1. RBSLA Constructs

<action>

The content model of the action role is defined as (Cterm | Assert | Retract | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic). The role is used in the content models of <ECA>, <Happens>, <Planned>, <Initiates> and <Terminates> (See ECA Example).

(See: eca_module.xsd)

<Any>

An Any event algebra operator. Content model is (oid?, event | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic, interval | Interval | Plex | Var).

(See: event-algebra_module.xsd)

<Aperiodic>

An Aperiodic event algebra operator. Content model is (oid?, event | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic, interval | Interval | Plex | Var).

(See: event-algebra_module.xsd)

<Attachment>

The Attachment element enables the integration of procedural attachments in RBSLA. The content model of the element is defined as ( oid?, ( Ind | Var | Cterm ) , Ind ). The elements <oid>, <Ind>, <Cterm> and <Var> are defined by RuleML. On
the eca layer of RBSLA the `<Cterm>` has been redefined so that `<Attachment>` is included. The content model of `<Cterm>` has been changed as follows:

```
(oid?, (op | Ctor | Attachment), (slot)*, (resl)?, (arg | Ind | Data | Skolem | Var | Reify | Cterm | Plex )* , (repo)?, (slot)*, (resl)?)
```

**Example:**
```
<Cterm>
  <Attachment>
    <oid> JavaPrintOut </oid>
    <Ind> System.out </Ind>
    <Ind> print </Ind>
  </Attachment>
  <Ind> Hello! </Ind>
</Cterm>
```

The `<Cterm>` redefinition enables nesting.

**Example:**
```
<Cterm>
  <Attachment>
    <Cterm>
      <Attachment>
        <Ind> java.Mobile.Car </Ind>
        <Ind> Car </Ind>
      </Attachment>
      <Ind> coupe </Ind>
    </Cterm>
    <Ind> refuel </Ind>
  </Attachment>
  <Ind> gas </Ind>
</Cterm>
```

The binding to a variable is enabled by `<Equal>` (defined by RuleML – see: [6]/0.9/xsd/modules/equality_module.xsd).

**Example:**
```
<Equal>
  <Var> Y </Var>
  <Cterm> [Attachment] </Cterm>
</Equal>
```

(See: attachment_module.xsd)
<Concurrent>

An **Concurrent** event algebra operator. Content model is (oid?, event | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic).

(See: event-algebra_module.xsd)

</Concurrent>

<condition>

The **condition** role has the following content model: (Naf | Neg | Cterm). The role is used in the content models of <ECA> element.

(See: eca_module.xsd)

<Conjunction>

An **conjunctive** event algebra operator (Conjunction). Content model is (oid?, event | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic).

(See: event-algebra_module.xsd)

<ECA>

ECA’s content model is (oid?, time?, event?, condition?, action, postcondition?, else?). The ECA element enables expressions for event condition action rules.

