УДК 004.896

ПОДСИСТЕМА НЕЧЁТКОГО ВЫВОДА ДЛЯ ОПТИМИЗАТОРА БАЗ ЗНАНИЙ НА ТЕХНОЛОГИЯХ МЯГКИХ ВЫЧИСЛЕНИЙ

Сорокин Сергей Владимирович[1], Нефедов Никита Юрьевич[2], Решетников Андрей Геннадьевич[3], Ульянов Сергей Викторович[4]

[1]Кандидат физико-математических наук, доцент кафедры информационных технологий;
Тверской государственный университет;
г. Тверь, Садовый переулок 35;
e-mail: sergey@tversu.ru.

[2]Аспирант;
ГБОУ ВПО Международный университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19,
e-mail: nefnukem@gmail.com.

[3]Аспирант;
ГБОУ ВПО Международный университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19,
e-mail: reshetnikovag@pochta.ru.

[4] Доктор физико-математических наук, профессор;
ГБОУ ВПО Международный университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19,
e-mail: ulyanovsv@mail.ru.

В статье рассматривается архитектура системы нечёткого вывода, реализованная в рамках инструментария оптимизатора баз знаний на мягких вычислениях. Разработанная архитектура подразумевает вынесение интерфейса алгоритмов, заинтересованных в получении списка активных правил, в отдельный класс. Сам алгоритм нечёткого вывода реализован в виде шаблонного метода, примитивные операции которого реализованы в классах конкретных моделей нечёткого вывода и баз правил. При этом алгоритм конфигурируется объектом, которому необходимо направлять список активных правил. Описаны три режима работы подсистемы: нечёткий вывод, создание неполных баз правил и анализ правил. Представлены результаты работы оптимизатора баз знаний на примере создания ИСУ неустойчивым динамическим объектом. Проведенное тестирование показало, что спроектированная в оптимизаторе ИСУ, обладает большей робастностью, чем ИСУ, спроектированные с использованием других современных средств.

Ключевые слова: нечёткий вывод, программная архитектура, интеллектуальные системы управления.

FUZZY INFERENCE SUBSYSTEM FOR SOFT COMPUTING OPTIMIZER OF KNOWLEDGE BASES

Sorokin S.[1], Nefedov N.[2], Reshetnikov A.[3], Ulyanov S.[4]

[1]Candidate of Science in Physics and Mathematics,, Associate Professor;
Tver State University;
Tver, Sadovy pereulok 35;
e-mail:sergey@tversu.ru.

[2]PhD student;
Dubna International University of Nature, Society, and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: nefnukem@gmail.com.

[3]*PhD student;*
Dubna International University of Nature, Society, and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: reshetnikovag@pochta.ru.

[4]*Doctor of Science in Physics and Mathematics, professor;*
Dubna International University of Nature, Society, and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: reshetnikovag@pochta.ru.

Software architecture of fuzzy inference subsystem for soft computing optimizer is considered. Proposed architecture is based on segregation of interface of algorithms, interested in active rules, to separate interface. Fuzzy inference algorithm is implemented as a template method; its primitive operations are implemented in concrete realizations of fuzzy inference models and fuzzy rule bases. Algorithm is also configurable by the object which will receive a list of active rules. Three operational modes of subsystem are described: fuzzy inference, LBRW rule database creation and rule analysis. Performance of soft computing optimizer is demonstrated on the task of creating control system for instable dynamic object. This control system exhibited increasing robustness comparing to systems created with other state-of-the-art tools.

Keywords: fuzzy inference, software architecture, intelligent control systems.

Введение

Инженерные методы теории управления и технологии проектирования систем автоматического управления (САУ) сформировались в прошлом столетии. Были заложены, в частности, основы стохастического и адаптивного управления сложными динамическими системами в условиях информационной неопределённости.

На основе теории лингвистической аппроксимации и нечеткого логического вывода (Л. Заде и др.), были разработаны принципы моделирования и проектирования нечётких САУ [1]. Разработанные в рамках этого подхода интеллектуальные САУ (ИСУ) первого поколения использовали базы знаний (БЗ), создаваемые экспертами, что ограничивало класс решаемых задач.

Для создания ИСУ, способных работать в условиях резкого изменения параметров внешней среды или старением структуры ОУ, были разработаны ИСУ второго поколения, объединяющие генетические алгоритмы (ГА), нечеткие нейронные сети (ННС) и нечёткие регуляторы (НР) в рамках концепции глобальной интеллектуальной обратной связи.

