معرفی سیستمهای خبره و سامانه های مبتنی بر فریم

مولف: بهرنگ قاسمی زاده
دانشکده کامپیوتر
دانشگاه علم و صنعت ایران
سال تحصیلی 1385

An Introduction to Expert Systems

Behrang QasemiZadeh
Computer Engineering Department, Iran University of Science and Technology
Academic Year 2005

فهـــرست مطـــالب

عنـــوان	صفحه
مقدمه	1
1. ارائه دانش و اهمیت آن	3
2. ساختـــار فریم ها در نخستین نظر	3
3. روشهای استنتاج مبتنی بر فریم	6
4. سیستم ارائه دانش بر پایه فریم از منظر عملیاتی	7
1.4 ذخیره و کسب دانش	7
2.4 حل مسائل و استنتاج	7
5. مسائل اصلی طراحی در سیستمهای ارائه دانش	9
6. آشنایی با بعضی خانواده های FKRS	10
7. انواع فریمها در سیستمهای FKRS	11
8. اصطلاحات فنی در اتصال(Link) فریمها	11
9. تنوع فریم ها	12
10. اسلات ها	14
1.10. SlotUnit	15
2.10. نوع داد ه های اسلات	15
11. تـــوارث	16
1.11. ناسازگاری در توارث	17
2.11. زمان محاسبه توارث	18
12. برنامه نویسی شی گرا و Access Oriented در FKRS	20
13. کلاس بندی	21

مقـــدمه

استفاده از سیستمهای خبره در کاربردهای متفاوت، افزایش بهره وری انسانی، سود مالی و جوابی بهتر به نیاز های کاربران را اثبات نموده است. سیستمهای خبره فراوانی به عنوان محیطی جامع برای مدیریت دانش پدید آمده اند.

سیستمهای خبره می توانند برای حل هر نوع مسئله ای بکار روند، از یک مسئله ساده انتخاب در میان گروههای مختلف، تا مسائل پیچیده تری همچون حمایت از مدیران جهت تصمیم گیری. بصورت سنتی گسترش و توسعه سیستمهای خبره چه از نظر زمانی و چه از نظر مالی، هزینه ای گزاف دارد. گاهی ساخت یک سیستم تنها، خود به پروژه ای بزرگ تبدیل می شود. هزینه بالای توسعه و استفاده یک سیستم خبره مانع از آن شده است که از سیستم های خبره جز در مواردی معدود استفاده نشود.

کلید گسترش و استفاده از سیستمهای خبره بصورت وسیع،کارآ و مناسب به همراه هزینه ای پایین، آن است که ابزاری آسان جهت توسعه و تولید سیستمهای خبره داشته باشیم که به سادگی توسط خبرگان مسئله قابل استفاده باشد. برای هرچه قدرتمند نمودن این ابزار می توان از ویژگیهای سطح بالا و پیشرفته همچون اجازه کنترل کامل بر روی موتور استنتاج و چگونگی روند اجرا، ماجولاریتی پایگاه دانش ، واسط کاربر و اجتماع سیستم خبره با دیگر برنامه ها ، استفاده نمود.

افرادی که در طراحی یک سیستم خبره در فازهای مختلف دخالت دارند، غالبا هرکدام از دانش کامپیوتری متفاوت بهره می برند، در نتیجه یک محیط مناسب بایستی ابزارهایی کارا را برای دسته های مختلفی از کاربران در دامنه خبرگان مسئله تا طراحان نرم افزار فراهم آورد. سیستمهای خبره را می توان به کمک پوسته های سیستم خبره (Expert system Shells) ایجاد نمود. یک پوسته خبره، یک نرم افزار محیط برنامه نویسی است، که اجازه ساختن یک سیستم خبره یا سیستم مبتنی بر پایگاه دانش را فراهم می آورد.

معمولا در معماری سیستمهای خبره سه واحد مجزا در نظر می گیرند 1-پایگاه دانش، 2- موتور استنتاج، 3- واسط کاربر گروهی دیگر علاوه بر آنچه ذکر شد واحدی به نام کسب دانش نیزدر نظر می گیرند. معماری ارائه شده از جهات مختلفی با معماری برنامه های کامپیوتری کلاسیک متفاوت است. وجود پایگاه دانش که حاوی دانش خاص در یک دامنه محدود حقیقی می باشد یکی از این تفاوتها می باشد. از آنجا که در سیستم های خبره کد برنامه از پایگاه دانش جدا شده است، و پایگاه دانش به آسانی توسط زبانهایی که توسط غیر برنامه نویسان قابل درک باشد ارائه می شود، سبب شده تا نیاز برای

مهارتهای برنامه نویسی کاهش یابد. با جدا نمودن پایگاه دانش از دیگر واحدهای سازنده یک سیستم خبره می توان یک پوسته سیستم خبره عمومی طراحی نمود. به عبارت دیگر این امکان وجود دارد که برنامه ای ساخت که شامل واحدهای دیگر باشد و غیر برنامه نویسان، خصوصا خبرگان مسئله، با تکمیل پایگاه دانش خاص خود و بهره گیری از واحد های از قبل موجود، سیستم های خبره را تکمیل و تولید نمایند. چنین برنامه هایی پوسته های سیستم خبره نامیده می شوند. گروهی حتی پا را از این فراتر نهاده و هر سیستم خبره را که پایگاه دانش آن تهی باشد یک پوسته سیستم خبره می نامند. به عبارت دیگر این عقیده وجود دارد که می توان موتور استنتاج یکسانی را روی پایگاههای دانش متفاوت به کار گرفت. به عنوان مثال موتور استنتاجی که برای MYCIN پیاده سازی شده بود به EMYCIN (Empty MYCIN) تغییر یافت تا یک سیستم خبره عمومی یا به عبارتی دیگر پوسته ای جهت تولید سیستمهای خبره مختلف در حوزه های مختلف باشد. در حقیقت، پوسته یک سیستم خبره یک معماری عمومی از یک سیستم خبره بدون در نظر گرفتن دانش زمینه ای محدود از یک خبره می باشد. چنین پوسته ای می تواند بوسیله دانش و یا قوانین دیگر برای کاربردهای مختلف به کار رود.

برای سیستمهای خبره تعاریف متعددی وجود دارد. تعریفی که در اینجا بر آن تکیه می کنیم عبارت است از سیستمهای خبره برنامه های کامپیوتری هستند که، از تکنیکهای هوش مصنوعی همچون ارائه سیمبلیک، استنتاج و جستجوی هیوریستیک، جهت انجام وظایفی پیچیده ،که در نظر اول تنها بوسیله انسان خبره انجام پذیر است، استفاده می شوند.(Buchanan,1985) سیستمهای خبره با برنامه های هوش مصنوعی برپایه general-problem-solver (Newell and Simon,1963) متفاوت است. برای تقابل بیشتر، سیستمهای خبره، سیستمهایی مبتنی بر پایگاه دانش" هستند که بر روی دانش زمینه ای خاص بنا شده اند(Feigenbaum,1977).

با توجه به تعریف ارائه شده و اهمیت پایگاه دانش، عموما سیستمهای خبره را با توجه به شیوه ارائه دانش در پایگاههای دانش آنها دسته بندی می نمایند. در این مقال ما تنها بر روی فریمها تمرکز نمودیم و سعی نمودیم با مطالعه آنها، ویژگیها و یا کاستی ایی که در این شیوه ارائه وجود دارد را یادآوری نموده و نحوه پیاده سازی یک پوسته مبتنی بر فریم را دنبال نماییم. ویژگیهای یک پوسته مبتنی بر فریم را مطالعه نماییم.؟؟؟؟؟

5. ارائه دانش و اهمیت آن

قبل از آنکه سیستم خبره بتواند به استدلالی سمبلیک درباره دانش بپردازد باید دانش به شیوه ای در کامپیوتر ارائه گردد. این ارائه می بایستی دارای ویژگیهای خاص باشد از جمله اطلاعات با کارایی لازم ذخیره و بازیابی شوند، دانش ارائه شده به گونه ای باشد که به مسئله پایبند مانده باشد، کامپیوتر بتواند پدیده را بفهمد یعنی، کامپیوتر بتواند اطلاعات را به روشهای معنی دار و متکی به خود ارائه نماید. مسائل ذکر شده عموما با نام مشکلات ارائه دانش خوانده شده و یکی از مسائل مهم در هوش مصنوعی را تشکیل می دهند.

