Моделирование и анализ информационных систем. Т. 26, №4 (2019), с. 534–549
Modeling and Analysis of Information Systems. Vol. 26, No 4 (2019), pp. 534–549

Theory of data

DOI: 10.18255/1818-1015-2019-4-534-549

УДК 004.822, 681.51

Методы специализации онтологии процессов, ориентированной на верификацию

Гаранина Н. О., Анureев И. С., Боровикова О. И., Зюбин В. Е.

Поступила в редакцию 11 сентября 2019
После доработки 16 ноября 2019
Принята к публикации 27 ноября 2019

Аннотация. Удобная для пользователя формальная спецификация и верификация параллельных и распределённых систем, принадлежащих различным предметным областям, таким как системы автоматического управления, телекоммуникации, бизнес-процессы, являются активными темами исследований в силу их практической значимости. В этой статье мы представляем методы разработки специализированных ориентированных на верификацию онтологий процессов, которые используются для описания параллельных и распределенных систем предметных областей. Одним из преимуществ таких онтологий является их формальная семантика, которая делает возможной формальную верификацию описанных систем. Наш метод основан на абстрактной онтологии процессов, ориентированной на верификацию. Мы используем два метода специализации абстрактной онтологии процессов. Декларативный метод с помощью специализации классов исходной онтологии, введения новых декларативных классов, а также системы аксиом задаёт ограничения для классов и отношений абстрактной онтологии. Конструктивный метод использует техники семантической разметки и сопоставления с образцом, чтобы связать понятия предметной области с классами абстрактной онтологии процессов. Мы даём подробные онтологические спецификации этих техник. Наши методы сохраняют формальную семантику исходной онтологии процессов и, следовательно, возможность применения формальных методов верификации к специализированным онтологиям процессов. Мы показываем, что конструктивный метод является уточнением декларативного метода. Построение онтологии типовых элементов систем автоматического управления иллюстрирует наши методы: разработано декларативное описание классов и ограничений специализированной онтологии в системе Protégé на языке OWL с использованием правил вывода на языке SWRL и построена система шаблонов семантической разметки, которая реализует типовые элементы систем автоматического управления.

Ключевые слова: онтология процессов, специализация, аксиомы онтологии, сопоставление с образцом, семантическая разметка, системы автоматического управления, верификация

Для цитирования: Гаранина Н. О., Анureев И. С., Боровикова О. И., Зюбин В. Е., "Методы специализации онтологии процессов, ориентированной на верификацию", *Моделирование и анализ информационных систем*, **26**:4 (2019), 534–549.

Об авторах:
Гаранина Наталья Олеговна, orcid.org/0000-0001-9734-3808, канд. физ.-мат. наук, с.н.с.,
Институт систем информатики им. А.П. Ершова СО РАН, Институт автоматики и электрометрии СО РАН,
пр. Акад. Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: garanina@iis.nsk.su

Анureев Игорь Сергеевич, orcid.org/0000-0001-9574-128X, канд. физ.-мат. наук, с.н.с.,
Институт систем информатики им. А.П. Ершова СО РАН, Институт автоматики и электрометрии СО РАН,
пр. Акад. Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: anureev@iis.nsk.su

Боровикова Олеся Игнатьевна, orcid.org/0000-0001-5456-8513, м.н.с.
Институт систем информатики им. А.П. Ершова СО РАН,
пр. акад. Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: olesya@iis.nsk.su

Зюбин Владимир Евгеньевич, orcid.org/0000-0002-8198-3197, доктор тех. наук, зав. лаб.,
Институт автоматики и электрометрии СО РАН,
пр. Акад. Коптюга, 1, г. Новосибирск, 630090 Россия, e-mail: zyubin@iae.nsk.su

Благодарности:
Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов № 17-07-01600 и № 19-07-00762, а также в рамках темы госзадания ИАиЭ СО РАН № АААА-А17-11706061006-6.

Введение

Наша долгосрочная цель – разработка комплексного подхода к поддержке формальной верификации параллельных систем для обеспечения их качества. Решение включает методы извлечения формальных моделей и свойств параллельных систем из текстов технической документации, а также инструменты для ручного исправления извлеченной информации и обогащения ее новыми сущностями.

Проектируемый нами интеллектуальный комплекс для поддержки формальной верификации параллельных систем автоматически извлекает и генерирует системные требования. Мы разработали онтологию требований *SO* как первый шаг к созданию этого комплекса [3]. Другим ключевым компонентом комплекса является онтология процессов *PO* для параллельных/распределенных систем [4]. Содержание этих онтологий, то есть множества экземпляров их классов, является онтологическим описанием некоторой параллельной/распределенной системы и требований к ней. Эти описания могут быть получены из технической документации с помощью нашей системы извлечения информации, которая использует структуру онтологий в правилах анализа документации и для хранения результата извлечения [5]. В силу неоднозначности естественного языка и недостаточной формализованности технической документации, может потребоваться корректирование извлечённой информации специальными редакторами. С помощью этих редакторов также можно, независимо от модуля извлечения информации, разрабатывать онтологические описания систем параллельных/распределенных процессов и требований. Эти описания являются основой для формальной верификации таких систем процессов, поскольку для онтологии требований и онтологии процессов можно задать формальную семантику. В частности, в работе [3] семантика онтологии *SO* задана как формулы темпоральной логики LTL [6], а в работе [4] семантика онтологии *PO* задана как помеченная система переходов. Для верификации системы процессов необходимо определить подходящий верификатор (например, инструмент проверки моделей) с учетом формальной семантики представления требований и процессов. Если он существует, мы транслируем онтологическое описание системы в язык спецификации модели выбранного верификатора, а описание требований переводится на язык спецификации свойств (обычно этот язык является темпоральной логикой). Работа с требованиями в нашей системе, кроме использования формальной семантики, включает также их представление на естественном языке и в графической форме.

Методы, предложенные в этой статье, связаны как с задачей извлечения информации о параллельной/распределенной системе процессов из технической документации, так и с разработкой редактора для построения и корректирования онтологического описания таких систем. В этих задачах используется онтология

процессов PO, которая описывает параллельные системы как состоящие из взаимодействующих параллельных процессов, характеризуемых локальными и разделяемыми переменными, каналами для обмена сообщениями, действиями обработки переменных и содержания каналов. Однако при рассмотрении конкретных предметных областей выразительные возможности онтологии PO оказываются слишком абстрактными, чтобы, с одной стороны, задавать корректные правила извлечения информации из технической документации, и с другой – чтобы описания системы процессов были исчерпывающими и понятными инженерам требований и разработчикам программных систем в заданных предметных областях.

