Аннотация

В статье представлен прототип системы, объединяющей методы синтаксического анализа предложений на русском языке и типизацию арифметических выражений на языке логического программирования Prolog. Система реализует декларативное описание грамматики через DCG, проверку согласования слов с использованием фактов, правил и предикатов, а также автоматическую проверку типов выражений. Особое внимание уделено механизму унификации и его роли в логическом выводе. Приведены формальные основы, алгоритмы работы системы, примеры работы и обсуждается практическая значимость проекта. Дополнительно выделены механизм объяснений и диагностики ошибок.

Оглавление

Глава 1

Введение

Обработка естественного языка (Natural Language Processing, NLP) требует не только распознавания слов, но и понимания их грамматической структуры. Одной из ключевых задач является *синтаксический анализ* — выявление структуры предложения и проверка согласованности его элементов.

Язык Prolog благодаря своей декларативной природе, поддержке **фактов**, **правил**, **предикатов** и механизма **унификации** позволяет эффективно реализовывать синтаксический анализ с помощью *Definite Clause Grammar* (DCG). Это позволяет компактно описывать грамматику и автоматически проверять корректность предложений.

Дополнение. Параллельно рассмотрим *типизацию арифметических выражений*: вычисление типа результата (`int/float/num`) по типам операндов и оператору. Здесь также используется унификация, но уже на уровне *типовых термов*.

1.1 Основные понятия и термины

Определение 1.1 (Факт)**.** *Факт — атомарное утверждение базы знаний (клауза с пустым телом), принимаемое истинным в пределах программы.*

Рассмотрим пример: лексические факты описывают словоформы и их грамматические признаки (листинг 1.1).

Определение 1.2 (Правило)**.** *Правило имеет вид* $H \leftarrow B_1, \ldots, B_n$ *и читается: «*H *истинно, если истинны все* B_i*». Факт — частный случай правила при* $n = 0$*.*

В проекте это представлено правилом DCG для предложения NP+VP.

Определение 1.3 (Предикат)**.** *Предикат — именованное отношение над термами; вычисление предиката — доказательство соответствующей цели.*

В проекте предикат `check_expr/4` проверяет корректность выражения, используя факты `result_type/4`.

Определение 1.4 (Терм)**.** *Терм — атом, число, переменная или составной терм* $f(t_1, \ldots, t_n)$*.*

Пример: `result_type('/', int, int, float)` — составной терм с функтором `result_type/4`.

Определение 1.5 (Подстановка, пример)**.** *Подстановка* $\theta = \{x_1/t_1, \ldots, x_k/t_k\}$ *заменяет переменные терма. Терм* B — пример *терма* A*, если* $B = A\theta$ *для некоторой* θ*.*

В проекте переменные `Number`, `Gender`, `Person` конкретизируются значениями из лексикона.

Определение 1.6 (Унификация)**.** *Унификация — процедура поиска наиболее общего унификатора (НОУ) пары термов.*

В проекте унификация используется для согласования признаков NP и VP: например, `noun/4` и `verb/4` унифицируются по числу и лицу или по числу и роду.

Определение 1.7 (DCG)**.** *DCG — нотация определённых клаузовых грамматик на основе разностных списков; правила транслируются в клаузы Prolog и могут включать семантические действия.*

В проекте DCG-правила описывают структуру предложений и арифметических выражений (см. листинги 1.1, 1.2).

Определение 1.8 (Типизация выражений)**.** *Тип бинарного выражения определяется фактами* `result_type/4`*; пропущенные или неизвестные аргументы получают* `default_type/1`*.*

В проекте это реализовано в модуле `types`: для операции «/» два `int` дают `float`, а неизвестные аргументы подставляются как `int`.

Пример (лексикон и DCG):

```
object(автобус, автобус, ед, муж, 3).
object(брат,    брат,    ед, муж, 3).
object(вода,    вода,    ед, жен, 3).
object(вопрос,  вопрос,  ед, муж, 3).

% DCG-правило для существительного с семантической проверкой
noun(брат, singular, masculine, third) -->
  [брат],
  { object(брат, брат, ед, муж, 3) }.
```

Листинг 1.1. Факты лексикона и порождение DCG для существительных

Пример (числовые термы):

```
num_term(int)   --> [5],   { normalize_number(5, 5),   integer(5) }.
num_term(float) --> [2.5], { normalize_number(2.5, 2.5), float(2.5) }.
```

Листинг 1.2. Составные термы для чисел и типов

Глава 2

Синтаксический анализатор на Prolog

Система опирается на три основных компонента.

