Правительство Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Санкт-Петербургский государственный университет»

Математическое обеспечение и администрирование
информационных систем

Системное программирование

Горохов Артем Владимирович

Применение синтаксического анализа в распознавании планов

Курсовая работа

Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
к. ф.-м. н., доцент Григорьев С. В.

Санкт-Петербург
2018

Оглавление

Введение

Интеллектуальные системы получили широкое распространение в современности, при этом одним из их важных аспектов является возможность распознавания различных действий людей и других систем. Рассмотрим задачу определения цели или планов агента (общеиспользуемый термин), по его действиям. Область науки изучающую этот вопрос называют распознаванием деятельности (Activity recognition). В этой области выделяют различные направления, среди которых *распознавание цели (goal recognition)* и *распознавание планов (plan recognition)* [8], основанные на использовании так называемых *библиотек планов* — наборов схем действий, которым может следовать агент.

Распознавание цели — это задача классификации, цель которой состоит в определении общей цели по выполняемым действиям на основании библиотеки планов. Для примера может быть рассмотрен случай наблюдения за действиями человека на кухне. Последовательность действий, начинающаяся с “взбить яйца”, “смешать с мукой” может быть распознана как “выпечка”, “выпечка торта”, “выпечка шоколадного торта” и т. д.

Задача *распознавания планов* включает в себя установление цели, но так же предполагает построение структуры действий согласно структуре библиотеки планов. Такой подход предлагает больше данных для последующего анализа. В рассмотренном выше примере система, выполняющая распознавание планов, способна в реальном времени выдать гипотезу о том, что человек собирается приготовить шоколадный торт. Кроме того, эта же система способна помочь повару предотвратить ошибки, в то время как система, решающая задачу распознавания цели, сможет только переклассифицировать действия с “выпечка шоколадного торта” на “неудачная попытка выпечки шоколадного торта”. В данной работе будет рассмотрена задача распознавания планов.

Алгоритмы распознавания планов могут применяться в различных областях: поддержка экипажей по время чрезвычайных ситуаций с по-

мощью распознавания используемого протокола действий [3], уведомление о пропущенных шагах в медицинском руководстве используемым врачом [4, 9], помощь в использовании программного обеспечения в обучении [1, 10] и обеспечение реакции в режиме реального времени на кибератаку [11, 12].

Задача распознавания планов связана с синтаксическим анализом: библиотека планов может быть представлена в виде контекстно-свободных грамматик, а процесс построения структуры действий — как построение дерева разбора для грамматики [7]. С другой стороны, существуют обстоятельства, не позволяющие адаптировать существующие алгоритмы синтаксического анализа для этой области: в распознавании планов рассматривают одновременно несколько возможных планов, а так же их чередование (interleaving) в последовательности рассматриваемых действий. Таким образом, в большинстве алгоритмов распознавания планов используются идеи синтаксического анализа, но не используются новейшие алгоритмы анализа. Одним из таких алгоритмов является Generalised LL [14]. Данный алгоритм позволяет осуществлять анализ строк для произвольных контекстно-свободных грамматик. Кроме того, в лаборатории языковых инструментов Jetbrains, СПбГУ, на базе инструмента для создания и изучения синтаксических анализаторов YaccConstructor [18], разработано расширение Generalised LL-алгоритма, позволяющее проводить анализ конечных автоматов, возможночть использования которого представляет интерес.

1. Постановка задачи

Целью данной работы является создание алгоритма распознавания планов, использующего известные высокопроизводительные алгоритмы синтаксического анализа. Для её достижения были поставлены следующие задачи.

- Изучить возможность использования алгоритма Generalised LL в решении задачи распознавания планов для получения высокопроизводительного решения.
- Разработать новый алгоритм распознавания планов на основе проведённых исследований.
- Выполнить реализацию разработанного алгоритма на базе инструмента YaccConstructor.
- Провести экспериментальное сравнение реализованного алгоритма с существующими.

2. Обзор

2.1. Распознавание планов

Формулировка задачи распознавания планов оперирует понятием библиотеки планов.

