Жаринов В.Н.

ОПИСАНИЕ ДЕЯТЕЛЬНОСТИ НА ОСНОВЕ МЕТОДОЛОГИИ ДРАКОН

Вводный цикл (извлечение)

Версия 09.1

ВВЕДЕНИЕ В ДОКУМЕНТ

Общие положения

1. Файл содержит выполняемый автоматизированным способом (в форме машинного оригинала МО) беловик целевого документа или его части (неотъемлемой), выделенной для удобства работы.

Документ в целом, кроме основного содержания, может включать приложения. Содержание документа, приложения (его выделенной части) составляют текст и/или иллюстрации (графчасть).

Конкретное наполнение файла определяется по его имени (полный формат имен см. шаблон документа)[1].

2. Содержание документа, приложения подразделено на структурные элементы по иерархии; её высшие 4 уровня стандартны. Элементы обычно имеют многоуровневую нумерацию и заголовки-абзацы, входящие в оглавление; возможны также элементы без нумерации, в т.ч. не входящие в оглавление, в т.ч. с заголовками в тексте.

В тексте применяются типовые приемы оформления, описанные в п/р 1.1 документа|шаблона.

3. В файл части из документа, приложения выделяется элемент структуры стандартного уровня иерархии (или ряд соседних элементов одного уровня) целиком (с заголовками).

Для многофайлового МО в имени каждого файла указаны индексы входящих элементов (формат: разделы <ЧN>, подразделы <пPNN>, пункты <ПNNN>, подпункты <пПNNNN>); файл первой части является *головным*.

При наличии приложений их форму (способ выполнения) указывают в отметках о наличии в составе единственного (или головного) файла основного документа (виды способов и формат отметок см. шаблон).

Приложения в МО могут выполняться как отдельные файлы *ПрилN* (что указывается в их отметках о наличии).

При наличии иллюстраций в документе, приложении (части) они также м.б. выполнены разными способами. Подрисуночные подписи включаются в оглавление для удобства поиска рисунков в документе.

Иллюстрации в МО могут содержаться в отдельном файле графчасти *Рисунки*; тогда текст содержится в файле *Текст*, и в нём дублируется подпись к каждой иллюстрации по месту её упоминания для отсылки к графчасти.

4. Оригинал документа (части) выполнен как настоящий файл (имя см. поле внизу) и другие необходимые (детальный состав многофайлового документа см. п. 1.1.4 в <настоящем файле|головном файле *Ч.1 Введ.*>).

Текст подготовлен в среде OpenOffice.org Writer или иной программы, совместимой по файлам; иллюстрации выполнены в той же программе и/или иными средствами, включая захват машобразов для МО.

Подлинник выполняется как твёрдая копия с заменой и/или добавлением листов к твёрдой копии предыдущих версий, либо как машинный образ файлов оригинала по листам, с которого делаются твёрдые дубликаты.

Информация, содержащаяся в документе, получена из открытых источников, рассматриваемых автором как надёжные. Возможное наличие секретных, конфиденциальных, а равно иных сведений ограниченного доступа следует рассматривать как результат предположения на массивах открытых сведений. Имея в виду возможные человеческие и технические ошибки, автор не может гарантировать абсолютную точность и полноту приводимых сведений, и не несет ответственности за возможные последствия, связанные с их использованием.

---

[1] Переменные части текста даются как поля в '< >', заменяемые на описание; общая часть (корень) поля пишется как есть, а изменяемые части как '*'. Файлы МО с однокоренным именем относятся к одному элементу структуры.

[2] Федеральный закон № 230-ФЗ от 18 декабря 2006 г.

[3] Если цитата состоит полностью из сведений, цитирующих другой источник – дать ссылку на первоисточник.

Назначение, сведения о версиях, языковые соглашения

1. Документ предназначен для представления содержания страницы гипертекста (либо ряда страниц, на которые делится его содержание) в виде обычного инфордока.

2. Версии документа выпускаются по мере обновления содержания цикла.

3. В тексте употребляются следующие типовые обозначения и сокращения:

англ.	английский;
букв.	буквально;
в т.ч.	в том числе;
и т.д.	и так далее;
и т.п.	и тому подобное;
к.-л.	какой-либо;
напр.	например;
см.	смотри;
т.е.	то есть;
т. зр.	точка зрения;
т.о.	таким образом;
разд.	раздел (документа);
п/р	подраздел (документа);
п.	пункт (документа);
п/п	подпункт (документа);
пред.	предыдущий;
след.	следующий;
ТЛ	титульный лист;

Сведения о терминологии, обозначениях и сокращениях данного документа см. в п/р 1.3.

Оглавление

ПРИЛОЖЕНИЯ:

1. Терминология, сокращения и условные обозначения (файл *Прил.1 Консультации по актуальным инфортехам*).

2. Элементы языкового обеспечения визуализации деятельности (файл *Прил.2 Консультации по актуальным инфортехам*).

—-резервная страница оглавления-—

2. ВИЗУАЛИЗАЦИЯ ДЕЯТЕЛЬНОСТИ В ТЕХНОЯЗЫКЕ

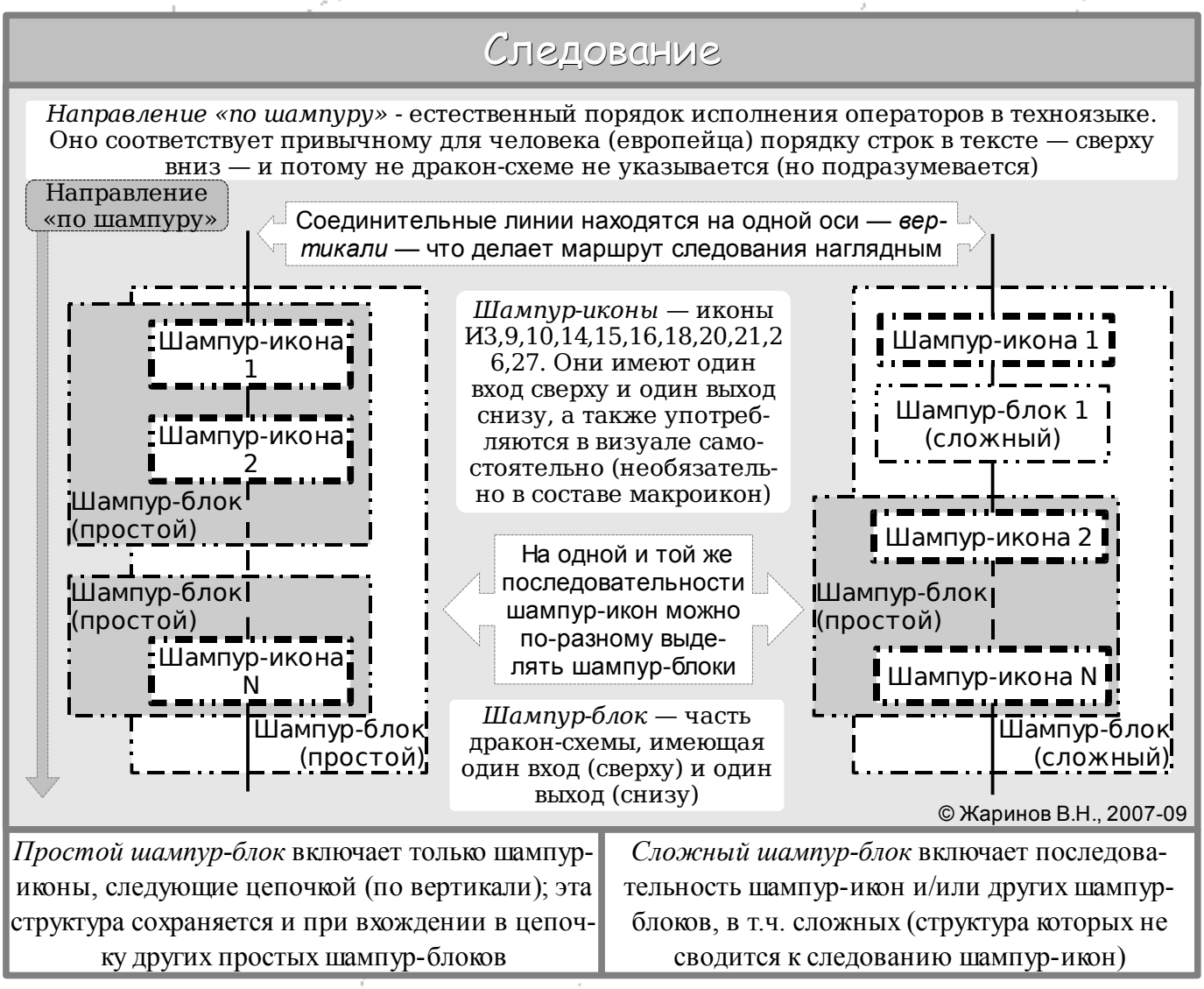
2.1. Основы ДРАКОН-информатики

Данная возможность не реализована. Пожалуйста, следите за обновлениями ресурса.

2.2. Базовые подструктуры деятельности

Для начала рассмотрим визуализацию последовательности операторов на техноязыке.

2.2.1. Следование



Следование в техноязыке – визуальное определение

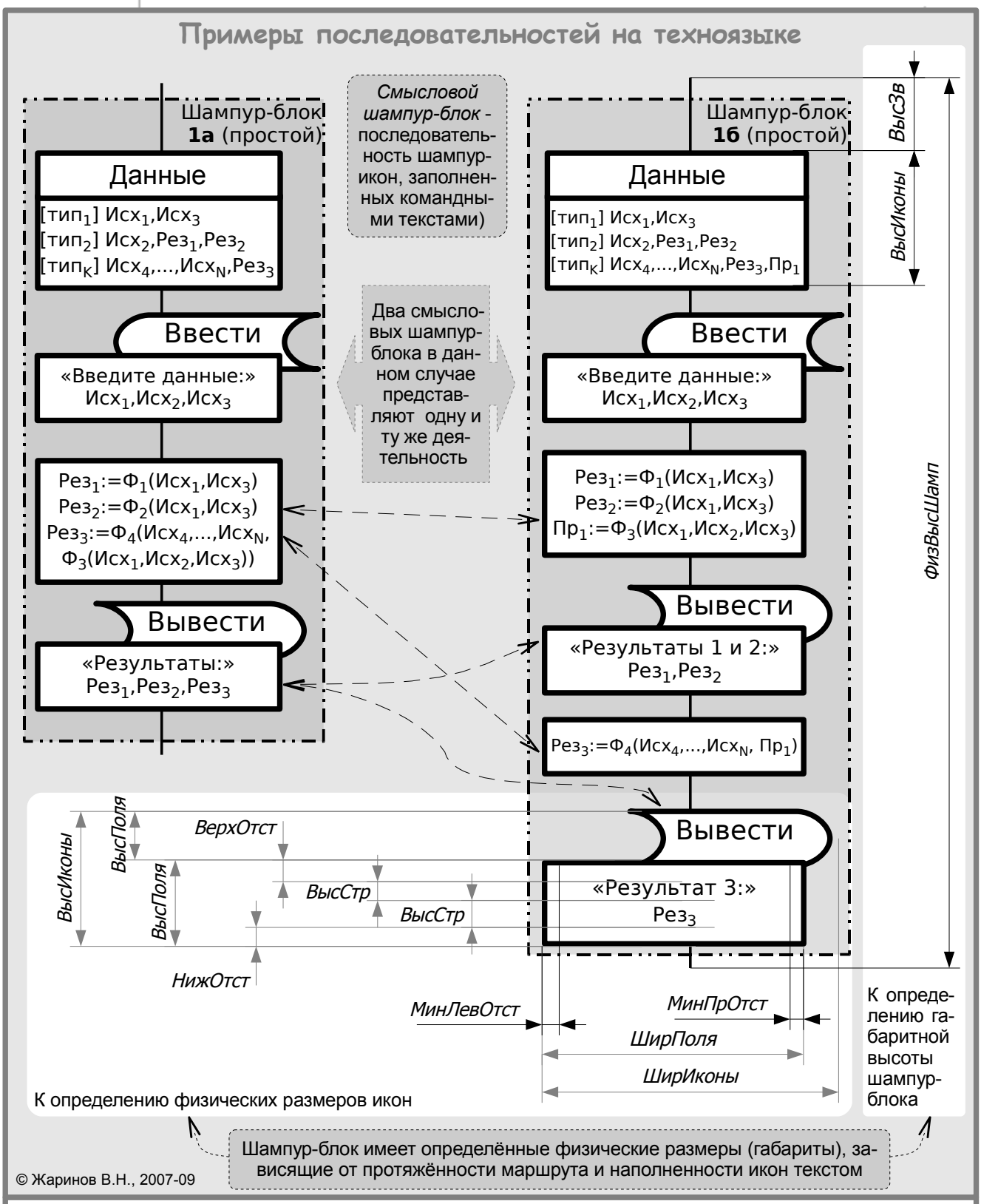
Последовательность неэлементарна потому, что в техноязыке она определяется как цепочка шампур-блоков, внутри которых содержатся не только шампур-иконы, но и другие шампур-блоки. Иначе, последовательность как цепочка икон ***Действие*** или других шампур-икон (соответствует следованию в блок-схемах) на техноязыке – только один из видов содержания шампур-блока (в частном случае она состоит из одной шампур-иконы); такой шампур-блок мы ранее в п. 2.1.2 назвали ***простым***. Там же мы определили и ***сложный*** шампур-блок; здесь мы можем уточнить, что его структура раскрывается как простые шампур-блоки, соединённые нешампур-иконами.

> Чем отличается простой шампур-блок от сложного? В первом мы можем выделять мелкие шампур-блоки вполне произвольно (с одним ограничением – чтобы границы их соприкасались, но не накладывались – можно только выделить более мелкий блок целиком внутри более крупного). Во втором этого недостаточно – мелкий блок не д.б. нешампур-блоком – т.е. разбиение ограничено структурой разбиваемого блока.

Далее мы будем иметь дело в основном со сложными шампур-блоками. Кроме того, в дракон-схемах формы силуэта допустимы блоки иного рода, называемые *адресными* – в которых вход по-прежнему один, но выходов два и более (подробно рассмотрены далее).

Итак, представление одной и той же деятельности в виде последовательности (цепочки действий) в общем случае неоднозначно; поэтому шампур-блок и принят за базовую конструкцию техноязыка в отличие от серии (композиции) в блок-схемах.

В качестве примера неоднозначности рассмотрим задачу вычисления некоторых величин на основании предварительно введённых исходных данных с последующей выдачей результатов.

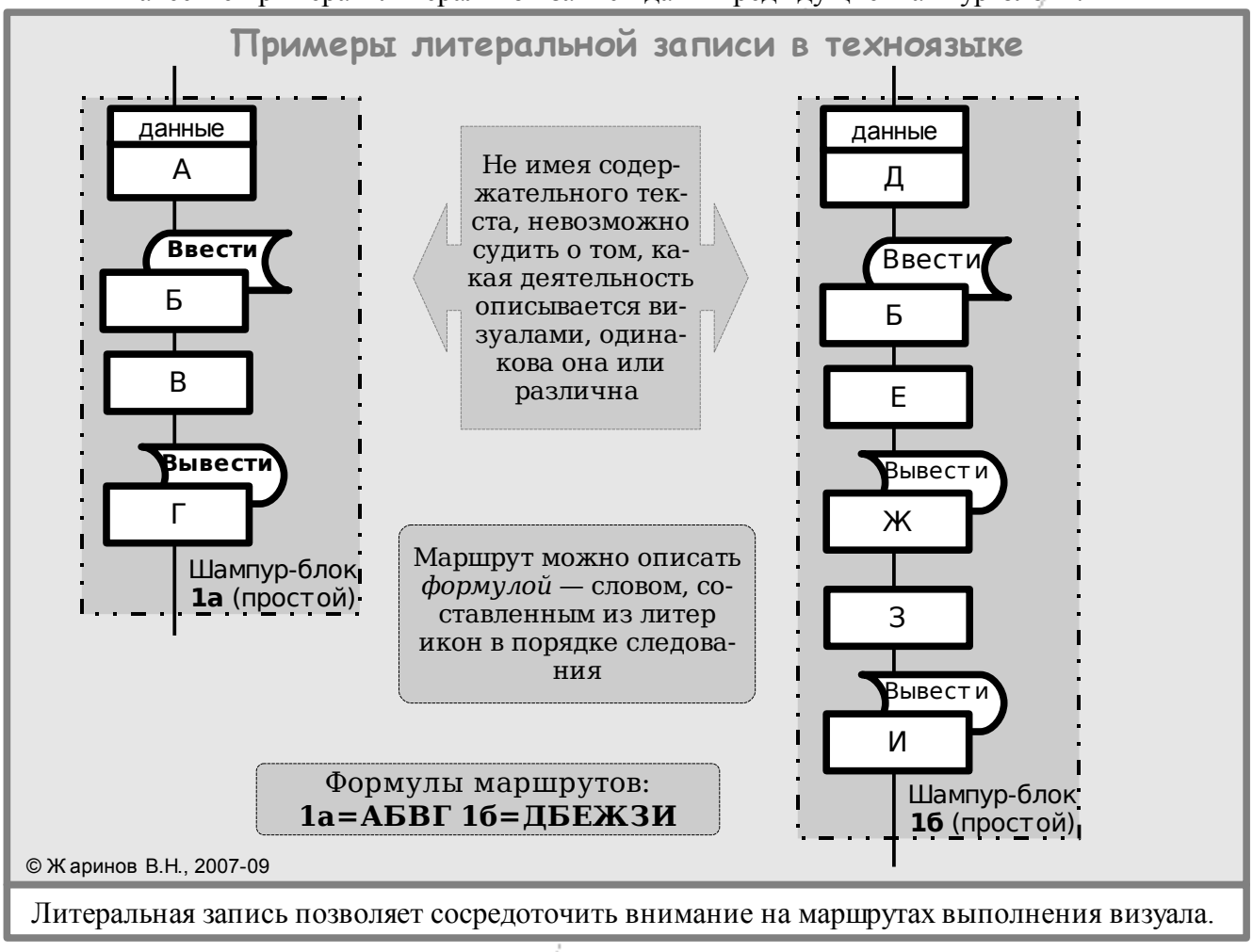


При алгоритмизации, как правило, для одной и той же *деятельности* - системы действий, ведущих к определённой цели (в данном случае - вычислить результаты на основании исходных данных) возможны варианты следования действий - смысловые шампур-блоки. В данном случае главное отличие двух вариантов заключается в сценарии вывода результатов исполнителем (напр. информашиной) во внешнюю среду (напр. оператору); одновременно мы изменили и порядок получения результатов (одно необязательно связано с другим).

Когда содержание икон неважно для работы над дракон-схемой, текст каждой иконы можно заменить литеральным обозначением (буквой некоторой азбуки), причем одинаковым текстам соответствуют разные буквы, а разным текстам – разные (то же делается, когда точные тексты икон ещё неясны). Такой вариант визуала называется *литеральной записью* (тогда как запись с содержательным текстом – *смысловой*).

Если разных текстов больше, чем букв азбуки, можно использовать аббревиатуры; для отражения к.-л. подразделения икон также можно усложнить правила обозначения (скажем, вводить цифровые или иные индексы, в т.ч. иерархические).

В качестве примера в литеральной записи даны предыдущие шампур-блоки.



Литеральная запись текста визуала

Литеральная запись экономит площадь за счет сокращения габаритов икон. Она позволяет сосредоточить внимание на структурах управления алгоритма, формально описать их.

> *Маршрут визуала* (линейного) – путь по дракон-схеме от начала до конца, проходящий через иконы и соединительные линии.
>
> *Формула маршрута* – последовательность литер, обозначающих иконы.

Также определим некоторые отношения между двумя произвольными алгоритмами.

> *Равносильные алгоритмы* – такие, у которых все маршруты попарно одинаковы (имеют совпадающие формулы).
>
> *Эквивалентные алгоритмы* – дающие одинаковые результаты при одних и тех же исходных данных.
>
> Равносильные алгоритмы всегда эквивалентны. Обратное утверждение неверно, в чем легко убедиться из вышеприведённых примеров (которые для этого утверждения служат контрпримерами, т.е. примерами-опровержениями).

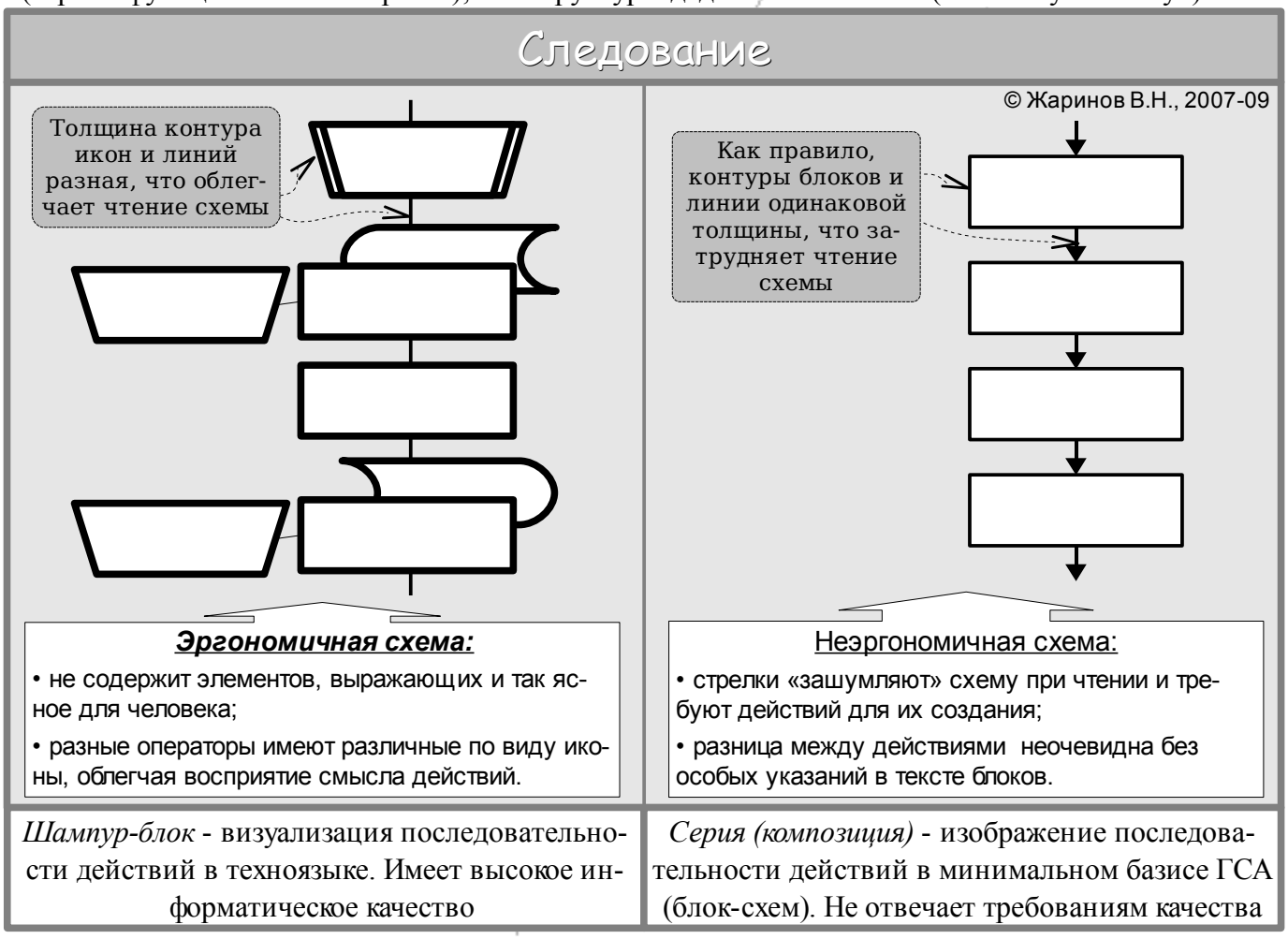
Пунктирные стрелки показывают эквивалентные преобразования линейного алгоритма – перераспределения действий по маршруту. Равносильное преобразование может состоять, напр. в замене

имен некоторых величин, функций в тексте всех икон; формула алгоритма от этого не меняется (поскольку тексты, которые были одинаковы до такой замены, остаются одинаковы и после неё, аналогично и различные тексты остаются различными).

