

# Quantitative Timed Pattern Matching Developers Manual

## 0.1.0

Generated by Doxygen 1.9.1



<b>1 Overview</b>	<b>1</b>
<b>2 Todo List</b>	<b>3</b>
<b>3 Class Index</b>	<b>5</b>
3.1 Class List . . . . .	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 ans_trait< T > Struct Template Reference . . . . .	7
4.2 ans_trait< MaxMinSemiring< double > > Struct Reference . . . . .	7
4.3 ans_trait< MaxMinSemiring< int > > Struct Reference . . . . .	7
4.4 ans_trait< MinPlusSemiring< double > > Struct Reference . . . . .	8
4.5 ans_trait< MinPlusSemiring< int > > Struct Reference . . . . .	8
4.6 BooleanSemiring Class Reference . . . . .	8
4.7 BoostTASState< SignalVariables > Struct Template Reference . . . . .	9
4.8 BoostTATransition< ClockVariables > Struct Template Reference . . . . .	9
4.8.1 Member Data Documentation . . . . .	9
4.8.1.1 resetVars . . . . .	9
4.9 BoostZoneGraphState< SignalVariables, ClockVariables, Value > Struct Template Reference . . . . .	10
4.9.1 Member Data Documentation . . . . .	10
4.9.1.1 jumpable . . . . .	10
4.10 Constraint< ClockVariables > Struct Template Reference . . . . .	11
4.10.1 Detailed Description . . . . .	11
4.11 ConstraintMaker< ClockVariables > Class Template Reference . . . . .	11
4.12 DBM Struct Reference . . . . .	12
4.12.1 Detailed Description . . . . .	13
4.12.2 Member Function Documentation . . . . .	13
4.12.2.1 convexUnion() . . . . .	13
4.12.2.2 elapse() . . . . .	13
4.12.2.3 getNumOfVar() . . . . .	14
4.12.2.4 isSatisfiableWithoutCanonize() . . . . .	14
4.12.2.5 merge() . . . . .	14
4.12.2.6 release() . . . . .	15
4.12.2.7 tightenWithoutClose() . . . . .	15
4.13 MaxMinSemiring< Base > Class Template Reference . . . . .	15
4.14 MaxPlusSemiring< Base > Class Template Reference . . . . .	16
4.15 MinPlusSemiring< Base > Class Template Reference . . . . .	16
4.16 num_type_trait< U > Struct Template Reference . . . . .	17
4.17 num_type_trait< signed char > Struct Reference . . . . .	17
4.18 num_type_trait< unsigned char > Struct Reference . . . . .	17
4.19 QuantitativeTimedPatternMatching< SignalVariables, ClockVariables, Weight, Value > Class Template Reference . . . . .	18
4.19.1 Detailed Description . . . . .	18

4.19.2 Outline of the Automata-Based Algorithm for Quantitative Timed Pattern Matching . . . . .	18
4.19.3 On the Usage of DBM in Quantitative Timed Pattern Matching . . . . .	19
4.19.4 Member Function Documentation . . . . .	19
4.19.4.1 feed() . . . . .	19
4.20 ResetVars< ClockVariables > Struct Template Reference . . . . .	20
4.21 weight_label_writer< Graph > Struct Template Reference . . . . .	20
4.22 ZoneGraphLabelWriter< ZoneGraph, TimedAutomaton, Weight > Struct Template Reference . . .	20
<b>Bibliography</b>	<b>21</b>
<b>Index</b>	<b>23</b>

# Chapter 1

## Overview

This is the documentation for developers of QTPM. The readers are supposed to be familiar with the concepts and the algorithm in [\[3\]](#).



