

Глава 4. ЛИНГВИСТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САПР

Содержание

4.1. Структура лингвистического обеспечения САПР	1
4.2. Языки программирования	2
4.2.1. Машинно–ориентированные языки.....	3
4.2.2. Алгоритмические языки высокого уровня.....	4
4.3. Языки проектирования	6
4.4. Структура трансляторов	8
4.5. Языки взаимодействия в САПР	11
4.6. Языки представления знаний	12

4.1. Структура лингвистического обеспечения САПР

Под *лингвистическим обеспечением САПР* понимают языковые средства общения пользователя с САПР и языки программирования, с помощью которых создается информационное и прикладное программное обеспечение.

Оно включает в себя языки для представления информации о проектируемых объектах, процессе и средствах проектирования.

Лингвистическое обеспечение включает в себя различные языковые средства, которые делятся на три группы:

- языки программирования;
- языки проектирования;
- языки взаимодействия (общения).

Под термином “*язык*” в данном случае понимается любое средство общения, любая система символов или знаков, используемых для обмена информацией. Языки программирования служат для записи программ. Ими пользуются главным образом при подготовке программ, а не при эксплуатации САПР.

Языки программирования используются для написания программ и применяются главным образом разработчиками САПР. *Языки проектирования* служат для описания исходной информации об объектах и задачах проектирования при выполнении проектных процедур с помощью программного обеспечения и являются средством общения пользователя САПР с ЭВМ. Особую группу составляют языки описания управляющей информации для программно–управляемого технологического оборудования (для фотонаборных установок, графопостроителей, металлообрабатывающих станков с ЧПУ и т.п.), называемые *языками управления*.

Языки могут быть *процедурными* и *непроцедурными*. *Процедурные языки* применяются для описания процессов в виде последовательностей действий и процедур. В частности, большинство языков программирования служит для описания вычислительных процессов и потому относится к процедурным языкам. Другой пример процедурного языка – язык имитационного моделирования GPSS. *Непроцедурные языки* применяются для описания семантических сетей, структур проектируемых объектов и других статических систем.

4.2. Языки программирования

С момента начала использования ЭВМ для расчетов и проектирования возникла проблема общения человека с машиной. Первоначально программа для ЭВМ готовилась в машинных кодах. Такие машинные программы могли разрабатываться только узкими специалистами – программистами, знающими устройство и особенности конкретной ЭВМ. Инженер–пользователь для выполнения расчетов на ЭВМ в своей проблемной области должен был обращаться к программисту. В этом случае имела место цепочка:

пользователь – программист – машинная программа – ЭВМ.

Такая цепочка приводила к большим затратам трудовых ресурсов и времени. Программирование задач на машинном языке ограничивало использование ЭВМ.

Эту проблему позволило решить создание алгоритмических языков высокого уровня, отличающихся универсальностью. Для того чтобы машина понимала языки высокого уровня, для нее необходим переводчик с этого языка на машинный. Таким переводчиком является *транслятор*, т. е. программа, которая преобразует программу, написанную на языке высокого уровня, в машинную программу. Таким образом, возникает следующая цепочка:

пользователь – программа на языке высокого уровня – транслятор – машинная программа–ЭВМ.

Эта цепочка сделала революцию в применении ЭВМ. Число активных пользователей ЭВМ стало быстро расти, что в свою очередь послужило толчком к еще более быстрому развитию ЭВМ.

Таким образом, среди языков программирования различают *машинно–ориентированные*, называемые языками ассемблера и автокодами, и *алгоритмические языки высокого уровня*.

Автокод – язык, предложения которого по структуре подобны машинным командам. *Язык ассемблера* – автокод, расширенный макрокомандами, выражениями, средствами, обеспечивающими модульность программ.

Алгоритмические языки высокого уровня не зависят от типа ЭВМ. Основной смысл алгоритмических языков состоит в том, что их символика и логика (правила записи) близки к принятым в математике и в естественном русском и английском языках. Вместе с тем эта символика и правила строго однозначны и могут автоматически (формализованно) переводиться в команды машины. *Алгоритмический язык*, согласно определению ГОСТа 19781–74, есть набор символов и система правил образования и истолкования конструкций из этих символов для задания алгоритмов.

Алгоритмический язык для записи программ и данных называют *языком программирования*. В качестве языков программирования в САПР находят применение машинно–ориентированные языки типа Ассемблера и алгоритмические языки высокого уровня.

