ДОПОЛНИТЕЛЬНЫЕ КОНСТРУКЦИЯ ЯЗЫКА L И ИХ СЕМАНТИКА

Существуют два способа добавления новых синтаксических конструкций в язык:

- ✓ **Первый способ** заключается в добавлении новых конструкций и соответствующих им семантических определений (операционных, аксиоматических или денотационных). Этот путь формально расширяет определение языка и усложняет доказательства, которые используют структурную индукцию.

ДОПОЛНИТЕЛЬНЫЕ КОНСТРУКЦИЯ ЯЗЫКА L И ИХ СЕМАНТИКА

- ✓ **Второй способ** заключается в сохранении ядра языка без изменения и введении дополнительных конструкций с помощью синтаксических равенств. Это означает, что программа может быть сведена к основным конструкциям ядра языка чисто синтаксическими средствами.

ЦИКЛ REPEAT–UNTIL

Введем в язык **L** новую синтаксическую конструкцию – цикл **repeat–until**:

C :: x:=E | C_1; C_2 | if B then C_1 else C_2 fi | while B do C od | repeat C until B

Зададим соответствующее правило вывода:

$$\{P\}\ C\ \{Q\},\ Q\ \&\ \neg B \rightarrow P$$

(R6) ———————————————————

$$\{P\}\ \text{repeat}\ C\ \text{until}\ B\ \{Q\ \&\ B\}$$

ЦИКЛ REPEAT–UNTIL

С другой стороны, можно определить цикл **repeat–until** через уже имеющиеся в языке **L** конструкции (цикл **while–do** и последовательность) следующим образом:

repeat C until B = C; while ¬B do C od

Используя эту связь, можно вывести правило вывода **R6** из **R2**, **R4** и **R5**.

ЦИКЛ REPEAT–UNTIL

Пусть задано **{P} C {Q}**. Тогда естественным является выбор **Q** в качестве соотношения в точке входа в цикл **while ¬B do C od**.

Далее, по правилу вывода **R4**, имеем:

$$\frac{\{Q \;\&\; \neg B\}\ C\ \{Q\}}{\{Q\}\ \underline{\textbf{while}}\ \neg B\ \underline{\textbf{do}}\ C\ \underline{\textbf{od}}\ \{Q \;\&\; B\}} \qquad (1)$$

ЦИКЛ REPEAT–UNTIL

Далее, если задано **Q & ¬B → P**, то, по правилу вывода **R5**, получим:

$$\frac{\mathbf{Q \,\&\, \neg B \rightarrow P,\ \{P\}\ C\ \{Q\}}}{\mathbf{\{Q \,\&\, \neg B\ \}\ C\ \{Q\}}} \qquad (2)$$

Комбинируя (1) и (2), получим:

Отсюда, по правилу вывода **R2**, имеем:

$$\frac{\mathbf{\{P\}\ C\ \{Q\},\ Q \,\&\, \neg B \rightarrow P}}{\mathbf{\{Q\}\ \underline{while}\ \neg B\ \underline{do}\ C\ \underline{od}\ \{Q \,\&\, B\}}} \qquad (3)$$

что соответствует правилу вывода **R6**.

ЗАВЕРШИМОСТЬ ЦИКЛА REPEAT–UNTIL

Аналогично циклу **while–do** доказательство утверждений **{P} C {Q}** и **Q & ¬B → P** для цикла **repeat–until** требуется для доказательства частичной корректности, т.е. корректности в случае завершимости цикла.

В то же время необходимо доказывать также и **завершимость цикла**.

Для этого необходимо сформулировать условия завершимости цикла **repeat–until**, аналогичные условиям для цикла **while–do**:

$$P \rightarrow (E>0) \qquad (1')$$

ЗАВЕРШИМОСТЬ ЦИКЛА REPEAT–UNTIL

Для любого целочисленного выражения $\mathbf{E_0}$ справедливо

$$\mathbf{\{E_0=E>0\}\ C\ \{E_0>E\geq 0\},} \qquad (2')$$

где **E** – некоторое целочисленное выражение.

Т.е. **E≥0** является инвариантом цикла **repeat–until** и **E>0** в течение циклического процесса. Кроме того, условие (2') означает, что каждое выполнение команды **C** уменьшает значение **E**. Таким образом, из (1') и (2'), а также **{P} C {Q}** и **Q & ¬B → P**, следует вывод о конечности циклического процесса, так как **E** не может уменьшаться бесконечное число раз и оставаться положительным, что требует условие (1').

ПРОЦЕДУРЫ И ФУНКЦИИ И ИХ АКСИОМАТИЧЕСКАЯ СЕМАНТИКА

Расширим язык **L**, добавлением в него описаний **процедур** и **функций**.

