ПОИСК СТРУКТУРНЫХ КОНФЛИКТОВ В ГРАФАХ БИЗНЕС-ПРОЦЕССОВ

И. В. Кучугуров[1], Н. Д. Кучугурова[2]

[1]*Российский государственный социальный университет*
Москва, Россия
E-mail: tws13@mail.ru
[2]*Московский педагогический государственный университет*
Москва, Россия
E-mail: dnkst@mail.ru

Рассматривается поиск конфликтов в графах бизнес-процессов. Предлагается переводить граф БП в программу на языке АФС, и затем ее исследовать на различные свойства.

Ключевые слова: верификация бизнес-процессов, конфликты в графах, язык АФС.

Современный экономический мир ставит перед компаниями новые стандарты по эффективности их работы. Чтобы быть успешным на рынке, зачастую недостаточно производить хороший продукт или оказывать хорошую услугу. Так же необходимо иметь адекватный подход к управлению, который можно рассматривать как совокупность бизнес-процессов (часто параллельных). В бизнес-процессах (БП) могут возникать различные конфликты при доступе к ограниченным ресурсам, процессы блокировки, зацикливания, проблемы синхронизации и т. д. Поэтому хорошо бы иметь возможность проверки этого БП на различные свойства до его внедрения в бизнес компании.

Покажем, как представить БП в виде графа, который можно перевести (транслировать) на язык асинхронно функциональных схем (АФС). Далее можно проводить анализ БП на интересующие нас свойства.

В нашем случае граф БП может состоять из множества элементов [2], представленных на рис. 1.

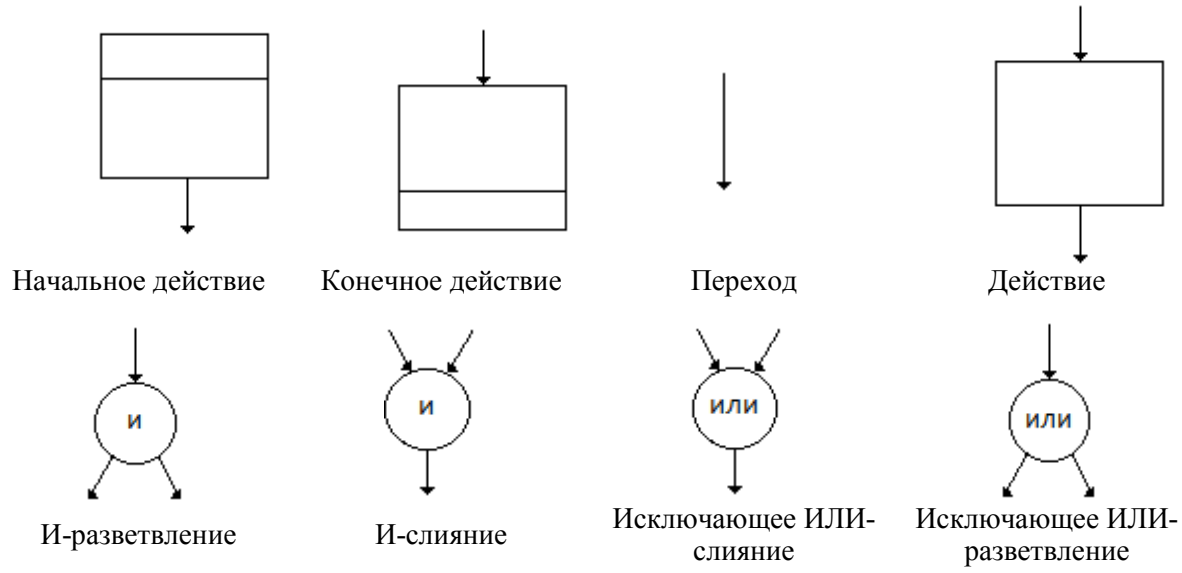


Рис. 1. Элементы графа БП

Элемент «И-разветвление» предназначен для разбиения потока на два или более независимых потока. «И-слияние» позволяет объединить параллельные потоки в один. Процесс продолжается, когда все входные потоки прибудут.

Элемент «исключающее ИЛИ-слияние» предназначен для объединения взаимно исключающих альтернативных потоков в один поток. «Исключающее ИЛИ-разветвление» имеет два или более исходящих перехода. Во время выполнения процесса на данном элементе осуществляется выбор, по какому из исходящих переходов продолжится процесс далее.

Теперь приведем краткое (местами неформальное) описание основных элементов языка АФС. Этого будет достаточно для понимания смысла. Подробное описание языка приведено в [1].

Множество АФС-программ Prog с типичным элементом pr определяется следующим образом:

pr ::= NET {can} BEGIN fproc END
can ::= CHAN j::type(k):type(l) | can1 ; can2
type ::= ALL | ANY
fproc ::= FUN i::c | fproc1 ; fproc2
c ::= a | skip | exit | break | wait(time) | read(i, l) | write(i, k) | SEQ(c {, c}) |
PAR(c {, c}) | ALT(gc) | LOOP(ALT(gc))
gc ::= g → c | gc {; gc}
g ::= tt | ff | b | wait(time) | read(i, l) | write(i, k)

Канал типа ALL(k):ALL(l) вначале принимает данные по всем своим k входам, после чего становится готовым для передачи данных по всем l выходам. Канал освобождается после передачи данных по всем своим выходам.

