УДК 007:519.816

В. В. Девятков, И. М. Сидякин

МУЛЬТИАГЕНТНАЯ СИСТЕМА АНАЛИЗА ТЕЛЕМЕТРИЧЕСКОЙ ИНФОРМАЦИИ

Рассмотрены теоретические и прикладные аспекты создания мультиагентной системы анализа телеметрической информации, включающие задачи анализа, принципы формирования и использования онтологий для ответа на запросы, а также мультиагентная архитектура программной системы, позволяющей осуществлять распределенный анализ телеметрической информации. Приведены иллюстративные примеры.

В последние годы все бóльшую популярность получают датчиковые сети [1] в самых различных областях применения, например, в городском хозяйстве, экологии, строительстве, интеллектуальных зданиях и т. п. В развитых странах ведутся интенсивные исследования датчиковых сетей [2]. Беспроводные датчики становятся все дешевле, что позволяет расставлять их в самых экзотических местах и в большом количестве. Число датчиков может достигать нескольких тысяч [3]. Датчики становятся интеллектуальнее. Области применения таких датчиков расширяются от таких традиционных, как анализ освещенности, температуры, движения, контроль доступа на объекты, до весьма экзотических, например контроль состояния мобильного телефона, автомобиля, личного кошелька с деньгами, наблюдения за детьми на детской площадке или за поведением собак. Обработка распределенной информации с удаленных датчиков может быть централизованной, децентрализованной или комбинированной. При большом числе датчиков централизованная обработка становится слишком трудоемкой, поэтому предпочтительнее децентрализованная и комбинированная обработка информации.

Данные, снимаемые с датчиков, образуют поток данных, называемых обычно в русскоязычных источниках телеметрическими. Такие применения датчиковых сетей, как мониторинг, слежение или наблюдение, требуют постоянного анализа непрерывно поступающих телеметрических данных, с целью ответить на запросы относительно ситуаций, возникающих в среде, охваченной датчиковой сетью.

Приведем примеры таких запросов.

1. Некоторая территория городской застройки охвачена датчиковой сетью, предназначенной для измерения уровня дождевых осадков. Среди датчиков сети имеются датчики, измеряющие скорость ветра (скорость движения дождевых облаков), направление ветра и степень загрязнения. Запросы пользователей касаются области загрязнения, которую создают дождевые осадки. Пользователя интересует, какая область в пределах территории, охваченной датчиками, была загрязнена за определенное время.

2. Движущиеся объекты фиксируются множеством видеокамер. Каждый объект однозначно идентифицируется различными видеокамерами. Запросы пользователей касаются скорости движения отдельных объектов за определенное время или разницы скоростей одних объектов по отношению к другим.

Анализ телеметрической информации осуществляется агентами. Агенты могут находиться в различных узлах сети и общаться друг с другом: собирать, передавать и обрабатывать телеметрическую информацию. Они также обслуживают различных потребителей, которые обращаются к ним с запросами с целью получить сведения о среде, где расположены датчики. Иными словами, запрос пользователя требует ответа, характеризующего ситуацию, в которой находится среда. Под ситуацией здесь понимается некоторое выводимое (устанавливаемое) в процессе анализа телеметрической информации свойство среды в определенное время или в определенном месте. Знания, на основе которых формируются ответы на запросы, называются онтологией [4]. Далее рассматриваются структура телеметрической информации (ТМИ), поступающей на вход мультиагентной системы анализа, задачи анализа ТМИ, принципы формирования и использования онтологий для ответа на запросы, касающиеся ТМИ, иллюстративные примеры, мультиагентная архитектура программной системы, позволяющей осуществлять распределенный анализ ТМИ на основе разработанных принципов.

Постановка задачи анализа телеметрической информации. Введем несколько понятий. Показания (данные), снимаемые с i-го датчика y_i, $i = 1, \ldots, m$, в некоторый момент времени t назовем *отсчетом* $y_i(t)$. Последовательность n+1 отсчетов $Y_i[t_0,t_n] = \{y_i(t_0), y_i(t_1), y_i(t_2), \ldots, y_i(t_n)\}$, снимаемых с одного датчика в течение нескольких подряд идущих моментов времени $t_0, t_1, t_2, \ldots, t_n$ (в течение временного интервала $[t_0,t_n]$) назовем *сигналом*. Множество отсчетов $K(t) = \{y_1(t), y_2(t),\ldots,y_m(t)\}$, снимаемых одновременно с m датчиков в момент времени t назовем *кадром*. Последовательность кадров $K(t_0), K(t_1),\ldots,K(t_n)$, снимаемых с m датчиков в течение нескольких подряд идущих моментов времени $t_0, t_1, t_2, \ldots, t_n$ (в течение

временного интервала $[t_0,t_n]$) назовем *потоком кадров*. Совокупность n+1 сигналов $Y_1[t_0, t_n]$, $Y_2[t_0,t_n]$,. . . , $Y_n[t_0,t_n]$, относящихся к одному временному интервалу $[t_0,t_n]$ назовем *потоком сигналов*. В реальных системах, вследствие необходимости экономить время и память, выполнять анализ в реальном масштабе времени, удовлетворять ограниченной пропускной способности каналов передачи данных, число отсчетов в кадрах может быть меньше числа опрашиваемых датчиков. В этом случае приходится иметь дело не с одним потоком кадров и сигналов, а с несколькими. Для простоты рассмотрения основных принципов, излагаемых в настоящей статье, положим, что мы имеем дело с одним потоком кадров и соответствующим ему одним потоком сигналов.

Сопоставим каждый отсчет $y_j(t_i)$ одного и того же сигнала с состоянием $b_j(t_i)$ конечного автомата M_j. Введем функцию выходов φ конечного автомата M_j

$$\varphi(b_j(t_i)) = y_j(t_i).$$

Введем также функцию переходов автомата M_j

$$f(b_j(t_i), (t_{i+1})) = b_j(t_{i+1}).$$

Таким образом, каждый отсчет является значением функции выхода $y_j(t) = \varphi(b_j(t))$ автомата M_j, каждый сигнал является последовательностью значений функций выхода $\boldsymbol{y}_j(t) = (y_j(t_0), y_j(t_1), \ldots, y_j(t_n))$ одного и того же автомата M_j, каждый кадр является набором $\boldsymbol{y}(t) = (y_1(t), \ldots, y_m(t))$ значений функций выхода различных автоматов $M_1, M_2, \ldots, M_m$, поток сигналов представляется набором последовательностей значений функций выхода $\boldsymbol{y}_1(t), \boldsymbol{y}_2(t), \ldots, \boldsymbol{y}_m(t)$ соответственно конечных автоматов $M_1, M_2, \ldots, M_m$, поток кадров — последовательностью кадров $\boldsymbol{y}(t_0), \boldsymbol{y}(t_1), \ldots, \boldsymbol{y}(t_n)$. Поскольку значение функции выхода $y_j(t)$ однозначно определяется функцией выхода $y_j(t) = \varphi(b_j(t))$, то наряду с введенными обозначениями будем также использовать следующие:

последовательность состояний $\boldsymbol{b}_j(t) = (b_j(t_0), b_j(t_1), \ldots, b_j(t_n))$ автомата M_j, соответствующую сигналу;

макросостояние $\boldsymbol{b}(t) = (b_1(t), \ldots, b_m(t))$ автоматов $M_1, M_2, \ldots$ $\ldots, M_m$, соответствующее кадру;

множество последовательностей состояний $\boldsymbol{b}_1(t), \boldsymbol{b}_2(t), \ldots, \boldsymbol{b}_m(t)$ автоматов $M_1, M_2, \ldots, M_m$, соответствующих потоку сигналов;

последовательность макросостояний $\boldsymbol{b}(t_0), \boldsymbol{b}(t_1), \ldots, \boldsymbol{b}(t_n)$ автоматов $M_1, M_2, \ldots, M_m$, соответствующую потоку кадров.

Нас интересует решение следующих задач анализа сигналов, кадров, потоков сигналов и кадров: распознавание потоков, выявление их характерных свойств; проверка свойств правильного поведения агентов, анализирующих потоки; эквивалентные преобразования потоков.

Исходными данными для решения этих задач являются потоки сигналов и кадров (отдельный сигнал или кадр рассматривается как частный случай соответствующих потоков), представляемые автоматами $M_1, M_2, \ldots, M_m$, потоки и агенты, обрабатывающие эти потоки. В дальнейшем, говоря о потоках, будем иметь в виду также представляющие их автоматы, не повторяя это каждый раз. Задачами агентов являются следующие.

Распознавание потоков осуществляется путем сравнения по определенным критериям потоков с эталонными потоками, которые заранее сформированы.

Выявление характерных свойств потоков осуществляется путем формального вывода (доказательства) наличия определенных отношений на потоках. В настоящей статье не рассматриваются процедуры вывода, а приводятся только принципы формирования онтологий и формулировки запросов на выявление характерных свойств потоков.

Проверка свойств правильного поведения агентов. При формулировке свойств правильного поведения объектов используются те же принципы, что и при формулировке характерных свойств потоков. Каждый из агентов является процессом, имеющим дело с потоками и другими агентами. Процессы выполняются параллельно или квазипараллельно и должны удовлетворять определенным свойствам правильного поведения. Проверка свойств правильного поведения осуществляется путем формального доказательства наличия этих свойств. В настоящей статье приведены наиболее типичные формулировки свойств правильного поведения, образующих элементы онтологий.

Эквивалентные преобразования потоков состоят в минимизации, композиции и кодировании состояний автоматов, представляющих собой потоки $M_1, M_2, \ldots, M_m$. Эти преобразования не рассматриваются в настоящей статье. Основой этих преобразований может служить метод, изложенный в работе [4].