## Chapter 2

## Todo List

**Class** **DBM**

configure include directory for eigen





## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">ans_trait&lt; T &gt;</a>	7
<a href="#">ans_trait&lt; MaxMinSemiring&lt; double &gt; &gt;</a>	7
<a href="#">ans_trait&lt; MaxMinSemiring&lt; int &gt; &gt;</a>	7
<a href="#">ans_trait&lt; MinPlusSemiring&lt; double &gt; &gt;</a>	8
<a href="#">ans_trait&lt; MinPlusSemiring&lt; int &gt; &gt;</a>	8
<a href="#">BooleanSemiring</a>	8
<a href="#">BoostTASState&lt; SignalVariables &gt;</a>	9
<a href="#">BoostTATransition&lt; ClockVariables &gt;</a>	9
<a href="#">BoostZoneGraphState&lt; SignalVariables, ClockVariables, Value &gt;</a>	10
<a href="#">Constraint&lt; ClockVariables &gt;</a>	
A constraint in a guard of transitions	11
<a href="#">ConstraintMaker&lt; ClockVariables &gt;</a>	11
<a href="#">DBM</a>	
Implementation of a zone with <a href="#">DBM</a> <a href="#">DBM</a> For the detail of DBMs, see for example <a href="#">[1]</a>	12
<a href="#">MaxMinSemiring&lt; Base &gt;</a>	15
<a href="#">MaxPlusSemiring&lt; Base &gt;</a>	16
<a href="#">MinPlusSemiring&lt; Base &gt;</a>	16
<a href="#">num_type_trait&lt; U &gt;</a>	17
<a href="#">num_type_trait&lt; signed char &gt;</a>	17
<a href="#">num_type_trait&lt; unsigned char &gt;</a>	17
<a href="#">QuantitativeTimedPatternMatching&lt; SignalVariables, ClockVariables, Weight, Value &gt;</a>	
A class to execute quantitative timed pattern matching	18
<a href="#">ResetVars&lt; ClockVariables &gt;</a>	20
<a href="#">weight_label_writer&lt; Graph &gt;</a>	20
<a href="#">ZoneGraphLabelWriter&lt; ZoneGraph, TimedAutomaton, Weight &gt;</a>	20



## Chapter 4

# Class Documentation

### 4.1 `ans_trait< T >` Struct Template Reference

The documentation for this struct was generated from the following file:

- `test/bellman_ford_test.cc`

### 4.2 `ans_trait< MaxMinSemiring< double > >` Struct Reference

#### Static Public Attributes

- `constexpr static const double ans = 3.0`

The documentation for this struct was generated from the following file:

- `test/warshall_froid_test.cc`

### 4.3 `ans_trait< MaxMinSemiring< int > >` Struct Reference

#### Static Public Attributes

- `static const int ans = 3`

The documentation for this struct was generated from the following files:

- `test/bellman_ford_test.cc`
- `test/robustness_test.cc`

## 4.4 `ans_trait< MinPlusSemiring< double > >` Struct Reference

### Static Public Attributes

- static const int **ans** = 2
- constexpr static const double **ans** = 2.0

The documentation for this struct was generated from the following files:

- test/bellman\_ford\_test.cc
- test/warshall\_froid\_test.cc

## 4.5 `ans_trait< MinPlusSemiring< int > >` Struct Reference

### Static Public Attributes

- static const int **ans** = 40

The documentation for this struct was generated from the following file:

- test/robustness\_test.cc

## 4.6 `BooleanSemiring` Class Reference

### Public Member Functions

- **BooleanSemiring** (double value)
- **BooleanSemiring** (bool data=false)
- **BooleanSemiring operator+** (const **BooleanSemiring** &x) const
- void **operator+=** (const **BooleanSemiring** &x)
- **BooleanSemiring operator\*** (const **BooleanSemiring** &x) const
- void **operator\*=** (const **BooleanSemiring** &x)
- bool **operator!=** (const **BooleanSemiring** &x) const
- bool **operator==** (const **BooleanSemiring** &x) const
- **BooleanSemiring star** () const

### Static Public Member Functions

- static **BooleanSemiring zero** ()
- static **BooleanSemiring one** ()

### Public Attributes

- bool **data**

The documentation for this class was generated from the following file:

- src/weighted\_graph.hh

## 4.7 BoostTASState< SignalVariables > Struct Template Reference

### Public Attributes

- bool **isInit**
- bool **isMatch**
- std::vector< [Constraint](#)< SignalVariables > > **label**

The documentation for this struct was generated from the following file:

- src/timed\_automaton.hh

## 4.8 BoostTATransition< ClockVariables > Struct Template Reference

### Public Attributes

- [ResetVars](#)< ClockVariables > [resetVars](#)
- std::vector< [Constraint](#)< ClockVariables > > **guard**

### 4.8.1 Member Data Documentation

#### 4.8.1.1 resetVars

```
template<class ClockVariables >  
ResetVars<ClockVariables> BoostTATransition< ClockVariables >::resetVars
```

#### Note

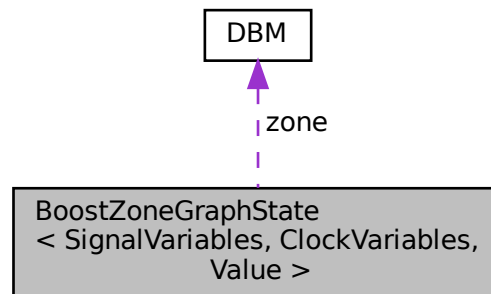
this structure is necessary because of some problem in boost graph

The documentation for this struct was generated from the following file:

- src/timed\_automaton.hh

## 4.9 BoostZoneGraphState< SignalVariables, ClockVariables, Value > Struct Template Reference

Collaboration diagram for BoostZoneGraphState< SignalVariables, ClockVariables, Value >:



### Public Attributes

- BoostTimedAutomaton< SignalVariables, ClockVariables >::vertex\_descriptor [vertex](#)  
*The corresponding state in the TA.*
- bool [jumpable](#)  
*The flag showing if one can fire a (discrete) transition. This is used to forbid having multiple jumps at the same time.*
- [DBM zone](#)  
*The corresponding zone.*
- std::vector< std::vector< Value > > [valuations](#)  
*The signal valuations observed after the latest (discrete) transition.*

### 4.9.1 Member Data Documentation

#### 4.9.1.1 jumpable

```
template<class SignalVariables , class ClockVariables , class Value >
bool BoostZoneGraphState< SignalVariables, ClockVariables, Value >::jumpable
```

The flag showing if one can fire a (discrete) transition. This is used to forbid having multiple jumps at the same time.

#### Note

In the current implementation, this flag is unnecessary because we have the following:  
[jumpable](#) == !([valuations.empty\(\)](#))

The documentation for this struct was generated from the following file:

- src/zone\_graph.hh

## 4.10 Constraint< ClockVariables > Struct Template Reference

A constraint in a guard of transitions.

```
#include <constraint.hh>
```

### Public Types

- enum class **Order** { **lt** , **le** , **ge** , **gt** }
- using **Interpretation** = std::vector< double >

### Public Member Functions

- bool **satisfy** (double d) const
- ::Order **operator()** (Interpretation val) const

### Public Attributes

- ClockVariables **x**
- Order **odr**
- int **c**

#### 4.10.1 Detailed Description

```
template<class ClockVariables>
struct Constraint< ClockVariables >
```

A constraint in a guard of transitions.

The documentation for this struct was generated from the following file:

- src/constraint.hh

## 4.11 ConstraintMaker< ClockVariables > Class Template Reference

### Public Member Functions

- **ConstraintMaker** (ClockVariables x)
- [Constraint](#)< ClockVariables > **operator**< (int c)
- [Constraint](#)< ClockVariables > **operator**<= (int c)
- [Constraint](#)< ClockVariables > **operator**> (int c)
- [Constraint](#)< ClockVariables > **operator**>= (int c)

The documentation for this class was generated from the following file:

- src/constraint.hh

## 4.12 DBM Struct Reference

Implementation of a zone with [DBM DBM](#) For the detail of DBMs, see for example [\[1\]](#).

```
#include <dbm.hh>
```

