



دانشکده مهندسی کامپیوتر

طراحی و پیاده سازی رابط گرافیکی کاربر برای نرم افزار کلیپس با ویژگی تطبیق پذیر بودن

پروژه پایانی برای دریافت درجه کارشناسی در رشته مهندسی کامپیوتر

علی اسدی

استاد راهنما

دکتر محمد رضا کنگاوری

پاییز ۱۴۰۲

الحمد لله رب العالمين
والصلاة والسلام على
سيدنا محمد وآله الطيبين
الطاهرين

MRTsoft

تأییدیه ی هیأت داوران جلسه ی دفاع از پایان نامه

نام دانشکده: دانشکده مهندسی کامپیوتر

نام دانشجو: علی اسدی

عنوان پایان نامه: طراحی و پیاده سازی رابط گرافیکی کاربر برای نرم افزار کلیپس با ویژگی تطبیق پذیر بودن

تاریخ دفاع: پاییز ۱۴۰۲

رشته: مهندسی کامپیوتر

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه	امضا
۱					
۲					
۳					

تأییدیه ی صحت و اصالت نتایج

باسمه تعالی

اینجانب علی اسدی به شماره دانشجویی ۹۶۵۲۱۰۳۸ دانشجوی رشته مهندسی کامپیوتر مقطع تحصیلی کارشناسی تأیید مینمایم که کلیه ی نتایج این پایان نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده ام. در صورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب مینمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی صلاح (اعم از اداری و قضایی) به عهده ی اینجانب خواهد بود و دانشگاه هیچ گونه مسؤولیتی در این خصوص نخواهد داشت

نام و نام خانوادگی: علی اسدی

تاریخ و امضا

مجوز بهره برداری از پایان نامه

بهره برداری از این پایان نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین میشود، بلامانع است:

- بهره برداری از این پایان نامه برای همگان بلامانع است.
- بهره برداری از این پایان نامه با اخذ مجوز از استاد راهنما، بلامانع است.
- بهره برداری از این پایان نامه تا تاریخ ممنوع است.

استاد راهنما

تاریخ:

امضا:

تقدیم به

پدر و مادرم

تشکر و قدردانی

سپاس خداوندگار حکیم را که با لطف بی کران خود، آدمی را زیور عقل آراست

در آغاز وظیفه خود می دانم از زحمات بی دریغ استاد راهنمای خود، جناب آقای دکتر کنگاوری صمیمانه

تشکر و قدردانی کنم که قطعاً بدون راهنمایی های ارزنده ایشان، این مجموعه به انجام نمیرسید

در پایان، بوسه میزنم بر دستان خداوندگاران مهر و مهربانی پدر و مادر عزیزم و بعد از خدا، ستایش میکنم وجود

مقدس شان را و تشکر می کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که

بهترین پشتیبان من بودند.

علی اسدی

پاییز ۱۴۰۲

چکیده

کلیپس CLIPS یک محیط نرم افزاری برای تولید سیستم های خبره است که فاقد یک رابط گرافیکی بر خط است. کلیپس یک زبان برنامه نویسی توصیفی است که برخلاف زبان های رویه ای که در آن ها باید روند و چگونگی کار برای سیستم بیان شود تا سیستم ساخته شود، در این زبانها تعریف صورت مسئله درسیستم، منجر به حل مسئله می گردد

هدف این پروژه طراحی و تولید یک رابط کاربری گرافیکی برای کلیپس است که استفاده از کلیپس را برای طراحان و کاربران تسهیل می نماید.

برای پیاده سازی یک برنامه خبره نیاز به یک سری اطلاعات داریم که چگونگی پیاده سازی این سیستم را برای ما تشریح کند. این اطلاعات توسط تحقیق و پژوهش از فرد خبره در آن زمینه به دست می آید که قوانین ما را تشکیل می دهند

پس از آن که این اطلاعات را کسب کردیم نیاز به دسته بندی و ایجاد یک درخت تصمیم برای سیستم خبره خود داریم. این درخت می تواند همان سناریو سیستم شما باشد که نحوه کسب دانش و واکنش ها به دانش ها را مشخص می کند.

برای تولید این رابط کاربری از زبان برنامه نویسی جاوا اسکریپت و کتابخانه react استفاده شده که ورودی ها را از کاربر گرفته و پس از پردازش آن ها در نهایت خروجی دو فایل به زبان های پایتون (.py) و کلیپس (.clp) بوده که کاربران می توانند از آن ها در محیط پایتون و برنامه کلیپس استفاده کنند. برای تولید فایل پایتون از api کتابخانه کلیپس clipspy استفاده شده و امکان استفاده از زبان کلیپس در پایتون را به ما می دهد.

فهرست مطالب

فصل ۱: مقدمه.....	۱۱
فصل ۲: سیستم خبره.....	۱۳
فصل ۳: برنامه کلیپس.....	۳۹
فصل ۴: رابط کاربری محاوره ای گرافیکی برای CLIPS.....	۵۶
فصل ۵: پیاده سازی یک سناریو.....	۶۴
فصل ۶: جمع بندی.....	۷۶
فصل ۷: منابع و مراجع.....	۷۸

فصل ۱

مقدمه

تا ابتدای دهه ۱۹۸۰ (م) کار چندانی در زمینه ساخت و ایجاد سامانه‌های خبره توسط پژوهش گران هوش مصنوعی صورت نگرفته بود. از آن زمان به بعد، کارهای زیادی در این راستا و در دو حوزه متفاوت ولی مرتبط سامانه‌های کوچک خبره و نیز سامانه‌های بزرگ خبره انجام شده است.

در دهه ۱۹۷۰، ادوارد فیگن بام در دانشگاه استنفورد به دنبال کشف روش حل مسئله ای بود که خیلی کلی و همه منظوره نباشد. پژوهشگران دریافتند که یک متخصص معمولاً دارای شماری رموز و فوت و فن خاص برای کار خود می‌باشد و در واقع از مجموعه‌ای از شگردهای سودمند و قواعد سرانگشتی در کار خود بهره می‌برد، این یافته مقدمه پیدایش سامانه خبره بود. سامانه خبره با برگرفتن این قواعد سر انگشتی از متخصصین و به تعبیری با تبدیل فرایند استدلال و تصمیم‌گیری متخصصین به برنامه‌های رایانه‌ای می‌تواند به عنوان ابزار راهنمای تصمیم‌گیری در اختیار غیرمتخصص و حتی متخصصین کم تجربه قرار گیرد.

فیگن بام توضیح داد که جهان از پردازش داده‌ها به «پردازش دانش» در حال حرکت است، انتقالی که با فناوری پردازنده‌های جدید و معماری‌های کامپیوتری امکان‌پذیر شده است. سیستم‌های خبره نقش زیادی در بسیاری از صنایع از جمله خدمات مالی، مخابرات، مراقبت‌های بهداشتی، خدمات مشتری، حمل و نقل، بازی‌های ویدئویی، تولید، حمل و نقل هوایی و ارتباطات نوشتاری ایفا کرده‌اند.

هوش مصنوعی: هوش مصنوعی روشی است در جهت هوشمند کردن رایانه تا قادر باشد در هر لحظه تصمیم‌گیری کرده و اقدام به بررسی یک مسئله نماید. هوش مصنوعی، رایانه را قادر به اندیشیدن می‌کند و روش آموختن انسان را رونوشت برداری می‌نماید؛ بنابراین اقدام به جذب اطلاعات جدید جهت به‌کارگیری در مراحل بعدی می‌پردازد.

مغز انسان به بخش‌هایی تقسیم شده است که هر بخش وظیفه خاص خود را جدا از بقیه انجام می‌دهد. آشفستگی در کار یک بخش تأثیری در دیگر بخش‌های مغز نخواهد گذاشت. در برنامه‌های هوش مصنوعی نیز این مسئله رعایت می‌شود درحالی که در برنامه‌های غیر هوش مصنوعی مثل C یا Pascal تغییر در برنامه روی سایر قسمت‌های برنامه و اطلاعات تأثیر دارد.

مباحث کاربردی و مهم در تحقق یک سامانه هوش مصنوعی:

سامانه‌های خبره (Expert Systems)

شبکه‌های عصبی (Neural Network)

الگوریتم‌های ژنتیک (Genetic Algorithms)

سامانه‌های منطق فازی (Fuzzy Logic Systems)

فصل ۲

سیستم خبره

«سیستم خبره» (Expert System) یکی از حوزه‌های مهم «هوش مصنوعی» (Artificial Intelligence) تلقی می‌شود. این نوع سیستم‌ها در حل مسائلی کاربرد دارند که به دانش تخصصی و استنتاج منطقی بر اساس داده‌ها و تجربه‌های پیشین نیازمند هستند. در مطلب حاضر به این پرسش پاسخ داده می‌شود که سیستم خبره چیست و از چه اجزایی تشکیل شده است. همچنین، در ادامه به کاربردها، ویژگی‌ها، مزایا و معایب این نوع سیستم‌ها پرداخته خواهد شد.

در ابتدا توسط محققان دانشگاه استنفورد معرفی شد و برای حل مسائل پیچیده در یک حوزه خاص توسعه یافت.

دانش مربوطه را از پایگاه دانش خود به دست می‌آورد و آن را بر اساس مشکل کاربر را تفسیر می‌کند. داده‌های موجود در پایگاه دانش اساساً توسط انسان‌هایی که در یک حوزه خاص متخصص هستند، اضافه می‌شوند. با این حال، این نرم افزار توسط افراد غیر متخصص برای به دست آوردن اطلاعات استفاده می‌شود. در تشخیص های پزشکی مختلف، حسابداری، کدنویسی، بازی و موارد دیگر استفاده می‌شود.

سیستم های خبره قادر به انجام تعدادی از اقدامات هستند، از جمله:

۱. مشاوره دادن
۲. کمک در تصمیم گیری انسانی
۳. تظاهرات و دستورالعمل
۴. استخراج راه حل ها
۵. تشخیص
۶. تفسیر ورودی ها و ارائه خروجی های مرتبط
۷. پیش بینی نتایج
۸. توجیه نتیجه گیری
۹. پیشنهادهایی برای راه حل های جایگزین برای یک مشکل

سیستم خبره چیست ؟

سیستم خبره، برنامه‌ای کامپیوتری است که به منظور حل مسائل پیچیده و گرفتن تصمیمات مختلف طراحی می‌شود. سیستم های خبره در راستای حل چالش‌ها، اطلاعاتی را از داده‌های موجود استخراج می‌کنند و با استدلال و استنتاج و بر اساس «پُرس‌مان» (کوئری) کاربر، به نتیجه‌گیری می‌پردازند.

سیستم های خبره بخشی از حوزه هوش مصنوعی هستند. نخستین پژوهش این حوزه در سال ۱۹۷۰ انجام شد و هدف آن طراحی سیستمی بود که بتواند بر پایه اطلاعات حقیقی و احتمالات، همانند انسان خبره به حل مسائل در حوزه‌ای خاص بپردازد.

سیستم خبره یک برنامه کامپیوتری هوشمند است که از دانش و روش های استنتاج برای حل مسائلی استفاده می‌کند که به دلیل مشکل بودن نیاز به تجربه و مهارت انسان دارد. این سیستم یک واقعیت (Fact) را از بیرون از سیستم دریافت می‌کند و با توجه به آن واقعیت با پاسخ و راه حل مناسب (خبرگی) را به عنوان خروجی می‌دهد. این سیستم در حالت کلی از ۲ قسمت تشکیل شده است:

پایگاه دانش Knowledge Base

موتور استنتاج Inference Engine

واقعیت fact

واقعیات همان اطلاعاتی است که به عنوان ورودی به سیستم خبره داده می‌شود.

قواعد rule

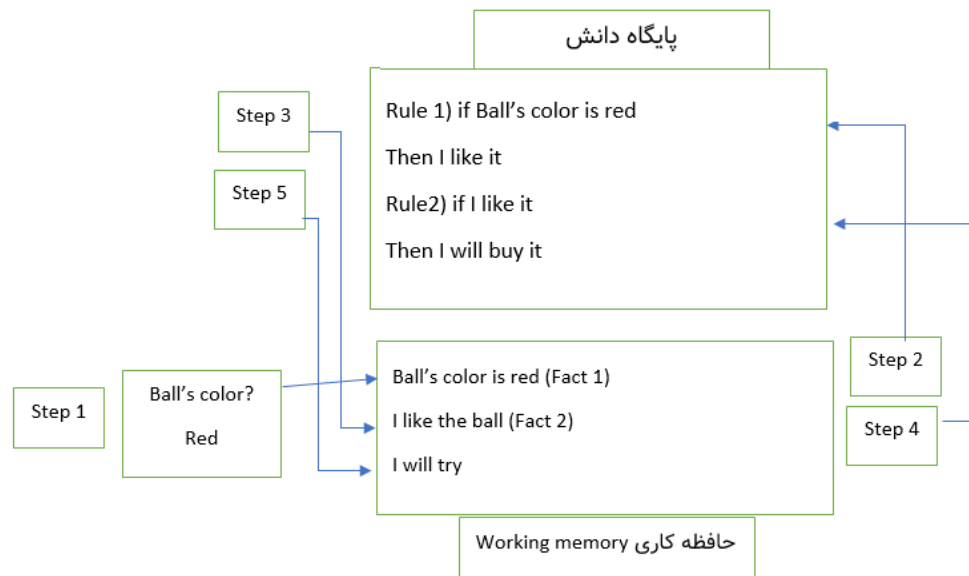
ساختار دانشی است که یعنی اطلاعات دانسته به اطلاعات دیگری می‌تواند به معلومات قبلی اضافه و یا دانسته فرض شوند. ساختار قاعده یا چند پیش فرض در قسمت "اگر" را با نتیجه‌گیری ها در قسمت "آنگاه" مرتبط می‌سازد. یک قاعده می‌تواند "در غیر این صورت" هم داشته باشد.

در واقع قاعده ساختاری است که با برقرار شدن قسمت الگو "pattern" قسمت عمل "action" اجرا می‌شود که اگر این عمل خودش معادل یک الگو دیگر شود عمل مربوط به آن اجرا می‌شود و به همین ترتیب تا آخر ادامه پیدا می‌کند.

If pattern

Then action

Else action



شکل ۱- مثال

در این مثال واقعیات (Facts) در حافظه کاری و قواعد (Rules) در پایگاه دانش قرار دارند. قاعده Rule1 اجرا می شود و قسمت الگوی آن Fact 1 تطبیق دارد یعنی (Ball's Color Is Red) پس چون قسمت الگو با Fact1 تطبیق داشت قسمت عمل Rule1 یعنی I like It انجام می شود. سپس Rule2 اجرا می شود چون الگوی آن با Fact 2 که این قسمت توسط عمل Rule 1 ساخته شده است تطبیق دارد قسمت عمل Rule2 یعنی Try will Buy It اجرا می شود.

سیستم های خبره دارای ویژگی هایی هستند که در ادامه به آن ها اشاره شده است:

- **کارایی بالا:** سیستم های خبره را می توان برای حل مسائل مختلف به کار برد و از نتیجه گیری و استدلال منطقی آن برای تصمیم گیری های مهم استفاده کرد.
- **ارائه نتایج قابل فهم:** سیستم خبره می تواند با زبان انسان با کاربر ارتباط برقرار کند و خروجی را نیز به زبان قابل درک انسان ارائه دهد.

- **ارائه نتایج معتبر:** از آنجا که سیستم های خبره بر اساس واقعیت ها، تجربه ها و استنتاج های منطقی گذشته به تحلیل مسائل می پردازند، نتایج و خروجی هایی را ارائه می دهند که تا حد زیادی دقیق و کارآمد هستند.
- **ارائه پاسخ در زمان کوتاه:** سیستم های خبره می توانند در کوتاه ترین زمان ممکن به حل پیچیده ترین مسائل بپردازند.

اجزای سیستم های خبره چیست ؟

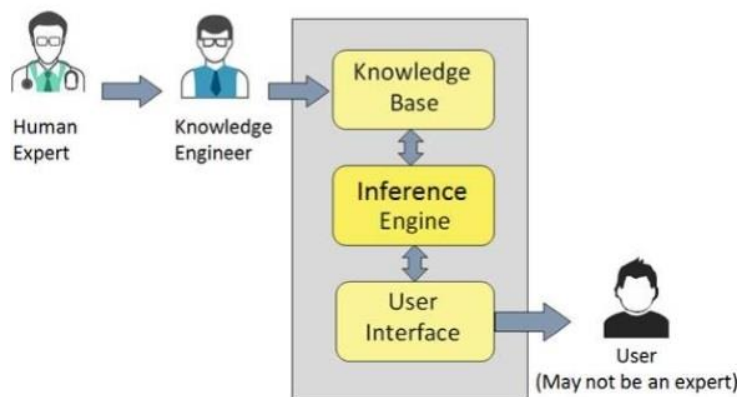
سیستم های خبره از سه جزء اصلی تشکیل شده اند که در ادامه فهرستی از آنها ارائه شده است.

۱. پایگاه دانش (Knowledge Base)

۲. رابط کاربری (User Interface)

۳. موتور استنتاج (Inference Engine)

در ادامه، پیش از آن که به توضیح اجزای سیستم های خبره پرداخته شود، مفهوم دانش را شرح می دهیم، زیرا این نوع سیستم ها بر مبنای دانش به تصمیم گیری و استنتاج می پردازند.