Распознавание потоков. Представим автомат M сигнала его графом переходов (рис. 1). Каждая вершина графа помечена символом b_i, $i = 0, 1, \ldots, 12$ (вершины обозначены кружками). Каждая пара соседних вершин b_i, b_{i+1}, $i = 0, 1, 2, \ldots, 11$, соединена дугой, направленной от вершины i к вершине $i + 1$. Дуги, направленные от вершины i к вершине $i + 1$, помечены символом t_i в алфавите $T = \{t_0, t_1, t_2, t_3, \ldots, t_{m-1}\}$. Если выписать обозначения

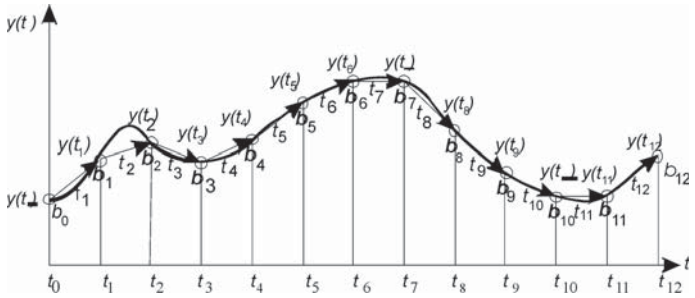

Рис. 1. Граф переходов сигнала

всех дуг слева направо, то получим последовательность символов $t_0t_1t_2t_3t_4t_5t_6t_7t_8t_9t_{10}t_{11}$. Эта последовательность может рассматриваться как слово или предложение некоторого языка $L = L(G)$, порождаемого автоматной грамматикой $G = \{V, T, P, S = b_0\}$, $V = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}\}$, $T=\{t_0,t_1,t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}\}$, $P = \{b_0 \to t_1b_1,\ b_1 \to t_2b_2,\ b_2 \to t_3b_3,\ b_3 \to t_4b_4,\ b_4 \to t_5b_5,\ b_5 \to t_6b_6,\ b_6 \to t_7b_7,\ b_7 \to t_8b_8,\ b_8 \to t_9b_9,\ b_9 \to t_{10}b_{10},\ b_{10} \to t_{11}b_{11},\ b_{11} \to t_{12}b_{12}\}$.

Для того чтобы перейти к представлению текущих и общих свойств сигналов, построим по четкой грамматике G нечеткую грамматику, базируясь на следующих принципах.

Каждой дуге графа соответствуют две инцидентные вершины b_i и $b_{i+1} = b_j$ (рис. 2). Координатами вершины b_i являются t_i и $\varphi(b_i(t_i)) = = y_i(t_i)$, а координатами вершины $b_j - t_j$ и $\varphi(b_j(t_j)) = y_j(t_j)$. Допустим, что координаты обеих вершин вследствие погрешности измерений или каких-либо других причин могут отклоняться по оси Y в пределах областей, приведенных на рис. 2 с шагами соответственно δ_y. Такое допущение означает, что вместо одной вершины b_i с координа-

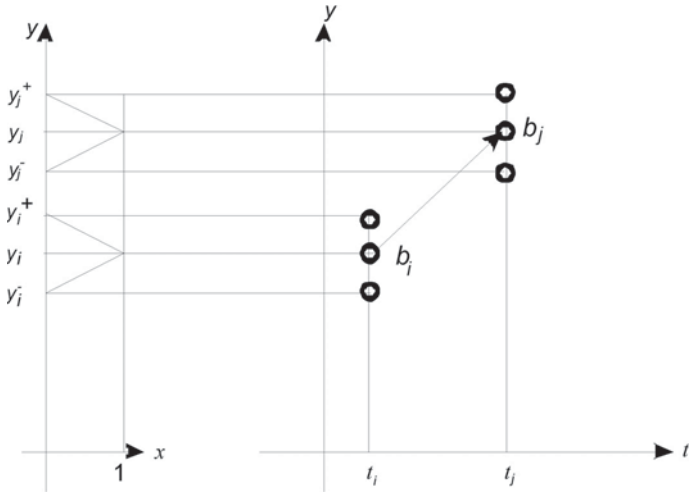

Рис. 2. Функции принадлежности вершин синтаксического графа

тами t_i, y_i будем иметь множество вершин $b_{ri} \in B(b_i)$ с координатами, изменяющимися в пределах области $y_i^+ - y_i^-$, причем мощность множества $B(b_i)$ будет

$$v = \frac{(y_i^+ - y_i^-)}{\delta_y}.$$

Вместо одной вершины b_j с координатами t_j, y_j при тех же отклонениях значений параметров и тех же значениях δ_y будем иметь множество вершин $b_{sj} \in B(b_j)$ с координатами, изменяющимися в пределах области $y_j^+ - y_j^-$. В результате вместо одной дуги t_j, ведущей из вершины b_i в вершину b_j, будем иметь множество дуг $t_{lj} \in T(t_j) = \{(b_{ri}, b_{sj}) | b_{ri} \in B(b_i),\ b_{sj} \in B(b_j)\}$ и соединяющих каждую вершину множества $B(b_i)$ с каждой вершиной множества $B(b_j)$.

Зададим две треугольные функции принадлежностей $\mu_i(y)$, $\mu_j(y)$, которые заданы на универсуме Y тройками значений соответственно в точках $\{y_i^-, y_i, y_i^+\}$, $\{y_j^-, y_j, y_j^+\}$ (см. рис. 2). Каждая из этих функций определяется следующим выражением:

$$\mu(y) = \begin{cases} \dfrac{y}{y_k - y_k^-} - \dfrac{y_k^-}{y_k - y_k^-}, & \text{если } y_k^- \leqslant y \leqslant y_k, \\ \dfrac{-y}{y_k^+ - y_k} + \dfrac{y_k^+}{y_k^+ - y_k}, & \text{если } y_k \leqslant y \leqslant y_k^+. \end{cases}$$

Здесь $k = i, j$. Эти функции определяют меру близости координат вершин графа к "наилучшим координатам", которым соответствует значение функций принадлежности, равное 1.

Будем полагать, что функция принадлежности каждой дуги $t_{lj} \in T(t_j)$, инцидентной вершинам $b_{ri} \in B(b_i)$ и $b_{sj} \in B(b_j)$, определяется следующим образом:

$$\mu_{T(t_j)}(t_{lj}) = \min\{\mu_i(y_{ri}),\ \mu_j(y_{sj})\}.$$

Нечеткая грамматика $G_F = \{V, T_F, P_F, S_F\}$ получается из четкой грамматики $G = \{V, T, P, S\}$ следующим образом:

$$T_F = \bigcup_j T(t_j).$$

Единственный начальный нетерминальный символ четкой грамматики заменяется множеством начальных нетерминальных символов:

$$S_F = B(b_0).$$

Множество правил P_F нечеткой грамматики G_F будет следующим:

$$P_F = \{b_{ri} \to t_{lj}b_{sj},\ \mu(b)_{ri} \to t_{lj}b_{sj} = \mu_{T(t_j)}(t_{lj}),\ t_{lj} \in T(t_j),\ j = 1, \ldots, n,\ l \leqslant v\}.$$

Нечеткая автоматная грамматика G_F порождает нечеткий язык $\{L(G_F),\ R_{L(G_F)}\}$:

$$L(G_F) = \{l^* | l^* = t_{l1}, t_{l2}, \ldots, t_{ln}, t_{lj} \in T(t_j), \quad l \leqslant v\},$$

$$R_{L(G_F)} = \{\mu_{L(G_F)}(l^*)/l^* |\, l^* = \\ = t_{l1}, t_{l2}, \ldots, t_{ln}, t_{lj} \in T(t_j),\ \mu_{L(G_F)}(l^*) = \min_{t_{lj} \in \{t_{l1}, \ldots, t_{l\,n}\}} \{\mu_{T(t_j)}(t_{lj})\}.$$

С учетом введенных понятий распознавание по эталонным сигналам может быть организовано следующим образом. Для каждого эталонного сигнала осуществляется построение эталонной нечеткой грамматики. Грамматик будет столько, сколько имеется эталонных сигналов. Если ограничиться синтаксическим уровнем распознавания по нечетким эталонным грамматикам, то процесс распознавания сигнала состоит в следующем.

1. Выбирается один из эталонных сигналов (точнее, соответствующая ему нечеткая эталонная грамматика).

2. Распознаваемый сигнал обрабатывается с теми же шагами дискретизации, что и выбранный эталонный.

3. Осуществляется синтаксический разбор распознаваемого сигнала с помощью нечеткой эталонной грамматики выбранного эталонного сигнала.

4. Если синтаксический разбор оказывается безуспешным, то выбирается следующий эталонный сигнал из числа нерассмотренных сигналов и все повторяется с п. 2.

5. Если синтаксический разбор оказался успешным, то вычисляется функция принадлежности разобранного распознаваемого сигнала и запоминается.

6. Если еще не все эталонные сигналы рассмотрены, то выбирается следующий эталонный сигнал из их числа и все повторяется с п. 2.

7. Если нерассмотренных эталонных сигналов не осталось, и не было ни одного успешного синтаксического разбора, то распознавание сигнала заканчивается неудачей (сигнал не был распознан).

8. Если были удачные синтаксические разборы, то распознавание закончилось успешно, и распознаваемый сигнал относится к классу того эталонного сигнала, при разборе которого с помощью грамматики функция принадлежности распознанного сигнала оказалась максимальной.

Известно [3], что правила любой грамматики легко представить правилами адекватного исчисления. Первоначальное построение грамматики является удобной формой хорошо формализуемого способа перехода от графического представления эталонных сигналов к описанию эталонных онтологий в нечетком исчислении. В случае автоматных грамматик легко организовать вывод на основе вычисления с помощью эталонных автоматов.

