The New Digital Economy – *How to Transform The Telco Networks*

JSON Schema for Attribute-based Access Control (ABAC) for Network Resource Security

Principal Investigator:

Greg Linklater

MSc @ Rhodes University

SATNAC 2017

Rhodes University
Where leaders learn

Coriant

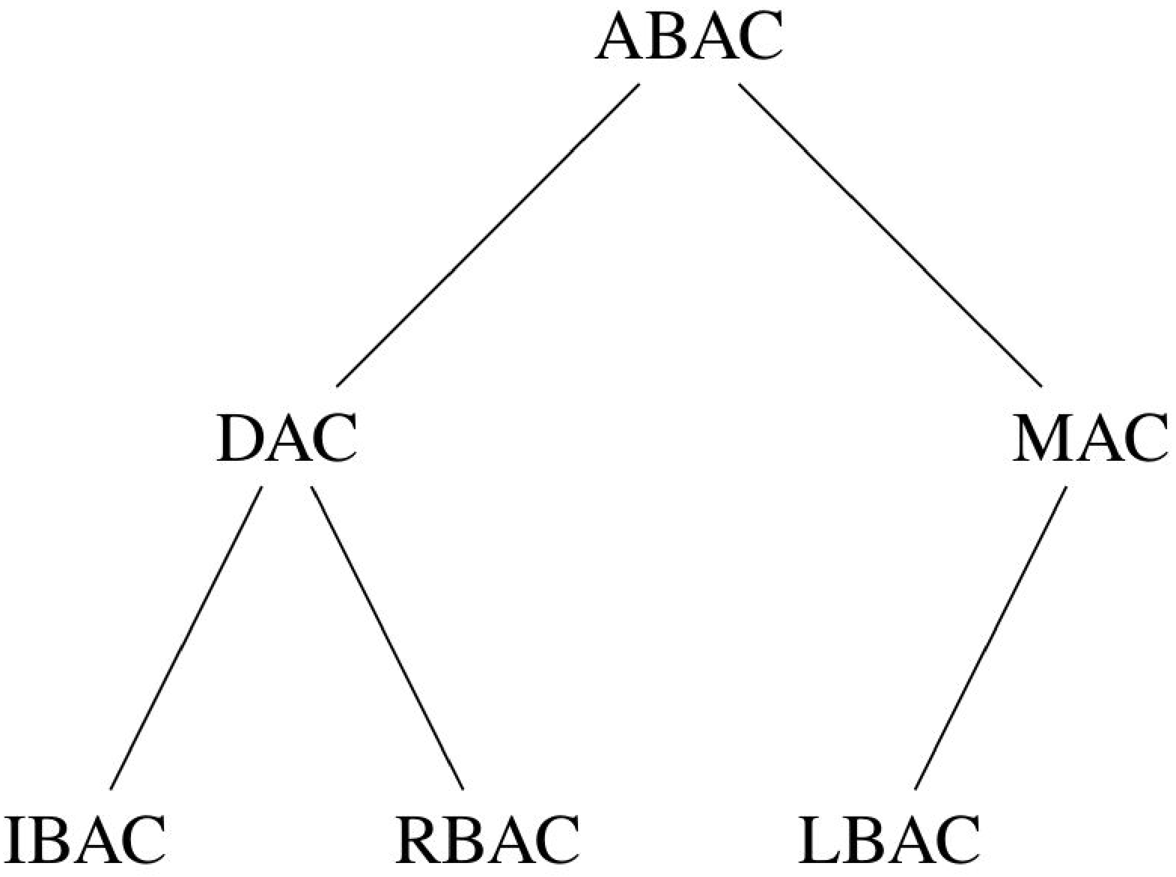
easttel
Bright Ideas Projects 39

Hosted by Telkom

Proceedings

1. Overview of Attribute-based Access Control (ABAC)
   - What is ABAC?
   - Benefits of ABAC
2. OASIS’ XACML
   - Overview and Caveats
3. JSON and JSON Schema
   - Why JSON? (JSON vs. XML)
   - Requirements and Functional Completeness
4. Current Affairs

Access Control



ABAC

DAC

MAC

IBAC

RBAC

LBAC

Figure 1. Non-Exhaustive Access Control Model Hierarchy

What is ABAC?

