Интеллектуализация ЭВМ

Решение задач на вычислительных моделях (работы Энна Харальдовича Тыугу – 1970-1985 гг.).
Реальная система синтеза программ – Инструментальная система программирования "**ПРИЗ**" ЕС ЭВМ.
Идея подхода. Синтез программ – составление алгоритма решения задачи на ***вычислительной модели*** предметной области (ПО).

Такая модель строится заранее. Она состоит из совокупности ***переменных***, соответствующих объектам ПО, и ***отношений вычислимости***.

Каждое отношение связывает значения нескольких переменных и интерпретируется так:
по известным значениям некоторых переменных (***входных переменных***) отношения можно вычислить значения других (***выходных переменных***).

Примеры объектов и отношений для ПО – "ГЕОМЕТРИЯ":

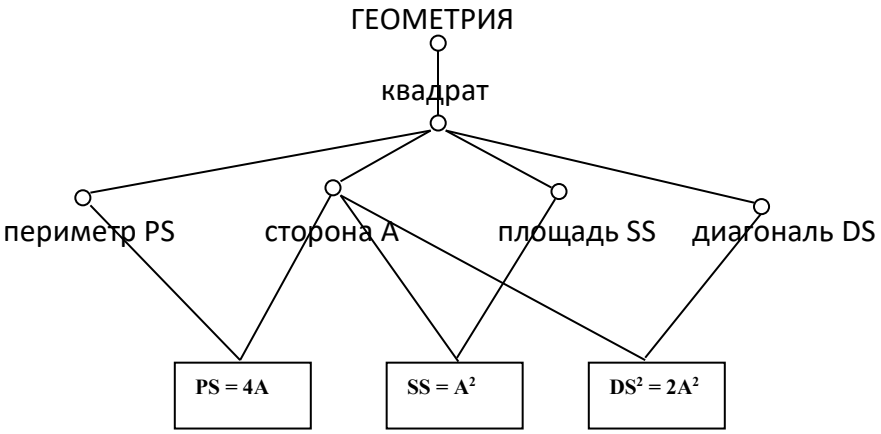
Объект КРУГ (C - Circle):

радиус (R)
диаметр (DC = 2 * R)
периметр (PC = 2 * pi * R)
площадь (SC = pi * R ↑ 2)

Объект КВАДРАТ (S-Squere):

сторона (A)
диагональ (DS = sqr(2) * A)
периметр (PS = 4 * A)
площадь (SS = A ↑ 2)

Схема отношения вычислимости, связывающего основные характеристики квадрата как объекта ПО – "ГЕОМЕТРИЯ":



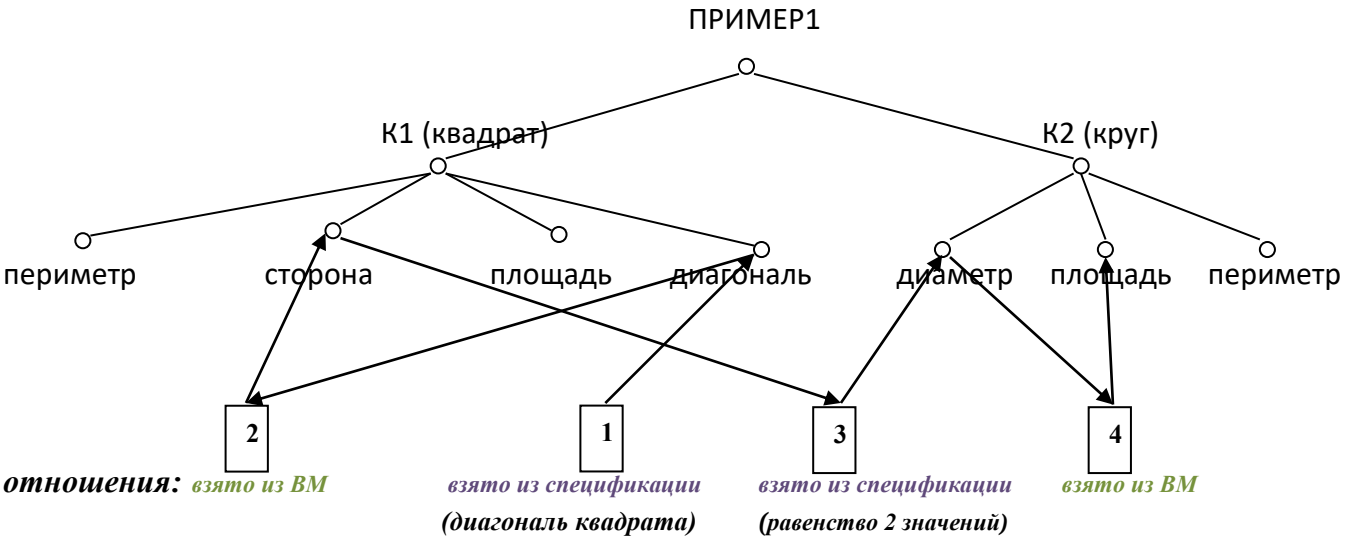
Задача на вычислительной модели – тройка (*V, U, M*), где *V* – множество входных переменных, *U* – множество выходных переменных, *M* – вычислительная модель.

