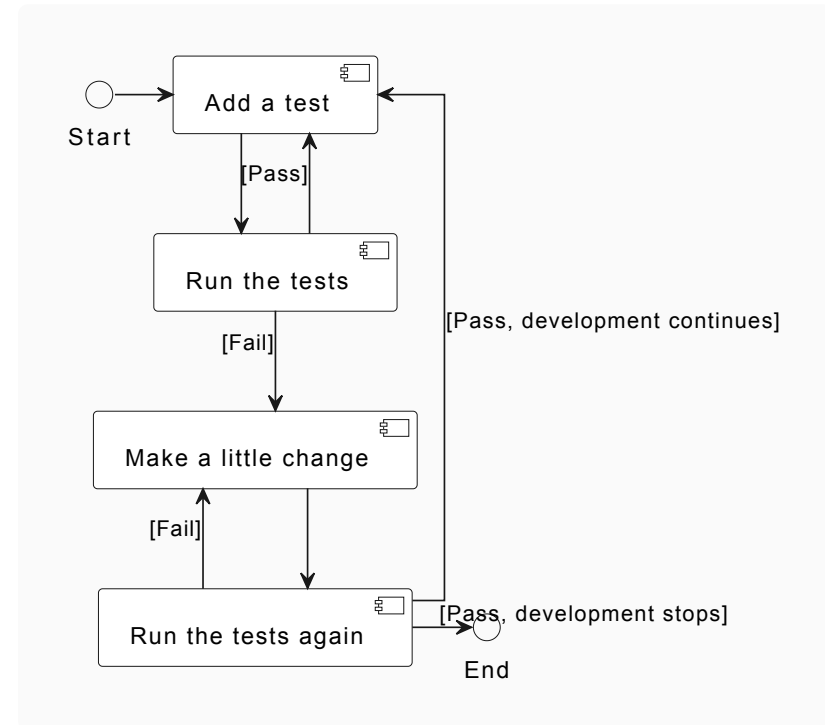# Type-Extensible Object Notation:

JSON with Syntax for Types

Test-Driven Development

"Testing has implications not only for the approach to implementation, but also the structure of the code itself and the overall practices surrounding projects. As each feature needs to be developed on a foundation of demonstrable necessity, the process must be divided into small enough steps for testing to be possible. The testing of a smaller component in a large codebase is referred to as unit tests , as each component becomes a testable unit ."

The testing of a unit either aims to demonstrate a flaw in the current system, or demonstrate the functional implementation of a feature that meets the requirements for acceptance.

A test should ideally be executed in a short amount of time, resulting in fast testing. Tests should be executable in isolation, resulting in unordered testing. Tests should use production data when applicable, and tests should represent a feature or component of an overall project and implementation.