Access decisions are made based on:

- Assigned attributes of the subject
  - isEmployee, dept == "HR", etc.
- Assigned attributes of the resource
  - location, protectedAssetValue, etc.
- Environmental conditions
  - date, time, planetaryAlignment, etc.
- Some arbitrarily complex boolean logic policy

-- Paraphrase of NIST Definition (2014)

ABAC Policy

“The subject may [action(s)] on [resource(s)]:

on a Friday

between the hours of 16h30 and 17h00

if they are:

an employee in the HR department

and the resource is:

worth less than subject’s annual salary

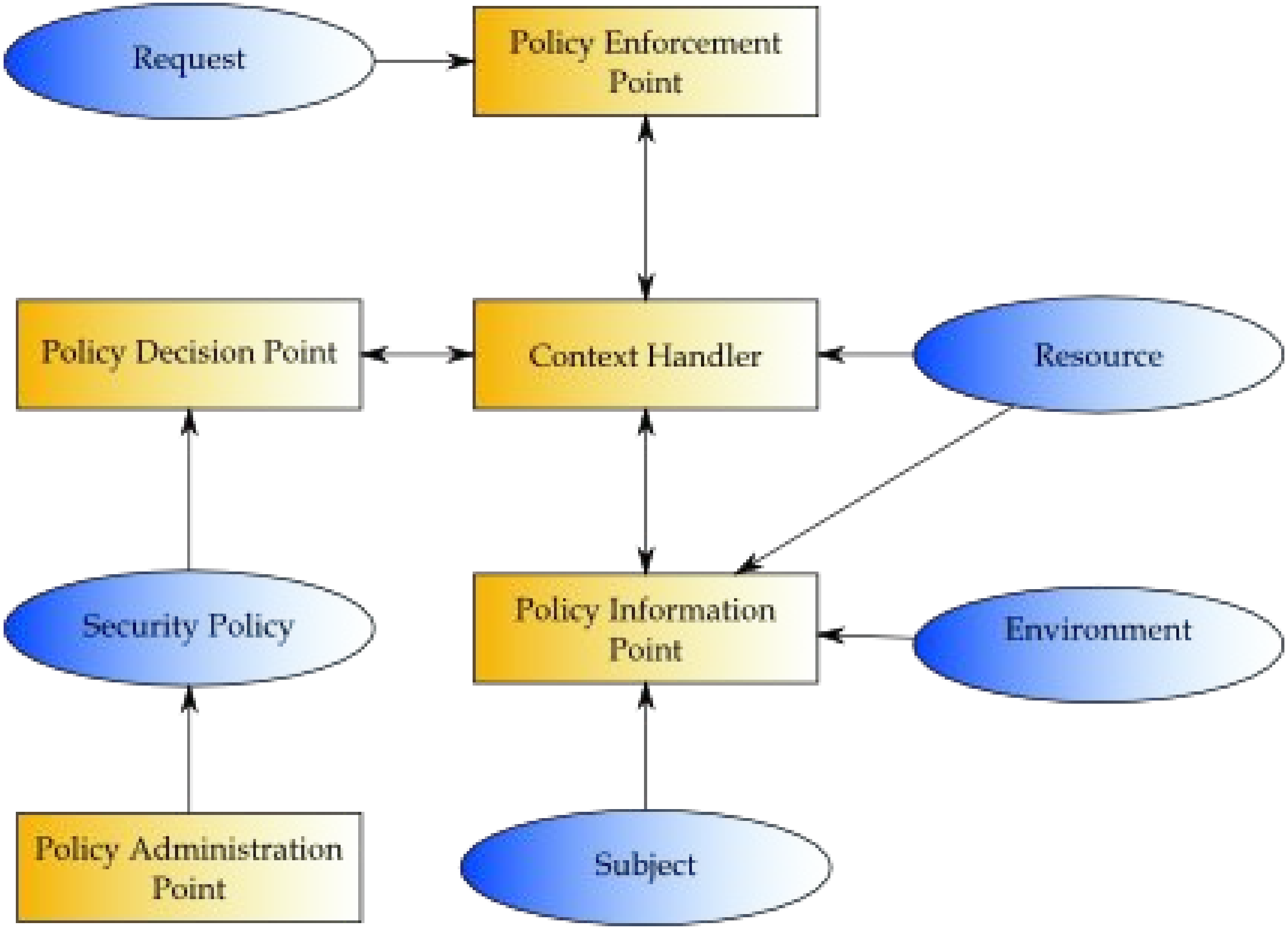
and is located at:

the focal point of a solar eclipse.”

