ASAM OpenODD Concept Project

Format

Dr Xizhe Zhang (Jason)
Lead Engineer, V&V group, Intelligent Vehicles
WMG, University of Warwick, UK

WP Lead – Format,
ASAM OpenODD Concept project

11 November 2021
Webinar, Online

ASAM Association for Standardization of Automation and Measuring Systems

Content

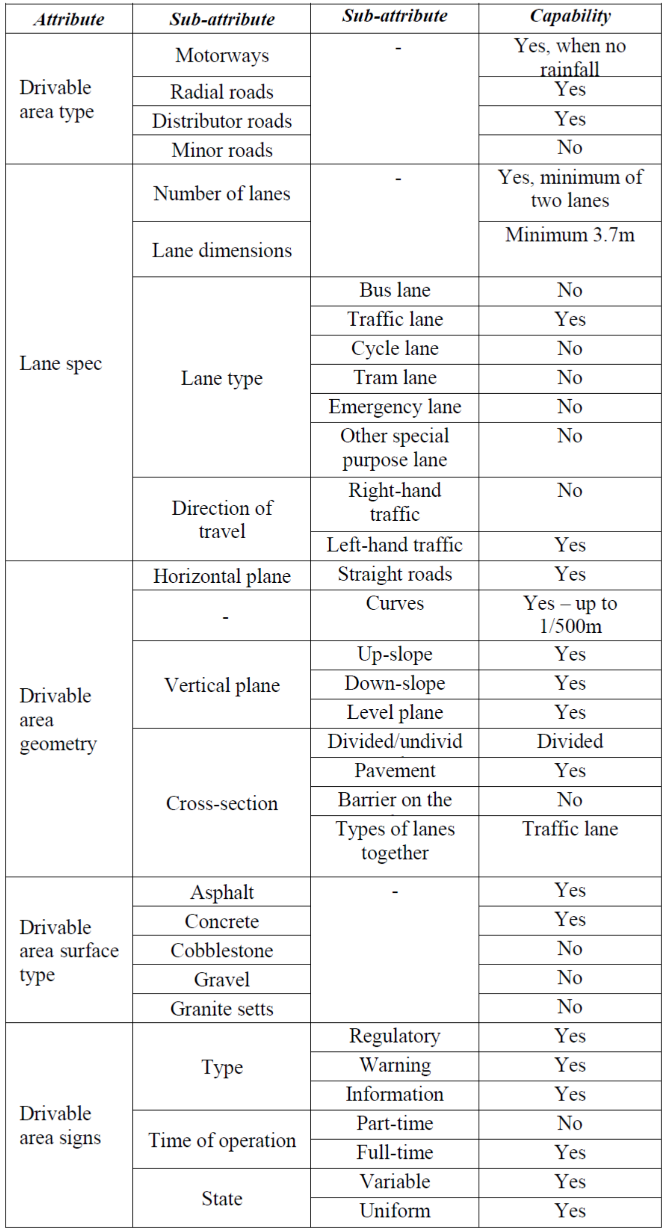
- Goal for format WP
- Why we need an ODD format
- Requirement clusters
- Two syntax approach
- Completeness of ODDs
  - Example 1: Permissive/restrictive
  - Example 2: Extensibility
  - Example 3: Composability
  - Example 4: Conditional statement

Goal for Format WP

*This WP will focus on the **semantic** and **syntactic** aspects of the ODD description format. It includes the capability to describe **conditional** ODD description using some or all ODD **attributes**. The format should be able to be used for, but not limited to, simulation execution*

Why we need an ODD format

- Using a tabular format example from BSI PAS 1883 Annex A [1]
- Lack of grammar/rules
- Need to interpretation of missing attributes
- Challenge to illustrate dependencies (conditional ODD statement)
- Limited implementation potentials
- Not necessary compact
- Mixture of data properties, units, classes
- …