Создание ИСУ является сложным процессом и требует наличия специализированного инструментария. Подобный инструментарий должен включать в себя средства для формирования и анализа баз знаний (БЗ), в том числе позволяющие автоматически формировать их структуру и параметры. При этом система должна поддерживать основные модели нечёткого вывода, иметь удобный пользовательский интерфейс и поддерживать интеграцию с другим программным обеспечением.

В данной статье рассматривается реализация подсистемы нечёткого вывода, входящая в оптимизатор БЗ SC Optimizer [2].

1. Модели нечёткого вывода

Рассмотрим основные модели нечёткого вывода, которые могут быть реализованы в ИСУ.

Правило нечёткого вывода в модели Мамдани имеет вид:

$$IF\ x_1\ is\ \mu_{j_l^1}^1(x_1)\ AND\ x_2\ is\ \mu_{j_l^2}^2(x_2)\ AND \ldots AND\ x_n\ is\ \mu_{j_l^n}^n(x_n)\ THEN\ y\ is\ \mu_{j_l^{n+1}}^{n+1},$$

где l – номер правила; $x_1,\ldots,x_n$ – входной вектор; μ_i^k – i-е терм-множество k-ой входной переменной, $k \in \{1,\ldots,n\}, i \in \{1,\ldots,m_k\}$; μ_i^{n+1} – i-е терм-множество выходной переменной; j_l^k – индекс,

показывающий какое из терм-множеств k-ой переменной используется в правиле l. Результат нечёткого вывода в модели Мамдани определяется следующим образом:

$$F(x_1,\dots x_n)=\frac{\sum_{l=1}^{M}\bar{y}_l\prod_{k=1}^{n}\mu_{j_l^k}^{k}(x_k)}{\sum_{l=1}^{M}\prod_{k=1}^{n}\mu_{j_l^k}^{k}(x_k)},$$

где $\bar{y}_l$ – модальное значение нечёткого множества $\mu_{j_l^{n+1}}^{n+1}$; П – используемая *t*-норма, M – число правил.

В модели Сугено правила имеют вид:

$$IF\ x_1\ is\ \mu_{j_l^1}^{1}(x_1)\ AND\ x_2\ is\ \mu_{j_l^2}^{2}(x_2)\ AND\dots AND\ x_n\ is\ \mu_{j_l^n}^{n}(x_n)\ THEN\ y\ is\ f^l(x_1,\dots,x_n),$$

где f^l – некоторая функция входного вектора. Результат нечёткого вывода в этой модели определяется по формуле:

$$F(x_1,\dots x_n)=\frac{\sum_{l=1}^{M}f^l(x_1,\dots,x_n)\prod_{k=1}^{n}\mu_{j_l^k}^{k}(x_k)}{\sum_{l=1}^{M}\prod_{k=1}^{n}\mu_{j_l^k}^{k}(x_k)}.$$

Как правило, при использовании модели Сугено, в качестве функций f рассматривают полиномы. Так, в модели Сугено 0-го порядка эта функция имеет вид: $f(x_1,\dots,x_n)=a_0$, а в модели Сугено 1-го порядка $f(x_1,\dots,x_n)=\sum_{k=1}^{n}a_kx_k+a_0$.

В реальных системах нечёткого вывода допускается наличие нескольких выходных переменных, значение каждой из которых определяется независимо друг от друга. Отметим следующие особенности, которые можно наблюдать на примере этих моделей:

- Вид предпосылки правила не зависит от модели нечёткого вывода и определяется только индексами выбранных для данного правила терм-множеств. Множество всех правил образуется декартовым произведением терм-множеств всех переменных и число правил $M=\prod_{k=1}^{n}m_k$.
- Вид заключения правила зависит от используемой модели нечёткого вывода, при этом некоторые модели используют терм-множества для выходных переменных, а некоторые — не используют.
- Для расчёта результата нечёткого вывода всегда необходимо вычислять значения $\prod_{k=1}^{n}\mu_{j_l^k}^{k}(x_k)$, которые будем называть уровнем активации правила l. При этом, правила с нулевым уровнем активации не оказывают влияния на результат.

Вычисление результата нечёткого вывода может быть организовано, как процесс накопления частичных результатов, вычисляемых для каждого активированного правила.

2. Требования к системе нечёткого вывода в рамках оптимизатора БЗ

Реализуемая в рамках инструментария БЗ система нечёткого вывода должна, в первую очередь, обеспечить возможность создания, хранения и расчётов с использованием БЗ, основанных на различных моделях нечёткого вывода. При этом желательно обеспечить возможность максимально простого расширения системы в случае добавления новых моделей.

