



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 نمی رسید

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 تولید این رابط کاربری از زبان برنامه نویسی جاوا اسکریپت و کتابخانه react استفاده شده و 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 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.

۱. مقدمه.....	۱۱
۲. سیستم خبره.....	۱۵
۳. برنامه کلیپس.....	۳۰
۴. راه اندازی برنامه.....	۴۷
۵. نحوه کار برنامه.....	۴۹
۶. پیاده سازی یک سناریو.....	۵۵
۷. جمع بندی.....	۶۶
۸. منابع و مراجع.....	۶۷

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 شده‌است.

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 شده‌است 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)

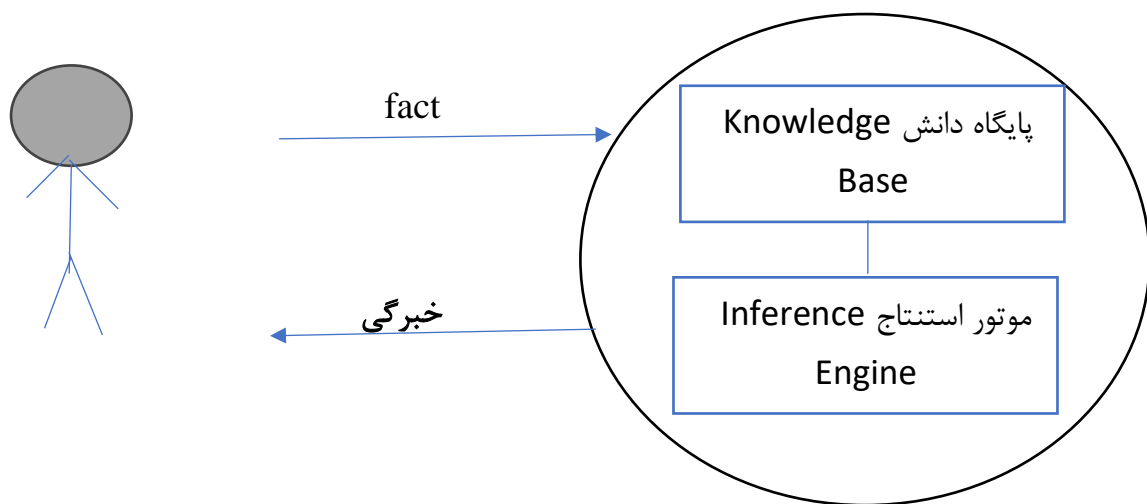
الگوریتم‌های ژنتیک (Genetic Algorithms)

سامانه‌های منطق فازی (Fuzzy Logic Systems)

سیستم خبره یک برنامه کامپیوتری هوشمند است که از دانش و روش‌های 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



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

قواعد rule

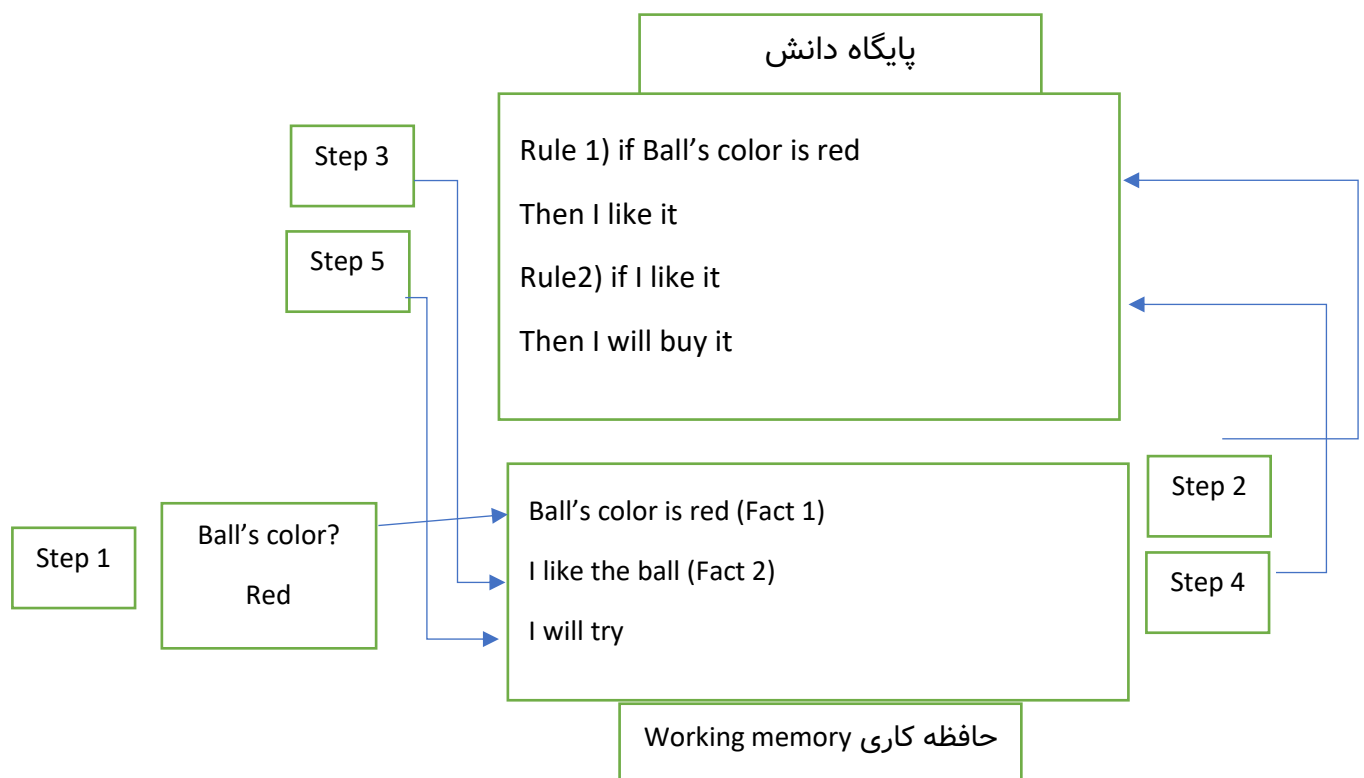
ساختار دانشی است که یعنی اطلاعات دانسته به اطلاعات دیگری می تواند به معلومات قبلی اضافه و یا دانسته فرض شوند. ساختار قاعده یا چند پیش فرض در قسمت "اگر" را با نتیجه گیری ها در قسمت "آنگاه" مرتبط می سازد. یک قاعده می تواند "در غیر این صورت" هم داشته باشد.

