АЛГОРИТМИЧЕСКИЕ СЕТИ КАК ВИЗУАЛЬНЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ

О. Ф. Королев

Санкт-Петербургский институт информатики и автоматизации РАН
199178, Санкт-Петербург, 14-я линия ВО, д.39
<korolf@rambler.ru>

УДК 681.3.06
О. Ф. Королев. **Алгоритмические сети как визуальный язык программирования** // Труды СПИИРАН, Вып. 2, т. 2. — СПб.: Наука, 2005.
Аннотация. *Предлагается расширение возможностей аппарата алгоритмических сетей путем сравнения его с классическими языками программирования.* — Библ. 4 назв.

UDC 681.3.06
O. F. Korolev. **The Algorithmic Networks as Visual Programming Language** // SPIIRAS Proceedings. Issue 2, vol. 2. — SPb.: Nauka, 2005.
Abstract. *The enhancement of algorithmic networks is offered through comparing with classic programming languages.*. — Bibl. 4 items.

Введение

Предложенный в лаборатории автоматизации моделирования СПИИРАН формализм алгоритмических сетей (АС), с успехом применяется в практике моделирования уже в течение более двадцати лет [1, 2]. На его основе разработано несколько версий систем автоматизации моделирования (последняя версия – система КОГНИТРОН [1, 2, 3]).

АС как формализм представления моделей ориентирован на конечного пользователя. С одной стороны, он должен давать возможность пользователю понять модель и сопоставить ее с моделируемым процессом, с другой стороны, обеспечить требуемую точность моделирования. АС является формализмом для представления алгоритмических моделей. То есть моделей, представленных в виде некоторого алгоритма, структура которого подобна структуре причинно-следственных связей в принятом сценарии моделируемого процесса, каждая переменная алгоритма проинтерпретирована в терминах предметных переменных. АС является формализмом представления алгоритмов и может рассматриваться как специальный язык программирования высокого уровня, предназначенный исключительно только для описания функциональной спецификации задачи.

АС [1, 2] – это ориентированный нагруженный граф, вершинам которого соответствуют операции или функции, дугам переменные, каждой вершине соответствует только одна функция или операция, одной дуге соответствует только одна переменная, но одна переменная может соответствовать нескольким дугам. Ограничений на типы функций и переменных нет. Входные дуги в вершину соответствуют аргументам функции в вершине, выходные результатам, дуга между вершинами указывает, что результат одной вершины используется в качестве аргумента в другой. Выделяются два основных типа вершин: вершины с операторами задержки и все остальные. Вычисления в вершине, не содержащей оператор задержки, производятся сразу, как стали известны значения всех входных ее переменных, срабатывание вершин с операторами задержки происходит только после того, как во всех остальных вершинах выпол-

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

нены все вычисления. Срабатывание происходит одновременно во всех вершинах с операторами задержки и переводит модель на следующий шаг моделирования, далее процесс повторяется. Это обеспечивает синхронизацию процесса вычислений. Вершина с оператором задержки осуществляет задержку появления на выходе значения входной переменной на один шаг моделирования. Состояние всех вершин с операторами задержки определяет состояние модели на шаге моделирования. В АС недопустимы контура, не содержащие вершины с операторами задержки, переменная может быть вычислена только в одной вершине. Переменные, не вычисляемые ни в одной из вершин АС, – входные переменные АС и должны быть заданы для каждого шага моделирования, значения выходов задержки задаются для первого шага моделирования как исходное состояние системы. АС может быть проинтерпретирована как графическая система представления рекуррентных выражений с функциями и переменными произвольного типа. Условие останова процесса моделирования вынесено за пределы АС и отслеживается в специальном шаблоне, к которому присоединяется синтезированная по АС расчетная программа, реализующая тело цикла моделирования.

Однако при этом возможности применения аппарата ограничены. Предлагалось много различных подходов для расширения границ применения АС [2, 3]. Тем не менее, АС как язык представления алгоритмов не имеют некоторых возможностей классических, текстовых языков программирования. Рассмотрим основные положения классических языков программирования и их возможную реализацию в АС.

1. Типы данных. Организация порядка вызова операторов

Исторически в разработанных на основе АС системах автоматизации моделирования использовались переменные, соответствующие вещественным скалярам. Предлагалось также матричное расширение типов данных [2].

