ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р МЭК 61131-7— 2017

КОНТРОЛЛЕРЫ ПРОГРАММИРУЕМЫЕ

Часть 7

Программирование нечеткого управления

(IEC 61131-7:2000, IDT)

Издание официальное

Москва
Стандартинформ
2017

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН Негосударственным образовательным частным учреждением дополнительного профессионального образования «Новая инженерная школа» (НОЧУ «НИШ») на основе собственного перевода на русский язык англоязычной версии указанного в пункте 4 стандарта, который выполнен Российской комиссией экспертов МЭК/ТК 65, и Федеральным государственным унитарным предприятием «Всероссийский научно-исследовательский институт стандартизации и сертификации в машиностроении» (ВНИИНМАШ)

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 306 «Измерения и управление в промышленных процессах»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 15 сентября 2017 г. № 1127-ст

4 Настоящий стандарт идентичен международному стандарту МЭК 61131-7:2000 «Контроллеры программируемые. Часть 7. Программирование нечеткого управления» (IEC 61131-7:2000 «Programmable controllers — Part 7: Fuzzy control programming», IDT).

Международный стандарт разработан Техническим комитетом МЭК ТК 65 «Измерение, управление и автоматизация в промышленных процессах».

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

*Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, 2017

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения 1
2 Нормативные ссылки 1
3 Термины и определения 1
4 Интеграция в программируемый контроллер 2
5 Язык нечеткого управления FCL 3
5.1 Обмен программами нечеткого управления 3
5.2 Элементы языка нечеткого управления 4
5.3 Пример FCL 12
5.4 Продукционные правила и ключевые слова языка нечеткого управления (FCL) 12
6 Соответствие 16
6.1 Классы соответствия языка нечеткого управления FCL 16
6.2 Контрольный перечень данных 18
Приложение А (справочное) Теория 20
Приложение В (справочное) Примеры 32
Приложение С (справочное) Промышленный пример: кран для контейнеров 34
Приложение D (справочное) Пример использования переменных в блоке правил 44
Приложение Е (справочное) Условные обозначения, аббревиатуры и синонимы 46
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов национальным стандартам 47

Введение

Теория нечеткой логики применительно к вопросам управления называется нечетким управлением. Нечеткое управление представляет собой технологию, которая позволяет расширить возможности промышленной автоматики и пригодна для решения задач уровня управления, выполняемых, как правило, программируемыми контроллерами.

Нечеткое управление, скорее, основано на практическом применении знаний, представленных в так называемых базах лингвистических правил, чем на использовании аналитических моделей (эмпирических или теоретических). Применение нечеткого управления возможно при наличии специальных знаний, которые могут быть представлены математически. При этом становится возможным использование имеющихся знаний для усовершенствования технологических процессов и выполнения различных задач. Например, для:

- управления (замкнутые или разомкнутые системы, одна или несколько переменных в линейных или нелинейных системах);
- настройки параметров систем в интерактивном или автономном режиме;
- классификации и распознавания образов;
- принятия решений в режиме реального времени (отправить изделие на станок А или Б?);
- помощи операторам в принятии решения или при регулировании параметров;
- обнаружения и диагностики неисправностей систем.

Широкий спектр вариантов применения и естественный подход, основанный на приобретенном человеком опыте, делает нечеткое управление основным инструментом, который должен стать стандартом для потребителей программируемых контроллеров.

Кроме того, нечеткое управление может комбинироваться с классическими методами управления.

Применение нечеткого управления может иметь преимущества в тех случаях, когда отсутствует точная модель технологического процесса, аналитическую модель сложно рассчитывать в режиме реального времени или трудно разработать в принципе.

Другим преимуществом нечеткого управления является возможность непосредственного использования опыта, приобретаемого потребителем. Кроме того, отсутствует необходимость моделировать весь контроллер с использованием нечеткого управления, так как в некоторых случаях нечеткое управление применяется для интерполяции между последовательностью локально линейных моделей, для динамической адаптации параметров линейного контроллера, превращающих его в нелинейный, или, наоборот, для концентрации на определенной, требующей оптимизации функции имеющегося контроллера.

Нечеткое управление является многозначным управлением, которое не ограничивается значениями «истина» и «ложь». Поэтому нечеткое управление полезно при создании модели эмпирических знаний, определяющей, какие управляющие воздействия должны выполняться при заданном наборе входных сигналов.

Современная теория и уже внедренные системы нечеткого управления существенно различаются в части терминологии (определения), функций (функциональные возможности) и практической реализации (инструментарий).

Нечеткое управление применяется как при решении небольших и простых задач, так и в сложных проектах. Чтобы охватить все варианты использования, в настоящем стандарте функции соответствующей системы нечеткого управления отображаются в определенных классах соответствия.

Базовый класс определяет минимальный набор функций, который должен обеспечиваться всеми совместимыми системами. За счет этого упрощается обмен программами нечеткого управления.

Дополнительные стандартные функции определяются в расширенном классе. Программы нечеткого управления, использующие такие функции, могут быть перенесены между системами с использованием такого же набора функций только полностью, в противном случае возможен лишь частичный обмен. Настоящий стандарт не требует, чтобы все совместимые системы выполняли все функции расширенного класса, однако допускает возможность (частичного) переноса и предотвращение использования нестандартных функций. Поэтому совместимая система не должна предлагать нестандартные функции, которые могут быть выполнены с использованием стандартных функций базового и расширенного классов.

Чтобы не исключать системы, применяющие свои сложные функции из числа соответствующих настоящему стандарту и не препятствовать процессу дальнейшего развития, настоящий стандарт разрешает применять дополнительные нестандартные функции, которые не охвачены базовым

и расширенным классами. Тем не менее такие функции должны быть перечислены стандартным способом для обеспечения простоты их распознавания как нестандартных функций.

Возможность переноса приложений нечеткого управления зависит от разных систем программирования, а также от характеристик систем управления. Эти зависимости приводятся в контрольном перечне данных, который должен быть предоставлен изготовителем.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

КОНТРОЛЛЕРЫ ПРОГРАММИРУЕМЫЕ

Часть 7

Программирование нечеткого управления

Programmable controllers. Part 7. Fuzzy control programming

Дата введения — 2018—09—01

1 Область применения

Настоящий стандарт устанавливает язык программирования приложений нечеткого управления, применяемый программируемыми контроллерами.

Целью настоящего стандарта является достижение изготовителями и потребителями общего понимания основных средств интегрирования приложений нечеткого управления в языки программируемых контроллеров в соответствии с МЭК 61131-3, а также возможности обмена переносимыми программами нечеткого управления между разными системами программирования.

Для достижения поставленной цели в приложении А приводится краткое описание теории нечеткого управления и нечеткой логики, необходимое для понимания настоящего стандарта. Пользователям настоящего стандарта, не знакомым с теорией нечеткого управления, может быть полезно сначала обратиться к приложению А.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты.

IEC 60050-351:1998, International Electrotechnical Vocabulary (IEV) — Part 351: Automatic control (Международный электротехнический словарь (IEV). Часть 351. Автоматическое управление)

IEC 61131-3:1993, Programmable controllers — Part 3: Programming languages (Программируемые контроллеры. Часть 3. Языки программирования)

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями. Определения для элементов языка дополнительно приведены в МЭК 61131-3.

Примечание — Термины, определения которых даны в настоящем разделе, выделяются курсивом в тексте определений.

3.1 **аккумуляция** (accumulation)

агрегирование результатов (result aggregation): Объединение результатов лингвистических правил в конечный результат.

3.2 **агрегирование** (aggregation)

определение степени выполнения (determination of degree of firing): Объединение степени принадлежности всех индивидуальных подусловий в правиле для расчета степени выполнения условия правила.

Издание официальное

3.3 **активизация** (activation): Процесс, которым степень выполнения условия воздействует на выходное нечеткое множество.

3.4 **результат** (conclusion)

следствие (consequent): Выходное значение *лингвистического правила*, то есть действие, которое должно выполняться (часть THEN правила нечеткого управления IF..THEN).

3.5 **условие** (condition)

предпосылка (antecedent): Выражение, включая подусловия, объединенные нечеткими операторами И, ИЛИ, НЕ.

3.6 **четкое множество** (crisp set): Особый случай нечеткого множества, в котором функция принадлежности принимает только два значения, обычно определяемые как 0 и 1.

3.7 **дефаззификация** (defuzzification): Преобразование нечеткого множества в числовое значение.

3.8 **степень принадлежности** (degree of membership): Значение функции принадлежности.

3.9 **фаззификация** (fuzzification): Определение степени принадлежности четкого входного значения лингвистических термов, определенных для каждой входной лингвистической переменной.

3.10 **нечеткое управление** (fuzzy control): Тип управления, алгоритм управления в котором основан на нечеткой логике.

[IEV 351-17-51, измененный]

3.11 **нечеткая логика** (fuzzy logic): Совокупность математических теорий, основанных на принципах нечеткого множества. Нечеткая логика является одним из типов бесконечнозначной логики.

3.12 **нечеткий оператор** (fuzzy operator): Оператор, используемый в теории нечеткой логики.

3.13 **нечеткое множество** (fuzzy set): Нечеткое множество A определяется как множество упорядоченных пар (x, μA(x)), где x — это элемент предметной области U, а μA(x) — функция принадлежности, которая сопоставляет каждому $x \in U$ действительное число $\in [0,1]$, определяющее степень того, насколько x принадлежит множеству.

3.14 **логический вывод** (inference): Применение лингвистических правил к входным значениям с целью определения выходных значений.

3.15 **лингвистическое правило** (linguistic rule): Правило IF-THEN (если ..., то ...) с условием и результатом, причем одно или оба являются лингвистическими.

3.16 **лингвистический терм** (linguistic term): В контексте лингвистического управления лингвистические термы определяются нечеткими множествами.

3.17 **лингвистическая переменная** (linguistic variable): Переменная, которая принимает значения из диапазона лингвистических термов.

3.18 **функция принадлежности** (membership function): Функция, которая в предметной области определяет степень принадлежности данному нечеткому множеству.

[IEV 351-17-52, измененный]

3.19 **синглтон** (singleton): Нечеткое множество, функция принадлежности которого равна единице в одной точке и нулю во всех других точках.

3.20 **подусловие** (subcondition): Элементарное выражение в форме переменной или терма «лингвистическая переменная IS лингвистический терм».

3.21 **база правил** (rule base): Набор лингвистических правил для достижения определенных целей.

3.22 **весовой коэффициент** (weighting factor): Значение в интервале 0... 1, которое устанавливает степень важности, достоверности, уверенности для лингвистического правила.

4 Интеграция в программируемый контроллер

В соответствии с разделом 5, приложения нечеткого управления, программируемые на языке нечеткого управления FCL, должны быть заключены в функциональные блоки (или программы), как это определено в МЭК 61131-3. В настоящем стандарте применяются понятия типов и экземпляров функциональных блоков, приведенные в МЭК 61131-3.

Типы функциональных блоков, определенные в языке нечеткого управления FCL, должны устанавливать входные и выходные параметры, а также специфические правила и описания нечеткого управления.

Соответствующие экземпляры функциональных блоков должны содержать специфические данные приложений нечеткого управления.

Функциональный блок, определенный в языке нечеткого управления FCL, может применяться в программах и функциональных блоках, написанных на любом из языков, приведенных в МЭК 61131-3,

например, на языке релейной логики, на языке программирования и т. д. Типы данных входных и выходных параметров функционального блока или программы, написанных на FCL, должны совпадать с соответствующей «вызывающей средой», как показано на рисунке 1.

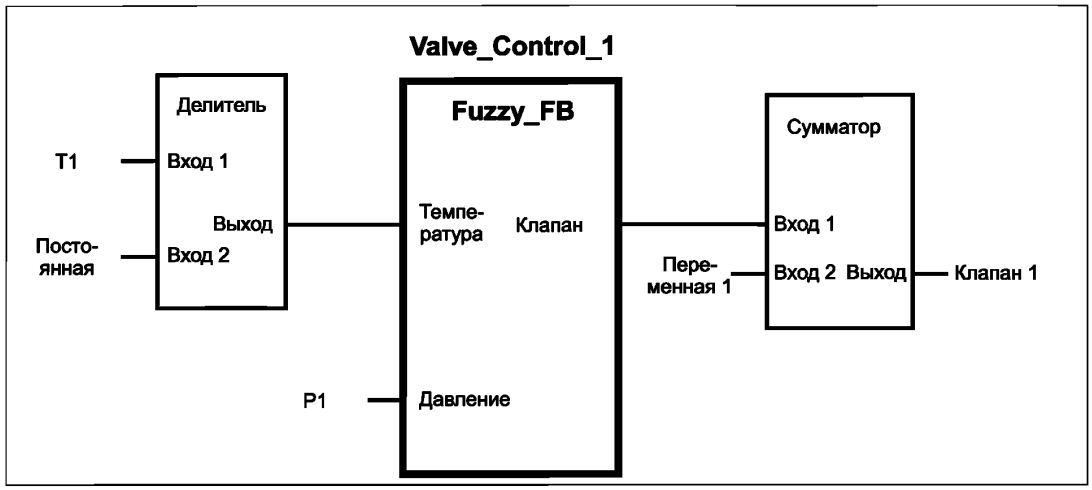


Рисунок 1 — Пример функционального блока нечеткого управления в представлении FBD

В данном примере Valve_Control_1 — это экземпляр определенного потребителем функционального блока, имеющий тип Fuzzy_FB. В соответствии с разделом 5, тип функционального блока Fuzzy_FB может программироваться на языке нечеткого управления FCL. В данном случае функциональный блок Fuzzy_FB используется в программе или в функциональном блоке, который представлен на графическом языке FBD (Function Block Diagram, диаграммы функциональных блоков) в МЭК 61131-3.

5 Язык нечеткого управления FCL

5.1 Обмен программами нечеткого управления

Определение языка нечеткого управления FCL основывается на определении языков программирования, определенных в МЭК 61131-3. Взаимодействие алгоритма нечеткого управления с его программной средой «прячет» его от программы. Поэтому в соответствии с МЭК 61131-3 внешне алгоритм нечеткого управления представлен в виде функционального блока. В соответствии с настоящим разделом должны быть определены элементы, необходимые для описания внутренних логических частей функционального блока нечеткого управления, например функции принадлежности, правила, операторы и методы.

Элементы языка FCL стандартизируют общее представление обмена данными между средствами настройки нечеткого управления разных изготовителей, как показано на рисунке 2. Используя данное общее представление, каждый изготовитель программируемых контроллеров может поддерживать аппаратное обеспечение, редакторы программного обеспечения и компиляторы. Изготовитель должен лишь применить интерфейс данных к его специфическому редактору. Заказчик должен иметь возможность обмениваться проектами нечеткого управления с разными изготовителями.

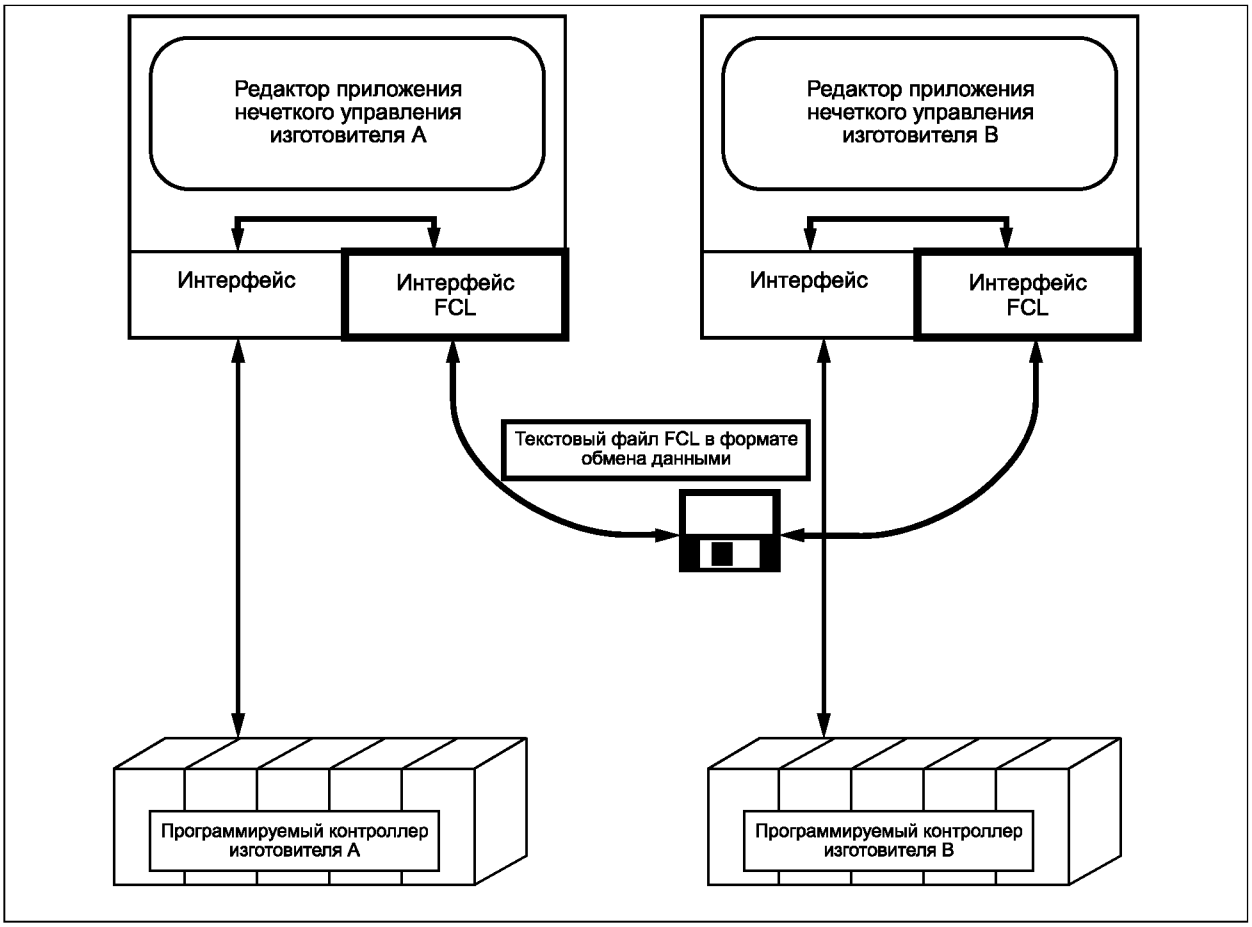


Рисунок 2 — Обмен данными программ в языке нечеткого управления (FCL)

5.2 Элементы языка нечеткого управления

В настоящем подразделе приведены примеры описания элементов языка нечеткого управления. Подробное продукционное правило приведено в подразделе 5.4.

5.2.1 Интерфейс функционального блока

В соответствии с разделом 4, внешнее представление нечеткого функционального блока требует, чтобы использовались следующие стандартные элементы языка, установленные в МЭК 61131-3:

FUNCTION_BLOCK *function_block_name*	Функциональный блок
VAR_INPUT *variable_name*: *data_type*; .... END_VAR	Описание входного параметра
VAR_OUTPUT variable_name: *data_type*; .... END_VAR	Описание выходного параметра
.... VAR *variable_name: data_type*; END_VAR END_FUNCTION_BLOCK	Локальные переменные

С помощью данных элементов языка можно описать интерфейс функционального блока. Интерфейс функционального блока определяется с помощью параметров, которые переходят в функциональный блок и выходят из него. Типы данных параметров должны определяться в соответствии с МЭК 61131-3.

