Это набросок теории и документации

К ОБОСНОВАНИЮ ЛОГИЧЕСКИХ МЕХАНИЗМОВ И СИСТЕМ МНОГОЗНАЧНОЙ ЛОГИКИ

Аксиоматика логики

Для обоснования логических механизмов необходимо на достаточном уровне формализовать материал, на котором они построены и с которым они работают. Материалом, на котором они построены, являются логические функции, а с которым они работают — истинностные значения (в общем случае, расположенные на отрезке от нуля до единицы).

Для аксиоматизации и должной формализации исчисления истинностных значений введём следующие определения, подобные колмогоровским в теории вероятностей:

Элементарное высказывание — высказывание о чём-либо, которое может быть истинным или ложным (в общем случае допустимы промежуточные вариации);

Истинность — неотрицательная, нормированная к единице мера, заданная на измеримом пространстве высказываний, характеризующая степень истинности высказываний;

Ω — универсум, множество всех возможных элементарных высказываний;

F — сигма-алгебра высказываний на множестве Ω;

T — сигма-аддитивная истинностная мера для $\forall \omega \in \Omega$.

Запишем свойства меры:

1) $\forall A \in \Omega \rightarrow 0 \leq T(A)$

2) $\forall A, B \in F : A \cap B = \emptyset \rightarrow T(A \cup B) = T(A) + T(B)$

3) $T(\emptyset) = 0$

К этим свойствам можно добавить ещё два свойства:

1) $\forall B \subseteq A \rightarrow T(B) \leq T(A)$

2) $A_1 \supset A_2 \supset \cdots : \bigcap_{i=1}^{\infty} A_i = \emptyset \; and \; T(A_1) < \infty \Rightarrow \lim_{i \to \infty} A_i = T(\emptyset) = 0$

Второе добавочное свойство — это непрерывность в нуле, которая следует из первого добавочного свойства, обеспечивающего нам сходимость, показанную во втором.

Зная, что $T(\Omega) = \sum_{\forall x \in \Omega} T(x)$, можно определить меру наибольшего множества и сам метод суммирования мер для пересекающихся и непересекающихся множеств (на самом деле мы уже неявно сделали это при определении понятия истинности). Известно, что истинностное значение в любой математической логике располагается на отрезке $[0,1]$. Тогда можно допустить $T(\Omega) = 1$, из чего следуют две возможности:

1) $\exists! \, x^* \in \Omega : T(x^*) = 1 \Rightarrow \forall x \in \Omega \backslash x^* \rightarrow T(x) = 0$

2) $\nexists x^* \in \Omega : T(x^*) = 1 \Rightarrow \sum_{\forall x \in \Omega} T(x) = 1$

Первая возможность эквивалентна закону исключённого третьего «Если *A* — истинно, то любое не-*A* — ложно», а вторая — утверждению, что сумма всех мер на множестве элементарных высказываний равна единице даже в отсутствие множества с истинностной мерой, равной единице.

Допущение $T(\Omega) \geq 1$ привело бы к тому, что в случае принадлежности универсуму двух множеств, мера одного из которых равнялась бы нулю, мы получили бы множество с мерой, большей или равной единице, что вступало бы в противоречие с канонами математической логики и сделало бы затруднительным определение закона исключённого третьего (также возникли бы трудности в интерпретации истинностей больших единицы).

Введём, аксиоматически и без доказательств, также в стиле Колмогорова условную истинность «Если B — истинно на $T(B)$, то A — истинно на $T(A|B)$»:

$$T(A|B) = \frac{T(A \cap B)}{T(B)}$$

Данное определение нам пригодится чуть далее, потому пока отложим его в сторону.

Теперь для дальнейшего удобства установим следующие соответствия между операциями алгебры множеств на Ω, базовыми логическими операциями и действиями с мерами:

$$\neg_A B \Leftrightarrow A \backslash B \Leftrightarrow T(A) - T(A \cap B)$$

$$A \wedge B \Leftrightarrow A \cap B \Leftrightarrow T(A) \times T(B)$$

$$A \vee B \Leftrightarrow A \cup B \Leftrightarrow T(A) + T(B) - T(A) \times T(B)$$

Здесь мы воспользовались известными свойствами алгебры множеств по типу формулы включений-исключений. Важно отметить, что в случае отрицания мы вычитаем из отрицающего множества именно пересечение отрицаемого и отрицающего, а не само отрицаемое. В случае, когда отрицаемое полностью лежит в отрицающем, вычитается отрицаемое, а в случае, когда эти множества не пересекаются, мы получаем вычитание пустого их пересечения, то есть отрицающее множество остаётся неизменным и мера получившегося результата равна его мере.

