SHACL Core Cheat Sheet

1) “Core Core” *(note: there’s no such thing as “core core,” we invented that)*

a) Node shapes
   i) sh:NodeShape

b) Property shapes
   i) sh:property
   ii) sh:path

c) Constraint components
   i) Cardinality
      1) sh:minCount
      2) sh:maxCount
   ii) Value types
      1) sh:datatype
         (a) xsd:
         (b) custom
      2) sh:class
      3) sh:nodeKind
         (a) sh:IRI
         (b) sh:BlankNode
         (c) sh:Literal
         (d) sh:BlankNodeOrLiteral
         (e) sh:BlankNodeOrIRI
         (f) sh:IRIOrLiteral
      4) *Sets:* sh:in
      5) *Specific value:* sh:hasValue
   iii) Value ranges
      1) sh:minInclusive
      2) sh:maxInclusive
      3) sh:minExclusive
      4) sh:maxExclusive
   iv) String-based
      1) sh:minLength
      2) sh:maxLength
      3) sh:length
      4) sh:pattern
         (a) *optional:* sh:flags
   v) Language-based
      1) sh:languageIn
      2) sh:uniqueLang
   vi) Logical
      1) sh:and
      2) sh:or
      3) sh:not
      4) sh:xone
   vii) Shape-based
      1) sh:node
      2) *(See “Intermediate Core” below):*
         (a) sh:property
         (b) sh:qualifiedValueShape
         (c) sh:qualifiedValueShapeDisjoint
         (d) sh:qualifiedMinCount
         (e) sh:qualifiedMaxCount
   viii) Closed shapes *(see “Intermediate Core” below)*
   ix) Property pairs *(see “Intermediate Core” below)*
  x) Non-validating *(see “Intermediate Core” below)*

d) Target declarations
   i) sh:targetNode
   ii) sh:targetClass
   iii) sh:targetSubjectsOf
   iv) sh:targetObjectsOf

e) Validation reporting
   i) sh:message
   ii) sh:severity
2) Intermediate Core  (note: there’s no such thing as “intermediate core,” we invented that)

f) Importing and referencing (Gayo 5.6.6)
   i) owl:imports
   ii) sh:deactivated

g) Combining logical operators (Gayo 5.11.5)
   i) If-then
   ii) If-then-else

h) Shape based constraints (Gayo 5.12)
   i) The constraints:
      (1) sh:node
      (2) sh:property
      (3) qualified value shapes:
         (a) sh:qualifiedValueShape
         (b) sh:qualifiedValueShapeDisjoint
         (c) sh:qualifiedMinCount
         (d) sh:qualifiedMaxCount
   ii) Shape references and recursion

i) Closed shapes
   i) sh:closed
   ii) sh:ignoredProperties

j) Property pair constraints
   i) sh:equals
   ii) sh:disjoint
   iii) sh:lessThan
   iv) sh:lessThanOrEquals

k) Non-validating constraints
   i) sh:name
   ii) sh:description
   iii) sh:order
   iv) sh:group
   v) sh:defaultValue

l) SHACL paths

<table>
<thead>
<tr>
<th>SHACL path</th>
<th>SPARQL path</th>
</tr>
</thead>
<tbody>
<tr>
<td>schema:name</td>
<td>schema:name</td>
</tr>
<tr>
<td>[sh:inversePath schema:knows]</td>
<td>^schema:knows</td>
</tr>
<tr>
<td>(schema:knows schema:name)</td>
<td>schema:knows/schema:name</td>
</tr>
<tr>
<td>[sh:alternativePath {schema:knows schema:follows}]</td>
<td>schema:knows { schema:follows</td>
</tr>
<tr>
<td>[sh:zeroOrOnePath schema:knows]</td>
<td>schema:knows?</td>
</tr>
<tr>
<td>[sh:oneOrMorePath schema:knows]</td>
<td>schema:knows+</td>
</tr>
<tr>
<td>([sh:zeroOrMorePath schema:knows] schema:name)</td>
<td>schema:knows*/schema:name</td>
</tr>
</tbody>
</table>