دانش ارائه شده در سیستم می تواند از انواع مختلفی باشد: واقعیت هایی در رابطه با حوزه مسئله ، تئوری هایی درباره حوزه مسئله، قوانین سخت و پروسیجرهایی وابسته به حدود مسائل عمومی، هیوریستیک هایی برای آنکه در یک موقعیت داده شده در مسئله چه باید کرد، استراتژیهای کلی برای حل انواع مسائل و فرا دانش (دانشی درباره دانش).

متعاقب این مسائل، مهندسی دانش نقش پررنگ تری را در طراحی و ساخت سیستم های خبره پیدا می کند. مسائلی از قبیل چگونگی پروسه کسب دانش از انسان خبره و یا دیگر منابع، پروراندن و استخراج دانش، متدهایی که برای نمایش دانش در سیستم خبره به کار می روند ، قرار دادن دانش در متدهای ارائه متفاوت از قبیل قوانین و فریمها، همگی اینها مسائلی هستند که در مهندسی دانش در نظر گرفته می شوند.

در نتیجه طراح یک پوسته سیستم خبره بایستی دیدی کامل و روشن از انواع متدهای نمایش، روشهای مختلف استنتاج، مهندسی دانش و ... داشته باشد و با در نظر گرفتن همگی این فاکتورها، معماری عمومی جهت یک پوسته سیستم خبره ارائه نماید. از آنجا که موضوع بحث ما پوسته ها و ابزارهای گسترش سیستمهای دانش و خبره مبتی بر ساختار فریم است از کلی گویی بیشتر اجتناب کرده و مستقیما به شرح و بسط مفهوم فریم و موضوعات وابسته به آن می پردازیم.

6. ساختـــار فریم ها در نخستین نظر

فریمها، نخستین بار توسط Minsky(1975) به عنوان یکی از استراتژیهای ارائه دانش معرفی شدند که در آن هر فریم یک دسته از اطلاعات مرتبط را نمایش می دهد.اطلاعات همبسته متعلق به یک شی یا رویداد بصورت مناسبی می توانند به کمک فریمها ذخیره شوند. ذخیره اطلاعات به چنین شکلی چه در هنگام ساخت ، چه در هنگام به روزرسانی و یا تصحیح اشتباهات آسان است. فریمها ابزاری متداول جهت ارائه دانش عمومی در حوزه هوش مصنوعی می باشند. فریم می تواند نه

تنها مقدار زیادی از دانش متداول را بدست آورد، همچنین می تواند اطلاعات مربوط به فرضیاتی که باید در نظر گرفته شود، اطلاعاتی که باید جستجو شوند و نحوه جستجوی این اطلاعات را فراهم نماید . استفاده از فریمها به عنوان ابزار ارائه دانش، به درج آسان دانش جدید و ساختن مدل کیفی برای آنالیزهای کارایی و خطایابی کمک می کند.

یک فریم ساختمان داده ای است که معمولا برای ارائه یک شی واحد، یا کلاسی از اشیا مرتبط یا یک مفهوم کلی بکار می رود. در بعضی از سیستمها تنها یک نوع فریم تعریف شده است حال آنکه در بعضی سیستمها انواع مختلفی از فریمها تعریف می شود مثلا دو نوع و یا بیشتر. به عنوان مثال فریم کلاس و مصداق.

فریمها غالبا در یک سلسله مراتب طبقه بندی شده ساماندهی می شوند که در آن هر فریم به فریم والد خود(در بعضی از سیستمها بیش از یک) متصل است. والد یک فریم A مفاهیم کلی تری را نسبت به فریم A در مورد آن شی ارائه می دهد و فرزند A مفاهیم دقیق تری را نسبت به A ارائه میکند. مجموعه ای از فریمها در یک یا چند سلسله مراتب توارثی یک پایگاه دانش است.(KB)

اجزا تشکیل دهنده فریمها Slot نام دارند. اسلات های یک فریم خصیصه و یا ویژگیهای چیزی که توسط فریم ارائه شده است، را شرح می دهند، در عین حال می تواند یک رابطه دودویی میان فریم و فریم دیگر را نشان دهد. به علاوه به منظور ذخیره مقادیر، اسلاتها همچنین حاوی محدودیتهایی بر روی مقادیر قابل قبول هستند. تعریف اسلات معمولا شامل اجزا دیگری علاوه بر نام ، مقدار و محدوده مقادیر است، مانند نام پروسیجری که می تواند برای محاسبه مقدار اسلات به کار رود و یا برای اعتبار سنجی مقدار محاسبه شده برای یک اسلات استفاده گردد. این اجزا مختلف از یک اسلات، Facet نامیده می شوند.

توارث سبب می شود تعریف اسلات در سلسله مراتب طبقه بندی به پایین منتشر شود. برای مثال هنگامیکه فریمی با نام کبوتربه عنوان فرزند فریم پرندگان ساخته می شود، آنگاه اسلات های فریم کبوتر به طور خودکار ویژگیهای اسلاتهای پرندگان را به ارث می برند.

توارث یک ابزار بسیار سودمند جهت مهندسی پایگاه های دانش پیچیده هستند. هنگامی که کاربر فریم جدیدی می سازد و آن فریم اسلات ها را از والد به ارث می برد، اسلات های ارث برده شده همانند الگویی کاربر را در پر کردن دانش در مورد مفهوم جدید راهنمایی می نماید. به دلیل آنکه اطلاعات همه اسلات ها و Facet ها در هنگام اجرا در دسترس است (برخلاف زبانهای برنامه نویسی شی گرا مانند C++)، برای یک برنامه همچون یک واسط کاربر این امکان وجود دارد که

بتواند کاربر را در وارد نمودن دانش جدید راهنمایی نماید. برای مثال واسط کاربر می تواند مستقیما نوع داده و محدوده مقادیر یک اسلات را تشخیص دهد. توارث همچنین به تغییرات سیستماتیک در یک دانش پیچیده کمک می نماید. برای مثال اگر بفهمیم که همه کامپیوتر های Cray می توانند از Unix علاوه بر CaryOS استفاده نمایند، این اطلاعات به راحتی می تواند در یک محل بوسیله تغییر اسلات مربوط به سیستم عامل در فریم مربوط به کامپیوترهای cary کد شود.

بعضی سیستمهای ارائه دانش با ساختارفریم (FKRS) رابطه میان فریمهای کلاس را محاسبه میکنند که Subsumption نامیده می شود و اجازه می دهد FKRS به طور خودکار محل صحیح یک کلاس را در طبقه بندی سلسله مراتبی تشخیص دهد(کلاس بندی یک کلاس). فریم A شامل (subsume) فریم B است در صورتیکه A مفهوم کلی تری را نسبت به B تعریف نماید، به این معنی که کلیه نمونه ها از مفهوم B نمونه ای از فریم A باشند.

اسلات ها و facetها از یک فریم می توانند بوسیله دانش ایستا و یا پویا پر شوند. دانش ایستا در هنگام ساخت فریم پر می شود و دانش پویا در هنگام اجرا. در تصویر1 ساختارنمونه ای پایه مربوط به فریم نشان داده شده است.

(<frame name>
(<slot1>([<facet1>] Value <value1>
<value2>……<valuek>)
([<facet2>] Value <value1>
<value2>……<valuek>)
. . . . . .
(<slot2>(Value
<value1><value2>…….<valuek>))
(<slot3> ([<facet1>] Value
<value1> <value2>…… ……<valuek>)
. . . . .)))
where:
sloti : is the *ith* attribute.
facetj : is the *jth* optional part of a slot.
valuek : is the *kth* value associated with a slot or facet.

شکل شماره 1 : نمونه ای برای ساختار عمومی یک فریم

با گذشت سالیان مشی Frame based از مقبولیت و محبوبیت فراوانی برخوردار شده است.

7. روشهای استنتاج مبتنی بر فریم

سه روش متفاوت جهت استدلال توسط فریمها به کار گرفته می شود: دانش قراردادی، توارث، و استفاده از قواین رویه ای توکار شده با فریمها.