Задача описывается на специальном входном языке УТОПИСТ (системы ПРИЗ).

Пример: описание задачи вычисления площади круга, вписанного в квадрат с диагональю 5.

```
ПРОГРАММА' ПРИМЕР1;
        ПО' ГЕОМ;
        ПУСТЬ' К1: КВАДРАТ ДИАГОНАЛЬ = 5;
                К2: КРУГ ДИАМЕТР = К1.СТОРОНА;
        ДЕЙСТВИЯ'
        НА ПРИМЕР1' ВЫЧИСЛИТЬ' К2.ПЛОЩАДЬ;
КОНЕЦ' +++
```

Схема вычислительной модели этой задачи:



Такая модель (во внутреннем представлении) формируется на первом этапе синтеза. Далее строится алгоритм решения задачи (по оператору задачи: НА ПРИМЕР1' ВЫЧИСЛИТЬ' К2.ПЛОЩАДЬ) и модели. Затем построенный алгоритм записывается на одном из выходных языков системы ПРИЗ.

В общем случае схема диалога с системой ПРИЗ такова (символ ? – признак факультативности строки):

```
        ПРИЗ К ВАШИМ УСЛУГАМ
ПРОГРАММА' <имя>;
        ?ПО' <предметная область>;
        ПУСТЬ' <входные данные>;
        ?ВВЕСТИ' <входные данные>;
        ДЕЙСТВИЯ'
        НА <имя> ВЫЧИСЛИТЬ <результат>;
КОНЕЦ' +++
        КОНЕЦ ОПИСАНИЯ ЗАДАЧИ
        ?<сообщения об ошибках>
        ВЫВЕСТИ АЛГОРИТМ (ДА, НЕТ)
        ?<сообщения об ошибках>
        ПРИЗ ЗАКОНЧИЛ РАБОТУ
        НУЖЕН ЛИ ПУСК? (ДА, НЕТ)
        ?<сообщения об ошибках во время работы ассемблера>
        ?<сообщения об ошибках во время работы загрузчика>
        <сообщения синтезированной программы>
```

При построении алгоритма решения задачи к входным переменным $v \in V$ (значения которых известны) применяются операторы, соответствующие отношениям вычислимости. Множество переменных с известными значениями расширяется. Решение задачи существует, если на некотором шаге это множество содержит все выходные переменные задачи. При движении в обратном направлении – от выходных переменных к входным – формируется последовательность операторов решения задачи.

Интересно, что логическая теория синтеза программ в системе ПРИЗ была разработана спустя несколько лет после создания системы. Эта теория специально ориентирована на синтез программ и более удобна в этой области, чем исчисление предикатов первого порядка.