В новой версии системы КОГНИТРОН [3] допускаются типы данных вещественный скаляр и одномерный массив вещественных чисел. В принципе, путем преобразования индекса его можно использовать как двух- и более мерный. Это осуществлялось при построении модели для решения системы линейных алгебраических уравнений [3].

Перспективным может оказаться реализация на АС строк и указателей.

В [3] был введен дополнительный строковый параметр оператора, широко используемый для задания имени модели в операторе ссылки на другую модель. В принципе данный параметр можно использовать для хранения строковых констант.

Теоретически возможна интерпретация переменных модели как указателей на динамическую память, выделенную специально введенными операторами. При этом, в зависимости от вида операторов это могут быть указатели на массивы вещественных чисел или символов. Тогда возможно организовать работу со строками следующим образом:

1) Вводится оператор получения строки `ns (new string)`. Вход — размер строки, выход указатель на строку. Оператор выделяет память под строку и инициализирует ее (вводится пользователем, читается из файла и т.д.).

2) Вводится оператор поиска подстроки в строке `sch (search)`. Первый вход — указатель на строку, в которой ищем, вторая — указатель на искомую строку. Возвращает указатель на найденную строку, либо нуль.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Поиск подстроки может осуществляться с помощью АС, изображенной на рис. 1.

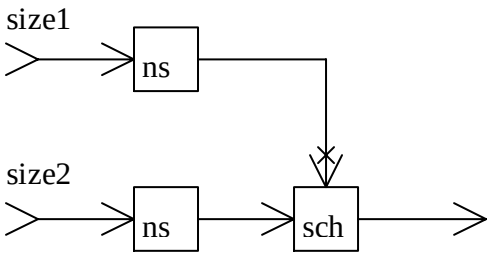

Рис. 1. Поиск подстроки.

Для АС строится план вычислений. Оператор не будет вызван, пока не вычислены все входящие в него переменные. Это дает возможность управлять порядком выполнения операторов. Например, возможно введение оператора генератора случайных чисел. В принципе данный оператор не имеет входа, однако если его необходимо вызвать после вычисления других операторов, мы можем задать ему формальный вход. Таким образом, в АС переменные могут иметь смысл управляющих сигналов.

2. Реализация условий

Рассмотрим реализацию условий в новой версии языка АС на примере аналогичных конструкций языка программирования С, в котором для этой цели используются условная операция и условный оператор.

2.1. Условная операция

Приведем аналог условной операции

$$x?y:z,$$

при x, не равном нулю, возвращающей значение y, в противном случае z. Аналогом данной операции является оператор вида, приведенного на рис. 2.

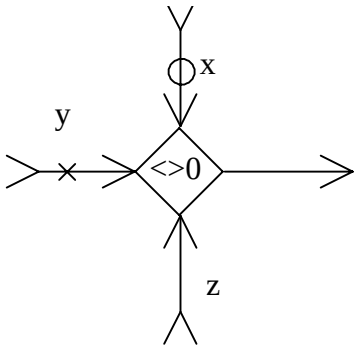

Рис. 2. Реализация условной операции.

При x, отличном от нуля, он возвращает y, иначе z. Подобные операторы реализовывались и в предыдущих версиях системы моделирования на основе АС [1].

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

2.2. Оператор условия

Однако операторов типа, приведенного на рис. 2, недостаточно для реализации полноценного классического оператора условия вида:

```
If(условие)
      Последовательность действий 1,
Else
      Последовательность действий 2.
```

Ранее, в [3] был предложен оператор ссылки на другую модель. Значение дополнительного параметра данного оператора имеет следующий формат:

```
:file<имя файла модели>:data<имя массива данных>:cp<число
периодов>:in<x1, y1; x2, y2; … xn, yn>:out< x1, y1; x2, y2; …
                          xn, yn >,
```

где параметр <:file> задает имя файла, в котором находится модель со своим шагом расчета; параметр <:data> задает имя набора данных вызываемой модели; параметр <:cp> задает число периодов, на которое считается модель; параметр <:in> задает через «;» список соответствия имен входных переменных в вызывающей и вызываемой моделях; параметр <:out> задает через «;» список соответствия имен выходных переменных в вызывающей и вызываемой моделях.