На рисунке 3 представлен пример описания функционального блока на языках структурированного текста (ST) и диаграмм функциональных блоков (FBD).

<table>
<tr>
<td>

```
FUNCTION_BLOCK Fuzzy-FB
VAR_INPUT
      Температура: REAL;
      Давление: REAL;
  END_VAR

  VAR_OUTPUT
     Клапан: REAL;
   END_VAR

END_FUNCTION_BLOCK
```

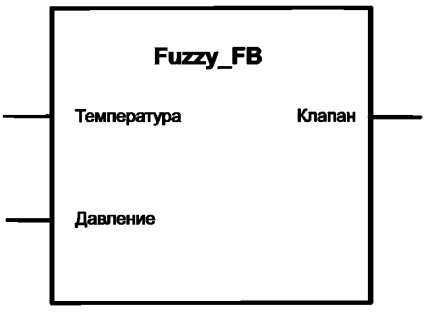
</td>
<td>
</td>
</tr>
<tr>
<td>Структурированный текст (ST)</td>
<td>Диаграммы функциональных блоков (FBD)</td>
</tr>
</table>

Рисунок 3 — Пример описания интерфейса функционального блока на языках ST и FBD

5.2.2 Фаззификация

Значения входных переменных должны быть преобразованы в *степень принадлежности* по *функции принадлежности*, определенной для переменной. Преобразование описывается между ключевыми словами FUZZIFY и END_FUZZIFY.

```
FUZZIFY variable_name
    TERM term_name:= membership_function;
....
END_FUZZIFY
```

После ключевого слова FUZZIFY должно быть указано имя переменной, используемой для фаззификации, т. е. имя переменной, предварительно определенной в разделе VAR_INPUT. *Лингвистическая переменная* должна быть описана одним или несколькими *лингвистическими термами*. *Лингвистические термы*, введенные ключевым словом TERM, с целью фаззификации переменной описываются *функцией принадлежности*. *Функция принадлежности* представляет собой кусочно-линейную функцию, которая определяется таблицей точек.

membership_function::= (*point i*), (*point* j), ...

Каждая точка представляет собой пару значений переменной и степень принадлежности данных значений, разделенные запятой. Пары заключены в скобки и разделены запятыми.

точка i::= значение входа i | *variable_name* входа i, *значение* i степени принадлежности

С помощью данного выражения могут быть определены все простые элементы, например быстрое изменение по линейному закону и по треугольнику. Точки должны быть заданы в порядке возрастания значения переменной. Функция принадлежности между очередными точками является линейной. Таким образом, степень принадлежности для каждого терма рассчитывается по четкому входному значению с помощью линейной интерполяции между двумя соответствующими соседними точками функции принадлежности.

Минимальное количество точек равно двум. Максимальное количество ограничивается согласно классам соответствия, см. раздел 6.

Пример *функции принадлежности* с тремя точками для *лингвистического терма* «тепло»:

```
TERM тепло: = (17,5; 0,0) (20,0; 1,0) (22,5; 0,0);
```

Если значение лингвистической переменной меньше, чем первая базовая точка в таблице соответствия, все значения ниже первой точки в таблице соответствия должны иметь такую же степень принадлежности, как и определенная для первой точки.

Если значение лингвистической переменной больше, чем последняя базовая точка в таблице соответствия, все значения больше последней точки в таблице соответствия должны иметь такую же степень принадлежности, как и определенная для последней точки.

```
Определение быстрого изменения по
линейному закону:
    FUZZIFY температура:
        TERM холодно: = (3, 1) (27,0);
        TERM тепло: = (3, 0) (27, 1);
    END_FUZZIFY
```

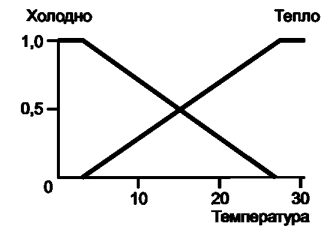


Рисунок 4 — Пример термов быстрого изменения по линейному закону

Примечание — Тип данных для точек функций принадлежности не определяется. Изготовитель должен предоставить компилятор, который поддерживает любые необходимые преобразования.

Для адаптации приложения нечеткого управления в интерактивном режиме базовые точки в функции принадлежности могут быть изменены. Это можно сделать с помощью переменных, которые вводятся в функциональный блок. Такие переменные должны быть описаны в разделе функционального блока VAR_INPUT. Пример использования переменных для определения точек для функций принадлежности приведен на рисунке 5.

Примечание — Значения точек функции принадлежности во время выполнения могут лежать вне последовательности.

```
VAR_INPUT
        Температура: REAL;          (* данный ввод должен быть фаззифицирован*)
        Давление: REAL;             (* данный ввод должен быть фаззифицирован*)
        bp_warm1, bp_warm2: REAL; (* данные вводы для адаптации в интерактивном режиме*)
END_VAR
FUZZIFY температура
        TERM тепло:= (bp_warm1, 0,0), (21,0; 1,0), (bp_warm2, 0,0);
..
END_FUZZIFY
```

Рисунок 5 — Пример использования переменных функции принадлежности

5.2.3 Дефаззификация

Лингвистическая переменная для выходной переменной должна быть преобразована в значение. Такое преобразование описывается между ключевыми словами DEFUZZIFY и END_DEFUZZIFY.

После ключевого слова DEFUZZIFY должно быть указано имя переменной, используемой для *дефаззификации*. Это имя предварительно определенной в разделе VAR_OUTPUT переменной.

```
DEFUZZIFY variable_name
    RANGE(min..max);
    TERM term_name:= membership_function;
    defuzzification_method;
    default_value;
END_DEFUZZIFY
```

Определение *лингвистических термов* дается в 5.2.2.

Синглтоны представляют собой специальные *функции принадлежности*, используемые для выходов с целью упрощения *дефаззификации*. Синглтоны описываются только одним значением *лингвистического терма*. На рисунке 6 приведены примеры термов синглтонов.

Определение синглтонов:

```
DEFUZZIFY клапан
RANGE(–100,100);
        TERM дренаж := –100;
        TERM закрыт : = 0;
        TERM вход : = 100;
END_DEFUZZIFY
```

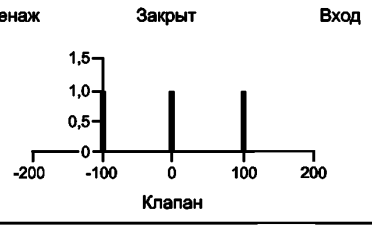


Рисунок 6 — Примеры термов синглтонов

Метод дефаззификации должен быть определен с помощью элемента языка METHOD.

```
METHOD: defuzzification_method;
```

Допускаются следующие методы дефаззификации (см. таблицы 1 и 2).

Таблица 1 — Методы дефаззификации

Ключевое слово	Расшифровка
CoG	Центр тяжести (примечание 1)
CoGS	Центр тяжести для синглтонов
CoA	Центр площади (примечания 2 и 3)
LM	Крайний левый максимум (примечание 4)
RM	Крайний правый максимум (примечание 4)

Примечание 1 — Центр тяжести эквивалентен центроиду.
Примечание 2 — Центр площади эквивалентен медиане.
Примечание 3 — При использовании синглтонов CoA неприменим.
Примечание 4 — Методы дефаззификации LM и RM асимметричны относительно нуля.

Т а б л и ц а 2 — Формулы методов дефаззификации

COG	$U = \frac{\int_{Min}^{Max} u\ \mu(u)\,du}{\int_{Min}^{Max} \mu(u)\,du}$
COGS	$U = \frac{\sum_{i=1}^{p} [u_i \mu_i]}{\sum_{i=1}^{p} [\mu_i]}$
COA	$U = u',\ \int_{Min}^{u'} \mu(u)du = \int_{u'}^{Max} \mu(u)du$
RM	$U = \sup(u'), \mu(u') = \sup_{u \in [Min, Max]} \mu(u)$
LM	$U = \inf(u'), \mu(u') = \inf_{u \in [Min, Max]} \mu(u)$

где:
U — результат дефаззификации;
u — выходная переменная;
p — количество синглтонов;
μ — функция принадлежности накопленных нечетких множеств;
i — индекс;
Min — минимальное значение дефаззификации, определенное в RANGE. В случае синглтонов Min = -∞;
Max — максимальное значение дефаззификации, определенное в RANGE. В случае синглтонов Min = +∞;
sup — наибольшее значение;
inf — наименьшее значение.

Если степень принадлежности равна 0 для всех *лингвистических термов* выходной переменной, то это значит, что активные правила для данной выходной переменной отсутствуют. В таком случае *дефаззификация* не может сгенерировать подходящее выходное значение. Поэтому для выходного значения допускается назначать значение по умолчанию. Такое значение по умолчанию является значением выходной переменной только в том случае, если не работает ни одно правило.

DEFAULT:= *значение* | NC;

Значение должно быть указано после ключевого слова DEFAULT. В противном случае должно быть задано NC (без изменений) для указания на то, что выходное значение останется неизменным, если не работает ни одно правило.

Диапазон — это описание минимального и максимального значений, которые разделяются двоеточием.

RANGE:= (*минимальное значение...* максимальное значение);

RANGE используется для ограничения каждой функции принадлежности до диапазона каждой выходной переменной. Если для выходных функций принадлежности используется синглтон, RANGE не действует.

Если диапазон не определен, по умолчанию должен использоваться диапазон типа данных переменной, указанный в МЭК 61131-3.

5.2.4 Блок правил

Логический вывод нечеткого алгоритма должен быть определен одним или несколькими блоками правил. Для надлежащей обработки и обеспечения возможности разделения базы правил на разные модули допускается использование нескольких блоков правил. Каждый блок правил должен иметь уникальное имя.

Правила должны быть указаны между ключевыми словами RULEBLOCK и END_RULEBLOCK.

RULEBLOCK *имя_блока_правил*
определение_операции;
[*метод_активизации;*]
метод_аккумуляции;
правила;
END_RULEBLOCK

Внутри блока правил используются нечеткие операции.

определение_операции ::= *операция: алгоритм*

Для соблюдения закона Моргана алгоритмы операций AND и OR должны применяться попарно; например, MAX необходимо применять с операцией OR, если с операцией AND применяется MIN.

Таблица 3 — Парные алгоритмы

Операция OR		Операция AND	
Ключевое слово для алгоритма	Алгоритм	Ключевое слово для алгоритма	Алгоритм
MAX	$Max(\mu_1(x), \mu_2(x))$	MIN	$Min(\mu_1(x), \mu_2(x))$
ASUM	$\mu_1(x)+ \mu_2(x) - \mu_1(x)\,\mu_2(x)$	PROD	$\mu_1(x)\,\mu_2(x)$
BSUM	$Min(1, \mu_1(x) + \mu_2(x))$	BDIF	$Max(0, \mu_1(x) + \mu_2(x) - 1)$

Пример блоков правил:
первый RULEBLOCK
AND: MIN;
..
END_RULEBLOCK
второй RULEBLOCK
AND: PROD;
..
END_RULEBLOCK

Метод активизации определяется следующим элементом языка:

ACT: *метод_активизации*;

Могут использоваться следующие методы активизации (см. таблицу 4):

Таблица 4 — Методы активизации

Название	Ключевое слово	Алгоритм
Продукт	PROD	$\mu_1(x)\,\mu_2(x)$
Минимальная	MIN	$Min(\mu_1(x), \mu_2(x))$
Примечание — Метод активизации не имеет отношения к синглтонам.		

Метод аккумуляции определяется следующим элементом языка:

ACCU: *метод_аккумуляции*;

Могут применяться следующие методы аккумуляции (см. таблицу 5):

Таблица 5 — Методы аккумуляции

Название	Ключевое слово	Формула
Максимум	MAX	$MAX(\mu_1(x), \mu_2(x))$
Ограниченная сумма	BSUM	$MIN(1, \mu_1(x) + \mu_2(x))$
Нормализованная сумма	NSUM	$\frac{\mu_1(x) + \mu_2(x)}{MAX(1, MAX_{x' \in X} (\mu_1(x') + \mu_2(x')))}$

Входными данными блока правил являются *лингвистические переменные* с набором *лингвистических термов*. Каждый терм имеет присвоенную ему степень принадлежности.

Правила определяются внутри блока правил. Каждое правило должно начинаться с ключевого слова RULE с идущим за ним названием правила, а заканчиваться должно точкой с запятой. Каждое правило внутри блока имеет уникальный номер.

Номера RULE: IF условие THEN результат [WITH весовой коэффициент];

Само правило должно начинаться с ключевого слова IF с идущим за ним *условием*. После *условия* следует *результат*, начинающийся с ключевого слова THEN.

Допускается комбинировать в одном правиле несколько *подусловий* и входных переменных. Назначение переменных состоит в разрешении импорта нечеткой степени принадлежности в нечеткий функциональный блок. Переменные должны быть определены между ключевыми словами IF и THEN и объединены операциями с ключевыми словами AND, OR или NOT.

Приоритет операций (см. таблицу 6) определяется в соответствии с булевой алгеброй, приведенной в таблице 3.

Таблица 6 — Приоритет операций

Приоритет	Операция
1	( ) скобки
2	NOT
3	AND
4	OR

Упрощенный пример правила:

RULE 1: IF *подусловие 1* AND *переменная 1* OR *переменная 2* THEN *результат*;

На базовом уровне соответствия операция OR может быть реализована определением двух правил:

RULE 3: IF *подусловие 1* OR *подусловие 2* THEN *результат*; заменено на: RULE 3a: IF *условие 1* THEN результат; RULE 3b: IF *условие 2* THEN результат.

Подусловие начинается с названия лингвистической переменной, за которым идет ключевое слово IS с необязательным NOT и одним *лингвистическим термом лингвистической переменной*, используемой в *условии*.

Подусловие := *лингвистическая_переменная* IS [NOT] *лингвистический_терм*

Лингвистические термы, применяемые в условии, должны соответствовать *лингвистической переменной* в том же *условии*. Применяемый терм должен быть предварительно определен с помощью ключевого слова TERM.

Пример подусловий:

```
Темп. IS жарко
Темп. IS NOT жарко
```

Кроме того, перед подусловием можно применять ключевое слово NOT. При этом могут применяться скобки.

```
IF NOT темп. IS жарко THEN ...        или    IF NOT (темп. IS жарко) THEN ...
```

Результат может быть разделен на несколько подрезультатов и выходных переменных.

Подрезультат должен начинаться с названия *лингвистической переменной*, за которым идет ключевое слово IS и один *лингвистический терм* данной *лингвистической переменной.*

Подрезультат := *лингвистическая_переменная* IS *лингвистический_терм*

Пример с несколькими подрезультатами из одной или нескольких строк:

Допускается присваивать каждому подрезультату *весовой коэффициент,* который является числом со значением, лежащим между 0,0 и 1,0, или переменной. Это делается с помощью ключевого слова WITH, за которым следует *весовой коэффициент*.

Весовой коэффициент должен уменьшать степень принадлежности (функция принадлежности) подрезультата путем умножения итога подрезультата на *весовой коэффициент*.

Для внешнего управления параметрами приложения нечеткого управления *весовой коэффициент* может быть переменным. В таком случае переменная должна быть описана в разделе VAR_INPUT. Это дает возможность во время работы менять *весовой коэффициент для* адаптации программы нечеткого управления к требованиям технологического процесса.

Если подрезультату не присвоено ключевое слово WITH, должен быть по умолчанию принят *весовой коэффициент* 1,0.

IF *условие* THEN *подрезультат* [WITH *весовой_коэффициент*] *подрезультат*;

Пример постоянного *весового коэффициента:*

```
F темп. IS холодно AND давление IS низкое THEN клапан 1 IS вход WITH 0,5,
                                                клапан 2 IS закрыт;
```

Пример переменного *весового коэффициента*:

```
VAR_INPUT
   w_мое_правило1: REAL:= 0,8;
END_VAR
RULEBLOCK правило_темп
   RULE 1:    IF темп.        IS холодная AND давление IS низкое
              THEN клапан     IS вход WITH w_мое_правило1;
   ..
END_RULEBLOCK
```

5.2.5 Необязательные параметры

Для реализации различных целевых систем может потребоваться предоставить системе дополнительную информацию, позволяющую осуществить наилучшее преобразование приложений нечеткого управления.

Такая дополнительная информация может быть необходима элементу языка, находящемуся между OPTIONS и END_OPTIONS.

```
OPTIONS
    спец_параметры_приложения
END_OPTIONS
```

Согласно разделу 6, данные элементы языка должны применяться для функций в классе соответствия открытого уровня.