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

If pattern

Then action

Else action



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 Twill Buy It اجرا می شود.

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 های خبره دارای ویژگی‌هایی هستند که در ادامه به آن‌ها اشاره شده است:

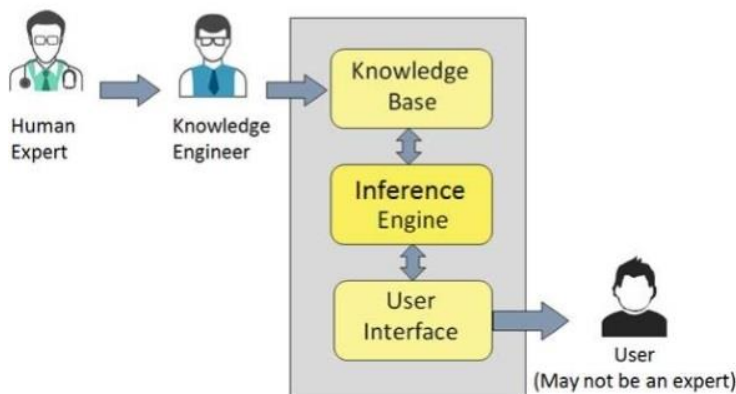
- **کارایی بالا**: سیستم های خبره را می‌توان برای حل مسائل مختلف به کار برد و از نتیجه‌گیری و استدلال منطقی آن برای تصمیم‌گیری‌های مهم استفاده کرد.
- **ارائه نتایج قابل فهم**: سیستم خبره می‌تواند با زبان انسان با کاربر ارتباط برقرار کند و خروجی را نیز به زبان قابل درک انسان ارائه دهد.
- **ارائه نتایج معتبر**: از آنجا که سیستم های خبره بر اساس واقعیت‌ها، تجربه‌ها و استنتاج‌های منطقی گذشته به تحلیل مسائل می‌پردازند، نتایج و خروجی‌هایی را ارائه می‌دهند که تا حد زیادی دقیق و کارآمد هستند.
- **ارائه پاسخ در زمان کوتاه**: سیستم های خبره می‌توانند در کوتاه‌ترین زمان ممکن به حل پیچیده‌ترین مسائل بپردازند.

اجزای سیستم های خبره چیست ؟

سیستم های خبره از سه جزء اصلی تشکیل شده‌اند 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.



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 می شود.

میزان موفقیت سیستم های 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، پژوهش گر و مهندس دانش آن ها را بررسی کرده و در نهایت مهندس دانش، اطلاعات را در قالبی مشخص در پایگاه دانش ذخیره می کند.

پایگاه دانش در سیستم های خبره چیست ؟

در سیستم های خبره، پایگاه دانش به عنوان حافظه ای محسوب می شود که دانش های استنتاج شده از سیستم های خبره مختلف را در خود نگهداری می کند. هر چقدر میزان اطلاعات موجود در این پایگاه های دانش بیشتر باشد، سیستم های خبره با دقت بیشتری درباره مسائل مختلف تصمیم می گیرند.

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 سیستم های خبره، رابط کاربری است که به منظور تعامل با کاربر و دریافت پرسش های آنها در قالبی مشخص طراحی می شوند. رابط کاربری پس از دریافت کوئری ها، آنها را به موتور استنتاج ارسال می کند. در نهایت، موتور استنتاج پاسخ خود را به رابط کاربری می فرستد تا آنها را به عنوان خروجی به کاربر نمایش دهد.

بدین ترتیب، می توان به طور کلی بیان کرد که رابط کاربری به کاربران مبتدی و غیر حرفه ای کمک می کند تا به منظور یافتن حل مسئله، با سیستم خبره ارتباط برقرار کنند. در طراحی رابط کاربری سیستم های خبره، از روش های «پردازش زبان طبیعی» (Natural Language Processing | NLP) استفاده می شود تا سیستم بتواند درخواست کاربر را درک کند.

موتور استنتاج در سیستم خبره

موتور استنتاج به عنوان مغز سیستم های خبره محسوب می شود و وظیفه پردازش اصلی سیستم را بر عهده دارد. موتور استنتاج از قوانین استنتاجی استفاده می کند تا با استخراج دانش از پایگاه دانش، درباره مسئله ای تصمیم بگیرد یا به اطلاعات جدیدی دست یابد.

