

SLA*

An Abstract Syntax for Service Level Agreements

Keven T. Kearney

Engineering Ingegneria Informatica SpA

Francesco Torelli

Engineering Ingegneria Informatica SpA

Constantinos Kotsokalis

ITC University of Dortmund

Developed for SLA@SOI
<http://sla-at-soi.eu/>

Supported by European Community's
7th Framework Programme
FP7/2007-2013
Grant agreement no. 216556

SLA*: Presentation Overview

- Modelling Objectives
- Core Expression Types
- Vocabularies
- Interface Specifications
- SLAs and SLA Templates : SLA(T)s
- QoS Metrics
- Concrete Implementations

SLA*: Presentation Overview

- Modelling Objectives
- Core Expression Types
- Vocabularies
- Interface Specifications
- SLAs and SLA Templates : SLA(T)s
- QoS Metrics
- Concrete Implementations

SLA*: Objectives

- Support for SLA Life-Cycle Management
 - QoS-based Discovery
 - (re)Negotiation (*cf.* WS-Agreement)
 - Provisioning (resource allocation)
 - Monitoring (effective SLA period / service execution)
 - domain-independence
- Extensibility & Customisability
 - domain-specific extensions
- Language Independence
 - e.g. supports XML serialisation, but not limited to XML

SLA*: SLA Life-Cycle Management

- Constraint Satisfaction

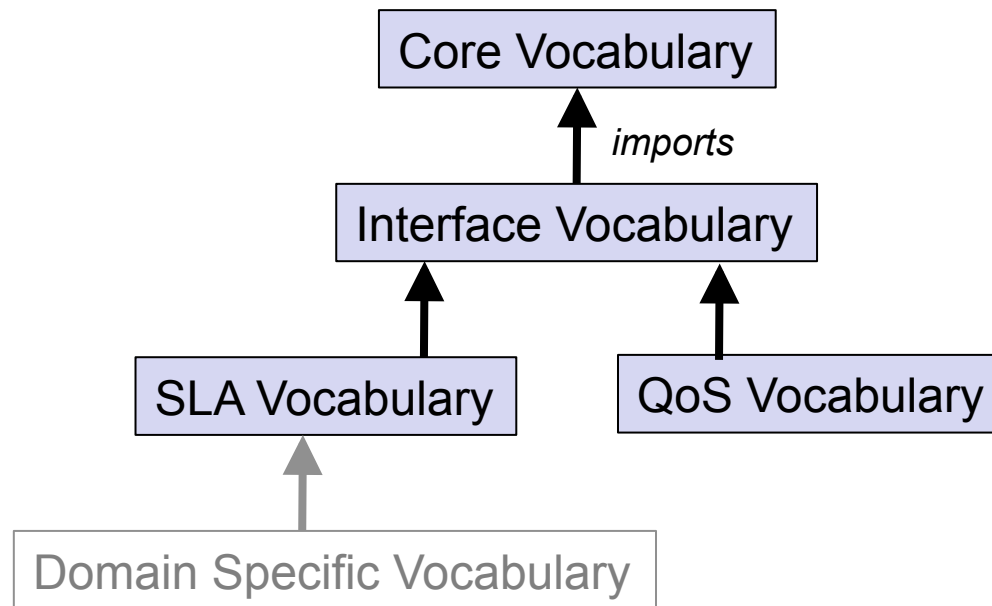
- e.g. QoS-based discovery
 - provider offers: “completion-time(S) < 10s”
 - customer requires: “completion-time(S) < 1 min”
- Constraint Language
 - fixed form constraint expressions
 - specification of:
 - constraint ‘evaluation’ procedures
 - semantic relations between operators (‘< ‘ is more stringent than ‘<= ‘)
 - data-conversion formula (e.g. 1min = 60s)

- Minimal Requirements for Document Content

- Functional Interface Specifications (*cf.* WSDL)
- SLAs & SLA Templates (SLATs)

SLA*: Extensibility & Customisability

- Pluggable Domain ‘Vocabularies’
 - Vocabulary specifies document content
 - Extensible (document ‘imports’)
 - Modular (use only the vocabularies required by the domain)



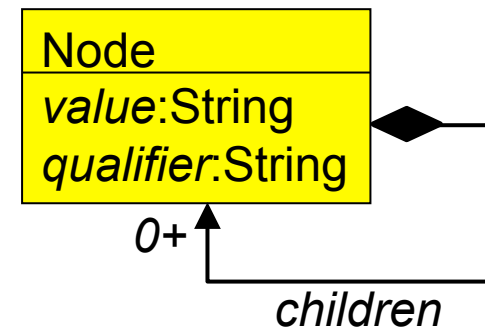
SLA*: Language Independence

- SLA* is an “Abstract Syntax”
 - Specifies high-level syntactic requirements (*expression types*)
 - Does not specify symbols (*expression tokens*)
 - e.g.

type { an ordered pair $\langle x, y \rangle$ where x is ... y is ...

token {
x(y)
<x y=“ ”>...</x>
<token>
 <x> ... </y>
 <x> ... </y>
</token>
The ‘x’ of
‘y’

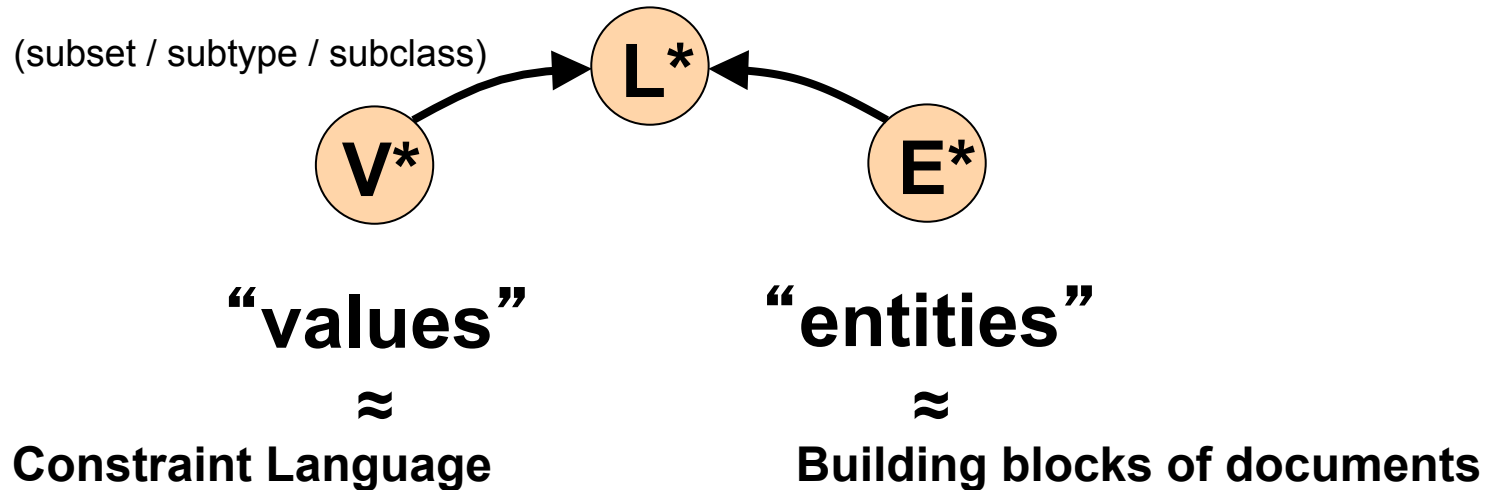
Underlying structure ...