Поскольку наш комплекс поддержки верификации может быть использован в разных предметных областях, необходимо разработать методы специализации абстрактной онтологии процессов PO для конкретных предметных областей, чтобы создавать экземпляры процессов, специфичные для предметной области, которые имеют переменные, каналы и действия, соответствующие их предметной специализации. Например, в системе автоматического управления процесс, задающий датчик, должен быть обязательно связан каналом связи по меньшей мере с одним процессом, задающим контроллер. В этой статье мы предлагаем два метода специализации абстрактной онтологии процессов PO (раздел 1.).

Декларативный метод, описанный в разделе 2., основан на добавлении к онтологии PO классов, специфических для выбранной предметной области, которые наследуют классы исходной онтологии, новых декларативных классов, не наследующих классы исходной онтологии, но необходимых для задания специфических ограничений отношений между наследующими классами, и аксиом, ограничивающих значения атрибутов этих классов и отношения между ними. Наследование здесь определено как расширение исходных классов декларативными предметно-специфическими атрибутами (к которым также относится и имя нового класса). В силу декларативности такого наследования, формальная семантика системы процессов, описанных новыми классами, не изменяется, поскольку она задаётся только в терминах классов онтологии PO, а новые декларативные классы не используются. Новые аксиомы ограничивают как значения наследуемых, так и новых декларативных атрибутов, что также не влияет на формальную семантику. Поэтому такая специализация PO для заданной предметной области сохраняет возможность формальной верификации системы процессов специализированной онтологии. Итак, специализированная онтология процессов отличается от исходной онтологии процессов PO декларативными предметно-специфическими атрибутами классов (включая ненаследующие классы) и набором аксиом и правил, которые определяют специфичные ограничения на атрибуты классов и отношения. Аксиомы и правила могут быть использованы как для настройки на предметную область системы извлечения информации из технической документации, так и для проверки целостности и согласованности содержания онтологии. В нашем случае онтология служит для представления параллельной/распределенной системы, поэтому целостность и согласованность означают, что экземпляры процессов соответствуют процессам предметной области, то есть имеют все необходимые переменные, каналы и действия.

Однако декларативный метод не даёт способа задания содержания специализированной онтологии. Поэтому мы предлагаем конструктивный метод, который может быть использован для создания предметно-ориентированного содержания он-

тологии процессов с использованием новой онтологии предметно-ориентированных шаблонов. Построение этого содержания включает в себя несколько этапов. На первом этапе мы используем абстрактную онтологию процессов PO для создания новой онтологии семантически размеченных процессов, за счёт обогащения её классов атрибутами семантической разметки, содержащими строковое описание терминов предметной области. Эта новая онтология дана в разделе 3. Затем множество экземпляров предметно-ориентированных процессов может быть определено с использованием ориентированной на процессы онтологии семантической разметки, описанной в разделе 4. Экземпляры этой онтологии определяют правила, с помощью которых задается набор экземпляров специализированных процессов. Декларативный метод специализированной онтологии процессов подходит для настройки процесса извлечения данных о параллельной системе и для проверки корректности описаний уже созданных или извлеченных систем. Для создания новой специализированной онтологии процессов и ее наполнения лучше использовать конструктивный метод, поскольку он позволяет на основе шаблонов строить экземпляры новой онтологии.

Мы иллюстрируем наши методы на примере предметной области систем автоматического управления (САУ). Для иллюстрации декларативного метода разработаны описания классов типовых элементов САУ, а также их свойств и аксиом, средствами языка OWL (Web Ontology Language [11]) в редакторе Protégé [12] с использованием языка правил SWRL (Semantic Web Rule Language [13]). Эти правила задают условия, которые обеспечивают в Protégé проверку корректности онтологического описания специализированных процессов с помощью машины логического вывода Hermit [9]. Разработка онтологии процессов для этой области особенно важна, потому что удобная для пользователя формальная спецификация и верификация систем автоматического управления и, в целом, кибер-физических систем, имеют важное практическое значение.

1. Онтология процессов

Мы рассматриваем *онтологию* как структуру, которая включает в себя следующие элементы: (1) конечное непустое множество *классов*, (2) конечный непустой набор *атрибутов данных и атрибутов отношений*, и (3) конечный непустой набор *доменов атрибутов данных*. Каждый класс определяется набором атрибутов. *Класс c является подклассом* другого класса C ($c < C$ наследует родительский класс), если все атрибуты класса-родителя принадлежат также классу-наследнику. Атрибуты данных получают значения из доменов, а значения атрибутов отношений являются экземплярами классов. *Экземпляр класса* определяется набором значений атрибутов для этого класса. *Содержание* онтологии — это набор экземпляров ее классов. *Аксиомы и правила* ограничивают значения атрибутов классов.

Содержание онтологии процессов PO представляет онтологическое описание параллельной/распределенной системы. Мы рассматриваем такую систему как множество взаимодействующих процессов. Процессы (описываемые классом $Process$) характеризуются набором локальных и разделяемых переменных, списком действий над этими переменными, которые меняют их значения, списком каналов для взаимодействия, а также списком коммуникационных действий по отправке сообщений.

