Санкт-Петербургский Государственный Университет

Математическое обеспечение и администрирование информационных систем

Зиннатулин Тимур Раифович

Реализация поиска путей с регулярными и контекстно-свободными ограничениями в графовой базе данных RedisGraph

Учебная практика

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

Санкт-Петербург
2021

Оглавление

Введение

Графовая модель представления данных [11] является альтернативой реляционной модели. Её особенность заключается в том, что данные представляются в виде графа. Его вершины задают сущности модели. Они могут содержать в себе метки и свойства вида "ключ-значение". Отношения, в свою очередь, задаются как ребра графа, которые содержат в себе метку — тип отношения. Графовые модели данных обладают большей гибкостью во внесении изменений, чем реляционные, в связи с чем часто являются приемлемым вариантом при проектировании баз данных. В частности, они успешно используются в социальных сетях [3], статическом анализе кода [10] и др.

Одним из способов обработки информации в таких базах данных является поиск путей между вершинами. В таком случае в запросе указываются ограничения на интересующие пользователя пути в графе. Имеется множество подходов к заданию в графе путей с ограничениями. Один из таких подходов исходит из теории формальных языков [2]. Пусть имеется помеченный граф, где метки являются символами некоторого алфавита. Тогда пути в графе соответствуют словам данного алфавита, так как являются последовательностью ребер с метками — символами алфавита. Далее, для введения ограничений на пути, задается грамматика, описывающая язык, содержащий только те слова, которые соответствуют нужным путям в графе.

Наибольший интерес среди таких грамматик представляют контекстно-свободные грамматики, так как они обладают большей выразительностью, чем регулярные грамматики. Они позволяют задавать более сложные отношения между вершинами в графе. К примеру, класс запросов поиска вершин, лежащих на одном уровне иерархии [4], задается контекстно-свободными, но не регулярными ограничениями. Запросы такого типа находят применение в биоинформатике [9] и при анализе вершин в RDF-хранилищах [4].

На данный момент в наиболее используемых графовых базах данных осуществляется поддержка в лучшем случае регулярных ограни-

чений, что сильно ограничивает выразительность языка запросов. Когда возникает необходимость задавать к базе более сложные запросы, разработчикам приходится вручную писать алгоритмы для решения задачи контекстно-свободной достижимости для их частного случая.

Единственная полноценная поддержка контекстно-свободных запросов была реализована Арсением Тереховым [12] на основе графовой базы данных RedisGraph [8]. В ней используется алгоритм Рустама Азимова [1], основанный на матричных операциях. На практике этот алгоритм является одним из самых быстрых алгоритмов выполнения контекстно-свободных запросов. Но для завершения интеграции алгоритма требуется написать тесты на корректность работы нововведенной функциональности, расширить валидацию запросов, поддерживаемых новым синтаксисом, а также обновить версию RedisGraph до более современной. В данной работе будет завершено внедрение матричного алгоритма в RedisGraph и проведено исследование полученной реализации в реальных условиях.

1. Постановка цели и задач

Целью данной работы является полная интеграция матричного алгоритма в RedisGraph и исследование его производительности. Для достижения цели были поставлены следующие задачи.

- Улучшить имеющуюся поддержку базой данных RedisGraph запросов с контекстно-свободными ограничениями.
- Внедрить реализацию матричного алгоритма в основной репозиторий RedisGraph.
- Исследовать производительность полученной реализации.

2. Обзор

Для описания работы матричного алгоритма введем понятие ослабленной нормальной формы Хомского.

Определение 2.1. Пусть G = ⟨Σ,N,P,S⟩ — контекстно-свободная грамматика, тогда G находится в ослабленной нормальной форме Хомского (ОНФХ), если содержит только правила вида:

- $A \to BC$, где $A, B, C \in N$
- $A \to a$, где $A \in N, a \in \Sigma$
- $A \to \epsilon$, где $A \in N$

ОНФХ отличается от нормальной формы Хомского наличием правил вида $A \to \epsilon$, где A — любой нетерминал, то есть A не обязательно является стартовым, а также допущением использовать стартовый нетерминал в правых частях правил.