Test cycle: Guernsey, M. (2013). Test-driven Database Development: Unlocking Agility. Addison-Wesley.
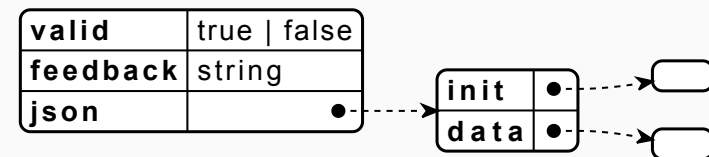
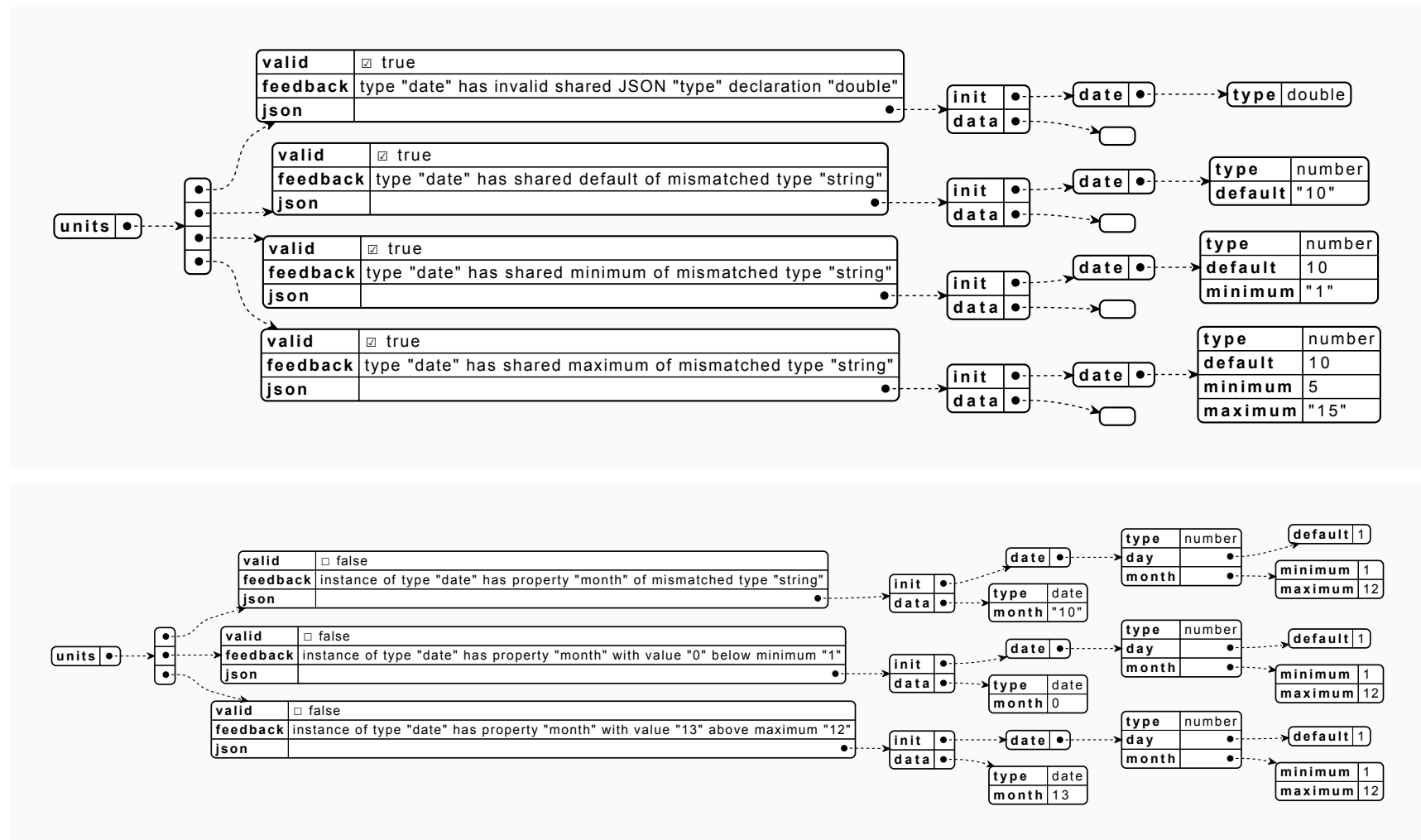2 of 10

Unit and Acceptance Testing

In this project the units are exclusively validation checks that return an error based on nonconformance to requirements or types in a data structure. A unit is a JSON data structure consisting of a sample nonconforming JSON object, its expected validity or invalidity, and an optional expected feedback string describing the source of nonconformance.

These units act as acceptance tests that describe the requirements for successfully implementing a feature through validation. This also results in implementing type semantics, as a feature can be utilised once its correct application can be verified.