سیستم های خبره جدید از الگوریتم های «یادگیری عمیق» (Deep Learning) و مدل های «یادگیری ماشین» (machine Learning) بهره گرفته اند تا در حل مسائل، رفتار و داوری انسان هوشمند را شبیه سازی کنند. با دریافت تجربه های بیشتر، سیستم های خبره عملکرد خود را بهبود می بخشند.

دو نوع موتور استنتاج در سیستم های خبره استفاده می شوند 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 واقعیتهای صورت می گیرد و استنتاج نهایی موتور، دقیق است.

۲. موتورهای استنتاج احتمالاتی (Probabilistic Inference Engine): این نوع از موتورهای استنتاجی، درباره مسائل مختلف بر پایه احتمالات نتیجه گیری می کنند و نتایج قطعی ارائه نمی دهند.

رویکرد حل مسئله در سیستمهای خبره

موتور استنتاج از دو روش برای استخراج اطلاعات از پایگاه دانش و یافتن راه حل برای مسئله استفاده می کنند.

این دو روش در ادامه فهرست شده اند:

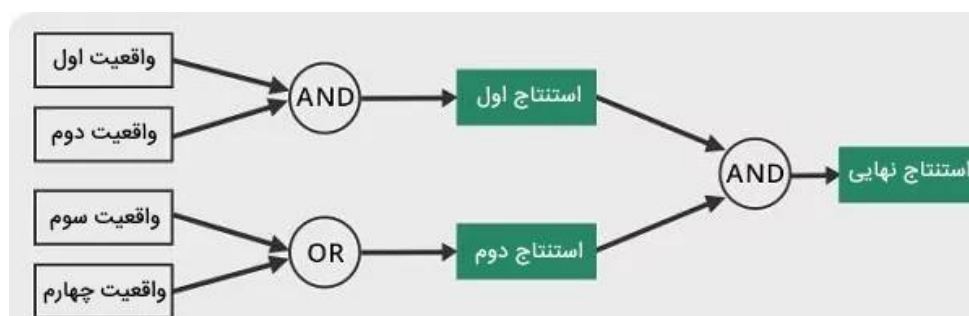
۱. زنجیرسازی رو به جلو (Forward Chaining)

۲. زنجیرسازی رو به عقب (Backward Chaining)

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

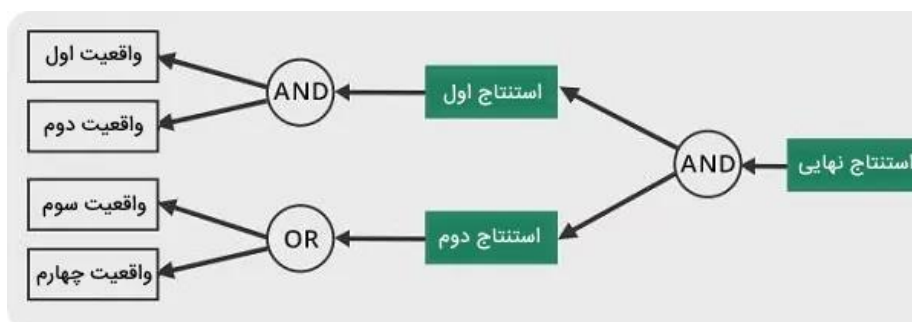
حل مسئله با رویکرد زنجیرسازی رو به جلو چیست ؟

موتور استنتاج در روش زنجیرسازی رو به جلو، بر اساس قواعد و واقعیت مشخص شروع به استدلال می کند و تا در نهایت نتیجه گیری خود را به واقعیات اضافه کند. به عبارتی، این نوع موتورها، زنجیره ای از شرطها را دنبال می کنند تا در نهایت به نتیجه گیری 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 تصمیم گیری موتورهای استنتاج بر پایه روش زنجیرسازی رو به جلو دیده می شود.



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.



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 سیستم های خبره چیست ؟

سیستم های خبره را می توان به ۶ نوع تقسیم کرد که در ادامه به توضیح هر یک از آن ها پرداخته شده است:

- سیستم های خبره قاعده مند (Rule Based Expert Systems)
- سیستم های خبره فازی (Fuzzy Expert Systems)
- سیستم های خبره مبتنی بر قاب (Frame Based Expert Systems)
- سیستم های خبره ترکیبی (Hybrid Expert Systems)
- سیستم های خبره عصبی (Neural Expert Systems)
- سیستم های خبره فازی - عصبی (Neuro - Fuzzy Expert Systems)

در ادامه مطلب، به توضیح ویژگی های هر یک از انواع سیستم های خبره پرداخته می شود.

سیستم خبره قاعده مند چیست ؟

سیستم خبره قاعده مند به عنوان اولین نوع از سیستم های خبره شناخته می شود. این نوع از سیستم های خبره شامل مجموعه ای از قواعد شرطی هستند که داده ها را می توان در قالب این دستورات شرطی قرار داد تا موتور استنتاج بر اساس این قواعد، به نتیجه گیری بپردازد.

سیستم های خبره فازی چه هستند؟

چنانچه داده های درون پایگاه دانش عباراتی را شامل شوند که مفاهیم آن ها دارای ابهام باشند، از سیستم های خبره فازی برای استنتاج استفاده می شود که بر اساس نظریه فازی پیاده سازی شده اند. عبارت هایی نظیر «بسیار بلند» یا «بسیار سبک» دارای مفاهیمی هستند که مقدار دقیق کمیت آن ها مشخص نیست. بدین ترتیب، برای استنتاج از چنین داده هایی باید از سیستم های خبره فازی استفاده شود.