Результат для конъюнкции и пересечения, потребовавшийся для дизъюнкции и объединения, уже был ранее неявно определён мерой условной истинности.

Очевидно, что для зависимых A и B мера B должна быть больше нуля, иначе условная истинность $T(A|B)$ не будет существовать. Чтобы вывести меру для $A \wedge B$ положим A и B независимыми и тогда, по определению условной истинности, получим:

$$T(A|B) = T(A) \Rightarrow \frac{T(A \cap B)}{T(B)} = T(A) \Rightarrow T(A \cap B) = T(A) \times T(B)$$

Теперь, обладая основополагающими операциями, мы можем выразить любую логическую функцию (или же операцию — мы полагаем здесь эти понятия эквивалентными), но перед этим оговоримся, что $T(A)^2 = T(A) \times T(A) = T(A \cap A) = T(A)$ по правилам алгебры множеств и ранее определённым соотношениям, то есть мера не изменяется при возведении в степень.

Для удобства, где не оговорено обратное, допустим возможным писать $T(A) = a$, при этом сохраняя за собой право применять к подобного рода записям операции, как логические, так и алгебраические (для унивёрсума условимся писать: $T(\Omega) = \omega$), то есть записи вида $\neg a = (1 - a)$ правомерны для ускорения написания.

Для конъюнкции и дизъюнкции в любой логике мы получаем уже известные соотношения:

1) $A \wedge B = a \times b$

2) $A \vee B = a + b - a \times b$

В случае отрицания вопрос обстоит сложнее. В общем случае его можно записать так:

$$\neg_A B = a - a \times b$$

В классической логике же отрицание имеет только один аргумент, причём важно, чтобы в бинарном и небинарном случае оно переводило единицу и ноль друг в друга. Наиболее просто этого можно достичь, означив отрицающим множеством унивёрсум:

$$\neg B = \neg_\Omega B = \omega - \omega \times b = 1 - b$$

Интересно, что тавтологию (или же тождественное отображение $id_\Omega(x) = x, \forall x \in \Omega$) можно записать тремя путями:

1) $id_\Omega(x) = \neg_x \emptyset = x - 0 = x$

2) $id_{\emptyset}(x) = |0 - x|$

3) $id_{\Omega}(X) = \Omega \cap X = T(X) \times T(\Omega) = x \times \omega = x \times 1 = x$

Второй способ нам пригодится, когда мы будем говорить о гипероперациях; при этом заметьте, что в случае отображения на пустом множестве мы доопределили функцию отображения, так как, в привычном понимании, она на пустом множестве не существует.

Теперь можно привести некоторые логические функции.

$$a \to b = \neg a \vee b = 1 - a + a \times b$$

$$a \downarrow b = \neg a \wedge \neg b = (1 - a) \times (1 - b)$$

$$a|b = a \uparrow b = \neg(a \wedge b) = \neg a \vee \neg b = (1 - a) + (1 - b) - (1 - a) \times (1 - b)$$

При желании вывод функций или же логических операций можно продолжать.

Итогом по аксиоматизации логик может являться то, что логика с любым заданным числом значений укладывается в эту простейшую модель, при этом на множестве 0 и 1 приведённая модель сводится к классической булевской логике.

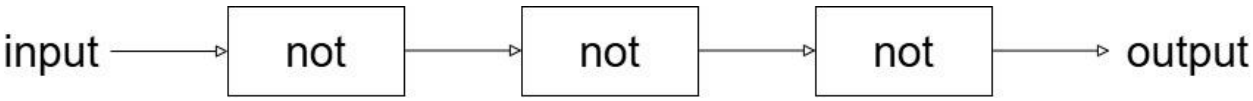
Логики с разными числами истинностных значений возникают сами собой в тех или иных пространствах элементарных высказываний.

Теперь можно перейти к обоснованию логических механизмов.

Модель логических механизмов

Логическим механизмом назовём любую последовательность операций, применённую к входным данным так, что выход предыдущей операции подаётся на вход последующей. Например, последовательность трёх отрицаний, применённая к входному значению может являться логическим механизмом.