Attribute	*Sub-attribute*	*Sub-attribute*	*Capability*
Drivable area type	Motorways	-	Yes, when no rainfall
	Radial roads		Yes
	Distributor roads		Yes
	Minor roads		No
Lane spec	Number of lanes	-	Yes, minimum of two lanes
	Lane dimensions		Minimum 3.7m
	Lane type	Bus lane	No
		Traffic lane	Yes
		Cycle lane	No
		Tram lane	No
		Emergency lane	No
		Other special purpose lane	No
	Direction of travel	Right-hand traffic	No
		Left-hand traffic	Yes
Drivable area geometry	Horizontal plane	Straight roads	Yes
	-	Curves	Yes – up to 1/500m
	Vertical plane	Up-slope	Yes
		Down-slope	Yes
		Level plane	Yes
	Cross-section	Divided/undivid	Divided
		Pavement	Yes
		Barrier on the	No
		Types of lanes together	Traffic lane
Drivable area surface type	Asphalt	-	Yes
	Concrete		Yes
	Cobblestone		No
	Gravel		No
	Granite setts		No
Drivable area signs	Type	Regulatory	Yes
		Warning	Yes
		Information	Yes
	Time of operation	Part-time	No
		Full-time	Yes
	State	Variable	Yes
		Uniform	Yes

[1] - "Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification," *The British Standards Institution, BSI PAS 1883*. 2020.