### Public Types

- using **Tuple** = std::tuple< std::vector< Bounds >, Bounds >

### Public Member Functions

- std::size\_t [getNumOfVar](#) () const  
*Returns the number of the variables represented by this zone.*
- void [cutVars](#) (std::shared\_ptr< [DBM](#) > &out, std::size\_t from, std::size\_t to) const
- std::tuple< std::vector< Bounds >, Bounds > [toTuple](#) () const  
*Return the tuple representation of the [DBM](#).*
- void [tightenWithoutClose](#) (uint8\_t x, uint8\_t y, Bounds c)  
*add the constraint  $x - y \leq (c, s)$  but does not close.*
- void [tighten](#) (uint8\_t x, uint8\_t y, Bounds c)  
*add the constraint  $x - y \leq (c, s)$*
- void [close1](#) (uint8\_t x)
- void [reset](#) (uint8\_t x)
- void [release](#) (uint8\_t x)
- void [elapse](#) ()  
*Assign the strongest post-condition of the delay.*
- void [canonize](#) ()  
*make the zone canonical*
- bool [isSatisfiableWithoutCanonize](#) () const  
*check if the zone is satisfiable*
- bool [isSatisfiable](#) ()  
*check if the zone is satisfiable*
- void [abstractize](#) ()  
*truncate the constraints compared with a constant greater than or equal to M*
- void [makeUnsat](#) ()  
*make the zone unsatisfiable*
- bool [operator==](#) ([DBM](#) z) const
- void [operator&=](#) (const [DBM](#) &z)
- bool [operator>](#) (const [DBM](#) &z) const
- bool [operator<=](#) (const [DBM](#) &z) const
- void [convexUnion](#) (const [DBM](#) &z, [DBM](#) &dest) const  
*Make the convex union of two DBMs.*
- bool [merge](#) (const [DBM](#) &z)  
*Try to merge the given [DBM](#) to this [DBM](#).*
- bool [isCanonized](#) () const

### Static Public Member Functions

- static [DBM zero](#) (int size)  
*Make the zone of size `size` such that all the values are zero.*



## Public Attributes

- `Eigen::Matrix< Bounds, Eigen::Dynamic, Eigen::Dynamic > value`  
The matrix representing the [DBM](#).
- `Bounds M`  
The threshold for the normalization.

### 4.12.1 Detailed Description

Implementation of a zone with [DBM DBM](#) For the detail of DBMs, see for example [\[1\]](#).

**Todo** configure include directory for eigen

#### Note

Internally, the variable 0 is used for the constant while externally, the actual clock variable is 0 origin, i.e., the variable 0 for the user is the variable 1 internally. So, we need increment or decrement to fill the gap.

### 4.12.2 Member Function Documentation

#### 4.12.2.1 `convexUnion()`

```
void DBM::convexUnion (
    const DBM & z,
    DBM & dest ) const [inline]
```

Make the convex union of two DBMs.

The convex union is the smallest [DBM](#) containing the given DBMs.

#### Parameters

in	z	The <a href="#">DBM</a> to take the convex union
out	dest	The <a href="#">DBM</a> to write the resulting convex union

#### Precondition

```
getNumOfVar() == z.getNumOfVar() == dest.getNumOfVar()
```

#### 4.12.2.2 `elapse()`

```
void DBM::elapse ( ) [inline]
```

Assign the strongest post-condition of the delay.

**Note**

We do not allow time elapse of duration zero

**4.12.2.3 getNumOfVar()**

```
std::size_t DBM::getNumOfVar ( ) const [inline]
```

Returns the number of the variables represented by this zone.

**Returns**

The number of the variables

**4.12.2.4 isSatisfiableWithoutCanonize()**

```
bool DBM::isSatisfiableWithoutCanonize ( ) const [inline]
```

check if the zone is satisfiable

**Precondition**

The zone is canonical

**4.12.2.5 merge()**

```
bool DBM::merge (
    const DBM & z ) [inline]
```

Try to merge the given [DBM](#) to this [DBM](#).