Benefits of ABAC

- Minimal Administrative Overhead
  - Set policy once, change data as necessary.
  - (Theoretically) Scalable
- Granular, *Real* (ℝ) Policies
  - Anything you can describe
  - Encompasses ***ALL*** existing access control methods
  - Improved control

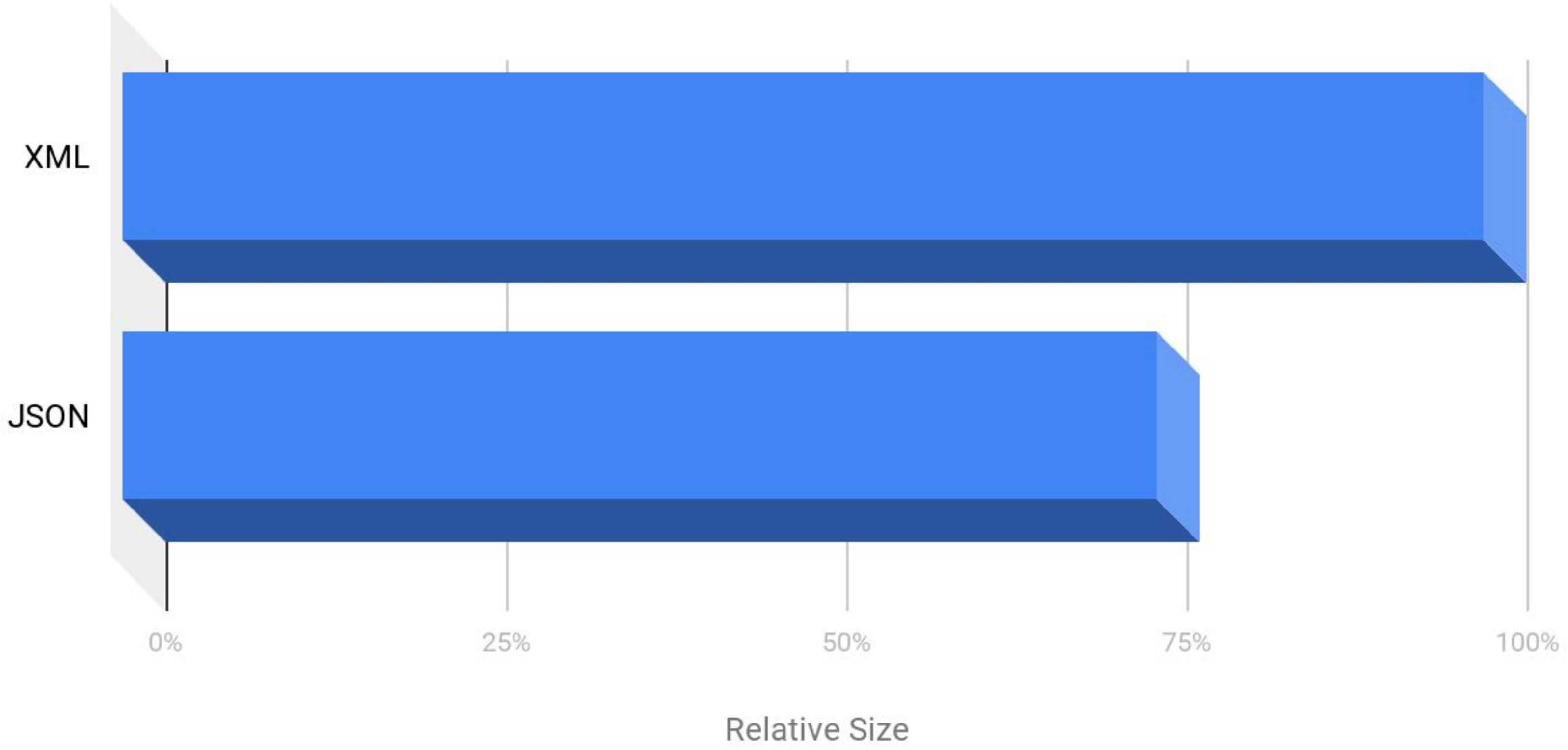
eXtensible Access Control Markup Language (XACML): Overview



eXtensible Access Control Markup Language (XACML): Caveats

- Data
- Complexity Tradeoff
- Monolithic Architecture
- Few implementations
- Highly specialised field
- Minimal adoption over the last >10 years
- XML
- **Expensive**

Why JSON? (and not XML?)