Определение 2.1 (База знаний)**.** *База знаний — совокупность фактов и правил, описывающих объекты предметной области и их свойства.*

В системе база знаний представлена фактами `noun/4`, `verb/4` и `object/5`, где фиксируются число, род, лицо и другие грамматические характеристики слов.

Определение 2.2 (Грамматика (DCG))**.** *Грамматика — система правил, определяющих структуру допустимых предложений.*

В проекте она реализована через правила DCG (листинг 2.1).

```
1 sentence -->
2     noun(Number, Gender, Person),
3     verb(Number, Gender, Person).
```

Листинг 2.1. Правило DCG для предложения

Определение 2.3 (Согласование)**.** *Согласование — проверка совпадения грамматических признаков (число, род, лицо и др.) между элементами предложения.*

В прототипе согласование обеспечивается унификацией переменных `Number`, `Gender`, `Person` при сопоставлении именной и глагольной группы, что реализовано в правиле (см. листинг 2.1).

2.1 Постановка задачи

Цель работы заключалась в создании на Prolog:
— базы знаний словоформ для проверки синтаксической корректности по числу, роду и лицу;
— механизма типизации арифметических выражений с учётом целых и вещественных чисел, переменных и пропусков.

2.2 Методологический подход

— Принцип *сквозных признаков*: единые переменные $Number, Gender, Person$ проходят через нетерминалы DCG и связываются с лексическими фактами.

— *Семантические действия* в DCG для проверки согласования и фильтрации недопустимых комбинаций.
— Для типизации: *рекурсивная* DCG, порождающая типовые уравнения, решаемые унификацией с таблицей совместимости.

Глава 3

Теоретическая основа

3.1 Синтаксический анализ

Каждое слово характеризуется признаками: лемма, число, род, лицо, время и т. д. Проверка предложения выполняется через согласование признаков существительного и глагола. Система считает предложение корректным, если существует успешная унификация термов `noun/4` и `verb/4` по числу, роду и лицу:

$$\exists\, n, g, p: \quad \texttt{noun}(\_, n, g, p) \ \wedge\ \texttt{verb}(\_, n, g, p).$$

Эквивалентно, существует НОУ для троек признаков NP/VP:

```
noun(Lemma, Number, Gender, Person) ~ verb(_, Number, Gender, Person).
```

3.2 Типизация арифметических выражений

Определение 3.1 (Базовые типы)**.** *В системе выделяются три базовых типа:* `int`*,* `float` *и обобщающий* `num`*.*

```
type(int).
type(float).
type(num).

default_type(int).
```

Листинг 3.1. Базовые типы и тип по умолчанию

Определение 3.2 (Совместимость операторов)**.** *Тип результата бинарной операции фиксируется фактами* `result_type/4`*.*

```
result_type('+', int,   int,   int).
result_type('+', int,   float, float).
result_type('+', float, int,   float).
result_type('+', float, float, float).

result_type('/', int,   int,   float).
result_type('/', int,   float, float).
```

```
result_type('/', float, int,   float).
result_type('/', float, float, float).
```

Листинг 3.2. Факты предиката `result_type/4`

Определение 3.3 (Проверка выражения)**.** *Вывод типа выражения осуществляется предикатом `check_expr/4`, который сопоставляет типы операндов с таблицей и подставляет `int`, если аргумент неизвестен.*

```
check_expr(AType0, BType0, Op, ResType) :-
    ( var(AType0) -> default_type(AType) ; AType = AType0 ),
    ( var(BType0) -> default_type(BType) ; BType = BType0 ),
    result_type(Op, AType, BType, ResType).
```

Листинг 3.3. Проверка выражения

Определение 3.4 (Числовые термы)**.** *Числовой терм может быть целым, вещественным или неизвестным идентификатором; в последнем случае подставляется `int`.*

```
num_term(int) -->
    [X],
    { normalize_number(X, N), integer(N) }.

num_term(float) -->
    [X],
    { normalize_number(X, N), float(N) }.

num_term(Type) -->
    [X],
    { unknown_token(X), default_type(Type) }.
```

Листинг 3.4. Числовые термы

Глава 4

Реализация

4.1 База знаний

Основой системы является база знаний, где каждому слову сопоставлены грамматические признаки: число, род, лицо, а для глаголов также время или залог. Фрагмент фактов показан в листинге 4.1.

```
noun(брат,    singular, masculine, third).
noun(вопросы, plural,   neuter,    third).

verb(работал,  singular, masculine, past).
verb(решается, singular, neuter,    present).
```