سیستم خبره مبتنی بر قاب چیست ؟

در سیستم های خبره مبتنی بر قاب ، از مفهوم «قاب (Frame)» به منظور نگهداری دانش استفاده می شود. قاب را می توان به عنوان «ساختمان داده (Data Structure)» تلقی کرد که به یک شیء یا مفهوم اشاره دارد. هر قاب دارای نام و مجموعه ای از ویژگی ها است که هر کدام از آن ها، دارای مقادیر خاصی هستند.

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

از قالب به منظور سازمان‌دهی دانش موجود در پایگاه دانش استفاده می‌شود. از این نوع ساختار می‌توان نیز به راحتی می‌توان در شی گزایی استفاده کرد.

سیستم خبره ترکیبی چیست ؟

در طراحی سیستم‌های خبره ترکیبی یا هیبریدی از مزیت‌های سیستم‌های خبره قاعده‌مند، فازی و سیستم خبره مبتنی بر قالب 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.

فناوری های لازم برای توسعه سیستم های خبره

به منظور طراحی سیستم خبره از چندین تکنولوژی استفاده می شود که در ادامه به آن ها می پردازیم.

- محیط توسعه (IDE) برای ساخت سیستم های خبره
 - ابزارها و سخت افزارهای لازم برای توسعه سیستم های خبره:
 - کامپیوتر، ریز کامپیوتر و MainFrame ها
- زبان های برنامه نویسی نظیر LISP و PROLOG
 - پایگاه های 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 می کنیم ؟

پیش از این که از هر گونه فناوری جدید استفاده کنیم، باید به این موضوع پردازیم که چرا چنین فناوری های جدیدی ارائه شده اند و چه نیازی وجود داشته است که علیرغم وجود پژوهشگران، دانشمندان و افراد خبره در حوزه های مختلف علوم، سیستم های کامپیوتری جدیدی برای حل مسائل طراحی می شوند؟

در ادامه، به مهم ترین دلایل ساخت سیستم های خبره اشاره شده است:

- عدم محدودیت حافظه :حجم زیادی از داده های مورد نیاز سیستم های کامپیوتری را می توان در حافظه ذخیره و به طور 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 به طور کامل به یاد آورد.
- بازدهی بالا :سیستم های مبتنی بر کامپیوتر می توانند در کوتاه ترین زمان به حل پیچیده ترین مسائل پردازند. همچنین، می توان با به روز رسانی مکرر داده های (دانش) آن ها، میزان دقت خروجی آن ها 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 به صورت 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 سیستم‌های خبره تصمیم بگیرند به چه کسانی وام تعلق گیرد.
- کاربرد سیستم‌های خبره در حوزه پزشکی: یکی از مهم‌ترین کاربردهای سیستم‌های خبره در مسائل پزشکی برای تشخیص بیماری مریضان و تجویز دارو و پیشنهاد روش درمان است.

- کاربرد سیستم های خبره در سازمان ها: مدیران سازمان ها در راستای اتخاذ تصمیمات مهم شرکت برای سوددهی بیشتر، می توانند 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 برنامه ریزی: یکی دیگر از کاربردهای وسیع سیستم های خبره در برنامه ریزی زمانی است که از آن ها در برنامه ریزی خطوط هوایی و آژانس های هواپیمایی استفاده می شود.

مثال هایی از سیستم خبره

برنامه های نرم افزاری مختلفی بر پایه سیستم های خبره طراحی شده اند که در ادامه به برخی از آن ها اشاره می شود:

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 می تواند باکتری های مختلفی را شناسایی کند که باعث عفونت شدید می شوند. به علاوه، این سیستم می تواند بر اساس وزن اشخاص، داروهای مختلفی را برای درمان بیماری آن ها پیشنهاد دهد.

سیستم DENDRAL: این سیستم خبره مبتنی بر روش های هوش مصنوعی است و از آن برای تحلیل مسائل شیمی استفاده می شود. به عبارتی، این سیستم می تواند با استفاده از دانش مربوط به طیف شناسی ماده های مخالف، ساختار مولکولی آن ها را پیش بینی کند.

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

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

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 کمتر 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 نمی‌توانند 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.

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.

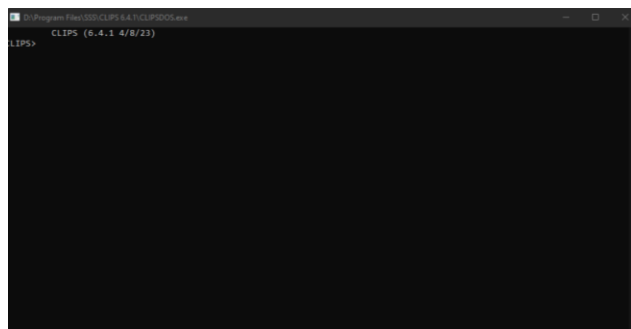
- از سر نام های عبارات C Language Implementation Production System گرفته شده است
- برای اولین بار توسط NASA در واحد فناوری نرم افزار (Software Technology Branch) تولید شده است.
- این نرم افزار رایگان بوده و در حال حاضر نسخه ۶.۳ آن موجود می باشد.
- این نرم افزار بر روی سیستم عامل های متفاوت مانند ویندوز 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 گرای (Object-oriented)
 - بر مبنای روال ها (Procedural programming)
- نرم افزار کلیپس فقط قواعد زنجیره ای پیشرو را پشتیبانی می کند.