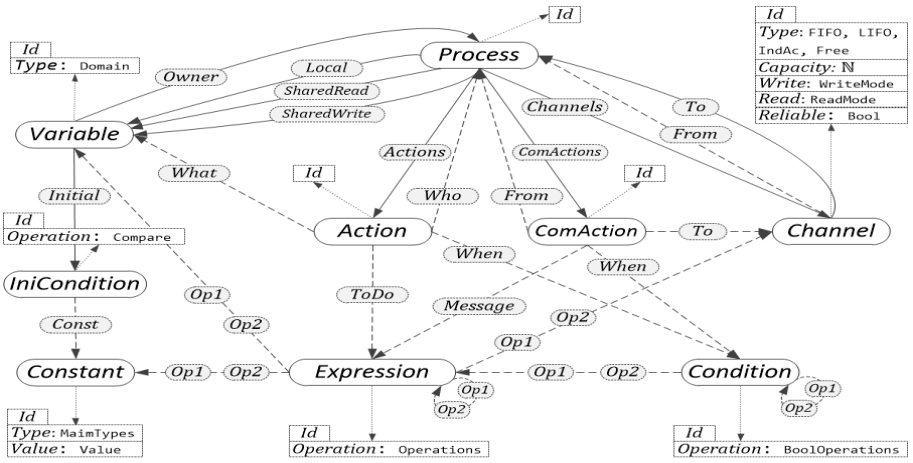

Рис. 1. Онтология процессов
Fig. 1. The Process Ontology

Переменные (класс *Variable*) и константы (класс *Constant*) процессов принимают значения в областях базовых типов. В зависимости от формальной семантики, приписываемой онтологии процессов, базовые типы и операции, определённые для переменных этих типов, могут различаться. В частности, при задании семантики как конечной системы помеченных переходов [4] в качестве базовых типов мы рассматриваем булевский тип, конечные подмножества целых и строк для перечислимых типов, а также конечные типы, производные из перечисленных. Далее в статье мы считаем, что предметные области таковы, что их системы процессов допускают упомянутую формальную семантику, т.е. не используют, например, бесконечные множества действительных чисел. Начальные значения переменных могут определяться сравнением с константами. Действия процессов (класс *Action*) изменяют значения переменных. Условие срабатывания для каждого действия определяется как охранное условие (класс *Condition*) относительно значений переменных и содержимого каналов. Процессы могут посылать сообщения через каналы (класс *Channel*) при выполнении охранных условий (класс *Condition*). Каналы коммуникации характеризуются типом чтения сообщений, ёмкостью, режимами записи/чтения и надежностью. Отметим, что при выборе формальной семантики онтологии как конечной системы помеченных переходов, ёмкость каналов должна быть ограничена. Рисунок 1 представляет онтологию процессов. Классы отображаются как белые овалы. Отношения между классами показаны пунктирными стрелками с именами в серых овалах. Эти стрелки являются сплошными, если отношение “один ко многим”, и пунктирными, если отношение “один к одному”. Атрибуты данных классов, помещённые в штрих-пунктирные прямоугольники, связаны с соответствующими классами штрих-пунктирными стрелками. Универсальные классы онтологии *PO* абстрагируются от особенностей предметной области. В следующем разделе мы определим метод специализации и расширения онтологии посредством добавления специализирующих классов и ограничивающих аксиом для предметной области.

2. Специализированная онтология процессов

Специализация онтологии – это построение иерархии классов [7]. В случае специализации онтологии процессов мы строим новую специализированную онтологию, расширяя исходную онтологию следующими элементами. Во-первых, необходимо задать множество классов для описания специфических для предметной области процессов, переменных, констант, каналов, действий процессов и т.п., которые наследуют классы исходной онтологии. При этом дополнительные атрибуты класса-наследника являются чисто декларативными, т.е. связывая абстрактные понятия онтологии процессов с понятиями предметной области, они не оказывают влияния на формальную семантику (в большинстве случаев дополнительные атрибуты отсутствуют). Например, для предметной области автоматических систем управления новый атрибут $SpecSen$ в описании процесса-датчика служит для спецификации вида и диапазона значений наблюдаемой им величины. Во-вторых, определяется множество классов, не наследующих классы онтологии PO, но необходимых для описания онтологических ограничений предметной области. Например, экземпляры класса $PhysicalQuantity$ является значениями атрибута $SpecSen$ процесса-датчика и могут служить для проверки соответствия типа его наблюдаемой величины. И, наконец, необходимо описать аксиомы и правила, ограничивающие значения атрибутов новых классов. В частности, явным образом нужно задать, что спецификация вида и диапазона значений процесса-датчика и спецификация вида и диапазона значений наблюдаемой им величины должны совпадать.

Формально специализация онтологии процессов PO описывается следующим образом. Обозначим специализированную онтологию как $SpecPO$, множество аксиом онтологии O как $Axioms(O)$, множество классов онтологии O как $Classes(O)$, множество атрибутов класса c как Atr_c, где $c \in O$ и $O \in \{PO, SpecPO\}$. Для онтологии O множество атрибутов классов, значения которых используются при описании её формальной семантики, обозначим как $FormSem(O)$. Отметим, что $FormSem(PO) = \cup_{c \in Classes(O)}(Atr_c \setminus Id)$, т.е. идентификаторы не влияют на формальную семантику (см. рис. 1 и [4]). Тогда:

1) $Classes(SpecPO) = InhCls \cup SpecCls$, где $\forall c \in InhCls\ \exists C \in Classes(PO) : c < C$ и $\forall c \in SpecCls\ \nexists C \in Classes(PO) : c < C$. При этом $FormSem(SpecPO) \subseteq FormSem(PO)$.
2) $Axioms(SpecPO) = Axioms(PO) \cup SpecAxs$, при этом $\forall c \in Classes(SpecPO)\ \exists a \in SpecAxs : a(uses)c$, где $a(uses)c$ означает, что в описании аксиомы a упоминается класс c. Аксиомы могут накладывать ограничения на количество значений атрибутов и на их возможный диапазон (включая точное соответствие). В случае атрибутов отношений этот диапазон определяет возможные классы (включая ограничения на их атрибуты), с которым данный класс связан отношениями.

Таким образом, классы исходной онтологии служат паттернами онтологического проектирования [7] для создания специализированных онтологий процессов, что иллюстрируется примером онтологии для типовых элементов систем автоматического управления в разделе 5. Далее мы опишем конструктивный подход к специализации абстрактной онтологии процессов для предметных областей.

3. Семантически-размеченная онтология процессов

В этом разделе мы формально определим наш метод семантической разметки онтологии процессов. Эта разметка используется для сопоставления абстрактных процессов PO со специфическими процессами выбранной предметной области. Разметка заключается в добавлении к классам онтологии PO специальных семантических меток и атрибутов, связывающих их с понятиями предметной области. Обогащённые классы вместе с несколькими вспомогательными классами образуют новую семантически-размеченную онтологию (онтологию $SMPO$). Экземпляры предметно-ориентированных процессов могут конструироваться, используя онтологию $SMPO$ и процесс-ориентированную онтологию шаблонов семантической разметки (онтологию $POSMPO$), описанную в следующем разделе.

Онтология $SMPO$ содержит домены $Classes$, $Domains$, $Types$, $Values$, $SLabel$ и $SAttribute$, классы $AValue$, $Element$, T и $Element\_T$ для каждого $T \in Domains$.