- دانش قراردادی: یکی از غالبهای قدرتمند دانش "دانش قراردادی" (default knowledge) می باشد. و این دانشی است که در آن فرض می شود همواره در رابطه با یک فریم درست است مگر آنکه خلاف آن ذکر شود. برای مثال، یک فریم شخص در یک سیستم خبره تشخیص طبی ممکن است حاوی چنین دانش قراردادی باشد: فرض می شود هر شخصی دو دست، دو چشم، دو کلیه، دو شش و ... است. تنها برای شخصی که یکی از این جفت ارگان های بدنش را از دست داده می بایستی اطلاعات اضافی ارائه شود.

  دلیل قدرت دانش قرار دادی به عنوان یک ابزارآنست که اجازه می دهد فریمها مقدار زیادی از دانش را بصورتی کارآ ارائه نمایند .برای مثال، فریمی که یک قطعه خاص از تجهیزات را نمایش می دهد می تواند بصورت قراردادی فرض نماید که تجهیزات سوار شده اند و به درستی کار می کنند. هنگامیکه دانش قراردادی به تعداد زیادی از حالتهای متداول جمعیت نمونه های ارائه شده اشاره نماید، بسیار کارا خواهد بود.

- تـــــوارث : یکی از قدرتمندترین اشکال استدلال در فریمها توارث می باشد. فریمهای خاص می توانند خصوصیات فریمهای عمومی تر را به "ارث" برند. در این مورد در قسمت های بعدی توضیحات بیشتر همراه با مثال ارائه می شود. توارث اجازه صرفه جویی زیادی در ارائه می دهد چراکه بسیاری از فریمهای خاص شده می توانند بسیاری از خصوصیات را از فریمهای کلی تر به صورت قراردادی به ارث برند بدون آنکه نیاز باشد آن خصوصیات در هر مصداق ذکر شود. تنها هنگامیکه مقادیر قراردادی ارث برده نمی شوند نیاز برای ذکر شرحی صریح در پایگاه دانش بوجود می آید.
- استفاده از قوانین رویه ای در فریمها: فریمها می توانند برای ارائه هر دو دانش رویه ای و توصیفی به کار روند. دانش رویه ای غالبا بوسیله قوانین تولید ارائه می شود، جاییکه پروسیجرها به یک اسلات خاص و یا کل یک فریم الحاق شده اند. روش متداول استفاده از قوانین رویه ای – شیطانک ها است که تنها در مواردی که شرایط خاصی برقرار باشد اجرا

می گردند. شیطانک حاوی قوانینی است برای مشخص نمودن مقداریک فریم در صورتیکه مقدار آن مشخص نباشد و مورد نیاز باشد.

8. سیستم ارائه دانش بر پایه فریم از منظر عملیاتی

در این قسمت دو کلاس عملیاتی که توسط FKSR برای حل مسائل قابل پیاده سازی اند معرفی می شوند. ذخیره مستقیم و کسب دانش، و حل مسئله وابسته به استنتاج.

1.4 ذخیره و کسب دانش

بیشترFKSRها کتابخانه ای از توابع را فراهم می آورند که برنامه های کاربردی می توانند با فراخوانی آنها عملیات مختلفی را انجام دهند. به عنوان مثال اضافه نمودن مقدار جدید به یک اسلات، حذف یک یا چند مقدار ازیک اسلات، بازیابی مقدارفعلی مربوط به یک اسلات، ساختن فریمی جدید با والد مشخص، عوض نمودن والدهای مربوط به یک فریم، حذف فریم، تغییر نام فریم، اضافه نمودن اسلات جدید به یک کلاس از فریمها و یا اضافه نمودن یک Facet به یک فریم. بعضی از سیستم ها همچون KEE ، Strobe ، Cycl و Kreme به کاربر اجازه ساخت واسط های گرافیکی را می دهند.

در مشی ای که Krypton در آن پیشقدم بود، بعضی FKSR ها زبانهای توصیفی محیا نمودند که کاربر بتواند از آن، هم برای پرسش از پایگاه دانش و هم برای اظهار نظر درباره حقایق جدید در پایگاه دانش، استفاده نماید.

2.4 حل مسائل و استنتاج

برنامه های کاربردی که در تعامل با یک FKRS هستند معمولا از قوانین ویا از کلاس بندی برای محیا نمودن استنتاج بر پایه دانش ذخیره شده در FKRS استفاده می نمایند. بیشتر FKRS ها یکی از قوانین یا کلاس بندی را پشتیبانی می نمایند و در مواردی بعضی از سیستمها همچون Loom و Classic از هر دو پشتیبانی می نمایند. در KEE و CLASS هر قانون تولید به تنهایی در یک فریم واحد کد می شود. در KEE و PROTEUS، پرسشها با کمک مفسر استنتاج رو به عقب، از قوانین برای استنتاج مقدار اسلات مورد نظر استفاده می کنند. به طور مشابه، در THEO کاربر می تواند قوانین PROLOG را به اسلات ها الحاق نماید در نتیجه آن THEOمی تواند با استنتاج رو به عقب

مقادیر اسلات ها را بدست آورد. KEE وPROTEUS همچنین می توانند از استنتاج رو به جلو هنگام ایجاد مقادیر برای اسلات های جدید استفاده نمایند.

کلاسبندی برای پشتیبانی از استنتاج در دو روش مختلف استفاده می شود. در اولین مورد، کلاس بندی می تواند عینا همان حل مسئله باشد مثلا در صورتیکه سیستم بتواند با توجه به توضیحات بیماری را به عنوان نمونه ای از یک کلاس بیماری تشخیص دهد، در این صورت سیستم یک تشخیص طبی را محاسبه نموده است. در روش دوم، در FKSR های خانواده KL-ONE، کلاسبندی یکی از اجزا کلیدی پردازشگر پرس و جو است که به سیستم اجازه میدهد درباره رابطه میان ترم های استفاده شده در پرس و جو و ترم های موجود در پایگاه دانش، استدلال نماید. این سیستمها ، یک پرس و جو را بوسیله ترجمه نمودن پرس و جو به شرح یک مفهوم و سپس کلاسه بندی کردن ان مفهوم جهت مشخص نمودن جایگاه آن در طبقه بندی سلسله مراتبی ، پاسخ می دهند. همه مفاهیم در پایین مفهوم پرس و جو در سلسه مراتب، بوسیله پرس و جو استنتاج می شود و بدین ترتیب جواب مربوط به پرسش را شامل می شود.

اصل مهم از ارائه دانش که تا به حال مورد توجه قرار گرفته است، پیچیدگی محاسباتی subsumption و در نتیجه کلاسبندی می باشد . محققین با مقایسه هزینه محاسباتی subsumption درتعداد زیادی زبانهای FKRS به این نتیجه رسیده اند که هر چه زبان expressive تر باشد، با هزینه بالاتری در محاسبات subsumption در آن زبان روبرو هستیم. این نتیجه با نام expressiveness tractability tradeoff شناخته می شود.

بعضی FKRS ها با ترکیب نمودن تواناییهای استنتاج خود با مکانیزمهای زمینه و سیستمهای truth-maintenance افزایش می دهند.

بسیاری از سیستمها بوجود آمده اند که جهت ارائه دانش و استنتاج از ترکیب منطق و فریم ها استفاده نموده اند. در اغلب این سیستمها به منصق تقدم بیشتری داده شده است و از این جهت سبب ایجاد یکسری مشکلات برای آنها شده است از جمله ناتوانی در نشان دادن استثنائات و محیا نمودن استنتاج قراردادی. در گروهی دیگر از سیستمها همچون SILO ازروشهایی مشابه با منطق جهت استنتاج و حل مسائل کمک گرفته شده است و برای این منظور توابعی کارآ تعریف شده است. همچنین از اثبات بر پایه کلاس که خود یک نوع عمومیت از جستجوی شی می باشد ، استفاده می گردد. از آنجا که در این سیستم از مفاهیم منطق استفاده شده است، کلیه روشهای اثبات در منطق همچون Resolution و Unification پشتیبانی می گردد. البته باید توجه نمود که اگر از محیطی مانند

Prolog جهت پیاده سازی استفاده شود، در نهایت هر یک از روشهای استنتاج که به کار برده شوند به نوعی توسط منطق پیاده سازی وحل خواهند شد.

