приятия внедряющего ГЛОНАСС. Очевидно, что даже предприятия одного направления имеют большое количество нюансов, которые необходимо учесть как, при постановке задачи, так и при программировании комплекса. Задачи же стоящие перед предприятиями различных направлений отличаются куда более существенно. Что касается пассажирского транспортного предприятия, то используемые в данный момент программные продукты разрабатывались с условием максимальной универсальности, вследствие чего не решают целый пласт специализированных задач. Если говорить об автотранспортных предприятиях, то их задачи в определенной степени могут быть решены и автоматизированы в комплексе поставляемого программного обеспечения. Однако городской электротранспорт имеет ряд существенных отличий от автобусных предприятий, что делает использование поставляемого программного обеспечения малоэффективным.

В работе рассматривается задача оценки, выборки, представления и формализации необходимых данных, поступающих от ГЛОНАСС для контроля производства и принятия управляющих решений в работе городского электрического транспорта.

Ошибка неполного (некорректного) описания процедуры предложений в программе на языке Пролог[4]

О.Н. Половикова
АлтГУ, г. Барнаул

Процесс создание программы на языке Пролог включает декларативный и процедурный уровни программирования. Можно создавать программы на языке Пролог, не зная, как в действительности происходит поиск заданного результата, как Пролог-система нашла выдаваемое решение. Декларативность языка Пролог позволяет определить в программе только необходимую совокупность отношений между объектами предметной области и требуемый результат ее работы. Но для того, чтобы разобраться, как Пролог-система осуществляет построение решения, следует понять процедурный уровень работы программы. Декларативный уровень программирования на языке Пролог не позволяет реализовать эффективный поиск альтернативы решения в случае их комбинаторного перебора (оптимизировать вычислительный процесс), а также выявить логические ошибки в программе [1].

[4] Работа выполнена при поддержке аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы (2009-2010 годы)» (код проекта №2.2.2.4/4278).

На процедурном уровне программирования важен порядок предложений внутри совокупности одноименных правил (в процедуре предложений), а также порядок хвостовых целей в теле предложений. От порядка предложений зависит порядок поиска решений и порядок, в котором будут находиться ответы на вопросы. Порядок целей влияет на количество проверок, выполняемых программой при решении, так как при попытке повторного согласования цели (в случае неудачного сопоставления какой-либо подцели) система возобновляет просмотр базы с предложения, непосредственно следующего за тем, которое обеспечивало согласование цели ранее.

Одной из распространенных ошибок в программах на языке Пролог является ошибка неполного (некорректного) описания условий выполнения правила в процедуре предложений для целевого утверждения (подцели). Сложность поиска подобных ошибок заключается в том, что некорректную работу программы можно обнаружить только, если произойдет повторное согласование утверждения, в описании процедуры предложений которого допущена неточность. Поиск такого рода ошибок возможен только, если разработчик знает не только декларативный, но и процедурный уровень программирования на Прологе. Рассмотрим пример программы с ошибкой.

Постановка задачи

Необходимо определить, является ли сумма моделей двух чисел меньше некоторой числовой константы (например, 12).

Пример программы для решения поставленной задачи (строки программы пронумерованы с использованием комментариев)

```
/*1*/?- read(X,"X= "),read(Y,"Y="),
abs(X,A),nl,write(A),abs(Y,B),nl,write(B),(A+B)<12.
/*2*/abs(X, X):- X>0.
/*3*/abs(X, -1*X).
```

В первой строке указана цель, в случае успешного согласования которой Пролог-система возвратит ответ «Да», в противном случае – «Нет». Во второй и третей строчке описана процедура правил для определения модуля числа: во второй строке указано правило (предложение) нахождения модуля для положительного значения переменной `X`, в третьей строке описано правило (факт), которое выполняется, если не выполнено первое правило. Если не выполняется условие `X>0`, второй параметр отношения `abs(X,A)` сопоставить со значением `-1*X`.

В данных рассуждениях нет ошибок. Пролог-система в процессе поиска решения (при согласовании цели программы, заданной в первой строке) будет пытаться согласовать каждую подцель с правилом или фактом из соответствующей процедуры предложений, рассматривая эти предложения в строгом порядке (в порядке следования в тексте программы). Но, при решении поставленной задачи не учитывается вся специфика работы механизма

возврата, на котором основан поиск решения Пролог-системой. Если попытка согласовать утверждение `(A+B)<12` потерпит неудачу, Пролог-система в цепочке конъюнкции подцелей будет заново пытаться согласовывать каждую подцель, пока либо все подцели не будут согласованы (ответ «Да»), либо все возможные варианты согласования подцелей уже рассмотрены, но решения не найдено (ответ «Нет»). Поэтому если утверждение `(A+B)<12` потерпит неудачу, то Пролог-система будет пытаться повторно согласовывать подцель `abs(Y,B)` или подцели `abs(X,B)` и `abs(X,A)`. Если значение переменной `Y` (или `X`) было положительным, то возникнет логическая ошибка, механизм возврата отношение (подцель) `abs(Y,B)` (или `abs(X,A)`) согласует с фактом `abs(X, -1*X)`.

Рассматриваемая программа на значениях `X=-7` и `Y=-6`, дает корректный результат: «Нет», так как подцели `abs(Y,B)` и `abs(X,B)` согласуются только с фактом `abs(X, -1*X)`, механизм возврата не будет заново пытаться перебирать предложения процедуры, так как все возможные варианты согласования подцелей уже рассмотрены. А вот на значениях `X=7` и `Y=-6` (или `X=-7` и `Y=6` или `X=7` и `Y=6`) программа выдаст некорректный результат «Да», так как в процессе поиска решения попытка согласовать утверждение `(A+B)<12` потерпит неудачу, произойдет возврат и повторное согласование всех указанных подцелей.

Существует несколько способов решения описанной выше ошибки. Например, можно воздействовать на процесс возврата, используя «отсечение». Данный встроенный предикат позволяет запретить просматривать Пролог-системе все альтернативы в процедуре предложений `abs(_,_)`, в случае согласования подцели с первым правилом. Кроме этого, можно втрое предложение записать в виде правила, чтобы гарантировать его выполнение только для неположительных значений первого параметра в отношении `abs(_,_)`.

```
/*Устранение ошибки, воздействием на процесс возврата*/
/*2*/abs(X, X):- X>0, !.
/*3*/abs(X, -1*X).
/*Устранение ошибки, заменив факт правилом*/
/*2*/abs(X, X):- X>0, !.
/*3*/abs(X, -1*X): X=<0.
```

Библиографический список

1. Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. с англ. – М. : Мир, 1990.