Однако, в рамках оптимизатора БЗ, следует обеспечить и функционирование других алгоритмов, которым необходимо иметь доступ к промежуточным результатам вычислений нечёткого вывода. Такими алгоритмами являются, в частности:

- алгоритм выбора наиболее значимых правил (LBRW);

- алгоритм сокращения обучающего сигнала;
- различные алгоритмы анализа качества БЗ.

Эти алгоритмы во время работы используют списки правил, активируемых теми или иными строками обучающего сигнала. Данная операция является также и основой вычисления нечёткого вывода, поэтому целесообразно использовать такую архитектуру, которая бы позволила использовать одну реализацию алгоритма поиска активных правил в различных целях. Это позволит сократить дублирование кода в ядре системы и упростит её поддержку.

3. Архитектура система нечёткого вывода в SCOptimizer

На рис. 1 показан фрагмент диаграммы классов SC Optimizer, отражающий существенные для системы нечёткого вывода моменты.

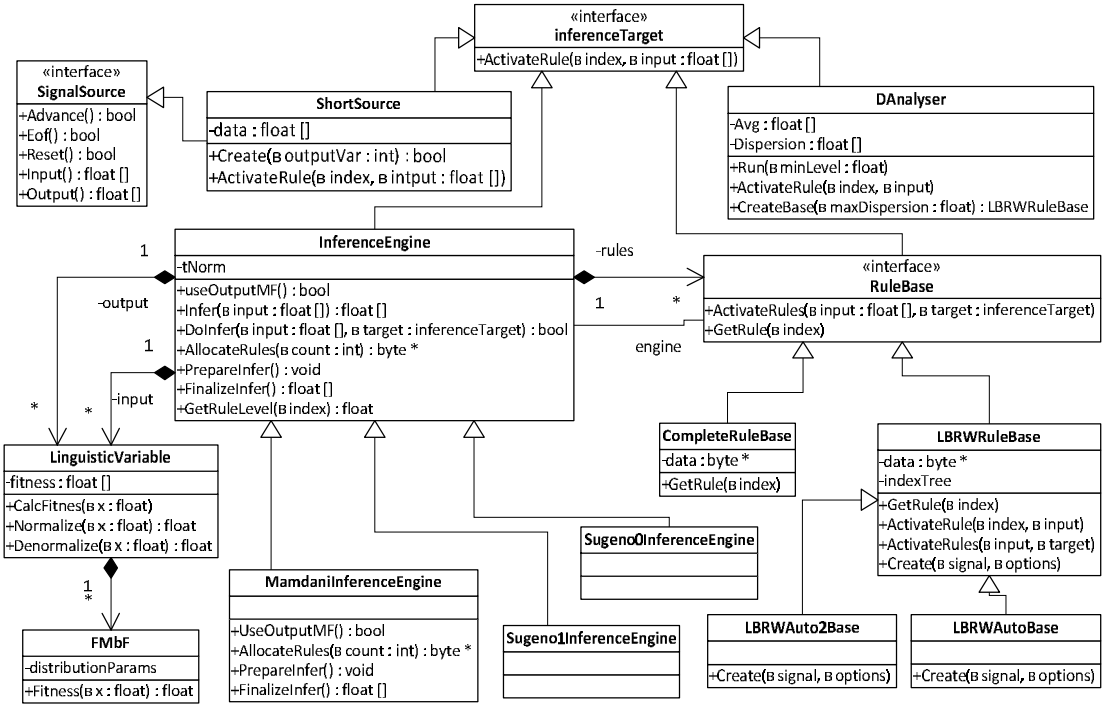

Рис. 1. Фрагмент диаграммы классов SC Optimizer

Рассмотрим классы, приведённые на этой диаграмме.

Класс InferenceTarget объявляет интерфейс объектов, заинтересованных в получении списка активных правил. Правила передаются этому объекту по одному, через функцию ActivateRule.

Класс InferenceEngine является базовым классом для систем нечёткого вывода. Он содержит ряд общих для разных моделей нечёткого вывода функций и наборов данных. В нём также объявляются виртуальные функции, переопределяемые в классах-наследниках, представляющих различные конкретные модели нечёткого вывода: MamdaniInferenceEngine, Sugeno0InferenceEngine и Sugeno1InferenceEngine. Так, например, функция UseOutputMF() позволяет определить используются ли данной моделью терм-множества для выходных переменных.

InferenceEngine также включает массивы input и output лингвистических переменных, представленных классом LinguisticVar. Лингвистическая переменная отвечает за нормализацию и денормализацию сигналов, а также за вычисление уровня соответствия входного сигнала распределениям терм-множеств, который сохраняется в массиве fitness[] и в дальнейшем используется для быстрого вычисления уровня активации правил вывода.