По литеральной записи неравносильных алгоритмов невозможно установить, эквивалентны они или нет.

Идя далее в абстрагировании записи, можно совсем исключить из дракон-схемы текст; получается чертеж, называемый *абстрактной дракон-схемой* («*слепышом*») и содержащий только операторы маршрутного подъязыка ДРАКОНа. Абстрактная запись удобна для представления и сравнительного анализа структур управления алгоритмов независимо от их содержания, а также формализованного описания самого техноязыка.

Сопоставим теперь шампур-блок с линейной конструкцией традиционных блок-схем (серией функциональных вершин); обе структуры дадим как слепыши (см. «очную ставку»).



Следование – «очная ставка» на техноязыке и языке блок-схем

Маршруты отражают структуру управления алгоритма. Кроме того, формулы маршрутов в текстовой форме описывают ***топологию*** дракон-схемы (это термин, означающий математическое описание пространственного строения любого объекта).

Топология – важная и сложная отрасль математики; но нам нужны лишь её отдельные элементы, которые часто можно визуализировать для лучшего понимания.

Применительно к линейным визуалам можно говорить о такой топологической характеристике, как длина маршрута. Очевидно, у эквивалентных алгоритмов эта длина м.б. неодинакова, хотя результат их исполнения один и тот же по определению. Что же меняется? Число и порядок звеньев вертикали, а значит, и естественных переходов (каждому соответствует звено), и точек вставки (помещённых посредине звена), и различимых состояний исполнителя.

Ограничения на топологию дракон-схем описаны в стандарте техноязыка как часть текста отдельных тезисов (см. /1, Гл.15/). Главное ограничение для линейной структуры – *не до-*

пускается разрыв шампура на части между разными диосценами (напр. листами, на которых чертится дракон-схема сочиняемого визуала). Для этого физическая высота схемы (шампура вместе с границами) не должна превышать высоты рабочего поля диосцены.

Обсудим, как удовлетворить этому. В основном высота схемы зависит от шампура:

ФизВысШамп = сумма(ДлЗвена) + сумма(ВысЭлИконы), где:

ДлЗвена – длина звена вертикали; ВысЭлИконы – высота элемента иконы, причем:

ВысЭлИконы = ВерхОтст + (КолСтр * ВысСтр) + НижОтст, здесь:

ВерхОтст – отступ первой строки текста от верхнего контура иконы (её элемента).

НижОтст – отступ последней строки текста от нижнего контура иконы (её элемента).

КолСтр – число строк текста в очередном по высоте (единственном) элементе иконы;

ВысСтр – полная высота строки текста (для данного абзаца) в очередном элементе иконы, причём:

ВысСтр = Кегль * ИнтСтр, здесь:

Кегль – (типографское) принятая высота шрифта текста; измеряется по ядру строчных букв;

ИнтСтр – межстрочный интервал для абзаца текста в очередном элементе иконы.

Все эти величины вместе можно назвать ***вертикальными параметрами компоновки***; для *ДлЗвена*, *Кегль*, *ИнтСтр*, **Отст* минимальные значения задаются из эргономических соображений (читабельности текста).

Из алфавита видно, что отдельные иконы имеют несколько элементов-полей, в т.ч. перекрывающихся; для них ширина задаётся по одному из полей, принятому за базовое, а высотой считается вертикальный размер видимой части элемента.

Дракон-схема предназначена человеку т.е. имеет символическую форму, а основным её представлением является бумажный (печатный) документ. При типографском качестве отображения (печати) или близком к нему (напр. лазерной принтописи) кегль текста выбирают в пределах 10...12 пунктов (зависит и от выбранной гарнитуры шрифта); допустим межстрочный интервал от 10...20% кегля, а отступ первой строки – порядка 0,1 см.

Кроме кегля, на итоговый пиксельный размер букв текста влияет и гарнитура шрифта. В разных полях икон она м.б. различна для удобства восприятия; мы выбрали конкретные гарнитуры. Тогда величина *КолСтр* определяется ***горизонтальными параметрами компоновки***: шириной иконы (поля) и отступами текста от контура слева и справа:

ШирСтр = ШирПоля – (МинЛевОтст + МинПрОтст)

Обычно для одной дракон-схемы выбирают эти параметры постоянными.

Напомним, что при вводе текста в фигуры, как и на страницу, приложение редактирования производит т.н. *выключку* – автоматическое размещение текста в строках заданной ширины.

Алгоритм выключки в фигурах для типичного офисного пакета следующий. По завершении ввода в абзац (как «жёсткий», заданный нажатием *<Enter>*, так и «мягкий» – с сохранением стиля, заданный нажатием *<Shift>+<Enter>*) текст абзаца построчно, начиная с места ввода, перераспределяется так, чтобы суммарная ширина букв в строке (включая знаки пробела) была меньше *ШирСтр*; при этом, если включена расстановка переносов, то последнее слово переносится, а если для абзаца задано выравнивание по ширине, то каждый пробел также «разгоняется» настолько, чтобы дополнить нехватку ширины до *ШирСтр*; в итоге *КолСтр* может измениться, и габаритная высота текста определяется заново по вышеприведённому выражению для *ВысЭлИконы*. Далее, если в фигуре включено свойство *Подогнать под текст* , то высота фигуры меняется так, чтобы удовлетворить новому *ВысЭлИконы*; иначе выключенный текст остаётся в новой габаритной высоте, «недотягивая» до нижнего/верхнего краёв контура либо «выпирая» за них, а высоту контура нужно изменять вручную.

Итак, меняя кегль, межстрочный интервал и отступы для некоторого элемента иконы, мы можем в определённой степени влиять на число строк и высоту строки, а значит, и на высоту текста и всего элемента. Последняя строка чаще всего неполная, т.н. «хвост».

От заданных параметров и зависит минимальный физический размер конкретного шампура. В целях рациональной компоновки можно управлять размером, меняя некоторые параметры.

Исходя из эргономичности, в реальной дракон-схеме промежутки между соседними

иконами д. б. различимы, а отступы текста от контура иконы (её части) – заметны на глаз.

Если такая подгонка не даёт результата (схема не умещается на выбранном формате целиком), то в частном случае можно пойти на отступления от правил компоновки; однако в общем случае это невозможно и возникает проблема. Для её решения можно применить подстановку, но уже не как формальную операцию оформления инварианта, а как неформальный эргономический приём. В этом случае границы фрагмента, выносимого во вставку, определяются тем, насколько нужно сократить высоту шампура.

Традиционно (в блок-схемах) допускается разрывать линии между листами, вставляя вершины-соединители. Такую схему читать неудобно, да и сочинять сложно; поэтому и запрещены разрывы в техноязыке. Однако это справедливо для диосцены (двумерного пространства); а как дело обстоит в одномерном (логически) пространстве? Рассмотрим это далее.

2.2.1.1. Произвольное следование: выкладка линейной дракон-модели

Ранее мы рассмотрели содержание подстановки и иконы Вставка. Безусловный переход м.б. использован внутри шампура и более общим образом; покажем это.

Мы уже изобрели новые визуальные операторы для обозначения шампур-блоков и нелинейных фрагментов дракон-схем, использованные при анализе в п/р 2.1. Теперь изобретём обозначения для безусловного перехода с возвратом; они нам понадобятся для начала. Логично доработать БП-иконы техноязыка, введя по аналогии со вставкой вторые вертикальные линии контура. Текст иконы Адрес вставки (назовём её так) аналогичен по формату тексту иконы Вставка; у иконы Имя вставки текст составляется из текстов икон Заголовок и Формальные параметры по тому же принципу. Тем самым мы исключаем икону И11 из нашей записи.

Перерисуем дракон-модель из п/п 2.3.1.3 в такой нотации; результат см. схему далее.

По принципу упорядочения вертикали БП располагаются слева от шампура.

Очевидно, что можно считать полученные структуры частями одного шампура. Теперь введём линейный порядок частей шампура, т.е. выложим нашу модель в цепочку. Оказывается, это можно сделать не единственным образом; имеются разные варианты следования частей (т.н. перестановки). Число вариантов определяется комбинаторной формулой перестановок и для нашего случая (3 части) равно 6, мы покажем три (на рисунке далее).

Теперь представим себе, что мы проделываем то же самое, разделяя шампур на части уже обычными БП; в отличие от БП с возвратом, здесь не требуется сохранять/восстанавливать состояния, поскольку исполняется всё тот же алгоритм.

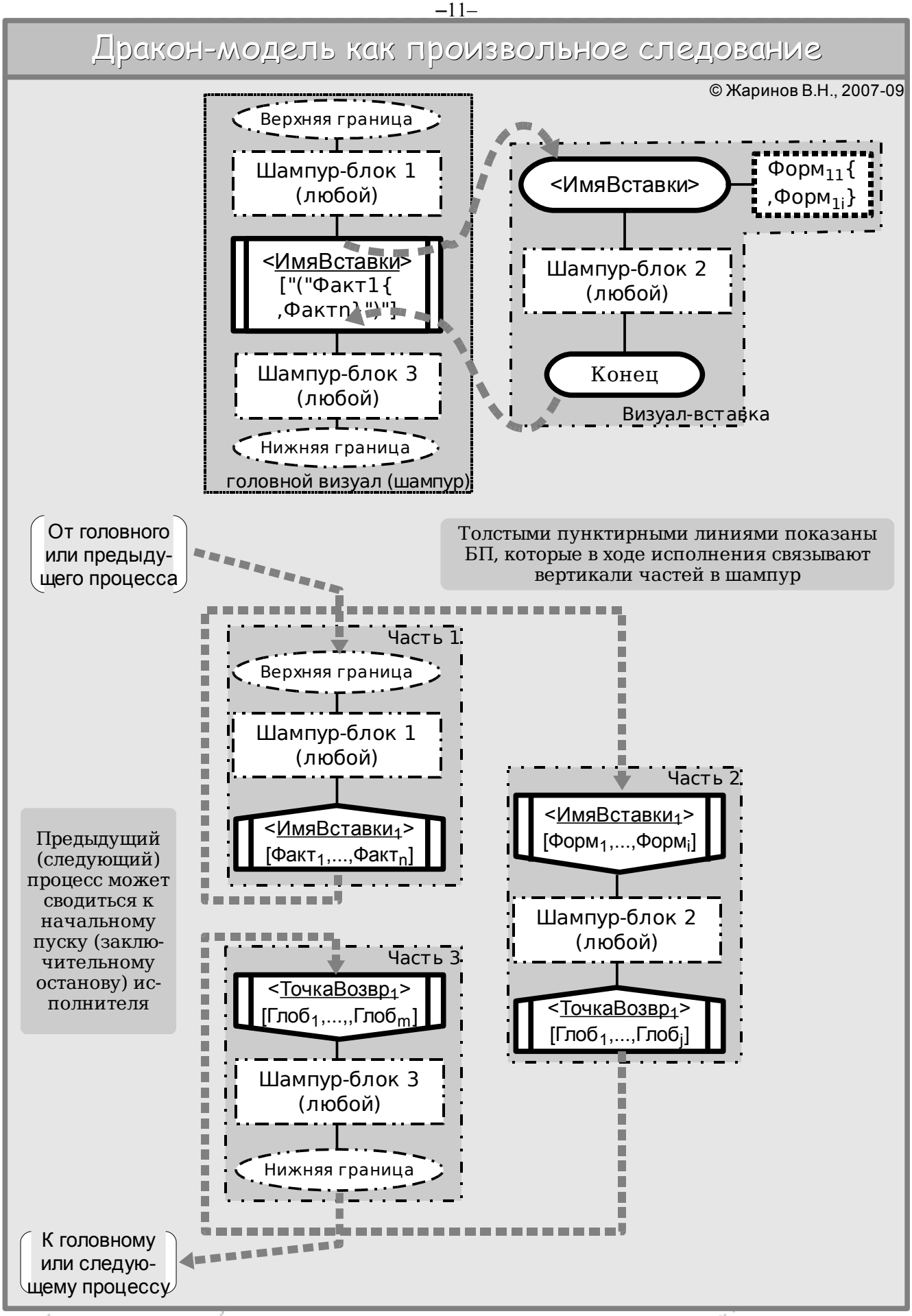
Эти переходы по содержанию эквивалентны совершаемым в петле силуэта разрешённым БП, но имеют иное назначение; поэтому изобретём обозначения и для них. Воспользуемся для иконы-метки графикой иконы Вариант, а для иконы-команды соответственно отразим эту фигуру по горизонтальной оси. Такое решение удобно будет при дальнейшем употреблении этого типа перехода.

С т. зр. классического техноязыка это вроде бы недопустимая схема: имеем линии, которые обрываются, не достигая иконы Конец (Пароннджанов называет их «хвосты»). Однако смысл безусловного перехода ставит всё на свои места; части можно рассматривать как аналоги визуалов-вставок, только без особенностей БП с возвратом. Правда, это уже не классический ДРАКОН; по сути, мы модифицировали шампур-метод.

Для чего это нужно? Вспомним, что выкладка согласует топологию алгоритма с пространственными свойствами формального исполнителя – линейностью и конечностью его памяти. При этом некоторые участки этой памяти м.б. уже заняты служебными величинами исполнителя, содержанием других алгоритмов.

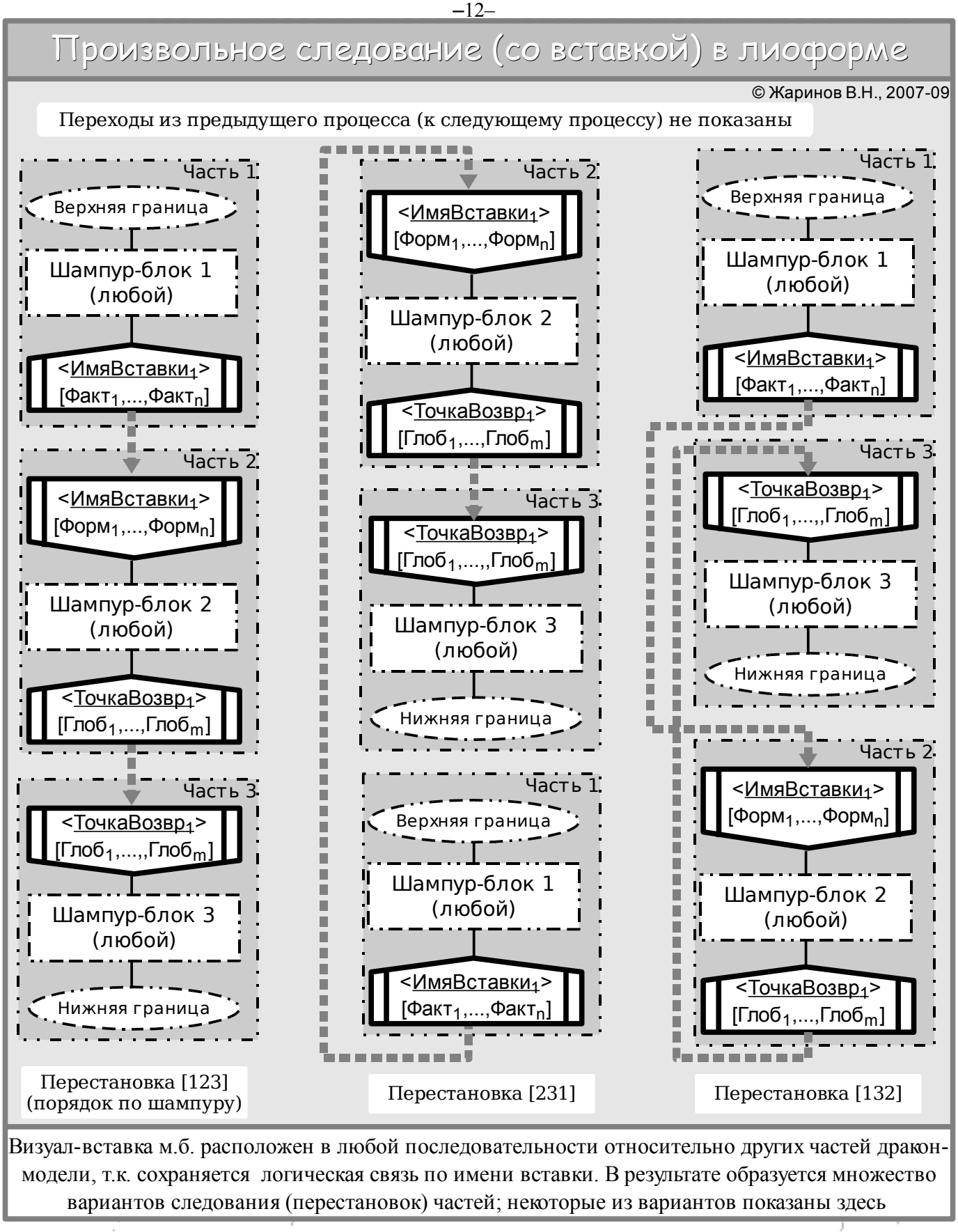
В общем случае заранее неизвестно, как расположатся эти участки и останется ли хотя бы одна свободная область, достаточная для размещения данного алгоритма. Значит, мы должны иметь возможность делить алгоритм на такие части, для каждой из которых найдётся место в памяти. Эту возможность нам и даёт введение в шампур обычных БП.

По сути, речь идёт о компоновке визуала в лиоформе и предметном представлении (где фигура и текст каждой иконы заменены машинными кодами действий, отвечающих её смыслу).



Безусловные переходы с возвратом в шампуре

Для выкладки каждая часть д.б. шампур-блоком (простым; над сложным вначале нужно выполнить некоторые операции, которых мы коснёмся далее).

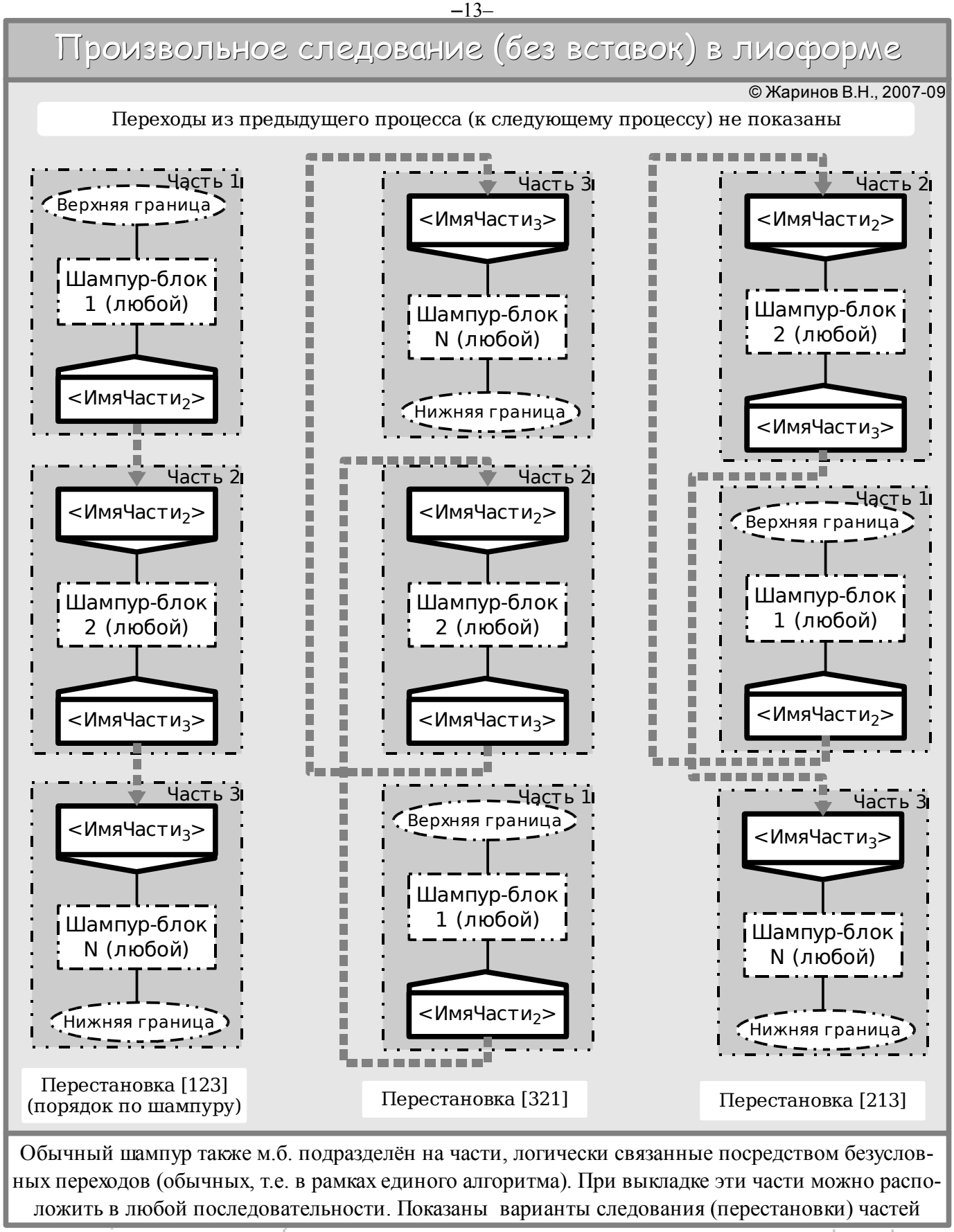


Визуал-вставка м.б. расположен в любой последовательности относительно других частей дракон-модели, т.к. сохраняется логическая связь по имени вставки. В результате образуется множество вариантов следования (перестановок) частей; некоторые из вариантов показаны здесь

Примеры выкладки шампура, содержащего безусловные переходы с возвратом

Здесь даны три перестановки; ещё две приведены на следующей схеме. Очевидно, читателю не представит труда обнаружить и изобразить единственную оставшуюся для самообразования:)

Это и есть случаи произвольного следования. Фактически мы объяснили принцип размещения машинного кода программы в основной памяти (ОП) информашины, не прибегая при этом к специфическим терминам и способам изложения, принятым в системном программировании; это лишний пример выразительной силы техноязыка (с введёнными выше знаками).