شکل ۲- اجزای سیستم های خبره

دانش چیست ؟

سیستم خبره بر اساس دانش موجود در پایگاه دانش خود، به تحلیل مسائل می پردازد. دانش مجموعه ای از داده های حقیقی و تجربه های حاصل شده از استنتاج های سیستم های خبره پیرامون موضوعی خاص است که در مواقع حل مسائل، مورد بررسی قرار می گیرند.

به منظور ذخیره دانش در پایگاه دانش از قالب دستورات شرطی «اگر ... آنگاه ... در غیر این صورت ... (IF ... THEN ... ELSE) استفاده می‌شود.

میزان موفقیت سیستم‌های خبره در حل مسائل، تا حد زیادی به کیفیت، جامع بودن و صحیح بودن دانش موجود در پایگاه داده بستگی دارد. به منظور تایید صحت اطلاعات پایگاه دانش، چندین متخصص، پژوهش‌گر و مهندس دانش آن‌ها را بررسی کرده و در نهایت مهندس دانش، اطلاعات را در قالبی مشخص در پایگاه دانش ذخیره می‌کند.

پایگاه دانش در سیستم‌های خبره چیست ؟

در سیستم‌های خبره، پایگاه دانش به عنوان حافظه‌ای محسوب می‌شود که دانش‌های استنتاج شده از سیستم‌های خبره مختلف را در خود نگهداری می‌کند. هر چقدر میزان اطلاعات موجود در این پایگاه‌های دانش بیشتر باشد، سیستم‌های خبره با دقت بیشتری درباره مسائل مختلف تصمیم می‌گیرند.

مؤلفه های پایگاه دانش

پایگاه دانش یک ES ذخیره ای از دانش واقعی و اکتشافی است.

۱. دانش واقعی - اطلاعاتی است که به طور گسترده توسط مهندسان دانش و محققان در حوزه وظیفه پذیرفته شده است.

۲. دانش اکتشافی - در مورد تمرین، قضاوت دقیق، توانایی فرد در ارزیابی و حدس زدن است.

رابط کاربری در سیستم خبره چیست ؟

یکی از اجزای سیستم‌های خبره، رابط کاربری است که به منظور تعامل با کاربر و دریافت پرسش‌های آن‌ها در قالبی مشخص طراحی می‌شوند. رابط کاربری پس از دریافت کوئری‌ها، آن‌ها را به موتور استنتاج ارسال می‌کند. در نهایت، موتور استنتاج پاسخ خود را به رابط کاربری می‌فرستد تا آن را به عنوان خروجی به کاربر نمایش دهد.

بدین ترتیب، می‌توان به‌طور کلی بیان کرد که رابط کاربری به کاربران مبتدی و غیرحرفه‌ای کمک می‌کند تا به منظور یافتن حل مسئله، با سیستم خبره ارتباط برقرار کنند. در طراحی رابط کاربری سیستم‌های خبره، از روش‌های «پردازش زبان طبیعی» (Natural Language Processing | NLP) استفاده می‌شود تا سیستم بتواند درخواست کاربر را درک کند.

موتور استنتاج در سیستم خبره

موتور استنتاج به عنوان مغز سیستم‌های خبره محسوب می‌شود و وظیفه پردازش اصلی سیستم را بر عهده دارد. موتور استنتاج از قوانین استنتاجی استفاده می‌کند تا با استخراج دانش از پایگاه دانش، درباره مسئله‌ای تصمیم بگیرد یا به اطلاعات جدیدی دست یابد.

سیستم‌های خبره جدید از الگوریتم‌های «یادگیری عمیق» (Deep Learning) و «مدل‌های یادگیری ماشین» (machine Learning) بهره گرفته‌اند تا در حل مسائل، رفتار و داوری انسان هوشمند را شبیه‌سازی کنند. با دریافت تجربه‌های بیشتر، سیستم‌های خبره عملکرد خود را بهبود می‌بخشند.

دو نوع موتور استنتاج در سیستم‌های خبره استفاده می‌شوند که در ادامه به آن‌ها اشاره شده است:

۱. موتورهای استنتاج قطعی: (Deterministic Inference Engine) در این نوع از موتورهای استنتاج، فرض بر این است که استنباط‌های موتور براساس قواعد و واقعیت‌ها صورت می‌گیرد و استنتاج نهایی موتور، دقیق است.

۲. موتورهای استنتاج احتمالاتی (Probabilistic Inference Engine): این نوع از موتورهای استنتاجی، درباره مسائل مختلف بر پایه احتمالات نتیجه‌گیری می‌کنند و نتایج قطعی ارائه نمی‌دهند.

رویکرد حل مسئله در سیستم‌های خبره

موتور استنتاج از دو روش برای استخراج اطلاعات از پایگاه دانش و یافتن راه‌حل برای مسئله استفاده می‌کنند.

این دو روش در ادامه فهرست شده‌اند:

۱. زنجیرسازی رو به جلو (Forward Chaining)

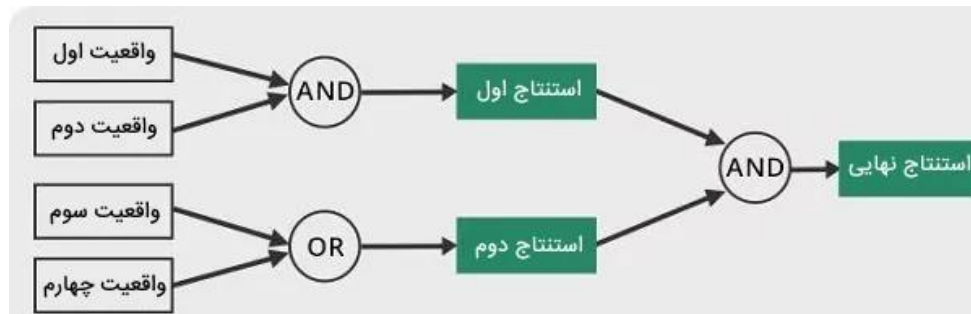
۲. زنجیرسازی رو به عقب (Backward Chaining)

در ادامه مطلب، به توضیح هر یک از رویکردهای ذکر شده در بالا پرداخته می‌شود.

حل مسئله با رویکرد زنجیرسازی رو به جلو چیست؟

موتور استنتاج در روش زنجیرسازی رو به جلو، بر اساس قواعد و واقعیت مشخص شروع به استدلال می‌کند و تا در نهایت نتیجه‌گیری خود را به واقعیات اضافه کند. به عبارتی، این نوع موتورها، زنجیره‌ای از شرط‌ها را دنبال می‌کنند تا در نهایت به نتیجه‌گیری بپردازند. در چنین روشی، موتور استنتاج در هر مرحله به دنبال پاسخ چنین

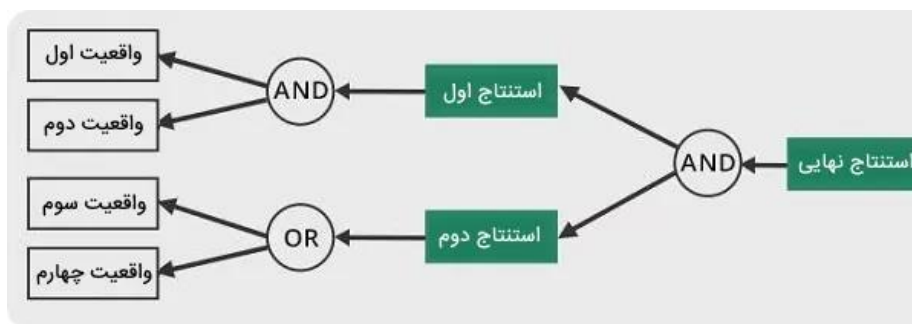
پرسشی است که «در گام بعدی چه اتفاقی خواهد افتاد». در تصویر زیر، روال تصمیم‌گیری موتورهای استنتاج بر پایه روش زنجیرسازی رو به جلو دیده می‌شود.



شکل ۳- رویکرد زنجیرسازی رو به جلو

حل مسئله با رویکرد زنجیرسازی رو به عقب

در روش استنتاجی زنجیرسازی رو به عقب، موتور جستجو کار تحلیل خود را از انتها (هدف) آغاز می‌کند تا به اثبات واقعیات شناخته شده برسد. به عبارتی، در این روش، موتور استنتاج به دنبال یافتن پاسخ چنین پرسشی است که «چرا این اتفاق افتاد». در پی پاسخ به این پرسش، موتور استنتاج به بررسی شرایط پیشین می‌پردازد که منجر به خروجی فعلی شده‌اند. تشخیص سرطان خون را می‌توان به عنوان مثالی در نظر گرفت که برای تشخیص دلایل آن، از روش زنجیرسازی رو به عقب استفاده می‌شود. در تصویر زیر، روال حل مسئله با رویکرد زنجیرسازی رو به عقب نمایش داده شده است.



شکل ۴- رویکرد زنجیرسازی رو به عقب

چه افرادی در توسعه سیستم‌های خبره مشارکت دارند ؟

افرادی که با سیستم‌های خبره کار می‌کنند و در توسعه آن‌ها مشارکت دارند، در فهرست زیر قرار می‌گیرند:

- **متخصصان حوزه خاص:** موفقیت سیستم‌های خبره در حل مسائل، تا حد زیادی به دانش گردآوری شده در پایگاه دانش وابسته است. به منظور تهیه دانش مورد نیاز این سیستم‌ها باید از متخصصان حوزه‌های مختلف کمک گرفت تا اطلاعات معتبری برای تهیه پایگاه دانش فراهم شود.
- **مهندس دانش:** افرادی که به عنوان مهندس دانش فعالیت می‌کنند، دانش معتبر تهیه شده توسط متخصصان حوزه‌های مختلف را در قالبی خاص فراهم می‌کنند تا برای سیستم خبره قابل فهم باشند.
- **کاربر نهایی:** کاربر نهایی، فردی است که برای حل مسائل مختلف، از سیستم‌های خبره کمک می‌گیرد. چنین فردی، الزاماً تخصص خاصی در زمینه‌های علمی ندارد و تنها به منظور یافتن پاسخ پرسش خود، با سیستم‌های خبره کار می‌کند.

انواع سیستم‌های خبره چیست ؟

سیستم‌های خبره را می‌توان به ۶ نوع تقسیم کرد که در ادامه به توضیح هر یک از آن‌ها پرداخته شده است:

- سیستم‌های خبره قاعده‌مند (Rule Based Expert Systems)
- سیستم‌های خبره فازی (Fuzzy Expert Systems)
- سیستم‌های خبره مبتنی بر قاب (Frame Based Expert Systems)
- سیستم‌های خبره ترکیبی (Hybrid Expert Systems)
- سیستم‌های خبره عصبی (Neural Expert Systems)
- سیستم‌های خبره فازی - عصبی (Neuro - Fuzzy Expert Systems)

در ادامه مطلب، به توضیح ویژگی‌های هر یک از انواع سیستم‌های خبره پرداخته می‌شود.

سیستم خبره قاعده مند چیست ؟

سیستم خبره قاعده‌مند به عنوان اولین نوع از سیستم‌های خبره شناخته می‌شود. این نوع از سیستم‌های خبره شامل مجموعه‌ای از قواعد شرطی هستند که داده‌ها را می‌توان در قالب این دستورات شرطی قرار داد تا موتور استنتاج بر اساس این قواعد، به نتیجه‌گیری بپردازد.

سیستم‌های خبره فازی چه هستند؟

چنانچه داده‌های درون پایگاه دانش عباراتی را شامل شوند که مفاهیم آن‌ها دارای ابهام باشند، از سیستم‌های خبره فازی برای استنتاج استفاده می‌شود که بر اساس نظریه فازی پیاده‌سازی شده‌اند. عبارت‌هایی نظیر «بسیار بلند» یا «بسیار سبک» دارای مفاهیمی هستند که مقدار دقیق کمیت آن‌ها مشخص نیست. بدین ترتیب، برای استنتاج از چنین داده‌هایی باید از سیستم‌های خبره فازی استفاده شود.

سیستم خبره مبتنی بر قاب چیست ؟

در سیستم‌های خبره مبتنی بر قاب ، از مفهوم «قاب (Frame)» به منظور نگهداری دانش استفاده می‌شود. قاب را می‌توان به عنوان «ساختمان داده (Data Structure)» تلقی کرد که به یک شیء یا مفهوم اشاره دارد. هر قاب دارای نام و مجموعه‌ای از ویژگی‌ها است که هر کدام از آن‌ها، دارای مقادیر خاصی هستند.

در تصویر زیر، دو قاب با نام‌های شخص و کامپیوتر ملاحظه می‌شود که هر کدام از این قاب‌ها دارای ویژگی‌های مختلفی هستند. مقادیر هر یک از مشخصه‌های قاب‌ها می‌توانند شامل مقادیر پیش‌فرض، اشاره‌گر به سایر قاب‌ها و مجموعه‌ای از قواعد باشند.

شخص	کامپیوتر
نام	مدل
سن	میزان حافظه رم
قد	مدل پردازنده
وزن	مدل کارت گرافیک
ملیت	رنگ
وضعیت تاهل	وزن

شکل ۵- مثالی از سیستم خبره مبتنی بر قاب

از قاب به منظور سازمان‌دهی دانش موجود در پایگاه دانش استفاده می‌شود. از این نوع ساختار می‌توان نیز به راحتی می‌توان در شیء گرابی استفاده کرد.

سیستم خبره ترکیبی چیست ؟

در طراحی سیستم‌های خبره ترکیبی یا هیبریدی از مزیت‌های سیستم‌های خبره قاعده‌مند، فازی و سیستم خبره مبتنی بر قاب استفاده شده است. با توجه به نوع طراحی سیستم‌های خبره ترکیبی، می‌توان آن‌ها را به دو دسته سیستم‌های خبره عصبی و سیستم‌های خبره عصبی - فازی تقسیم کرد.

- سیستم خبره عصبی: در طراحی این نوع از سیستم‌های خبره، از ساختار سیستم‌های خبره قاعده‌مند و شبکه عصبی استفاده شده است. در این نوع از سیستم‌ها، دانش‌های مورد نیاز موتور استنتاج، در قالب وزن‌های شبکه عصبی ذخیره می‌شوند.
- سیستم‌های خبره عصبی فازی: در طراحی این نوع از سیستم‌های خبره، ویژگی‌های سیستم‌های خبره فازی و شبکه عصبی مصنوعی به کار رفته است.

مراحل توسعه سیستم خبره چیست ؟

مراحل توسعه و ارائه سیستم‌های خبره را می‌توان در ۶ گام خلاصه کرد که در ادامه به توضیح هر یک از این گام‌ها پرداخته شده است:

شناسایی حوزه مسئله:

- مسئله تعریف شده باید جزء مسائلی باشد که بتوان آن را با استفاده از سیستم‌های خبره حل کرد.
- مشورت گرفتن از متخصص برای مسئله مطرح شده انجام می‌شود.
- ارائه مدلی برای طراحی سیستم‌های خبره که به لحاظ هزینه به صرفه است.

طراحی سیستم خبره

- شناسایی ابزارها و فناوری‌های توسعه سیستم‌های خبره صورت می‌گیرد.
- شناسایی روش‌های استفاده از سیستم‌های خبره در سایر سیستم‌های سخت‌افزاری یا شیوه استفاده از سایر پایگاه‌های داده و بانک‌های اطلاعاتی به عنوان پایگاه دانش برای سیستم خبره انجام می‌شود.
- شناختن مفاهیم و دانش‌های تخصصی برای مسئله مطرح شده در این مرحله ضرورت دارد.

توسعه نمونه اولیه از سیستم‌های خبره

- همکاری با متخصصان مربوط به مسئله برای جمع‌آوری دانش لازم برای پایگاه دانش انجام می‌شود.
- آماده‌سازی دانش در قالب قواعد شرطی و ذخیره‌سازی آن‌ها در پایگاه دانش صورت می‌گیرد.

آزمایش و اصلاح نمونه اولیه سیستم‌های خبره

- تست از نمونه اولیه سیستم‌های خبره توسط مهندس دانش به منظور شناسایی خطاها انجام می‌شود.

- تست از نمونه اولیه سیستم خبره توسط کاربر نهایی اجرا می‌شود.

توسعه و تکمیل سیستم خبره

- تکمیل سیستم‌های خبره و تست نهایی آن برای بررسی صحت عملکرد اجزای سیستم صورت می‌گیرد.
- مستندسازی و تهیه گزارش نهایی از پروژه نیز در این مرحله انجام می‌شود.
- آموزش به کاربر نهایی برای استفاده از سیستم‌های خبره باید انجام شود.

نگهداری سیستم‌های خبره

- به‌روزرسانی دانش پایگاه دانش را انجام می‌دهند.
- تهیه رابط‌های ارتباطی جدید برای سیستم‌های مختلف صورت می‌پذیرد.

فناوری‌های لازم برای توسعه سیستم‌های خبره

به منظور طراحی سیستم خبره از چندین تکنولوژی استفاده می‌شود که در ادامه به آن‌ها می‌پردازیم.

