УДК 681.3.06

КУЗНЕЦОВ А.В

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В КОНЦЕПЦИИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Предложена модель представления знаний на основе классического объектно-ориентированного формализма, охваченного семантической сетью, которая названа ST-сетью. Рассмотрены некоторые особенности реализации модели в языке объектно-ориентированного логического программирования Logtalk.

The model of knowledge representation is offered on the basis of the classic object-oriented formalism overcame a semantic network that is adopted a ST-network. Some of model realization features are considered in Logtalk object-oriented logical programming language.

Неструктурированность и процедурная избыточность программ в Прологе

Интерпретация семантических сетей (СС) [1] на Прологе сопровождается рядом трудностей и недостатков, связанных с тем, что решение задачи проблемной области (ПО) реализуется в виде *неструктурированной* совокупности отношений, представляющих множества фактов и правил вывода. Это приводит к громоздкости и запутанности программ, особенно в случаях, когда требуется спецификация понятий СС, которые включают динамически изменяющиеся свойства. Кроме того, использование исключительно классической логики для записи исходных формул СС, осложняет программирование выбора *возможных* и *требуемых* правил вывода, что ведет к порождению лишних неконструктивных решений и существенно снижает эффективность программ. С целью преодоления этих трудностей и улучшения возможностей инструментальной среды исследуются и разрабатываются средства объектно-ориентированного логического программирования (ООЛП), одним из представителей которых является система Logtalk.

Рассмотрим особенности и преимущества Logtalk с точки зрения реализации декларативно-процедурной семантики, т.е. формулировки семантики понятий СС при помощи определений и программ на основе концепции ООЛП.

Концептуальная основа Logtalk

Система Logtalk представляет собой расширение стандартного Пролога и взаимодействует со многими современными компиляторами. Основные идеи объектного подхода к программированию, заложенные в Logtalk можно кратко свести к следующим положениям [2]:

1. Система Logtalk объединяет основные преимущества двух парадигм программирования. С одной стороны, объектная ориентация позволяет работать с одним и тем же набором логически целостных элементов в последовательных фазах разработки приложения, предоставляя возможность организации и инкапсуляции знаний о каждом объекте в пределах заданной ПО. С другой стороны, логическое программирование предусматривает представление знаний о каждом объекте декларативным способом. Соединенные вместе, эти два преимущества существенно приближают разработанную программную модель к ПО, включая упрощение написания и сопровождения программ, а также повышение производительности разработчиков.
2. Обеспечена интеграция событийно-управляемого и объектно-ориентированного программирования.
3. Практически все доступные сегодня объектно-ориентированные языки программирования базируются, либо на классах, либо на прототипах, с преимущественным распространением языков базируемых на классах. В Logtalk предусмотрена поддержка обеих типовых иерархий. Это значит, что в одном и том же приложении допускается как иерархия прототипов, так и иерархия классов. Прототипы решают проблему систем базируемых на классах в случаях, когда необходимо определить класс с единственным экземпляром, который уместен при повторном использовании части кода. Классы решают проблему дуализма, характерную для систем, базируемых на прототипах, где невозможно инкапсулировать некоторый код для повторного использования другими объектами, а не только объектом, выделенным в самостоятельный элемент. Автономными считаются объекты, которые не принадлежат какой либо иерархии и дают удобные решения для инкапсуляции кода при по-

вторном использовании несколькими независимыми объектами.

4. В языках подобных Smalltalk-80, Objective-C и Java описывается единственная иерархия с корневым классом, который обычно называется Object. Это гарантирует, что все объекты разделяют общее поведение, но имеет тенденцию к созданию длинных иерархических описаний, где достаточно сложно описывать объекты с исключительным поведением. В Logtalk предусмотрены составные, независимые иерархии объектов. Некоторая часть таких иерархий может быть основана на прототипах, тогда как другие основаны на классах. Кроме того, автономные объекты обеспечивают простой способ инкапсуляции полезных предикатов, которые не требуют включения в иерархии объектов.

5. Logtalk предусматривает гибкую поддержку в разделении интерфейса и реализации так, что протокольные инструкции могут быть помещены в объект, категорию или протокол (первичные сущности Logtalk) или могут быть разбросаны (рассредоточены) по объектам, категориям и протоколам.

6. Подобно C++, Lagtalk поддерживает директивы наследования private, protected и public, предоставляя возможность ограничивать диапазон унаследованных, импортированных или включенных предикатов.

