee P_{2,1}) \wedge (\sim (P_{1,2} \vee P_{2,1}) \vee B_{11})$$

۳. CNF نیازمند ظهور~ تنها در لیترال هاست، بنابراین ~ را به داخل حرکت می دهیم و این کار به کمک قوانین هم ارزی زیر انجام می شود:

$$\sim(\sim\alpha) \equiv \alpha$$
$$\sim(\alpha \wedge \beta) \equiv \sim\alpha \vee \sim\beta$$
$$\sim(\alpha \vee \beta) \equiv \sim\alpha \wedge \sim\beta$$

در این مثال تنها قانون آخری کاربرد دارد:

$$((\sim B_{11} \vee P_{1,2} \vee P_{2,1}) \wedge \left((\sim P_{1,2} \wedge \sim P_{2,1}) \vee B_{11}\right)$$

۴. حال جمله ای شامل ∨,∧ های تو در تو که به لیترال ها اعمال شده اند، داریم. قوانین توزیع پذیری را استفاده کرده و ∨ را روی ∧ تا جای ممکن توزیع می دهیم.

$$(\sim B_{11} \vee P_{1,2} \vee P_{2,1}) \wedge (\sim P_{1,2} \vee B_{11}) \wedge (\sim P_{2,1} \vee B_{11})$$

جمله اصلی حالا به یک *CNF* تبدیل شده که عطفی از سه کلاز است. این می تواند به عنوان ورودی روال رزولوشن استفاده شود.

استنتاج براساس رزولوشن و فرم نرمال عطفی:

روال استنتاج با رزولوشن برای اثبات هر جمله از برهان خلف استفاده می کند. یعنی برای نمایش اینکه $kB \models \propto$ است نشان می دهد که $(kB \wedge \sim \propto)$ دارای تناقض بوده و ارضا ناپذیر است.

الگوریتم رزولوشن اول $(kB \wedge \sim \propto)$ را به *CNF* تبدیل می کند. سپس قانون رزولوشن به کلازهای حاصل اعمال خواهد شد. هر جفت که شامل لیترال های متضاد است با هم ادغام شده تا تولید یک کلاز جدید بکند، که به مجموعه اضافه خواهد شد. این فرآیند ادامه یافته تا یکی از موارد زیر روی دهد:

- کلاز جدیدی برای افزودن وجود نداشته باشد، که در این حالت β, α را مستلزم نمی سازد و اثبات صورت نمی گیرد.
- به کارگیری قانون رزولوشن یک کلاز تهی تولید کند که در این حالت β, α را مستلزم خواهد ساخت. چون کلاز تهی هم ارز با نادرست است. و بنابراین

$kB \wedge \sim\alpha \equiv F$
$\sim \propto \equiv F$
$\propto \equiv T$

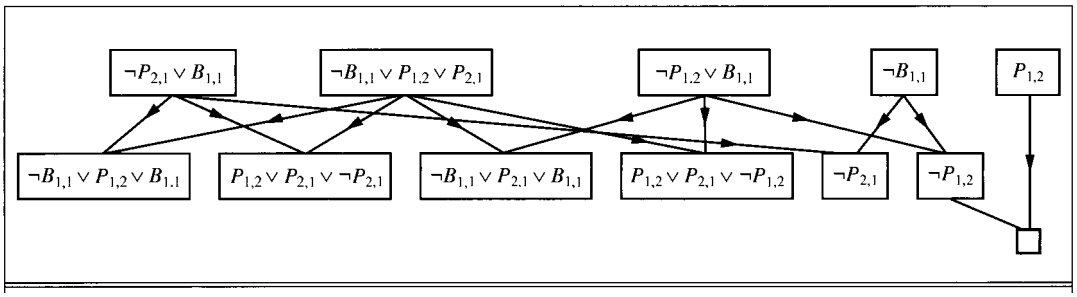
مثال در دنیای هیولا:

وقتی عامل در [۱و۱] است هیچ نسیمی نیست پس چاهی در همسایگی نمی تواند وجود داشته باشد ثابت کنید $\sim P_{1,2}$

پایگاه معرفت مربوط به قرار:

$$KB = R_2 \wedge R_4 = \left(\left(B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}\right) \wedge \sim B_{1,1}\right)$$

