**هوش مصنوعي**

**Artificial Intelligence**

**نام كتاب :**

هوش مصنوعي رهياتي نوين

**مولف :**

راسل و نورويگ

**مترجم :**

رامين رهنمون آناهيتا هماوندي

فصل اول

**هوش مصنوعي**

**Artificial Intelligence**

AI: به طور رسمي در سال 1956 مطرح شده است.

# علل مطالعه Al:

AI سعي دارد تا موجوديت‌هاي هوشمند را درک کند. از اين رو يکي از علل مطالعه آن يادگيري بيشتر در مورد خودمان است.

جالب و مفيد بودن موجوديت‌هاي هوشمند .

# AI چيست؟‌

تعاريفي از AI که به چهار قسمت تقسيم شده‌اند:

پردازش فکري و استدلالي

پردازش رفتاري

ايده‌آل هوشمندي (منطقي بودن)

ارائه انساني

**ايده‌آل هوشمندي**

**تمرکز بر روي پردازش‌هاي رفتاري**

**پردازش‌هاي فکري و استدلالي**

سيستم‌هايي که مانند انسان فکر مي‌کنند

سيستم‌هايي که به طور منطقي فکر مي‌کنند

سيستم‌هايي که مانند انسان عمل مي‌کنند

سيستم‌هايي که به طور منطقي عمل مي‌کنند

**ارائه انساني**

# 1- انسان گونه عمل کردن: رهيافت آزمون تورينگ

آزموني از کامپيوتر به عمل آيد، و آزمون گيرنده نتواند دريابد که در آن طرف انسان قرار دارد يا کامپيوتر.

براي اين کار کامپيوتر بايد قابليت‌هاي زير را داشته باشد:

* پردازش زبان طبيعي = محاوره
* بازنمايي دانش= ذخيره اطلاعات
* استدلال خودکار= استدلال و استخراج
* يادگيري ماشيني= کشف الگو و برون ريزي

# تست تورينگ:

اين آزمون از ارتباط فيزيکي مستقيم بين کامپيوتر و محقق اجتناب مي‌کند.

به منظور قبول شدن در تست تورينگ کلي، کامپيوتر به موارد زير احتياج دارد:

* بينايي ماشين براي درک اشياء
* روباتيک به منظور حرکت آنها

# 2. انساني فکر کردن-: رهيافت مدلسازي شناختي:

* چگونگي شناسايي عملکرد افکار انسان:
* 1- درون گرايي
* 2- تجارب روانشناسي
* علوم شناختي : مدل‌هاي کامپيوتر از AI و همچنين تکنيک‌هاي روانشناختي را گرد هم مي‌آورد تا بتواند تئوري‌هاي دقيقي از کارکرد ذهن انسان به دست آورند.

# 3. منطقي فکر کردن: قوانين رهيافت تفکر

* رمز «تفکر درست»: ارسطو سعي در کشف آن داشت.
* قياس: از موضوعات مطرح شده توسط ارسطو مي‌باشد، که الگوهايي براي ساختار توافقي ايجاد کرد که همواره نتايج صحيحي به اندازه مقدمات صحيح به دست مي‌آورد.
* مثال: «سقراط انسان است، تمام انسان‌ها مي‌ميرند، پس سقراط خواهد مرد.»

# دو مشکل عمده در اين رسم منطق‌گرايي وجود دارد:

* تبديل دانش غير رسمي به شکل رسمي توسط اعلام، منطقي ساده نيست.
* تفاوت عمده‌اي بين قادر به حل مسئله بودن در اصول و انجام آن در عمل وجود دارد.

# 4. منطقي عمل کردن: رهيافت عامل منطقي

* عامل: در اصل چيزي است که ابتدا درک مي‌کند و سپس عمل مي‌کند.
* در نگرش «قوانين تفکر» تأکيد عمده بر روي استنتاج‌هاي صحيح بوده است.
* «مهارت‌هاي شناخت» که براي آزمون تورينگ موردنياز است، براي انجام فعاليت‌هاي منطقي وجود دارند.

# مزاياي مطالعه AI به‌عنوان طراحي عامل منطقي:

* عمومي‌تر از رهيافت «قوانين تفکر»
* پيشرفت علمي، بسيار قانون‌پذيرتر از رهيافت‌هايي است که بر تفکر يا رفتار انساني متکي هستند.

# زيربناي هوش مصنوعي:

AI، از علوم مختلفي بهره مي‌برد که از ميان آنها علوم زير مهم‌تر شناخته شده‌اند:

* علم فلسفه
* علم رياضي
* علم روانشناسي
* علم زبان‌شناسي
* علم کامپيوتر

# فلسفه: (428 قبل از ميلاد مسيح – تاکنون)

پايه‌هاي تفکر و فرهنگ غرب تشکيل شده است از: افلاطون، استادش سقراط، و شاگردش ارسطو.

قياس: ارسطو، سيستمي غيررسمي از قياس براي استدلال مناسب توسعه داد، امکان توليد نتايج، بر پايه فرضيات اوليه به طور مکانيکي وجود داشت.

در نظر گرفتن ذهن به‌عنوان سيستمي فيزيکي

رنه دکارت مدافع سرسخت قدرت استدلال بود؛ و همچنين طرفدار مکتب دواليسم.

ماترياليسم: در مقابل دواليسم قرار دارد و معتقد است تمامي جهان مطابق قوانين فيزيکي عمل مي‌کنند.

**ويلهم لايبنيز:**

* تبديل موقعيت ماترياليستي به نتايج منطقي
* ساخت ابزاري مکانيکي براي انجام عمليات منطقي

# ايجاد منبع دانش:

فرانسيس بيکن، جنبش آزمون‌گرايان را آغاز کرد. و با شعار جان لاک مفهوم يافت:

«هيچ چيز قابل فهم نيست اگر ابتدا در حس نباشد.»

اصل استقراي امروزي، در حقيقت از کتاب ديويد هيوم نشأت مي‌گيرد: "رسانه‌اي از طبيعت انسان"

برتراندراسل، پايه‌گذار پوزيوتيزم منطقي، ارائه‌دهندة اين تئوري بود که:

«قوانين عمومي توسط تکرار ارتباطات بين عناصر آنها به وجود مي‌آيند.»

# ارتباط بين دانش و عمل

اشياء را با تحليل، دسته‌بندي مي‌کنيم و در اطراف آنها، کارکرد مورد نيازشان نوسان مي‌نمايد.

در اين ميان پايه سيستم‌مکاشفه‌اي GPS بنيان گذارده مي‌شود.

# رياضيات (800. C-تاکنون)

براي ارتباط فلسفه با دانش نظري، نياز به فرمول‌سازي رياضي در سه زمينه اصلي است:

* محاسبات
* منطق
* احتمالات

# محاسبات:

نظريه اظهار محاسبات به عنوان الگوريتمي رسمي به خوارزمي برمي‌گردد، رياضيدان عربي قرن نهم که نوشته‌هاي وي، جبر و تئوري اعداد عربي را به اروپا معرفي کرد.

# منطق:

در اين زمينه، دانشمندان زيادي بر چگونگي شکل‌گيري و هدايت آن، نقش داشته‌اند که به چند نفر از آنها اشاره مي‌کنيم:

**ارسطو:** دانشمندي که بيشترين شکل‌گيري نگرش فلسفي منطق را به او نسبت مي‌دهند.

**جورج بول:** يک زبان رسمي براي ساخت استنتاج منطقي ارائه داد.

**FREGE:** منطق مرتبه اول را به شکلي مطرح نمود که در بيشتر سيستم‌هاي نمايش دانش پايه استفاده مي‌شود.

آلفرد تارسکي: تئوري چگونگي ارتباط بين اشياء موجود در محيط منطقي، و اشياء موجود در دنياي واقعي را ارائه نمود.

**ديويد هيلبرت:** رياضيدان بزرگي بود که شهرت وي به دليل مسائلي است که نتوانست حل کند.

**راسل:** قضيه کامل نبودن (incompleteness) را مطرح نمود.

**تورينگ:** ماشين تورينگ قادر به محاسبه هر تابع محاسبه‌پذيري است.

**تئوري پيچيدگي:**

انجام‌ناپذيري ، استحاله

**استيون کوک و ريچارد کارپ:** تئوري NP-completeness را مطرح کردند.

# احتمالات:

**گاردنيوي:** اولين کسي بود که ايده احتمال را مطرح کرد.

**پير فرمت، پاسکال، برنولي، لاپلاس** و ديگر دانشمندان بر رشد و توسعه اين ايده تأثير داشتند.

برنولي: ديدگاه «درجه باور» ذهني را در مقايسه با نرخ نتايج عيني مطرح کرد.

**بيس:** قانوني براي بهنگام‌سازي احتمالات ذهني را به وجود آورد.

**نيومن و مورگنسترن:** تئوري تصميم‌گيري را آغاز کردند. و از ترکيب تئوري احتمال، و تئوري سودمندي حاصل مي‌شود.

# روانشناسي (1879- تاکنون):

**هلمولتز:** روشي علمي براي مطالعه بينايي انسان به کار برد؛ که اين کتاب به عنوان مرجع بينايي فيزيولوژيک و حتي به‌عنوان «مهمترين رساله فيزيکي و روانشناختي بينايي انسان تا به امروز» شناخته مي‌شود.

**وندت:** اولين آزمايشگاه روانشناسي تجربي را در دانشگاه لايپزيک راه‌اندازي کرد.

**داتسون و تورن دايک:** حرکت رفتارگرايي (behaviorism) را مطرح کردند.

اساس مشخصه روانشناسي شناختي(congnitive psychology)، اين نگرش است که مغز دارنده و پردازش‌کننده اطلاعات است.

کريک، کتاب ماهيت بيان را منتشر کرد. و سه مرحله کليدي را براي عامل مبتني بر داشن معين کرد:

* محرک‌ها بايد به شکل دروني تبديل شوند.
* بازنمايي توسط پردازش‌هاي شناختي بازنمايي‌هاي داخلي جديدي را مشتق کند.
* اينها دوباره به صورت عمل برگردند.

**مهندسي کامپيوتر (1940- تاکنون)**

براي پيشرفت هوش مصنوعي، به دو چيز احتياج داريم:

هوش ، محصول مصنوعي

در اين تقسيم‌بندي، کامپيوتر مي‌تواند به عنوان محصول مصنوعي محسوب گردد.

**Heath Robinson:** اولين کامپيوتر مدرن عملياتي بود که در سال 1940 توسط تيم آلن تورينگ به منظور کدگشايي پيام‌هاي آلمان‌ها ساخته شد.

**Colossus:** نام ماشين بعدي بود که تيوپ‌هاي مکنده در آن به کار برده شد.

**Z-3:** اولين کامپيوتر قابل برنامه‌ريزي که توسط کنراد زوس در 1941 اختراع شد.

**اعداد با مميز شناور و زبان Plankalkul** نيز توسط زوس اختراع شدند.

**ABC:** اولين کامپيوتر الکترونيک در امريکا توسط جان آتاناسف و کليفورد در دانشگاه ايالتي ايوا ساخته شد.

**MARK I , II , III:** توسط تيمي به رهبري هوراد ايکن در هاروارد توسعه داده شد.

**ENIAC:** اولين کامپيوترديجيتال الکترونيک چند منظوره، توسط تيمي به سرپرستي ماچلي و اکرت در دانشگاه پنسيلوانيا ساخته شد.

**IBM 701:** اولين کامپيوتر سودآور، توسط ناتانيل روچتر در 1952 ساخته شد.

**چارلز بابيج:** طراحي ماشيني که جداول لگاريتمي را محاسبه کند.

* طراحي موتور آناليتيکي
* طرح حافظه قابل‌آدرس‌دهي، برنامه ذخيره شده و پرش‌هاي شرطي

کار در زمينه AI منجر به ايده‌هاي بسيار متعددي شد که به علوم کامپيوتر برگشت؛ مانند:

اشتراک زماني – مفسرهاي دوسويه – نوع داده ليست پيوندي – مديريت حافظه خودکار و برخي نکات کليدي برنامه‌نويسي شيءگرا و محيط‌هاي توسعه برنامه مجتمع با واسط کاربر گرافيکي.

# زبان‌شناسي (1975- تاکنون)

اسکينر در سال 1975 کتابي در زمينه رفتارگرايان براي يادگيري زبان، با نام «رفتار زباني» منتشر کرد.

نوآم چامسکي بر اساس تئوري خودش يعني ساختارهاي ترکيبي، اين کتاب را تجديد نظر و چاپ کرد. که به اندازه اصل کتاب شهرت پيدا کرد.

تئوري چامسکي بر اساس مدل‌هاي نحوي قرار دارد.

زبان‌شناسي مدرن و AI در يک زمان متولد شدند، بنابراين زبان‌شناسي نقش مهمي در رشد AI بازي نمي‌کند.

اين دو دريک زمينه مشترک به نام

زبان‌شناسي محاسباتي(Computatioal linguistics) يا پردازش زبان طبيعي (natural language processing) بهم تنيده شده‌اند که در آن بر روي مسئله استفاده زبان تمرکز شده است.

# تاريخچه هوش مصنوعي

پيدايش هوش مصنوعي (1943- 1956)

اشتياق زودهنگام، آرزوهاي بزرگ (1952-1969)

مقداري واقعيت (1974-1966)

سيستم‌هاي مبتني بر دانش: کليد قدرت؟ (1969-1979)

بازگشت شبکه‌هاي عصبي (1986- تاکنون)

حوادث اخير (1987- تاکنون)

# پيدايش هوش مصنوعي

* اولين کار جدي در حيطه AI، توسط وارن مک‌کلود و والتر پيتز انجام شد.
* سه منبع استفاده شده توسط آنها:
  + دانش فيزيولوژي پايه و عملکرد نرون در مغز
  + تحليل رسمي منطق گزاره‌ها متعلق به راسل و رايت هد
  + تئوري محاسبات تورينگ

در 1949 دونالد هب، قانون ساده بهنگام‌سازي براي تغيير تقويت اتصالات بين نرون‌ها را تعريف کرد که از طريق آن يادگيري ميسر مي‌گردد.

در زماني که کلود شانون و آلن تورينگ، برنامه بازي شطرنج را نوشتند ، SNARC، اولين کامپيوتر شبکه عصبي در دانشگاه پرينستون توسط مينسکي و ادموندز ساخته شد.

اين کامپيوتر، از 3 هزار تيوپ مکشي و مکانيزم خلباني خودکار اضافي که مربوط به بمب‌افکن‌هاي B24 مي‌باشد براي شبيه‌سازي شبکه 40 نروني استفاده کرد.

محققين علاقمند به تئوري آتوماتا، شبکه‌هاي عصبي و مطالعه هوش، گرد يکديگر جمع شدند و در کارگاهي در دورت موند مشغول فعاليت شدند. که در اين ميان نام هوش مصنوعي براي حيطه فعاليت آنها انتخاب شد.

# اشتياق زودهنگام، آرزوهاي بزرگ (1952-1969)

فعالان در عرصه AI:

روچستو و تيمش در IBM

هربرت جلونتر: با ساخت Geometry Theorem Prover

آرتور ساموئل: ساخت برنامه براي بازي چکر

جان مک کارتي در MIT:

تعريف زبان ليسپ (Lisp) مهمترين زبان هوش مصنوعي

مفهوم اشتراک زماني (time sharing)

نشر مقاله‌اي با عنوان "برنامه‌ها با حواس مشترک"

تشريح يک سيستم فرضي به نام Advice Taker ، که به اصول پايه بازنمايي معرفت و استدلال تجسم بخشيد؛

کار بر روي سيستم برنامه‌ريزي سؤال-جواب

کار بر روي پروژه روبات‌هاي shakey

مينسکي: کار بر روي ميکرو ورلدها و همکاري با مک‌کارتي، ولي بر سر اختلاف بر نگرش منطقي و ضدمنطقي کار تحقيقاتي خود را از هم جدا کردند.

مينسکي با گروهي از دانشجويان بر روي ميکروورلدها کار کرد که برخي از آنها عبارتند از:

جيمز اسلاگل، SAINT، قادر به حل مسائل انتگرال‌گيري فرم بسته

اوانز: ANALOGY، حل مسائل مشابهت هندسي در تست‌هاي هوش

رافائل: SIR: پاسخ به قضاياي پرسشي جملات ورودي

بابرو: STUDENT: حل مسائل داستاني جبر

# مقداري واقعيت (1966-1974)

مشکلات تقريباً تمام پروژه‌ها تحقيقي AI وقتي پديدار مي‌شدند که مسائل گسترده‌تري براي حل توسط آنها مطرح مي‌شد:

برنامه‌هاي اوليه اغلب داراي دانش محدود يا فاقد دانش در مورد موضوع کار بودند.

انجام ناپذيري بسياري از مسائل

به دليل اعمال برخي محدوديت‌هاي پايه‌اي بر روي ساختار پايه مورد استفاده براي توليد رفتار هوشمند

# سيستم‌هاي مبتني بر دانش: کليد قدرت؟ (1969-1979)

روش‌هاي ضعيف: مبتني بر يک جستجوي همه‌منظوره مي‌باشند که قدم‌هاي اوليه يادگيري را برمي‌دارند اما تلاشي در جهت يافتن راه‌حل‌هاي کامل ندارند.

به اين دليل که اطلاعات ضعيفي را در مورد دامنه فعاليت خود به کار مي‌برند.

پس براي حل مسائل دشوار، تقريباً جواب را از قبل بايد بدانيم.

برنامه DENDRAL از برنامه‌هايي است که از اين رهيافت استفاده مي‌کند.

اهميت برنامه DENDRAL در اين بود که اولين سيستم موفق با دانش غني بود، يعني تبحر سيستم بر پايه تعداد بسيار زيادي قانون ايجاد شده بود. سيستم‌هاي بعدي ايده اصلي رهيافت Advice taker مک کارتي را دنبال مي‌کردند يعني جداسازي دانش (در شکل قوانين) و مؤلفه استدلال.

# MYCIN نسبت به DENDRAL دو تفاوت عمده دارد:

برخلاف قوانين DENDRAL، هيچ مدل تئوري‌وار عمومي براي آنکه قوانين MYCIN استنتاج شود، وجود نداشت.

قوانين مي‌بايست عدم قطعيت مربوط به دانش پزشکي را منعکس مي‌کرد.

# AI به يک صنعت تبديل مي‌شود (1980-1988)

RI: اولين سيستم خبره تجاري موفق از شرکت DEC که سودآوري زيادي را براي شرکت بهمراه داشت.

پروژه «نسل پنجم»: اين پروژه ژاپني به منظور ساخت کامپيوترهاي هوشمندي که پرولوگ را به جاي کد ماشين اجرا مي‌کردند، انجام شد.

شرکت‌هاي ديگر جهان از جمله ميکروالکترونيک، MCC، ليسپ ماشين، تگزاس اينسترومنت، سمبوليکس، زيراکس و غيره در ساخت ايستگاه‌هاي کاري بهينه شده در اين عرصه فعاليت داشتند.

# بازگشت شبکه‌هاي عصبي:

دانشمندان فعال در اين عرصه:

هاپ فيلد: که به آناليز خواص ذخيره‌سازي و بهينه‌سازي شبکه‌ها پرداخت.

راسل هارت و هينتون: مطالعه مدل‌هاي شبکه عصبي را ادامه دادند.

بريسون و هو: الگوريتم يادگيري انتشار به عقب را مجدداً مطرح کردند.

# حوادث اخير:

رهيافت HMM: رهيافت غالب در سال‌هاي اخير مي‌باشد که توسط مايکف به وجود آمده است.

اين رهيافت از دو جنبه زير حائز اهميت است:

مبتني بر نظريه رياضي محض است.

طي فرايندي با يادگيري گروه عظيمي از داده گفتار واقعي خود را بهبود مي‌بخشد.

برنامه‌ريزي: در دهه 70 فقط براي ميکرووردها مناسب بودند، اکنون براي زمانبندي کار در کارخانه‌ها و مأموريت‌هاي فضايي استفاده مي‌شوند.

بيان شبکه باور: استدلال کارا را در مورد ترکيب رويدادهاي غيرمنطقي ممکن ساخت.

ايده سيستم‌هاي خبره فرماتيو توسط کار جوداپير و ارديک هوروتيز و ديويد هکرمن مطرح شد:

"سيستم‌هايي که مطابق قوانين تئوري تصميم‌گيري به طور منطقي عمل مي‌کنند و سعي ندارند که تبحر انساني را تقليد کنند."

# شرايط کنوني:

برخي از سيستم‌هايي موجود در جهان که از هوش مصنوعي استفاده مي‌کنند:

HITECH: اولين برنامه کامپيوتري که موفق به شکست استاد بزرگ شطرنج جهان، آرنولد دنکر شده است.

PEGASUS: يک برنامه درک گفتار که سؤالات کاربر را جواب مي‌دهد و تمامي برنامه‌هاي مسافرتي شخص را با يک برنامه‌ريزي درست، مقرون به صرفه مي‌کند.

MARVEL: سيستم خبره‌اي که داده‌هاي ارسالي از سفينه فضايي را تحليل نموده و در صورت بروز مشکلات جدي، پيغام هشدار به تحليلگران مي‌دهد.

فصل دوم

عامل های هوشمند

# عامل:

به هر چيزي اطلاق مي‌شود، که قادر به درک محيط پيرامون خود از طريق حس‌گرها(sensor) و اثرگذاري‌ بر روي محيط از طريق اثرکننده‌ها (effector) باشد.

# عامل نرم‌افزاري:

عامل نرم‌افزاري رشته‌هاي بيتي را به عنوان درک محيط و عمل، کدگذاري مي‌کند.

# عوامل انساني

حس کردن: گوش، چشم، ديگر ارگان‌ها

اثرگذاري: دست، پا، بيني، اندام‌هاي ديگر

# عوامل روباتيک

حس کردن: دوربين، يابنده‌هاي مادون قرمز

اثرگذاري: موتور

**environment**

**?**

**agent**

**sensors**

**effectors**

**percepts**

**actions**

# عامل‌ها چگونه بايد عمل کنند؟

عامل منطقي: چيزي است که کار درست انجام مي‌دهد.

عمل درست: آن است که باعث موفق‌ترين عامل گردد.

کارايي: چگونگي موفقيت يک عامل را تعيين مي‌کند.

# تفاوت ميان منطقي بودن و دانش کل (omniscience):

عامل داناي کل معني خروجي واقعي اعمال خود را دانسته و بر پايه آن عمل مي‌کند اما دانش کل در واقعيت غيرممکن است.

اگر معين کنيم که هر عامل هوشمند همواره بايد همان کاري را انجام دهد که در عمل مناسب است، هيچگاه نمي‌توان عاملي را طراحي نمود که اين مشخصات را مرتفع سازد.

آن چه در هر زماني منطقي است به چهار چيز وابسته است:

* معيار کارايي که درجه موفقيت را تعيين مي‌کند.
* هر چيزي که تا کنون عامل، ادراک نموده است. ما اين تاريخچه کامل ادراکي را دنباله ادراکي مي‌ناميم.
* آنچه که عامل درباره محيط خود مي‌داند.
* اعمالي که عامل مي‌تواند صورت دهد.

# رفتار عامل وابسته به دنباله ادراکي تا حال است.

عامل را بايد به‌عنوان ابزاري براي تحليل سيستم‌ها قلمداد کرد؛ نه شخصيتي مطلق که جهان را به دو بخش عامل و غير‌عامل‌ها تقسيم مي‌کند.

