ТЕХНОЛОГИЧЕСКОЕ ОБРАЗОВАНИЕ

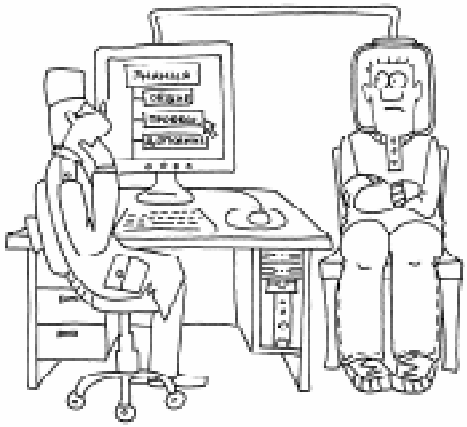
Сафонов Владимир Олегович,
Новиков Антон Владимирович,
Сигалин Максим Владимирович,
Смоляков Алексей Леонидович,
Черепанов Дмитрий Геннадьевич

РЕДАКТОР ЗНАНИЙ KNOWLEDGE EDITOR

От редакции: Статья является продолжением статьи «Интеграция методов инженерных знаний и инженерии программ: система управвления знаниями Knowledge.Net», опубликованной в № 5 за 2005 г.

I. РЕДАКТОР ЗНАНИЙ

Редактор знаний предназначен для визуализации, ввода и модификации знаний на языке Knowledge.NET. Редактор реализован как *расширение (add-in)* для Visual Studio.NET 2005, то есть он запускается автоматически вместе с Visual Studio, а его графический пользовательский интерфейс (GUI) фактически становится частью Visual Studio GUI.



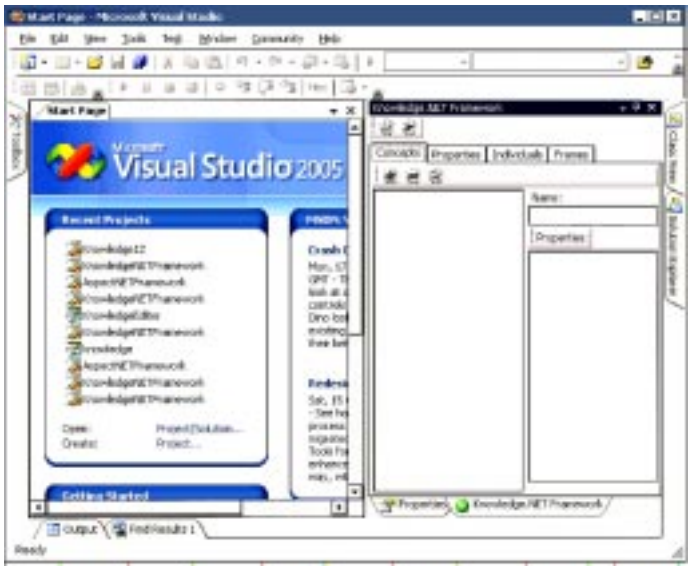


Рисунок 1. Окно add-in при запуске Visual Studio.

1. СОЗДАНИЕ ПРОЕКТА

Такой подход позволяет пользователям просматривать и редактировать знания, представленные как в текстовом виде, так и в графическом.

Окно add-in появляется при запуске Visual Studio (рисунок 1).

Для удобства пользователей системы Knowledge.NET стандартный для Visual Studio.NET набор видов проекта расширен новым – *Knowledge*. Для того чтобы создать новый проект в Knowledge.NET, пользователь может воспользоваться шаблоном «Knowledge»,

который является Visual C# шаблоном (рисунок 2).

Для поддержки этапов проектирования и реализации, мастер (wizard) генерирует заглушку (шаблон) в файле *template.expert* для будущего описания знаний пользователем на языке Knowledge.NET (рисунок 3).

Рисунок 2. Окно выбора шаблона.

2. ЗАПУСК КОНВЕРТОРА

Интерфейс Knowledge.NET включает следующие кнопки на панели инструментов:  («Refresh») и  («Convert»).

Функция «Refresh» («Обновить») позволяет выполнить синхронизацию текстового и графического представлений. Более подробно представления рассматриваются ниже.