Свойства потоков. Характерные свойства могут выявляться для сигналов, потоков сигналов, кадров и потоков кадров. Рассмотрим наиболее типичные свойства для потока сигналов, представляемого множеством автоматов $M_1, M_2, \ldots, M_m$. О процедуре выявления характерных свойств речь пойдет ниже. А пока рассмотрим логическую формулировку наиболее типичных из этих свойств.

Инвариантные свойства. Свойства, которые всегда должны выполняться для потока сигналов, называются инвариантными свойствами. Они выражаются модальной формулой вида

$$\Vdash \Box w,$$

которая означает, что формула w должна быть всегда истинна для потока сигналов, определяемого начальными условиями этой формулы.

Корректное завершение. Это свойство имеет значение только для автоматов M_j, $j = 1, 2, \ldots, m$, каждый из которых содержит терминальное состояние (состояние, из которого не выходит ни одна дуга). Такие автоматы назовем терминальными в отличие от нетерминальных автоматов, которые не содержат терминальных состояний. Пусть $\boldsymbol{b}(t) = \{b_{j1}(t), \ldots, b_{jk}(t)\}$ — множество состояний различных терминальных автоматов $\{M_{j1}, \ldots, M_{jk}\} \subseteq \{M_1, M_2, \ldots, M_m\}$, называемое начальным внутренним макросостоянием; $y_j(t) = \varphi(b_j(t))$, $\boldsymbol{y}(t) = \{y_{j1}(t), \ldots, y_{jk}(t)\}$, $\boldsymbol{s}(\boldsymbol{b}(t))$ — предикат, принимающий истинное значение, когда автоматы находятся в макросостоянии $\boldsymbol{b}(t)$, $\boldsymbol{\phi}(\boldsymbol{x}(t))$ — отношение на множестве переменных $\boldsymbol{x}$, не являющихся отсчетами, $\boldsymbol{\varphi}(\boldsymbol{x}(t), \boldsymbol{y}(t))$ — утверждение о частичной корректности, т.е. отношение, которое должно установиться между множеством переменных $\boldsymbol{x}(t)$ и выходным макросостоянием $\boldsymbol{y}(t) = (y_{j1}(t), \ldots, y_{jk}(t))$. Используя введенные обозначения, свойство частичной корректности записывается следующим образом:

$$\Vdash (\boldsymbol{s}(\boldsymbol{b}(t_0)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t_0))) \supset \Box(\boldsymbol{s}(\boldsymbol{b}(t_e)) \supset \boldsymbol{\varphi}(\boldsymbol{x}(t_e), \boldsymbol{y}(t_e))).$$

Эта формула означает, что если выполняется предусловие $\boldsymbol{s}(\boldsymbol{b}(t_0)) \wedge$ $\wedge \boldsymbol{\phi}(\boldsymbol{x}(t_0))$ в начальном состоянии $\boldsymbol{b}(t_0)$, то в терминальном макросостоянии $\boldsymbol{b}(t_e)$, достижимом из начального состояния $\boldsymbol{b}(t_0)$, имеет место $\boldsymbol{\varphi}(\boldsymbol{x}(t_e), \boldsymbol{y}(t_e))$. Например, это может быть равенство $\boldsymbol{x}(t_e) = \boldsymbol{y}(t_e)$.

Чистое поведение. Свойство чистого поведения обусловливает выполнение определенных условий $\psi(\boldsymbol{x}(t_i), \boldsymbol{y}(t_i))$, если i пробегает множество значений $i \in I \subseteq \{1, 2, \ldots, T\}$. Используя введенные обозначения, свойство чистого поведения записывается следующим образом:

$$\Vdash (\boldsymbol{s}(\boldsymbol{b}(t_0)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t_0))) \supset \Box \bigwedge_{i \in I} (\boldsymbol{s}(\boldsymbol{b}(t_i)) \supset \boldsymbol{\psi}(\boldsymbol{x}(t_i), \boldsymbol{y}(t_i))).$$

Например, с помощью свойства чистого поведения можно указать, что значения $y_j(t_i)$ не должны превосходить определенный порог, иметь общий делитель и т.п

Глобальная инвариантность. Очень часто некоторые инвариантные свойства не зависят от какого-либо конкретного внутреннего состояния или макросостояния. Свойство глобальной инвариантности в общем виде выглядит следующим образом:

$$\Vdash \Box \boldsymbol{\phi}(\boldsymbol{x}(t)) \supset \Box \boldsymbol{\varphi}(\boldsymbol{x}(t), \boldsymbol{y}(t)).$$

т.е. всегда, когда имеет место $\boldsymbol{\phi}(\boldsymbol{x}(t))$, имеет также место и $\boldsymbol{\varphi}(\boldsymbol{x}(t), \boldsymbol{y}(t))$.

Сигнальное взаимное исключение. Понятие сигнального взаимного исключения связано с понятием сигнальной критической секции. Напомним суть этого понятия применительно к предмету нашего рассмотрения. Как уже говорилось, каждый отсчет является значением функции выхода $y_j(t) = \varphi(b_j(t))$ автомата M_j, каждый сигнал является последовательностью значений функций выхода $\boldsymbol{y}_j(t) = (y_j(t_0), y_j(t_1), \ldots, y_j(t_e))$ одного и того же автомата M_j. Рассмотрим любую пару автоматов $\{M_l, M_k\} \subseteq \{M_1, M_2, \ldots, M_m\}$. Пусть автоматы M_l, M_k достигли одновременно состояний $b_l(t), b_k(t)$. Назовем последовательности состояний $DS_l = (b_l(t), \ldots, b_l(t+r))$ и $DS_k = (b_k(t), \ldots, b_k(t+s))$ взаимно исключающими, если какие-либо условия не должны выполняться, когда автоматы находятся в каких-либо состояниях этих последовательностей. Обозначим $\pi(DS_j)$ условие, которое истинно, если автомат M_j находится в одном из состояний последовательности DS_j. Тогда свойство сигнального взаимного исключения DS_l и DS_k может быть представлено следующим образом:

$$\Vdash \boldsymbol{s}(\boldsymbol{b}(t_0)) \supset \Box \neg (\pi(DS_l) \wedge \pi(DS_k)) \wedge t \geqslant t_0.$$

Иначе его можно переписать следующим образом:

$$\Vdash \boldsymbol{s}(\boldsymbol{b}(t_0)) \supset \Box \neg [(s(b_l(t)) \vee \ldots \vee s(b_l(t+r))) \wedge (s(b_k(t)) \vee \ldots \\ \ldots \vee s(b_k(t+r)))] \wedge t \geqslant t_0.$$

Кадровое взаимное исключение. Каждый кадр — это набор $\boldsymbol{y}(t) = (y_1(t), \ldots, y_m(t))$ значений функций выхода различных автоматов $M_1, M_2, \ldots, M_m$, определенный на внутреннем макросостоянии $\boldsymbol{b}(t)$, т.е. $\boldsymbol{y}(t) = \boldsymbol{\varphi}(\boldsymbol{b}(t))$. Поток кадров является последовательностью кадров. Назовем последовательности макросостояний $DK_l = (\boldsymbol{b}_l(t_1), \ldots, \boldsymbol{b}_l(t_1 + r))$ и $DK_k = (\boldsymbol{b}_k(t_2), \ldots, \boldsymbol{b}_k(t_2 + s))$, где $\{(t_1+r)-t_1\} \cap \{(t_2+s)-t_2\} = \emptyset$, взаимно исключающими, если какие-либо условия не должны выполняться, когда автоматы находятся во внутренних макросостояниях этих последовательностей. Обозначим $\pi(DK_j)$ условие, которое истинно, если автоматы $M_1, M_2, \ldots, M_m$ находятся в одном из макросостояний последовательности DK_j. Тогда свойство кадрового взаимного исключения может быть представлено следующим образом:

$$\Vdash \boldsymbol{s}(\boldsymbol{b}(t_0)) \supset \Box\neg[\pi(\boldsymbol{b}_l(t_1), \ldots, \boldsymbol{b}_l(t_1 + r)) \wedge \pi(\boldsymbol{b}_k(t_2), \ldots \ldots, \boldsymbol{b}_k(t_2 + s))] \wedge (t_1 + r < t_2).$$

Свойства осуществимости. Эта категория свойств выражается общей формулой

$$\Vdash w_1 \supset \Diamond w_2,$$

означающей, что если формула w_1 для некоторого потока сигналов или кадров в какой-то момент времени станет истинной, то w_2 должна, в конце концов, также стать истинной. В отличие от свойств инвариантности, которые описывают только сохранение определенных свойств, свойства осуществимости говорят о том, что некоторые события, в конце концов, должны осуществиться. Эта формула в определенном смысле задает цель w_2, которая обязательно должна быть достигнута после истечения некоторого, в общем случае, неизвестного времени.

Абсолютная корректность. Это свойство похоже на свойство корректного завершения и имеет смысл только для терминальных автоматов. Абсолютная корректность выражается следующим образом:

$$\Vdash [\boldsymbol{s}(\boldsymbol{b}(t_0)) \wedge \boldsymbol{\varphi}(\boldsymbol{x}(t_0), \boldsymbol{y}(t_0))] \supset \Diamond[\boldsymbol{s}(\boldsymbol{b}(t_e)) \supset \boldsymbol{\varphi}(\boldsymbol{x}(t_e), \boldsymbol{y}(t_e))],$$

Это означает, что если имеется поток, для которого в момент времени t_0 имеет место истинность $\boldsymbol{s}(\boldsymbol{b}(t_0)) \wedge \boldsymbol{\varphi}(\boldsymbol{x}(t_0), \boldsymbol{y}(t_0))$, то позднее обязательно наступит время t_e, когда все автоматы окажутся в терминальных состояниях и будет иметь место отношение $\boldsymbol{\varphi}(\boldsymbol{x}(t_e), \boldsymbol{y}(t_e))$ (будет истинна формула $\boldsymbol{\varphi}(\boldsymbol{x}(t_e), \boldsymbol{y}(t_e))$).