Формулы этой логической теории строятся из логических связок и отношений вычислимости, имеющих вид: $u \underset{f}{\rightarrow} v$. Интерпретация такого отношения: v числимо по u применением функции f. Теорема существования решения – формула $\exists f\,(u \underset{f}{\rightarrow} v)$. Она выводится из аксиом, образующих вычислительную модель задачи.

При поиске доказательства помимо правила вывода modus ponens:

$$\frac{A \rightarrow B;\ A}{B}$$

используются и специальные логические правила вывода.

Пример: правило вывода системы ПРИЗ для синтеза линейной последовательности операторов:

$$\frac{x \underset{f1}{\rightarrow} y;\ y \underset{f2}{\rightarrow} z}{x \underset{f1,f2}{\longrightarrow} z}$$

где *f1, f2* - композиция двух функций *f1* и *f2: f2 (f1 (x)).*

Ветвление появляется в синтезируемой программе, если условие вычислимости является предикатом *P(z)*. Если в ходе доказательства, условием вычислимости является теорема существования решения другой задачи (подзадачи), в синтезируемой программе появляется вызов процедуры, решающей эту подзадачу. Если в подзадаче фигурирует та же теорема существования, что и доказываемая, возможно построение рекурсивной процедуры. Для основных видов циклов их управляющие структуры запрограммированы заранее и представлены в вычислительной модели в виде аксиом (построение тела цикла рассматривается как подзадача).

Выдающиеся исследования Энна Тыугу по решению задач на вычислительных моделях и синтезу программ намного опередили время. При этом система ПРИЗ не нашла практического применения.

Система ЛИСС (работы М.Г.Мальковского и Е.И.Большаковой – 1980-1985 гг.)

Экспериментальная система синтеза лисп-программ ***ЛИСС*** разрабатывалась на кафедре алгоритмических языков факультета ВМК МГУ.

Основные особенности подхода:

Опыт показывает, что пользователю обычно трудно сразу дать точную спецификацию программы. Он часто опускает в спецификации очевидные или незначащие с его точки зрения дета ли, которые существенны для синтеза. Поэтому необходимым свойством системы синтеза является ее способность в случаях, когда человек не был исчерпывающе точен на этапе постановки задачи, – начинать работу с неполной формулировкой задачи и уточнять ее автоматически или с помощью пользователя. Целесообразным часто оказывается и вмешательство человека в процесс синтеза для внесения дополнений, поправок и т.п. Так как не исключается возможность ошибки в исходной спецификации, в системе нужны средства проверки соответствия спецификации, а при необходимости и самой программы. замыслу пользователя, а также средства исправления найденных ошибок. Они должны обеспечивать возможность изменения исходной спецификации и повторения процесса синтеза.

Своеобразие системы ЛИСС заключается именно в наличии таких возможностей и гибкой схеме диалогового взаимодействия пользователя с системой на всех этапах их совместного решения задачи синтеза: на этапе постановки задачи, в ходе самого синтеза, на этапе проверки полученных результатов.

Система ЛИСС **предназначена для синтеза типичных лисп-программ** (функций) обработки списков, используемых как вспомогательные при построении более сложных реальных программ. Пользователю системы необходимо знание основных понятий Лиспа (атом, список и т.п.), в терминах которых и формулируется задание на синтез. Знание же приемов программирования и элементарных функций Лиспа, из которых строится программа, не требуется.

Основным принципом при разработке интерфейса пользователя с системой ЛИСС было требование естественности, простоты и удобства общения для человека. Для того, чтобы удовлетворить этому требованию и повысить эффективность работы системы (с точки зрения достижения конечного результата – синтеза программы), при проектировании интерфейса были приняты во внимание как общие закономерности человеческого общения (краткость и фрагментарность реплик диалога, способность человека исправлять речевые ошибки собеседника), так и специфические особенности общения в рассматриваемой проблемной области, технические условия (работа пользователя за терминалом).

Остановимся на факторах, предопределивших выбор языка (языков) общения, схему функционирования системы и другие ее характеристики.