# نگاشت ايده‌آل از دنباله‌هاي ادراکي به عمليات

هر عامل خاصي را به وسيله جدولي توصيف مي‌کنيم، که در آن عمل آن در پاسخ به هر دنباله ادراکي قرار مي‌گيرد.

اين بدان معني نيست که ما جدول خاصي با يک ورودي براي هر دنباله ادراک ممکني توليد کنيم. مي‌توان مشخصات نگاشت را بدون شمارش خسته‌کننده آنها انجام داد.

مثال: تابع ريشه دوم

دنباله ادراکي: دنباله‌اي از کليدهاي زده شده

نگاشت ايده‌آل:براي مقادير مثبت x نشان داده شده توسط ادراک، z نيز مثبت باشد و عمل مناسب نمايش نشان داده شود.

# خودمختاري:

در اينجا تعريف عامل بايد کامل‌تر شود و بخش دانش دروني به آن اضافه مي‌گردد.

رفتار عامل مي‌تواند متکي بر دو پايه تجربه خود و دانش دروني بنا نهاده شود.

اين رفتار، در ساخت عامل براي شرايط محيطي خاص که در آن عمل خواهد کرد، استفاده مي‌شود.

سيستم به وسعتي خود مختار است که رفتار آن بر اساس تجربه خودش تعيين مي‌کند. زماني که عامل فاقد تجربه و يا کم تجربه‌ است، مسلماً تصادفي عمل خواهد کرد، مگر آنکه طرح‌ کمک‌هايي به آن داده باشد.

عامل هوشمند واقعاً خود مختار بايد قادر به عمل موفقيت‌آميز در دامنه وسيعي از محيط‌ها باشد و البته بايد زمان کافي براي تطبيق نيز به آن داده شود.

# ساختار عامل‌هاي هوشمند

وظيفه هوش مصنوعي طراحي برنامه عامل است؛

اين طراحي شامل تابعي است که نگاشت عامل از ادراک به عمليات را پياده سازي مي‌کند.

معماري: فرض مي‌کنيم برنامه عامل بر روي نوعي ابزار محاسبه‌گر اجرا مي‌گردد که آن را معماري مي‌ناميم.

برنامه‌ عامل، بايد توسط معماري قابل پذيرش و اجرا باشد.

عموماً، معماري ادراک از طريق حس‌گرها را براي برنامه آماده ساخته، برنامه را اجرا نموده و اعمال انتخابي برنامه را به عمل‌کننده‌هاي سيستم منتقل مي‌کند.

ارتباط بين عامل‌ها، معماري‌ها و برنامه‌ها را مي‌توان به صورت ذيل جمع بندي نمود:

برنامه+ معماري= عامل

در اينجا مسئله تمايز بين محيط واقعي و مصنوعي مطرح مي‌شود؛ اما

**مسأله اصلي، پيچيدگي مابين:**

ارتباط رفتار عامل، دنباله ادراکي توليد شده بوسيله محيط، و اهدافي که عامل قصد حصول آن را دارد، است.

مشهور‌ترين محيط مصنوعي، محيط تست تورينگ (turing) است.

# برنامه‌هاي عامل:

**تشابهات عامل‌هاي هوشمند:**

دريافت ادراک محيطي

توليد اعمال لازم

**دو نکته در مورد شالوده برنامه قابل ذکر هستند:**

برنامه عامل تنها يک درک از شرايط محيطي واحد را به عنوان ورودي دريافت مي‌کند.

هدف يا معيار کارايي بخشي از برنامه شالوده نخواهد بود.

**چرا تنها به پاسخ‌ها نگاه نمي‌کنيم؟**

جدول مراجعه بايد بر پايه حفظ کامل دنباله ادراکي در حافظه عمل نموده و از آن براي ايندکس‌سازي داخل جدول استفاده کند.

جدول عامل نوع راننده تاکسي

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **محيط** | **اهداف** | **عمليات** | **ادراکات** | **نوع عامل** |
| جاده، پياده‌رو، ترافيک، مشتري | ايمني، سرعت، قانونمندي، راحتي، افزايش سودمندي | راهنمايي کردن، شتاب‌دهنده، ترمز، صحبت با مسافر | دوربين‌ها، سرعت سنج، GPS، Sonar ميکروفون | راننده تاکسي |

جنبه‌هاي مختلف يک عمل، انواع مختلف برنامه‌هاي عامل را پيشنهاد خواهد کرد.

براي مثال، 4 عامل را مورد بررسي قرار مي دهيم:

* عامل‌هاي واکنشي ساده
* عامل‌هايي که اثرات دنيا را حفظ مي‌کنند
* عامل‌هاي هدف‌گرا
* عامل‌هاي سودمند

# عامل‌هاي واکنشي ساده

در اينجا جدول رجوع بايد مورد توجه قرار گرفته و فيلدهاي مختلف آن توسط اطلاعات ورودي پر شود.

اتصالاتي (واکنش‌هايي) وجود دارند که انسان‌ها بسياري از آنها را دارا بوده:

برخي از آنها قابل يادگيري و برخي ديگر غريزي است.

**دياگرام شماتيک از عامل ساده واکنشي**

**Environment**

What the world is like now

What action I should do now

**Condition-action rules**

**Agent**

Sensors

Effectors

مربع مستطيل: نشان‌دهنده وضعيت داخلي جاري فرايند تصميم‌گيري عامل

بيضي: نشان‌دهنده وضعيت اطلاعات پس‌زمينه

# عامل‌هايي که اثرات دنيا را حفظ مي‌کنند

از آنجايي ناشي مي‌شود که حسگرها نمي‌توانند دسترسي کامل به وضعيت دنيا را به وجود آورند.

در چنين شرايطي، عامل ممکن است نيازمند دستکاري برخي اطلاعات وضعيت داخلي باشد تا از طريق آن تمايز بين وضعيت‌هاي دنيا که در ظاهر ورودي ادراکي يکساني مي‌کنند ولي در واقع معني کاملاً متفاوتي دارند را ميسر سازد.

**بهنگام‌سازي اطلاعات وضعيت داخلي همزمان** با گذر زمان نيازمند دو نوع دانش کد شده در برنامه عامل است.

اول: نيازمند آنيم که برخي اطلاعات درباره چگونگي تغيير جهان مستقل از عامل را داشته باشيم.

دوم: نيازمند اطلاعات درباره اعمال خود هستيم که بر روي دنيا اثرگذار است.

**عامل واکنشي با حالت داخلي**

**Environment**

What the world is like now

What action I should do now

**Agent**

**sensors**

**Effectors**

**What my action do**

**How the world evolves**

**State**

**Condition-action rules**

# عامل‌هاي هدف گرا:

دانستن درباره وضعيت کنوني محيط همواره براي تصميم‌گيري عمل نمي‌تواند کافي باشد.

به همان گونه که عامل نيازمند شرح وضعيت جاري است، به نوعي نيازمند اطلاعات هدف(goal) مي‌باشد که توضيح موقعيت مطلوب است.

برنامه‌ عامل مي‌تواند اين اطلاعات را با اطلاعاتي درباره نتايج اعمال ممکن (همانند اطلاعاتي که در عامل واکنش براي بهنگام‌سازي وضعيت داخلي استفاده شد) ترکيب نموده تا اعمال مناسب را براي دسترسي به هدف انتخاب نمايد.

در مواقعي ساده است: که رضايت از هدف بلافاصله از عمل واحد توليد گردد.

در مواقعي پيچيده است: که عامل بايد دنباله‌هاي طولاني را در نظرگرفته تا راهي براي دستيابي به هدف پيدا کند.

در مواقع پيچيده، جستجو و برنامه‌ريزي به يافتن دنباله اعمال منجر خواهند شد.

# تفاوت عامل‌هاي واکنشي و هدف‌گرا:

در طراحي عامل‌هاي واکنشي طراح براي حالات متفاوت عملي درست را پيش محاسبه مي‌کند. در عامل‌هاي هدف‌گرا، عامل مي‌تواند دانش خود را در مورد چگونگي واکنش بهنگام سازد.

1. براي عامل واکنشي ما مجبور به دوباره نويسي تعداد زيادي قوانين شرط –عمل خواهيم بود.
2. عامل هدف‌گرا نسبت به رسيدن به مقاصد متفاوت انعطاف پذير است.
3. بسادگي با تعيين يک هدف تازه، مي‌توانيم عامل هدف‌گرا را به رفتار تازه برسانيم.

**عاملي با اهداف دقيق**

What it will be like if I do action A

**Environment**

What the world is like now

What action I should do now

**Agent**

**sensors**

**Effectors**

**Goals**

**What my action do**

**How the world evolves**

**State**

# عامل‌هاي سودمند:

اهداف به تنهايي براي توليد رفتار با کيفيت بالا کافي نيستند.

ملاک کارايي عومي بايد مقايسه‌اي بين وضعيت‌هاي دنياي متفاوت (يا دنباله‌ حالات) را بر پايه چگونگي رضايت عامل در صورت حصول هدف بدهد.

بنابراين اگر يک وضعيت دنيا به ديگري ترجيح داده مي‌شود، آنگاه آن براي عامل سودمند‌تر خواهد بود

سودمندي: تابعي است که يک وضعيت را به عدد حقيقي نگاشت مي‌دهد، که درجه رضايت مربوط را تشريح مي‌کند.

مشخصات کامل تابع سودمندي امکان تصميم‌گيري منطقي را براي دو نوع حالتي که هدف مشکل دارد، اجازه مي‌دهد.

1. زماني که اهداف متناقص وجود دارند.
2. زماني که چندين هدف دارند که عامل مي‌تواند آنها را هدف قرار دهد و هيچکدام از آنها با قطعيت قابل حصول نيست.

ارتباط بين عامل و محيط: اعمال بوسيله عامل بر محيط انجام مي‌شود، که خود ادراک عامل را مهيا مي‌سازد.

**خواص محيط:**

* قابل دسترسي در مقابل غير دسترسي
* قطعي در برابر غير قطعي
* اپيزوديک در مقابل غيراپيزوديک
* ايستا در مقابل پويا
* گسسته در مقابل پيوسته

**قابل دسترسي در مقابل غيرقابل دسترسي**

محيط قابل دسترسي: محيطي که عامل آن توسط ابزار حس‌کننده‌اش امکان دسترسي به وضعيت کامل محيط را داشته باشد.

محيط قابل دسترسي راحت است، زيرا عامل نيازمند دستکاري هيچ وضعيت داخلي براي حفظ دنيا را نخواهد داشت.

**قطعي در مقابل غير قطعي**

محيط قطعي: محيطي است که اگر وضعيت بعدي محيط بوسيله وضعيت کنوني و اعمالي که با عامل‌ها انتخاب گردد، تعيين شود.

بهتر است به قطعي يا غير قطعي بودن محيط از ديدگاه عامل نگاه کنيم.

**اپيزوديک در مقابل غير اپيزوديک**

* محيط اپيزوديک (episodic)، تجربه عامل به اپيزودهايي تقسيم مي‌گردد.
* هر اپيزود شامل درک و عمل عامل است.
* کيفيت اعمال آن تنها به خود اپيزود وابسته است.
* محيط‌هاي اپيزودي بسيار ساده‌ترند زيرا عامل نبايد به جلوتر فکر کند.

**ايستا در مقابل پويا**

محيط پويا: محيطي که در حين سنجيدن عامل تغيير مي‌کند.

محيط نيمه‌پويا: محيطي که با گذر زمان تغيير نمي‌کند اما امتياز کارايي تغيير مي‌کند.

محيط‌هاي ايستا براي کار ساده هستند زيرا عامل نياز به نگاه‌کردن به دنيا در حين تصميم‌گيري عملي نداشته و همچنين در مورد گذر زمان نيز نگران نمي‌باشد.

**گسسته در مقابل پيوسته**

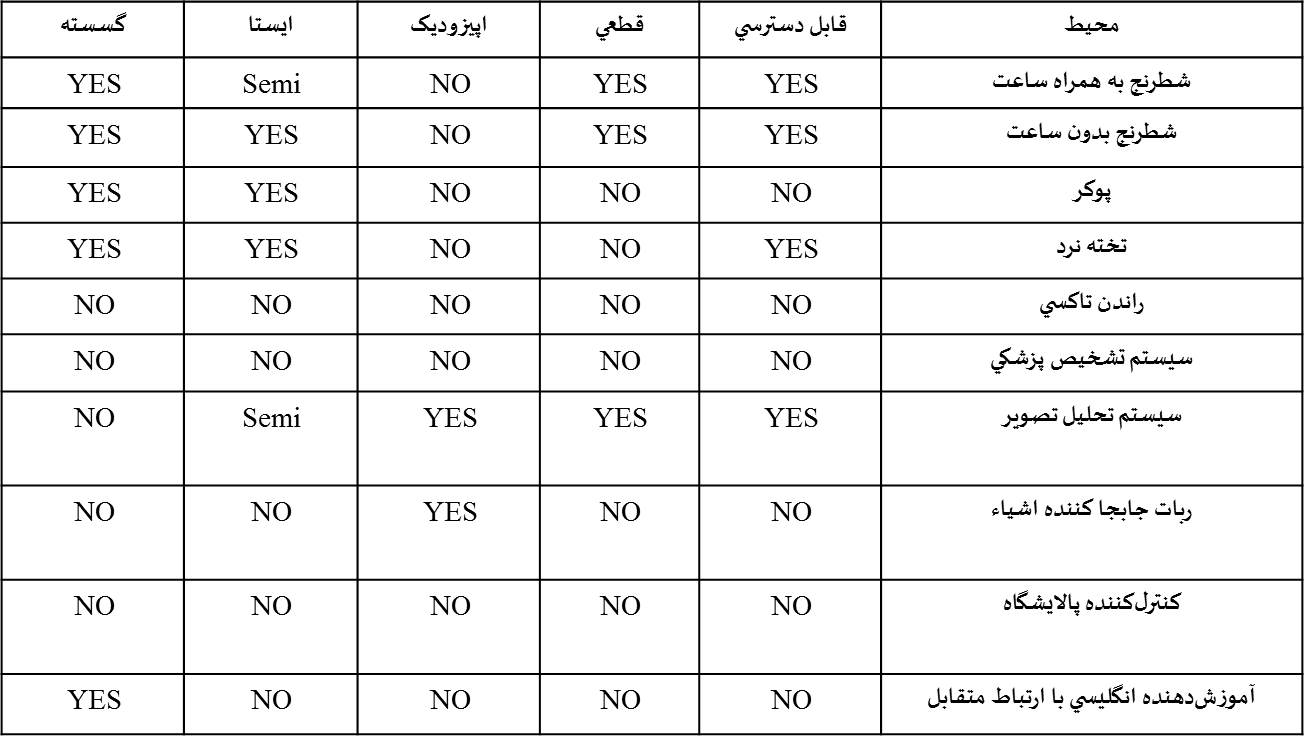
محيط گسسته: اگر تعداد محدود و مجزا از ادراک و اعمال بوضوح تعريف شده باشد.

- بازي شطرنج گسسته است.

- رانندگي تاکسي پيوسته است.

سخت‌ترين حالت در بين حالات موجود براي محيط:

غير قابل دسترسي، غير اپيزوديک، پويا و پيوسته

**مثال‌هايي از انواع محيط و ويژگي‌هاي آنها**

# برنامه‌هاي محيط

شبيه‌ساز يک يا چند عامل را به عنوان ورودي گرفته و بگونه‌اي عمل مي‌کند که هر عامل ادراک درست و نتيجه بازگشتي عمل خود را بدست آورد.

شبيه‌ساز محيط را بر اساس اعمال و ديگر فرآيند‌هاي پوياي محيط بهنگام مي‌سازد.

محيط با وضعيت آغازين و تابع بهنگام‌سازي تعريف مي‌گردد.

فصل سوم

حل مسائل توسط جستجو

يک نوع عامل هدفگرا، عامل حل مسئله ناميده مي‌شود.

عامل‌هاي حل مسئله توسط يافتن ترتيب عمليات تصميم مي‌گيرند که چه انجام دهند تا آنها را به حالت‌هاي مطلوب سوق دهد.

**عامل‌هاي حل مسئله**

عامل‌هاي هوشمند به طريقي عمل مي‌کنند که محيط مستقيماً به داخل دنباله حالت‌هايي وارد شود که معيار کارآرايي را افزايش مي‌دهند.

عمليات به گونه‌اي ساده‌سازي مي‌شوند که عامل قادر باشد تا هدفي را قبول کرده و به آن برسد.

الگوريتم جستجو مسئله‌اي را به عنوان ورودي دريافت نموده و راه‌حلي را به صورت دنباله عمليات بر مي‌‌گرداند.

فاز اجرايي: مرحله‌اي است که در آن زمان، راه‌حلي‌ پيدا مي‌شود و عمليات پيشنهادي مي‌توانند انجام شوند.

به طور ساده براي طرح يک عامل مراحل «فرموله‌سازي، جستجو، اجرا» را در نظر مي‌گيريم.

پس از فرموله‌سازي يک هدف و يک مسئله براي حل عامل،

1. رويه جستجويي را براي حل آن مسئله فراخواني مي‌کند.
2. از راه حل براي راهنمايي عملياتش استفاده مي‌کند و هرآنچه که راه حل پيشنهاد مي‌کند را انجام مي‌دهد.
3. آن مرحله را از دنباله حذف مي‌کند.
4. زماني که راه‌حل اجرا شد، عامل هدف جديدي را پيدا مي‌کند.

**چهار نوع اساسي از مسائل وجود دارند:**

* مسائل تک حالته (Single-state)
* مسائل چند حالته (Multiple-state)
* مسائل احتمالي (Contingency)
* مسائل اکتشافي (Exploration)

# دانش و انواع مسئله دنياي مکش (جاروبرقي):

اگر دنيا حاوي دو محل باشد:

هر محل ممکن است که شامل خاک باشد و يا نباشد و عامل ممکن است که در يک محل يا ديگر محل‌ها باشد؛ که داراي هشت حالت متفاوت خواهد بود.

هدف تميز کردن تمام خاک‌هاست که در اينجا معادل با مجموعه حالت‌ {8و 7} است.

# مدل‌هاي مختلف براي مسئله جاروبرقي:

**- مدل تک حالته:**

حس‌گرهاي عامل به آن اطلاعات کافي مي‌دهند تا وضعيت دقيق مشخص شود. (دنيا قابل دسترسي است). عامل مي‌تواند محاسبه کند که کدام وضعيت پس از هر دنباله از عمليات قرار خواهد گرفت.

**- مدل چند حالته:**

عامل تمام اثرهاي عملياتش را مي‌داند اما دسترسي به حالت دنيا را محدود کرده است.

زماني که دنيا تماماً قابل دسترسي نيست عامل بايد در مورد مجموعه حالت‌هايي که ممکن است به آن برسد استدلال کند.

**- مدل احتمالي:**

با اين مدل حل مسئله، حس‌گرهايي را در طول فاز اجرايي نياز داريم. عامل اکنون بايد تمام درخت عملياتي را بر خلاف دنباله عملياتي منفرد، محاسبه کند. که به طور کلي هر شاخه درخت، با يک امکان احتمالي که از آن ناشي مي‌شود، بررسي مي‌شود.

**مدل اکتشافي:**

عاملي که هيچ اطلاعاتي در مورد اثرات عملياتش ندارد.

در اين حالت، عامل بايد تجربه کند و به تدريج کشف کند که چه عملياتي بايد انجام شود و چه وضعيت‌هايي وجود دارند. اين روش يک نوع جستجو است.

اگر عامل نجات يابد، «نقشه‌اي» از محيط را ياد مي‌گيرد که مي‌تواند مسائل بعدي را حل کند.

*مسائل و راه‌حل‌هاي خوب تعريف شده*

مسئله: در واقع مجموعه‌اي از اطلاعات است که عامل از آنها براي تصميم‌گيري در مورد اينکه چه کاري انجام دهد، استفاده مي‌کند.

عناصر اوليه تعريف يک مسئله، وضعيتها عمليات هستند.

**براي تعريف يک مسئله موارد زير نياز داريم:**

* وضعيت آغازين (initial state) که عامل خودش از بودن در آن آگاه است.
* مجموعه‌اي از عمليات ممکن، که براي عامل قابل دسترسي باشد.
* آزمون هدف (goal test)، که عامل مي‌تواند در يک تعريف وضعيت منفرد آن را تقاضا کند تا تعيين گردد که آن حالت، وضعيت هدف است يا خير.
* تابع هزينه مسير، تابعي است که براي هر مسير، هزينه‌اي را در نظر مي‌گيرد؛ و با حرف g مشخص مي‌شود.

هزينه يک سفر= مجموع هزينه‌هاي عمليات اختصاصي در طول مسير

**براي حل مسئله چند حالته، فقط به يک اصلاح جزئي نياز داريم:**

يک مسئله شامل:

* يک مجموعه حالت اوليه
* مجموعه‌اي از عملگرهاي ويژه براي هر عمل به گونه‌اي که از هر وضعيت داده شده مجموعه‌اي حالات رسيده شده و يک آزمون هدف و تابع هزينه مسير را معين کند.

**يک عملگر:**توسط اجتماع نتايج اعمال عملگر در هر وضعيت مجموعه، به کار برده مي‌شود.

**يک مسير:** مجموعه حالات را مرتبط مي‌کند.

**يک راه حل:** مسيري است که به مجموعه‌اي از حالات که تمام آنها، وضعيت هدف هستند، سوق مي‌دهند.

**اندازه‌گيري کارايي حل مسئله:**

*کارايي يک جستجو، حداقل از سه طريق مي‌تواند اندازه‌گيري شود:*

* آيا اين جستجو راه حلي پيدا مي‌کند؟
* آيا راه حلي مناسبي است؟
* هزينه جستجو از نظر زماني و حافظه مورد نياز براي يافتن راه حل چيست؟

مجموع هزينه جستجو= هزينه مسير + هزينه جستجو

عامل بايد تصميم بگيرد که چه منابعي را فداي جستجو و چه منابعي را صرف اجرا کند.

**انتخاب حالات و عمليات**

هنر واقعي حل مسئله، تصميم‌گيري در مورد اين است که چه چيزهايي در تعريف حالات و عملگرها بايد به حساب آورده شوند و چه چيزهايي بايد کنار گذاشته شوند.

**انتزاع:**

فرآيند حذف جزئيات از يک بارنمايي انتزاع (abstraction) ناميده مي‌شود.

همانگونه که تعريف را خلاصه مي‌کنيم مي‌بايست عمليات را نيز خلاصه نمائيم.

انتزاع به اين دليل مفيد است، که انجام هر کدام از عمليات آسانتر از مسئله اصلي است.

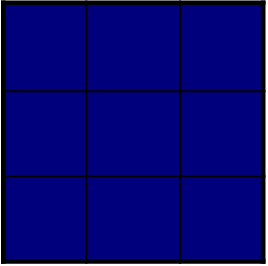
انتخاب يک انتزاع خوب از اين رو شامل حذف تا حد ممکن مي‌شود تا زماني که عمليات خلاصه شده براي انجام آسان باشند.

**مسائل نمونه: مسائل اسباب‌بازي**

# معماي 8:

معماي 8 نمونه‌اي است شامل يک صفحة 3\*3 با 8 مربع شماره دار در يک صفحه خالي.

