ДОПОВІДІ
НАЦІОНАЛЬНОЇ
АКАДЕМІЇ НАУК
УКРАЇНИ

ІНФОРМАТИКА

doi: https://doi.org/10.15407/dopovidi2018.11.019
УДК 004.8

А.Ф. Кургаев

Институт кибернетики им. В.М. Глушкова НАН Украины, Киев
E-mail: afkurgaev@ukr.net

Новое определение SPARQL — языка запросов Semantic Web

Представлено академиком НАН Украины А.В. Палагиным

Дано в метаязыке нормальных форм знаний (НФЗ) текстовое и часть графического описания синтаксиса языка SPARQL структурированных запросов Semantic Web. Наличие таких описаний гарантирует реализуемость языка SPARQL с реализацией интерпретатора метаязыка НФЗ. Показано, что выразительные возможности метаязыка НФЗ для формального описания SPARQL вполне сопоставимы с выразительными возможностями метаязыка Extended Backus-Naur Form.

Ключевые слова: *метаязык нормальных форм знаний, формальное описание, язык SPARQL, Semantic Web.*

Для извлечения нужных данных из непрерывно возрастающих коллекций публичных и частных данных все шире используется язык запросов SPARQL, упрощающий доступ как к структурированным данным семантического веб-проекта, так и к данным на разных платформах.

SPARQL (рекурсивный акроним от англ. SPARQL Protocol and RDF Query Language) — язык запросов к данным, представленным согласно модели RDF, а также протокол для передачи этих запросов и ответов на них.

SPARQL упрощает интеграцию хранилища данных на многих предприятиях за счет широкого выбора инструментов и библиотек приложений для извлечения, обновления, микширования и сопоставления RDF доступных данных [1].

Целью статьи является экспериментальное исследование выразительных возможностей метаязыка нормальных форм знаний (НФЗ) [2—5] на примере описания языка запросов SPARQL.

1. Общая характеристика SPARQL. SPARQL — это язык запросов Semantic Web, который позволяет [6—8]:

извлекать значение из структурированных и полуструктурированных данных;

исследовать данные, запрашивая неизвестные отношения;

выполнять сложные объединения разрозненных баз данных в одном простом запросе;

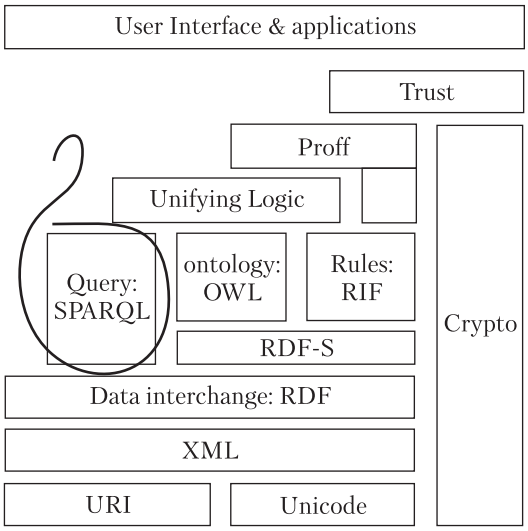

Рис. 1. Место SPARQL в семантическом стеке Тима Бернерс-Ли [9]

преобразовывать RDF данные из одного словаря в другой.

Запрос SPARQL включает в себя:

префиксные декларации для сокращения URI;

определение набора данных, которого касается RDF запрос;

утверждение результата, определяющего информацию, возвращаемую из запроса;

шаблон запроса для поиска в базовом наборе данных;

модификаторы запросов, нарезку, упорядочение и другие изменения результатов запроса.

SPARQL базируется на сопоставлении графов, имеет возможности для запроса необходимых и необязательных графических паттернов (англ. **pattern** — образец, шаблон; форма, модель; схема, диаграмма), также поддерживает агрегацию, вложенные запросы, отрицания, создание значений с помощью выражений тестирования и исходного RDF-графа. Результатами SPARQL запросов могут быть наборы результатов или RDF графы.

SPARQL имеет четыре формы запросов:

SELECT возвращает все или подмножество переменных, связанных со значениями согласно шаблону запроса. Может вводить и новые переменные.

CONSTRUCT возвращает RDF граф, сконструированный заменой переменных в наборе триплета шаблонов.

ASK возвращает логический результат проверки наличия решения для шаблона запроса.

DESCRIBE возвращает один результирующий RDF граф, описывающий найденные ресурсы.

Ключевое слово FILTER выражает ограничение на решения по всей группе, выбранной графовым шаблоном. В частности, FILTER исключает произвольные решения, которые при подстановке в выражение приведут к эффективному булевому значению false или вызовут ошибку.

Выражения SPARQL строятся согласно грамматике и обеспечивают доступ к функциям (именованным IRI) и операторам (вызываются ключевыми словами и символами в грамматике SPARQL). Грамматика SPARQL идентифицирует набор операторов (например, &&, *, isIRI), используемых для построения ограничений. Операторы SPARQL [6, разделы 17.3, 17.4] могут использоваться для сопоставления значений типизированных литералов.

Место SPARQL в структуре языков Semantic Web дано на рис. 1.

2. Формальное определение языка SPARQL в метаязыке НФЗ. В данном изложении использована последняя, доступная автору, официальная редакция нормативных документов языка SPARQL, принятая консорциумом W3C [6]. В нормативном описании языка SPARQL правила с именами в верхнем регистре употребляются в качестве терминалов. В грамматику есть две точки входа: Queryunit для запросов и Updateunit для обновления.