Домены $Classes$ и $Domains$ включают имена классов и доменов онтологии PO, соответственно. Домен $Types = Classes \cup Domains$ включает все имена онтологии PO. Домен $Values$ включает все значения атрибутов из онтологии PO: $Values = \cup_{T \in Types} Val(T)$, где $Val(T)$ — множество значений для T, которые являются экземплярами $T \in Classes$ или соответствующими значениями для $T \in Domains$.

Домен $SLabel$ — это конечное множество семантических меток, представленных строками. Метки специфицируют информацию, связанную с экземплярами классов онтологии PO. Эта информация может быть о предметной области (“sensor” или “pressure”) или особенностях моделирующих процессов (“periodic start”).

Домен $SAttribute$ — это конечное множество семантических атрибутов (S-атрибутов), представленных строками. Подобно меткам, S-атрибуты специфицируют информацию о предметной области, связанную с экземплярами классов онтологии PO. Разница состоит в том, что семантические атрибуты имеют значения (“100”, “$true$” или “экземпляр класса $Controller$”).

Класс $AValue$ имеет два однозначных атрибута: $Attribute$ со значениями в домене $SAttribute$ и $Value$ со значениями в $Values$, которые специфицируют имя S-атрибута и его значение. Его экземпляры называются атрибутными значениями.

Класс T онтологии $SMPO$ — это класс T онтологии PO, обогащённый двумя многозначными атрибутами: $SLabels$ со значениями в $SLabel$ и $SAattributes$ со значениями в $AValue$. Эти атрибуты семантически размечают экземпляры классов онтологии PO, специализируя их для конкретной предметной области, и называются размечающими атрибутами (M-атрибутами). Атрибут $SLabels$ специфицирует множество семантических меток. Атрибут $SAttributes$ специфицирует множество S-атрибутов с их значениями с естественным ограничением, что любой S-атрибут может входить в его значение не более одного раза. Атрибуты класса T, не являющиеся M-атрибутами, называются *базовыми атрибутами*.

Класс $Element\_T$ имеет только M-атрибуты и однозначный атрибут $Value$ со значением в T, специфицирующий значение из домена T. Таким образом, в онтологии $SMPO$ значения доменов онтологии PO могут включать информацию о предметной области, задаваемую M-атрибутами.

Класс $Element$ имеет только M-атрибуты. Экземпляры этого класса описывают

информацию о предметной области, которая не выражается естественным образом как специализация классов онтологии PO.

Проиллюстрируем добавление информации о предметной области к элементам онтологии PO на примере датчика, измеряющего температуру в градусах Цельсия в диапазоне от 0 до 1000. Такой датчик задаётся следующим экземпляром класса `Process` онтологии $SMPO$:

```
Process(BAVs, SLabels:{"sensor"},
SAttributes:{AValue (Attribute:"Dimension", Value:"temperature"),
AValue(Attribute:"unit", Value:"Celsius"),
AValue(Attribute:"range",
Value:Element(SLabels:{"range"},
SAttributes:{AValue(Attribute:"left", Value:"0"),
AValue(Attribute:"right", Value:"1000")})})
```

Листинг 1. Экземпляр датчика
Listing 1. The Instance of a Sensor

Здесь элемент $T(A_1 : V_1, \ldots, A_n : V_n)$ обозначает экземпляр класса T со значениями V_1, ..., V_n атрибутов A_1, ..., A_n, множество $\{V_1, \ldots, V_n\}$ перечисляет значения многозначного атрибута, и $BAVs$ — базовые атрибуты из онтологии PO.

Таким образом, онтология $SMPO$ позволяет описывать экземпляры понятий предметных областей в контексте онтологии PO. Однако, она не позволяет описывать ограничения на эти экземпляры. В следующем разделе мы определим онтологию $POSMPO$, которая накладывает ограничения на семантическую разметку, и тем самым на экземпляры понятий предметной области, специфицируемые с помощью этой разметки. Эта онтология определяет понятия предметной области, используя шаблоны для накладывания ограничений в экземплярах онтологии $SMPO$ на местность и значения их атрибутов.

4. Процесс-ориентированная онтология шаблонов семантической разметки

Онтология $POSMPO$ включает все домены и классы онтологии $SMPO$, дополнительные домены $AMatchSizes$ и $AMatchOperations$, а также класс $AMatch$.

Пусть n, m — неотрицательные целые числа. Домен $AMatchArities = \{$"m", "m|0","$m-k$","$m-k|0$","$m-$","$m-|0$","$-m$"$\}$ описывает ограничения на число значений атрибутов классов онтологии $SMPO$.

Домен $AMatchOperations = \{$" = "," < "," <= ","! = "," > "," => ","in", "$oneof$","all"$\}$ описывает множество операций сопоставления. Они сопоставляют значение атрибута экземпляра класса онтологии $SMPO$ со значением соответствующего атрибута экземпляра класса онтологии $POSMPO$. Множество значений этого домена может расширяться для конкретной предметной области.

Экземпляры классов онтологии $POSMPO$ называются шаблонами семантической разметки. Каждый шаблон специфицирует множество экземпляров классов онтологии $SMPO$, сопоставляемых с этим шаблоном. Класс T онтологии $POSMPO$ имеет те же самые атрибуты, как и класс T онтологии $SMPO$, но при этом их значения содержатся в $AMatch$. Класс $AMatch$ специфицирует правила для сопоставления значений атрибутов онтологии $SMPO$ с шаблонами для них и имеет

три атрибута. Однозначный атрибут Ar со значениями в $AMatchArities$ ограничивает число сопоставляемых значений. Однозначный атрибут Op со значениями в $AMatchOperations$ определяет операцию сопоставления. Многозначный атрибут Pat со значениями в $Values[SMPO]$ задаёт шаблоны для значений атрибутов.