Терм-множества представлены классом FMbF, который отвечает за хранение параметров распределений и вычисление уровня принадлежности для заданного входного значения.

Ещё одна иерархия классов связана с БЗ. Хранение данных правил и их извлечение по указанному индексу (соответствующему предпосылке правила) обеспечивается классами, унаследованными от RuleBase. CompleteRuleBase предоставляет полную базу, хранящую все возможные правила. LBRWRuleBase даёт возможность использовать частичные базы, хранящие наиболее существенные правила. Подклассы LBRWRuleBase определяют различные версии алгоритмов отбора правил, переопределяя функцию Create().

Так как формат и объем данных правой части правил существенно различается для разных моделей нечёткого вывода, то непосредственное выделение памяти осуществляется с помощью фабричной функции [3] AllocateRules(), объявленной в классе InferenceEngine и определённой в его наследниках.

Класс ShortSource реализует алгоритм сокращения обучающего сигнала. Он реализует интерфейс с InferenceTarget, а также интерфейс SignalSource, определяющий операции доступа к обучающему сигналу. Создание сокращённой версии обучающего сигнала производится функцией Create(). Сокращённый сигнал создаётся индивидуально для каждого выхода модели.

Класс DAnalyser позволяет проанализировать значения дисперсии ожидаемых выходных значений для правил модели. Функция Run() осуществляет вычисление дисперсии. Функция CreateBase() создаёт базу, в которую входят правила, отобранные согласно заданного критерию.

4. Алгоритм нечёткого вывода

Работа алгоритма нечёткого вывода показана на рис. 2.

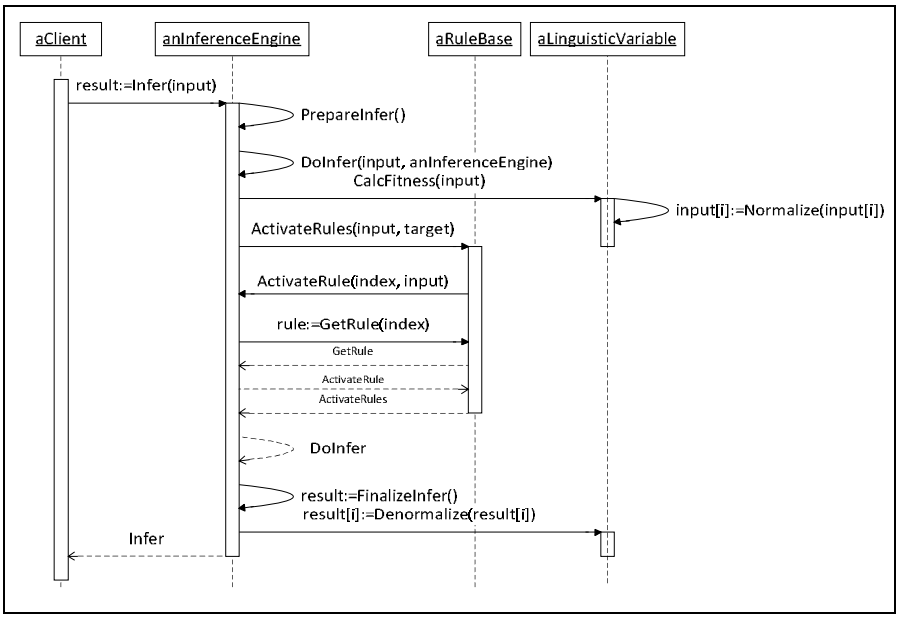

Рис. 2. Схема работы алгоритма нечёткого вывода.

Процесс начинается с вызова функции Infer() класса InferenceEngine, которой передаётся вектор входных значений, и состоит из трех стадий.

Первая стадия включает подготовку модели нечёткого вывода к началу вычислений, и обеспечивается вызовом операции PrepareInfer(), определённой в дочерних классах.

Второй этап реализован в функции DoInfer. В первую очередь она вычисляет лингвистическое представление входного сигнала, используя функцию CalcFitness входных переменных. Затем вызывается функция ActivateRules(), объявленная в классе RuleBase, которая перечисляет все активные правила. Эта функция может переопределяться в классах-наследниках RuleBase для реализации более эффективных вариантов перебора правил в зависимости от используемых структур хранения.

Так как в качестве цели для перечисления правил передаётся указатель на саму систему нечёткого вывода, то информация об активных правилах передаётся конкретной модели нечёткого вывода,

которая реализует функцию ActivateRule() и накапливает результаты вычислений в своих внутренних переменных.