Рассмотрим процедуры без параметров, чтобы отделить вызов процедуры от механизмов задания областей действия переменных и передачи параметров. Также ограничимся рассмотрением случая объявления лишь одной процедуры. Полученные результаты могут быть легко обобщены на случай объявления более, чем одной процедуры.

Пусть имеется нерекурсивная процедура без параметров **P**, объявление которой имеет следующий вид:

procedure P; C_0

где $\mathbf{C_0}$ – тело процедуры **P**.

Расширим множество программ языка **L** программами, допускающими вызов процедур:

C :: x:=E | C_1; C_2 | if B then C_1 else C_2 fi | while B do C od | repeat C until B | P

ПРОЦЕДУРНЫЕ ВЫЗОВЫ

Каждый вызов процедуры **P** относится к объявлению **procedure P; C_0**.

Для работы с процедурными вызовами, если C_0 нерекурсивно, в систему **H** вводится следующее правило:

(R7)

$$\frac{\{R\}\ C_0\ \{Q\}}{\{R\}\ P\ \{Q\}}$$

Для определения непротиворечивости полученной системы **H** необходимо расширить значение функции M_I применительно к программам, содержащим процедурные вызовы.

НЕПРОТИВОРЕЧИВОСТЬ СИСТЕМЫ H С ПРОЦЕДУРНЫМИ ВЫЗОВАМИ

Пусть для любой программы **C** в языке **L** запись $\mathbf{C[C_0/P]}$ означает программу, получающуюся в результате замещения всех вхождений **P** в **C** программой $\mathbf{C_0}$. Иными словами $\mathbf{C[C_0/P]}$ является макрорасширением **C**.

Тогда, для любой программы **C**, содержащей вызов процедуры **P**, значение $\mathbf{M_I(C)}$ определяется равным значению $\mathbf{M_I(C[C_0/P])}$.

Отсюда следует, что $\mathbf{M_I(P) = M_I(C_0)}$, а следовательно, правило **R7** является непротиворечивым, поскольку $\mathbf{C_0}$ не содержит вызова процедуры **P**.

Таким образом, непротиворечивость полученной системы **H** следует из непротиворечивости правил вывода **R2–R6** и определения $\mathbf{M_I(C)}$ для программ, содержащих вызов процедуры **P**.

РЕКУРСИВНОЕ ОБЪЯВЛЕНИЕ ПРОЦЕДУР

Для случая, когда объявление процедуры **prooedure P;** C_0 является **рекурсивным**, необходимо ввести в систему **H** следующее правило:

$$\{R\}\ P\ \{Q\} \vdash_H \{R\}\ C_0\ \{Q\}$$

(R8)

$$\{R\}\ P\ \{Q\}$$

Это правило читается как: «Справедливость **{R} P {Q}** вытекает из того факта, что справедливость **{R}** C_0 **{Q}** может быть доказана из предположения о справедливости **{R} P {Q}**».

Отметим, что полученная формальная система **H**, содержащая правило вывода **R8**, не является полной.

ОБЪЯВЛЕНИЕ ФУНКЦИИ

Пусть имеется объявление функции

function f(X): T; C,

где **f** – имя функции;

X – раздел спецификации формальных параметров;

T – тип результата;

C – тело функции.

Предположим, что функция **f** не приводит к побочному эффекту, т.е. внутри **C** не осуществляется присваиваний ни параметрам-переменным, ни глобальным переменным.

ПРАВИЛО ВЫВОДА ДЛЯ ФУНКЦИЙ

При вызове функции **f** осуществляется выполнение тела функции **C** с фактическими параметрами, задаваемыми списком **x**, помещенными на место формальных. Результат выполнения функции присваивается переменной с именем **f**.

Пусть для тела **C** функции **f** доказано, что **{P} C {Q}**, где **Q** содержит имя функции **f**, обозначающее результат функции. Тогда правило вывода для функции **f** имеет следующий вид:

(R9)

$$\frac{\{P\}\ C\ \{Q\}}{\forall x\ (P \rightarrow Q[f(x)/f])}$$

Неформально это правило может быть прочитано как: «Из того, что некоторое свойство справедливо для тела функции, следует, что это свойство будет справедливо для каждого вызова этой функции».

ПРОБЛЕМА ЗАВЕРШЕНИЯ ПРИ РАЗРАБОТКЕ ФУНКЦИЙ

Следует обратить особое внимание на проблему завершения при разработке функций.

Если для некоторого **x** при вызове функции **f(x)** ее выполнение не завершается, то присваивание вида **y:= f(x)** не имеет смысла, поскольку значение **f(x)** не определено.

В случае незавершения выполнения тела функции правило вывода **R9** может привести к противоречию.