Пусть $V.A$ обозначает значение атрибута A экземпляра V и $|S|$ обозначает мощность множества S. Экземпляр V класса T онтологии $SMPO$ сопоставляется с шаблоном P класса T онтологии $POSMPO$ тогда и только тогда, когда для каждого атрибута A шаблона P такого, что $P.A = AMatch(Ar : R, Op : O, Pat : \{V_1, \ldots, V_n\})$, выполнены следующие свойства:

– Если $R =$ "m", то $|V.A| = m$.
– Если $R =$ "$m-$", то $|V.A| \geq m$.
– Если $R =$ "$-m$", то $|V.A| \leq m$.
– Если $R =$ "$m-k$", то $k \leq |V.A| \leq m$.
– Если $R =$ "$m|0$", то $|V.A| = m$, или $|V.A| = 0$.
– Если $R =$ "$m-|0$", то $|V.A| \geq m$, или $|V.A| = 0$.
– Если $R =$ "$m-k|0$", то $n \leq |V.A| \leq m$, или $|V.A| = 0$.
– Если $O =$ " = ", то $n = 1$, и $V' = V_1$ для каждого $V' \in V.A$. Случаи, когда $O \in \{$" = "," < "," <= ","! = "," > "," > "$\}$ определяются аналогично. Эти случаи ограничивают сравниваемые значения атрибутов.
– Если $O =$ "in", то $n = 1$, и $V' \in V_1$ для каждого $V' \in V.A$. Этот случай определяет принадлежность значений атрибутов.
– Если $O =$ "$oneof$", то V сопоставляется с $upd(P, A, V_i)$ для некоторого $1 \leq i \leq n$, где $upd(Q, B, U)$ обозначает результат присваивания значения U атрибуту B экземпляра Q. Этот случай выбирает некоторое значение шаблона для атрибутов.
– Если $O =$ "all", то имеются S_1, ..., S_n такие, что $V.A = \{S_1, \ldots, S_n\}$, $S_i \cap S_j = \emptyset$ для $S_i \neq \emptyset$, $S_j \neq \emptyset$, и $upd(V, A, S_i)$ сопоставляется с $upd(P, A, V_i)$ для каждого $1 \leq i \leq n$. Этот случай выбирает все значения шаблона для атрибутов.
– Если O не определено, и $A \neq SLabels$, или $T = AValue$, и $A \neq Attribute$, то V' сопоставляется с $P.A$ для каждого $V' \in V.A$. Этот случай сводит сопоставление множества значений атрибутов к сопоставлению отдельных значений атрибутов. Оставшиеся случаи — специальные случаи для классов $SLabels$, $AValue$ и $SAttributes$.
– Если O не определено и $A = SLabels$ то $P.SLabels \subseteq V.SLabels$.
– Если O не определено, $T = AValue$, и $A = Attribute$ то $n = 1$, и $V.A = V_1$.
– Если O не определено и $A = SAttributes$ и $sattributes(P) = sattributes(V)$, где $attributes(i)$ — множество имен S-атрибутов в экземпляре i некоторого класса.

Мы определили онтологию $POSMPO$, которая комбинирует онтологию PO с описаниями понятий предметной области. Конкретное множество экземпляров этой онтологии задает правила для построения соответствующей предметно-ориентированной онтологии процессов. Классы и домены онтологии $POSMPO$ задают язык для конструктивного использования аксиом, которые ограничивают абстрактные классы онтологии PO относительно предметной области, так как эти аксиомы могут специфицировать только число значений атрибутов и их области изменения.

Конструктивный метод, как и декларативный, сохраняет формальную семантику исходной онтологии PO, поскольку он не изменяет базовых атрибутов классов этой онтологии, в терминах которых описывается формальная операционная семан-

тика онтологии PO [4], а только добавляет новые семантические атрибуты, отображающие экземпляры классов онтологии PO на понятия предметной области.

Декларативный и конструктивный методы специализации связаны следующим образом: в декларативном методе имена новых классов, наследующих классы онтологии процессов, а также имена их новых атрибутов согласуются со строковыми значениями доменов $SLabel$ и $SAttribute$. Поскольку декларативный метод задаёт семейство систем процессов, а конструктивный позволяет строить конкретную систему, то значения атрибутов, соответствующие строкам из домена $SAttributes$, согласуются только с типичными для этого семейства ограничениями на значения атрибутов. Классы декларативной специализации, не наследующие классы исходной онтологии, согласуются с экземплярами класса $Element$ в той мере, в которой значения их семантических меток описывают общие ограничения семейства систем. Декларативные ограничения специализации на значения атрибутов, т.е. аксиомы и правила, согласуются с экземплярами класса $AMatch$ в той мере, в которой значения их атрибутов описывают общие ограничения семейства систем.

В следующем разделе мы опишем типовые элементы систем автоматического управления (САУ) посредством метода декларативной специализации онтологии процессов и сконструируем эти элементы, используя классы онтологии $POSMPO$.

5. Специализированная онтология процессов для типовых элементов систем автоматического управления

В этом разделе мы определим классы, аксиомы и шаблоны семантической разметки для типовых элементов систем автоматического управления, таких как датчики, контроллеры, исполнительные устройства (актуаторы) и объект управления.

Простые и сложные датчики характеризуются следующими ограничениями. Датчики должны считывать наблюдаемые значения из переменных, разделяемых с объектом управления, и не могут их изменять. Для датчиков должны быть заданы коммуникационные действия для отправки сообщений контроллерам через исходящие каналы. У каждого датчика должна быть хотя бы одна разделяемая переменная и связь с хотя бы одним контроллером. У простых датчиков нет локальных переменных и действий. Они могут наблюдать ровно одну разделяемую переменную и отправлять наблюдаемое значение контроллерам без изменений. Сложные датчики могут трансформировать наблюдаемые значения нескольких разделяемых переменных для передачи контроллеру, используя локальные переменные и действия. Для датчиков должны быть определены физические величины, которые они обрабатывают. Эти величины характеризуются размерами (`"temperature"`, `"pressure"`, `"density"`, etc.), единицами измерения (`"centimeter"`, `"kilogram"`, `"volt"`, etc.) и диапазонами. Эти ограничения и сопутствующие понятия онтологии определяются декларативно классами и аксиомами из листинга 2 и конструктивно — шаблонами из листинга 3.

```
NewClass Sensor: Process
NewAttribute:
SpecSen: PhQuantity;

NewClass SimSensor: Sensor;

NewClass PhQuantity
Attributes:
Dimension: String;
Unit: String;
Range: {Int,Int};

SimSensor: local   exactly 0 Variable
shared  exactly 1 Variable
actions exactly 0 Action
specSen exactly 1 PhQuantity

Sensor(?s)^Variable(?v)^Shared(?s,?v) -> ControlledObject(?o)^Shared(?o,?v)
Sensor(?s)^Channel(?ch)^Channels(?s,?ch)^ComAction(?ca)^From(?ca,?s)^To(?ca,?ch) ->
    Controller(?c)^Channels(?c,?ch)
Sensor(?s)^Variable(?v)^Shared(?s,?v)^PhQuantity(?s,?q) -> Type(?v,?q)
```