دستیابی به نرم افزار

این نرم افزار یک برنامه رایگان می باشد که هم اکنون نسخه ۶.۳ 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.



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 Symbols، 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 $2.73E3$ ، 0.215 ، $-25.E7$ 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 واقعیات 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)

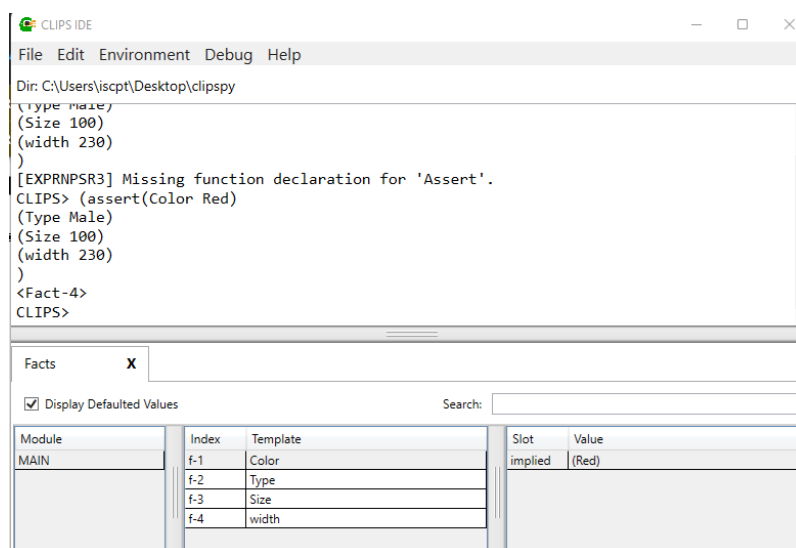
(Type Male)

(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



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>) |
 (allowed-lexemes <lexeme-list>) |
 (allowedintegers <integer-list>) |
 (allowedfloats <float-list>) |
 (allowednumbers <number-list>) |
 (allowed-instance-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 **_aseFact1** 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 **(Terminal)**

tمانیتور 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 الگوهای 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 ننماییم 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 فایل 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 قبلا 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.

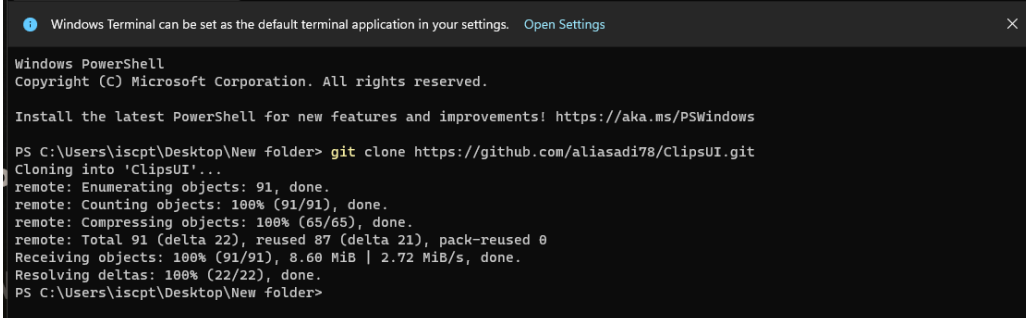
راه اندازی برنامه

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>
```

با دستور `cd CLIPSUI` وارد پوشه می شویم

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>

```

و با دستور زیر برنامه رو اجرا می کنیم.

npm start

برنامه در آدرس localhost:3000 اجرا می شود.



نحوه کار برنامه

برنامه از ۳ قسمت اصلی تشکیل شده است:

قسمت Fact

قسمت Template

قسمت Rule

The screenshot displays the application's user interface, which is organized into three primary functional areas, each with its own form and submission button:

- Fact Section:** Located at the top, it contains a single text input field labeled 'fact' and a green button labeled 'SUBMIT FACT'.
- Template Section:** Positioned on the bottom left, it includes three input fields: 'templateName', 'templateDescription', and 'templateDetailNumber'. Below these fields is a green button labeled 'SUBMIT TEMPLATE'. An 'ADD TEMPLATE' button is located below the entire section.
- Rule Section:** Positioned on 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. A green button labeled 'SUBMIT RULE' is at the bottom of the form. An 'ADD RULE' button is located below the entire section.

Between the Fact and Template sections is an 'ADD FACTS' button. At the bottom center of the interface is a 'GENERATE CODE' button.

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.

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.

Template

templateName

person

templateDescription

templateDetailNumber

3

typeOfValue-0

slot

detailName-0

name

detailType-0

STRING

typeOfValue-1

slot

detailName-1

surname

detailType-1

STRING

typeOfValue-2

slot

detailName-2

birthdate

detailType-2

SYMBOL

SUBMIT TEMPLATE

Template

templateName

templateDescription

templateDetailNumber

SUBMIT TEMPLATE

ADD TEMPLATE

Rule

templateName

templateDescription

ruleDetailNumber

operation

operationName

question

SUBMIT RULE

ADD RULE

GENERATE CODE

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.