Одним из частых свойств формулируемых человеком спецификаций программ является их содержательная неполнота. Чтобы не ограничивать пользователя в выборе способов формулирования задачи синтеза и избавить его от необходимости составления исчерпывающе полных формальных спецификаций, в качестве входного языка системы ЛИСС выбран естественный (русский) язык. Возможность спецификации программы на естественном языке привлекательна для неподготовленного пользователя, весьма смутно представляющего, что и в какой форме должно быть указано в формальной спецификации. Удобна эта возможность и для профессионального программиста, который, разрабатывая большую программу, не хочет отвлекаться на написание вспомогательных процедур. Синтез таких процедур он поручает системе (своего рода "программисту-подмастерью" – этот термин использует К.Хьюитт, правда, в несколько ином контексте и в ином смысле) и не занимается продумыванием различных мелких деталей их работы, которые нужно было бы указать в формальной спецификации.

Используемый в системе ЛИСС лингвистический процессор позволяет задавать исходную спецификацию в достаточно произвольной форме и обеспечивает устойчивое функционирование системы даже в случаях появления в тексте незнакомых ей слов, оборотов, форм с ошибками, характерными при работе человека за терминалом (нажатие соседней клавиши, пропуск буквы и т.п.). Кроме того, пользователь может ввести удобные для него условные обозначения, позволяющие сократить длину набираемых на клавиатуре сообщений.

Если словесная спецификация оказалась неполной, ЛИСС начинает ее уточнять, выявляя и восстанавливая отсутствующие детали (нужные для синтеза). Для того, чтобы система синтеза построила программу, в точности соответствующую замыслу пользователя, в идеале необходимо разрешение всех обнаруженных неясностей и противоречий в диалоге с пользователем. Однако такой исчерпывающий уточняющий диалог реально может оказаться очень длительным, а следовательно, обременительным для пользователя.

Система ЛИСС реализует компромисс между требованиями надежности результата (соответствия программы и цели пользователя) и краткости диалога. На основе заложенных в нее знаний она автоматически пополняет исходную спецификацию и выбирает наиболее вероятную интерпретацию неоднозначностей, обращаясь к помощи пользователя только в наиболее сложных случаях. В принципе построенная системой программа может не соответствовать замыслу пользователя. Это происходит либо в том случае, когда уточняющие стандартные предположения системы оказываются при решении очередной задачи неверными, либо когда сама исходная спецификация составлена пользователем неточно, то есть описывает не ту программу, в создании которой заинтересован человек.

В распоряжении пользователя ЛИСС имеются средства обнаружения и исправления подобных ошибок. Он может, в частности, проверить текущее состояние процесса синтеза, исправить спецификацию, отменить сделанные системой предположения. Так как желательно возможно более раннее выявление всех ошибок, в ЛИСС разрешено осуществлять проверки и исправления на различных этапах синтеза программы. Чтобы повысить уверенность пользователя в правильности построенной программы в качестве одного из методов контроля используется традиционное тестирование.

Синтез очередной программы начинается с перевода словесной спецификации на формальный внутренний язык системы. Как в процессе анализа исходной формулировки и перевода во внутреннее представление, так и во время построения программы может обнаружиться неполнота или противоречивость описания. В этих случаях система делает уточняющие предположения или же обращается к пользователю с вопросами. Таким образом, система сохраняет инициативу при совместном решении задачи синтеза, если только пользователь не прерывает последовательность уточняющих вопросов запросом контрольной информации о сделанных системой предположениях или о текущей (с учетом принятых предположений) спецификации. Это происходит, например, если очередной вопрос системы кажется пользователю неуместным. После проверок он может внести необходимые исправления, и синтез возобновится.

При успешном окончании синтеза полученная программа выдается на экран терминала (и, разумеется, запоминается), и пользователь может проверить ее работу на заданных им тестах. Если пользователь не удовлетворен результатами тестирования, он продолжает отладочные действия, возобновляя работу системы по синтезу нужной ему программы.