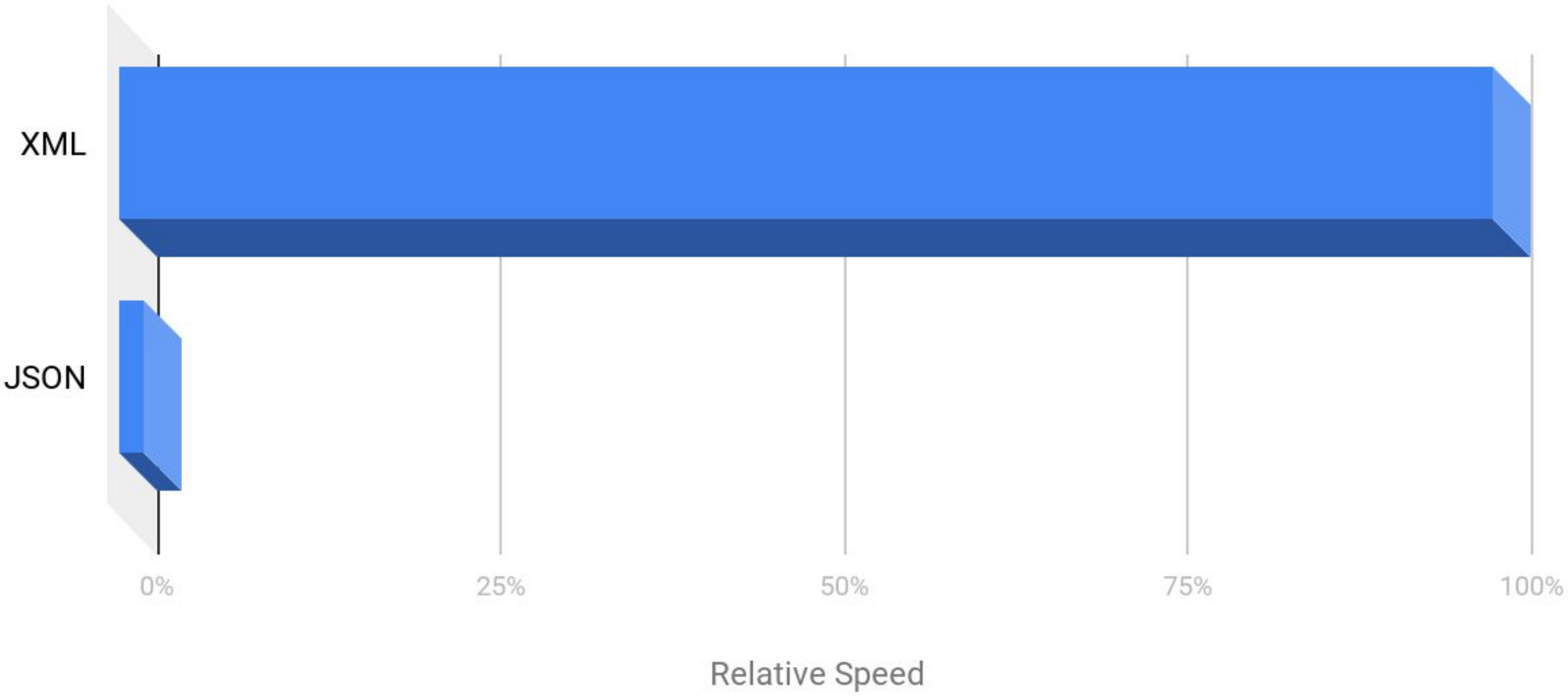
Average Relative Size of Equivalent Documents (Hameseder, 2011)

Smaller is better. For display purposes only.