در دسته ای دیگر از سیستمها که با ابزارهای فازی همراه شده اند، از روشهای استدلال فازی همچون استفاده از ماتریسهای تصمیم و به کارگیری متغیرهای زبانی جهت استدلال و حل مسئله استفاده می شود. نمونه ای از این معماری توسط آقایان A.K.Sharma, Chander Kumar, Khurram Mustaf , Ashok Kumar در National Workshop on IT Services and Applications (WITSA2003) Feb 27-28, 2003 ارائه شده است.

10. مسائل اصلی طراحی در سیستمهای ارائه دانش

محدوده وسیع طراحی FKRS بطور ضمنی به این واقعیت اشاره دارد که طراح یک FKRS می بایستی درباره مسائل متعددی تصمیم بگیرد. برای مثال اینکه چه مدلی از فریم و اسلات به کار گرفته شود، از چه نوع مکانیزم توارث استفاده شود، از کلاسبندی استفاده نماید یا نه، چه نوع الگوریتمی را برای Subsupmtion به کار گیرد و ... . می بایستی مجموعه جامعی از اصول ارائه دانش وجود داشته باشد که طراح و کاربر FKRS را در میان گستره ای از انتخابات پیچیده راهنمایی نماید. کاربر FKRS احتیاج دارد تا بداند چه ترکیبی از ساختار نمایشی به او اجازه می دهد تا یک نرم افزار کاربردی را به سرعت و با کارایی قابل قبول بسازد. همچنین احتیاج دارد تا بداند چه نوع ساختار نمایشی قادر است دانش موجود در دامنه کاری او را موجزتر و طبیعی تر کد نماید، تا در نتیجه آن نرم افزار بتواند آسانتر نگهداری شود. کاربران احتیاج دارند تا زمینه های تئوری هزینه و سود ناشی از ویژگیهای FKRS بدانند، همانطور که نیازمندند بانند چگونه یک FKRSخاص، بر حسب احتیاجات نرم افزار کاربردی آنها آماده و محیا خواهد شد. اصول ارائه دانش باید چنان باشد که بتواند کاربر را در انتخاب بهترین FKRS برای حل یک مسئله خاص – سیستمی که دارای بیشترین سود و کمترین هزینه باشد- راهنمایی نماید.

هنگام طراحی یک FKRS برای حل یک یا چند کلاس از مسائل، مجریان می بایستی فرا مجموعه ای از این نظریات را اداره نمایند. مجریان همچنین برای پیش بینی اینکه چه ترکیبی از شیوه ارائه، متضمن اعجاز و کارایی کافی برای برنامه خواهد بود، می بایستی تعدادی تصمیمات مهندسی بگیرند. برای مثال مجریان باید میان استراتژی های پیاده سازی انتخابی برای روشهای نمایشی که انتخاب کرده اند تصمیم گیرند. و همچنین می بایستی امیدوار باشند که پیاده سازی هریک از ویژگیهای

FKRS مستقل از یکدیگرند، در صورتیکه معمولا این ویژگیها بر یکدیگر تاثیر گذار بوده و این بیان می کند که FKRS ماجولار نیست و در نتیجه گسترش و پیاده سازی، عیب یابی، نگهداری و اصلاح آن مشکل خواهد بود.

از دیگر خصوصیات اصلی یک سیستم ارائه دانش همانطورکه قبلا در مورد ان بحث شد expressiveness tractability tradeoff است که به رابطه میان میزان مشعر بودن زبان ارائه دانش با میزان محاسبات کلاسبندی در آن زبان می پردازد.

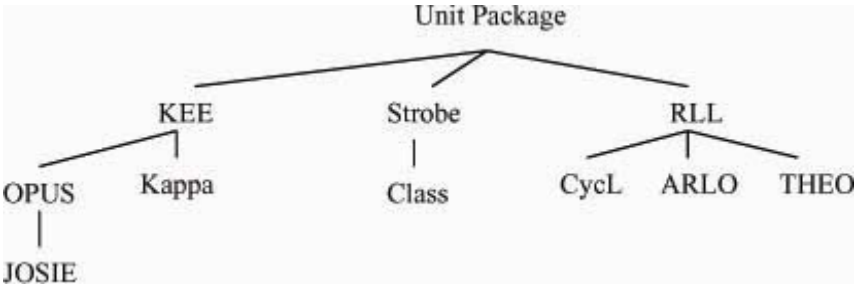
11. آشنایی با بعضی خانواده های FKRS

در اینجا بصورت خیلی مختصر به هر یک از خانواده های سیستمهای ارائه دانش بر پایه فریم پرداخته شده ودرباره ویژگیهای کلی هر یک بحث شده است. شکل زیر چندین فامیلی از FKRS را نشان می دهد. خانواده UNITS در دانشگاه استنفورد در اواخر 1970 بنیان گذاشته شد. بخشهای آن شامل UNIT Package، STROBE،CLASS، RLL، CYCL،ARLO، THEO، JOSIE، OPUS و سیستمهای تجاری KEE و KAPPA میباشد.

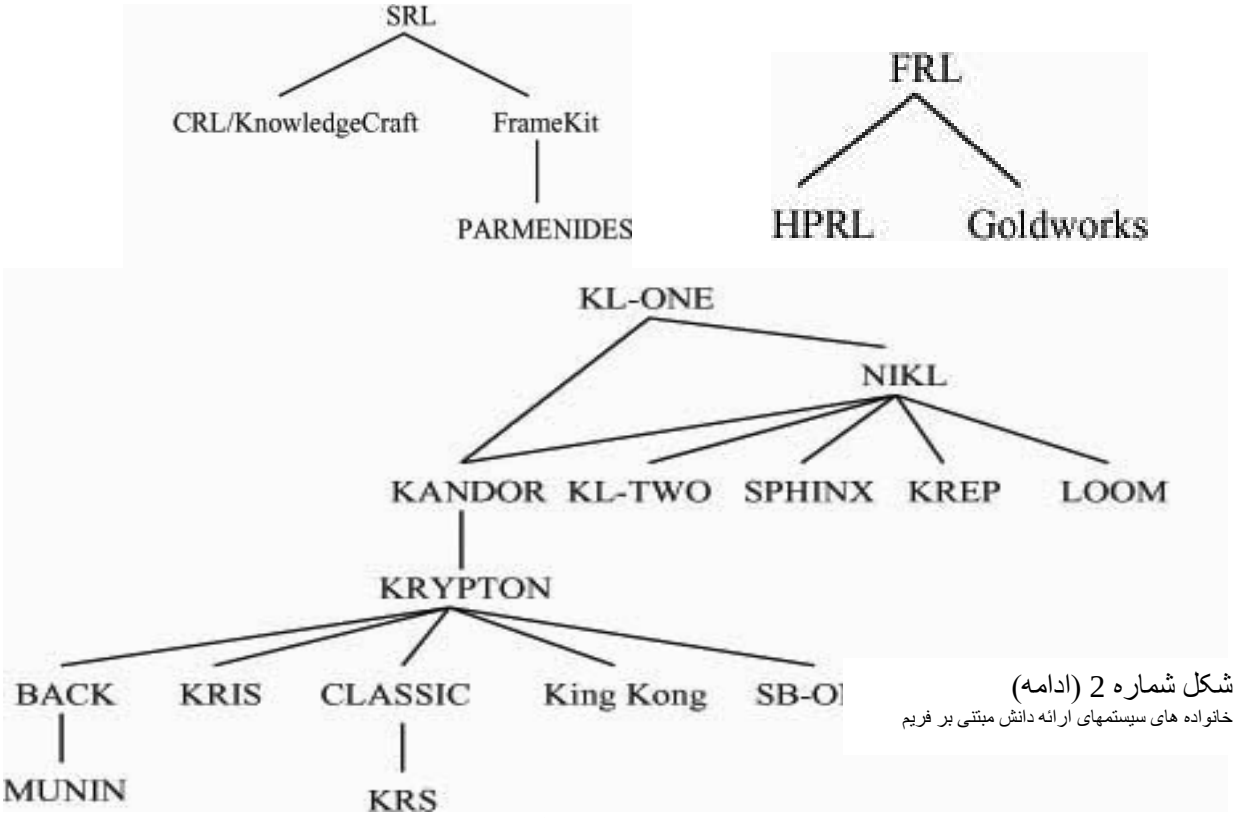
خانواده KL-ONE در دانشگاه هاروارد در دهه 70 پایه گذاری شده است. بخشهای ان شامل KL-ONE، NIKL، KANDOR، KL-TWO، K-REP، KREME، BACK، MUNIN، SPHINX، KRIS، MESON، SB-ONE، KRYPTON، LOOM، CLASSIC می باشد.