Канал типа ALL(k):ANY(l) аналогичен по входу каналу типа ALL(k):ALL(l). Канал освобождается после передачи данных по любому из выходов.

Канал типа ANY(k):ALL(l) становится готовым для передачи данных, если он принял данные по одному из своих входов, и освобождается после передачи данных по всем своим выходам.

Канал типа ANY(k): ANY (l) становится готовым для передачи данных, если он принял данные по одному из входов, и освобождается после передачи данных по любому из своих выходов.

Пустая команда skip не приводит к выполнению каких-либо действий. ALT(gc{;gc}) – альтернативная команда, означающая выполнение одной из защищенных команд, перечисленных в скобках, для которой значения защиты истинны. LOOP(ALT(gc{;gc})) – команда цикла, завершающаяся, когда все защиты ложные или при выполнении оператора break.

Команда wait(time) задерживает выполнение функционального процесса на время, задаваемое выражением time.

Команда SEQ(c{, c}) означает последовательное выполнение команд, перечисленных внутри скобок. PAR(c{, c}) – параллельное.

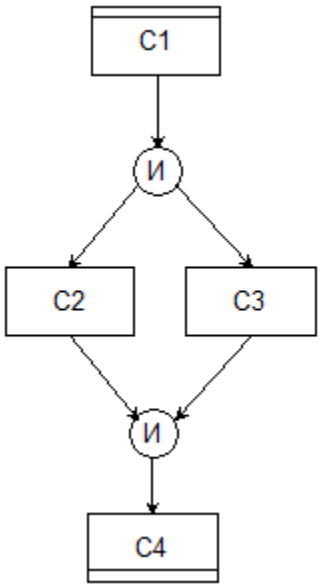
Команда чтения read(i, l) осуществляет запрос на прием данных из l-го выхода i-го канала, а команда write(i, k) осуществляет запрос на передачу данных на k-й вход i-го канала. Если l-й выход (k-вход) канала i готов к выдаче (приему) данных, то осуществляется чтение данных из канала (запись данных в канал), после чего функциональный процесс продолжает свое выполнение. В противном случае выполнение функционального процесса задерживается.

Выполнение АФС-программы предполагает одновременный запуск на выполнение всех функциональных процессов. В начале выполнения программы каналы не содержат данных и готовы к приему информации. АФС-программа

завершается, если все функциональные процессы находятся в пассивном состоянии. Функциональный процесс находится в пассивном состоянии, если он либо завершил свое выполнение, либо находится в состоянии ожидания ввода или вывода информации. Заметим, что выполнение команды wait(time) хотя и приостанавливает выполнение процесса, не означает, что процесс находится в пассивном состоянии. Завершение всех функциональных процессов означает успешное завершение АФС-программы. В противном случае имеет место блокировка.

Итак, у нас есть граф БП, который следует перевести в язык АФС. До перевода в язык АФС необходимо четко представлять, на какие свойства будем анализировать граф БП. Это позволит нам при переводе абстрагироваться от ненужных нам элементов. Обычно для анализа на различные свойства абстрагироваться следует по-разному, учитывая то одни, то другие элементы. В этом случае имеет смысл сделать несколько вариантов абстракций, и выполнить перевод в язык АФС необходимое количество раз. В итоге можно получить несколько программ на языке АФС, анализ каждой из которых даст ответ на наличие или отсутствие проверяемого свойства в исходном графе БП. Легче проверять две-три простых программы, чем одну большую.

Рассмотрим пример перевода графа БП в язык АФС (рис. 2). Выполнение программы начинается с запуска четырех функциональных процессов (ФП). Первый ФП выполнит действие C1 и запишет информацию в первый вход первого канала. В момент запуска программы второй, третий и четвертый ФП ожидают получения входной информации. Когда первый ФП выполнит запись в канал 1, канал отдаст полученную информацию по двум своим выходам. И одновременно запустится второй и третий ФП. Когда они отработают, они запишут в два разных входа второго канала информацию. Получив по всем своим входам информацию второй канал передаст информацию в четвертый ФП. И в нем выполнится конечный оператор, на этом выполнение программы остановится.




```
NET
  CHAN1::ALL(1):ALL(2)
  CHAN2::ALL(2):ALL(1)
BEGIN
  FUN1::SEQ(C1; write(1,1))
  FUN2::ALT(read(1,1)→SEQ(C2;write(2,1)))
  FUN3::ALT(read(1,2)→SEQ(C3;write(2,2)))
  FUN4::ALT(read(2,1)→C4)
END
```

Рис. 2. Пример графа БП и соответствующей ему программы на языке АФС

Если в графе БП содержится «исключающее ИЛИ-разветвление», то оно на языке АФС будет представляться каналом типа ALL(1):ANY(n). Где n – количество выхо-

дов. Соответственно элемент «исключающее ИЛИ-слияние» будет представляться каналом типа ANY(n):ALL(1), где n – количество входов канала.

После получения программы на языке АФС можно проводить исследование на интересующие нас свойства.

Библиографические ссылки

1. *Кораблин Ю. П.* Семантика языков распределенного программирования. М. : Изд-во МЭИ, 1996.
2. *Коробков К. Н.* Исследование и разработка системы управления бизнес-процессами, автореф. дис. … канд. техн. наук: 05.13.11. М. : РГСУ, 2008.