Example:

```
RBSLA:

<ECA>
    <time>
        <Cterm>
            <Ctor> everySec </Ctor>
            <Ind> 10 </Ind>
        </Cterm>
    </time>
    <action>
        <Cterm>
            <Ctor> updateKnowledge </Ctor>
        </Cterm>
    </action>
    <postcondition>
```

```
The **else** role has the following content model: \textbf{(Cterm | Assert | Retract)}. The role is one of the parts of the \textbf{<ECA>} element.

(See: eca_module.xsd)

**<Else>**

\textit{Else} is a part of \textbf{<Rule>}. The content model of \textit{Else} is the same as the content model of \textbf{<Naf>} which is defined by RuleML and redefined by RBSLA at the hornlog2rbsla layer. The content model is: \textbf{(oid?, (Atom | Cterm))}. The renaming of \textbf{<Naf>} is just for better understanding and easier writing of rules on the top layer. See the \textbf{<Rule>} example.

(See: if_then_else_module.xsd, naf_module.xsd, hornlog2rbsla.xsd and connective_module.xsd)

**<event>**

The \textbf{event} role has the following content model: \textbf{(Naf | Neg | Ind | Var | Cterm | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic)}. The role is one of the parts of the content models of \textbf{<ECA>}, \textbf{<Happens>}, \textbf{<Planned>}, \textbf{<Initiates>} and \textbf{<Terminates>} elements (See ECA Example).

(See: eca_module.xsd)
Content model is \( \text{Cterm} \). The role occurs under \(<\text{Happens}>\), \(<\text{Planned}>\), \(<\text{Initiates}>\) and \(<\text{Terminates}>\) by their redefinition on the deontic layer.

(See: deontic_module.xsd)

\(<\text{Fact}>\)

\textbf{Fact} has the same content model as \(<\text{Atom}>\) (defined by RuleML in atom_module.xsd). The reason to declare a separate element is to make the top layer easier to understand. Its content model is registered as follows: \( (\text{oid}?, (\text{op} | \text{Rel}), (\text{slot})^*, (\text{arg} | \text{Ind} | \text{Data} | \text{Skolem} | \text{Var} | \text{Reify})^*, (\text{slot})^*) \). \(<\text{Atom}>\) and corresponding \(<\text{Fact}>\) has the @closure attribute.

\textbf{Example:}

\begin{verbatim}
<Fact>
  <Rel> father </Rel>
  <Ind> John </Ind>
  <slot>
    <Ind> daughter </Ind>
    <Ind> Mary </Ind>
  </slot>
</Fact>
\end{verbatim}

(See: if_then_else_module.xsd and [6]/0.9/xsd/modules/atom_module.xsd)

\(<\text{fluent}>\)

The fluent role is defined with its content model \( (\text{Ind} | \text{Var} | \text{Cterm}) \) in the events_module of the RBSLA language. However, this has been redefined at the deontic layer by adding deontic norms. The top layer content model of fluent is as follows: \( (\text{Ind} | \text{Var} | \text{Cterm} | \text{norm} | \text{Oblige} | \text{Permit} | \text{Forbid} | \text{Waived}) \).

(See: events_module.xsd and deontic.xsd)

\(<\text{Forbid}>\)

\textbf{Forbid} is one of the four deontic norms which content model is: \( (((\text{Ind} | \text{Var} | \text{Cterm}), (\text{Ind} | \text{Var} | \text{Cterm}), \text{action}) \).

(See: deontic_module.xsd)
<Happens>