2.1. Алгоритм, основанный на матричном умножении

Алгоритм Рустама Азимова [1] решения задачи контекстно-свободной достижимости основан на обычном произведении матриц, благодаря чему он достигает хорошей производительности. На вход алгоритму поступает помеченный граф, который представлен в виде набора булевых матриц смежности для каждой метки, и контекстно-свободная грамматика, представленная в ОНФХ.

Основная идея алгоритма заключается в рассмотрении отдельно правил, где в правой части находится один терминал, вида $A \to a$, и правил, где в правой части находятся два нетерминала, вида $A \to BC$. Для первого типа правил о наличии искомого пути говорит наличие дуги с меткой в виде терминала a. Для второго типа правил предлагается умножать матрицы смежности, соответствующие двум нетерминалам

B и C. Такая операция позволяет соединить дугой вершины, между которыми есть путь, состоящий из 2 дуг, первая из которых с меткой B, а вторая — с C, тем самым гарантируя обнаружение путей, выводимых из нетерминала A.

2.2. GraphBLAS

GraphBLAS [6] — это матричный фреймворк, позволяющий работать с графами в терминах линейной алгебры. Он задает набор базовых операций для разреженных матриц над полукольцами.

Стандарт GraphBLAS отлично подходит для реализации алгоритма, основанного на матричном умножении. В частности, при его реализации была использована библиотека SuiteSparse[1], реализующая стандарт GraphBLAS. Эта библиотека является многопоточной и хорошо оптимизирована, что позволяет добиться высокой производительности.

2.3. RedisGraph

RedisGraph [8] — это высокопроизводительная графовая база данных, поддерживающая язык запросов Cypher. RedisGraph работает с графами в виде разреженных матриц смежности и транслирует запросы языка Cypher в матричные выражения.

Из-за представления графов в виде разреженных матриц смежности RedisGraph идеально подходит для реализации на его основе матричного алгоритма, в связи с чем была выбрана для полноценной реализации этого алгоритма на основе графовой базы данных.

В рамках работы Арсения Терехова [12] была реализована поддержка матричного алгоритма. До полного внедрения требуется написать тесты на работоспособность новой функциональности и расширить валидатор запросов.

[1]SuiteSparse: A Suite of Sparse Matrix Software: `https://people.engr.tamu.edu/davis/suitesparse.html` Accessed: 2021-11-11

3. Описание реализации

В рамках учебной практики была улучшена текущая реализация матричного алгоритма на базе RedisGraph: были написаны тесты для проверки работоспособности алгоритма и был расширен валидатор запросов.

3.1. Тесты

В RedisGraph тестирование корректности запросов релазиовано на языке Python с использованием библиотеки RLTest[2], разработанной компанией RedisLabs.

К тестовому графу делаются запросы на языке Cypher с реализованным синтаксисом из работы прошлого года [13]. Проверку проходят как обычные шаблоны путей, так и реализованные с их помощью контекстно-свободные запросы.

Также после тестирования с помощью библиотеки Valgrind[3] проводится проверка на утечки памяти.

3.2. Валидация запросов

В RedisGraph основной частью обработки запроса является построение плана его выполнения. Её часть, которая относится к работе с контекстно-свободными ограничениями, приведена на рисунке 1. В ней зелёным цветом выделено то, что было добавлено или расширено в работе [12]. Сразу после получения запроса на языке Cypher строится абстрактное синтаксическое дерево (AST). Перед тем, как строить что-либо по полученному дереву, необходимо провести его валидацию — проверку на корректность введенного запроса. Желтым цветом выделен этап валидации, который был реализован для нового синтаксиса языка Cypher.

[2]RLTest — Redis Labs Test Framework: `https://github.com/RedisLabsModules/RLTest` Accessed: 2020-12-14

[3]Valgrind — system for debugging and profiling Linux programs: `https://valgrind.org/` Accessed: 2020-12-14