4.2.1.Машинно–ориентированные языки

Эти языки позволяют сокращать время составления программы за счет использования более удобных средств описания алгоритмов, передавать ЭВМ работу по распределению оперативной и долговременной памяти, обнаруживать самой ЭВМ ошибки кодирования и выполнения программ и выдавать в удобной для программиста форме информацию. К таким языкам программирования относятся язык Ассемблер (для всех классов ЭВМ) и др. Языки уровня Ассемблера соответствуют системам команд конкретных ЭВМ и позволяют составлять программы в форме, удобной для человека.

В основе машинно–ориентированных языков лежит символическая адресация и кодирование, а это шаг в направлении автоматизации программирования. В то же время эти языки программирования имеют существенный недостаток машинных языков – они сильно отличаются от традиционных математических языков. Поэтому время составления программ остается значительным, сами программы громоздки.

В последнее время языки Ассемблера нашли широкое применение при программировании на мини–, микро– и персональных ЭВМ из–за необходимости написания эффективных программ, например, программ, входящих в состав графического редактора.

Однако языки типа Ассемблера отличаются большей универсальностью, выражающейся в широких возможностях описания кодов различных форматов, логических операций и процедур. В то же время при использовании этих языков требуются меньшие затраты машинного времени и памяти. Так, например, для транслятора с языка ПАСКАЛЬ требуется увеличение затрат машинного времени в 1,5...2,5 раза по сравнению с машинно–ориентированным языком типа Ассемблера.

Учитывая положительные особенности машинно–ориентированных и алгоритмических языков высокого уровня, их можно применять одновременно при разработке САПР для решения различных задач. При этом язык Ассемблера используют:

при разработке модулей с большим количеством логических операций и операций над отдельными группами разрядов машинных слов, так как в этой ситуации возможности алгоритмических языков высокого уровня недостаточны;

при жестких требованиях к модулю по показателям затрат машинных времени и памяти.

В остальных случаях определяющими требованиями становятся производительность труда программистов и инвариантности к типам ЭВМ, что обуславливает применение языков высокого уровня.

Использование машинно–ориентированных языков позволяет достигать наивысшей эффективности объектных программ с точки зрения затрат вычислительных ресурсов – машинных времени и памяти. Эти языки универсальны в смысле применимости к решению задач различных классов – научно–технических и экономических, системных и прикладных.

Однако программирование на этих языках требует высокой квалификации программиста и приводит к увеличению сроков разработки прикладного программного обеспечения. Главный недостаток этих языков – непереносимость программ на ЭВМ с системой команд, отличной от той, на которую ориентирован язык.

4.2.2. Алгоритмические языки высокого уровня

Алгоритмические языки высокого уровня в сравнении с машинно–ориентированными языками удобнее для реализации алгоритмов численного анализа, легче осваиваются инженерами, позволяют повысить производительность труда программистов при разработке программ и их адаптации к различным типам ЭВМ. Алгоритмические языки высокого уровня – основное средство разработки прикладного программного обеспечения. В САПР наибольшее распространение получили языки ФОРТРАН, ПЛ/1, ПАСКАЛЬ, СИ, АДА, ЛИСП, ПРОЛОГ, БЭЙСИК.

Выделяется два подкласса языков программирования: проблемно–ориентированные и процедурно–ориентированные.

Проблемно–ориентированные языки программирования направлены на решение узкого класса задач. Программирование ведется в понятиях, характерных для конкретной проблемной области. Данный подкласс языков программирования тесно связан с языками общения в САПР.

Процедурно–ориентированные языки программирования созданы для описания алгоритмов решения задач. Среди них можно выделить машинно–зависимые языки программирования высокого уровня и алгоритмические.

Машинно–зависимые языки программирования высокого уровня характеризуются максимальным использованием аппаратуры микроЭВМ и удобством записи и компоновки программ. Эти языки в меньшей степени зависят от конкретной ЭВМ, чем языки Ассемблер, хотя набор разрешенных оператором выражений находится в прямом соответствии с набором команд и методов адресации аппаратуры. Данный подкласс принадлежит к классу языков программирования ПЛ/1 и нашел самое широкое распространение при программировании микропроцессорных наборов. К машинно–зависимым языкам программирования относятся SMAL/80 (с системой команд ЭВМ класса INTEL8080) и МИСТРАЛ (Motorola–6800), PL/M, MPL, PLZ–SYS.