Например, если в момент времени t_0 среднее значение всех функций выхода (отсчетов), входящих в $y(t_0)$, не более определенной величины, то в конце концов в момент времени t_e оно станет равным

среднему значению отсчетов в момент времени t_e, а все автоматы окажутся в терминальных состояниях.

Гарантированное осуществление. Свойства осуществимости позволяют выразить причинно-следственную связь между любыми двумя событиями, возникающими в потоках сигналов или кадров, а не только между их началом и окончанием. Такая возможность особенно существенна, когда потоки являются циклическими. Общий вид формулы, выражающей свойство гарантированного осуществления, следующий:

$$\Vdash (\boldsymbol{s}(\boldsymbol{b}(t)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t), \boldsymbol{y}(t))) \supset \Diamond[\boldsymbol{s}(\boldsymbol{b}'(t'')) \wedge \boldsymbol{\varphi}(\boldsymbol{y}(t'), \boldsymbol{x}(t')) \wedge t' > t].$$

Свойства предшествования. Третий класс рассматриваемых свойств составляют свойства, которые называются свойствами предшествования и выражаются с помощью оператора **U** (“до тех пор, пока”) и дополнительного оператора предшествования **P** (“предшествует”), который можно рассматривать как частный случай оператора **U**. В простейшем случае свойства предшествования описываются следующим выражением:

$$\Vdash w_1 \mathbf{U} w_2.$$

Это выражение означает, что для потока сигналов или кадров в будущем наступит такое время, когда формула w_2 станет истинной, причем до этого времени формула w_2 была ложной, а w_1 была истинной и перестанет быть таковой с наступлением этого времени.

Для оператора предшествования **P** выражение $w_1 \mathbf{P} w_2$ эквивалентно выражению $\neg(\neg w_1 \mathbf{U} w_2)$. Из определения оператора **U** следует смысл оператора **P**: не может быть такой ситуации, когда формула w_2 была ложной и стала истинной, а формула $\neg w_1$ была до этого истинной (т.е. w_1 была ложной) и стала ложной после этого. Это означает, что если w_2 когда-либо станет истинной, то это произойдет не раньше, чем истинной станет w_1. Выражение $w_1 \mathbf{P} w_2$ читается как “w_1 предшествует w_2”.

Рассмотрим некоторые из свойств предшествования, использующих операторы **U** и **P**.

Безопасность. Свойство безопасности, использующее оператор **U**, является уточнением свойства гарантированного осуществления:

$$\Vdash (\boldsymbol{s}(\boldsymbol{b}(t)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t), \boldsymbol{y}(t))) \supset \Diamond[\boldsymbol{s}(\boldsymbol{b}'(t'')) \wedge \boldsymbol{\varphi}(\boldsymbol{y}(t'), \boldsymbol{x}(t')) \wedge t' > t],$$

используемого, чтобы выразить достижение потоком некоторого состояния $b'(t')$, в котором формула $\boldsymbol{\varphi}(\boldsymbol{y}(t'), \boldsymbol{x}(t'))$ будет истинна, если

во время t поток находился в состоянии $\boldsymbol{b}(t)$ и формула $\boldsymbol{\phi}$ была истинна. Состояние $\boldsymbol{b}'(t')$ при этом не обязательно достигается в первый раз.

В отличие от свойства гарантированного осуществления свойство безопасности

$$\Vdash [\boldsymbol{s}(\boldsymbol{b}(t)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t), \boldsymbol{y}(t))] \supset [(\neg \boldsymbol{s}(\boldsymbol{b}'(t'))\mathbf{U}[\boldsymbol{s}(\boldsymbol{b}'(t')) \wedge \boldsymbol{\phi}'\boldsymbol{y}(t), \boldsymbol{x}(t)])]$$

означает следующее: если поток находится в состоянии $\boldsymbol{b}(t)$ и формула $\boldsymbol{\phi}(\boldsymbol{x}(t))$ истинна, то через некоторое время поток должен достичь состояния $\boldsymbol{b}'(t')$ и при первом же приходе в это состояние формула $\boldsymbol{\phi}'$ должна стать истинной.

Предусмотренная реакция. Свойства, выражаемые формулой $w_1 \supset \Diamond w_2$, относятся к числу свойств, которые гарантируют, что в случае истинности формулы w_1, истинной станет формула w_2. Часто желательно дополнить это свойство требованием, чтобы формула w_2 никогда не становилась истинной, если этому не предшествует истинность формулы w_1. Это свойство в общем случае выразимо с помощью оператора предшествования, но требует в каждом конкретном случае тщательного выбора времени наблюдения. Хорошим примером выражения, задающего свойство предусмотренной реакции, является следующее:

$$\Vdash [\boldsymbol{s}(\boldsymbol{b}(t_0)) \supset w_1\mathbf{P}w_2] \wedge [(w_2 \wedge \mathbf{N}\neg w_2) \supset \mathbf{N}(w_1\mathbf{P}w_2)].$$

Этот пример иллюстрирует тщательный выбор времени наблюдения, когда предшествование w_1 по отношению к w_2 может быть наблюдаемым либо из начального состояния, либо во время, когда w_2 истинна и изменяет свое значение на ложное в следующий момент времени (здесь $\mathbf{N}$ означает “в следующий момент времени”). В большинстве практических случаев имеется дополнительная информация о формулах w_1 и w_2, которая помогает сформулировать требования в более простой форме.

Справедливая реакция. Во многих ситуациях предшествование двух событий φ_1 и φ_2 связано с более ранними событиями ϕ_1 и ϕ_2, такими, что ϕ_1 предшествует событию ϕ_2. Такую ситуацию будем называть условным предшествованием. Она представляется выражением

$$\Vdash (\phi_1\mathbf{P}\phi_2) \supset (\varphi_1\mathbf{P}\varphi_2).$$

Оно означает, что если ϕ_1 предшествует ϕ_2, то φ_1 предшествует φ_2. Вместе с выражениями

$$\Vdash \phi_1 \supset \Diamond\varphi_1,$$

$$\Vdash \phi_2 \supset \Diamond\varphi_2$$

выражение условного предшествования называется справедливой реакцией. Иначе говоря, если $\Vdash \phi_1 \supset \Diamond\varphi_1$ и $\Vdash \phi_2 \supset \Diamond\varphi_2$ интерпретируются, как ответ φ_i на требование ϕ_i, то справедливая реакция дополнительно требует: если ϕ_1 предшествует ϕ_2, то ответ на ϕ_1, а именно φ_1, должен предшествовать ответу на ϕ_2, а именно φ_2.

Свойства агентов. Агенты — это интеллектуальные параллельно работающие процессы. Поэтому проверка свойств поведения агентов — это, по существу, проверка определенных свойств этих процессов. Практически все из рассмотренных свойств потоков несколько в иной интерпретации применимы и для проверки свойств поведения объектов. Чтобы это было более очевидным, будем полагать, что вычислительной моделью процессов являются конечные автоматы. В отличие от автомата M_j:, представляющего поток, функции перехода и выхода агента A_j: имеют следующий вид:

$$\varphi(b_j(t_i)) = c_j(t_i),$$
$$f(b_j(t_i),\ \ a(t_{i+1})) = b_j(t_{i+1}).$$

Здесь $c_j(t_i)$ является значением функции выхода φ, а значением функции выхода в общеупотребительном смысле теории автоматов $a(t_{i+1})$ — входное условие при истинном значении которого происходит переход из состояния $b_j(t_i)$ в состояние $b_j(t_{i+1})$, являющееся значением функции переходов f. Далее те свойства поведения агентов, которые с точностью до обозначений совпадают со свойствами потоков, только перечислим в новых обозначениях. Большее внимание уделим свойствам поведения агентов, не имеющим аналогов среди свойств потоков.

Инвариантные свойства.

Корректное завершение

$$\Vdash (\boldsymbol{s}(\boldsymbol{b}(t_0)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t_0))) \supset \Box[\boldsymbol{s}(\boldsymbol{b}(t_e)) \supset \boldsymbol{\varphi}(\boldsymbol{x}(t_e),\, c(t_e))].$$

Чистое поведение

$$\Vdash (\boldsymbol{s}(\boldsymbol{b}(t_0)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t_0))) \supset \Box \bigwedge_{i\in I}(\boldsymbol{s}(\boldsymbol{b}(t_i)) \supset \boldsymbol{\psi}(\boldsymbol{x}(t_i),\, c(t_i))).$$

Глобальная инвариантность

$$\Vdash \Box(\boldsymbol{s}(\boldsymbol{b}(t)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t))) \supset \Box(\boldsymbol{\varphi}(\boldsymbol{x}(t),\, c(t))).$$

Взаимное исключение критических секций. Каждый агент A_j является процессом P_j. Если процессы P_l, P_k, являющиеся автоматами,

одновременно достигли состояний $b_l(t)$, $b_k(t)$, требующих одного и того же ресурса на период обработки состояний соответственно в диапазонах $D_l = (b_l(t), \ldots, b_l(t+r))$ и $D_k = (b_k(t), \ldots, b_k(t+s))$, то только один процесс из указанных двух может получить доступ к ресурсу для обработки своего диапазона состояний. Эти диапазоны называются критическими секциями. Только один из процессов может обрабатывать свою критическую секцию. Это и называется взаимным исключением критических секций. Взаимное исключение критических секций, таким образом, необходимо, когда двум или более процессам необходим доступ к распределяемым переменным или ресурсам, таким, например, как диски, и нам необходимо защититься от взаимовлияния этих процессов или от попыток других процессов получить доступ к тому ресурсу, с которым данный процесс уже имеет дело. Обозначим $\pi(D_j)$ условие, которое истинно, если процесс P_j находится в одном из состояний критической секции D_j. Тогда свойство взаимного исключения критических секций D_l и D_k может быть представлено следующим образом:

$$\Vdash \boldsymbol{\phi}(\boldsymbol{x}(t_0)) \supset \Box\neg(\pi(D_l) \wedge \pi(D_k)) \wedge t \geqslant t_0.$$

Это свойство означает, что всегда, когда процесс P_l обрабатывает состояния автомата A_l из секции D_l, то процесс P_k не может обрабатывать состояния автомата A_k из секции D_k. Если ввести предикат: $s(b(t))$ — истинный, когда автомат находится в состоянии $b(t)$, то свойство сигнального взаимного исключения можно выразить иначе:

$$\Vdash \boldsymbol{\phi}(\boldsymbol{x}(t_0)) \supset \Box\neg[(s(b^l(t)) \vee \ldots \vee s(b^l(t+r))) \wedge (s(b^k(t)) \vee \ldots \\ \ldots \vee s(b^k(t+r)))] \wedge t \geqslant t_0.$$

Свойства осуществимости.