خانواده FRL درMIT در اواسط دهه 70 بنیان گذاشته شده است. اعضای آن شامل FRL، HPRL وGOLDWORKS می باشد.

چندین FKRS وجود دارند که در خانواده بزرگتری جای نمی گیرند. نمونه ای از اینها شامل



شکل شماره 2
خانواده های سیستمهای ارائه دانش مبتنی بر فریم



شکل شماره 2 (ادامه)
خانواده های سیستمهای ارائه دانش مبتنی بر فریم

PROTEUS، OZONE، LOOPS، RHET و ... می باشد.

12. انواع فریمها در سیستمهای FKRS

غالبا FKRSها دو نوع مختلف از فریمها را تعریف می نمایند: فریمهای کلاس که یک کلاس یا مجموعهای از یک مفهوم کلی وا یک معنی را ارائه می نمایند. به عنوان مثال : کلاس همه کامپیوتر ها، مجموعه همه کامپیوترهای تولید شده توسط IBM، مفهوم پدر.

فریمهای مصداق چیزهای منحصربه فرد- موجودیتهای واقعی موجود در جهان را نمایش می دهند، مثلا: کامپیوتر خاصی که با آن کار می کنم، شخصی که پدر من است. محققین مختلف دیدگاههای متفاوتی به معنای فریمهای کلاس و مصداق دارند، علاوه بر اینکه انواع مختلف دیگری از فریمها را تعریف نموده اند.

13.اصطلاحات فنی در اتصال(Link) فریمها

در این بخش در مورد اصطلاحاتی که به رابطه میان کلاسها و مصداقها در یک ساختار طبقه بندی سلسله مراتبی می پردازد، بحث می شود. به طور مجازی هر خانواده از FKRSها اصطلاحات

خاص خود را دارند. استفاده از اصطلاحات ناسازگار و دارای افزونگی، ارتباط میان پژوهندگان ارائه دانش را مختل نموده است، همانگونه که دیگر محققین در زمینه های مختلف علوم کامپیوتر را سردرگم نموده است، چنانکه می پندارند اصطلاحات متفاوت می بایستی مفاهیم متفاوتی را بیان نمایند. اصطلاحاتی که در اینجا آورده شده است، شامل اصطلاحاتی است که توسط آقای Russinoff ارائه گشته است.

همواره علاقمند بوده ایم که رابطه موجود میان فریمهای کلاس و فریمهای مصداق را نام گذاری نماییم. در صورتیکه فریم کلاس C1 مستقیما بالای کلاس فریم C2 در سلسله مراتب، متصل شده باشد، آنگاه گفته می شود که C1 یک direct-super از C2 می باشد و C2 یک direct-sub از C1 می باشد. به این نوع رابطه رابطه super-sub گفته می شود. در صورتیکه یک فریم کلاس C مستقیما بالای فریم مصداق I متصل باشد، آنگاه گفته می شود C یک template از I است وI یک Instance از C می باشد. توجه شود که رابطه های تعریف شده در بالا، رابطه هایی متعدی هستند.

گفته می شود A یک والد از B است، چنانکه A یک Direct-super ویا یک template از B باشد (که در اینصورت B فرزند A است). Ancestor(جد) به عنوان یک تعدی از رابطه والد تعریف می شود و descendant(نسل) یک تعدی از رابطه فرزندی تعریف می شود.

14. تنوع فریم ها

بعضی FKRS تنها یک نوع از فریم را به جای فریم های کلاس و مصداق بکار می برند . مثلا NIKL تنها فریم کلاس را فراهم نموده است، زیرا بوجود آورندگان آن ماموریت NIKL را تنها پشتیبانی از تعاریف مفاهیم و روابط میان آنها در نظر گرفته اند، نه به عنوان ابزاری برای نتیجه گیری درباره یک نمونه خاص.(توجه شود که در این سیستم برای تعریف یک فریم نمونه و یا تنها، فریم کلاسی در نظر می گیرند که تنها یک مصداق دارد.) گروهی دیگر همانند THEO نیز تنها از یک نوع فریم استفاده نموده اند زیرا معتقد بودند که تفاوت میان کلاس و مصداق چیزی است که خوش تعریف نیست.

دیگر سسیستم های FKRS حداقل شامل فریم هایی از نوع مصداق و کلاس می باشند، اما بعضی از سیستمها انواع اضافه تری از فریم ها را معرفی نموده اند:

- متا کلاس. PROTEUS از فریمهایی که به نام متا کلاس شناخته می شوند، برای تعریف مجموعه ای از کلاسهای PROTEUS استفاده می نماید. هر PROTEUSکلاس یک نمونه

از متا کلاسی است که کلاس نامیده می شود. کاربران می توانند متا کلاسهای دیگر را به عنوان direct-sub از متاکلاس **کلاس**، تعریف نمایند. برای مثال می توان متا کلاسی تعریف کرد به نام computer family که VAX family (یک کلاس) یک مصداق از آن باشد. **کلاس** یک کلاس از همه کلاسهاست و یا بطور مشابه مجموعه ای از همه مجموعه ها. LOOPS و CYCL نیز از متاکلاسها استفاده می نمایند: هر فریم CYCL یک مصداق از فریم Collection ویا فریم IndividualObject است. Collection متا کلاسی است که معادل کلاس در PROTEUS می باشد. در CYCL تنها فریمهای کلاس (مصداقهای Collection) می توانند مصداق داشته باشند.

- Indefinites. UNIT Package و CLASS از فریمهای Indefinite برای نمایش نمونه هایی که هویت آنها نامشخص است استفاده می نمایند. Indefiniteها به کاربر اجازه میدهند تا بگوید که دو نمونه همانند، هستند بدون آنکه از هویت آنها مطلع باشند.
- Description. این فریمها کلاسهای متغیر شده ای هستند که در UNIT Package برای نمایش هدفها در مسائل برنامه ریزی شده به کار می روند.
- Prototypes. KRL،RLL و JOSIE از فریمهای Prototype برای نمایش اطلاعات درباره یک مصداق معمول از یک کلاس استفاده می نمایند، هنگامیکه آن مصداق با کلاس خود درتناقض است.
- SlotUnits. در CYCL، SlotUnits فریمهایی هستند که اطلاعاتی را درباره اسلات ها نگهداری می کنند.
- SeeUnits . CYCL از SeeUnits، به عنوان پی نوشت یا شرحی بر دیگر فریمها استفاده می کند.
- CompactUnits. فریمهای مصداقی هستند در KEE، که فضای ذخیره سازی کمتری را اشغال می کنند و سرعت دسترسی بالاتری نسبت مصداقهای نرمال دارند. یک CompactUnits می بایست تنها یک Template داشته باشد و حتما مجموعه اسلاتهای آن مانند template مربوطه اش باشد. در LOOM|، مصداقهای CLOS حالت مشابه را داشته و می توانند کارایی بالاتری در پیاده سازی فریمهای مصداق داشته باشند.

15. اسلات ها

هر فریم از تعدادی اسلات تشکیل شده است که معمولا ویژگیها و یا خصیصه های مربوط به آن شی یا مفهوم را بوسیله فریم ارائه می نمایند. اسلات ها همچنین برای برای نمایش روابط دودویی میان فریم حاوی آنها و دیگر فریمها به کار می روند. در ساده ترین مدل به هر اسلات یک نام داده می شود و مقداری متناظر با آن در نظر گرفته می شود.