Таким образом в общем случае диалог пользователя с системой ЛИСС состоит из уточняющих постановку задачи синтеза вопросов системы, которые могут чередоваться с контролирующими

действиями пользователя. При этом система достаточно гибко реагирует на степень и характер неполноты исходной спецификации, и ход диалога зависит в конечном итоге от степени программистской подготовки пользователя, его знаний о ПО, его опыта работы с системой. Так, если исходная словесная спецификация оказалась содержательно полной, то уточняющих вопросов нет, и после ввода исходной формулировки пользователь получает синтезированную программу.

Чтобы обеспечить эффективную работу с системой как новичка, так и опытного пользователя, оказалось целесообразным выделить два режима работы ЛИСС. **Первый режим** предназначен для пользователей, слабо знающих язык Лисп и нечетко представляющих себе возможности системы. Кроме непосредственного решения задачи пользователя (синтеза программы) цель ЛИСС при работе в этом режиме − помочь человеку ориентироваться и проблемной области и научить его строить спецификации "понятные" системе без каких бы то ни было уточнений. Если пользователь работает с системой достаточно регулярно и заинтересован в том, чтобы максимально сократить диалог, он должен соответствующим образом адаптироваться к возможностям системы.

При работе в первом режиме инициатива в диалоге отдана системе, все уточняющие вопросы задаются по схеме "меню". Помимо вариантов ответа, указываемых в вопросе, допускаются и некоторые директивы. По директиве ИНФОРМАЦИЯ на экран терминала выдается текст, информирующий о режимах и правилах работы с системой ЛИСС, по директиве ПОМОЩЬ описываются текущее состояние диалога и ожидаемые действия пользователя. Директивы позволяют определить нужные пользователю проверки состояния процесса синтеза (директива ПРОВЕРКА), закончить проверки и исправления и продолжить синтез (ПРОДОЛЖИТЬ) и т.п.

Эксперименты показали, что нескольких сеансов работы с системой ЛИСС в первом режиме достаточно для ознакомления с возможностями системы и перехода ко второму, основному, режиму работы, в котором пользователь управляет синтезом с помощью небольшого набора приказов. Подавляющее большинство этих приказов эквивалентно репликам пользователя при выборе пунктов меню в первом режиме работы (доступны в те же моменты диалога и влекут за собой те же действия системы). Часть приказов позволяет производить нужные проверки и исправления (например, по приказу ПРЕДПОЛОЖЕНИЯ на экран терминала выдается список сделанных системой уточняющих предположений). Другие приказы позволяют задавать степень подробности уточняющего диалога; указывать, какие предположения системы нужно согласовывать с пользователем; менять форму уточняющих вопросов (меню или свободная) и др.

Пример 1. Работа с ЛИСС ведется в первом режиме

Какое преобразование осуществляет ваша функция?
Можно привести пример ее работы

ОПРЕДЕЛЯЕТ ЧИСЛО ЗАДАННЫХ ЭЛЕМЕНТОВ В СПИСКЕ

Как задаются нужные элементы? выберите вариант:
- задано свойство, которым должны обладать элементы
- задан сам элемент (лисп-выражение)
Возможные ответы: свойс / выр

ВЫР

Являются ли подсчитываемые элементы атомами?
Ответ: да / нет

ПРОВЕРКА

Что будете проверять:
- интерпретацию системой исходной формулировки
- сделанные системой предположения
- текущую спецификацию функции
- все уточнения исходной формулировки
Ответы: инт / пред / спец / уточ

ИНТ

Ваша исходная словесная формулировка понята системой так:
1 - функция в списке, являющемся значением одного ее аргумента,
2 - подсчитывает количество элементов,

З - определяемых другим аргументом функции.
Ответы: правильно / ошибка

ПРАВИЛЬНО

Синтез функции продолжен
Являются ли подсчитываемые элементы атомами?
Ответ да / нет

ДА

Функция построена. ее имя - C2, аргументы:
X - анализируемый список, Y - подсчитываемые элементы
Определение (текст) функции:

```
(SEXPR C2 (LAMBDA (X Y)
          (COND  ((NULL X) 0)
                 ((EQ (CAR X) Y) (ADD1 (C2 (CDR X) Y)))
                 (T (C2 (CDR X) Y)))))
```