Пусть значение дополнительного параметра будет являться форматной строкой вида

```
<текст1>{x1}<текст2>{x2}….<текстn>{xn},
```

где вместо фигурных скобок будут подставляться значения переменных, указанных в них, вычисленные на момент вызова данного оператора. Эти значения могут подставляться в имя файла модели, имя массива данных или число периодов, на которое считается модель.

Тогда условный оператор можно реализовать с помощью оператора ссылки на другую модель, считаемую со своим шагом расчета (рис. 3):

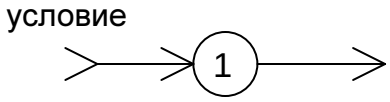

Рис. 3. Реализация условного оператора.

Значение дополнительного параметра будет являться форматной строкой вида

```
:file<model{условие}.cg>:data<…>:cp<…>:in<…>:out<…>.
```

Поскольку условие принимает значения 0 или 1, следует создать две модели — model0.cg и model1.cg. В зависимости от значения входной переменной будет вызываться либо одна, либо другая модель.

С помощью реализованного таким образом условного оператора возможно осуществить останов счета при достижении заданного условия. Пусть, например, модель model0.cg является пустой. Пока условие является истинным, будут выполняться действия, соответствующие model1.cg. Как только условие станет ложным, произойдет останов счета.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Следует отметить, что при реализации форматных строк дополнительного параметра встает теоретическая проблема синтаксического контроля создаваемых моделей, поскольку мы узнаем имя вызываемой модели только на этапе расчета модели.

3. Оператор выбора

Для реализации полноценного классического оператора выбора вида

```
switch(x)
<значение 1>
  Последовательность действий 1
…
<значение n>
  Последовательность действий n
```

достаточно подготовить модели `model1.cg` … `modeln.cg` и использовать оператор ссылки на другую модель со своим шагом расчета (рис. 4).

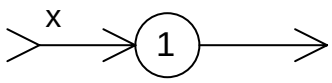

Рис. 4. Реализация оператора выбора.

Значение дополнительного параметра будет являться форматной строкой вида

```
:file<model{x}.cg>:data<…>:cp<…>:in<…>:out<…>
```

4. Вызов подпрограммы

В языке программирования C в качестве подпрограммы используется функция, допускающая пустые аргументы и результат. Аналогом вызова функции на алгоритмических сетях является ссылка на другую модель, считаемую со своим шагом расчета (см. 2.2). Дополнительный параметр задает имя вызываемой модели и соответствие внешних и внутренних переменных, т.е. соответствие формальных и фактических параметров.

К сожалению, из-за принятого на данный момент ограничения на количество дуг оператора (общее число входящих и выходящих дуг не более восьми) ограничивается число возвращаемых значений. Для передаваемых переменных такой проблемы нет, поскольку они формально могут и не являться входами оператора. Несколько снять проблему числа возвращаемых значений могут массивы. Возвращаемые значения запаковываются в массив, который анализируется вызывающей АС.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

5. Организация циклов

Организация циклов, пожалуй, является слабым местом АС. Первоначально АС позволяли с помощью контура, содержащего оператор задержки, организовать один внешний цикл (см. введение).

Для реализации вложенных циклов было предложено использовать операторы ссылки на модель со своим шагом расчета [3, 4] — происходит передача управления другой модели, число шагов моделирования которой равно числу шагов вложенного цикла (см. 2.2). Использование такого оператора дало возможность реализовать аналог вызова подпрограммы (см. 4), а с определенными допущениями операторы условия и выбора (см. 2.2, 3).

6. Операторы ввода–вывода

Ввод и вывод данных является важной составляющей языков программирования. Для его реализации введем два оператора.

