Nevertheless, the opening summary from my proposal expresses the "find and fetch" use case which the IETF community lacks for its more than 8,000 specifications (referred to as "Request for Comments" (RFC). As I am the author of this proposal, I'll reproduce a lengthy excerpt here:

> "Many specifications relevant to technical projects are maintained and promulgated by global and national standards bodies. But it is difficult for technical designers and developers to efficiently recall and to keep abreast of updates to the large number of standards that are in effect, and directly applicable to their tasks or components at any given time.
>
> We propose a collaborative project of iterative experimentation using selected standards documents of the Internet Engineering Task Force (IETF) as sample requirements (i.e. BCP, Best Current Practice documents). Factors that invoke particular requirements of a small representative sample of the standards will be expressed as computable 'control tables'. Sample unit tests will be designed and run to simulate the normal technical work of creating, maintaining or diagnosing Internet subsystems or services. The free/libre/open "Internet of Rules" specifications and components (currently at alpha testing stage through Xalgorithms Foundation) will provide the automated rules discovery method.
>
> The purpose of the experiment will be to determine if sufficient contextual and operational data can be generated from unit tests to automatically find and return the correct technical requirements to the working developer, as needed. If this method of automated rules discovery can be made to work with some initial very simple tests, additional tests will be designed with incrementally increasing complexity and uncertainty, until a practical limit is determined.
>
> The objective of this experiment would be to determine if Xalgorithms "Internet of Rules" specifications and component may provide a practicable and effective method for designers and developers to receive a discrete, helpful interface to the specific requirement statements from the applicable standards 'in the moment' as they are testing some element of their implementation. This is the scenario within which they are most likely to be paying attention to conformance considerations, and will have the opportunity to act on such information to enhance conformance." (Potvin, 2019b)

5.4 Rules as Data

5.4.1 Data Structure of [rule.dwd] Records

Earlier in this dissertation it was put forward that:

- A rule is an *imperative* statement of obligation, permission or encouragement among people;
- Documentation about a rule is a *declarative* statement as a 'normative proposition';
- Applicability and invocation of a rule to a circumstance is an *empirical* statement of deduction.

In the DWDS, a rule-maker agent communicates imperative assertions with normative propositions to assist rule-taker agents with empirical deductions. Obligation, permission or encouragement among human and machine 'agents' may be communicated with optimal efficiency in a complex dynamic multi-objective multi-constraint setting.[62] The end-to-end information transmission must be intuitive enough for a broad

62 This is unlike 'maximum' efficiency in terms of a single criterion.

population of human rule-maker agents and rule-taker agents to communicate normative propositions among themselves without having to know formal data processing or computer programming methods, and yet it must also structure the transmitted information precisely enough to be readily parsed and processed on any computing platform the rule-taker agents may prefer to employ or delegate to.

The default deployment of any node in a network which implements the DWDS includes all three loosely coupled components, RuleMaker, RuleReserve and RuleTaker. RuleReserve provides a passive data storage service to RuleMaker, and a passive data sifting service to RuleTaker. Users of RuleMaker applications and of RuleTaker components can have their own Subset RuleReserve nodes, or they may decide to have external third-party suppliers of RuleReserve nodes bundled with services for quality assurance, security, up-time guarantees, and error and omissions insurance. These function together as a type of data 'pipeline' (von Landesberger et al., 2017). RuleReserve receives an [is.dwd] request message from RT instances, and employs its descriptive data about a particular circumstance as a virtual sieve to sift 'in effect' and 'applicable' [rule.dwd] records from its entire collection, as well as any [lookup.dwd] tables that those rules require to operate. First RuleReserve sifts for rules 'in effect' to get an intermediate list, then it sifts again for rules 'applicable'. What remains is packaged into an [ought1.dwd] message and provided back to the requesting RuleTaker instance. At that point, RuleTaker will then sift the logic gates of each [rule.dwd] record in the [ought1.dwd] rows to determine what output assertions are actually 'invoked', and from this generate an [ought2.dwd] message that is delivered to the end-user, or their application or machine. (End users have the option to have RT run an additional round of 'in effect' and/or 'applicable' sifting operations with a revised or refined [sieve2.dwd] prior to resolving the logic gates.)

The DWDS enables three parallel representations of the same 'rules-as-data':

- General users get a graphical interface that prioritizes their comprehension of the information;
- Technical users and machines get a JSON record prioritizing data integrity and transmissibility;
- Machines get an indexed record that prioritizes storage efficiency and processing speed.

Every human-accessible [rule.dwd] record, upon commit from a RuleMaker application to a RuleReserve node, is immediately pre-parsed into directly-processable data, so that it does not have to be parsed again at compute time. A parser in RuleMaker uses a pre-defined grammatical framework to transcribe it into a hierarchical data structure, and in RuleReserve this data is splayed out along a single row of the wide-column distributed database. In this form of storage, each row of data can be processed in aggregate with any number of other rows containing [rule.dwd] or [lookup.dwd] records. RuleReserve nodes have two functions: immutable storage and fast columnar data sifting.[63]

63 Early experimental prototyping employed MongoDB for storing JSON files, and Cassandra for fast columnar queries, which could be swapped for ScyllaDB for faster performance.

In order to accommodate input variability that can be expected from diverse rule authors on a decentralized network, it would be optimal to employ a ‘recursive descent’ style of parser configured with several parsing algorithms. In the event one method fails, it returns to the beginning of the record and attempts an alternative available parsing method. When a record cannot be parsed, a notification with diagnostic evidence is provided to the current user and to the event log. Every RR node interacts with RuleTaker (RT) clients through network data streaming that logs requests and responses.[64] The event log can be analyzed for patterns, which could indicate potential improvements.

The sifting operations of DWDS depend upon the data structure of [rule.dwd] records. We’ll use the JSON format to illustrate this, but the reader should keep in mind that a CBOR representation, and the human-optimized graphical form, and the machine-optimized horizontal tape form are concurrent and informationally equivalent. Tables 25 and 26 show the sample rule (Grocery Store Delivery Policy) in eight sections, which are coloured for clarity. The first five sections provide classes of metadata about each rule, which are used for sifting operations of the RR network. The rule logic of sections 6 through 8 are used for logic gate sifting within RT components.

Table 25: Sections of a [rule.dwd] Record in JSON

Metadata Used by RuleReserve *Data used to sift for rules ‘In Effect’ and ‘Applicable’*	**Logic Data Used by RuleTaker** *Data used to sift for rules ‘Invoked’*
1. Rule Identity	
2. RuleMaker Identities	6. WHEN these Input Conditions
3. Linked Rules or Lookups	7. THEN these Output Assertions
4. GIVEN this Context: Where and when this rule is 'in effect'.	8. Output Weights and Characteristics
5. WHEN these Categories: Activities and things to which this rule is 'applicable'.	

64 This can be implemented with Kafka, Pulsar, or equivalent.

Table 26: JSON Representation of a Sample [rule.dwd] Record


```
{
  "id": "24f44897-b6ad-4ca0-8f7d-03c059b08e86",
  "uuid": "24f44897-b6ad-4ca0-8f7d-03c059b08e86",
  "rule_id": "24f44897-b6ad-4ca0-8f7d-03c059b08e86",
  "rulereserve_nodes": "*",
  "version_standard_url": "https://semver.org/",
  "dwds_schema_version": "0.0.0",
  "properties": {
    "id": "24f44897-b6ad-4ca0-8f7d-03c059b08e86"
  },
  "metadata": {
    "rule": {
      "120_title": "Grocery Store Delivery Policy",
      "240_summary": "",
      "960_explanation": "When our standard delivery box is
more than half full and also contains at least $100.00 in
value of groceries, we provide free delivery. This does not
apply to non-standard boxes. For all non-standard boxes,
when delivery is provided we do",
      "version": "0.4.0",
      "criticality": "",
      "url": "https://www.grocersonline.com/deliverypolicy",
      "rulemaker_entity": [
        {
          "name": "Xalgorithms Foundation",
          "url": "https://www.xalgorithms.org",
          "uuid": "2171eb5f-a819-4ff9-bcda-e3edb4dc7e4d"
        }
      ],
      "rulemaker_manager": [
        {
          "name": "Joseph Potvin",
          "email": "jpotvin@xalgorithms.org",
          "contact": "",
          "uuid": "0d304b5d-9c3c-4606-8abe-45fe1835dfbe"
        }
      ],
      "rulemaker_author": [
        {
          "name": "Joseph Potvin",
          "email": "jpotvin@xalgorithms.org",
          "contact": "",
          "uuid": "78042f34-ebe1-4aa4-851c-a0563ee1c423"
        }
      ],
      "rulemaker_maintainer": [
        {
          "name": "Joseph Potvin",
          "email": "jpotvin@xalgorithms.org",
          "contact": "",
          "uuid": "69c3e2e9-3dc1-453b-a568-3172b80c9d18"
        }
      ],
 "linked_rules_or_lookups": [
 {
 "dwds": "",
 "column": [],
 "row": [],
 "value": []
 }
 ],
 "in_effect": [
    {
      "country": "CA",
      "subcountry": "CA-ON",
      "timezone": "UTC-05:00",
      "start": "2021-12-31T05:00:01.000Z",
      "end": "2023-12-31T04:59:59.000Z"
    }
  ],
  "category_applicable": {"industry_classifications": [
      {
        "isic_code": "4721",
        "isic_name": "Retail sale of food in specialized
stores"
      }
    ],
    "good_service_asset": [
      {
        "unspsc_code": "78142100",
        "unspsc_name": "Logistics operation management"
      }
    ]
  },
"input_conditions": [
    {
      "sentence":[
                {"determiner":: "The"},
                {"noun": "capacity" },
                {"description": "of this box" },
                {"past_participle_verb": "used" },
                {"predicate_verb": "is" },
                {"attribute": ">=half"}
            ],
            "scenarios": ["00", "01", "01"]
        },
        {
            "sentence":[
                {"determiner":: "This" },
                {"past_participle_verb": "measured" },
                {"noun": "box" },
                {"attribute": "type" },
                {"predicate_verb": "is" },
                {"description": "standard"}
            ],
            "scenarios": ["11", "11", "01"]
        },
        {
            "sentence":[
                {"determiner":: "The" },
                {"noun": "value" },
                {"past_participle_verb": "contained"},
                {"description": "in this box"},
                {"predicate_verb": "is"},
                {"attribute": ">=$100"
            }
            ],
            "scenarios": ["11","00","01"]
    }
  ],
"output_assertions": [
    {
      "sentence":[
                {"determiner":: "The"},
                {"past_participle_verb": "advertized"},
                {"noun": "delivery"},
                {"description": "of groceries"},
                {"predicate_verb": "is"},
                {"attribute": "offered"
            }
            ],
            "scenarios": ["00","01","01"]
    },
        {
      "sentence":[
                {"determiner":: "The"},
                {"past_participle_verb": "advertized"},
                {"noun": "price"},
                {"description": "of the delivery service"},
                {"predicate_verb": "is"},
                {"attribute": "charged"
            }
            ],
            "scenarios": ["00","01","00"]
    }
  ],
  "output_weight": {
    "character": "0",
    "enforcement": "8",
    "consequences": "17",
    "rule_group": ""
  },
  "output_characteristics": {
    "ultimate_responsibility": "rule-taker",
    "primary_normative_verb": "may"
    "normative_orientation": "affirmative",
    "primary_action_verb": "to_do",
    "rule_rationale": "practical",
    "rule mood": "declarative"
  }
}
```

Any [lookup.dwd] table can be similarly represented in JSON, CBOR, human-optimized graphical form, and machine-optimized horizontal tape form without the syntax. Table 27 shows a small [lookup.dwd] table with two ISO 3166-1 country codes, and a column for the current value of the xyz_index.

Table 27: A Simple Lookup Table

xyz_index	
3166-1 CA	24.07
3166-1 CL	23.65

This can be written in JSON as follows:

```
[
 { "3166-1" : "CA", "xyz_index" : "24.07" },
 { "3166-1" : "CL", "xyz_index" : "23.65" }
]
```

When autonomous parties on a decentralized network are publishing their own [lookup.dwd] records for general use, there is a natural incentive to use standard data schemas, in effect, standard application programming interfaces (API), so that their tables will operate for the intended users.

5.4.2 Transmission Protocols for Data with Direction

The default network connection configuration of the RuleMaker, RuleTaker and RuleReserve components is “hypertext transfer protocol - secure” (https:) over transmission control protocol (TCP) port 443 for encrypted network transmission of [is.dwd] and [ought1.dwd] transitory messages, and the “InterPlanetary File System” (ipfs:) over port 4001 for network storage and retrieval of whole [rulereserve.dwd] and [lookup.dwd] persistent records to populate SupersetRuleReserve nodes. In this scenario, all messages and transmitted records mingle with general Internet traffic. An Internet of Rules can be operationalized with existing firewall and Internet traffic management settings, and network administrators have no unconventional configurations to deal with.

DWDS uses IPFS as a general-purpose resilient content delivery network (CDN), that’s to say, a geographically distributed network of servers choreographed to provide simple efficient storage and fast delivery of whole files of tabular data over the Internet for data processing on SQLite by RuleReserve nodes and by RuleTaker components.[65]

A chosen design premise of the DWDS is that data which embodies intrinsic normative direction (obligation, permission and encouragement) is a distinct class of data. One may reasonably consider whether the communication of rules might usefully shift to a network path that is dedicated to this class of data, in order to enable more effective and efficient end-user monitoring. This could be appropriate when [is.dwd] and [ought1.dwd] messages and [rule.dwd] and [lookup.dwd] resources carry data that stakeholders deem to carry significant weight for monetary, safety, security, ecological and liberty standards.

The DWDS does not require, but describes for consideration the potential for a new *‘Data With Direction*

65 The initial suggestion for our design to use IPFS came from Calvin Hutcheon, and the choice to employ it as our persistent storage method was made jointly with Don Kelly.