Листинг 2. Декларативное описание датчиков
Listing 2. Declarative Description of Sensors

В листинге выше описаны три новых класса, два из которых наследуют класс *Process* исходной онтологии. Здесь и далее декларативные ограничения на количество значений атрибутов приведены в той форме, в какой они отображаются в редакторе онтологий Protégé, так как формальное описание на языке OWL выглядит громоздким. Правила, ограничивающие использование каналов, описаны более формально на языке задания правил SWRL.

```
Process( // Simple sensor
Local:AMatch("0"),
SharedRead:AMatch("1", Variable(SLabels:{"Observed value"})),
SharedWrite:AMatch("0"),
Actions:AMatch("0"),
Channels:AMatch("1-",
Channel(SLabels:{"Channel from sensor to controller"})),
ComActs:AMatch("1-", ComAction(SLabels:{"Sending observed value from simple sensor"})),
SLabels:{"Simple sensor"},
SAttributes: {AValue("Physical quantity", Element(SLabels:{"Physical quantity"})})

Element( // Physical quantity
SLabels:{"Physical quantity"},
SAttributes: {AValue("Dimension", AMatch(Op:"in", Pat:Dimension)),
AValue("Unit", AMatch(Op:"in", Pat:Unit)),
AValue("Range", Element{SLabels:{"Range"}})})

Element( // Range
SLabels:{"Range"},
SAttributes: {AValue("Left", AMatch(Op:"in", Pat:Float)),
AValue("Right", AMatch(Op:"in", Pat:Float))})

Variable( // Observed value
Users:AMatch(Op:"all",
Pat:{AMatch("1", Process(SLabels:{"Controlled Object"})),
AMatch("1-", Process(SLabels:{"Simple sensor"}))}),
SLabels:{"Observed value"},
SAttributes: {AValue("Physical quantity", Element(SLabels:{"Physical quantity"})})

Channel( // Channel from sensor to controller
From:AMatch("1", "oneof", {Process(Slabels:{"Simple sensor"}),
Process(Slabels:{"Complex sensor"})}),
To:AMatch("1-", Process(SLabels:{"Controller"})),
Type:AMatch("1", "=", "FIFO"),
Capacity:AMatch("1", "=", 1),
Write:AMatch("1", "=", "Old"),
Read:AMatch("1", "=", "Keep"),
Reliable:AMatch("1", "=", "true"),
SLabels:{"Channel from sensor to controller"})

ComAction( // Sending observed value from simple sensor
From:AMatch("1", Process(Slabels:{"Simple sensor"})),
```

```
To:AMatch("1-", Channel(SLabels:{"Controller"})),
Message:AMatch("1", Expression(Op1:AMatch("1", Variable(SLabels:{"Observed value"}))))
SLabels:{"Sending observed value from simple sensor"})

Process( // Complex sensor
SharedRead:AMatch("1-", Variable(SLabels:{"Observed value"})),
SharedWrite:AMatch("0"),
Channels:AMatch("1-",
Variable(SLabels:{"Channel from sensor to controller"})),
ComActs:AMatch("1-", ComAction(SLabels:{"Sending message from complex sensor"})),
SLabels:{"Complex sensor"},
SAttributes: {AValue("Physical quantity", Element(SLabels:{"Physical quantity"})})

ComAction( // Sending message from complex sensor
From:AMatch("1", Process(Slabels:{"Complex sensor"})),
To:AMatch("1-", Channel(SLabels:{"Controller"})),
SLabels:{"Sending message from complex sensor"})
```

Листинг 3. Конструктивное описание датчиков
Listing 3. Constructive Description of Sensors

В этом и следующих листингах мы используем следующие сокращения: `SLabels:S` для `SLabels:AMatch(Value:S)`, `AMatch(R, O, P)` для `AMatch(Ar:R, op:O, Pat:P)`, где `R`, `O`, или `P` могут опускаться, `AMatch(R, O, P)` для `{AMatch(R, O, {P})}`, `AValue(A, V)` для `AValue(Attribute:A, Value:V)` и `T` для `AMatch(Op:"in", Pat:T)`.

Ограничения на *контроллеры*, *актуаторы* и *объекты управления* состоят в следующем. Контроллеры и актуаторы не должны иметь общих переменных. Контроллеры должны иметь выходные каналы, соединяющие их с другими контроллерами и актуаторами, и входные каналы, соединяющие их с датчиками и актуаторами. Актуаторы должны иметь выходные каналы, соединяющие их с контроллерами и объектом управления, и входные каналы, соединяющие их с контроллерами. Должен быть, по меньшей мере, один датчик и, по меньшей мере, один актуатор, соединенный с контроллером через входной и выходной каналы соответственно. Должен быть хотя бы один контроллер и единственный объект управления, связанный с актуатором через входной и выходной каналы соответственно. Объект управления должен быть связан с актуаторами входными каналами. С объектом управления должны быть связаны по крайней мере одна общая переменная, один датчик и один актуатор. Эти ограничения описываются декларативно классами и аксиомами из листинга 4 и конструктивно — шаблонами из листинга 5.

```
NewClass Controller: Process
NewClass Actuator: Process
NewClass Object: Process

Controller: shared exactly 0 Variable

Actuator: shared exactly 0 Variable

Object: shared  min 1 Variable
actions min 1 Action

Controller(?c)^Channel(?ch)^Channels(?c,?ch)^ComAction(?ca)^From(?ca,?c)^To(?ca,?ch)
-> Controller(?d)^Channels(?d,?ch)
Controller(?c)^Channel(?ch)^Channels(?c,?ch)^ComAction(?ca)^From(?ca,?c)^To(?ca,?ch)
-> Actuator(?a)^Channels(?a,?ch)
Controller(?c)^Channel(?ch)^Channels(?c,?ch)^ComAction(?ca)^From(?ca,?c)^To(?ca,?ch)^
Actuator(?a)^Channels(?a,?ch)
-> Channels(?a,?ch1)^ComAction(?ca1)^From(?ca1,?a)^To(?ca1,?ch1)^ Channels(?c,?ch1)
Controller(?c) -> Sensor(?s)^Channel(?ch)^Channels(?s,?ch)^
ComAction(?ca)^From(?ca,?s)^To(?ca,?ch)^Channels(?c,?ch)

Actuator(?a)^Channel(?ch)^Channels(?a,?ch)^ComAction(?ca)^From(?ca,?a)^To(?ca,?ch)
-> Controller(?c)^Channels(?c,?ch)
Actuator(?a)^Channel(?ch)^Channels(?a,?ch)^ComAction(?ca)^From(?ca,?a)^To(?ca,?ch)^
Controller(?c)^Channels(?c,?ch)
```

```
-> Channels(?c,?ch1)^ComAction(?ca1)^From(?ca1,?c)^To(?ca1,?ch1)^ Channels(?a,?ch1)
Actuator(?a) -> Object(?o)^Channel(?ch)^
Channels(?a,?ch)^Channels(?o,?ch)^ComAction(?ca)^From(?ca,?a)^To(?ca,?ch)

Object(?o)^Channel(?ch)^Channels(?o,?ch)
-> Actuator(?a)^Channels(?a,?ch)^ ComAction(?ca)^From(?ca,?a)^To(?ca,?ch)
Object(?o)^Variable(?v)^Shared(?o,?v) -> Sensor(?s)^Shared(?s,?v)
```