Абсолютная корректность. Это свойство похоже на свойство корректного завершения и имеет смысл только для терминальных автоматов. Абсолютная корректность выражается следующим образом:

$$\Vdash \boldsymbol{\phi}(\boldsymbol{x}(t_0)) \supset \Diamond\boldsymbol{\varphi}(\boldsymbol{b}(t_e),\, c(t_e)).$$

Гарантированное осуществление

$$\Vdash [\boldsymbol{s}(\boldsymbol{b}(t)) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t))] \supset \Diamond[\boldsymbol{s}(\boldsymbol{b}(t')) \wedge \boldsymbol{\varphi}(c(t')) \wedge \boldsymbol{\phi}(\boldsymbol{x}(t'))] \wedge t' > t.$$

Доступность. Рассмотрим снова процесс P_j, который имеет критическую секцию D_j. При изучении свойства взаимного исключения было показано, как установить защиту для такой секции от вмешательства со стороны других процессов. Продолжающим и дополняющим

свойством к свойству взаимного исключения является свойство доступности, означающее следующее: если процесс желает войти в свою критическую секцию, он, в конечном счете, попадает туда и не будет неопределенно долго удерживаться механизмом защиты. Очевидно, что защитный механизм является бесполезным, если он не позволяет, в конце концов, процессу войти в его критическую секцию.

Пусть $b(t_0)$ — состояние перед критической секцией. Тот факт, что процесс находится в $b(t_0)$, отражает намерение войти в критическую секцию D_j. Тогда свойство доступности выглядит следующим образом:

$$\Vdash s(b(t_0)) \supset \Box[\pi(D_j) \wedge t \geqslant t_0].$$

Правильные конструкции критических секций должны гарантировать выполнение двух полезных свойств: взаимного исключения и доступности.

Беступиковость

$$\Vdash \Diamond\neg s(b^j(t)),$$

или

$$\Vdash \neg\Box s(b^j(t)).$$

Отзывчивость. Важными классами являются бесконечные процессы (например, циклически бесконечно повторяющиеся или настолько длинные, что могут считаться бесконечными). Они не имеют терминальных состояний. Остановка этих процессов является ошибочной. Следовательно, свойства корректного завершения или абсолютной корректности для таких процессов неприменимы. Вместо них используется свойство отзывчивости. Для описания свойства отзывчивости одной формулы недостаточно, так как помимо процессов, обрабатывающих потоки, в этом процессе участвует диспетчер (операционная система). Кроме того, свойства отзывчивости зависят от принципов взаимодействия процессов с диспетчером.

Рассмотрим один из вариантов такого взаимодействия и соответствующее ему описание свойства отзывчивости.

Предположим, что диспетчер G взаимодействует с процессами $P_1, \ldots, P_A$, распределяя между ними ресурсы.. Ресурсами могут быть жесткие диски, главная память, каналы передачи информации и т.д. Процессы взаимодействуют с диспетчером, запрашивая ресурсы через пропозициональные переменные $\{r_j,\ g_j\}$, $j = 1, \ldots, A$. Переменная r_j устанавливается истинной процессом P_j, чтобы сигнализировать о запросе ресурса. Переменная g_j устанавливается истинной диспетчером G, сообщающим, тем самым, процессу P_j, что он может взять ресурс. После использования ресурса процесс P_j возвращает ресурс обратно

диспетчеру G, присваивая переменной r_j ложное значение. Диспетчер признает освобождение ресурса присвоением переменной g_j ложного значения.

Суммируя сказанное, имеем:

P_j выдает запрос $\supset r_j{:=}\,истина$;

G предоставляет ресурс $\supset g_j{:=}\,истина$;

P_j освобождает ресурс $\supset r_j{:=}\,ложь$;

G признает освобождение $\supset g_j{:=}\,ложь$.

Утверждением, означающим, что диспетчер справедливо реагирует на запрос процесса P_j, является

$$r_j \supset \Diamond g_j,$$

т.е. когда бы r_j не стала истинной, в конце концов g_jбудет истинной.

Подобным образом необходимо обеспечить подтверждение освобождения ресурса диспетчером установкой g_j в *ложь*:

$$b_j{:}\neg r_j \supset \Diamond(\neg g_j).$$

Более того, диспетчер не может работать нормально, если не будет обеспечено должного взаимодействия между процессами. Например, диспетчер не может обещать процессу P_j предоставления, в конце концов, ресурса, если процесс P_i, который в это время занимает его, в конце концов, его не освободит. Следовательно, требуется, чтобы поведение процесса P_j удовлетворяло определенным условиям, а именно:

$$g_j \supset \Diamond(\neg r_j).$$

Это утверждение говорит о том, что в случае предоставления процессу P_j ресурса r_j, этот ресурс, в конце концов, будет освобожден. К сформулированным утверждениям обычно добавляются утверждения, гарантирующие правильное непрерывное поведение диспетчера G. Одним из таких утверждений является следующее:

$$\Box\left(\sum_{j=1}^{j=m} g_j \leqslant 1\right),$$

означающее, что в любое время на интервале $[t_0, t_e]$ диспетчер предоставляет ресурс самое большее одному процессу.

Описание корректного поведения диспетчера G в целом будет выглядеть следующим образом:

$$\xi = \bigwedge_{j=1}^{j=m}(r_j \supset \Diamond g_j) \wedge \bigwedge_{j=1}^{j=m}(b_j{:}\neg r_j \supset \Diamond(\neg g_j)) \wedge \Box\left(\sum_{j=1}^{j=m} g_j \leqslant 1\right).$$

Описание корректного поведения процессов P_j имеет вид

$$\varphi = \bigwedge_{j=1}^{j=m}(g_j \supset \Diamond(\neg r_j)).$$

Общая формулировка свойства отзывчивости может быть получена, исходя из следующих соображений. Переменные r_j и g_j являются соответственно входными и выходными переменными диспетчера. Значения переменных r_j устанавливаются процессами $P_1, \ldots, P_A$.

Тогда свойство отзывчивости выглядит следующим образом:

$$\Vdash \Box\varphi \supset \Box\xi.$$

Свойства предшествования.

Предусмотренная реакция

$$\Vdash (s(b(t_0) \supset w_1 \mathbf{P} w_2) \wedge [(w_2 \wedge \mathbf{N}\neg w_2) \supset \mathbf{N}(w_1 \mathbf{P} w_2)].$$

Пример. Распределение ресурсов. Вернемся к рассмотренному ранее примеру с диспетчером G, который распределяет ресурс между процессами $P_1, \ldots, P_t$. Простейший протокол взаимодействия между процессом P_i и диспетчером определяется следующими четырьмя выражениями:

$\Vdash (r_i \wedge \neg g_i) \supset \mathbf{N}\, r_i.$ (1)

Это выражение говорит о том, что если r_i истинна, а g_i ложна, т.е. запрос на ресурс со стороны P_i имеется, но он еще не предоставлен, то процесс P_i обязан настаивать на своем требовании, подтверждая значение r_iв следующем такте.

$\Vdash (r_i \wedge g_i) \supset \mathbf{N} g_i.$ (2)

Это выражение означает, что если ресурс был предоставлен процессу P_i, то диспетчер не имеет права отнять его до тех пор, пока пользователь не присвоит r_i ложное значение.

$\Vdash (r_i \wedge g_i) \supset \mathbf{N}\neg r_i.$ (3)

Это утверждение означает, что если диспетчер еще не признал отказа от ресурса пользователя P_i, то P_i не может выставлять нового требования.

$\Vdash (\neg r_i \wedge \neg g_i) \supset \mathbf{N}\neg g_i.$ (4)

Это утверждение означает, что если ресурс не занят процессом P_i и не затребован им, то ресурс не предоставляется процессу, который его не требует. Все это и есть описание свойства предусмотренной реакции. Выполнение перечисленных свойств вместе со следующими дополнительными выражениями гарантирует правильное поведение программы:

$\Vdash r_i \supset \Diamond g_i$; (5)
$\Vdash g_i \supset \Diamond \neg r_i$; (6)
$\Vdash \neg r_i \supset \Diamond \neg g_i$. (7)

Четыре первых утверждения характеризуют поведение программы в соседние моменты времени или состояния. В более глобальном стиле свойства предусмотренной реакции можно выразить следующими выражениями:

$\Vdash r_i \supset [r_i \mathbf{U}(g_i \wedge r_i)]$; (a)
$\Vdash g_i \supset [g_i \mathbf{U}(\neg r_i \wedge g_i)]$; (b)
$\Vdash \neg r_i \supset [r_i \mathbf{U}(\neg g_i \wedge \neg r_i)]$; (c)
$\Vdash g_i \supset (r_i \mathbf{P} g_i)$. (d)

Эти выражения заменяют все остальные. Утверждение (a) говорит о том, что если r_i истинно, то оно должно оставаться истинным до тех пор, пока g_i истинно. Утверждение (b) говорит о том, что если ресурс предоставлен, то он остается предоставленным до тех пор, пока от него не откажутся. Утверждение (c) говорит о том, что если от ресурса отказались, то он не может быть затребован до тех пор, пока отказ от него не будет признан. Утверждение (d) говорит о том, что если g_i в данный такт не назначено, то его следующему значению должен предшествовать запрос.