Requirement clusters

Development engineer

Test engineer

Data scientists

Tool developer

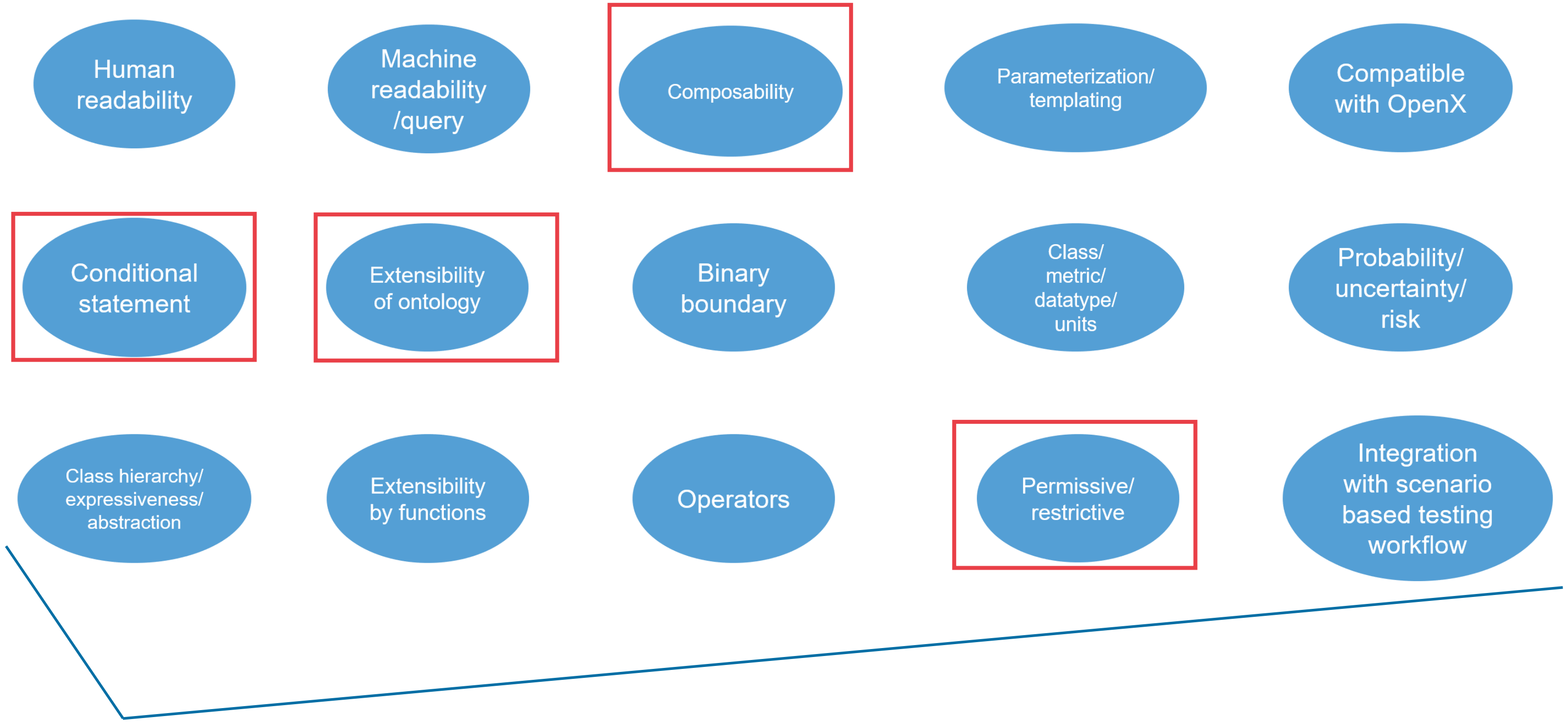
Scenario editor/creator

Data annotation engineer

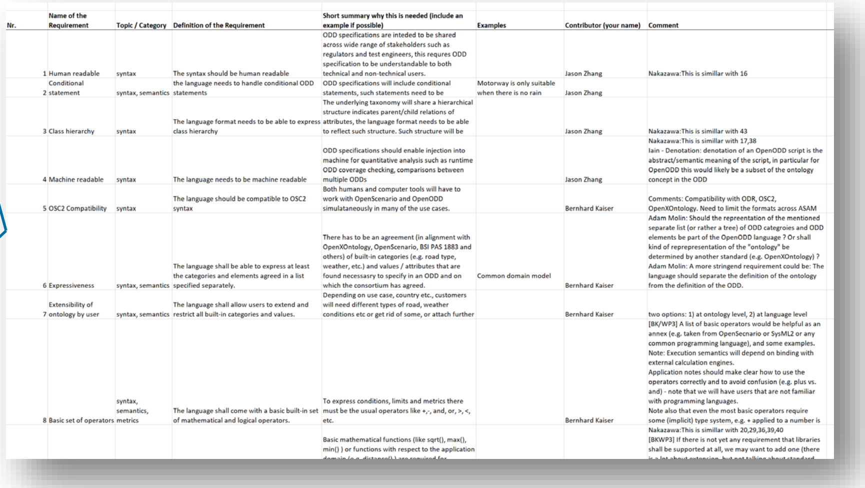
Safety engineer

Infrastructure operator

Requirement clusters


Human readability
Machine readability /query
Composability
Parameterization/ templating
Compatible with OpenX
Conditional statement
Extensibility of ontology
Binary boundary
Class/ metric/ datatype/ units
Probability/ uncertainty/ risk
Class hierarchy/ expressiveness/ abstraction
Extensibility by functions
Operators
Permissive/ restrictive
Integration with scenario based testing workflow