Template

templateName

person

templateDescription

templateDetailNumber

3

typeOfValue-0

slot

detailName-0

name

detailType-0

STRING

typeOfValue-1

slot

detailName-1

surname

detailType-1

STRING

typeOfValue-2

slot

detailName-2

birthdate

detailType-2

SYMBOL

SUBMIT TEMPLATE

Template

templateName

templateDescription

templateDetailNumber

SUBMIT TEMPLATE

ADD TEMPLATE

Rule

templateName

my-rule

templateDescription

ruleDetailNumber

1

pattern-0

my-fact first-slot

operation

t "My Rule fired!" crlf

operationName

print

question

next rule?

SUBMIT RULE

ADD RULE

GENERATE CODE

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

تشریح کد

```
<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>
```

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>
)
```

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>
)
```

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>
)
```

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()`
```

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.

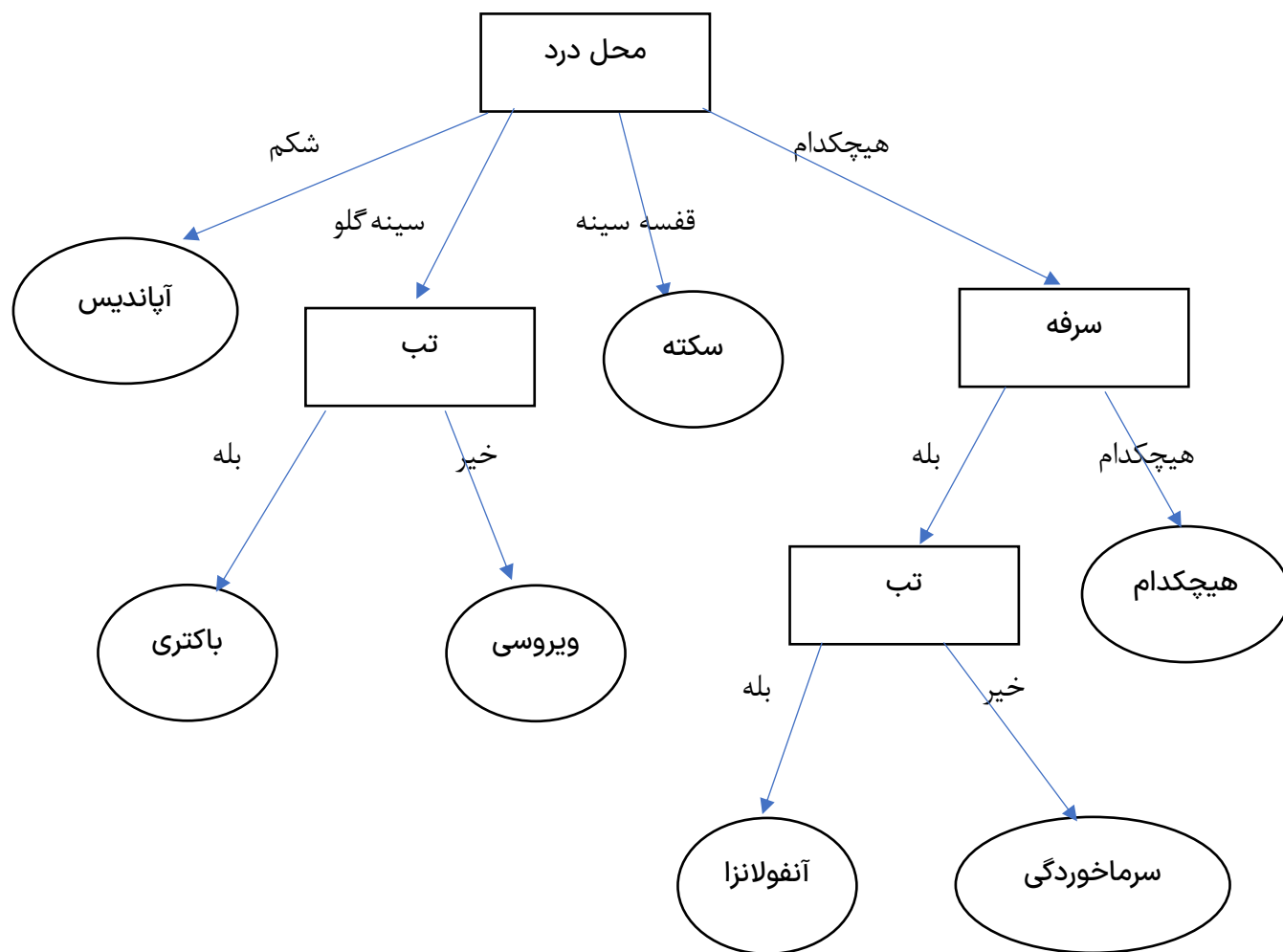
یا اگر محل درد “سایر” انتخاب شود در مرحله بعدی از فرد پرسیده شود که آیا “سرفه” می کند؟ از در جواب این سوال بگوید “خیر”، این سیستم ساده نتوانسته بیماری فرد را تشخیص دهد ولی اگر “بله” انتخاب شود در مرحله بعدی از فرد پرسیده شود که آیا “تب” دارد؟ اگر در جواب این سوال بگوید “بله”، تشخیص داده شود که بیماری “آنفلوآنزا” 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 آورده اید یک درخت تصمیم گیری ایجاد کنید. البته توجه کنید این درخت سلسله مراتبی است و از ریشه به سمت برگ‌ها به جلو می‌رود. بنابراین در بعضی سیستم‌ها ترتیب این مراحل می‌تواند در تصمیم‌گیری بسیار مهم باشد.