است و ما می خواهیم α را ثابت کنیم که همان $\sim P_{1,2}$ است. در زمان تبدیل $(kB \wedge \sim\alpha)$ به CNF کلازهای نشان داده شده در بالای شکل ۶-۹ تولید خواهد شد. سطر دوم شکل تمامی کلازهای حاصل از اعمال رزولوشن جفتی بر سطر اول را نشان می دهد. اگر $P_{1,2}$ را با $\sim P_{1,2}$ ادغام کنیم به کلاز تهی منجر می شود که با مربع کوچکی نشان داده شده است. وارسی شکل ۶-۹ آشکار می سازد که مراحل رزولوشن پایان ناپذیرند. برای مثال کلاز $B_{1,1} \vee \sim B_{1,1} \vee P_{1,2}$ با $Trune \vee P_{1,2}$ هم ارز است که با True درست است کمک چندانی نمی کند. پس هر کلازی را که در آن دو لیترال متضاد وجود دارد دور می ریزیم.



شکل ۶-۹ یک استنتاج ساده در دنیای هیولا، $\sim P_{1,2}$ از چهار کلاز اول در سطر بالا به دست آمده است.

کلاز هورن

پایگاه های معرفت دنیای واقعی اغلب تنها شامل کلازهای محدود شده ای به نام کلازهای هورن هستند. یک کلاز هورن از فصل لیترال هایی تشکیل شده است که در آن حداکثر یکی مثبت است. برای مثال کلاز $(\sim L_{1,1} \vee \sim \text{Breeze} \vee B_{1,1})$ که $L_{1,1}$ به معنی مکان عامل در [۱و۱] است، یک کلاز هورن است

حال آنکه ($\sim B_{1,1}\ Vp_{2,1}\ \vee P_{2,1}$) نیست. کلازهای هورن به این شکل که دقیقاً یک لیترال مثبت دارند، کلازهای صریح نامیده می شوند. لیترال های مثبت سر(head) و لیترالهای منفی بدنه کلاز (body) نامیده می شوند. کلاز صریح که هیچ لیترال منفی ندارد تنها یک گزاره ساده است و واقعیت (fact) نامیده می شود.

محدودیت تنها یک لیترال مثبت از جنبه ای به نظر جالب نیست ولی به سه دلیل فوق العاده مهم است:

۱. هر کلاز هورن به عنوان یک شرط نوشته می شود که مقدم آن عطف لیترال های مثبت و تالی آن یک لیترال مثبت واحد است. برای مثال، کلاز هورن($\sim L_{1,1} \vee \sim Breeze\ Vb_{1,1}$) می تواند به صورت شرط $(L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$ نوشته شود. در شکل آخری جمله راحت تر قابل خواندن است، و می گویند که اگر عامل در [۱و۱] باشد و آنجا نسیم بیاید، آنگاه [۱و۱] نسیمی است. انسان می تواند به راحتی جملات را برای دامنه های متعددی از دانش بخواند و بنویسد.

یک کلاز هورن با هیچ لیترال مثبتی می تواند به صورت شرطی که نتیجه آن لیترال false است نوشته شود. برای مثال کلاز ($\sim w_{1,1} \vee \sim w_{1,2}$) هم ارز با $w_{1,1} \wedge w_{1,2} \Rightarrow false$ است. چنین جملاتی محدودیت های عمومی در دنیای پایگاه داده نامیده می شوند که معرف خطا در داده هستند. در الگوریتم ذکر شده آتی ، برای سادگی فرض می کنیم که پایگاه معرفت تنها شامل کلازهای صریح است و فاقد هر نوع محدودیت های عمومی است. می گوییم این پایگاه های معرفت در شکل هورن هستند.