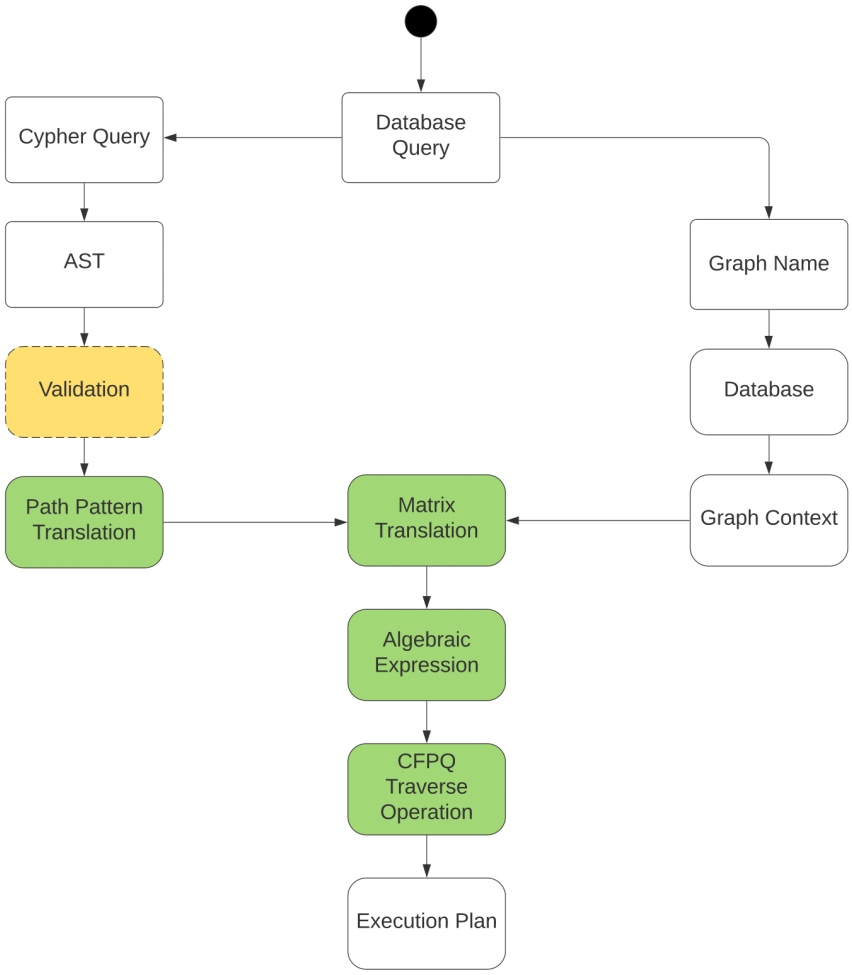

Рис. 1: Улучшение поддержки базой данных RedisGraph контекстно-свободных запросов

Программа валидации запросов написана на языке C. В ней задействуется реализация библиотеки libcypher-parser, описанная в работе [13]. Эта библиотека предназначена для разбора языка Cypher в абстрактное синтаксическое дерево и его анализа[4].

Валидатор был расширен до состояния поддержки всех возможных вариантов поддерева ACT, отвечающих за запросы с контекстно-свободными ограничениями.

[4]Используемая версия: `https://github.com/YaccConstructor/libcypher-parser/tree/path-patterns-support-for-redisgraph` Accessed: 2020-12-14

3.3. Интеграция с RedisGraph

Была проведена работа по интеграции в текущую на тот момент версию RedisGraph. По итогу работы Арсением Тереховым был создан pull request[5] в исходный репозиторий.

3.4. Доработка реализации

Также было замечено, что в полученной реализации при вычислении транзитивного замыкания использовалось полукольцо LOR_LAND, в то время как полукольцо ANY_PAIR в контексте использования в реализации алгоритма выполняет те же функции, что и LOR_LAND, но его операторы считают значения быстрее[6].

Вследствие этого была создана еще одна версия реализации матричного алгоритма, где при вычислении транзитивного замыкания вместо полукольца LOR_LAND было использовано полукольцо ANY_PAIR, что потенциально должно было ускорить работу алгоритма.

[5]Pull request: `https://github.com/RedisGraph/RedisGraph/pull/1640` Accessed: 2021-10-10

[6]SuiteSparse User Guide:`https://fossies.org/linux/SuiteSparse/GraphBLAS/Doc/GraphBLAS_UserGuide.pdf` Accessed: 2021-11-11

4. Эксперименты

После завершения интеграции для двух полученных решений были произведены замеры производительности при обработке запросов с регулярными грамматиками. Затем на одной из реализаций было проведено исследование производительности на запросах с контекстно-свободными ограничениями.