Requirement list



Nr.	Name of the Requirement	Topic / Category	Definition of the Requirement	Short summary why this is needed (include an example if possible)	Examples	Contributor (your name)	Comment
1	Human readable	syntax	The syntax should be human readable	ODD specifications are inteded to be shared across wide range of stakeholders such as regulators and test engineers, this requres ODD specification to be understandable to both technical and non-technical users.		Jason Zhang	Nakazawa:This is simillar with 16
2	Conditional statement	syntax, semantics	the language needs to handle conditional ODD statements	ODD specifications will include conditional statements, such statements need to be	Motorway is only suitable when there is no rain	Jason Zhang	
3	Class hierarchy	syntax	The language format needs to be able to express class hierarchy	The underlying taxonomy will share a hierarchical structure indicates parent/child relations of attributes, the language format needs to be able to reflect such structure. Such structure will be		Jason Zhang	Nakazawa:This is simillar with 43
4	Machine readable	syntax	The language needs to be machine readable	ODD specifications should enable injection into machine for quantitative analysis such as runtime ODD coverage checking, comparisons between multiple ODDs		Jason Zhang	Nakazawa:This is simillar with 17,38 Iain - Denotation: denotation of an OpenODD script is the abstract/semantic meaning of the script, in particular for OpenODD this would likely be a subset of the ontology concept in the ODD
5	OSC2 Compatibility	syntax	The language should be compatible to OSC2 syntax	Both humans and computer tools will have to work with OpenScenario and OpenODD simulataneously in many of the use cases.		Bernhard Kaiser	Comments: Compatibility with ODR, OSC2, OpenXOntology. Need to limit the formats across ASAM
6	Expressiveness	syntax, semantics	The language shall be able to express at least the categories and elements agreed in a list specified separately.	There has to be an agreement (in alignment with OpenXOntology, OpenScenario, BSI PAS 1883 and others) of built-in categories (e.g. road type, weather, etc.) and values / attributes that are found necessasry to specify in an ODD and on which the consortium has agreed.	Common domain model	Bernhard Kaiser	Adam Molin: Should the repreentation of the mentioned separate list (or rather a tree) of ODD categroies and ODD elements be part of the OpenODD language ? Or shall kind of reprepresentation of the "ontology" be determined by another standard (e.g. OpenXOntology) ? Adam Molin: A more stringend requirement could be: The language should separate the definition of the ontology from the definition of the ODD.
7	Extensibility of ontology by user	syntax, semantics	The language shall allow users to extend and restrict all built-in categories and values.	Depending on use case, country etc., customers will need different types of road, weather conditions etc or get rid of some, or attach further		Bernhard Kaiser	two options: 1) at ontology level, 2) at language level
8	Basic set of operators	syntax, semantics, metrics	The language shall come with a basic built-in set of mathematical and logical operators.	To express conditions, limits and metrics there must be the usual operators like +,-, and, or, >, <, etc.		Bernhard Kaiser	[BK/WP3] A list of basic operators would be helpful as an annex (e.g. taken from OpenSecnario or SysML2 or any common programming language), and some examples. Note: Execution semantics will depend on binding with external calculation engines. Application notes should make clear how to use the operators correctly and to avoid confusion (e.g. plus vs. and) - note that we will have users that are not familiar with programming languages. Note also that even the most basic operators require some (implicit) type system, e.g. + applied to a number is
				Basic mathematical functions (like sqrt(), max(), min()) or functions with respect to the application domain (e.g. distance()) are required for			Nakazawa:This is simillar with 20,29,36,39,40 [BKWP3] If there is not yet any requirement that libraries shall be supported at all, we may want to add one (there is a lot about extension, but not talking about standard

Two syntax proposals

Syntax 1 – Utilises OpenSCENARIO 2.0 format

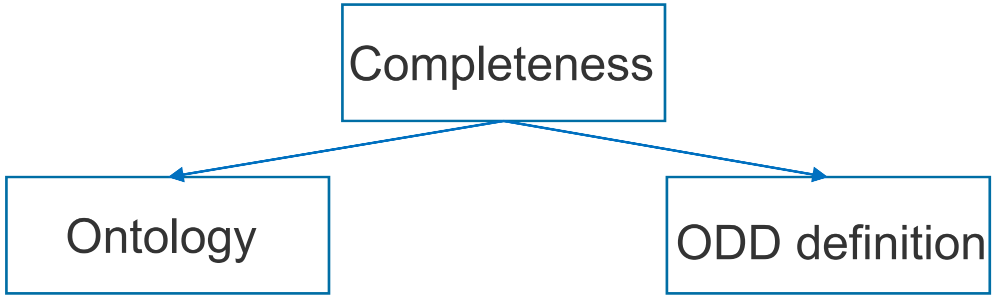
- Object oriented, declarative, domain specific programming language.
- ODD types are declared using a subtype of OpenSCENARIO struct called odd_struct.
- More details please refer to OpenSCENARIO 2.0 documentation.