В приведенном ниже формальном описании языка SPARQL использованы следующие метасимволы метаязыка НФЗ [2, 3, 5]:

«=» — разделитель, отделяет имя понятия (нетерминала) от его определения;

«;» — конец определения понятия;

« » (пробел) — отношение конкатенации;

«/» — отношение альтернативного выбора;

«(«, »)» — итерационные скобки обрамляют повторяемую (нуль или больше раз) структуру понятий;

«^» — отношение отрицания примыкающего понятия;

« ′ » — текстовая кавычка;

true — тождественно истинное понятие с пустым объемом;

знаки /* и */ обрамляют комментарий.

В нотации метаязыка НФЗ текстуальное определение SPARQL дано в таблице, а первые две страницы графического определения — на рис. 2 и рис. 3.

Определение языка SPARQL в метаязыке НФЗ

1	QueryUnit	=	Query;
2	Query	=	Prologue Query_ ValuesClause;
2a	Query_		SelectQuery / ConstructQuery / DescribeQuery / AskQuery;
3	UpdateUnit	=	Update;
4	Prologue	=	(BaseDecl / PrefixDecl);
5	BaseDecl	=	'BASE' IRIREF;
6	PrefixDecl	=	'PREFIX' PNAME_NS IRIREF;
7	SelectQuery	=	SelectClause (DatasetClause) WhereClause SolutionModifier;
8	SubSelect	=	SelectClause WhereClause SolutionModifier ValuesClause;
9	SelectClause	=	'SELECT' SelectClause_a SelectClause_b;
9a	SelectClause_a	=	'DISTINCT' / 'REDUCED' / true;
9b	SelectClause_b	=	SelectClause_c (SelectClause_c) / '*' ;
9c	SelectClause_c	=	Var / '(' Expression 'AS' Var ')' ;
10	ConstructQuery	=	'CONSTRUCT' ConstructQuery_a;
10a	ConstructQuery_a	=	ConstructTemplate (DatasetClause) WhereClause SolutionModifier / (DatasetClause) 'WHERE' '{' ConstructQuery_b '}' SolutionModifier ;
10b	ConstructQuery_b	=	TriplesTemplate / true;
11	DescribeQuery	=	'DESCRIBE' DescribeQuery_a (DatasetClause) DescribeQuery_b SolutionModifier;
11a	DescribeQuery_a	=	VarOrIri (VarOrIri) / '*';
11b	DescribeQuery_b	=	WhereClause / true;
12	AskQuery	=	'ASK' (DatasetClause) WhereClause SolutionModifier;
13	DatasetClause	=	'FROM' DatasetClause_a;
13a	DatasetClause_a	=	DefaultGraphClause / NamedGraphClause ;
14	DefaultGraphClause	=	SourceSelector;
15	NamedGraphClause	=	'NAMED' SourceSelector;
16	SourceSelector	=	iri;
17	WhereClause	=	WhereCl GroupGraphPattern;

Продолжение табл.

17a	WhereCl	=	'WHERE'/ true;
18	SolutionModifier	=	SolMod_a SolMod_b SolMod_c SolMod_d;
18a	SolMod_a	=	GroupClause / true;
18b	SolMod_b	=	HavingClause / true;
18c	SolMod_c	=	OrderClause / true;
18d	SolMod_d	=	LimitOffsetClauses / true;
19	GroupClause	=	'GROUP' 'BY' (GroupCondition);
20	GroupCondition	=	BuiltInCall / FunctionCall / '(' Expression GrCond ')' / Var;
20a	GrCond	=	'AS' Var / true;
21	HavingClause	=	'HAVING' (HavingCondition);
22	HavingCondition	=	Constraint;
23	OrderClause	=	'ORDER' 'BY' (OrderCondition);
24	OrderCondition	=	OrdCond Bracketted Expression / Constraint / Var ;
24a	OrdCond	=	'ASC' / 'DESC';
25	LimitOffsetClauses	=	LimitClause LimOffCl_a / OffsetClause LimOffCl_b;
25a	LimOffCl_a	=	OffsetClause / true;
25b	LimOffCl_b	=	LimitClause / true;
26	LimitClause	=	'LIMIT' INTEGER;
27	OffsetClause	=	'OFFSET' INTEGER;
28	ValuesClause	=	'VALUES' DataBlock / true;
29	Update	=	Prologue Upd_a ;
29a	Upd_a	=	Update1 Upd_b / true;
29b	Upd_b	=	';' Update / true;
30	Update1	=	Load / Clear / Drop / Add / Move / Copy / Create / InsertData / DeleteData / DeleteWhere / Modify;
31	Load	=	'LOAD' Load_a iri Load_b;
31a	Load_a	=	'SILENT'/ true;
31b	Load_b	=	'INTO' GraphRef / true;
32	Clear	=	'CLEAR' Load_a GraphRefAll;
33	Drop	=	'DROP' Load_a GraphRefAll;
34	Create	=	'CREATE' Load_a GraphRef;
35	Add	=	'ADD' Load_a GraphOrDefault 'TO' GraphOrDefault;
36	Move	=	'MOVE' Load_a GraphOrDefault 'TO' GraphOrDefault;
37	Copy	=	'COPY' Load_a GraphOrDefault 'TO' GraphOrDefault;
38	InsertData	=	'INSERT DATA' QuadData;
39	DeleteData	=	'DELETE DATA' QuadData;
40	DeleteWhere	=	'DELETE WHERE' QuadPattern;
41	Modify	=	Modify_a Modify_b (UsingClause) 'WHERE' GroupGraphPattern;
41a	Modify_a	=	'WITH' iri / true;
41b	Modify_b	=	DeleteClause Modify_c / InsertClause ;
41c	Modify_c	=	InsertClause / true;
42	DeleteClause	=	'DELETE' QuadPattern;
43	InsertClause	=	'INSERT' QuadPattern;
44	UsingClause	=	'USING' UsingClause_a;

Продолжение табл.