These units do not act as tests demonstrating the necessity for an implementation or evolution of the JSON specification, which I identify as a flaw of the experiment documented in this project.

```
{
    "valid": "true | false",
    "feedback": "string",
    "json": {
        "init": {},
        "data": {}
    }
}
```

Top section (type declarations):

units •

- valid ☑ true
  feedback type "date" has invalid shared JSON "type" declaration "double"
  json •
  - init •
    data •
    - date •
      - type double

- valid ☑ true
  feedback type "date" has shared default of mismatched type "string"
  json •
  - init •
    data •
    - date •
      - type number
        default "10"

- valid ☑ true
  feedback type "date" has shared minimum of mismatched type "string"
  json •
  - init •
    data •
    - date •
      - type number
        default 10
        minimum "1"

- valid ☑ true
  feedback type "date" has shared maximum of mismatched type "string"
  json •
  - init •
    data •
    - date •
      - type number
        default 10
        minimum 5
        maximum "15"

Bottom section (type instances):

units •

- valid ☐ false
  feedback instance of type "date" has property "month" of mismatched type "string"
  json •
  - init •
    data •
    - date •
      - type number
        day •
        month •
        - default 1
          minimum 1
          maximum 12
    - type date
      month "10"

- valid ☐ false
  feedback instance of type "date" has property "month" with value "0" below minimum "1"
  json •
  - init •
    data •
    - date •
      - type number
        day •
        month •
        - default 1
          minimum 1
          maximum 12
    - type date
      month 0

- valid ☐ false
  feedback instance of type "date" has property "month" with value "13" above maximum "12"
  json •
  - init •
    data •
    - date •
      - type number
        day •
        month •
        - default 1
          minimum 1
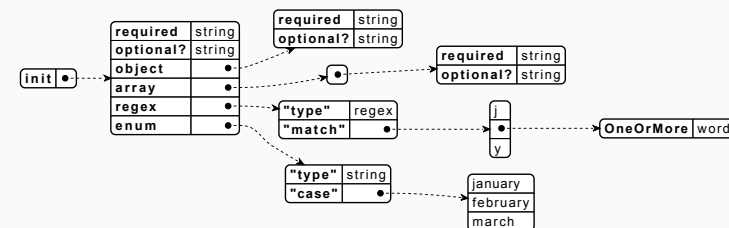   maximum 12
    - type date
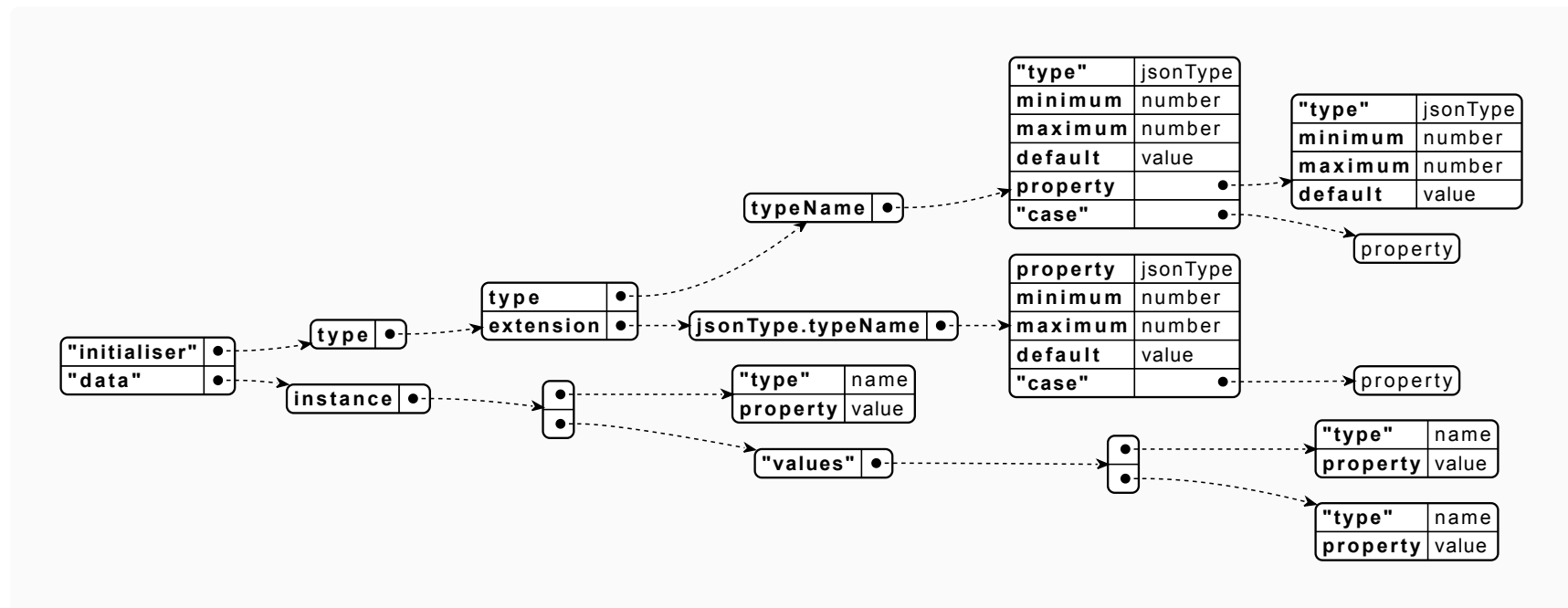   month 13