SLA*: Presentation Overview

- Modelling Objectives
- **Core Expression Types**
- Vocabularies
- Interface Specifications
- SLAs and SLA Templates : SLA(T)s
- QoS Metrics
- Concrete Implementations

SLA*: Expression Types

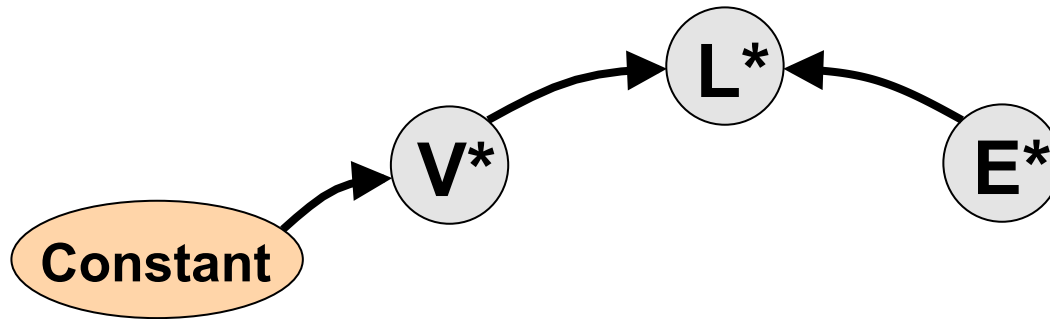


L* = the set (type/class) of all *legal* expressions

V* = the set of all legal *value* expressions

E* = the set of all legal *entity* expressions

SLA*: Constants & Datatypes



Constant

abstract supertype of all constant (literal) values

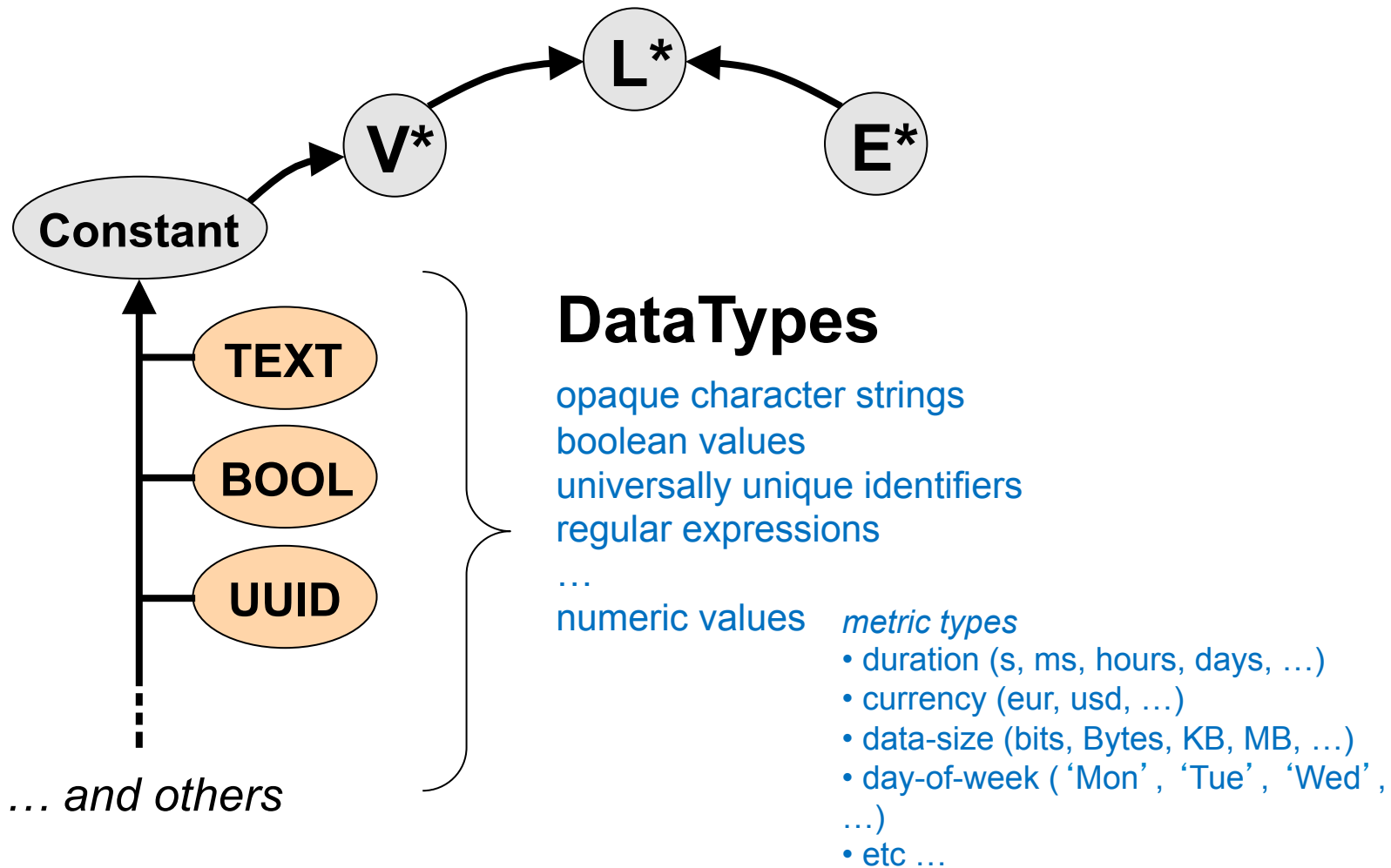
e.g.

<http://www.slaatsoi.com>

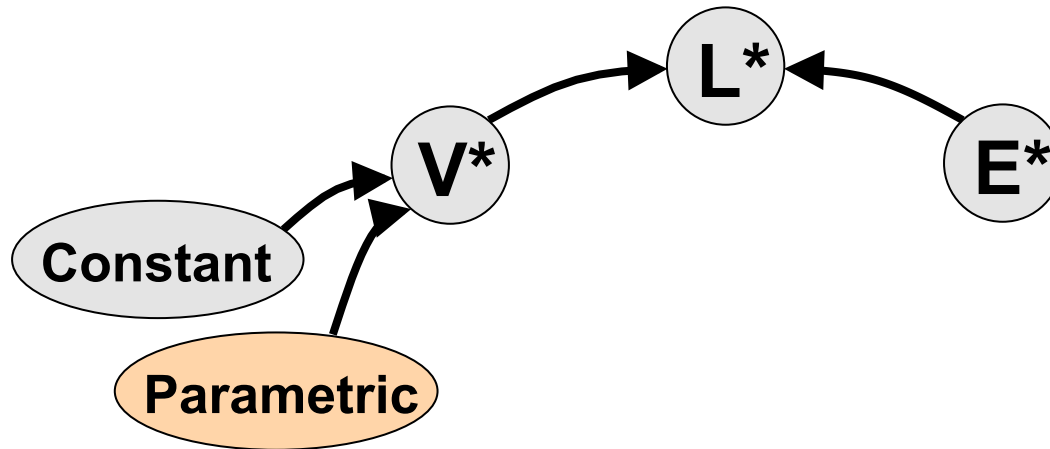
42 seconds

26th October 2010 15:00.00

SLA*: Constants & Datatypes



SLA*: Parametric Expressions



Parametric

“functional” or “predicate” form expressions

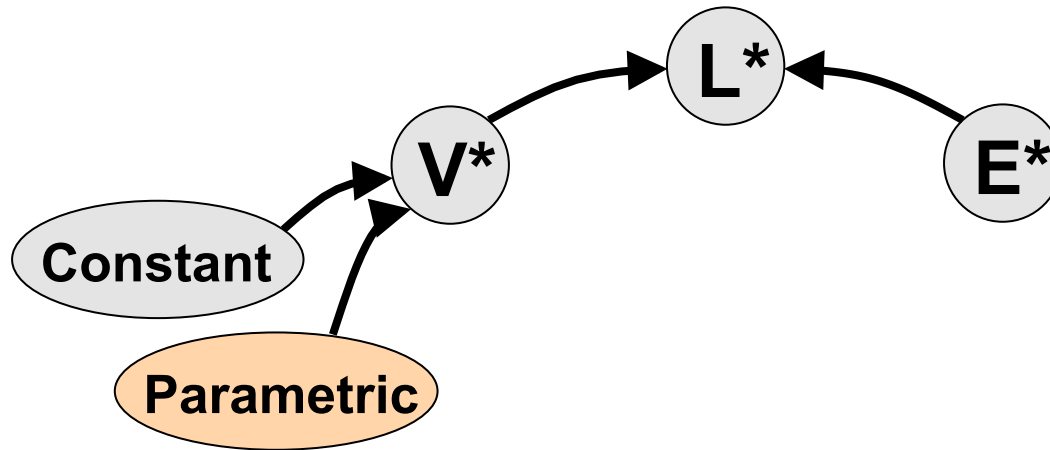
e.g.

sum(4, 5.6, 78)

completion-time(service-operation-X)

time-series(mean(completion-time(X)), period(24 hrs), 10)