44a	UsingClause_a	=	iri / 'NAMED' iri ;
45	GraphOrDefault	=	'DEFAULT' / GrOrDef iri;
45a	GrOrDef	=	'GRAPH' / true;
46	GraphRef	=	'GRAPH' iri;
47	GraphRefAll	=	GraphRef / 'DEFAULT' / 'NAMED' / 'ALL';
48	QuadPattern	=	'{' Quads '}';
49	QuadData	=	'{' Quads '}';
50	Quads	=	TriplTemp (QuadsNotTriples point TriplTemp);
50a	TriplTemp	=	TriplesTemplate / true;
50b	point	=	'.' / true;
51	QuadsNotTriples	=	'GRAPH' VarOrIri '{' TriplTemp '}';
52	TriplesTemplate	=	TriplesSameSubject point TriplTemp;
53	GroupGraphPattern	=	'{' GroupGraphPat '}';
53a	GroupGraphPat	=	SubSelect / GroupGraphPatternSub ;
54	GroupGraphPatternSub	=	TriplBlock (GraphPatternNotTriples point TriplBlock);
54a	TriplBlock	=	TriplesBlock / true;
55	TriplesBlock	=	TriplesSameSubjectPath point TriplBlock;
56	GraphPatternNotTriples	=	GroupOrUnionGraphPattern / OptionalGraphPattern / MinusGraphPattern / GraphGraphPattern / ServiceGraphPattern / Filter / Bind / InlineData;
57	OptionalGraphPattern	=	'OPTIONAL' GroupGraphPattern;
58	GraphGraphPattern	=	'GRAPH' VarOrIri GroupGraphPattern;
59	ServiceGraphPattern	=	'SERVICE' SIL VarOrIri GroupGraphPattern;
59a	SIL	=	'SILENT' / true;
60	Bind	=	'BIND' '(' Expression 'AS' Var ')';
61	InlineData	=	'VALUES' DataBlock;
62	DataBlock	=	InlineDataOneVar / InlineDataFull;
63	InlineDataOneVar	=	Var '{' (DataBlockValue) '}';
64	InlineDataFull	=	InlineDataFull_a '{' ('(' (DataBlockValue) ')' / NIL) '}';
64a	InlineDataFull_a	=	'(' (Var) ')' / NIL ;
65	DataBlockValue	=	iri / RDFLiteral / NumericLiteral / BooleanLiteral / 'UNDEF';
66	MinusGraphPattern	=	'MINUS' GroupGraphPattern;
67	GroupOrUnionGraphPattern	=	GroupGraphPattern ('UNION' GroupGraphPattern);
68	Filter	=	'FILTER' Constraint;
69	Constraint	=	BracketedExpression / BuiltInCall / FunctionCall;
70	FunctionCall	=	iri ArgList;
71	ArgList	=	'(' DIST Expression (',' Expression) ')' / NIL ;
71a	DIST	=	'DISTINCT' / true ;
72	ExpressionList	=	'(' Expression (',' Expression) ')' / NIL;
73	ConstructTemplate	=	'{' ConstrTempl '}';
73a	ConstrTempl	=	ConstructTriples / true;
74	ConstructTriples	=	TriplesSameSubject point ConstrTempl ;
75	TriplesSameSubject	=	VarOrTerm PropertyListNotEmpty / TriplesNode PropertyList;
76	PropertyList	=	PropList;
76a	PropList	=	PropertyListNotEmpty / true;

Продолжение табл.

77	PropertyListNotEmpty	=	Verb ObjectList (';' VerbObjList);
77a	VerbObjList	=	Verb ObjectList / true;
78	Verb	=	VarOrIri / 'a';
79	ObjectList	=	Object (',' Object);
80	Object	=	GraphNode;
81	TriplesSameSubjectPath	=	VarOrTerm PropertyListPathNotEmpty / TriplesNodePath PropertyListPath;
82	PropertyListPath	=	PropertyListPathNotEmpty / true;
83	PropertyListPathNotEmpty	=	VerbPathSimple_a ObjectListPath (';' VerbPathSimple_b);
83a	VerbPathSimple_a	=	VerbPath / VerbSimple ;
83b	VerbPathSimple_b	=	VerbPathSimple_a ObjectList / true;
84	VerbPath	=	Path;
85	VerbSimple	=	Var;
86	ObjectListPath	=	ObjectPath (',' ObjectPath);
87	ObjectPath	=	GraphNodePath;
88	Path	=	PathAlternative;
89	PathAlternative	=	PathSequence ('/' PathSequence);
90	PathSequence	=	PathEltOrInverse ('/' PathEltOrInverse);
91	PathElt	=	PathPrimary PathMod_a;
91a	PathMod_a	=	PathMod / true;
92	PathEltOrInverse	=	PathElt / '^' PathElt;
93	PathMod	=	'?' / '*' / '+';
94	PathPrimary	=	iri / 'a' / '!' PathNegatedPropertySet / '(' Path ')';
95	PathNegatedPropertySet	=	PathOneInPropertySet / '(' PathNegatedPropertySet_a ')';
95a	PathNegatedPropertySet_a	=	PathOneInPropertySet ('/' PathOneInPropertySet) / true;
96	PathOneInPropertySet	=	PathOneInPropertySet_a / '^' PathOneInPropertySet_a;
96a	PathOneInPropertySet_a	=	iri / 'a' ;
97	Integer	=	INTEGER;
98	TriplesNode	=	Collection / BlankNodePropertyList;
99	BlankNodePropertyList	=	'[' PropertyListNotEmpty ']';
100	TriplesNodePath	=	CollectionPath / BlankNodePropertyListPath;
101	BlankNodePropertyListPath	=	'[' PropertyListPathNotEmpty ']';
102	Collection	=	'(' GraphNode (GraphNode) ')';
103	CollectionPath	=	'(' GraphNodePath (GraphNodePath) ')';
104	GraphNode	=	VarOrTerm / TriplesNode;
105	GraphNodePath	=	VarOrTerm / TriplesNodePath;
106	VarOrTerm	=	Var / GraphTerm;
107	VarOrIri	=	Var / iri;
108	Var	=	VAR1 / VAR2;
109	GraphTerm	=	iri / RDFLiteral / NumericLiteral / BooleanLiteral / BlankNode / NIL;
110	Expression	=	ConditionalOrExpression;
111	ConditionalOrExpression	=	ConditionalAndExpression ('‖' ConditionalAndExpression);
112	ConditionalAndExpression	=	ValueLogical ('&&' ValueLogical);