Алгоритмические языки программирования являются машинно–независимыми языками. К данному подклассу языков можно отнести ФОРТРАН и его модификации, язык СИ, ПАСКАЛЬ, БЕЙСИК, АДА, ЛИСП. Достоинства перечисленных языков программирования:

- высокая производительность труда программиста,
- самодокументируемость программ,
- простота эксплуатации,
- возможность переноса программ с одной ЭВМ на другую (свойство

мобильности),

наличие средств контроля и отладки.

Рассмотрим некоторые особенности использования алгоритмических языков программирования. Если создаваемая САПР предназначена для решения инженерных задач, описываемых действительными числами, то лучше воспользоваться языками программирования СИ, ФОРТРАН; для обработки экономической информации – КОБОЛ либо СУБД, для обработки геометрических моделей – ЛИСП.

Для выбора языка программирования можно воспользоваться и другими факторами. Например, если несколько языков программирования удовлетворяют тематике задач, то предпочтение отдается языку с более быстродействующим транслятором или снабженному лучшей системой отладки, или дающему более эффективные программы. Под параметрами эффективности подразумевается быстродействие, объем используемой оперативной памяти, количество требуемых операторов для обработки литературного и вычислительного текстов. Исходя из этих критериев было выявлено, что СИ с соответствующим транслятором обеспечивает самые короткие объектные модули, на последнем месте ПАСКАЛЬ (программы на данном языке программирования в 3,25 раза длиннее программ на СИ).

Может так оказаться, что ни один из известных или доступных программисту языков не удовлетворяет его практическим потребностям. В этом случае возможна разработка собственного языка программирования, максимально учитывающего специфику структур данных и алгоритмов определенного класса задач, и его реализации.

В настоящее время разрабатываются языки четвертого уровня, включающие в себя средства для автоматического выбора метода решения той или иной задачи с помощью ЭВМ. Такие языки программирования позволят описать формулировку задачи и некоторые требования к методу решения. Дальнейшая работа по подготовке задачи и ее решению на ЭВМ будет выполняться автоматически. В основе таких языков лежат программы автоматического выбора методов решения задач.

Алгоритмический язык ФОРТРАН (FORmula TRANslation – переводчик формул) предназначен для научных и инженерных задач, решаемых на ЭВМ. Этот язык разработан в 1956 г. Язык программирования ФОРТРАН стал первым языком, в котором материализовалось понятие “модульность”. Хорошо разработанные библиотеки стандартных подпрограмм являются ярким примером преимущества модульного принципа программирования. Но этот язык программирования не лишен многочисленных недостатков: в нем нет средств для удобного описания разнообразных структур данных, запрещены рекурсивные обращения к процедурам, нет строгого описания языка. Несмотря на недостатки, ФОРТРАН широко использовался в САПР, особенно разработанных в 60–70-е годы, благодаря простоте разработки эффективных трансляторов. В настоящее время применяется усовершенствованная версия языка – ФОРТРАН-77.

Вторым языком, получающим все большее распространение в САПР, является алгоритмический язык PL/1 (Programming Language/1 – язык программирования – 1), разработанный в 1966 г. В отличие от языка ФОРТРАН, этот язык имеет более широкие возможности при обработке больших массивов информации и описании структур исходных данных. Этот язык ориентирован на крупные модели ЭВМ.

В начале для малых и микро-ЭВМ получил большое распространение язык БЕЙСИК, разработанный в 1965 г. Этот язык очень похож на ФОРТРАН, только значительно проще. Особенно большие возможности БЕЙСИКА проявляются при решении задач в режиме диалога с ЭВМ. Простота и доступность этого языка позволяют быстро осваивать его начинающим пользователям САПР.

Язык ПАСКАЛЬ является претендентом на роль основного языка для написания прикладного программного обеспечения. Положительные свойства этих языков – развитые средства для написания хорошо структурированных программ, для представления различных типов и структур данных, удачное сочетание простоты и строгости в описании языков.

Язык СИ является другим претендентом на роль основного языка программирования в САПР. Он сочетает черты языков высокого уровня и языков ассемблера, что делает удобным его применение при разработке системного программного обеспечения (ПО). Язык СИ остается машинно-независимым и, следовательно, обеспечивает создание мобильных (переносимых) программ.

Язык АДА можно назвать наиболее универсальным среди созданных языков. В этот язык включены средства для описания параллельных процессов. Однако трансляторы с этого языка пока не получили достаточного распространения.