Syntax 2 – Evolved during the OpenODD concept project, utilises the query semantic approach

- Mapping all attributes to True/False → inclusion/exclusion from the ODD specification.
- Functions include importing, assigning permissive/restrictive at global and local level, extension of the ontology, packaging/composing new ODDs.
- Eight keywords:

`INCLUDE`	Include external files such as ontology file
`MODE`	Assign permissive/restrictive at the global level
`SUITABLE` / `UNSUITABLE`	Assign permissive/restrictive at the individual statement level
`ADDCOND`	Add custom extension to an existing domain model or ontology
`DETERMINE`	Construct new classes/or to assign existing classes using a set of values
`MEASURE`	Assign measuring unit and data type to an ODD attributes
`TAG`	Compose multiple ODDs/parameters into a new ODD name

Completeness of ODDs



Completeness

Ontology

ODD definition

- ODD ontology
  - A set of ODD attributes together with their hierarchy, measuring units and data types
  - Always evolving, needs to be extensible, can never be exhaustive

- ODD definition
  - Utilise an ODD ontology to further define the constraints across the attributes
  - Must assign all the available ODD attributes into inclusion/exclusion → Complete coverage

- **ODD ontology cannot be exhaustive, but ODD definition must be complete**

Example 1

Permissive/ restrictive

Example 2

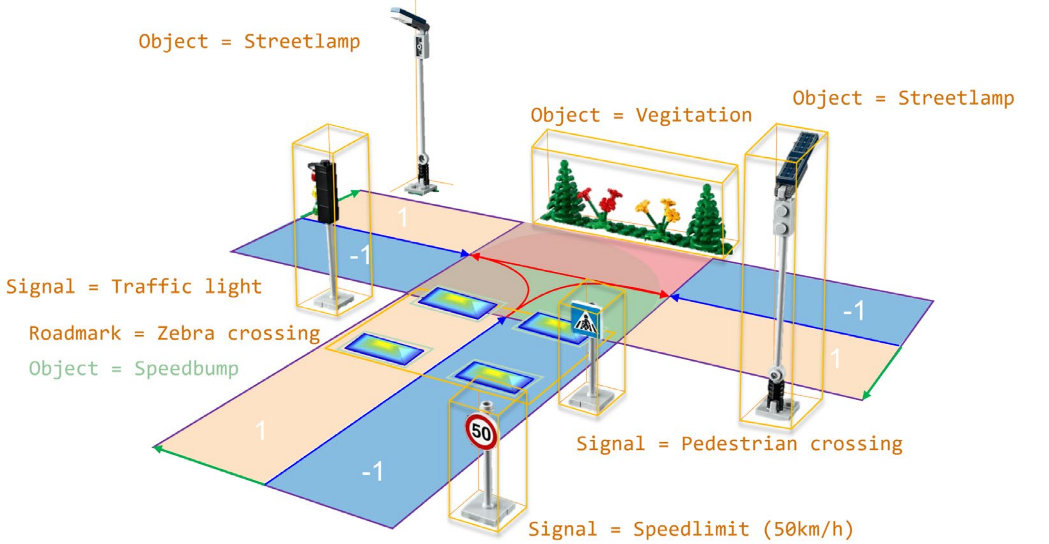
Extensibility of ontology

Permissive/restrictive

Permissive/ restrictive

- Two levels
  - At the global level
  - At individual statement level

Element specifically stated?	Restrictive	Permissive	Level
Yes	Elements is outside ODD (such as UNSUITABLE)	Element is inside ODD (such as SUITABLE)	Individual statement
No, but element is in the ontology	Element is outside ODD	Element is inside ODD	Global



Syntax 1

```
permissive_odd: ODD:
    keep(road_topology.properties.lane_dimensions in [3..5]m)
    keep(road_topology.properties.number_of_lanes in [1..2])
```