Обычный шампур также м.б. подразделён на части, логически связанные посредством безусловных переходов (обычных, т.е. в рамках единого алгоритма). При выкладке эти части можно расположить в любой последовательности. Показаны варианты следования (перестановки) частей

Примеры выкладки шампура, содержащего простые безусловные переходы

В такого рода структуре неважен и физический порядок следования частей в памяти друг относительно друга («как выложено»); адреса переходов логически свяжут части по порядку в шампуре («как сочинено»). Это даёт нам необходимую свободу выкладки при том, что объёмы частей м.б. различны; необязательно, что подходящие по объёму участки в памяти будут следовать в том же порядке, что и части выкладываемого шампура.

2.2.2. Ветвление

В основе определение ветвления в техноязыке совпадает с традиционным: это шампур-блок, содержащий одну или более развилок на основе икон ветвления, причем выполнение (или невыполнение) условия развилки ведет к передаче управления на тот или иной участок маршрута за ней. Условие зависит от обстановки выполнения алгоритма, т.е. включает к-л. из его величин. Структура связей внутри ветвления нелинейна (несводима к цепочке операторов).

Для нелинейных алгоритмов определение маршрутов будет более общим:

> *Маршрут визуала* – путь от начала до конца, проходящий через иконы и соединительные линии, причем при каждом ветвлении выбирается один из выходов (вариантов).
>
> *Формула маршрута* – последовательность литер, обозначающих иконы, и результатов проверки условий ветвления (ответов на вопросы развилок, вариантов переключателя).

Понятно, что нелинейный визуал будет иметь набор маршрутов, полученных перебором всех возможных сочетаний ответов для имеющихся икон ветвления. В совокупности все они отражают структуру управления алгоритма и по-разному относятся к его цели.

> *Главный маршрут* – ведущий к цели нелинейного визуала (тому, что надо по условию задачи, решаемой данным алгоритмом) наверняка либо с наибольшей вероятностью.
>
> Главный маршрут можно указать только для смысловой записи визуала.

Главное эргономическое требование к нелинейным визуалам: *главный маршрут должен проходить по шампуру;* тогда дракон-схему легче читать.

> Основные нелинейные структуры при визуальном анализе мы будем сопровождать компактными схемами связей, своего рода «маршрутными картами» из фигурных стрелок. По-прежнему на схемах не показываем, а подразумеваем точки ввода (нейтральные) в середине каждого звена вертикали.

Этот тип структуры в техноязыке имеет эргономические особенности записи. Начнём с **простого ветвления** (см. диосцену далее). Формально его структура распадается на развилку, шампур-блоки альтернатив и точку слияния, связанные естественными переходами друг с другом. Введём определения (см. /1, Гл.15, Тезис 26, 27/):

> *Лиана* – часть дракон-схемы, имеющая один вход – *начало* и один выход – *конец*. Началом лианы м.б. любой выход иконы *Вопрос* (выход любой иконы *Вариант*), если выход не соединён с петлёй цикла. Концом лианы м.б. точка слияния с другой линией, если эта точка не есть неразветвлённый вход иконы.
>
> *Нагруженная лиана* – последовательность икон/макроикон, звеньев вертикали и ломаных линий (связей между иконами, включающих наряду с вертикальными также горизонтальные участки и изломы, не являющиеся вершинами дракон-схемы).
>
> *Ненагруженная лиана* – не содержащая икон; есть звено вертикали или ломаная.
>
> Отметим, что главная вертикаль также попадает под определение лианы.

Очевидно, что ломаные линии возникают только у лиан, содержащих побочные вертикали. Число изломов линии д.б. минимально; оправданным считается один излом.

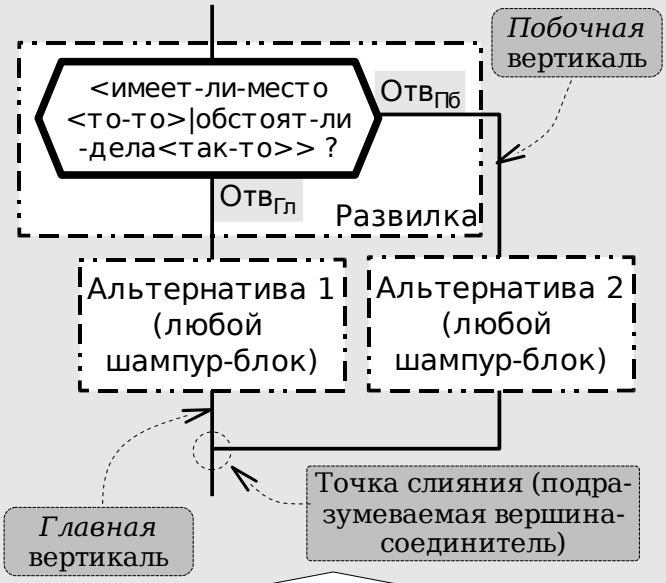
Сложное ветвление по-человечески естественно изображать как мультиграф, используя условную вершину со многими выходами (мультиикону, см. схему-прототип); в техноязыке можно остаться в рамках бинарных ГСА и визуализировать эту вершину как цепочку развилок (см. внизу справа), однако получим неэргономичную и к тому же сравнительно громоздкую, плохо компонуемую конструкцию. Чтобы устранить эти недостатки, был разработан переключатель (общий вид см. внизу слева). На «очной ставке» (см. далее) видно, что запись цепочки развилок на языке блок-схем ещё усиливает отмеченные недостатки.

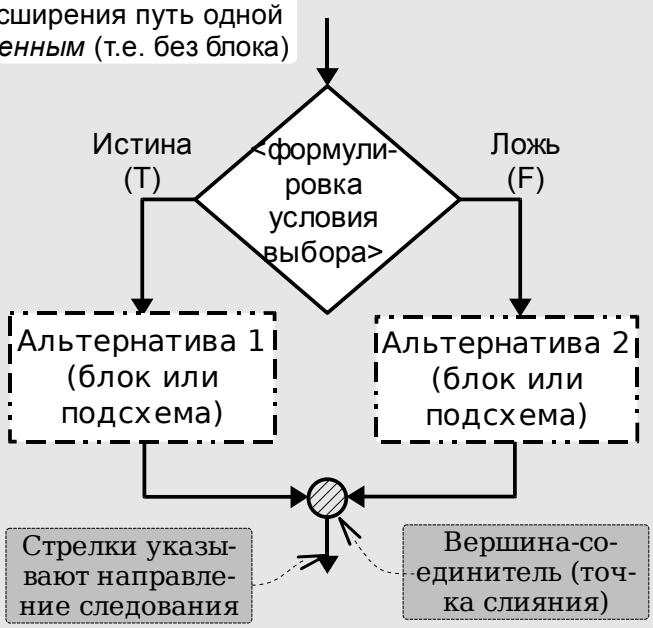
Из схемы множественного ветвления на базе мультиикоины «Вопрос» легко выводится сложная развилка в минимальном базисе ГСА – мультиикона преобразуется в дерево индивидуальных условий на каждое направление. Из неё выводится переключатель – дерево условий замещается системой рёбер и вершин, образующих «шапку» переключателя, а на схеме в явном виде от системы выражений выбора остаются только объявление переменной выбора и её требуемых значений.

Ветвление (простое)

Выбор одного из альтернативных продолжений маршрута происходит по результату проверки (исполнителем) условия ветвления (истинно или ложно при текущих значениях входящих величин)

По правилам структурного расширения путь одной из альтернатив м.б. *ненагруженным* (т.е. без блока)





Эргономичная схема:

- не содержит элементов, выражающих и так ясное для человека;
- различие (неравная значимость) альтернатив визуально очевидно за счет структуры;
- главная вертикаль всегда расположена по шампуру, что облегчает чтение схемы;
- формы блоков удобны для записи текста.

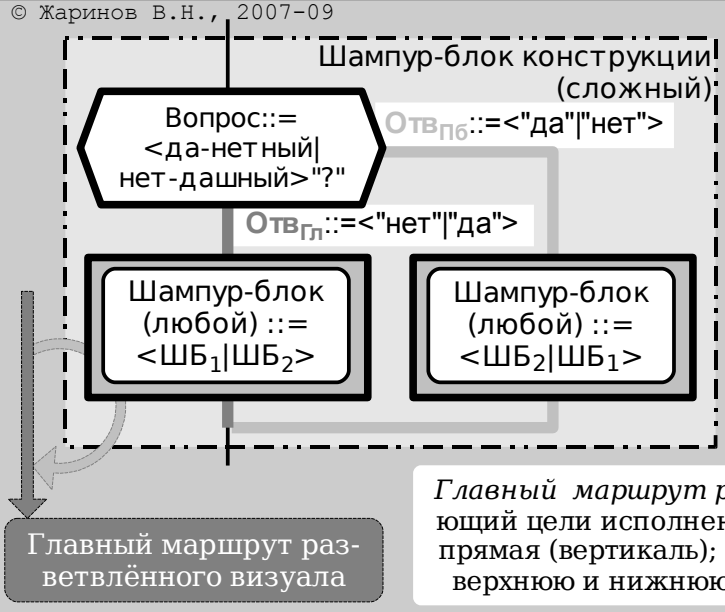
Неэргономичная схема:

- стрелки, соединитель и «мудрёные» подписи плеч требуют лишних усилий;
- в силу симметрии разница альтернатив неочевидна без особых указаний;
- обе вертикали «ломаются», что затрудняет прослеживание маршрутов исполнения;
- форма блока выбора неудобна для записи.

Развилка — визуализация простого ветвления на техноязыке. Имеет высокое информатическое качество (эргономична и технологична)

Выбор (альтернатива) — изображение простого ветвления на языке блок-схем (бинарных). Не отвечает требованиям информатического качества

Формальная структура развилки



Каждая из линий от выхода развилки до точки слияния (выделены) вместе со своей нагрузкой образует *лиану*

Ответы на вопрос могут меняться местами; тогда в *смысловой схеме* меняется местами и содержание вертикалей (шампур-блоки), т.е. происходит *рокировка*; в её результате на главный маршрут попадает содержание другой альтернативы

Главный маршрут разветвлённого визуала (обычно достигающий цели исполнения) *должен идти по шампуру*. Его лиана прямая (вертикаль); остальные лианы имеют и горизонтали — верхнюю и нижнюю — а также точки излома (оправданные)

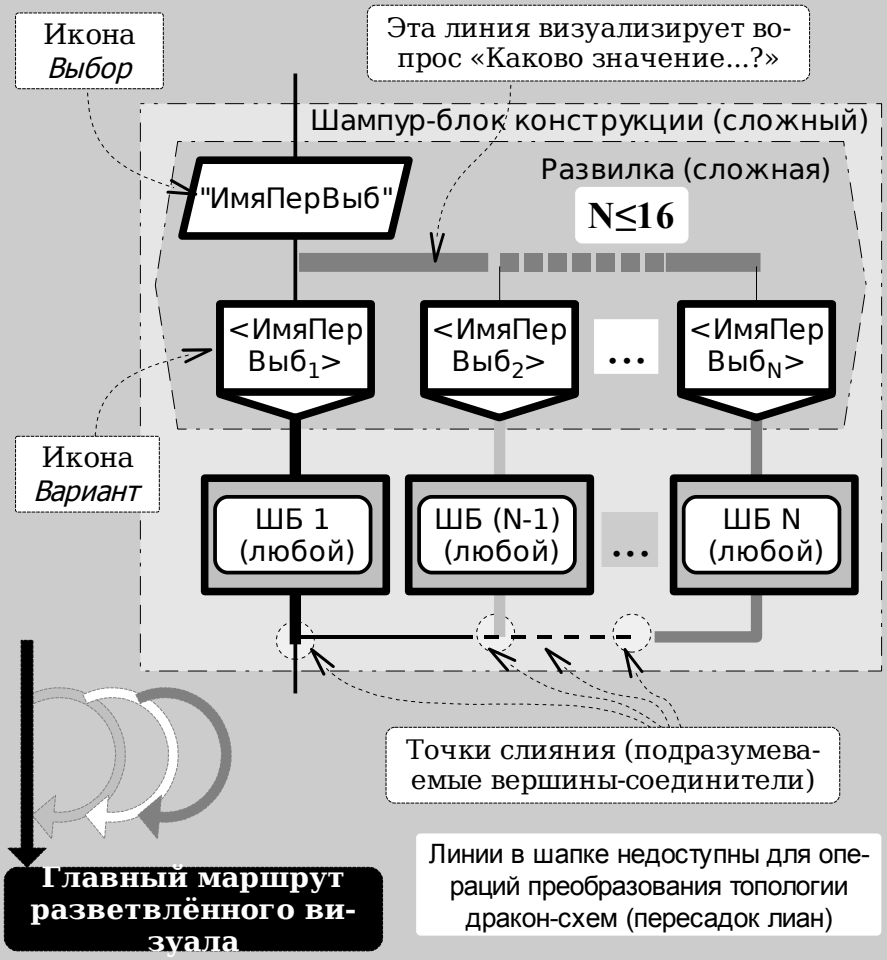
Простое ветвление – формальная структура на техноязыке и «очная ставка»

Ветвление (сложное)

Драконоподобная схема-прототип

Показано, что значения (ответы) можно задать как прямо (указанием точного значения, диапазона шкалы либо перечислением), так и косвенно - через выражение (формулу, здесь - логическую). Проверяется отношение значения переменной выбора к указанию (обычно равенство) и выбирается первый совпавший вариант, причём: порядок проверки всегда один и тот же (допустим, слева направо); подмножества указаний не должны перекрываться; значения переменной, не совпавшие ни с каким указанием (неопределённые), образуют отдельный вариант (здесь - крайний правый)

Формальная структура в техноязыке



В переключателе выбор среди неопределённых значений *ИмяПерВыб* д.б. указан явно как вариант "любое другое"

Выделенные линии от выходов (вместе с их нагрузкой) составляют ***лианы***

В *смысловой схеме* можно менять местами содержимое икон Вариант, тогда меняется местами и нагрузка вертикалей (шампур-блоки), т.е. происходит *рокировка*

Вершины сложной развилки вместе с их соединениями составляют т.н. ***шапку переключателя***. Точки слияния и линии между ними и нагрузкой аналогично составляют ***подвал***

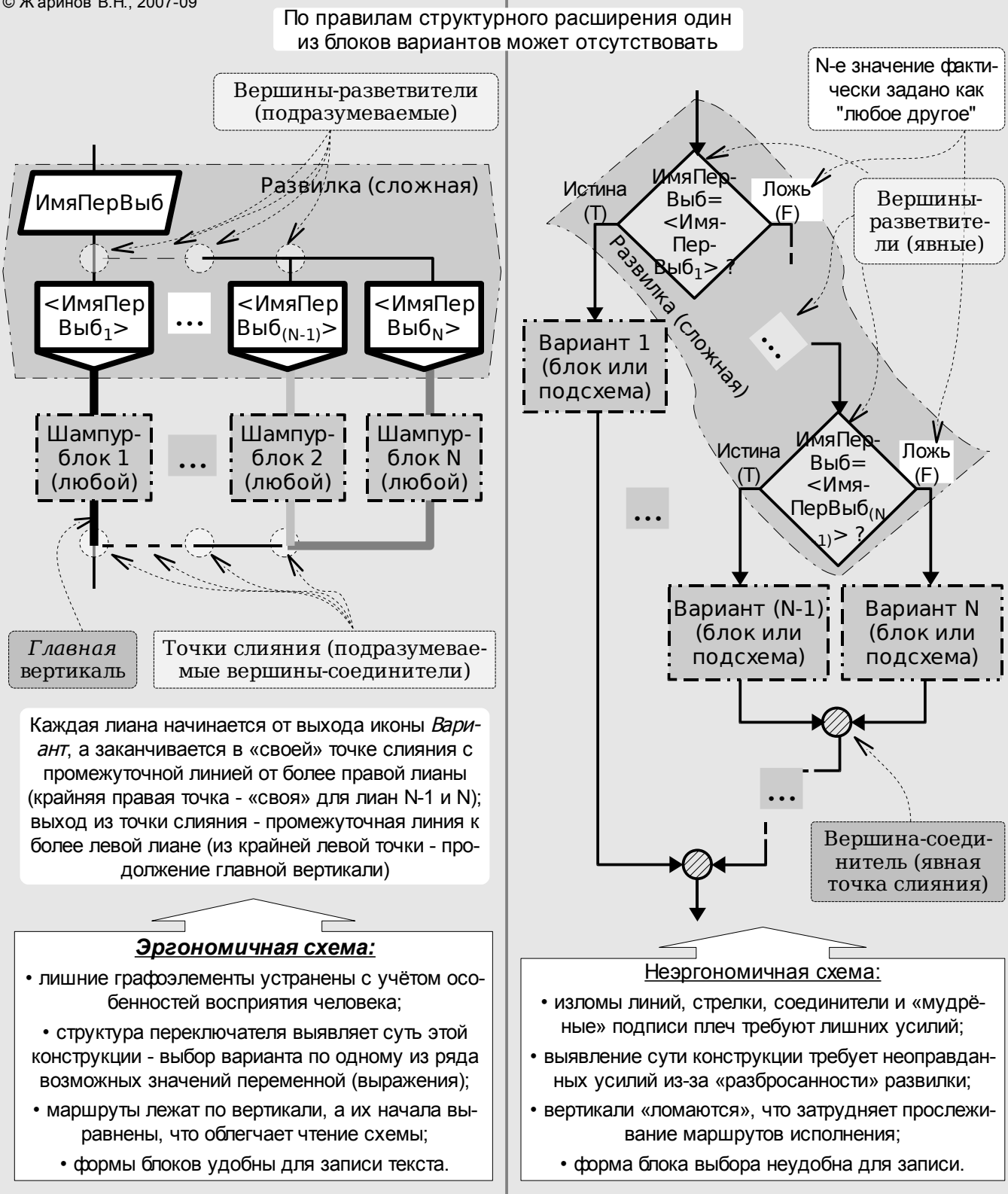
Линии в шапке недоступны для операций преобразования топологии дракон-схем (пересадок лиан)

Линии в шапке/подвале «частично выполняемые» - каждый раз только до/от иконы выбранного варианта

Множественное ветвление – происхождение и формальная структура в техноязыке

«Шапка» переключателя, как мы помним, логически остаётся единой (что показано взятием в контур-вопрос).

Ветвление (сложное)



Каждая лиана начинается от выхода иконы *Вариант*, а заканчивается в «своей» точке слияния с промежуточной линией от более правой лианы (крайняя правая точка - «своя» для лиан N-1 и N); выход из точки слияния - промежуточная линия к более левой лиане (из крайней левой точки - продолжение главной вертикали)

Эргономичная схема:

- лишние графоэлементы устранены с учётом особенностей восприятия человека;
- структура переключателя выявляет суть этой конструкции - выбор варианта по одному из ряда возможных значений переменной (выражения);
- маршруты лежат по вертикали, а их начала выравнены, что облегчает чтение схемы;
- формы блоков удобны для записи текста.

Неэргономичная схема:

- изломы линий, стрелки, соединители и «мудрёные» подписи плеч требуют лишних усилий;
- выявление сути конструкции требует неоправданных усилий из-за «разбросанности» развилки;
- вертикали «ломаются», что затрудняет прослеживание маршрутов исполнения;
- форма блока выбора неудобна для записи.

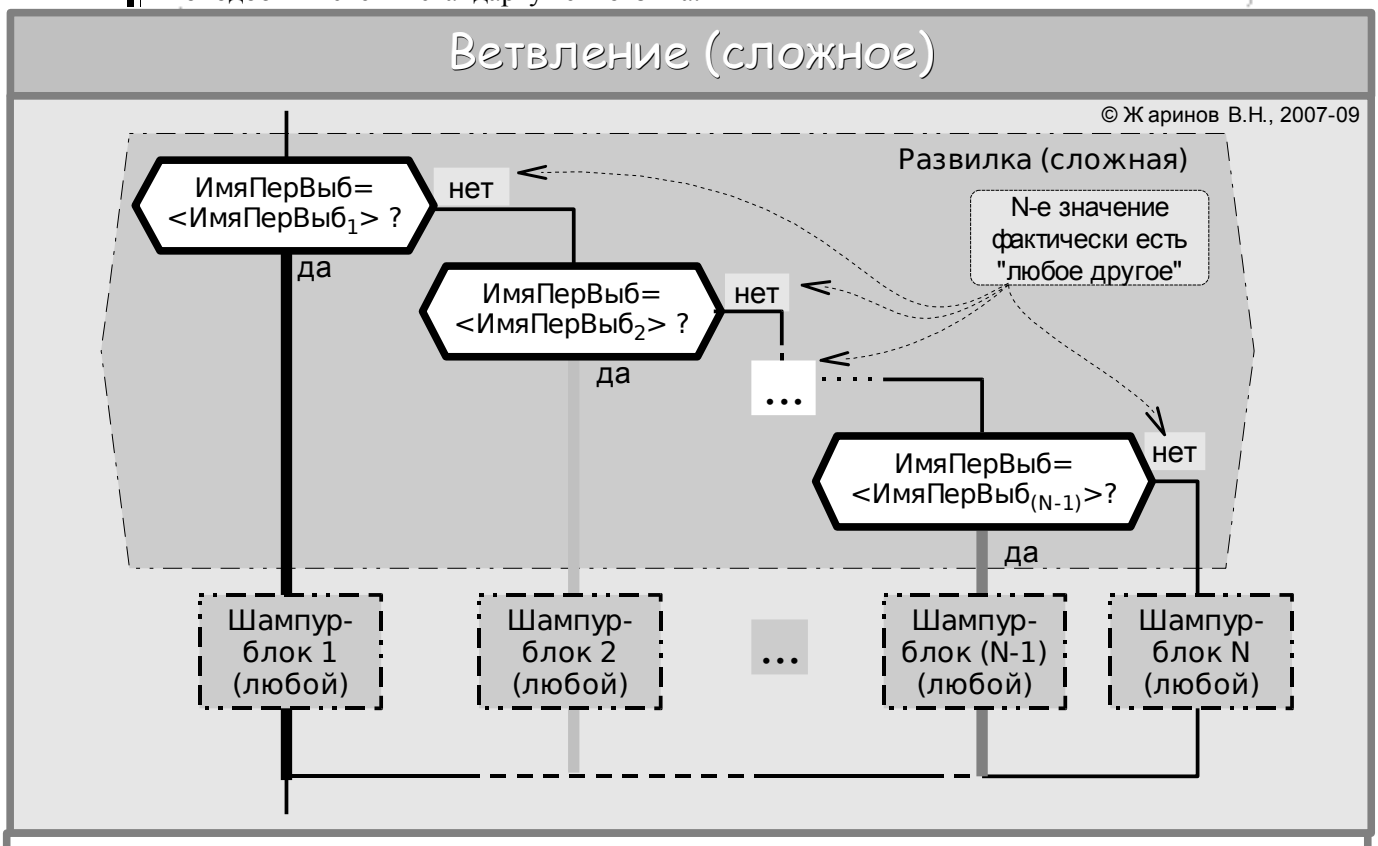
Переключатель - визуализация множественного ветвления в техноязыке. Имеет высокое информатическое качество	*Выбор* - изображение множественного ветвления в минимальном базисе ГСА (бинарных блок-схем). Не отвечает требованиям качества