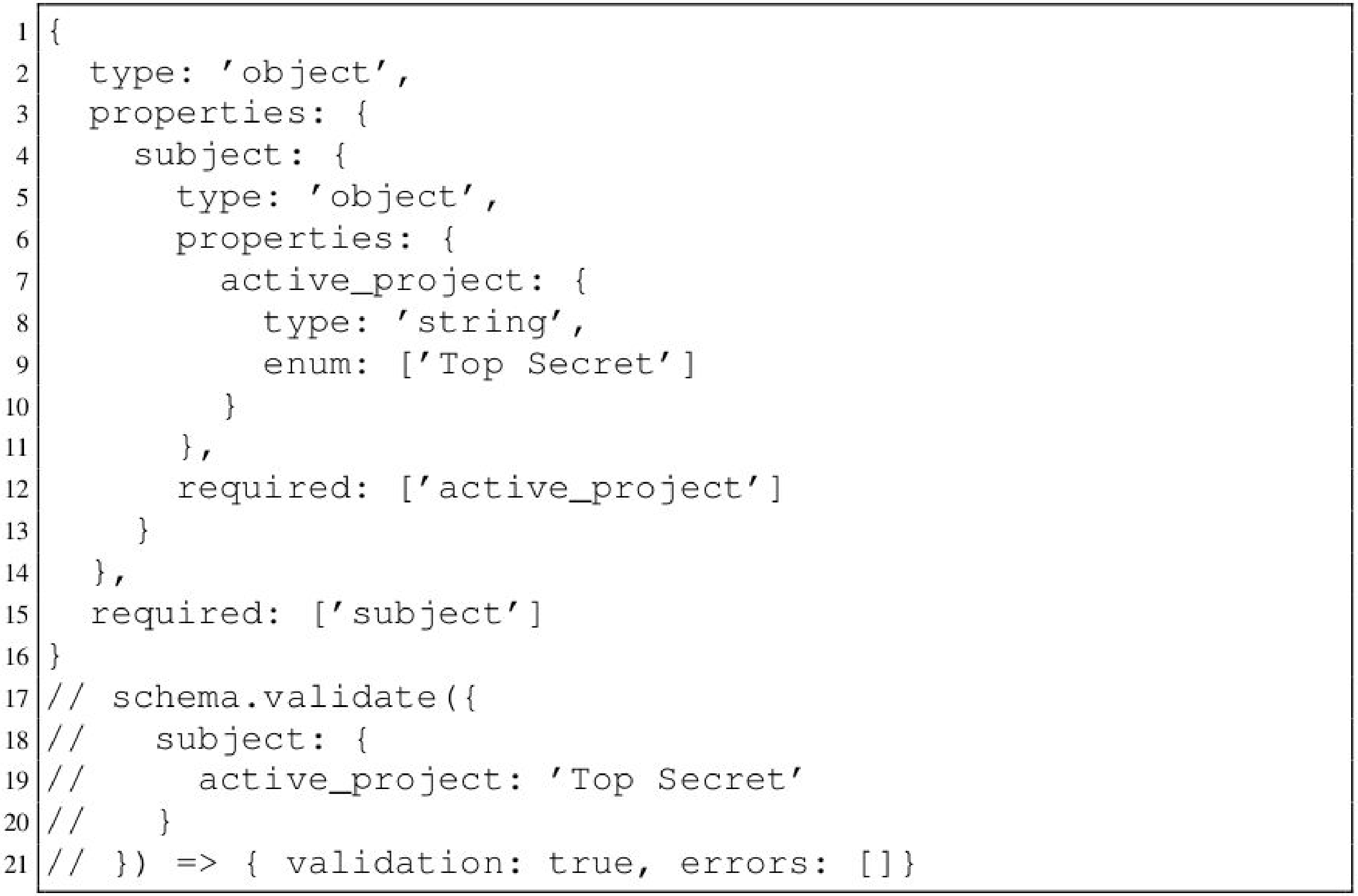
Why JSON? (and not XML?)

Average Relative Decode Speed of Equivalent Documents (Nursitov, 2011)

Smaller is better. For display purposes only.



JSON Schema



```
{
  type: ’object’,
  properties: {
    subject: {
      type: ’object’,
      properties: {
        active_project: {
          type: ’string’,
          enum: [’Top Secret’]
        }
      },
      required: [’active_project’]
    }
  },
  required: [’subject’]
}
// schema.validate({
//   subject: {
//     active_project: ’Top Secret’
//   }
// }) => { validation: true, errors: []}
```

Listing 1. Example JSON Schema Rule

Schemas & ABAC Rules

Schema Definition Language	ABAC Rule Syntax
Model	Rule
Data Interaction	Attribute Interaction
Validation	Evaluation
Serialization	Serialization

Two-Parameter Binary Logic Functions

“The set is functionally complete if and only if every boolean operation can be expressed by combining functions in the set.”