هر مربع که مجاور خانه خالي است. مي‌تواند به درون آن خانه برود. هدف رسيدن به ساختاري است که در سمت راست شکل نشان داده شده است. نکته مهم اين است که بجاي اينکه بگوييم «مربع شماره 4 را به داخل فضاي خالي حرکت بده» بهتر است بگوييم «فضاي خالي جايش را با مربع سمت چپش عوض کند.»



**5**

**4**

**1**

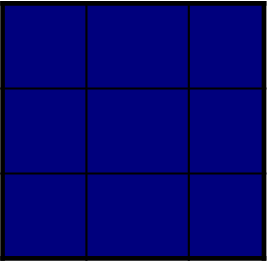
**6**

**8**

**7**

**3**

**2**



**1**

**2**

**3**

**8**

**4**

**7**

**6**

**5**

**Start State**

**Goal State**

**حالت‌ها:** توصيف وضعيت مکان هر 8 مربع را در يکي از 6 خانة صفحه مشخص مي‌کند. براي کارايي بيشتر، بهتر است که فضاهاي خالي نيز ذکر شود.

**عملگر‌ها:** فضاي خالي به چپ، راست، بالا و پائين حرکت کند.

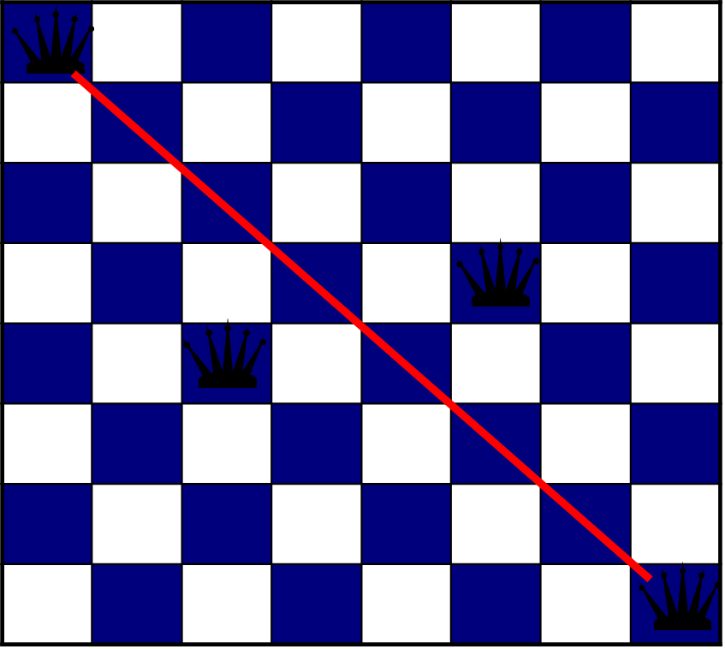
**آزمون هدف:** وضعيت با ساختار هدف مطابقت مي‌کند.

**هزينه مسير:** هر قدم ارزش 1 دارد، بنابراين هزينه مسير همان طول مسير است.

# مسئله 8 وزير:

هدف از مسئله 8 وزير، قرار دادن 8 وزير بر روي صفحه شطرنج به صورتي است که هيچ وزيري نتواند به ديگري حمله کند.

دو نوع بيان رياضي اصلي وجود دارد بيان افزايشي که با جايگزيني وزيرها، به صورت يکي يکي کار مي‌کند و ديگري بيان وضعيت کامل که با تمام 8 وزير روي صفحه شروع مي‌کند و آنها را حرکت مي‌دهد.

در اين فرمول ما 64 امکان داريم.

بنابراين ما تست هدف و هزينه مسير را به صورت زير خواهيم داشت:

**آزمون هدف:** 8 وزير روي صفحه، که با هم برخورد ندارند.

**هزينه مسير:** صفر.

**حالات:** ترتيب از صفر تا 8 وزير بدون هيچ برخورد.

**عملگرها:** يک وزير را در خالي‌ترين ستون سمت چپ جايگزين کنيد که هيچ برخوردي با بقيه نداشته باشد.

**Cryptarithmetic :**

در مسائل کريپتاريتمتيک، حروف به جاي ارقام مي‌نشينند و هدف يافتن جايگزيني از اعداد براي حروف است که مجموع نتيجه از نظر رياضي درست باشد. معمولاً هر حرف بايد به جاي يک رقم مختلف بنشينند.

مثال:

**FORTY**

**+ TEN**

**+ TEN**

**----------**

**SIXTY**

**29786**

**+ 850**

**+ 850**

**----------**

**31486**

**F=2, O=9, R=7, etc.**

# يک فرمول ساده:

حالات: يک معماي Cryptarithmetic با چند حروف جايگزين شده توسط ارقام.

عملگرها: وقوع يک حروف را با يک رقم جايگزين کنيد که قبلاً در معما ظاهر نشده باشد.

آزمون هدف: معما فقط شامل ارقام است و يک مجموع صحيح را بر مي‌گرداند.

هزينه مسير: صفر- تمام راه حل‌هاي صحيح است.

*مي‌خواهيم که از تبديل جايگزيني‌هاي مشابه اجتناب کنيم:*

* قبول يک ترتيب ثابت مانند ترتيب الفبايي.
* هر کدام که بيشترين محدوديت جايگزيني را دارد، انتخاب کنيم؛ يعني حرفي که کمترين امکان مجاز را دارند، محدوديت‌هاي معما را مي‌دهد.

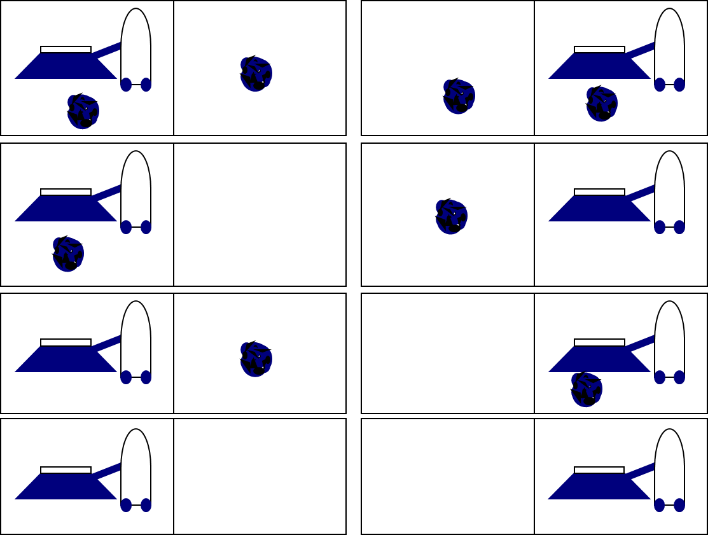
# دنياي مکش:

مسئله تک حالته: عامل از جاي خودش اطلاع دارد و تمام مکان‌هاي آلوده را مي‌شناسد و دستگاه مکنده ما درست کار مي‌کند.

حالات: يکي از 8 حالت نشان داده شده.

عملگرها: حرکت به چپ، حرکت به راست، عمل مکش.

آزمون هدف: هيچ خاکي در چهار گوش‌ها نباشد.

هزينه مسير: هر عمل ارزش 1 دارد.

**مسئله چند حالته:** عامل داراي حسگر نمي‌باشد.

**مجموعه وضعيت‌ها :** زير مجموعه‌اي از حالات.

**عملگرها:** حرکت به چپ، حرکت به راست، عمل مکش.

**آزمون هدف:** تمام حالات در مجموعه حالت‌ها فاقد خاک باشند.

**هزينه مسير:** هر عمل هزينه 1 دارد.

# مسئله کشيش‌ها و آدمخوارها:

سه کشيش و سه آدم خوار در يک طرف رودخانه قرار دارند و هم چنين قايقي که قادر است يک يا دو نفر را حمل کند. راهي را بيابيد که هر نفر به سمت ديگر رودخانه برود، بدون آنکه تعداد کشيش‌ها در يکجا کمتر از آدم خوارها شود.

حالات: يک حالت شامل يک دنبالة مرتب شده از عدد است که تعداد کشيش‌ها، تعداد آدمخوارها و محل قايق در ساحلي از رودخانه که از آنجا مسئله شروع شده را نمايش مي‌دهد.

عملگرها: از هر حالت، عملگرهاي ممکن يک کشيش، يک آدمخوار، دو کشيش، دو آدمخوار، يا يکي از هر کدام را در قايق جا مي‌دهند.

آزمون هدف: رسيدن به حالت(0و 0 و 0).

هزينه مسير: تعداد دفعات عبور از رودخانه.

# مسائل دنياي واقعي

**مسيريابي:**

الگوريتم‌هاي مسير يابي کاربردهاي زيادي دراند، مانند مسيريابي در شبکه‌هاي کامپيوتري، سيستم‌هاي خودکار مسافرتي و سيستم‌هاي برنامه‌نويسي مسافرتي هوايي.

**مسائل فروشنده دوره گرد و تور :**

مسئله فروشنده دوره گرد مسئله مشهوري است که در آن هر شهر حداقل يکبار بايد ملاقات شود هدف يافتن کوتاهترين مسير است.

علاوه بر مکان عامل، هر حالت بايد مجموعه شهرهايي را که عامل ملاقات کرده، نگه دارد.

علاوه بر برنامه‌ريزي صفر براي فروشنده دوره‌گرد، اين الگوريتم‌ها براي اعمالي نظير برنامه‌ريزي حرکات مته خوردکار سوراخ‌کننده برد مدار استفاده مي‌شود.

**طرح VISI :**

ابزار طراحي کمکي کامپيوتري در هر فازي از پردازش استفاده مي‌شود دو وظيفه بسيار مشکل عبارتند از:

Channel routing

Cell layout

که بعد از اينکه ارتباطات و اتصالات مدار کامل شد، اين دو قسمت انجام مي‌شوند.

هدف طراحي مداري روي تراشه است که کمترين مساحت و طول اتصالات و بيشترين سرعت را داشته باشد.

هدف قرار دادن سلول‌ها روي تراشه به گونه‌اي است که آنها روي هم قرار نگيرند و بنابراين فضايي نيز براي سيم‌هاي ارتباطي وجود دارد که بايد بين سلول‌ها قرار گيرند.

کانال‌يابي، مسير ويژه‌اي را براي هر سيم که از فواصل بين سلول‌ها استفاده مي‌کند، پيدا مي‌کند.

**هدايت ربات:**

يک ربات مي‌تواند در يک فضاي پيوسته با يک مجموعه نامحدودي از حالات و عمليات ممکن حرکت کند.

ربات‌هاي واقعي بايد قابليت تصحيح اشتباهات را در خواندن حسگرها و کنترل موتور داشته باشند.

# خط توليد خودکار:

در مسائل سرهم‌بندي، مشکل يافتن قانوني است که تکه‌هاي چند شيئي را جمع کند. اگر ترتيب نادرست انتخاب شود، راهي نيست که بتوان قسمت‌هاي بعدي را بدون از نو انجام دادن قسمت‌هاي قبلي، اضافه کرد.

کنترل يک مرحله در دنباله، يک مسئله جستجوي پيچيدة هندسي است که ارتباط نزديکي با هدايت ربات دارد. از اين رو توليد مابعدهاي مجاز گران‌ترين قسمت دنباله سرهم‌بندي است و استفاده از الگوريتم‌هاي آگاهانه براي کاهش جستجو، ضروري است.

**جستجو براي راه‌حل:**

نگهداري و گسترش يک مجموعه از دنباله‌هاي راه حل ناتمام.

جستجوي حالت‌هاي موجود و يافتن راه‌حل بنا بر اصل جستجو.

**توليد دنباله‌هاي عمل:**

**فرايند گسترش حالت:** فرايندي که از طريق توليد مجموعه جديدي از حالات، عملگرها در حالت جاري را به کار گرفته، و نتيجتاً حالت هدف را در مجموعه وارد مي‌کند.

**اصل جستجو:** انتخاب يک حالت و کنار گذاشتن بقيه براي بعد، زماني که اولين انتخاب به حل مسئله منجر نشود.

**ريشه درخت جستجو:** يک گره جستجو است که با حالت اوليه مطابقت دارد.

**گره‌هاي برگي درخت:** حالاتي هستند که داراي فرزندي در درخت نيستند.

**ساختارهاي داده براي درخت‌هاي جستجو:**

گره به عنوان يک ساختار داده با پنج قسمت به شرح زير است:

* وضعيتي که گره در فضاي حالات دارا مي‌باشند.
* گره‌اي که در جستجوي درخت، گره جديدي را توليد کرده است (گره والد).
* عملگري که براي توليد گره به کار رفته است.
* تعداد گره‌هاي مسير، از ريشه تا گره موردنظر (عمق گره).
* هزينه مسير، از حالت اوليه تا گره.

تفاوت بين گره‌ها و حالت‌ها:

گره‌ها عمق و والد دارند؛ در صورتي که حالت‌ها شامل چنين چيزهايي نيستند.

**استراتژي جستجو:**

استراتژي‌ها بايد داراي 4 معيار زير باشند:

* کامل بودن
* پيچيدگي زماني
* پيچيدگي فضا
* بهينگي

ما 6 استراتژي را بررسي خواهيم کرد:

* جستجوي سطحي
* جستجوي با هزينه يکسان
* جستجوي عمقي
* جستجوي عمقي محدود شده
* جستجوي عميق‌کننده تکراري
* جستجوي دوطرفه
* **جستجوي سطحي:**
* در اين استراتژي که بسيار سيستماتيک است، ابتدا گره ريشه، و سپس تمام گره‌هاي ديگر گسترش داده مي‌شوند.
* به عبارت کلي‌تر، تمام گره‌هاي عميق d، قبل از گره‌هاي عميق d+1 گسترش داده مي‌شوند.
* **مزايا:**
* جستجوي سطحي، کامل و بهينه مي‌باشد زيرا هزينه مسير، يک تابع کاهش‌نيابنده از عمق گره است.
* **معايب:**
* مرتبه زماني O(bd) مي باشد که نمايي است.
* نياز به حافظه زياد.

**جستجوي با هزينه يکسان:**

در اين استراتژي، در شرايط عمومي، اولين راه حل، ارزان‌ترين راه نيز هست.

اگر هزينه مسير توسط تابع g(n) اندازه‌گيري شود، در اين صورت جستجوي سطحي همان جستجوي با هزينه يکسان است با:

g(n)=DEPTH(n)

**جستجوي عمقي:**

اين استراتژي، يکي از گره‌ها را در پائين‌ترين سطح درخت بسط مي‌دهد؛ اما اگر به نتيجه نرسيد، به سراغ گره‌هايي در سطوح کم عميق‌تر مي‌رود.

**مزايا:**

اين جستجو، نياز به حافظه نسبتاً کمي فقط براي ذخيره مسير واحدي از ريشه به يک گره برگي، و گره‌هاي باقي‌مانده بسط داده نشده دارد.

پيچيدگي زماني O(bm) مي‌باشد. به طوريکه b فاکتور انشعاب فضاي حالت، و m حداکثر عمق درخت باشد.

**معايب:**

اگر مسيري را اشتباه طي کند، هنگام پائين رفتن گير مي‌کند.

جستجوي عمقي نه کامل و نه بهينه است.

در درخت‌هاي با عمق نامحدود و بزرگ اين استراتژي کار نمي‌کند.

**جستجوي عمقي محدود شده:**

اين استراتژي، براي رهايي از دامي که جستجوي عمقي در آن گرفتار مي‌شد، از يک برش استفاده مي‌کند.

جستجوي عمقي محدود شده کامل است اما بهينه نيست.

زمان و پيچيدگي فضاي جستجوي عمقي محدودشده، مشابه جستجوي عمقي است. اين جستجو پيچيدگي زماني O(bL) و فضاي O(bL) را خواهد داشت، که L محدودة عمق است.

در يک درخت جستجوي نمايي، تقريباً تمام گره‌ها در سطح پائين هستند، بنابراين موردي ندارد که سطوح بالايي چندين مرتبه بسط داده شوند. تعداد بسط‌ها در يک جستجوي عمقي محدود شده با عمق d و فاکتور انشعاب b به قرار زير است:

1+b+b2+…+bd-2+bd-1+bd

**جستجوي عميق‌کننده تکراري:**

قسمت دشوار جستجوي عمقي محدود شده، انتخاب يک محدودة خوب است.

اگر محدودة عمق بهتري را پيدا کنيم، اين محدوده، ما را به سوي جستجوي کاراتري سوق مي‌دهد. اما براي بيشتر مسائل، محدودة عمقي مناسب را تا زماني که مسئله حل نشده است، نمي‌شناسيم.

جستجوي عميق‌کنندة تکراري استراتژي است که نظريه انتخاب بهترين محدودة عمقي، توسط امتحان تمام محدودة مسيرهاي ممکن را يادآوري مي‌کند.

**مزايا:**

ترکيبي از مزاياي جستجوي سطحي و عمقي را دارد.

اين جستجو مانند جستجوي سطحي کامل و بهينه است، اما فقط مزيت درخواست حافظه اندک را از جستجوي عمقي دارد.

مرتبه بسط حالات مشابه جستجوي سطحي است، به جز اينکه بعضي حالات چند بار بسط داده مي‌شوند.

در جستجوي عميق‌کننده تکراري، گره‌هاي سطوح پائيني يک بار بسط داده مي‌شوند، آنهايي که يک سطح بالاتر قرار دارند دوبار بسط داده مي‌شوند و الي‌آخر تا به ريشه درخت جستجو برسد، که d+1 بار بسط داده مي‌شوند.

**بنابراين مجموع دفعات بسط در اين جستجو عبارتست از:**

(d+1)1+(d)b+(d-1)b2+…+3bd-2+2bd-1+1bd

پيچيدگي زماني اين جستجو هنوز O(bd) است، و پيچيدگي فضا O(bd) است.

در حالت کلي، عميق‌کننده تکراري، روش جستجوي برتري است؛ زماني که فضاي جستجوي بزرگي وجود دارد و عمق راه حل نيز مجهول است.

**جستجوي دوطرفه:**

ايده جستجوي دوطرفه در واقع شبيه‌سازي جستجويي به سمت جلو از حالت اوليه و به سمت عقب از هدف است و زماني که اين دو جستجو به هم برسند، متوقف مي‌شود.

براي پياده‌سازي الگوريتم سؤالات زير بايد پاسخ داده شوند:

1- سؤال اصلي اين است که، جستجو از سمت هدف به چه معني است؟ ماقبل‌هاي (predeccessors) يک گره n را گره‌هايي درنظر مي‌گيريم که n مابعد (successor) آنها باشد. جستجو به سمت عقب بدين معناست که توليد ماقبل‌ها از گرة هدف آغاز شود.

2- زماني که تمام عملگرها، قابل وارونه‌شدن باشند، مجموعه ماقبل‌ها و مابعدها يکسان هستند.

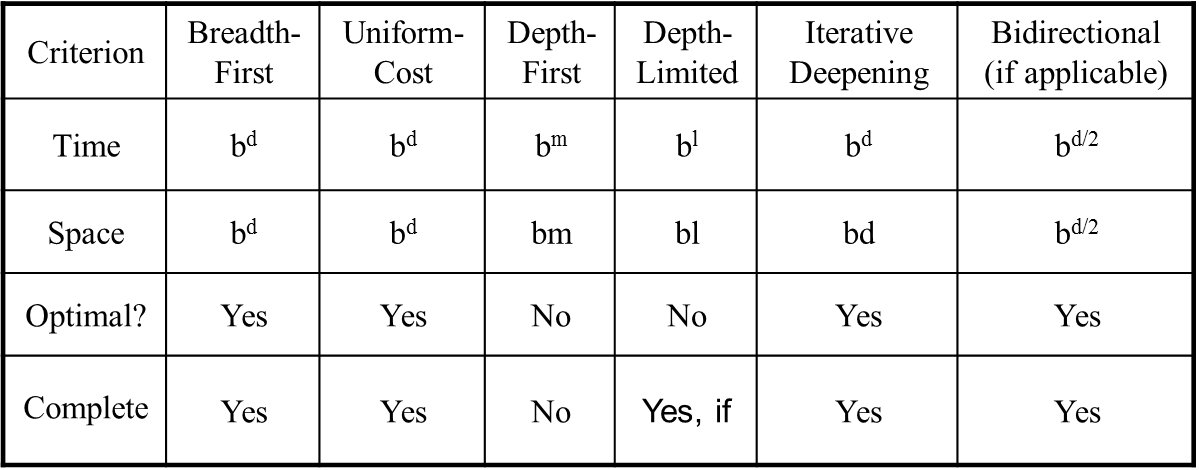
3- چه کار مي‌توان کرد زماني که هدف‌هاي متفاوتي وجود داشته باشد؟ اگر ليست صريحي از حالت‌هاي هدف وجود داشته باشد، مي‌توانيم يک تابع ماقبل براي مجموعه حالت تقاضا کنيم در حاليکه تابع مابعد يا (جانشين) در جستجوي مسائل چندوضعيته به کار مي‌رود.

4- بايد يک راه موثر براي کنترل هر گره جديد وجود داشته باشد تا متوجه شويم که آيا اين گره قبلاً در درخت جستجو توسط جستجوي طرف ديگر، ظاهر شده است يا خير.

5- نياز داريم که تصميم بگيريم که چه نوع جستجويي در هر نيمه قصد انجام دارد.

**مقايسه استراتژي‌هاي جستجو:**

ارزيابي استراتژي‌هاي جستجو. b فاکتور انشعاب، d عمل پاسخ، m ماکزيمم عمق درخت جستجو، l محدوديت عمق است.



**اجتناب از حالات تکراري:**

براي مسائل زيادي، حالات تکراري غيرقابل اجتناب هستند. اين شامل تمام مسائلي مي‌شود که عملگرها قابل وارونه شدن باشند، مانند مسائل مسيريابي و کشيش‌ها و آدمخوارها.

**سه راه براي حل مشکل حالات تکراري براي مقابله با افزايش مرتبه و سرريزي فشار کار کامپيوتر وجود دارد:**

به حالتي که هم اکنون از آن آمده‌ايد، برنگرديد. داشتن تابع بسط (يا مجموعه عملگرها) از توليد مابعدهايي که مشابه حالتي هستند که در آنجا نيز والدين اين گره‌ها وجود دارند، جلوگيري مي‌کند.

1. از ايجاد مسيرهاي دوار بپرهيزيد. داشتن تابع بسط (يا مجموعه عملگرها) از توليد مابعدهاي يک گره که مشابه اجداد آن گره است، جلوگيري مي‌کند.

2. حالتي را که قبلاً توليد شده است، مجدداً توليد نکنيد. اين مسئله باعث مي‌شود که هر حالت در حافظه نگهداري شود، پيچيدگي فضايي O(bd) داشته باشد. بهتر است که به O(s) توجه کنيد که s تعداد کل حالات در فضاي حالت ورودي است.

**جستجوي ارضاء محدوديت (Constraint Satisfaction Problem):**

نوع خاصي از مسئله است که CSP، حالات توسط مقادير مجموعه‌اي از متغيرها تعريف مي‌شوند و آزمون هدف مجموعه‌اي از محدوديت‌ها را به آنها اختصاص مي‌دهد که متغير ملزم به پيروي از آنها هستند.