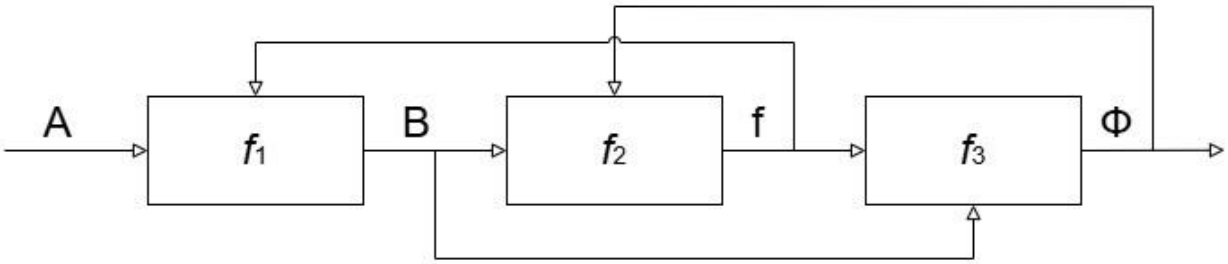
Применение к входному значению конъюнкции с некоторым фиксированным значением тоже может быть логическим механизмом. Проиллюстрируем два этих примера.



Три последовательных отрицания

Три конъюнкции входа с некоторым X

Формализация механизмов подобного рода не представляет никакого математического интереса. Гораздо более интересные задачи ставит реляционная биология и (M,R) системы, как её формализация для случая одной клетки. Известна модель логического механизма с самоопределением операций, представленная ниже.



Механизм на основе (M,R) системы Роберта Розена (усечённый)

Механизм на основе (M,R) системы Роберта Розена отличается от известных нам механизмов тем, что в нём происходит самоопределение операций. Выходы трёх функций подаются не только на входы друг друга, но ещё и меняют друг друга, как указано на усечённой схеме (далее станет ясно, почему она усечённая и будет представлена полная схема).

Для начала определимся, что для достижения самоопределения операций в рамках любого заданного логического механизма нужно поместить в функциональные блоки на схеме такие операции, которые могут меняться в

зависимости от управляющих значений (показаны вертикальными стрелками на схеме). Математически такие управляющие значения ничем не будут отличаться от обычных аргументов функции, задающей операцию.

Таким образом, ***гипероперация*** — это операция, которая в зависимости от значения управляющего параметра/аргумента становится той или иной логической операцией.

Введём и понятия *потоков*. ***Поток*** — это, в терминах механизма, последовательность пересылки значений с выходов функциональных блоков на входы после отработки соответствующих блоков. В терминах логических функций — это простая пересылка получившихся результатов в аргументы.

Введённые понятия потока и гипероперации позволяют нам теперь определить уточняющие понятия:

> ***Поток на преобразование*** — последовательная отработка функциональных блоков с передачей значения с выхода на обрабатывающий вход.
>
> ***Поток на изменение*** — последовательная пересылка значений с выходов на управляющие/изменяющие входы с проведением соответствующего изменения блоков, на которые пришло управляющее значение.

Для примера запишем поток преобразования/обработки, описываемый следующим образом: «Три блока последовательно обрабатывают значение *x*; на выходе получаем значение *y*».

$$x \to 1 \to 2 \to 3 \to y$$

Числа — это номера функциональных блоков, в которых лежат операции (вообще-то более точно говорить «определённые гипероперации», но мы упрощаем для простоты изложения).

Теперь запишем поток управления или же поток на изменение, который описывается так: «Блок 3 изменяет блок 2, блок 2 изменяет блок 1, а блок 1 изменяет блок 3».

Сначала определимся, что речь идёт, фактически, о том, что каждый последующий изменяет предыдущий. Для первого предыдущего нет, потому он изменяет третий, являющийся первым в описанном потоке. Очевидно, что изменение внутреннего устройства блоков (их функций, переопределение состояний их гипероперациий) никак не затронет то, что располагается на их выходах после отработки. В случае, когда наши потоки разделены, то есть сначала преобразуем входы, а потом блоки, порядок изменений блоков не важен, но будем придерживаться описания.

