Notation3 (N3): A readable RDF syntax

W3C Team Submission 28 March 2011

This version:
http://www.w3.org/TeamSubmission/2011/SUBM-n3-20110328/
Latest version:
http://www.w3.org/TeamSubmission/n3/
Previous version:
http://www.w3.org/TeamSubmission/2008/SUBM-n3-20080114/
Authors:
Tim Berners-Lee, W3C
Dan Connolly, W3C

Abstract

The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web.

This document defines *Notation 3* (also known as *N3*), an assertion and logic language which is a superset of RDF. N3 extends the RDF datamodel by adding formulae (literals which are graphs themselves), variables, logical implication, and functional predicates, as well as providing an textual syntax alternative to RDF/XML.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications can be found in the W3C technical reports index at http://www.w3.org/TR/.

This January 2008 W3C Team Submission, with its linked grammars and test suites, defines the N3 language. It is the product of experience with working code since 2000, and much discusssion in secifically the W3C RDF Interest Group and Semantic Web

Interest Groups. It may be put into W3C note form, or even Recommendation track form, in the future. There is a large amount of implementation with N3 and its subsets. There also a large number of tests. However, the test suite has not been completely vaidated against the grammar.

Table of Contents

Appendices

Introduction

This is the specification of the Notation3 language, of internet Media Type text/n3. Normative parts of the specification are thus. Non-normative parts and comments thus.

This is a language which is a compact and readable alternative to RDF's XML syntax, but also is extended to allow greater expressiveness. It has subsets, one of which is RDF 1.0 equivalent, and one of which is RDF plus a form of RDF rules.

This document is a specification of the language suitable for those familiar with the general concepts. The developer learning N3 is invited to try the A tutorial, while implementers looking for for a particular detail of the definition of the logic are steered toward the operational semantics. There is also a list of other N3 resources.

This language has ben developed in the context of the Semantic Web Interest Group. Comments on this document should be sent to public-cwm-talk@w3.org

The aims of the language are

- to optimize expression of data and logic in the same language,
- to allow RDF to be expressed,
- to allow rules to be integrated smoothly with RDF,
- to allow quoting so that statements about statements can be made, and
- to be as readable, natural, and symmetrical as possible.

The language achieves these with the following features:

- URI abbreviation using prefixes which are bound to a namespace (using @prefix) a bit like in XML,
- Repetition of another object for the same subject and predicate using a comma ","
- Repetition of another predicate for the same subject using a semicolon ";"
- Bnode syntax with a certain properties just put the properties between [and]
- Formulae allowing N3 graphs to be quoted within N3 graphs using { and }
- Variables and quantification to allow rules, etc to be expressed
- A simple and consistent grammar.

Grammar

The grammar of N3 is defined by a context free grammar:

- Context-free grammar in N3

While this obviously provides a bootstrap problem for implementers, it is a common format one would expect N3 developers to share. Non-authoritative generated copies of the same grammar are provided in the following dialects:

N3	RDF/XML	EBNF as used in XML 1.1	HTML	yacc

Test Suite

The N3 test suite provides a check for implementers.

- Test suite space

The work of validating the grammar and test suite against each other is not complete.

Encoding

N3 files are encoded in UTF-8 (See RFC2279), in normalized in Normalization Form C. The language is defined in terms of a sequence of Unicode characters.

(Implementations may chose to implement using 8-bit bytes, passing bytes >7F transparently, but this will not allow them to check the validity of embedded non-ASCII characters. Note also that some systems internally use UTF-16 encoding which uses 16 bits as a rule, but allows full 20-bit unicode code points to be encoded as two 16-bit units. This may be a suitable implementation for a system which passes such characters without counting them or breaking apart strings of them. A full 32 unicode implementation would normally be preferable.)

The exact set of unicode character to be allowed in (a) literal strings or (b) identifiers may need to be revised if the unicode specification is revised in unanticipated ways in the future)

MIME type text/n3; charset=utf-8

This document defines, in the linked grammar, the allowable syntax and, for documents of valid syntax, the meaning, of a document in the `text/n3` MIME type (Internet Content Type). It also defines the fragment identifier syntax and semantics.

As the type is in the text tree, agents which do not understand it will and should display it as text for human consumption.

MIME parameter: charset

This MIME type is used with a *charset* parameter: the encoding is always *utf-8.* The only occasion that the encoding can be omitted is when it happens (and is guaranteed by the sender) that all of the unicode characters in the file are in fact that subset which is the 7-bit ASCII set. This is because the default encoding for types in the text/* tree is ASCII.

Fragment identifier syntax and semantics

The fragment identifier part of a URI where the rest of the URI is that of an information resource which has a representation in N3 is used to identify any thing whatever, abstract or concrete.

The N3 representation may give information about that thing by including its URI as one of the terms in a statement.

The fragment identifier syntax should match the syntax for the part of a qname after the colon. The qname syntax can be used within the document to refer to the thing the full URI identifies.

It is good practice to use URIs of this form to identify things, and to ensure that the N3 document which is served to the inquirer contains useful information which the publisher of the information, and owner of the URI, considers relevant, useful and interesting to any agent (human or machine) dereferencing the URI. It is good practice to include statements relating the thing to other things, also identified by URIs. The use of URIs whose part before the hash sign is a different information resource constitutes a form of link, leading an agent to possibly dereference the other URI and gain more information.

Syntax details

This section describes the syntax which is not given by the grammar. If a parser is generated automatically from the grammar, the tokenizer must be written to take the following into account.

Whitespace

Tokenizing and white space is not specified in the grammar for simplicity. White space must be inserted whenever the following token could start with a character which could extend the preceding token. Whitespace may not be inserted within a token, except that it may be freely added and removed whitespace within a URI between angle brackets. This allows complex URIs to be broken onto several lines, which in turn allows N3 to be sent for example of email systems in which a limited line length.

All URIs are quoted with angle brackets. Whitespace within the <> is to be ignored. Whitespace may therefore be used on output to split a long URI between lines.

Base URI

The ***@base*** directive sets the base URI to be used for the parsing of relative URIs. It takes, itself, a relative URI, so it can be used to change the base URI relative to the previous one.

Example

```
@base  <http://example.org/products/>.
...
@base  <prod123/>.
...
@base  <../>.
```

Note that if files are to be concatenated, the base URI of one may adversely affect parsing of another, unless @base is avoided, or used consistently with absolute URIs.

Namespaces

The ***@prefix*** directive binds a prefix to a namespace URI. It indicates that a qualified name (qname) with that prefix will thereafter be a shorthand for a URI consisting of the concatenation of the namespace identifier and the bit of the qname to the right of the (only allowed) colon.

The namespace prefix may be empty, in which case the qname starts with a colon. This is known as the default namespace. The empty prefix "" is by default , bound to "#" -- the local namespace of the file. The parser behaves as though there were a

```
@prefix : <#>.
```

just before the file. This means that <#foo> can be written `:foo` and using `@keywords` one can reduce that to `foo`.

@keywords

Keywords are a very limited set of alphanumeric strings which are in the grammar prefixed by an at sign "@".

Qnames using the default namespace which has an empty prefix start with a colon. The language becomes more readable when keywords and/or qnames can be written without the at sign or colon prefix, but clearly one cannot do both.

In many languages similar to N3, there is a risk of ambiguity as to whether a naked alphanumeric string is a keyword or an identifier. Serious version management problems occur when new keywords are added to a language, changing things which were identifiers into keywords. N3 is designed to be extended in the future, and possibly branched into specific languages such as query languages. For this reason, an N3 document can declare which keywords it uses without the at sign. This allows N3 to be extended without the danger that existing documents be incorrectly interpreted by future systems, or future documents by existing systems.

If no @keywords directive is given, qualified names all have colons, and unquoted alphanumerics are all keywords. Only the keywords `is`, `of`, and `a` may be used naked.

If the **@keywords** directive is given, the keywords given will thereafter be recognized without a "@" prefix, and anything else is a local name in the default namespace. Any keyword may still be given, even if not in the keyword list, by prefixing it with "@".

Because keywords are declared in this way, we will have the freedom later to make extensions to the syntax using new keywords without fear of ambiguity. However, the tokenizer has to be aware of the `@keywords` setting. The grammar is written as without reference to the keywords system at all, on the assumption that the string has been preprocessed by a keyword processor to put a "@" on all keywords and a ":" on all qnames in the default namespace.

Strings

The """value""" string form is used simply for multi-line values or values containing quote marks.

The Unicode sets have been defined to be compatible with those used in XML 1.1. Unicode has changed over the years, and the intention of this specification would be to change if necessary. However, it is understood that the character ranges in the grammar should be stable even though the introduction of a few new code points into the Unicode system.

String escaping

Escaping in strings uses the same conventions as Python strings except for a \U extension introduced by NTriples spec. N3 strings represent ordered sequences of Unicode characters.

Some escapes (\a, \b, \f, \v) should be avoided because the corresponding characters are not allowed in RDF.

Escape Sequence	Meaning
`\`*`newline`*	Ignored
`\\`	Backslash (`\`)
`\'`	Single quote (`'`)
`\"`	Double quote (`"`)
`\n`	ASCII Linefeed (LF)
`\r`	ASCII Carriage Return (CR)
`\t`	ASCII Horizontal Tab (TAB)
`\u`*`hhhh`*	character in BMP with Unicode value U+*hhhh*
`\U00`*`hhhhhh`*	character in plane 1-16 with Unicode value U+*hhhhhh*

In N3, the double quote character is used for strings. The single quote character is reserved for future use. The single quote character does not need to be escaped in an N3 string.

**RDF and N3 are defined in terms of characters, not bytes. Therefore, the \ooo and \xhh escapes are not used.

The hexadecimal digit as in unicode escapes are UPPERCASE. This is designed to match the NTriples strings.

Semantics

This section describes the meaning of the productions in the grammar

An N3 document encodes a set of statements, and its meaning is the conjunction of the meaning of the statements.

The statement of the form `x p y.` asserts that the relation p holds between x and y. The semantics of statements, where they are valid RDF statements, are those described in the RDF abstract syntax document [RDFAS].

When p is identified by a URI, and that URI when dereferenced in the Web gives some information, that information may in practice be used to determine more information about the semantics of the statement.

In property lists, the semicolon *;* is shorthand for repeating the subject. In object lists *,* is shorthand for repeating the verb.

Shorthand for common predicates

For three URIs commonly used as predicates, N3 provides a special shorthand syntax. This may only be used in the predicate position in a statement. The RDF type predicate shorthand is the keyword `a`. (This needs no @ sign, unless @keyword is given and it is not in the keyword list.)

Shorthand	**stands for**
`a`	`<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`
`=`	`<http://www.w3.org/2002/07/owl#sameAs>`
`=>`	`<http://www.w3.org/2000/10/swap/log#implies>`
`<=`	`<http://www.w3.org/2000/10/swap/log#implies>` but in the inverse direction

Blank nodes

There are several ways in N3 of representing a blank, or unnamed node: the underscore namespace, the square bracket syntax, and the path syntax.

Underscore namespace

N3 allows has a special _: namespace prefix. An identifier of such a form (e.g. `_:a17`) represents an blank node. The name a17 is only used in the serialization to connect different mentionings of the same node. There is no commitment to it as a name. It may not be used as a URI. A serialization which uses arbitrary different node identifiers is completely equivalent, as is one which uses the other blank node syntaxes below for the same data.

When formulae are nested, _: blank nodes syntax used to only identify blank node in the formula it occurs directly in. It is an arbitrary temporary name for a symbol which is

existentially quantified within the current formula (not the whole file). They can only be used within a single formula, and not within nested formulae.

This is the one blank node syntax available even in the NTriples minimal subset of N3.

Square bracket blank node syntax

`[ pl ]` means *x*, where there exists some *x* such that x has properties in the property list `pl`. For example,

```
[ :firstname "Ora" ] dc:wrote [ dc:title "Moby Dick" ] .
```

is a statement which would be means in math

exists x, y . firstname(x, "Ora") & dc:wrote(x,y) & dc:title (y, "Moby Dick")

or in english "Some person who has a first name Ora wrote a book entitled "Moby Dick". Note **not** "*the* book" or "*the* person".

This can equally well be written

[x:firstname "Ora" ; dc:wrote [dc:title "Moby Dick"]] .

or

[] x:firstname "Ora" ; dc:wrote [dc:title "Moby Dick"].

The [] maybe read loosely as "something".

Paths

These are just shorthand. *x*!*p* means [is *p* of *x*] in the above anonymous node notation. You can read it as "x's p". This is a little reminiscent of the "." in object oriented programming "object.slot" syntax.

The reverse traversal, *x*^*p* means [*p x*] . For either forward or backward traversal, *p* is a property, and *x* can be a whole path with both ! and ^ in it.

Example:

`:joe!fam:mother!loc:office!loc:zip` Joe's mother's office's zipcode

`:joe!fam:mother^fam:mother` Anyone whose mother is Joe's mother.

Formulae

An RDF document parses to a set of statements, or graph. However RDF itelf has no datatype allowing a graph as a literal value. N3 extends RDF allows a graph itself to be

referred to within the language, where it is known as a formula. A { statementlist } is a formula whose meaning is the the logical conjunction (equivalent to syntactic juxtaposition) of the statements in the list. It is a set, as the same statement occuring more than once has no meaning. It is unordered set.

Example:

```
{ [ x:firstname  "Ora" ] dc:wrote [ dc:title  "Moby Dick" ] } a n3:falsehood .
```

This claims that the expression in {braces} is false - that there is nothing called Ora which wrote anything titled "Moby Dick".

A formula is considered, like a literal string, to be defined only by its contents.

Quantification

Apart from the set of statements, a formula also has a set of URIs of symbols which are universally quantified, and a set of URIs of symbols which are existentially quanitified. Variables are then in general symbols which have been quantified. There is a also a shorthand syntax ?x which is the same as :x except that it implies that x is universally quantified not in the formula but in its parent formula

The semantics of a formula are than the contents are quoted. Variable substitution **does** recursively take place within a formula, but substitution of equals does **not**. The variable substitution is used for example when formulae are used for rules, and patch file formats. See the tutorial introduction to rules.

Certain propoerties may, by their semantics, allow the propagataion of substitution of equals, by agents which are aware of that semantics. So for example, if the statement { F ex:or G } is true, where F and G are formulae, then it ise useful to define a disjunction operator ex:or such that if a = b, then it is also true that { F ex:or G' } where G' is the result or substituiting b for a in G.

The `@forAll` directives declare variables which are universally quantified: the formula is true for any value of the variable. Similarly, `@forSome` gives an existential quantification: there exists some value of the variable for which the formula is true. (Note also that the blank node notations above all introduce introduce a blank node, which is an unnamed existential variable)

```
@forSome <#g>. <#g> <#loves> <#you>
```

is equivalent to

```
[ <#loves> <#you> ] .
```

If both universal and existential quantification are specified for the same formula, then the scope of the universal quantification is outside the scope of the existentials:

```
@forAll <#h>. @forSome <#g>. <#g> <#loves> <#h> .
```

means

```
for all h

   for some g

     g loves h
```

("Every has someone who loves them" rather than "Somebody loves everybody")

which you might think of as

∀h(∃g(loves(g,h))

Lists

Notation3 in practice uses lists frequently, as ordered containers, as argument lists to Nary functions, and so on. Implementations may treat *list* as a data type rather than just a ladder of rdf:first and rdf:rest properties. The use of rdf:first and rdf:rest can be regarded as a reification of the list datatype.
This use of lists requires more axioms than are actually defined in the RDF specification.

1. All lists exist. The statement `[rdf:first <a>; rdf:rest rdf:nil].` carries no information in that the list (,a>) exists and this expression carries no new information about it.
2. A list has only one rdf:first. rdf:first is functional. If the same thing has rdf:first arcs from it, they must be to nodes which are RDF equivalent - are the same RDF node.
3. Lists are the same RDF node if they have the same the:first and the same rdf:rest.

This may seem obvious when you think of a list as a datatype. Occasionally people express surprise that statements like the one above which have a list as a subject but no information about it disappear when loaded into an N3 store.

Notes on Numbers

The BNF syntax above describes the syntax for various literal productions. These syntax strings identify values which are members of various classes of number. The BNF productions should not be confused with the relationship between number classes. In the syntax, integer, rational and real productions are distinct. When it comes to the values they represent, all integers are also rational numbers, and all nationals are also real numbers.

There is no distinction between the rationals 1.0 and 1/1 and the the integer 1. The semantics of rational numbers are true to normal mathematical equality.

The issue of reals is more complicated, as any real literal is necessary approximate. So while there is a real number which is equal to the rationals 1 or 1/3, reals do not support this comparison. (See XML Schema Datatypes)

Boolean literals

The words `true` and `false` are boolean literals.

Appendix: N3 Subsets

For various purposes it is useful to define limited subsets of the language. These are defined in terms of N3 by grammar.

NTriples is an *extremely* constrained language which uses hardly any syntactic sugar at all: it is optimized for reading by scripts, and comparing using text tools. It just allows one triple on each line. It was designed for the RDF test suite parser reference output.

[The RDF test suite format is not committed to be an exact subset of N3, and currently (2004) specified uses a \uXXXX form for encoding unicode characters in URIs, in variance with N3, and the IRI drafts which use %XX encoding of utf-8. Currently this output option is a --n3=u flag in cwm].

Turtle is another subset, for only expressing RDF. It is like n3-rdf below except that it does not have the path syntax. It is a subejct of a searate document.

The fact that the turtle language is a subset of the N3 language has not been checked from the grammars.

Grammars for N3 subsets are linked below.

BNF grammar descriptions of N3

N3 the full language	N3	RDF/XML	HTML	yacc
N3-rdf (under development). This is an N3 language which is constrained so that only correct RDF graphs can be defined. This is all you need for data.	N3	RDF/XML	HTML	
N3-rules (under development). This subset allows {} only for making rules like {...} =>{...}, to be equivalent to various other rule languages out there.	N3	RDF/XML	HTML	
N3-QL (under development) Restrictions very similar to N3-Rules	N3			

Comparison of N3 subsets

Feature	Expresses RDF 1.0	@prefix [], ; a	Collections	Numeric literals	Literal subj	RDF Path	Rules	Formulae
syntax:			(<a> <b>)	2	7 a n:prime.	x!y^z	{?x}=> {?x}	{} @forAll
NTriples	y							
Turtle	y	y	y					
N3 RDF	y	y	y	y	y	y		

N3 Rules	y plus	y	y	y	y	y	y	
N3	y plus	y	y	y	y	y	y	y

Acknowledgements

Sean Palmer and other folks on irc://openprojects.net/rdfig and later irc://openprojects.net/swig suggested many things and reviewed new ideas (and scrapped old ones!). Yosi Scharf created the parser test suites and worked on consistency between grammar, test suites and code. Thanks to all implementers on N3 software in all countries and langauges!

Historical Notes

Numbers

Whilst complex numbers (with integer, rational or real parts) are a reasonable class to add, it is not seen as a priority at the moment.

The literal syntax for decimals as 1.0 as opposed to double length reals as 1.0e0 or 1e0 was decided in coordination with the DAWG working group in January 2006.

Boolean truth values

They were added to Notation3 in 2006-02 in discussion with the SPARQL language developers, the Data Access Working Group. Note that no existing documents will have used a naked true or false word, without a `@keyword` statement which would make it clear that they were not to be treated as keywords. Furthermore any old parser encountering true or false naked or in a @keywords statement would throw an error, so misunderstanding is avoided

Mime Type History

[The type application/n3 was applied for in 2002 and again in 2006. The IETF-W3C liaison meeting established that the mime type would be formally awarded when the N3 specification was in a 'published' form. In the mean time, it should be used as part of the point of N3 is to be human readable, and so the text tree is indicated. The application for text/n3 within the IANA registry is pending as of 2006-02 as IANA #5004. While registration is pending, applications should use the proposed type in anticipation of registration, **not** an x- type.]

In 2008-01, this specification was submitted as a W3C Team Submission. Some discussion of the mime type occured, in hich the consesnsus appeared to be that text/n3 was preferable to text/rdf+n3, and so that change was adopted.

In 2008-01, there was some discussion also as to whether the encoding parameter really

was likely to be used in practical systems. That general IETF architecture is outside the scope of the document.

Appendix: Internet Media Type, File Extension and Macintosh File Type (Normative)

Contact:
Eric Prud'hommeaux
See also:
How to Register a Media Type for a W3C Specification
Internet Media Type registration, consistency of use
TAG Finding 3 June 2002 (Revised 4 September 2002)

The Internet Media Type / MIME Type for Notation3 is "text/n3".

It is recommended that Notation3 files have the extension ".n3" (all lowercase) on all platforms.

It is recommended that Notation3 files stored on Macintosh HFS file systems be given a file type of "TEXT".

This information that follows has been submitted to the IESG for review, approval, and registration with IANA.

Type name:
text
Subtype name:
n3
Required parameters:
None
Optional parameters:
`charset` — this parameter is required when transferring non-ASCII data. If present, the value of `charset` is always `UTF-8`.
Encoding considerations:
The syntax of Notation3 is expressed over code points in Unicode [UNICODE]. The encoding is always UTF-8 [RFC3629].
Unicode code points may also be expressed using an \uXXXX (U+0 to U+FFFF) or \UXXXXXXXX syntax (for U+10000 onwards) where X is a hexadecimal digit [0-9A-F]
Security considerations:
Notation3 is a general-purpose assertion and logic language; built-in functions evaluate given data to infer more assertions. Certain built-in functions may dereference URIs, invoking the security considerations of the scheme for that URI. Note in particular, the privacy issues in [RFC3023] section 10 for HTTP URIs. Data obtained from an inaccurate or malicious data source may lead to inaccurate or misleading conclusions, as well as the dereferencing of unintended URIs. Built-in functions allow one to parameterize the incorporation of data: one example of more

careful behavior would be to incorporate a source document only ifs hash matches a stated value and one may incorporate only those assertions matching a stated pattern. Care must be taken to align the trust in consulted resources with the sensitivity of the intended use of the data; inferences of potential medical treatments would likely require different trust than inferences for trip planning.
Notation3 is used to express arbitrary application data; security considerations will vary by domain of use. Security tools and protocols applicable to text (e.g. PGP encryption, MD5 sum validation, password-protected compression) may also be used on Notation3 documents. Security/privacy protocols must be imposed which reflect the sensitivity of the embedded information.
Notation3 can express data which is presented to the user, for example, RDF Schema labels. Application rendering strings retrieved from untrusted Notation3 documents must ensure that malignant strings may not be used to mislead the reader. The security considerations in the media type registration for XML ([RFC3023] section 10) provide additional guidance around the expression of arbitrary data and markup.
Notation3 uses IRIs as term identifiers. Applications interpreting data expressed in Notation3 should address the security issues of Internationalized Resource Identifiers (IRIs) [RFC3987] Section 8, as well as Uniform Resource Identifier (URI): Generic Syntax [RFC3986] Section 7.
Multiple IRIs may have the same appearance. Characters in different scripts may look similar (a Cyrillic "o" may appear similar to a Latin "o"). A character followed by combining characters may have the same visual representation as another character (LATIN SMALL LETTER E followed by COMBINING ACUTE ACCENT has the same visual representation as LATIN SMALL LETTER E WITH ACUTE). Any person or application that is writing or interpreting data in Notation3 must take care to use the IRI that matches the intended semantics, and avoid IRIs that make look similar. Further information about matching of similar characters can be found in Unicode Security Considerations [UNISEC] and Internationalized Resource Identifiers (IRIs) [RFC3987] Section 8.

Interoperability considerations:

There are no known interoperability issues.

Published specification:

This specification.

Applications which use this media type:

No widely deployed applications are known to use this media type. It may be used by some web services and clients consuming their data.

Additional information:

Magic number(s):

Notation3 documents may have the strings '@prefix' or '@base' (case dependent) near the beginning of the document.

File extension(s):

".n3"

Base URI:

The Notation3 '@base <IRIref>' term can change the current base URI for relative IRIrefs in the query language that are used sequentially later in the document.

Macintosh file type code(s):

"TEXT"

Person & email address to contact for further information:

Eric Prud'hommeaux <eric@w3.org>

Intended usage:

COMMON

Restrictions on usage:

None

Author/Change controller:

The Notation3 specification is the product of Tim Berners-Lee. A W3C Working Group may assume maintenance of this document; W3C reserves change control over this specifications.

Appendix: Change History

2007/10

@base introduced, and @prefix clarified.

2006/03

N3Resources document split from this one.

2006/03

The \ooo and \xhh escapes from dprocated to removed.

2006/01

Added decimal literals, and true and false, in coordination with SPARQL.

2004/03

The empty prefix is now bound by default to <#>

2003/03

Added literal numbers

2003/02

Added @forall @forsome

2001/04/10

Removed \a, \b, \f, \v because they are not allowed in XML. Changed "ASCII character" to "byte" for \ooo and \xhh. (duerst)

2000/12/29

Switched from bind to @prefix. Code still support sbind.
Added (node node ...) list shorthand, as code now reads and writes it.

Added a little about self-describing documents.

References

Normative

[RDFAS] *Resource Description Framework (RDF): Concepts and Abstract Syntax* W3C Recommendation 10 February 2004

[UTF8] F. Yergeau, *UTF-8, a transformation format of ISO 10646*, IETF, RFC2279

[UNF] Mark Davis, Martin Dürst, *Unicode Normalization Forms* , Unicode Consortium: Unicode Standard Annex #15

Informative

[NT]Jan Grant, David Beckett *RDF Test Cases* W3C Recommendation, Section 3: "NTriples"

[BECK] D. Beckett, New Syntaxes for RDF summarizes the state of the art as of 2004.

[TURT] David Becket, *Turtle - Terse RDF Triple Language*

[N3QL] N3QL, a draft proposal for an RDF query language.

[TUT]A tutorial

[OPSEM]*Notation 3 Logic* An operational semantics for N3.