CSPها مي‌توانند توسط الگوريتم‌هاي جستجوي geneal-purpose حل شوند، اما به علت ساختار خاص آنها، الگوريتم‌هايي صرفاً براي CSPهايي طرح مي‌شوند که از الگوريتم‌هاي عمومي کارآيي بهتري دارند.

محدوديت‌ها به گونه‌هاي مختلفي ظاهر مي‌شوند.

* محدوديت‌هاي يکتا
* محدوديت‌هاي دودويي
* محدوديت‌هاي مطلق
* محدوديت‌هاي اولويت‌دار
* در CSPهاي گسسته که دامنه‌هاي آن محدود هستند، محدوديت‌ها مي‌توانند به سادگي توسط شمردن ترکيبات مجاز مقادير نمايش داده شوند.
* با استفاده از يک شماره‌گذاري، هر CSP گسسته مي‌تواند به يک CSP دودويي تبديل شود.

چطور يک الگوريتم جستجوي همه منظوره را در يک CSP به کار ببريم.:

در حالتي که تمام متغيرها، تعيين نشده‌اند:

عملگرها مقداري را به يک متغير از مجموعه‌ مقادير ممکن، نسبت مي‌دهند.

آزمون هدف تمام متغيرها کنترل مي‌کند که آيا مقدار گرفته‌اند و تمام محدوديت‌ها از بين رفته‌اند يا خير.

* توجه کنيد که حداکثر عمق درخت جستجو در n و تعداد متغيرها و تمام راه‌حل‌ها در عمق n هستند.
* جستجوي عمقي روي يک CSP زمان جستجو را تلف مي‌کند زماني که محدوديت‌ها قبلاً مختلف شده باشند.

**فصل چهارم**

**روش‌هاي جستجو آگاهانه**

**جستجوي بهترين:**

اين استراتژي به اين صورت بيان مي‌شود که در يک درخت، زماني که گره‌ها مرتب مي‌شوند، گره‌اي که بهترين ارزيابي را داشته باشد، قبل از ديگر گره‌ها بسط داده مي‌شود.

هدف: يافتن راه‌حل‌هاي کم‌هزينه است، اين الگوريتم‌ها عموماً از تعدادي معيار تخمين براي هزينه راه‌حل‌ها استفاده مي‌‌کنند و سعي بر حداقل کردن آنها دارند.

**حداقل هزينه تخمين زده شده براي رسيدن به هدف: جستجوي حريصانه**

يکي از ساده‌ترين استراتژي‌هاي جستجوي بهترين، به حداقل رساندن هزينه تخمين زده شده براي رسيدن به هدف است. بدين صورت که حالت گره‌اي که به حالت هدف نزديک‌تر است، ابتدا بسط داده مي‌شود.

تابع کشف‌کننده: هزينه رسيدن به هدف از يک حالت ويژه مي‌تواند تخمين زده شود اما دقيقاً تعيين نمي‌شود. تابعي که چنين هزينه‌هايي را محاسبه مي‌کند تابع کشف‌کننده h ناميده مي‌شود.

جستجوي حريصانه: جستجوي بهترين که h را به منظور انتخاب گره بعدي براي بسط استفاده مي‌کند، جستجوي حريصانه (greedy search) ناميده مي‌شود.

ويژگي‌هاي جستجوي حريصانه:

* جستجوي حريصانه از لحاظ دنبال کردن يک مسير ويژه در تمام طول راه به طرف هدف، مانند جستجوي عمقي است، اما زماني که به بن‌بست مي‌رسد، برمي‌گردد.
* اين جستجو بهينه نيست و ناکامل است.
* پيچيدگي زماني در بدترين حالت براي جستجوي حريصانه O(bm)، که m حداکثر عمق فضاي جستجو است.
* جستجوي حريصانه تمام گره‌ها را در حافظه نگه مي‌دارد، بنابراين پيچيدگي فضاي آن مشابه پيچيدگي زماني آن است.
* ميزان کاهش پيچيدگي به مسئله و کيفيت تابع h بستگي دارد.

**حداقل‌سازي مجموع هزينه مسير: جستجوي A\***

جستجو با هزينه يکسان، هزينه مسير، g(n) را نيز حداقل مي‌کند.

با ترکيب دو تابع ارزيابي داريم:

f(n) = g(n) + h(n)

:g(n) هزينه مسير از گره آغازين به گره n را به ما مي‌دهد.

h(n): هزينه تخمين زده شده از ارزانترين مسير از n به هدف است

و ما داريم:

هزينه تخمين زده شده ارزانترين راه حل از طريق n = f(n)

**کشف‌کنندگي قابل قبول:**

تابع hاي را که هزينه‌اي بيش از تخمين براي رسيدن به هدف نداشته باشد، يک کشف‌کنندگي قابل قبول (admissible heuristic) گويند.

**جستجوي A\*:**

جستجوي بهترين که f به عنوان تابع ارزياب و يک تابع h قابل قبول استفاده مي‌کند، به عنوان جستجوي A\* شناخته مي‌شود.

**رفتار جستجوي A\***

نگاهي گذرا به اثبات کامل و بهينه بودن A\*:

مشاهده مقدماتي:

تقريباً تمام کشف‌کنندگي‌هاي مجاز داراي اين ويژگي هستند که در طول هر مسيري از ريشه، هزينه f هرگز کاهش پيدا نمي‌کند.

اين خاصيت براي کشف‌کنندگي، خاصيت يکنوايي (monotonicity) گفته مي‌شود.

اگر يکنوا نباشد، با ايجاد يک اصلاح جزئي آن را يکنوا مي‌کنيم.

بنابراين هر گره جديدي که توليد مي‌شود، بايد کنترل کنيم که آيا هزينة f اين گره از هزينه f پدرش کمتر است يا خير. اگر کمتر باشد، هزينة f پدر به جاي فرزند مي‌نشيند:

**بنابراين:**

f هميشه در طول هر مسيري از ريشه غيرکاهشي خواهد بود، مشروط بر اينکه h امکان‌پذير باشد.

h\*(n) : هزينه واقعي رسيدن از n به هدف است.

در استفاده عملي، خطاها با هزينه مسير متناسب هستند، و سرانجام رشد نمايي هر کامپيوتر را تسخير مي‌کند. البته، استفاده از يک کشف‌کنندگي خوب هنوز باعث صرفه‌جويي زيادي نسبت به جستجوي ناآگاهانه مي‌شود.

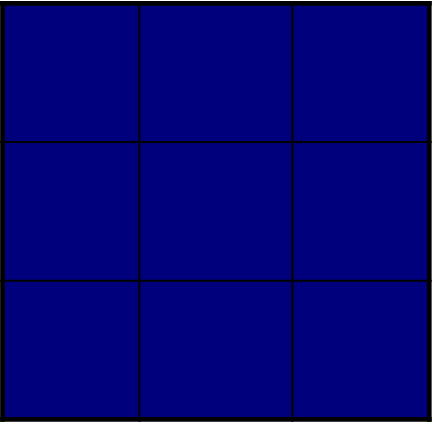
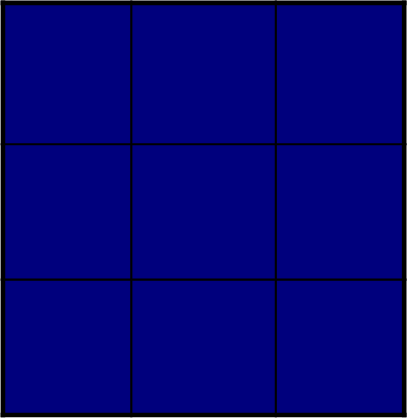
A\* معمولاً قبل از اينکه دچار کمبود زمان شود، دچار کمبود فضا مي‌شود. زيرا اين جستجو تمام گره‌هاي توليد شده را در حافظه ذخيره مي‌کند.

# توابع کشف‌کننده:

مسئله 8 را بررسي مي‌کنيم:

معماي 8 يکي از مسائل اوليه کشف‌کنندگي بود.

**هدف:** لغزاندن چهارخانه‌ها به طور افقي يا عمودي به طرف فضاي خالي است تا زماني که ساختار کلي مطابق با هدف (goal) باشد.



**1**

**2**

**3**

**8**

**4**

**7**

**6**

**5**

**5**

**4**

**1**

**6**

**8**

**7**

**3**

**2**

**Start State**

**Goal State**

اگر خواستار يافتن راه‌حل‌هاي کوتاه باشيم، به يک تابع کشف‌کننده نياز داريم که هرگز در تعداد مراحل به هدف اغراق نکند. در اينجا ما دو کانديد داريم:

h1 = تعداد چهارخانه‌هايي که در مکان‌هاي نادرست هستند. h1 يک کشف‌کننده مجاز است، زيرا واضح است که هر چهارخانه‌اي که خارج از مکان درست باشد حداقل يکبار بايد جابجا شود.

h2 = مجموع فواصل چهارخانه‌ها از مکان‌هاي هدف صحيحشان است. فاصله‌اي که ما حساب مي‌کنيم، مجموع فواصل عمودي و افقي است که بعضي وقتها city block distance و يا Manhattan distance ناميده مي‌شود.

**اثر صحت کشف‌کنندگي بر کارايي:**

يک راه براي تشخيص کيفيت کشف‌کنندگي فاکتور انشعاب مؤثر b\* است. اگر مجموع تعداد گره‌هاي بسط داده شده توسط A\*براي يک مسئله ويژه N باشد و عمق راه حل d، سپسb\* فاکتور انشعابي است که يک درخت يکنواخت با عمق d خواهد داشت تا گره‌هاي N را نگهدارد. بنابراين:

N=1+ b\*+( b\*)2…+( b\*)d

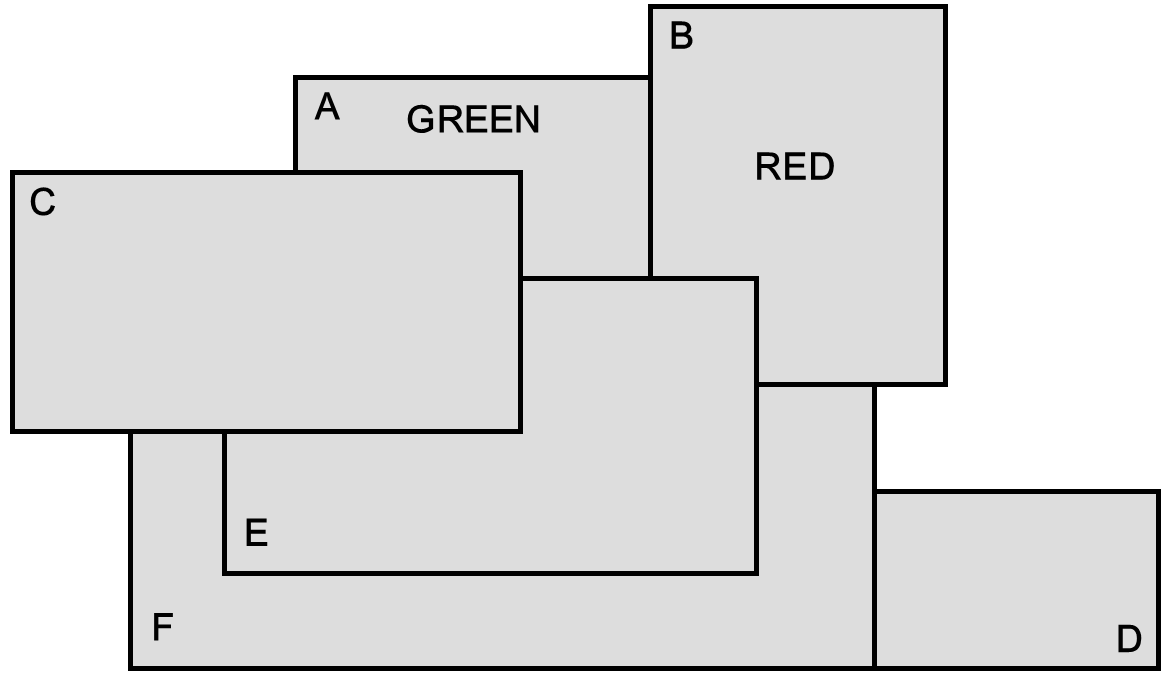
* معمولاً فاکتور انشعاب مؤثر که توسط کشف‌کنندگي نمايش داده مي‌شود، مقدار ثابتي دارد.
* يک کشف‌کنندگي خوب طراحي شده، b\* در حدود 1 دارد.

**کشف‌کننده‌ها براي مسائل ارضا محدوديت:**

مسئله ارضاء محدوديت شامل يک سري از متغيرهايي است که ,ويژگي‌هاي زير را دارا هستند:

* مي‌توانند مقاديري را از دامنة داده شده دريافت کنند.
* با يک سري از محدوديت‌ها، ويژگي‌هاي راه حل را مشخص کنند.

رنگ‌آميزي نقشه نمونه‌اي از اين کشف‌کننده‌هاست:

هدف رنگ‌آميزي نقشه، اجتناب از رنگ‌آميزي مشابه دو کشور همسايه است.

**ما حداکثر از سه رنگ (قرمز، آبي، سبز) مي‌توانيم استفاده کنيم:**

اگر رنگ سبز را براي کشور A، قرمز را براي B، انتخاب کنيم، کشور E بايد آبي باشد. در اين صورت ما ناچاريم که C را به رنگ قرمز درآوريم و F سبز باشد. رنگ‌آميزي D با رنگ آبي يا قرمز، بستگي به راه‌حل دارد. در اين حالت مسئله بدون هيچگونه جستجويي حل مي‌شود.

**جستجوي SMA\*:**

الگوريتم SMA\*، حافظه محدود A\* ساده شده (Simplified-Memory-BoundedA\*) مي‌باشد.

اين الگوريتم، قادر است تا از تمام حافظه موجود براي اجراي جستجو استفاده کند. استفاده از حافظه بيشتر کارايي جستجو را وسعت مي‌بخشد. مي‌توان هميشه از فضاي اضافي صرفنظر کرد.

**SMA\* داراي خواص زير است:**

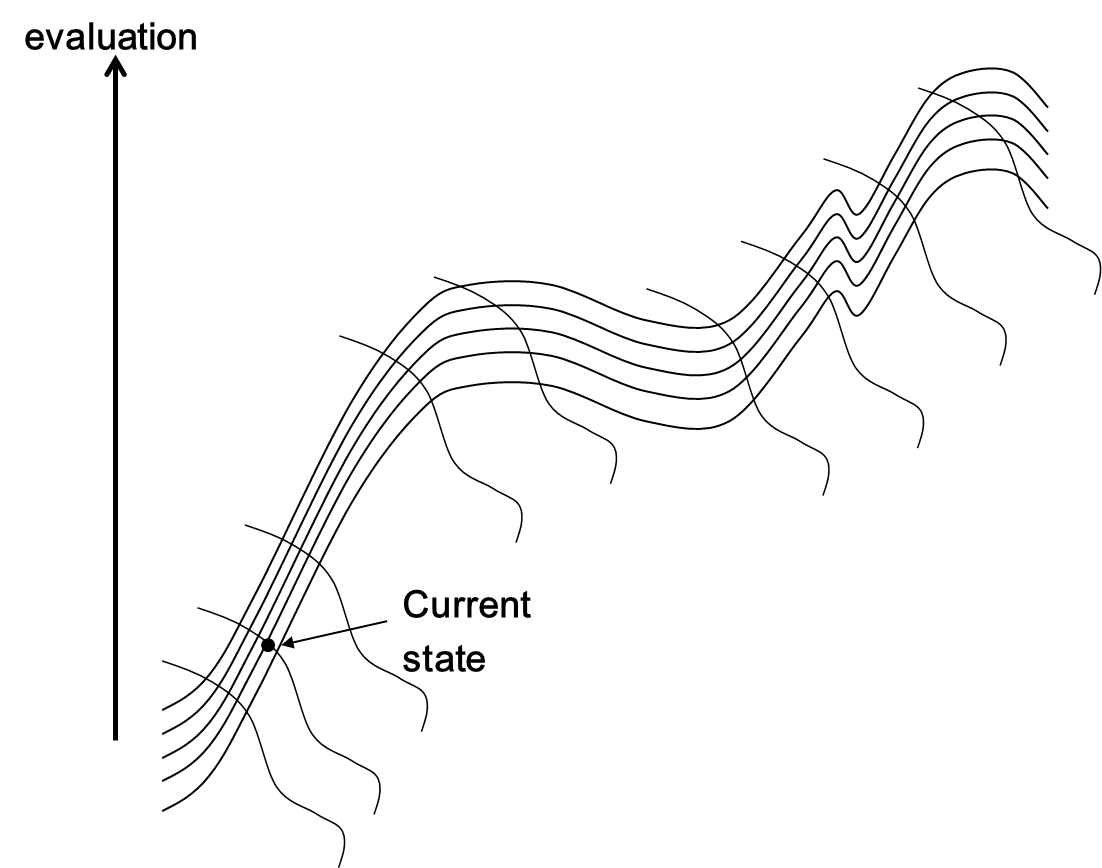
* مي‌تواند از تمام حافظه قابل دسترس استفاده کند.
* از حالات تکراري تا جايي که حافظه اجازه مي‌دهد، جلوگيري مي‌کند.
* اين الگوريتم کامل است به شرط آنکه حافظه براي ذخيره کم عمق‌ترين مسير راه حل کافي باشد.
* اين الگوريتم بهينه است، اگر حافظه کافي براي ذخيره کم‌عمق‌ترين مسير راه‌حل کافي باشد. بعلاوه بهترين راه‌حلي را برمي‌گرداند که بتواند با حافظه موجود مطابقت داشته باشد.
* زماني که حافظه موجود براي درخت جستجوي کامل کافي باشد، جستجو Optimally efficient است.

طراحي SMA\* ساده است.

* زماني که نياز به توليد فرزند داشته باشد ولي حافظه‌اي نداشته باشد، نياز به ساختن فضا بر روي صف دارد. براي انجام اين امر، يک گره را حذف مي‌کند. گره‌هايي که به اين طريق از صف حذف مي‌شوند، گره‌هاي فراموش‌شده يا (forgotten nodes) ناميده مي‌شوند.
* براي اجتناب از جستجوي مجدد زيردرخت‌هايي که از حافظه حذف شده‌اند، در گره‌هاي اجدادي، اطلاعاتي در مورد کيفيت بهترين مسير در زير درخت فراموش شده، نگهداري مي‌شود.

**الگوريتم‌هاي اصلاح تکراري**

* بهترين راه براي فهم الگوريتم‌هاي اصلاح تکراري درنظر داشتن تمام حالاتي است که روي سطح يک دورنمايي در معرض ديد قرار داده شده است. ارتفاع هر نقطه در دورنما مطابق با تابع ارزياب حالت آن نقطه است. ايده اصلاح تکراري، حرکت کردن در اطراف دورنما و سعي بر يافتن قله‌هاي مرتفع است، که همانا راه‌حل‌هاي بهينه هستند.
* الگوريتم‌هاي اصلاح تکراري معمولاً اثر حالت جاري را فقط حفظ مي‌کنند، و توجهي فراتر از همسايگي آن حالت ندارند.

**الگوريتم‌هاي اصلاح تکراري سعي بر يافتن قله‌هايي بروي سطح حالات دارند، جائي که ارتفاع توسط تابع ارزيابي تعريف مي‌شود.**

**اين الگوريتم‌ها به دو گره اصلي تقسيم مي‌شوند.**

* الگوريتم‌هاي تپه‌نوردي (Hill-climbing)
* Simulated annealing

**1- الگوريتم‌هاي جستجوي تپه‌نوردي (Hill-climbing)**

يک اصلاح خوب اين است زماني که بيش از يک فرزند خوب براي انتخاب وجود دارد، الگوريتم بتواند به طور تصادفي از ميان آنها يکي را انتخاب کند.

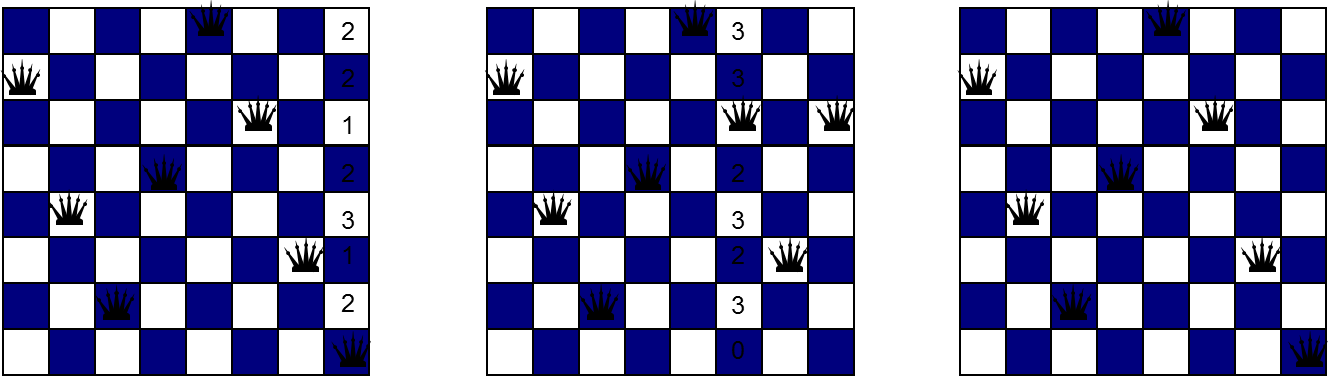
اين سياست ساده، سه زيان عمده دارد:

* Local Maxima : يک ماکزيمم محلي، برخلاف ماکزيمم عمومي، قله‌اي است که پائين‌تر از بلندترين قله درفضاي حالت است. زماني که روي ماکزيمم محلي هستيم، الگوريتم توقف خواهد نمود. اگرچه راه حل نيز ممکن است دور از انتظار باشد.
* Plateaux : يک فلات محوطه‌اي از فضاي حالت است که تابع ارزياب يکنواخت باشد. جستجو يک قدم تصادفي را برخواهد داشت.
* Ridges : نوک کوه، داراي لبه‌هاي سراشيب است. بنابراين جستجو به بالاي نوک کوه به آساني مي‌رسد، اما بعد با ملايمت به سمت قله مي‌رود. مگر اينکه عملگرهايي موجود باشند که مستقيماً به سمت بالاي نوک کوه حرکت کنند. جستجو ممکن است از لبه‌اي به لبه ديگر نوسان داشته باشد و پيشرفت کمي را حاصل شود.

در هر مورد، الگوريتم به نقطه‌اي مي‌رسد که هيچ پيشرفتي نيست. اگر اين اتفاق بيفتد، تنها کار ممکن براي انجام دادن آغاز مجدد از نقطه شروع ديگري دوباره آغاز مي‌شود.

موفقيت hill-climbing خيلي به ظاهر فضاي حالت «سطح» بستگي دارد: اگر فقط ماکزيمم‌هاي محلي کمي وجود داشته باشد، تپه‌نوردي با شروع تصادفي خيلي سريع راه‌حل خوبي را پيدا خواهد کرد.

**2- Simulated annealing**

در اين گروه از الگوريتم‌ها به جاي شروع دوباره به طور تصادفي زماني که در يک ماکزيمم محلي گير افتاديم، مي‌توانيم اجازه دهيم که جستجو چند قدم به طرف پائين بردارد تا از ماکزيمم محلي فرار کند.

راه‌حل دو مرحله‌اي براي مسئله 8 وزير با استفاده از حداقل برخوردها را نشان مي‌دهد. در هر مرحله يک وزير به منظور تعيين مجدد ستون انتخاب مي‌شود. تعداد برخوردها (در اين مورد، تعداد وزيرهاي حمله‌کننده) در هر چهارخانه نشان داده شده است. الگوريتم وزير را به چهارخانه‌اي که حداقل برخورد را داشته باشد، براي از بين بردن تصادفي برخوردها، حرکت مي‌دهد.

اگر حرکت واقعاً شرايط را بهبود بخشد، آن حرکت هميشه اجرا مي‌شود.

پارامتر‌هاي مؤثر به شرح زير مي‌باشند:

1- : چگونگي ارزيابي.

2- T: تعيين احتمال.

الگوريتم شباهت صريحي با annealing (پردازشي که به طور آهسته مايعي را تا زماني که يخ ببندد سرد مي‌کند)، گسترش يافته است. مقدار تابع مطابق با انرژي ورودي اتم‌هاي ماده است، و T با دما مطابقت دارد. جدول ميزان دما را در جايي که پائين آمده است، تعيين مي‌کند.

**کاربردها در مسائل ارضا محدوديت**

مسائل ارضاء محدوديت (CSP)، مي‌توانند توسط روش‌هاي اصلاح تکراري با استفاده از موارد زير حل شوند.

* مقدار دادن به تمام متغيرها.
* به کاربردن عملگرهاي تغيير به منظور حرکت دادن ساختار به طرف يک راه‌حل.

الگوريتم‌هايي که CSPها را حل مي‌کنند، روش‌هاي تصحيح کشف‌کنندگي، ناميده مي‌شوند، زيرا آنها تناقضات را در ساختار جاري مسئله اصلاح مي‌کنند.

در انتخاب مقدار جديد براي يک متغير، واضح‌ترين کشف‌کنندگي انتخاب مقداري است که کمترين مقدار تناقضات را با ديگر متغيرها نتيجه دهد، که همان کشف‌کنندگي مينيمم تناقضات است.

فصل 5

**تئوري بازي**

**بازي‌ها در نقش مسائل جستجو**

رقابت انتزاعي، که در بازي‌هاي صفحه‌اي ديده مي‌شود، موجب شده تا تئوري بازي جزء تحقيقات AI قرار بگيرد.

وضعيت بازي براي بازنمايي آسان است و عامل‌ها معمولاً به تعداد کمي از عمليات محدود مي‌شوند.

دلايلي که محققين قديم، شطرنج را به‌عنوان موضوعي در AI برگزيدند:

* بازي شطرنج کامپيوتري اثباتي بر وجود ماشيني است که اعمال هوشمندانه‌اي را انجام مي‌دهند.
* سادگي قوانين
* وضعيت دنيا کاملاً براي برنامه شناخته شده است. (بازنمايي بازي به عنوان يک جستجو از طريق فضاي موقعيت‌هاي ممکن بازي، ساده است.)
* پيچيدگي بازي‌ها، به طور کامل نوعي از عدم قطعيت را معرفي مي‌کنند.
* عدم قطعيت به علت وجود اطلاعات گم شده رخ نمي‌دهد، اما به علت اينکه فرد زماني براي محاسبه دقيق نتايج حرکت ندارد عدم قطعيت بوجود مي‌آيد.
* در اين مورد، فرد بر اساس تجربيات گذشته مي‌تواند بهترين حدس را بزند.
* **تصميمات کامل در بازي‌هاي دونفره:**
* مورد کلي از يک بازي با دو بازيکن را در نظر مي‌گيريم که آن را MIN,MAX مي‌ناميم.

يک بازي به طور رسمي مي‌تواند به عنوان نوعي از مسئله جستجو به همراه قسمت‌هاي زير تعريف شود:

* حالت اوليه شامل مکان صفحه وتعيين نوبت حرکت هر بازيکن است.
* مجموعه‌اي از عملگرها که حرکات صحيح را که بازيکن مي‌تواند انجام دهد، تعيين مي‌کند.
* آزمون پاياني زمان بازي را تعيين مي‌کند. حالاتي را که بازي درآنها به پايان رسيده است حالات پاياني ناميده مي‌شوند.
* تابع سودمندي (تابع امتياز payoff) که مقدار عددي براي نتيجه بازي را تعيين مي‌کند.

اگر به آن به عنوان يک مسئله جستجو نگاه شود، جستجو براي دنباله‌اي از حرکات که منتهي به حالت پاياني مي‌شد (مطابق با تابع سودمندي)، و سپس پيشروي و ساخت اولين حرکت در دنباله بود.

اما حرکات MIN غير قابل پيش‌بيني است؛

بنابراين:

MAX بايد استراتژي‌اي را بيابد که به يک حالت پاياني برنده بدون توجه به عملکرد MIN منجر شود، که اين استراتژي شامل حرکات درست براي MAX براي هر حرکت ممکن از MIN مي‌باشد.

الگوريتم MINMAX به منظور تعيين استراتژي بهينه براي MAX طراحي شده است و از اين رو مي‌توان بهترين حرکت را تصميم‌‌گيري کرد. الگوريتم شامل 5 مرحله است:

1. توليد درخت کامل بازي، تمام راه تا مراحل پاياني
2. درخواست تابع سودمندي براي هر حالت پاياني به منظور بدست آوردن مقدارش.
3. از سودمندي حالات پاياني به منظور تعيين سودمندي گره‌ها يک مرحله بالاتر دردرخت جستجو استفاده کنيد.
4. بررسي مقادير را از گره‌هاي برگي تا ريشه، يک لايه در هر لحظه، ادامه دهيد.
5. احتمالاً مقادير به بالاي درخت مي‌رسند، MAX حرکتي را انتخاب مي‌کند که به بالاترين مقدار منتهي مي‌شود.

**اگر:**

m: حداکثر عمق درخت،

b: تعداد حرکات قانوني در هر نقطه،

**آنگاه:**

زمان پيچيدگي الگوريتم minimax ، O(bm) است.

الگوريتم يک جستجو عمقي است.

**تصميمات ناقص:**

الگوريتم minimax فرض مي‌کند که برنامه زمان لازم براي جستجوي تمامي راه‌هاي ممکن وضعيت‌هاي پاياني را دارد که اين فرض معمولاً عملي نيست.

**الگوريتم ميني‌ماکس، به دو راه تغيير يابد:**

* تابع سودمندي با تابع ارزيابي EVAL جايگزين شود.
* آزمون پاياني با آزمون قطع CUTOFF-TEST جايگزين گردد.

تابع ارزيابي:

تابع ارزيابي تخميني از سودمندي مورد انتظار بازي را ازموقعيت داده شده برمي‌گرداند.

واضح است که ارائه يک برنامه بازي بي نهايت به کيفيت تابع ارزيابي بستگي دارد.

چگونه به طور دقيق کيفيت را مي‌توان اندازه گرفت؟

1. تابع ارزيابي با تابع سودمندي در مورد حالت پاياني بايد به توافق برسند.
2. نبايد زياد طول بکشد! (اگر پيچيدگي را محدود نکنيم minimax به عنوان يک زيربرنامه فراخواني مي‌شود و مقدار دقيق وضعيت محاسبه مي‌شود.) از اين رو، معامله‌اي بين صحت تابع ارزيابي و هزينه زمان آن وجود دارد.
3. تابع ارزيابي بايد به درستي شانس‌هاي واقعي براي برد را منعکس کند.

تابع ارزيابي فرض مي‌گيرد که مقدار هر مهره مي‌تواند به طور مستقل از ديگر مهره‌ها روي صفحه قضاوت شود. اين نوع از تابع ارزيابي، تابع خطي وزن دار ناميده مي‌شود.

اين تابع مي‌تواند به صورتي ذکر شود که Wها وزن ها هستند و fها اعداد هر نوع مهره روي صفحه خواهند بود.

**قطع جستجو:**

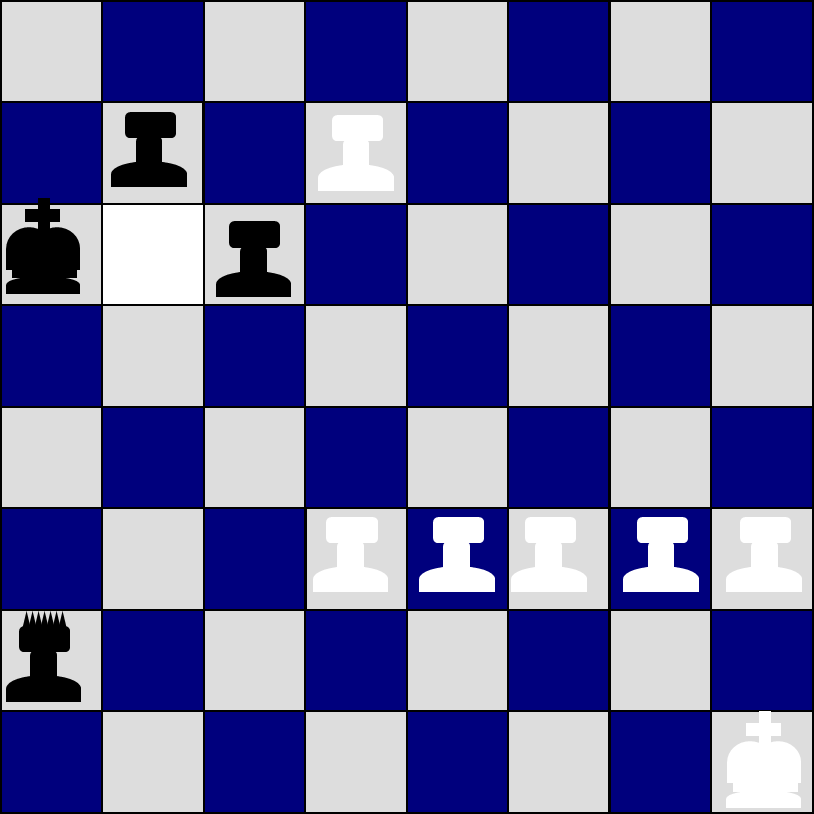
صريح‌ترين رهيافت براي کنترل ميزان جستجو قراردادن محدوديتي براي داشتن يک عمق ثابت است، بنابراين تست قطع براي تمام گره‌ها در زير عمق d موفق مي‌شود. عمق طوري انتخاب مي‌شود که ميزان زمان استفاده شده از آنچه که قوانين بازي اجازه مي‌دهد تجاوز نکند.

زماني که، وقت تمام مي‌شود، برنامه حرکت انتخابي توسط عميق‌ترين جستجوي کامل شده را برمي‌گرداند.

به دليل تخميني بودن توابع ارزيابي اين رهيافتها مي‌توانند نتايج ناخوشايندي را به همراه داشته باشند.

تابع ارزيابي فقط بايد براي مواقعي به کاربرده شود که خاموش هستند، يعني اينکه تفاوتهاي چشم‌گير در مقدار، در آينده نزديک بعيد به نظر مي‌رسد.

اين جستجوي فوق العاده **جستجوي خاموش** ناميده مي‌شود.

**مسئله افقي (horizonproblem):** رخ سياه مانع از حرکت وزير سفيد به حالت افقي شده است و اين موقعيت به نفع سياه است. در حالي که برگ برنده در دست سفيد است.

**هرس آلفا-بتا:**

هرس درخت جستجو:

پردازش حذف شاخه‌اي از درخت جستجو، با در نظر داشتن و بدون آزمايش، هرس درخت جستجو ناميده مي‌شود.

زماني که اين تکنيک براي يک درخت minimax استاندارد، به کار برده مي‌شود، حرکت مشابهي همانطور که minimax انجام مي‌داد، برمي‌گرداند؛ اما شاخه‌هايي که در تصميم‌ نهايي دخالتي ندارند را هرس مي‌کند.

جستجوي minimax عمقي است، بنابراين، در هر لحظه، بايد گره‌هايي در نظر گرفته شوند که در طول يک مسير مجزا در درخت هستند.

α مقدار بهترين انتخابي باشد که تا کنون در طول مسير براي MAX پيدا شده است. و β مقدار بهترين (به طور مثال، پايين‌ترين مقدار) انتخابي باشد که در طول مسير تا اين لحظه براي MIN پيدا شده است.

درخت جستجوي آلفا-بتا:

اين درخت، مقدار α و β را همچنانکه جلو مي‌رود، به روز درمي‌آورد، و زير درخت را هرس مي‌کند (فراخواني بازگشتي را قطع مي‌کند) به محض اينکه معلوم مي‌شود که اين زير درخت بدتر از مقدار α يا β جاري است.

**مزاياي هرس آلفا-بتا**

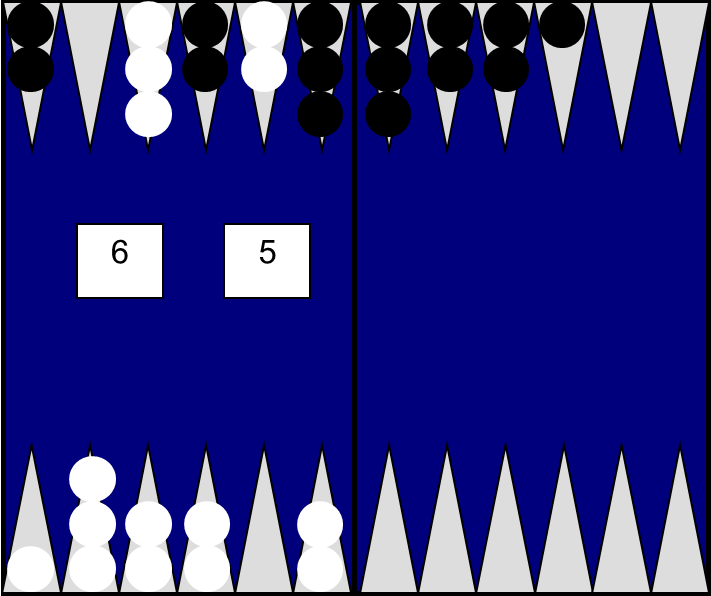
مزاياي آلفا-بتا به مرتبه‌اي که در آن گره‌هاي فرزندي آزمايش شده‌اند، برمي‌گردد.

پيچيدگي O(b/log b)d مي‌باشد.

1. در عمل، يک تابع ساده مرتب‌کننده شما را به نتيجه بهترين حالت بر خلاف نتيجه تصادفي سوق مي‌دهد.
2. رهيافت مشهور ديگر انجام جستجوي عميق‌کننده تکراري و استفاده از مقادير backed-up از يک تکرار براي تعيين ترتيب جانشين‌ها در تکرار بعدي است.

نتايج بازي‌ها نيز قابل ملاحظه هستند (و در حقيقت، مسائل جستجو در حالت کلي) و بايد به صورت يک مدل درخت مطلوب فرض شوند تا نتايجشان را به دست آورند.

**بازي‌هايي که شامل عنصر شانس هستند:**

تخته نرد يک بازي عمومي است که شانس و مهارت را با هم ترکيب مي‌کند.

تاس‌هاي سفيد 5-6، چهار حرکت زير را مي‌تواند انجام دهد:

(16-10 و 10-5) و (24-19 و 11-5) و (11-5 و 10-5) و (16-11 و 11-5)

درخت بازي در تخته نرد بايد شامل گره‌هاي شانس براي گره‌هاي MIN و MAX‌ باشد.

مرحله بعدي فهم چگونگي ساخت تصميمات صحيح است.

محاسبه مقادير انتظاري گره‌ها، صريح است. براي گره‌هاي پاياني، از تابع سودمندي مانند بازيهاي قطعي استفاده مي‌کنيم.

با پيشروي در درخت جستجو به اندازه يک مرحله، به يک گره شانس برخورد مي‌کنيم.

اگر ما فرض کنيم که S(C,di) مجموعه موقعيت‌هاي توليد شده توسط اعمال حرکات قانوني براي پرتاب P(di) در موقعيت C باشد، مي‌تواند مقدار expectimax از ‍C را با استفاده از فرمول زير محاسبه نمود:

Expectimax (c)=∑I P(di) maxsε S(c,di) (utility(s))

اين فرمول، سودمندي مورد انتظار در موقعيت c را با فرض بهترين بازي ارائه مي‌دهد.

**ارزيابي موقعيت در بازي‌ها با گره‌هاي شانس:**

حضور گره‌هاي شانس بدين معناست که بايد در مورد آنچه که به معناي مقادير ارزيابي است، دقيق بود.

اگر ما تغييري را در مقياس مقادير ارزيابي ايجاد کنيم، برنامه در مجموع به طور متفاوت رفتار مي‌کند.

**پيچيدگي:**

بدليل اينکه expectiminimax تمام دنباله‌هاي پرتاب تاس را در نظر مي‌گيرد، زماني معادل O(bmnm) مي‌برد، که n تعداد پرتابهاي محدود است.

مزيت آلفا- بتا، با داشتن بهترين بازي ناديده گرفتن پيشرفت‌ها در آينده است که احتمال وقوعشان کم است.

در بازيهاي به همراه تاس، دنباله‌هاي محتملي از حرکات وجود ندارد، چون براي آن حرکاتي که بايد انجام بگيرند، ابتدا تاس بايد به روش درستي پرتاب شود تا آن حرکات منطقي شوند.

اگر بگوئيم که تمام مقادير سودمندي بين 1+ و 1- هستند، سپس مقدار گره‌هاي برگي محدود مي‌شوند و در عوض‌ ما مي‌توانيم حد بالايي روي مقدار گره شانسي بدون توجه به فرزندانش قرار دهيم.

فصل ششم

عامل‌هاييکه به طور منطقي استدلال مي‌‌کنند

**معرفي طراحي پايه‌اي براي يک عامل مبتني بر دانش:**

رهيافت مبتني بر دانش روش قدرتمندي از ساخت برنامه عامل است. هدف آن پياده‌سازي نمايي از عامل است که بتواند به عنوان دانش در مورد دنياي آنها و استدلال در مورد گونه‌هايي ممکن از رفتار آنها به کار مي‌رود.

عامل‌هاي مبتني بر دانش قادرند که:

1. وظايف جديد را به صورت اهداف تعريف شده صريح قبول کنند.
2. آنها مي‌توانند به سرعت توسط گفتن يا يادگيري دانش جديد درمورد حيطه، به رقابت برسند.
3. آنها مي‌توانند خود شانس را با تغييرات محيط، توسط به روز در آوردن دانش مربوطه، تطبيق دهند.

عامل مبتني بر دانش به موارد زير نياز دارد:

1. چه چيزهايي را بداند؟
2. وضعيت جاري دنيا؟
3. چطور توسط ادراک به خواص ناديده دنيا رجوع کند؟
4. چطور دنيا زمان را مي‌گشايد؟
5. عامل به چيزي مي‌خواهد برسد؟
6. فعاليت‌هايي که در شرايط مختلف انجام مي‌دهد چيست؟

بخش مرکزي عامل مبتني بر دانش پايگاه دانش (knowledge base) آن، يا KB است.

**پايگاه دانش:** مجموعه‌اي از نمايش حقايق در مورد نياز است.

**جمله:** هر نمايش اختصاصي يک جمله (sentence) ناميده مي‌شود.

**جملات:** جملات در يک زباني که زبان بازنمايي دانش (knowledge representation) ناميده مي‌شود، بيان مي‌شوند.

**ASK:** به منظور افزودن جملات جديد به پايگاه دانش به کار برده مي‌شود.

**TELL:** به منظور پرسش اينکه چه چيزهايي شناخته شده است.

تشخيص اينکه چه چيزي بايد پس از TELLed به KB دنبال شود، مسئوليت مکانيزمي به نام استنتاج (inference) است، که قسمت مهم ديگر عامل مبتني بر دانش را تشکيل مي‌دهد.

هر زمان که برنامه دانش صدا زده مي‌شود، دو عمل انجام مي‌شود:

1. به پايگاه دانش گفته مي‌شود (TELL) که چه دريافت کرده است.
2. از پايگاه دانش سؤال مي‌‌شود (ASK) که چه عملي بايد انجام شود.

در فرآيند پاسخ به اين پرسش، استدلال منطقي براي اثبات اينکه کدام عمل بهتر از بقيه است استفاده مي‌شود و دانسته‌هاي عامل و اهداف آن مشخص مي‌شوند.

مي‌توانيم يک عامل مبتني بر دانش را در سه سطح تعريف کنيم:

1. سطح دانش knowledge level يا سطح epistemological که خلاصه‌ترين سطح است؛ مي‌توانيم عامل را توسط گفتن اينکه عامل چه مي‌داند، تعريف ‌نماييم.
2. سطح منطقي logical level سطحي است که دانش به صورت جملات رمزگذاري مي‌شود.
3. سطح پياده سازي Implementation Level سطحي است که در معماري عامل اجرا مي‌شود و بازنمايي‌هاي فيزيکي از جملات سطح منطقي، در اين سطح وجود دارد.

انتخاب پياده‌سازي در کارآيي بهتر عامل بسيار اهميت دارد، اما به سطح منطقي و سطح دانش مربوط نمي‌‌شود.

**دنياي WUMPUS:**

مشابه دنياي مکش، دنياي Wumpus شبکه‌اي از مربع است که توسط ديوارهايي احاطه شده‌اند، که هر مربع مي‌تواند شامل عامل‌ها و اشياء باشد.

وظيفه عامل يافتن طلا و بازگشتن به نقطه شروع و بالا رفتن از غار است.

براي مشخص نمودن وظيفه عامل، ادراکات، عمليات و اهداف آن را بايد مشخص کنيم. در دنياي Wumpus، اينها به صورت زير هستند:

* از مربعي که شامل Wumpus است و مربع‌هاي مجاور (نه قطري) عامل بوي بدي را دريافت مي‌کند .
* در مربعهايي که مستقيماً مجاور با چاله‌ها هستند، عامل نسيمي را دريافت مي‌کند.
* در مربعي که طلا وجود دارد، عامل يک درخششي را درک مي‌کند.
* زماني که يک عامل به داخل ديواره قدم بر مي‌دارد، ضربه‌اي را دريافت مي‌کند.
* زماني که Wumpus کشته مي‌شود، فريادي سر مي‌دهد که هر جايي از غار شنيده مي‌شود.
* ادراکات به عامل به صورت ليستي از پنج سيمبول داده مي‌شود.
* مانند دنياي مکش، عملايتي براي جلو رفتن، چرخيدن 90 به سمت چپ، چرخيدن 90 به سمت راست وجود دارد.
* عامل نابود خواهد شد زماني که وارد يک مربع شامل سياده چاله و يا کي Wumpus زنده مي‌‌شود.
* هدف عامل يافتن طلا و برگرداندن آن به خانه شروع با سرعت تمام است، بدون آنکه کشته شود.

**بازنمايي، استدلال و منطق:**

بازنمايي و استدلال با همديگر،‌ عملکرد يک عامل مبتني بر دانش را حمايت خواهند کرد.

بازنمايي دانش (knowledge representation) دانش را در فرم حل شدني کامپيوتر مطرح مي‌سازد، که به عامل‌ها کمک مي‌کند تا ارائه بهتري داشته باشند.

**زبان بازنمايي دانش متوسط دو خاصيت تعريف مي‌شود:**

**نحو (Syntax):** يک زبان ساختاري ممکن براي تشکيل جملات را ايجاد مي‌کند.

بازنمايي واقعي در داخل کامپيوتر: هر جمله توسط يک ساختار فيزيکي يا خاصيت فيزيکي قسمتي از عامل پياده‌سازي مي‌شود.

**معني (Semantic):** تعيين مي‌کند که حقايق موجود در دنيا به چه جملاتي نسبت داده شوند.

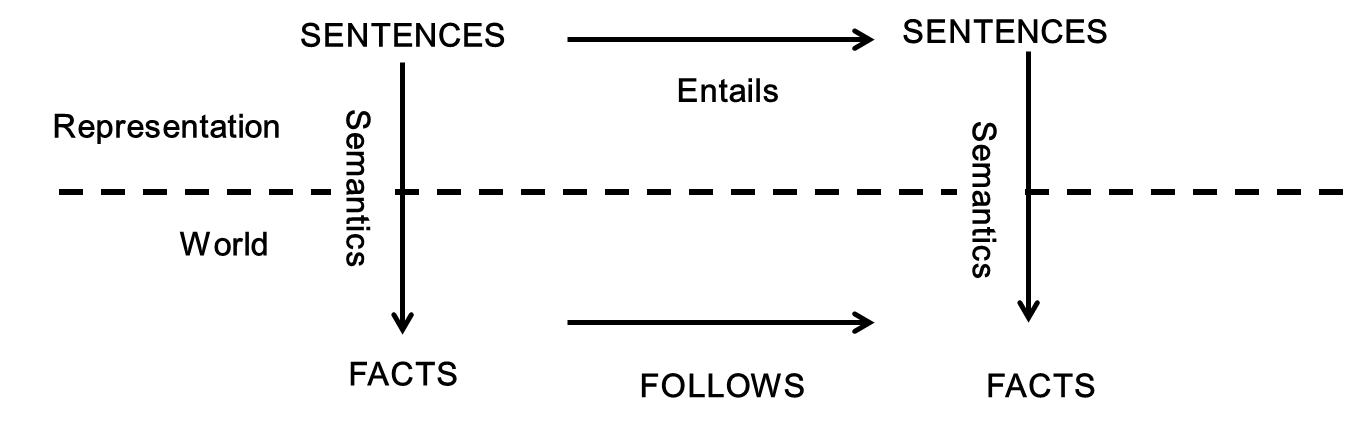
با Semanticها، مي‌توانيم بگوييم زماني که ساختار ويژه با يک عامل وجود دارد، عامل به جملات مربوطه، اعتقاد دارد.

معني‌هاي زبان تعيين مي‌کند که حقايق به کدام جملات مربوط مي‌شوند.

**تفاوت بين حقايق و بازنمايي‌هاي آنها:**

حقايق قسمتي از دنياي واقعي را تشکيل مي‌دهند، اما بازنمايي‌هاي آنها بايد به صورتي کد شوند که بتواند به طور فيزيکي در يک عامل ذخيره شود.

جملات قسمتي از ساختار فيزيکي عامل هستند و استدلال بايد پردازشي از ايجاد ساختار جديد فيزيکي از نمونه‌هاي قديمي‌تر باشد.

استدلال مطلوب بايد اين اطمينان را حاصل کند که ساختار جديد حقايقي را بازنمايي مي‌کند که از حقايقي که ساختار قديمي ايجاد کرده بود، پيروي کنند.

**استلزام:**

ارتباط بين حقايقي که دنباله رو يکديگر هستند را نشان مي‌دهد.

در علائم رياضي، ارتباط استلزام بين يک پايگاه دانش KB و يک جمله a به صورت «KB مستلزم a است» تلفظ مي‌شود و به صورتKB|= a نوشته مي‌‌شود.

رويه استنتاج مي‌تواند يکي از دو عامل ذيل را انجام دهد:

1. با داشتن پايگاه دانش KB مي‌تواند جملات تازه‌اي از a توليد کند که مفهوم آن استلزام توسط KB باشد.
2. يا با داشتن يک پايگاه دانش KB و جمله a ديگري، اين رويه مي‌تواند گزارش دهد که a توسط KB مستلزم شده است يا خير.

رويه استنتاج i مي‌تواند توسط جملاتي که آنها را مشتق مي‌کند، تعريف شود. اگر i بتواند a را از KB مشتق کند، منطق‌دان مي‌تواند بنويسيد: |\_ I a KB که خوانده مي‌‌شود «آلفا از KB توسط i مشتق شده است يا «i مشتق مي‌کند آلفا از KB».

ثبت عمليات رويه استنتاج صحيح، اثبات (Proof) ناميده مي‌شود.

**کليد استنتاج صحيح:**

داشتن مراحل استنتاج است که به جملات مورد عمل قرار گرفته، توجه داشته باشد.

**بازنمايي:**

زبان‌هاي برنامه‌نويسي (مانند C يا پاسکال يا Lips) براي تعريف الگوريتم‌ها مناسب هستند و بين ساختارهاي داده پيوستگي ايجاد مي‌کنند.

زبان‌هاي طبيعي بيش‌تر محتاج محاوره بر خلاف بازنمايي هستند.

**مزايا و معايب زبان طبيعي:**

زبان طبيعي راهي خوب براي سخنگو است تا مخاطب را متوجه منظور خود سازد؛ اما اغلب اين تقسيم دانش بدون بازنمايي صريح خود دانش انجام مي‌شود. زبان‌هاي طبيعي هم چنين از ابهامات رنج مي‌برند، مانند عبارت «سگ‌ها و گربه‌هاي کوچک»، روشن نيست که آيا سگ‌ها نيز کوچک هستند يا خير.

**يک زبان بازنمايي خوب مي‌بايست:**

مزاياي زبان‌هاي طبيعي و رسمي را با هم داشته باشد.

پرمعني و رسا باشد.

دقيق و غير مبهم

مستقل از متن

قابل استنتاج

**معاني:**

يک جمله خودش به تنهاي معنايي ندارد.

مي‌توان زباني را تعريف نمود که در آن هر جمله يک تفسير اختياري داشته باشد. اما در عمل تمام زبان‌هاي بازنمايي ارتباط سيستماتيکي بين جملات اعمال مي‌کنند.

**صدق‌پذيري:**

يک جمله معتبر (Valid) يا لزوماً صحيح است اگر و فقط اگر تحت تمام تفسيرهاي ممکن در تمام دنياي ممکن، بدون توجه از آنچه که تصور مي‌شد که معنا دهد و بدون توجه به حالت آن مطلب در کل، تعريف شده باشد.

يک جمله صدق‌پذير (satisfiable) است اگر و فقط اگر تفسيري در دنيايي براي صحت آن وجود داشته باشد. جمله در خانه [1,2] Wumpus وجود دارد " Satisfiable است زيرا امکان دارد که Wumpus در آن خانه باشد، حتي اگر چنين اتفاقي نيفتاده باشد، جمله‌اي که صدق‌پذير نباشد صدق ناپذير (unsatisfiable) است.

جملات خود تناقضي صدق‌ناپذير هستند، اگر تناقض به معناي سيمبول‌ها بستگي نداشته باشد.

**استنتاج در کامپيوترها:**

معتبر بودن و صدق ناپذيري به قابليت کامپيوتري که استدلال مي‌کند، بستگي دارد.

**کامپيوتر‌ها از دور نقطه ضعف رنج مي‌برند:**

کامپيوتر لزوماً تفسيري را که شما براي جملات در پايگاه دانش به کار مي‌برديد، نمي‌داند.

چيزي در مورد دنيا نمي‌داند به جز آنچه که در پايگاه دانش ظاهر مي‌شود.

چيزي که استنتاج رسمي را قدرت مي‌بخشد، نبودن محدوديت بر روي پيچيدگي جملاتي است که کاميپوتر بايد آنها را مورد عمل قرار دهد.

بزرگترين چيز در مورد استنتاج رسمي، قابليت آن براي بدست آوردن نتايج صحيح است حتي زماني که کامپيوتر اطلاعي از تفسير استفاده شده توسط شما نداشته باشد.

کامپيوتر فقط نتايج معتبر را گزارش مي‌کند، که بايست بدون توجه به تفسير شما، صحيح باشد.

**منطق شامل موارد زير مي‌شود:**

1- يک سيستم رسمي براي تعريف حالتهاي مطلب که شامل:

الف- نحو (syntax) زبان، که روش درست کردن جملات را شرح مي‌دهد.

ب- معاني (semantic) زبان، که محدوديت‌هاي سيستماتيکي را روي چگونگي ارتباط جملات با حالات موضوع قرار مي‌دهند.

2- تئوري اثبات- مجموعه‌اي از قوانين براي استنباط استلزامي يک سري از جملات.

ما روي دو نوع منطق تمرکز خواهيم کرد:

* منطق بولين يا گزاره‌اي،
* منطق مرتبه اول (دقيق تر بگوييم، حساب گزاره‌ مرتبه اول با تساوي.)
* در منطق گزاره‌اي سيمبول‌ها تمام گزاره‌ها را بازنمايي مي‌کنند.
* سيمبولهاي گزاره‌اي مي‌توانند با استفاه از ربط‌دهنده‌هاي بولين (Boolin connevtives) جملات را با معناهاي پيچيده‌ترين توليد کنند.

منطق مرتبه اول با بازنمايي دنياهايي به نام اشياء (objects) و گزاره ها روي اشياء (به عنوان مثال، خواص اشياء يا ارتباط بين اشياء)، به خوبي استفاده از ربط دهنده ها و سورها (quantifiers)، به جملات اجازه مي‌دهند تا در مورد چيزي در دنيا به سرعت نوشته شوند.

در منطق مرتبه اول گزاره‌اي يک جمله يک حقيقت را بيان مي‌کند و عامل باور دارد که جمله صحيح است، يا جمله نادرست است يا قادر نيست تا از راه ديگري تنيجه‌گيري کند.

سيستم‌هايي که مبتني بر منطق شولا (Fuzzy) هستند، مي‌توانند در جايي از اعتقاد را در يک جمله داشته باشند و هم‌چنين به درجات حقيقت نيز اجاره دهند: يک حقيقت نيازي به درست يا نادرست بودن در دنيا ندارد، اما مي تواند تا يک ميزاني صحت داشته باشد.

**منطق گزاره‌اي: يک منطق بسيار ساده:**

علائم منطق گزاره‌اي:

ثابتهاي منطقي (true, False)

علائم گزاره‌اي: Q, P

رابط‌هاي

پرانتز ()

تمام جملات توسط قرار دادن اين علائم با هم و با استفاده از قوانين زير، ساخته ‌مي‌شوند:

* ثابتهاي منطقي (true, False) خودشان جمله محسوب مي‌شوند.
* علامات گزاره‌اي نظير Q, P هر کدام به تنهايي يک جمله هستند.
* پرانتزهاي اطراف يک عبارت، آن عبارت را تبديل به يک جمله واحد مي‌سازند مثل (P ^ Q).
* يک جمله مي‌تواند توسط ترکيب جملات ساده‌تر با يکي از پنج رابط منطقي ايجاد مي‌شود.

روش رفع ابهام منطق گزاره‌اي بسيار شبيه عبارت رياضي است.

**معاني:**

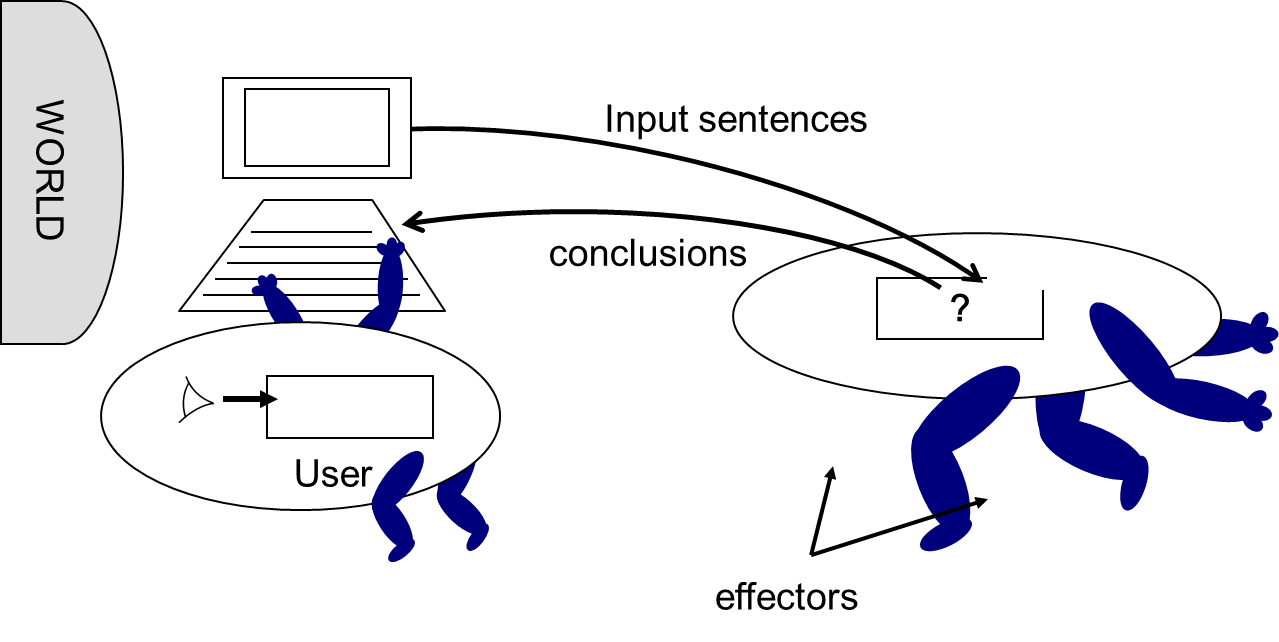
يک سيمبول گزاره‌اي مي‌تواند آنچه که خواست شما است، معني بدهد. يعني اينکه، تفسير آن هر حقيقت اختياري مي‌تواند باشد.

يک جمله پيچيده، معنايي مرکب از معناهاي هر قسمت از جمله را دارد، هر رابط مي‌تواند به عنوان يک تابع تصور شود.

**اعتبار و استنتاج:**

جدول درستي براي تعريف رابطها و براي کنترل جملات معتبر به کار مي‌رود.

ماشين‌ هيچ ايده‌اي از معناي نتايج ندارد، کاربر مي‌تواند نتايج را بخواند و از تفسير خود براي سيمبولهاي گزاره‌اي به معناي نتيجه پي ببرد.

وجود يک يک سيستم استدلال ضروري است تا قادر باشد، نتايجي را استخراج کند که از مقدم‌ها، بدون توجه به دنيا که اولويت رجوع جملات را مشخص مي‌کند، پيروي کنند.

جملات اغلب به دنيايي رجوع مي‌کنند که عامل دسترسي مستقلي به آن نداشته باشد.

**مدل‌ها Models:**

دنيايي که در آن جمله‌اي تحت تفسيري ويژه، درست باشد. يک مدل (Model) از آن جمله ناميده مي‌شود.

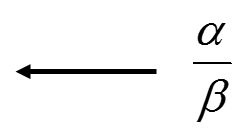
مدل‌ها در منطق بسيار حائز اهميت هستند زيرا، دوباره استلزام را مطرح مي‌کنند، جمله a توسط يک پايگاه دانش KB مستلزم مي‌شود، اگر مدل‌هاي KB تمام مدلهاي a باشند. سپس زماني که KB درست باشد، a نيز درست خواهد بود.

«دنياهاي واقعي» متفاوت بسياري وجود دارند که مقادير درستي مشابهي براي آن سيمبول‌ها دارند. تنها تقاضايي که براي کامل شدن تصفيه لازم است، درستي يا نادرستي هر سيمبول گزاره‌اي در هر دنيا است

**قوانين استنتاج براي منطق گزاره‌اي:**

پردازشي که توسط هر کدام از آنها، صحت يک استنباط از طريق جداول درستي بدست آمده است، مي‌توند به کلاس‌هاي استنتاج‌ها گسترش داده شود.

نمونه‌هاي مطمئني از استنتاج‌ها وجود دارند. که بيشتر و بيش‌تر بوجود مي‌آيند، و صحت آنها مي‌تواند يکبار براي هميشه نشان داده شوند.

زماني که يک قانون پياده شد، مي‌توان به منظور ساخت استنتاج‌ها بدون ساخت جداول درستي، استفاده شود.

اين جمله نيست، اما يک قانون استنتاج است.

يک قانون استنتاج زماني درست است اگر نتيجه آن در تمام موارد درست باشد و مقدم‌ها نيز درست باشند.

يک اثبات منطقي شامل دنباله‌اي از کاربردهاي قوانين استنتاج است که ابتدا با جمله‌هاي موجود در KB آغاز مي‌شود، و منجر به توليد جمله‌اي مي‌شود که اثبات را پايان مي‌دهد.

يکنوايي:

استفاده قوانين استنتاج به منظور يافتن نتيجه از يک پايگاه دانش، به طور صريح مبتني بر خواص عمومي منطقهاي قطعي (شامل گزاره‌اي و منطق مرتبه اول) است که يکنوايي (monotonicity) ناميده مي‌شود.

مي‌توانيم خواص يکنوايي منطق را به طور زير شرح دهيم:



منطق مرتبه اول وگزاره‌اي دراين حالت، يکنوا هستند.

تئوري احتمال، يکنوا نيست.

کلاس مفيدي از جملات براي زماني که رويه استنتاجي با زمان چند جمله‌اي وجود دارد که اين کلاس جملات هورن (Horn sentences) ناميده مي‌شود. يک جمله هورن فرمي به صورت زير دارد:



که Pi و Q اتمهاي خنثي هستند. دو مورد مهم وجود دارد: اول، زماني که Qثابت False است.

ما به جمله‌اي مي‌رسيم که برابر است با:



دوم اينکه، زماني کهn=1 و P1=True ما به True=>Qمي‌رسيم که برابر است با جمله اتمي Q.

منطق گزاره‌اي به ما اجازه مي‌دهد که به تمام نکات مهم درمورد منطق و چگونگي استفاده از آن به منظور ارائه استنتاج که نهايتاً به عمليات تبديل مي‌شود، برسيم. اما منطق گزاره‌اي بسيار ضعيف است.

**مشکل کند شدن رويه استنتاج:**

1) مشکل فقط نوشتن اين قوانين نيست بلکه تعداد زياد آنها، باعث مشکل مي‌شود.

2) مشکل ديگر، روبرو شدن با تغييرات محيط است. ما جزيي از عامل استدلال کننده را در يک مکان و زمان ويژه نشان داديم، و تمام گزاره‌ها در پايگاه دانش در آن زمان خاص، درست بودند. اما در حالت کلي، دنيا هر لحظه در حال تغيير است.

اندازه يک جدول درستي n2 است. که n تعداد سيمبولهاي گزاره‌اي در پايگاه دانش است.

براي اجتناب از سردرگمي، ما به سيمبول‌هاي گزاره‌اي متفاوتي، براي تشخيص مکان عامل در هر مرحله نيازداريم.

1- ما نمي‌دانيم که بازي چه مدت طول خواهد کشيد، بنابراين نمي‌دانيم که چه تعداد از اين گزاره‌هاي وابسته به زمان، نياز داريم.

2- اکنون بايد برگرديم و حالتهاي وابسته به زمان از هر قانون را بنويسيم.

فصل هفتم

**منطق مرتبه اول**

منطق گزاره‌اي هستي شناسي بسيارمحدودي دارد و فقط براي دنيايي که شامل حقايق باشد، تعهد قبول مي‌کند و اين امر بازنمايي مسائل ساده را نيز مشکل ساخته است.

**منطق مرتبه اول First-Order\_logic))** تعهدات هستي شناسانه قوي‌تري را نسبت به منطق گزاره‌اي ايجاد مي‌کند.

**اجزايي که در اين منطق وجود دارند:**

**اشياء(Objects):** مردم، خانه‌ها، اعداد، تئوريها، رنگها، بازيهاي بيس‌بال، جنگلها، کشورها...

**روابط(Relations):** برادرِ، بزرگتر از، داخل، قسمتي از، رنگ ...دارد، بدهکار است، اتفاق افتاد بعد از...

**خواص(Properties):** قرمز، گرد: غيرواقعي، رسمي...

**توابع(Functions):** پدرِ، بهترين دوست، يکي بيشتر ‌از، نوبت سوم...

ما ادعا نمي‌کنيم که دنيا واقعاً از اشياء و روابط بين آنها ساخته شده است، بلکه اين جداسازي به ما کمک مي‌کند با بهتر در مورد دنيا قضاوت کنيم.

* منطق مرتبه اول قادر است تا حقايقي را در مورد تمام اشياء جهان بيان دارد.
* اگرچه منطق مرتبه اول، موجوديت اشياء و روابط آنها را ممکن مي‌سازد، اما هيچ تعهد هستي‌شناسي را براي چيزهايي مثل طبقات، زمان و حوادث قبول نمي‌کند.
* منطق مرتبه اول از اين نظر جهاني است که قادر است تا هر چيزي را که قابل برنامه‌ريزي باشد، بيان کند.

**نحو و معاني:**

منطق مرتبه اول جملاتي دارد، اما همچنين واژه‌هايي term نيز دارد که اشياء را بازنمايي مي‌کنند.

سيمبول‌هاي ثابت، متغيرها و سيمبول‌هاي تابع براي ساخت واژه‌ها استفاده مي‌شوند، و کميت‌سنجها و سيمبولهاي گزاره‌اي براي ساخت جملات به کار برده مي‌شوند.

تعريف دقيق هر عنصر به صورت زير است:

**سيمبولهاي ثابت (Constant Symbols):**

يک تفسير مي‌بايست معين کند که کدام شيء توسط کدام سيمبول ثابت در اشياء ارجاع داده مي‌شود.

هر سيمبول ثابت، دقيقاً به اسم يک شيء نامگذاري مي‌شود، اما تمام اشياء نيازي به داشتن نام ندارند و بعضي از آنها مي‌توانند چند اسم داشته باشند.

**سيمبولهاي گزاره (Predicate Symbols):**

يک تفسير معين مي‌کند که يک سيمبول گزاره به يک رابطه ويژه درمدل رجوع مي‌کند.

**سيمبولهاي تابع (Function Symbols):**

بعضي از روابط تابع هستند، بدين معنا که هر شيئ دقيقاً به شيئ ديگري توسط رابطه رجوع مي‌کند.

انتخاب **ثابت**، **گزاره**، و **سيمبولهاي تابع** به کلي به **کاربرد** بستگي دارد.

**ترم‌ها (Terms):**

يک ترم، يک عبارت منطقي است که به يک شيئ رجوع مي‌کند.

معاني رسمي ترم‌ها بسيار صريح است. تفسير، يک رابطه تابعي ارجاع داده شده توسط سيمبول تابع، و اشياء ارجاع داده شده توسط واژه‌ها را اختصاص مي‌دهد که آرگومان‌هايش هستند. از اين رو، تمام ترم به شيئ رجوع مي‌کند که به عنوان (n+1) امين مدخل در آن tuple در رابطه‌اي که اولين n عنصر آن اشياء ارجاع شده توسط آرگومان‌ها هستند، ظاهر مي‌شود.

**جملات اتمي (Atomic sentences):**

مي‌توانيم با استفاده از ترم‌هايي براي ارجاع به اشياء و گزاره‌هايي براي ارجاع به روابط، جملات اتمي به وجود آوريم، که حقايق را پايه‌گذاري مي‌کنند.

يک جمله اتمي از يک **سيمبول گزاره‌اي** تشکيل يافته و توسط يک **ليست پرانتز از واژه‌ها** دنبال مي‌شود.