Так как у нас здесь ничто не входит и не выходит, и все преобразования происходят в уже заданной системе, то записать такой поток можно следующим образом:

$$3 \to 2 \to 1 \to 3$$

Стоит заметить, что несмотря на тройку в начале и конце зацикливания не происходит — оно происходило бы, если бы после второй тройки стояла бы стрелка с многоточием после неё.

После определения всех необходимых понятий можно перейти к приведению примеров гипероперациий и формальному определению «самоопределения операций».

Нетрудно заметить, что некоторые логические функции «симметричны» (неформально говоря).

Например, тавтология и отрицание:

A	$\neg A$
0	1
1	0

A	id(A)
0	0
1	1

Или конъюнкция и стрела Пирса:

A	B	and
0	0	0
0	1	0
1	0	0
1	1	1

A	B	↓
0	0	1
0	1	0
1	0	0
1	1	0

Гипероперацией является такая формула, которая при крайних (0 или 1) значениях управляющего параметра вырождается в одну из «симметричных» функций по типу тех, что представлены выше.

Для отрицания и тавтологии (*NegTau*) этой формулой является:

$$NegTau(x, c) = |c - x|$$

где c — управляющий параметр, а x — аргумент.

Для конъюнкции и стрелки Пирса (*ConPir*) этой формулой будет:

$$ConPir(a, b, c) = |(c - a) \times (c - b)|$$

где c — управляющий параметр, а a, b — аргументы.

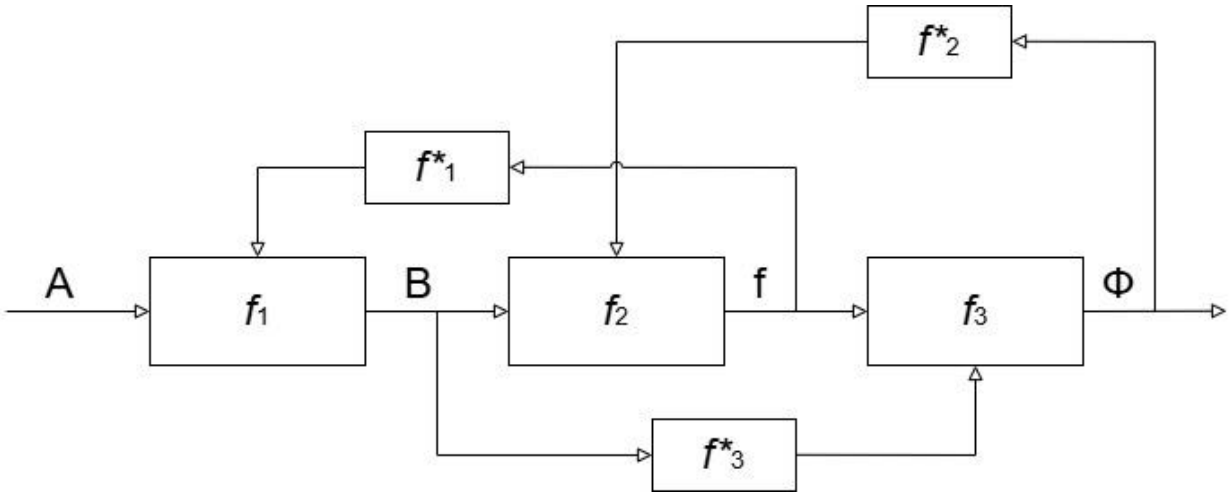
Нетрудно увидеть, что в случае, когда единица — управляющий параметр, мы получаем отрицание и стрелку Пирса в представленных гипероперациях, а когда управляющий параметр — ноль, благодаря модулю (абсолютному значению), получаем тавтологию и конъюнкцию соответственно.

Таким образом, мы обладаем уже двумя гипероперациями, на которых можем построить логический механизм, но будет ли этого достаточно для достижения самоопределения операций, как того требует минимальная модель живой системы?

Если мы будем просто так передавать значения между блоками, то ничего интересного не добьёмся — лишь создадим усложнённую композицию функций с сложными внутренними отношениями, однако мы в любом случае сможем просчитать все дальнейшие состояния логического механизма по начальным условиям. Проще говоря, такая система (системой называем механизм) всецело будет зависеть от начальных условий и определяться ими.

Чтобы избежать такой детерминированности и хоть чуть-чуть уподобиться живым системам, следует ввести вероятность, как фактор, влияющий на управляющие значения (можно, конечно, сделать стохастическими сами функциональные блоки, но тогда система выродится в генератор случайных чисел с очень сложной начинкой, а оно нам не очень нужно).