محققین به روشهای گوناگونی این مدل را آراسته اند. تقریبا همه سیستم ها چند خصیصه دیگر علاوه بر مقدار را برای هر اسلات در نظر می گیرند، مانند نوع داده اسلات و محدودیت بر روی دامنه مقادیر نسبت داده شده به اسلات. گروهی با کلی تر کردن این ایده اجازه داده اند تا اسلات ها خصوصیت هایی اختیاری وقردادی داشته باشند که Facet نامیده می شود، نام، مقدار، نوع داده و محدوده مقادیر به طور معمول مکمل آن است. دیگر facetهای معمول که استفاده شده اند شامل: شرح، میزان درستی مانند آنچه در MYCIN به عنوان ضریب قطعیت مطرح شده است، توضیح و توجیه که مقادیر اسلاتهای دیگر که این مقدار برای اسلات کنونی از آنها به دست آمده است را مشخص می نماید(به کار برده شده در THEO) ، پروسیجرهای الحاقی، و یژگیهای مکانیزم توارث برای آن کلاس و ... .

THEO از ایده کلی تری استفاده نموده است: facetها خودشان می توانند facet داشته باشند. در هر سطحی THEO اجازه اسلاتهای دلخواه تو در تو در اسلاتهایی در اسلاتهای دیگر را می دهد. بعلاوه THEO ابزارهایی برای الحاق فرا اطلاعات فراهم نموده است. THEO بطور گسترده ای از subslotها برای کسب اطلاعات مشتق شده بوسیله مکانیزمهای استنتاج مختلف استفاده می نماید. برای مثال فرض کنید که کاربر پرسشی برای مقدار Frank.children دارد. اگر هیچ مقدار محلی پیدا نشود، یکی از مکانیزمهای استنتاج THEO سعی می کند این مقدار را از Frank.Daughters بدست آورد البته در صورتیکه درسیستم مشخص شود که رابطه Daughters، یک رابطه خاص از رابطه Children است.

OZONE از مشی دیگری استفاده می کند: اسلاتها به گروههای جداگانه که فضا نامیده می شوند تقسیم بندی می گردند. هر فریم به طور معمول سه فضا دارد که به نامهای فضای سیستم (این اسلاتها نام فریم و والدهای فریم را لیست می کند)، فضای تابع (این اسلات ها حاوی پروسیجرهای الحاقی است)، فضای متغیر(برای اسلاتهای تعریف شده توسط کاربر) شناخته می شوند. فضاها دو حسن بزرگ دارند: فضاهای نام جداگانه، برای تعریف انواع مختلف از اسلات ها، تعریف می کنند،

برای مثال جهت جلوگیری از برخورد نام میان اسلات های سیستم و، اسلاتهای تعریف شده توسط کاربر. مزیت بعدی امکان استفاده بیشتر از فضای ذخیر ه سازی ثانویه و فراهم آمدن استفاده از Incremental loading است – اسلاتهای فضای سیستم می توانند مستقل از اسلاتهای فضاهای دیگر بار شوند.

سیستم JOSIE امکان مشاهده مقادیراسلات به عنوان کلاس ها را فراهم می نماید، که سبب آن می شود که بتوان مجموعه ای از اثبات ها را برای مقادیر اسلات ساخت. این به معنی آن است که اگر مقادیر یک اسلات شامل مجموعه ای باشد، این این مکانیزم اجازه می دهد با این مجموعه همچون یک کلاس برخورد نمود و بتوان از زبان "تعریف کلاس" در سیستم، برای ساخت دلائل و اثبات ها استفاده نمود.

SlotUnit .1.10

اکثر FKRS حاوی نوعی از فریم هستند که سازندگان CYCL آن را SlotUnit می نامند. یک SlotUnit فریمی است که اطلاعاتی درباره یک اسلات خاص نگهداری می کند و چگونگی استفاده از آن اسلات را در یک پایگاه دانش توضیح می دهد. در خانواده سیستم های UNITS، ایده مربوط به SlotUnit ها در دو سیستم RLL و OPUS وجود دارد، همچنین THEO نیز از آن بهره می برد. SRL از SlotUnit ها تنها برای اسلاتهایی که یک رابطه دودوئی میان دو فریم را نمایش می دهند، استفده می شود.

2.10. نوع داد ه های اسلات

بیشتر FKRSها اسلاتها را در غالب مجموعه ها مدل می کنند.در گروهی دیگر، با اسلات ها همچون لیستی از مقادیررفتار می شود. تفاوت مجموعه ها با لیست ها اینست که در لیست ها، ترتیب قرار گیری اجزا مهم است و امکان وجود مقادیر تکراری اجازه داده می شود.LOOM با مقادیر اسلات ها همچون مجموعه های مرتب رفتار می نماید، تکرار مقادیر جایز نیست اما ترتیب اجزا مهم است و در نظر گرفته می شود. نکته جالب آنست که هیچ کدام از FKRSها بطور مجازی به کار اجازه نمی دهند صراحتا یکی از انواع داده را از میان انواع مختلف لیست، مجموعه، مجموعه مرتب انتخاب نمایند. البته بعضی از سیستمها همچون K-REP از این قاعده مستثنی هستند. در بعضی از سیستم ها این امکان به کاربر داده می شود که تک مقداری یا چند مقداری بودن اسلات را تعیین نمایند.

مجموعه ها نسبت به لیست ها دارای این برتری هستند که منطق موجود در فریم ها را ساده نموده و روابط Subsupmtion خوش تعریف دارند.

بعضی از سیستمها به کاربر اجازه می دهند تا نوع داده هر یک از مقادیر در مجموعه یا لیست را مشخص نمایند. برای مثال این نوع داده می تواند اتم، بولین، عدد صحیح، بازه، عدد، رشته، جدول، متن و ... باشد.

12. تــوارث

توارث یک مکانیزم استنتاج است که اطلاعات درباره یک فریم، در یک ساختار سلسله مراتبی از طریق والد هایش بدست می آید. سوالات مهمی که در این زمینه مطرح می شوند شامل زمان محاسبه توارث، امکان وجود ناسازگاری و چگونگی برخورد با آن ، انواع معانی توارث میباشد.

هنگام تعریف یک اسلات S در یک فریم کلاس C، توارث سبب می شود کلیه فرزندان C حاوی اسلات S باشند. به عبارت دیگر هر فرزند C حاوی اسلاتی با نام S خواهد بود که دارای نوع داده و محدوده مقادیر ثابت همانند S در کلاس C خواهد بود. هدف خواسته شده از این عملیات این است که اگر یک مشخصه خاص یا رابطه به یک کلاس از اشیا، یا به یک مفهوم اعمال شود، آنگاه آن مشخصه یا رابطه بایستی به همه فرزندان آن کلاس یا مفهوم، با حداقل محدوده مقادیر صریح همچنان که به کلاس یا مفهوم اعمل گشته است، اعمال شود.

تفاوت اساسی میان FKRS در اینست که آیا فریم فرزند مقداراسلات را در هنگام توارث بدست می آورد یا خیر. در سیستمهای موجود در خانواده UNITS و SRL ارث بری مقادیر اجازه داده شده است. ایده این عملیات اینست که، هنگامیکه یک مقدار اسلات در فریم کلاس تعریف شده است، ترجیح می دهیم سیستم مقدار نسبت داده شده به کلاس را برای فرزندان در نظر بگیرد مگر آنکه کاربر یک مقدار خاص را برای آن اسلات در نظر بگیرد.

تفاوت دیگر در اینست که بعضی FKRSها اجازه توارث خصوصیات از میان روابط دیگر به جز رابطه Super-sub و مصداق را می دهند. در SRL و CYCL کاربران با بکار گیری SlotUnit چگونگی محاسبه توارث برای هر رابطه را شرح می دهند. SlotUnit مشخص مینماید که کدام اسلاتها از طریق این رابطه ارث بری دارند. علاوه بر این در SRL ،این مکانیزم اجازه می دهد کاربران نگاشت مقادیر اسلاتها را نیز مشخص نمایند.

در سیستم JOSIE، توانایی رفتار با اسلات ها همانند کلاسها، اجازه انواع دیگری از روابط توارث را می دهد. این اجازه می دهد که بتوان ادعا نمود که مقادیر داده شده در یک اسلات از یک فریم داده شده، زیر مجموعه ای از مقادیراسلات دیگر در یک فریم دیگر است.