Продолжение табл.

113	ValueLogical	=	RelationalExpression;
114	RelationalExpression	=	NumericExpression RelationalExpression_a;
114a	RelationalExpression_a	=	'=' NumericExpression / '!=' NumericExpression / '<' NumericExpression / '>' NumericExpression / '<=' NumericExpression / >=' NumericExpression / 'IN' ExpressionList / 'NOT' 'IN' ExpressionList / true;
115	NumericExpression	=	AdditiveExpression;
116	AdditiveExpression	=	MultiplicativeExpression ('+' MultiplicativeExpression / '-' MultiplicativeExpression / NumLiteral ('*' UnaryExpression / '/' UnaryExpression));
116a	NumLiteral	=	NumericLiteralPositive / NumericLiteralNegative;
117	MultiplicativeExpression	=	UnaryExpression ('*' UnaryExpression / '/' UnaryExpression);
118	UnaryExpression	=	'!' PrimaryExpression / '+' PrimaryExpression / '-' PrimaryExpression / PrimaryExpression;
119	PrimaryExpression	=	BracketedExpression / BuiltInCall / iriOrFunction / RDFLiteral / NumericLiteral / BooleanLiteral / Var;
120	BracketedExpression	=	'(' Expression ')';
121	BuiltInCall	=	BuiltInCall_1 / BuiltInCall_2;
121a	BuiltInCall_1	=	Aggregate / 'STR' '(' Expression ')' / 'LANG' '(' Expression ')' / 'LANGMATCHES' '(' Expression ',' Expression ')' / 'DATATYPE' '(' Expression ')' / 'BOUND' '(' Var ')' / 'IRI' '(' Expression ')' / 'URI' '(' Expression ')' / 'BNODE' Expr / 'RAND' NIL / 'ABS' '(' Expression ')' / 'CEIL' '(' Expression ')' / 'FLOOR' '(' Expression ')' / 'ROUND' '(' Expression ')' / 'CONCAT' ExpressionList / SubstringExpression / 'STRLEN' '(' Expression ')' / StrReplaceExpression / 'UCASE' '(' Expression ')' / 'LCASE' '(' Expression ')' / 'YEAR' '(' Expression ')' / 'MONTH' '(' Expression ')' / 'DAY' '(' Expression ')' / 'HOURS' '(' Expression ')' / 'MINUTES' '(' Expression ')' / 'SECONDS' '(' Expression ')' / 'TIMEZONE' '(' Expression ')' / 'TZ' '(' Expression ')' / 'ENCODE_FOR_URI' '(' Expression ')';
121b	BuiltInCall_2		'NOW' NIL / 'UUID' NIL / 'STRUUID' NIL / 'MD5' '(' Expression ')' / 'SHA1' '(' Expression ')' / 'SHA256' '(' Expression ')' / 'SHA384' '(' Expression ')' / 'SHA512' '(' Expression ')' / 'COALESCE' ExpressionList / 'isIRI' '(' Expression ')' / 'isURI' '(' Expression ')' / 'isBLANK' '(' Expression ')' / 'isNUMERIC' '(' Expression ')' / 'CONTAINS' '(' Expression ',' Expression ')' / 'STRSTARTS' '(' Expression ',' Expression ')' / 'STRENDS' '(' Expression ',' Expression ')' / 'STRBEFORE' '(' Expression ',' Expression ')' / 'STRAFTER' '(' Expression ',' Expression ')' / 'IF' '(' Expression ',' Expression ',' Expression ')' / 'STRLANG' '(' Expression ',' Expression ')' / 'STRDT' '(' Expression ',' Expression ')' / 'sameTerm' '(' Expression ',' Expression ')'/ RegexExpression / ExistsFunc / NotExistsFunc;

Продолжение табл.