Пример 2. Работа также ведется в первом режиме

Какое преобразование осуществляет ваша функция? **(1)**
Можно привести пример ее работы

ЗАМЕНЯЕТ В СПИСКЕ АТОМ A НА ВЫРАЖЕНИЕ E

Необходимо уточнить:
A - конкретный атом или обозначение аргумента функции?
Возможные ответы: конк / арг **(2)**

КОНК

Надо ли заменять только первый по порядку указанный элемент списка
или все такие элементы?
Ответы: перв / все **(3)**

ПРОВ

Что будете проверять:
- интерпретацию системой исходной формулировки
- сделанные системой предположения
- текущую спецификацию функции
- все уточнения исходной формулировки
Ответы: инт / пред / спец / уточ **(4)**

ПРЕД

В процессе синтеза системой сделано предположение:
E - имя аргумента функции, а не конкретное выражение
Ответы: прав (правильно) / отм (отменить) **(5)**

ПРАВИЛЬНО

Будете проверять дальше или продолжить синтез?
Ответы: пров / прод **(6)**

ПРОД

Синтез продолжен
Надо ли заменить только первый по порядку элемент списка
или все такие элементы?
Ответы: перв / все

ПЕРВ

Функция построена. ее имя - B12,
Аргументы: X - преобразуемый список, E - замена.
Определение (текст) функции: **(7)**

```
(SEXPR B12 (LAMBDA (X E)
           (COND ((NULL X) NIL)
```

((EQUAL (CAR X) (QUOTE A)) (CONS E (CDR X)))
(T (CONS (CAR X) (B12 (CDR X) E))))))

Будете проверять полученную функцию или начать синтез новой?

Ответы: пров / нач **(8)**

ПРОВ

Что будете проверять:
- сделанные системой предположения
- точную спецификацию функции
- все уточнения исходной формулировки
- работу функции на тестах

Ответы: пред / спец / уточ / тест **(9)**

ТЕСТ

Введите тест (можно в словесной форме) **(10)**

АРГУМЕНТЫ ФУНКЦИИ - (A C (A)), B;

Результат - (B C (A))

ПРАВИЛЬНО (11)

В ходе анализа ответа пользователя на вопрос системы (1) Анализатор системы ЛИСС обнаруживает две неточности словесной спецификации лисп-функции. Одна из неточностей (АТОМ А – это либо атом, имеющий печатное наименование А, либо аргумент функции, имеющий имя А и тип атома) разрешается обращением к пользователю (2). Другая же неопределенность уточняется автоматически (ВЫРАЖЕНИЕ Е – это скорее всего некоторое лисповское выражение, являющееся значением аргумента Е), при этом запоминается сделанное предположение. По окончании работы Анализатор получает фрейм задачи синтеза, содержащий такой фрагмент:

((ХАРД ЗАМ)
(ИССП (АРГ (ТИП СПИС)))
(ХАРЭЛ (ФОРМЗ КОНК) (СПОСЗ (ХАР (= (ВЫР АТОМ А)))))
(В/ЗОБ (АРГ (ТИП ВЫР) (ИМЯ Е))))

Известны описатели: характеристика действия функции (слот ХАРД) – замена (ЗАМ); исходный или преобразуемый объект (слот ИССП) – объект типа список; форма и способ задания обрабатываемого элемента (слот ХАРЭЛ, подслоты ФОРМЗ и СПОСЗ) – задан конкретный элемент (КОНК), он является атомом А; сам объект-замена (слот В/ЗОБ) – выражение с именем Е.

На этапе предварительной обработки полученного фрейма, после оценочного сопоставления с известными фреймами типа задач обнаруживается один допустимый фрейм (имя типа – РЕД1). В ходе полного сопоставления фреймов фрейм задачи пополняется, и описатели ИССП и В/ЗОБ приобретают следующий вид:

(ИССП (АРГ (ТИП СПИС) (ИМЯ Х) (ПНОМ 1)))
(В/ЗОБ (АРГ (ТИП ВЫР) (ИМЯ Е) (ПНОМ 2)))

Первый аргумент функции – список с именем Х, второй – выражение с именем Е.