Happens is declared in its module with the following content model: (oid?, (event | action | Ind | Var | Cterm ), (time | Ind | Var | Cterm)). However, this is not the top level content model because it has been redefined at the deontic layer by adding <exception> and <violation>. The new content model is: (oid?, (event | action | Ind | Var | Cterm | violation | exception ), (time | Ind | Var | Cterm)).

(See: events_module.xsd and deontic.xsd)

<HoldsAt>

The primary structure of HoldsAt is declared as follows: (oid?, (fluent | Ind | Var | Cterm ), (time | Ind | Var | Cterm)). The element is redefined on the next layer – the deontic layer- and its new content model is: (oid?, (fluent | Ind | Var | Cterm | norm | Oblige | Permit | Forbid | Waived ), (time | Ind | Var | Cterm)).

(See: events_module.xsd and deontic.xsd)

<HoldsInterval>

Query construct to ask if an event interval holds in between a time interval. The primary structure of HoldsInterval is declared as follows: ( oid?, interval | Interval | Plex | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic | Cterm, interval | Interval | Plex | Var ).

(See: events_module.xsd)

<If>

If is part of <Rule> and just like <Else> and <Then> serves for better understanding and easier writing of rules on the top layer. Its content model is the same as this of the body role that is part of RuleML however RBSLA is redefining it. The structure of If is: (Atom | Conjunction | Disjunction | Assert | Retract | RetractAll). See the <Rule> example.
<Initially>

Its primary content model as declared in events_module is (oid?, (fluent | Ind | Var | Cterm)). However, this is overwritten at the deontic layer and the new structure of the element is (oid?, (fluent | Ind | Var | Cterm | norm | Oblige | Permit | Forbid | Waived)).

(See: events_module.xsd and deontic.xsd)

<Initiates>

The top level content model of Initiates is (oid?, (event | action | Ind | Var | Cterm), (fluent | Ind | Var | Cterm | norm | Oblige | Permit | Forbid | Waived), (time | Ind | Var | Cterm)). Its primary structure as implemented in events_module is (oid?, (event | action | Ind | Var | Cterm), (fluent | Ind | Var | Cterm), (time | Ind | Var | Cterm)).

(See: events_module.xsd and deontic.xsd)

<Interval>

The Interval denotes a time or event interval. Events might be complex. The content model is (oid?, event | time | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic, event | time | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic)

(See: event-algebra_module.xsd)

<interval>

The interval role of a time or event interval. The content model is (Interval | Plex | Var)
The role of the `mode` attribute is to show if a variable is intended to be an input or an output. The attribute is a restriction on string base to the following three values: “?” undefined, “+” to be input and “−” to be output. Its use is optional. The attribute is added to the attribute list of the `<Var>` element at the `hornlog2rbsla` layer.

(See: `attribute_module.xsd`)
Like `<Forbid>`, `<Oblige>` is one of the deontic norms. Its content model is `((Ind | Var | Cterm), (Ind | Var | Cterm), action)`.

Example:

```xml
<Oblige>
  <Ind> provider </Ind>
  <Ind> consumer </Ind>
  <Cterm>
    <Ctor> pay </Ctor>
    <Var> penalty </Var>
  </Cterm>
</Oblige>
```

(See: deontic_module.xsd)

**<operator>**

The `operator` role is used to denote an event algebra operator. The content model is:

```xml
(Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic | Cterm)
```

(See: event-algebra_module.xsd)

**<Disjunction>**

An `Disjunction` event algebra operator. Content model is `oid?, event | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic)

(See: event-algebra_module.xsd)

**< Overrides>**
The Overrides element should provide a structure for ranking of rules, facts and others. Its content model is ((oid | Neg | Naf | Atom | Happens | Planned | Initially | Initiates | Terminates | HoldsAt | ValueAt), (oid | Neg | Naf | Atom | Happens | Planned | Initially | Initiates | Terminates | HoldsAt | ValueAt)).

Example:

```xml
<Overrides>
  <oid>discount10</oid>
  <oid>discount5</oid>
</Overrides>
```

(See: defeasible_module.xsd)

<parameter>

Its structure is described by the following content model: (Ind | Var | Cterm).

(See: events_module.xsd)

<Periodic>

An Periodic event algebra operator. Content model is (oid?, event | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic, interval | Interval | Plex | Var).

(See: event-algebra_module.xsd)

<Permit>

Like <Forbid> and <Oblige> is Permit also one of the deontic norms. Its content model is ((Ind | Var | Cterm), (Ind | Var | Cterm), action).

(See: deontic_module.xsd)

<Planned>
The primary structure of Planned is defined by the events_module as 
\((oid?, (event | action | Ind | Var | Cterm ), (time | Ind | Var | Cterm))\). This is not the top level content model of the element because it has been redefined at the deontic layer by adding \(<\text{violation}>\) and \(<\text{exception}>\). The top level structure is as follows: 
\((oid?, (event | action | Ind | Var | Cterm ), (time | Ind | Var | Cterm))\).
(See: events_module.xsd)

\(<\text{postcondition}>\)

The postcondition role has the following content model: 
\((\text{Naf} | \text{Neg} | \text{Cterm} | \text{Assert} | \text{Retract} | \text{RetractAll})\). The role is one of the parts of the \(<\text{ECA}>\) element (ECA Example).
(See: eca_module.xsd)

\(<\text{RBSLA}>\)

RBSLA is the top element of the RBSLA language. It is defined at the top layer – RBSLA layer. RBSLA’s content model is as following: 
\((\text{Assert}^*, \text{Query}^*, \text{Protect}^*)\).
(See: root_module.xsd)

\(<\text{Repository}>\)

The repository element is a part of the optional layer contract_manager, which should provide connectivity between the RBSLA language and the contract manager application. The content model of the element is: 
\((\text{Predicates}, \text{Functions}, \text{Fact_templates}, \text{Rule_templates}, \text{Jndi_contexts}, \text{Datasources}, \text{Variable_names}, \text{Swing_editors}, \text{Blueprints})\). The elements from the content model are nonspecific for the RBSLA language, therefor they are not contained in the glossary. For detailed information about them please consider the repository_module.xsd.

\(<\text{Retract}>\)
The Retract element is defined as follows: ((oid | Atom)*, TestCase?). However, the content model of the element has been changed three times - once on the eca, once on the event_calculus and once on the defeasible layer. After the first redefining the content model of Retract is ((oid | Atom| ECA)*, TestCase?) and after the second one ((oid | Atom | ECA | Happens | Planned | Initially | Initiates | Terminates | HoldsAt | ValueAt )* , TestCase? ). The top level content model of Retract is ((oid | Atom | ECA | Happens | Planned | Initially | Initiates | Terminates | HoldsAt | ValueAt | Overrides)*, TestCase? ). (See: update_module.xsd, eca.xsd and event_calculus.xsd)

<Rule>

Rule’s content model is the following: (If, Then, Else?). <Rule> contains the optional attribute @variety. The <Rule> element should make the definition of rules constructs easier for not advanced users.

Example:

```xml
<Rule variety="strict">
  <If>
    <Conjunction>
      <Atom>
        <Rel> premium </Rel>
        <Var> customer </Var>
      </Atom>
      <Atom>
        <Rel> regular </Rel>
        <Var> product </Var>
      </Atom>
    </Conjunction>
  </If>
  <Then>
    <Atom>
      <Rel> discount </Rel>
      <Var> customer </Var>
      <Var> product </Var>
      <Ind> 5.0 percent </Ind>
    </Atom>
  </Then>
  <Else>
    <Atom>
      <Rel> discount </Rel>
      <Var> customer </Var>
      <Var> product </Var>
      <Ind> 1.0 percent </Ind>
    </Atom>
  </Else>
</Rule>
```
<Rulebase>

The content model of `<Rulebase>` is: (Fact*, Rule*, ECA*, Query*, Integrity*, Overrides*, Assert*, TestCase*, Retract*, RetractAll*). Its role is to provide structures in RBSLA syntax for saving facts and rules from the contract manager application.

(See: repository_module.xsd)

@safety

The `safety` attribute is restricted on string base to the values transaction and normal. Its role is to indicate when the function must be started as transaction and when not. The `safety` attribute is included by redefining of `<Assert>` in its attribute list on the hornlog2rbsla layer. The attribute is part of the attribute lists of `<Retract>` and `<RetractAll>`.

(See: attribute_module.xsd)

@semantics

The `semantics` attribute is restricted to string values. Its role is to provide information about different semantics. It occurs just in `<TestCase>`.

(See: testcases_module.xsd)

<Sequence>
A sequence event algebra operator. Content model is: 

(oid?, event | Ind | Var | Cterm | operator | Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic)

(See: event-algebra_module.xsd)

<Terminates>

The top level structure of Terminates is (oid?, (event | action | Ind | Var | Cterm), (fluent | Ind | Var | Cterm | norm | Oblige | Permit | Forbid | Waived), (time | Ind | Var | Cterm)). However, the element is redefined on the deontic layer that’s why the primary content model has been changed. In the events_module Terminates is implemented as follows: (oid?, (event | Ind | Var | Cterm), (fluent | Ind | Var | Cterm | interval | Interval), (time | Ind | Var | Cterm | interval | Interval)).

(See: events_module.xsd and deontic.xsd)

<Test>

Test’s content model is as follows: (oid?, Ind?, Query). The Test element is part of <TestCase>.

(See: testcases_module.xsd)

<TestCase>

The TestCase element is defined in the testcases_module with the following content model: (oid?, Test+, Atom*, Implies*, Integrity*). The usage of the @semantics attribute is optional.

(See: testcases_module.xsd)

<time>
The time role has the following content model: (Naf | Neg | Cterm | Assert | Retract | RetractAll). The role is one of the parts of the content models of <ECA>, <Happens>, <Planned>, <Initiates>, <Terminates>, <HoldsAt> and <ValueAt> elements (See ECA Example).

(See: eca_module.xsd)

<Then>

Then is like <If> and <Else> one of the parts of <Rule>. Its structure is the same as this of the head role which is part of RuleML. The content model is: (Atom | formula). Just like the other two parts of <Rule> and <Rule> self the renaming of the head role has the main aim to make understanding and writing of rules on the top level easier. See the <Rule> example.

(See: if_then_else_module.xsd and [6]/0.9/xsd/modules/connective_module.xsd)

<ValueAt>

The content model is (oid?, (parameter | Ind | Var | Cterm), (time | Ind | Var | Cterm), (Ind | Var | Cterm)) (See: events_module.xsd)

@variety

The variety attribute is restricted on string base to the values strict and defeasible. Its role is to show which <Implies> must be regard as defeasible and which as strict. The variety attribute has been included to the attribute list of <Implies> (defined by RuleML) on the defeasible layer.

(See: defeasible_module.xsd)

<violation>
Content model is \texttt{(Cterm)}. The role occurs under \texttt{<Happens>}, \texttt{<Planned>}, \texttt{<Initiates>} and \texttt{<Terminates>} by their redefining on the deontic layer.

(See: deontic_module.xsd)

\texttt{<Waived>}

Waived is the forth of the deontic norms. Its content model is \texttt{((Ind \mid Var \mid Cterm), (Ind \mid Var \mid Cterm), action)}.

(See: deontic_module.xsd)

\texttt{<Xor>}

An \texttt{Xor} event algebra operator. Content model is \texttt{(oid?, event \mid Ind \mid Var \mid Cterm \mid operator \mid Sequence \mid Disjunction \mid Xor \mid Conjunction \mid Concurrent \mid Not \mid Any \mid Aperiodic \mid Periodic)}.

(See: event-algebra_module.xsd)
2. RBSLA Extensions to the RuleML Schemas

The RBSLA language builds on the existing XML derivation language RuleML. A little glossary of the extended RuleML elements in RBSLA follows in this section.

**Glossary**

**<Assert>**

The **<Assert>** element is defined by RuleML and redefined and extended by RBSLA. The original content model of the element at the hornlog layer is: `(oid?, (formula | Atom | Implies | Equivalent | Forall)*)`. The new top level content model of `<Assert>` in RBSLA is: `(oid?, (formula | Atom | Implies | Equivalent | Forall | TestCase | ECA | Happens | Planned | Initially | Initiates | Terminates | HoldsAt | ValueAt | Overrides)*)`. `< Assert>` provides the structure for adding of new knowledge in the knowledgebase and is defined under the `<RuleML>` element in RuleML and under the `<RBSLA>` element in the RBSLA language. `<Assert>` is the element that should provide connectivity between the different contract modules.

**Example:**

**Assert in a module definition:**

```xml
<Assert>
  <oid> new knowledge </oid>
  <Atom>
    <Rel> consumption </Rel>
    <Ind> 1er BMW </Ind>
    <Ind> max 6,5l </Ind>
    <Ind> per 100 km </Ind>
  </Atom>
</Assert>
```

**Assert as reference to a module definition:**