7. Еще одной характерной особенностью Logtalk является то, что имена объектов могут быть составными термами (вместо атомов), обеспечивая способ параметризации объектных предикатов. Но, доступ к значениям параметров осуществляется посредством встроенного метода вместо создания глобального диапазона параметров всего объекта.

В остальном система Logtalk спроектирована из расчета совместимости с большинством компиляторов Пролога.

Семантические сети в объектно-ориентированном проектировании

Прежде всего, остановимся на некоторых аспектах, связанных с введением соглашений, относящихся к использованию СС в объектно-ориентированном проектировании.

Как средство представления знаний СС исторически развивались, постепенно концентрируя в своей концепции такие понятия, как класс, объект, обобщение, иерархия, наследование свойств и т.п., которые характерны для существующего на сегодняшний день подхода к построению объектных моделей. Такая тенденция дает основание для рассмотрения выразительных возможностей СС с точки зрения их применения в классическом объектно-ориентированном проектировании, исходя из современных взглядов на процесс разработки. Отметим также, что семантические сети ориентированы на решение задач, относящихся к машинному анализу смысла фраз естественного языка, тогда как в объектно-ориентированном проектировании внимание разработчика акцентируется на структурно-функциональном аспекте ПО. Это значит, что объединение в ходе разработки двух формализмов решает задачу сближения языка проблемной области с объектной моделью. Другими словами, будем говорить о дополнении канонической формы сложной декомпозиции [3] семантической сетью, как это показано на рис. 1.

Представление иерархий и связей между объектной моделью и СС, требует рассмотрения некоторых терминологических и эксплуатационных отличий принятых в Logtalk по сравнению с другими языками объектно-ориентированного программирования (ООП).

Во-первых: Logtalk оперирует только с тремя видами сущностей: *объекты*, *протоколы*, и *категории*. Термины *объект*, *прототип*, *родитель*, *класс*, *подкласс*, *суперкласс*, *метакласс*, *экземпляр* и *предок* всегда обозначают объект. Различные имена используются скорее для того, чтобы сделать акцент на роли, которую играет объект в специфическом контексте. Роль объекта зависит, в свою очередь, от его отношения с другими объектами. Термин, отличный от термина *объект*, используется только в случаях, когда необходимо обратить внимание на явное отношение с другими объектами.

Во-вторых: также как и в других языках ООП, в Logtalk иерархии объектов предоставляют возможность общего использования интерфейса и реализации посредством наследования. В контексте ООЛП, наследование может быть интерпретировано как форма теории расширения: объект фактически содержит, в добавок к его собственным предикатам, все предикаты унаследованные от других объектов. Протоколы инкапсулируют объявления предиката, отделяя интерфейс от реализации, а категории инкапсулируют и объявления и определения предикатов, которые могут быть импортированы любым объектом. Таким образом, подобно тому, как протоколы поддерживают многократное использование интерфейса, категории поддерживают много-

кратное использование реализации, обеспечивая альтернативу решения проблемы множественного наследования, характерной для других языков программирования. Тем не менее Logtalk поддерживает и множественное наследование для тех случаев, где его композиционные механизмы не могут обеспечить подходящее решение.