Type Semantics

The extensible implementation of types through this project meant that nodes in a data structure were only validated once they were explicitly typed. This choice ensures that untyped data structures remain valid JSON data, but has implications for designing the semantics of types.

As a semantic feature was added to the implementation, the possible combinations increased exponentially. As such the validation had to test for all possible cases and combinations of semantics, which in turn informed the design of type semantics.

In reflecting on this implementation it became evident that it is missing some features, and that there are better grammatical alternatives to the type semantics derived from testing. This includes semantics for single optional arrayrised regex and enumerated values.

```
{
    "init": {
        "required": "string", "optional?": "string",
        "object": { "required": "string", "optional?": "string" },
        "array": [ { "required": "string", "optional?": "string" } ],
        "regex": { "type": "regex", "match": [ "j", { "OneOrMore": "word" }, "y" ] },
        "enum": { "type": "string", "case": [ "january", "february", "march" ] }
    }
}
```
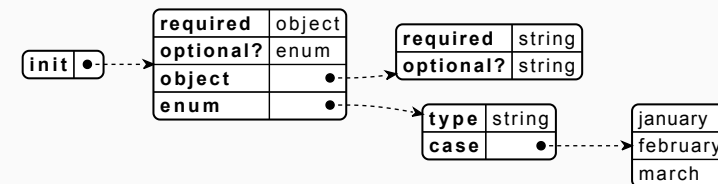
Relational Type

Phasellus aliquet convallis arcu in aliquam. Pellentesque lectus orci, vulputate a blandit eget, pretium vel lectus. Aenean ultricies augue at elit rhoncus, quis hendrerit erat egestas. Sed sit amet efficitur sem. Pellentesque euismod, odio eu imperdiet facilisis, mi tortor mattis arcu, a tincidunt sem mauris id est. Maecenas sollicitudin eros ut bibendum sagittis. Suspendisse mattis, risus et luctus lacinia, nisl neque dictum nibh, non hendrerit orci libero non nibh. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Sed volutpat, ex a sodales semper, turpis mauris accumsan erat, quis malesuada tellus mi non augue.

In ultrices dictum dui, quis lacinia mauris hendrerit ac. Donec malesuada viverra elementum. Cras erat turpis, euismod eu tempus vel, cursus non neque. Sed semper tellus bibendum mi consectetur aliquet. Suspendisse interdum auctor turpis sed ultrices. Vestibulum sollicitudin nisi at tellus dictum auctor vel sit amet tellus.

```
{
  "init": {
    "required": "object",
    "optional?": "enum",
    "object": {
      "required": "string", "optional?": "string"
    },
    "enum": {
      "type": "string", "case": [ "january", "february", "march" ]
    }
  }
}
```
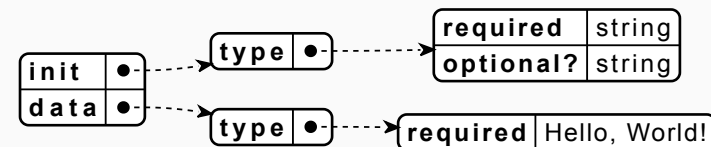
Optional Type

Phasellus aliquet convallis arcu in aliquam. Pellentesque lectus orci, vulputate a blandit eget, pretium vel lectus. Aenean ultricies augue at elit rhoncus, quis hendrerit erat egestas. Sed sit amet efficitur sem. Pellentesque euismod, odio eu imperdiet facilisis, mi tortor mattis arcu, a tincidunt sem mauris id est. Maecenas sollicitudin eros ut bibendum sagittis. Suspendisse mattis, risus et luctus lacinia, nisl neque dictum nibh, non hendrerit orci libero non nibh. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Sed volutpat, ex a sodales semper, turpis mauris accumsan erat, quis malesuada tellus mi non augue.