5.3 Пример FCL

На рисунке 7 приведен пример языка нечеткого управления.

```
FUNCTION_BLOCK Нечеткий_ФБ
VAR_INPUT
            температура: REAL;
            давление:    REAL;
END_VAR
VAR_OUTPUT
            клапан:      REAL;
END_VAR
FUZZIFY температура
            TERM холодно  := (3, 1) (27, 0);
            TERM жарко    := (3, 0) (27, 1);
END_FUZZIFY
FUZZIFY давление
            TERM низкое   := (55, 1) (95, 0);
            TERM высокое  := (55, 0) (95, 1);
END_FUZZIFY
DEFUZZIFY клапан
            TERM дренаж   := -100;
            TERM закрыт   := 0;
            TERM вход     := 100;
            METHOD: CoGS;
            DEFAULT       := 0;
END_DEFUZZIFY
RULEBLOCK №1
            AND: MIN;
            ACCU: MAX;
            RULE 1: IF температура IS холодно AND давление IS низкое   THEN клапан IS вход;
            RULE 2: IF температура IS холодно AND давление IS высокое  THEN клапан IS закрыт WITH 0,8;
            RULE 3: IF температура IS жарко AND давление IS низкое     THEN клапан IS закрыт;
            RULE 4: IF температура IS жарко AND давление IS высокое    THEN клапан IS дренаж;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

Рисунок 7 — Пример нечеткого функционального блока

5.4 Продукционные правила и ключевые слова языка нечеткого управления (FCL)

Метод спецификации текстовых языков для программируемых контроллеров определяется в соответствии с МЭК 61131-3, приложение А. Данный метод спецификации применяется в настоящем стандарте для FCL.

Формальные характеристики элементов текстовых языков программирования, установленных в МЭК 61131-3, определяются в соответствии с МЭК 61131-3, приложение В. Для FCL применяется следующее подмножество элементов языка в соответствии с МЭК 61131-3, приложение В:

В. 1.1 Буквы, цифры и идентификаторы;
В.1.2 Константы;
В.1.3 Типы данных;
В.1.4 Переменные.

5.4.1 Продукционные правила

Дополнительно к перечисленным выше элементам языка, установленным в МЭК 61131-3, могут применяться следующие:

описание_функционального_блока ::= 'FUNCTION_BLOCK' имя_функционального_блока
{описания_пер_ввода_вывода_фб}
{другие_описания_пер}
тело_функционального_блока
'END_FUNCTION_BLOCK'

описания_пер_ввода_вывода_фб ::= входные_описания | выходные_описания

другие_описания_пер ::= описания_пер

тело_функционального_блока ::= {блок_фаззификации}
{блок_дефаззификации}
{блок_правил}
{доп_блок}

блок_фаззификации ::= 'FUZZIFY' имя_переменной
{лингвистический_терм}
'END_FUZZIFY

блок_дефаззификации ::= DEFUZZIFY' f_имя_переменной
[диапазон]
{лингвистический_терм}
метод_дефаззификации
значение_по_умолчанию
'END_DEFUZZIFY'

блок_правил ::= 'RULEBLOCK' имя_блока_правил
определение_операции
[метод_активизации]
метод_аккумуляции
{правило}
'END_RULEBLOCK'

доп_блок ::= 'OPTION'
любой указанный изготовителем параметр
'END_OPTION'

лингвистический_терм ::= 'TERM' имя_терма ':=' функция_принадлежности ';'

функция_принадлежности ::= синглтон | точки

Примечание — Информацию об использовании синглтонов см. в 5.2.3.

синглтон ::= числовая_константа | имя_переменной

точки ::= {'('числовая_константа | имя_переменной ','
числовая_константа ')'}

Примечание — Допускаемые номера точек см. в 5.2.2.

метод_дефаззификации ::= 'METHOD' ':' 'CoG' | 'CoGS' | 'CoA' | 'LM' | 'RM' ';'

значение_по_умолчанию ::= 'DEFAULT' ':=' числовая_константа | 'NC' ';'

диапазон ::= 'RANGE('числовая_константа'..' числовая_константа')' ';'

определение_операции ::= [('OR' ':' 'MAX' | 'ASUM' | 'BSUM')]
[('AND' ':' 'MIN' | 'PROD' | 'BDIF')] ';'

Примечание — Парные алгоритмы см. в таблице 3.

метод_активизации ::= 'ACT' ':' 'PROD' | 'MIN' ';'

метод_аккумуляции ::= 'ACCU' ':' 'MAX' | 'BSUM' | 'NSUM' ';'

правило ::= 'RULE' целочисленный_литерал ':'
IF условие THEN результат [WITH весовой_коэффициент];

условие ::= x{('AND' x)|('OR' x)}

x ::= ['NOT'] (подусловие | (' (' условие ')'))

подусловие ::= имя_переменной | (имя_переменной 'IS' [`NOT'] имя_терма)

результат ::= { (имя_переменной | (имя_переменной 'IS' имя_терма)) ','}
(имя_переменной | имя_переменной 'IS' имя_терма)

весовой_коэффициент ::= переменная | числовая_константа

имя_функционального_блока ::= идентификатор

имя_блока_правил ::= идентификатор

имя_терма ::= идентификатор

f_имя_переменной ::= идентификатор

имя_переменной ::= идентификатор

числовая_константа ::= целочисленный_литерал | действительный_литерал

входные_описания ::= см. МЭК 61131-3, приложение B

выходные_описания ::= см. МЭК 61131-3, приложение B

описания_пер ::= см. МЭК 61131-3, приложение B

идентификатор ::= см. МЭК 61131-3, приложение B

5.4.2 Ключевые слова

Таблица 7 — Зарезервированные ключевые слова для FCL

Ключевое слово	Значение	Пункт
( )	Скобки в условии, терме, диапазоне	5.2.4
ACCU	Метод аккумуляции	5.2.4
ACT	Метод активизации	5.2.4
AND	Операция AND	5.2.4

Окончание таблицы 7

Ключевое слово	Значение	Пункт
ASUM	Операция OR, алгебраическая сумма	5.2.4
BDIF	Операция AND, ограниченная разность	5.2.4
BSUM	Метод аккумуляции, операция OR, ограниченная сумма	5.2.4
CoA	Дефаззификация методом центра площади	5.2.3
CoG	Дефаззификация методом центра тяжести	5.2.3
CoGS	Дефаззификация синглтонов методом центра тяжести	5.2.3
DEFAULT	Выходное значение по умолчанию в случае, когда не работает ни одно правило	5.2.3
DEFUZZIFY	Дефаззификация выходной переменной	5.2.3
END_DEFUZZIFY	Конец спецификации дефаззификации	5.2.3
END_FUNCTION_BLOCK	Конец спецификации функционального блока	5.2.1
END_FUZZIFY	Конец спецификаций фаззификации	5.2.2
END_OPTIONS	Конец спецификаций дополнений	5.2.5
END_RULEBLOCK	Конец спецификаций блока правил	5.2.4
END_VAR	Конец определений входной/выходной переменной	5.2.1
FUNCTION_BLOCK	Конец спецификаций функционального блока	5.2.1
FUZZIFY	Фаззификация входной переменной	5.2.2
IF	Начало правила, за которым приводится условие	5.2.4
IS	Ставится после лингвистической переменной в условии и результате	5.2.4
LM	Метод дефаззификации «Крайний правый максимум»	5.2.3
MAX	Метод аккумуляции по формуле максимума, операция OR	5.2.4
METHOD	Метод дефаззификации	5.2.3
MIN	Минимум в качестве логической операции AND	5.2.4
NC	Выходная переменная остается без изменений (No Change) в случае, когда не работает ни одно правило	5.2.3
NOT	Операция NOT	5.2.4
NSUM	Метод аккумуляции по формуле нормализованной суммы	5.2.4
OPTIONS	Определение дополнительных параметров	5.2.5
OR	Операция OR	5.2.4
PROD	Произведение в качестве операции AND, метод активизации	5.2.4
RANGE	Пределы функции принадлежности	5.2.3
RM	Метод дефаззификации «Крайний правый максимум»	5.2.3
RULE	Начало спецификации нечеткого правила	5.2.4
RULEBLOCK	Начало спецификации блока правил	5.2.4
TERM	Определение лингвистического терма (функция принадлежности) для лингвистической переменной	5.2.2
THEN	Отделяет условие от результата	5.2.4
VAR	Определение локальной переменной (s)	5.2.1
VAR_INPUT	Определение входной переменной (s)	5.2.1
VAR_OUTPUT	Определение выходной переменной (s)	5.2.1
WITH	Определение весового коэффициента	5.2.4

6 Соответствие

6.1 Классы соответствия языка нечеткого управления FCL

Уровни соответствия системы управления, использующей язык нечеткого управления FCL, приведены на рисунке 8. Иерархическая структура состоит из следующих уровней:

- базовый уровень, включающий в себя определения функционального блока и типов данных в соответствии с МЭК 61131-3;
- расширенный уровень с необязательными функциями, приведенными в таблице 9, дополнительно разрешенными на данном уровне;
- открытый уровень, охватывающий дополнительные функции, не определенные в настоящем стандарте, но перечисленные изготовителем.

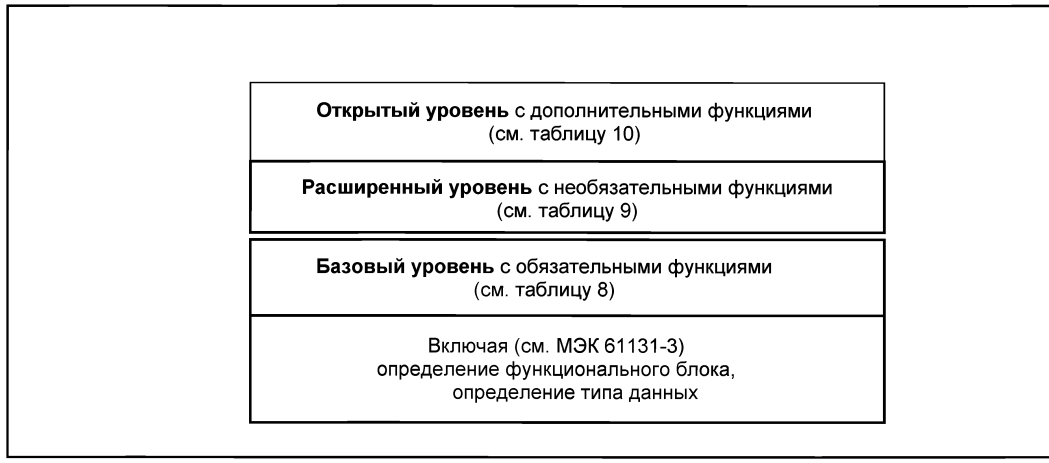


Рисунок 8 — Уровни соответствия

Система управления, применяющая язык нечеткого управления FCL и требующая соответствия настоящему стандарту, должна удовлетворять приведенным ниже правилам:

a) для реализации нечеткого управления система управления должна применять средства функционального блока в соответствии с МЭК 61131-3. Поэтому определение функциональных блоков и типов данных, необходимых для входных и выходных параметров функционального блока нечеткого управления, должно соответствовать МЭК 61131-3;

b) все функции нечеткого управления, приведенные в таблице 8, должны быть реализованы в соответствии с положениями настоящего стандарта. Данная таблица определяет множество элементов базового уровня, которые в общем случае должны иметь все системы управления, соответствующие МЭК 61131;