```xml
<Assert>
  <oid> rules/module.rbsla </oid>
</Assert>
```

Thereby, the oid element contains a reference to the file where the definition of the imported module is made.

(See: [6]/0.9/xsd/modules/performative_module.xsd, hornlog2rbsa.xsd, eca.xsd, event_calculus.xsd and defeasible.xsd)
The **Cterm** element is redefined by the first layer of RBSLA. The RBSLA element **Attachment** is added and the new content model of **Cterm** is: 
\[(\text{oid}?, (\text{op} | \text{Ctor} | \text{Attachment}), (\text{slot})^*, (\text{resl})?, (\text{arg}|\text{Ind}|\text{Data}|\text{Skolem}|\text{Var}|\text{Reify}|\text{Cterm}|\text{Plex})^*, (\text{repo})?, (\text{slot})^*, (\text{resl})?)\]

(See: hornlog2rbsla.xsd)

The **Implies** element is already well-known. It is redefined by RBSLA to meet the requirements. The content model at the hornlog layer is defined as follows: 
\[(\text{oid}?, (\text{head}, \text{body}) | (\text{body}, \text{head}) | ((\text{Atom} | \text{Conjunction} | \text{Disjunction}), \text{Atom}))\]. The new top level content model in RBSLA is: 
\[(\text{oid}?, (\text{head}, \text{body}) | (\text{body}, \text{head}) | ((\text{Atom} | \text{Conjunction} | \text{Disjunction} | \text{ Assert} | \text{ Retract} | \text{ RetractAll} | \text{ Happens} | \text{ Planned} | \text{ Initially} | \text{ Initiates} | \text{ Terminates} | \text{ HoldsAt} | \text{ ValueAt}), (\text{Atom} | \text{formula} | \text{Happens} | \text{ Planned} | \text{ Initially} | \text{ Initiates} | \text{ Terminates} | \text{ HoldsAt} | \text{ ValueAt}))\]. The attributes are \@closure, \@direction, \@kind and \@variety.

(See: [6]/0.9/xsd/modules/connectiv_moule.xsd, hornlog2rbsla.xsd, event_calculus.xsd and defeasible.xsd)

The **Integrity** element is used to define constraints like as follows:

**Example:**

```
<Integrity>
  <Neg>
    <Atom>
      <Rel> cold </Rel>
      <Var> object </Var>
    </Atom>
    <Atom>
      <Rel> hot </Rel>
      <Var> object </Var>
    </Atom>
  </Neg>
</Integrity>
```
The content model at top level of RBSLA language is: (oid?, (formula | Atom | Conjunction | Disjunction | Implies | Happens | Planned | Initially | Initiates | Terminates | HoldsAt | ValueAt )+)

(See: [6]/0.9/xsd/modules/connective_module.xsd, hornlog2rbsla.xsd and event_calculus.xsd)

<Naf>

The RBSLA content model of <Naf> is: (oid?, (Atom | Cterm)).

(See: [6]/0.9/xsd/modules/naf_module.xsd and ornlog2rbsla.xsd)

<Neg>

<Neg> is the construct that provides the classical negation. Its RBSLA content model is: (Atom | Equal | Cterm)

(See: [6]/0.9/xsd/modules/neg_module.xsd and hornlog2rbsla.xsd)

<Query>

The Query element is already well known. The RBSLA language extends it by adding the constructs for event processing. The top level content model becomes (oid?, (formula | Atom | Conjunction | Disjunction | Exists | Happens | Planned | initially | Initiates | Terminates | HoldsAt | ValueAt)*).

(See: [6]/0.9/xsd/modules/performative_module.xsd and event_calculus.xsd)

<Var>

<Var> is extended at the first RBSLA layer by adding the @mode attribute.

(See: hornlog2rbsla.xsd)
Appendix A - RuleML

Figure 1: RuleML schema's structure
Appendix B - RBSLA

Figure 2: Overview of the RBSLA language