121c	Expr	=	‘(‘ Expression ‘)’ / NIL ;
122	RegexExpression	=	‘REGEX’ ‘(‘ Expression ‘,’ Expression Express ‘)’;
122a	Express	=	‘,’ Expression / true;
123	SubstringExpression	=	‘SUBSTR’ ‘(‘ Expression ‘,’ Expression Express ‘)’;
124	StrReplaceExpression	=	‘REPLACE’ ‘(‘ Expression ‘,’ Expression ‘,’ Expression Express ')';
125	ExistsFunc	=	‘EXISTS’ GroupGraphPattern;
126	NotExistsFunc	=	‘NOT’ ‘EXISTS’ GroupGraphPattern;
127	Aggregate	=	‘COUNT’ ‘(‘ DIST Expr_a ‘)’ / ‘SUM’ ‘(‘ DIST Expression ‘)’ / ‘MIN’ ‘(‘ DIST Expression ‘)’ / ‘MAX’ ‘(‘ DIST Expression ‘)’/ ‘AVG’ ‘(‘ DIST Expression ‘)’ / ‘SAMPLE’ ‘(‘ DIST Expression ‘)’ / ‘GROUP_CONCAT’ ‘(‘ DIST Expression Aggreg ‘)’;
127a	Aggreg	=	‘;’ ‘SEPARATOR’ ‘=’ String / true;
127b	Expr_a	=	‘*’ / Expression ;
128	iriOrFunction	=	iri ArgList / iri;
129	RDFLiteral	=	String RDFLit;
129a	RDFLit	=	LANGTAG / ‘^^’ iri / true;
130	NumericLiteral	=	NumericLiteralUnsigned / NumericLiteralPositive / NumericLiteralNegative;
131	NumericLiteralUnsigned	=	INTEGER / DECIMAL / DOUBLE;
132	NumericLiteralPositive	=	INTEGER_POSITIVE / DECIMAL_POSITIVE / DOUBLE_POSITIVE;
133	NumericLiteralNegative	=	INTEGER_NEGATIVE / DECIMAL_NEGATIVE / DOUBLE_NEGATIVE;
134	BooleanLiteral	=	‘true’ / 'false';
135	String	=	STRING_LITERAL1 / STRING_LITERAL2 / STRING_LITERAL_LONG1 / STRING_LITERAL_LONG2;
136	iri	=	IRIREF / PrefixedName;
137	PrefixedName	=	PNAME_LN / PNAME_NS;
138	BlankNode	=	BLANK_NODE_LABEL / ANON;
	Продукции для терминалов		
139	IRIREF	=	‘<’ (^IRIREF_a IRIREF_b) ‘>’;
139a	IRIREF_a	=	[#x00-#x20];
139b	IRIREF_b	=	[^<>”{}] / [^`\] ;
140	PNAME_NS	=	PN_PREFIX ‘:’ / ‘:’;
141	PNAME_LN	=	PNAME_NS PN_LOCAL;
142	BLANK_NODE_LABEL	=	‘_:’ BL_NODE_LAB (PN_CHARS / ‘.’) ;
142a	BL_NODE_LAB	=	PN_CHARS_U / [0-9] ;
143	VAR1	=	‘?’ VARNAME;
144	VAR2	=	‘$’ VARNAME;
145	LANGTAG	=	‘@’ [a-zA-Z] ([a-zA-Z]) (‘-’ [a-zA-Z0-9] ([a-zA-Z0-9]));
146	INTEGER	=	[0-9] ([0-9]);
147	DECIMAL	=	([0-9]) '.' [0-9] ([0-9]);

Окончание табл.

148	DOUBLE	=	[0-9] ([0-9]) '.' ([0-9]) EXPONENT / '.' [0-9] ([0-9]) EXPONENT / [0-9] ([0-9]) EXPONENT;
149	INTEGER_POSITIVE	=	'+' INTEGER;
150	DECIMAL_POSITIVE	=	'+' DECIMAL;
151	DOUBLE_POSITIVE	=	'+' DOUBLE;
152	INTEGER_NEGATIVE	=	'-' INTEGER;
153	DECIMAL_NEGATIVE	=	'-' DECIMAL;
154	DOUBLE_NEGATIVE	=	'-' DOUBLE;
155	EXPONENT	=	[eE] sign [0-9] ([0-9]);
155a	sign	=	[+-] / true;
156	STRING_LITERAL1	=	"'" ([^#x27#x5C#xA#xD] / ECHAR) "'";
157	STRING_LITERAL2	=	'"' ([^#x22#x5C#xA#xD] / ECHAR) '"';
158	STRING_LITERAL_LONG1	=	"'''" (STR_LIT_LONG1_a STR_LIT_LONG1_b) "'''";
158a	STR_LIT_LONG1_a	=	"'" / "''" / true;
158b	STR_LIT_LONG1_b	=	[^'\] / ECHAR ;
159	STRING_LITERAL_LONG2	=	'"""' (STR_LIT_LONG2_a STR_LIT_LONG2_b) '"""';
159a	STR_LIT_LONG2_a	=	'"' / '""' / true;
159b	STR_LIT_LONG2_b	=	[^"\] / ECHAR ;
160	ECHAR	=	'\' [tbnrf\"'];
161	NIL	=	'(' (WS) ')';
162	WS	=	#x20 / #x9 / #xD / #xA;
163	ANON	=	'[' (WS) ']';
164	PN_CHARS_BASE	=	[A-Z] / [a-z] / [#x00C0-#x00D6] / [#x00D8-#x00F6] / [#x00F8-#x02FF] / [#x0370-#x037D] / [#x037F-#x1FFF] / [#x200C-#x200D] / [#x2070-#x218F] / [#x2C00-#x2FEF] / [#x3001-#xD7FF] / [#xF900-#xFDCF] / [#xFDF0-#xFFFD] / [#x10000-#xEFFFF];
165	PN_CHARS_U	=	PN_CHARS_BASE / '_';
166	VARNAME	=	BL_NODE_LAB (PN_CHARS_U / [0-9] / #x00B7 / [#x0300-#x036F] / [#x203F-#x2040]);
167	PN_CHARS	=	PN_CHARS_U / '-' / [0-9] / #x00B7 / [#x0300-#x036F] / [#x203F-#x2040];
168	PN_PREFIX	=	PN_CHARS_BASE PN_PR;
168a	PN_PR	=	(PN_CHARS / '.') PN_CHARS / true;
169	PN_LOCAL	=	PN_LOC_a (PN_CHARS / '.' / ':' / PLX) PN_LOC_b;
169a	PN_LOC_a	=	PN_CHARS_U / ':' / [0-9] / PLX ;
169b	PN_LOC_b	=	PN_CHARS / ':' / PLX / true;
170	PLX	=	PERCENT / PN_LOCAL_ESC;
171	PERCENT	=	'%' HEX HEX;
172	HEX	=	[0-9] / [A-F] / [a-f];
173	PN_LOCAL_ESC	=	'\' PN_LOC_ESC;
173a	PN_LOC_ESC	=	'_' / '~' / '.' / '-' / '!' / '$' / '&' / "'" / '(' / ')' / '*' / '+' / ',' / ';' / '=' / '/' / '?' / '#' / '@' / '%' ;