\*this is updated to the convex union of \*this and z if it is the union of them. This happens if and only if one of them includes the other or two zones are adjacent.

**Parameters**

<i>in</i>	<i>z</i>	The <a href="#">DBM</a> to merge
-----------	----------	----------------------------------

**Return values**

<i>true</i>	when the convex union is the union
<i>false</i>	when the convex union is not the union

#### 4.12.2.6 release()

```
void DBM::release (
    uint8_t x ) [inline]
```

##### Note

the result is not canonized

#### 4.12.2.7 tightenWithoutClose()

```
void DBM::tightenWithoutClose (
    uint8_t x,
    uint8_t y,
    Bounds c ) [inline]
```

add the constraint  $x - y \leq (c,s)$  but does not close.

##### Note

The result is not canonized

The documentation for this struct was generated from the following file:

- src/dbm.hh

## 4.13 MaxMinSemiring< Base > Class Template Reference

### Public Member Functions

- **MaxMinSemiring** (Base data=std::numeric\_limits< Base >::infinity())
- **MaxMinSemiring operator+** (const [MaxMinSemiring](#) &x) const
- void **operator+=** (const [MaxMinSemiring](#) &x)
- **MaxMinSemiring operator\*** (const [MaxMinSemiring](#) &x) const
- void **operator\*=** (const [MaxMinSemiring](#) &x)
- bool **operator!=** (const [MaxMinSemiring](#) &x) const
- bool **operator==** (const [MaxMinSemiring](#) &x) const
- **MaxMinSemiring star** () const

### Static Public Member Functions

- static [MaxMinSemiring](#) **one** ()
- static [MaxMinSemiring](#) **zero** ()

## Public Attributes

- Base **data**

The documentation for this class was generated from the following file:

- src/weighted\_graph.hh

## 4.14 MaxPlusSemiring< Base > Class Template Reference

### Public Member Functions

- **MaxPlusSemiring** (Base data=0)
- **MaxPlusSemiring** **operator+** (const **MaxPlusSemiring** &x) const
- void **operator+=** (const **MaxPlusSemiring** &x)
- **MaxPlusSemiring** **operator\*** (const **MaxPlusSemiring** &x) const
- void **operator\*=** (const **MaxPlusSemiring** &x)
- bool **operator!=** (const **MaxPlusSemiring** &x) const
- bool **operator==** (const **MaxPlusSemiring** &x) const
- **MaxPlusSemiring** **star** () const

### Static Public Member Functions

- static **MaxPlusSemiring** **zero** ()
- static **MaxPlusSemiring** **one** ()

## Public Attributes

- Base **data**

The documentation for this class was generated from the following file:

- src/weighted\_graph.hh

## 4.15 MinPlusSemiring< Base > Class Template Reference

### Public Member Functions

- **MinPlusSemiring** (Base data=0)
- **MinPlusSemiring** **operator+** (const **MinPlusSemiring** &x) const
- void **operator+=** (const **MinPlusSemiring** &x)
- **MinPlusSemiring** **operator\*** (const **MinPlusSemiring** &x) const
- void **operator\*=** (const **MinPlusSemiring** &x)
- bool **operator!=** (const **MinPlusSemiring** &x) const
- bool **operator==** (const **MinPlusSemiring** &x) const
- **MinPlusSemiring** **star** () const

### Static Public Member Functions

- static [MinPlusSemiring](#) zero ()
- static [MinPlusSemiring](#) one ()

### Public Attributes

- Base **data**

The documentation for this class was generated from the following file:

- src/weighted\_graph.hh

## 4.16 num\_type\_trait< U > Struct Template Reference

### Public Types

- using **num\_type** = U

The documentation for this struct was generated from the following file:

- src/timed\_automaton.hh

## 4.17 num\_type\_trait< signed char > Struct Reference

### Public Types

- using **num\_type** = signed int

The documentation for this struct was generated from the following file:

- src/timed\_automaton.hh

## 4.18 num\_type\_trait< unsigned char > Struct Reference

### Public Types

- using **num\_type** = unsigned int

The documentation for this struct was generated from the following file:

- src/timed\_automaton.hh

## 4.19 QuantitativeTimedPatternMatching< SignalVariables, ClockVariables, Weight, Value > Class Template Reference

A class to execute quantitative timed pattern matching.

```
#include <quantitative_timed_pattern_matching.hh>
```

### Public Types

- using **ResultMatrix** = std::array< Bounds, 6 >

### Public Member Functions

- **QuantitativeTimedPatternMatching** (const TimedAutomaton &TA, const std::vector< TState > &initStates, const std::function< Weight(const std::vector< Constraint< ClockVariables >> &, const std::vector< Value >> &)> &cost, const bool ignoreZero=false)
- void **feed** (const std::vector< Value > &valuation, const double duration)  
*feed one valuation with dwell time*
- void **getResult** (boost::unordered\_map< ResultMatrix, Weight > &v) const
- boost::unordered\_map< ResultMatrix, Weight > &**getResultRef** ()

#### 4.19.1 Detailed Description

```
template<class SignalVariables, class ClockVariables, class Weight, class Value>
class QuantitativeTimedPatternMatching< SignalVariables, ClockVariables, Weight, Value >
```

A class to execute quantitative timed pattern matching.

#### Note

This class works in an incremental way.

#### 4.19.2 Outline of the Automata-Based Algorithm for Quantitative Timed Pattern Matching

The following shows the outline of the algorithm for quantitative timed pattern matching:

1. A piece  $(u_i, \tau_i)$  of the monitored piecewise constant signal is given to **feed**.
2. The configurations corresponding to the matching begging from the current piece are added to the "pool" of the current configurations.
3. The zone graph of duration  $\tau_i$  with values  $u_i$  is constructed.
4. For each node of the zone graph, we compute the shortest distance to it by the generalized Bellman-Ford algorithm [2].
5. By forcing the dwell time, we construct the configuration just after the current piece.
6. For the configurations reaching accepting states, we put the resulting matching to this->result.

We use the zone graph for generalized reachability analysis since the transition and the switching of the signal values are asynchronous.

### 4.19.3 On the Usage of DBM in Quantitative Timed Pattern Matching

The usage of each element in the [DBM](#) is as follows.

- 0:  $x_0 == 0$ 
  - i.e., the special element showing the constant 0.
- 1-N:  $x$  (usual variables)
- N+1: the duration from the actual start
  - i.e.,  $\text{NOW} - t$ , where  $t$  is the beginning of the matching
- N+2: the dwell time
  - i.e.,  $\text{NOW} - \tau_i$
  - should be released at first
  - THIS SHOULD NOT RESET in zone construction

### 4.19.4 Member Function Documentation

#### 4.19.4.1 feed()

```
template<class SignalVariables , class ClockVariables , class Weight , class Value >
void QuantitativeTimedPatternMatching< SignalVariables, ClockVariables, Weight, Value >::feed
(
    const std::vector< Value > & valuation,
    const double duration ) [inline]
```

feed one valuation with dwell time

In this function, a piece of the entire piecewise constant function is fed and the matching ending in this piece is added to this->result. See [Outline of the Automata-Based Algorithm for Quantitative Timed Pattern Matching](#) for the outline of the algorithm.

#### Parameters

in	<i>valuation</i>	The new signal valuation
in	<i>duration</i>	The duration of the given signal valuation

#### Note

It is not a problem to give the same valuation consecutively.