Необходимой стадией разработки знаний является конвертирование знаний, описанных на языке Knowledge.NET, в исходный код C#. Запуск конвертора осуществляется нажатием на кнопку «Convert». Результат конвертирования пользователь может увидеть в стандартном окне Output (рисунок 4).

Запуск компилятора C# осуществляется при помощи стандартного интерфейса Visual Studio.NET 2005.

3. ТЕКСТОВОЕ ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

Система Knowledge.NET позволяет пользователю редактировать текстовое представление знаний на языке Knowledge.NET. Создание и редактирование осуществляется с использованием стан-

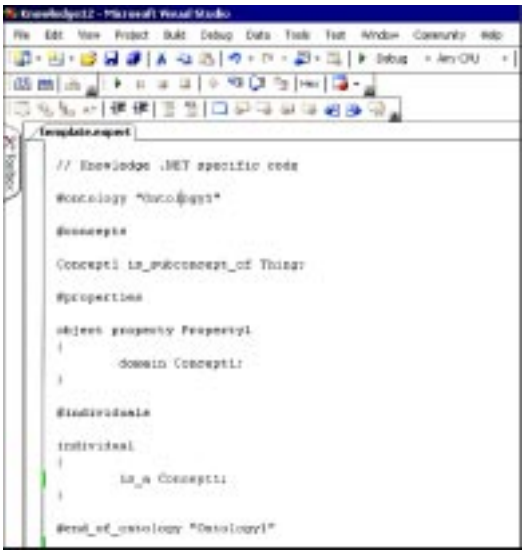


Рисунок 3. Пример исходного кода на языке Knowledge.NET

Рисунок 4. Сохранение результата конвертирования в стандартном окне Output.

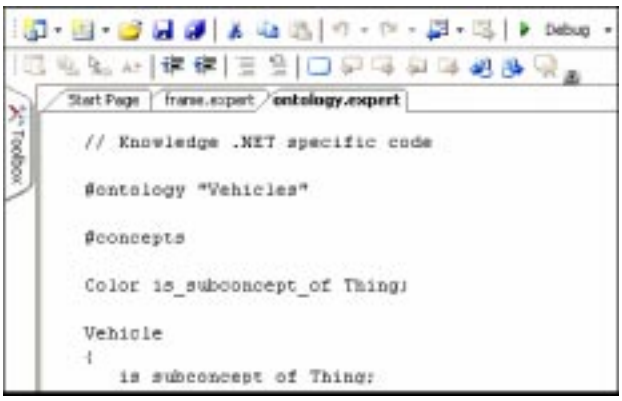


Рисунок 5. Фрагмент текстового представления знаний на языке Knowledge.NET.

дартного интерфейса среды Visual Studio 2005 (рисунок 5).

В системе реализована функциональность, позволяющая автоматически синхронизировать текстовое и графическое представления. При редактировании текста происходит обновление графического представления «на лету». В свою очередь, при модификации графического представления пользователь может проследить соответствующие изменения в текстовом представлении.

4. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

Интерфейс Knowledge.NET включает следующие вкладки:

Каждая вкладка содержит графические компоненты, позволяющие пользователю просматривать и редактировать соответствующие виды знаний. Первые три вкладки предоставляют интерфейс с онтологическими знаниями, четвертая вкладка – с фреймовыми знаниями.

Вкладка Concepts

В данной вкладке (рисунок 6) пользователь может просматривать иерархическую структуру концептов, а также информацию о свойствах, которые принадлежат данному концепту.

Любой концепт в системе Knowledge.NET должен быть подконцептом концепта «Thing».

Новый концепт может быть добавлен пользователем нажатием на кнопку () («Add»), расположенную на панели инструментов вкладки «Concepts». Удалить концепт можно нажатием на кнопку () («Delete»).

При создании концепта необходимо:

– специфицировать имя нового концепта;

– выбрать концепты, которые будут родителями нового концепта.

При создании концепта появляется модальный диалог (рисунок 7), в котором пользователю необходимо указать эту информацию.

Вкладка Properties

В данной вкладке (рисунок 8) пользователь может просматривать иерархическую структуру свойств, а также информацию о свойствах – имя, тип значений, домен, область значений.