SLA*: Parametric Expressions



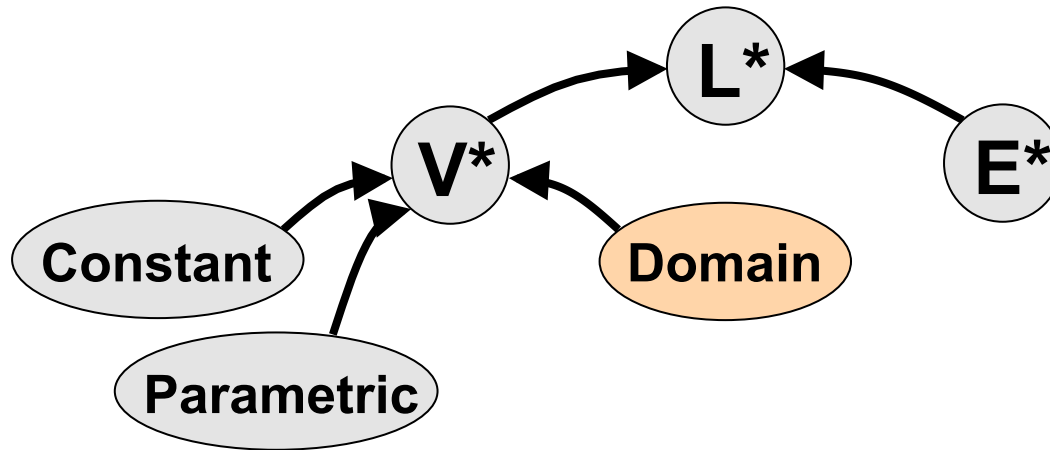
Parametric

an ordered pair $\langle f, P \rangle$ where:

$f \in \mathbf{UUID}$ uniquely identifies the “function”

$P \subset V^*$ is an ordered set of parameters

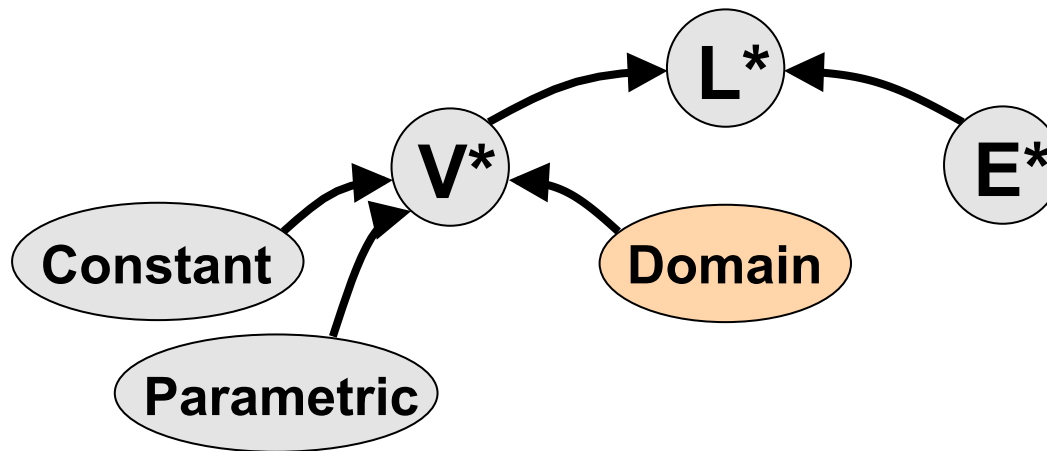
SLA*: Domain Expressions



e.g.
< 4 seconds
matches "[a-zA-Z0-9]*"
one-of [A, B, C]

e.g.
> 2s **and** < 4s
one-of [A, B, C] **or** one-of [E, F, G]
not (>= 20 KB and < 30 KB)

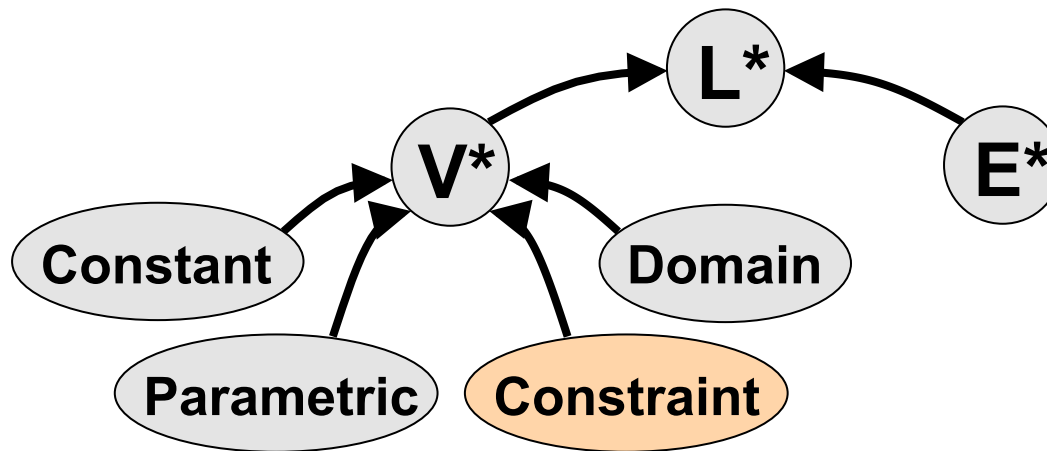
SLA*: Domain Expressions



an ordered pair $\langle o, v \rangle$ where:
 o uniquely identifies a 'comparison operator', and $v \in V^*$ specifies a domain boundary

an ordered pair $\langle o, D \rangle$ where:
 o uniquely identifies a 'logical operator', and $D \in \text{Domain}$ is an unordered set of sub-domains

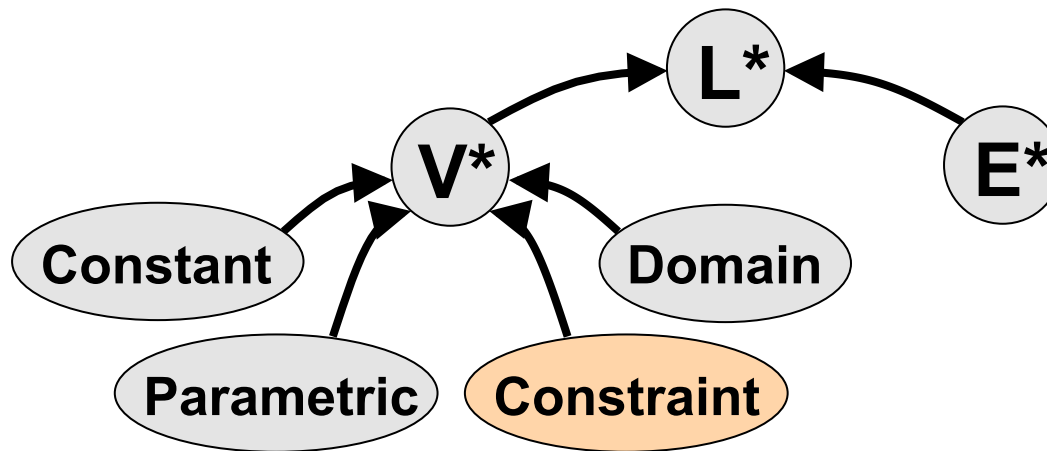
SLA*: Constraint Expressions



e.g.
completion-time(X) < 4 seconds
“xyz40” matches “[a-zA-Z0-9]*”
X one-of [A, B, C]

e.g.
x > 2s and x < 4s
X one-of [A, B, C] or X one-of [E, F, G]
not (“a” matches “b”)

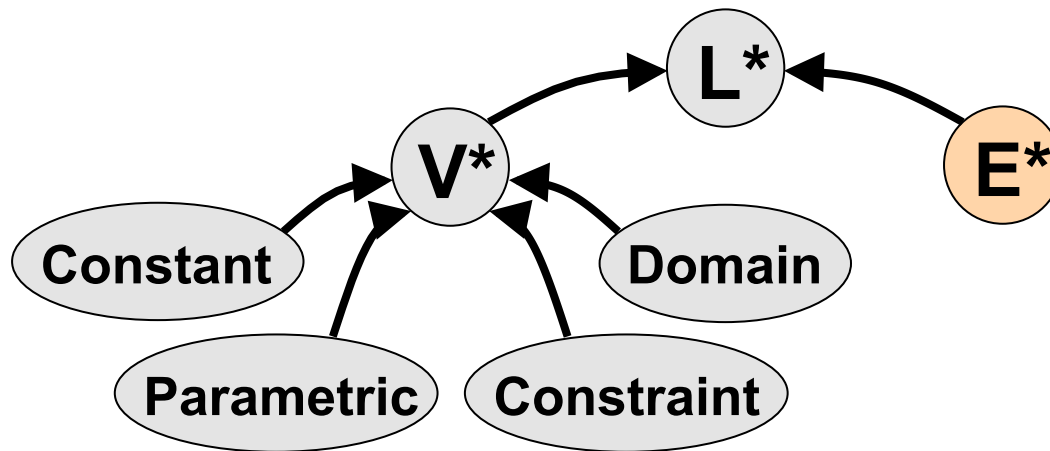
SLA*: Constraint Expressions



an ordered pair $\langle v, d \rangle$ where:
 $v \in V^*$ is a value constrained to lie
in the domain $d \in \text{Domain}$

an ordered pair $\langle o, C \rangle$ where:
 o uniquely identifies a 'logical
operator', and $C \in \text{Constraint}$ is
an unordered set of sub-constraints