Ниже представлена полная схема логического механизма с изменением выходных значений перед тем, как они попадут на управляющий вход (на обработку значения, при этом, идут неизменёнными).



Полная схема логического механизма на основе (M,R) системы Роберта Розена

Функциональные блоки, помеченные звёздочками, то есть блоки меняющие значение перед тем, как оно попадёт на управляющий вход, и есть

те самые блоки, в которых содержится стохастическое преобразование вошедших в них значений.

Пока отвлечёмся от их устройства и уточним, что теперь запись потока на управление (или изменение) должна включать в себя и блоки, преобразующие управляющие значения. Тогда уже известная нам запись примет вид:

$$3 \to 2^* \to 2 \to 1^* \to 1 \to 3^* \to 3$$

Таким образом, одна итерация работы механизма (когда отрабатывают все шесть показанных блоков в данном случае), со схемы записывается следующим образом:

$$x \to 1 \to 2 \to 3 \to y$$

$$3 \to 2^* \to 2 \to 1^* \to 1 \to 3^* \to 3$$

Здесь ***итерацией работы*** назовём полный проход по двум потокам, что эквивалентно отработке всех блоков механизма, как мы назвали это в данном случае.

Не зря мы использовали термин итерация, отсылающий к циклам. Механизм можно зациклить следующим образом:

$$x \to 1 \to 2 \to 3 \to x$$

Можно после прохода по потокам подавать на вход первого блока выход третьего блока и продолжать так раз за разом, пока управляющие параметры, лежащие в блоках (c, определённые на предыдущей итерации) не повторятся (каждую итерацию мы можем записывать текущее значение параметров, а затем останавливать работу механизма, если две записи повторились).

Вообще функцией, лежащей в блоках со звёздочкой может быть любая функция, которая переводит отрезок от нуля до единицы в себя же случайным образом (не обязательно, чтобы область значений функции покрывала весь отрезок).

Такой функцией может являться, например, распределение Бернулли, вероятностью выпадения единицы из которого принимается значение с выхода меняющего функционального блока. Затем ноль или один, выпавшие с заданной вероятностью, подаются на управляющий вход изменяемого блока.

Либо же можно задать такую функцию:

$$P(f(x) = 1) = \frac{x}{2}, \qquad P(f(x) = 0) = 1 - \frac{x}{2}$$

То есть функция $f(x)$ с вероятностью $\frac{x}{2}$, дающая единицу. Установив её область определения, равной отрезку [0,1] в бинарном случае получим, что при нуле на выходе изменяющего блока изменяющее значение будет гарантированно нулём, а при единице — единицей с вероятностью 0.5.

Теперь у нас есть механизм, потоки, логические функции, а также стохастические функции, позволяющие добавить случайность, зависящую от выходных значений внутри механизма. Теперь мы можем формально определить, чем является самоопределение операций в рамках логического механизма.

Самоопределение операций — это определение состояний блоков логического механизма такое, что оно полностью не зависит от начальных условий и внешних условий, но при этом стохастически определяется внутренними выходными значениями; в системе с самоопределением операций принципиально невозможно предсказать, какое её состояние будет следующим, при этом эта система неразложима на части без потери свойства самоопределения операций и возможности каждой из её частей продолжать работать так же, как внутри системы.

Здесь можно сразу озвучить несколько необходимых, но недостаточных условий наличия самоопределения операций в подобного рода механизмах:

1) Механизм должен пройти, как минимум, две итерации цикла работы, чтобы самоопределение возымело эффект;

2) Поток на управление не должен быть сонаправлен с потоком на обработку;

3) В потоке на управление должны быть задействованы случайные процессы, зависящие от потока на обработку.

Лишь в совокупности эти три условия позволяют нам говорить о самоопределении операций в рассматриваемых логических механизмах. Чтобы доказать, что это так, попробуем отрицать по отдельности каждое из них.

В случае первого всё предельно очевидно — при одном «проходе» (одной итерации работы) мы просто не используем то, что получилось в результате потока на управление, даже при соблюдении второго и третьего условий.

В случае третьего условия всё так же предельно ясно — отсутствие вероятности сделает все происходящие внутри процессы внешне определёнными; отсутствие же связи вероятности с потоком на обработку превратит систему во внешнее определённую с встроенными генераторами случайных чисел или же шума.