بعضی از سیستمهای ارائه دانش امکان توارث در مدهای معنایی مختلف را می دهند. به عنوان مثال در سیستمهای KL_ONE و SRL بعضی از این مدها عبارتند از:

- OVERRIDE.VALUES : مقادیر محلی بر مقادیر ارث برده شده ارجحیت دارند.
- UNION: مقادیر محلی با مقادیر ارث برده شده اجتماع می شوند.
- SAME.VALUES: مقادیر محلی متفاوت اجازه داده نمی شود.
- MINIMUM و MAXIMUM: مقدار حداقل یا حداکثر میان مقادیرارث برده شده ومقادیر محلی نتخاب می شود.
- METHOD: برای پروسیجرهای الحاقی به کار می رود.
- و ...

در میان مدهای ذکر شده OVERRIDE.VALUES در بیشتر FKRSها یافت می شود، هر چند که بقیه مد ها نیز کاربرد دارند.

به عنوان مثالی دیگر، در سیستم SILO ، مکانیزم توارث یک مکانیزم بنیادی است. بصورت معمول، امکان توارث چندگانه وجود دارد. یک مصداق فرم دانش مربوط به همه کلاسها و والد های خود را به ارث می برد.در صورتیکه ناسازگاری بوجود آید مکانیزمهای جهت رفع آن به کار خواهد افتاد.

3.11. ناسازگاری در توارث

درسیستمهای اولیه مانند UNIT، فریم تنها می توانست یک والد در سلسله مراتب توارث داشته باشد. سیستمهای بعدی همانند KEE، CYCL و SRL، هر فریم می توانست چند والد داشته باشد و در نتیجه اطلاعات را از بیش از یک والد به ارث برد. بیشتر FKRSها که اجازه توارث از خصیصه های ناسازگار کننده را می دهند مکانیزم های مختلفی را را برای حل ناسازگاریهای بالقوه ارائه می دهند. در بعضی از سیستمها، کاربر والدی که باید مقدار از آن ارث برده شود را مشخص می نماید، بدینگونه زمینه ای که مقدار باید از آن گرفته شود مشخص می شود. در بعضی دیگراز سیستمها مکانیزمی به کار می رود که در آن اولین مقدار که در جستجوی اول-عرض از اجداد فریم فرزند یافت شود، به عنوان مقدار ارث برده شده مشخص می نمایند. در بعضی دیگر از سیستم ها

مـی تـوان از اسـتراتژیهای جسـتجو متفـاوتی اسـتفاده نمـود. در گـروهـی دیگـر نیـز از توابـع تعریـف شـده توسط کاربر برای این منظور استفاده می شود.

مشـکل حـل ناسـازگاری هنگامیکـه FKRS ایـن تـوارث را روی یـک رابطـه چنـدتایی محاسـبه می نماید بزرگتر خواهد بود. در سیستم SRL برای حل این مشکل می توان ترتیب جستجوی هر یـک از اتصالات را برای یک فریم داده شده مشخص نمود.

در سیسـتم SILO جهـت رفـع ناسـازگاری در تـوارث، خـاص تـرین اطلاعـات در بـاره یـک خصیصه، در مقابل اطلاعاتی که کلی ترند، باطل مـی شـوند. در اینجـا دو مشـکل پیش مـی آیـد، چگونـه دانش ناسـازگار را تشـخیص دهیم و چگونـه خـاص تـرین حالـت پیش آمـد آن را، پیـدا نمـاییم. بایـد توجـه داشت که توارث در این سیستم دو جنبه دارد، اول توارث محتویات، یعنی کدام قسمت از دانش دربـاره یـک شـی، ارث بـرده شـده اسـت و دوم، ترتیـب تـوارث در یـک تـوارث چندگانـه اسـت. در SILO، مکانیزمهایی جهت مشخص نمودن این جنبه های توارث وجود دارد و ناسازگاری توسط این مکانیزمها و تعاریف ارائه شده تشخیص داده می شوند. پس از تشـخیص ناسـازگاری جهت رفـع آن، اصـلی کـه در سلسله مراتب طبقه بندی بـالاتر اسـت، بـه ارث بـرده مـی شـود. همچنین کـاربر مـی توانـد بوسیله دانش کنترلی و فراتوابع نحوه توارث را تعریف نماید.آنچـه گفتـه شد زمـانی انجـام پـذیر اسـت کـه ناسـازگاری توسط اشیائی در یـک سلسـله مراتـب بوجـود آیـد. در غیـر اینصـورت ابتـدا بایـد ترتیـب تـوارث مشـخص شود. بعد از تشخیص توارث با تقدم بالاتر،اصلی که در این سلسله مراتب بالاتر و کلی تر است به ارث برده می شود.

4.11. زمان محاسبه توارث

FKRSهای مختلف توارث اسلاتها را در زمانهای متفاوت محاسبه مـی نماینـد. در سیسـتمهایی که توارث در هنگام واکشی اتفاق می افتد، توارث هنگامی اتفاق می افتد که برنامه کاربردی مقدار یک اسـلات را درخواسـت مـی کنـد. سیسـتم تـوارث، از سلسـله مراتـب تـوارث، بـرای یـافتن مقـداری جهـت ارث بری، بالا می رود. آنگاه مقدار ارث برده شده بـه برنامـه برگردانده مـی شـود، امـا بطـور فیزیکـی هیچگاه در فریم فرزنـد ذخیره نمـی گـردد. بـالعکس، در سیسـتمی کـه تـوارث در هنگـام اعـلان را محیـا می نماید، هنگامیکه فریم والد تعریف می شـود یـا کلاسبندی مـی گـردد، اطلاعـات اسـلاتهای ارث بـرده شده بطور فیزیکی در کلیه فریمهای فرزنـد کپـی مـی شـود. در بسیاری از سیسـتم هـا کـه از تـوارث در هنگام اعلان استفاده می نمایند، در صورتیکه کاربر بخواهد تعریف اسلات در فریم والد را تغییر دهد،

کاربر بایستی کل پایگاه دانش را دوباره بارگذاری نماید، زیرا سیستم توانایی به روزرسانی توسعه دهنده برای اطلاعات بدست آمده را ندارد.

در نهایت اینکه توارث در زمان واکشی و یا اعلان، دارای سرعت و نیازهای فضایی متفاوت و همچنین هریک دارای انعطاف پذیری متفاوت در زمان اجرا هستند. برای مطالب بیشتر در مورد مقایسه این دو روش می توان به مقاله ارائه شده توسط Schon مراجعه نمایید. با بکارگیری توارث در زمان اعلان، سیستم ازنیازبه جستجوی سلسله مراتب توارث در زمان واکشی رها خواهد شد اما در عوض ، هنگام تغییر تعاریف مربوط به کلاس، کاربر مجبور خواهد شد تا کل پایگاه دانش را دوباره بار نماید، در نتیجه گرچه با کاهش زمان اجرا در این روش، یک انعطاف پذیری حاصل شده است اما پیاده سازی این مشی هنگامیکه از یک پایگاه دانش خیلی بزرگ استفاده می نماییم امکان پذیر نخواهد بود. روشهای متعددی برای بالا بردن سرعت پیشنهاد شده است از جمله بالا بردن فضای ذخیره سازی و یا استفاده از روشهای HASHING در جاییکه امکان پیاده سازی آن وجو دارد، اما در هر یک از روشهای ارائه شده کاملا مشخص نیست که میزان افزایش سرعت در مقابل چه مقدار از فضای ذخیره سازی امکان پذیر است. در سیستم PARMENIDES از توارث در هنگام اعلان استفاده می شود، علاوه بر اینکه توارث دوباره در هنگام تغییرات در کلاسها محاسبه می شود. گرچه این روش باعث بالا رفتن انعطاف پذیری می شود اما باید توجه نمود که پیچیدگی و هزینه پیاده سازی چنین سیستمی بسیار بالاست. متاسفانه در عمل اطلاعات زیادی در مورد کارایی استفاده از توارث چندتایی و یا هزینه پیاده سازی آن وجود ندارد اما مطمئنا محاسبات مربوط به Subsumption را با پیچیدگی زیاد همراه خواهد نمود، همانگونه که در خانواده سیستمهای KL-ONE مشاهده شد. می توان ویژگیهای موجود در توارث چندگانه را به کمک قوانین تولید پیاده سازی نمود، اما آنچه سبب ارجحیت مکانیزم توارث می شود این است که کارایی چنین نوع خاصی از استنتاج، بهینه می باشد. استفاده از قوانین جهت پیاده سازی چنین مکانیزمی، همراه با استنتاج رو به عقب و جستجو در میان یک فضای بزرگ از قوانین است.