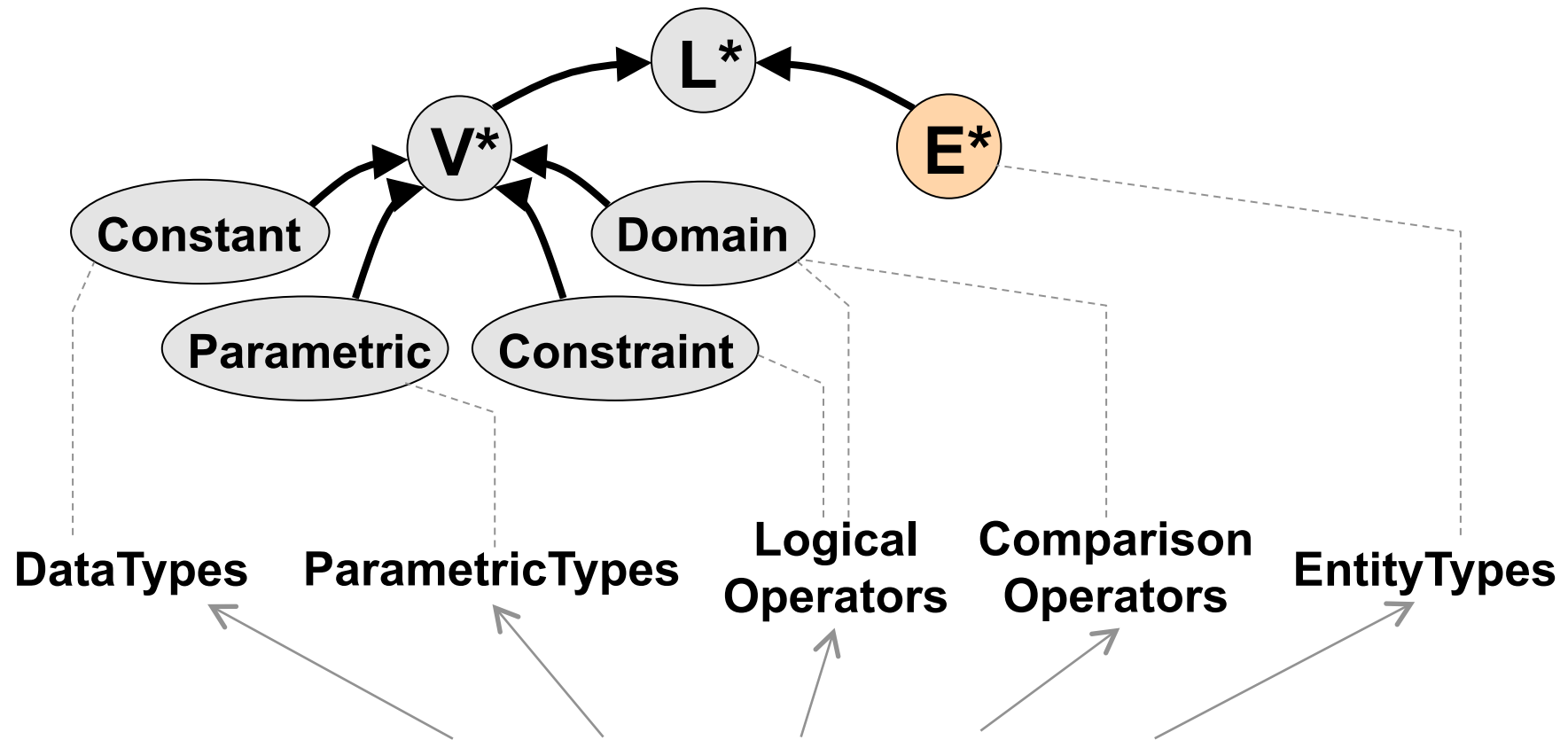
SLA*: Entity Expressions



E (entity)

an 'entity' in document content hierarchy, comprising:
an unordered set of ordered key/value attribute pairs
 $\langle k, v \rangle$, where: $k \in \mathbf{NAME}$ is the name (key) of the
attribute, and $v \in L^*$ gives the attribute's value

SLA*: Vocabularies

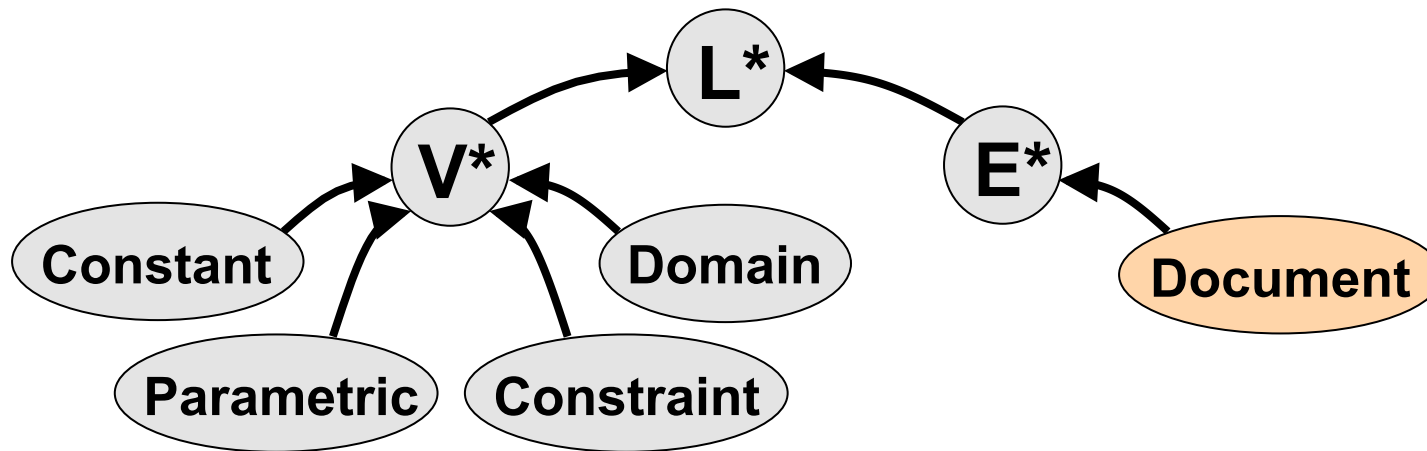


Subtypes, operators, data-formats, etc ..
are specified in **Vocabularies**

SLA*: Presentation Overview

- Modelling Objectives
- Core Expression Types
- **Vocabularies**
- Interface Specifications
- SLAs and SLA Templates : SLA(T)s
- QoS Metrics
- Concrete Implementations

SLA*: Vocabularies



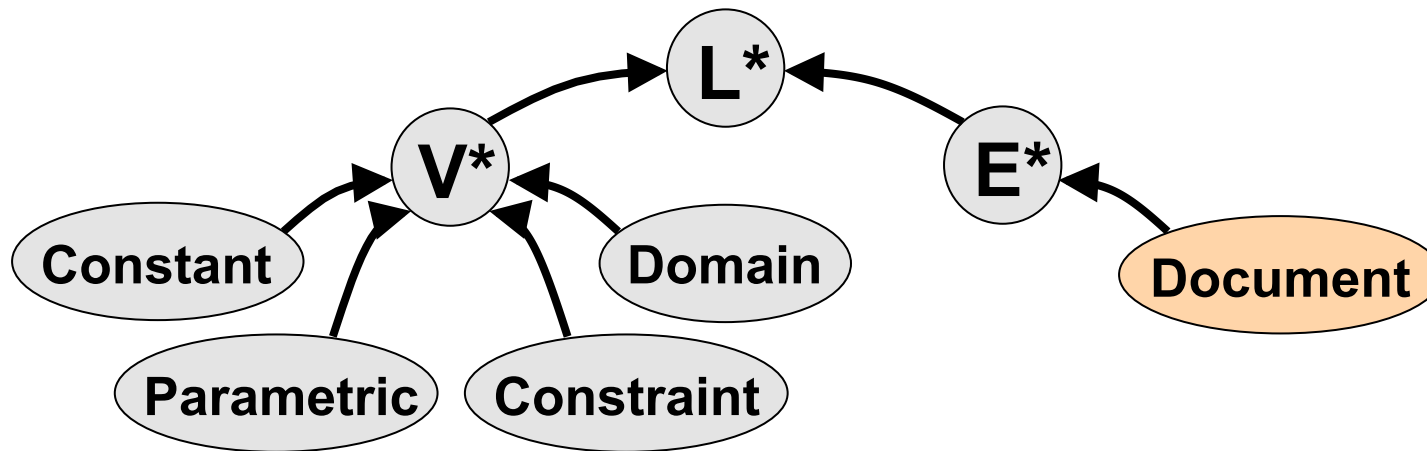
ENTITY-TYPE DEFINITION

core:Document \subset *core:E**

vocabularies \subset (^)*core:Vocabulary* [0+]

macros \subset *core:Macro* [0+]