Второе условие более интересно и менее очевидно. Представим потоки:

$$x \to 1 \to 2 \to 3 \to y$$

$$3 \to 1^* \to 1 \to 2^* \to 2 \to 3^* \to 3$$

Блоки поочерёдно отрабатывают и поочерёдно меняются (то есть меняются лежащие в них функции, управляющим значением определяются гипероперации). Это приведёт к тому, что значение, пройдя через блок поменяется и пойдёт дальше, но перед этим «оставит след», предопределит возможность случайного изменения блока, в котором было. Фактически, мы тогда получим не систему блоков, взаимно меняющих друг друга на основе

того, что с ними происходит, а последовательность обрабатывающих блоков, которые сами меняются случайным образом. Блоки и блоки со звёздочкой, обладающие одними и теми же номерами в потоках, можно будет, семантически, склеить. Так получим, что перед тем, как войти в блок, значение сначала скажет, должен ли тот измениться после того, как оно в нём обработается. Это разрушит саму концепцию неразложимости системы на части и сложной взаимосвязи её компонентов.

Так мы приходим к тому, что ни одно из условий нельзя выбросить без потери самоопределения операций. При желании можно обозначить это символами, но мы воздержимся от этого.

Стоит при этом заметить, что логические механизмы, в привычном нам понимании, не проводят никаких вычислений, потому сравнивать их с абстрактными вычислителями по типу машин Тьюринга, Поста и Минского бессмысленно. Что же касается темпоральных клеточных автоматов, то распечатка состояний функциональных блоков может претендовать на визуализацию изменения состояний взаимодействующих друг с другом клеток во времени, однако сама идея такого сравнения не совсем ясна.

Если же говорить об автопоэтическом поведении, то описанное самоопределение операций как раз им и является.

Что же касается циклов, то они возможны в детерминированном случае (потому остановка при повторении состояний именно в нём имеет особый смысл, сигнализируя о том, что система зациклилась). Можно назвать стационарной точкой цепочку из одного повторяющегося состояния, а предельным циклом — цепочку из двух и далее состояний.

В стохастическом случае механизм просто «прыгает» между последовательностями, которые проходил бы, начиная с тех или иных начальных условий, потому формирование ярковыраженных циклов в стохастическом случае невозможно, только если не брать случай с теоремой о

бесконечных обезьянах, когда мы ждём достаточное количество времени, чтобы все состояния повторились, включая заранее заданные цепочки.

Можно при этом поставить вопрос о том, через сколько итераций стохастический механизм переберёт все возможные состояния блоков (их управляющих значений) или не только переберёт, но и повторит их.

Здесь теория пока что заканчивается.

КРАТКОЕ ОПИСАНИЕ КОДА

Честно говоря, не представляю, как описать кратко, чтобы и в статью влезло, и смысл передало. Есть начало попытки с расписыванием всех методов и полей, но там описание выйдет в 5-10 раз длиннее самого кода.

У нас есть класс Block, содержащий в себе поля двух входов (a и b, обрабатывающий и управляющий соответственно), состояния блока и выхода. У этого класса есть методы (функции работающие с полями), отвечающие за преобразование значения с поля a путём его преобразования логической функцией, в определении формулы которой используется значение из поля состояния блока; полученный результат помещается значением в поле выхода. Этих методов, преобразующих вход в выход через формулу с участием состояния, может быть сколько угодно в коде, и вызов одного из них определяет, какой гипероператор будет использован в качестве находящегося в блоке.

Также этот класс имеет методы для работы со значением, пришедшим на управляющий вход (b). Эти методы преобразуют значение с входа b и помещают полученный результат в поле с состоянием блока. То, какой метод будет вызван для работы с управляющим значением, то и будет определять, как оно преобразуется в потоке управления.

Сам же логический механизм представлен в классе Henkamono, у которого есть поля для хранения количества блоков, двух чисел, отвечающих за конфигурацию, а также вектора, в котором хранятся реализации класса Block.

У класса Henkamono есть методы расчёта номера следующего блока для передачи значений в управляющем потоке. У этого класса также есть методы передачи значений между блоками (передачи на те или иные входы). Также присутствует метод, отвечающий за прокрутку в цикле последовательной отработки всех блоков с промежуточной передачей значений, причём порядок передачи и отработки определяется заданной конфигурацией.