Доступ к правой части правила осуществляется с помощью вызова функции GetRule() БЗ.

В завершение, управление возвращается в функцию Infer(), которая вызывает операцию FinalizeInfer(), реализованную в конкретной модели нечёткого вывода, которая проводит необходимые итоговые вычисления и возвращает результат.

5. Алгоритм отбора наиболее значимых правил

При использовании другими алгоритмами, взаимодействие компонентов происходит несколько другим способом. Например, на рис. 3 показана работа системы при исполнении алгоритма отбора наиболее значимых правил LBRW.

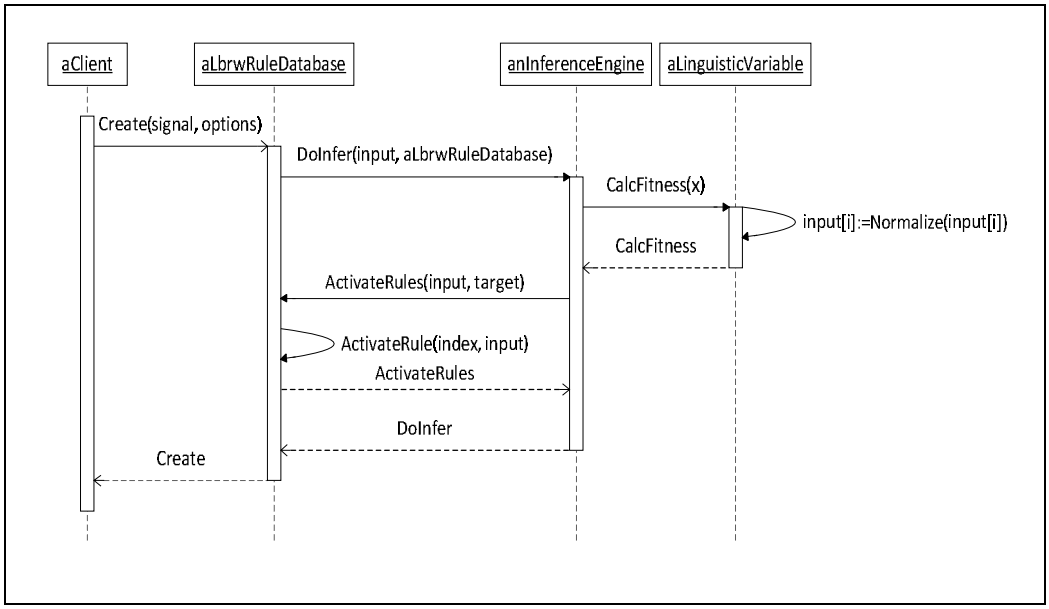

Рис. 3. Схема работы системы нечёткого вывода с алгоритмом LBRW

Алгоритм определён в функции Create() класса LBRWRuleBase или его наследниках.

Во время работы алгоритма просматривается весь набор обучающих данных, и для каждой строки вызывается функция DoInfer(). Однако, целью перечисления правил в данном случае является не потомок InferenceEngine, а база правил LBRWRuleBase. Поэтому информация об активных правилах передаётся базе правил, которая накапливает необходимую ей информацию. В дальнейшем эта информация используется функцией Create() для отбора правил, удовлетворяющих тем или иным критериям.

6. Алгоритм сокращения обучающего сигнала

При работе алгоритма сокращения обучающего сигнала реализуется третий режим работы нечёткого вывода (рис. 4).