***Определение* 1.** *Библиотека планов — это кортеж* (Σ, NT, R, P)*, состоящий из следующих элементов:*

- Σ *и* NT *— непересекающиеся конечные алфавиты терминальных действий и нетерминальных целей;*
- $R \subseteq NT$ *— множество корневых целей;*
- P *— множество продукций вида* $(n, \beta, \sqsubset)$*, где*
  - $n \in NT$*;*
  - β *— строка символов из* $\Sigma \bigcup NT$*, для которой* $(|\beta| = 1)\&(\beta \subseteq \Sigma)$ *или* $(|\beta| > 1)\&(\beta \subseteq NT)$*;*
  - $\sqsubset$ *— асимметричное отношение над индексами* $[1, |\beta|]$*, описывающее допустимый порядок целей.*

Библиотека планов описывает иерархии которые можно построить из различных действий. Задача распознавания планов состоит в построении всех возможных иерархий для наблюдаемых действий, исходддя из информации в библиотека планов. При этом на задачу могут накладываться различные ограничения: *множества действий в иерархиях не должны пересекаться*; *иерархии должны содержать все наблюдаемые действия* и другие.

Во многих задачах необходима работа в реальном времени — алгоритм строит неполные иерархии действий, а с поступлением новых достраивает или отвергает уже построенные. Существующие алгоритмы распознавания планов для работы в реальном времени способны обрабатывать только небольшие последовательности действий, работать

на ограниченных доменах или ограничивать полноту выведенных гипотез [5, 17]. Это связано с тем, что число гипотез может расти быстро даже для небольшого числа действий [16]. Как пример этого может быть рассмотрен сценарий выпечки, в котором повар начинает с муки и сахара — многие рецепты начинаются с этих ингредиентов, поэтому объяснение действий повара потребует генерации большого количества гипотез-иерархий.

2.2. Контекстно-свободные грамматики с шаффлом

Задачу распознавания планов можно рассматривать с точки зрения синтаксического анализа: библиотека планов представима набором контекстно-свободных грамматик со стартовыми нетерминалами R_i для каждой корневой цели из R, а продукции $(n, \beta, \sqsubset)$ — как продукции грамматики $n \to \beta'_1|...|\beta'_v$, где $\beta'_1...\beta'_v \in \beta'$ — все перестановки β, в которых сохраняется отношение $\sqsubset$. Тогда иерархии — это деревья разбора последовательностей терминалов (наблюдаемых действий).

В данной работе будет рассмотрена задача распознавания планов с условием, что каждое действие принадлежит иерархиям с единой корневой целью. Такая формулировка позволяет строить множества непересекающихся иерархий, и рассматривается в последних работах исследующих задачу распознавания планов [7, 16, 8]. Такая постановка задачи также использовалась в работе [7], в которой описывается связь задачи распознавания планов с синтаксическим анализом контекстно-свободных грамматик с *шаффлом* (Context-Free Shuffle Grammars, CFSG). Для определения CFSG введём понятие *шаффла*.

Операция *шаффла* ($\odot$) строк определяется индуктивно:

- $\epsilon \odot u = u \odot \epsilon = u, \forall u \in \Sigma^*$ (Σ — алфавит);
- $\alpha_1 u_1 \odot \alpha_2 u_2 = \{\alpha_1 w | w \in (u_1 \odot \alpha_2 u_2)\} \cup \{\alpha_2 w | w \in (\alpha_1 u1 \odot u_2)\}$, $\forall \alpha_1, \alpha_2 \in \Sigma$ и $u_1, u_2 \in \Sigma^*$.

Эта операция расширяется на языки следующим образом:
$L_1 \odot L_2 = \bigcup_{u_1 \in L_1, u_2 \in L_2} u_1 \odot u_2$.

Единственное структурное отличие контекстно-свободных грамматик с шаффлом (CFSG) от контекстно-свободных грамматик — это структура продукции стартового нетерминала: в CFSG она имеет следующий вид: $S = A_1 \odot A_2 \odot ... \odot A_n$ и задаёт шаффл языков порождаемых нетерминалами $\{A_i | i \in 1..n\}$. Таким образом, можно осуществить переход от задачи распознавания планов к задаче синтаксического анализа CFSG.