1) `MessageBox` — оператор вывода. Определим его как имеющий обязательно один вход и один выход. Вход необходим для организации порядка вызова операторов АС (см. 1), а также для передачи параметра для выводимой строки (см. 2.2). Выход нужен для организации порядка вызова операторов АС. Выводимый текст (строковая константа) будет задаваться в дополнительном параметре оператора. В файле инициализации системы «Когнитрон» ему будут соответствовать строки

```
#out 1 1
#in 1 1
#calc1 msg(x1)
#extra 1
```

2) `InputBox` — оператор ввода. Определим его как имеющий обязательно один вход и один выход. Вход необходим для организации порядка вызова операторов АС, а также для передачи параметра для выводимой при вводе подсказки (задается в дополнительном параметре оператора). Выход нужен для получения введенной переменной. В файле инициализации системы «Когнитрон» ему будут соответствовать строки

```
#out 1 1
#in 1 1
#calc1 inp(x1)
#extra 1
```

7. Прочие расширения возможностей АС

Возможности языка АС можно увеличить, работая по следующим направлениям:

1) введение в алгоритмический базис операторов для работы с файлами: открытие файла, запись строки, чтение строки, запись файла и т.д.;
2) реализация построения простейшего диалогового интерфейса (диалоговые окна, вывод графиков с результатами расчета);
3) введение оператора вызова внешних приложений.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

8. Пример АС, реализующей игровую программу

В качестве примера, в котором использованы предложенные идеи, приведем АС, реализующую игру в 23 спички. Сначала имеется 23 спички. Можно взять 1, 2 или 3 спички. Пользователь ходит первым, затем компьютер. Проигрывает игрок, взявший последнюю спичку.

Главная модель изображена на рис. 5.

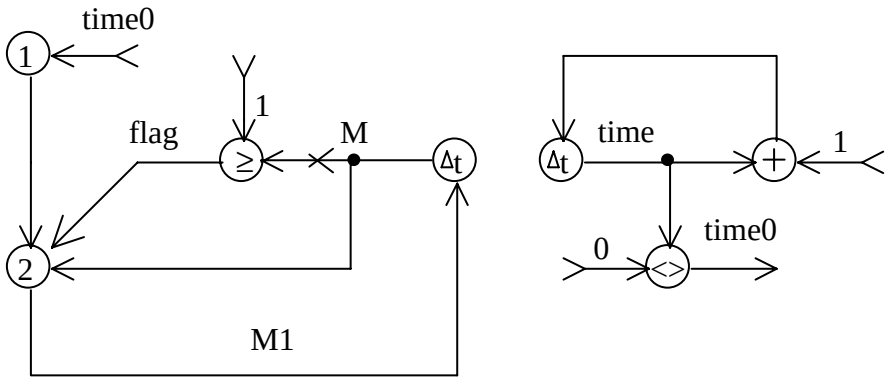

Рис. 5. Игра в 23 спички.

Число спичек задается переменной `M`. Начальное приветствие и правила игры задаются оператором 1. Его дополнительный параметр равен

```
:num<1>:file<models/matches/start{time0}.cg>:data<data>:cp<1>:
                              in<>:out<>
```

Модель `start0.cg` выводит правила игры (рис. 6), `start1.cg` является пустой.

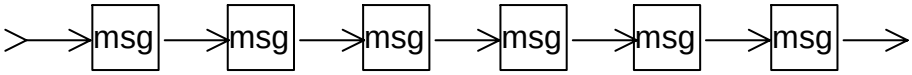

Рис. 6. Вывод правил игры в 23 спички.

Оператор 2 имеет следующий дополнительный параметр

```
:num<2>:file<models/matches/model{flag}.cg>:data<data>:cp<1>:i
                          n<M,M>:out<M1,M1>
```

Модель model1.cg отвечает за основные действия (рис. 7), `model0.cg` является пустой и вызывается тогда, когда спички кончились (останов счета).

На рис. 7 при необходимости оператор `wn` выводит сообщение о выигрыше компьютера (создается модель с сообщениями и пустая модель), оператор `fl` выводит сообщение о проигрыше компьютера (создается модель с сообщениями и пустая модель), оператор `st` выдает сообщение о количестве спичек, взятом компьютером (если человек не взял последние спички). Также имеются операторы `msg` для вывода текущего количества спичек и `inp` для ввода числа спичек, взятых пользователем. Оператор `rnd` генерирует случайное число.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Заключение

Перечислим основные предложенные в статье новшества:

1) работа со строками,

2) выполнение в зависимости от условия нужной модели,

3) останов счета при достижении результата,

4) запрос действий пользователя и вывод подсказок,

5) новые возможности в представлении результатов (вывод информации, запись в файлы, вызов внешних программ).