Листинг 4. Декларативное описание контроллеров, актуаторов и объектов
Listing 4. Declarative Description of Controllers, Actuators and Objects

В листинге выше перечислены три новых класса, наследующих класс *Process* исходной онтологии. Правила, задающие ограничения на каналы сообщений, оказываются довольно громоздкими, поскольку должны описывать как входные, так и выходные каналы. Декларативное описание классов и ограничений, специализирующих абстрактную онтологию процессов, даёт возможность уточнить детали методов извлечения информации, использующих онтологию для сохранения результата (например, [5]), а также проверить корректность данных о системах процессов, извлечённых из технической документации.

```
Process( // Controller
SharedRead:AMatch("0"), SharedWrite:AMatch("0"),
Channels:AMatch("all",
{AMatch("1-", Channel(SLabels:{"Channel from sensor to controller"}),
AMatch("0-", Channel(SLabels:{"Channel from actuator to controller"},
AMatch("1-", Channel(SLabels:{"Channel from controller to actuator"},
AMatch("0-", Channel(SLabels:{"Channel from controller to controller"})}),
ComActs:AMatch("1-", ComAction(SLabels:{"Sending message from  controller"})),
SLabels:{"Controller"})

Process( // Actuator
SharedRead:AMatch("0"), SharedWrite:AMatch("0"),
Channels:AMatch("all",
{AMatch("1-", Channel(SLabels:{"Channel from controller to actuator"}),
AMatch("0-", Channel(SLabels:{"Channel from actuator to controller"},
AMatch("1", Channel(SLabels:{"Channel from actuator to controlled object"})}),
ComActs:AMatch("1-", ComAction(SLabels:{"Sending message from actuator"})),
SLabels:{"Actuator"})

Process( // Controlled object
SharedRead:AMatch("0"),
SharedWrite:AMatch("1", Variable(SLabels:{"Observed value"})),
Channels:AMatch("1-", Channel(SLabels:{"Channel from actuator to controlled object"}),
ComActs:AMatch("0"),
SLabels:{"Controlled object"})
```

Листинг 5. Конструктивное описание контроллеров, актуаторов и объектов
Listing 5. Constructive Description of Controllers, Actuators and Objects

На рисунке 2 представлена иерархия классов специализированной онтологии процессов для типовых элементов САУ, описанная с помощью редактора *Protégé*. Фрагмент описания класса *SimSensor* содержит ограничения на количество значений его атрибутов, перечисленных в листинге 2.

Каждый шаблон дает правила для определения элемента САУ в контексте онтологии процессов *PO*. По набору таких шаблонов можно задать систему параллельных процессов, которые реализуют типичные элементы САУ. Таким образом, мы показали, как конструктивный метод специализации может использоваться для определения специфических процессов в заданной предметной области.

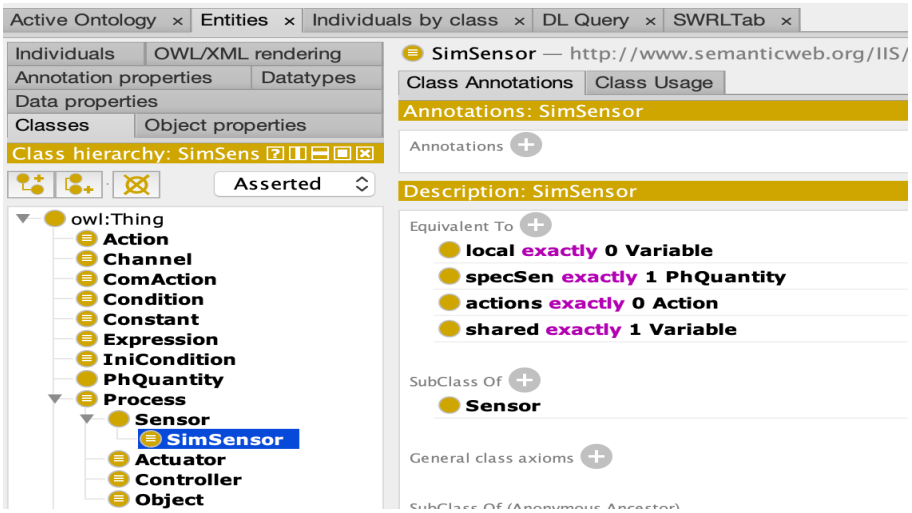

Рис. 2. Фрагмент описания специализированной онтологии процессов
Fig. 2. A fragment of the Specialized Process Ontology

Заключение

Методы разработки специализированных онтологий, основанные на трех базовых онтологиях [8], имеет несколько полезных свойств. Ориентированная на верификацию онтология процессов специфицирует компактную универсальную модель процессов с возможностью определения формальной семантики как помеченной системы переходов, что сделано в работе [4], так и других формализмов, например, машин абстрактных состояний [1] и др. Эта семантика обеспечивает возможность применения формальных методов верификации, в частности, дедуктивную верификацию и проверку моделей. Декларативная специализация необходима для задания правил извлечения информации из технической документации методами, основанными на онтологиях. Онтология $SMPO$ обеспечивает возможность связывания абстрактных процессов и их элементов с понятиями предметной области и описания новых предметно-ориентированных классов. Кроме того, разметка значений доменов онтологии PO даёт возможность описывать семантически-нагруженные значения, например, именованные числа (числа с указанием единицы измерения). Онтология $POSMPO$ шаблонов семантической разметки задаёт ограничения на семантическую разметку экземпляров онтологии $SMPO$, определяя понятия предметной области, связанные с этими экземплярами. В отличие от декларативного подхода, описывающего предметно-ориентированную онтологию процессов множеством новых классов, аксиом и правил, этот подход задаёт онтологию как множество шаблонов (экземпляров онтологии $POSMPO$) для определения предметно-ориентированных процессов конструктивно как экземпляров шаблонов из этого множества. Все три онтологии базируются на простых понятиях, которые могут быть использованы как онтологические шаблоны проектирования [2, 7, 10].