2.3. Построение леса разбора с помощью Generalised LL

Одним из наиболее производительных алгоритмов синтаксического анализа контекстно-свободных грамматик является Generalised LL (GLL). Данный алгоритм позволяет строить деревья разбора строки для любой контекстно-свободной грамматики. Если грамматика неоднозначна (возможно несколько вариантов вывода некоторых строк), алгоритм GLL строит все возможные деревья. Для построения используется структура данных Shared Packed Parse Forest (SPPF) [15], позволяющая хранить лес разбора (множество деревьев разбора), переиспользуя построенные узлы дерева. SPPF содержит следующие типы узлов (i и j — это начало и конец выведенной подстроки для данного узла).

- *Упакованный узел* (M, k), где M — позиция в грамматике, k — начало выведенной подстроки правого ребёнка — символьного узла. Левый ребёнок опционален и может быть представлен символьным или промежуточным узлом.

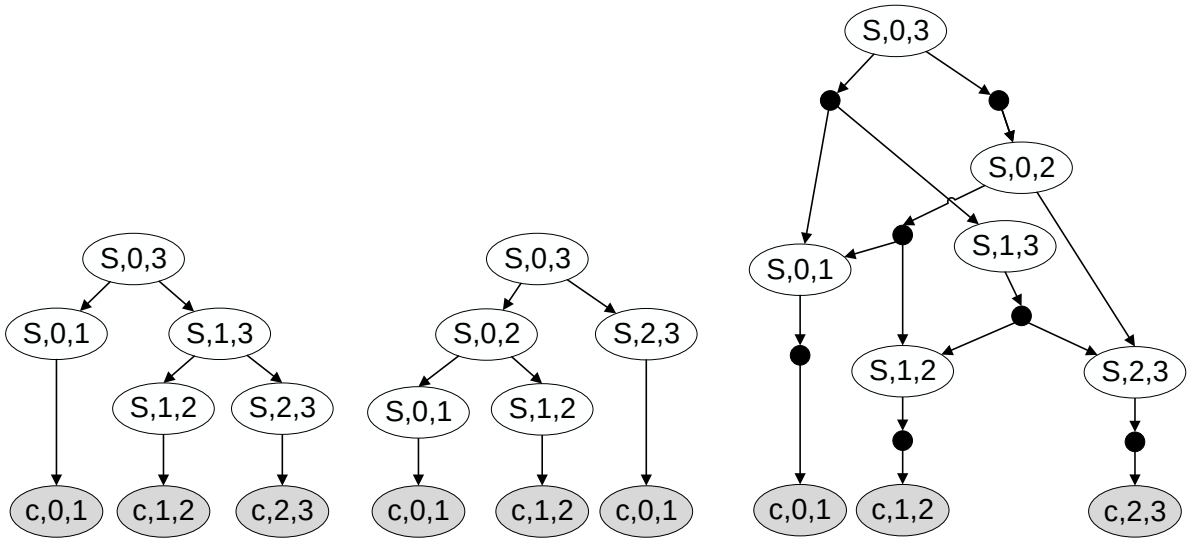
- *Символьный узел* (X, i, j), где X — терминал или нетерминал. Терминальные символьные узлы — листья.

- *Промежуточный узел* (M, i, j), где M — позиция в грамматике.

$$S ::= S\ S \mid c$$

Рис. 1: Неоднозначная грамматика G_0

Символьные и промежуточные узлы имеют 1 или более детей — упакованных узлов. Различные дети этих узлов — различные варианты выведенных поддеревьев. Если у узла (символьного или промежуточного) или его потомков более одного ребёнка, то для него было построено несколько вариантов разбора строки. Так, деревья, представленные на рис. 2a, объединяются в SPPF показанный на рис. 2b. Промежуточные и упакованные узлы необходимы для обеспечения переиспользования узлов SPPF — такое представление существенно снижает затраты памяти для некоторых грамматик, благодаря этому пространственная сложность алгоритма Generalised LL: $O(N^3)$ — как и временная.