SLA*: Vocabularies

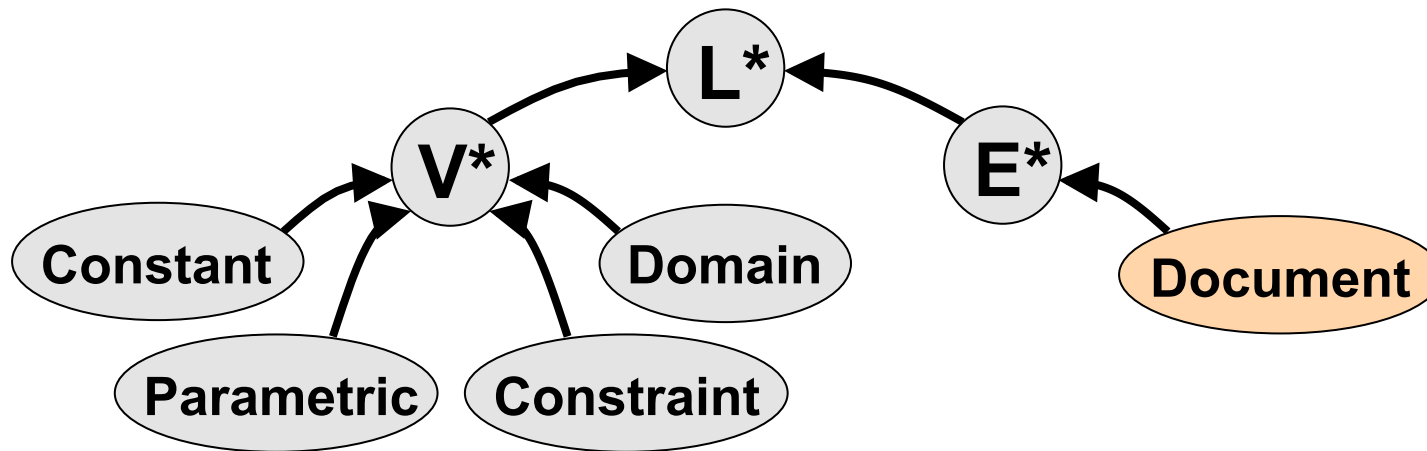


core:Document \subset core:E* Super-type
vocabularies \subset (^)core:Vocabulary [0+]
macros \subset core:Macro [0+] attribute type

UUID of the core vocabulary "Document" entity type

attribute type

SLA*: Vocabularies



core:Document \subset *core:E**

vocabularies \subset (^) *core:Vocabulary* [0+]

macros \subset *core:Macro* [0+]

Cardinality on values

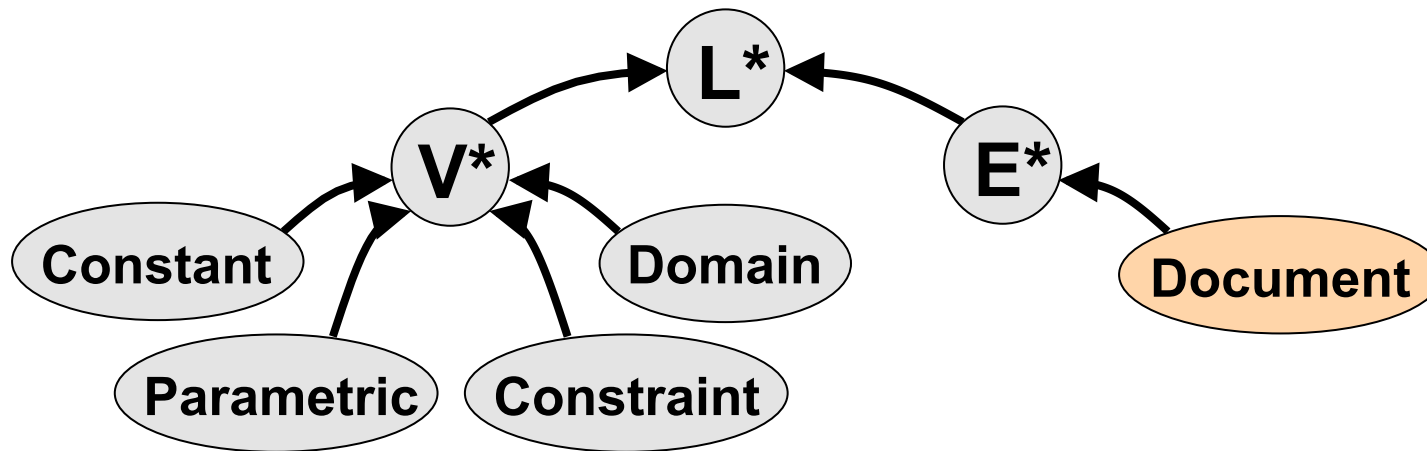
NAME of the attribute (key)

The attribute's value type

Prefix indicating "reference type"

In this case: signifies *either* an instance *or* a reference to an instance of a vocabulary document

SLA*: Vocabularies



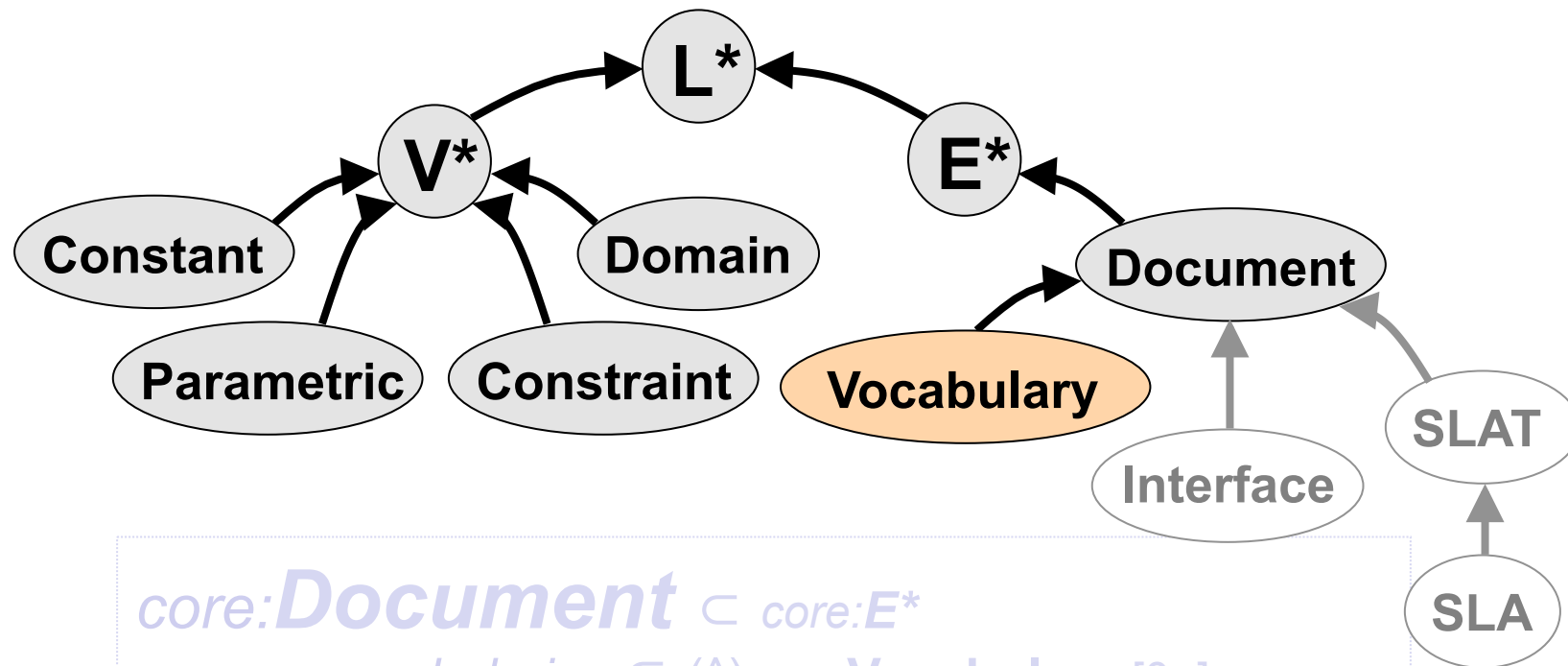
core:Document \subset *core:E**

vocabularies \subset (^)*core:Vocabulary* [0+]

macros \subset *core:Macro* [0+]