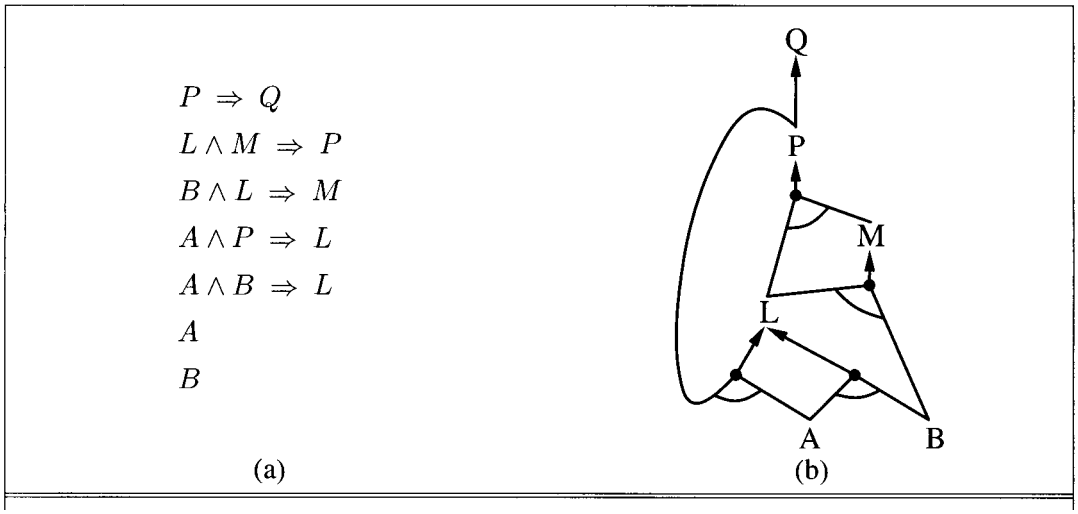
۲.استنتاج با کلازهای هورن می تواند از طریق الگوریتم های زنجیره سازی به جلو و عقب که متعاقبا توضیح داده شده، انجام شود. هر دوی این الگوریتم ها بسیار طبیعی هستند از این منظر که مراحل استنتاج آشکار بوده و برای انسان به راحتی قابل تعقیب هستند.

۳.تصمیم گیری استلزام با کلازهای هورن در زمان خطی نسبت به اندازه پایگاه معرفت قابل انجام است.

استنتاج از طریق زنجیره سازی روبه جلو وعقب

در الگوریتم زنجیره سازی روبه جلو PL-FC-Entails?(KB,q) تعیین می کند که آیا یک گزاره واحد با نماد q در یک پایگاه معرفت از کلازهای هورن مستلزم شده یا خیر. ابتدا از واقعیات شناخته شده شروع کرده، سپس نتایج جدید به مجموعه واقعیات شناخته شده اضافه خواهد شد. برای مثال اگر $L_{1,1}$ و Breeze شناخته باشند و $(L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$ در پایگاه معرفت باشد، $B_{1,1}$ می تواند اضافه شود. این فرایند تا جایی که درخواست q اضافه شود و یا استنتاج دیگری وجود نداشته باشد، ادامه خواهد یافت. نکته مهم برای یادآوری آنکه اجرا در زمان خطی انجام می شود.

بهترین راه درک الگوریتم از طریق یک مثال و تصویر است. در شکل ۶-۱۰-a یک پایگاه معرفت ساده از کلازهای هورن و A وB به عنوان واقعیات شناخته شده است. شکل ۶-۱۰-b همان پایگاه معرفت را به صورت گراف AND-OR نشان داده است. در گراف های AND-OR اتصالات متعدد می توانند توسط یک قوس که معرف عطف است به هم متصل شوند، در حالی که اتصالات متعدد بدون قوس معرف فصل هستند. به سادگی می توان دید چگونه زنجیره سازی رو به جلو روی گراف کار می کند. برگ های شناخته شده(A وB در اینجا) واقعیت هستند و انتشار استنتاج تا جای ممکن به بالای گراف صورت می گیرد. هر گاه یک عطف پدیدار شد، انتشار تا برقراری تمامی شرط های عطف متوقف خواهد شد. خواننده با تعقیب مثال با جزئیات آشنا خواهد شد. در شکل ۶- ۱۱ الگوریتم زنجیره سازی به جلونشان داده شده است.



شکل۶-۱۰: (a) یک پایگاه معرفت مبتنی بر کلازهای هورن (b) گراف AND-OR وابسته به آن
زنجیره سازی رو به عقب، همان طور که از نام آن پیدا است، درخواست به عقب بر می گردد.

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional Horn clauses
          q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true in KB