(a) Возможные деревья вывода (b) SPPF

Рис. 2: Пример для входа ccc и грамматики G_0

Кроме того, было предложено расширение алгоритма Generalised LL [13], позволяющее для контекстно-свободной грамматики производить анализ регулярного множества, представленного в виде конечного автомата. Так, для грамматики $S \to a(b|d)^*cd$ и конечного автомата M_2 (рис. 3) алгоритм построит лес разбора, содержащий все строки вида ab^*cd. Далее будет описано применение этого расширения алгоритма Generalised LL для анализа контекстно-свободных грамматик с

шаффлом.

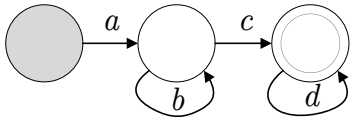


Рис. 3: Конечный автомат M_2

2.4. NP-полнота задачи

Известно, что задача синтаксического анализа CFSG является NP-сложной [2]. Одним из известных подходов к решению NP-сложных задач является сведение к задаче выполнимости булевых формул (Boolean satisfiability problem, SAT) [6]. Задача SAT заключается в определении возможности задания всем переменным, встречающимся в булевой формуле, значения ложь и истина так, чтобы формула стала истинной. Для решения задачи SAT существует множество так называемых SAT-решателей, и с каждым годом появляются новые работы по увеличению их производительности. В связи с этим, представляет интерес изучение возможности использования SAT-решателей в решении задачи синтаксического анализа CFSG, так как такой подход может значительно повысить производительность в сравнении с существующими алгоритмами.

3. Алгоритм распознавания планов

В этой главе описан предложенный в данной работе алгоритм синтаксического анализа контекстно-свободных грамматик с шаффлом, а так же его применение в задаче распознавания планов.

3.1. Синтаксический анализ CFSG

Задачу синтаксического анализа контекстно-свободных грамматик с шаффлом можно сформулировать следующим образом: для CFSG имеющей структуру представленную на рис. 4 и входной строки $L = l_1 l_2 ... l_k$, необходимо предоставить все возможные множества деревьев вывода $R = \{T_1, T_2, ..., T_n\}$, где T_i — дерево порождённое нетерминалом A_i, таких, что:

- объединение элементов крон $T_1, T_2, ..., T_n$ образует L;
- порядок терминалов в кронах согласован с порядком в L;
- пересечение элементов крон $T_1, T_2, ..., T_n = \emptyset$.

Для решения этой задачи с применением SAT-решателя необходимо осуществить её конвертацию в экземпляр задачи SAT. Но так как в общем случае SAT-решатель осуществляет полный перебор, целесообразно предварительно сузить пространство поиска решения. Рассмотрим преобразование нашей задачи, которое будет использоваться для конвертации в SAT.

$$
\begin{aligned}
S &\to A_1 \odot A_2 \odot ... \odot A_n \\
A_1 &\to ... \\
A_2 &\to ... \\
&... \\
A_n &\to ...
\end{aligned}
$$

Рис. 4: Структура контекстно-свободной грамматики с шаффлом.

Описанные выше множества деревьев $\{T_1, T_2, ..., T_n\}$ строятся на основе информации из грамматик и входной строки, то есть производится синтаксический анализ строк, шаффл которых образует входную строку, при этом происходит разбор по контекстно-свободной грамматике. Таким образом, перед осуществлением поиска множеств деревьев $\{T_1, T_2, ..., T_n\}$ можно построить множества $T'_1, T'_2, ..., T'_n$, содержащие всевозможные деревья для данной входной строки и каждой из грамматик. Для некоторых грамматик такой подход может значительно сузить пространство поиска. Данное решение предполагает использование синтаксического анализ множества строк $L' = \{l_1 l_2 ... l_m \mid l_i \in L;\ m \leq k;$
$pos(l_i) < pos(l_{i+n});\ i, n \in N\}$, где $pos(l_i)$ — позиция l_i в L. Так, для строки $a12b$ искомым множеством является $\{a, 1, 2, b, a1, a2, ab, 12, 1b, 2b, a12, a2b, a1b, 12b, a12b, \epsilon\}$.

Анализ множества строк возможен с использованием расширения алгоритма Generalised LL, предложенного в работе [13]. Данный алгоритм позволяет осуществлять анализ конечного автомата и строит лес разбора в форме SPPF для всех строк, порождаемых этим автоматом. Для его использования при анализе множества L' необходимо построить порождающий его конечный автомат. Для этого следует:

- построить конечный автомат M_0 (рис. 5a) порождающий строку L;
- дополнить все переходы автомата ϵ-переходами. Результат — автомат M_0 (рис. 5b).