Формируется также часть фрейма задачи, описывающая внешние характеристики синтезируемой функции (имя, число и тип аргументов):

((=ИМЯ В12) (=ЧАРГ 2) (=ТАРГ СПИС)).

Затем Конструктор начинает дедуктивный синтез программы по сформированному фрейму задачи.

По окончании синтеза полученная функция выводится на экран терминала (7). ЛИСС уточняет, какие проверки пользователь будет осуществлять (9), запрашивает нужные тестовые данные (10) и сообщает результат проверки функции на тесте (11).

Пример 3. Работа ведется во втором режиме

Опишите действие вашей функции

Можно привести пример ее работы

ПОЛУЧАЕТ ИЗ СПИСКА (A B (C) E K) СПИСОК (A B E K)

Чем определяется исключаемый из списка элемент?

Выберите вариант:
- элемент определяется номером
- элемент определяется некоторым свойством
Ответы: ном / свойс

СВОЙС

Функция построена. ее имя - B5, аргументы: X - исходный список
Определяющее выражение функции

```
(SEXPR B5 (LAMBDA (X)
        (COND ((NULL X) NIL)((NOT (ATOM (CAR X))) (CDR X))
            (T (CONS (CAR X) (B5 (CDR X)))))))
```

СПЕЦ

Спецификация построенной функции:
1 - функция в списке, являющемся значением ее аргумента x,
2 - исключает
3 - первый по порядку элемент,
4 - не являющийся атомом

ПРАВИЛЬНО

Поскольку работа ведется во втором режиме, по умолчанию устанавливается "краткий" диалог. В нашем случае пользователь описывает функцию примером ее работы. Работа Конструктора заканчивается формированием условного выражения. Построенная функция выводится на экран терминала, и пользователь просматривает спецификацию этой функции.

Пример 4. Работа ведется во втором режиме

Введите спецификацию **(1)**

ЗАКЛЮЧАЕТ В СКОБКИ ЭЛЕМЕНТ У СПИСКА X

Функция построена. ее имя - B6, аргументы:
X - исходный список, У - обрабатываемый элемент
Определяющее выражение функции:

```
(SEXPR B6 (LAMBDA (X Y)                                    (2)
        (COND ((NULL X) NIL)
            ((EQUAL Y (CAR X)) (CONS (LIST (CAR X)) (CDR X)))
            (T (CONS (CAR X) (B6 (CDR X) Y)))))
```

ТЕСТ:
АРГУМЕНТ - (Р А К А) И АТОМ А,
РЕЗУЛЬТАТ - (Р (А) К (А))

Неправильно, на тестовых данных получен результат:
(Р (А) К А) **(3)**

ПРЕД

Сделанные предположения:
1 - У - имя аргумента, а не конкретный атом
2 - функцией обрабатывается только первый
по порядку (от начала) элемент списка **(4)**

ОТМ 2

Синтез возобновлен с учетом исправлений
Функция построена. ее имя - B6, аргументы:
X - исходный список,
У - обрабатываемый элемент.
Определяющее выражение функции:

```
(SEXPR B6 (LAMBDA (X Y)                                    (5)
        (COND ((NULL X) NIL)
            ((EQUAL Y (CAR X)) (CONS (LIST (CAR X)) (B6(CDR X))))
            (T (CONS (CAR X) (B6 (CDR X) Y)))))
```

В приведенном примере после ввода пользователем исходной спецификации система распечатывает текст функции (2).

Однако пользователь обнаруживает на тесте несоответствие ее своему замыслу (3) и проверяет предположения, сделанные во время работы системы (4). После отмены пользователем неправильного предположения (предположение № 2) ЛИСС возобновляет процесс синтеза, начиная с шага, на котором было принято и использовано это предположение. Полученная после исправления программа распечатывается (5).