4.3. Языки проектирования

Для обеспечения процесса проектирования объектов в САПР используются следующие виды языков проектирования:

- входной язык проектирования;
- выходной язык проектирования;
- сопровождения,
- промежуточные,
- внутренние.

Входной язык – язык проектирования, предназначенный для представления задания на проектирование, т.е. информации об объектах и задачах проектирования, передаваемой от человека к ЭВМ. В этом языке для задания исходной информации в САПР должны быть предусмотрены средства описания объектов проектирования в форме, удобной для отображения и ввода в ЭВМ. В большинстве входных языков САПР можно выделить две части: непроцедурную, служащую для описания структур объектов, и процедурную, предназначенную

для описания заданий на выполнение определенных проектных операций и процедур. Языковые средства в этих двух частях составляют соответственно язык описания объекта и язык описания заданий. Среди языков описания объекта различают языки описания схем, чертежей, процессов функционирования. Названия этих разновидностей язык описания объекта соответственно схемные, графические, моделирования.

Выходной язык – язык проектирования, предназначенный для представления какого-либо проектного решения, включая результат проектирования в форме, удовлетворяющей требованиям его дальнейшего применения. В состав этого вида языков входят различные средства описания результатов проектирования в виде чертежей, технических карт, схем наладок, таблиц, текстовой документации, а также представление формы промежуточных результатов проектирования, используемых в различных подсистемах САПР.

Языки сопровождения применяются для корректировки и редактирования данных при выполнении проектных процедур. В диалоговых режимах работы с ЭВМ средства языков входного, выходного и сопровождения тесно связаны и объединяются под названием диалогового языка.

Промежуточные языки используются для описания информации о задачах проектирования на определенной стадии трансляции. Введение единого для программно-методического комплекса промежуточного языка облегчает адаптацию комплекса к новым входным языкам, т. е. делает комплекс открытым по отношению к новым составляющим лингвистического обеспечения.

Внутренние языки являются языками внутреннего представления данных. Введение единого внутреннего представления данных означает принятие определенных соглашений об интерфейсах отдельных программ в программно-методическом комплексе и делает программно-методический комплекс открытым по отношению к новым элементам программного обеспечения.

Разрабатываемые или применяемые при создании САПР *языки проектирования* и, в первую очередь, входные языки, должны отвечать следующим *требованиям*:

универсальности – возможности описания на входном языке любых объектов проблемной области, на которую ориентирована САПР;

удобству пользования – язык должен иметь проблемную ориентацию, обеспечивая пользователю максимальные удобства для описания и восприятия используемых при проектировании данных;

максимальной лаконичности описания;

однозначности истолкования элементов и конструкций языка;

возможности развития и расширения языка;

совместимости с другими входными и выходными языками.

Входной язык для технологического проектирования. Входной язык для

технологического проектирования, используемый в САПР технологических процессов, предназначен для описания информации о предметах и процессах технологического проектирования. При этом основным объектом описания является информация о детали. Среди различных видов информации, описываемой с помощью входного языка для технологического проектирования в САПР ТП, наибольшую сложность представляют формализация и описание геометрических образцов детали.

Первоначально проблема создания входных языков для описания геометрической информации, и прежде всего конструкторских чертежей, возникла при кодировании информации при разработке управляющих программ для станков с ЧПУ. В качестве таких языков, широко используемых в настоящее время, могут быть названы АРТ (США), ЕКАРТ (ФРГ), САП-2, САПП, ТЕХТРАН и ряд других, разработанных в СССР. К наиболее развитым языкам описания объектов, рассматриваемых при решении задач технической подготовки производства в машиностроении, относится язык ОГРА – язык “Описания ГРАфики”.

Состав и особенности построения входного языка для технологического проектирования регламентированы ГОСТами 14.417–81. ЕСТПП. “Проектирование автоматизированное. Входной язык для технологического проектирования. Язык описания детали”.

4.4. Структура трансляторов

Исполнение на ЭВМ заданий, представленных на каком-либо языке, отличном от машинного, требует интерпретации или предварительной трансляции исходной информации. Эти преобразования информации осуществляются программами или техническими устройствами – *интерпретаторами* или *трансляторами*. Объединяющее название для интерпретаторов и трансляторов – *языковые процессоры*.