Для проведения экспериментов с над полученной реализацией был использован ПК с операционной системой Ubuntu 18.04 и конфигурацией: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz CPU, DDR4 64 Gb RAM. Нижепредставленные графы и грамматики для построения шаблонов путей были взяты из набора данных CFPQ_Data[7], собранного исследователями лаборатории языковых инструментов JetBrains Research.

4.1. Сравнение реализаций на регулярных запросах

Для запросов с регулярными ограничениями был выбран граф geospecies. Этот граф использовался для исследования алгоритма, основанного на тензорном произведении [5] и оказался одним из наиболее показательных графов на регулярных запросах. Метаданные графа geospecies представлены в таблице 1.

По этому графу была сгенерирована статистика количества вхождения меток ребер, и самые частовстречающиеся из них были выбраны как символы алфавита. По этому алфавиту были составлены регулярные выражения, которые использовались для задания запросов. Эти выражения представлены в таблице 2.

Graph name	#vertices	#edges	#A	#B	#C
geospecies	450 609	2 201 532	127 098	127 055	109 608

Таблица 1: Метаданные используемого графа

На рис. 2 представлен вид запроса на языке Cypher с расширенным синтаксисом, задающий ограничение на пути в виде регулярного

[7]CFPQ_Data: Data for Context-Free Path Querying Evaluation: `https://jetbrains-research.github.io/CFPQ_Data/index.html` Accessed: 2021-10-11

выражения Q_2.

Name	Query
Q_1	A*
Q_2	A · B*
Q_3	A · B* · C*
Q_4	(A | B)*
Q_5	A · B* · C
Q_6	A* · B*
Q_7	A · B · C*

Таблица 2: Регулярные выражения для задания запрсов

```
PATH PATTERN S = ()-/ :A :B* /->()
MATCH ()-/ ~S /->()
RETURN COUNT(*)
```

Рис. 2: Вид запроса на примере Q_2

Query	Result	Binary_Op		Any_Pair	
		time(s)	SD(s)	time(s)	SD(s)
Q_1	577 707	20,9	0,2	20,9	0,1
Q_2	6 588 862	8,7	0,1	8,7	0,1
Q_3	6 779 512	24,0	0,2	24,0	0,2
Q_4	7 280 991	55,4	0,2	55,3	0,1
Q_5	15 252	8,6	0,1	8,6	0,1
Q_6	7 163 984	56,0	0,0	56,0	0,1
Q_7	6 652 414	11,7	0,0	11,6	0,0

Таблица 3: Сравнение разных версий RedisGraph на регулярных запросах

Для каждой версии RedisGraph создавался отдельный docker-образ, на который посылались запросы. Каждый запрос запускался 5 раз, и время его работы усреднялось. Результирующее время было взято с точностью до десятых.

В таблице 3 представлены результаты замеров регулярных запросов на двух разных версиях реализации матричного алгоритма в RedisGraph.

В колонках "Binary_Op" представлены время выполнения запросов $Q_{1..7}$ старой версией алгоритма и их стандартное отклонение в секундах, а в колонках "Any_Pair" — время работы версии алгоритма с использованием полукольца ANY_PAIR и стандартное отклонение, также в секундах.

Эксперименты показали, что использование полукольца ANY_PAIR никак ощутимо не повлияло на время исполнения запросов с регулярными ограничениями. В дальнейшем для замеров будет использована версия алгоритма с полукольцом ANY_PAIR.

4.2. Эксперименты с контекстно-свободными запросами

Далее были проведено исследование работоспособности и производительности полностью интегрированного матричного алгоритма на базе RedisGraph при обработке контекстно-свободных запросов.

4.2.1. MemoryAliases

Для проведения замеров запросов с контекстно-свободными ограничениями были выбраны графы класса MemoryAliases — графы, использующиеся для анализа указателей языка Си. Метаданные графов MemoryAliases представлены в таблице 4.