Syntax 2

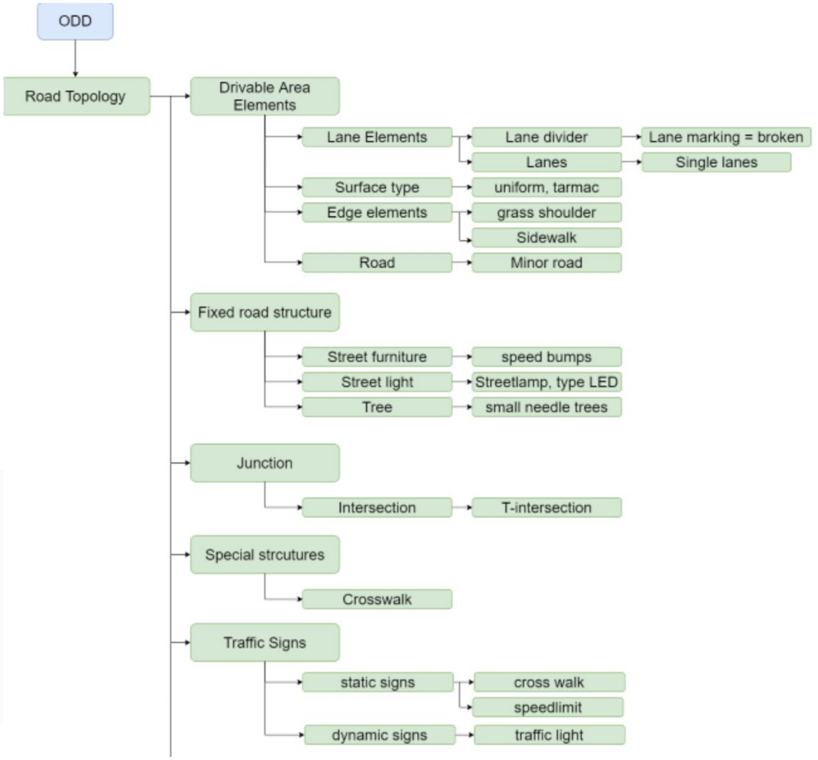
```
INCLUDE odd_ontology
MODE Permissive
SUITABLE laneDimensions is {min: 3.0, max: 5.0}
SUITABLE numberOfLanes is {min: 1.0, max: 2.0}
```

Permissive mode

```
restrictive_odd: ODD:
    keep(road_topology.drivable_area_elements.lane_elements.
         lane_divider.lane_marking == broken)
    keep(road_topology.drivable_area_elements.lane_elements.lanes == single_lane)
    keep(road_topology.properties.driving_direction == left_hand_traffic)
    keep(road_topology.properties.lane_dimensions in [3..5]m)
    keep(all_default_values()) # makes it restrictive
```

```
INCLUDE odd_ontology
MODE Restrictive
SUITABLE Lanemarking is ‘broken‘;
SUITABLE Lanes is ‘single lanes‘;
…
UNSUITABLE driving_direction is ‘left_hand_traffic‘;
SUITABLE laneDimensions is {min: 3.0, max: 5.0};
```

Restrictive mode



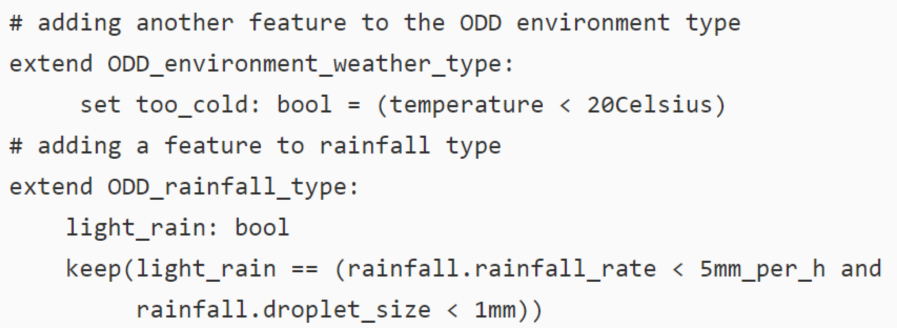
Extensibility

Extensibility of ontology

Summary:
Due to the diverse stakeholders and use cases, as well as an ever expanding ODD ontology, ODD definition language needs to support the modification of the underlying ontology from a base model.