c) подмножество элементов расширенного уровня, приведенное в таблице 9, представляет собой вспомогательные элементы, которые могут быть реализованы в дополнение ко всем функциям базового уровня. Реализация должна строго соответствовать положениям настоящего стандарта. Данные функции должны маркироваться как стандартные расширения; кроме того, перечень реализованных функций, как показано в таблице 9, должен быть частью документации системы;

d) сверх базового и расширенного уровней могут быть реализованы другие функции, если во избежание любых возможных ошибок эти функции не имеют таких же или аналогичных стандартным функциям возможностей или представлений. Данные функции должны маркироваться как функции открытого уровня; кроме того, перечень, как показано в таблице 10, должен быть частью документации системы;

e) обмен прикладными программами между разными системами нечеткого управления должен осуществляться в текстовой форме языка нечеткого управления FCL в соответствии с положениями настоящего стандарта. Данный формат, как формат ввода и вывода, должен быть доступным в системах, соответствующих настоящему стандарту;

f) чтобы получить наиболее подходящий и удобный пользовательский интерфейс и не препятствовать дальнейшему развитию, внешнее представление разработки, ввода, тестирования и т. д.

программ приложения нечеткого управления может быть реализовано любыми графическими или текстовыми средствами.

Элементы таблицы 8 являются базовым множеством функций, которые должны быть реализованы во всех системах нечеткого управления, соответствующих настоящему стандарту.

Таблица 8 — Элементы языка базового уровня FCL (обязательные)

Элемент языка	Ключевое слово	Описание
Описание функционального блока	VAR_INPUT, VAR_OUTPUT	Содержит входные и выходные переменные
Функция принадлежности	Входная переменная: TERM	Максимальная из трех постоянных точек (координата степени принадлежности равна 0 или 1)
	Выходная переменная: TERM	Только постоянные синглтоны
Условное агрегирование	операция: AND	Алгоритм: MIN
Активизация	—	Не применяется, так как используются только синглтоны
Аккумуляция (агрегирование результатов)	операция: ACCU	Алгоритм: MAX
Дефаззификация	METHOD	Алгоритм: CoGS
Значение по умолчанию	DEFAULT	NC, значение
Блок правил	RULEBLOCK	Только один блок правил
Условие	IF ... IS ...	n подусловий
Результат	THEN	Только один подрезультат

Элементы таблицы 9 представляют собой расширенное множество функций, которые могут быть дополнительно реализованы в стандартной системе нечеткого управления (например, для операции AND должен быть выбран алгоритм PROD или BDIF либо оба). Данные необязательные функции являются вспомогательными для всех функций базового уровня.

Таблица 9 — Элементы языка базового уровня FCL (необязательные)

Элемент языка	Ключевое слово	Описание
Описание функционального блока	VAR	Содержит локальные переменные
Функция принадлежности	Входная переменная: TERM	Максимальная из четырех постоянных или переменных точек (координата степени принадлежности равна 0 или 1)
	Выходная переменная: TERM	Максимальная из четырех постоянных или переменных точек (координата степени принадлежности равна 0 или 1)
Условное агрегирование	Операция: AND	Алгоритм: PROD, BDIF
	Операция: OR	Алгоритм: ASUM, BSUM
	Операция: NOT	1 – {аргумент}
	Скобки	( )
Активизация	Операция: ACT	Алгоритм: MIN, PROD
Аккумуляция	Операция: ACCU	Алгоритм: BSUM, NSUM
Диапазон фаззификации	Операция: RANGE	RANGE (минимальное значение... максимальное значение) ограничивает диапазон функции принадлежности для выходной переменной
Метод дефаззификации	Операция: METHOD	Алгоритм: CoG, CoA, LM, RM

Окончание таблицы 9

Элемент языка	Ключевое слово	Описание
Блок правил	Операция: RULEBLOCK	n блоков правил
Условие	IF	n подусловий, n входных переменных
Результат	THEN	n подусловий, n выходных переменных
Весовой коэффициент	WITH	Постоянное значение и значение, присвоенное переменной в разделе описания VAR_INPUT.....END_VAR

В таблице 10 приведен пример перечня элементов языка открытого уровня. Данный перечень должен быть частью документации системы.

Т а б л и ц а 10 — Примеры перечня элементов языка открытого уровня

Функции принадлежности свободного ввода/вывода (например, Гаусса, экспоненциальные)
Более четырех точек функции принадлежности
Координата степени принадлежности от 0 до 1
Переменные значения принадлежности

6.2 Контрольный перечень данных

Данный контрольный перечень данных (см. таблицу 11) должен предоставляться вместе с технической документацией. В данном перечне изготовитель программируемых контроллеров, программных средств нечеткого управления и прикладного программного обеспечения должен описывать специфические рабочие характеристики системы нечеткого управления. Для обеспечения передачи приложений нечеткого управления между системами разных изготовителей в качестве средства для проверки возможности передачи программ используется контрольный перечень данных. Данный перечень не является исчерпывающим, он служит примером и может быть расширен изготовителями.

Т а б л и ц а 11 — Контрольный перечень данных

Технические характеристики	Информация изготовителя (примеры)
Типы данных входов и выходов функционального блока	REAL, INT
Строка комментариев программы FCL	YES, NO
Время выполнения, мс	20, 30
Требования к объему памяти, Кбайт	3, 4
Сопоставление переменных значений весовых коэффициентов и степени принадлежности от 0,0 до 1,0 с диапазоном целых чисел	0—200, 0—400
Длины идентификаторов (например, имена переменных, блоков правил, термов)	6, 8
Максимальное значение входных переменных для фаззификации	6, 8
Максимальное количество термов функции принадлежности на входную переменную	5, 7
Максимальное количество термов функции принадлежности всех входных переменных	30, 56
Максимальное количество точек функции принадлежности, соответствующих каждому терму входной переменной	3, 4, 10
Максимальное общее количество точек функции принадлежности, соответствующих всем термам входной переменной	90, 224
Максимальное значение выходных переменных для дефаззификации	6, 8
Максимальное количество термов функции принадлежности на выходную переменную	5, 7

Окончание таблицы 11

Технические характеристики	Информация изготовителя (примеры)
Максимальное общее количество термов функции принадлежности всех выходных переменных	30, 56
Максимальное количество точек функции принадлежности, соответствующих каждому терму выходной переменной	1, 4, 10
Максимальное общее количество точек функции принадлежности, соответствующих всем термам выходной переменной	90, 224
Максимальное количество блоков правил	1, 10
Максимальное количество правил в блоке	10
Максимальное количество подусловий для правила	4, 10
Максиальное количество всех правил	15
Максимальное количество подрезультатов для правила	4
Уровень вложенности ( )	1, 3

Приложение А
(справочное)

Теория

В настоящем приложении дается пояснение к определениям, приведенным в разделе 3.

А.1 Нечеткая логика

В *нечеткой логике* для описания физических переменных вместо имен и чисел (как правило, действительных чисел), применяемых в обычных разомкнутых и замкнутых системах управления, используются лингвистические значения и выражения. Термы «низкая» и «полностью открытый» являются *лингвистическими термами* физических значений «температура» или «степень открытия терморегулирующего клапана». Если входная переменная описывается *лингвистическими термами*, то она называется *лингвистическим значением.*

Каждый *лингвистический терм* описывается *нечетким множеством* М. Поэтому он математически однозначно определяется базовым множеством G из двух утверждений и *функцией принадлежности* μ. *Функция принадлежности* устанавливает принадлежность каждого элемента предметной области G (например, числовые значения шкалы времени — возраст и годы) в множестве М (например, «молодой») в форме числового значения от нуля до единицы. Если *функция принадлежности* конкретного значения равна единице, то лингвистическое утверждение, соответствующее *лингвистическому терму*, применимо во всех отношениях (например, «молодой» для возраста 20 лет). И наоборот, если она равна нулю, то нет никакого соответствия (например, «очень старый» для возраста 20 лет).

Для описания *нечетких множеств* применяются приведенные ниже формы записи.

Для конечных множеств: неупорядоченные парные множества в дифференциальной форме:

$$M = \{(x_1, \mu M(x_1)), (x_2, \mu_M(x_2)), \ldots, (x_n, \mu_M(x_n))\}, x_i \in G, i = 1,2, \ldots n \quad (A.1)$$