Справедливая реакция

$$\Vdash (\phi_1 \mathbf{P} \phi_2) \supset (\varphi_1 \mathbf{P} \varphi_2);$$
$$\Vdash \phi_1 \supset \Diamond \varphi_1;$$
$$\Vdash \phi_2 \supset \Diamond \varphi_2.$$

Пример. Распределение ресурсов. Рассмотрим снова обслуживаемые диспетчером процессы. Наложим на них требование условного предшествования: $\Vdash (r_i \mathbf{P} r_\varphi) \supset (g_i \mathbf{P} g_\varphi)$, которое означает, что если процесс P_i поместил свой запрос раньше процесса P_φ, то он раньше и будет обслужен. Однако здесь необходимо быть осторожным, ставя жесткие условия. Например, если g_i истинно, в то время, как оба r_i и r_φ ложны, то может иметь место ситуация, когда нельзя обещать, что g_i будет предшествовать g_φ. Для исключения такой ситуации необходимо исходить из ложности g_φ. Тогда выражение условного предшествования будет выглядеть следующим образом:

$$\Vdash (\neg g_\varphi) \supset [(r_i \mathbf{P} r_\varphi) \supset (g_i \mathbf{P} g_\varphi)].$$

Пример формулировки свойств правильного поведения агентов. Имеется k агентов. Агент A_1 является отправителем кадров, агенты $A_2, A_3, \ldots, A_k$ — передатчиками кадров. Агент A_1 помещает кадры

в буфер, а агенты $A_2, A_3, \ldots, A_k$ берут их из него и рассылают потребителям. Все агенты являются процессами и функционируют параллельно. Наша задача — формулировать свойства их правильного поведения. Прежде чем рассматривать механизм взаимодействия агентов, напомним, что такое семафорные переменные. Семафорными переменными являются особые переменные, используемые для синхронизации работы параллельных процессов (в нашем случае агентов). Над каждой из этих переменных x могут выполняться только две команды: *занять* (x) и *освободить* (x). Переменная может быть занята, если ее значение не равно 0. Если переменная занята, то до ее освобождения никто ее занять не может. При занятии значение семафорной переменной увеличивается на 1, при освобождении — уменьшаться на 1 (указывается в комментариях). Команда *освободить* может выполняться и над незанятой семафорной переменной, изменяя ее значение. Никакие другие операции над семафорными переменными не допускаются. Процесс, достигший команды *занять* (x), будет выполняться дальше только в том случае, если нет ни одного другого процесса, который уже занял эту переменную и еще не освободил. После освобождения переменной она может быть занята любым другим процессом. При попытке занятия свободной семафорной переменной сразу несколькими процессами занять ее может один из них. Для синхронизации взаимодействия агентов $A_1, A_2, \ldots, A_k$ введем три семафорных переменных:

• переменную x_1, используемую для исключения одновременного доступа агентов в буфер;

• переменную x_2, указывающую число свободных мест в буфере. Она защищает буфер от переполнения. Отправитель не может положить кадр в буфер, если x_2=0. Значение переменной x_2 увеличивается на 1 после каждого помещения отправителем кадра в буфер. В начале работы агента A_1 значение $x_2 = N$. Отправитель не может положить в буфер более, чем N кадров. Каждый потребитель уменьшает x_2 на 1 после взятия одного кадра;

• переменную x_3, содержащую текущее количество кадров в буфере; ее начальное значение равно 0; оно увеличивается отправителем на 1 всякий раз, когда он кладет туда кадр, и уменьшается передатчиком на 1 всякий раз, когда он берет из буфера кадр. Эта переменная используется для того, чтобы потребитель никогда не пытался взять кадр из пустого буфера.

Агенты используют буферB, который имеет N ячеек памяти $B(l)$, $l = 1, \ldots, N$. Буфер может быть пуст и тогда переменная $x_3 = 0$. Если $x_3 \neq$0, то в буфере содержится x_3 кадров в ячейках $B(1), \ldots, B(x_3)$. Ячейка памяти $B(1)$ в этом случае называется головной и обознача-

ется *голова*(B). Ячейка памяти $B(x_3)$ буфера B называется хвостом и обозначается *хвост*(B).

Агент A_1 при желании поместить кадр в буфер ведет себя следующим образом. Сначала он пытается занять семафорную переменную x_2. Если ему это удается и значение семафорной переменной x_2 не равно 0, то это означает, что агент A_1 может поместить кадр c в буфер, поскольку там есть свободное место. Для того чтобы это сделать, он должен занять семафорную переменную x_1. Если ему удается ее занять, то сначала содержимое каждой ячейки $B(j)$, $j = 1, \ldots, x_3$, перемещается в ячейку $B(j+1)$, затем агент A_1 помещает кадр в освободившуюся ячейку *голова*(B), семафорная переменная x_1 освобождается, значение переменной x_2 уменьшается на 1, а значение переменной x_3 увеличивается на 1. Такую комплексную операцию будем обозначать *голова*$(B) + B$. Каждый из агентов A_p, $p = 2, \ldots, k$, при желании взять кадр из буфера ведет себя следующим образом. Он пытается занять семафорную переменную x_3. Если ему удается это сделать (переменная не равна 0), то это означает, что буфер не пуст и из него можно взять кадр. Тогда агент A_p переходит к попытке занять семафорную переменную x_1. Если ему и это удастся сделать, то он забирает содержимое ячейки *хвост*(B). Такую операцию будем обозначать B — *хвост*(B)_. Затем агент освобождает сначала переменную x_1, а затем переменную x_2. После этого он производит ряд вычислений, для того, чтобы получить значение ω_p времени передачи извлеченного из буфера кадра (сек) передатчиком A_p.

Таким образом, поведение агентов A_1 и A_p может быть представлено следующей программой.

Программа “Отправитель–передатчики”

$x_1 = 1$, ∗семафорная переменная доступа к буферу∗
$x_2 = N$, ∗семафорная переменная числа пустых ячеек буфера∗
x_3=0, ∗семафорная переменная числа заполненных ячеек буфера∗
$v = C$, ∗переменная пропускной способности канала передачи кадров (бит/с)∗
$\tau = T$, ∗переменная максимально допустимого времени передачи кадра (с)∗
$\beta = 0$, ∗ переменная числа бит в кадре∗
$\varepsilon = 0$, ∗переменная времени записи кадра в буфер (с)∗
$\delta = 0$, ∗переменная времени извлечения кадра из буфера (с)∗
$\omega_p = 0$, ∗ переменная времени передачи извлеченного из буфера кадра (с) передатчиком A_p∗

Отправитель $\mathbf{A}_1$

b_1^1: *Вычислить* $\boldsymbol{y}_1 = \boldsymbol{\varphi}[\boldsymbol{b}(t)]$;

b_2^1: *Вычислить* $\varepsilon = \varepsilon[\boldsymbol{y}_1]$;
b_3^1: *Занять*(x_2); $*x_2 := x_2 - 1*$
b_4^1: *Занять*(x_1); $*x_1 := x_1 - 1*$
b_5^1: *голова*(B):=$\boldsymbol{y}_1$;
b_6^1: $B :=$ *голова*$(B) + B$;
b_7^1: *Освободить*(x_1); $*x_1 := x_1 + 1*$
b_8^1: *Освободить*(x_3); $*x_3 := x_3 + 1*$
b_9^1: *Вычислить* $\varepsilon = \varepsilon[\boldsymbol{y}_1]$;
b_{10}^1: *Перейти на* b_1;

Передатчик $\mathbf{A}_p$

b_1^p: *Занять*(x_3); $*x_3 := x_3 + 1*$
b_2^p: *Занять*(x_1); $*x_1 := x_1 - 1*$
b_3^p: $\boldsymbol{y}_2 :=$ *хвост*(B);
b_4^p: $B := B -$ хвост(B);
b_5^p: *Освободить*(x_1); $*x_1 := x_1 + 1*$
b_6^p: *Освободить*(x_2); $*x_2 := x_2 + 1*$
b_7^p: *Вычислить* $\beta = \beta[\boldsymbol{y}_2]$;
b_8^p: *Вычислить* $\delta = \delta[\boldsymbol{y}_2]$;
b_9^p: *Вычислить* $\omega_p = \beta v + \delta + \varepsilon$;
b_{10}^p: *Перейти на* b_1^p.