Полученный автомат порождает множество строк L'. Так, например, для строки $a12b$ искомое множество порождается автоматом M_1 (рис. 6).

Результатом работы алгоритма для каждой из грамматик со стартовым нетерминалом A_i является лес разбора, описывающий искомое множество T'_i. Таким образом, после применения алгоритма, необходимо отыскать такие $T_1, T_2, ..., T_n$, что будут выполнены следующие условия

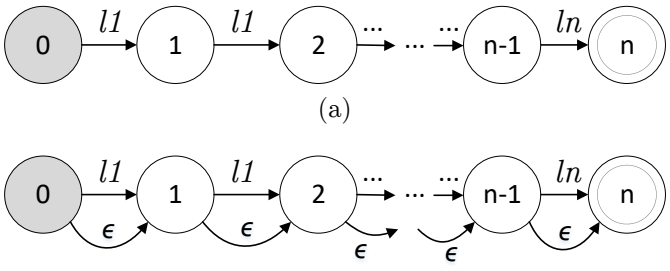


Рис. 5: Автомат M_0 (a) и его замыкание (b).

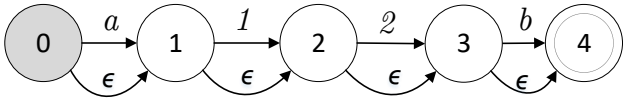


Рис. 6: Автомат M_1 — для строки $a12b$.

- $\forall T_i, i \in 1..n : T_i \in T'_i$;
- объединение элементов крон $T_1, T_2, ..., T_n$ образует L;
- порядок терминалов в кронах согласован с порядком в L;
- пересечение элементов крон $T_1, T_2, ..., T_n$ — ϵ.

Представим эту задачу в форме задачи о выполнимости булевой формулы (SAT). Для множества лесов разбора $T'_1, T'_2, ..., T'_n$ и входной строки L формула состоит из двух частей связанных конъюнкцией(&). Первая часть состоит из конъюнкции формул описывающих каждый из лесов разбора T'_i, которая строится рекурсивно по структуре SPPF, псевдокод этой процедуры приведён в листинге 1. Имена переменных формируются из имён терминалов, их позиций во входной строке и номера текущего SPPF. Заметим, что для упакованного узла добавляются отрицания переменных, соответствующих терминалам которые не использованы в данном дереве.

Вторая часть формулы описывает условие принадлежности каждого терминала ровно одному SPPF и имеет следующий вид: $(t_1^j \ xor \ t_2^j \ xor \ ... \ xor \ t_n^j)$, где t_i^j — терминал в позиции i из SPPF T'_j.

```
function nodeToFormula(node, SPPFn) =
  match node with
```

```
  | TerminalNode(terminal,(posL,posR)) ->
    Variable(SPPFn,terminal,posL))
  | NonterminalNode(children,_)
  | IntermediateNode(children,_) ->
    result = nodeToFormula(child[0])
    for child in children[1..length(children)] do
      result = AND(result, nodeToFormula(child))
    result
  | PackedNode(left,right,(posL,posR)) ->
    if (left != NULL)
    then
      result = AND(nodeToFormula(left), nodeToFormula(right))
      if (left.posR != right.posL)
      then
        for i in [left.posR..right.posL-1] do
          result = AND(result, NOT(Variable(SPPFn,input[i],i)))
      result
    else
      nodeToFormula(right)

function SPPFtoFormula(root : NonterminalNode, SPPFn) =
  result = nodeToFormula(root, SPPFn)
  for i in [0 .. root.posL]@[root.posR .. input.Length] do
    result = AND(result, NOT(Variable(SPPFn,input[i],i)))
  result
```