Множественное ветвление – «очная ставка» на техноязыке и языке блок-схем

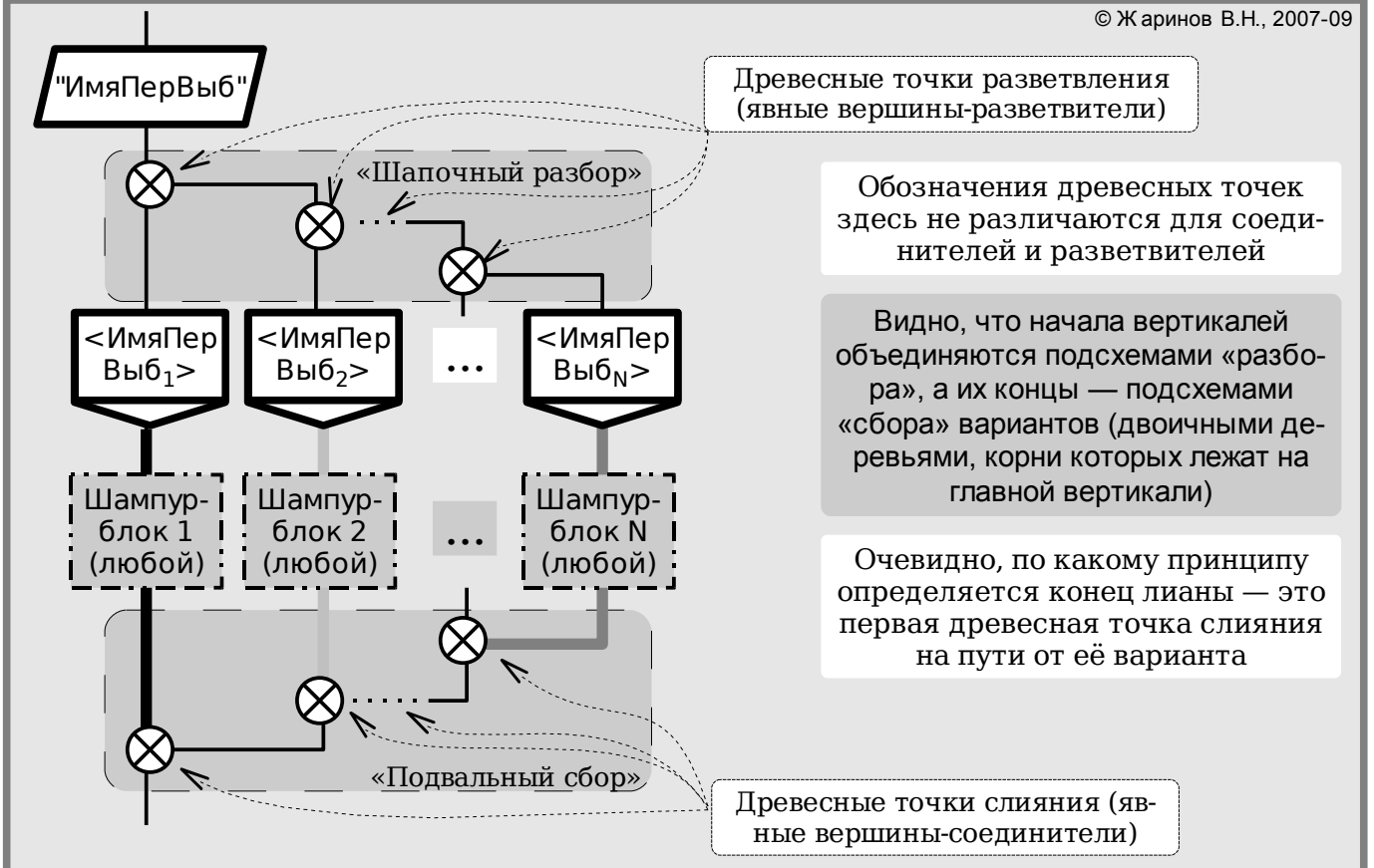
Во всех случаях мы выделяем штрихпунктиром обобщённые блоки – составные части, на которые структурируем конструкции.

На отдельных схемах мы пользуемся ранее описанным приёмом – представлением

обобщённого шампур-блока через шампур-икону *Комментарий* – для приведения драконоподобных схем к стандарту техноязыка.



Множественное ветвление можно визуализировать и на базе развилок (в стиле бинарных блок-схем). Результат менее качествен в сравнении с переключателем (громоздок и хуже читается)



Можно показать соединители и разветвители явно как т.н. древесные точки (без принятых в техноязыке эргономических упрощений показа маршрутов переключателя)

Раскрытие структуры сложного ветвления в техноязыке

Видно, что конструкцию ветвления в целом можно свести к шампур-блоку независимо от её внутренней структуры.

> В техноязыке нелинейные конструкции определяются как шампур-блоки через словарь макроикон, из которого нужно брать их начальные формы.

Если при визуализации реального алгоритма правило главного маршрута не соблюдено сразу, выполняют *рокировку* – перемену местами содержимого вертикалей схемы; при этом в развилке надо менять и формулировку вопроса на обратную, а в переключателе – варианты.

> Рокировка не предусмотрена стандартом техноязыка; это технологическая операция, относящаяся к равносильным преобразованиям.

Рокировку можно определить и как замену содержания блоков ветвления в обобщённой записи (см. формальную структуру данной конструкции).

Ещё одна проблема смысла сложного ветвления – значение «любое другое». Прежде всего возникает вопрос – любое «вообще»? Конечно, нет – только в пределах области допустимых значений величины *ИмяПерВыб*; в информатике любая такая область конечна, поэтому конечна будет и область «любых других» значений. Рассматривая области изменения переменной выбора и вариантов как множества, можем говорить о «любом другом» как о разности множества *ИмяПерВыб* и множеств всех вариантов.

Следующий вопрос – как определяется «любое другое» в дракон-переключателе? На рисунках мы видим разницу между ним и записью сложного ветвления через развилки. Во втором случае вариант «любое другое» логически вытекает из совокупности отрицательных ответов на вопросы всех развилок и задан неотъемлемо, не требуя явной формулировки. В переключателе же имеем только то, что явно определили в иконах Вариант как константы или выражения выбора. Из последней схемы, где сняты эргономические упрощения, можно понять, почему так происходит: шапка переключателя есть также дерево, но его промежуточные вершины – не условные операторы, а древесные точки-соединители, которые не участвуют в выборе маршрута; эта роль отведена вершинам-листьям дерева – иконам Вариант.

> Если позволить себе электрическую аналогию, то каждая икона Вопрос подобна тумблеру, соединяющему вход либо с главным выходом (по ответу «да» на тематический подвопрос развилки, т.е. совпадению текущего значения *ИмяПерВыб* со значением варианта), либо с побочным выходом (по ответу «нет»); икона Вариант же м.б. уподоблена кнопке, замыкаемой по совпадению значений. Тогда, если подать на схемы сложного ветвления ток, то он потечёт по тому маршруту, который предопределён значением *ИмяПерВыб* при заданных значениях вариантов.
>
> Будучи поверхностной, эта аналогия для переключателя оказывается верной лишь частично: если «замкнуть» не один размыкатель-вариант, то и ток потечёт по всем замкнутым ветвям, тогда как вариант для исполнения д.б. выбран единственный. При пересечении значений разных вариантов это достигается тем, что варианты просматриваются для выбора в определённом порядке – слева направо (как читается текст в европейских естественных языках). На нашей «электросхеме» придётся либо вводить ещё по кнопке-замыкателю между данной и следующей древесными точками... либо также наделять древесные разветвители свойствами тумблеров (от чего ушли, к тому и пришли).

Как влияет эта особенность переключателя на алгоритмизацию? В принципе никак – просто если требуется выбор по «любому другому» значению *ИмяПерВыб*, то нужно вводить в переключатель отдельный вариант.

> В реальных задачах зачастую «любое другое» значение вообще не должно присутствовать как возможность ветвления, т.к. вся область значений *ИмяПерВыб* должна исчерпываться явно заданными значениями вариантов (т.е. как результат пересечения их множеств со множеством значений *ИмяПерВыб* должно получаться пустое множество).

Далее для целей программирования из визуала получается импер-текст на некотором прогязыке.

> Важно понимать, что в прогязыке сложное ветвление (case-оператор) реализуется именно через цепочку развилок (так требуется для информашины-исполнителя); поэтому

вариант «любое другое» будет присутствовать всегда как последний по тексту ветвления (в некоторых прогязыках, скажем, в Perl, он явно выделяется в case-операторе как вариант «по умолчанию»). Поэтому если по логике задачи он не является «любым другим», то сочинитель должен сам обеспечить, чтобы область значений для этого варианта формировалась правильно – множества значений *ИмяПерВыб* и остальных вариантов д.б. такими, чтобы «остаток» от вычитания их из множества значений *ИмяПерВыб* давал нужное множество.

Итак, физическая реализация математической модели может накладывать определённые ограничения. Заметим, что наша электрическая аналогия показала именно это для уровня машинного исполнения сложного ветвления.

Возможна и по крайней мере одно ограничение уже при сочинении разветвлённой структуры (не обязательно сложной); это всё та же проблема компоновки реальных шампуров.

Физический размер вертикали определяется суммарной протяжённостью звеньев (их числом при заданной минимальной длине звена) и суммарной высотой икон (числом строк их текста при заданных параметрах вёрстки). Возможно, что некая побочная вертикаль физически длиннее охватываемого точками её присоединения участка главной вертикали (и вообще более правая вертикаль – длиннее более левой), если первая более насыщена иконами и/или её иконы – текстом. Тогда возникает задача согласования длин вертикалей на дракон-схеме.

По правилам топологии *наклонные линии и изломы вертикалей «против шампура» (т.е. участки обратного следования) запрещены*. Поэтому правильным решением будет «выключка», т.е. пропорциональное изменение звеньев вертикалей данного шампур-блока.

2.2.2.1. Произвольное ветвление: матрёшки и пересадки лиан

Поскольку мы определили, что нагрузкой любой вертикали ветвления м.б. любой шампур-блок, то как один из случаев это м.б. другое ветвление; так получается частный случай т.н. ***матрёшек*** – иерархически вложенных нелинейных конструкций.

Какие задачи приводят к матрёшкам? Очевидный случай – при сложном ветвлении у нас получилось более 16 вариантов выбора – такой переключатель запрещён в техноязыке в силу громоздкости. Для удобства восприятия он требует декомпозиции – представления в виде вложенных (входящих в вертикали друг друга) ветвлений.

Часто само множество вариантов выбора логически поддаётся классификации; иначе его можно разбить на подмножества принудительно и ввести переменную-критерий выбора между подмножествами. В результате визуализации такой системы условий у нас получатся переключатели, вложенные в другие переключатели. Аналогичная ситуация – условие ветвления у нас громоздкое, но его можно разбить на этапы проверки (подвопросы); эти этапы можно «разнести» по вложенным развилкам (и/или переключателям – это зависит от от того, до двух или более двух вариантов окажется в каждой группе классификации и можно ли поставить подвопрос как да-нетный). Однако это внешние (формальные) основания, связанные с формой представления визуала (здесь – с эргономизацией этой формы).

Может статься, что нам нужно за проверкой одного условия выполнить некие действия, а только потом проверять следующее; аналогично может понадобиться выполнить какие-то действия после прохождения ветвления перед следующим. Это уже внутренние (содержательные) основания для произвольного ветвления, вытекающие из логики решения задачи.

> Перед и после ветвления м.б. и другие ветвления (развилки, переключатели), образуя цепочку. Вертикаль, частью которой является главная вертикаль некоего ветвления, определим как *базовую вертикаль* данного ветвления, а ветвление, которое уже не охватывается другим ветвлением (т.е. сверху его вершиной-разветвителем, а снизу его же точкой слияния), считаем *внешним* в данной матрёшке. Базовой вертикалью в конце концов всегда является шампур визуала, поэтому любая матрёшка, выделенная не относительно шампура, является лишь частью к.-л. матрёшки, базированной на шампуре; конечно, матрёшки, как и просто ветвления, в любой вертикали могут следовать подряд цепочкой.

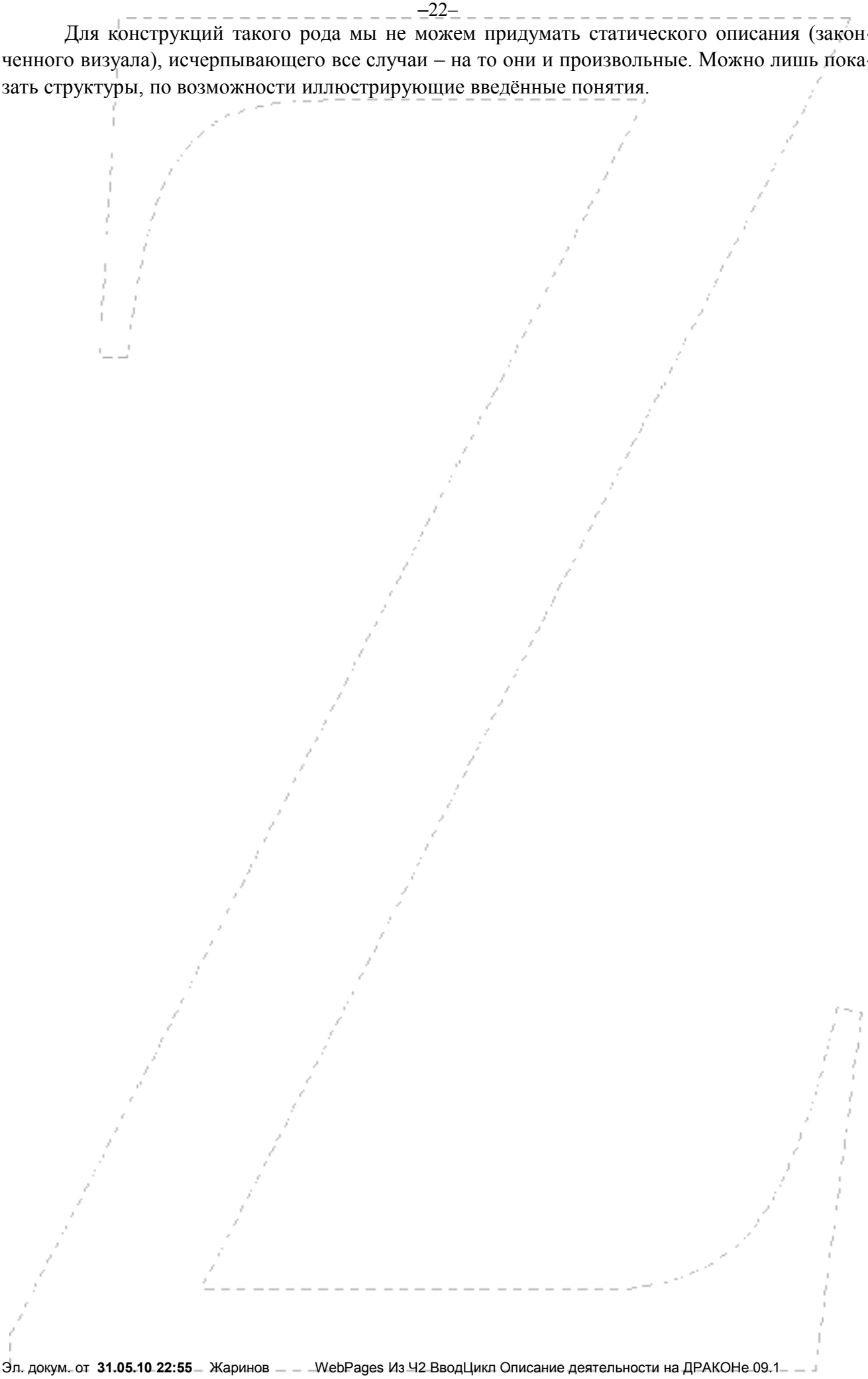
Любая вертикаль ветвления (необязательно в матрёшке) м.б. пустой либо нагруженной шампур-иконами, визуализирующими операции решения данной задачи; назовём такую нагрузку *линейной*. Вертикаль матрёшки, содержащая ветвление с ненагруженной главной вертикалью, сама уже нагружена, но эта нагрузка *чисто нелинейная*, не совершающая действий для решения задачи. Наконец, в матрёшке возможна *смешанная* нагрузка вертикали (когда в неё входят и ветвления, и шампур-иконы).

В любом случае мы продумываем структуру ветвления как иерархию макроикон и только потом строим её, начиная «сверху», т.е. с внешней макроиконы; далее в её вертикали вставляем макроиконы следующего уровня иерархии и т.д. Затем в переключатели можно добавить варианты до нужного числа и вставлять шампур-иконы в вертикали, требующие линейной нагрузки. Напомним, что ненагруженной можно оставлять лишь одну вертикаль ветвления.

Теперь посмотрим, что приводит нас к необходимости пересадки лиан. Опять-таки простейший случай – в нагруженной развилке нам нужно перенести точку слияния выше (тогда часть содержимого главной вертикали становится «общим достоянием», выполняясь при любом результате ветвления как часть вертикали, базовой для данной развилки) либо ниже (тогда наоборот, часть содержимого базовой вертикали переходит внутрь развилки и не выполняется при выборе её побочной вертикали).

Более сложный случай – если нужно проделать то же в переключателе; тогда точка слияния переносится на вертикаль другого варианта (только соседнего справа или слева, т.к. иначе нарушится запрет на пересечения). Можно перенести точку и на вертикаль развилки (соседнюю).

Для конструкций такого рода мы не можем придумать статического описания (законченного визуала), исчерпывающего все случаи – на то они и произвольные. Можно лишь показать структуры, по возможности иллюстрирующие введённые понятия.



2.2.2.2. Ветвление как произвольное следование

Подумаем теперь над смыслом двоичных деревьев. Для этого вспомним, что говорилось о безусловном переходе в текстовых алгоязыках. Очевидно, что каждую древесную точку сбора мы можем заменить БП-обходом с конца данной побочной вертикали переключателя на главную – в точку, предшествующую следующей за переключателем иконе. Также каждую древесную точку разбора вместе с её вариантом можно заменить на развилку, побочный выход которой оканчивается иконой БП на следующую вертикаль. Такое преобразование назовём прямым. Полученная схема показана на рисунке (см. далее).

Можно преобразовать переключатель в квазилинейную структуру и иначе. Для этого сначала введём безусловные переходы, оторвав все вертикали от сложной развилки (дерева «шапочного разбора»). А затем заместим это дерево его представлением через цепочку развилок, преобразованную операцией «инверсия ответов» – разновидностью рокировки, перемещающей и нижележащие иконы Развилка – как бы «вывернем шапку» (подобную операцию, но не затрагивающую положения самих условных вершин, Пароджанов называет «да-нет»).

Получится, что все развилки лежат цепочкой на главной вертикали (проходятся по ответу «нет», ставшему главным), а побочный выход каждой из них (по ответу «да») ведёт к БП на «свой» вариант; после цепочки лежит шампур-блок варианта «любое другое». Такое преобразование назовём инверсным. Конструкция показана на рисунке (нижняя область).

Очевидно, то же самое можно проделать для простого ветвления; вся разница в числе вынесенных фрагментов (он будет единственным).

Что у нас получилось? Некий вариант произвольного следования; с его помощью мы создали возможность выкладки алгоструктуры ветвления в линию. Точно также, как и для произвольного следования, побочные варианты можно рассматривать как аналоги визуалов-вставок; также и варианты выкладки образуют множество перестановок на полученном наборе фрагментов. Это не что иное, как лиоформа разветвлённой дракон-схемы (шампур-блока).

Если имеются вставки в вертикали, они точно также раскладываются на фрагменты, и в результирующей лиоформе сочетаются БП обычные и с возвратом.

Очевидно, эту операцию мы можем применить и к переключателю, т.к. ранее мы показали, что его можно свести к цепочке вложенных развилок.

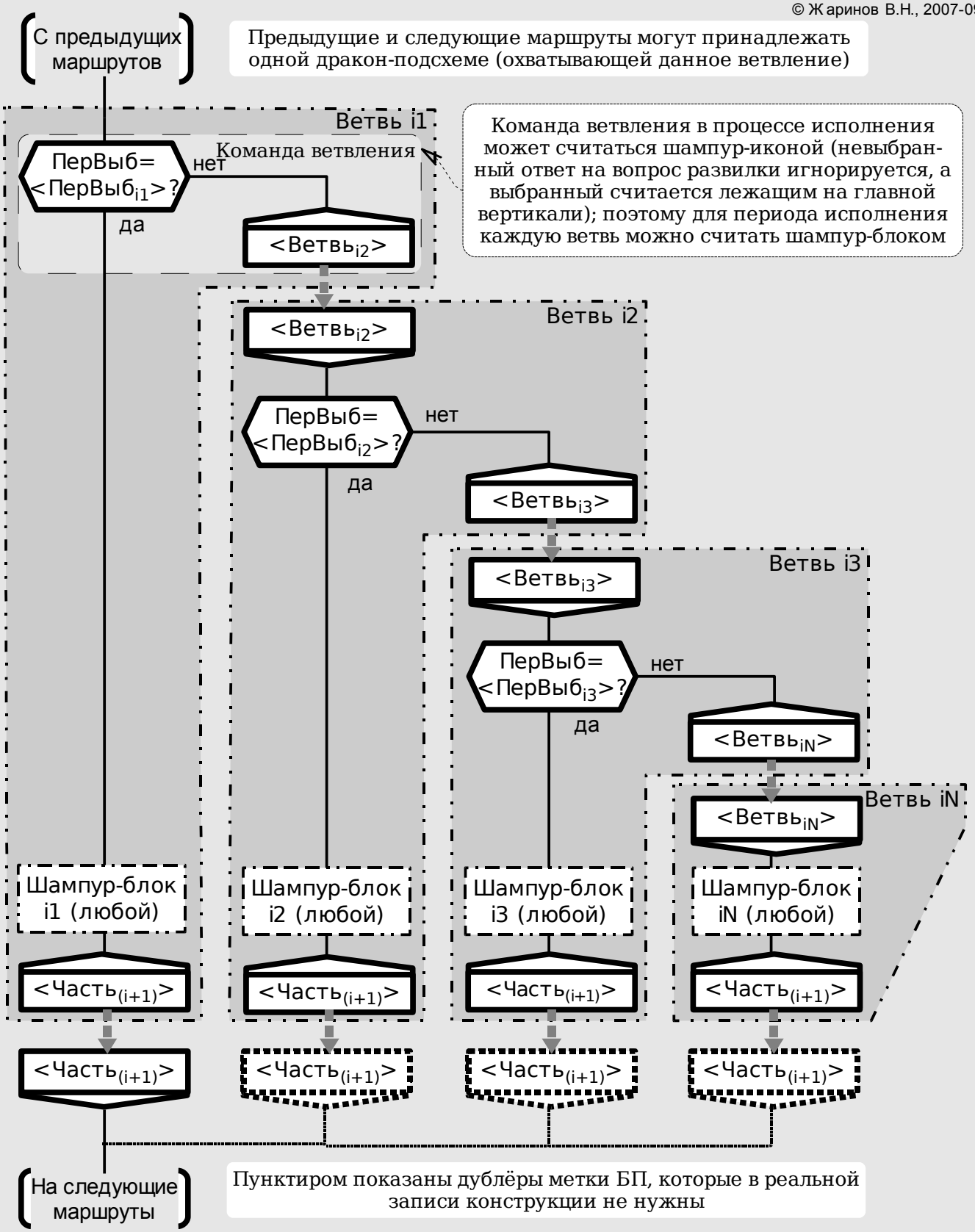
С другой стороны, получилась визуализация «устройства» конструкции сложного ветвления (case-оператора) на программном уровне формализации (в форме машинного кода). При этом мы показываем то, что в текстовых операторах ветвления обычно «спрятано» – присутствие безусловного перехода для целей обхода вариантов, не выбираемых при ветвлении.