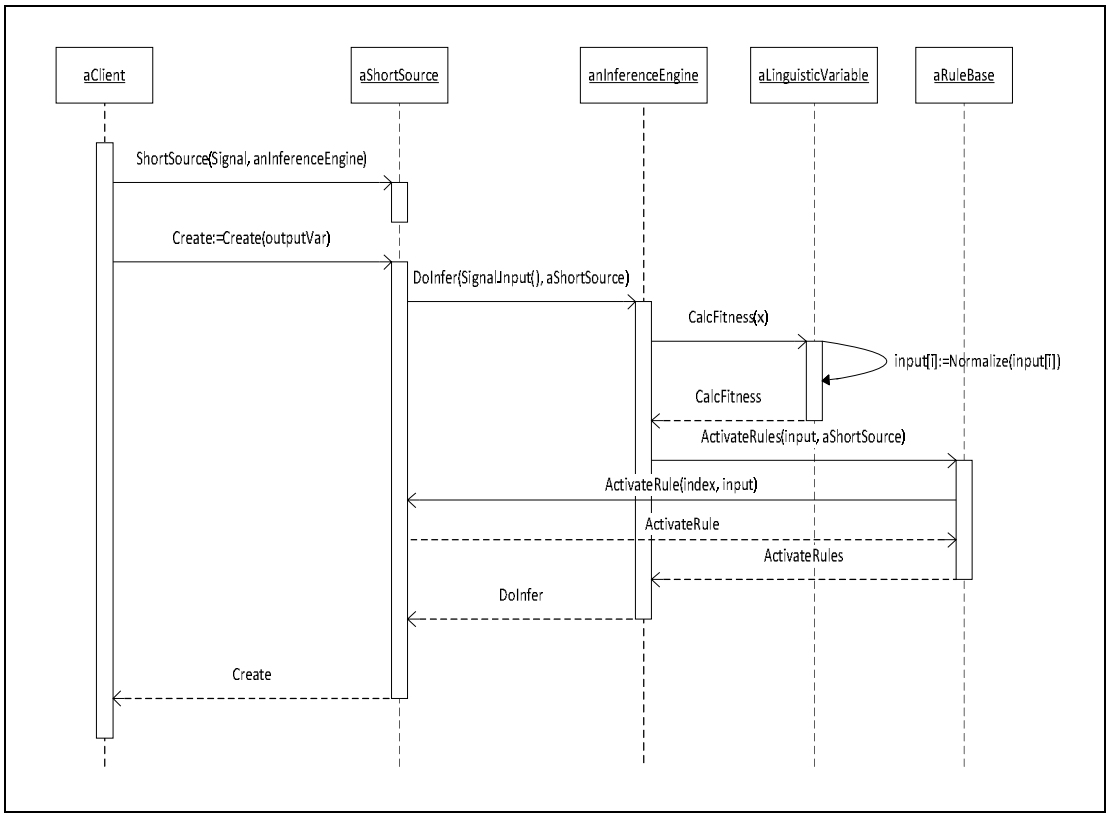

Рис. 4. Схема работы нечёткого вывода с алгоритмом сокращения обучающего сигнала

В данном случае, в процессе нечёткого вывода задействованы три класса: класс InferenceEngine управляет процессом нечёткого вывода, одна из баз правил (потомок RuleBase) перечисляет активированные правила, а обработку правил ведёт класс ShortSource.

Для создания сокращенного набора обучающих данных клиент создаёт объект aShortSource класса ShortSource, инициализируя его полным набором обучающих данных Signal и текущей системой нечёткого вывода anInferenceEngine.

Затем вызывается функция Create() объекта aShortSource.

Эта функция вызывает нечёткий вывод для всех строк полного обучающего сигнала, передавая ссылку на себя в качестве цели вывода. Таким образом, все активированные правила передаются функции ActivateRule объекта aShortSource. Эта функция вычисляет и запоминает лингвистическое представление заданного выхода обучающего сигнала.

После окончания нечёткого вывода функция Create() осуществляет фильтрацию обучающего сигнала, в процессе которой удаляются строки с дублирующим лингвистическим представлением. Отфильтрованный сигнал сохраняется в объекте aShortSource и используется для оптимизации базы правил для соответствующего выхода.

Для создания сигнала для другого выхода повторяется запуск функции Create().

7. Алгоритм анализа дисперсии

Режим работы системы с алгоритмом анализа дисперсии в целом совпадает с режимом, в котором работает алгоритм сокращения обучающего сигнала. Но в данном случае процедура нечёткого вывода повторяется два раза (рис. 5): первый раз для вычисления математического ожидания, а второй раз – дисперсии выходных значений каждого правила согласно обучающему сигналу.