Интерпретатор поочередно анализирует и исполняет указания, выраженные предложениями входного языка. В оперативной памяти ЭВМ при решении задачи присутствуют прикладная программа на входном языке и интерпретатор.

Транслятор преобразует заданную информацию с одного языка на другой. Программа на входе транслятора и ее язык называются *исходными*, на выходе транслятора – *объектными*. Если объектный язык – машинный или близкий к машинному, то трансляция и транслятор называются компиляцией и *компилятором* соответственно. Если исходный язык – язык ассемблера, то транслятор на машинный язык – *ассемблер*. Если исходный и объектный языки относятся к одному и тому же уровню языков, то транслятор называют *конвертором*. Решение задач по методу компиляции происходит в два этапа. Сначала в оперативной памяти размещаются исходная программа и компилятор, результатом работы компилятора будет рабочая программа. Затем скомпилированная рабочая программа исполняется.

При интерпретации циклических участков, неизбежно присутствующих в реальных программах, приходится многократно обрабатывать одни и те же предложения исходного

текста, что обуславливает повышенные затраты машинного времени по сравнению с затратами при трансляции. Но в большинстве случаев интерпретация экономичнее по затратам памяти, так как не расходует память под скомпилированную программу.

Типичные функции трансляторов – контроль правильности исходной информации, генерация текста объектной программы. Процесс трансляции состоит из нескольких этапов, называемых *фазами трансляции*. Основные этапы – лексический и синтаксический анализ, генерация кода.

Лексический анализ, называемый также сканированием, служит для разделения исходного текста на отдельные элементарные языковые единицы – лексемы, в качестве которых фигурируют идентификаторы, числа, метки, знаки операций и т.п. Составляются таблицы лексем, используемые при дальнейшей трансляции. Выявляются недопустимые сочетания символов языка, которые нельзя выделить как лексемы (например, идентификатор, начинающийся с цифры, неразрешенный символ и т. п.).

Синтаксический анализ (грамматический разбор) – фаза, на которой проверяется соблюдение синтаксиса языка, т.е. проверяется правильность построения предложений. В процессе анализа должны выявляться все ошибки в исходном описании, которые можно обнаружить по формальным признакам, и выдаваться пользователю соответствующие диагностические сообщения. Результат синтаксического анализа – представление информации на промежуточном языке.

Генерация кода осуществляется генератором кода, который использует данные синтаксического анализа для построения объектной программы.

Одна или несколько фаз трансляции, заканчивающиеся формированием файла, направляемого во внешнюю память, называются *проходом*.

Однопроходные трансляторы наиболее экономичны по затратам времени. В них входная информация порциями проходит все фазы трансляции, так что к моменту обработки новой порции уже закончена генерация кода для предыдущей порции. Это ограничивает возможности генерирования высокоэффективных рабочих программ.

Оптимизирующие трансляторы являются многопроходными, в них на очередном проходе могут быть использованы особенности всей программы, просмотренной на предыдущем проходе, для принятия мер по повышению эффективности генерируемой программы. Такими особенностями могут быть наличие повторяющихся участков программ, которые можно не дублировать, повторяющихся вычислений, которые можно выполнить однократно и т. п. В двухпроходных трансляторах обычно за первый проход выполняются лексический и синтаксический анализы, за второй – генерация кода. В трехпроходных трансляторах фазы лексического и синтаксического анализа составляют разные проходы.

В некоторых программно–методических комплексах, способных адаптироваться к новым условиям применения, предусматривается единый промежуточный язык на который

могут транслироваться описания задач с нескольких входных языков. Промежуточный язык должен быть по своему синтаксису достаточно близким к возможным входным языкам. Соблюдение этого условия обуславливает сравнительную простоту трансляции со входных языков на промежуточный.

Наиболее сложная фаза при трансляции – синтаксический анализ. Математическим аппаратом, используемым при построении синтаксических анализаторов, является аппарат *формальных грамматик*. Аппарат формальных грамматик близок к теории автоматов и искусственного интеллекта. Известны попытки его использования для разработки алгоритмов структурного синтеза.

Грамматика – система правил, задающая множество правильных цепочек из символов языка, т.е. грамматика – конечное множество правил, определяющих язык. *Синтаксис* – часть грамматики, определяющая построение предложений из слов. *Семантика* – толкование синтаксических правил.