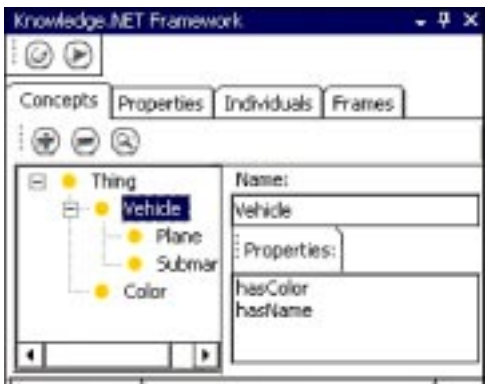


Рисунок 6. Вкладка Concepts. Создание концептов.

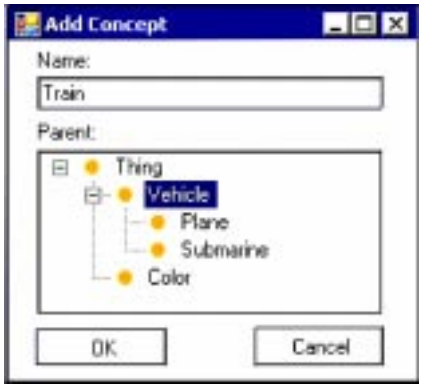


Рисунок 7. Модальное окно при добавлении концепта.

Рисунок 8. Вкладка Properties.

Рисунок 9. Модальное окно при добавлении свойства.

Новое свойство может быть добавлено пользователем нажатием на кнопку («Delete»).

При создании свойства необходимо:

– специфицировать имя нового свойства;

– специфицировать домен, тип и область значений.

При создании свойства появляется модальный диалог (рисунок 9), в котором пользователю необходимо указать эту информацию.

Вкладка Individuals

В данной вкладке (рисунок 10) пользователь может просматривать иерархическую структуру концептов, а также экземпляры, реализующие концепты.

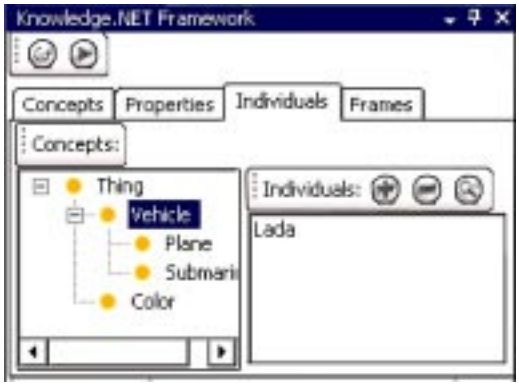


Рисунок 10. Вкладка Individuals.

Новый экземпляр может быть добавлен пользователем нажатием на кнопку «Add», расположенную на панели инструментов вкладки «Individuals». Удалить экземпляр можно нажатием на кнопку «Delete».

При создании экземпляра необходимо:

– специфицировать имя нового экземпляра;

– выбрать концепт, который реализуется экземпляром;

– специфицировать значения свойств.

При создании экземпляра появляется модальный диалог (рисунок 11), в котором пользователю необходимо указать информацию.

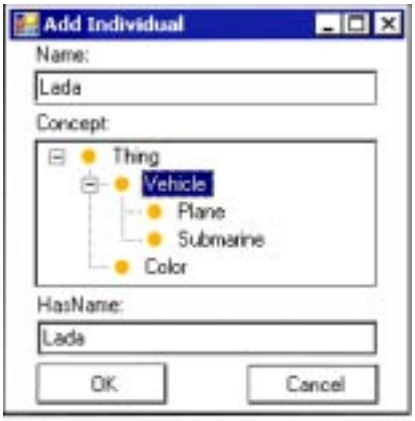


Рисунок 11. Модальное окно при добавлении экземпляра.

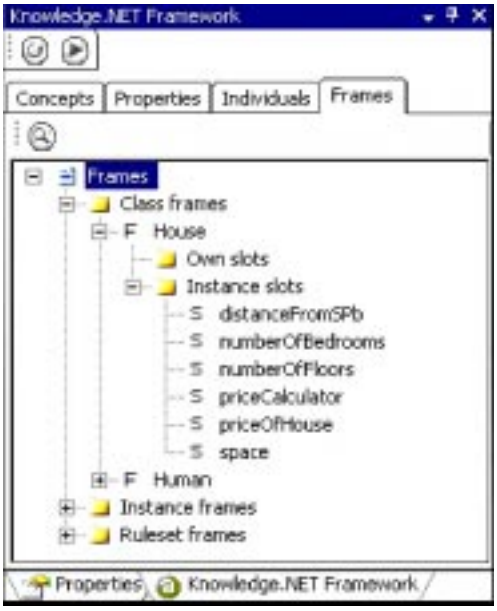


Рисунок 12. Вкладка Frames.

После добавления экземпляра обновляется как графическое, так и текстовое представление.

Вкладка Frames

В данной вкладке (рисунок 12) пользователь может просмотреть общую структуру фреймовых знаний.

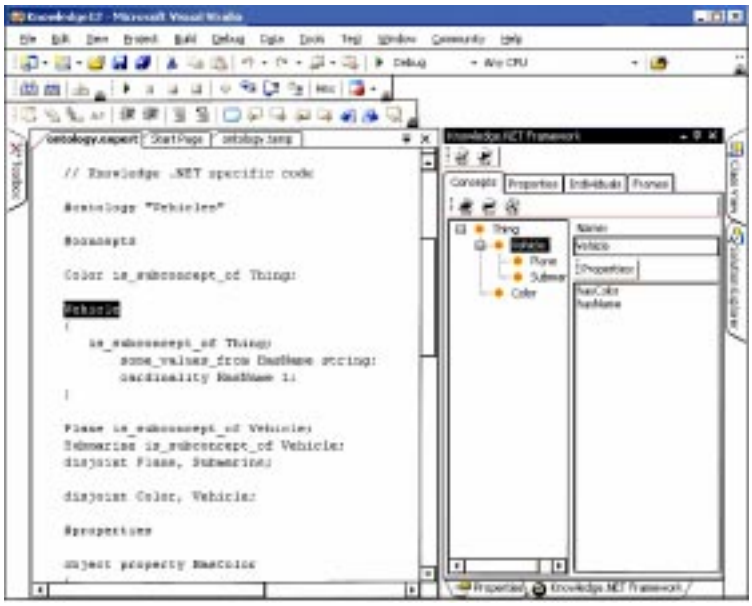


Рисунок 13. Пример навигации от графического представления к текстовому представлению.

5. СВЯЗЬ ПРЕДСТАВЛЕНИЙ

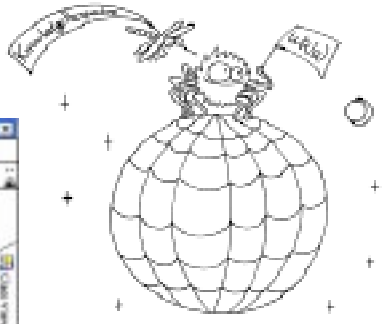
Как было сказано ранее, обновление представлений происходит синхронно. Таким образом, пользователь может выбрать и использовать наиболее удобную для себя форму представления.

Навигация в браузере знаний

Выбрав существующий объект из графического представления (концепт, слот, экземпляр, фрейм), пользователь может нажатием на кнопку () («Browse») мышкой перейти к определению этого объекта в текстовом представлении (рисунок 13).

II. СИСТЕМА ИЗВЛЕЧЕНИЯ ЗНАНИЙ KNOWLEDGE PROSPECTOR

Система извлечения знаний KnowledgeProspector, входящая в состав проекта Knowledge.NET, предназначена для извлечения знаний из текстовых (в частности, HTML) документов, и их представления в формате Knowledge.NET. Основное назначение системы – извлечение знаний из Интернета.



Система основана на алгоритме извлечения знаний, состоящего из трех основных частей:

- *Морфологический анализ* текста и построение по его результатам *набора сущностей*.
- *Семантический анализ* наборов сущностей и построение по его результатам *графа знаний*.
- *Анализ графа знаний.*

1. ОПИСАНИЕ РАБОТЫ АЛГОРИТМА ИЗВЛЕЧЕНИЯ ЗНАНИЙ

Первый этап работы алгоритма – это перевод текста с естественного языка в набор *сущностей*.

В текущей версии системы реализован анализ текста на русском языке.

Второй этап работы заключается в анализе набора сущностей. Для анализа наборов сущностей используются *правила построения графа знаний*. Одним из самых эффективных правил, с точки зрения получения количества связей и удобства настройки, является правило применения шаблонов обработки сущностей. Результат работы второго этапа – построение графа знаний. Вершины графа представляют сущности, а дуги – связи между ними.

Последующий этап заключается в анализе графа знаний. В задачи данного этапа входит:

– объединение различных свойств в единые классы (например, свойства «большой», «огромный», «маленький» имеет смысл объединить в один класс);

– удаление из графа избыточных связей (например, если у родительского класса имеется некоторое свойство, то у его наследников его можно не указывать).

2. ПЕРВИЧНЫЙ АНАЛИЗ ВХОДНОГО ТЕКСТА

Словари

Данный этап осуществляется при помощи разного вида словарей:

• *MRD-словаря*. Данный словарь содержит около 170 тысяч лексем русского языка. Словарь позволяет производить морфологический анализ слов и определять грамматические характеристики слова.

• *XML-словаря*. Это словарь, который содержит дополнительные конструкции для перевода слов языка в сущности. Например, для слова «подкласс» можно задать свойство, которое позволит при последующем анализе сущностей рассуждать о связях между словами, связанными с этим словом.

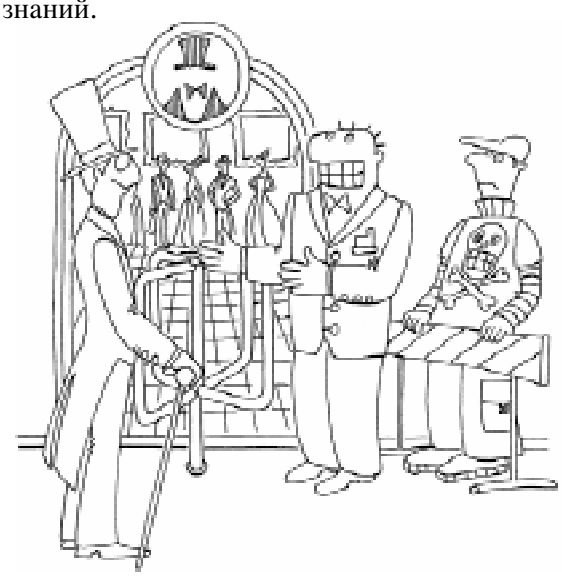
• Словарей, создаваемых пользователем. Для создания словаря необходимо реализовать интерфейс *IDictionary* и подключить словарь к экземпляру класса *DictionaryManager*.

Сущности

Выделение сущностей является результатом первичного анализа входного текста. Существует два вида сущностей:

• *Обычные* сущности («неизвестная», «разделитель», «связь»). Эти сущности не могут быть добавлены в граф знаний.

• *Настоящие* сущности («класс», «свойство», «тип данных», «индивидуал»). Сущности могут быть вершинами графа знаний.



В простейшем случае сущность содержит только слово естественного языка, из которого она получена («неизвестная» сущность). Если слово было найдено в морфологическом словаре, то ему приписываются морфологические и грамматические свойства, которые впоследствии можно использовать при анализе наборов сущностей.

Сущность «класс» (class)

То же самое, что и понятие «класс» в системе OWL. Если в словаре указано, что слово является существительным, то оно автоматически будет преобразовано в сущность типа «класс».

Сущность «свойство» (property)

То же самое, что и понятие «свойство» в OWL. Если в словаре указано, что

слово является прилагательным, то оно автоматически будет преобразовано в сущность типа «свойство».

Сущность «разделитель» (separator)

Представляет различные символы пунктуации. Разделителями считаются следующие знаки:

«.» – точка.

« », «\t», «\n» – пробел, символ табуляции, символ новой строки.

«,» – запятая.

«:» – двоеточие.

«;» – точка с запятой.

«(», «)» – открывающая и закрывающая круглые скобки.

«[», «]» – открывающая и закрывающая квадратные скобки.

«{», «}» – открывающая и закрывающая фигурные скобки.

«<», «>» – знаки «меньше» и «больше».

Сущность «время» (datetime)

Представляет дату и время. Содержит следующие поля:

Год
Месяц
День
Час
Минута
Секунда

Сущность *время,* в отличие от типа данных C# *DateTime*, позволяет указывать значения только для некоторых полей, например *месяц*; все остальные поля могут быть пустыми.

Сущность «Целое число» (integer)

Представляет целое число. Если в тексте встретится число, то оно будет преобразовано в сущность «целое число» на этапе первичного анализа.

Сущность «индивидуал» (individual)

То же самое, что и понятие «индивидуал» в OWL.

Сущность «связь» (relationship)

Слово, которое было описано в XML-словаре как связь между другими словами. Содержит такие параметры, как свойства и тип связи. Свойства могут быть:

Транзитивными
Симметричными
Функциональными.

Тип связи – это то «отношение», которое будет построено при семантическом анализе сущностей.

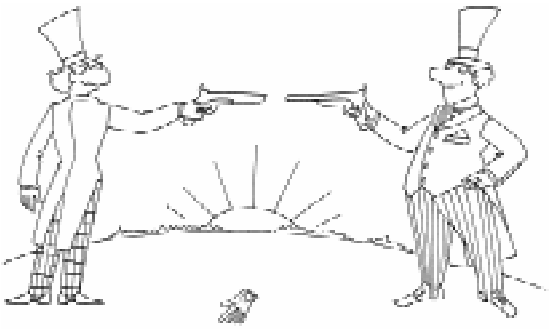
Сущность «неизвестная» (unknown)

Если для слова не удалось определить сущность одного из предыдущих типов, то ему ставится в соответствие «неизвестная» сущность.

3. СЕМАНТИЧЕСКИЙ АНАЛИЗ НАБОРОВ СУЩНОСТЕЙ

Отношения

Отношения используются для связей между сущностями в графе знаний. Отношение может быть построено только между *настоящими сущностями*.



Отношение между «свойством» и «классом»

Связывает сущность типа «свойство» с сущностью типа «класс». Фактически при этом *классу* приписывается данное *свойство*.

Отношение «подкласс»

Обозначает, что одна сущность является подклассом другой сущности. Применяется для *классов* и *свойств*.

Отношение «эквивалентность»

Обозначает эквивалентность между двумя сущностями. Применяется к сущностям типа «класс».

Правила построения графа знаний

Построение графа знаний основано на наборе правил, которые последовательно

выполняются на всём наборе сущностей. В текущей версии системы реализованы и применяются следующие правила:

- Правило применения шаблонов обработки наборов сущностей.
- Правило добавления всех *настоящих сущностей* в граф знаний.

Шаблоны обработки набора сущностей

Шаблоны являются эффективным и удобным средством построения графа знаний. С помощью шаблонов может быть реализовано, например, распознавание дат, чисел, фамилий, имен и отчеств людей. Также шаблоны позволяют создавать новые сущности, которые могут участвовать в дальнейшем анализе текста. С помощью шаблонов можно организовать особую обработку частицы «не» и вопросительных предложений. Многие из шаблонов не зависят от языка, на котором написан исходный текст.

Шаблон состоит из трех частей, отделенных друг от друга с помощью двух последовательных символов: «->». Первая часть шаблона – целое число, которое представляет приоритет шаблона. Шаблон с более высоким приоритетом будет выполнен раньше, чем шаблон с более низким приоритетом. Если приоритеты совпадают, то шаблоны выполняются параллельно. Не рекомендуется, чтобы два шаблона, которые выполняют схожие действия, имели одинаковый приоритет, так как это нарушит их работу. Пример – шаблон, который может удалить некоторую сущность из набора, и шаблон, который будет использовать эту сущность при построении связи. Во второй части шаблона описываются ограничения на набор сущностей (будем называть эту часть *регулярным выражением над сущностями*), для которых будут применены действия из третьей части шаблона (*обработчик набора сущностей*).

Язык описания регулярных выражений над сущностями

Описание

Язык предназначен для описания требований к структуре набора сущностей.

Если набор удовлетворяет этим требованиям, к нему применяются правила обработки, описанные далее.

Регулярное выражение состоит из последовательности элементов. Элементы отделяются друг от друга пробелами. Каждый элемент состоит из двух частей - ограничений на сущность и ограничений на встречаемость этого элемента.

Ограничения на сущность.

Ограничения на сущность заключаются в квадратные скобки. В случае, если ограничение состоит из одного элемента, скобки можно опустить.

Ограничения на сущность описываются с помощью бинарных логических операций:

«&» – логическое «И».

«|» – логическое «ИЛИ».

Если бинарная операция не указана, по умолчанию подразумевается операция ИЛИ.

Порядок выполнения операндов указывается с помощью круглых скобок. Если скобки не указаны, то порядок определяется традиционным соотношением приоритетов операндов (самый высокий приоритет у операции «НЕ», затем – «И», затем «ИЛИ»). Порядок выполнения в случае одинаковых приоритетов операций – слева направо.

Каждый элемент ограничения может быть двух видов:

- *Мета-элемент*, проверка которого будет осуществляться с помощью одного из ниже указанных правил. Указывается с

помощью символа «#» и последующего описания правила.

• *Слово на естественном языке* (любая последовательность символов, начинающаяся не с символа «#»). В этом случае проверка элемента будет считаться успешной, если сущность построена из того же слова, что и указанное в элементе.

Существуют следующие виды правил:

• «#E» – проверки принадлежности сущности к некоторому классу. Класс указывается через точку после «#E». Доступны следующие сокращения для классов:

«P» – сущность является *свойством* (сокращение от property).

«C» – сущность является *классом* (сокращение от class).

«I» – сущность является *индивидуалом* (сокращение от individual).

«S»– сущность является *разделителем* (сокращение от separator).

«U» – неизвестная сущность (сокращение от unknown).

«Int» – сущность является *целым числом* (сокращение от Integer).

«DT» – сущность является *временем* (сокращение от DateTime).

• «#M» – проверка части речи, к которой относится слово, из которого была построена сущность. Часть речи указывается через точку после «#M». Используются англоязычные обозначения частей речи:

Numeral – числительное

Noun – существительное

Adjective – прилагательное

Verb – глагол.

• «#S» – проверка принадлежности к одному из специальных классов слов. Класс указывается через точку после «#S».

Пример:

Month – все слова, обозначающие месяц. Все слова указаны в специальном файле.

Ограничение на встречаемость.

Ограничение на встречаемость записывается сразу же после ограничений на сущность.

Пустой символ – обозначает, что данному элементу должна удовлетворять ровно одна сущность.

Символ «+» – что элементу должна удовлетворять как минимум одна сущность.

Символ «*» – что элементу может удовлетворять любое число сущностей (в том числе – 0).

Символ «?» – элементу может соответствовать не более чем одна сущность.

Примеры

#P+ #C – означает все наборы сущностей, которые можно разбить на два непустых подмножества – в первом из них все сущности будут *свойствами*, а во втором будет только одна сущность типа *класс*.

[#P #M.Adjective]+ #C – означает все наборы сущностей, которые можно разбить на два непустых подмножества, в первом из которых все сущности являются *свойствами* или должны быть представлены *прилагательными*, а во втором содержится только одна сущность типа *класс*.

#C *является подмножество* #C – все наборы из четырех сущностей, первая и четвертая из которых являются классами, а значения второй и третьей равны соответственно «является» и «подмножество». Несоответствие русской орфографии в примере объясняется тем, что для модуля морфологической обработки слов все слова в подобном шаблоне указываются в *нормальной форме* (единственном числе, именительном падеже и т. д.). При морфологическом анализе будут правильно распознаны другие формы данного слова.

Язык описания обработчиков набора сущностей

Описание

Язык предназначен для описания действий, которые требуется выполнить над набором сущностей, удовлетворяющих регулярному выражению шаблона. Обработчик сущностей состоит из набора *действий*, отделяемых друг от друга пробелами. Действия делятся на два вида:

• Действия, создающие новые сущности

• Действия, модифицирующие набор сущностей или строящие отношения между ними.

Ниже приведены список всех возможных действий и их описание.

Действие **Add**
Назначение: Используется для добавления новых сущностей в граф знаний. Эти сущности будут добавлены в набор и не будут анализироваться шаблонами или другими правилами второго этапа.
Синтаксис: *Add (действие-создающее-сущность)*
Описание: Выполняет действие, создающее сущность, и добавляет полученную сущность в граф знаний.

Действие **Replace**
Назначение: Используется для замены нескольких существующих сущностей новой сущностью.
Синтаксис: *Replace (index, count, действие-создающее-сущность).*
Описание: Заменяет *count* сущностей, начиная с *index*, на новую сущность, созданную указанным действием.

Действие **NewDateTime**
Назначение: Используется для создания новой сущности типа *время*.
Синтаксис: *NewDateTime (параметр= индекс, ...)*
Пример: *NewDateTime (d=1, m=2, y=3)*
Описание: Создает сущность *время* и инициализирует его параметры значениями сущностей, указанными в качестве индексов.
Допустимы следующие параметры:
«Y» – год.
«M» – месяц.
«D» – день.
«H» – час.
«Min» – минута.
«S» – секунда.

Действие **PR (Property Relationship)**
Назначение: Используется для связывания *свойства* с *классом*.
Синтаксис: *PR (index1, index2)*
Описание: В качестве аргументов указываются индексы элемента регулярного выражения. Строится декартово произведение множеств сущностей, удовлетворяющих первому элементу (номер которого указан в первом аргументе), и второму элементу (его номер – второй аргумент). Все пары из построенного произведения связываются между собой с помощью объекта класса *PropertyRelationship*.

Действие **SC (Subclass Relationship)**
Назначение: Используется для обозначения того факта, что один *класс является* подклассом другого.
Синтаксис: *SC (index1, index2)*
Описание: Действие аналогично действию PR, за исключением того, что сущности в данном случае связываются при помощи экземпляра класса *SubclassRelationship*.

Примеры работы шаблонов:

Примеры шаблонов без использования слов на естественном языке:

1) Набор сущностей: P1 P2 P3 C1
   Пример на естественном языке: большой красивый светлый дом
   Шаблон: #E.P+ # E.C -> PR(1, 2)
   Построенные связи: PR(P1, C1)
   PR(P2, C1)
   PR(P3, C1)

2) Набор сущностей: Int1(15) C1(«январь») Int2(1994) C2(«год»)
   Пример на естественном языке: 15 января 1994 года.
   Шаблон: #E.INT #S.Month #E.INT -> Replace(0, 3, NewDateTime(d=2, m=1, y=0))
   Выполненные действия: Сущности Int1, C1, Int2 будут заменены на новую сущность DateTime, у которой будут установлены следующие значения параметров: день = 15, месяц = 1, год = 1994.

Примеры шаблонов со словами на естественном языке:

1) Набор сущностей: C1 E(«бывают») P1
   Пример на естественном языке: дома бывают кирпичными
   Шаблон: #E.C бывают #E.P -> PR(3, 1)
   Построенные связи: PR(P1, C1)

2) Набор сущностей: C1 E(«(») P1 E(«и») P2 E(«)»)
   Пример на естественном языке: дома (кирпичные и панельные)
   Шаблон: #E.C (#E.P и #E.P) -> PR(3,1) PR(5,1)
   Построенные связи: PR(P1, C1)
   PR(P2, C2)

От редакции: на диске к журналу находится заключительная – более специальная – часть статьи «Конвертер знаний в формате KIF».

Литература

См. Компьютерные инструменты в образовании № 5, 2005. С. 68.

Сафонов Владимир Олегович,
доктор технических наук,
профессор кафедры информатики
СПбГУ, руководитель лаборатории
Java-технологии.

Новиков Антон Владимирович,
Сигалин Максим Владимирович,
Смоляков Алексей Леонидович,
Черепанов Дмитрий Геннадьевич –
аспиранты кафедры математико-
механического факультета СПбГУ.

© Наши авторы, 2005.
Our authors, 2005.