Остальные передатчики имеют аналогичные программы. Очевидно, что в рассмотренной программе секции $D_1 = \{b_4^1, b_5^1, b_6^1\}$ агента A_1 и $D_p = \{b_2^p, b_3^p, b_4^p\}$ агента A_p являются критическими. Для того чтобы получить корректный результат, необходимо гарантировать для них выполнение свойства взаимного исключения:

$$\phi(x) \supset \Box\neg(\pi(D_1) \wedge \pi(D_p)),$$

где

$$\phi(x) = s(b_1^1) \wedge s(b_1^p) \wedge B = \varnothing \wedge x_1 = 1 \wedge x_2 = N \wedge x_3 = 0.$$

Эта программа будет свободна от зависания, если выполняется условие:

$$\begin{aligned}
&\Box\{[(s(b_2^1) \wedge s(b_1^p)) \supset (x_1 > 0 \wedge x_2 > 0)];\\
&\wedge\,[(s(b_2^1) \wedge s(b_2^p)) \supset (x_1 > 0 \wedge x > 0)];\\
&\wedge\,[(s(b_3^1) \wedge s(b_1^p)) \supset (x > 0 \wedge x2 > 0)];\\
&\wedge\,[(s(b_3^1) \wedge s(b_2^p)) \supset (x > 0)]\}.
\end{aligned}$$

Свойство доступности программы будет следующим:

$$[s(b_2^1) \supset \Diamond s(b_4^1)] \wedge [s(b_1^p) \supset \Diamond s(b_3^p)].$$

Доказательство свойств. Известно, что для того чтобы иметь возможность проверять наличие тех или иных свойств агентов и потоков из числа рассмотренных или каких-либо иных дедуктивным способом, необходимо иметь некоторый адекватный решаемым задачам проверки свойств агентов и потоков язык исчисления и формулировки на этом языке:

— базовых общезначимых аксиом (если они необходимы);

— аксиом, описывающих общие свойства моделей мультиагентной системы (программы, состоящей из параллельных процессов), не зависящие от конкретного типа системы. Эти аксиомы назовем общими агентскими аксиомами;

— аксиом, описывающих свойства конкретной модели мультиагентной системы, зависящей от решаемой ею задачи. Эти аксиомы назовем частными агентскими аксиомами. Частные агентские аксиомы описывают графы переходов агентов;

— аксиом, описывающих общие свойства моделей потоков, не зависящие от конкретного типа потоков. Эти аксиомы назовем общими потоковыми аксиомами;

— аксиом, описывающих свойства конкретной модели потоков. Эти аксиомы назовем частными потоковыми аксиомами. Частные потоковые аксиомы описывают графы переходов потоков;

— правил вывода, состоятельных для данного исчисления.

Аксиомы образуют базу знаний или онтологию, а свойства (теоремы), которые требуется доказать, являются запросами к этой базе знаний.

Онтология хранится в определенных форматах и может быть распределена между агентами. Каждый из агентов решает некоторую частную задачу в соответствии со своей ролью. Часть из них решает общую задачу ответа на запрос. Каждый агент имеет входной и выходной порты, собственную базу знаний (онтологию) и решатель, или машину вывода, которая использует правила вывода. Через входной порт агент получает сообщения и запросы от других агентов, а также информацию из других источников, не являющихся агентами в указанном смысле. Через выходной порт агент выдает сообщения и запросы другим агентам, а также информацию получателям, не являющимися агентами.

При описании свойств потоков и агентов использовалась некоторая версия временной модальной логики. Эти описания носят теоретический характер, и использование напрямую рассмотренного языка

временной модальной логики нецелесообразно. Для конкретных реализаций может быть использован более выразительный язык ситуационного модального нечеткого исчисления [5], где синтаксическое описание служит средством обмена между агентами и представляет собой онтологию на формальном языке ситуационного модального нечеткого исчисления. Семантическое описание используется агентами для вычислений и представляет онтологию на языке нечетких отношений. Его можно рассматривать как формальную систему со своим языком, правилами вывода и интерпретацией. Вывод заключений делается на семантическом уровне. Связь между синтаксическим и семантическим уровнями осуществляется посредством процедуры семантической трансляции, которая выполняет перевод текста, представленного на синтаксическом уровне, на язык нечетких отношений, и процедуры лингвистической аппроксимации, которая выполняет перевод заключений, полученных на семантическом уровне, на язык синтаксического уровня. Агенты обмениваются информацией на синтаксическом уровне.

Запрос является теоремой о наличии у потоков или агентов некоторых свойств. Установление наличия этих свойств осуществляется с помощью процедуры вывода (доказательства) в нечетком исчислении. Подробно вариант исчисления и процедура вывода описаны в работе [5]. Для осуществления вывода агенты, отвечающие за обработку потоков, формируют и помещают по мере необходимости факты в онтологии. В процессе вывода могут участвовать несколько агентов, обменивающихся друг с другом сообщениями. Архитектура мультиагентной системы сбора, обработки и передачи потоков телеметрической информации кратко рассмотрена в следующем разделе.

Архитектура мультиагентной системы сбора, обработки и передачи потоков телеметрической информации. Общая архитектура мультиагентной системы сбора, обработки и передачи потоков телеметрической информации, которую кратко будем называть МАСТИ (Мультиагентной Системой Телеметрической Информации), представляет собой множество программных агентов. Каждый агент решает определенную задачу в системе.

Существует два возможных типа конфигураций МАСТИ:

- локальная конфигурация, включающая все программные компоненты, расположенные на отдельной рабочей станции;
- распределенная конфигурация, включающая системные компоненты, расположенные на нескольких рабочих станциях, объединенных в вычислительную сеть. Сеть может включать локальные или глобальные каналы связи. Каждая рабочая станция имеет собственную локальную конфигурацию, структура которой определяется ролью, от-

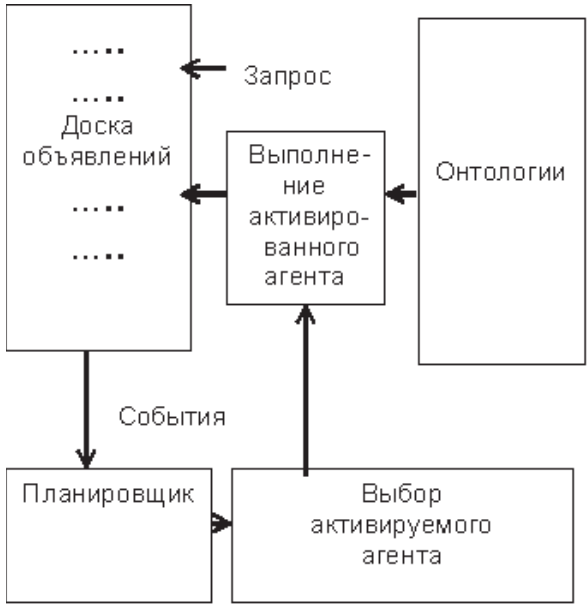

Рис. 3. Архитектура МАСТИ

веденной рабочей станции в системе.

Распределенная конфигурация использует сеть для передачи потоков и команд между рабочими станциями. Поддержка сетевых операций обеспечивается сетевыми агентами, включаемыми в локальные конфигурации. Архитектура МАСТИ для проверки свойств потоков и агентов построена на основе использования доски объявлений (рис. 3).

Планировщик системы выделяет агентам время для работы. Используется режим квазипараллельной работы агентов. Агенты получают для исполнения периодически короткие интервалы времени. На доску объявлений помещается текущая информация о состоянии агентов и потоков, используемая планировщиком для вызова (активации) соответствующих агентов.

Вся система и отдельные локальные подсистемы на каждой рабочей станции могут быть реконфигурированы при необходимости для решения различных задач по сбору, обработке и анализу свойств потоков и агентов. Различные конфигурации могут быть созданы на основе заданного набора агентов. Каждый агент имеет собственные настройки и панель управления, поддерживает COM интерфейс обмена данными и программный интерфейс, управляемый командами JScript. Коллекция агентов может быть расширена. В настоящее время агенты разрабатываются на языке C++. В качестве средства проектирования используется Microsoft Visual Studio .NET. Шаблон проекта агента создается с помощью дизайнера проектов Microsoft .NET Project Wizard.

С помощью программных инструментов, входящих в состав системы, можно создавать пользовательский интерфейс. Пользователь взаимодействует с системой через графический интерфейс, создание которого является составной частью процесса проектирования конфигурации. Каждый агент реализует две компоненты пользовательского интерфейса: панель управления и окно редактирования настроек. Эти компоненты являются основой для построения пользовательского интерфейса.

Кадры потоков имеют фиксированное число слов и, как правило, передаются с постоянной частотой. Частота формирования кадра определяет частоту опроса передаваемых в кадре параметров. На рис. 4 показан формат кадра ТМИ.

На рис. 5 показан формат заголовка кадра ТМИ. Признак достоверности кадра используется для указания сбоев кадровой синхронизации. Признак устанавливается аппаратурой.

На рис. 6 показана последовательность размещения телеметрических слов в кадре. Каждое слово содержит значение одного из каналов. Адрес канала определяется номером канала локального коммутатора (ЛК) и номером канала основного коммутатора (ОК). Например, слово Л3К1 содержит показание датчика, подключенного к первому каналу

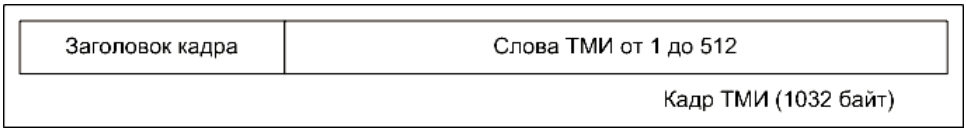

Рис. 4. Формат кадра ТМИ

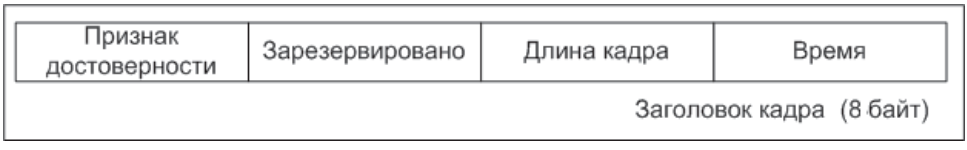

Рис. 5. Заголовок кадра

Рис. 6. Порядок слов в кадре

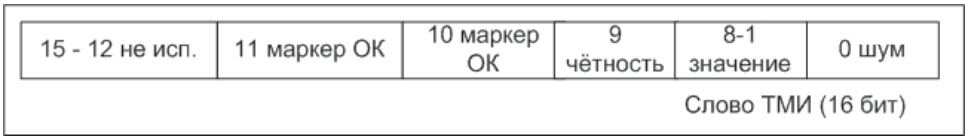

Рис. 7. Формат слова в кадре

локального коммутатора, который, в свою очередь, подключен к третьему каналу основного коммутатора (см. также рис. 1 и 2).