$\mu_M(x_i)$ считаются числовыми значениями.

Для бесконечных множеств:

$$M = \{x, \mu_M(x)\}, \quad x \in G \quad (A.2)$$

Для описания различий между *четкими* и *нечеткими* терминами на рисунке А.1 приведены *лингвистические термы* для *лингвистической переменной* «возраст». Если терм «совершеннолетний» однозначно определяется законом и тем самым отражает дискретный переход в отношении *функции принадлежности*, для терма «взрослый» не может быть назначен четкий предел возраста.



Рисунок А.1 — Функция принадлежности термов «совершеннолетний» и «взрослый»

Например, на рисунке А.2 представлены описание *лингвистической переменной* «возраст» *лингвистическими термами* и их соотношение на шкале времени «возраст в годах» с помощью *значений принадлежности*.

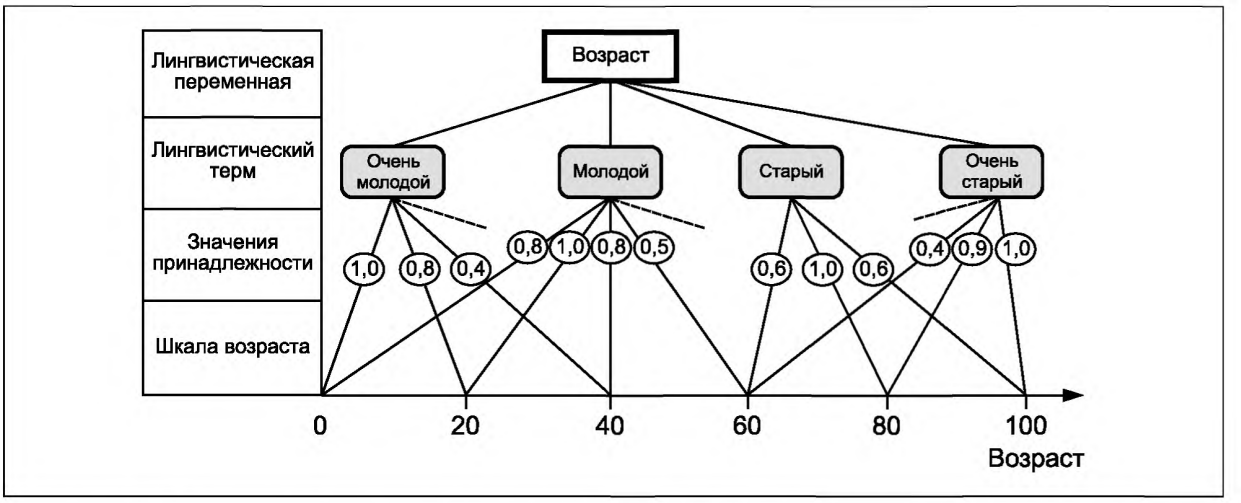


Рисунок А.2 — Описание лингвистической переменной «возраст» лингвистическими термами и их соотношение на шкале времени (возраст в годах)

Типовые формы *функции принадлежности* представлены на рисунке А.3. Следующие формы функции принадлежности относятся к специальным формам:

- определение через прямоугольник (например, интервал) с целью описания значений, а также;
- одноэлементное множество (синглтон) для альтернативного представления выходных переменных.

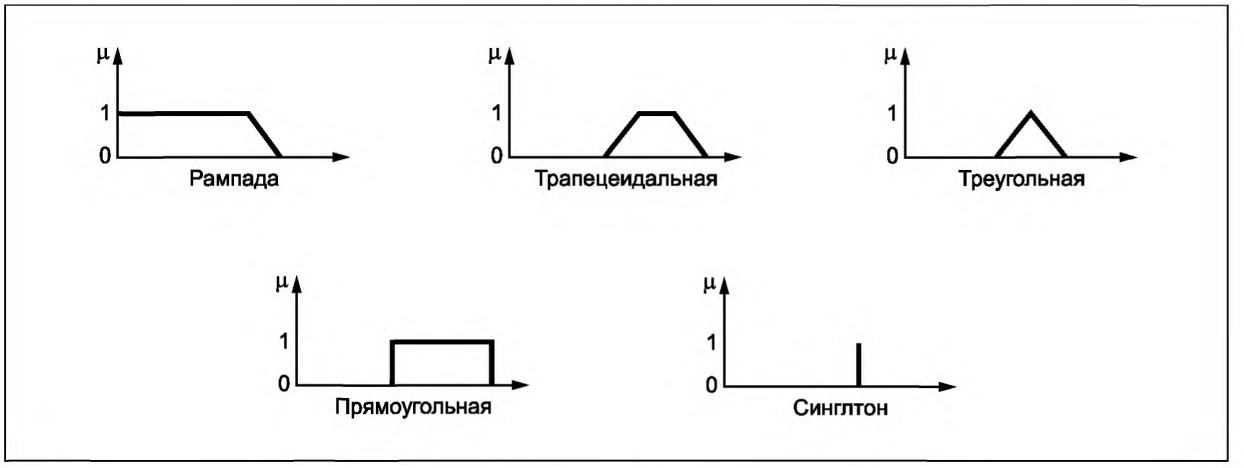


Рисунок А.3 — Типовые формы функций принадлежности

В *нечеткой логике* выражение, связывающее *лингвистические переменные* с *лингвистическими термами*, представляет собой *лингвистическое утверждение*. Такие выражения, как «температура высокая» или «температура низкая» с простой базовой структурой, в настоящем стандарте называются лингвистическим утверждением.

Лингвистическая переменная — символ сравнения — лингвистический терм (А.3)
(температура низкая)

В отличие от классической логики, в которой утверждения принимают одно из логических значений, «истина» или «ложь», лингвистические утверждения нечеткой логики характеризуются степенью принадлежности.

Знания, приобретенные на основе опыта, могут быть определены в правилах. Правило R_k имеет следующий вид:

R_k: IF *условие* P_k THEN *результат* C_k (А.4)

В данном контексте *условие* каждого *правила* включает лингвистическое утверждение или комбинацию утверждений при посредстве входных переменных, тогда как *результат* определяет выходную переменную в смысле инструкции к действию.

$$P_K = A\ AND\ B\ OR\ (NOT\ C) \quad (A.5)$$

Если *условие* и/или *результат* определяются *лингвистическими утверждениями*, *правило* называется также *лингвистическим правилом*. *База правил*, в свою очередь, состоит из нескольких *правил*. В целом в отличие от классических систем, основанных на правилах, одновременно применяются (выполняются) несколько правил. Таким образом, результаты правил должны комбинироваться друг с другом через соответствующие математические *операции*.

Соотношения между *нечеткими множествами* и операциями с *нечеткими множествами* определяются с помощью *функций принадлежности*.

Для двух нечетких множеств A и B, элементы x которых берутся из базового множества G, применяются следующие соотношения:

равенство $A = B$

истинно, если $\mu_A(x) = \mu_B(x)$, для всех $x \in G$ (A.6)

полное вхождение $A \subseteq B$

истинно, если $\mu_A(x) = \leq \mu_B(x)$, для всех $x \in G$ (A.7)

частичное вхождение $A \subset B$

истинно, если $\mu_A(x) = \leq \mu_B(x)$, для всех $x \in G$
и $\mu_A(x) < \mu_B(x)$ по крайней мере, для одного $x \in G$. (A.8)

Для двух *нечетких множеств,* A и B, элементы x которых берутся из базового множества G, могут применяться следующие *операции*:

пересечение

$A \cap B$ определяется $\mu_{A \cap B}(x) = I(\mu_A(x), \mu_B(x))$, (A.9)

где I называется операцией пересечения

объединение

$A \cup B$ определяется $\mu_{A \cup B}(x) = U(\mu_A(x), \mu_B(x))$, (A.10)

где U называется операцией объединения

дополнение

определяется с помощью $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$ (A.11)

Как в случае классических (*четких*) множеств, к нечетким множествам применяются следующие интерпретации:

пересечение относится к нечеткой операции AND;

объединение — к нечеткой операции OR;

дополнение — к нечеткой операции NOT.

Элементарные алгоритмы для математического представления пересечения, объединения и дополнения имеют следующий вид (см. также рисунок А.4):

для *пересечения*, минимум

$\mu_{A \cap B}(x) = Min\ \{\mu_A(x), \mu_B(x)\}, \quad x \in G$ (A.12)

для *объединения*, максимум

$\mu_{A \cup B}(x) = Max\ \{\mu_A(x), \mu_B(x)\}, \quad x \in G$ (A.13)

для *дополнения*, вычитание из единицы

$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \quad x \in G$ (A.14)

Для нечетких операторов AND и OR существует много возможных алгоритмов. При этом следует отметить, что операции AND и OR не могут быть выбраны произвольным способом.

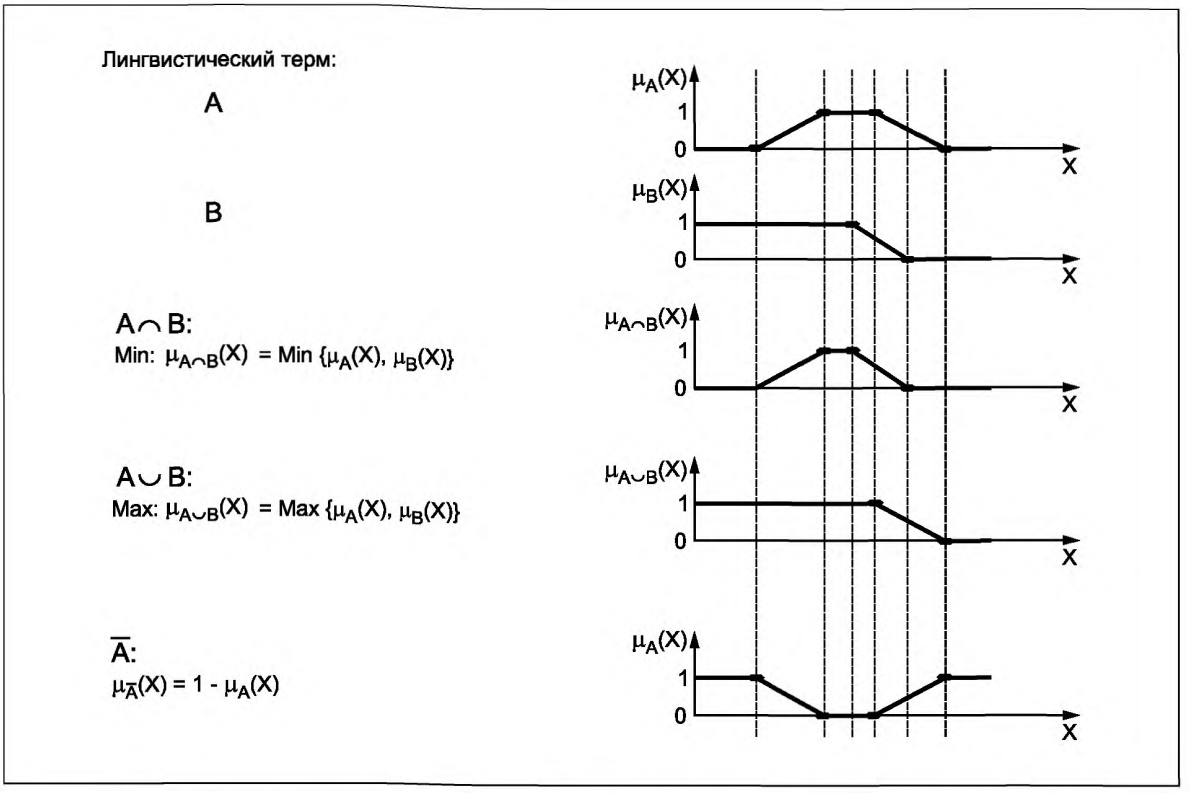


Рисунок А.4 — Алгоритмы представления операций с двумя функциями принадлежности

А.2 Нечеткое управление

Нечеткое управление означает управление техническими процессами в замкнутом и разомкнутом контурах, включая обработку измеряемых значений, которое основано на использовании нечетких правил и их преобразовании с помощью нечеткой логики.

Входная информация содержит действительные переменные в форме измеряемых переменных технологического процесса, производных переменных, а также уставок. Выходные переменные являются действительными переменными в форме регулирующих переменных. Между входными и выходными переменными технологического процесса и нечеткой средой должны быть выполнены преобразования (фаззификация, дефаззификация). Основной компонент нечеткого управления состоит из лингвистических правил из базы правил и логического вывода.

Функциональные элементы *нечеткого управления* приведены на рисунке А.5.

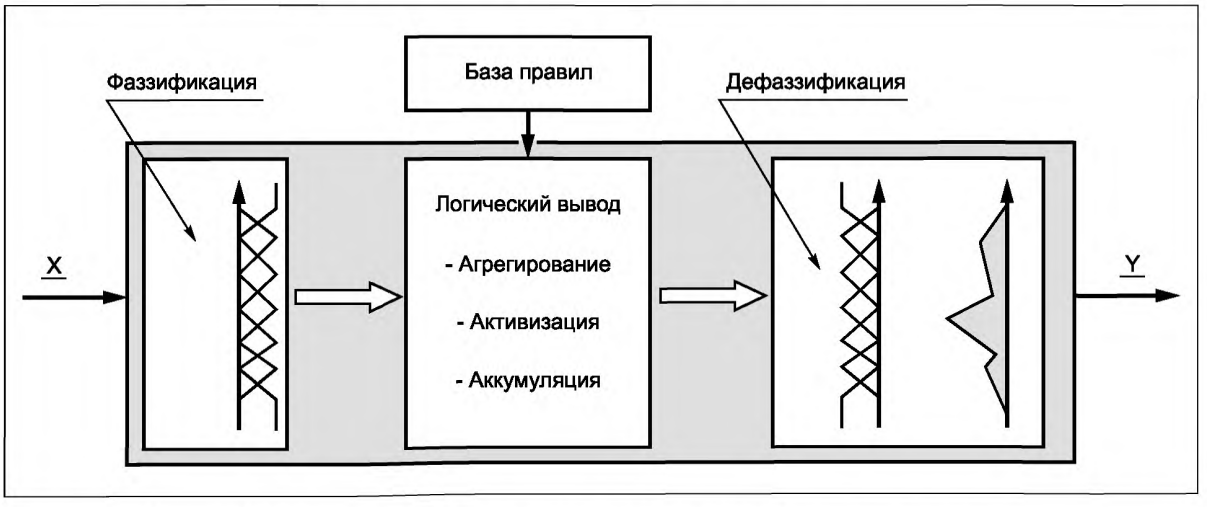


Рисунок А.5 — Структура и функциональные элементы нечеткого управления

А.2.1 Фаззификация

Фаззификацией называется сопоставление входных переменных с лингвистическими термами. Для этого устанавливается фактическая степень принадлежности входных переменных каждого *лингвистического терма* соответствующей *лингвистической переменной.*

Пример *фаззификации* приведен на рисунке А.6.

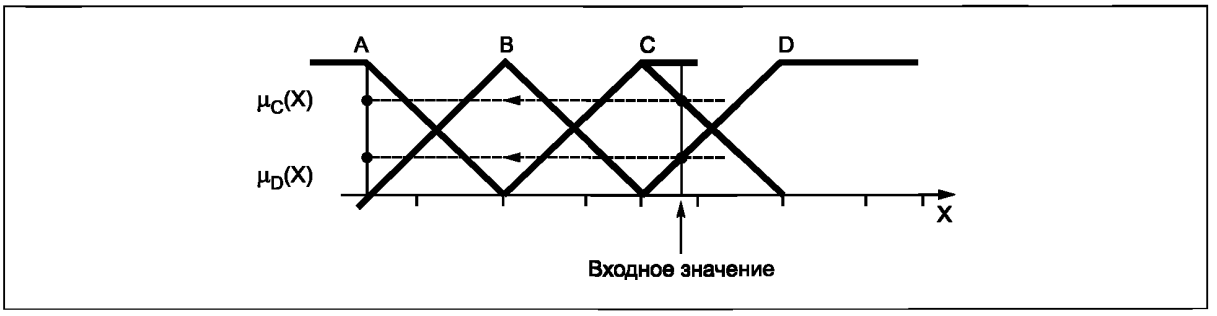


Рисунок А.6 — Принцип фаззификации (пример)

А.2.2 База правил

База правил содержит эмпирические знания, относящиеся к конкретному рассматриваемому технологическому процессу. Для интерпретации знаний используются *лингвистические правила*. При условии, что нечеткая операция AND означает MIN, а нечеткая операция OR означает MAX, можно показать, что нечеткое правило R_j, основанное на комбинации OR из m утверждений, может быть представлено m правилами, утверждения которых комбинируются только с помощью AND. На рисунках А.7 и А.8 приведены примеры различной интерпретации правил.

База правил

– Определение базы правил в текстовой форме:

R1:	IF	Условие P1	THEN	Результат C1
R2:	IF	Условие P2	THEN	Результат C2
R3:	IF	Условие P3	THEN	Результат C3
....				
R_n:	IF	Условие P_n	THEN	Результат C_n

когда условие может состоять из комбинации лингвистических утверждений, например, для правила k:

$P_k = P_{k1}$ AND P_k2 OR (NOT P_{k3})

а для результата – может быть более одного, например:

$C_k = C_{k1}, C_{k2}$

Рисунок А.7— Интерпретация базы знаний в лингвистической форме

Если имеются две входные переменные и одна выходная переменная, причем эти две входные переменные комбинируются только с помощью AND, *база правил* может быть задана в форме матрицы, в которой значения входных переменных приведены в столбцах и/или в строках, а поля матрицы содержат значения выходной переменной.

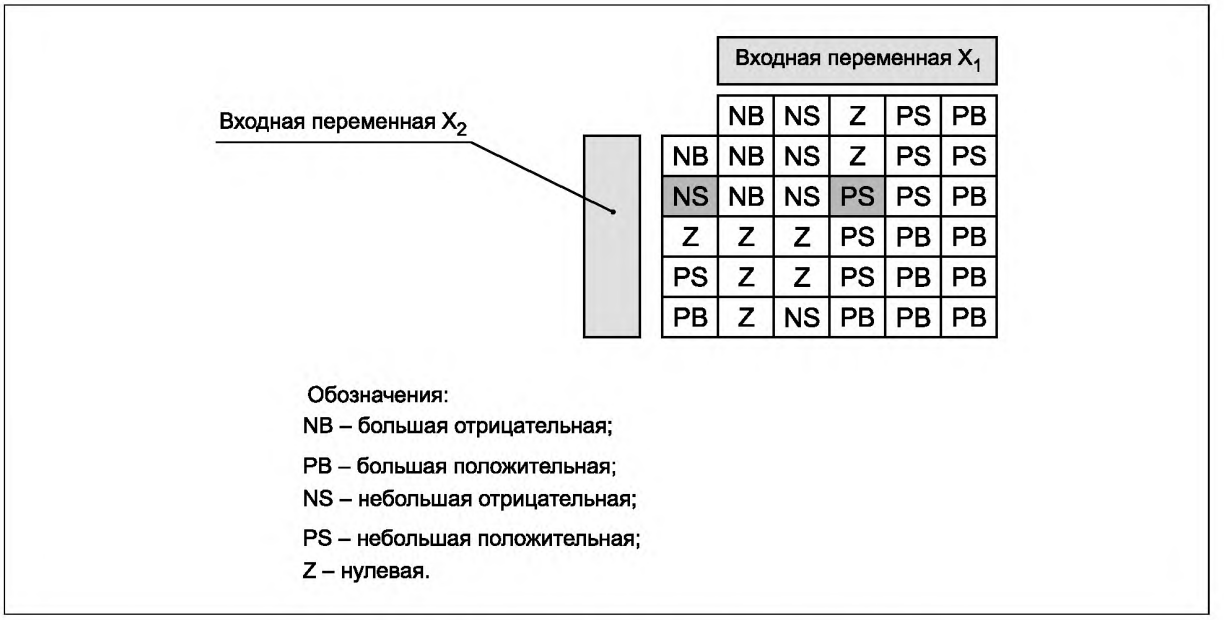


Рисунок А.8 — Матричное представление двух переменных

А.2.3 Логический вывод

Принципы и определения, используемые в настоящем стандарте, действительны для упрощенного случая нечеткой базы правил, подобной широко используемой схеме логических выводов Мамдани. Более сложные схемы логических выводов в настоящем стандарте не рассматриваются.

Логический вывод состоит в агрегировании трех подфункций, активизации и аккумуляции, приведенных на рисунке А.9. См. также рисунок А.10.

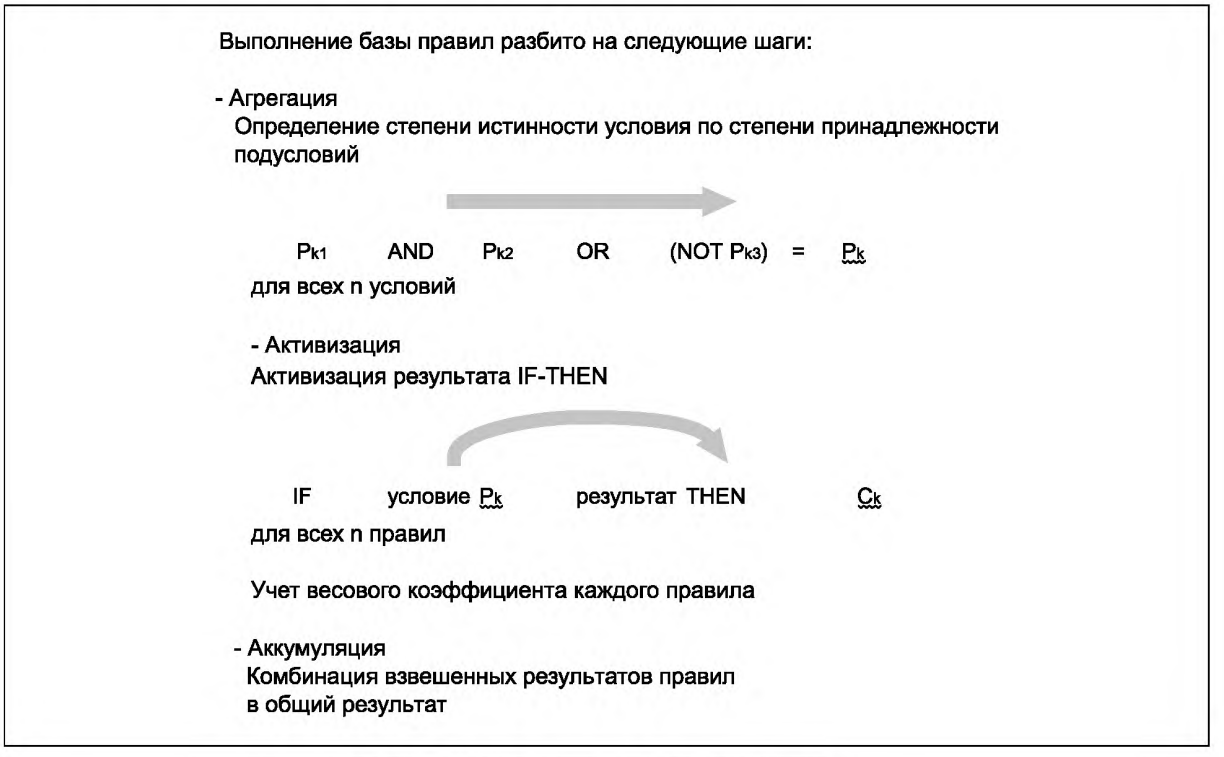


Рисунок А.9 — Элементы логического вывода

- *Агрегирование*:

Если условие состоит из одного подусловия, действительность условия идентична действительности подусловия, т. е. степень истинности условия соответствует истинности подусловия. Однако, если условие состоит из комбинации нескольких подусловий, степень истинности должна определяться агрегированием отдельных значений. Если существует комбинация AND подусловий, степень истинности рассчитывается с помощью *нечеткой операции* AND.

- *Активизация*:

Подрезультаты в *результате* относятся к выходным переменным. Затем на основе степени истинности *условия*, полученного при агрегировании, определяется *степень принадлежности результата* (*результат* IF A THEN B). Как правило, для активизации используется MIN или PROD.

Если база правил содержит правила с весовым коэффициентом w_k, где $w_k \in [0, 1]$, это может быть реализовано с помощью умножения.

$$c_k^* = w_k \times c_k \qquad (A.15)$$

- *Аккумуляция*:

Результаты применения *правил* комбинируются для получения общего результата. Для *аккумуляции, как правило,* используется *алгоритм максимума*. В таблице А.1 приведены *операции*, используемые, как правило, для отдельных шагов *логического вывода*.

В зависимости от комбинации *операций* на отдельных шагах получаются разные стратегии *логического вывода*. Наиболее известные так называемые методы *максимума-минимума* (MaxMin Inference) и *максимума-произведения* (MaxProd Inference), которые применяют *максимум* для *аккумуляции* и *минимум* или алгебраическое произведение для *активизации*. В случае метода *максимума-минимума функции принадлежности нечетких множеств результатов* ограничены *степенью истинности условия* и комбинируются для создания нечеткого множества с помощью формирования максимума. В случае метода *максимума-произведения*, наоборот, *функции принадлежности нечеткого множества результатов* взвешиваются, т. е. умножаются по степени истинности условия, а затем комбинируются.

Таблица А.1 — Шаги логического вывода и широко используемые алгоритмы

Шаг логического вывода	Операции	Алгоритмы
Агрегирование		
для AND	*Минимальная*	$a_k = \text{Min}\{a_{K1}(\times 1), a_{K2}(\times 2)\}$
для OR	*Максимальный*	$a_k = \text{Max}\{a_{K1}(\times 1), a_{K2}(\times 2)\}$
Активизация		
преобразование IF-THEN-результата		
	Минимальная	$c_k' = \text{Min}\{a_K, \mu_{oc}(u)\}$
весовой коэффициент каждого правила		
	Умножение	$c_k = \text{Mult}\{\omega K, c_K'\} = \omega_K \times c_K'$
Аккумуляция	*Максимальный*	$\mu = \text{MAX}\{c_i(u)\}$

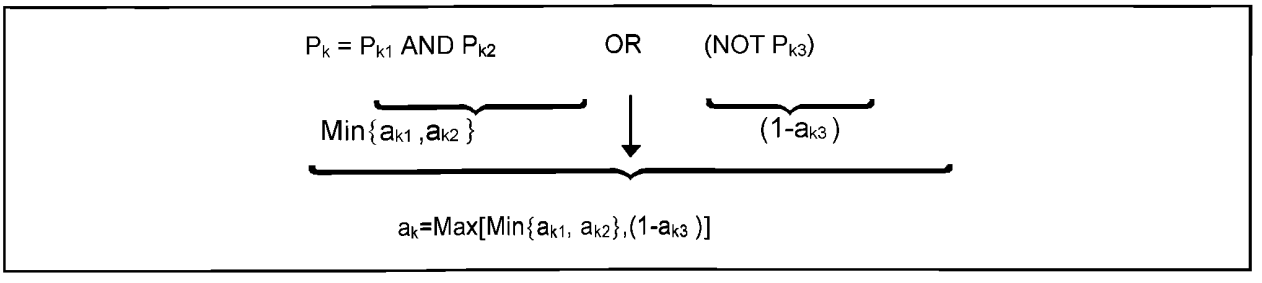
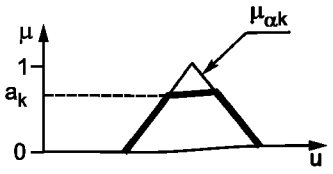


Рисунок А.10а — Пример, демонстрирующий принципы агрегирования

- Активизация

- Преобразование результата IF-THEN, например, операция минимума



Функция принадлежности лингвистического терма α выходной переменной u, связанной с правилом k

например, алгебраическое произведение

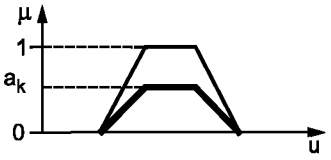


Рисунок А.10b — Принципы активизации (пример)

- Аккумуляция

например, операция максимизации

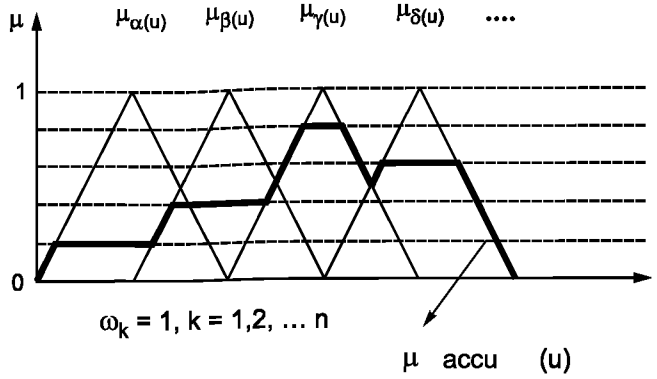


Рисунок А.10c — Принципы аккумуляции (пример)

А.2.4 Дефаззификация (см. рисунок А.11)

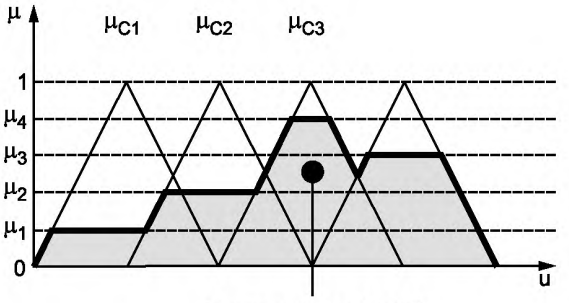
Логический вывод в качестве результата дает *нечеткое множество* или его функцию принадлежности. Элемент управления не может напрямую обрабатывать данную нечеткую информацию, поэтому результат процесса *логического вывода* должен быть преобразован в *четкие* числовые значения. В этой связи *четкое* число, которое необходимо определить (как правило, действительное число), должно обеспечить представление информации, содержащейся в *нечетком множестве*.

Дефаззификация

Преобразование нечеткого результата логического вывода в четкую выходную переменную U.
Дефаззификация методом центра тяжести (CoG).

Четкая выходная переменная определяется как значение абсциссы U центра тяжести площади под функцией принадлежности:

- пример метода центра тяжести.



$$U = \frac{\int_{Min}^{Max} u*\mu(u)*du}{\int_{Min}^{Max} \mu(u)*du}$$

Центр тяжести (CoG)

Рисунок А.11а — Методы дефаззификации

Общими методами являются также следующие:

Крайний левый максимум LM

Определяется значение выходной переменной, для которого *функция принадлежности* выхода достигает ее крайнего левого максимума.

Крайний правый максимум RM

Определяется значение выходной переменной, для которого *функция принадлежности* выхода достигает ее крайнего правого максимума.

Крайний правый максимум: $\sup(X')/\mu(X') = \sup_{x \in [Min, Max]} \mu(X)$

Крайний левый максимум: $\inf(X')/\mu(X') = \sup_{x \in [Min, Max]} \mu(X)$

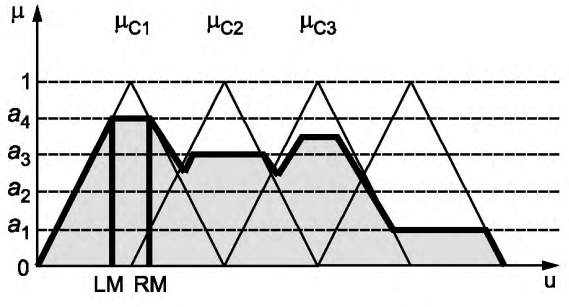


Рисунок А.11b — Разница между методами крайнего левого максимума и крайнего правого максимума

Метод центра площади

В данном методе выходное значение определяется как значение абсциссы центра, которое делит площадь под *функцией принадлежности* на две области одинакового размера.

Примечание 1 — Центр тяжести эквивалентен центроиду.

Примечание 2 — Центр площади эквивалентен медиане.

Примечание 3 — При использовании синглтонов CoA не применяется.

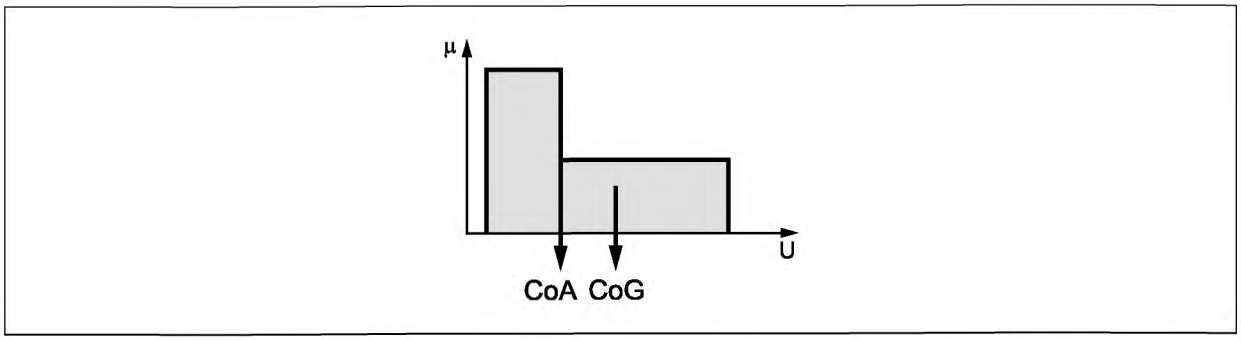


Рисунок А.11c — Разница между методами центра площади и центра тяжести

Примечание — Формулы методов дефаззификации CoA и CoG приведены в таблице 2.

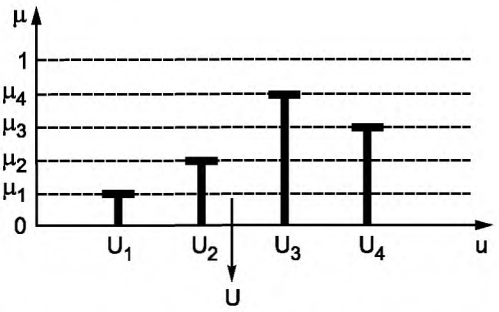
Дефаззификация

Если функции принадлежности выходных переменных являются синглтонами, расчет выполняется с помощью:

- метода центра тяжести для синглтонов (COGS)

$$U = \frac{\sum_{i=1}^{p} [U_i^* \mu_i]}{\sum_{i=1}^{p} [\mu_i]},$$

где μ – результаты аккумуляции



В случае синглтонов метод центра площади применяться не может. Пример для 4 синглтонов выше:

$$U = \frac{[U_1^* \mu_1 + U_2^* \mu_2 + U_3^* \mu_4 + U_4^* \mu_3]}{[\mu_1 + \mu_2 + \mu_3 + \mu_4]}$$

Рисунок А.11d — Методы дефаззификации

А.3 Осуществление нечеткого управления

С точки зрения информационной технологии нечеткое управление является экспертной системой, основанной на правилах. С точки зрения технологии систем управления это, как правило, полевой контроллер с нелинейной характеристикой. На рисунке А.12 приведены примеры характеристических кривых нечеткого управления. Текущие значения его выходных переменных зависят исключительно от текущих значений входных переменных, а не от предыдущих значений, за исключением случая, когда нет активных правил и не было определено значение по умолчанию. Если контроллер должен обладать динамическими свойствами, должны быть предусмотрены внешние по отношению к функциональному блоку динамические функции.

Как правило, это дифференцирующие и интегрирующие элементы. Выходные переменные этих функций являются для нечеткого управления дополнительными входными переменными. Это также применимо к ошибкам

управления, которые аналогичным образом должны формироваться вне нечеткого управления. С другой стороны, выходные переменные нечеткого управления могут быть переданы операторам для обработки корректирующих переменных (например, интегрирующий элемент для алгоритмов скорости) или распределены между различными элементами управления.

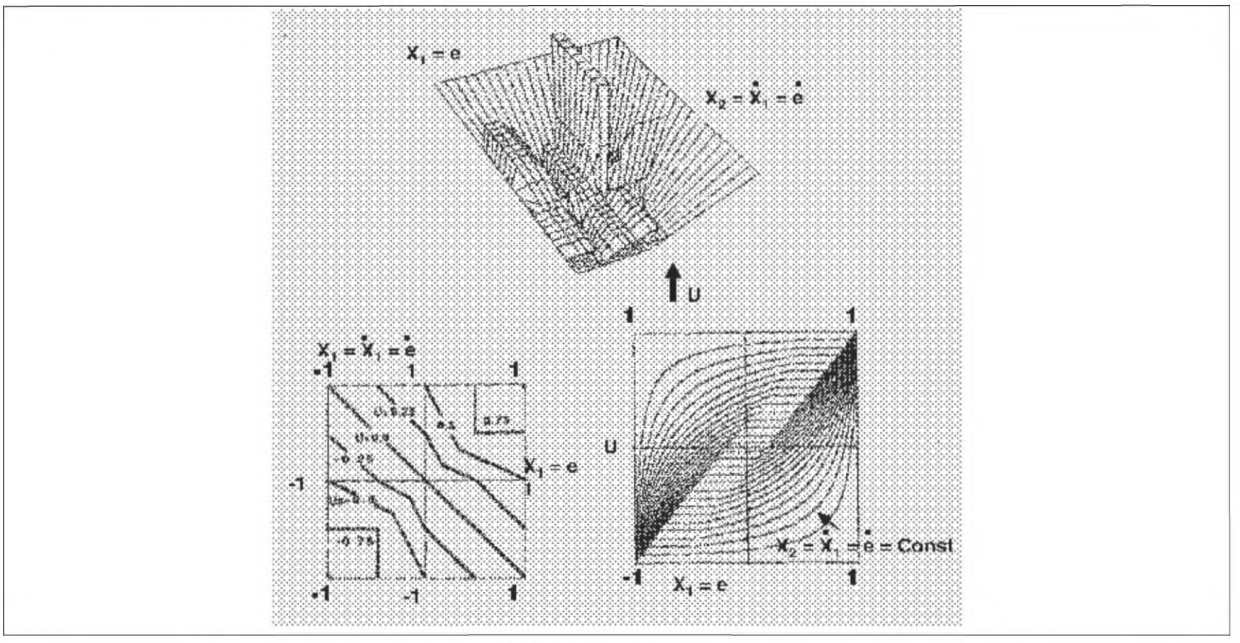


Рисунок А.12 — Примеры характеристических кривых нечеткого управления

Базовая структура основанных на нечетком управлении контроллеров приведена на рисунке А.13а, а на рисунке А.13b приведен пример, показывающий, как из разности между уставкой и переменной технологического процесса формируется ошибка. Эта разность, как и ее производная по времени, передается нечеткому управлению в виде двух переменных, являющихся независимыми от ее объективного восприятия. Корректирующая переменная получается из выходной переменной интегрированием по времени. Если база правил разработана таким образом, что она описывает алгоритм скорости, то основанный на нечетком управлении контроллер отображает динамическое поведение, подобное пропорциональному интегрированию.

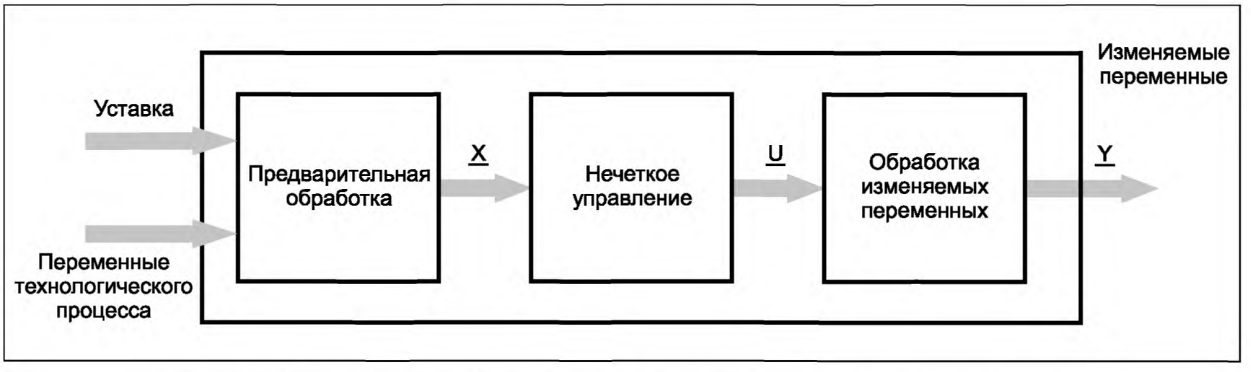


Рисунок А.13а — Основанный на нечетком управлении контроллер: базовая структура

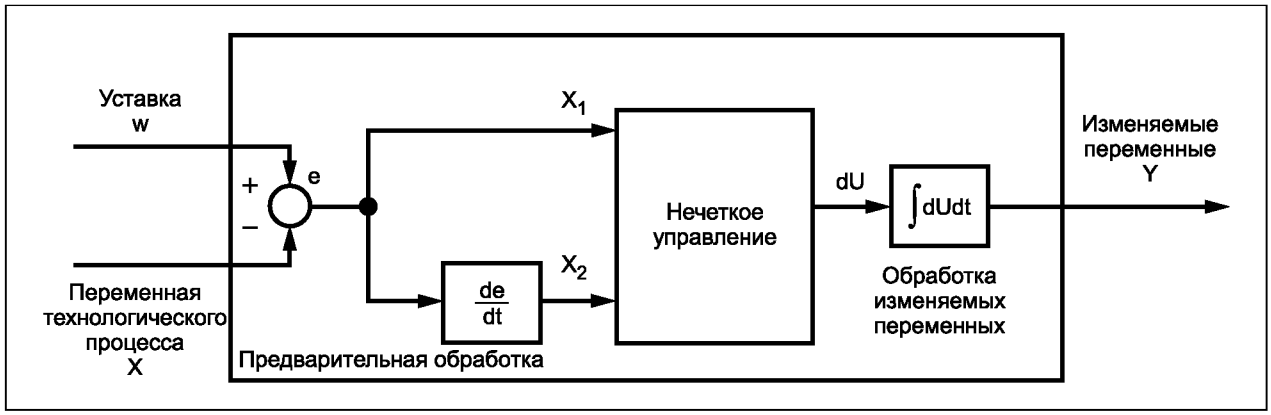


Рисунок А.13b — Пример основанного на нечетком управлении контроллера

Приложение В
(справочное)

Примеры

Одним из главных применений программируемых контроллеров является их использование совместно с обычными ПИД-регуляторами для улучшения качества управления ПИД-регулятора. Приведенные ниже примеры показывают принципиальные возможности, дающие общее представление о том, где может быть использовано нечеткое управление.

