استفاده از زبان منطقی پرولاگ در کاربردهای زمان-واقعی شبکه‌های قدرت

حسین جلالی کوشکی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

چکیده :

هوش مصنوعی کاربردهای فراوانی در شبکه‌های قدرت دارد که از جمله می‌توان از موارد استفاده سیستمهای خبره نام برد که علاوه بر کاربردهائی در زمینه‌های طراحی، برنامه‌ریزی، و نگهداری و تعمیرات سیستمهای قدرت، می‌توانند در پائیدن (Monitoring) و کنترل شبکه‌ها نیز مفید واقع گردند. کاربردهائی از قبیل پائیدن، کنترل و عیب‌یابی روی خط (on-line) بصورت زمان-واقعی (Real-time) بوده و پیاده‌کردن آنها مستلزم دستیابی به آخرین اطلاعات وضعیت شبکه می‌باشد. این اطلاعات معمولا با استفاده از روشهای الگوریتمی محاسبه می‌شوند و عموما بر روی کامپیوتری جدا از کامپیوتر مورد استفاده سیستم خبره اجرا می‌گردند. مسئله ارتباط بین دو سیستم کامپیوتری مزبور، و نحوه پیاده‌سازی سیستم‌های خبره زمان-واقعی با استفاده از زبان منطقی پرولاگ مورد بحث این مقاله می‌باشد. روش ارائه شده در مقاله جنبه نسبتا عام داشته و می‌تواند در کلیه مواردیکه سیستم خبره بر روی کامپیوترهای شخصی قابل پیاده‌شدن است، بکار گرفته شود.

۱ - مقدمه:

یکی از اهداف تحقیقات هوش مصنوعی، بوجود آوردن برنامه‌های کامپیوتری است که عملکردی مشابه انسان داشته باشند بنحوی که یک کاربر نتواند تشخیص دهد که آیا طرف خطابش یک برنامه کامپیوتری است یا یک انسان. دامنه مسائلی که در حوزه هوش مصنوعی قرار می‌گیرند بسیار وسیع است و شامل مواردی از قبیل بازیها (Games)، دیدن و شنیدن (Vision and Hearing)، درک زبان طبیعی (Natural Language Understanding)، برنامه‌ریزی روباتها (Robotics)، و سیستمهای خبره (Expert Systems) می‌شود[1].

در قسمت اعظم زمینه‌های هوش مصنوعی، هنوزبرنامه‌های کامپیوتری نسبت به انسان (و حتی نسبت به حیوانات) از قابلیتهای بسیار پائینتری برخوردارند. بعنوان مثال، برنامه‌های

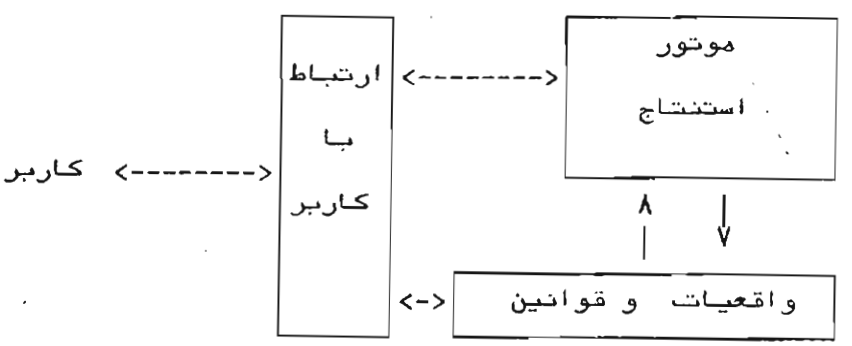
کامپیوتری که برای دیدن اشیا یاشنیدن صداها ویاتشخیص صحبت (Speech Recognition) تهیه شده‌اند هنوز در مراحل نسبتا مقدماتی توسعه خود می‌باشند. بطور کلی شاید بتوان گفت که این مطلب در کلیه مواردیکه مسئله درک عمومی (Common Sense) و قابلیتهای عمومی افراد بشر مطرح است، صادق می‌باشد. برعکس در زمینه‌هائی که استدلال و استنتاج در یک حوزه خاص و محدود مورد نظر است (نظیر سیستمهای خبره)، برنامه‌های کامپیوتری بسیار موفقی تهیه شده‌است که نه تنها به حد عملکرد انسان می‌رسند بلکه در مواردی قادرند از این حد نیز فراتر بروند.