Graph	#vertices	#edges	#A	#D
wc	332	269	113	156
bzip2	632	556	259	297
pr	815	692	333	359
ls	1687	1453	703	750
gzip	2687	2293	1075	1218

Таблица 4: Метаданные графов MemoryAliases

Запрос, используемый для замеров, представлен на рис. 3. Для каждого графа данный запрос запускался 5 раз, было взято среднее время работы с точностью до десятых.

```
PATH PATTERN V1 = ()-/ [ ~V2 <:A ~V1 ] | () /->()
PATH PATTERN V2 = ()-/ ~S | () /->()
PATH PATTERN V3 = ()-/ [ :A ~V2 ~V3 ] | () /->()
PATH PATTERN S = ()-/ <:D ~V1 ~V2 ~V3 :D /->()
MATCH ()-/ ~S /->()
RETURN COUNT(*)
```

Рис. 3: Структура запроса для MemoryAliases

Graph	Result	Time(ms)	SD(ms)
wc	156	33,1	1,2
bzip2	315	67,6	13,5
pr	385	124,2	5,1
ls	854	98,9	1,5
gzip	1458	162,0	1,8

Таблица 5: Результаты запросов к графам MemoryAliases

В таблице 5 представлены результаты экспериментов над графами MemoryAliases. Результаты экспериментов соотносятся с результатами, полученными в предыдущих исследованиях [7].

4.2.2. RDF

Также для исследования были взяты графы из набора RDF — реальные биологические данные. Их метаданные представлены в таблице 6. Для них были составлены классические запросы same-generation, используемые в других работах по исследованию запросов с контекстно-свободными ограничениями. Данные запросы представлены в синтаксисе Cypher на рис. 4.

Graph	#vertices	#edges	#SCO	#T	#BT
pathways	6238	12 363	3117	3118	—
gohierarchy	45 007	490 109	490 109	—	—
enzyme	48 815	86 543	8163	14 989	—
eclass514_en	239 111	360 248	90 962	72 517	—
go	582 929	1 758 432	94 514	226 481	—
geospecies	450 609	2 201 532	—	89 062	20 867

Таблица 6: Метаданные графов с RDF-данными

```
#Query_1
PATH PATTERN S1 = ()-/ [ <:SCO [ ~S1 | () ] :SCO ] |
                       [ <:T [ ~S1 | () ]  :T ] /->()
MATCH ()-/ ~S1 /->()
RETURN COUNT(*)

#Query_2
PATH PATTERN S2 = ()-/ <:SCO [ ~S2 | () ] :SCO /->()
MATCH ()-/ ~S2 /->()
RETURN COUNT(*)

#Query_geo
PATH PATTERN S3 = ()-/ :BT [ ~S3 | () ]  <:BT /->()
MATCH ()-/ ~S3 /->()
RETURN COUNT(*)
```

Рис. 4: Запросы same-generation

Замеры по каждому графу и каждому запросу проводились таким же образом, как в испытаниях на графах MemoryAliases.

Graph	Query_1			Query_2			Query_geo		
	result	time(s)	SD(s)	result	time(s)	SD(s)	result	time(s)	SD(s)
pathways	884	0,1	0,0	887	0,1	0,0	—	—	—
gohierarchy	588 976	0,8	0,0	588 976	0,7	0,0	—	—	—
enzyme	396	0,3	0,0	394	0,2	0,0	—	—	—
eclass514_en	90 994	6,7	0,0	90 988	4,1	0,0	—	—	—
go	640 316	149,3	0,6	640 305	96,9	0,1	—	—	—
geospecies	85	6,5	0,1	—	—	—	226 669 749	58,1	0,8

Таблица 7: Результаты экспериментов с контекстно-свободными запросами к RDF-данным

Результаты экспериментов соотносятся с результатами, полученными в предыдущих исследованиях, проведенных на этих наборах данных [12]. Время исполнения запросов на RedisGraph с полностью интегрированным алгоритмом больше, чем в других исследованиях [1] того же алгоритма, но это ожидаемый результат в силу издержек, возникающих при выполнении запроса внутри графовой базы данных.

В дальнейшем требуется сравнить скорость работы полученной реализации с другими графовыми базами данных, поддерживающими запросы с регулярными ограничениями.

Заключение

В рамках учебной практики были выполнены следующие задачи:

- Улучшена имеющаяся поддержка базой данных RedisGraph запросов с контекстно-свободными ограничениями.
- Завершено внедрение матричного алгоритма в RedisGraph.
- Проведено экспериментальное исследование производительности полученной реализации.
- Результаты работы изложены в статье "Multiple-Source Context-Free Path Querying in Terms of Linear Algebra", опубликованной на конференции EDBT 2021

В дальнейшем планируется продолжить проведение экспериментального исследования производительности алгоритма на регулярных запросах и контекстно-свободных запросах и сравнить результаты с существующими аналогами. Также в будущем планируется разработать подробную пользовательскую документацию запросов в расширенном синтаксисе Cypher.