Листинг 1: Функция SPPFtoFormula, преобразующая SPPF в булеву формулу

Описанная формула — экземпляр задачи SAT. Решение полученной таким образом задачи SAT следует интерпретировать так. Если решение не может быть построено, то строка не принадлежит языку, который задан изначальной контекстно-свободной грамматикой с шаффлом. В случае существования множества решений F, искомое множество деревьев разбора $R = \{T_1, T_2, ..., T_n\}$ может быть восстановлено. Для этого необходимо извлечь из каждого SPPF T'_j множество деревьев T^f_j, крона которых состоит из терминалов t_i , таких, что переменной t^j_i в решении установлено значение истины. Этот процесс следует произвести для каждого из найденных решений. Тогда

$R = \bigcup_{f \in F} (T_1^f \bigtimes T_2^f \bigtimes ... \bigtimes T_n^f)$, где F — множество всех решений задачи SAT. Таким образом, получен алгоритм для решения задачи синтаксического анализа контекстно-свободных грамматик с шаффлом.

3.2. Применение анализа CFSG в распознавании планов

Предложенный алгоритм синтаксического анализа CFSG может использоваться в решении задачи распознавания планов. Для этого необходимо построить контекстно-свободную грамматику с шаффлом для библиотеки планов данной в задаче. Этот процесс осуществляется следующими шагами:

- описанным ранее способом, построить для библиотеки планов множество контекстно-свободных грамматик со стартовыми нетерминалами $A_1 ... A_n$;
- создать продукцию вида $S \to A_1 \odot A_2 \odot ... \odot A_n$, где S — стартовый нетерминал CFSG, который не содержится в алфавите нетерминалов грамматик, построенных на предыдущем шаге.

После применения предложенного ранее алгоритма к этой грамматике и последовательности действий, будет получено множество решений — наборов деревьев, которые описывают иерархию действий в соответствии с библиотекой планов.

4. Реализация алгоритма

Предложенный в данной работе алгоритм реализован на языке F#, на базе проекта YaccConstructor. Структура решения приведена в диаграмме изображённой на рис. 7. Выделенные модули были реализованы в рамках данной работы, алгоритм Generalised LL был реализован в проекте и был переиспользован. Кроме того, для решения задачи SAT была использована библиотека “Z3” [19] от Microsoft.

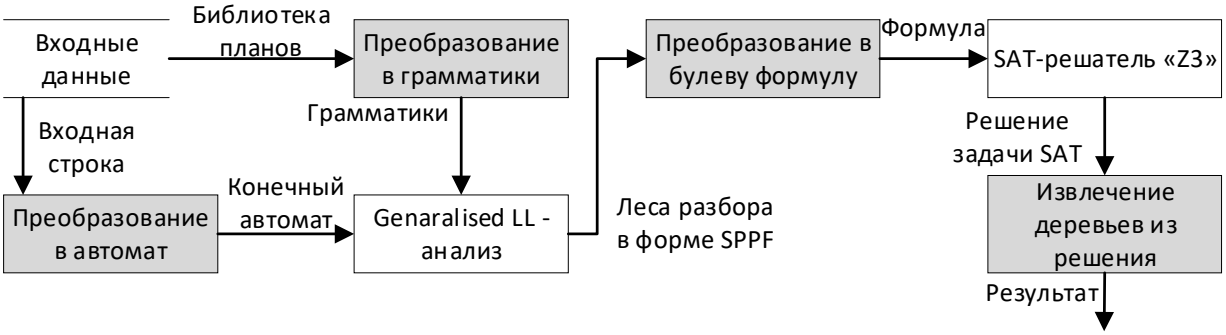


Рис. 7: Структура реализации.

5. Эксперименты

На текущий момент наиболее производительным в решении задачи распознавания планов представляется алгоритм SLIM. В связи с этим было проведено сравнение времени работы описанного алгоритма со SLIM на одной библиотеке планов. Сравнение осуществлялось с результатами авторов предложенными в статье [8]. Результаты представлены в таблице 1 (GLL + SAT — результат данной работы) и показывают превосходство алгоритма предложенного в данной работе.

Кол-во действий	1	2	3	4	5	6	7	8	9
SLIM	0.16	0.16	0.18	0.25	0.43	0.74	1.21	1.92	4.45
GLL + SAT	0.17	0.24	0.33	0.38	0.51	0.62	0.63	1.19	1.47

Таблица 1: Время работы алгоритмов.

6. Результаты

В данной работе были получены следующие результаты.