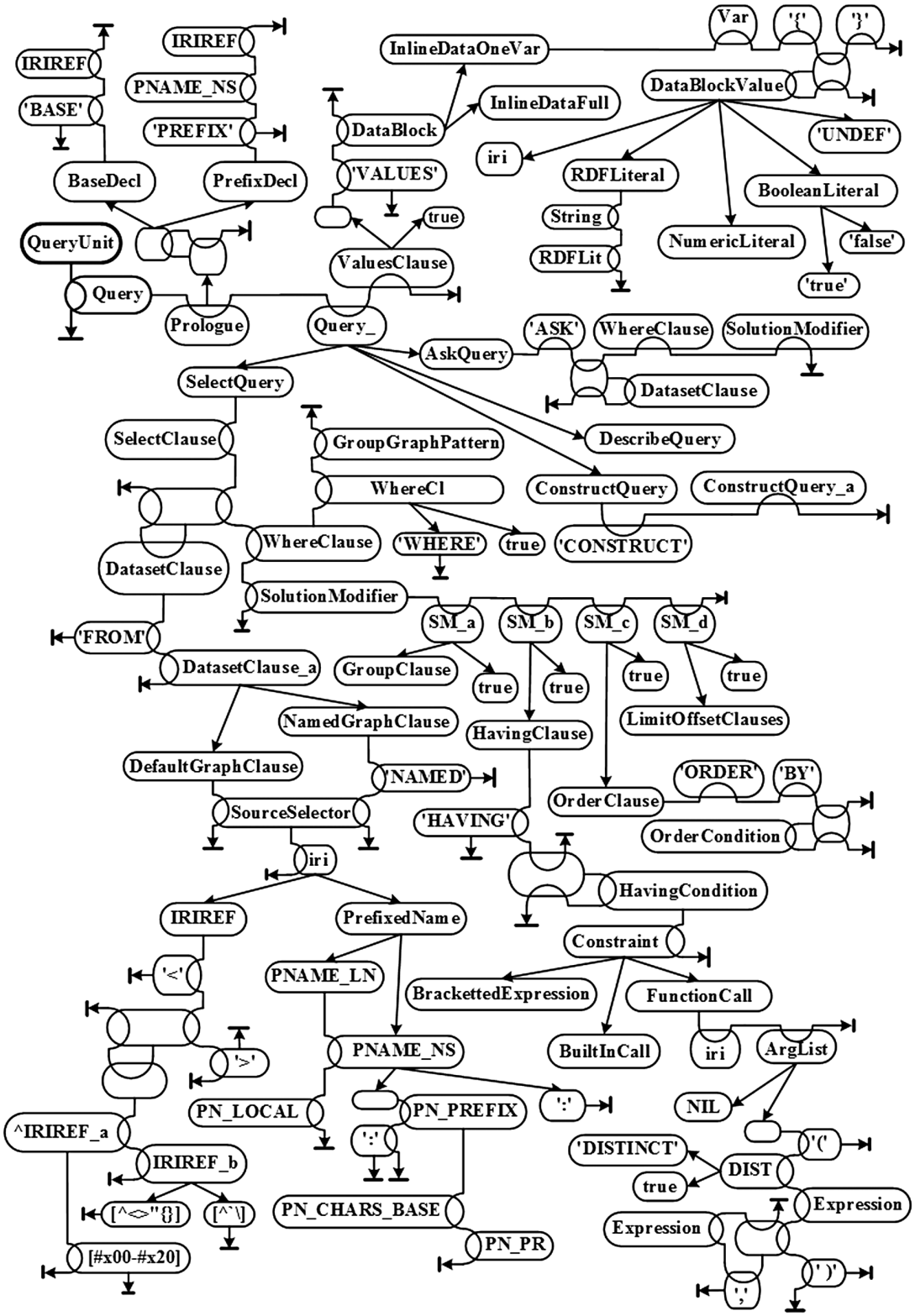

Рис. 2. Первая страница графического описания в метаязыке НФЗ синтаксиса языка SPARQL