- محیط توسعه (IDE) برای ساخت سیستم‌های خبره
 - ابزارها و سخت‌افزارهای لازم برای توسعه سیستم‌های خبره:
 - کامپیوتر، ریز کامپیوتر و MainFrame ها
- زبان‌های برنامه نویسی نظیر LISP و PROLOG
 - پایگاه‌های داده بزرگ
 - ابزارهای رفع خطا و ویرایشگر قوی
- شل (Shell)
 - شل سیستم خبره جاوا (Java Expert System Shell | JESS)
 - شل Vidwan

چرا از سیستم خبره استفاده می‌کنیم؟

پیش از این که از هر گونه فناوری جدید استفاده کنیم، باید به این موضوع پردازیم که چرا چنین فناوری‌های جدیدی ارائه شده‌اند و چه نیازی وجود داشته است که علیرغم وجود پژوهشگران، دانشمندان و افراد خبره در حوزه‌های مختلف علوم، سیستم‌های کامپیوتری جدیدی برای حل مسائل طراحی می‌شوند؟

در ادامه، به مهم‌ترین دلایل ساخت سیستم‌های خبره اشاره شده است:

- عدم محدودیت حافظه: حجم زیادی از داده‌های مورد نیاز سیستم‌های کامپیوتری را می‌توان در حافظه ذخیره و به‌طور کامل و دقیق این اطلاعات را بازیابی کرد. انسان در مقایسه با کامپیوتر برای به خاطر سپردن حجم عظیمی از اطلاعات، دارای محدودیت است و به راحتی قادر نیست در هر زمان تمامی اطلاعات را به‌طور کامل به یاد آورد.
- بازدهی بالا: سیستم‌های مبتنی بر کامپیوتر می‌توانند در کوتاه‌ترین زمان به حل پیچیده‌ترین مسائل بپردازند. همچنین، می‌توان با به‌روزرسانی مکرر داده‌های (دانش) آن‌ها، میزان دقت خروجی آن‌ها را بالا برد.
- حل مسائل تخصصی: متخصصان زیادی در حوزه‌های تخصصی مختلف مشغول به کار هستند که دانش و مهارت‌های متفاوتی دارند. با این حال، برای حل مسئله‌ای پیچیده، نیاز است از چندین فرد متخصص کمک گرفته شود تا درباره موضوع مطرح شده، از دیدگاه‌ها و رویکردهای متفاوت به تجزیه و تحلیل بپردازند. این امر، نیازمند زمان طولانی است. سیستم‌های خبره می‌توانند با ترکیب دانش هر یک از حوزه‌های تخصصی و تحلیل مسائل از جنبه‌های مختلف، در زمان کوتاه، به نتیجه‌گیری موثری با دقت بالا برسند.
- ارائه خروجی بدون لحاظ کردن احساسات: سیستم‌های خبره و کلیه سیستم‌های کامپیوتری در زمان تصمیم‌گیری پیرامون موضوعی خاص، تحت تاثیر عواطف و احساسات نیستند. همچنین، انسان ممکن است به دلیل خستگی یا بیماری نتواند برای حل مسائل، تصمیم درستی بگیرد. سیستم‌های کامپیوتری فارغ از چنین عواملی به‌صورت شبانه‌روزی با بالاترین دقت، مسائل را حل می‌کنند.
- به‌روزرسانی پایگاه دانش: چنانچه خروجی‌های سیستم خبره از دقت قابل قبولی برخوردار نباشند، می‌توان با به‌روزرسانی پایگاه دانش، عملکرد سیستم را بهبود بخشید.

تفاوت سیستم‌های خبره و هوش مصنوعی

سیستم خبره به عنوان یکی از حوزه‌های اصلی هوش مصنوعی محسوب می‌شود. هدف هوش مصنوعی، شبیه‌سازی هوش انسان در تفکر، احساسات و یادگیری است و سیستم‌های خبره به عنوان یکی از نخستین پژوهش‌هایی به حساب می‌آیند که هدف حوزه هوش مصنوعی را محقق کرد. تفاوت جزئی‌تر این دو حیطه پژوهشی را می‌توان در ادامه ملاحظه کرد:

تمرکز هوش مصنوعی بر روی توانمند ساختن ماشین و برنامه‌های کامپیوتری است تا بتوانند مشابه انسان هوشمند، رفتار کنند. سیستم‌های خبره می‌توانند به عنوان روشی برای محقق ساختن هدف هوش مصنوعی در نظر گرفته شوند.

هوش مصنوعی شامل روش‌هایی بر پایه عملکرد هوشمندانه بشر برای حل مسائل مختلف است، در حالی که سیستم خبره به عنوان برنامه‌های کامپیوتری تلقی می‌شود که به منظور حل مسائل مختلف مورد استفاده قرار می‌گیرد.

زیر بخش‌های هوش مصنوعی را می‌توان به پردازش تصویر، پردازش زبان طبیعی، یادگیری ماشین، یادگیری عمیق و سیستم‌های خبره تقسیم کرد. زیر بخش‌های سیستم‌های خبره عبارت‌اند از:

رابط کاربری

موتور استنتاج

پایگاه دانش

از آنجایی که هوش مصنوعی مفهوم جامع‌تری را در بر می‌گیرد، کاربرد سیستم‌های کامپیوتری هوشمند را می‌توان در حوزه‌های گسترده‌ای نظیر تشخیص چهره، تشخیص اشیا در تصاویر، ترجمه ماشینی، تبدیل متن به گفتار و گفتار به متن و سایر موارد ملاحظه کرد. سیستم‌های خبره به عنوان زیر شاخه هوش مصنوعی تلقی می‌شوند و صرفاً از آن‌ها می‌توان برای مشورت گرفتن برای یافتن پاسخ مسائل و تصمیم‌گیری استفاده کرد.

کاربردهای سیستم‌های خبره چیست ؟

از سیستم‌های خبره به منظور گرفتن مشاوره برای یافتن پاسخ مسائل استفاده می‌شود. امروزه، شاهد کاربرد این سیستم‌ها در حوزه‌های مختلفی هستیم که در ادامه به برخی از آن‌ها اشاره شده است:

- کاربرد سیستم‌های خبره در طراحی و ساخت قطعات سخت‌افزاری: امروزه، از سیستم‌های خبره به‌طور گسترده در طراحی و تولید دستگاه‌های مختلفی نظیر لنزهای دوربین و اتومبیل‌های خودران استفاده می‌شود.
- کاربرد سیستم‌های خبره در حوزه مالی: به منظور تشخیص کلاهبرداری و فعالیت‌های مشکوک مالی می‌توان از سیستم خبره استفاده کرد. همچنین، مدیران بانک می‌توانند از این سیستم‌ها به منظور

تخصیص وام‌های کلان بانکی به درخواست‌کنندگان بهره‌گیرند و با توجه به خروجی سیستم های خبره تصمیم بگیرند به چه کسانی وام تعلق گیرد.

- کاربرد سیستم های خبره در حوزه پزشکی: یکی از مهم‌ترین کاربردهای سیستم های خبره در مسائل پزشکی برای تشخیص بیماری مریضان و تجویز دارو و پیشنهاد روش درمان است.
- کاربرد سیستم های خبره در سازمان‌ها: مدیران سازمان‌ها در راستای اتخاذ تصمیمات مهم شرکت برای سوددهی بیشتر، می‌توانند از نتایج سیستم های خبره استفاده کنند تا در مسیر تحقق اهداف سازمان، متحمل کم‌ترین خطر و اشتباه شوند.
- کاربرد سیستم های خبره در مدیریت زمان و برنامه‌ریزی: یکی دیگر از کاربردهای وسیع سیستم های خبره در برنامه‌ریزی زمانی است که از آن‌ها در برنامه‌ریزی خطوط هوایی و آژانس‌های هواپیمایی استفاده می‌شود.

مثال هایی از سیستم خبره

برنامه‌های نرم‌افزاری مختلفی بر پایه سیستم های خبره طراحی شده‌اند که در ادامه به برخی از آن‌ها اشاره می‌شود:

سیستم MYCIN: این سیستم، یکی از سیستم های خبره اولیه بود که برای طراحی آن از رویکرد زنجیرسازی رو به عقب استفاده شده است. این سیستم می‌تواند باکتری‌های مختلفی را شناسایی کند که باعث عفونت شدید می‌شوند. به علاوه، این سیستم می‌تواند بر اساس وزن اشخاص، داروهای مختلفی را برای درمان بیماری آن‌ها پیشنهاد دهد.

سیستم DENDRAL: این سیستم خبره مبتنی بر روش‌های هوش مصنوعی است و از آن برای تحلیل مسائل شیمی استفاده می‌شود. به عبارتی، این سیستم می‌تواند با استفاده از دانش مربوط به طیف‌شناسی ماده‌های مخالف، ساختار مولکولی آن‌ها را پیش‌بینی کند.

سیستم PXDES: از این سیستم برای تشخیص نوع و میزان وخیم بودن سرطان ریه بیماران استفاده می‌شود.

سیستم CaDet: از این نوع سیستم به منظور تشخیص سرطان در مراحل اولیه آن استفاده می‌شود.

R1/XCON: این ES توانایی انتخاب نرم افزار خاصی را برای تولید یک سیستم کامپیوتری بر اساس ترجیح کاربر داشت.

DXplain: این نیز یک سیستم پشتیبانی بالینی است که می تواند انواع بیماری ها را بر اساس یافته های پزشک پیشنهاد دهد.

مزایای سیستم های خبره

سیستم خبره به عنوان یکی از شاخه های پژوهشی هوش مصنوعی است که به دلیل مزیت های مختلف آن، مطالعات زیادی را به خود اختصاص داده است. در ادامه، به برخی از مهم ترین مزیت های سیستم های خبره اشاره می شود.

- با بسط دانش این سیستم ها، می توان دقت خروجی آن ها را افزایش داد و دانش حوزه های تخصصی مختلف را به آن اضافه کرد.
- از این نوع سیستم ها می توان در شرایط بحرانی و حادی استفاده کرد که نیاز به تصمیم گیری با حداقل ریسک وجود دارد.
- میزان خطای موجود در استنتاج و نتیجه گیری سیستم با افزایش اطلاعات پایگاه دانش به مراتب کم تر می شود.
- خروجی این سیستم ها تحت تاثیر عوامل مختلف احساسی نظیر ترس، عصبانیت، دلسوزی و مواردی از این قبیل قرار نمی گیرد.
- با استفاده از سیستم های خبره می توان در کوتاه ترین زمان ممکن، مسائل پیچیده را با دقت بالا حل کرد.
- سیستم های خبره با بیان ادله و توضیحات مناسب، خروجی خود را به کاربر ارائه می دهند.

محدودیت های سیستم های خبره

علی رغم مزیت های مهمی که سیستم های خبره دارند، می توان به مواردی اشاره کرد که به عنوان معایب این نوع سیستم ها محسوب می شوند.

در ادامه، برخی از مهم ترین محدودیت ها اشاره شده است:

- استنتاج و نتیجه‌گیری سیستم‌های خبره تا حد زیادی به دانش موجود در پایگاه دانش وابسته است. بدین ترتیب، چنانچه پایگاه دانش شامل دانش نادرست باشد، نتیجه استنتاج سیستم‌های خبره نیز نادرست خواهد بود.
 - سیستم‌های خبره نمی‌تواند همانند انسان خبره، بر اساس سناریوهای مختلف، نتیجه‌گیری‌های خلاقانه ارائه دهد.
 - هزینه‌های توسعه و نگهداری چنین سیستم‌هایی بسیار بالا هستند.
 - برای هر حوزه تخصصی، باید پایگاه دانش مجزایی تهیه شود که آماده‌سازی آن‌ها بسیار زمان‌بر و هزینه‌بر خواهد بود.
 - به منظور افزایش کارایی سیستم‌های خبره، باید دانش مورد نیاز آن‌ها را به‌روزرسانی کرد، زیرا چنین سیستم‌هایی نمی‌توانند دانش جدید را یاد بگیرند و باید مهندس دانش، دانش مورد نیازشان را به‌طور دستی تهیه کند.
- زبان‌هایی که برای توسعه سیستم‌های خبره به کار می‌روند به دو دسته تقسیم می‌شوند:

۱- زبان‌های قراردادی

۲- زبان‌های هوش مصنوعی

زبان‌های قراردادی :

زبان‌های قراردادی به نام زبان‌های مسئله‌گرا (مبتنی برمسئله) نیز نامیده می‌شوند. مثل C یکی از فوائد اصلی استفاده از زبان‌های قراردادی قابلیت دسترسی واسطه‌ها به نرم افزارهای قراردادی مانند پایگاه‌های داده و صفحات گسترده است. اگرچه بسیاری از ابزارهای توسعه سیستم خبره ، هم اکنون از نظر تجاری در دسترس هستند و این امکانات را دارا می‌باشند، اما ممکن است که برای ساخت یک سیستم خبره از یک زبان برنامه نویسی قراردادی مثل C یا پاسکال وحتى کوبول هم استفاده شود . همان طوری که می‌توان از یک قیچی هم برای بریدن چمن استفاده کرد. این زبان‌ها برای ساخت سیستم‌های خبره مناسب نیستند. چرا که قدرت اداره ساختار دانش (که قبلاً بحث آن گذشت) را ندارند. برای مثال کوبول برای پردازش داده‌ها طراحی شده است نه برای نمایش و کنترل دانش. با این وجود از زبان‌هایی مانند زبان C که مزیت اصلی آن سرعت می‌باشد در طراحی سیستم خبره استفاده می‌شود.

زبان‌های هوش مصنوعی :

زبان های هوش مصنوعی ، زبان های سمبلیک نامیده می شوند. چون آنها برای کاربرد های هوش مصنوعی ایجاد شده اند. رایج ترین این زبان ها Lisp و PROLOG هستند. اولین گروه از سیستم های خبره اغلب از Lisp استفاده می کردند. ساخت سیستم های خبره با زبان های سطح بالابسیار وقت گیر است و اغلب چندین سال طول می کشد. چرا که سازنده مجبور است واسط کاربر، پایگاه دانش و موتور استنتاج را با استفاده از ساختارهای موجود در زبان توسعه دهد. یک مثال از چنین سیستم هایی DENDRAL می باشد که عبارت است از یک سیستم خبره تحلیل شیمیایی، که برای استنتاج ساختارهای مولکولی ناشناخته به کار می رفت. PROSPECTOR مثالی دیگر از سیستم خبره استخراج معدن می باشد که مانند یک مشاور فنی به زمین شناسان در استخراج معدن کمک می کند مثال دیگر Xcon می باشد که DEC از آن برای سفارشات مشتریان استفاده می کند.

عمومی ترین ابزارهایی که امروزه برای توسعه سیستم های خبره به کار می رود پوسته های سیستم خبره نام دارند.

پوسته های سیستم خبره:

پوسته های یک خبره شروع آسان و راحت برای ساخت سیستم های خبره ایجاد می کنند چرا که کاربرد آن ها ساده می باشد. در واقع آن سیستم های خبره ای هستند که خالی از قوانین می باشند. به این معنی که توسعه دهندگان می توانند روی ورودی پایگاه دانش متمرکز شوند بدون اینکه مجبور باشند چیزی مانند موتور استنتاج و یا واسط کاربر را بسازند. حتی خبرگانی که برنامه نویسی را نیز بلد نیستند می توانند به سرعت با پوسته ها آشنا شوند. همچنین بسیاری از پوسته های سیستم های خبره شامل امکاناتی است که اکتساب دانش را نیز راحت می سازند.

چندین نمونه پوسته عموماً در دسترس هستند عبارتند از:

AM برای ویندوز (که قبلاً تحت عنوان Crystal شناخته می شد) Leonardo و EXSYS که همه این ها روی سخت افزارهای کامپیوترهای شخصی تحت ویندوز کار می کنند.

پوسته ها در زمینه های تخصصی :

هم اکنون ابزارهای متعددی برای توسعه زمینه های کاربردی خاص در دسترس می باشند. برای مثال پوسته هایی برای سیستم های تشخیصی، سیستم های وضعیتی، سیستم های برنامه ریزی، سیستم های کمکی راهنما و غیره وجود دارد. یک مثال از این ابزارهای خاص که در فصل قبلی ذکر شد ، CBR Express نام دارد.

یک CBR خصوصاً برای کاربرد در ابزارهای کمکی راهنما بسیار مناسب است. پوسته زمینه های خاص ، بسیار گرانتر از پوسته های معمولی می باشد.

پوسته های توسعه CBR در دسترس می باشند. هارمون (در سال ۱۹۹۲) به بازنگری در مورد چهار نمونه از این پوسته ها پرداخت. آن ها عبارتند از : ReMind برای سیستم های تشخیصی - CBR Express/ART- IM از محصولات استنتاج، Esteem از شرکت نرم افزاری Esteem و Caspower برای راه حل های استقرایی.