В.1 Предварительное регулирование

Устройство нечеткого управления дополняет обычный контроллер замкнутой системы корректирующим сигналом для управляемого клапана (см. рисунок В.1).

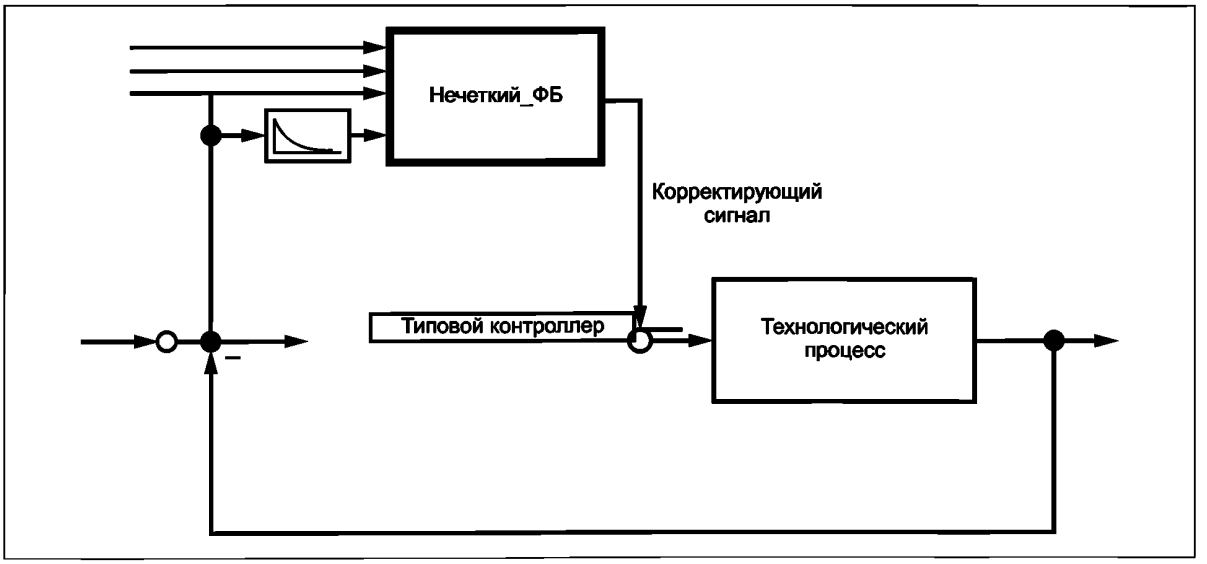


Рисунок В.1 — Пример предварительного регулирования

В.2 Адаптация характеристик обычных ПИД-регуляторов

Устройство нечеткого управления используется для адаптации параметров управления ПИД-регулятора (см. рисунок В.2).

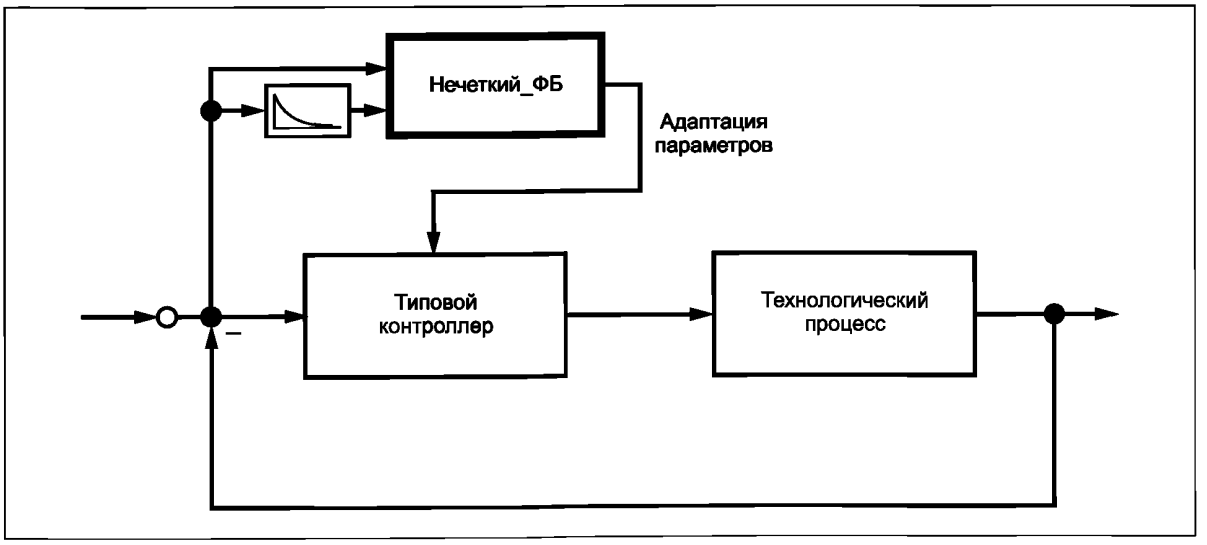


Рисунок В.2 — Пример адаптации параметров

В.3 Прямое нечеткое управление технологическим процессом

Другой областью применения является включение эмпирических знаний о процессе и стратегий лингвистического управления непосредственно в промышленную автоматизацию. Данный метод используется во многих технологических процессах, в которых необходимо вмешательство оператора (см. рисунок В.3).

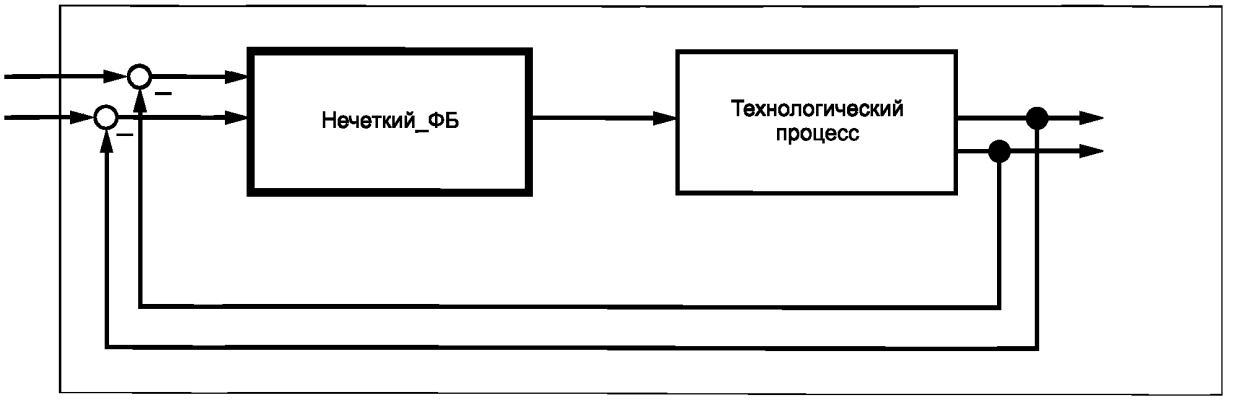


Рисунок В.3 — Пример прямого нечеткого управления

Приложение С
(справочное)

Промышленный пример: кран для контейнеров

Краны для контейнеров применяются в большинстве портов при погрузке контейнеров на суда и при их выгрузке. Краны подхватывают отдельные контейнеры тросами, закрепленными на тележке крана. Тележка крана перемещается по горизонтальному рельсовому пути. Как показано на рисунке С.1, когда контейнер подхвачен и тележка начинает двигаться, контейнер начинает раскачиваться. Раскачивающийся контейнер возможно перемещать, но невозможно поставить.

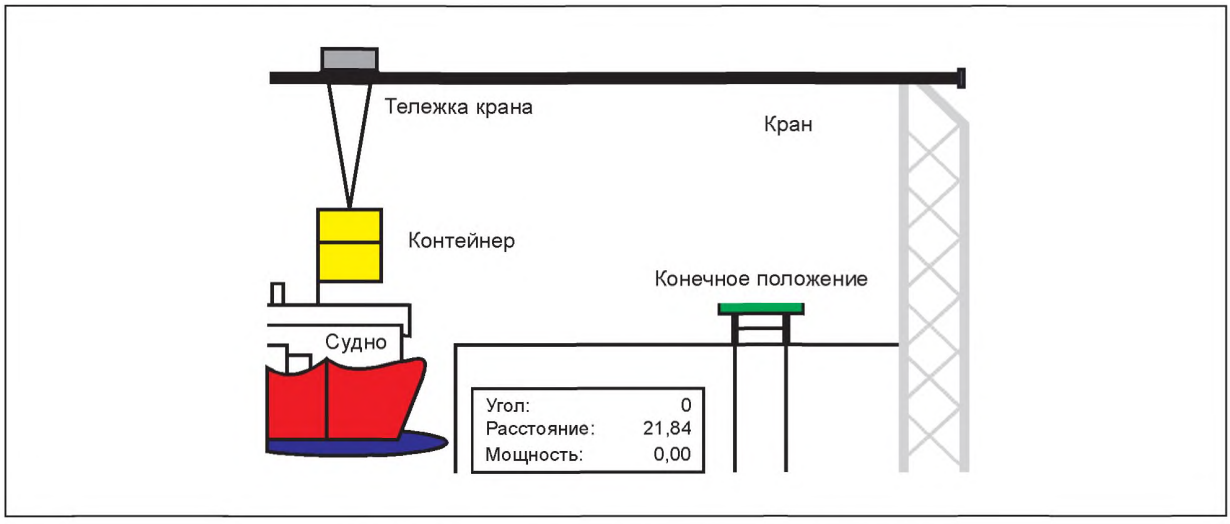


Рисунок С.1 — Промышленный пример: кран для контейнеров

Анализ действий оператора показывает, что эти действия подчиняются ряду общих правил:

a) Начало движения тележки при средней мощности двигателя.

b) Если в начале движения тележка крана находится далеко от цели, необходимо отрегулировать мощность двигателя так, чтобы контейнер был чуть сзади тележки.

c) Если тележка крана находится ближе к цели, необходимо уменьшить мощность двигателя так, чтобы контейнер был чуть впереди тележки.

d) Когда контейнер находится рядом с целевым положением, необходимо прибавлять или убавлять мощность в области среднего значения, в зависимости от направления раскачивания.

e) Когда контейнер находится над целью и раскачивание равно нулю, необходимо остановить двигатель.

Для автоматизации управления таким краном используются датчики положения («расстояние») тележки крана и угла раскачивания контейнера («угол»). Выходной величиной является мощность двигателя. Сначала для всех переменных должны быть определены лингвистические переменные. Лингвистические переменные «расстояние», «угол» и «мощность двигателя» (см. рисунки С.2, С.3 и С.4) делятся на пять лингвистических термов. Для функции принадлежности используются формы рампады, треугольника и синглтоны. На приведенных ниже рисунках представлены лингвистические переменные и лингвистические термы.

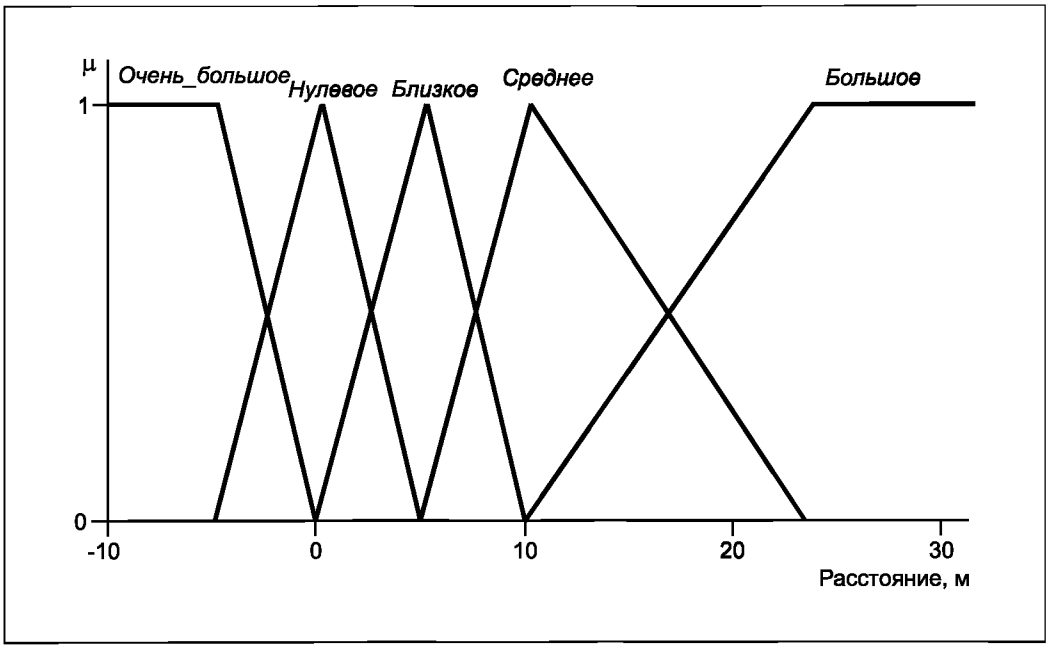


Рисунок С.2 — Лингвистическая переменная «расстояние» между тележкой крана и целевым положением

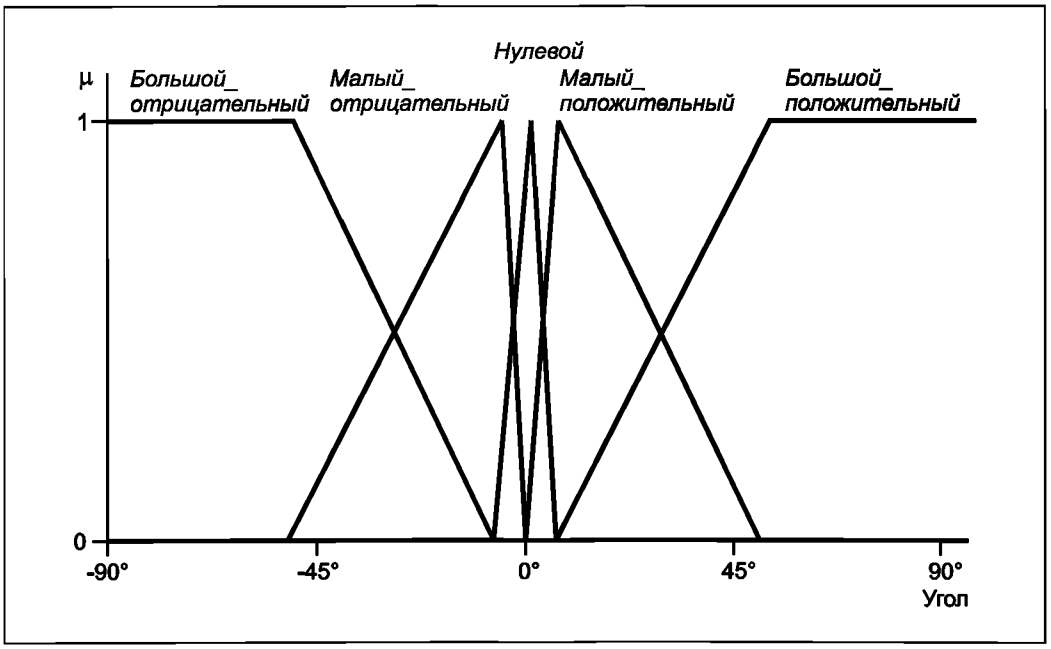


Рисунок С.3 — Лингвистическая переменная «угол» между контейнером и тележкой крана

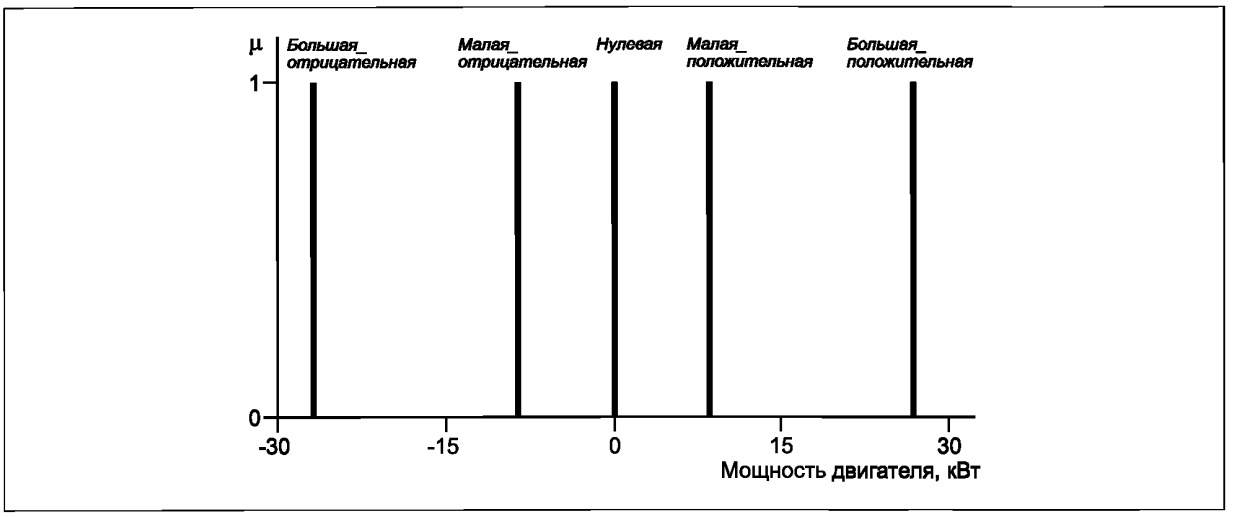


Рисунок С.4 — Лингвистическая переменная «мощность»

С помощью данных лингвистических термов для описания текущего состояния крана в базу правил могут быть преобразованы пять общих правил. На рисунке С.5 дано определение базы правил в записи FCL. Правило 2 было преобразовано в два правила, чтобы соответствовать формату «if-then». Здесь не используются некоторые из определенных функций принадлежности, и система должна быть проверена на наличие неопределенных выходов, например расстояние: очень_дальнее; угол: большой_положительный.

Правило 1: IF расстояние IS большое	AND угол IS нулевой	THEN мощность IS средняя_положительная
Правило 2: IF расстояние IS большое	AND угол IS малый_отрицательный	THEN мощность IS большая_положительная
Правило 3: IF расстояние IS большое	AND угол IS большой_отрицательный	THEN мощность IS средняя_положительная
Правило 4: IF расстояние IS среднее	AND угол IS малый_отрицательный	THEN мощность IS средняя_отрицательная
Правило 5: IF расстояние IS близкое	AND угол IS малый_положительный	THEN мощность IS средняя_положительная
Правило 6: IF расстояние IS нулевое	AND угол IS нулевой	THEN мощность IS нулевая

Рисунок С.5 — База правил

В таблице С.1 приведены шаги логического вывода и использованные операции соответственно.

Т а б л и ц а С.1 — Шаги логического вывода и присвоенная операция

Шаг логического вывода	*Операции*
Агрегирование	
AND	*Минимальная*
Активизация	
преобразование IF-THEN-результата	
	Минимальная
Аккумуляция	*Максимальный*

Необходимо учитывать текущее состояние крана, когда расстояние тележки крана до целевого положения равно 12 м, а угол контейнера составляет плюс 4°. Для иллюстрации будет принято подмножество трех правил.

На рисунках С.6 и С.7 показано, как для данного случая рассчитывается фаззификация.

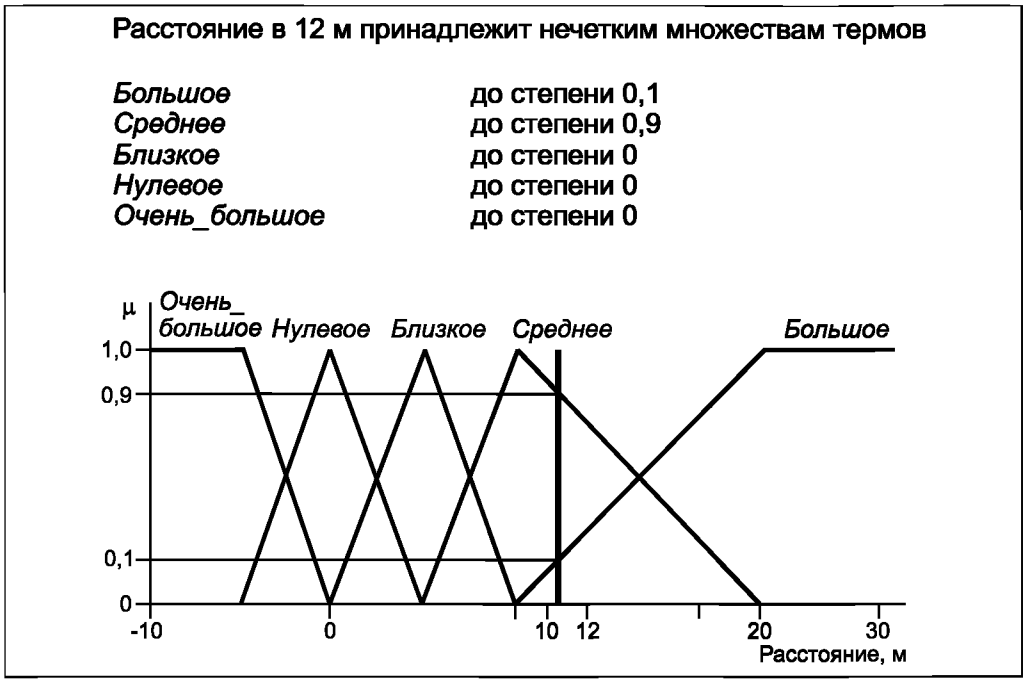


Рисунок С.6 — Фаззификация лингвистической переменной «расстояние»

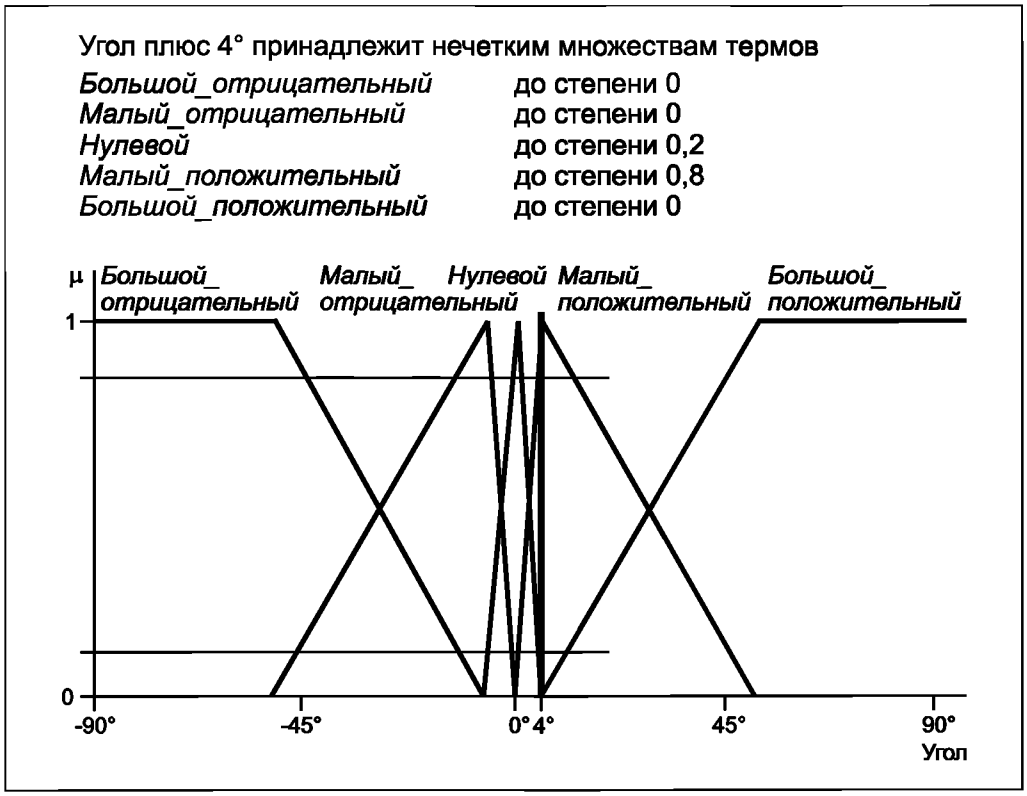


Рисунок С.7 — Фаззификация лингвистической переменной «угол»

Расстояние 12 м преобразуется в значение лингвистической переменной {0,1; 0,9; 0, 0, 0}, которое может быть интерпретировано как «еще среднее, но уже немного большое». Угол плюс 4° преобразуется в значение лингвистической переменной {0; 0; 0,2, 0,8, 0}, которое может быть интерпретировано как «небольшое положительное, почти нулевое».

- Логический вывод

После того, как все входные переменные были преобразованы в значения лингвистических переменных, шаг нечеткого логического вывода дает возможность идентифицировать правила, применяемые к текущему состо-

янию, и вычислить значения выходной лингвистической переменной. На рисунке С.8 для иллюстрации показано подмножество трех правил:

RULE 1: IF расстояние IS среднее	AND угол IS малый_ положительный	THEN мощность IS средняя_положительная
RULE 2: IF расстояние IS среднее	AND угол IS нулевой	THEN мощность IS нулевая
RULE 3: IF расстояние IS дальнее	AND угол IS нулевой	THEN мощность IS средняя_положительная

Рисунок С.8 — Подмножество трех правил

Логический вывод состоит из трех подфункций: агрегирования, активизации и аккумуляции.

- Агрегирование (см. рисунок С.10)

Определение степени истинности предположения по степени принадлежности элементарных утверждений.

RULE 1: Расстояние = среднее	AND	Угол = малый_положительный	=	P1
RULE 2: Расстояние = среднее	AND	Угол = нулевой	=	P2
RULE 3: Расстояние = дальнее	AND	Угол = нулевой	=	P3
Для всех предположений				

Рисунок С.9 — Элементы агрегирования

Операция Min соответствует агрегированию AND. На рисунке С.9 показано, как для данного случая вычисляется агрегирование.

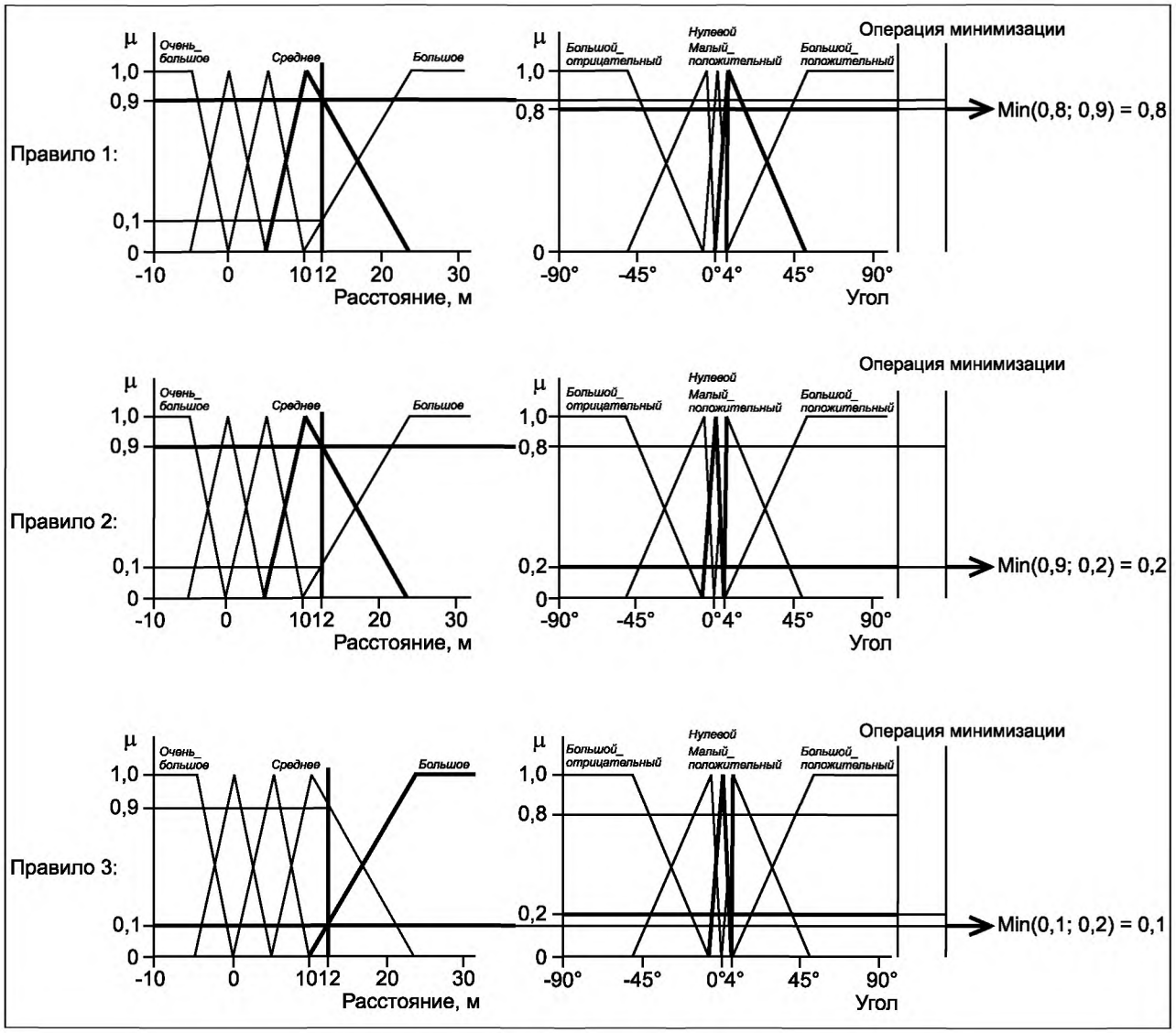


Рисунок С.10 — Принципы агрегирования

- Активизация (см. рисунок С.12)

Преобразование результата IF-THEN

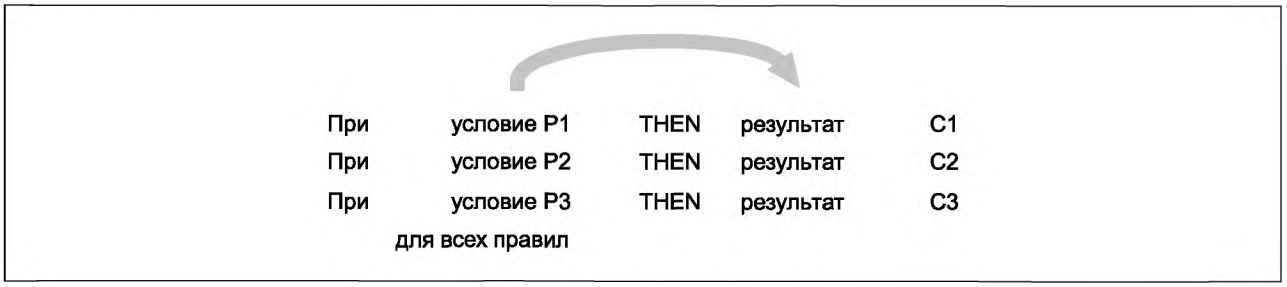


Рисунок С.11 — Элементы активизации

На рисунке С.11 показано, как для данного случая рассчитывается активизация. Результат агрегирования описан с левой стороны, результат активизации — с правой стороны.

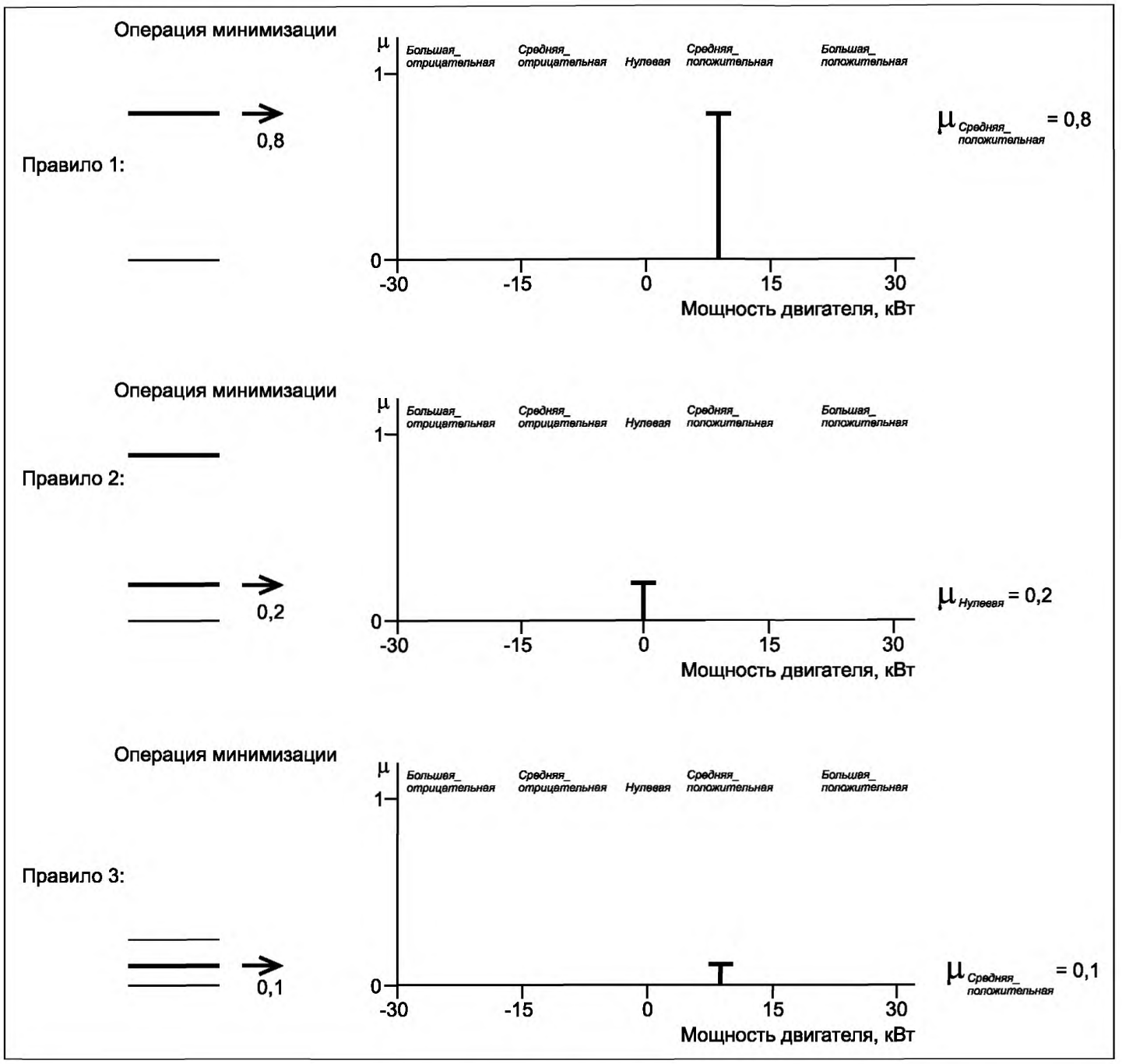


Рисунок С.12 — Принципы активизации

- Аккумуляция (см. рисунок С.14)

Комбинация взвешенных результатов правил в общий результат.

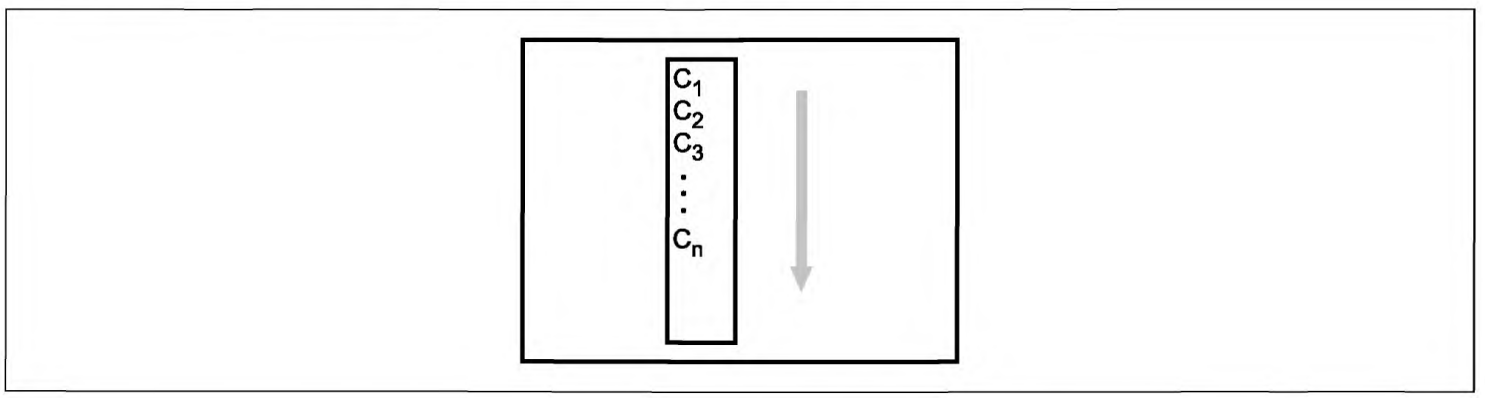


Рисунок С.13 — Элементы аккумуляции

Результат аккумуляции правил с 1 по 3 показан в нижней части рисунка С.13. Результат среднего_положительного синглтона рассчитывается, например, как Max(0,8; 0,1) = 0,8.

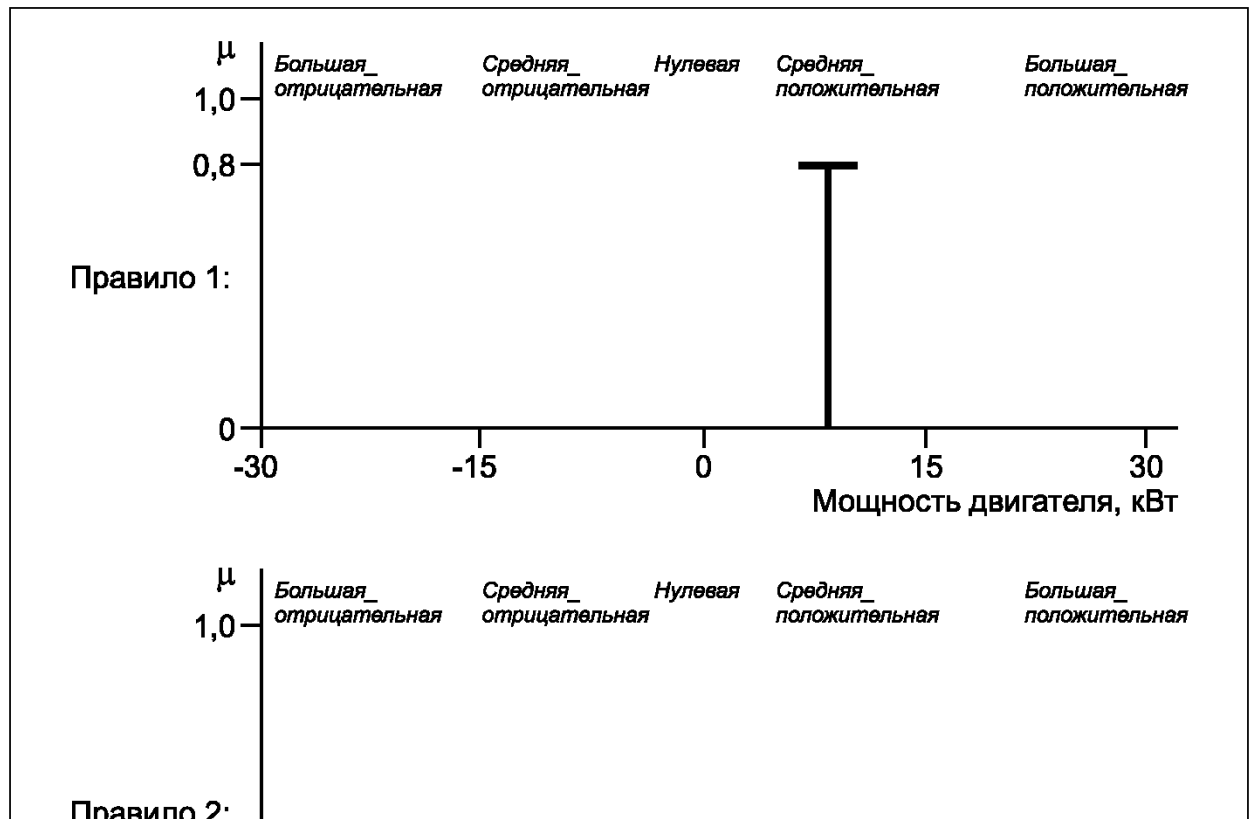


Рисунок С.14 — Принципы аккумуляции

- Дефаззификация (см. рисунок С.15)

Логический вывод в качестве результата дает нечеткое множество или его функцию принадлежности. Чтобы использовать его для установки мощности двигателя, он должен быть преобразован в четкое числовое значение. Значение, которое необходимо определить (как правило, действительное число), должно обеспечить наилучшее представление информации, содержащейся в полученном нечетком множестве. Используя метод максимальной высоты, четкую выходную переменную мощность двигателя рассчитывают следующим образом

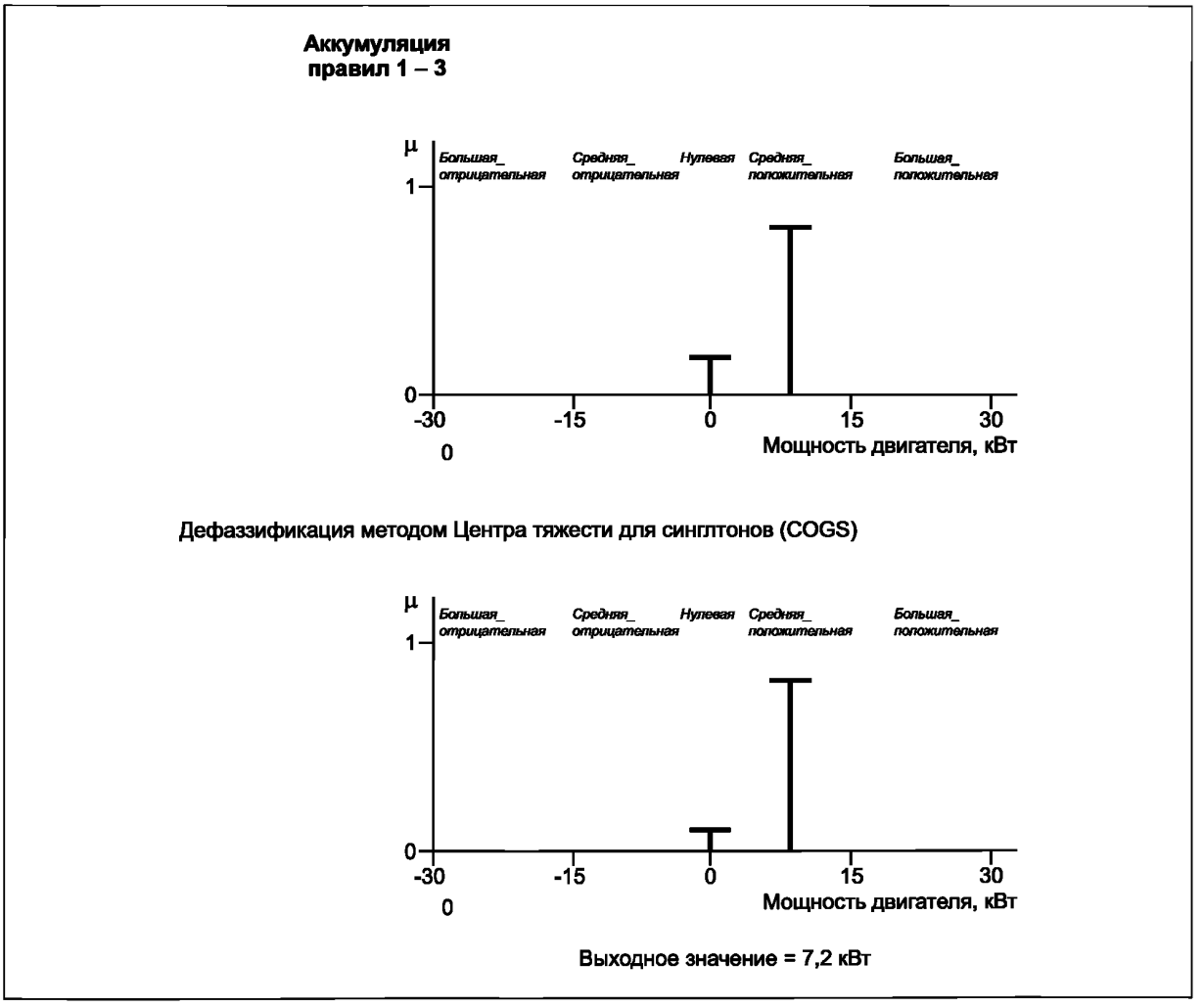


Рисунок С.15 — Дефаззификация

- Реализация примера с краном для контейнеров в FCL (см. рисунок С.16):