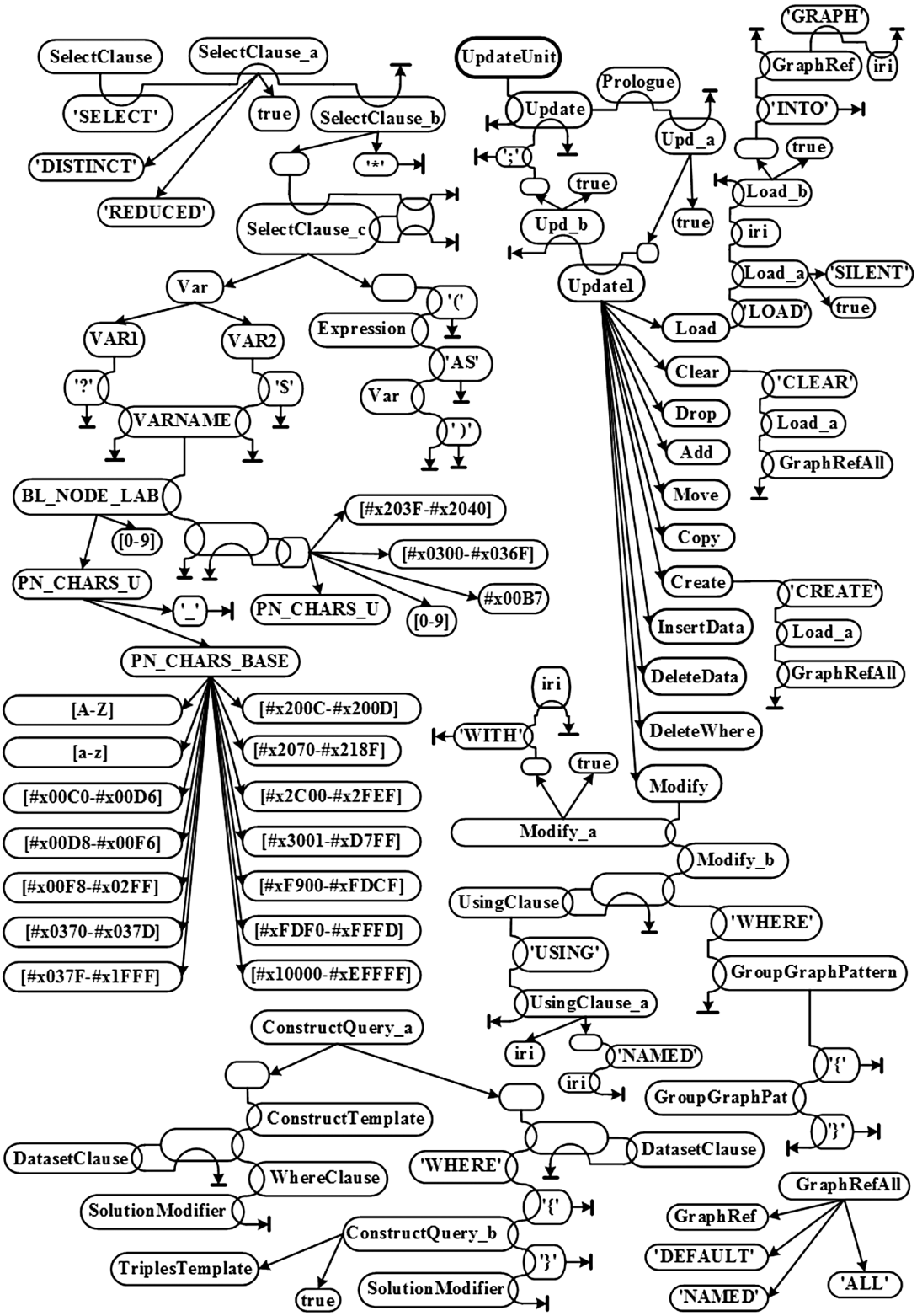

Рис. 3. Вторая страница графического описания в метаязыке НФЗ синтаксиса языка SPARQL

3. Сопоставление предлагаемого с нормативным описанием SPARQL. Нормативное описание синтаксиса языка SPARQL метаязыком EBNF включает 173 [6], а аналогичное описание метаязыком НФЗ (см. таблицу) — 236 продукций. Дополнительные правила НФЗ-описания языка SPARQL использованы для моделирования:

структурных скобок 20 правил: 2a, 9c, 10a, 13a, 24a, 41b, 44a, 53a, 64a, 83a, 96a, 114a, 116a, 121c, 127b, 142a, 158b, 159b, 169a, 173a;

необязательности 36 правил: 9a, 10b, 17a, 18a, 18b, 18c, 18d, 20a, 25a, 25b, 29a, 29b, 31a, 31b, 41a, 41c, 45a, 50a, 50b, 54a, 59a, 71a, 73a, 76a, 77a, 83b, 91a, 95a, 122a, 127a, 129a, 155a, 158a, 159a, 168a, 169b;

одного или большего числа вхождений нетерминала и структурных скобок — правило 9b;

отрицания понятий два правила 139a, 139b;

для упрощения изображения графа два правила 121a, 121b.

Главным результатом сопоставления рассмотренных описаний является

Утверждение 1. Выразительных возможностей метаязыка НФЗ достаточно для формального описания языка SPARQL запросов Semantic Web.

Таким образом исследованы выразительные возможности метаязыка НФЗ по отношению к нормативному описанию синтаксиса языка запросов SPARQL. Даны формальные текстовое и первые две страницы графического описания этого языка, чье наличие гарантирует реализуемость языка SPARQL с реализацией интерпретатора метаязыка НФЗ. Показано, что выразительные возможности метаязыка НФЗ для формального описания языка SPARQL близки выразительным возможностям метаязыка Extended Backus-Naur Form, уступая ему по числу правил из-за отсутствия структурных скобок и метасимвола необязательности.