> В некоторых прогязыках (диалектах) БП-обход в конце варианта не реализуется (напр. для классического Си); тогда возможности маршрутизации сужаются (можно только пропустить один или более вариантов, которые предшествуют по тексту выбранному; если за ним по тексту есть последующие, то они будут выполнены).

Теперь проясняется причина выбора именно такой графики икон обычного БП; тем самым подчёркивается необходимость его для выкладки ветвлений.

Все шампур-блоки перед выкладкой нужно свести к простым; при этом в общем случае имеем конструкцию произвольного ветвления.

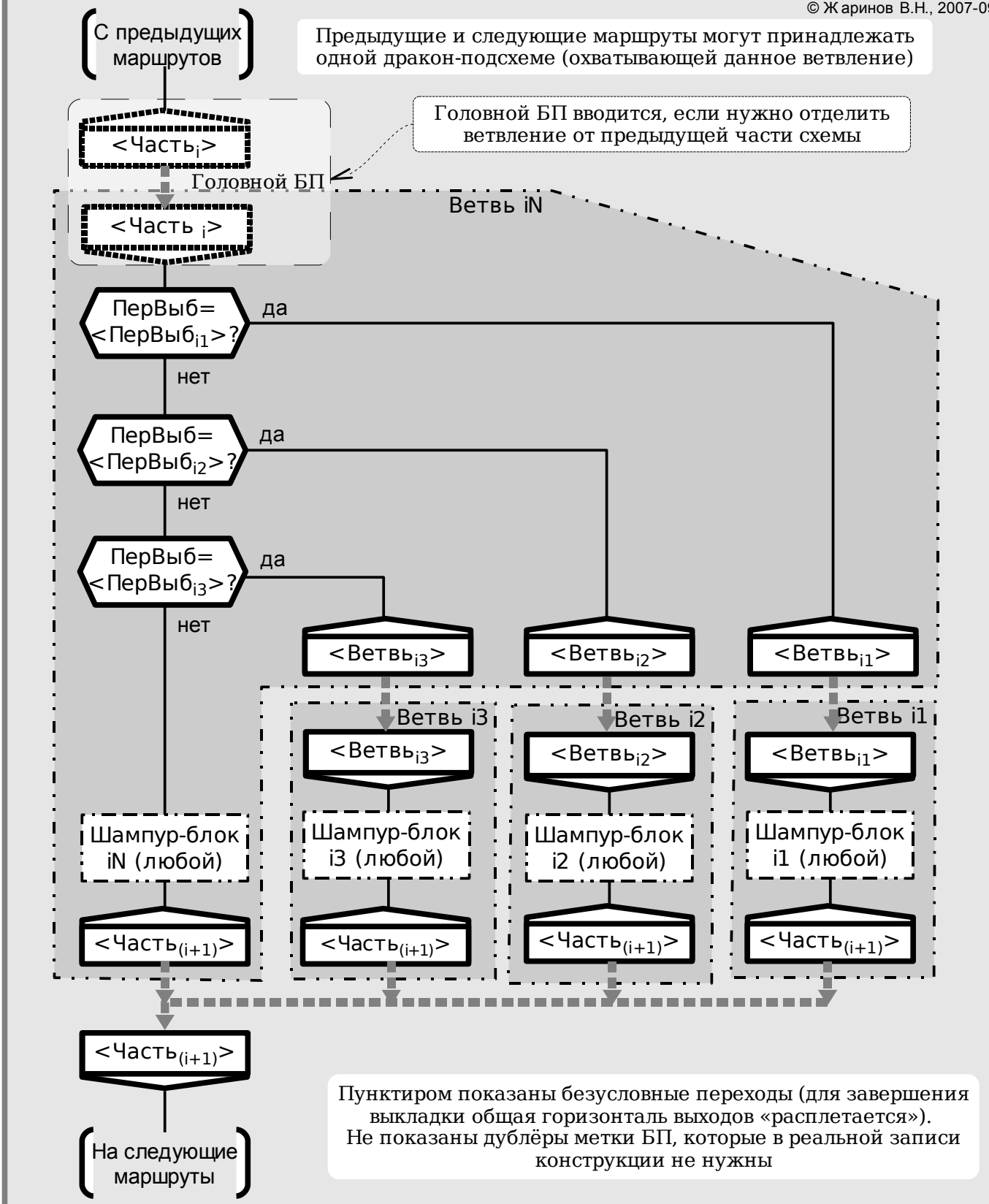
Сложное ветвление в лиоформе (прямой)



Показана прямая выкладка цепочки развилок в лиоформу, при которой иконы БП вводятся непосредственно в начала и концы вертикалей исходной конструкции

Прямое преобразование сложного ветвления в произвольное следование

Сложное ветвление в лиоформе (инверсной)



При инверсной выкладке в лиоформу цепочки развилок вначале исходная конструкция преобразуется так, что развилки оказываются на главной вертикали, а уже затем вводятся иконы БП

Инверсное преобразование сложного ветвления в произвольное следование

2.1.3. Цикл

Является нелинейной структурой с *обратной связью (ОС)* – маршрутом, включающим участки как с прямым следованием, так и с обратным, причём последний замыкается на главную вертикаль конструкции выше ветвления. Выбор этого маршрута приводит к *итерации* – очередному выполнению охваченных ОС операторов, но, возможно, в иной алгоритмической обстановке (которая может изменяться в ходе итерации); выбор другого маршрута означает выход из цикла. Как следствие, маршрут циклической структуры включает повторяющиеся фрагменты; определение же маршрута цикла (и его формулы) аналогично ветвлению.

Традиционные варианты этой структуры в техноязыке записаны эргономично. Так, замыкание обратной связи оформлено как икона *Петля цикла*; т.о. в ней невозможно размещение икон. В результате сохраняется принцип, по которому *вертикаль обратного следования не считается шампуром* (отсюда и топологический запрет на загибы вертикалей шампур-блоков вверх, и отдельная икона *Период* для паузы в петле цикла), а расположение вертикалей такое же, как в развилке; то и другое удобно для составления и чтения схем.

Традиционные «наукообразные» названия обычных циклов «с предусловием» и «с постусловием» в техноязыке заменяются соответственно на *цикл ПОКА* и *цикл ДО*, а цикл «с параметром» называется *циклом ДЛЯ*. Кроме того, вводятся новые виды циклов, максимально использующие возможности языка.

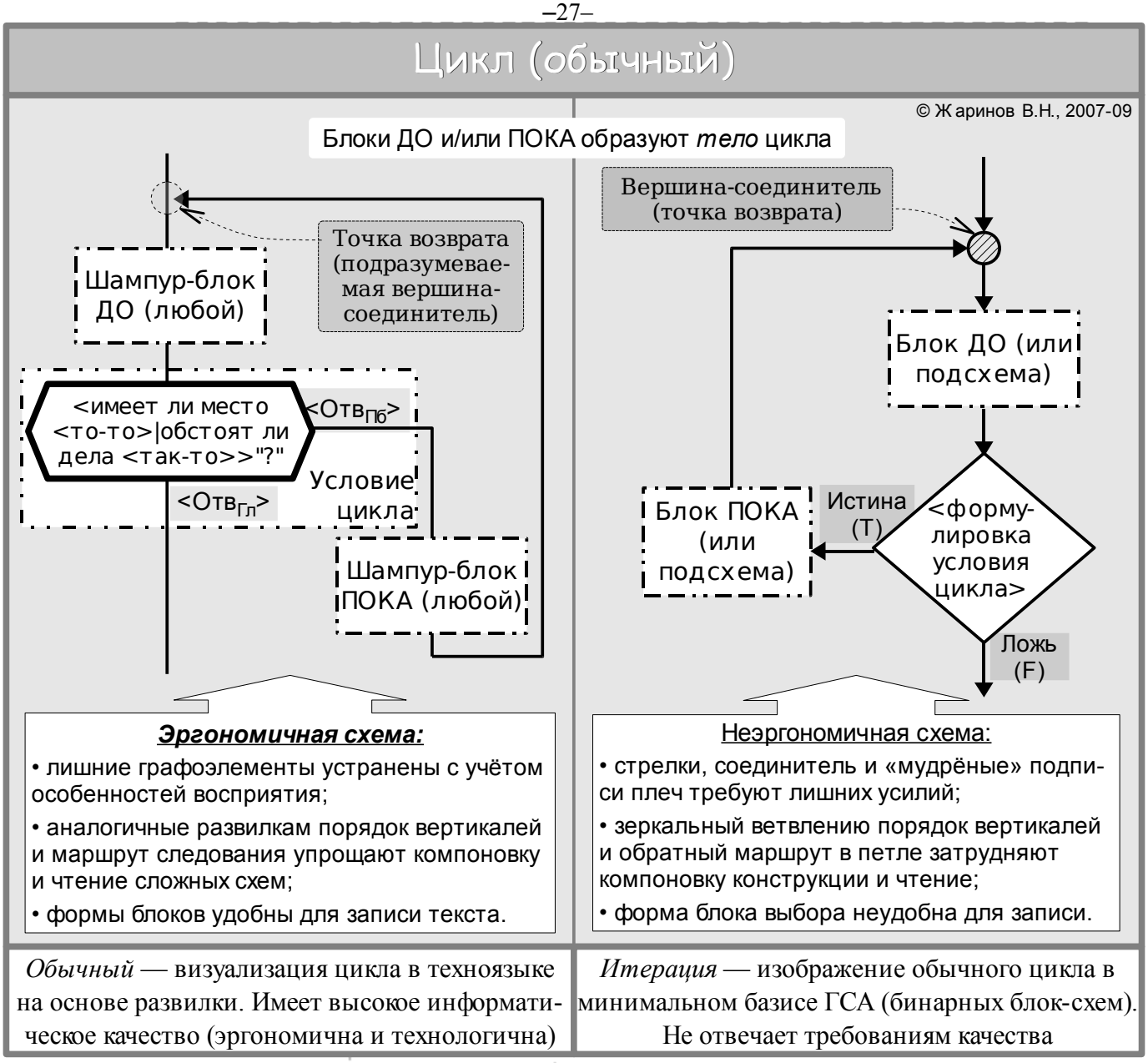
> Важно уяснить себе, что в техноязыке цикл любого вида организуется только добавлением макроиконы в нужную вертикаль. Пересадки лианы возможны в теле цикла для организации т.н. досрочных выходов; образовывать новый цикл при этом не допускается.

Рассмотрение начнём с **гибридного цикла**, представляющего собой общую форму структуры обычного цикла (см. схему далее).

Обычный цикл – формальная структура на техноязыке

В цикле развилка играет иную роль, чем в ветвлении – условия выбора между маршрутом, который завершается возвратом, и маршрутом, продолжающим следование (до конца шампура либо до следующего ветвления). Поэтому здесь в общем случае не имеет смысла понятие главного маршрута и операция рокировки.

Далее проведём «очную ставку» с блок-схемами, результаты которой показаны на рисунке.



Обычный цикл (гибридный) – «очная ставка» с блок-схемной записью

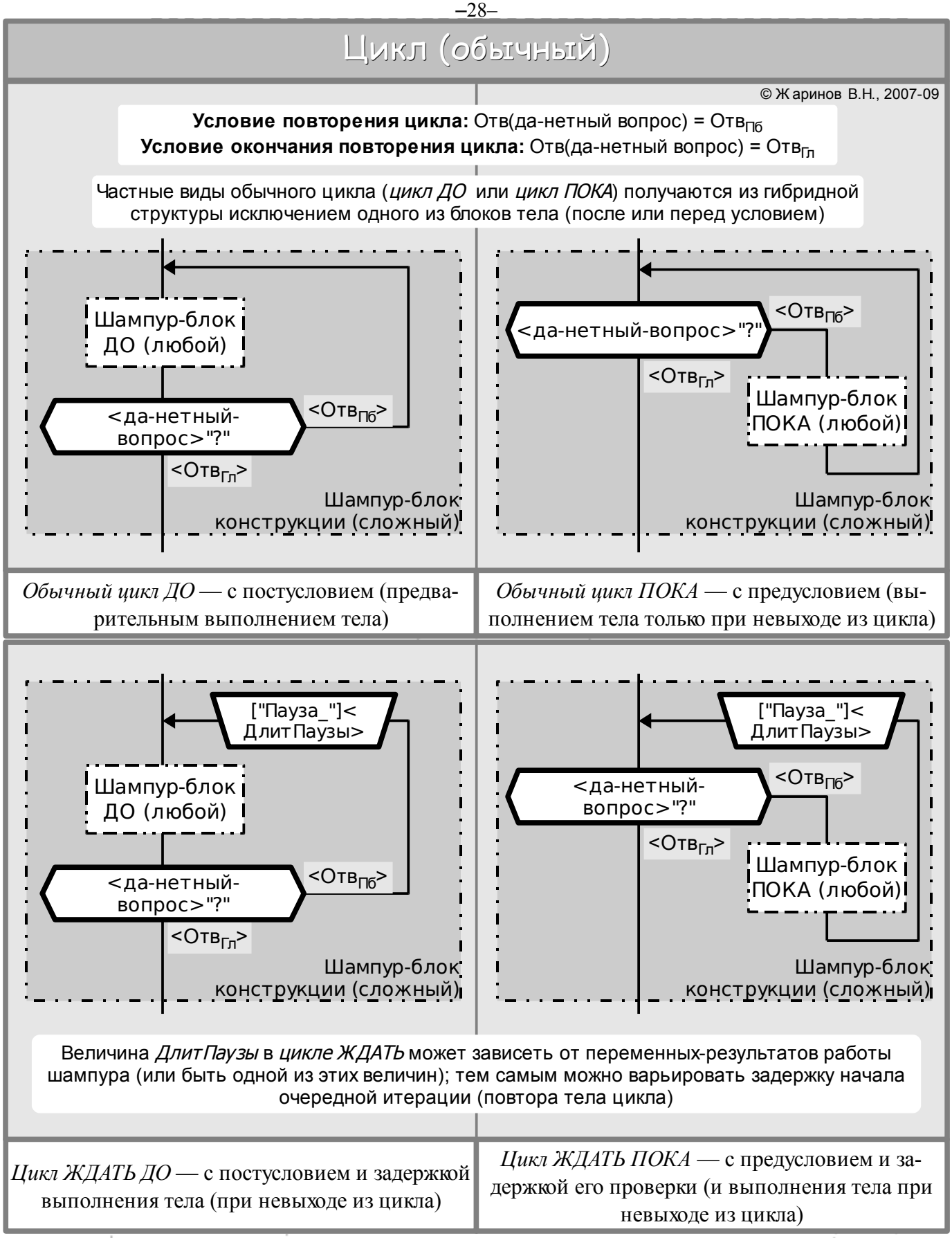
> Здесь мы также выделяем штрихпунктиром обобщённые блоки, на которые структурируем конструкцию.

Цикл ЖДАТЬ позволяет регулировать темп итераций за счёт введения в цепь обратной связи оператора задержки.

Также м.б. необходимо расположить тело полностью до или после условия цикла; так приходим к частным формам обычного цикла.

Цикл ДО и **цикл ПОКА** отличаются, кроме блочного состава тела, также конфигурацией соединительных линий (см. схемы); изменения объяснены в формальной структуре (см.).

Как частные подвиды возникают также циклы ЖДАТЬ ДО и ЖДАТЬ ПОКА.



Конструкции обычного цикла частных типов (ПОКА и ДО) на техноязыке

Цикл ДЛЯ применяется, когда число повторений либо заранее известно, либо м.б. вычислено по известным граничным значениям некой числовой переменной и величине её приращения за один повтор цикла. Всё это иллюстрирует формальная структура цикла, записанная через цикл ДО (см. схему).

Цикл ДЛЯ — визуализация цикла с параметром (известным числом повторений). Между иконами начала и конца подразумевается петля цикла (выделена белым)	*Обычный цикл ДЛЯ* — визуализация цикла с параметром позволяет наглядно представить его выполнение, а также перейти к изображению в базисе бинарных ГСА

Конструкции цикла с параметром – формы записи на техноязыке

> Здесь мы применили другой прием приведения обобщённой схемы к стандарту техноязыка – замену шампур-блока иконой *Вставка*; так целесообразно поступать, когда содержание блока нам известно.

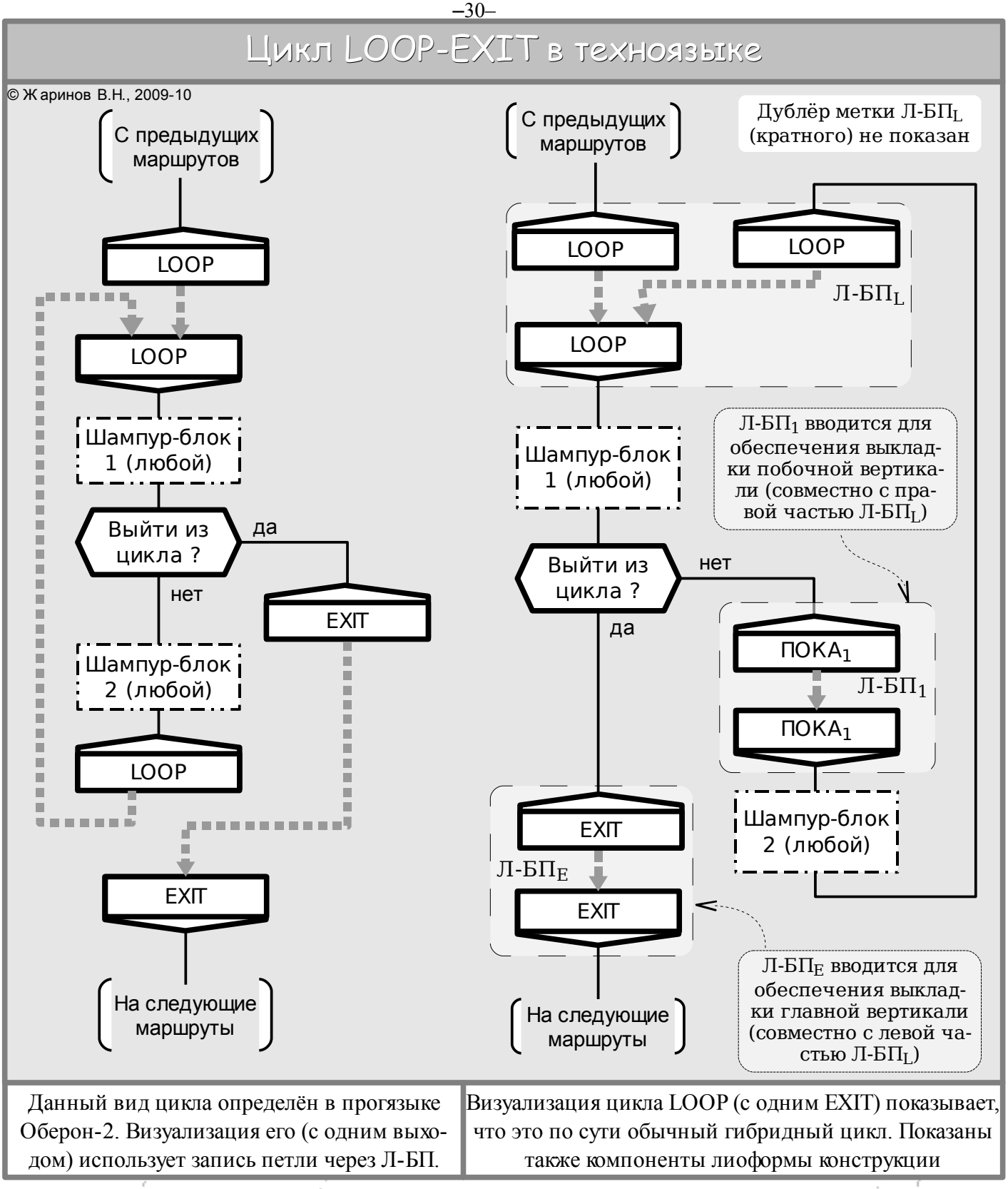
Наконец, в Обероне-2 имеется конструкция **LOOP-EXIT-цикла**. Его также можно записать с использованием БП. Результаты показаны ниже:

Здесь мы для наглядности сопоставления с ТЯП-формой записали метки линейных БП иначе, чем ранее для сложных ветвлений. Естественно, что переход на начало цикла (метку входа в цикл) оказывается кратным (мы это обсуждали для иконы Петля цикла).

По сути, получено инвариантное представление для выкладки простых циклов; это очевидно на правой схеме, полученной путём рокировки условия цикла в левой. Её мы показали в полной лиоформе (готовой для выкладки на «ленту памяти» исполнителя-информашины).

А что получится, если любые развилки ведут внутрь тела цикла? Тогда не имеем выхода и получаем «зацикленную» (логически бесконечную) конструкцию. Она также имеет и практическое, и теоретическое значение; подробнее см. п. 2.3.1 .

По правилам Оберона, данный цикл может иметь и более одного выхода, но любой из них должен вести на следующий за циклом по ходу следования оператор (икону). При этом получится сложный цикл; визуализацию и анализ его оставляем читателю.