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

Rule

templateName

Menu

templateDescription

ruleDetailNumber

0

operation

t crlf crlf crlf " mahal dard ki

operationName

print

question

selectedindex

SUBMIT RULE

Rule

templateName

A-shekam

templateDescription

ruleDetailNumber

1

pattern-0

selectedindex 1

operation

t crlf crlf crlf" shoma bimary

operationName

print

question

SUBMIT RULE

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-sorfeh-no	templateDescription	ruleDetailNumber 2
pattern-0 selectedIndex 4		
pattern-1 ifYesNoSolfeh no		
operation t " moteasefane nmitavanam	operationName print	question

SUBMIT RULE

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

```

(defrule Menu ""
  ">"
  (printout t "crif crif " " mahl dard kujest?Lutfeh yk az gozinshaye zir ra entetab kard" "crif crif " 1.) shekam. " crif crif " 2.) galo." "crif crif " 3.) sikh." "crif crif " 4.) sayer." "crif crif " 5.) EXIT OP SYSTEM.." "crif crif " " Your answer: ")
  (assert ( selectIndex (read)))
  (defrule A-shekam ""
    (selectIndex 1)
    "")
  (printout t "crif crif crif" shoma kinary Apandis darid " crif crif " Thank you for using my Program... "crif crif ")
  ""
  (defrule A-shekam ""
    (selectIndex 1)
    (printout t "crif crif crif" shoma kinary Apandis darid " crif crif " Thank you for using my Program... "crif crif ")
  )
  (defrule Q-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 2)
      (ifreshTab yes)
      ">"
      (printout t "crif crif crif" shoma glindard bakhtesai darid " crif crif)
    )
    (defrule A-galo-tab-no ""
      (selectIndex 2)
      (ifreshTab no)
      ">"
      (printout t "crif crif crif" shoma glindard virusi darid " crif crif)
    )
    (defrule A-sikh ""
      (selectIndex 3)
      ">"
      (printout t "crif crif crif" shoma saikh kardid " crif crif " Thank you for using my Program... "crif crif)
    )
    (defrule Q-sayer-soifeh ""
      (selectIndex 4)
      ">"
      (printout t "crif crif crif " aya shoma soife mikand? (Yes | No) " crif crif " Your answer: ")
      (assert ( ifreshSoife (read)))
      (defrule A-sayer-soifeh-yes-tab ""
        (selectIndex 4)
        (ifreshSoife yes)
        ">"
        (printout t "crif crif crif " aya shoma tab darid? (Yes | No) " crif crif " Your answer: ")
        (assert ( ifreshTab (read)))
        (defrule A-sayer-soifeh-yes-tab-yes ""
          (selectIndex 4)
          (ifreshSoife yes)
          (ifreshTab yes)
          ">"
          (printout t "crif crif crif" shoma anfaridza darid " crif crif " Thank you for using my Program... "crif crif)
        )
        (defrule A-sayer-soifeh-yes-tab-no ""
          (selectIndex 4)
          ""
        )
      )
    )
  )
  (import clips
    environment <- clips.environment()
    DEFALTE_STRING "" (defrule Menu ""
      ">"
      (printout t "crif crif crif " mahl dard kujest?Lutfeh yk az gozinshaye zir ra entetab kard" "crif crif " 1.) shekam. " crif crif " 2.) galo." "crif crif " 3.) sikh." "crif crif " 4.) sayer." "crif crif " 5.) EXIT OP SYSTEM.." "crif crif " " Your answer: ")
      (assert ( selectIndex (read)))
      environment build[DEFALTE_STRING]
      DEFALTE_STRING "" (defrule A-shekam ""
        (selectIndex 1)
        "")
      (printout t "crif crif crif" shoma kinary Apandis darid " crif crif " Thank you for using my Program... "crif crif ")
      ""
      environment build[DEFALTE_STRING]
      DEFALTE_STRING "" (defrule Q-galo-tab ""
        (selectIndex 2)
        ">"
        (printout t "crif crif crif " aya shoma tab darid? (Yes | No) " crif crif " Your answer: ")
        (assert ( ifreshTab (read)))
        environment build[DEFALTE_STRING]
        DEFALTE_STRING "" (defrule A-galo-tab-yes ""
          (selectIndex 2)
          (ifreshTab yes)
          ">"
          (printout t "crif crif crif" shoma glindard bakhtesai darid " crif crif)
        )
        environment build[DEFALTE_STRING]
        DEFALTE_STRING "" (defrule A-galo-tab-no ""
          (selectIndex 2)
          (ifreshTab no)
          ">"
          (printout t "crif crif crif" shoma glindard virusi darid " crif crif)
        )
        environment build[DEFALTE_STRING]
        DEFALTE_STRING "" (defrule A-sikh ""
          (selectIndex 3)
          ">"
          (printout t "crif crif crif" shoma saikh kardid " crif crif " Thank you for using my Program... "crif crif)
        )
        environment build[DEFALTE_STRING]
        DEFALTE_STRING "" (defrule Q-sayer-soifeh ""
          (selectIndex 4)
          ">"
          (printout t "crif crif crif " aya shoma soife mikand? (Yes | No) " crif crif " Your answer: ")
          (assert ( ifreshSoife (read)))
          environment build[DEFALTE_STRING]
          DEFALTE_STRING "" (defrule A-sayer-soifeh-yes-tab ""
            (selectIndex 4)
            (ifreshSoife yes)
            ">"
            (printout t "crif crif crif " aya shoma tab darid? (Yes | No) " crif crif " Your answer: ")
            (assert ( ifreshTab (read)))
            environment build[DEFALTE_STRING]
            DEFALTE_STRING "" (defrule A-sayer-soifeh-yes-tab-yes ""
              (selectIndex 4)
              (ifreshSoife yes)
              (ifreshTab yes)
              ">"
              (ifreshSoife yes)
            )
          )
        )
      )
    )
  )