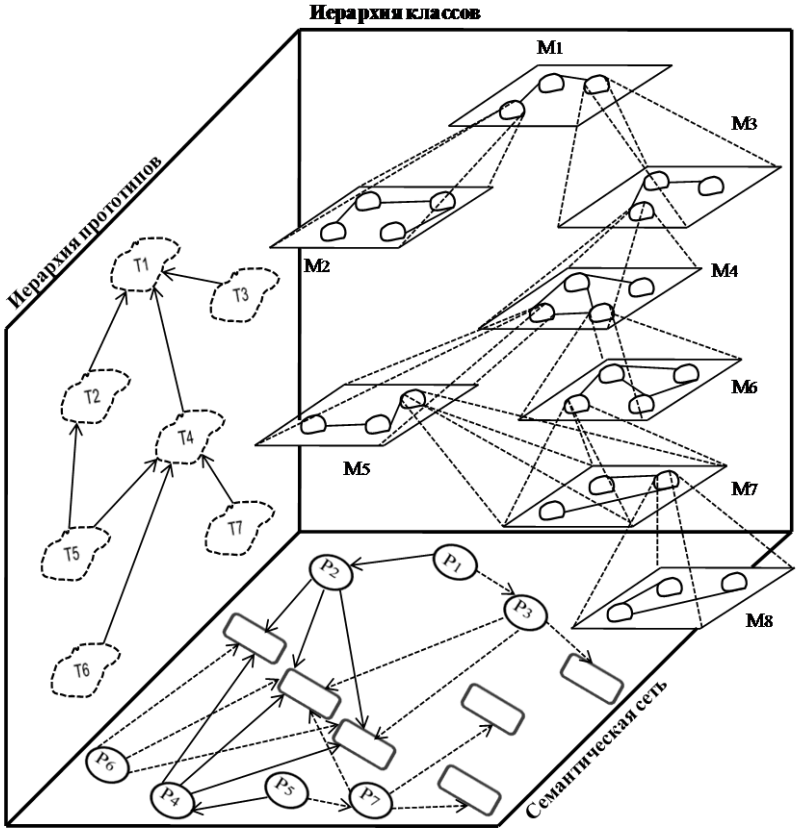

Рис. 1 Каноническая форма сложной декомпозиции, охваченной семантической сетью

В-третьих: для представления иерархий прототипов и классов в Logtalk используется три вида отношений. Иерархии прототипов конструируют определение отношений *расширения* (extension), тогда как иерархии классов, определяют отношения *экземпляра* (instantiation) и *специализации* (specialization) между объектами. На рис. 1 иллюстрируется общий случай построения комбинированной модели, основанной, как на иерархи прототипов, так и на иерархии классов. Такая форма сложной декомпозиции соответствует модели абстрактной рефлективной системы с динамически порождаемыми объектами, которая содержит метаклассы (классы интерпретируются как объекты более общих классов) и суперклассы (классы с которыми другие классы прямо или косвенно связаны отношением специализации). Так, метаклассы M_5 и M_6, представленные на рисунке, одновременно являются суперклассами метакласса M_7.

Механизм связи объектной модели с семантической сетью в Logtalk

Связь объектной модели с семантической сетью наиболее выразительно иллюстрируется возможностью Logtalk задавать имена объектов в виде составных термов, содержащих свободные переменные. Эти переменные играют роль объектных параметров, а объектные предикаты кодируются так, чтобы зависеть от их значений. Иначе говоря, параметрический объект может рассматриваться как родовой объект, из которого порождаются специфические экземпляры путем конкретизации параметров. Схему передачи параметров объектным предикатам рассмотрим на следующем примере:

```
:- object(foo1_obj).
  :- public(pred_pub/1).
  pred_pub1(something) .
...
:- end_object.
:- object(foo2_obj(_par_one, _par_two),
   specializes(protected::foo1_obj)).
```

