УДК 519.95

А.Е.С о л о в ь е в

ЛОГИЧЕСКИЕ УСЛОВИЯ В ЯЗЫКАХ, ОРИЕНТИРОВАННЫХ НА НЕПРОФЕССИОНАЛЬНОГО ПРОГРАММИСТА

(г.Пе р м ь)

Традиционный путь привлечения непрофессионалов к использованию ЭВМ для обработки результатов исследований, вспомогательных расчетов и т.п. связан с созданием проблемно-ориентированных языков. В настоящее время это основной и наиболее разумный путь, однако приходится мириться с узкой специализацией таких языков. Работа пользователя может быть облегчена за счет использования достаточно внушительных по объему проблемно-ориентированных пакетов. Но исследовательская работа нередко требует выхода за возможности пакета и здесь приходится возвращаться к традиционным и традиционно неудобным для непрофессионала процедурно-ориентированным языкам.

В данной статье рассматривается ряд аспектов иного подхода к созданию таких языков, связанного с поднятием концептуального уровня некоторых конструкций процедурно-ориентированных языков.

Наличие в алгоритмических языках средств обработки арифметических выражений делает процедуру вычисления по формуле элементарной не только для профессионального программиста, но и для непрофессионала. Если отвлечься от вопросов, связанных с работой внешних устройств и т.п. и проанализировать лишь особенности проектирования обрабатывающих алгоритмов, то следует констатировать, что основной проблемой для непрофессионала является формирование управляющей структуры. Недаром многие методики оценки сложности алгоритмов учитывают количество логических условий в тексте программы или логических блоков в блок-схеме алгоритма.

Существует объективная сложность задачи, а теория сложности вычислений дает оценки этой сложности, которые принципиально нельзя улучшить. Однако в данном случае речь идет о сложности восприятия алгоритма человеком, а это не одно и то же.

Итак, если сложность алгоритма есть монотонная функция от числа логических условий, то следует рассматривать возможности сокращения числа условий за счет укрупнения их "емкости". При этом следует отметить два момента:

укрупнение логических условий совсем не обязательно должно приводить к усложнению восприятия человеком таких условий, поскольку обычно логические условия чрезвычайно "мелки". Так что укрупнение может быть даже желательно;

укрупнение за счет простого удлинения условий, т.е. использования сложных логических выражений с применением связок, вряд ли может существенно упростить алгоритм, так как количество элементарных условий не уменьшается, они лишь лучше локализуются.

Напрашивается естественный вывод – попытаться "увеличить мощность" самих элементарных условий, чтобы они были более адекватны логическим условиям, используемым в предметной области исследования. Существует традиционный прием, связанный с включением процедур в логическое условие. Однако здесь его следует признать не более, чем некой разновидностью макроса, не снимающей поставленной проблемы. В данном случае в число элементарных условий можно, например, включить не только проверку выполнения конкретных отношений, но и проверку выполнения с в о й с т в отношений, что позволяет сделать принципиальный шаг в поднятии уровня языка.

Пример. Пусть ρ – некоторое отношение, тогда свойство рефлексивности в математической (!) литературе записывают как $x\rho x$. Типичная ошибка новичка заключается в уверенности, что поскольку антирефлексивность записывается как $\neg x\rho x$, то антирефлексивность есть отрицание рефлексивности. Однако более строгая математическая запись для свойства рефлексивности будет $\forall x(x\rho x)$, а для антирефлексивности $\forall x \neg(x\rho x)$. Если же взять отрицание утверждения о рефлексивности, то получим

$$\neg\forall x(x\rho x) = \exists x \neg(x\rho x) \neq \forall x \neg(x\rho x).$$

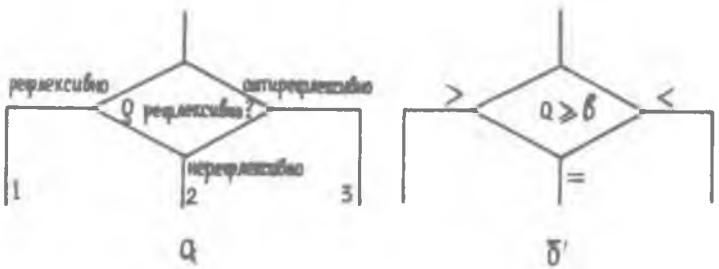
Отрицание антирефлексивности также не дает ожидаемого

$$\neg\forall x \neg(x\rho x) = \exists x(x\rho x) \neq \forall x(x\rho x).$$

Значит для замыкания необходимо введение свойства нерефлексивности, которое вообще нельзя выразить в сокращенной записи, а в квантифицированной форме будет иметь вид

$$\exists x \neg (x \rho x) \& \exists x (x \rho x).$$

Если рассматривать как элементарный логический блок проверки условия рефлексивности некоторого отношения ρ (рисунок (а)), то для



Р и с. Примеры обощенного логического блока: а–проверки условия рефлексивности некоторого отношения ρ ; б–с тремя выходами

такого блока естественно наличие трех выходов. Вопросы "управления" выходами будут рассмотрены позже. Для того, чтобы убедиться, что традиционный логический блок неадекватен не только предлагаемым новым логическим условиям, но и самим традиционным условиям, обсудим еще один пример.