Данный вид цикла определён в прогязыке Оберон-2. Визуализация его (с одним выходом) использует запись петли через Л-БП.	Визуализация цикла LOOP (с одним EXIT) показывает, что это по сути обычный гибридный цикл. Показаны также компоненты лиоформы конструкции

Цикл LOOP-EXIT – формы записи через безусловные переходы

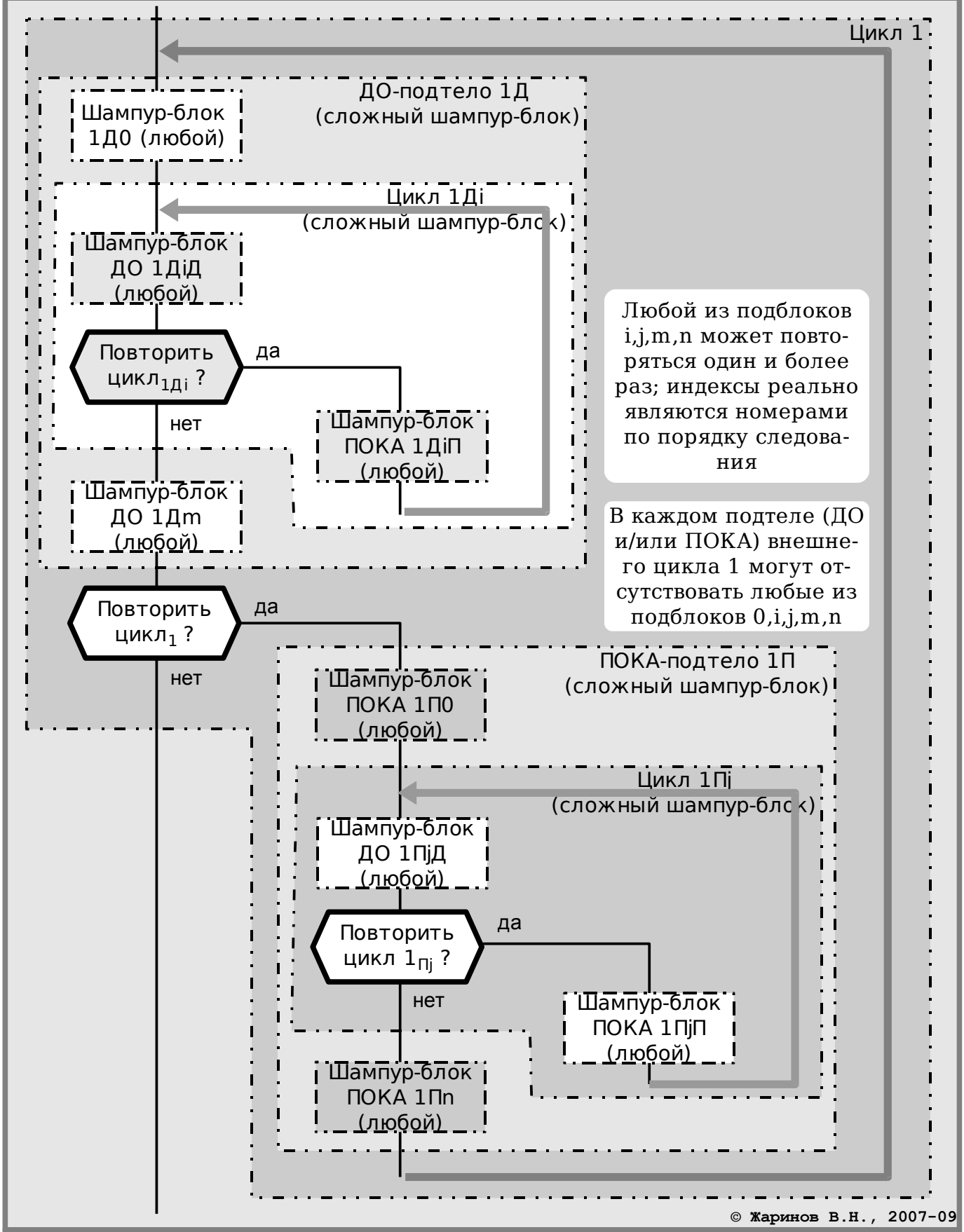
Далее займёмся другими сложными (и в то же время регулярными) циклическими алгоструктурами.

2.2.3.1. Сложные циклы: диоформы и произвольное следование

К сложным прежде всего относятся **циклы в цикле**, как иерархия вложенных друг в друга простых циклов. Такую конструкцию называют ещё «гнездом циклов».

> *Цикл в цикле* – конструкция, в которой тело цикла, входящего в её главную вертикаль (внешнего), содержит в себе другой цикл (внутренний); при этом внутренний цикл может располагаться как до условия цикла, так и после него.

Циклы в цикле (общий вид)



Показано, как в гибридный цикл м.б. вложены (однократно) другие гибридные циклы. По тому же принципу можно наращивать глубину вложения (вводя циклы в шампур-блоки внутри циклов).

Обобщённая структура вида "цикл в цикле" (с одинарным вложением)

И внешний, и любой из внутренних циклов м.б. любого типа. Схематически простые циклы в цикле (имеющие только один уровень внутренних циклов) показаны в /1, Гл.9/. Ниже показана общая структура гнезда циклов для одного уровня вложенности.

Каждый шампур блок на схемах, конечно, в свою очередь может включать в себя циклы; так образуются цепочки циклов и циклы в цикле более глубокой вложенности (гнёзда циклов).

Переключающий цикл вводится на основе переключателя. Его условие также можно представить через систему развилок.

Переключающий цикл можно разложить на петлю цикла и переключатель-условие цикла (специально изменённый переключатель, см. верхнюю форму записи); но можно выделить в нём (см. нижнюю форму) условие цикла (первую развилку) и особый блок, который включает ветвление внутри тела цикла по остальным условиям; его связь с главной вертикалью конструкции можно получить только пересадкой лианы (поэтому блок и назван л и а н н ы м).

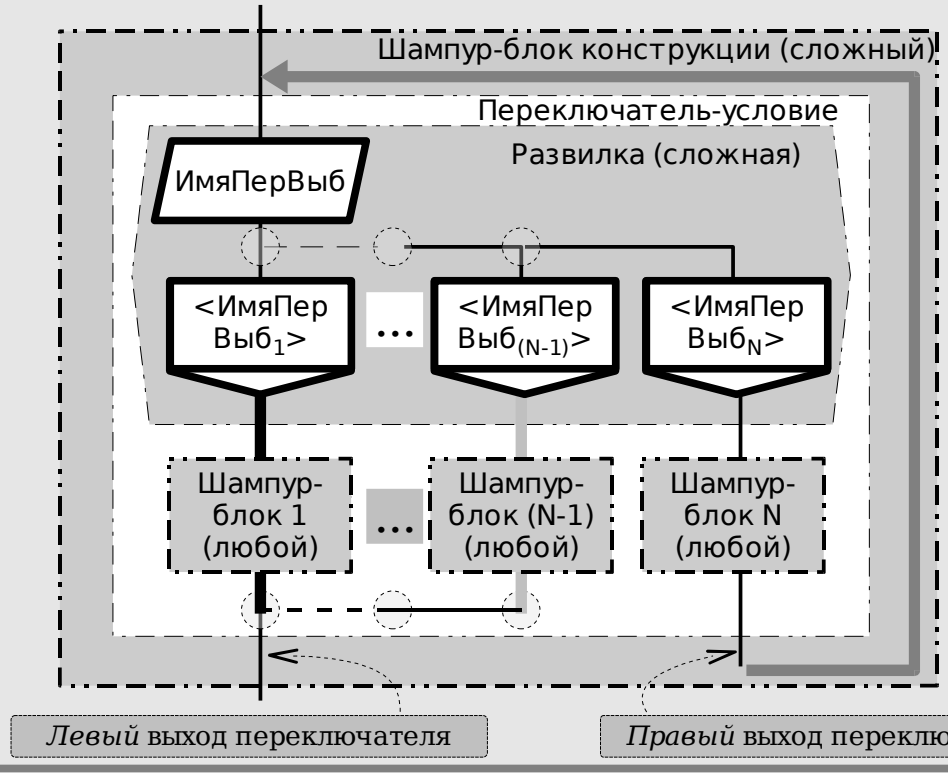
Видно, что в переключающем цикле возможно прекращение очередного выполнения тела – досрочный выход из цикла – но маршрут перед этим каждый раз проходит единственный шампур-блок (как и для повторения). А можно ли задать произвольный маршрут внутри тела? Да, и для этого рационально основываться как раз на последней форме записи переключающего цикла.

Как мы увидим далее, по этому принципу можно построить и иную разновидность переключающего цикла; поэтому рассмотренную будем называть циклом Пароджанова.

Ещё один сложный **цикл Дейкстры** был предложен как структурная конструкция Э. Дейкстрой ещё для текстовых прогязыков. В ТЯП-форме он описан, в частности, Виртом[4]. Визуализация этого описания возможна, если считать условие цикла сложной развилкой.

[4] Вирт Н., 2010. – Приложение С.

Цикл переключающий



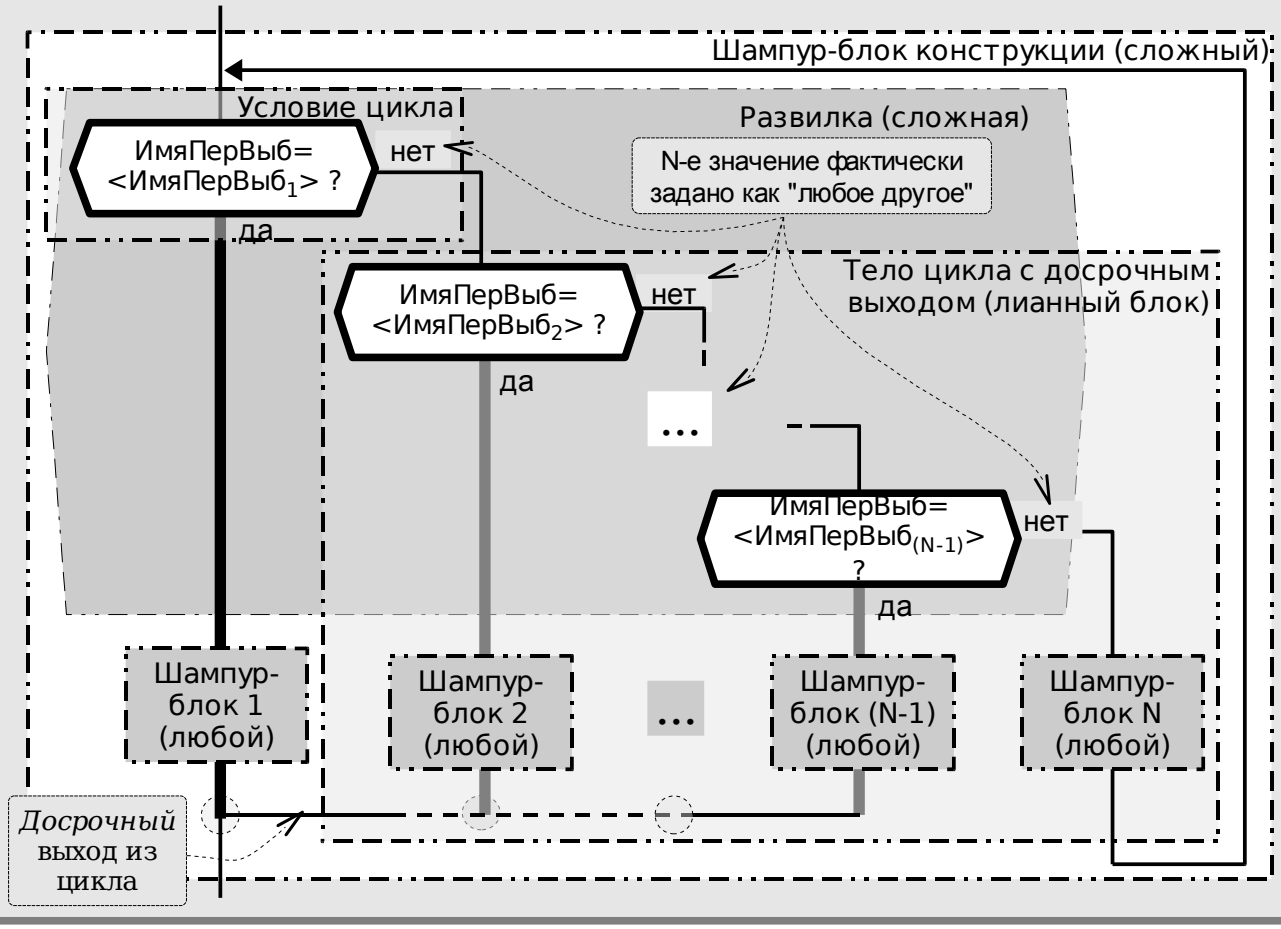
Структура связей задаётся заранее как макроикона *Переключающий цикл*

Неопределённое значение ИмяПерВыб д.б. указано явно

Шампур-блок одного из вариантов может отсутствовать

Зацикливается всегда один (крайний правый) вариант

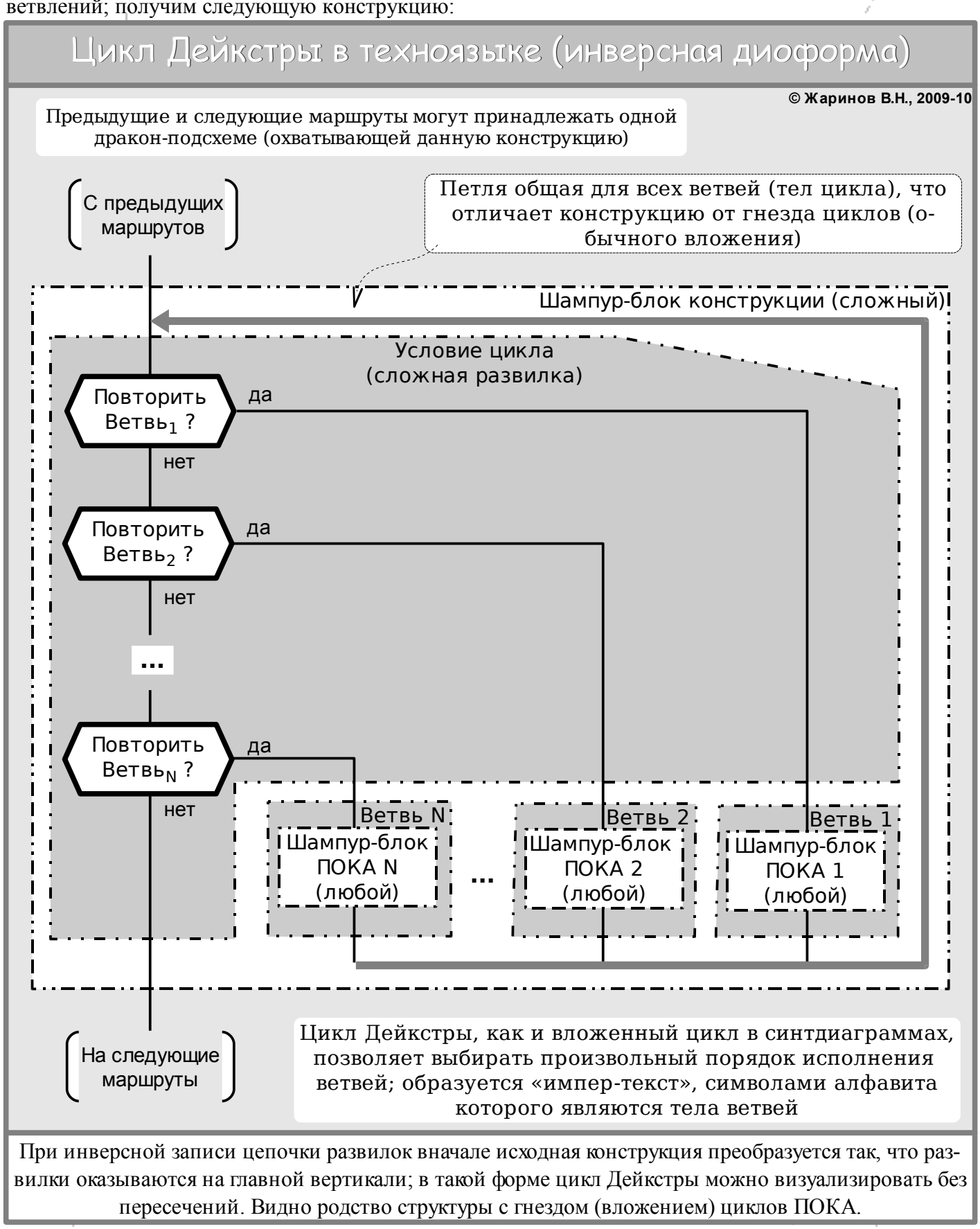
Переключающий цикл имеет в основе переключатель, преобразованный так, что первый вариант становится условием выхода из цикла, а последний вариант - условием повторения цикла.



Переключающий цикл можно визуализировать и на основе сложной развилки. Видно, что можно выделить блок разветвлённой структуры, левые вертикали которого ведут к ***досрочному выходу*** из цикла

Переключающий цикл – формы записи на техноязыке

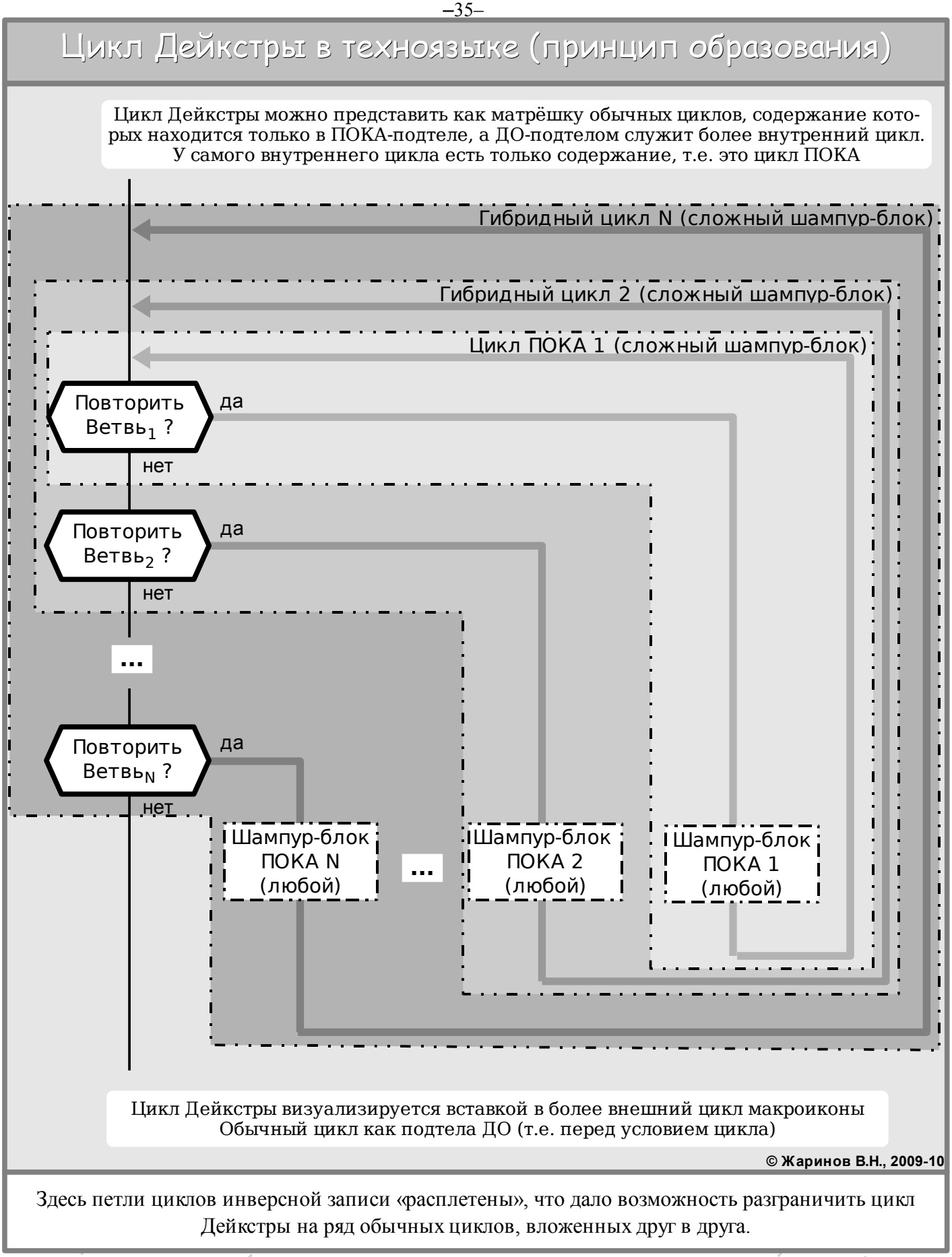
Выберем инверсную запись этой развилки, которую мы использовали для сложных ветвлений; получим следующую конструкцию:



Цикл Дейкстры в инверсной записи

Видно, что цикл многоветочный, но каждый раз исполняется одна из веток.

А как построить такую конструкцию по шампур-методу? Для этого надо выделить петли составляющих циклов.

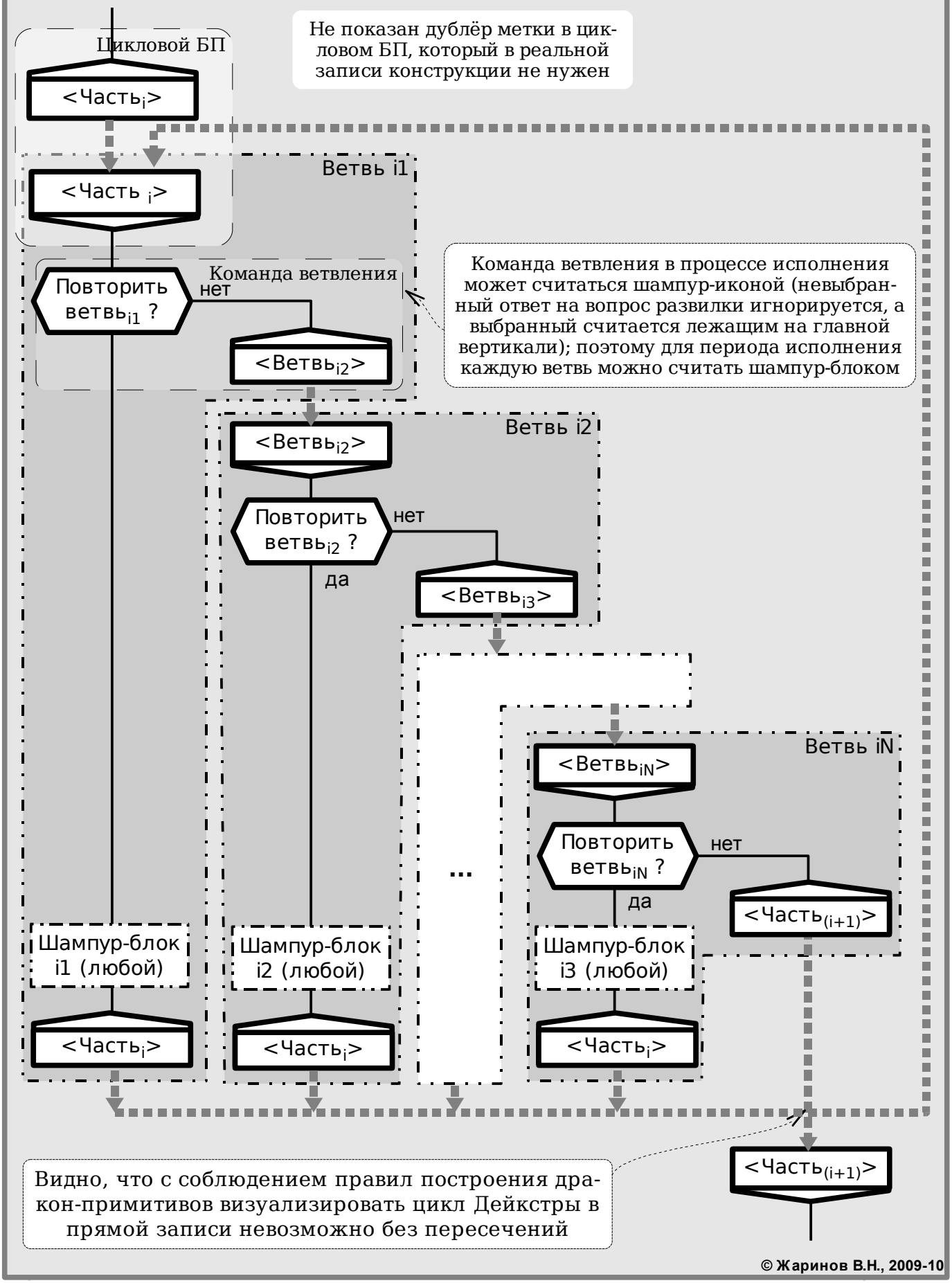


Здесь петли циклов инверсной записи «расплетены», что дало возможность разграничить цикл Дейкстры на ряд обычных циклов, вложенных друг в друга.

Принцип образования цикла Дейкстры

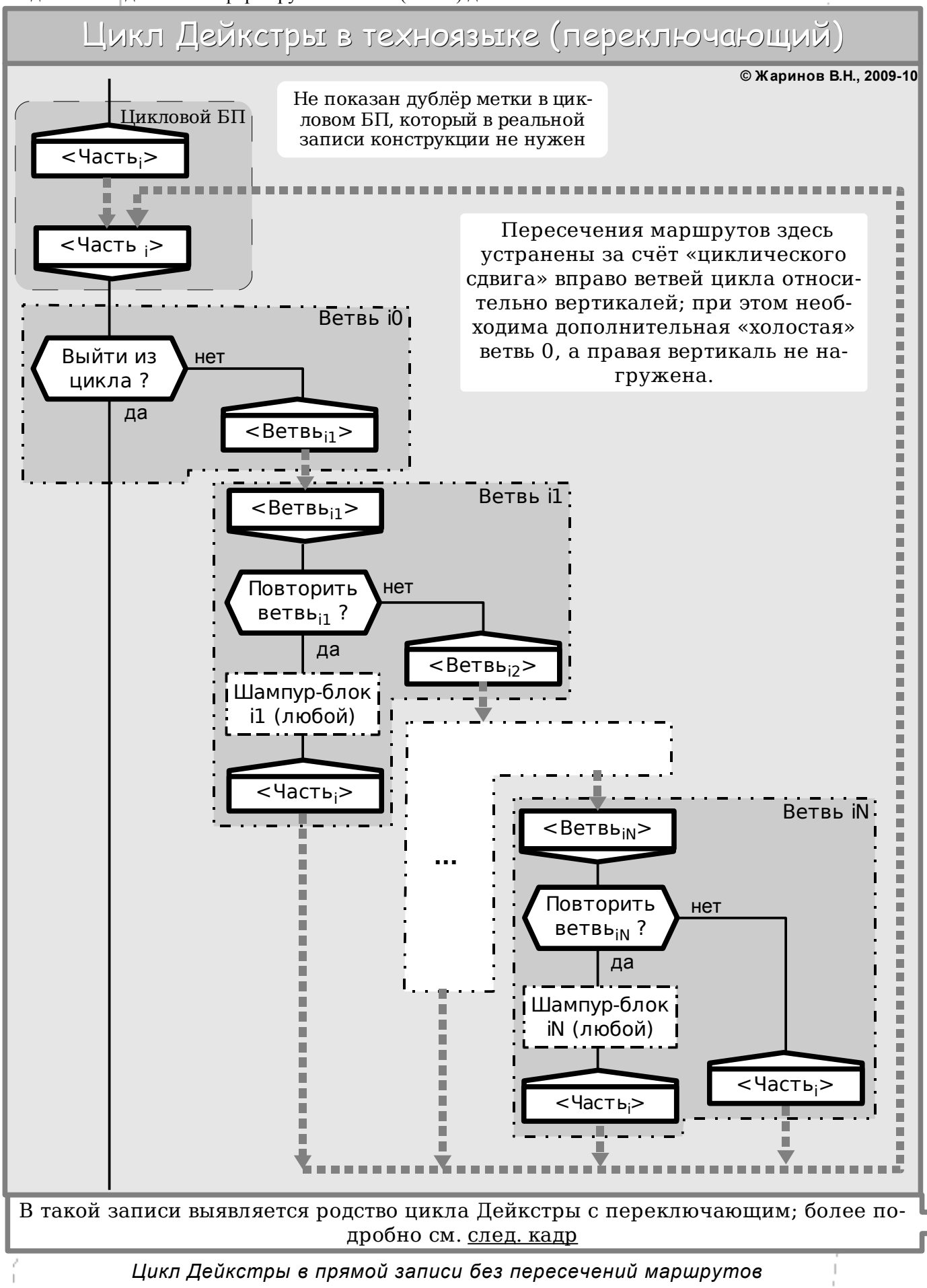
Теперь построим цикл с условием в прямой записи. Для этого введём в предыдущую схему линейные БП, затем обратим развилку условия и подставим тела ветвей. Результат показан на схеме ниже:

Цикл Дейкстры в техноязыке (прямая запись)



Цикл Дейкстры в прямой записи

Как и ранее при введении линейных БП в дракон-схемы, считаем, что рассматриваемая конструкция есть i-тая из N частей, на которые содержащий её некий визуал поделён такими переходами. Исходя из этого формируются имена (метки) для линейных БП.



В такой записи выявляется родство цикла Дейкстры с переключающим; более подробно см. след. кадр

Цикл Дейкстры в прямой записи без пересечений маршрутов

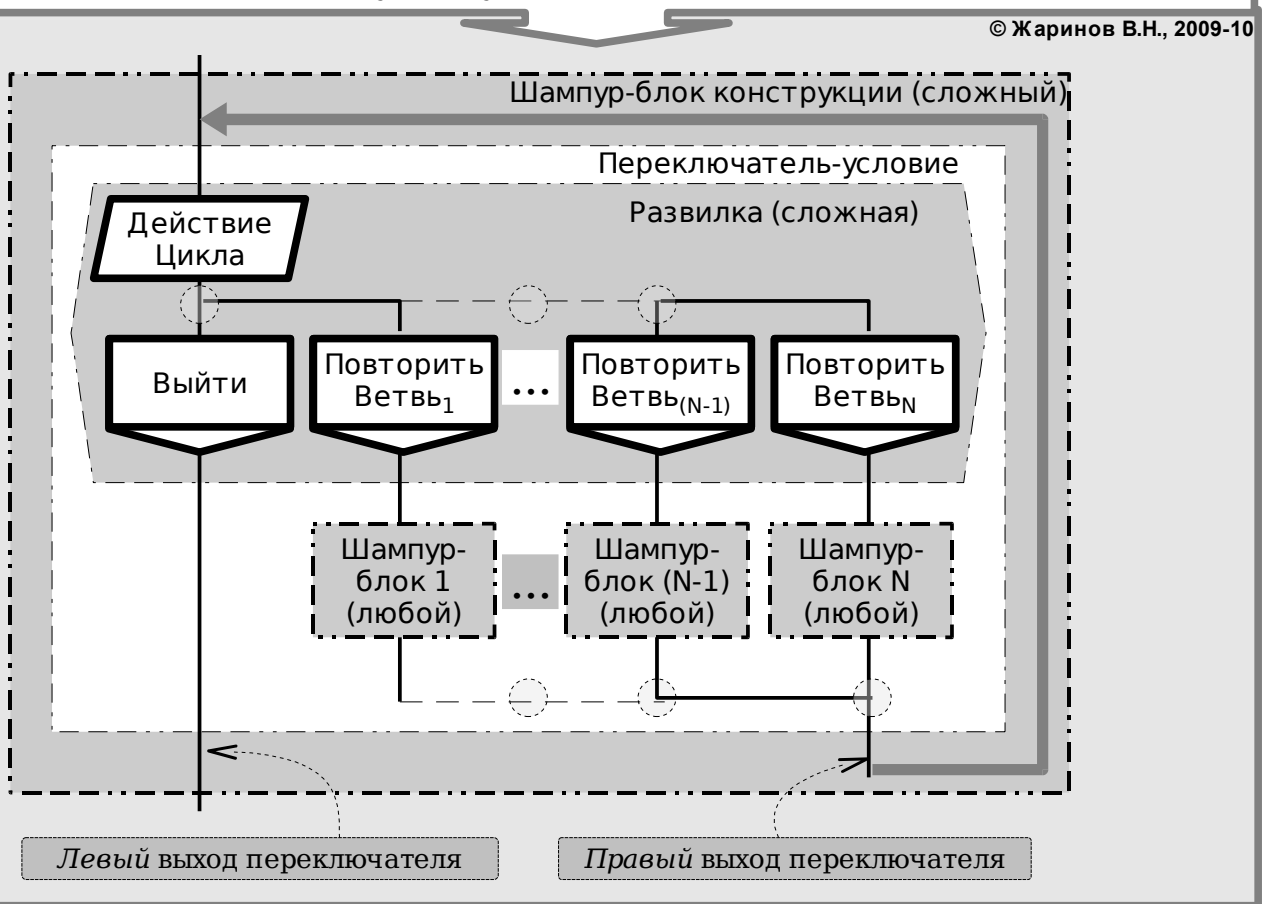
По сути, здесь выбирается один из возможных маршрутов, но не однократно, как в сложном ветвлении, а с возможностью повтора. Эта возможность определяется анализом состояния при очередном проходе условия; при этом каждая развилка в общем случае проверяет величины состояния (переменные алгоритма), существенные для выбора «своего» маршрута.

Очевидно, на этой конструкции можно построить маршрутную логику любого алгоритма. При этом в алгосостоянии возникает дополнительная переменная выбора следующей ветви.

А можно ли устранить возникшие пересечения? Ответ нам подсказывает запись цикла Паронджанова через цепочку развилок – попробуем сдвинуть содержимое ветвей «по кругу» относительно вертикалей на одну позицию вправо – т.е. та ветвь, что была на первой вертикали, д.б. на второй и т.д... а то, что было на последней вертикали, окажется на первой. При этом нужно также добавить новую развилку, чтобы сохранить смысл вопросов выбора ветвей, а соответствующая «любому другому» значению крайняя правая вертикаль оставляется ненагруженной. Результат показан выше.

Наконец, покажем, как можно представить цикл Дейкстры в виде переключающего.

Цикл Дейкстры можно построить на базе переключателя, подобно переключающему циклу Паронджанова; см. ниже



Практически цикл Дейкстры можно визуализировать на базе переключателя, добавляя в макроикону Переключающий цикл варианты изнутри справа (тогда как в цикле Паронджанова нужно добавлять слева).

Цикл Дейкстры как переключающий

Сопоставление этих циклов показывает, что цикл Дейкстры обеспечивает выбор варианта содержания в каждой итерации, тогда как цикл Паронджанова даёт возможность выбора варианта лишь при выходе; итерация же всегда одинакова по содержанию.

Интересен вопрос: а как выбирать нужную ветвь? Очевидным представляется использовать порядковый номер ветви как значение переменной выбора; тогда вопрос для очередной ветви сводится к проверке равенства этого значения её номеру. Формируется значение в конце очередной ветви как результат т.н. (восстановления/продвижения) инварианта цикла; проще говоря, сочинитель определяет зависимость номера следующей ветви цикла от результатов (со-

стояния алгоритма), полученных при исполнении текущей ветви. Процедура вычисления по этой зависимости оформляется в конце ветви и получаемое значение присваивается переменной выбора; далее в развилках по цепочке происходит просмотр и выбор нужной ветви (а при определённом значении – выход из цикла Дейкстры).

Понятно, что текущая ветвь м.б. нелинейной; тогда номер следующей ветви может зависеть и от маршрута, по которому текущая исполнялась на этот раз (возможно, простейшим образом – в конце маршрута переменной выбора просто присваивается номер следующей ветви без дополнительных вычислений).

По сути, в цикле Дейкстры можно структурировать содержимое на логические части так же, как в дракон-силуэте; только выбор части условный, а каждая часть м.б. только шампур-блоком. Логика же получается равносильной; роль адреса ветки в «подвале» силуэта играет процедура вычисления номера, в общем случае разная на разных маршрутах внутри ветви. В то же время условность перехода даёт возможность выбора по более сложным основаниям, чем просто маршрут, пройденный в текущей ветви; в пределе можно анализировать алгосостояние целиком. В силуэте аналогичная возможность обеспечивается созданием в конце ветки адресного блока, в котором происходит условный выбор одного из маршрутов, ведущих в «подвал», а далее безусловно по петле силуэта – на нужную часть.

Понятно, что любой из рассмотренных сложных циклов (а равно и любой простой цикл, который можно выделить в составе сложного) м.б. типа ЖДАТЬ.

> Мы для простоты не показывали в петлях циклов икону Период. Однако попытка показать её выявит неравнозначность схем с общей и раздельными петлями вложенных циклов — в первом случае мы приходим на каждом выходе из цикла к одной и той же иконе Период, во втором — для каждого выхода к своей.
>
> Какое изображение правильное? Очевидно, с раздельными петлями — оно исходит строго из шампур-синтаксиса дракон-схем. Отсюда видно, как важно при визуализации целостно определять смысл исходных конструкций.

Преобразуя структуры, получаемые как комбинации простых и сложных циклических конструкций, получаем произвольные циклы.

2.2.3.1. Произвольные циклы: матрёшки и пересадки лиан

Цикл с досрочным выходом является допустимой в техноязыке произвольной циклической структурой. К такому циклу приводит необходимость произвольного ветвления на пути к петле цикла и/или к досрочному выходу, а также нескольких досрочных выходов (точек слияния с главной вертикалью, разделённых шампур-блоками). Общая цель – учесть факторы, возникающие по ходу выполнении тела цикла, как условия окончания повторения и/или изменения маршрута в блоке ПОКА (пропуска каких-то частей блока). Досрочные выходы образуются вливанием в главную вертикаль конструкции ниже основного выхода из цикла вертикалей отдельных развилок блока ПОКА; остальные вертикали ведут в петлю цикла. Для вливания пересаживаются лианы; т.о. образуется разновидность лианного макроблока дракон-схемы. Точек слияния с петлёй м.б. несколько; в любом случае это соединительные вершины, связанные между собой горизонтальными и/или ломаными линиями (конечно же, не содержащими икон).

Как и для произвольных ветвлений, для произвольных циклов можно придумать лишь частные иллюстративные примеры, см. напр. /1, Гл.9/.

Видно, что ранее показанный переключающий цикл Пароннджанова можно рассматривать как частный случай цикла с единым досрочным выходом по общей переменной условия и с ограничением на размещение шампур-блоков тела относительно развилок.

Кроме того, возможны **произвольные циклы в цикле**, как иерархия вложенных друг в друга простых и/или произвольных циклов.

> Как и произвольные ветвления, произвольные циклы образуются из матрёшек; здесь они включают макроиконы циклов и, возможно, макроиконы ветвлений в блоках тел циклов.

2.3. Базовые структуры деятельности в техноязыке

Как уже говорилось, одной из ключевых идей, «изюминок» техноязыка является существование различных изображений структуры ГСА на плоскости, или символических диоформ визуалов – примитива и силуэта. Изначально даются их исходные формы – заготовки дракон-схем; одно из назначений заготовки – задавать границы шампура. Рассмотрим теперь каждую из диоформ в целом, опираясь на то, что мы узнали в предыдущем подразделе о подструктурах, составляющих содержание визуалов, т.е. входящих в шампуры.

2.3.1. Примитив

Эта диоформа похожа на обычную блок-схему: заголовок и конец, между которыми помещается тело – шампур-блок содержания визуала (пока оно нас не интересует, поэтому замещено одним блоком).

Визуал формы примитива – общий вид и способ образования

Примитив произвольной структуры мы рассмотрим позже.

Суммируем основные свойства базовых конструкций для примитива.

Следование изображается упорядочением икон в цепочку, соединённую звеньями. Формально при этом происходит ***естественный переход*** исполнителя алгоритма от текущей иконы к следующей в цепочке «по шампуру», т.е. по вертикали сверху вниз.

Формула маршрута следования – это слово, составленное из буквенных обозначений икон.

Горизонтальные линии в начале и конце любой побочной вертикали являются частью соответствующего звена (естественного перехода).

Ветвления изображаются посредством *развилок* – визуальных операторов, имеющих один вход и несколько выходов (плеч). Простейшая развилка строится на основе иконы И4, сложная – макроиконы 3. Формально при этом происходит ***условный переход*** на один из выходов в зависимости от результата проверки некоторого условия.

Если алгоритм содержит ветвления, то в его дракон-схеме можно выделить элементы:

- *главная вертикаль* – проходящая от выхода иконы *Заголовок* ко входу иконы *Конец*, остальные вертикали считаются *побочными*.

- *лиана* – соединительная линия или последовательность шампур-блоков, началом которой является выход иконы *Вопрос* или *Вариант*.

Проверка условия ветвления происходит в условном операторе, изображаемом иконой И4 либо «шапкой» макроикон 3/11 или 5/13. В иконе *Вопрос* записывается т.н. *да-нетный* вопрос, т.е. такой, на который можно дать ответы «Да» или «Нет». Для переключателя составляется т.н. *тематический* вопрос, т.е. такой, на который можно дать ряд конкретных ответов (перечень из двух и более понятий – вариантов); предмет вопроса записывается в иконе *Выбор*, а каждый из вариантов – в своей иконе *Вариант*.

Циклы, вводимые на основе иконы И24, изменяют порядок исполнения; в петле цикла движение идет в направлении, обратном естественному – снизу вверх и справа налево, что отмечается стрелкой.

Формально цикл можно представить как возвращение на уже пройденную точку маршрута, но в иной ситуации (состоянии исполнителя), чем при предыдущем проходе.

Петля цикла может начинаться не сразу от выхода развилки, а после некоторого шампур-блока, нагружающего побочную вертикаль (такая структура называется *цикл ДО*); аналогично и между концом петли цикла и входом развилки м.б. не пустое звено, а шампур-блок (такая структура называется *цикл ПОКА*); если имеет место и то, и другое, имеем *гибридный цикл*.

При наличии ветвлений в визуале имеется два и более маршрутов, проходящих через разные вертикали и плечи развилок. В смысловом визуале (иконы которого заполнены конкретным текстом) один из маршрутов считается *главным*; он выбирается разработчиком схемы, обычно по значимости для целей алгоритма.

Формулы маршрутов разветвлённого визуала записываются с участием значений выходов ветвлений (ответов на вопросы); очевидно, каждому маршруту будет соответствовать уникальный набор ответов на все вопросы.

Все побочные вертикали и петли циклов упорядочиваются строго вправо от главной вертикали, из которой они исходят. Тем самым исполнение визуала соответствует естественному порядку чтения дракон-схемы – справа налево и сверху вниз (за исключением прослеживания циклов).

Исходя из визуального порядка вертикалей ветвления (цикла), определяется и положение горизонталей рёбер, ограничивающих побочную вертикаль – всегда справа от главной.

Теперь коснёмся структуры примитива в целом. Она м.б. линейной или нелинейной (содержащей ветвления и/или циклы).

Изначально нелинейная структура тела примитива образуется как матрёшка – вложение одних макроикон ветвления в другие. Из схем переключающего цикла видно, что матрёшки образуются и как комбинации ветвлений с циклами. Нам также знаком ещё один случай матрёшек – циклы в цикле.

> *Матрёшка* в общем случае определяется как вложение нелинейных структур (ветвлений и/или циклов) друг в друга.

В **примитиве реального времени** употребляются иконы РВ, не являющиеся шампур-блоками. Такая икона *Период* может располагаться в петле цикла; кроме того, такие иконы *Синхронизатор* м.б. присоединены к некоторым шампур-иконам (макроиконам). Тем самым образуются макроиконы реального времени (по стандарту языка имеют номера 7…20).

Кроме обычных примитивов, возможны и произвольные.

2.3.1.1. Произвольный примитив: макроцикл и пересадки лиан

Произвольным примитивом будем считать визуал с нелинейным телом, полученный из заготовки-примитива как вставкой макроикон и шампур-икон, так и преобразованием структуры связей в теле. Это общий случай данной диоформы визуалов.

Тело в произвольном примитиве состоит из макроструктурных блоков следующих видов:

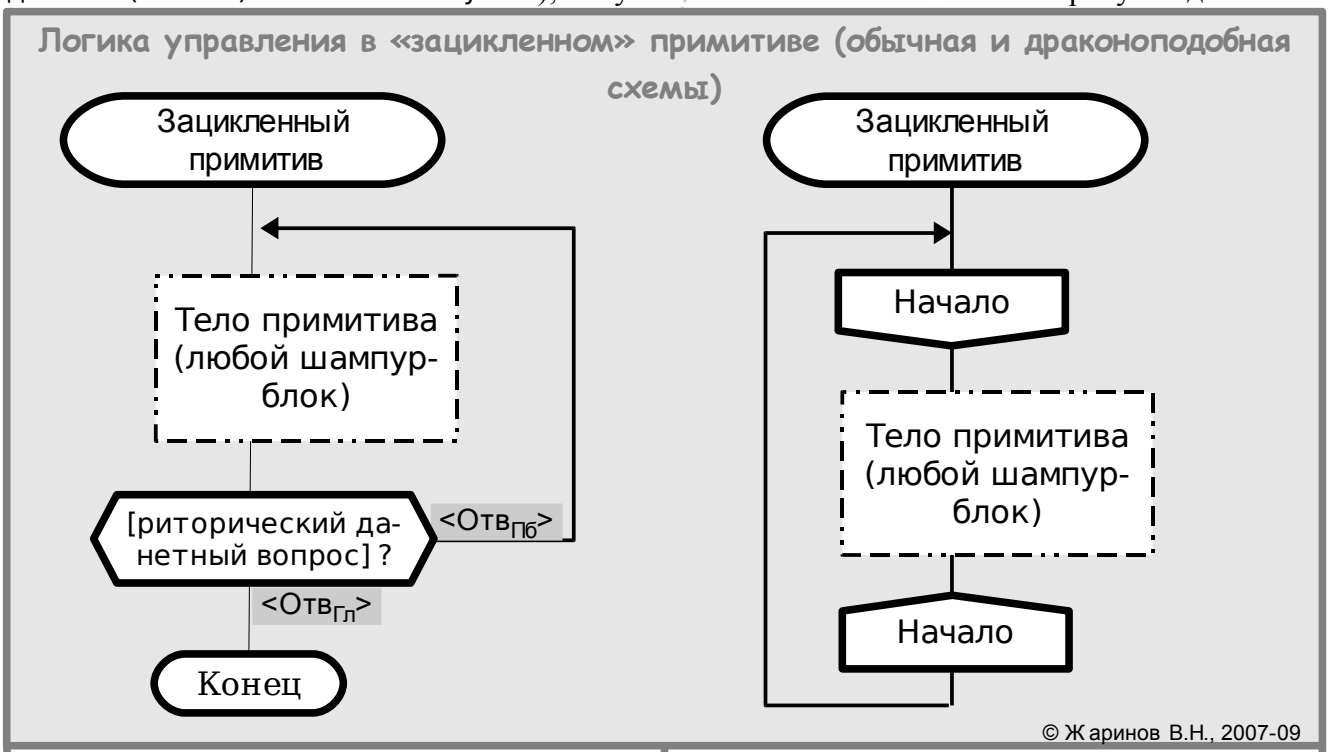
- *структурные* – организуются методом добавления к шампуру и/или побочным вертикалям икон (макроикон) между уже имеющимися;
- *лианные* – образуются из дракон-схемы, содержащей ветвления, путём выполнения модификации, называемой пересадкой лианы.

> *Пересадка лианы* – отрыв нижнего конца лианы и присоединение его в другой точке дракон-схемы при соблюдении определённых условий (см. /2, гл. 15, Тезис 28/).

Очевидно, что после пересадки изменится совокупность маршрутов визуала (и их формул). Пересадке лианы может сопутствовать переформулировка условных вопросов (и вариантов ответа).

В частном случае (если тело примитива начинается с ветвления и в нём определённым образом пересаживались лианы) лианным блоком становится всё тело.

В примитиве, кроме того, возможно «зацикливание» алгоритма операцией Удаление конца примитива; при этом образуется макроцикл – неявный (не заданный через макроикону) цикл по всему телу до верхней границы шампура. Этот макроцикл мы можем представить себе как явный цикл, вопрос которого д.б. *риторическим*, т.е. иметь в любой ситуации выполнения единственный верный ответ, приводящий на петлю (наиболее очевидная формулировка – Продолжать (или нет) выполнение визуала?); полученная схема показана слева на рисунке далее.



Дракон-схема визуализирует логику «зацикливания» примитива как обычный цикл по всему шампуру; вопрос цикла ставится так, что при исполнении визуала он всегда должен иметь ответ <Отв$_{Пб}$> (потому он и назван *риторическим*); иначе говоря, происходит *безусловный переход* в начало алгоритма

Драконоподобная схема показывает, что реально происходит в примитиве с удалённым концом при достижении выхода тела; иконы *Имя ветки* и *Адрес* в совокупности с петлёй безусловного перехода назад (расположена вправо от вертикали, чтобы отличать её от петли цикла) изображают возврат

Варианты визуализации логики «зацикленного» примитива

Указанную схему можно считать корректной, но вообще-то ответ на риторический вопрос бессмыслен. Фактически при удалении конца условный переход на него превращается в ***безусловный переход назад*** (возврат управления) к началу алгоритма. Это можно изобразить, используя иконы *Имя ветки* и *Адрес*, визуализирующие безусловный переход в техноязыке (см. схему справа на рисунке выше). Эта схема может считаться лишь драконоподобной, поскольку БП назад в техноязыке запрещён.

2.3.2. Силуэт

Силуэт – форма алгоритма на диосцене, допускающая безусловные переходы с определёнными ограничениями (т.н. разрешённые БП). Их совокупность организована как специальная икона техноязыка *Петля силуэта*.

Данная диоформа считается наиболее общей для алгоритма.

> *Силуэт* – форма сложного визуала, подразделяющая его на смысловые части – ветки и петлю силуэта – которые образуют метаструктуру деятельности.
>
> *Ветка* – составной оператор языка ДРАКОН, состоящий из начала, тела и конца. Тело ветки имеет один вход и один или более выходов. Вход обязательно соединен с петлёй силуэта через икону *Имя ветки*, а выход – через икону *Адрес*.
>
> *Петля силуэта* – икона, визуализирующая обобщённые безусловные переходы между выходами веток и их входами.

Понятие главного маршрута применительно к силуэту уточняется:

> *Главный маршрут силуэта* – последовательность маршрутов поочерёдно работающих веток.

Замыкающая стрелка петли обозначает направление перехода – снизу вверх и слева направо (ср. с петлёй цикла), а также отмечает начальную точку исполнения визуала – со входа в крайнюю левую ветку. Горизонталь, лежащую в петле справа от этой стрелки, будем называть *заглавной*; вместе с иконой *Заголовок* и комплектом икон *Имя ветки* она образует т.н. *шапку* силуэта, которая позволяет оперативно уяснить метаструктуру описываемой деятельности (её название, число частей и имя каждой части). Нижнюю горизонталь петли называют «*землей*» силуэта; вместе с комплектом икон *Имя ветки*» она образует т.н. *подвал* силуэта.

Силуэт имеет две и более веток; в каждой иконе *Адрес* пишется имя одной из веток данного визуала. Мы можем обобщённо показать структуру дракон-схемы в этой форме (см.).

Как показано на схеме, крайняя правая ветка м.б. *конечной*; тело её представляет собой шампур, соединяемый не с петлёй, а с иконой *Конец*. Остальные ветки будем называть *полными*.

Тело полной ветки обычного силуэта образуется так же, как и шампур в обычном примитиве.

> *Шампур ветки* – вертикаль, соединяющая икону Имя ветки с иконой Адрес .

Рассмотрим отличия базовых алгоритмических конструкций в силуэте от примитива и сопутствующие ограничения.

В силуэте **следование** отображается, кроме соединения смежных икон звеньями вертикали, также через совпадение имён в иконах *Имя ветки* и *Адрес* разных веток. Тем самым происходит безусловный переход по петле силуэта от ветки к ветке, вплоть до конечной.

Ветвление возможно внутри ветки, т.е. побочные вертикали, начинающиеся в данной ветке, должны в ней и заканчиваться; кроме того, через заземлённые лианы возможен условный переход с побочных вертикалей на другие ветки. *Передачи управления от выходов развилок напрямую на другие ветки, минуя петлю силуэта, запрещены*.

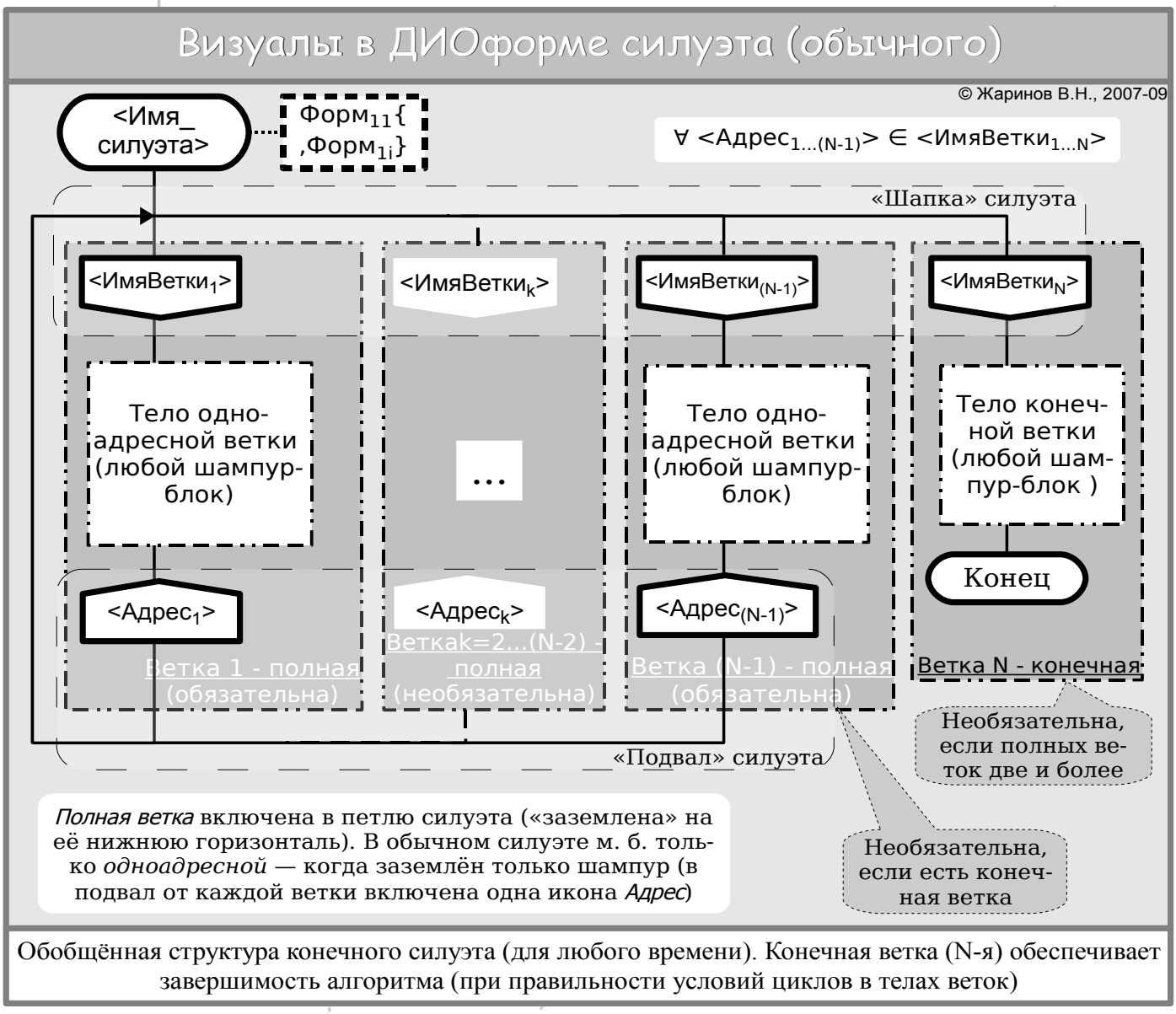
Цикл также может иметь другую форму. В частном случае имя в иконе *Адрес* совпадает с именем, присвоенным данной ветке; тем самым образуется макроцикл, называемый *веточным* циклом.

Силуэт можно описать на текстовом языке; разумеется, в него следует ввести операторы, соответствующие иконам *Имя ветки* и *Адрес*. Пример описания силуэта на псевдокоде /1, с.91-92/.

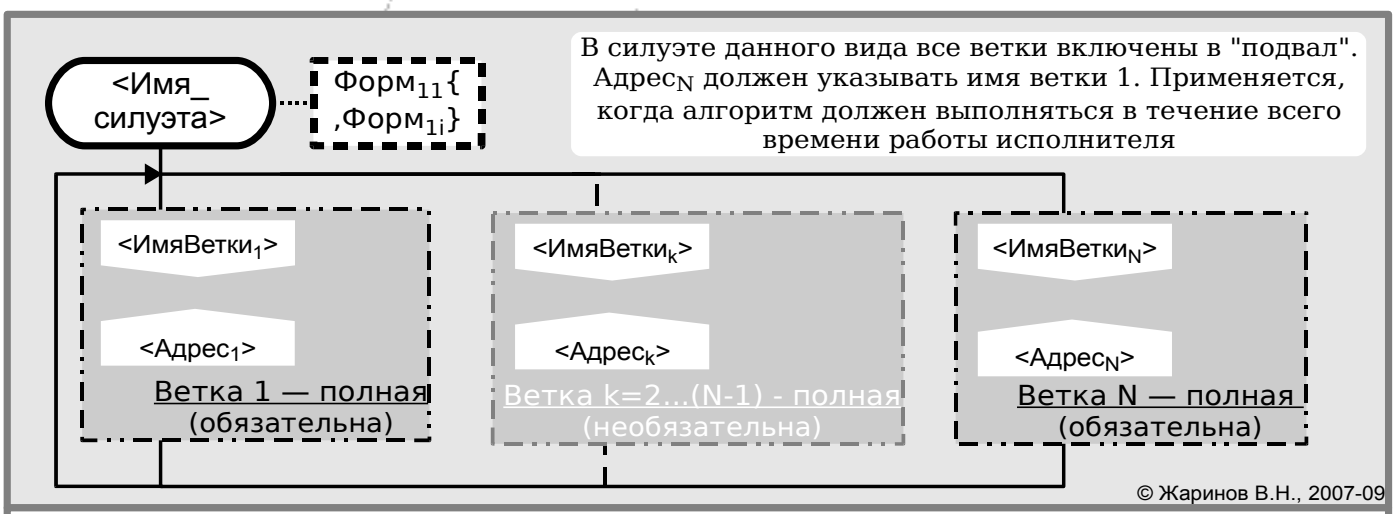
В **силуэте реального времени** допускается употребление в телах веток рассмотренных для примитива шампур-икон и макроикон РВ. Кроме того, конечная ветка может отсутствовать; тогда силуэт является "зацикленным", т.е. выполняется бесконечно (практически – пока может работать исполнитель алгоритма). Общий вид данного силуэта представлен далее.

Исходной формой силуэта является *заготовка-силуэт*, имеющая только начальную и конечную ветки, тела которых замещены точками ввода. Заготовка-силуэт даётся в языке акси-

оматически, как и заготовка-примитив; очевидно, что путём несложной редукции из первой получается вторая.



Обобщённая структура конечного силуэта (для любого времени). Конечная ветка (N-я) обеспечивает завершимость алгоритма (при правильности условий циклов в телах веток)



Обобщённая структура «зацикленного» силуэта (для реального времени). Получается в результате удаления последней ветки конечного силуэта (см. схему выше).

Визуал формы силуэта – обобщённые структуры для обычного вида

Смысл перехода от примитива к силуэту – визуальное структурирование деятельности (алгоритма, техпроцесса) таким образом, чтобы исключить пересечения соединительных линий независимо от сложности структуры. Устраняются и пересечения, неустранимые в примитиве, например, "заходы" петель циклов друг за друга и за побочные вертикали. При этом используется то, что замыкание петли силуэта одной-единственной линией означает всю совокупность переходов от ветки к ветке, которые логически разрешаются по совпадению значений икон *Имя ветки* и *Адрес*.

Примеры перехода от примитива к силуэту см. Каталог примеров.

2.3.2.1. Произвольный силуэт: макроциклы и заземления лиан

К произвольным будем относить силуэты с расширенными возможностями маршрутизации выполнения – множественностью выходов веток и/или входов в визуал.

Тело полной ветки произвольного силуэта, в отличие от тела примитива, может иметь несколько выходов; в этом случае ветка называется *многоадресной*. Один из этих выходов является концом шампура ветки; остальные представляют собой т.н. *заземлённые лианы*.

Кроме структурных и лианных блоков, в макроструктуру ветки входит новый тип блока:

- *адресный* – образуется из структурного блока за счет применения операции Заземление лианы и, возможно, Пересадка лианы.

> *Заземление лианы* – отрыв конца лианы и присоединение его к "земле" силуэта через икону *Адрес* с соблюдением определённых условий (см. /1, Гл. 15, Тезис 29/).

Операция Заземление лианы делает ветку многоадресной. При наличии хотя бы одной многоадресной ветки силуэт является разветвлённым.

В разветвлённом силуэте (с многоадресными ветками) уточняются некоторые понятия, и прежде всего понятие ***главного маршрута,*** что подробно рассмотрено в /1, Гл.6/.

> *Шампур ветки* – вертикаль, соединяющая икону *Имя ветки* с крайней левой из икон *Адрес* той же ветки.
>
> *Главный маршрут силуэта* (разветвлённого) – последовательность маршрутов поочерёдно работающих веток, взятая для набора конкретных выходов каждой ветки.

В разветвлённом силуэте можно образовать веточный цикл . Он может проходить и через последовательность двух и более веток; тогда иконы с совпадающим именем относятся к разным веткам. Началом веточного цикла является начало (икона *Имя ветки*) крайней левой из входящих в него веток, а концом – икона *Адрес* крайней правой.

Как и обычные, веточные циклы м.б. вложены друг в друга (конечно, *N-веточный цикл можно вложить в цикл с числом веток, также не меньшим N*).

Иконы *Имя ветки* и *Адрес*, обозначающие начало и конец веточного цикла, принято помечать специальным образом (возможные варианты см. /2, рис. 51 и 105/). Мы принимаем второй вариант, что видно из используемого алфавита техноязыка.

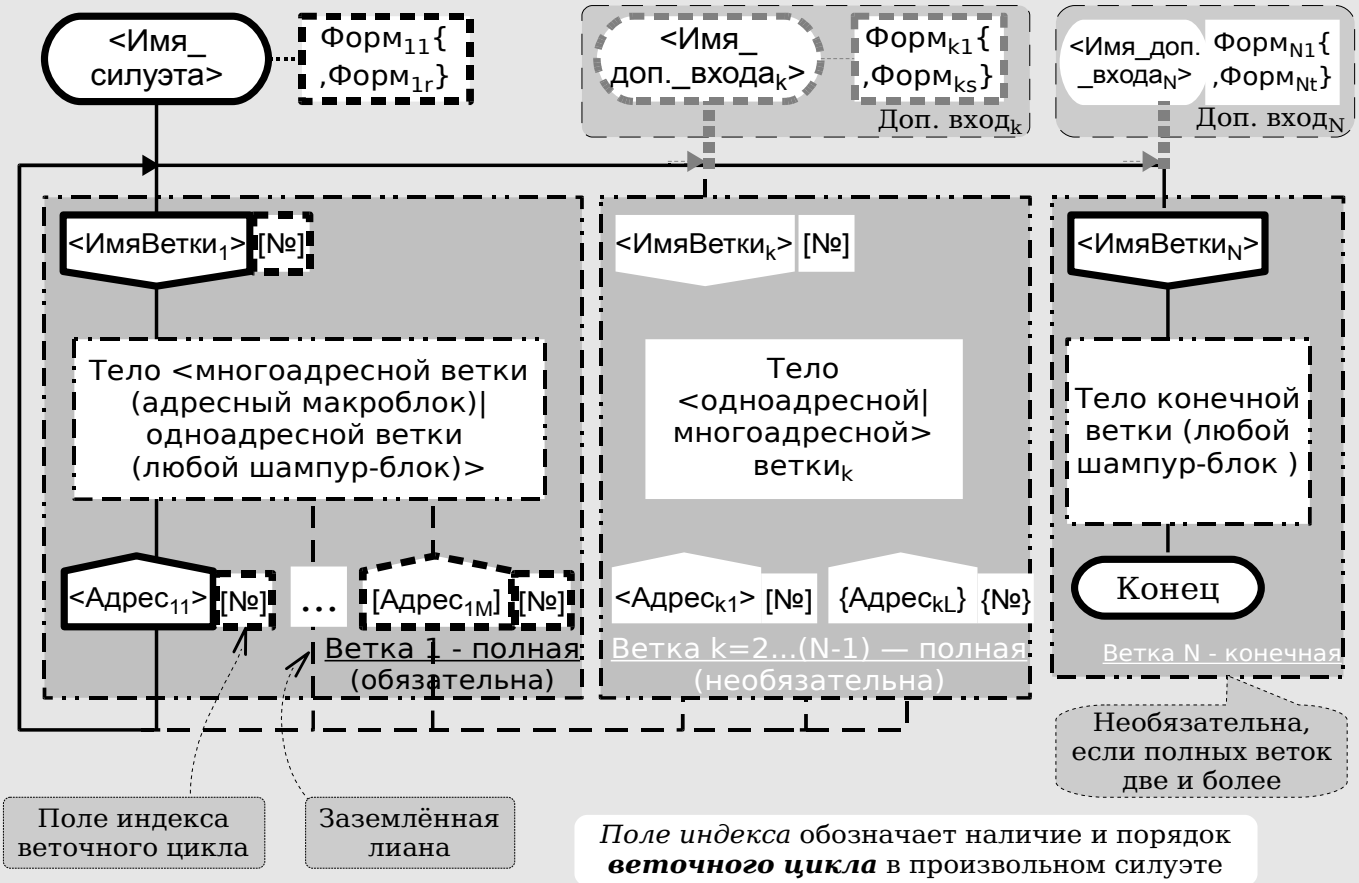
> Вообще говоря, ничто не мешает нам образовать веточный цикл в неразветвлённом силуэте; однако единственный эффект этого будет состоять в "зацикливании" части, на которую он распространяется, в пределе – всего алгоритма, кроме конечной ветки (а для этого более логично удалять её); т.о. нет логичных оснований пользоваться таким приёмом.

К отдельным веткам, кроме главной, можно присоединить иконы *Заголовок* (макроиконы *Заголовок с параметрами*); тем самым образуются ***дополнительные входы*** (допвходы) через эти ветки. *Допвход в веточный цикл возможен только через его начало*.

Также можно преобразовать конечный силуэт к бесконечному операцией техноязыка Удаление последней ветки; ссылки на последнюю ветку в «подвале» изменяются на первую.

Визуалы в ДИОформе силуэта (произвольного)

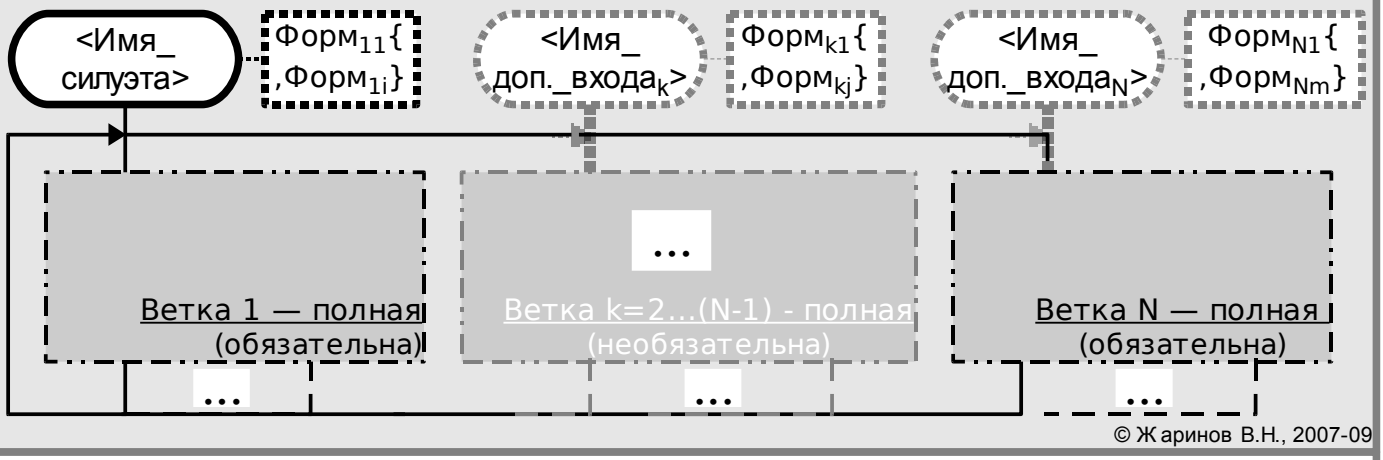
Допвходы возможны *только в начало ветки,* которая *не входит* в веточный цикл либо является *первой* в нём (т.е. её *икона Имя ветки содержит поле индекса*)



Поле индекса обозначает наличие и порядок ***веточного цикла*** в произвольном силуэте

Любая полная ветка в произвольном силуэте м. б. *одноадресной* — когда в подвал включён только шампур либо *многоадресной* — когда включены и другие вертикали. В многоадресной ветке две и более иконы *Адрес,* а тело обязательно разветвлённое

Структура конечного силуэта (для любого времени). Отличается от конечного обычного силуэта наличием допвходов и/или многоадресных веток (где возможны веточные циклы)



Структура «зацикленного» силуэта (реального времени), полученная в результате удаления последней ветки конечного силуэта. Допвходы здесь возможны с теми же ограничениями

Визуал формы силуэта – обобщённые структуры для произвольного вида

6. ИСТОЧНИКИ

Документы для ссылок

1. Пароджанов В.Д. Как улучшить работу ума. Алгоритмы без программистов – это очень просто! – М.: Дело, 2001.

2. Пароджанов В.Д. Занимательная информатика. – М.: Дрофа, 2007.

3. Баранцев Р.Г. Становление тринитарного мышления. – М.-Ижевск: НИЦ «Регулярная и хаотическая динамика», 2005.

4. Фридланд А.Я. Информатика: процессы, системы, ресурсы. – М.: БИНОМ. Лаборатория базовых знаний, 2003.

5. Функциональное моделирование. Методология IDEF0: Стандарт/русская редакция. – М.: МетаТехнология, 1993.

6. С.В. Черемных, И.О. Семенов, В.С. Ручкин. Структурный анализ систем: IDEF-технологии. – М.: Финансы и статистика, 2001.

Полезные ресурсы

mgopu.ru/PVU/2.2/Sprint-Inform – здесь можно изучить современную информатическую терминологию, введенную в /4/.

forum.oberoncore.ru – на форуме обсуждается среди прочего и ДРАКОН как элемент культуры и практический инструмент.

Усилители интеллекта – рассылка, посвящённая средствам облегчения умственного труда и в частности, техноязыку.