ЦИТИРОВАННАЯ ЛИТЕРАТУРА

1. DuCharme B. Learning SPARQL, 2nd Edition. USA: O'Reilly Media, Inc. 2013. 386 p.
2. Кургаев А.Ф., Григорьев С.Н. Нормальные формы знаний. *Допов. Нац. акад. наук Укр.* 2015. № 11. С. 36—43.
3. Кургаев А.Ф., Григорьев С.Н. Метаязык нормальных форм знаний. *Кибернетика и системный анализ.* 2016. **52**, № 6. С. 11—20. doi: http://link.springer.com/article/10.1007/s10559-016-9885-3
4. Кургаев А.Ф., Григорьев С.Н. Интерпретатор универсальной машины Тьюринга. *Допов. Нац. акад. наук Укр.* 2016. № 10. С. 30—36. doi: http://dx.doi.org/10.15407/dopovidi2016.10.030
5. Кургаев А.Ф., Григорьев С.Н. Определение формальных языков в метаязыке нормальных форм знаний. *Проблеми програмування.* 2017. № 4. С. 37—50.
6. SPARQL 1.1: Query Language. W3C Recommendation 21 March 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/. (Дата обращения: 01.02.2018)
7. Feigenbaum L. SPARQL By Example: A Tutorial. VP Technology & Standards, Cambridge Semantics. URL: http://www.cambridgesemantics.com/2008/09/sparql-by-example/. (Дата звернення: 01.02.2018)
8. Introduction aux RDF & SPARQL: Training Module 1.3. Open Data Support: 2014 European Commission. URL: https://www.europeandataportal.eu/sites/default/files/d2.1.2_training_module_1.3_introduction_to_rdf_sparql_en_edp.pdf. (Дата обращения: 01.02.2018)
9. Berners-Lee T., Hall W., Hendler J.A., O'Hara K., Shadbolt N., Weitzner D.J. A Framework for Web Science. *Foundations and Trends in Web Science.* 2006. **1**, № 1. P. 1—130. doi: https://doi.org/10.1561/1800000001

Поступило в редакцию 15.05.2018

REFERENCES

1. DuCharme, B. (2013). Learning SPARQL, 2nd Edition. USA: O'Reilly Media, Inc.
2. Kurgaev, A., Grygoryev, S. (November 2015). The normal forms of knowledge. Dopov. Nac. akad. nauk. Ukr., No. 11, pp. 36-43 (in Russian).
3. Kurgaev, A., Grygoryev, S. (November 2016). Metalanguage of Normal Forms of Knowledge. Cybernetics and Systems Analysis. 52, No. 6, pp. 839-848. doi: http://link.springer.com/article/10.1007/s10559-016-9885-3
4. Kurgaev, A. & Grygoryev, S. (October 2016). The universal turing machine interpreter. Dopov. Nac. akad. nauk. Ukr., No. 10, pp. 30-36 (in Russian). doi: http://dx.doi.org/10.15407/dopovidi2016.10.030
5. Kurgaev, A. & Grygoryev, S. (2017). The definition of formal languages in the meta language of normal forms of knowledge. Programming problems. No. 4, pp. 37-50 (in Russian).
6. SPARQL 1.1: Query Language. W3C Recommendation 21 March 2013. Retrieved from: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
7. Lee Feigenbaum. SPARQL By Example: A Tutorial. VP Technology & Standards, Cambridge Semantics. Retrieved from: http://www.cambridgesemantics.com/2008/09/sparql-by-example/.
8. Introduction aux RDF & SPARQL: Training Module 1.3. Open Data Support: 2014 European Commission. R etrieved from: https://www.europeandataportal.eu/sites/default/files/d2.1.2_training_module_1.3_introduction_to_rdf_sparql_en_edp.pdf.
9. Berners-Lee, T., Hall, W., Hendler, J. A., O'Hara, K., Shadbolt, N. & Weitzner, D. J. (2006) A Framework for Web Science. Foundations and Trends in Web Science, 1, No. 1, pp. 1-130. doi: https://doi.org/10.1561/1800000001

Received 15.05.2018

О.П. Кургаєв

Інститут кібернетики ім. В.М. Глушкова НАН України, Київ
E-mail: afkurgaev@ukr.net

НОВЕ ВИЗНАЧЕННЯ SPARQL —
МОВИ ЗАПИТІВ SEMANTIC WEB

Дано у метамові нормальних форм знань (НФЗ) текстовий і частина графічного опису синтаксису мови SPARQL структурованих запитів Semantic Web. Наявність таких описів гарантує реалізуємість мови SPARQL з реалізацією інтерпретатора метамови НФЗ. Показано, що виразні можливості метамови НФЗ для формального опису SPARQL цілком порівнянні з виразними можливостями метамови Extended Backus-Naur Form.

Ключові слова: *метамова нормальних форм знань, формальний опис, мова SPARQL, Semantic Web.*

A.F. Kurgaev

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kiev
E-mail: afkurgaev@ukr.net

NEW DEFINITION OF THE SPARQL —
QUERY LANGUAGE FOR THE SEMANTIC WEB

The text and the part of a graphic description of the syntax of the SPARQL structured Query Language for the Semantic Web are described in the metalanguage of normal forms of knowledge. These descriptions guarantee that the SPARQL language can be implemented, when the interpreter of the metalanguage of normal forms of knowledge is implemented. It is demonstrated that the expressive possibilities of the metalanguage of normal forms of knowledge for the formal description of the SPARQL language are quite comparable with those of the Extended Backus-Naur Form Metalanguage.

Keywords: *metalanguage of normal forms of knowledge, formal description, SPARQL language, Semantic Web.*