В САПР применяются порождающие формальные грамматики. Порождающая грамматика – это четверка объектов $\Gamma = \langle V_T, V_N, P, S \rangle$, где V_T, V_N – множества соответственно терминальных и нетерминальных символов; P – правила грамматики, называемые продукциями; S – начальный нетерминальный символ. Множество V_T включает все слова языка, из которых строятся предложения. Множество V_N включает символы, которые играют вспомогательную роль; нетерминальные символы используются при записи синтаксических правил и обозначают классы терминальных символов. Словарь грамматики V объединяет множества V_T и V_N . Продукции можно представить как преобразования $\phi \rightarrow \Psi$, называемые также *правилами подстановки* и читаемые "заменить ϕ на Ψ ", где ϕ, Ψ – некоторые цепочки слов из V . В ϕ должен быть хотя бы один нетерминальный символ.

Последовательность $S = \phi_0, \phi_1, \phi_2, \dots, \phi_n$ называется выводом цепочки ϕ_n , если в указанной последовательности каждая цепочка ϕ_{i+1} получается из ϕ_i по правилам подстановки, здесь n – длина вывода. Множество цепочек, выводимых в грамматике Γ , называется *языком*, порожденным грамматикой Γ .

Правила подстановки, как и в других системах продукций, не являются предписаниями, а представляют собой разрешения, поэтому порождающая грамматика не может рассматриваться как алгоритм.

Синтаксический анализатор в соответствии с описанием языка имеет в своем составе подпрограммы на каждое правило подстановки. Кроме проверки соответствия исходной программы синтаксическим правилам синтаксический анализатор осуществляет некоторые преобразования, облегчающие последующую генерацию объектной программы. При трансляции с процедурных алгоритмических языков целесообразно преобразование выражений из инфиксной формы в постфиксную. Понятие об этих формах дается следующим примером: $A+B*C$ – привычная для исходных программ инфиксная форма; $ABC*+$ – постфиксная форма.

Проще будет осуществлена реализация выражения в постфиксной форме, а также генерация объектного кода.

4.5. Языки взаимодействия в САПР

На современном этапе разработки и использования САПР обозначилось положение, когда эффективность любой САПР характеризуется тремя основными параметрами:

- степенью удобства языка взаимодействия пользователя с САПР,
- временем ожидания решения поставленной задачи,
- удобством формы представления результатов проектирования.

Подробно остановимся на анализе только первого параметра. Степень удобства характеризует затраты человеческих ресурсов на подготовку исходных данных для проектирования того или иного объекта. Язык взаимодействия является средством, гарантирующим максимально удобный контакт пользователя с САПР и упорядочивающим процесс проектирования. Форма представления исходных данных на языке взаимодействия должна:

- сокращать или устранять полностью необходимость каких-либо знаний программирования;
- приближаться к естественной форме описания объекта в выражениях, максимально понятных пользователю;
- способствовать сокращению времени подготовки исходных данных, одновременно уменьшая возможность внесения ошибок;
- обладать гибкостью, некоторой универсальностью, легкостью изучения и простотой кодирования;
- сохранять способность к дальнейшему развитию и совершенствованию.

Анализируя приведенные требования, необходимо помнить, что любой язык представляет собой знаковую систему, описание которой задается алфавитом и синтаксисом, с помощью которых представляются исходные данные. Синтаксис определяет набор правил объединения слов в предложения. Языки взаимодействия представляются либо в жестком (позиционно-зависимые или контекстно-связанные), либо в свободном формате с произвольным расположением слов в предложении (позиционно-независимые или контекстно-свободные).

Язык взаимодействия является совокупностью трех взаимосвязанных языков:

- языка описания объекта, предназначенного для подготовки данных о проектируемом объекте;
- языка описания заданий, предназначенного для задания последовательности действий, составляющих выбранный маршрут проектирования объекта;
- языка отображения информации, предназначенного для формирования и

отображения (на АЦПУ, графопостроителе, экране буквенно–цифрового или графического дисплея) входных, выходных и промежуточных протоколов. Под протоколом понимают ту или иную форму представления данных, требуемых в процессе и после окончания проектирования. Например, входной протокол содержит данные об объекте проектирования, указания об ошибках и т. д.

По своему характеру языки взаимодействия делятся на пассивные и активные. Под *пассивным языком взаимодействия* понимают язык, используя который пользователю приходится описать объект проектирования, задать последовательность выполнения заданий и форму отображения информации сразу при подготовке одного этапа или нескольких этапов проектирования и ожидать результатов после выполнения последнего задания без оперативного вмешательства для исправления ошибок или изменения последовательности выполнения заданий. Такой режим взаимодействия пользователя с САПР называется пакетным. Пассивные языки взаимодействия были обусловлены низким уровнем развития технических средств и системного программного обеспечения.