```

```

import clips
environment = clips.Environment()
environment.build(DEFRULE_STRING0)
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 " %)" EXIT @? SYSTEM." crlf crlf crlf " Your answer: ")
(assert ( selectedindex (read))))""
environment.build(DEFRULE_STRING1)
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 bakterai 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 karded " 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 binari shoma ra tashkhis daham" crlf crlf " Thank you for using my Program..."crlf crlf)
)""
environment.build(DEFRULE_STRING11)
environment.run()

```

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

```
1.) shekam.
```

```
2.) galo.
```

```
3.) sineh.
```

```
4.) sayar.
```

```
5.) EXIT OF SYSTEM..
```

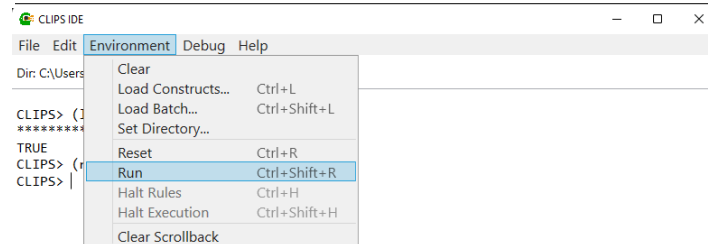
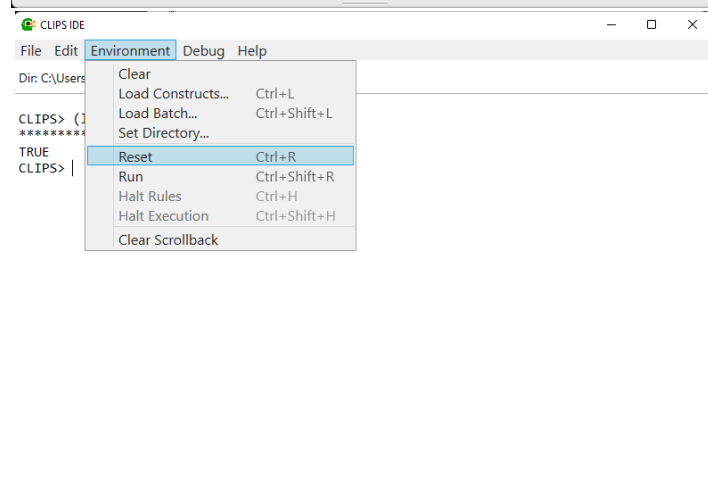
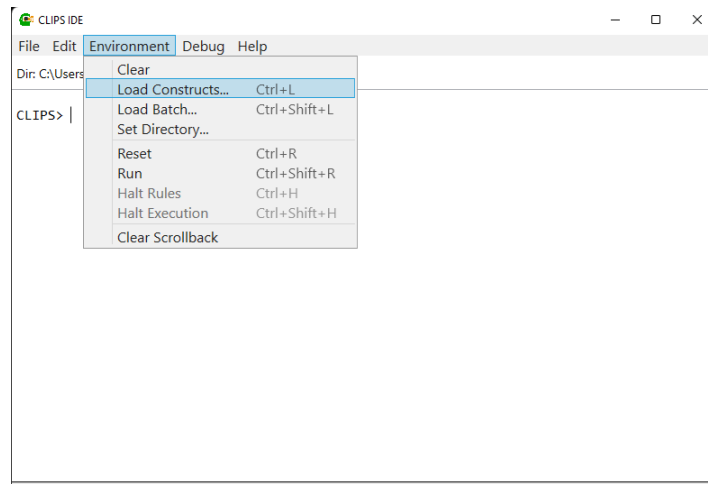
```
Your answer: 1
```

```
█
```

```
shoma bimary Apandis darid
```

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

برای اجرای برنامه در کلیپ مانند شکل عمل می کنیم



```
CLIPS IDE
File Edit Environment Debug Help
Dir: C:\Users\iscpt\Desktop\clipsy

CLIPS (6.4.1 4/8/23)
CLIPS> (load "final.clp")
*****
TRUE
CLIPS> (reset)
CLIPS> (run)

mahal dard kojast?lotfan yki az gozinehay zir ra entekhab konid

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

Your answer:
```

```
CLIPS IDE
File Edit Environment Debug Help
Dir: C:\Users\iscpt\Desktop\clipsy

mahal dard kojast?lotfan yki az gozinehay zir ra entekhab konid

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

Your answer: 3

shoma sekteh kardeid
Thank you for using my Program...

CLIPS>
```

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، 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.

<https://www.clipsrules.net>

<https://clipspy.readthedocs.io/en/latest/>

<https://www.mygreatlearning.com/blog/expert-systems-in-artificial-intelligence/>

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm

<https://www.javatpoint.com/expert-systems-in-artificial-intelligence>

<https://www.guru99.com/expert-systems-with-applications.html#11>

<https://digitalthinkerhelp.com/expert-system-in-artificial-intelligence-with-applications-examples-types-uses/>

<http://Parsbook.org>

<http://tehranit.net/آموزش-سریع-کلیپس-سیستم-خبره-۴-ساعت/>

<https://blog.faradars.org/سیستم-خبره-چیست/>



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