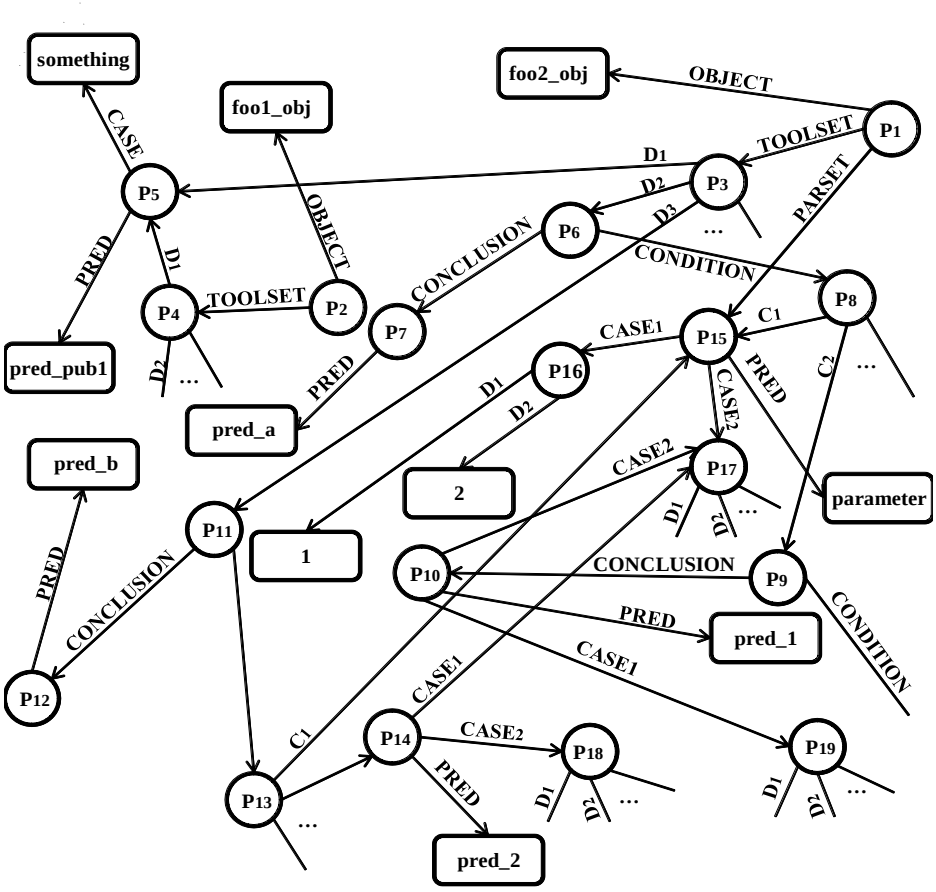

Рис. 2 Параметрический объект foo2_obj в семантической сети

CASE1 указывает на дизъюнктную вершину P16, которая объединяет возможные значения номеров позиций параметров, а связь CASE2 указывает на недоопределенную дизъюнктную вершину P17, которая объединяет возможные значения самих параметров. Подчеркнем еще раз, что в семантических падежах параметрической вершины могут быть указаны лишь возможные номера позиций параметров и их значений. Требуемые значения определяются только декларативно-процедурной семантикой объектных предикатов.

Оставшиеся недоопределенными две дизъюнктные вершины P18 и P19, объединяют возможные значения переменных Var_a и Var_b предикатов pred_1 и pred_2 соответственно.

Анализируя структуру сети на рис. 2 можно сказать, что она отображает семантику объектных моделей в ООЛП, так как охватывает объекты и связи с их инструментальными предикатами. Такие сети далее будем называть *ST-сетями*. В то же время на ST-сети никак не отражены иерархические межуровневые связи, которые формируют объектную модель ПО в целом. Требуемые связи создаются на этапе сложной декомпозиции ПО, как это показано на рис. 1, и реализуются программой Logtalk на основе декларативно-процедурной семантики. Например, в приведенном выше фрагменте программы объекты foo1_obj и foo2_obj компилируются как классы, так как они связаны отношением специализации. Тогда, объявленный в суперклассе foo1_obj public-предикат pred_pub1 становится protected-предикатом в подклассе foo2_obj, где наследование уточняется директивой protected.

В заключение отметим, что помимо возможности управлять вызовом правил вывода на основе инкапсуляции и наследования, определяемого директивами public, protected и private, в Logtalk предусмотрен гибкий механизм переопределения объектных предикатов с использованием оператора ^^/1 для организации супервызова [2]. С помощью супервызова можно управлять вызовом унаследованных предикатов, организуя *специализированное (specialization), объединенное (union)* и *выборочное (selective)* наследования.

Специализированное наследование предусматривает добавление унаследованного определения к новому коду.

При объединенном наследовании, соответственно, объединяются новое и унаследованное определения предикатов. Этот механизм реализуется путем вызова единственного унаследованного определения с оператором ^^/1 в новом определении.

Выборочное наследование позволяет организовать сокрытие некоторых унаследованнных определений. Эта форма наследования обычно используется для представления исключений в общих определениях.

Выводы

1. Смешанное представление знаний на основе объектных моделей и СС позволяет использовать преимущества объектно-ориентированного проектирования при создании программных систем, требующих семантической обработки понятий ПО.

2. В соответствии с концепцией ООЛП, принятой в Logtalk, интерпретация СС на Прологе упрощается за счет введения структурных отношений на объектах ПО согласно принципам объектно-ориентированного проектирования.

3. ST-сеть представляет собой разновидность семантической сети, в которой отображены связи объектов ПО с инструментальными предикатами этих объектов.

4. Процедурная семантика объектов графически не отражается в ST-сети, и в Logtalk реализуется только механизмами ООЛП.

Список литературы

1. Вагин В.Н. Дедукция и обобщение в системах принятия решений.–М.: Наука. гл. ред. физ. мат. лит. 1988.– 384 с.– (Пробл. искусств. интеллекта).
2. Lopes de Moura PJ (2003) Logtalk: Design of an object-oriented logic programming language. [Electronic resource] // PhD Thesis, Departamento de Informatica, Universidade da Beira Interior, Portugal. – Mode of access: logtalk.org/papers/thesis.pdf. – Last access: 2008. – Title from the screen.
3. Буч. Г. Объектно-ориентированное проектирование с примерами применения: Пер. с англ. – М.: Конкорд, 1992. – 519с.