A document is an entity which has a “vocabularies” attribute whose value is a (possible empty) list of either vocabularies or references to vocabularies

SLA*: Vocabularies



core:Document \subset *core:E**

vocabularies \subset (^) *core:Vocabulary* [0+]

macros \subset *core:Macro* [0+]

core:Vocabulary \subset *core:Document*

terms \subset *core:Term* [1+]

SLA*: Vocabulary Terms

Term $\subset E^*$
definition $\subset \text{TEXT}$ [1]

Abstract supertype of vocabulary terms

- human readable semantic definition

EntityType $\subset \text{Term}$
uuid $\subset \text{UUID}$ [1]
supertype $\subset \wedge E^*$ [1]
concrete $\subset \text{BOOL}$ [1]
attributeTypes $\subset \text{AttributeType}$ [0+]
observableTypes $\subset \text{ObservableType}$ [0+]

Defines an entity type

- universally unique identifier for the type
- reference to the supertype
- 'yes' if the type can be instantiated
- list of attribute types for the entity
- (see later)

AttributeType $\subset \text{NamedEntity}$
valuetype $\subset \wedge L^*$ [1]
cardinality $\subset \text{CARD}$ [1]
def $\subset \text{TEXT}$ [1+]

Defines an attribute type

- reference to the attribute's value type
- cardinality on the attribute's value
- human readable semantic definition

NamedEntity $\subset E^*$
name $\subset \text{NAME}$ [1]

An entity with a locally unique name

- the name of the entity

SLA*: Vocabulary Terms (continued)

DataType \subset Term

uuid \subset UUID [1]

supertype \subset ^^Constant [1]

concrete \subset BOOL [1]

Defines a data type

- *universally unique identifier for the type*
- *reference to the supertype*
- *'yes' if the type can be instantiated*

DataValue \subset Term

regex \subset REGEX [1]

role \subset ^^Constant [1]

stnd \subset STND [0..1]

Defines a data format / enumeration

- *regular expression constraining data values*
- *the DataType to which this format applies*
- *standard form (data conversion formula)*

```
DataValue{
  regex : [D%i1-31]/[M%i1-12]/[Y%i] [h%i0-23]:[m%i0-59]:[s] CET
  role : DATETIME
  def : Regular expression for DATETIME values
}
DataValue{
  regex : [x] ms
  role : DURATION
  stnd : [x/1000] s
  def : format for durations expressed in units of milliseconds
}
```

*Concrete
Examples
"BNF
syntax"*

SLA*: Vocabulary Terms (continued)

ParametricType \subset Term

uuid \subset UUID [1]

role \subset $\wedge\wedge$ Constant [1]

cardinality \subset CARD [1]

macros \subset Macro [0+]

parameterTypes \subset ParameterType [0+]

Defines a parametric type

- *universally unique identifier for the type*
- *the type to which this parametric evaluates*
- *cardinality on the role*
- *list of local macros*
- *ordered list of parameter types*

ParameterType \subset E*

valuetype \subset $\wedge\wedge$ V* [1]

cardinality \subset CARD [1]

Defines an parameter type

- *reference to the parameter's type*
- *cardinality on the parameter*

DomainOp \subset ParametricType

evaluator \subset PROC [1]

broader \subset $\wedge\wedge$ DomainOp [0+]

Defines a domain operator type

- *procedure for evaluating the operator*
- *references to any broader domain operators*

CompoundOp \subset Term

uuid \subset UUID [1]

cardinality \subset CARD [1]

evaluator \subset PROC [1]

Defines a compound operator type

- *universally unique identifier for the type*
- *arity of the operator*
- *procedure for evaluating the operator*

SLA*: Vocabulary Terms (continued)

ObservableType \subset *ParametricType*
name \subset NAME [1]

Defines an observable property type

- *name of the type*

Observed \subset E^*
observed \subset E^* [1]
observableTypes \subset *ObservableType* [1+]

Associates observable with entities

- *reference to the (named) entity being observed*
- *list of observable properties for that entity*

Observed types specify properties that are in principle monitorable. They can be defined for:

- *types (see *EntityType*) : in which case they apply to all instances of that type*
- *instances (see *Observed*) : in which case they apply just to that instance*

Macro \subset *NamedEntity*
expression \subset V^* [1]

A macro ('variable') expression

- *the expression denoted by this macro*

A convenience mechanism permitting complex expressions to be denoted by short names (cf. the use of macro statements in C)

SLA*: The Core Vocabulary

Vocabulary Terms

- **EntityType**
 - AttributeType
 - ObservableType
- **DataType**
- **DataValue**
- **ParametricType**
 - ParameterType
- **DomainOp**
- **CompoundOp**

Set-based Parametrics

- **sum**(a:N₁₊) → N₁ (NUMERIC)
- **std**(a:N₁₊) → N₁ (NUMERIC)
- **mean**(a:N₁₊) → N₁ (NUMERIC)
- **median**(a:N₁₊) → N₁ (NUMERIC)
- **mode**(a:N₁₊) → N₁ (NUMERIC)
- **max**(a:N₁₊) → N₁ (NUMERIC)
- **min**(a:N₁₊) → N₁ (NUMERIC)
- **count**(a:Constant₁₊,d:Domain₁) → COUNT₁
- **percent**(a:N₁₊,d:Domain₁) → PERCENT₁

Entities

- **Document**
- **Vocabulary**
- **NamedEntity**
- **Macro**
- **Observed**

DataTypes

- **TEXT**
- **NAME**
- **REGEX**
- **UUID**
- **ENUM**
- **CARD**
- **BOOL**
- **STND**
- **PROC**
- **DATETIME**
- **TIME**
- **NUMERIC**

Arithmetic Parametrics

- **add**(a:N₁,b:N₁) → N (NUMERIC)
- **subtract**(a:N₁,b:N₁) → N (NUMERIC)
- **multiply**(a:N₁,b:N₁) → N (NUMERIC)
- **divide**(a:N₁,b:N₁) → N (NUMERIC)
- **modulo**(a:N₁,b:N₁) → N (NUMERIC)
- **round**(a:N₁,b:COUNT₁) → N (NUMERIC)

Miscellaneous Parametrics

- **time_is**() → TIME₁
- **day_is**() → DAY₁
- **month_is**() → MONTH₁
- **time_series**(f:F₁,e:EventClass₁,c:COUNT₁) → F₁₊ (Constant)
- **specialisation**(e:EventClass₁,c:Constraint₁) → EventClass₁
- **periodic**(s:DATETIME₁,d:DURATION₁) → EventClass₁
- **schedule**(s:DATETIME₁₊) → EventClass₁

Metric Units

- **QUANTITY**
- **COUNT**
- **PERCENT**
- **DATARATE**
- **TXRATE**
- **LENGTH**
- **AREA**
- **FREQUENCY**
- **WEIGHT**
- **POWER**
- **ENERGY**
- **DAY**
- **MONTH**
- **YEAR**
- **DURATION**
- **CURRENCY**
- **DATASIZE**

SLA*: Presentation Overview

- Modelling Objectives
- Core Expression Types
- Vocabularies
- **Interface Specifications**
- SLAs and SLA Templates : SLA(T)s
- QoS Metrics
- Concrete Implementations

SLA*: The Interface Vocabulary

Functional interface specifications ...

Interface \subset Document
extended \subset (^)Interface [0+]
operations \subset Operation [0+]

An interface specification document

- *list of interfaces extended by this interface*
- *list of operations exposed by this interface*

Operation \subset Service
faults \subset ^^L [0+]
inputs \subset Property [0+]
outputs \subset Property [0+]
related \subset Property [0+]

An interface operation

- *list of faults / exceptions thrown*
- *list of inputs to the operation*
- *list of operation outputs*
- *list of other properties related to the operation*

Property \subset NamedEntity
datatype \subset ^^L [1]
domain \subset Domain [0..1]
auxiliary \subset BOOL [1]

An interface operation property

- *reference to the property's datatype*
- *optional domain limiting the property's value*
- *'true' if null values are accepted*

(Generalisation of WSDL 2.0)

SLA*: The Interface Vocabulary

Abstract 'service' type ...

***Service** \subset NamedEntity*

ObservableTypes:

*invocation_uuid : UUID,
request_time : DATETIME,
reply_time : DATETIME,
endpoint_uuid : UUID,
consumer_uuid : UUID*

Abstract supertype of "services"

- observable properties of service invocations ...*
 - unique identifier for the invocation*
 - time invocation was received*
 - time invocation was complete*
 - EPR at which the invocation occurred*
 - unique identifier of the consumer*

Parametrics ...

service(^Service [0+]) \rightarrow ^Service [1]

fault(^Service [1] , ^^L [1]) \rightarrow EventClass [1]*

SLA*: Presentation Overview

- Modelling Objectives
- Core Expression Types
- Vocabularies
- Interface Specifications
- **SLAs and SLA Templates : SLA(T)s**
- QoS Metrics
- Concrete Implementations

SLA*: The SLA Vocabulary

SLAs & SLA Templates...