Листинг 4.1. Факты о словах (фрагмент)

4.2 Грамматика (DCG) и согласование

Структура простого предложения описана правилом DCG: оно объединяет именную и глагольную группы, передавая между ними общие переменные для согласования признаков. Если значения не совпадают, унификация завершается неудачей и предложение считается некорректным (листинг 4.2).

```
sentence -->
  noun(Number, Gender, Person),
  verb(Number, Gender, Person).
```

Листинг 4.2. DCG-правило для предложения

4.3 Типизация выражений

Арифметические выражения обрабатываются аналогично синтаксису предложений. DCG–правила рекурсивно строят выражения и сопоставляют типы операндов с таблицей совместимости. Целые числа интерпретируются как `int`, вещественные — как `float`. При делении результат всегда имеет тип `float`. Пропущенные или неизвестные аргументы получают тип `int` по умолчанию.

```
expr(int)   --> [X], { integer(X) }.
expr(float) --> [X], { float(X) }.

expr(int)   --> expr(int),   [+], expr(int).
expr(float) --> expr(int),   [+], expr(float).
expr(float) --> expr(_),     [/],  expr(_).
```

Листинг 4.3. Фрагмент правил типизации

4.4 Нормализация входных данных

Перед анализом строковые представления чисел преобразуются в числовые значения. Это позволяет одинаково корректно обрабатывать, например, `5` и `'5'`.

```
normalize_number(Str, Num) :-
    ( number_string(Num, Str)
    -> true
    ; atom_string(Atom, Str), atom_number(Atom, Num)
    ).
```

Листинг 4.4. Нормализация числовых термов

Глава 5

Примеры работы системы

5.1 Синтаксический анализ

— «брат работал» → `true`: используются `noun(брат, singular, masculine, third)` и `verb(работал, singular, masculine, past)`; правило `sentence -> noun(...), verb(...)`; унификация успешна.
— «братья работают» → `true`: `noun(братья, plural, masculine, third)` и `verb(работают, plural, masculine, third)` унифицируются.
— «вопросы решается» → `false`: `noun(вопросы, plural, neuter, third)` и `verb(решается, singular, neuter, present)` конфликтуют по числу.

5.1.1 Дерево разбора DCG

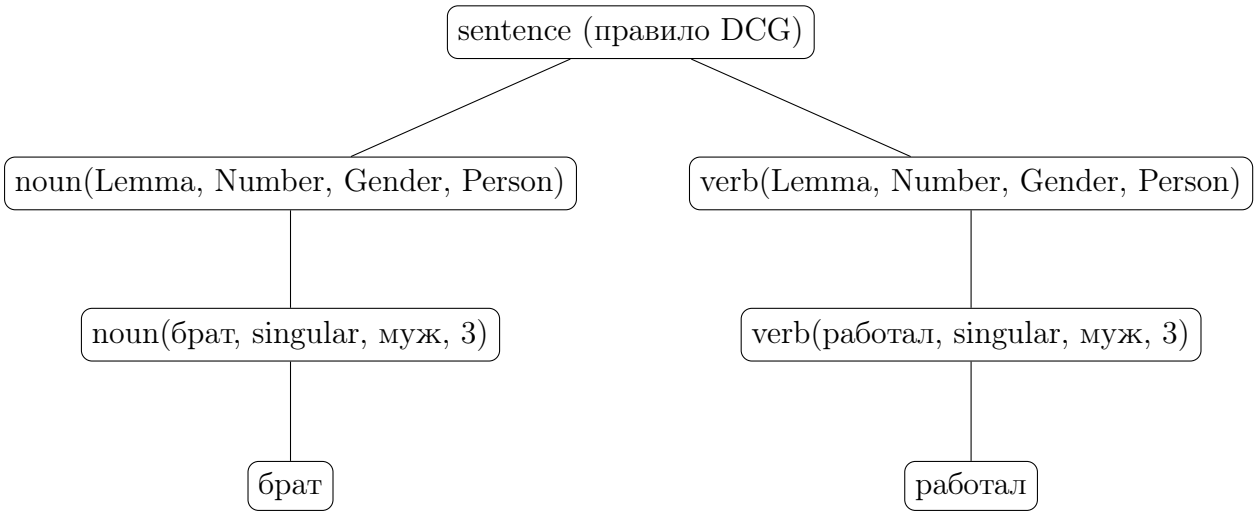

Рис. 5.1: Дерево разбора предложения «брат работал»