Для многих рассуждений в повседневной практике "не верно, что больше" означает "меньше", а "не верно, что $a \leqslant b$ " означает "$b \leqslant a$". Это, кстати, хорошие иллюстрации невыполнения логического закона исключенного третьего (т.е. $(a > b) \vee (a < b) \not\equiv 1$, так как возможен еще и случай, когда $a = b$) и закона противоречия (т.е. $(a \geqslant b) \& (b \geqslant a) \not\equiv 0$, поскольку опять же в случае $a = b$ выражение слева равно I).

Однако с более строгой математической точки зрения" $\neg (a > b)$ " означает "$(a \leqslant b)$ ", а " $\neg (a \leqslant b)$ " означает "$(a > b)$ ". И то и другое справедливо лишь при условии, что a и b сравнимы. Многих ошибок или описок здесь можно избежать, если опять же взять для традиционного отношения $\geqslant$ логический блок с тремя выходами, как показано на рисунке (б), что,правда,напоминает многократно раскритико-

ванную конструкцию Фортрана. Есть смысл снова внимательно на нее посмотреть!

Выполненные автором исследования позволяют сделать предположение, что многие сложности, связанные с использованием логических условий, сводятся к неадекватности математической операции отрицания и многообразному использованию понятия отрицания в повседневной практике. В результате предложена система из четырех операций отрицания, которые содержательно описываются следующим образом:

N_0 – неотрицающее отрицание, соответствует фразе "не только это";
N_1 – категорическое отрицание, соответствует фразе "все наоборот";
N_2 – дополняющее отрицание, соответствует фразе "только не это";
N_3 – некатегорическое отрицание, соответствует фразе " не то, чтобы это".

Система обладает многими интересными свойствами, однако в качестве примера рассмотрим лишь результат ее применения для анализа свойства рефлексивности:

$$N_0 \forall x (x \rho x) \equiv \forall x N_0 (x \rho x) \approx \forall x (x \rho x).$$

Проверка условия рефлексивности при N_0 не имеет смысла, так как в этом случае рефлексивность и ее отрицание совпадают:

$$N_1 \forall x (x \rho x) \equiv \forall x N_1 (x \rho x).$$

Отрицание N_1 применимо к отношениям, которые могут быть либо рефлексивными, либо антирефлексивными. Логический блок для этого случая содержит лишь выходные ветви I и 3 (см.рисунок (а)):

$$N_2 \forall x (x \rho x) \equiv \exists x N_2 (x \rho x).$$

Орицание N_2 совпадает с традиционным, поэтому здесь возможны все три случая:

$$N_3 \forall x (x \rho x) \equiv N_3 (x \rho x).$$

Отрицание N_3 соответствует таким отношениям, для которых возможна либо рефлексивность, либо нерефлексивность, т.е. ветви I и 2.

Рассуждения относительно не какого-то свойства вообще, а конкретного отношения $\geq$ совершенно аналогичны.

Решения, рассмотренные в связи с логическим блоком для процедурных языков, могут быть просто адаптированы для функциональных языков типа ЛИСП.

Наличие нескольких отрицаний позволяет более гибко и точно формулировать логические условия, что, с одной стороны, может исключить необходимость последующих дополнительных проверок, а с другой - обеспечивает возможность проверки не только отдельных отношений, но и совокупностей отношений, обладающих общим свойством.

Об особенностях реализации логических блоков можно сказать следующее.

Если проверяемое отношение задано перечислением кортежей, то проверка наличия нужного свойства, например той же рефлексивности, может осуществляться через перебор всех кортежей: необходимо убедиться, что для каждого элемента a есть пара $<a, a>$. Это простой способ, однако сам перебор может быть неприемлемо долгим. Например, в общеинститутском файле "Результаты сессии" поиск фамилии человека, сдавшего экзамен самому себе, может быть весьма продолжительным и, скорее всего, безуспешным.

В качестве промежуточного можно рассматривать вариант, когда первоначально определяются свойства отношения, они сохраняются, а затем лишь корректируются при изменениях базового множества.

Более перспективным представляется задание отношений с помощью описаний свойств или аксиоматически, тогда проверка логических условий будет осуществляться с помощью логического вывода.

При наличии таких свойств некоторого отношения φ, как антирефлексивность и транзитивность, может быть получено подтверждение, что это отношение антисимметрично. Запись же этих рассуждений с помощью традиционного языка с элементарными отношениями была бы весьма громоздкой. Наличие свойства антирефлексивности у отношения "сдавать сессию" позволяет сразу дать отрицательный ответ на вопрос об официальной сдаче экзамена самому себе.

Еще более очевидными становятся преимущества данного подхода, если использовать нечеткие отношения типа "немного больше", "почти симметрично" и т.п. Здесь естественно вписывается аппарат нечеткой логики. Кроме того, работа с множествами кортежей вызывает сильные аналогии с реляционным подходом.

Введение в алгоритмический язык "сильных" логических условий делает принципиально возможным существенно упростить запись многих алгоритмов, облегчить общение с ЭВМ непрофессионалов.