В будущем мы планируем уточнить онтологию PO и развить методы специализаций как декларативный, так и конструктивный, для построения различных предметно-ориентированных онтологий процессов. Также мы добавим новые виды операций сопоставления в конструктивном методе (например, текущее множество

операций не позволяет нам выражать свойство, что различные атрибуты имеют один и тот же экземпляр в качестве значения).

Список литературы / References

[1] Börger E., Stärk R., *Abstract State Machines: A Method for High-Level System Design and Analysis*, Springer-Verlag, 2003.

[2] Gangemi A., Presutti V., "Ontology Design Patterns", *Handbook on Ontologies, Springer, Second Edition*, 2009, 221–243.

[3] Garanina N. O., Zubin V., Lyakh T., Gorlatch S., "An Ontology of Specification Patterns for Verification of Concurrent Systems", *Proceedings of the 17th International Conference SoMeT-18, New Trends in Intelligent Software Methodologies, Tools and Techniques, Series: Frontiers in Artificial Intelligence and Applications*, **303** (2018), 515-528.

[4] Garanina N., Anureev I., "Verification Oriented Process Ontology", *Proceedings of the 9th Workshop "Program Semantics, Specification and Verification: Theory and Applications", PSSV 2018*, 2018, 58–67.

[5] Garanina N., Sidorova E., Bodin E., "A Multi-agent Text Analysis Based on Ontology of Subject Domain", *Perspectives of System Informatics. PSI 2014. Lecture Notes in Computer Science*, **8974** (2015), 102–110.

[6] Clarke E. M., Henzinger Th. A., Veith H., Bloem R., *Handbook of Model Checking*, **10**, Springer, 2018.

[7] Staab S., Studer R., *Handbook on Ontologies*, Springer Science & Business Media, 2010.

[8] Scherp. A., Saathoff C., Franz T., Staab S, "Designing Core Ontologies", *Applied Ontology, IOS Press*, **6**:3 (2011), 177–221.

[9] HermiT OWL Reasoner, www.hermit-reasoner.com, (29.09.2019).

[10] Ontology design patterns, www.ontologydesignpatterns.org, (29.09.2019).

[11] OWL Web Ontology Language Overview: W3C Recommendation 10 February 2004, www.w3.org/TR/owl-features/, (29.09.2019).

[12] Protégé. A Free, Open-source Ontology Editor and Framework for Building Intelligent Systems, protege.stanford.edu, (29.09.2019).

[13] Horrocks I. et al., "SWRL: A Semantic Web Rule Language Combining OWL and RuleML", www.w3.org/Submission/SWRL, (29.09.2019).

Garanina N. O., Anureev I. S., Borovikova O. I., Zyubin E. V., "Methods for Domain Specification of Verification-Oriented Process Ontology", *Modeling and Analysis of Information Systems*, **26**:4 (2019), 534–549.

DOI: 10.18255/1818-1015-2019-4-534-549

Abstract. User-friendly formal specifications and verification of parallel and distributed systems from various subject fields, such as automatic control, telecommunications, business processes, are active research topics due to its practical significance. In this paper, we present methods for the development of verification-oriented domain-specific process ontologies which are used to describe parallel and distributed systems of subject fields. One of the advantages of such ontologies is their formal semantics which make possible formal verification of the described systems. Our method is based on the abstract verification-oriented process ontology. We use two methods of specialization of the abstract process ontology. The declarative method uses the specialization of the classes of the original ontology, introduction of new declarative classes, as well as use of new axioms system, which restrict the classes and relations of the abstract ontology. The constructive method uses semantic markup and pattern matching techniques to link sublect fields with classes of the abstract process ontology. We provide detailed ontological specifications for these techniques. Our methods preserve the formal semantics of the original process ontology and, therefore, the possibility of applying formal verification methods to the specialized

process ontologies. We show that the constructive method is a refinement of the declarative method. The construction of ontology of the typical elements of automatic control systems illustrates our methods: we develop a declarative description of the classes and restrictions for the specialized ontology in the Protégé system in the OWL language using the deriving rules written in the SWRL language and we construct the system of semantic markup templates which implements typical elements of automatic control systems.

Keywords: process ontology, specialization, ontology axioms, pattern matching, semantic markup, automatic control system, formal verification

On the authors:
Natalia O. Garanina, orcid.org/0000-0001-9734-3808, PhD, senior researcher
A.P. Ershov Institute of Informatics Systems SB RAS, Institute of Automation and Electrometry SB RAS,
6 Lavrent'eva ave., Novosibirsk 630060, Russia, e-mail: garanina@iis.nsk.su
Igor S. Anureev, orcid.org/0000-0001-9574-128X, PhD, senior researcher
A.P. Ershov Institute of Informatics Systems SB RAS, Institute of Automation and Electrometry SB RAS,
6 Lavrent'eva ave., Novosibirsk 630060, Russia, e-mail: anureev@iis.nsk.su
Olesya I. Borovikova, orcid.org/0000-0001-5456-8513, junior researcher
A.P. Ershov Institute of Informatics Systems SB RAS,
6 Lavrent'eva ave., Novosibirsk 630060, Russia, e-mail: olesya@iis.nsk.su
Vladimir E. Zyubin, orcid.org/0000-0002-8198-3197, DSci, head of laboratory,
Institute of Automation and Electrometry SB RAS,
1 Academician Koptyug ave., Novosibirsk 630090, Russia, e-mail: zyubin@iae.nsk.su

Acknowledgments:
This work was funded by the RFBR according to the research № 17-07-01600, № 19-07-00762 and Funding State budget of the Russian Federation (IA&E project № AAAA-A17-11706061006-6).