На рис. 7 показан формат слова в кадре. Бит 11 содержит признак маркера основного коммутатора. Он устанавливается в каждом восьмом слове. Бит 10 содержит признак маркера локального коммутатора. Он устанавливается в последних восьми словах кадра. Оба признака указывают на конец кадра. Маркеры обеспечивают кадровую синхронизацию. Бит 9 содержит дополнение до четности. Он используется для обнаружения битовых ошибок. Биты с 1 по 8 используются для хранения значения канала. Бит 0 не используется и принимает случайное значение.

МАСТИ должна обеспечивать правильную обработку потоков в рамках заданной конфигурации на всех рабочих станциях, а также передачу потоков между отдельными рабочими станциями в режиме реального времени. Для передачи потоков между рабочими станциями используются каналы с ограниченной пропускной способностью. В ряде случаев пропускная способность канала может быть недостаточна для передачи полного потока в темпе поступления от аппаратуры. Выходом из этой ситуации является оптимизация объема передаваемой информации, в частности на основе эквивалентных преобразований потоков, о которых упоминалось выше.

В заключение рассмотрим **пример выявления характерных свойств сигналов** с использованием нечеткого модального исчисления.

Координаты объекта определяются двумя датчиками. Датчики выдают периодические сигналы $X(t)$ и $C(t)$ с периодом 3, определяющие в каждый момент времени t координаты объекта соответственно по осям абсцисс и ординат. Будем считать их лингвистическими переменными. Универсумы $U(x)$ и $U(c)$ сигналов $X(t)$ и $C(t)$ будем считать одинаковыми для всех t. Для всех t на универсуме $U(x)$ заданы одинаковые функции принадлежности $\mu_l(x)$, $\mu_c(x)$, $\mu_r(x)$, на универсуме $U(c)$ — функции принадлежности $\mu_l(c)$, $\mu_c(c)$, $\mu_r(c)$. Будем полагать, что этим функциям принадлежности оответствуют лингвистические значения *Слева_*$X(t)$, *В_центре_*$X(t)$, *Справа_*$X(t)$; *Слева_*$C(t)$, *В_центре_*$C(t)$, *Справа_*$C(t)$ лингвистических переменных $X(t)$ и $C(t)$. Нечеткими грамматиками, соответствующими значе-

ниям сигналов $X(t)$ и $C(t)$ в моменты времени $t_0+k\Delta_t$, $t_0+(k+1)\Delta_t$, где $k = 1, 3, 5, \ldots$, являются соответственно $G_x = \{V_x, A_x, P_x, B_x\}$, $G_c = \{V_c, A_c, P_c, B_c\}$, где $V_x = \{b_1, b_2\}$, $A_x = \{t_0 = t_0 + k\Delta_t$, $t_1 = t_0 + (k+1)\Delta_t\}$, $P_x = \{B_x \to t_0 b_1,\ b_1 \to t_1 b_2\}$, $V_c = \{A_1, A_2\}$, $A_c = \{t_0 = t_0 + k\Delta_t, t_1 = t_0 + (k+1)\Delta_t\}$, $P_c = \{B_c \to t_0 A_1,\ A_1 \to t_1 A_2\}$. Эти правила можно представить следующими формулами нечеткого исчисления, изложенного в работе [5], каждая из которых задает свойство, относящееся только к одному сигналу (показаны только правила для сигнала $X(t)$),

$(t\ есть\ t_0) \supset (b_x\ есть\ B_x)$,

$(b_x\ есть\ B_x) \supset ((X(t_0)\ есть\ Слева\_X(t_0)) \vee (X(t_0)\ есть\ B\_центре\_X(t_0)) \vee (X(t_0)\ есть\ Справа\_X(t_0)))$,

$(b_x\ есть\ B_x) \supset k = 1$,

$(b_x\ есть\ B_x)\&(t\ есть\ t_0) \supset (b_x\ есть\ b_1)$,

$(b_x\ есть\ b_1) \supset ((X(t_1)\ есть\ Слева\_X(t_1)) \vee (X(t_1)\ есть\ B\_центре\_X(t_1)) \vee (X(t_1)\ есть\ Справа\_X(t_1)))$,

$(b_x\ есть\ b_1)\&\ (t\ есть\ t_1) \supset (b_x\ есть\ b_2)$,

$(b_x\ есть\ b_2) \supset k = k + 1$,

$(b_x\ есть\ b_2) \supset b_2\ есть\ B_x$,

$(b_x\ есть\ b_2) \supset k = k + 1$.

Кроме этих свойств, могут быть известны свойства, относящиеся к обоим сигналам, например следующие:

$(X(t)\ есть\ Слева\_X(t) \supset C(t)\ есть\ Справа\_C(t))$,

$(X(t)\ есть\ B\_центре\_X(t) \supset C(t)\ есть\ B\_центре\_C(t))$,

$(X(t)\ есть\ Справа\_X(t) \supset C(t)\ есть\ Слева\_C(t))$.

Все перечисленные правила для отдельных сигналов и потока из двух сигналов образуют онтологию. Правила вывода, рассмотренные в работе [5], вместе с онтологией образуют теорию. Используя теорию, требуется найти ответ на запрос, содержательная суть которого следующая: “Если известна онтология, то где находится объект в момент времени t?”. Формально этот запрос с использованием модальностей может быть задан следующей формулой:

$(A_1X(t)\ есть\ Слева\_X(t)\&\ (A_2X(t)\ есть\ \ B\_центре\_X(t)\& A_3X(t)\ есть\ Справа\_X(t)))\&(n_1\ C(t)\ есть\ Слева\_C(t)\&\ (n_2C(t)\ есть\ B\_центре\_C(t)\&\ n_3\ C(t)\ есть\ Справа\_C(t)))?$

В этом выражении надо определить значения модальностей A_1, A_2, A_3, характеризующих расположение объекта по оси X и модальностей n_1, n_2, n_3, характеризующих расположение объекта по оси C.

Выводы. Распределенные беспроводные и комбинированные сети интеллектуальных датчиков и связанные с ними системы сбора, обработки и передачи информации становятся новой областью исследова-

ний. В статье были рассмотрены принципы представления и использования информации, собираемой с датчиков, для анализа свойств потоков и свойств мультиагентной системы, обрабатывающей эти потоки. Проверка свойств осуществляется путем запросов к базам знаний (онтологиям). Онтологии хранятся не обязательно централизованно. Они могут быть распределены по агентам, каждый из которых отвечает за свою часть анализа свойств потоков или мультиагентной системы. Для формирования онтологий предлагается использовать нечеткое модальное ситуационное исчисление. Все свойства представляются на языке этого исчисления как теория, ответ на запросы является выводом в этом исчислении.

Изложенные принципы были частично апробированы в описанной мультиагентной системе анализа потоков.

Настоящая работа частично была выполнена по теме "Интегрированный телекоммуникационный узел сбора, обработки и анализа телеметрической информации для удаленного предоставления результатов анализа службам городского хозяйства" в рамках государственного контракта на выполнение работ по федеральной целевой программе "Интеграция науки и высшего образования России на 2002–2006 годы".

СПИСОК ЛИТЕРАТУРЫ

1. Estrin D., Govindan R., Heidemann J., Kumar S. Next century challenges: Scalable coordination in sensor networks. ASM MOBICOM, 1999.
2. Bonnet P., Gehrke J. E., Seshadri P. Towards sensor database systems. ASM Mobile Data Management. 2001.
3. Akyildiz F., Su W., Sankarasubramaniam Y., Cayirci E. A servey on Sensor Networks / Ieee Communication Magazine, August 2002, p. 102–114.
4. Девятков В. В., Мак-Кормик М. Формирование алгоритмов управления по элементам их поведения / Вестник МГТУ им. Н.Э. Баумана. Серия "Приборостроение". – 1995. – № 1. – С. 14–23.
5. Девятков В. В., Румбешт В. В. Динамическое прогнозирование и распознавание ситуации на основе аппарата нечеткой логики и конечно-автоматной модели представления временных последовательностей / Вестник МГТУ им. Н.Э. Баумана. Серия "Приборостроение". – 1995. – № 1. – С. 66–74.
6. Девятков В. В. Онтологии и их применение / Программные продукты и системы. – 2000. – № 3.
7. Девятков В. В. Системы искусственного интеллекта. -М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 350 с.
8. Девятков В. В. Румбешт В. В. Нечеткое модальное ситуационное исчисление для анализа сложных объектов / Вестник МГТУ им. Н.Э. Баумана. Серия "Приборостроение". – 2001. – № 3. – С. 3–22.
9. Девятков В. В. Мультиагентное иерархическое распознавание на основе нечеткого ситуационного исчисления // Тр. ИПУ РАН. – Т.XX. – 2003. – С. 78–99.

10. Девяткова В. В. Нечеткий логический анализ телеметрической информации // Труды шестого международного симпозиума “Интеллектуальные системы” INTELS’2004. – М.: РУСАКИ, 2004. – С. 72–79.

11. Девяткова В. В. Мультиагентная архитектура обслуживания запросов к базам данных удаленного восприятия. Проблемы управления и моделирования в сложных системах // Тр. VI Международной конференции: Под. ред. акад. Е.А. Федосова, акад. Н.А. Кузнецова, проф. В.А. Виттиха. – Самара: Самарский научный центр РАН, 2004. – С. 282–293.

Статья поступила в редакцию 26.05.2005