SLA \subset SLAT

agreedAt \subset DATETIME [1]

effectiveFrom \subset DATETIME [1]

effectiveUntil \subset DATETIME [1]

template \subset ^SLAT [0..1]

ObservableTypes:

state : ENUM

A Service Level Agreement (SLA)

- *date SLA was agreed*
- *date SLA comes into effect*
- *date SLA ceases to be effective*
- *reference to SLAT from which SLA was derived*

- *observable states of the SLA*

DataValues:

SLA state : 'agreed', 'terminated', 'violated', 'warned', 'satisfied'

SLAT \subset Document

parties \subset Party [2+]

interfaceDecls \subset InterfaceDeclr [1+]

agreementTerm \subset AgreementTerm [1+]

An SLA Template

- *list of parties*
- *declarations of relevant interface specifications*
- *list of agreement terms*

SLA*: The SLA Vocabulary

SLA Parties...

Actor \subset NamedEntity

Abstract supertype of SLA actors

Party \subset Actor

role \subset ENUM [1]

operatives \subset Operative [0+]

A Party to the SLA

- *the role of the party in the agreement*
- *list of party operatives*

DataValues:

Party role : 'provider', 'customer'

Operative \subset Actor

A sub-category of party specific actors

Customisables ...

Customisable \subset Macro

domain \subset Domain [1]

*A customisable ('variable')
expression with the permitted values*

SLA*: The SLA Vocabulary

Interface Declarations...

InterfaceDeclr \subset Service

provider \subset ^Actor [1]

consumers \subset ^Actor [1+]

endpoints \subset Endpoint [1+]

interface \subset (^)Interface [1]

Declaration of a functional interface

- *obligated provider of the interface*
- *consumers permitted to use the interface*
- *endpoints for invocation messages*
- *functional specification of the interface*

Endpoint \subset Service

location \subset TEXT [0..1]

protocol \subset ENUM [1]

An endpoint for interface invocations

- *the address (EPR) of the endpoint*
- *communications protocol*

DataValues:

Endpoint protocol : 'telephone', 'HTTP', 'SOAP', 'email', 'SMS',
'REST', 'XMPP', 'postal', 'fax',
'SSH'

SLA*: The SLA Vocabulary

Agreement Terms...

AgreementTerm \subset *NamedEntity*

pre \subset *Constraint* [0..1]

guarantees \subset *Guarantee* [1+]

macros \subset *Macro* [0+]

ObservableTypes:

state : *ENUM*

An SLA Agreement Term

- *preconditions on the applicability of the term*
- *list of guarantees constituting the term*
- *local macro ('variable') definitions*
- *observable state of the agreement term*

DataValues:

AgreementTerm state : 'violated', 'warned', 'satisfied'

Guarantee \subset *NamedEntity*

obligated \subset *^Actor* [1]

ObservableTypes:

state : *ENUM*

Abstract supertype of SLA guarantees

- *the SLA party obligated to satisfy the guarantee*
- *observable state of the guarantee*

DataValues:

Guaranteed state : 'violated', 'warned', 'satisfied'

SLA*: The SLA Vocabulary

Guarantees...

State \subset Guarantee
priority \subset Constant [0..1]
pre \subset Constraint [0..1]
post \subset Constraint [1]

A guaranteed 'state of affairs'

- *optional priority level*
- *precondition on the applicability of the guarantee*
- *the guaranteed state*

Action \subset Guarantee
policy \subset ENUM [1]
pre \subset EventClass [1]
limit \subset DURATION [1]
post \subset ActionDef [1]

A guaranteed action

- *action policy*
- *the conditions under which the action is triggered*
- *a time limit on performing the action*
- *definition of the action to be performed*

DataValues:

Action policy : 'mandatory', 'optional', 'forbidden'

ActionDef \subset E*

Abstract supertype of action definitions

SLA*: The SLA Vocabulary

Action postconditions...

Invocation \subset *ActionDef*

operation \subset \wedge Operation [1]

parameters \subset InvocationParameter [0+]

Invocation of an interface operation

- *the operation to be invoked*
- *list of input values to the operation*

InvocationParameter \subset E^*

property \subset \wedge ObservableType [1]

value \subset L [1]

Value of an operation property

- *the (monitorable) property of the operation*
- *the value of the input*

Payment \subset *ActionDef*

recipient \subset \wedge Actor [1]

value \subset V^* [1]

Economic transfer of goods

- *reference to the actor receiving the goods*
- *expression denoting the value of the goods*

Termination \subset *ActionDef*

Termination of an SLA

SLA*: Presentation Overview

- Modelling Objectives
- Core Expression Types
- Vocabularies
- Interface Specifications
- SLAs and SLA Templates : SLA(T)s
- **QoS Metrics**
- Concrete Implementations

SLA*: The QoS Vocabulary

QoS Parametrics (definitions not shown)...

availability(^Service [1]) → QUANTITY [1]

accessibility(^Service [1]) → QUANTITY [1]

arrival_rate(^Service [1]) → TXRATE [1]

throughput(^Service [1]) → TXRATE [1]

completion_time(^Service [1]) → DURATION [1]

mttr(^Service [1]) → DURATION [1]

mttf(^Service [1]) → DURATION [1]

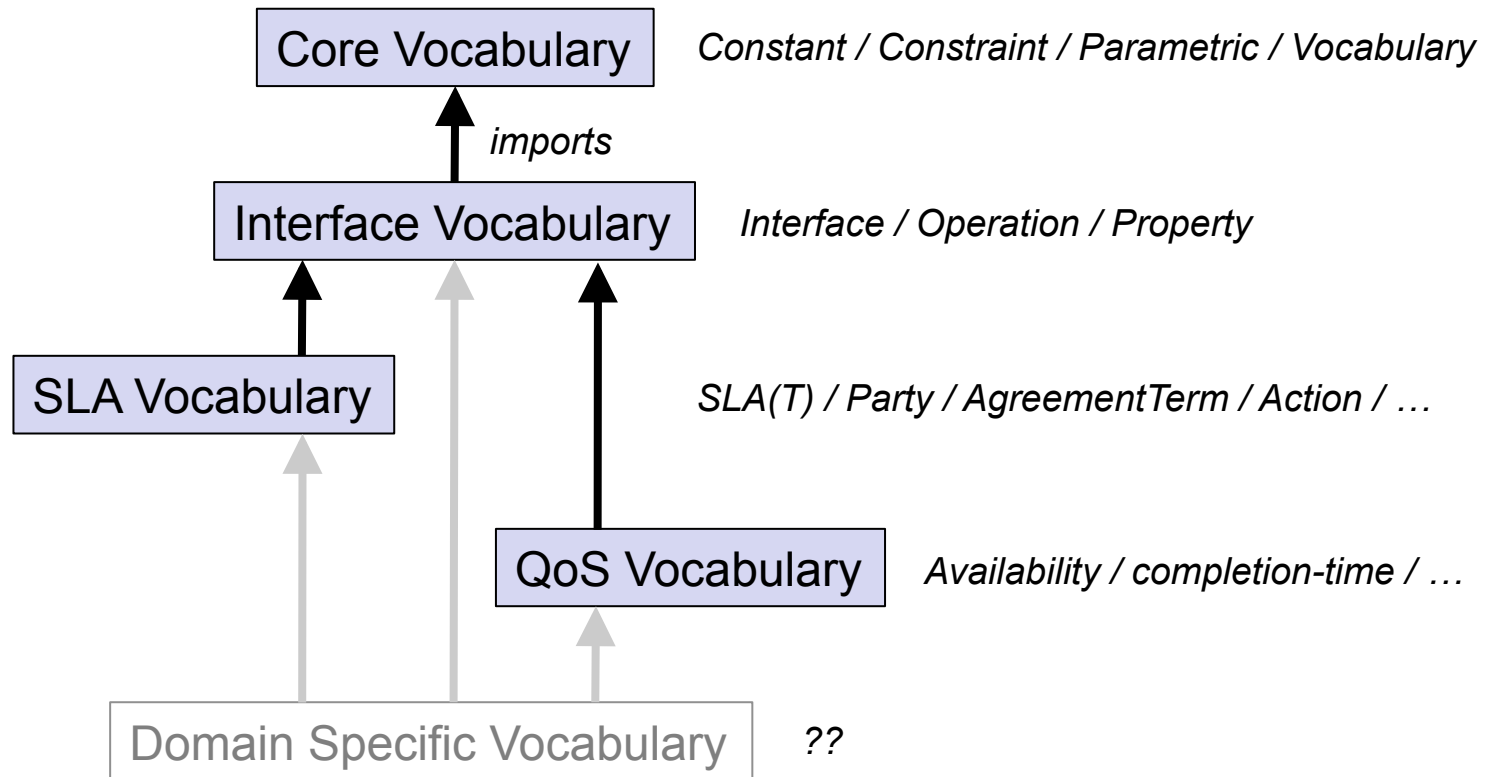
isolation(^Service [1]) → BOOL [1]

non_repudiation(^Service [1]) → TEXT [1+]

supported_standards(^Service [1]) → TEXT [1+]

regulatory(^Service [1]) → TEXT [1+]

SLA*: Summary of Vocabularies



SLA*: Presentation Overview

- Modelling Objectives
- Core Expression Types
- Vocabularies
- Interface Specifications
- SLAs and SLA Templates : SLA(T)s
- QoS Metrics
- **Concrete Implementations**

SLA*: Compact Vocabulary Syntax

e.g. the previous vocabulary slides ...