После появления дисплеев, интерактивно–графических станций, специального программного обеспечения для поддержания интерактивного режима работы системы человек–компьютер стали разрабатывать языки взаимодействия на основе диалога. Диалог между человеком и компьютером подобен диалогу между двумя людьми, только более проблемно–ориентированный. Диалог можно организовать в следующих формах: программируемого вопросника; меню; вопросов и ответов типа “да – нет”; свободного общения. Главные достоинства диалога состоят в том, что пользователь быстро обучается языку взаимодействия и может вмешаться в любой момент в процесс проектирования с целью его окончания или изменения тех или иных исходных данных.

4.6. Языки представления знаний

Для описания декларативных знаний используются *ролевые, реляционные и логические языки*. *Декларативные знания* представляют собой сведения о предметной области в виде совокупности понятий и отношений между ними. Различия между языками заключаются в форме представления этих знаний.

Ролевые языки порождаются фреймовыми представлениями, *реляционные* описывают семантические сети, а *логические* основаны на исчислении предикатов первого порядка.

Фрейм – структура данных, в которой в определенном порядке представлены сведения о свойствах объекта. Типичный вид фрейма

$$\langle \text{имя фрейма}; a_1=p_1; a_2=p_2; \dots; a_n=p_n; q_1, q_2, \dots, q_n \rangle,$$

где a_i – имя i -го атрибута, p_i – его значение, q_i – ссылка на некоторый другой фрейм или процедуру.

Семантическая сеть – форма представления знаний в виде совокупности понятий и

отношений между ними в некоторой предметной области. Семантическую сеть удобно изображать графом, в котором вершины отображают понятия, а ребра – отношения между ними. Семантические сети можно использовать для представления структур проектируемых объектов, если вершинам поставить в соответствие элементы, а ребрам – соединения элементов или другие взаимодействия. В качестве вершин могут фигурировать фреймы, представляющие сведения об элементах.

Элементами *логического языка* являются символы для представления отношений (предикатные), переменных, констант, функций, а также скобки, запятые и другие вспомогательные символы. Формулы, состоящие из этих элементов, являются высказываниями и называются *литералами*. Пример литерала: "дисплей есть часть вычислительного комплекса", где "есть часть" – предикатный символ, выражающий отношение "часть–целое" между понятиями (переменными) "дисплей" и "вычислительный комплекс". Литералы связываются в логические формулы с помощью функциональных символов \vee (ИЛИ), $\&$ (И), \Rightarrow (импликация), \sim (НЕ), и кванторов существования \exists и общности \forall . С помощью логических формул можно задать некоторую базу данных, пополнять ее, строить запросы на поиск в ней нужных элементов, в том числе осуществлять поиск по образцу. С помощью эквивалентных преобразований логических формул можно выполнять операции по обработке информации, представляющей знания, реализуя тем самым функции, присущие системам искусственного интеллекта.

Для пользователей наиболее естественны и удобны ролевые языки, наименее удобны – логические.

Для представления процедурных знаний применяются *языки productions*.

Использование большинства алгоритмических языков общего назначения в качестве языков представления знаний возможно, но мало удобно. Целесообразна разработка специальных языков. Одним из первых языков представления знаний стал язык ЛИСП (создан в 1959 г.). В настоящее время в системах искусственного интеллекта широко распространен язык ПРОЛОГ.

Описание задачи на языке ПРОЛОГ включает: данные о предметной области в виде совокупности фактов и правил; формулировку задания. *Фактом* называется простое утверждение, выражающее отношение, применяемое к одному или нескольким понятиям (атрибутам), например *ТИП (ЭВМ, IBM PC)*, где *ТИП* – отношение классификации ("род–вид") между атрибутами *ЭВМ* и *IBM PC*; *ЕСТЬ (IBM PC)* – этот факт интерпретируется как наличие объекта *IBM PC*, т.е. объект *IBM PC* обладает свойством "есть в наличии". *Правило* (редуктор) состоит из заголовка и тела. Справедливость утверждения в заголовке правила определяется справедливостью фактов, перечисленных в его теле. *Формулировка задания* – это один или несколько вопросов или команд.