В результате алгоритмические сети превращаются в полноценный визуальный язык функционального программирования, что позволит существенно расширить область применения АС и повысить эффективность расчета.

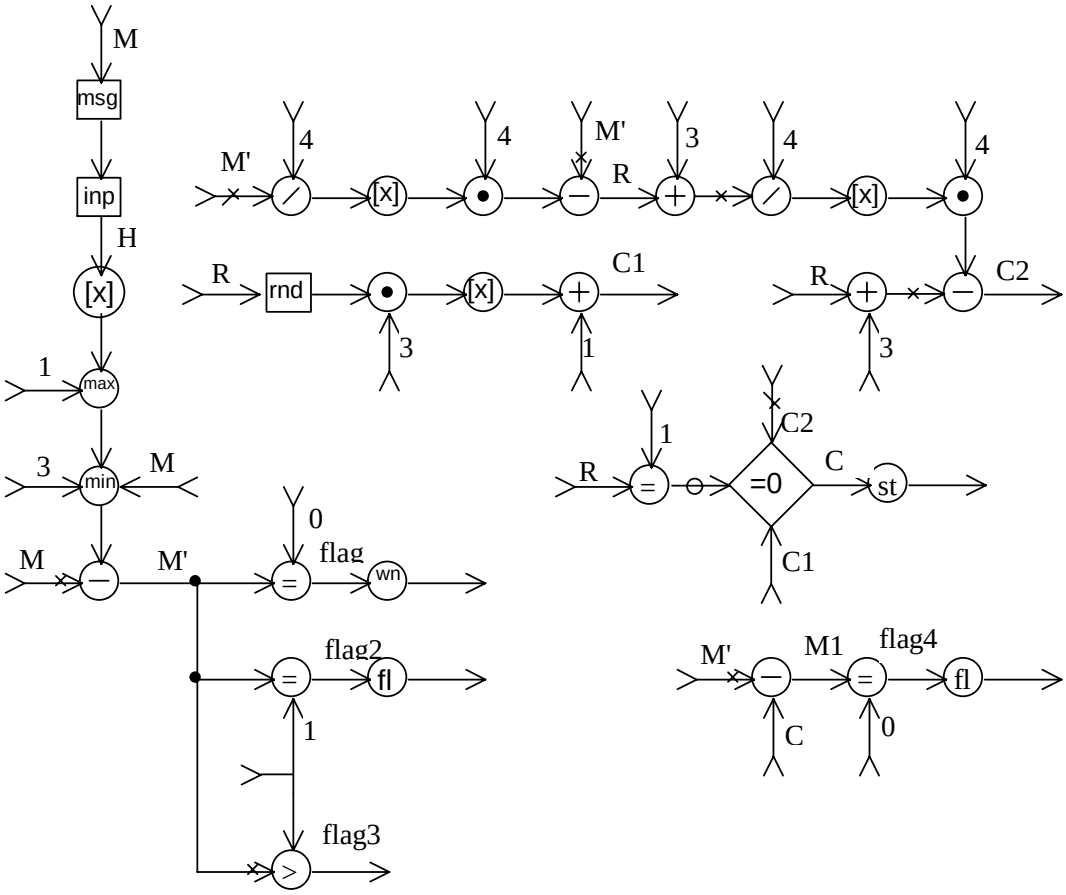

Рис. 7. Основной шаг игры в 23 спички.

Литература

[1] *Иванищев В. В., Марлей В. Е.* Введение в теорию алгоритмических сетей. СПб.: Изд-во СПбГТУ, 2000. — 180 с.

[2] *Иванищев В. В., Михайлов В. В.* Автоматизация моделирования экологических систем. СПб.: Изд-во СПбГТУ, 2000. — 172 с.

[3] *Королев О. Ф.* Методы графического представления моделей на основе алгоритмических сетей и их программная реализация. Диссертация на соискание ученой степени кандидата технических наук, 2003.

[4] *Янив Р. И.* Организация вычислений на алгоритмических сетях. // Информационные технологии и интеллектуальные методы. Выпуск №3. — СПб.: СПИИРАН, 1999, с.140–146.