جعبه های ابزار هوش مصنوعی

ابزارهای دیگر برای سیستم ها خبره جعبه های ابزار هوش مصنوعی یا محیط ها می باشند. این ابزارها ابزارهای پیشرفته ای هستند که نوعاً شامل ساختارهای مختلف کد، برای مجموعه ای از کارهای سیستم خبره می باشند. آنها از قوانین ، قاب ها، برنامه نویسی شی گرا (OOP) و شبکه های معنایی یا منطق استفاده می کنند و ممکن است از زنجیره های روبه عقب یا رو به جلو CBR و یا تکنیک های متنوع توارث نیز استفاده نمایند جعبه های ابزار هوش مصنوعی پیش رفته تر از پوسته ها (shell) هستند بنابراین آنها میزان تولید و بهره‌وری را افزایش می دهند گرچه به علت پیچیدگی ، جعبه های ابزار هوش مصنوعی بیشتر از زبان های برنامه نویسی یا پوسته ها نیاز به مهارت دارند نهایتاً می توان گفت که پوسته ها برای کاربرد های کوچک مناسب هستند و ابزارهای هوش مصنوعی نیز بیشتر برای کاربرد های بزرگ (همانند گروه های کاری Client/Server) مناسب هستند. دو نمونه از رایج ترین جعبه های ابزار هوش مصنوعی عبارتند از :

۱- ART-IM که یک جعبه ابزار هوش مصنوعی بر پایه لیست و برای کامپیوترهای شخصی بوده که در محیط ویندوز قابل اجراست. ART-IM شامل ۴ قسمت اصلی، قوانین (اصولاً برای دانش رویه ای بکار می رود) واقعیت ها، قاب ها ، نظریه ها (برای دانش اعلانی) می باشد ART برای کاربردهای پیچیده و مخصوصاً کاربردهای بلادرنگ مثل تفسیر هوشمند داده های دریافتی از حس گرها^۱ در محیط کارخانه مناسب می باشد

۲- Level 5 Object که یک جعبه ابزار کامپیوتر های شخصی می باشدو برای کاربردهای Client/Server مناسب می باشد. همانند Level 5 Object ART-IM نیز مدل های مختلف دانش، شامل: قوانین، قاب ها و دامنه را حمایت می کند. همچنین شامل قابلیت های استدلال ارزنده ای می باشد. محیط توسعه شامل ویرایشگر قانون، ویرایشگر اشیاء و ویرایشگر نمایش برای طراحی واسط کاربرد محیط ویندوز می باشد. ویرایشگر قانون جهت نگهداری قوانین و نمایش درختی دانش استفاده می شود. درخت دانش به معنای نمایش گرافیکی دانش بر مبنای درخت تصمیم گیری می باشد. ویرایشگر شی اجازه می دهد که توسعه دهنده کلاس های اشیاء را ایجاد کند و ویرایش نماید. امکانات ویرایشگر به توسعه دهنده اجازه ایجاد و

ویرایش موضوعات مختلف را می دهد. امکان کنترل جعبه ها، دکمه های رادویی، پنجره های متن، ناحیه ها و ابزارهای توسعه ویندوز همگی امکاناتی است که در اختیار توسعه دهنده قرار دارد.

ابزارهای کمکی ساخت سیستم :

ابزارهای کمکی ساخت سیستم ، برنامه های مختلفی هستند که کلیه مراحل توسعه را شامل می شوند. آن ها در عملیات مختلف مربوط به اخذ و نیز نمایش دانش خبره به کمک مهندس دانش می آیند. این ابزارها با امکانات مختلف ترسیم فلوچارت و غیره به طراحی سیستم خبره توسط مهندس دانش کمک می نمایند.

یک نمونه ابزار کمکی موجود جهت ساخت سیستم عبارت است از : Xpert Rule که Attar Sftware ایجاد شده است.

ابزارهای اکتساب دانش:

روش های مختلف اکتساب دانش بسیارگران و وقت گیر هستند. تلاش های زیادی صورت گرفته شده است تا ابزاری ایجاد شود که اکتساب دانش را آسان کند. یک موتور استقرایی رایج ترین وسیله ای است که قادر می باشد قوانین را از مثال های موجود استخراج نماید. هرچند که موتور استقرایی بخش ضروری یک سیستم خبره نمی باشد ، با این حال یک بخش مفید ضمیمه شده است که توسط بسیاری از پوسته های سیستم خبره مورد استفاده قرار می گیرد. موتور استقرایی مجموعه ای از نمونه ها و مثال های مختلف را که یک حوزه خاص وجود دارد بررسی نموده ارتباطات بین مفاهیم مختلف را تحلیل می نماید. سپس موتور استقرایی قوانین که این مثال ها را به یکدیگر پیوند می دهد ایجاد می کند و آن ها در پایگاه دانش جای می دهد.

برای ساخت آسان سیستم های خبره، توصیه می شود از پوسته سیستم خبره استفاده کنید که هم در زمان و هم در تلاش صرفه جویی می کند. به سادگی یک پوسته سیستم خبره (ES Shell) یک جعبه ابزار است که از اجزای ضروری داخلی یک سیستم خبره معمولی تشکیل شده است.

پوسته های سیستم خبره

پوسته های سیستم خبره ابزارهایی هستند که می توانند برای توسعه سیستم های خبره استفاده شوند. آنها از برخی اجزای سیستم خبره ساخته شده با پایگاه دانش خالی تشکیل شده اند. از این رو، در بیشتر موارد، مهندس دانش تنها با پر کردن پایگاه دانش باقی می ماند. این اساسا یک ابزار با هدف خاص است که مطابق با الزامات و استانداردهای یک دامنه خاص یا برنامه های کاربردی حوزه دانش تخصصی تعبیه شده است. ممکن است به عنوان یک بسته نرم افزاری تعریف شود که ساخت سیستم های خبره مبتنی بر دانش را با ارائه یک طرح بازنمایی دانش و یک موتور استنتاج تسهیل می کند.

تفاوت پوسته‌های ES راه‌های مختلفی برای مدل‌سازی دانش در پایگاه دانش ارائه می‌دهند. آن‌ها به صورت های زیر هستند

۱- به عنوان قوانین

Rule 1: If the ambient temperature is above 90F

Then the weather is hot.

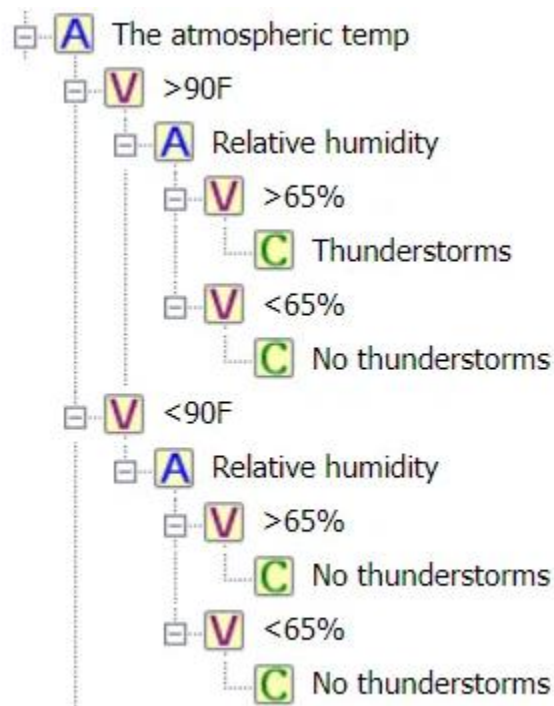
Rule 2: IF the relative humidity is greater than 65%

Then the atmosphere is humid.

Rule 3: If the weather is hot and the atmosphere is humid

Then the thunderstorms are likely to be developed

۲- در قالب درخت تصمیم



شکل ۶- یک سیستم خبره مبتنی بر درخت که بر اساس قوانین ذکر شده در بالا ایجاد شده است

۳- به عنوان اشیاء (فریم) - یک ساختار داده با دانش معمولی در مورد یک شی یا مفهوم خاص.

پوسته ES یک گزینه ایده آل برای توسعه سریع سیستم های خبره است. به همین دلیل می توان آن را یکی از سودمندترین رویکردهای تجاری در پیاده سازی سیستم خبره دانست.

ساختار پوسته سیستم خبره

Expert System Shell به یک ماژول نرم افزاری اشاره دارد که شامل موارد زیر است:

- رابط کاربری (توکار)(built-in)
 - موتور استنتاج (توکار)(built-in)
 - یک اسکلت ساختار یافته از یک پایگاه دانش (در حالت خالی آن) با امکانات بازنمایی دانش مناس
- نه تنها اجزای فوق، بلکه برخی از پوسته های ES نیز امکاناتی را برای اتصال به پایگاه داده از طریق ویژگی های مترجم، یکپارچه سازی وب و پردازش زبان طبیعی (NLP) فراهم می کنند.
- رابط کاربری پورتالی است که هم برای کاربران نهایی (که از سیستم خبره برای دریافت راه حل استفاده می کنند) و هم برای مهندس دانش (که مهندسی دانش و مدل سازی را انجام می دهد) در دسترس است.
- پایگاه دانش را می توان با یک پایگاه داده خارجی (مانند MySQL) متصل کرد زیرا پایگاه دانش برای ذخیره داده های گسترده بهینه نیست. پایگاه دانش نمی تواند مستقیماً به پایگاه داده دسترسی داشته باشد و این ویژگی های دسترسی از طریق یک مترجم انجام می شود.
- برخی از ES Shell ها دارای ویرایشگرهای پایگاه دانش داخلی هستند که به مهندس دانش کمک می کند تا به راحتی پایگاه دانش را به روز کند و بررسی کند. مهندس دانش تخصصی را در یک حوزه خاص جمع آوری می کند و در پر کردن پایگاه دانش مدل می کند.
- موتور استنتاج که مهمترین بخش یک سیستم خبره است به پایگاه دانش دسترسی پیدا می کند و مشکل را با زنجیره ای به عقب و یا زنجیره جلویی حقایق و قوانین موجود در پایگاه دانش حل می کند. در ES Shells، موتور استنتاج نیز یک جزء داخلی است که معمولاً در ProLog برنامه ریزی می شود.
- اکثر پوسته های ES از مؤلفه دیگری به نام «سیستم توضیح» تشکیل شده اند که با در نظر گرفتن «داده های مشخصات موردی» در دسترس، دلایل و توضیحاتی را برای ارائه یک پاسخ مشخص در اختیار کاربر قرار می دهد.

در یک پوسته سیستم خبره، طراحی رابط کاربری و سایر اجزای نرم افزار توسط مهندس نرم افزار برنامه ریزی می شود. بنابراین یک سیستم خبره طراحی مشترکی از ۳ طرف اصلی است: متخصص، مهندس دانش و مهندس نرم افزار.

پوسته های محبوب Expert System

در زیر چند نمونه از سیستم های خبره موجود برای توسعه دهندگان برای ساختن سیستم های خبره در مقیاس های مختلف آورده شده است. بسته به پوسته ES، مزایا و معایب مختلفی برای هر پوسته وجود دارد.

۱-ES-Builder

ES Builder مخفف Expert System Builder است. این یکی از مشهورترین Expert System Shell ها است. این یک پوسته رایگان ES است که به ویژه برای دانشجویان و محققان برای توسعه پوسته های سیستم خبره پیاده سازی شده است. پوسته شامل یک رابط وب بهبود یافته است که با استفاده از چارچوب AJAX ساخته شده است. ES-Builder حقایق و قوانین پایگاه دانش را در یک پایگاه داده آنلاین MySQL ذخیره می کند. یک موتور استنتاج داخلی (نوشته شده در Prolog) وجود دارد و رابط های کاربری با استفاده از HTML و CSS ساده توسعه داده می شوند. اتصال پایگاه داده برای دسترسی به پایگاه دانش با استفاده از PHP حفظ می شود.

مزایای ES Builder:

رابط های کاربری خودکار تولید شده (طراحی ساده HTML CSS رابط کاربری)

امکان ایجاد فایل های پشتیبان از پایگاه دانش و ادغام آسان با پایگاه داده MySQL.

Ability پایگاه دانش را با استفاده از درخت تصمیم پر می کند.

نمای بهینه از پایگاه دانش مبتنی بر قانون.

محدودیت های ES-Builder:

فقط قوانین زنجیره ای رو به جلو پشتیبانی می شوند قوانین زنجیره ای به عقب پشتیبانی نمی شوند.

ابزارهای دسکتاپ فقط از ویندوز ۹۸ و ویندوز ۷ پشتیبانی می کنند.

CLIPS-۲

مخفف C-Language Integrated Production System است. همانطور که از معنی پیداست پوسته سیستم خبره CLIPS به زبان رویه C نوشته شده است. CLIPS در سال ۱۹۸۵ در مرکز فضایی جانسون ناسا توسعه یافت. این یک زبان برنامه نویسی مبتنی بر قانون است که برای ایجاد یک سیستم خبره استفاده می شود. CLIPS در سیستم‌هایی استفاده می‌شود که پیاده‌سازی و نگهداری راه‌حل اکتشافی آسان‌تر از رویکرد الگوریتمی سنتی است.

CLIPS ۳ ابزار مختلف را برای بازنمایی دانش در قالب روش های برنامه نویسی / پارادایم های برنامه نویسی ارائه می دهد. روش ها عبارتند از:

۱- رویه ای

۲- شی گرا

۳- برنامه نویسی مبتنی بر قانون

از آنجایی که CLIPS به زبان C نوشته شده است، سیستم خبره توسعه یافته توسط CLIPS به کامپایلر ANSI نیز نیاز دارد. از آنجایی که سیستم هایی که از قبل دارای کامپایلر ANSI هستند می توانند به راحتی سیستم خبره را بدون تغییر کد منبع اجرا کنند.

مزایای CLIPS:

می تواند به راحتی در سایر برنامه های فرعی ساخته شده در زبان های رویه ای برای گسترش عملکرد موجود جاسازی شود.

از روش هایی مانند منطق فازی و عوامل پشتیبانی کنید.

محدودیت های CLIPS:

فقط از قوانین زنجیره‌ای رو به جلو پشتیبانی می‌کند اما از قوانین زنجیره‌ای رو به عقب پشتیبانی نمی‌کند.

JESS -۳

JESS مخفف Java Expert System Shell است. JESS یکی دیگر از پوسته های سیستم خبره و یک زبان برنامه نویسی است که به طور کامل توسط زبان برنامه نویسی جاوا نوشته شده است. JESS به شدت برای

ساختن سیستم‌های خبره مبتنی بر قانون استفاده می‌شود که با سیستم‌هایی که روی جاوا نوشته شده‌اند پیوند محکمی دارد. سیستم‌های خبره ای که با استفاده از JESS ساخته شده‌اند می‌توانند بر روی رابط خط فرمان اجرا شوند یا به عنوان اپلت مورد استفاده قرار گیرند.

سیستم‌های خبره مبتنی بر JESS به طور گسترده به عنوان عوامل هوشمند، در سیستم‌های برنامه ریزی منابع سازمانی و برای اعتبار سنجی پلت فرم‌های تجارت الکترونیک استفاده می‌شوند.

مزایای JESS

بهتر است در مسائل بزرگتر که عملکرد تحت سلطه کیفیت الگوریتم است، اعمال شود.

شامل جستجوهای مربوط به زنجیره عقب و حافظه کاری است.

سریعتر از سیستم‌های خبره که با استفاده از پوسته‌های سیستم خبره نوشته شده بر روی زبان C توسعه یافته‌اند.

حاوی دستوراتی است که باعث می‌شود در هنگام اجرای سیستم از حافظه کمتری استفاده شود.

محدودیت‌های JESS

برخی از قابلیت‌ها هنگام استفاده در مرورگر محدود هستند.

PyKE-۴

PyKE مخفف Python Knowledge Engine است. PyKE از برنامه نویسی منطقی الهام گرفته شده از Prolog استفاده می‌کند، اما PyKE کاملاً به زبان برنامه نویسی Python نوشته شده است. توابع پایتون، قوانین PyKE، و متغیرهای الگوی PyKE، طرح‌های نموداری از ویژگی‌های اصلی پایگاه دانش PyKE هستند. این یک موتور استنتاج است که از قواعد و حقایق برای ایجاد حقایق اضافی با استفاده از زنجیره پیشروی قوانین برای اثبات اهداف استفاده می‌کند.

و همچنین، از طریق زنجیره‌ای به عقب، توابع پایتون را در نمودارهای فراخوانی قابل تنظیم که به نام پلان نیز شناخته می‌شوند، مونتاژ می‌کند.

مزایای PyKE:

بسیار قابل تنظیم و سازگاری کد.

عملکرد بالا.

قابلیت استفاده مجدد کد

هر دو زنجیر به جلو و زنجیر عقب پشتیبانی می شوند.

امکان اجرای دستورات در سیستم محلی برای هدایت استنتاج.

قابلیت پرسش کاربر نهایی را فراهم کنید.

محدودیت های PyKE:

رابط کاربری باید به صورت دستی ایجاد شود.

مشکلات یکپارچه سازی هنگام ادغام موتورهای استنتاج با رابط های کاربری مختلف رخ می دهد.

ارزیابی پوسته های سیستم خبره:

معیارهای کلی انتخاب یک پوسته خاص برای توسعه سیستم خبره عبارتند از:

سهولت استفاده :

ابزارهای انتخاب شده برای استفاده باید آسان باشد و نیز نمونه سازی را با سرعت انجام دهد.

توانایی تکنیکی :

پوسته ها به طرق مختلف به نمایش دانش می پردازند و استراتژی های مختلفی را برای استنتاج بکار می برند. برای مثال VP-Expert کاملاً بر مبنای قوانین است و دیگر پوسته ها مثل Leonardo قاب ها یا شیء گرایی را پشتیبانی می کنند . قابلیت های تکنیکی باید توسط توسعه دهنده پس از درک کامل از ماهیت حوزه کاربرد شناسایی گردد.

محیط پشتیبانی توسعه:

ساختار یک پوسته خبره که محیط را پشتیبانی می کند حداقل باید شامل یک ویرایشگر به منظور ایجاد پایگاه دانش ، ابزار اشکال زدایی و ردیابی و نیز یک واسط کاربر دارای امکانات توضیح باشد.

امکانات واسط کاربر:

پوسته ها باید شامل ابزاری برای توسعه کیفیت واسط کاربر باشند. بطوری که کاربر بتواند به هنگام اجرا امکانات مختلفی از قبیل مسیریابی متن های به هم پیوسته و دستیابی به اطلاعات مختلف استفاده کند. همانند امکانات دسترسی به صفحات وب و یا استفاده از فرامتن.