```
FUNCTION_BLOCK кран_для_контейнеров

VAR_INPUT
        расстояние: REAL;
        угол: REAL;
END_VAR

VAR_OUTPUT
        мощность: REAL;
END_VAR

FUZZIFY расстояние
        TERM слишком_большое        := (-5, 1) (0, 0);
        TERM нулевое                := (-5, 0) ( 0, 1) ( 5,0);
        TERM близкое                := ( 0, 0) ( 5, 1) (10,0);
        TERM среднее                := ( 5, 0) (10, 1) (22,0);
        TERM большое                := (10, 0) (22,1);
END_FUZZIFY

FUZZIFY угол
        TERM большой_отрицательный      := (-50, 1) (-5, 0);
        TERM небольшой_отрицательный    := (-50, 0) (-5, 1) (0,0);
        TERM нулевой                    := (-5, 0) (0, 1) (5,0);
        TERM небольшой_положительный    := (0, 0) (5, 1) (50,0);
        TERM большой_положительный      := (5, 0) (50, 1);
END_FUZZIFY

DEFUZZIFY мощность
        TERM большая_отрицательная      := -27;
        TERM средняя_отрицательная      := -9;
        TERM нулевая                    := 0;
        TERM средняя_положительная      := 9;
        TERM большая_положительная      := 27;
        METHOD: CoGS;
        DEFAULT:= 0;
END_DEFUZZIFY

RULEBLOCK №1
        AND: MIN;
        ACCU: MAX;
        RULE 1: IF расстояние IS большое AND угол IS нулевой THEN мощность IS средняя_положительная;
        RULE 2: IF расстояние IS большое AND угол IS небольшой_отрицательный THEN мощность IS
        большая_положительная;
        RULE 3: IF расстояние IS большое AND угол IS большой_отрицательный THEN мощность IS
        средняя_положительная;
        RULE 4: IF расстояние IS среднее AND угол IS небольшой_отрицательный THEN мощность IS
        средняя_отрицательная;
        RULE 5: IF расстояние IS большое AND угол IS небольшой_положительный THEN мощность IS
        средняя_положительная;
        RULE 6: IF расстояние IS нулевое AND угол IS нулевой THEN мощность IS нулевая;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Рисунок С.16 — Пример на SCL

Функциональный блок FCL должен вызываться так же, как и с помощью программ, в соответствии с МЭК 61131-3. Это обосновано тем, что с внешней точки зрения нечеткий блок имеет единственный интерфейс через свои переменные.

В соответствии с МЭК 61131-3 вызов для приведенного выше примера может быть:

```
кран_для_контейнеров (расстояние: = INP_DIS, угол:= INP_ANG);
A:= кран_для_контейнеров.мощность;
```

Переменные INP_DIS и INP_ANG могут быть привязаны непосредственно к словам ввода контроллера или вычисляться из других значений.

Приложение D
(справочное)

Пример использования переменных в блоке правил

Печенье готовят в туннельной печи. Цвет печенья определяется с помощью датчика цвета в конце печи. Измерение цвета осуществляется с помощью трехмерной величины, для определения принадлежности измеренного цвета к одному из трех приведенных ниже классов используется нечеткая классификация: коричневый, светлый, темный. В печи также измеряется влажность. Печь может управляться двумя температурными контурами: один в первой и один во второй половине печи.

Принцип управления приведен на рисунках D.1 и D.2.

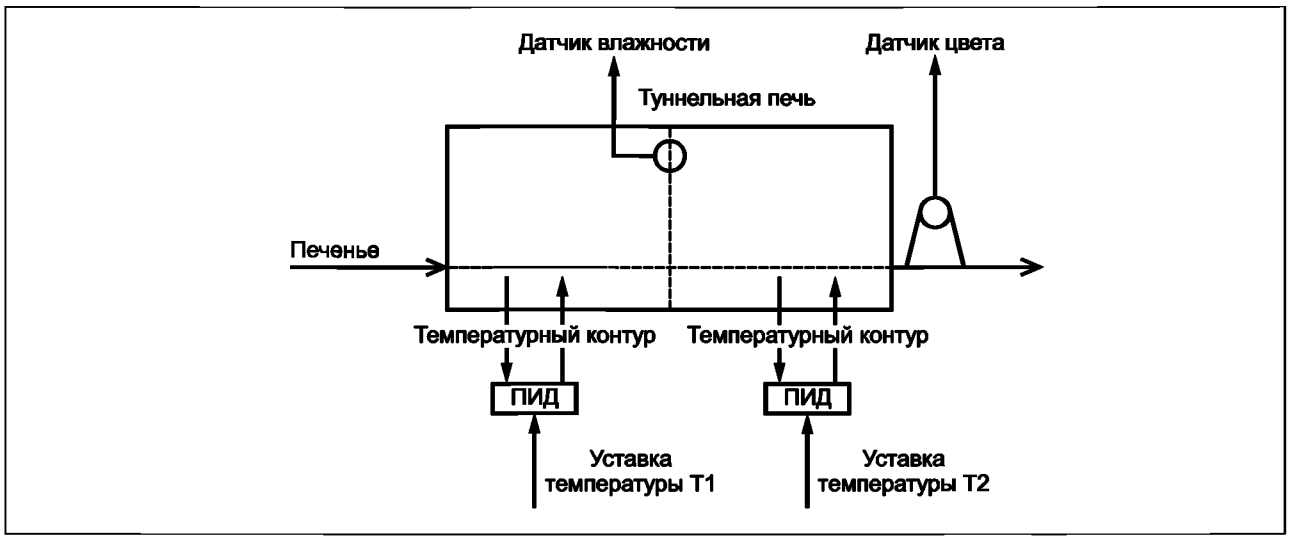


Рисунок D.1 — Принцип управления системой

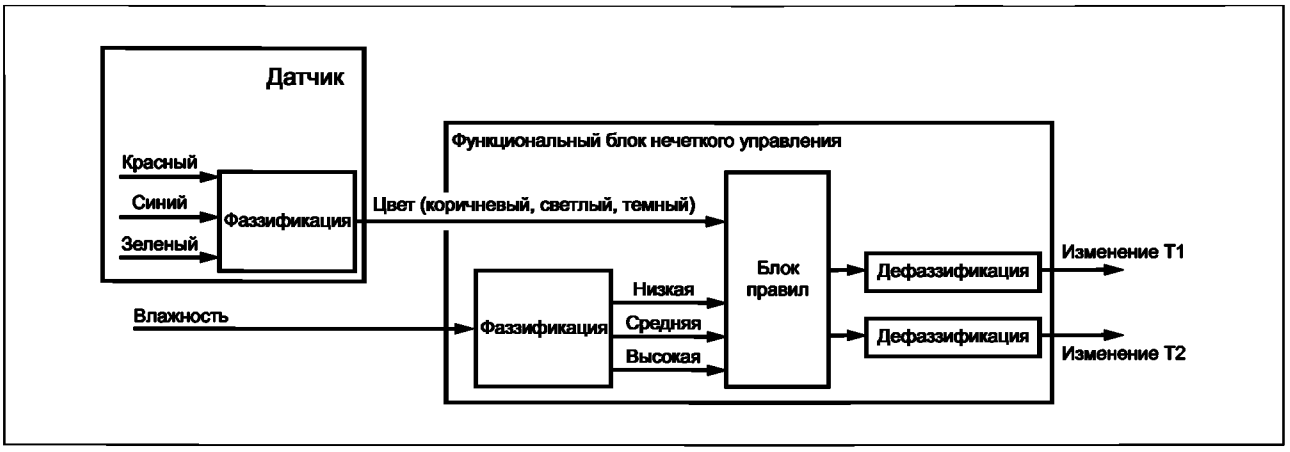


Рисунок D.2 — Принцип управления печью на основе нечеткой логики

Блок правил содержит следующие пять правил (см. рисунок D.3):

IF влажность IS средняя AND цвет IS коричневый	THEN dT1 IS ноль AND dT2 IS ноль
IF влажность IS высокая	THEN dT1 IS положительная
IF влажность IS низкая	THEN dT1 IS отрицательная
IF влажность IS средняя AND цвет IS светлый	THEN dT2 IS положительная
IF влажность IS средняя AND цвет IS темный	THEN dT2 IS отрицательная

Рисунок D.3 — Блок правил

Синтаксис FCL для данного примера приведен на рисунке D.4.

Примечание — Вместо использования трех переменных — светлый, коричневый и темный — для цвета может также использоваться только одна переменная перечисляемого типа данных, как показано ниже. В данном примере dT2 не определяется, если влажность высокая или низкая.

```
TYPE
STRUCT тип_цвета
        коричневый REAL;
        светлый: REAL;
        темный: REAL;
END_STRUCT
END_TYPE

FUNCTION_BLOCK управление_печью
VAR_INPUT
        влажность:      REAL;
        цвет : тип_цвета;
END_VAR

VAR_OUTPUT
        dT1: REAL;
        dT2: REAL;
END_VAR

FUZZIFY влажность
        TERM низкая             := (30,1) (50,0);
        TERM средняя            := (30,0) (50,1) (70,1) (80,0);
        TERM высокая            := (70,0) (80,1);
END_FUZZIFY

DEFUZZIFY dT1
        TERM отрицательная      := -5;
        TERM нулевая            := 0;
        TERM положительная      := 5;
        METHOD: CoGS
        DEFAULT:= 0;
END_DEFUZZIFY

DEFUZZIFY dT2
          TERM отрицательная    := -3:
          TERM нулевая          := 0;
          TERM положительная    := 3;
          METHOD: CoGS
          DEFAULT:= 0;
END_DEFUZZIFY

RULEBLOCK логический вывод
AND: MIN;
ACCU: MAX;
RULE 1 IF влажность IS средняя AND цвет = коричневый THEN dT1 IS нулевая
                                                        AND dT2 IS нулевая;
RULE 2: IF влажность IS высокая                          THEN dT1 IS положительная;
RULE 3: IF влажность IS низкая                           THEN dT1 IS отрицательная;
RULE 4: IF влажность IS средняя AND цвет = светлый       THEN dT2 IS положительная;
RULE 5: IF влажность IS средняя AND цвет = темный        THEN dT2 IS отрицательная;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

Рисунок D.4 — Пример на FCL

Приложение Е
(справочное)

Условные обозначения, аббревиатуры и синонимы

Т а б л и ц а Е.1 — Условные обозначения и аббревиатуры

CoA	Центр площади	
CoG	Центр тяжести	
FB	Функциональный блок	
FBD	Диаграммы функциональных блоков	
FCL	Язык нечеткого управления	
IL	Перечень команд	
ИСО	Международная организация по стандартизации	
MAX	Операция максимизации	
MIN	Операция минимизации	
PROD	Операция умножения	
ST	Структурированный текст	
μ	Степень принадлежности	
ω	Весовой коэффициент	

Т а б л и ц а Е.2 — Синонимы

Результат	Следствие
Аккумуляция	Агрегирование результатов
Активизация	Композиция
Условие	Предпосылка
Центр тяжести	Центроид
Центр площади	Медиана

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
IEC 60050-351:1998	—	*
IEC 61131-3:2013	IDT	ГОСТ Р МЭК 61131-3—2016 «Контроллеры программируемые. Часть 3. Языки программирования»

*Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандарта:

- IDT — идентичный стандарт.

УДК 681.58:681.3 ОКС 35.240.50 25.040.40 IDT

Ключевые слова: программируемые контроллеры, нечеткое управление, нечеткая логика, язык нечеткого управления

БЗ 10—2017/155

Редактор *В.Н. Шмельков*
Технический редактор *В.Н. Прусакова*
Корректор *С.И. Фирсова*
Компьютерная верстка *Е.О. Асташина*

Сдано в набор 18.09.2017. Подписано в печать 09.10.2017. Формат 60×84 1/8. Гарнитура Ариал.
Усл. печ. л. 6,05. Уч.-изд. л. 5,62. Тираж 21 экз. Зак. 1905.
Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Издано и отпечатано во ФГУП «СТАНДАРТИНФОРМ», 123001 Москва, Гранатный пер., 4.

gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru