Обзор теоретико-типового подхода для построения диалоговых систем

Г. Ю. Лобанов
mail@gleblobanov.ru

Аннотация. В статье представлен обзор теоретико-типового подхода для построения целеориентированных диалоговых систем. Он опирается на встроенную в программную платформу Grammatical Framework — систему редактирования доказательств. Взаимодействие диалогового агента и пользователя представляет собой пошаговое конструирование корректно типизированного выражения. Описаны способы сохранения и обновления информационного состояния диалога для учета контекста.
DOI: 10.52119/LPHS.2024.87.26.008.
Ключевые слова: теория типов, диалоговая система, редактор доказательств, Grammatical Framework.

A Review of the Type-Theoretical Approach to the Design of Dialogue Systems

G. Y. Lobanov

Abstract. The paper reviews the type-theoretical approach to the design of task-oriented dialogue systems. It is based on proof editor capabilities of Grammatical Framework. Interactions of a dialogue system and a user are the incremental construction of correctly typed expressions. Some means of storage and update of the information state of a dialogue for the account of a context are described.
Keywords: type theory, dialogue system, proof editor, Grammatical Framework.

Введение. Диалоговые системы делят на два класса. Целеориентированные диалоговые системы предназначены для решения некоторой задачи. Они выполняют поиск информации, заказ билетов, тестирование. Пользователь общается с системой ради некоторой функции. Когда результат получен, диалог прекращается. Это отличает диалоговые системы от чатботов. Последних разрабатывают для ведения продолжительного разговора, который имитирует разговор с человеком. Такие системы находят применение в развлечении, попытках пройти тест Тьюринга, тестировании теорий психологического консультирования [2].

В статье описан подход к разработке целеориентированных диалоговых систем на основе системы редактирования доказательств изложенный в статьях Арне Ранта, Робина Купера [5] и развитый Питером Юнглёфом [3]. Подход использует возможности редактирования доказательств программной платформы Grammatical Framework (GF) [5; 6]. Эта программная платформа является реализацией интуиционистской теории типов Пера Мартина-Лёфа [4]. Она является также функциональным языком программирования с зависимыми типами для описания формальных грамматик естественных языков. В дополнение к ней идет библиотека готовых грамматик естественных языков GF Resource Grammar Library.

Подход заключается в пошаговом построении корректно типизированного выражения, которое представляет коммуникативную цель диалога. Изначально выражение имеет тип сложной структуры и состоит только из метапеременных. Каждой метапеременной соответствует вопрос. Их последовательно задают пользователю. Из корректно типизированных ответов достраивают главное выражение. Возможны ответвления на сторонние диалоги и непоследовательные ответы пользователя. Когда конструирование главного выражения завершено, система выполняет соответствующую ему задачу.

В некотором смысле этот подход является фреймовым и традиционным для данного типа диалоговых систем. Однако теоретико-типовая составляющая, выражающаяся в использовании редактора доказательств GF, позволяет избежать ручного кодирования условий заполнения слотов во фреймах. Можно положиться на поиск по типу, проверку типа и другие уже имеющиеся инструменты, доступные через программный интерфейс GF.

1. Характеристики диалоговых систем. Для проверки диалоговых систем в ходе проекта TRINDI был выработан список Trindi Tick List [1]. Его задача — удостовериться, что создаваемая система будет удобна для пользователя. Она не должна уступать заполнению простой формы, при взаимодействии с которой пользователь может выбрать поле для заполнения произвольно. Общение с диалоговой системой должно предоставлять ему по крайней мере те же возможности по не скованному однообразной процедурой внесению информации.
В этом разделе и далее приводятся примеры взаимодействия диалоговой системы с пользователем. Предметной областью является автоматизация учета студентов на курсах. Высказывания диалоговой системы даны курсивом. Примеры иллюстрируют указанный список и приведены по образцу [5] ввиду изначального выбора авторами теоретико-типовой модели такой системы оценки диалоговых систем.

1.1. Система чувствительна к контексту. Система должна конструировать контекст из ответов пользователя. Следующие его ответы должны быть соотнесены с уже предоставленной информацией.

— *В каком году вы начали посещать курс по логике?*
— В 2016-м.
— *В каком году вы закончили посещать курс по логике?*
— Год спустя.

1.2. Система принимает больше информации, чем необходимо. Пользователь может ответить на несколько вопросов, поддерживаемых системой. Если был задан только один вопрос, остальные ответы все равно будут приняты.

— *Какой курс вы хотите изучать?*
— «Логику» с этого года.

1.3. Система принимает ответ не на актуальный вопрос. Пользователь может ответить на другой вопрос, поддерживаемый системой.

— *Какой курс вы хотите изучать?*
— В следующем году.

1.4. Система принимает ответ с неполной информацией. Пользователь может дать не всю информацию в ответе. Это приведет к новой череде вопросов для уточнения.

— *Введите название экзамена и оценку*
— «История философии».
— *Введите оценку*
— 5.

1.5. Система принимает неопределенный ответ. Ответы пользователя могут быть неопределенными в силу широты понятия или омонимии. Система должна обрабатывать серией уточняющих вопросов.

— *Какой курс вы хотите изучать?*
— По математике.

1.6. Сообщение с отрицанием правильно обрабатывается. Система должна распознавать отрицание, а не строить ответ на выделении ключевых слов в высказывании пользователя.

— *Какой курс вы хотите изучать?*
— Не логику.

1.7. Система продолжает работу без информации от пользователя. При длительной паузе со стороны пользователя система должна повторить вопрос или прекратить работу.

— *Какой курс вы хотите изучать?*
— …

1.8. Система обрабатывает помехи. Система должна вычленять в высказывании пользователя только значимые элементы.

— *Какой курс вы хотите изучать?*
— ### Логику ###

1.9. Пользователь может инициировать вспомогательные диалоги о работе системы. Для помощи пользователю система должна быть готова выдавать подсказки, ориентирующие в предметной области и функционале.

— *Какой курс вы хотите изучать?*
— Какие есть курсы на выбор?
— *«Логика», «Философия истории», «Риторика».*

1.10. Пользователь может инициировать вспомогательные диалоги, обращенные к состоянию реальности. Система должна предоставлять возможность пользователю задавать вспомогательные вопросы не только о работе системы, но и о состоянии дел. Они касаются не ее функционала, а реальности вне ее.

— *Какой курс вы хотите изучать?*
— Есть ли еще места для записи на курсе?
— *Да.*

1.11. Система задает только подходящие последовательные вопросы. Система следит за ответами пользователя и задает вопросы, соответствующие его ответам. Это возможно при сохранении и обновлении информационного состояния диалога.

— *Какой курс вы хотите изучать?*
— «Логика».

— *Когда вы хотите начать?*
— В 2019-м.
— *В вашем расписании на 2019 год есть пересечении курсов по времени.*

1.12. Система может обрабатывать несогласованную информацию. Система распознает противоречивую информацию и задает уточняющие вопросы.

— *Введите название экзамена и оценку.*
— «Логика», 5.
— *Вы не записаны на курс «Логика».*

2. Grammatical Framework. Grammatical Framework (GF) является программной платформой, в центре которой находится функциональный язык программирования с зависимыми типами [6; 8]. Его используют для описания формальных грамматик естественных языков. Он находит применение в прикладных задачах математической лингвистики: автоматическом анализе, производстве, переводе текстов. Его основы были заложены работой Арне Ранта «Type-Theoretical Grammar» [7], в которой он предложил использовать интуиционистскую теорию типов Пера Мартина-Лёфа для представления семантики естественных языков. GF впервые появился в качестве программного продукта в 1998 г. в лаборатории Xerox Research Centre Europe в Гренобле в рамках проекта «Multilingual Document Authoring». В настоящее время GF объединяет международную группу разработчиков и исследователей. Библиотека грамматик GF Resource Grammar Library насчитывает 35 грамматик естественных языков, включая русский [9].
Исходный код GF, формальная грамматика, состоит из нескольких модулей: один абстрактный синтаксис и несколько конкретных синтаксисов. Именно абстрактный синтаксис и является реализацией интуиционистской теории типов в платформе GF. Он служит для представления объектов и связей в тексте, которые не зависят от синтаксических особенностей конкретного естественного языка: семантических, логических, прагматических. Синтаксический уровень представляют модули конкретных синтаксисов. Грамматика компилируется в единый файл, и обращение к ней осуществляется через программный интерфейс.

2.1. Абстрактный синтаксис. Абстрактный синтаксис представляет собой перечисление базовых категорий, типов, функций и их определений. Это основные элементы, из которых строится и от которых зависит каждое его выражение. В листинге 1 приведен фрагмент абстрактного синтаксиса грамматики, на примере которой дальше разобран алгоритм диалоговой системы.

2.2. Конкретный синтаксис. Один или несколько конкретных синтаксисов связаны с одним абстрактным синтаксисом. Каждой функции и категории соответствует способ линеаризации — преобразования в строку. В листинге 2 представлен фрагмент конкретного синтаксиса грамматики, соответствующий разобранному далее примеру диалоговой системы.

3. Алгоритм работы диалоговой системы. Алгоритм диалоговой системы проиллюстрируем простейшим случаем регистрации на курс «Логика» на 2019 год. В представленных выше листингах 1 и 2 описан фрагмент ее грамматики. Диалоговая система начинает работу с определения коммуникативной цели. В случае рассматриваемого примера это запись на

```
abstract Dialog = {
cat Action , Course , Year ;
fun SignUp : Course -> Year -> Action ;
def SignUp course year = SignUpAnswer course year ;
fun SignUpAnswer : Course -> Year -> Action ;
fun Logic : Course ;
fun y2019 : Year ;
}
```

Листинг 1: Абстрактный синтаксис диалоговой системы. В строке 2 перечислены категории, или базовые типы, из которых могут быть составлены функциональные типы. Строки, начинающиеся с `fun`, описывают функции. `SignUp` конструирует коммуникативную цель диалога. Строка, начинающаяся с `def`, определяет функцию `SignUp` и задает значение — ответ диалоговой системы

```
concrete DialogRus of Dialog = {
lincat Action , Course , Length = { s : Str } ;
lin SignUp course year = { s = course . s ++ year . s } ;
lin SignUpAnswer course year { s = " Вы записаны на курс
" ++ course . s ++ " на " ++ year . s ++ " год ."} ;
lin Logic = { s = " Логика " } ;
lin y2019 = { s = "2019" } ;
}
```

Листинг 2: Конкретный синтаксис диалоговой системы. Строки, начинающиеся с `lin`, задают способ линеаризации соответствующих функций абстрактного синтаксиса. Одновременно они, обратно, задают способ анализа строк. Таким образом, конкретный синтаксис описывает преобразование в обе стороны: из выражения абстрактного синтаксиса в строку и из строки в выражение абстрактного синтаксиса

курс. Она представлена функцией `SignUp` типа `Action`. Приложение двух аргументов типов `Course` и `Year` к этой функции и будет целевым выражением, построение которого приведет к выполнению задачи. Построив это выражение после нескольких итераций обмена сообщений с пользователем, система запустит вычисление функции `SignUp`. Но пока на месте аргументов у функции `SignUp` стоят метапеременные. Обозначим их именем целевого типа с вопросом: `Course?`, `Year?`.

Система актуализирует первую метапеременную `Course?` и задает соответствующий вопрос пользователю. Допустим, пользователь вводит строку:

```
> "Логика"
```

Система анализирует ее и представляет в виде выражения абстрактного синтаксиса `Logic` типа `Course`. Так как его тип совпадает с типом метапеременной, итерация успешно завершается. Фокус перемещается на следующую метапеременную `Year?`. Допустим, пользователь вводит строку:

```
> 2019
```

Система анализирует ее и представляет в виде выражения абстрактного синтаксиса `y2019` типа `Year`. Так как его тип совпадает с типом метапеременной, итерация успешно завер-

шается. Все метапеременные выражения `SignUp` заполнены. Запускается вычисление функции `SignUp`. Как определено в листинге 1 в строке 4, его результатом окажется выражение `SignUpAnswer` с аргументами `course` и `year`. Система может использовать это выражение для производства строки ответа по схеме из листинга 2, строка 4.

```
> Вы записаны на курс "Логика" на 2019 год.
```

Также это выражение может быть использовано для записи в информации в базу данных и для других целей.

3.1. Информационное состояние диалога. Питер Юнглёф в [3] предлагает использовать интерактивно конструируемые деревья для сохранения и обновления информационного состояния диалога. Основной задачей остается построение главного выражения абстрактного синтаксиса, представляющего коммуникативную цель диалога. Однако теперь его предлагают рассматривать в совокупности со множеством деревьев, возникающих в процессе коммуникации системы с пользователем. Вводятся две сущности: нефиксированная вершина и связанное дерево.

Нефиксированная вершина создается тогда, когда пользователь в ответ на вопрос системы вводит сообщение, которое не соответствует типу актуальной метапеременной. Затем принимается попытка соединить нефиксированную вершину с остальным деревом. Предложено три стратегии: «сверху вниз», «снизу вверх», «снизу вниз». Стратегия «сверху вниз» соединяет доминирующую вершину метапеременной главного дерева с нефиксированной вершиной последовательными уточнениями. Стратегия «снизу вверх» соединяет доминируемые вершины последовательными уточнениями. Стратегия «снизу вниз» пытается закончить предложенное пользователем дерево, затем возвращается к главному вопросу.

Связанное дерево создается тогда, когда пользователь задает вопрос в ответ на вопрос системы. Новый вопрос пользователя становится главным вопросом поддиалога. Его результат связывается с вершиной метапеременной, которая была в фокусе на момент возникновения вопроса у пользователя. Второе применение связанных деревьев — в разрешении анафоры.

Заключение. В [5] Арне Ранта и Робин Купер утверждают, что проиллюстрированный в статье подход позволяет создать диалоговую систему, которая пройдет Trindi Tick List. Кроме того, модель позволяет добавить две характеристики, не указанные в списке. Во-первых, система может выводить ответы на незаданные вопросы. Во-вторых, она позволяет пользователю отменять свои ответы и вводить их заново.

С момента выхода статьи [5] модель получила развитие в работе Питера Юнглёфа [3]. Использование динамически конструируемых деревьев позволило сделать систему более гибкой, однако остались некоторые нерешенные проблемы. Так, в новой модели не прояснен способ исправления пользователем некорректного ввода. Кроме того, система ждет своей имплементации.

Литература

1. Bohlin P. et al. *Survey of existing interactive systems.* Trindi deliverable D1.3, 1999.
2. Jurafsky D., Martin J. H. *Speech and Language Processing.* Third edition, unpublished draft.
3. Ljunglöf P. Dialogue management as interactive tree building. *Proceedings of the 13th Workshop on the Semantics and Pragmatics of Dialogue, June 24–26, 2009, Stockholm, Sweden.* 2009, p. 83–90.

4. Martin-Löf P., Sambin G. *Intuitionistic type theory.* Napoli, Bibliopolis, 1984.
5. Ranta A., Cooper R. Dialogue systems as proof editors. *Journal of Logic, Language and Information* 13.2, 2004, p. 225–240.
6. Ranta A. Grammatical Framework: A Type-Theoretical Grammar Formalism. *Journal of Functional Programming* 14.2, 2004, p. 145–189.
7. Ranta A. *Type-Theoretical Grammar.* Oxford Science Publications, 1995.
8. GF—Grammatical Framework. 2018. URL: `www.grammaticalframework.org` (accessed: 28.05.2024).
9. GF Resource Grammar Library: Synopsis. 2018. URL: `www.grammaticalframework.org/lib/doc/synopsis.html` (accessed: 28.05.2024).