واسط خارجی:

بیشتر پروژه های سیستم خبره نیاز به درجه ای از ارتباط با خارج از محیط سیستم دارند. از قبیل :

نیاز به فایل های صفحه گسترده، پایگاه های داده و یا حتی روال های ++C. برای مثال در یک سیستم خبره پزشکی ممکن است به اطلاعاتی در رابطه با بیمار نیاز داشته باشیم که این اطلاعات در یک پایگاه داده ای ذخیره شده باشد.

مجوز اجرا و پشتیبانی فروشنده:

در واقع این پارامتر چگونگی توسعه یک محصول را از لحاظ کاربردی بررسی می نماید. مثلاً برنامه روی نرم افزارها و سخت افزارهای مختلف بدون هزینه های اضافی (افزایش قیمت) قابلیت توسعه و توزیع داشته باشد (قابلیت حمل) . به عنوان مثال یک سیستم خبره توسعه یافته بر روی یک کامپیوتر پنتیوم تحت ویندوز ۹۸ ممکن است روی یک کامپیوتر ۴۸۶ تحت DOS اجرا نشود.

فصل ۳

نرم افزار کلیپس (CLIPS)

از این نرم افزار برای ساخت سیستم های خبره استفاده می شود.

برخی از ویژگی های نرم افزار به شرح زیر می باشد.

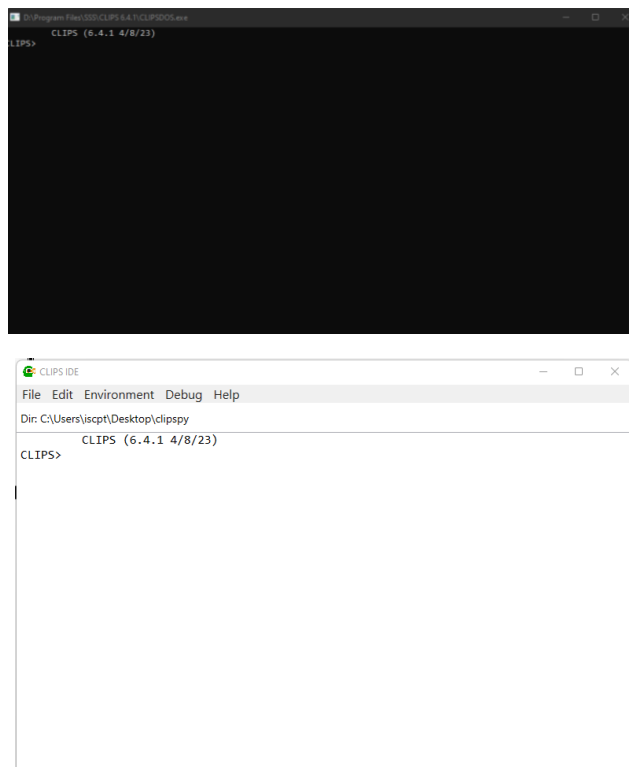
- از سر نام های عبارات C Language Implementation Production System گرفته شده است
- برای اولین بار توسط NASA در واحد فناوری نرم افزار (Software Technology Branch) تولید شده است.
- این نرم افزار رایگان بوده و در حال حاضر نسخه ۶.۴.۱ آن موجود می باشد.
- این نرم افزار بر روی سیستم عامل های متفاوت مانند ویندوز لینوکس. مک نصب می گردد.
- این نرم افزار با استفاده از زبان C ساخته شده است.
- این نرم افزار چند کاره می باشد یعنی می توانید بصورت مدل های زیر با آن برنامه نویسی کنید.
 - بر مبنای قواعد (Rule-based)
 - بر مبنای شی گرایی (Object-oriented)
 - بر مبنای روال ها (Procedural programming)
- نرم افزار کلیپس فقط قواعد زنجیره ای بشرو را پشتیبانی می کند.

دستیابی به نرم افزار

این نرم افزار یک برنامه رایگان می باشد که هم اکنون نسخه ۶.۴.۱ آن موجود می باشد می توانید این نرم افزار را برای سیستم عامل های موجود یعنی ویندوز. لینوکس و مک تهیه نمایید به همین منظور می توانید به سایت زیر مراجعه کرده و آن را دانلود نمایید.

<http://clipsrules.sourceforge.net>

برنامه کلیپس شامل دو کنسول برای نوشتن و اجرا نمودن برنامه ها می باشد. یک کنسول تحت داس می باشد محیط شبیه سیستم عامل داس Command Prompt دارد و تمام محیط تعاملی به صورت دستوری می باشد. و دارای یک محیط ویژوالی مانند دیگر نرم افزار های تحت ویندوز می باشد.



شکل ۷-نمایی از برنامه کلیپس

مولفه های پایه نرم افزار

- فیلد field
- واقعیات fact
- قواعد rule
- الگو template
- کلاس class

فیلدها

برای ساخت پایگاه دانش باید اطلاعات از ورودی (صفحه کلید یا فایل) خوانده شود سپس دستورات اجرا شوند. در طی فرایند اجرای برنامه، نرم افزار کلیپس نشانه ها Symbol، کاراکتر هایی که یک معنی می دهد را با یکدیگر ترکیب و گروه بندی می کند و یک توکن می سازد. در واقع فیلد یک نوع خاصی از توکن می باشد.

فیلد عددی

دو نوع فیلد عددی وجود دارد.

عدد صحیح Integer مانند ۲۳۸ ، -۳۲

عدد ممیز شناور Float مانند $2.73E3$ ، 0.15 ، $-25.E7$ همان نماد علمی می باشد.

فیلد نشانه ای

- فیلد های نشانه ای حاوی کاراکتر های اسکی می باشند.
- این فیلد ها دارای محدودیت هایی می باشند.
- نشانه ها در کلیپس حساس به حروف Case Sensitive هستند.
- نشانه ها نمی توانند با کاراکتر های ؟ و ؟\$ شروع شوند چون این ۲ کاراکتر برای معرفی متغیر ها رزرو شده اند.
- نمونه هایی از نشانه ها : foo ،

محدودیت فیلد نشانه ای

فیلد های نشانه ای نمی توانند شامل سمبل های زیر باشند.

- کاراکتر های اسکی چاپ نشدنی (مانند کاراکتر فاصله Space)
- دابل کوتیشن ” ”
- پرانتز با و بسته ()
- امپرسند &
- خط عمودی |
- علامت کوچکتر <
- علامت تیلدا ~
- سمی کولن ;

فیلد رشته ای

فیلد های رشته ای در بین ۲ دابل کوتیشن " " قرار می گیرند. تنها محدودیت در این فیلد ها استفاده از بک اسلش \ می باشد. که برای چاپ بک اسلش از \\ و برای چاپ " از \ استفاده می شود. در نمونه هایی آورده شده است.

- "foo"
- "a and b"
- "a\"quote"
- "1 number"

فیلد آدرس

این فیلد ها آدرس داده ها را نگه داری می کنند مانند زبان C.

واقعیات (Fact)

Fact ها در واقع واقعیت هایی هستند که کاربر به عنوان ورودی به سیستم خبره به می دهد و از سیستم خبره ، خبرگی دریافت می کنند. کلیپس برای حل مسئله ها نیاز به اطلاعات و داده ها دارد. پایه واحد های داده که در قواعد (Rule) استفاده می شود همان Fact می باشد. هر Fact از قسمت های زیر تشکیل شده است.

- نام واقعیت (Relation name)
- تعداد صفر یا بیشتر اسلات (Slot name) به همراه مقدار (Slot value)

Syntax:

Relation Name(

(SlotName1 SlotValue1)

(SlotName2 SlotValue2)

)

مثال

Person (

(Family “Asghari”)

(Age 24)

)

ساختار Deftemplate:

با استفاده از این دستور می توان ساختار fact ها را بصورت گروهی ایجاد نمود. باید دقت نمود که این دستور فقط ساختار اولیه Fact را می سازد و هیچ Fact را ایجاد نمی کند و باید با دستور Assert آن ساختار ایجاد شده توسط Deftemplate را مقدار دهی نماییم.

Syntax:

(deftemplate <relation-name> [<optional-comment>]

<slot-definition>*)

<slot-definition>

(slot <slot-name>) | (multislot <slot-name>)

)

<relation-name>: نام الگو می باشد.

<optional-comment>: می توان در این قسمت توضیحاتی برای الگو قرار داد. این قسمت اختیاری می

باشد.

(slot <slot-name>): این قسمت اجازه می دهد یک فیلد با یک مقدار ذخیره گردد.

(multislot <slot-name>): این فیلد اجازه می دهد چندین مقدار در یک اسلات ذخیره شود.

در مثال زیر یک فقط ساختار ایجاد شده است و تا زمان مقدار دهی با دستور Assert این الگو به Fact تبدیل نمی شود.

(Deftemplate Person “This Template is For one Person”

(MultiSlot Name)

(Slot Age)

(Slot Eye-Color)

(Slot Hair-Color)

)

دستور Assert

مقداردهی ساختار Template با دستور Assert انجام می شود.

Example:

```
(Assert Person(  
  (Name John B. Jakson)  
  (Age 25)  
  (Eye-Color Brown)  
  (Hair-Color Black)  
)  
)
```

با دستور Assert می تواند Fact هایی بصورت جداگانه نیز تعریف کرد که در این صورت نیازی به تعریف Template نمی باشد.

(Assert(Color Red))

مثال فوق یک Fact با نام Color با مقدار Red به Fact-List اضافه می کند.

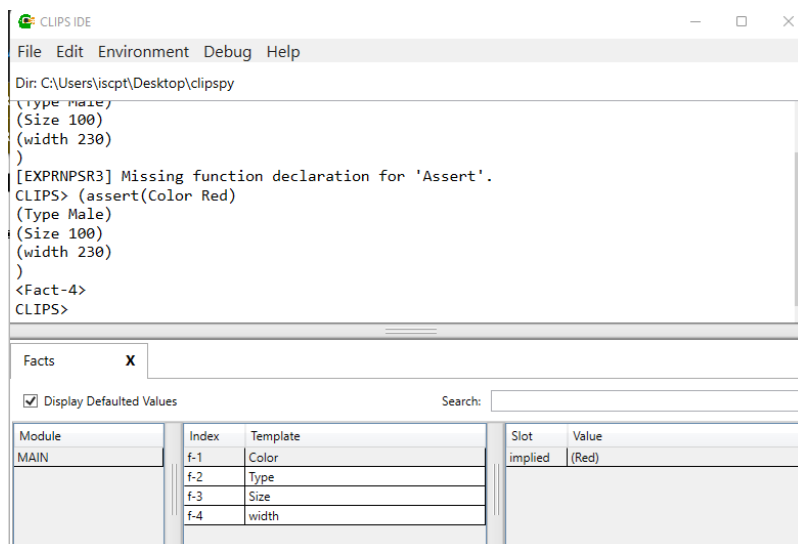
(Assert(Color Red)

(Type Male)

(Size 100)

(width 230))

مثال فوق چهار fact به ترتیب به نام های Color و Type و Size و width با مقادیر Red و Male و ۱۰۰ و ۲۳۰ به طور همزمان اضافه می نماید



شکل ۸- فکت های ذخیره شده در کلیپس

محدودیت ها برای اسلات

با اعمال محدودیت ها می توان اسلات های یک تمپلیت را مجبور به وارد کردن داده های با قالب خاص نمود و یا مقداری پیش فرض برای اسلات ها در نظر گرفت.

constraint-attribute> ::= <type-attribute> |

< allowed-constant-attribute | >

< range-attribute | >

< cardinality-attribute >

< default-attribute >

) < type-attribute> ::= (type <type-specification>) >

< <type-specification> ::= <allowed-type>+ | ?VARIABLE

< allowed-type> ::= SYMBOL | STRING | LEXEME |

INTEGER | FLOAT | NUMBER |

INSTANCE-NAME | INSTANCE-ADDRESS |

INSTANCE | FACT-ADDRESS |

EXTERNAL-ADDRESS

<allowed-constant-attribute>

::= (allowedsymbols <symbol-list>) |

(allowedstrings <string-list>) |

(allowedlexemes <lexeme-list>) |

(allowedintegers <integer-list>) |

(allowedfloats <float-list>) |

(allowednumbers <number-list>) |

(allowedinstance-names <instance-list>) |

(allowedvalues <value-list>)

مثال

deftemplate person

(slot name (type STRING))

(slot age (type INTEGER))

(slot gender (allowed-symbols male female))

)

نام فقط می تواند نوع

رشته ای قبول نماید

سن فقط می تواند نوع

عدد صحیح قبول نماید

جنیست Gender فقط می تواند شامل یکی از ۲
مقدار "Male" یا "Female" باشد.

به عنوان مثال در صورتی که سعی کنید مقداری بر خلاف محدودیت تعریف شده وارد نمایید با خطایی از سوی نرم افزار کلیپس مواجه خواهید شد.

متغیر ها Variable

در نرم افزار کلیپس میتوان ۲ نوع متغیر تعریف نمود.

- متغیر تک مقداری (Single Value) <Symbol>?
- متغیر چند مقداری (Multi Values) <Symbol> \$?

ویلکارد ها (Wildcard)

ویلکاردها علامت هایی هستند که بجای هر کاراکتر یا کاراکتر خاصی بکار می روند. مانند * که در هنگام جستجو به معنای ”هر” بکار می رود.

بدست آوردن شماره ایندکس Fact

برای انجام بعضی از عملیات ها نیاز به شماره ایندکس Fact می باشد به همین دلیل از روش زیر شماره ایندکس را بدست می آوریم. باید بدانیم که شماره ایندکس منظور همان شماره ای است که در پنجره Fact-List در سمت چپ Fact نشان داده می شود و این شماره را خود نرم افزار کلیپس بصورت اتوماتیک به Fact ها اختصاص می دهد.

<variable-symbol> <- <pattern-CE>

<variable-symbol>: نام یک متغیر است که شماره ایندکس در آن قرار می گیرد.

<pattern-CE>: نام Fact است که می خواهیم شماره ایندکس آنرا بدست بیاوریم.

<- : علامت انتساب می باشد.

f-0 (initial-fact)

f-1 (data 1 blue)

f-2 (data 1 blue red)

f-3 (data 1 blue red 6.9)

در لیست بالا تعداد چهار Fact وجود دارد شماره ایندکس 0 یعنی f-0 که توسط خود نرم افزار کلیپس رزرو می شود و در واقع یک مقدار دهی اولیه انجام می دهد حتی اگر هیچ Fact تعریف نشده باشد این شماره به عنوان f-0 (initial-fact) توسط خود نرم افزار استفاده می شود و Fact ها از شماره ۱ به بعد شماره گذاری می شوند.

حذف fact با دستور Retract

با استفاده از این دستور می توان یک Fact را حذف نمود. باید دقت نمود که این دستور با داشتن شماره ایندکس Fact می تواند آنرا حذف نماید که از روش قبل شماره ایندکس را بدست می آوریم.

(retract <retract-specifier>+ | *)

<retract-specifier>: شماره ایندکس Fact ای را که قرار است حذف نماییم، مشخص می کنیم.

دستور Save-Fact جهت ذخیره Fact ها

با استفاده از این دستور می توان کلیه Fact های موجود (Fact ها در پنجره Fact) را در داخل فایل ذخیره نمود.

دستور Load-Facts جهت باز نمودن Fact ها

با استفاده از این دستور می توان Fact هایی که قبلا توسط دستور Save-Facts در فایل ذخیره نموده ایم را به لیست Fact-list اضافه نماییم.

تعریف قواعد (Rules)

با این دستور می توان قواعد را تعریف نماییم. نرم افزار کلیپس یک برنامه Rule-based نیز می باشد یعنی بر مبنای قواعد نیز کار می کند.

(defrule <rule-name> [<optional comment>]

<patterns>*

=>

<actions>*

)

<rule-name>:این قسمت شامل نام قاعده یا Rule می باشد.

<optional comment>:در این قسمت می توان توضیحاتی در مورد آن قاعده نوشت. توضیحات اختیاری می باشد.

<patterns>:این قسمت الگو را مشخص می نماید یعنی در واقع همان قسمت الگو عبارت شرطی می باشد.

<actions>:این قسمت اعمال و دستوراتی می باشد که در صورت برقرار شدن (True) قسمت Pattern اجرا می شود. نکته:علامت * در Syntax ها به معنی تعداد ۰ یا بیشتر می باشد *<patterns> یعنی چندین الگو می توان تعریف نمود.

(Defrule Rule1

(car(color red))

=>

(assert(Action ok))

)

یعنی اگر fact با نام car مقدار فیلد Color آن Red بود سپس یک Fact با دستور Assert به نام Action ایجاد کن و مقدار آنرا ok قرار بده.

نکته:علامت => معادل کلمه کلیدی Then در شبه کد می باشد

ایجاد Fact گروهی با دستور Deffacts

با استفاده از این دستور می توان Fact هایی بصورت گروهی تعریف کرد و به عنوان دانش اولیه با پایگاه دانش اضافه می شود.

(deffacts <deffacts-name> [<comment>]

<pattern>*)

<deffacts-name>: نام fact را مشخص می کند.

نکته: Fact هایی که با دستور Deffact تعریف می شوند در پنجره Facts نمایش داده نمی شوند.