مطالب منتشر شده که در آن راه های انتخابی برای حل مشکل اطلاعات ارث برده شده ناسازگار تشریح شده اند، گزارشی از اینکه تا چه حد استنتاجهای تولید شده توسط متدهای ارائه شده در عمل قابل قبول است، ارائه نمی دهند. همچنانکه اطلاعی در مورد هزینه های محاسباتی ارائه نمی کنند.

14. برنامه نویسی شی گرا و Access Oriented در FKRS

گروهی از FKRSها حاوی ابزارهایی جهت برنامه نویسی شی گرا می باشند و گروهی دیگر، دست یابی گرا و یا هر دو را پشتیبانی می نمایند.

در گروهی از این سیستمها همچون UNIT، امکان الحاق توابع و یا متدهایی از LISP وجود دارد. کاربر می تواند یک متد را با فرستادن پیغام به فریم یا اسلات درخواست نماید. برای مثال می توان یک متد با نام if-deleted را به یک فریم اضافه نمود، با ساختن یک اسلات در ان فریم و نامگذاری آن به همین نام و ذخیره نمودن تعریف تابعی از LISP در آن اسلات. سپس این متد را می توان بوسله فرستادن پیغامی از طریق تابعی به نام UNITMSG درخواست نمود.

برنامه نویسی access-oriented در FKRS متاول تر ازOOP می باشد، و در سیستمهایی مانند خانواده UNITS، LOOM و OZONE استفاده شده است. امکان AOP به برنامه نویس اجازه می دهد تا حاشیه نویسی های(annotations) وابسته به برنامه را همراه داده ها نماید، به این ترتیب که این حاشیه نویسی ها بصورت خودکار هنگامی که کلاسهای متفاوتی از عملیات بر روی داده ها اعمال می شوند، اجرا می شوند(در FKRS این داده ها همان مقادیر اسلات ها می باشد). این حاشیه نویسی ها می توانند بصورت پویا به داده ها متصل و یا جدا شوند و وجود این حاشیه نویسی ها از دید برنامه یی که سعی نمی کنند مستقیما با آنها کار کنند پنهان است (برای مثال برنامه ای که تنها سعی در دستکاری داده ها دارد). معمولا این حاشیه نویسی ها پروسیجرهای LISP هستند اما در بعضی از سیستم ها این حاشیه نویسی ها می توانند قوانین PROLOG باشند.

یکی از رایجترین انواع این حاشیه نویسی ها توابعی هستند که در هنگام درخواست کاربر برای مقدار یک اسلات فراخوانی می شوند ؛ تابع مقدار مربوط به اسلات را محاسبه نموده و آن را به کاربر می دهد در حالیکه از دید کاربر مخفی است. نوع دیگر این حاشیه نویسی، توابعی هستند که هنگام تغییر مقادیر توسط کاربر فعال می شوند. این حاشیه نویسی ها ممکن است مقادیر اسلات ها را به روز رسانی نماید و یا حتی تغییراتی در یک بانک اطلاعاتی که مقدار اسلات ها به آن وابسته است ایجاد نماید.

به عنوان یک مثال، در FRAMEKIT کاربر این حاشیه نویسی روی اسلات ها را از طریق توابع LISP ذخیره شده در facetهایی به نام if-needed ، if-added ، if-erased و if-accessed انجام می دهد. If-needed در صورتی اجرا خواهد شد که کاربر سعی نماید مقدار نسبت داده شده به یک اسلات را بگیرد اما، هنوز آن اسلات مقداری نداشته باشد. در صورتیکه به اسلات مقداری نسبت

داده شده باشد آنگاه if-accessed به جای آن اجرا می شود. به همین صورت if-add در صورتی اجرا می شود که کاربر قصد داشته باشد مقدار جدیدی به اسلات اضافه نماید، و در صورتیکه کاربر قصد پاک نمودن مقدار یک اسلات را داشته باشد if-erased اجرا می شود.

گروهی بر این عقیده اند که استفاده از AOP، هزینه محاسباتی بالایی را به دنبال خواهد داشت. بیشتر این هزینه هنگام جستجو برای پیدا کردن این حاشیه نویسی ها، در سلسله مراتب جستجو می باشد. برای جلوگیری از تکرار این جستجو روشهایی پیشنهاد شده است از جمله ذخیره یکسری اطلاعات جهانی برای مشخص نمودن اینگونه اسلاتها.

15. کلاس بندی

به طور شهودی کلاس یا مفهوم A ، شامل (subsumes) کلاس یا مفهوم B می شود، در صورتیکه مفهومی که A ارائه می دهد کلی تر از آن چیزی باشد که توسط B ارائه می گردد. برای مثال کلاس مردها ، کلاس پدر را شامل می شود چرا که همه پدرها ، مرد هستند. Subsumption را می توان بصورت دقیق تر بوسیله ترمهای مجموعه ای بیان نمود. KL_ONE بصورت خودکار بررسی می نماید که آیا یک کلاس شامل کلاس دیگری می شود یا نه، که این کار بر پایه تعاریف ارائه شده برای اسلاتهای آن کلاس صورت می گیرد. محاسبات مربوط به Subsumption اساس کار کلاس بندی می باشد. عملیات فریم A را در محل مناسب در سلسله مراتب طبقه بندی شده قرار می دهد.

تعریف یک مفهوم همانند C شامل دو قسمت است: لیستی از مفاهیم که کلی تر از مفهوم C هستند، و لیستی از شرایط که C را از مفهوم جد خود، متمایز می نماید. این شرایط، متغیری از محدودیتهای اعمال شده بر مقدار اسلات هستند. برای مثال در KL-ONE تعریف Father به عنوان یک زیر مجموعه پوشا از Man است که در آن، حداقل یک ارزش به Child نسبت داده شده باشد.

FKRSهایی که از کلاسه بندی استفاده می کنند معمولا terminological reasoners (استدلال کننده های وابسته به اصطلاحات) نامیده می شوند، زیرا از روابط میان مجموعه ای ازتعاریف ترم ها – کلاس یا مفهوم، نگهداری می نمایند.

با اینکه عده زیادی از محققین کلاسبندی را به عنوان محوری ترین عملیات در FKRSها می دانند، نقش کلاسه بندی هنوز واضح نیست. گروهی بر این باورند که کلاس بندی سبب کاهش زمان مورد نیاز وهزینه های نگهداری پایگاه های دانش بزرگ می شود اما در حقیقت هیچ نوع مطالعات

سیستماتیک در این مورد موجود نیست. علاوه بر این سیستمهای صنعتی بزرگی وجود دارند که بر پایه FKRS گسترش یافته اند اما ازکلاس بندی استفاده نمی نمایند. همگی نمونه های تجاری از FKRS ها که از نمونه خانواده های UNIT، یا SLR ویا FRL هستند، از کلاس بندی استفاده نمی نمایند. موفقیت سیستمهایی که از کلاس بندی استفاده نمی کنند به صورت تجربی اثبات نموده است که کلاس بندی یکی از شرایط لازم یا کافی برای موفقیت FKRS نمی باشد.

استفاده ازکلاس بندی در کاربردهای مختلف متفاوت است. مثلا در برنامه های کاربردی زبانهای طبیعی بیشتر جهت اداره پایگاه دانش و پاسخ به پرسشها از کلاس بندی استفاده می شود حال آنکه در یک برنامه تشخیص طبی بیشتر برای حل مسئله و نگهداری از پایگاه دانش از آن استفاده می گردد، حال آنکه یک تحلیل گر کیفی تنها برای نگهداری پایگاه دانش از آن استفاده می نماید.