Interface \subset Document
extended \subset (^)Interface [0+]
operations \subset Operation [0+]

An interface specification document

- *list of interfaces extended by this interface*
- *list of operations exposed by this interface*

Operation \subset Service
faults \subset ^^L [0+]
inputs \subset Property [0+]
outputs \subset Property [0+]
related \subset Property [0+]

An interface operation

- *list of faults / exceptions thrown*
- *list of inputs to the operation*
- *list of operation outputs*
- *list of other properties related to the operation*

Property \subset NamedEntity
datatype \subset ^^L [1]
domain \subset Domain [0..1]
auxiliary \subset BOOL [1]

An interface operation property

- *reference to the property's datatype*
- *optional domain limiting the property's value*
- *'true' if null values are accepted*

SLA*: BNF Example

```
sla#SLA {
  sla#agreedAt : 26.10.2010 15:00.0 CET
  sla#parties :
    "Engineering" : sla#Party {
      sla#role : provider
    }
  sla#interfaceDecls :
    "THE_SERVICE" : sla#InterfaceDeclr {
      sla#provider : Engineering
      sla#interface : http://www.eng.it/aservice.wsdl
      sla#endpoints :
        "EPR-1" : sla#Endpoint {
          sla#protocol : SOAP
        }
      }
  sla#agreementTerms :
    "Term01" : sla#AgreementTerm {
      sla#guarantees :
        "State01" : sla#State {
          sla#obligated : Engineering
          sla#post :
            qos#completion_time( THE_SERVICE ) <= 5 ms
        }
      }
    }
}
```

(not all details shown)

SLA*: XML Example

```
<sla:SLA>
  <sla:agreedAt>26.10.2010 15:00.0 CET</sla:agreedAt>
  ...
  <sla:parties>
    <sla:Party>
      <core:name>Engineering</core:name>
      <sla:role>provider</sla:role>
    ...
  <sla:agreementTerms>
    <sla:AgreementTerm>
      <core:name>Term01</core:name>
      <sla:guarantees>
        <sla:State>
          <core:name>State01</core:name>
          <sla:obligated>Engineering</sla:obligated>
          <sla:post>
            <core:AtomicConstraint>
              <core:Parametric op="&qos;completion_time">
                THE_SERVICE
              </core:Parametric>
              <core:AtomicDomain op="&core;less_than_or_equals">
                5 ms
              </core:AtomicDomain>
            ...
  ...
```

(not all details shown)

SLA*: Human-Readable Example

The following service level agreement, henceforth “the agreement”, was agreed on **26th October 2010** at **15:00.0 CET**. The agreement comes into effect on the **1st November 2010** at **00:00.0 CET** and ceases to be effective on **1st November 2011** at **00:00.0 CET**.

The signatory parties to the agreement are:

The first party, Engineering Ingegneria Informatica S.p.A, henceforth referred to as **Engineering**, acting in the role of service **provider**.

The second party ...

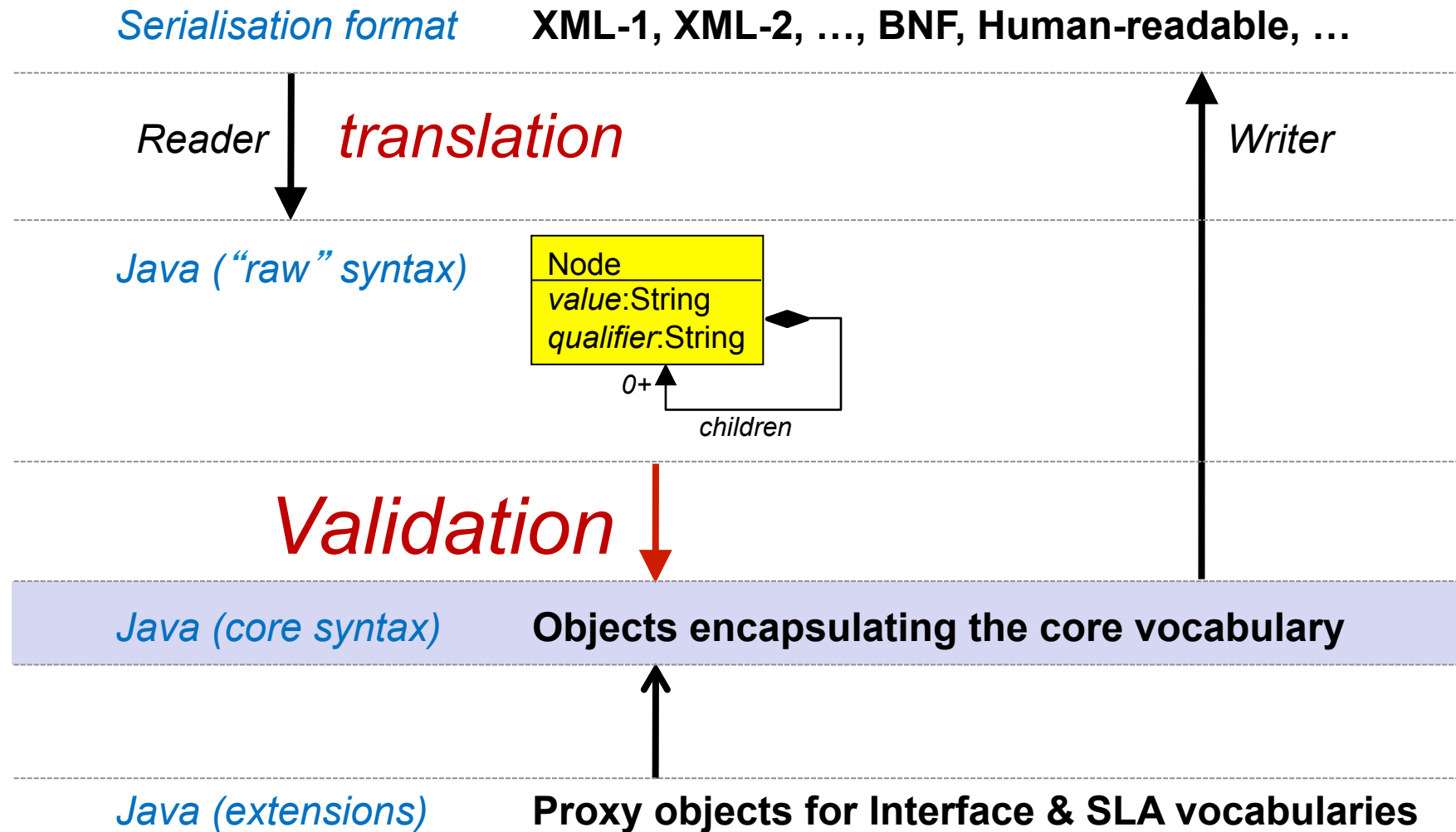
The agreement concerns the use of the following services:

Service 1, henceforth referred to as **THE_SERVICE**, is provided by **Engineering** and described by the functional interface specification **available at <http://www.eng.it/aservice.wsdl>**. **THE_SERVICE** is accessible from the following endpoints ...

... etc ...

(not all details shown)

SLA*: Java Implementation



SLA*: Summary

- Support for SLA Life-Cycle Management
 - domain-independence
 - Common constraint language
 - Minimal document specifications
- Extensibility & Customisability
 - domain-specific extensions
 - Modular pluggable vocabularies
 - Default core operators, datatypes & metric-units
 - Default QoS vocabulary (common QoS metrics)
- Language Independence
 - Abstract Syntax
 - Default concrete implemenations (BNF, XML, Java)

Thank you