در این موارد، برنامه‌های کامپیوتری با استفاده از مجموعه‌ای از واقعیتها، قوانین تجربی، و اطلاعات دیگر مربوط به یک مسئله خاص، و با بکارگیری ترتیب مناسب اعمال این قوانین، استدلال و استنتاج می‌نمایند. نکته اساسی در این است که در اینگونه سیستمها، برخلاف تلاشهای اولیه در زمینه هوش مصنوعی، برنامه‌های عام و کلی مورد نظر نبوده بلکه از دانش خاص مربوط به مسئله استفاده شده و دامنه جستجو محدود می‌گردد. بعضی از برنامه‌های کامپیوتری که در زمینه‌های خاص نظیر تشخیص پزشکی، تشخیص ساختار مولکولی اجسام، معدن‌یابی، برنامه ریزی ساخت کامپیوتر، و طراحی VLSI نوشته شده و مورد استفاده قرارگرفته‌اند عملکردی موفقیت‌آمیزتر از افراد خبره در رشته‌های مزبور داشته‌اند و این امر سبب پیشرفت نسبتا سریع سیستمهای خبره گردیده‌است بطوریکه امروزه کاربردهای مختلف این تکنیک را در رشته‌های مختلف شاهد هستیم که از جمله می‌توان از کاربردهای سیستمهای خبره در شبکه‌های قدرت [2] و همچنین کاربردهای زمان-واقعی اینگونه سیستمها در کنترل پروسه‌ها نام برد[5-3]. برنامه‌های اعلام خطر و کمک به اپراتورهای اطاق کنترل در تشخیص عیب و تعیین محل خطا، مواردی از این قبیلند.

سیستمهای خبره زمان-واقعی، علاوه بر دانش حل مسئله، که عموما بصورت یک سری قوانین بیان می‌شوند، نیاز به آخرین اطلاعات وضعیت سیستم کنترل شونده دارند. سیستم کامپیوتری که عهده‌دار وظیفه حساس جمع‌آوری و یا محاسبه الگوریتمی مقادیر مزبور است چنانچه درگیر محاسبات وقتگیر (و عموما کند) مربوط به سیستم خبره شود، ممکن است از وظیفه اصلی خود باز بماند. بنابراین سیستمهای خبره زمان-واقعی معمولا برروی کامپیوتری جدا از کامپیوتر اصلی سیستم کنترل پیاده می‌شوند و در اینصورت مسئله ارتباط بین دو سیستم کامپیوتری بایستی به نحوی حل شود که از یکطرف تاثیر نامطلوب بر محاسبات و عملیات سیستم اصلی نگذارد و ازطرف دیگر بتواند نیازهای سیستم خبره را به نحو مطلوب برآورده سازد. در بخشهای بعدی این مقاله، پس از مقایسه خصوصیات سیستمهای خبره و زبان منطقی پرولاگ، روشی بیان خواهد شد که در اکثر مواردی که دامنه سیستم خبره محدود بوده و برروی کامپیوترهای شخصی قابل پیاده‌شدن باشد، می‌تواند مورد استفاده واقع شود.

۲ - خصوصیات سیستمهای تولید (Production Systems):

در سیستمهای "مبتنی بر قوانین" (Rule-Based Systems) که به سیستمهای تولید نیز مشهورند، دانش حل مسئله بصورت یک سری قوانین (rules) بیان می‌شود و این متداول‌ترین روش برای بیان دانش حل مسئله در سیستمهای خبره است [6]. در این سیستمها وضعیت فعلی سیستم بصورت مجموعه‌ای از واقعیات (facts)، و وضعیت مورد نظر بصورت مجموعه‌ای از اهداف (goals) بیان می‌شوند. وظیفه پیداکردن مسیری از وضعیت فعلی به وضعیت هدف، به عهده موتور استنتاج (Inference Engine) است که درعین‌حال می‌تواند نحوه رسیدن به جواب را توضیح دهد (ازطریق بیان سلسله قوانین اعمال شده در مسیر پیموده‌شده). قسمت ارتباط با کاربر (User Interface) وظیفه تبادل اطلاعات با کاربر را بر عهده دارد. اجزا مختلف یک سیستم تولید و ارتباط بین آنها، در شکل (۱) نشان داده‌شده‌است.