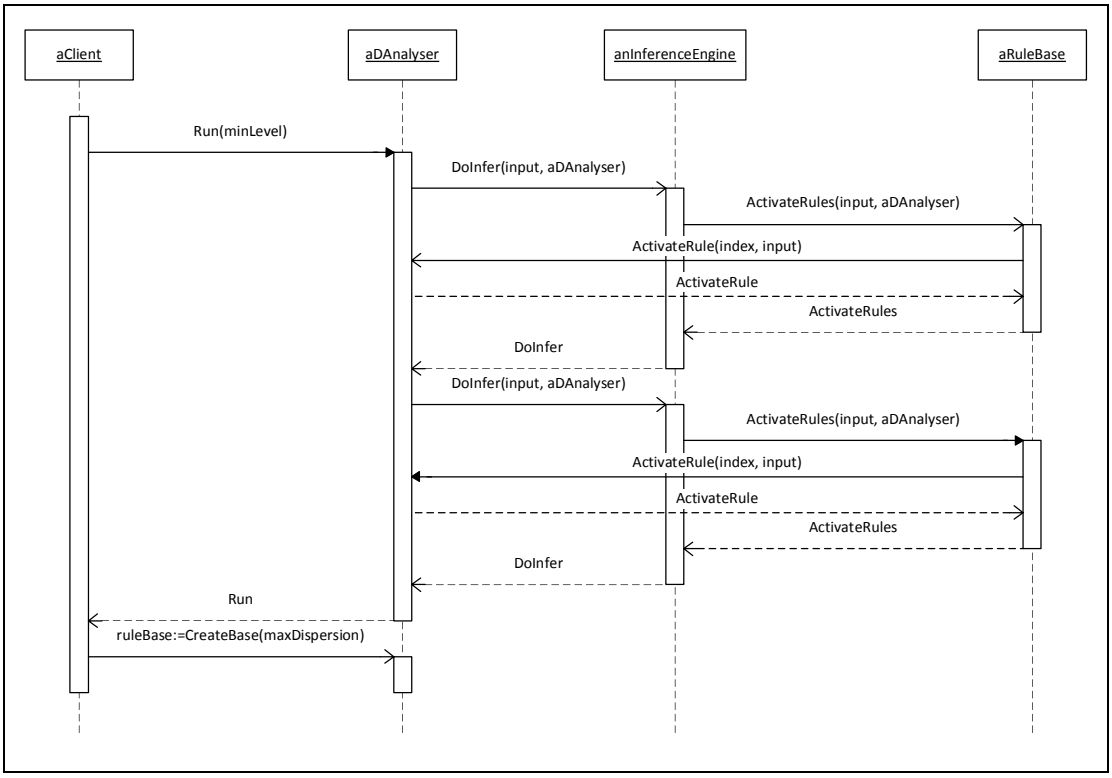

Рис. 5. Схема работы нечёткого вывода с алгоритмом анализа дисперсии

Работа анализатора начинается с вызова функции Run(). Она дважды проводит цикл нечёткого вывода, устанавливая анализатор в качестве цели. При первом проходе анализатор аккумулирует данные для вычисления среднего значения требуемого выхода, при втором – вычисляет дисперсию. Правила, с уровнем активации ниже заданного порогового значения minLevel игнорируются.

После выполнения функции Create() система предоставляет пользователю возможность выбрать пороговое значение для отбора правил. Процедура отбора реализована в функции CreateBase(), которая создаёт неполную базу и заполняет её правилами, дисперсия выходного значения для которых превосходит значение maxDispersion.

8. Применение оптимизатора баз знаний для создания ИСУ неустойчивым динамическим объектом

В качестве примера применения оптимизатора БЗ рассмотрим задачу управления неустойчивой динамической системой «движущаяся каретка – перевернутый маятник» (рис. 6).

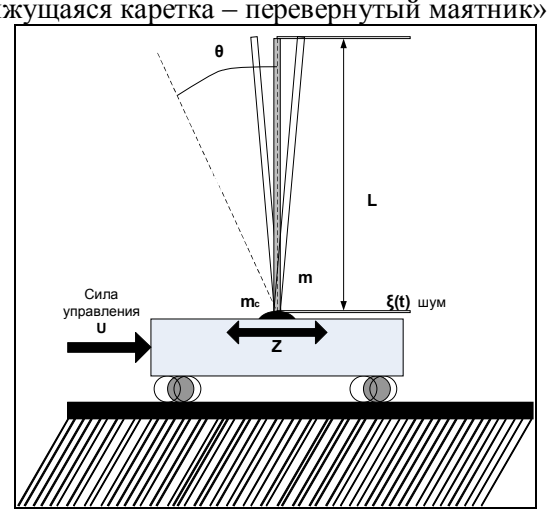

Рис. 6. Динамическая система «движущаяся каретка – перевернутый маятник»

Динамическое поведение этой системы при воздействии силы управления *u* описывается системой дифференциальных уравнений второго порядка:

$$\begin{cases} \ddot{\theta} = \dfrac{g\sin\theta + \cos\theta\left(\dfrac{u+\xi(t)+a_1\dot{z}+a_2 z - ml\dot{\theta}^2\sin\theta}{m_c+m}\right) - k\dot{\theta}}{l\left(\dfrac{4}{3}-\dfrac{m\cos^2\theta}{m_c+m}\right)}; \\ \ddot{z} = \dfrac{u+\xi(t)-a_1\dot{z}-a_2 z + ml(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta)}{m_c+m}. \end{cases} \quad (1)$$

В системе уравнений (1) z и θ – обобщенные координаты; g – ускорение свободного падения ($9.8\ m/\sec^2$), m_c – масса оси вращения (аналог массы подвижной каретки), m – масса перевернутого маятника (называемого «шест»), l – половина эффективной длины маятника, k и a_1 коэффициенты трения о поверхность горизонтального перемещения вдоль оси z и в оси вращения θ, соответственно, a_2 – сила упругости, препятствующая перемещению тележки, $\xi(t)$ – внешний стохастический шум, а u – сила управления.