Текущее состояние репозитория можно увидеть на GitHub[8].

[8]Рабочая ветвь: `https://github.com/YaccConstructor/RedisGraph/tree/any_pair_traversal` Accessed: 2021-11-04.

Список литературы

[1] Azimov Rustam, Grigorev Semyon. Context-Free Path Querying by Matrix Multiplication // Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences Systems (GRADES) and Network Data Analytics (NDA). — GRADES-NDA '18. — New York, NY, USA : Association for Computing Machinery, 2018. — Accessed: 2020-12-14. Access mode: https://doi.org/10.1145/3210259.3210264.

[2] Barrett Chris, Jacob Riko, Marathe Madhav. Formal-Language-Constrained Path Problems // SIAM J. Comput. — 2000. — 01. — Vol. 30. — P. 809–837. — Accessed: 2020-12-14.

[3] Chaudhary Anoop, FAISAL ABDUL. Role of graph databases in social networks. — 2016. — 06. — Accessed: 2020-12-14.

[4] Zhang Xiaowang, Feng Zhiyong, Wang Xin et al. Context-Free Path Queries on RDF Graphs. — 2016. — Accessed: 2020-12-14. 1506.00743.

[5] Context-Free Path Querying by Kronecker Product / Egor Orachev, Ilya Epelbaum, Rustam Azimov, Semyon Grigorev. — 2020. — 08. — P. 49–59. — ISBN: 978-3-030-54831-5. — Accessed: 2020-12-14.

[6] Jeremy Kepner. GraphBLAS Mathematics - Provisional Release 1.0. — 2017. — http://www.mit.edu/~kepner/GraphBLAS/GraphBLAS-Math-release.pdf. — Accessed: 2020-12-14.

[7] Parser Combinators for Context-Free Path Querying / Ekaterina Verbitskaia, Ilya Kirillov, Ilya Nozkin, Semyon Grigorev // Proceedings of the 9th ACM SIGPLAN International Symposium on Scala. — Scala 2018. — New York, NY, USA : Association for Computing Machinery, 2018. — P. 13–23. — Accessed: 2021-11-09. Access mode: https://doi.org/10.1145/3241653.3241655.

[8] RedisGraph - a graph database module for Redis. — https://oss.redislabs.com/redisgraph/. — Accessed: 2020-12-14.

[9] Sevon Petteri, Eronen Lauri. Subgraph Queries by Context-free Grammars // Journal of Integrative Bioinformatics. — 2008. — 06. — Vol. 5. — Accessed: 2020-12-14.

[10] Urma Raoul-Gabriel, Mycroft Alan. Source-code queries with graph databases—with application to programming language usage and evolution // Science of Computer Programming. — 2015. — Vol. 97. — P. 127 – 134. — Special Issue on New Ideas and Emerging Results in Understanding Software. Access mode: http://www.sciencedirect.com/science/article/pii/S0167642313002943 (online; accessed: 2020-12-14).

[11] Yannakakis Mihalis. Graph-Theoretic Methods in Database Theory // Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. — PODS '90. — New York, NY, USA : Association for Computing Machinery, 1990. — P. 230–242. — Access mode: https://doi.org/10.1145/298514.298576 (online; accessed: 2020-12-14).

[12] Выпускная квалификационная работа "Разработка матричного алгоритма поиска путей с контекстно-свободными ограничениями для RedisGraph" Терехова А.К. — https://se.math.spbu.ru/thesis/texts/Terehov_Arsenij_Konstantinovich_Bachelor_Thesis_2020_text.pdf. — Accessed: 2021-10-10.

[13] Учебная практика "Поддержка контекстно-свободных ограничений в языке Cypher" Зиннатулин Т.Р. — https://oops.math.spbu.ru/SE/YearlyProjects/vesna-2020/YearlyProjects/vesna-2020/mo-2nd-course/Zinatullin-report.pdf. — Accessed: 2020-12-14.