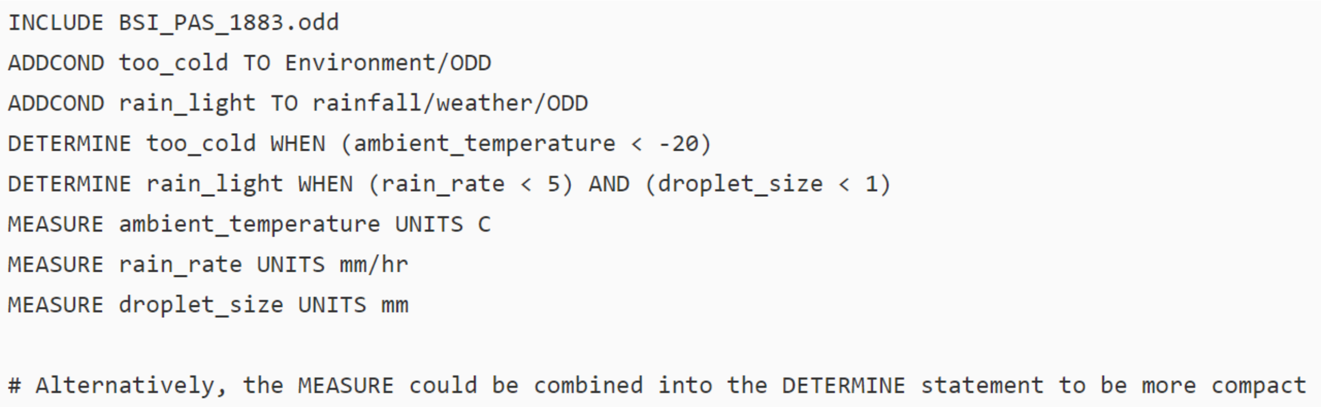
Challenge:
Base on BSI PAS 1883 taxonomy, user wants to add ‘too_cold’ attribute onto the existing ‘environment’ class, it is defined when the ambient temperature is below 20 degree C. Additionally add ‘rain_light’ attribute onto ‘rainfall’ class, defined as when rain rate is less than 5 mm/hr and droplet size less than 1mm.

Syntax 1



```
# adding another feature to the ODD environment type
extend ODD_environment_weather_type:
     set too_cold: bool = (temperature < 20Celsius)
# adding a feature to rainfall type
extend ODD_rainfall_type:
    light_rain: bool
    keep(light_rain == (rainfall.rainfall_rate < 5mm_per_h and
         rainfall.droplet_size < 1mm))
```

Syntax 2



```
INCLUDE BSI_PAS_1883.odd
ADDCOND too_cold TO Environment/ODD
ADDCOND rain_light TO rainfall/weather/ODD
DETERMINE too_cold WHEN (ambient_temperature < -20)
DETERMINE rain_light WHEN (rain_rate < 5) AND (droplet_size < 1)
MEASURE ambient_temperature UNITS C
MEASURE rain_rate UNITS mm/hr
MEASURE droplet_size UNITS mm

# Alternatively, the MEASURE could be combined into the DETERMINE statement to be more compact
```

Composability

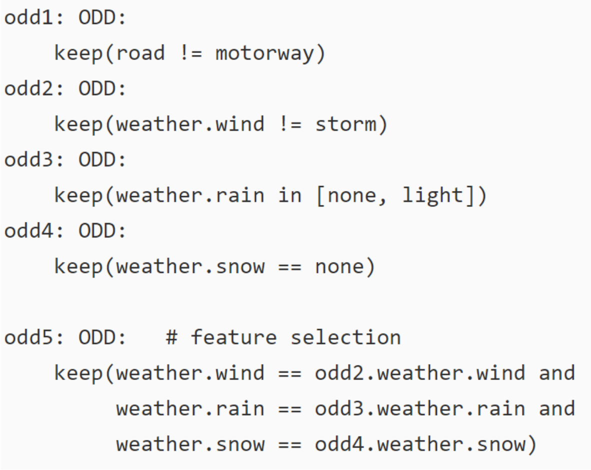
Composability

Summary:
The ASAM OpenODD language format should be composable, (i.e., combine ODD definitions into a new, wider ODD). In addition, the ability to re-use a defined section of an ODD should also be possible.

Challenge:
Given ODD_1 (motorway ODD), ODD_2 (specify wind conditions) , ODD_3 (specify rainfall conditions), ODD_4 (specify snowfall conditions).