In ultrices dictum dui, quis lacinia mauris hendrerit ac. Donec malesuada viverra elementum. Cras erat turpis, euismod eu tempus vel, cursus non neque. Sed semper tellus bibendum mi consectetur aliquet. Suspendisse interdum auctor turpis sed ultrices. Vestibulum sollicitudin nisi at tellus dictum auctor vel sit amet tellus.

```json
{
  "init": {
    "type": {
      "required": "string", "optional?": "string"
    }
  },
  "data": {
    "type": {
      "required": "Hello, World!"
    }
  }
}
```
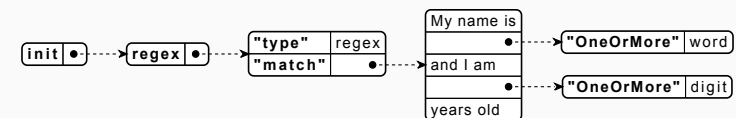
Regular Expression (RegEx)

Phasellus aliquet convallis arcu in aliquam. Pellentesque lectus orci, vulputate a blandit eget, pretium vel lectus. Aenean ultricies augue at elit rhoncus, quis hendrerit erat egestas. Sed sit amet efficitur sem. Pellentesque euismod, odio eu imperdiet facilisis, mi tortor mattis arcu, a tincidunt sem mauris id est. Maecenas sollicitudin eros ut bibendum sagittis. Suspendisse mattis, risus et luctus lacinia, nisl neque dictum nibh, non hendrerit orci libero non nibh. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Sed volutpat, ex a sodales semper, turpis mauris accumsan erat, quis malesuada tellus mi non augue.

In ultrices dictum dui, quis lacinia mauris hendrerit ac. Donec malesuada viverra elementum. Cras erat turpis, euismod eu tempus vel, cursus non neque. Sed semper tellus bibendum mi consectetur aliquet. Suspendisse interdum auctor turpis sed ultrices. Vestibulum sollicitudin nisi at tellus dictum auctor vel sit amet tellus.

```
{
  "init": {
    "regex": {
      "type": "regex",
      "match": [
        "My name is",
        { "OneOrMore": "word" },
        "and I am",
        { "OneOrMore": "digit" },
        "years old"
      ]
    }
  }
}
```

Enumerated Type

Phasellus aliquet convallis arcu in aliquam. Pellentesque lectus orci, vulputate a blandit eget, pretium vel lectus. Aenean ultricies augue at elit rhoncus, quis hendrerit erat egestas. Sed sit amet efficitur sem. Pellentesque euismod, odio eu imperdiet facilisis, mi tortor mattis arcu, a tincidunt sem mauris id est. Maecenas sollicitudin eros ut bibendum sagittis. Suspendisse mattis, risus et luctus lacinia, nisl neque dictum nibh, non hendrerit orci libero non nibh. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Sed volutpat, ex a sodales semper, turpis mauris accumsan erat, quis malesuada tellus mi non augue.

In ultrices dictum dui, quis lacinia mauris hendrerit ac. Donec malesuada viverra elementum. Cras erat turpis, euismod eu tempus vel, cursus non neque. Sed semper tellus bibendum mi consectetur aliquet. Suspendisse interdum auctor turpis sed ultrices. Vestibulum sollicitudin nisi at tellus dictum auctor vel sit amet tellus.

```
{
  "init": {
    "date": {
      "month": {
        "type": "string", "case": [ "january", "february", "march" ]
      }
    }
  },
  "data": {
    "date": {
      "month": "january"
    }
  }
}
```