Без управления маятник не может быть удержан в вертикальном положении, $\theta = 0$ [4].

Цель управления – сбалансировать положение маятника в условиях существенных ограничений на скорость и положение тележки, а также в условиях ограниченной силы управления. Рассмотрим модель динамической системы с параметрами, представленными в таблице 1, и начальными условиями $\left[\theta_0\dot{\theta}_0\right] = [100.1](grad); \left[\zeta_0\dot{\zeta}_0\right] = [00]$.

Таблица 1. Параметры системы «движущаяся каретка — перевернутый маятник»

m_c [кг]	m [кг]	l [м]	Коэфф. трения в θ, k	Коэфф. трения в z, a_2	Сила упругости, a_1
1.0	0.1	0.5	0.4	0.1	5.0

Сравним базу, спроектированную в оптимизаторе БЗ, с базой, спроектированной в ANFIS (встроенный инструментарий Матлаб) и ПИД-регулятором (рис. 7).

Рис. 7. Графики динамики движения маятника, и интегральной ошибка управления

Результаты моделирования показали, что только интеллектуальный регулятор, спроектированный с помощью оптимизатора БЗ, является робастным и выполнил поставленную задачу управления.

Заключение

Особенностью архитектуры системы нечёткого вывода в рамках SC Optimizer является реализация алгоритма нечёткого вывода в виде распределённого шаблонного метода [3]. При этом ответственность за выполнение операций распределяется следующим образом:

- общая схема работы алгоритма определяется базовым классом моделей нечёткого вывода InferenceEngine;
- функция поиска и перечисления активных правил делегирована классам, отвечающими за хранение БЗ;
- вычисления, необходимые для проведения нечёткого вывода, реализованы в конкретных классах-реализациях моделей нечёткого вывода.

Возможность конфигурирования алгоритма нечёткого вывода классом-приёмником списка активированных правил даёт возможность использовать реализацию нечёткого вывода в рамках других алгоритмов, выполняющих различные задачи в процессе автоматизированного построения ИСУ.

В SC Optimizer система нечёткого вывода может работать в трёх режимах, информация о которых обобщена в таблице 2.

Таблица 2. Распределение ролей в разных режимах работы системы нечёткого вывода

	Нечёткий вывод	**Алгоритм LBRW**	**Алгоритмы анализа**
Координация	InferenceEngine	InferenceEngine	InferenceEngine
Перечисление правил	CompleteRuleBase или LBRWRuleBase	CompleteRuleBase или LBRWRuleBase	CompleteRuleBase или LBRWRuleBase
Обработка правил	Наследники InferenceEngine	LBRWRuleBase или его наследники	Классы реализации алгоритма анализа

Учитывая число алгоритмов анализа и соответствующих классов-наследников, общее число вариантов, в которых может быть сконфигурирована и использована система нечёткого вывода во время выполнения программы равно 16.

Реализованная в рамках SC Optimizer архитектура системы нечёткого вывода позволила решить ряд инженерных задач, возникших при разработке программного комплекса. Высокая эффективность проектных решений позволила создать систему проектирования ИСУ, превосходящую аналогичный инструментарий МатЛаб [1], что подтверждается результатами тестирования.

Список литературы

1. Ульянов С.В., Литвинцева Л.В., Добрынин В.Н., Мишин А.А., Интеллектуальное робастное управление: Технологии мягких вычислений. М.: – ВНИИгеосистем. 2011. – С. 408.
2. Сорокин С.В., Литвинцева Л.В., Ульянов С.В. Технология интеллектуальных мягких вычислений в проектировании робастных нечётких систем управления: оптимизатор баз знаний // Нечёткие системы и мягкие вычисления. – Тверь: ТвГУ, 2008. – Том 3. – №1.
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2008. – С. 366.
4. Ульянов С.В., Мишин А.А., Миногин А.А. и др. Информационная технология проектирования робастных баз знаний нечетких регуляторов. Ч. III: квантовый нечёткий вывод и квантовая информация. // Системный анализ в науке и образовании: сетевое научное издание. – 2010. – №3. – [Электронный ресурс]. URL: http:/www.sanse.ru/archive/17. – 0421000111\0029.