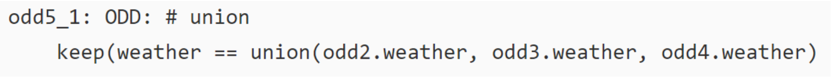
1. Construct a weather ODD (ODD_5) by combining ODD_2, ODD_3 and ODD_4
2. Illustrate how to obtain the intersection, union and difference between two ODDs

Syntax 1



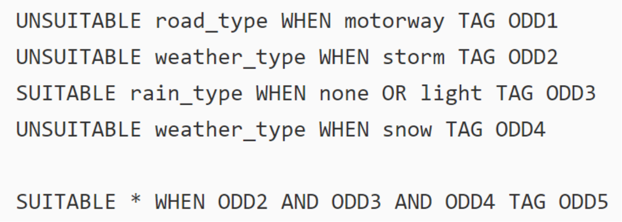
```
odd1: ODD:
    keep(road != motorway)
odd2: ODD:
    keep(weather.wind != storm)
odd3: ODD:
    keep(weather.rain in [none, light])
odd4: ODD:
    keep(weather.snow == none)

odd5: ODD:   # feature selection
    keep(weather.wind == odd2.weather.wind and
         weather.rain == odd3.weather.rain and
         weather.snow == odd4.weather.snow)
```



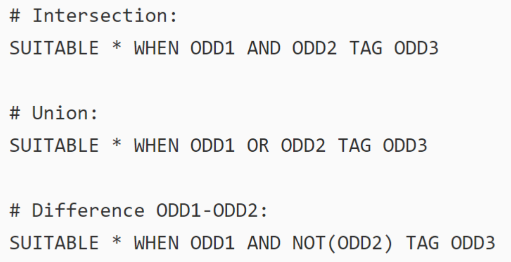
```
odd5_1: ODD: # union
    keep(weather == union(odd2.weather, odd3.weather, odd4.weather)
```

Syntax 2



```
UNSUITABLE road_type WHEN motorway TAG ODD1
UNSUITABLE weather_type WHEN storm TAG ODD2
SUITABLE rain_type WHEN none OR light TAG ODD3
UNSUITABLE weather_type WHEN snow TAG ODD4

SUITABLE * WHEN ODD2 AND ODD3 AND ODD4 TAG ODD5
```



```
# Intersection:
SUITABLE * WHEN ODD1 AND ODD2 TAG ODD3

# Union:
SUITABLE * WHEN ODD1 OR ODD2 TAG ODD3

# Difference ODD1-ODD2:
SUITABLE * WHEN ODD1 AND NOT(ODD2) TAG ODD3
```

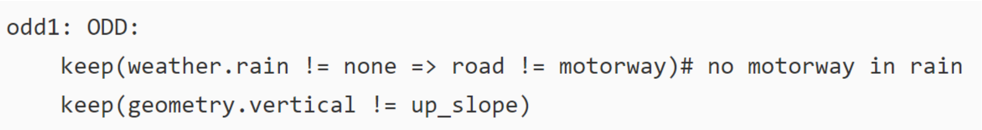
Conditional statement

Conditional statement

Summary:
Language specification needs to support conditional statement or reduced ODD. This can be tackled by imposing constraints on the full operational range of specific attributes.

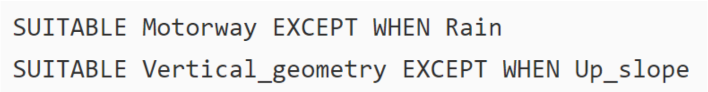
Challenge:
Given an ODD hierarchical tree, we would like to express that motorway is only suitable when there is no rain (conditional ODD statement), within the geometry, up-slope is not suitable, all weather conditions are suitable

Syntax 1



```
odd1: ODD:
    keep(weather.rain != none => road != motorway)# no motorway in rain
    keep(geometry.vertical != up_slope)
```

Syntax 2



```
SUITABLE Motorway EXCEPT WHEN Rain
SUITABLE Vertical_geometry EXCEPT WHEN Up_slope
```

Thank you for your attention!

Dr Xizhe Zhang
Format WP lead – ASAM OpenODD Concept project

Lead engineer (simulation lead), Intelligent Vehicles,
WMG, University of Warwick, UK

Email: jason.zhang@warwick.ac.uk

ASAM