- Исследована возможность применения расширения алгоритма Generalised LL в задаче распознавания планов, путём комбинирования с SAT-решателем.
- Разработан алгоритм распознавания планов, на основе синтаксического анализа контекстно-свободных грамматик с шаффлом (CFSG). После предварительного сужения пространства поиска с помощью алгоритма Generalised LL, анализ производится SAT-решателем.
- Выполнена реализация предложенного подхода в рамках проекта YaccConstructor. Исходный код доступен в репозитории проекта: https://github.com/YaccConstructor/YaccConstructor
- Экспериментальное сравнение реализованного алгоритма с аналогом показало существенный прирост производительности.

Список литературы

[1] Amir Ofra, Gal Ya'akov Kobi. Plan recognition and visualization in exploratory learning environments // ACM Transactions on Interactive Intelligent Systems (TiiS). — 2013. — Vol. 3, no. 3. — P. 16.

[2] Berglund Martin, Björklund Henrik, Björklund Johanna. Shuffled languages—Representation and recognition // Theoretical Computer Science. — 2013. — Vol. 489-490. — P. 1 – 20.

[3] Blaylock Nate, Allen James. Fast hierarchical goal schema recognition // Proceedings of the National Conference on Artificial Intelligence / Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. — Vol. 21. — 2006. — P. 796.

[4] Charniak Eugene, Goldman Robert P. Plan recognition in stories and in life // arXiv preprint arXiv:1304.1497. — 2013.

[5] Controlling the Hypothesis Space in Probabilistic Plan Recognition. / Froduald Kabanza, Julien Filion, Abder Rezak Benaskeur, Hengameh Irandoust // IJCAI. — 2013. — P. 2306–2312.

[6] Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications / A. Biere, A. Biere, M. Heule et al. — Amsterdam, The Netherlands, The Netherlands : IOS Press, 2009. — ISBN: 1586039296, 9781586039295.

[7] Maraist John. String Shuffling over a Gap between Parsing and Plan Recognition. — 2016.

[8] Mirsky Retuh et al. SLIM: Semi-lazy inference mechanism for plan recognition // arXiv preprint arXiv:1703.00838. — 2017.

[9] Ng Hwee Tou, Mooney Raymond J. Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. // KR. — 1992. — Vol. 92. — P. 499–508.

[10] Plan recognition for exploratory learning environments using interleaved temporal search / Oriel Uzan, Reuth Dekel, Or Seri et al. // AI Magazine. — 2015. — Vol. 36, no. 2. — P. 10–21.

[11] Provoking Opponents to Facilitate the Recognition of their Intentions. / Francis Bisson, Froduald Kabanza, Abder Rezak Benaskeur, Hengameh Irandoust // AAAI. — 2011.

[12] Qin Xinzhou, Lee Wenke. Attack plan recognition and prediction using causal networks // Computer Security Applications Conference, 2004. 20th Annual / IEEE. — 2004. — P. 370–379.

[13] Ragozina Anastasiya. GLL-based relaxed parsing of dynamically generated code : Master's Thesis / Anastasiya Ragozina ; SpBU. — 2016.

[14] Scott Elizabeth, Johnstone Adrian. GLL parsing // Electronic Notes in Theoretical Computer Science. — 2010. — Vol. 253, no. 7. — P. 177–189.

[15] Scott Elizabeth, Johnstone Adrian. GLL parse-tree generation // Science of Computer Programming. — 2013. — Vol. 78, no. 10. — P. 1828 – 1844.

[16] Sequential plan recognition / Reuth Mirsky, Ya'akov Kobi Gal, Roni Stern, Meir Kalech // Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems / International Foundation for Autonomous Agents and Multiagent Systems. — 2016. — P. 1347–1348.

[17] Wiseman Sam, Shieber Stuart M. Discriminatively Reranking Abductive Proofs for Plan Recognition. // ICAPS. — 2014.

[18] YaccConstructor [Репозиторий проекта]. — Режим доступа: `https://github.com/YaccConstructor/` (дата обращения: 20.04.2018).

[19] Z3 [Репозиторий проекта]. — Режим доступа: `https://github.com/Z3Prover/z3` (дата обращения: 20.04.2018).