يک جمله اتمي درست است **اگر** رابطه ارجاع شده توسط سيمبول گزاره با اشياء ارجاع شده توسط آرگومان‌ها مطابقت داشته باشد.

رابطه در صورتي صحت دارد که tuple اشياء در رابطه باشد.

حقيقت يک جمله بنابراين هم به تفسير و هم به دنيا بستگي دارد.

**جملات پيچيده:**

ما مي‌توانيم از رابط‌هاي منطقي براي تشکيل جملات پيچيده‌تر فقط در محاسبات گزاره‌اي استفاده کنيم.

**معاني** جملات که با استفاده از رابطهاي منطقي فرم گرفته‌اند، ازلحاظ گزاره‌اي با آن يکسان هستند.

**سورها (Quantifires):**

زماني که ما منطقي در اختيار داريم که شامل اشياء است، طبيعي است که ذکر خواص کلي اشياء را بر شمارش اشياء توسط نام ترجيح مي‌دهيم. سورها به ما اجازه اين کار را مي‌دهند.

منطق مرتبه اول دو سور استاندارد دارد:

* عمومي (universal)
* وجودي (existential)

**سور عمومي: (Universal Quantification)**

A

معمولاً به معني «براي تمام» است.

A

شما يک جمله را مي‌توانيد به صورت xP که P يک عبارت منطقي است تصور کنيد. و P معادل با ترکيب عطفي  تمام جملات حاصل شده توسط جانشيني نام يک شيئ براي متغير x هرجا که درP ظاهر شود، است.

**سور وجودي (Existential):**

E

به صورت «وجود دارد...» تلفظ مي‌شود. درحالت کلي xP زماني درست است که P براي بعضي از اشياء در دنيا درست باشد. بنابراين مي‌تواند به عنوان معادلي براي ترکيب فصلي جملات بدست آمده توسط جانشيني اسم يک اشياء براي متغير x، تصور شود.

E

بنابراين، يک جمله شرطي با سور وجودي در دنيايي شامل هر شيئ که مقدم آن ترکيب شرطي نادرست باشد، درست است. از اين رو همچنين جملاتي اصلاً چيزي براي گفتن ندارند.

**سورهاي لانه‌اي (Nested Quantifiers):**

x , y معادل با x وy است

ترتيب سورها بسيار مهم است. اگر ما آنها را در پرانتز قرار دهيم روشن‌تر مي‌شود.

در حالت کلي، x( y P(x,y)) جمله دلخواهي است که شامل x,y مي‌باشد. مي‌گويد که هر شيئي در دنيا يک خاصيت ويژه‌اي دارد، و آن خاصيت به چند شيئي توسط رابطه p مربوط مي‌شود.

از طرف ديگر، x( y P(x,y)) مي‌گويد که در دنيا شيئي وجود دارد که خاصيت ويژه‌اي دارد و خاصيت توسط p به هر شيئي در دنيا مربوط مي‌شود

مشکل اساسي زماني بوجود ميآيد ، که دو سور با يک متغير استفاده مي‌شوند.

قانون اين است که متغير به داخلي‌ترين سور که آن را بيان مي‌کند، پس اين متغير ارتباطي با ديگر سورها نخواهد داشت.

**ارتباط بين و**

در واقع دو سور وجودي و عمومي از طريق تناقض با هم در ارتباط هستند.

بدليل اينکه در واقع رابط عاطفي در دنياي اشياء است و رابط فصلي است، تعجب آور نخواهد بود که آنها از قوانين دمورگان پيروي کنند. قوانين دمورگان در ارتباط با جملات سوري به شرح زير است:



براي اهداف AI، **محتوا** و از اين رو **قابليت خواندن جملات** مهم هستند.

**بنابراين:**

ما هر دو سور را نگه مي‌داريم.

**تساوي (Equality):**

به غير از گزاره‌ها و ترم‌هايي که قبلاً به آنها اشاره مي‌توانيم از سيمبول تساوي (equality symbol) براي ساختن عباراتي که دو ترم به شيئي مشابه رجوع کنند، استفاده مي‌کنيم.

سيمبول تساوي : مي‌تواند به منظور شرح خواص يک تابع داده شده، استفاده شود. اين سمبول هم چنين مي‌تواند با علامت نقيض براي نشان دادن عدم تشابه دو شيئي استفاده شود.

**توسعه‌ها و تمايزات نگارشي:**

سه نوع از روشهاي که روي منطق مرتبه اول اعمال مي‌شود:

1- منطق مرتبه بالاتر

2-1 عبارات تابعي و گزاره‌اي با استفاده از عملگر λ

2-2 سور يکتايي

2-3 عملگر يکتايي

3- انواع علائم

**منطق مرتبه بالاتر:**

* ما را قادر مي‌سازد تا بتوانيم کيفيت روابط و توابع اشياء را به خوبي تعيين کنيم.
* قدرت معنا دارتري نسبت به منطق اول دارد.

**عبارات تابعي و گزاره‌اي با استفاده از عملگر λ :**

* اغلب مفيد است که توابع و گزاره‌هاي پيچيده را از قسمت هاي ساده‌تري تشکيل دهيم.
* عملگر λ مرسوم است که براي اين منظور استفاده شود.
* اين –expression λ مي‌تواند براي آرگومان‌ها نيز به کار برده شود تا به يک ترم منطقي منتهي شود.

براي مثال گزارة «از جنيست متفاوت و از‌ آدرس مشابه هستند.» را مي‌تواند به صورت زير نوشت:



**سور يکتايي:**

راه دقيقي براي گفتن اينکه يک شيئي منحصر به فرد يک گزاره را قانع مي‌کند، وجود ندارد. بعضي از مؤلفان علامت ! x King(x) را استفاده مي‌کنند.

جمله بالا بدين معناست که «يک شيئي منحصر به فرد x وجود دارد که King(x) را قانع مي‌کند «يا غير رسمي تر بگوييم» دقيقاً يک King وجود دارد.

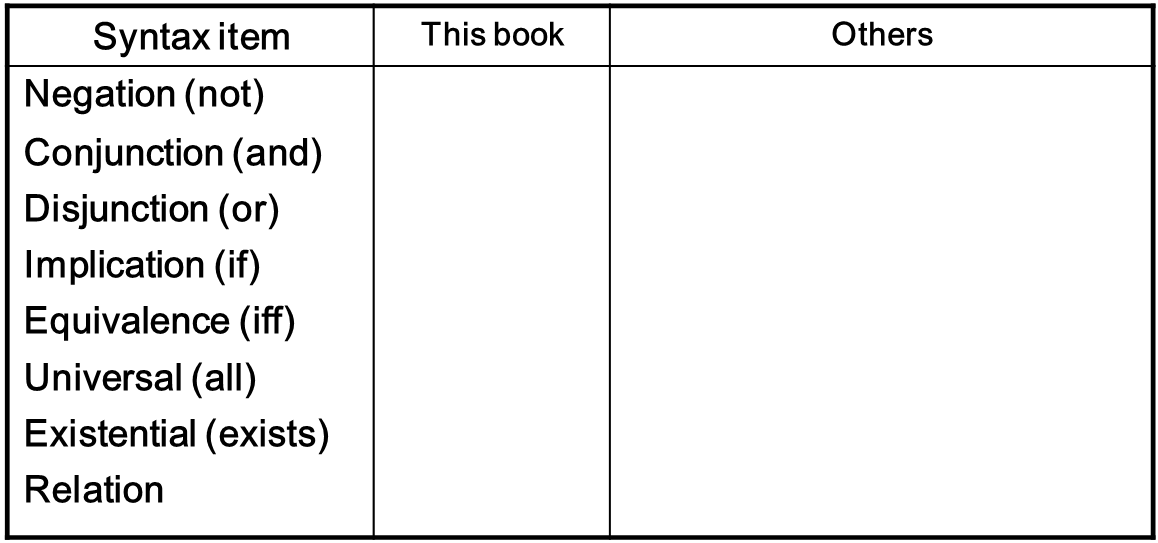
**عملگر يکتايي:**

براي مفهوم يکتايي استفاده مي‌کنيم.

علامتxp(x) ﺎ عموماً براي بازنمايي مستقيم شيئي مورد نظر استفاده مي‌شود.

**انواع علائم:**

تعدادي از علائم رايج در منطق مرتبه اول:



**استفاده از منطق مرتبه اول:**

* دامنه Kinship
* اصل موضوعات، تعاريف و قضايا
* دامنه مجموعه‌ها
* علائم خاص براي مجموعه‌ها، ليست‌ها و محاسبات
* طرح پرسش و گرفتن پاسخ

**عامل‌هاي منطق براي دنياي Wumpus:**

ما معماري سه عامل را در نظر مي‌گيريم:

1. عامل‌هاي (reflex) که فقط ادراکات و عملياتشان رامطابق هم طبقه‌بندي مي‌کنند.
2. عامل‌هاي مبتني بر مدل (model-based) که بازنمايي داخلي از دنيا را تشکيل مي‌دهند و از آن براي عملکردشان استفاده مي‌کنند.
3. عامل‌هاي مبتني بر هدف goal-based که اهداف را صورت مي‌دهند و سعي دارند تا به آنها برسند. (عامل‌هاي مبتني بر هدف معمولاً عامل‌هاي مبتني بر مدل نيز هستند.)

**عامل واکنشي ساده:**

ساده‌ترين نوع ممکن عامل، قوانيني دارد که مستقيماً ادراکات را به عمليات مرتبط مي‌سازد. اين قوانين مشابه واکنش يا غرايز هستند.

**محدوديت‌هاي عامل‌هاي واکنشي ساده:**

* وجود مسائلي که بايد به عامل از طريق بازنمايي دنيا فهمانده شود.
* عامل‌هاي واکنشي نمي‌توانند از حلقه‌هاي نامحدود اجتناب ورزند.

**بازنمايي تغيير در دنيا:**

در طراحي عامل، تمام ادراکات به پايگاه دانش اضافه مي‌شود، و در اصل تاريخچه ادراک تمام آن چيزهايي است که در مورد دنيا بايد دانسته شود. اگر ما قوانيني داشته باشيم که به گذشته به همان خوبي زمان جاري رجوع کنند، مي‌توانيم قابليت‌هاي يک عامل را براي يافتن جايي که عملکرد بهينه دارد، افزايش دهيم.

هر سيستمي که تصميماتي را بر پايه ادراکات گذشته مي‌گيرد، مي‌تواند براي استفاده مجدد از جملاتي در مورد حالت جاري، دوباره نوشته شود، به شرط اينکه اين جملات به محض رسيدن هر درک تازه‌اي و در عمل تازه‌اي که انجام مي‌شود، به روز درآورده شود.

قوانيني که روش‌هايي در آن دنيا مي‌تواند تغيير کند (تغيير نکند) را تعريف مي‌کنند، قوانين diachronic ناميده مي‌شوند که از زبان يوناني به معناي «سرتاسر زمان» برگرفته شده است. بازنمايي تغييرات يکي از مهم‌ترين حيطه‌ها در بازنمايي دانش است.

ساده‌ترين راه براي کنار آمدن با تغييرات، **تغيير پايگاه دانش** است.

يک عامل مي‌تواند در فضاي گذشته و حالات ممکن آينده، به جستجو بپردازد، و هر حالت توسط پايگاه دانش متفاوتي بازنمايي مي‌شود.

در اصل، بازنمايي موقعيت و عمليات تفاوتي با بازنمايي اشياء واقعي يا روابط واقعي ندارد.

ما نياز داريم که در مورد اشياء و روابط مناسب، تصميم‌گيري کنيم و سپس قضايايي در رابطه با آنها بنويسيم.

**محاسبه موقعيت:**

محاسبه موقعيت (Situation Calculus) روش خاصي براي تعريف تغييرات در منطق مرتبه اول است.

تصوري که از دنيا مي‌شود، آن را به صورت دنباله‌اي از موقعيت‌ها در نظر مي‌گيرد، که هر کدام از آنها يک "snapshot" از حالت دنيا است.

**استنتاج خواص پنهاني دنيا:**

زماني که عامل بتواند تشخيص دهد که کجا قرار دارد، مي‌تواند کيفيت‌ها را با محل، به جاي موقعيت تطبيق دهد.

**قوانين همزمان:**

قضايايي را که ما براي تسخير اطلاعات ضروري براي اين استنباط‌ها خواهيم داشت، قوانين همزمان (Synchronic) ناميده مي‌شوند، زيرا آنها خواص حالت يک دنيا را به ديگر خواص حالت دنياي مشابه، مربوط مي‌کنند.

دو نوع اصلي از قوانين همزمان وجود دارند:

**قوانين Causal:**

قوانين سببي جهت مفروض شده علت را در دنيا منعکس مي‌کنند: بعضي از خواص پنهاني دنيا، ادراکات مطمئني را براي توليد شدن باعث مي‌شوند.

**2) قوانين تشخيصي (Diagnostic rules):**

قوانين تشخيصي مستقيماً دلالت بر حضور خواص پنهان شده از اطلاعات مبتني بر ادراک دارند.

اگرچه قوانين تشخيصي به نظر مي‌آيد که اطلاعات مطلوبي را مستقيماً توليد کنند، خيلي حيله‌گيرانه است، اطمينان داشته باشيم که آنها قوي‌ترين نتايج ممکن را از اطلاعات موجود به دست مي‌آورند.

مهم‌ترين مسئله براي به خاطر سپردن اين است که اگر قضايا به درستي و کمال، روش عملکرد دنيا و روشي که ادراکات توليد مي‌شوند را تعريف کنند، رويه استنتاج به درستي قوي‌ترين شرح ممکن از حالت دنيا با ادراکات داده شده را استخراج خواهد کرد.

**اولويت بين عمليات:**

تغييرات عقايد عامل در مورد بعضي از چهره‌هاي دنيا نياز به تغييرات در قوانيني که با ديگر چهره‌ها سروکار دارند، دارد.

* عامل ما به سادگي توسط پرسش براي رسيدن به چيزي متفاوت، مي‌تواند دو مرتبه برنامه‌ريزي شود.
* اهداف، مطلوب بودن حالات حاصل را بدون توجه به روش به دست آمدن آنها توضيح مي‌دهند.

اولين قدم، شرح مطلوبيت خود عمليات (action)، و ترک ماشين براي انتخاب بهترين عمل است.

**از يک مقياس ساده استفاده مي‌کنيم:**

عمليات مي‌توانند عالي، خوب، متوسط ، ريسکي و يا مهلک باشند.

عامل هميشه بايد يک عمل فوق‌العاده‌اي را در صورت يافتن انجام دهد؛ در غير اينصورت، يک عمل خوب در غير اينصورت، يک عمل متوسط، و يک عمل ريسک‌دار اگر تمام قبلي‌ها شکست بخورند.

سيستم مقدار عملياتي:

سيستمي که حاوي قوانيني از اين نوع است يک سيستم مقدار عملياتي (action-value) ناميده مي‌شود.

* توجه کنيد که قوانين به آنچه که واقعاً عمليات انجام مي‌دهند، رجوع نمي‌کنند، فقط به مطلوب بودن آنها توجه دارند.

**به سوي يک عامل هدفدار:**

حضور يک هدف دقيق به عامل اجازه مي‌دهد تا دنباله‌اي از عملياتي که منجر رسيدن به هدف مي‌شوند را پيدا کند.

حداقل سه روش براي يافتن چنين دنباله‌اي وجود دارد:

1. استنتاج
2. جستجو
3. برنامه‌ريزي

**استنتاج:**

نوشتن قضايايي که به ما اجازه ASK از KB را براي دنباله‌اي از عمليات بدهد که ضمانت رسيدن به هدف را به طور امن بکند، چندان مشکل نيست.

**مشکلات اين روش:**

- براي دنياهاي بزرگ، تقاضاهاي محاسباتي بسيار زياد است.

- مشکل تشخيص راه‌حل‌هاي خوب از راه‌حل‌هاي بيهوده وجود دارد.

**جستجو:**

ما مي‌توانيم از رويه جستجوي سطحي براي يافتن مسيري به هدف استفاده کنيم. اين از عامل درخواست مي‌کند تا دانش خود را به صورت مجموعه‌اي از عملگرها درآورد، و بازنمايي حالات را دنبال کند، بنابراين الگوريتم جستجو مي‌تواند به کار برده شود.

**برنامه ريزي:**

شامل استفاده از سيستم‌هاي استدلال خاصي مي‌شود که براي استدلال در مورد عمليات طراحي شده‌اند.

فصل هشتم

**استنتاج در منطق مرتبه اول**

# قوانين استنتاج مربوط به سورها:

قوانين استنتاج براي منطق گزاره‌اي:

Modus Ponens

And – Elimination

And – Introduction

Or – Introduction

Resolution

**سه قانون استنتاجي جديد:**

**1- حذف سور عمومي (Universal Elimination):**

براي هر جمله α متغير v، و ترم زميني g داريم:



**2- حذف سور وجودي:**

براي هر جمله α، متغير v، و سيمبول ثابت k که جاي ديگري از پايگاه دانش ظاهر نشده است، داريم:



**3- (Existential Introduction):**

براي هر جمله α، متغير v که در α واقع نباشد، و ترم زميني g‌که در α واقع نشود داريم:



**مي‌توان اين قوانين را با استفاده از:**

يک جمله با سور عمومي به عنوان ترکيب عطفي تمام مقداردهي‌هاي ممکن آن، و تعريف يک جمله با سور وجودي به عنوان ترکيب فصلي تمام مقداردهي‌هاي ممکن آن، کنترل کرد.

کاربرد قوانين استنتاج، در واقع پرسشي از مطابقت نمونه‌هاي پيش‌فرضيات آنها با جملات موجود در KB و سپس افزودن نمونه‌هاي جديد آنهاست.

اگر ما فرايند يافتن اثبات را به عنوان يک پردازش جستجو فرموله‌سازي کنيم، پس واضح است که اثبات همان راه حل مسئله جستجو است و روشن است که بايد برنامه‌اي هوشمند براي يافتن اثبات بدون دنبال کردن هر گونه مسير نادرست موجود باشد.

# Modus Ponens تعميم يافته:

* فرم Canonical
* يکسان‌سازي (Unificaiton)

# فرم Canonical:

تمام جملات موجود در پايگاه دانش بايد به صورتي باشند که با يکي از پيش‌فرضيات قانون Modus Ponens مطابقت داشته بشاند، فرم Canonical براي Modus Ponens متضمن اين نکته است که هر جمله در پايگاه دانش از چه نوع اتمي يا شرطي با يک ترکيب عطفي از جملات اتمي در طرف چپ و يک اتم منفرد در طرف راست بايد باشد.

ما جملات را به جملات Horn زماني تبديل مي‌کنيم که ابتدا وارد پايگاه دانش، با استفاده از حذف سور وجودي و حذف And شده باشند.

يکسان‌سازي (Unificaiton):

وظيفه روتين يکسان‌ساز Unify، گرفتن دو جمله اتمي p، q و برگرداندن يک جانشين که p، q را مشابه هم خواهد ساخت، است. (اگر چنين جانشيني موجود نباشد، Unify، fail برمي‌گرداند.)



UNIFY، عمومي‌ترين يکسان‌ساز (Most General Unifier) يا (MGU) را برمي‌گرداند، که جانشيني است که کمترين تعهد را در قبل محدودسازي متغيرها دارد.

**زنجيره‌سازي به جلو و عقب (Forward AND Backward Chaining):**

**زنجيره‌سازي به جلو (forward chaining):**

قانون Modus Ponens تعميم يافته به دو صورت استفاده مي‌شود. مي‌توانيم با جملات موجود در پايگاه دانش شروع کنيم و نتايج جديدي را که مي‌توانند استنباط‌هاي بيشتري را بسازند، توليد کنيم. اين روش زنجيره‌سازي به جلو ناميده مي‌شود.

اين روش زماني استفاده مي‌شود که حقيقت جديدي به پايگاه داده ما اضافه شده باشد و خواسته باشيم نتايج آن را توليد کنيم.

**زنجيره‌سازي به عقب (Backward Chaining):**

مي‌توانيم با چيزي که قصد اثباتش را داريم آغاز کنيم و جملات شرطي را پيدا کنيم که به ما اجازه بدهند نتيجه را از آنها استنتاج کنيم، و سپس سعي در ايجاد پيش‌فرضيات آنها داشته باشيم.

اين روش زماني استفاده مي‌شود که هدفي براي اثبات وجود داشته باشد.

**الگوريتم زنجيره‌سازي به جلو:**

زنجيره‌سازي به جلو توسط افزودن يک حقيقت جديد p به پايگاه دانش، فعال مي‌شود و مي‌تواند به عنوان قسمتي از پردازش TELL براي مثال، همکاري داشته باشد. در اينجا ايده، يافتن تمام ترکيبات شرطي است که P را به‌عنوان پيش‌فرض داشته باشد، سپس اگر بقيه پيش‌فرضيات برقرار باشند، مي‌توانيم نتيجه ترکيب شرطي را به پايگاه دانش توسط راه‌اندازي استنتاج‌هاي بعدي اضافه کنيم.

ما به ايدة ترکيب (Composition) جانشيني نيز نياز داريم.

جانشيني است که اثر آن با اثر اعمال هر جانشيني به نوبت، برابر است. زيرا:



زنجيره‌سازي به جلو، تصويري تدريجي از شرايط در حالي که داده‌هاي جديد وارد مي‌شوند، مي‌سازد.

پردازش‌هاي استنتاجي آن مستقيماً با حل مسئله ويژه در ارتباط نيستند،

به همين دليل روية data-driven يا data-directed ناميده مي‌شود.

**الگوريتم زنجيره‌سازي به عقب:**

زنجيره‌سازي به عقب به منظور يافتن تمام پاسخ‌ها براي سؤال طرح شده، به وجود آمده است. بنابراين زنجيره‌سازي به عقب، وظيفه‌اي که از رويه ASK خواسته شده را انجام مي‌دهد. الگوريتم زنجيره‌سازي به عقب BACK-CHAIN ابتدا توسط کنترل درمي‌يابد که آيا پاسخ‌ها مستقيماً از جملات پايگاه دانش، توليد مي‌شوند يا خير. سپس تمام ترکيبات شرطي که نتايجشان با پرسش (query) مطابقت دارد را پيدا مي‌کند و سعي دارد تا پيش‌فرض‌هاي آن ترکيبات شرطي را توسط زنجيره‌سازي به عقب ايجادکند.

اگر پيش‌فرض، يک ترکيب عطفي باشد، سپس BACK-CHAIN ترکيبات عطفي را عطف به عطف پردازش مي‌کند، تا يکسان‌ساز را براي تمام پيش‌فرض بسازد.

کامل بودن Completeness:

تصور کنيد که ما پايگاه دانش زير را در اختيار داريم:



سپس ما مي‌خواهيم که S(A) را نتيجه بگيريم، S(A) درست است،

اگر Q(A)‌يا R(A) درست باشد، و يکي از آنها بايد درست باشد زيرا:

يا P(A) يا P(A) ¬ درست است.

متأسفانه، زنجيره‌سازي با Modus Ponens نمي‌تواند S(A) را نتيجه بگيرد.

مشکل اين است که  نمي‌تواند به صورت Horn دربيايد، و از اين رو توسط Modus Ponens نمي‌تواند استفاده شود.