شکل (۱)- اجزا یک سیستم خبره و ارتباط بین آنها

به سیستمی که کلیه اجزا را به استثنای پایگاه معرفت (مجموعه قوانین و واقعیات) داشته باشد بنحوی که بتوان آن را برای پیاده‌کردن سیستمهای خبره مختلف بکار گرفت، "پوسته" (Shell) می‌گویند.

در سیستمهای تولید، هر قانون بصورت IF...THEN... می‌باشد که پس از کلمه IF قسمت شرایط قانون (conditions) وپس از کلمه THEN قسمت عملکرد قانون (actions) نوشته می‌شود که هر قسمت ممکن است شامل چند عبارت باشد که با AND و OR به هم‌دیگر مربوطند. هر قانون درصورتی قابل اجرا است که کلیه شرایط مشخص شده در آن برقرار باشد. درهرلحظه با توجه به وضعیت کنونی مسئله (که توسط واقعیات مشخص می‌شود) تعدادی از

قوانین قابل اجرا خواهندبود که مجموعه قوانین مزبور "مجموعه تضاد" (Conflict Set) رابوجودمی‌آورندوسپس بااستفاده ازاستراتژی معینی جهت رفع تضاد (Conflict Resolution) یکی از قوانین مزبور انتخاب و اجرا خواهد شد. اجرای یک قانون دراکثرموارد سبب خواهد شد که "پایگاه معرفت" (Knowledge Base) مسئله تغییر نماید بنحوی‌که در سیکل بعدی مجموعه دیگری ازقوانین قابل اجرا خواهند شد و قانون جدیدی برای اجرا انتخاب خواهد گردید. این عمل آنقدر تکرار خواهد شد تا اینکه یا به هدف برسیم ویااینکه هیچ قانون دیگری قابل اجرا نباشد (یعنی مجموعه تضاد خالی باشد) که در آنصورت برنامه متوقف می‌گردد. برای شرح کامل و دقیق نحوه عملکرد سیستمهای تولید می‌توانید به یکی از مراجع متعدد دراین زمینه (از جمله [6]) مراجعه فرمائید.

در این سیستمها ترتیب قرارگرفتن قوانین حائز اهمیت نیست و بعلاوه سیستم می‌تواند از کاربر در موارد لزوم سئوال نماید و یا به سئوالهای کاربر، ازقبیل "چرا" (WHY) و "چطور" (HOW) پاسخ گوید. ضمنا طرزکار این سیستمها عموما بصورت محاوره‌ای است (گرچه برخی از سیستمها قابلیت کامپایل شدن را نیز دارا می‌باشند). بعلاوه، برخی از پوسته‌ها این امکان را در اختیار استفاده‌کننده قرار می‌دهند که قسمتهائی از برنامه را بصورت الگوریتمی (با استفاده از یکی از زبانهای برنامه‌نویسی متداول) نوشته و آن قسمت را ملحق به برنامه خود نماید.

۳ - زبان منطقی پرولاگ[7]:

زبان پرولاگ، برخلاف اکثر زبانهای برنامه‌نویسی موجود، یک زبان "روشی" (Procedural) نیست بلکه یک زبان "اخباری" (Declarative) است که در آن واقعیات و قوانین مربوط به مسئله بصورت "گزاره‌های منطقی" (Logical Predicates) بیان می‌شوند. همچنین هدف بصورت مجموعه‌ای از گزاره‌ها (که باید اثبات گردند) نوشته می‌شود. هر قانون دارای دو قسمت است که در قسمت "مقدم" (antecedent) شرایط قانون، و در قسمت "تالی" (consequent) نتیجه قانون نوشته می‌شود. در قسمت مقدم می‌توان چندین شرط را بیان نمود که بوسیله اپراتورهای منطقی AND و OR به یکدیگر مربوطند. موتور استنتاج پرولاگ، با استفاده از قوانین مسئله سعی خواهد نمود که گزاره (یا گزاره‌های) هدف را به اثبات برساند. تشابه زیادی بین سیستمهای مبتنی بر قوانین، که شرح مختصرشان در بخش قبل گذشت، و برنامه‌های پرولاگ وجود دارد. در هر دو مورد دانش مربوط به مسئله بصورت مجموعه‌ای از واقعیات و قوانین بیان می‌گردد و در هر دو مورد موتور استنتاج با استفاده از قوانین مسئله، مسیری از وضعیت فعلی به وضعیت هدف خواهد یافت. از طرف دیگر تفاوتهائی نیز بین این دو وجود دارد که اهم آنها عبارتند از:

آ) در برنامه‌های پرولاگ، هر عبارت فقط می‌تواند یک نتیجه داشته باشد و نوشتن نتیجه‌هائی که بصورت AND و OR چند گزاره هستند در قسمت تالی مجاز نیست. لازم به تذکر است که این مطلب تاثیر چندانی بر قابلیت سیستم نخواهد گذاشت چون می‌توان قوانینی

که بیش از یک گزاره در قسمت نتیجه‌شان موجود است را بصورت چند قانون مجزا در پرولاگ نوشت بنحوی که عینا همان نتیجه را داشته باشد.

ب) در برنامه‌های پرولاگ، ترتیب قرارگرفتن قوانین اهمیت دارد و در زمان اجرا، قوانین به همان ترتیبی که نوشته شده‌اند، مورد بررسی قرار خواهند گرفت.

ج) از موتور استنتاج پرولاگ نمی‌توان سئوالات "چرا" و "چگونه" نمود مگر اینکه این امکان توسط برنامه‌نویس فراهم آمده باشد.

د) در سیستمهای تولید عموما امکان تعریف و بکارگیری ضریب اطمینان برای استنتاج غیردقیق وجود دارد درحالیکه در پرولاگ چنین امکانی فراهم نیست (ولی می‌تواند توسط برنامه‌نویس فراهم گردد).

از طرف دیگر مزایای استفاده از زبان پرولاگ بدینقرارند:

آ) پرولاگ بر مبنای منطق گزاره‌ها استوار است و بدینترتیب از مبنای تئوریک قوی ریاضی برخوردار می‌باشد.

ب) پرولاگ یک زبان برنامه‌نویسی است و نسبت به پوسته‌های موجود از انعطاف بیشتری برخوردار است.

ج) سرعت اجرا در برنامه‌های پرولاگ بیشتر است، بخصوص در انواع کامپایل شده آن.

د) امکان الحاق برنامه‌هائی که به زبان Turbo C نوشته شده‌اند (در مورد توربو پرولاگ) وجود دارد [8].

استفاده از پرولاگ جهت طراحی و پیاده‌سازی سیستمهای خبره، مطلب جدیدی نیست و کتب و مقالات مختلفی در این زمینه نوشته شده‌است.

٤ - سیستمهای خبره زمان-واقعی:

کاربردهای زمان-واقعی سیستمهای خبره فراوانند. از آنجمله می‌توان از پائیدن شبکه (Monitoring)، جلوگیری از بروز وضعیت خطر (Alarm Pervention)، بررسی علت اعلام خطر (Alarm Analysis)، بهینه‌سازی فرایندها (Process Optimization)، عیب‌یابی سیستم و دستگاهها (System and Equipment Diagnosis) و نظایر آن نام برد. بعنوان مثال در یک سیستم بررسی علت اعلام خطر، سیستم خبره به اپراتور کمک می‌کند تا از میان تعداد فراوانی علامت خطر، علت اصلی را دریابد و در رفع آن بکوشد. چنین سیستمی می‌تواند

با استفاده از تجربیات اپراتورهای با سابقه و خبره طراحی و ایجاد شود و در مواقعی که اپراتورهای کم تجربه‌تر مسئولیت نگهداری سیستم را بعهده دارند، بعنوان کمک به آنان مورد استفاده قرار گیرد.

در سیستمهائی که زمان-واقعی نیستند، عموما واقعیتهای مسئله در ابتدا مشخص شده و تا پایان حل مسئله به قوت خود باقی خواهند بود. بعلاوه، در اینگونه سیستمها، می‌توان پاره‌ای از واقعیتها را در حین یافتن جواب از کاربر سئوال نمود. بعنوان مثال در سیستم خبره معروف MYCIN [9]، واقعیتهای مسئله (مشخصات بیمار و نتیجه آزمایشات مختلف) از کاربر سئوال می‌شود، و در سیستمهای طراحی و برنامه‌ریزی، محدودیتهای مسئله که نمایانگر قسمت اعظم واقعیتها می‌باشند، از ابتداء مشخص شده و تا پایان ثابت خواهند ماند.