[<comment>]: توضیحاتی برای Fact می باشد که نوشتن این توضیحات اختیاری می باشد.

<pattern>: نام الگو یا همان Fact ها را مشخص مینماید.

(deffacts BaseFact1

(Color Red)

(type Stuff)

(Height 100)

)

مثال فوق Fact با نام BaseFact1 شامل فیلدهای اولیه Color, type, Height را به عنوان دانش اولیه به پایگاه داده اضافه می نماید.

چاپ رشته / متغیر با دستور Printout

(printout <logical-name> <print-items>*)

<logical-name>: در این قسمت می توان نوع خروجی را مشخص نمود با کاراکتر t (Terminal)

مانیتور به عنوان خروجی در نظر گرفته می شود.

<print-items>: در این قسمت متن مورد نظر با آیتم هایی که قرار است چاپ شوند را مشخص می نماییم.

نکته: عبارت **CrLf** تعیین می کند که بعد از چاپ نمودن آیتم ها مکان نما به خط بعدی رود. این کلمه معادل **\n** در زبان **c++** میباشد.

نمایش لیست دستورات

با استفاده از دستورات زیر می توان لیست بعضی از ساختارهایی که تا بحال یاد گرفته ایم را مشاهده نماییم.
(list-defrules): این دستور لیست قواعد موجود را نمایش می دهد.

(list-deftemplates): این دستور لیست الگو های موجود را نمایش می دهد.

(list-deffacts): این دستور لیست Fact هایی را که با دستور Deffacts تعریف کرده ایم را نمایش می دهد.

(Facts): این دستورات لیست تمام fact(Fact) های موجود در پنجره Facts که با دستورات assert یا Deftemplate اضافه نموده ایم) را نمایش می دهد.

دستورات اساسی

(clear): این دستور تمام Fact ها و Rule ها را از حافظه کاری حذف می کند معادل این است یکبار از نرم افزار خارج شویم و دوباره وارد شویم(Shutdown + Restart).

(reset): این دستور اطلاعات مربوط به Fact ها را حذف میکند و Agenda را ریست می کند.

(Agenda): قواعد فعال را نمایش می دهد.

(Refresh): این دستور قواعد را بروز رسانی می نماید.

(Run [<Limit>]): با این دستور قواعد فعال در حافظه کاری اجرا می شود که در قسمت <Limit> تعداد قواعدی را که می خواهیم اجرا شود را مشخص می کنیم. مشخص نمودن تعداد اختیاری می باشد. در صورتی که تعداد را مشخص نماییم یعنی (Run) در این صورت کلیه دستورات فعال در پنجره Agenda اجرا می شوند.

بعضی توابع سودمند

تابع save

با این تابع مجموعه از ساختارهایی که وجود دارند(Fact و قواعد و الگوها و ...) را در فایلی ذخیره می نماید.

(save <file-name>)

<file-name>: مسیر و نام فایل به همراه پسوند آن را مشخص می نماید.

تابع Load

توسط این دستور می توان ساختار هایی که قبلا ذخیره نموده ایم (Save) را باز نماید و آماده اجرا کند.

(load <file-name>)

<file-name>: مسیر و نام فایل به همراه پسوند آن را مشخص می نماید.

تابع Open

با استفاده از این تابع می توان فایل برنامه کلیپس که قبلا ذخیره نموده ایم را باز نماییم.

(open <file-name> <logical-name> [<mode>])

<file-name>: این قسمت شامل نام فایل به همراه آدرس کامل و پسوند فایل می باشد. حتما اگر در مسیر علامت بک اسش وجود دارد باید آنرا مشخص نماییم.

نکته: برای مشخص نمودن علامت \ باید این علامت را ۲ بار پشت سر هم تایپ نماییم تا برای کلیپس معنی \ این بدهد یعنی \ \ را تایپ می کنیم.

<logical-name>: این قسمت یک نام منطقی می باشد که قبلا در برنامه جاری نباید استفاده شده باشد.

[<mode>]: این قسمت نحوه دستیابی به فایل را مشخص می نماید. این قسمت اختیاری می باشد.

انواع حالت های Mode

"r" read access only

"w" write access only

"r+" read and write access

"a" append access only

در صورتی که تابع Open با موفقیت اجرا شود مقدار True و در غیر اینصورت مقدار False را بر می گرداند.

تابع Close

این تابع بر عکس تابع Open کار میکند و فایلی را که قبلاً با تابع Open باز شده است را می بندد.

(close [<logical-name>])

<logical-name>: نام منطقی فایل است که هنگام Open کردن به آن اختصاص دادیم.

- اگر تابع Close بدون پارامتر فراخوانی شود تمام فایل های باز را می بندد.
- در صورتی که تابع Open با موفقیت اجرا شود مقدار True و در غیر اینصورت مقدار False را بر می گرداند.

تابع Read

با استفاده از این تابع می توان مقداری را از صفحه کلید دریافت نمود یا محتویات یک فایل را خواند.

(read [<logical-name>])

<logical-name>: نام منطقی فایل است که هنگام Open کردن به آن اختصاص دادیم. این پارامتر اختیاری است و اگر فایل را مشخص نکنیم مقداری را از صفحه کلید دریافت می کند.

تابع Bind

با استفاده از این تابع یک مقدار را به یک متغیر منتقل می نماییم.

(bind <variable> <expression>*)

<variable>: نام یک متغیر را مشخص می نماید.

<expression>: یک عبارت را که قرار است در متغیر قرار گیرد مشخص می نماید.

محدودیت های کاربری CLIPS

پوسته های سیستم خبره یک وسیله سریع و کم هزینه را برای آشنایی با تکنولوژی مهیا می سازد. اگرچه برای محدوده های خاصی درسیستم های پیچیده استفاده از پوسته های سیستم خبره میسر نمی باشد. اما با تمام این محدودیت ها، پوسته ها بطور زیادی مورد استفاده قرار می گیرند.

در بعضی موارد آنها در ابتدای چرخه یعنی در مراحل اولیه کار، به منظور ایجاد درک بهتر از نیازمندی های مسأله استفاده می شوند و در سطوح بالاتر کنار گذاشته می شوند.

قانون زیر به اختصار چگونگی انتخاب یک ابزار برای توسعه سیستم خبره را بیان می کند:

اگر می توانید پوسته ها را به کار برید . درجایی که مجبورید از یک جعبه ابزار استفاده کنید و وقتی که چاره ای ندارید، یک زبان هوش مصنوعی را به کار گیرید.

برای توسعه پروژه های CLIPS، برنامه نویسان باید یا از یک ابزار مبتنی بر دستور یا یک IDE استفاده کنند که برای کمک به برنامه نویسان ایجاد شده است، اما، این محیط برای کاربران نهایی کاربر پسند نیست، بنابراین، با ایجاد یک رابط برای کد خود، می توانیم کاربران نهایی را حفظ کنیم. کاربر را از پیچیدگی های سیستم در پشت صحنه دور کرده و همچنین به آنها کمک می کند تا راحت تر از برنامه استفاده کنند.

ورژن های CLIPS

نسخه ۶.۲ CLIPS

شامل دو پیشرفت عمده است. ابتدا، مکانیزمی که به یک برنامه کاربردی تعبیه شده اجازه می دهد تا چندین محیط ایجاد کند که در آن برنامه ها قابل بارگیری هستند. دوم، یک رابط ویندوز XP CLIPS/۲۰۰۰ بهبود یافته است موجود است و رابط مکینتاش CLIPS برای پشتیبانی از MacOS X بهبود یافته است

نسخه ۶.۳ CLIPS

شامل یک پیشرفت عمده (بهبود عملکرد قوانین) و چندین مورد است پیشرفت/تغییر جزئی

نسخه ۶.۴ CLIPS

شامل سه پیشرفت عمده است: یک برنامه C بازطراحی شده

رابط برنامه نویسی (API)؛ کلاس های wrapper و برنامه های نمونه برای دات نت و جاوا. و

محیط های توسعه یکپارچه (IDE) با پشتیبانی یونیکد برای ویندوز و جاوا.

فصل ۴

رابط کاربری محاوره ای گرافیکی برای CLIPS

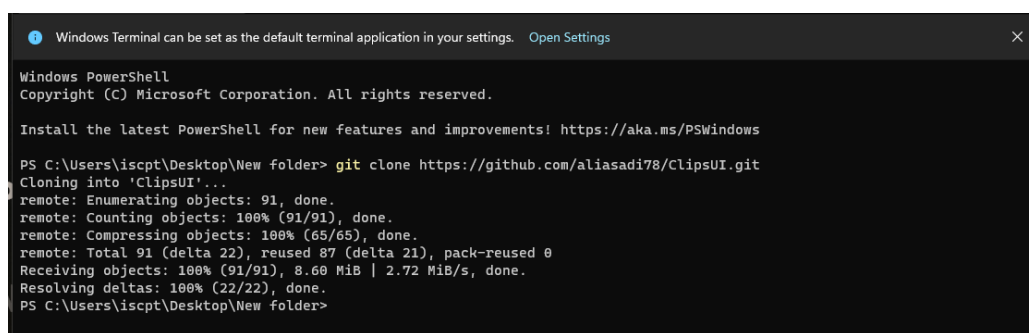
راه اندازی برنامه

برنامه کلیپس را از آدرس زیر دانلود کنید

<https://www.clipsrules.net>

ابتدا با استفاده از دستور زیر پروژه را دانلود می کنیم.

`git clone https://github.com/aliasadi78/ClipsUI.git`



```
Windows Terminal can be set as the default terminal application in your settings. Open Settings X
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\iscpt\Desktop\New folder> git clone https://github.com/aliasadi78/ClipsUI.git
Cloning into 'ClipsUI'...
remote: Enumerating objects: 91, done.
remote: Counting objects: 100% (91/91), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 91 (delta 22), reused 87 (delta 21), pack-reused 0
Receiving objects: 100% (91/91), 8.60 MiB | 2.72 MiB/s, done.
Resolving deltas: 100% (22/22), done.
PS C:\Users\iscpt\Desktop\New folder>
```

شکل ۹ - دانلود برنامه از گیت هاب

با دستور `cd CLIPSUI` وارد پوشه می شویم

<https://nodejs.org/en/download>

برای اجرا پروژه نیاز به نصب Node js داریم

اگر روی سیستم نصب دارید با دستور زیر نیازمندی های پروژه را نصب می کنیم.

`npm install`


```

PS C:\Users\iscpt\Desktop\clipsy\CLIPS_UI> npm install
npm WARN deprecated @babel/plugin-proposal-private-methods@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-private-methods instead.
npm WARN deprecated @babel/plugin-proposal-numeric-separator@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-numeric-separator instead.
npm WARN deprecated @babel/plugin-proposal-nullish-coalescing-operator@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-nullish-coalescing-operator instead.
npm WARN deprecated @babel/plugin-proposal-class-properties@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-class-properties instead.
npm WARN deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility
npm WARN deprecated @babel/plugin-proposal-optional-chaining@7.21.0: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-optional-chaining instead.
npm WARN deprecated rollup-plugin-terser@7.0.2: This package has been deprecated and is no longer maintained. Please use @rollup/plugin-terser
npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead
npm WARN deprecated w3c-hr-time@1.0.2: Use your platform's native performance.now() and performance.timeOrigin.
npm WARN deprecated workbox-cacheable-response@6.6.0: workbox-background-sync@6.6.0
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.

added 1568 packages, and audited 1569 packages in 22s

254 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\iscpt\Desktop\clipsy\CLIPS_UI>

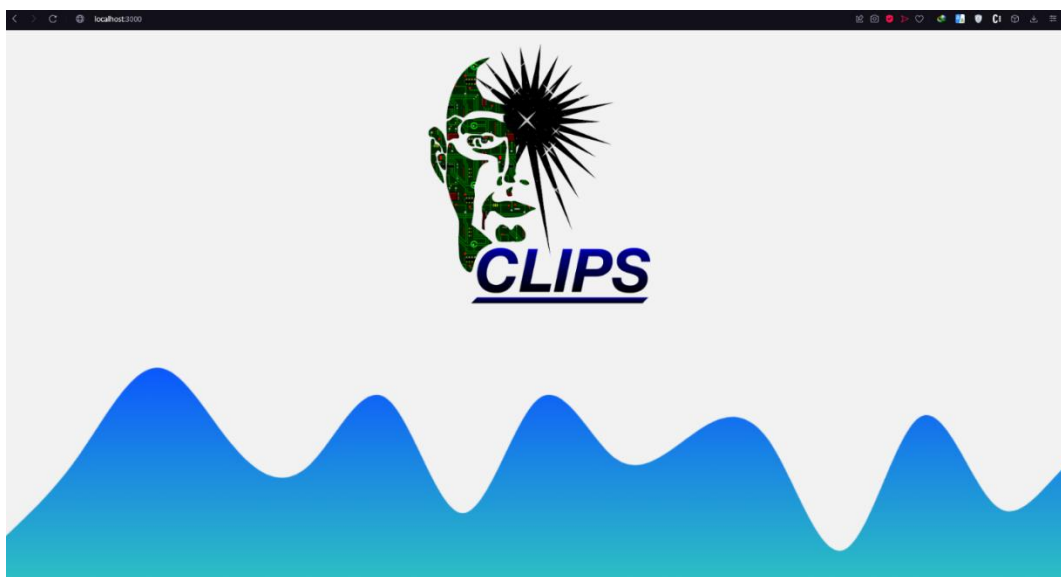
```

شکل ۱۰ - نصب نیازمندی ها

و با دستور زیر برنامه رو اجرا می کنیم.

npm start

برنامه در آدرس localhost:3000 اجرا می شود.



شکل ۱۱ - اجرا شدن برنامه

نحوه کار برنامه

برنامه از ۳ قسمت اصلی تشکیل شده است:

قسمت Fact

قسمت Template

قسمت Rule

The screenshot displays the user interface of the program, which is organized into three main functional areas, each with a corresponding 'ADD' button below it:

- Fact Section:** Located at the top, it contains a text input field labeled 'fact' and a green button labeled 'SUBMIT FACT'.
- Template Section:** Located at the bottom left, it contains three input fields: 'templateName', 'templateDescription', and 'templateDetailNumber'. Below these fields is a green button labeled 'SUBMIT TEMPLATE'. An 'ADD TEMPLATE' button is positioned below the entire section.
- Rule Section:** Located at the bottom right, it contains six input fields arranged in two rows: 'templateName', 'templateDescription', and 'ruleDetailNumber' in the top row; 'operation', 'operationName', and 'question' in the bottom row. Below these fields is a green button labeled 'SUBMIT RULE'. An 'ADD RULE' button is positioned below the entire section.

At the very bottom center of the interface is a green button labeled 'GENERATE CODE'.

شکل ۱۲ - نمای اولیه برنامه

برای نوشتن حقایق حقیقت را نوشته و دکمه submit fact را می زنیم تا حقیقت ثبت شود و برای اضافه کردن حقایق بیشتر دکمه add fact را می زنیم.

شکل ۱۳ – قسمت Fact

Template از ۳ قسمت نام تمپلیت، توضیح تمپلیت و تعداد جزئیات تشکیل شده است. تعداد جزئیات را وارد می کنیم

قسمت اول تایپ مقدار را مشخص می کنیم که Slot است یا MultiSlot

قسمت دوم نام را وارد می کنیم

قسمت سوم هم تایپ داده را وارد می کنیم که مقادیر معتبر آن عبارتند از:

INTEGER
FLOAT
STRING
SYMBOL
MULTIFIELD
FACT_ADDRESS
INSTANCE_NAME
INSTANCE_ADDRESS
EXTERNAL_ADDRESS

مانند قسمت قبل تمپلیت را ثبت و اضافه می کنیم.

شکل ۱۴ - اضافه کردن Template

Rule از ۶ قسمت نام قاعده، توضیح قاعده و تعداد الگوها، عملیات بعد برقراری الگوها و نام عملیات و قسمت پرسش سوال برای گرفتن ورودی تشکیل شده است. تعداد الگوها و خود الگوها را وارد می کنیم.

شکل ۱۵ - اضافه کردن Rule

و در آخر دکمه Generate Code را می زنیم تا خروجی را مشاهده کنیم.

```

<div>
  <resultContext.Provider value={result}>
    <factContext.Provider value={facts}>
      <div style={{backgroundColor: '#2b6777', margin: '1rem', padding: '1rem', borderRadius: '1rem'}}>
        <FactCreate/>
        <div>{factList.map(item => {
          return item
        })}</div></div>
        <div style={center}>
          <Button className="btn" type="submit" variant="contained" sx={{mt: 3, mb: 2}}
            onClick={handleChangeFacts}>
            add Facts
          </Button>
        </div>
      </factContext.Provider>
    <div className="create">
      <div>
        <templateContext.Provider value={templates}>
          <div style={{backgroundColor: '#2b6777', margin: '1rem', padding: '1rem', borderRadius: '1rem'}}>
            <CreateTemplate/>
            <div>{templateList.map(item => {
              return item
            })}</div></div>
            <div style={center}>
              <Button className="btn" type="submit" variant="contained" sx={{mt: 3, mb: 2}}
                onClick={handleChangeTemplate}>
                add template
              </Button>
            </div>
          </templateContext.Provider>
        </div>
      </div>
    </div>
  </div>

```

شکل ۱۶ – بخش اصلی کد

بخش اصلی کد از ۳ کامپوننت FactCreate، CreateTemplate، RuleCreate تشکیل شده است که هر یک وظایف گفته شده را انجام می دهند. برای مدیریت داده ها هم از هوک context استفاده شده و کامپوننت ها را درون Context.Provider قرار گرفته اند.