5.2 Типизация выражений

На основании таблицы совместимости:
— $5 + 2.5 \rightarrow$ `float`
— $2 * 2.5 \rightarrow$ `float`

— $x + 5 \to$ `int` (если x неизвестен, `int` по умолчанию)
— $y - 2.5 \to$ `float`
— $z * w \to$ `int` (оба `int`)
— $var/3 \to$ `float`
— $5 + \_ \to$ `int` (эвристика по умолчанию)

5.2.1 Правило вывода типов

$$e_1 : \tau_1, \;\; e_2 : \tau_2, \;\; \text{result\_type}(op, \tau_1, \tau_2) = \tau \;\Rightarrow\; e_1 \; op \; e_2 : \tau$$

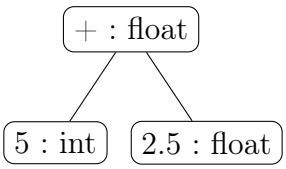

Рис. 5.2: Типизация примера $5 + 2.5$

5.3 Обоснование выводов

Для практического применения системы важно не только определить истинность высказывания или тип выражения, но и уметь объяснить, почему получен именно такой результат. Это позволяет использовать прототип в образовательных и прикладных задачах. Возможны следующие подходы:

1. При несогласованности грамматических признаков фиксировать конфликтующие характеристики (например, `plural` и `singular`) и формировать краткое пояснение причины отказа.
2. При выводе типа арифметического выражения указывать правило таблицы `result_type/4`, по которому сделано заключение, или сообщать о невозможности сопоставления.
3. Для повышения надёжности поддерживать набор тестовых примеров «вход -> ожидаемый результат», что обеспечивает воспроизводимость экспериментов.

Глава 6

Актуальность и практическое применение

Прототип системы на языке Prolog демонстрирует широкий спектр практических применений. В образовательной сфере он может использоваться при обучении русскому языку: база фактов и правил грамматики позволяет автоматически объяснять структуру предложений и согласование слов, делая процесс обучения наглядным и интерактивным.

В задачах проверки текста система выявляет и корректирует грамматические несогласования, опираясь на предикаты и механизм унификации признаков. Это обеспечивает возможность автоматизированного контроля качества письменной речи.

При обработке больших массивов текстов, например медицинских или юридических, масштабируемая база знаний и декларативное описание грамматики упрощают анализ и делают его воспроизводимым.

Кроме того, система может быть интегрирована с современными NLP–платформами: логический вывод дополняет методы машинного обучения, повышая объяснимость и интерпретируемость получаемых решений.

Глава 7

Результаты

Разработанный прототип подтвердил эффективность выбранного подхода. Компактное описание грамматики и правил типизации облегчает расширение системы и добавление новых конструкций. Использование типа по умолчанию обеспечивает корректную обработку неполных или динамических выражений, а нормализация входных данных повышает устойчивость анализа.

Декларативное представление знаний через факты, правила и предикаты в сочетании с механизмом унификации делает вывод прозрачным и наглядным. Это упрощает понимание работы системы и открывает возможности для её применения в обучении, автоматической проверке текстов и обработке больших объёмов данных.

Глава 8

Перспективы развития

Дальнейшее развитие системы связано с расширением её выразительных возможностей. Планируется поддержка более сложных синтаксических конструкций, расширение словаря до масштабов полноценных корпусов, а также введение дополнительных типов данных и операторов, включая логические и сравнительные.

Особое внимание предполагается уделить интеграции с современными NLP–библиотеками и моделями машинного обучения. Это позволит использовать прототип в качестве основы для интеллектуальных платформ анализа текста, где логический вывод будет органично сочетаться с вероятностными методами и обеспечивать интерпретируемость получаемых решений.

Глава 9

Заключение

Представлен исследовательский прототип на Prolog, объединяющий DCG–синтаксис для русского языка и модуль типизации арифметических выражений. Сохранены и формализованы ключевые определения и методика, показаны примеры и ограничения. Унификация выступает единым механизмом как для согласования грамматических признаков, так и для вывода типов, что делает подход наглядным и расширяемым.

Литература

[1] Братко И. *Программирование на языке Prolog для искусственного интеллекта.* М.: Вильямс, 2004.

[2] Sterling L., Shapiro E. *The Art of Prolog.* MIT Press, 1994.

[3] SWI-Prolog. *Using Definite Clause Grammar (DCG).* URL: `https://nweb42.com/books/prolog/ispolzovanie-dcg-definite-clause-grammar/`