LTL Plugin3 Input Syntax

From FSL

The MOP LTL plugin syntax instantiates the generic <Logic Name>, <Logic Syntax>, and <Logic State> from the Logic Repository Syntax. It is used in conjunction with the <Logic Repository I/O> syntax, and defined using BNF (http://en.wikipedia.org/wiki/Backus-Naur_form)

```
// BNF below is extended with {p} for zero or more and [p] for zero or one repetitions of p

// The mandatory MOP logic syntax
 <LTL Name>       ::= "ltl"
 <LTL Syntax>     ::= "true" | "false"
                  | <Event Name>                               // events are atomic propositions
                  | <Not> <LTL Syntax>                         // negation
                  | <LTL Syntax> "and" <LTL Syntax>            // conjunction
                  | <LTL Syntax> "or" <LTL Syntax>             // disjunction
                  | <LTL Syntax> "xor" <LTL Syntax>            // XOR: eXclusive OR
                  | <LTL Syntax> "=>" <LTL Syntax>             // implies
                  | <LTL Syntax> "<=>" <LTL Syntax>            // if and only if
                  | "[]" <LTL Syntax>                          // always
                  | "<>" <LTL Syntax>                          // eventually
                  | "o" <LTL Syntax>                           // next
                  | <LTL Syntax> "U" <LTL Syntax>              // until
                  | <LTL Syntax> "~U" <LTL Syntax>              // dual until (|>release)|>
                  | <LTL Syntax> "R" <LTL Syntax>              // release
                  | "<*>" <LTL Syntax>                         // eventually in the past
                  | "(*)" <LTL Syntax>                         // previously
                  | <LTL Syntax> "S" <LTL Syntax>              // since
                  | <LTL Syntax> "~S" <LTL Syntax>              // dual since
 <LTL State>      ::= "violation"
```

<LTL Name>

<LTL Name> instantiates <Logic Name> from the MOP Syntax. It denotes the LTL logic using the string "ltl".

<LTL Syntax>

<LTL Syntax> instantiates <Logic Syntax> from the MOP Syntax. <LTL Syntax> is based on constants and atomic propositions with boolean operators and temporal operators. The different operators in decreasing order of precedence are [], [*], <>, <*>, o, (*), !, not > U > S > <And> > <Xor> > <Or> > <Implies> ><->.

The last eight operators from <LTL Syntax> are called *temporal* and have the following interpretation:

- [] X holds if X holds in all time points
- <> X holds if X holds in some future time point
- X U Y holds if Y holds in some future time point, and X has holds until Y holds (strict since)
- o X holds if X holds at the next time point
- [*] X holds if X holds in all past time points
- <*> X holds if X holds in some past time point

- X S Y holds if Y holds in some past time point, and since then X has held (strict since)
- (*) X holds if X holds at the previous time point

<LTL State>

<LTL State> instantiates <Logic State> from the MOP Syntax. In a LTL specification, <LTL State> can be either the special state violation. The special state violation occurs when the trace is not a prefix of any trace that satisfy the give formula.

<Not>

The LTL plugin supports various kinds of not operators

<And>

The LTL plugin supports various kinds of and operators

<Or>

The LTL plugin supports various kinds of or operators

<Xor>

The LTL plugin supports various kinds of xor operators

<Implies>

The LTL plugin supports various kinds of implies operators

Example

```
<mop>
        <Client>Web</Client>
        <Events>acquire request</Events>
        <Property>
                <Logic>ltl</Logic>
                <Formula>[](acquire => (*) request)</Formula>
        </Property>
        <Categories>violation</Categories>
</mop>
```

Retrieved from "http://fsl.cs.illinois.edu/index.php?title=LTL_Plugin3_Input_Syntax&oldid=15824"

- This page was last modified on 3 June 2012, at 00:37.
- This page has been accessed 401 times.