-- Paraphrase of Wernick’s Definition (1942)

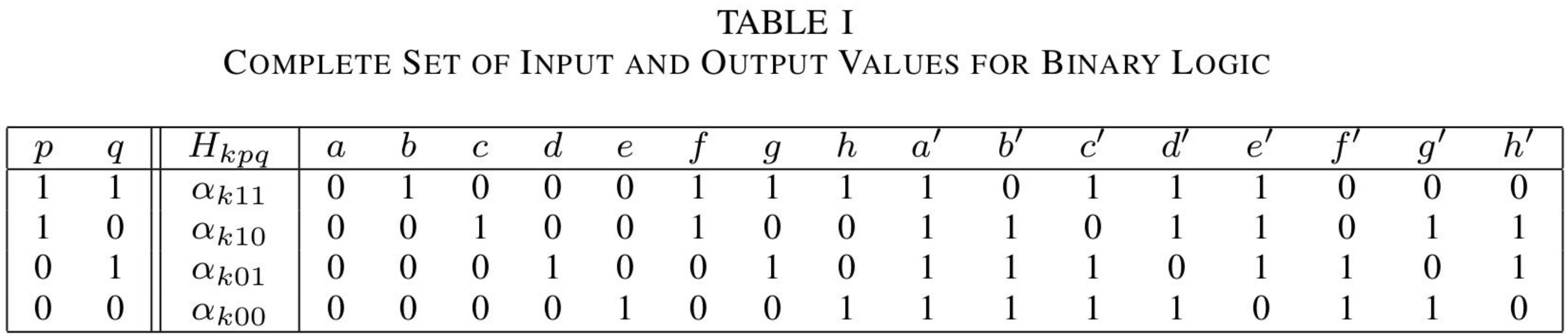


TABLE I
COMPLETE SET OF INPUT AND OUTPUT VALUES FOR BINARY LOGIC

p	q	H_{kpq}	a	b	c	d	e	f	g	h	a'	b'	c'	d'	e'	f'	g'	h'
1	1	α_{k11}	0	1	0	0	0	1	1	1	1	0	1	1	1	0	0	0
1	0	α_{k10}	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1
0	1	α_{k01}	0	0	0	1	0	0	1	0	1	1	1	0	1	1	0	1
0	0	α_{k00}	0	0	0	0	1	0	0	1	1	1	1	1	0	1	1	0

Functional Completeness

$$H = \{a, b, ..., h, a', b', ..., h'\} \tag{1}$$

where

$$a'(p,q) = \neg a(p,q), \quad b'(p,q) = \neg b(p,q), \quad \ldots \tag{2}$$

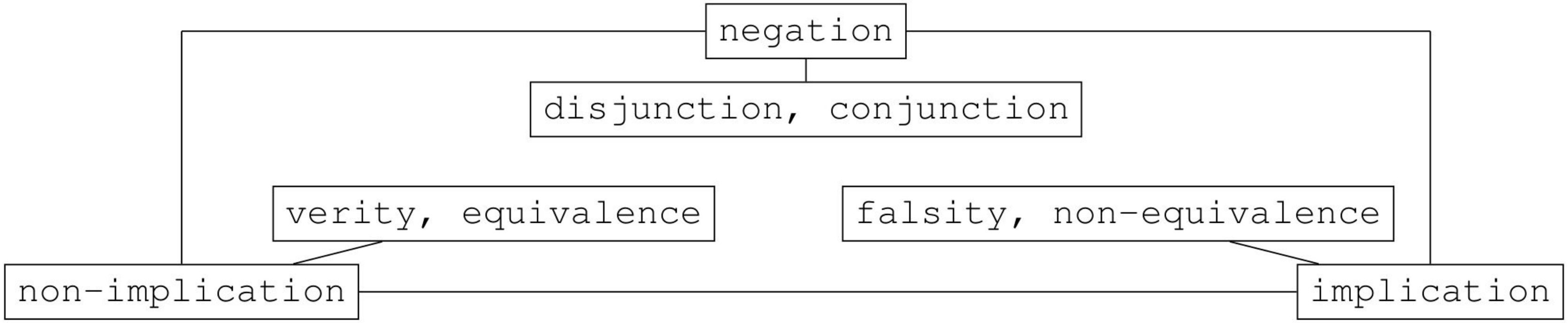
and

$$H_k(p,q) = \alpha_k \tag{3}$$

$$F(\Omega_i(p,q), \Omega_j(p,q)) \tag{4}$$

$$= H_k(p,q) = \alpha_k, \; \Omega \subset H, \; \exists i, \; \exists j, \; \exists F, \; \forall k$$

Wernick's Two-Function Sets for Functional Completeness



negation

disjunction, conjunction

verity, equivalence

falsity, non-equivalence

non-implication

implication

Figure 2. Wernick's Functionally Complete Two-Function Sets

Current Affairs

Decentralisation -- User Centric Identity

- W3C Verified Claims
- JSON/CBOR Object Signing and Encryption
- Open Algorithms (OPAL)
- Attestation Exchange
  - Trust Networks