RuleCreate

```

return (
  <div className="rule">
    <div>
      Rule
    </div>
    <div>
      <Box component="form" onSubmit={handleSubmit} label="template" sx={{m: 2}}>
        <TextField className="field" id="templateName" label="templateName" name="templateName" variant="outlined" sx={{m: 2}}/>
        <TextField
          id="templateDescription"
          label="templateDescription"
          name="templateDescription"
          multiline
          maxRows={7}
          sx={{m: 2}}
          className="field"
        />
        <TextField className="field" id="ruleDetailNumber" label="ruleDetailNumber" name="ruleDetailNumber" variant="outlined" onChange={handleChange} sx={{m: 2}}/>
        <div>{ruleDetail.map(item => {
          return item
        })}</div>
        <TextField className="field" id="operation" label="operation" name="operation" variant="outlined" sx={{m: 2}}/>
        <TextField className="field" id="operationName" label="operationName" name="operationName" variant="outlined" sx={{m: 2}}/>
        <TextField className="field" id="question" label="question" name="question" variant="outlined" sx={{m: 2}}/>
        <Button className="btn" type="submit" fullWidth variant="contained" sx={{mt: 2, mb: 2}}>
          Submit Rule
        </Button>
      </Box>
    </div>
  </div>
)

```

شکل ۱۷ – کد RuleCreate

RuleCreate از ۶ textfield تشکیل شده که ورودی های قواعد را از کاربر می گیرند.

CreateTemplate از ۳ textfield تشکیل شده که ورودی های تمپلیت را از کاربر می گیرند

```
return (
  <div className='template'>
    <div>
      Template
    </div>
    <div>
      <Box component="form" onSubmit={handleSubmit} label="template" sx={{m: 2}}>
        <TextField className="field" id="templateName" label="templateName" name="templateName"
          variant="outlined" sx={{m: 2}} />
        <TextField
          id="templateDescription"
          label="templateDescription"
          name="templateDescription"
          multiline
          maxRows={7}
          sx={{m: 2}}
          className="field"
        />
        <TextField className="field" id="templateDetailNumber" label="templateDetailNumber" name="templateDetailNumber" variant="outlined" onChange={handleChange}
          sx={{m: 2}}/>
        <div>{templateDetail.map(item => {
          return item
        })}</div>
        <Button className="btn" type="submit" fullWidth variant="contained" sx={{mt: 2, mb: 2}}>
          Submit Template
        </Button>
      </Box>
    </div>
  </div>
)
```

شکل ۱۸ – کد CreateTemplate

FactCreate از ۱ textfield تشکیل شده که ورودی حقایق را از کاربر می گیرند

```
return (
  <div className='rule'>
    <div>
      Fact
    </div>
    <div>
      <div className=''>
        <Box component="form" onSubmit={handleSubmit} label="template" sx={{m: 2}}>
          <TextField className="field" id="fact" label="fact" name="fact" variant="outlined" sx={{m: 2}}/>
          <Button className="btn" type="submit" variant="contained" sx={{mt: 3, mb: 2}}>
            Submit Fact
          </Button>
        </Box>
      </div>
    </div>
  </div>
)
```

شکل ۱۹ – کد FactCreate

این سه کامپوننت درون هوک کانتکست قرار می گیرند و درون کامپوننت اصلی Create قرار می گیرند.

قسمت کد هم طبق API دستورات را تولید می کند.

```

facts.forEach((item, index : number) : void => {
    pythonCode += `environment.assert_string('${item.fact}')\n`
    clipsCode += `(assert (${item.fact}))\n`
})
rules.forEach((item, index : number) : void => {
    // console.log(item)
    pythonCode += `DEFRULE_STRING${index}= """ (defrule ${item.ruleName} "${item.ruleDescription}" \n`
    clipsCode += `(defrule ${item.ruleName} "${item.ruleDescription}" \n`
    item?.ruleDetail.forEach(d : T => {
        // console.log(d)
        pythonCode += `(${d.valueType})\n`
        clipsCode += `(${d.valueType})\n`
    })
    pythonCode += `)\n`
    clipsCode += `)\n`
    pythonCode += `(printout ${item.operation})\n`
    clipsCode += `(printout ${item.operation})\n`
    if (item?.question) clipsCode += `(assert ( ${item.question} (read)))\n`
    else clipsCode += `)\n`
    if (item?.question) pythonCode += `(assert ( ${item.question} (read)))"""\n`
    else pythonCode += `)\n`
    pythonCode += `environment.build(DEFRULE_STRING${index})\n`
})
templates.forEach((item, index : number) : void => {
    console.log(item)
    pythonCode += `template_string${index}= """ (deftemplate ${item.templateName} ${item.templateDescription} \n`
    clipsCode += `(deftemplate ${item.templateName} ${item.templateDescription} \n`
    item?.templateDetail.forEach(d : T => {
        pythonCode += `(${d.valueType} ${d.name} ${d.type})\n`
        clipsCode += `(${d.valueType} ${d.name} ${d.type})\n`
    })
    pythonCode += `"""\n`
})
pythonCode += `environment.run()`

```

شکل ۲۰ - قسمت ایجاد کد نهایی

فصل ۵

پیاده سازی یک سناریو

سیستم ساده پزشکی

فرض کنید با مطالعه و پژوهش در مورد یک سیستم ساده پزشکی و یا مصاحبه با پزشکان (منظور افراد خبره در این زمینه) به اطلاعات زیر دست یافتید:

پزشک برای تشخیص بیماری فرد مراجعه کننده، ابتدا از او چند سوال پرسیده و با توجه به پاسخ های فرد بیمار (که همان حقایق می باشد)، اقدام به تشخیص بیماری می کند.

از جمله این سوال ها این است که “محل درد شما کجاست؟”

مثلا فرد در جواب می تواند بگوید “شکم”، “گلو”، “سینه” و “سایر جاها”. حال با توجه به جوابی که در قبال این سوال داده می شود (حقیقت تولید شده) مرحله بعدی شروع می شود.

برای هر جواب در این مرحله، مراحل بعدی می تواند متفاوت باشد. مثلا اگر محل درد “شکم” انتخاب شود، تشخیص داده شود که فرد “آپاندیس” دارد

یا اگر محل درد “گلو” انتخاب شود در مرحله بعدی از فرد پرسیده شود که آیا “تب” دارد؟ اگر در جواب این سوال بگوید “بله”، تشخیص داده شود که بیماری “گلودرد باکتریایی” است و اگر “خیر”، تشخیص داده شود که فرد “گلودرد ویروسی” دارد

اگر محل درد “سینه” انتخاب شود، تشخیص داده شود که فرد “سکته” کرده است.

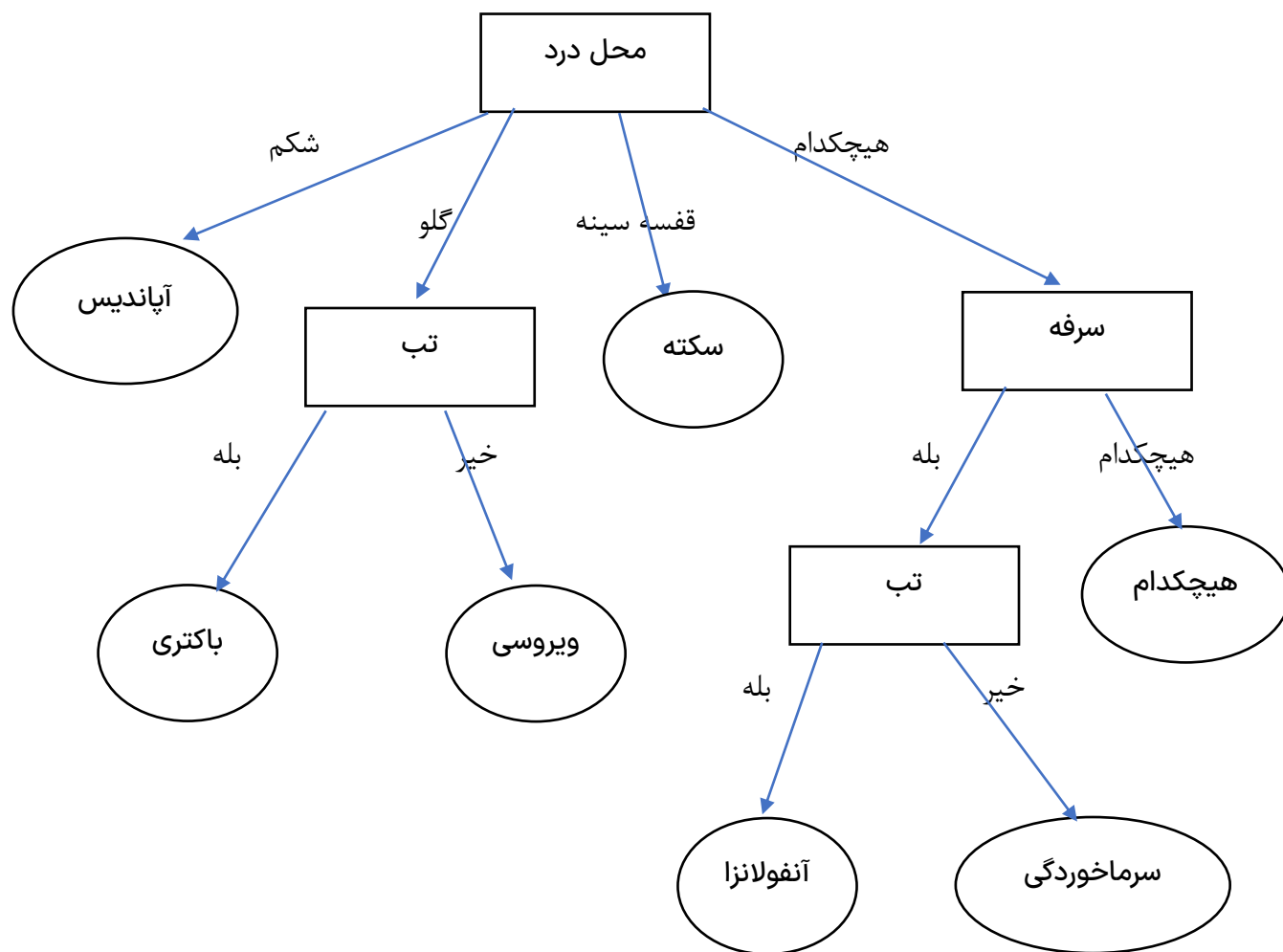
یا اگر محل درد “سایر” انتخاب شود در مرحله بعدی از فرد پرسیده شود که آیا “سرفه” می کند؟ از در جواب این سوال بگوید “خیر”، این سیستم ساده نتوانسته بیماری فرد را تشخیص دهد ولی اگر “بله” انتخاب شود در مرحله بعدی از فرد پرسیده شود که آیا “تب” دارد؟ اگر در جواب این سوال بگوید “بله”، تشخیص داده شود که بیماری “آنفلانزا” است و اگر “خیر”، تشخیص داده شود که فرد دچار “سرماخوردگی” شده است.

نکته: در هر مرحله هر سوال را یک قانون تصور نمود که با توجه به حقایق بدست آمده در مرحله قبلی اجرا

می‌شود.

درخت تصمیم گیری

شما می‌توانید از آنچه به عنوان اطلاعات در بخش قبلی به دست آورده‌اید یک درخت تصمیم‌گیری ایجاد کنید. البته توجه کنید این درخت سلسله‌مراتبی است و از ریشه به سمت برگ‌ها به جلو می‌رود. بنابراین در بعضی سیستم‌ها ترتیب این مراحل می‌تواند در تصمیم‌گیری بسیار مهم باشد.



شکل ۲۱ - درخت تصمیم‌گیری سناریو

حالا قواعد را مانند شکل وارد برنامه می‌کنیم.

Menu

=>

t crlf crlf crlf

" mahal dard kojast?lotfan yki az gozinehaye zir ra entekhab konid" crlf crlf

" 1.) shekam. " crlf crlf

" 2.) galo." crlf crlf

" 3.) sineh." crlf crlf

" 4.) sayer." crlf crlf

" 5.) EXIT OF SYSTEM.." crlf crlf crlf

" Your answer: "

(selectedindex)

A-shekam

selectedindex 1

=>

t crlf crlf crlf" shoma bimary Apandis darid " crlf

crlf " Thank you for using my Program...

"crlf crlf

The image displays two instances of a 'Rule' configuration form. Each form contains several input fields and a 'SUBMIT RULE' button.

Top Form:

- templateName: Menu
- templateDescription: (empty)
- ruleDetailNumber: 0
- operation: t crlf crlf crlf " mahal dard ki
- operationName: print
- question: selectedindex

Bottom Form:

- templateName: A-shekam
- templateDescription: (empty)
- ruleDetailNumber: 1
- pattern-0: selectedindex 1
- operation: t crlf crlf crlf " shoma bimary
- operationName: print
- question: question

شکل ۲۲-وارد کردن قواعد سناریو

Q-galo-tab

selectedindex 2

=>

t crlf crlf crlf " aya shoma tab darid? (Yes | No) " crlf crlf " Your answer: "
ifYesNoTab

A-galo-tab-yes

(selectedindex 2)

(ifYesNoTab yes)

=>

t crlf crlf crlf "shoma glodard bakterraei darid " crlf crlf

A-galo-tab-no

(selectedIndex 2)

(ifYesNoTab no)

=>

t crlf crlf crlf "shoma glodard virosi darid " crlf crlf

A-sineh

(selectedIndex 3)

=>

t crlf crlf crlf" shoma sekteh kardeid " crlf crlf " Thank you for using my
Program... "crlf crlf

Q-sayer-sorfeh

(selectedIndex 4)

=>

t crlf crlf crlf " aya shoma solfe mikonid? (Yes | No) " crlf crlf " Your answer: "
(ifYesNoSorfeh)

Q-sayer-sorfeh-yes-tab ""

(selectedIndex 4)

(ifYesNoSorfeh yes)

=>

t crlf crlf crlf " aya shoma tab darid? (Yes | No) " crlf crlf " Your answer: "

(ifYesNoTab)

A-sayer-sorfeh-yes-tab-yes ""

(selectedIndex 4)

(ifYesNoSorfeh yes)

(ifYesNoTab yes)

=>

t crlf crlf crlf" shoma anfolanza darid " crlf crlf " Thank you for using my
Program... "crlf crlf

A-sayer-sorfeh-yes-tab-no ""

(selectedIndex 4)

(ifYesNoSorfeh yes)

(ifYesNoTab no)

=>

t crlf crlf crlf" shoma sarma khordid" crlf crlf " Thank you for using my Program...
"crlf crlf

A-sayer-sorfeh-no ""

(selectedIndex 4)

(ifYesNoSorfeh no)

=>

t " moteasefane nmitavanam bimari shoma ra tashkhis daham" crlf crlf " Thank
you for using my Program... "crlf crlf

Rule

templateName A-sayer-solfeh-no	templateDescription	ruleDetailNumber 2
pattern-0 selectedindex 4		
pattern-1 ifYesNoSolfeh no		
operation t " moteasefane nmitavanam bimari shoma ra tashkhis daham" crlf crlf " Thank you for using my Program... "crlf crlf	operationName print	question