The documentation for this class was generated from the following file:

- src/quantitative\_timed\_pattern\_matching.hh

## 4.20 ResetVars< ClockVariables > Struct Template Reference

### Public Attributes

- `std::vector< ClockVariables > resetVars`

The documentation for this struct was generated from the following file:

- `src/timed_automaton.hh`

## 4.21 weight\_label\_writer< Graph > Struct Template Reference

### Public Member Functions

- **weight\_label\_writer** (const Graph &g)
- `template<class Edge >`  
`void operator() (std::ostream &out, const Edge &edge) const`

The documentation for this struct was generated from the following file:

- `src/zone_graph.hh`

## 4.22 ZoneGraphLabelWriter< ZoneGraph, TimedAutomaton, Weight > Struct Template Reference

### Public Member Functions

- **ZoneGraphLabelWriter** (const ZoneGraph &ZG, const TimedAutomaton &TA, const std::unordered\_map< typename ZoneGraph::vertex\_descriptor, Weight > &distance)
- `template<class Vertex >`  
`void operator() (std::ostream &out, const Vertex &vertex) const`

The documentation for this struct was generated from the following file:

- `src/zone_graph.hh`



# Bibliography

- [1] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets, Advances in Petri Nets [This tutorial volume originates from the 4th Advanced Course on Petri Nets, ACPN 2003, held in Eichstätt, Germany in September 2003. In addition to lectures given at ACPN 2003, additional chapters have been commissioned]*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003. [5](#), [12](#), [13](#)
- [2] Mehryar Mohri. *Weighted Automata Algorithms*, pages 213–254. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. [18](#)
- [3] Masaki Waga. Online quantitative timed pattern matching with semiring-valued weighted automata. In Étienne André and Mariëlle Stoelinga, editors, *Formal Modeling and Analysis of Timed Systems - 17th International Conference, FORMATS 2019, Amsterdam, The Netherlands, August 27-29, 2019, Proceedings*, volume 11750 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2019. [1](#)



# Index

`ans_trait< MaxMinSemiring< double > >`, [7](#)  
`ans_trait< MaxMinSemiring< int > >`, [7](#)  
`ans_trait< MinPlusSemiring< double > >`, [8](#)  
`ans_trait< MinPlusSemiring< int > >`, [8](#)  
`ans_trait< T >`, [7](#)

`BooleanSemiring`, [8](#)  
`BoostTASState< SignalVariables >`, [9](#)  
`BoostTATransition< ClockVariables >`, [9](#)  
    `resetVars`, [9](#)  
`BoostZoneGraphState< SignalVariables, ClockVariables, Value >`, [10](#)  
    `jumpable`, [10](#)

`Constraint< ClockVariables >`, [11](#)  
`ConstraintMaker< ClockVariables >`, [11](#)  
`convexUnion`  
    `DBM`, [13](#)

`DBM`, [12](#)  
    `convexUnion`, [13](#)  
    `elapse`, [13](#)  
    `getNumOfVar`, [14](#)  
    `isSatisfiableWithoutCanonize`, [14](#)  
    `merge`, [14](#)  
    `release`, [15](#)  
    `tightenWithoutClose`, [15](#)

`elapse`  
    `DBM`, [13](#)

`feed`  
    `QuantitativeTimedPatternMatching< SignalVariables, ClockVariables, Weight, Value >`, [19](#)

`getNumOfVar`  
    `DBM`, [14](#)

`isSatisfiableWithoutCanonize`  
    `DBM`, [14](#)

`jumpable`  
    `BoostZoneGraphState< SignalVariables, ClockVariables, Value >`, [10](#)

`MaxMinSemiring< Base >`, [15](#)  
`MaxPlusSemiring< Base >`, [16](#)  
`merge`  
    `DBM`, [14](#)  
`MinPlusSemiring< Base >`, [16](#)

`num_type_trait< signed char >`, [17](#)  
`num_type_trait< U >`, [17](#)  
`num_type_trait< unsigned char >`, [17](#)

`QuantitativeTimedPatternMatching< SignalVariables, ClockVariables, Weight, Value >`, [18](#)  
    `feed`, [19](#)

`release`  
    `DBM`, [15](#)  
`resetVars`  
    `BoostTATransition< ClockVariables >`, [9](#)  
`ResetVars< ClockVariables >`, [20](#)

`tightenWithoutClose`  
    `DBM`, [15](#)

`weight_label_writer< Graph >`, [20](#)

`ZoneGraphLabelWriter< ZoneGraph, TimedAutomaton, Weight >`, [20](#)