در سیستمهای زمان-واقعی، ازیکطرف واقعیتهای مسئله (مثلا مقادیر ولتاژها و جریانهای مختلف) دائما در حال تغییرند و ازطرف دیگر عموما امکان سئوال از کاربر وجود ندارد. بعنوان مثال در یک سیستم بررسی علت اعلام خطر، سیستم خبره بایستی وضعیت شبکه را دائما تحت نظر داشته باشد تا درصورت بروز خطا، بتواند علت را بررسی نموده و به اپراتور کمک نماید. در چنین سیستمی باید:

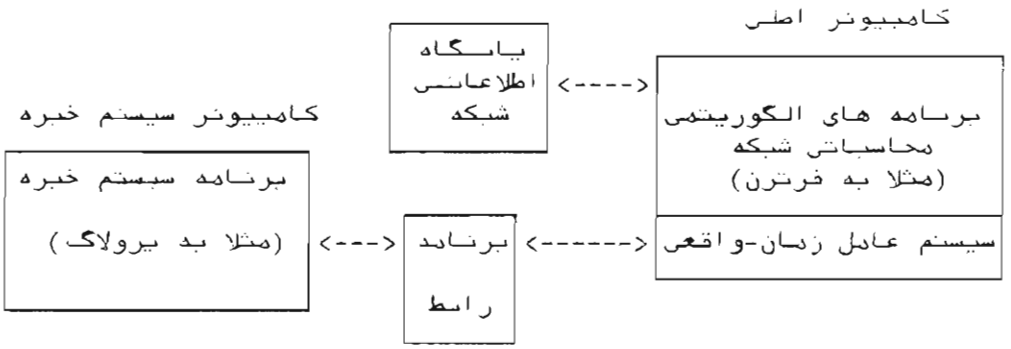
آ) وضعیت شبکه، که واقعیتهای مسئله را تشکیل می‌دهند، بطور اتوماتیک در اختیار سیستم خبره قرارگیرند. این مطلب از طریق ارتباط سیستم خبره با کامپیوتری که وضعیت شبکه را ثبت می‌کند، امکانپذیر است.

ب) سیستم بایستی بتواند تجزیه و تحلیل را در زمان مناسب (عموما خیلی سریعتر از آنچه در سیستمهای غیر زمان-واقعی قابل قبول است) به انجام رساند.

ج) محاوره با اپراتور و سئوال کردن از وی امکانپذیر نیست و اپراتور نیز به احتمال زیاد، از سیستم سئوالات "چرا" و "چگونه" نخواهد کرد.

سیستمهای خبره زمان-واقعی عموما بر روی کامپیوتری متفاوت با کامپیوتر اصلی شبکه پیاده می‌شوند تا اجرای آنها موجب تاثیر نامطلوب بر عملکرد کامپیوتر اصلی شبکه نگردد. شکل(۲) اجزاء چنین سیستمی را نشان می‌دهد.

مشخصه‌های سیستمهای زمان-واقعی به نحوی است که محدودیتهای استفاده از پرولاگ (عدم امکان پرسش های "چرا" و "چگونه"، اهمیت ترتیب قوانین) تاثیر چندانی بر این سیستمها نخواهد داشت و برعکس مزیتهای استفاده از پرولاگ (سرعت بیشتر، امکان الحاق برنامه‌های به زبان دیگر) درچنین سیستمهائی اهمیت فراوان دارد.



شکل (۲) - پیاده کردن یک سیستم خبره زمان-واقعی

۵ - روش پیشنهادی:

در مواردیکه دانش سیستم خبره زیاد گسترده نباشد بنحوی که بتوان آن را با استفاده از کامپیوترهای شخصی پیاده کرد می‌توان از روش زیر استفاده نمود. لازم به تذکر است که با توجه به قدرت و سرعت روزافزون کامپیوترهای شخصی، این محدودیت از عملی بودن سیستمها جلوگیری نخواهد نمود. ضمنا روش پیشنهادی جنبه عام داشته و حتی در مواردیکه امکان اتصال برنامه‌های دیگر به سیستم اصلی به راحتی میسر نباشد، قابل پیاده‌شدن است و کمترین تاثیر نامطلوب را بر عملکرد کامپیوتر اصلی خواهد گذاشت.