SUBMIT RULE

شکل ۲۳ -وارد کردن قواعد سناریو

و نتیجه دو کد به زبان های پایتون و کلیپس خروجی می دهد که باید آن ها را اجرا کنیم.

```

defrule ""
  (printout t "crif crif crif " ahal dard kujestjefsten jil as goinehaye zir ra entehab kundi" crif crif " 1.) shekam. " crif crif " 2.) galo." crif crif " 3.) simeh." crif crif " 4.) sayer." crif crif " 5.) EXIT OR SYSTEM..." crif crif " Your answer: ")
  (defrule a-shekam ""
    (selectindex 1)
  )
  (printout t "crif crif crif" shoma binary Agendis darid " crif crif " Thank you for using my Program..." "crif crif ")
  (defrule a-shekam ""
    (selectindex 1)
  )
  (printout t "crif crif crif" shoma binary Agendis darid " crif crif " Thank you for using my Program..." "crif crif ")
  (defrule a-galo-tab ""
    (selectindex 2)
  )
  (printout t "crif crif crif " aya shoma tab darid? (Yes | No) " crif crif " Your answer: ")
  (assert ( ifreshTab (read)))
  (defrule a-galo-tab-yes ""
    (selectindex 3)
  )
  (ifreshTab yes)
  )
  (printout t "crif crif crif " shoma glendar baterasi darid " crif crif ")
  (defrule a-galo-tab-no ""
    (selectindex 3)
  )
  (ifreshTab no)
  )
  (printout t "crif crif crif" shoma glendar virezi darid " crif crif ")
  (defrule a-simeh ""
    (selectindex 3)
  )
  (printout t "crif crif crif" shoma seteh kareid " crif crif " Thank you for using my Program..." "crif crif ")
  (defrule a-sayer-solfen ""
    (selectindex 4)
  )
  (assert ( ifreshSolfen (read)))
  (defrule a-sayer-solfen-yes-tab ""
    (selectindex 4)
  )
  (ifreshSolfen yes)
  )
  (printout t "crif crif crif " aya shoma tab darid? (Yes | No) " crif crif " Your answer: ")
  (assert ( ifreshTab (read)))
  (defrule a-sayer-solfen-yes-tab-yes ""
    (selectindex 4)
  )
  (ifreshSolfen yes)
  )
  (ifreshTab yes)
  )
  (printout t "crif crif crif" shoma enfance darid " crif crif " Thank you for using my Program..." "crif crif ")
  (defrule a-sayer-solfen-yes-tab-no ""
    (selectindex 4)
  )
  )

Export clips
environment < clips.environment()
DEFPRLG_STRINGS "" (defrule memo ""
  (printout t "crif crif crif " ahal dard kujestjefsten jil as goinehaye zir ra entehab kundi" crif crif " 1.) shekam. " crif crif " 2.) galo." crif crif " 3.) simeh." crif crif " 4.) sayer." crif crif " 5.) EXIT OR SYSTEM..." crif crif " Your answer: ")
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-shekam ""
    (selectindex 1)
  )
  (printout t "crif crif crif" shoma binary Agendis darid " crif crif " Thank you for using my Program..." "crif crif ")
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-galo-tab ""
    (selectindex 2)
  )
  (printout t "crif crif crif " aya shoma tab darid? (Yes | No) " crif crif " Your answer: ")
  (assert ( ifreshTab (read)))
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-galo-tab-yes ""
    (selectindex 2)
  )
  (ifreshTab yes)
  )
  (printout t "crif crif crif " shoma glendar baterasi darid " crif crif ")
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-galo-tab-no ""
    (selectindex 2)
  )
  (ifreshTab no)
  )
  (printout t "crif crif crif " shoma glendar virezi darid " crif crif ")
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-simeh ""
    (selectindex 3)
  )
  (printout t "crif crif crif" shoma seteh kareid " crif crif " Thank you for using my Program..." "crif crif ")
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-sayer-solfen ""
    (selectindex 4)
  )
  (assert ( ifreshSolfen (read)))
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-sayer-solfen-yes-tab ""
    (selectindex 4)
  )
  (ifreshSolfen yes)
  )
  (printout t "crif crif crif " aya shoma tab darid? (Yes | No) " crif crif " Your answer: ")
  (assert ( ifreshTab (read)))
  environment.built(DEFPRLG_STRINGS)
  DEFPRLG_STRINGS "" (defrule a-sayer-solfen-yes-tab-yes ""
    (selectindex 4)
  )
  (ifreshSolfen yes)
  )
  (ifreshTab yes)
  )
  )

```

شکل ۲۴ - کد نهایی به زبان پایتون و کلیپس

```

import clips
environment = clips.Environment()
DEFRULE_STRING0= "" (defrule Menu ""
=>
(printout t crlf crlf crlf " mahal dard kojast?lotfan yki az gozinehaye zir ra entekhab konid" crlf crlf " 1.) shekam. " crlf crlf " 2.) galo," crlf crlf " 3.) sineh," crlf crlf " 4.)
sayer," crlf crlf " 5.) EXIT or SYSTEN.." crlf crlf crlf " Your answer: ")
(assert ( selectedindex (read))))""
environment.build(DEFRULE_STRING0)
DEFRULE_STRING1= "" (defrule A-shekan ""
(selectedindex 1)
=>
(printout t crlf crlf crlf" shoma binary Apanidis darid " crlf crlf " Thank you for using my Program... "crlf crlf )
)""
environment.build(DEFRULE_STRING1)
DEFRULE_STRING2= "" (defrule A-shekan ""
(selectedindex 1)
=>
(printout t crlf crlf crlf" shoma binary Apanidis darid " crlf crlf " Thank you for using my Program... "crlf crlf )
)""
environment.build(DEFRULE_STRING2)
DEFRULE_STRING3= "" (defrule Q-galo-tab ""
(selectedindex 2)
=>
(printout t crlf crlf crlf " aya shoma tab darid? (Yes | No) " crlf crlf " Your answer: " )
(assert ( ifYesNoTab (read))))""
environment.build(DEFRULE_STRING3)
DEFRULE_STRING4= "" (defrule A-galo-tab-yes ""
(selectedindex 2)
(ifYesNoTab yes)
=>
(printout t crlf crlf crlf "shoma glodard bakteraei darid " crlf crlf)
)""
environment.build(DEFRULE_STRING4)
DEFRULE_STRING5= "" (defrule A-galo-tab-no ""
(selectedindex 2)
(ifYesNoTab no)
=>
(printout t crlf crlf crlf "shoma glodard virosi darid " crlf crlf)
)""
environment.build(DEFRULE_STRING5)
DEFRULE_STRING6= "" (defrule A-sineh ""
(selectedindex 3)
=>
(printout t crlf crlf crlf" shoma sekteh kardaid " crlf crlf " Thank you for using my Program... "crlf crlf)
)""
environment.build(DEFRULE_STRING6)
DEFRULE_STRING7= "" (defrule Q-sayer-solfeh ""
(selectedindex 4)
=>
(printout t crlf crlf crlf " aya shoma solfe mikonid? (Yes | No) " crlf crlf " Your answer: ")
(assert ( ifYesNoSolfeh (read))))""
environment.build(DEFRULE_STRING7)
DEFRULE_STRING8= "" (defrule Q-sayer-solfeh-yes-tab ""
(selectedindex 4)
(ifYesNoSolfeh yes)
=>
(printout t crlf crlf crlf " aya shoma tab darid? (Yes | No) " crlf crlf " Your answer: ")
(assert ( ifYesNoTab (read))))""
environment.build(DEFRULE_STRING8)
DEFRULE_STRING9= "" (defrule A-sayer-solfeh-yes-tab-yes ""
(selectedindex 4)
(ifYesNoSolfeh yes)
(ifYesNoTab yes)
=>
(printout t crlf crlf crlf" shoma anfolanza darid " crlf crlf " Thank you for using my Program... "crlf crlf)
)""
environment.build(DEFRULE_STRING9)
DEFRULE_STRING10= "" (defrule A-sayer-solfeh-yes-tab-no ""
(selectedindex 4)
(ifYesNoSolfeh yes)
(ifYesNoTab no)
=>
(printout t crlf crlf crlf" shoma sarma khordid" crlf crlf " Thank you for using my Program... "crlf crlf)
)""
environment.build(DEFRULE_STRING10)
DEFRULE_STRING11= "" (defrule A-sayer-solfeh-no ""
(selectedindex 4)
(ifYesNoSolfeh no)
=>
(printout t " moteasefane nmitavanam bimari shoma ra tashkhis daham" crlf crlf " Thank you for using my Program... "crlf crlf)
)""
environment.build(DEFRULE_STRING11)
environment.run()

```

شکل ۲۵- کد نهایی در محیط پایتون


```
mahal dard kojast?lotfan yki az gozinehaye zir ra entekhab konid

1.) shekam.
2.) galo.
3.) sineh.
4.) sayar.
5.) EXIT OF SYSTEM..

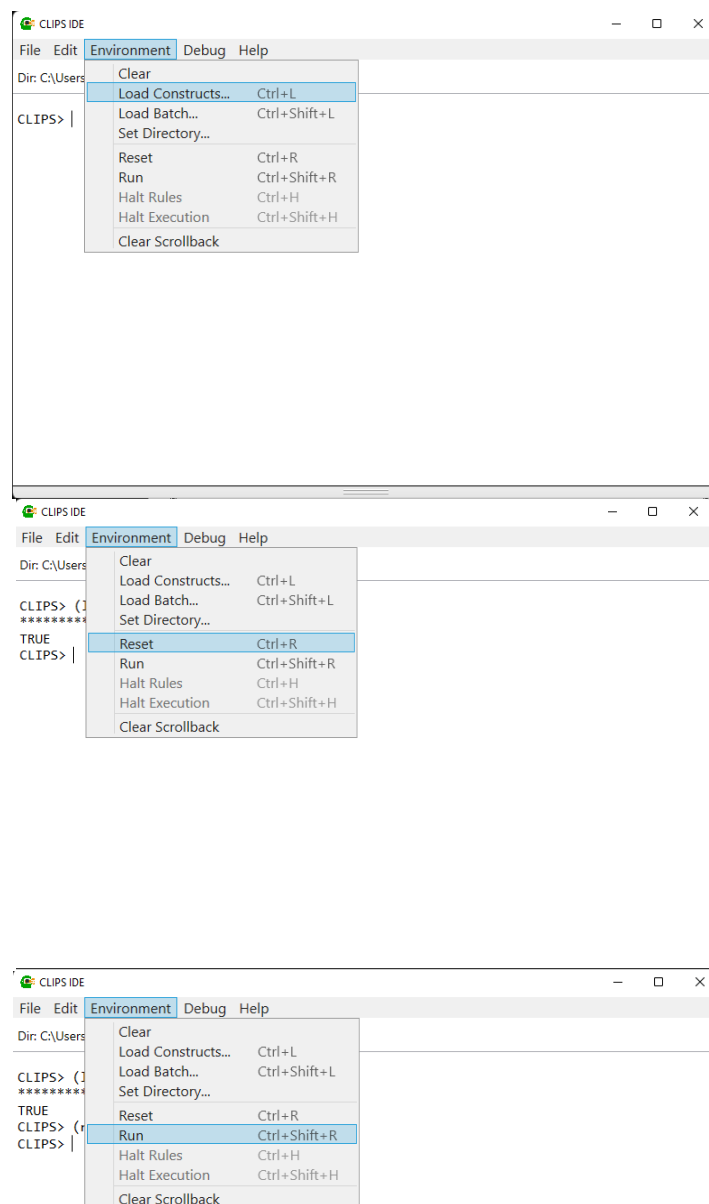
Your answer: 1

shoma bimary Apandis darid

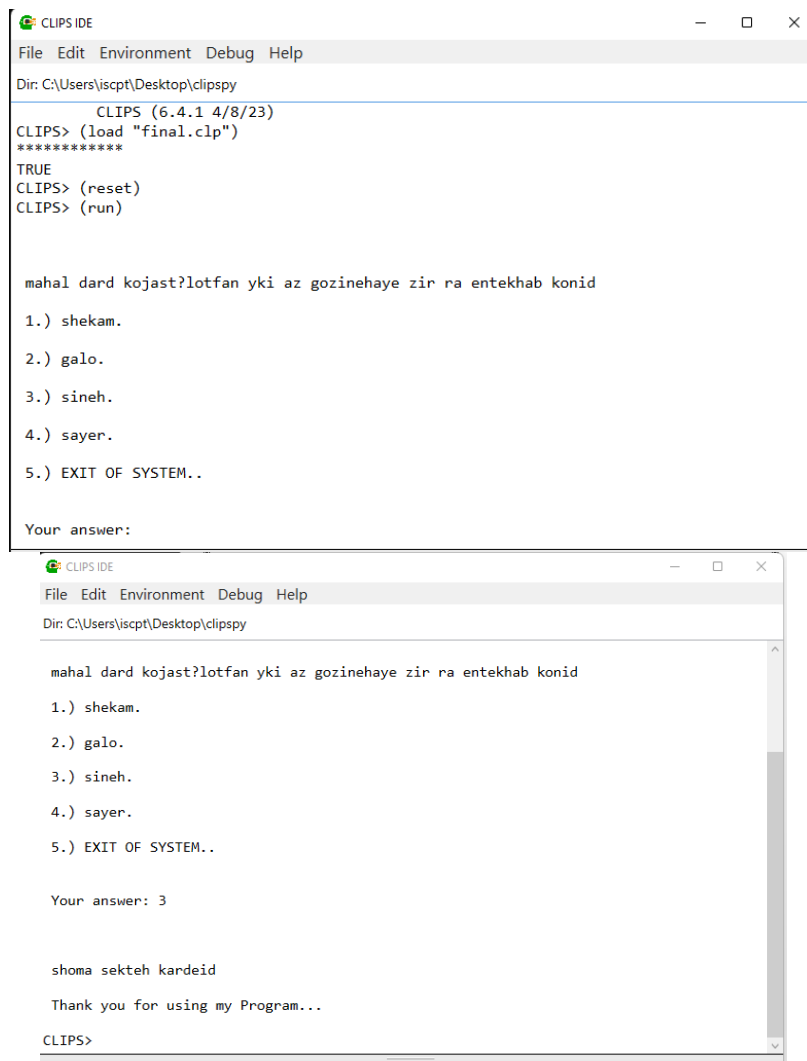
Thank you for using my Program...
```

شکل ۲۶ – اجرای کد پایتون و استفاده از کلیپس در محیط پایتون

برای اجرای برنامه در کلیپ مانند شکل عمل می کنیم



شکل ۲۷- اجرای کد در محیط کلیپس



شکل ۲۸- اجرای کد در محیط کلیپس

فصل ۶

جمع‌بندی

سیستم خبره به عنوان یکی از شاخه‌های مهم هوش مصنوعی محسوب می‌شود که کاربرد آن، ارائه راه‌حل منطقی و صحیح برای حل مسائل تخصصی است. این نوع سیستم‌ها نقش مهمی در حل مسائل بحرانی مانند تشخیص بیماری، تشخیص درمان بیماری، تشخیص کلاهبرداری و افزایش میزان سوددهی بیشتر دارند. با توجه به اهمیت این شاخه از علوم کامپیوتر و پژوهش‌های بسیاری که به این حوزه در سال‌های اخیر تخصیص داده شده است.

یک سیستم خبره یک سیستم تصمیم‌گیری مبتنی بر کامپیوتر تعاملی و قابل اعتماد است که از حقایق و اکتشافات برای حل مسائل پیچیده تصمیم‌گیری استفاده می‌کند.

اجزای کلیدی یک سیستم خبره عبارتند از: (۱) رابط کاربری، (۲) موتور استنتاج، (۳) پایگاه دانش شرکت کنندگان کلیدی در توسعه سیستم‌های خبره هوش مصنوعی عبارتند از: (۱) متخصص دامنه (۲) مهندس دانش (۳) کاربر نهایی

بهبود کیفیت تصمیم‌گیری، کاهش هزینه، ثبات، قابلیت اطمینان، سرعت از مزایای کلیدی یک سیستم خبره است

یک سیستم خبره نمی‌تواند راه حل‌های خلاقانه‌ای ارائه دهد و نگهداری آن هزینه‌بر است.

واضح است که ES Shells به جای توسعه یک سیستم خبره از ابتدا گزینه بسیار بهتری است. ES Shell های زیادی وجود دارند که از زبان‌های برنامه‌نویسی مختلف پشتیبانی می‌کنند که توسعه یافته‌اند و دروازه‌هایی را برای توسعه دهندگانی که با زبان‌های برنامه‌نویسی مختلف آشنا هستند باز می‌کنند. پوسته‌های ES باید عمده‌تاً با توجه به الزامات و مقیاس/اندازه/پیچیدگی سیستم خبره‌ای که باید توسعه داده شود به درستی انتخاب شوند.

رابط کاربری حیاتی‌ترین بخش نرم‌افزار سیستم خبره است. این کامپوننت پرس و جوی کاربر را به شکلی خوانا می‌گیرد و به موتور استنتاج ارسال می‌کند. پس از آن، نتایج را به کاربر نمایش می‌دهد. به عبارت دیگر، این یک رابط است که به کاربر کمک می‌کند تا با سیستم خبره ارتباط برقرار کند.

در مطلب حاضر سعی داشتیم به معرفی جامعی از این حوزه بپردازیم و به اجزای اصلی این سیستم‌ها، ویژگی‌ها و کاربردهای آن‌ها اشاره کنیم تا علاقه‌مندان به این حوزه بتوانند با مطالعه این مطلب، به اطلاعات اولیه‌ای از آن دست پیدا کنند.

- 1- <https://www.clipsrules.net>
- 2- <https://clipspy.readthedocs.io/en/latest/>
- 3- <https://www.clipsrules.net/documentation/v641/bpg641.pdf>
- 4- <https://www.clipsrules.net/documentation/v631/bpg631.pdf>
- 5- <https://www.clipsrules.net/documentation/v624/bpg624.pdf>
- 6- <https://www.mygreatlearning.com/blog/expert-systems-in-artificial-intelligence/>
- 7- https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm
- 8- <https://www.javatpoint.com/expert-systems-in-artificial-intelligence>
- 9- <https://www.guru99.com/expert-systems-with-applications.html#11>
- 10- <https://digitalthinkerhelp.com/expert-system-in-artificial-intelligence-with-applications-examples-types-uses/>
- 11- <https://medium.com/nerd-for-tech/an-introduction-to-expert-system-shells-530043914ec0>
- 12- https://www.researchgate.net/publication/263278128_Basic_Concepts_of_Expert_System_Shells_and_an_efficient_Model_for_Knowledge_Acquisition
- 13- https://cinuresearch.tripod.com/ai/www-cee-hw-ac-uk/alison/ai3notes/subsection2_5_2_1.html
- 14- https://www.researchgate.net/publication/235028224_The_Applicability_and_Limitations_of_Expert_System_Shells
- 15- <http://Parsbook.org>
- 16- <http://tehranit.net/آموزش-سریع-کلیپس-سیستم-خبره-۴-ساعت/>
- 17- <https://blog.faradars.org/سیستم-خبره-چیست/>
- 18- <https://rk-export.blogspot.com/1393/01/28/post-1-سیستم-های-خبره-1/>

<https://github.com/aliasadi78/ClipsUI.git>

کد برنامه در آدرس



Computer Engineering Department

Bachelor's final project

Design and implementation of graphic user interface for Clips software with
adaptability feature

Ali Asadi

Supervisor

Dr. Kangavari