  while agenda is not empty do
    p ← POP(agenda)
    unless inferred[p] do
      inferred[p] ← true
      for each Horn clause c in whose premise p appears do
        decrement count[c]
        if count[c] = 0 then do
          if HEAD[c] = q then return true
          PUSH(HEAD[c], agenda)
  return false
```

شکل ۶-۱۱ الگوریتم زنجیره سازی روبه جلو برای منطق گزاره ها. Agenda نمادهای درست شناخته شده را حفظ می کند، اما هنوز پردازش نمی کند. جدول count تعداد مقدم های هر شرط را که ناشناخته است حفظ می کند. هر گاه نماد جدیدی مثل p از دستور کار، پردازش شد، شمارش برای هر شرطی که در مقدم آن، p ظاهر شده، یکی کم می شود. (این کار در زمان ثابت در صورت شاخص بندی مناسب KB انجام می شود) اگر شمارش به صفر رسید، تمامی مقدم های شرط شناخته شده اند، پس می تواند به دستور اضافه شود. نهایتاً، می بایست نمادهای مورد نظر برای پردازش را حفظ کنیم. یک نماد نتیجه گرفته شده اگر قبلاً پردازش شده نمی بایست به دستور کار اضافه شود. این جلوی کار اضافی را می گیرد، همچنین از حلقه های نامتناهی که بر اثر شرط هایی مثل $P \Rightarrow Q$ و $Q \Rightarrow P$ تولید می شوند، جلوگیری خواهد کرد.

زنجیره سازی رو به عقب همانطور که از نام آن پیداست از درخواست به عقب بر می گردد. اگر درخواست Q درست شناخته شود نیاز به هیچ کاری نیست. در غیر این صورت، الگوریتم آن شرط هایی که

در پایگاه معروف Q را نتیجه می گیرند، خواهد یافت. اگر تمامی مقدم های یکی از آن شرط ها قابل اثبات درستی بود(با زنجیره سازی به عقب) آنگاه q نیز درست است. اگر درخواست Q درشکل۶-۱۰ اعمال کنیم، در گراف به عقب برگشته تا به مجموعه واقعیات شناخته شده که تشکیل پایه اثبات را می دهند، برسد. همانند زنجیره سازی رو به جلو در صورت پیاده سازی کارا زمان خطی خواهد بود.

زنجیره سازی روبه عقب شکلی از استدلال هدف گرا است.این برای پاسخ به پرسش های مثل"اکنون چه کنیم ؟" و " کلیدهای من کجا هستند؟" مفید است. اغلب هزینه زنجیره سازی رو به عقب بسیار کمتر از خطی نسبت به اندازه پایگاه معرفت است، زیرا فرایند تنها واقعیات را لمس می کند.

استنتاج گزاره ای بر اساس کنترل مدل

الگوریتم های این قسمت در جهت کنترل ارضاء پذیری کار می کنند. در استنتاج $KB \models \alpha$ ، این الگوریتم ها به دنبال مدل های ارضاء پذیر برای KB هستند. سپس چک می کنند که آیا در این مدلها α درست است یا خیر.

الگوریتم پی جویی به عقب کامل DPLL

این الگوریتم توسط دیویس- پونتام- لوگمن- لاونر(۱۹۶۲) مطرح شد و بنام چهار نفر طراح آن ثبت شده است. DPLL یک جمله در شکل نرمال عطفی به عنوان ورودی دریافت می کند که همان مجموعه کلازهاست. سپس یک پویشگربازگشتی اول عمق، مدلهای ممکن را آزمایش می کند تا مدلهایی را بیابد که ورودی را ارضاء می سازند. اگر مدل بدست آمده α را نیز ارضاء سازد جمله α استلزام خواهد شد.

این الگوریتم دارای مزیت پایان زود هنگام است زیرا کلاز درست است اگر هر لیترال درست باشد، حتی اگر دیگر لیترالها هنوز جدول درستی ندارند. پس جمله در کل حتی قبل از آنکه مدل کامل شود، مورد قضاوت کلی قرار خواهد گرفت. برای مثال جمله $(A \vee B) \wedge (A \vee C)$ درست است اگر A درست باشد، با

وجود اینکه مقادیر B و C هرچه باشند. مشابهاً جمله نادرست است، اگرهر کلازی نادرست باشد که زمانی روی می دهد که هر لیترال آن نادرست باشد.

الگوریتم جستجوی محلی WALKSAT

تاکنون الگوریتم های جستجوی محلی متعددی همانند تپه نوردی و بازپخت شبیه سازی شده را دیدیم. این الگوریتم ها می توانند مستقیماً برای مسائل ارضاء پذیری به کاررروند، با این فرض که تابع ارزیابی درستی انتخاب کنیم. از آنجا که هدف یافتن یک جایگذاری است که هر کلازی را ارضاء کند، تابع ارزیابی که تعداد کلازهای ارضاء نشده را شمارش می کند، کار را انجام می دهد.

یکی از ساده ترین و موثرترین الگوریتم های تولید شده از تمام این کارهاWALKSAT نام دارد. در هر تکرار، الگوریتم یک کلاز ارضاء نشده و یک نماد را در کلاز برای معکوس کردن انتخاب می کند. دو راه برای انتخاب نماد مورد نظر به طور تصادفی استفاده می شود:

۱. " حداقل تناقض" که تعداد کلازهای ارضاء ناپذیری را در حالت جدید به حداقل می رساند.

۲. حرکت تصادفی