نحوه عمل بدینترتیب است که ابتدا سیستم خبره با استفاده از زبان منطقی پرولاگ نوشته شده و برروی یک کامپیوتر شخصی، که در عین حال از طریق "برابرسازی" (emulation) بعنوان یک ترمینال سیستم کامپیوتری اصلی عمل می‌کند، اجرا می‌شود. مثلا اگر کامپیوتر اصلی از سری IBM/370 باشد، می‌توان با استفاده از کارت IRMA و برنامه برابرساز مربوطه، کامپیوتر شخصی را در عین حال بعنوان یک ترمینال IBM 3278 مورد استفاده قرار داد. سپس این ترمینال، بعنوان یکی از ترمینالهای سیستم اصلی، طوری برنامه‌ریزی می‌گردد که اطلاعات مورد نیاز سیستم خبره را برروی صفحه تصویر نمایش دهد. کلیه این کار بسته به امکانات سیستم اصلی مورد استفاده، به طرق مختلف امکانپذیر است و درجه سهولت آن نیز در سیستمهای مختلف یکسان نخواهد بود. مرحله بعدی آن است که برنامه‌ای به یک زبان روشی (مثلا C) بنویسیم که با استفاده از دستورالعمل‌های ارتباطی که جهت دسترسی از PC به صفحه تصویر مربوط به سیستم اصلی (host) توسط برنامه برابرساز فراهم آمده‌است، اطلاعات مربوط به شبکه را مستقیما از حافظه صفحه تصویر دریافت نموده و با استفاده از آخرین اطلاعات فراهت شده، واقعیات مربوط به آن وضعیت را بصورت گزاره‌های پرولاگ در قالبی قراردهد. اطلاعات این فایل که بصورت تکراری با فرکانس مطلوب (مثلا یکبار در ثانیه) محتویاتش از نو ساخته می‌شود، می‌تواند ازطرف برنامه پرولاگ مورد "مشاوره" (consult) واقع گردد که در نتیجه تازه‌ترین اطلاعات شبکه به برنامه پرولاگ انتقال می‌یابد بدون آنکه تاثیر نامطلوبی بر سیستم اصلی ایجاد شود.

۶ - بررسی نتایج:

در نمونه‌ای که به منظور آزمایش این نحوه عمل تهیه شد، سیستم اصلی IBM 4381، کامپیوتر مربوط به سیستم خبره IBM PS/2 Mod 70-121، کارت برابر سازی IBM 3270 Connection، و برنامه مربوطه IBM Graphics Workstation Program بودند. زبان مورد استفاده جهت برنامه سیستم خبره Turbo Prolog V2.0 و زبان مورد استفاده جهت برنامه ارتباطی Turbo C بود. تعداد قانونهای مورد استفاده ۲۰ عدد و تعداد پارامترهائی که از طرف سیستم خبره پائیده می‌شدند ۱۷ عدد بود. در چنین شرایطی عملکرد سیستم بسیار مناسب بود و امکان تغییر واقعیتها تا سه بار در ثانیه بدون هیچگونه تاثیر نامطلوب بر شبکه اصلی تایید گردید.

۷ - مراجع:

[1] E. Rich, Artificial Intelligence, McGraw Hill Book Company, 1983.

[2] B.F. Wollenberg and T. Sakaguchi, "Artificial Intelligence in Power System Operations", Proceedings of the IEEE, Vol. 75, No. 12, Dec. 1987, pp.1678-1635.

[3] R.L. Moore et al, "Expert Systems: Are They the Next Generation of Process Control?", InTech, May 1985, pp.55-57.

[4] J.A. Bernard, "Use of a Rule-Based System for Process Control", IEEE Control System Magazine, Oct. 1988, pp.3-13.

[5] R.S. Shirley, "Some Lessons Learned Using Expert Systems for Process Control", IEEE Control System Magazine, Dec. 1987.

[6] L. Brownstone et al, Programming Expert Systems in OPS5, Addison Weseley Publishing Company, 1986.

[7] W.F. Clocksin and C.S. Mellish, Programming in Prolog, Springer-Verlag, 1987.

[8] Turbo Prolog 2.0 Manuals, Borland Corporation, 1988.

[9] B.G. Buchanan and R.O. Duda, "Principles of Rule-Based Expert Systems", Stanford University, Heuristic Programming Project Report No. HPP-82-14, Aug. 1982.