اين بدان معني است که رويه اثباتي که از Modus Ponens استفاده مي‌کند ناکامل (incomplete) است:

جملاتي که در پايگاه دانش مستلزم شده‌اند ولي رويه نمي‌تواند آنها را استنتاج کند.

پرسش در مورد وجود رويه‌هاي اثبات کامل بحثي است که ارتباط مستقيم با رياضيات دارد. اگر يک رويه اثبات کامل بتواند براي عبارات رياضي پيدا شود، دو چيز دنبال مي‌شود:

* تمام مفروضات مي‌توانند به طور مکانيکي ايجاد شوند.
* تمام رياضيات مي‌توانند به عنوان نتيجة منطقي مجموعه‌اي از اصل موضوع‌هاي پايه‌اي ايجاد شوند.

يک رويه اثبات کامل براي منطق مرتبه اول ارزش بسياري در AI دارد:

* نظريه‌هاي عملي در رابطه با پيچيدگي کامپيوتري.
* فعال ساختن يک ماشين براي حل هر گونه مسئله که در زبان مي‌تواند قرار داده شود.

قضيه کامل بودن گودل نشان داد که، براي منطق مرتبه اول، هر جمله‌اي که توسط مجموعه جملات ديگري مستلزم شود مي‌تواند از آن مجموعه اثبات شود. بنابراين مي‌توانيم قوانين استنتاجي را که به يک رويه اثبات کامل R اجازه مي‌دهد، پيدا کنيم:



اين قضيه کامل بودن مشابه اين است که بگوييم رويه‌اي براي يافتن سوزني در يک پشته کاه وجود دارد و اين ادعاي بيهوده نيست زيرا جملات با سود عمومي و سيمبول‌هاي تابع لانه‌اي دلخواهي در پشته‌هاي کاه با اندازه نامحدود، استفاده مي‌شوند.

گودل نشان داد که رويه اثباتي وجود دارد اما هيچ رويه‌اي را ذکر نکرد.

استلزام در منطق مرتبه اول، نيمه تصميم‌پذير (Semidecidable) بنابراين مي‌توانيم نشان دهيم که جملات از پيش‌فرضيات تبعيت مي‌کنند، اما هميشه نمي‌توانيم نشان دهيم که آنها از پيش‌فرضيات تبعيت نمي‌کنند.

به‌عنوان يک فرضيه، سازگاري (consistency) مجموعه جملات (سؤالي در مورد وجود راه‌حلي براي تبديل تمام جملات به جملات درست) نيز نيمه تصميم‌پذير است.

**Resolution: يک رويه استنتاج کامل**

از دو ترکيب شرطي مي‌توانيم ترکيب سومي را مشتق کنيم که پيش‌فرض اولي را به نتيجه دومي متصل مي‌کند.

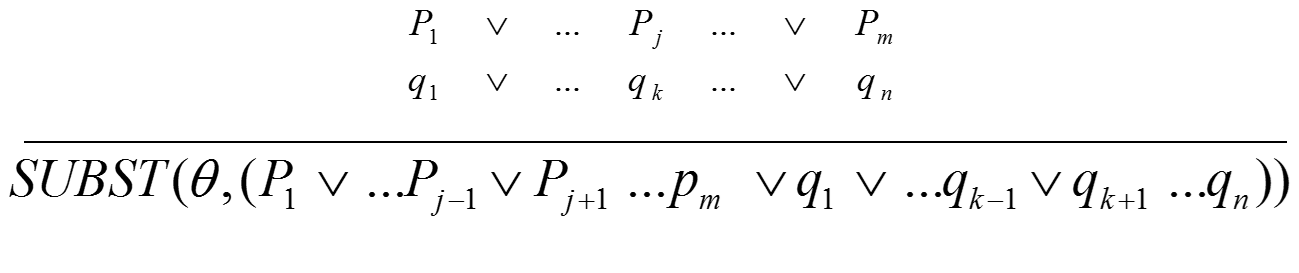
Modus Ponens به ما اجازه استخراج ترکيب شرطي جديد را نمي‌دهد و فقط نتايج اتمي را استخراج مي‌کند. از اين رو قانون resolution قدرتمندتر از Modus Ponens است.

**قانون استنتاج resolution:**

در فرم ساده قانون resolution، پيش‌فرضيات داراي دقيقاً دو ترکيب فصلي هستند. ما مي‌توانيم اين قانون را براي دو ترکيب فصلي به هر طولي وسعت بخشيم، که اگر يکي از قسمت‌هاي ترکيب فصلي در يکclause(Pj) با نقيض قسمت ديگر ترکيب فصلي (qk) يکسان باشند، سپس ترکيب فصلي از تمام قسمت‌ها استنتاج مي‌شود بغير از آن دو:

Resolution تعميم يافته (ترکيبات فصلي)

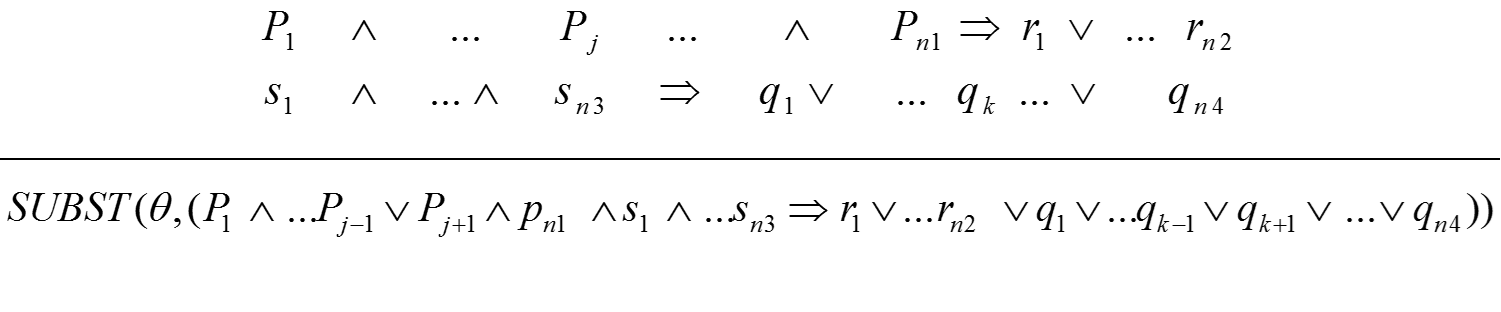
Resolution تعميمي يافته (ترکيبات شرطي)

Resolution‌تعميم يافته (ترکيبات فصلي):

براي Pi و qi فرضي که UNIFY (Pj ¬ qk)=θ

معادلاً، مي‌توانيم اين عبارت را به صورت ترکيب شرطي بنويسيم.

Resolution تعميم يافته (ترکيبات شرطي):

براي اتم‌هاي Pi و qi و ri و si که

UNIFY (Pj , qk)=θ

**فرم‌هاي Canonical براي resolution:**

**در نسخه اوليه قانون resolution،** هر جمله يک ترکيب فصلي از حروف فرضي است.

تمام ترکيبات فصلي در KB فرض شده‌اند که در يک ترکيب عطفي صريح (مانند يک KB‌معمولي) به هم متصل شده‌اند، بنابراين اين فرم، فرم نرمال عطفي Conjunctive normal form (CNF) ناميده مي‌شود.

اگرچه هر جمله به تنهايي يک ترکيب فصلي است.

**در صورت ثانويه قانون resolution،** هر جمله يک ترکيب شرطي با يک ترکيب عطفي از اتم‌ها در سمت چپ و يک ترکيب فصلي از اتم‌ها در طرف راست است.

اين حالت، فرم نرمال شرطي(implicative normal form (INF) ناميده مي‌شود.

هر مجموعه از جملات مي‌توانند به دو فرم ترجمه شوند. فرم نرمال عطفي رايج‌تر است، اما فرم نرمال شرطي طبيعي‌تر به نظر مي‌آيد.

Resolution تعميمي از Modus Ponens است.

فرم نرمال شرطي رايج‌تر از فرم Horn است، به دليل اينکه طرف سمت راست مي‌تواند يک ترکيب شرطي باشد و نه فقط يک اتم تنها.

Modus Ponens قابليت ترکيب اتم‌ها با يک ترکيب شرطي را به منظور استخراج نتيجه به صورتي دارد که resolution قادر به انجام آن نيست.

زنجيره‌سازي با resolution قدرتمندتر از زنجيره‌سازي با Modus Ponens است، اما هنوز کامل نيست.

برهان خلف:

رويه استنتاج کاملي که از resolution استفاده مي‌کند برهان خلف (refutation) ناميده مي‌شود و هم‌چنين به عنوان اثبات توسط تناقض (proof by contradiction) و (reduction and absurdum شناخته شده است.

**تبديل به فرم نرمال:**

* هر جمله مرتبه اولي مي‌تواند به صورت فرم نرمال شرطي (يا عطفي) دربيايد.
* از يک مجموعه از جملات به فرم نرمال مي‌توانيم اثبات کنيم که يک جمله نرمال از مجموعه پيروي خواهد کرد.

**رويه‌اي براي تبديل به فرم نرمال:**

1) حذف ترکيب شرطي:

مي‌توان تمام ترکيبات شرطي را با معادل فصلي جايگزين نمود.

2) حذف ¬:

نقيض فقط براي فرم نرمال عطفي مجاز است، و براي تمام فرم‌هاي نرمال شرطي قدغن است.

3) استاندارد کردن متغيرها:

اين عمل بعداً از ايجاد ابهام زمان حذف سورها جلوگيري مي‌کند.

4) انتقال سورها به سمت چپ:

5) Skolemize:

Skolemization پردازشي است که در آن تمام سورهاي وجودي حذف مي‌شوند.

6) توزيع Λ بر ν :

7) ترکيبات فصلي و عطفي لانه‌اي مسطح شده:

در اين مورد، جمله به فرم نرمال عطفي (CNF)‌است.

8) تبديل ترکيبات فصلي به ترکيب شرطي:

**برخورد با مسئله تساوي:**

يکسان‌سازي يک تست نحوي مبتني بر ظاهر ترم‌هاي آرگوماني است و تست صحيح معنايي مبتني بر اشيايي که نمايش مي‌دهند، نيست.

دو روش براي انجام اين امر:

**1) بديهي نمودن تساوي به وسيله ذکر خواص آن:**

بايد ذکر شود که تساوي، انعطاف‌پذير، متقارن و (متعدي) است.

**2) استفاده از يک قانون استنتاج از يک قانون استنتاج:**

مي‌توانيم قانون استنتاج را به صورت زير تعريف کنيم:

Demodulation: براي تمام ترم‌هاي z,y,x که UNIFY (x,y) = θ



**استراتژي‌هاي Resolution:**

4 استراتژي که براي راهنمايي جستجو به سمت يک اثبات استفاده مي‌شوند، را بررسي خواهيم کرد:

1. **Unit preference:**

در اينجا ما سعي بر توليد جمله کوتاهي به صورت True => False‌داريم.

اين استراتژي يک کشف‌کننده مفيد است که مي‌تواند با ديگر استراتژي‌ها ترکيب شود.

**2) مجموعه Support**

هر resolution جمله‌اي را از مجموعه Support با جمله ديگري ترکيب مي‌کند و نتيجه را به مجموعه Support اضافه مي‌کند. اگر مجموعه Support به نسبت تمام پايگاه دانش کوچک باشد، فضاي جستجو را قطع خواهد کرد.

يک انتخاب بد براي مجموعه Support الگوريتم را ناکامل خواهد ساخت.

استراتژي مجموعه Support داراي اين مزيت است که درخت‌هاي اثباتي توليد مي‌کند که اغلب براي درک افراد آسان هستند، زيرا آنها هدف‌گرا هستند.

**3) Resolution‌ورودي:**

در استراتژي resolution ورودي هر resolution يکي از جملات ورودي را (از KB يا query) با جمله ديگر ترکيب مي‌کند.

در پايگاه‌هاي دانش Horn، Modus Ponens نوعي از استراتژي resolution ورودي است، زيرا يک ترکيب شرطي از KB اصلي را با ديگر جملات ترکيب مي‌کند. از اين رو شگفتي‌آور نخواهد بود که resolution ورودي براي پايگاه‌هاي دانشي که به صورت Horn هستند، کامل باشد اما در حالت کلي ناکامل است.

**4) Subsumption:**

متد Subsumption تمام جملاتي که توسط يک جمله موجود در KB، Subsume مي‌شوند، را حذف مي‌کند.

Subsumption به نگهداري KB به صورت کوچک کمک مي‌کند، و در نتيجه فضاي جستجو را کوچک مي‌سازد.

فصل نهم

**برنامه‌ريزي**

تفاوت عامل برنامه‌ريزي با عامل حل مسئله در سه چيز است:

بازنمايي اهداف، حالات و عمليات

استفاده از بازنمايي‌هاي منطقي و صريح برنامه‌ريز را قادر مي‌سازد تا سنجش عامل را معقولانه هدايت کند.

عامل برنامه‌ريزي همچنين در روش بازنمايي و جستجو براي راه‌حل‌ها نيز تفاوت دارد.

**يک عامل ساده برنامه‌ريزي:**

زماني که حالت دنيا قابل دسترسي است، عامل مي‌تواند از ادراکات توليد شده توسط محيط استفاده کرده و مدل کامل و صحيحي از حالت دنياي جاري بسازد. سپس، با داشتن هدف، مي‌تواند الگوريتم برنامه‌ريزي مناسبي را براي توليد برنامه عمل فراخواني کند. عامل سپس مي‌تواند در طي مراحل برنامه، هر لحظه يک عمل را اجرا کند.

عامل با محيط از طريق يک روش حداقل در عمل است و از ادراکاتش براي شرح حالت اوليه استفاده مي‌کند و از اين رو هدف اوليه را دنبال مي‌کند؛ اما به سادگي توانسته مراحل برنامه را تشکيل بدهد.

**از حل مسئله به برنامه‌ريزي:**

**برنامه‌ريزي** و **حل مسئله** موضعات متفاوتي هستند زيرا در بازنمايي اهداف و حالات و عمليات و هم چنين بازنمايي ساختار دنباله‌هاي عملياتي متفاوت عمل مي‌کنند.

**عناصر اوليه يک حل مسئله مبتني بر جستجو:**

* بازنمايي عمليات.
* بازنمايي حالات.
* بازنمايي اهداف.
* بازنمايي برنامه‌ها.

**بازنمايي عمليات:**

عمليات توسط برنامه‌هايي که شرح حالت مابعد را توليد مي‌کنند، تعريف مي‌شود.

**بازنمايي حالات:**

در حل مسئله، شرح کامل حالت اوليه داده شده است و عمليات توسط برنامه‌اي که شرح کامل حالت را توليد مي‌کنند، بازنمايي مي‌شوند.

**بنابراين:**

تمام بازنمايي‌هاي حالت، کامل هستند.

**بازنمايي اهداف:**

تنها دانشي که عامل در مورد هدف در اختيار دارد، **تست هدف** و **تابع کشف‌کننده** است. هر دو اينها بر روي حالت‌ها اعمال مي‌شوند تا مطلوبيت آنها مورد ارزيابي قرار گيرد.

**بازنمايي برنامه‌ها:**

در حل مسئله يک راه‌حل دنباله‌اي از عمليات است. در طول تشکيل راه‌حل‌ها، الگوريتم‌هاي جستجو فقط دنباله‌هاي پيوسته عمليات را که از حالت اوليه آغاز مي‌شوند (يا در مورد جستجوي دوطرفه، خاتمه دادن به حالت هدف) در نظر مي‌گيرند.

حال ببينيم چطور اين تصميمات بر روي قابليت عامل تأثير مي‌گذارند، تا مسئله ساده زير را حل کنند:

«يک ليتر شير و يک خوشه موز و يک مته چندسرعته را بخر.»

حالت اوليه: عامل در خانه است اما بدون هيچ يک از اشياء موردنظر.

عملگر: تمام کارهايي که عامل قادر به انجام آن است.

تابع کشف‌کننده: تعداد چيزهايي که هنوز به دست آورده نشده‌اند.

**اولين ايده کليدي در وراي برنامه‌ريزي:**

«بسط دادن» بازنمايي حالات، اهداف و عمليات است. الگوريتم‌هاي برنامه‌ريزي از تعاريفي به زبان‌هاي رسمي استفاده مي‌کنند که معمولاً منطق مرتبه اول و يا زيرمجموعه‌اي از آن است.

حالات و اهداف توسط مجموعه‌هايي از جملات بازنمايي مي‌شوند و عمليات توسط شرح پيش‌شرط‌ها و تأثيرات منطقي بازنمايي مي‌شوند که برنامه‌ريزي را قادر مي‌سازد تا ارتباطات بين حالات و عمليات را هدايت کند.

**دومين ايده کليدي در وراي برنامه‌ريزي:**

اين است که برنامه‌ريز آزاد است تا عمليات را به برنامه هر زمان که لازم باشد، اضافه کند. هرچند که دنباله افزايشي در حالت اوليه وجود داشته باشد.

هيچ الزامي بر وجود ارتباط بين مرتبه برنامه ريزي و مرتبه اجرا نيست. با ساختن تصميمات «مشخص» و «مهم» در ابتدا، برنامه‌ريزي مي‌تواند فاکتور انشعاب را براي انتخاب‌هاي بعدي و نياز به پي‌جويي به عقب را براي تصميمات اختياري کاهش دهد.

**سومين ايده کليدي در وراي برنامه‌ريزي:**

اين است که بيشتر بخش‌هاي دنيا مستقل از ديگر بخش‌ها هستند.و اين امر داشتن يک هدف عطفي را ممکن مي‌سازد و مي‌توان آن را با يک استراتژي تقسيم و غلبه حل نمود.

الگوريتم‌هاي تقسيم و غلبه مؤثر هستند؛ زيرا تقريباً هميشه حل چندين زيرمسئله کوچک آسان‌تر از يک مسئله بزرگ است. بهر حال تقسيم و غلبه در مواردي که هزينه ترکيب راه‌حل‌هاي زيرمسائل زياد باشد، با شکست مواجه مي‌شود. بسياري از معماها داراي اين خاصيت هستند.

دليل اينکه معماها «گول‌زننده» هستند، اين است که قرار دادن زيربرنامه‌ها کنار هم کار دشواري است.

فصل دهم

**عدم قطعيت**

**مسئله‌اي که با منطق مرتبه اول و بنابراين با رهيافت عامل منطق‌گرا وجود دارد اين است که:**

**عامل‌ها اغلب هيچگاه دسترسي کامل به تمام حقيقت درباره محيط خود را ندارند.**

سؤالات بسيار مهمي وجود دارند که عامل نمي‌تواند پاسخ طبقه‌بندي شده به آن را بيابد. بنابراين بايد تحت **عدم قطعيت (uncertainity)** عمل کند.

**عدم قطعيت به علت کامل نبودن**، و

**عدم صحت درک عامل از خواص محيط** ناشي مي‌شود.

**مسئله کيفيت:**

قوانين بسياري در دامنه کامل نيستند؛ زيرا:

1) شرايط بسيار زيادي بايد دقيقاً شمارش شوند،

يا

2) برخي از شرايط ناشناخته هستند.

**برخورد با دانش غيرقطعي:**

ابزار اصلي ما براي کنار آمدن با درجات باور، تئوري احتمالات خواهد بود که درجه عددي از باور را بين 0 و 1 به جملات اختصاص مي‌دهد.

**احتمالات** روشي از خلاصه‌سازي عدم قطعيت را به وجود مي‌آورد که از تنبلي و جهل ما ناشي مي‌شوند.

احتمال 0 براي باور مبهمي که داراي جملات نادرست است، و

احتمال 1 براي باور مبهمي که داراي جمله درست است، تخصيص داده مي‌شود.

**جمله** در حقيقت خودش هم درست و هم نادرست است.

مهم است توجه داشته باشيم که **درجه باور** با **درجه درستي** متفاوت است.

تئوري احتمالات تعهد ontological را همانند منطق ايجاد مي‌کند، که حقايق در دنيا هم وجود دارند و هم ندارند.

**درجه درستي که با درجه باور در تضاد است**، موضوع منطق فازي است.

در منطق مرتبه اول و گزاره‌اي، جمله بسته به تعبير و دنيا، درست يا نادرست خواهد بود و زماني درست است که حقيقتي را که به آن رجوع مي‌کند، موضوع اصلي باشد.

عبارات احتمالي کاملاً مشابه اين نوع معناها نيستند. به آن علت است که احتمالاتي که عامل به يک گزاره تخصيص مي‌دهد به ادراکاتي که تا آن لحظه دريافت کرده است بستگي دارد.

در بحث استدلال غيرقطعي، ما آن را **شاهد (evidence)** مي ناميم.

همانطور که وضعيت استلزام زماني که جملات بيشتري به پايگاه دانش اضافه مي‌شوند تغيير مي‌کند، احتمالات نيز در صورت وجود شواهد بيشتر، تغيير خواهند کرد.

تمام عبارات احتمالي بايد شواهدي را با توجه به اينکه کدام احتمال تشخيص داده شده است، تعيين کنند. همانگونه که عامل ادراکات جديدي را دريافت مي‌کند، ارزيابي‌هاي احتمالي به منظور انعکاس شاهد جديدي، به روز درآورده مي‌شوند.

**عدم قطعيت و تصميمات عقلاني:**

حضور عدم قطعيت روش‌هاي تصميم‌گيري عامل را تغيير داده است. عامل منطقي عموماً هدف واحدي دارد و هر برنامه‌اي که امکان رسيدن به آن قطعي است را اجرا مي‌کند. يک عمل مي‌تواند انتخاب و يا رد شود، چه به هدف برسد و چه نرسد و بدون توجه به آنچه که ديگر عمليات انجام مي‌دهند.

**تئوري سودمندي:**

اين تئوري اين گونه بيان مي‌شود:

هر وضعيت درجه‌اي از فايده يا سودمندي را براي يک عامل دارد وعامل به حالاتي با سودمندي بالاتر رجوع خواهد کرد.

سودمندي يک حالت به عاملي وابسته است که مفروضاتش توسط تابع سودمندي بازنمايي شده است.

تئوري سودمندي رعايت حال ديگران را نيز مي‌کند. براي يک عامل کاملاً منطقي است که سودمندي بالاتر را به وضعيتي اختصاص دهد که عامل خودش از آن رنج ولي ديگران منفعت مي‌برند.

مفروضات که به عنوان سودمندي‌ها، مطرح شدند با احتمالات در تئوري عمومي تصميمات عقلاني که تئوري تصميم‌گيري ناميده مي‌شود، ترکيب مي‌شوند:

**Dicision theory=probalility+utility theory**

ايده اساسي در مورد تئوري تصميم‌گيري اين است که يک عامل منطقي است

**اگر و فقط اگر**

عملي را که منتهي به بالاترين سودمندي مي‌شود، انتخاب کند.

اين اصل سودمندي مورد انتظار ماکزيمم(MEU) ناميده مي‌شود.

احتمالات و سودمندي‌ها در ارزيابي يک عمل توسط توزين سودمندي يک نتيجه ويژه و با احتمالي که پديد آورده است، ترکيب مي‌شوند.

**طراحي براي يک عامل تصميم‌گيري نظري:**

ساختار عاملي که از تئوري تصميم‌گيري براي انتخاب عمليات استفاده مي‌کند، در سطح انتزاعي با عامل منطقي يکسان است.