

# Namespace Aplib.Core

## Classes

### [Combinators](#)

Convenience class containing static methods for creating goal structures and tactics.

### [LiftingExtensionMethods](#)

Contains extension methods for lifting BDI cycle components into higher-order components.

### [Metadata](#)

Data structure to store information about a component which may be useful for debugging or logging.

## Interfaces

### [ICompletable](#)

Defines an object that can be completed.

### [IDocumented](#)

Represents an object that contains general information on an instance, such as [IMetadata](#).

### [IMetadata](#)

A collection of generic metadata for unique instances which should help visualise the instance with human-readable information.

## Enums

### [CompletionStatus](#)

Represents the state of a completable object.

# Class Combinators

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

Convenience class containing static methods for creating goal structures and tactics.

```
public static class Combinators
```

## Inheritance

[object](#)  ← Combinators

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Methods

**FirstOf<TBeliefSet>(params  
IGoalStructure<TBeliefSet>[])**

Initializes a new instance of the [FirstOfGoalStructure<TBeliefSet>](#) class.

```
public static FirstOfGoalStructure<TBeliefSet> FirstOf<TBeliefSet>(params  
IGoalStructure<TBeliefSet>[] children) where TBeliefSet : IBeliefSet
```

## Parameters

**children** [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

## Returns

[FirstOfGoalStructure](#)<TBeliefSet>

## Type Parameters

TBeliefSet

## FirstOf<TBeliefSet>(IMetadata, params IGoalStructure<TBeliefSet>[])

Initializes a new instance of the [FirstOfGoalStructure<TBeliefSet>](#) class.

```
public static FirstOfGoalStructure<TBeliefSet> FirstOf<TBeliefSet>(IMetadata metadata, params IGoalStructure<TBeliefSet>[] children) where TBeliefSet : IBeliefSet
```

### Parameters

metadata [IMetadata](#)

Metadata about this GoalStructure, used to quickly display the goal in several contexts.

children [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

### Returns

[FirstOfGoalStructure](#)<TBeliefSet>

### Type Parameters

TBeliefSet

## FirstOf<TBeliefSet>(IMetadata, params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public static FirstOfTactic<TBeliefSet> FirstOf<TBeliefSet>(IMetadata metadata, params ITactic<TBeliefSet>[] subTactics) where TBeliefSet : IBeliefSet
```

## Parameters

**metadata** [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

**subTactics** [ITactic](#)<TBeliefSet>[]

## Returns

[FirstOfTactic](#)<TBeliefSet>

## Type Parameters

**TBeliefSet**

# FirstOf<TBeliefSet>(IMetadata, Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public static FirstOfTactic<TBeliefSet> FirstOf<TBeliefSet>(IMetadata metadata,
Predicate<TBeliefSet> guard, params ITactic<TBeliefSet>[] subTactics) where
TBeliefSet : IBeliefSet
```

## Parameters

**metadata** [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

**guard** [Predicate](#)<TBeliefSet>

The guard condition.

**subTactics** [ITactic](#)<TBeliefSet>[]

## Returns

[FirstOfTactic](#)<TBeliefSet>

## Type Parameters

TBeliefSet

### FirstOf<TBeliefSet>(params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public static FirstOfTactic<TBeliefSet> FirstOf<TBeliefSet>(params  
ITactic<TBeliefSet>[] subTactics) where TBeliefSet : IBeliefSet
```

## Parameters

subTactics [ITactic](#)<TBeliefSet>[]

## Returns

[FirstOfTactic](#)<TBeliefSet>

## Type Parameters

TBeliefSet

### FirstOf<TBeliefSet>(Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public static FirstOfTactic<TBeliefSet> FirstOf<TBeliefSet>(Predicate<TBeliefSet>  
guard, params ITactic<TBeliefSet>[] subTactics) where TBeliefSet : IBeliefSet
```

## Parameters

guard [Predicate](#)<TBeliefSet>

The guard condition.

subTactics [ITactic](#)<TBeliefSet>[]

Returns

[FirstOfTactic](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Primitive<TBeliefSet>(IGoal<TBeliefSet>)

Initializes a new instance of the [PrimitiveGoalStructure<TBeliefSet>](#) class.

```
public static PrimitiveGoalStructure<TBeliefSet> Primitive<TBeliefSet>  
(IGoal<TBeliefSet> goal) where TBeliefSet : IBeliefSet
```

Parameters

goal [IGoal](#)<TBeliefSet>

The goal to fulfill.

Returns

[PrimitiveGoalStructure](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Primitive<TBeliefSet>(IMetadata, IGoal<TBeliefSet>)

Initializes a new instance of the [PrimitiveGoalStructure<TBeliefSet>](#) class.

```
public static PrimitiveGoalStructure<TBeliefSet> Primitive<TBeliefSet>(IMetadata  
metadata, IGoal<TBeliefSet> goal) where TBeliefSet : IBeliefSet
```

## Parameters

**metadata** [IMetadata](#)

Metadata about this GoalStructure, used to quickly display the goal in several contexts.

**goal** [IGoal](#)<TBeliefSet>

The goal to fulfill.

## Returns

[PrimitiveGoalStructure](#)<TBeliefSet>

## Type Parameters

**TBeliefSet**

# Primitive<TBeliefSet>(IMetadata, IAction<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>(IMetadata metadata,
    IAction<TBeliefSet> action) where TBeliefSet : IBeliefSet
```

## Parameters

**metadata** [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

**action** [IAction](#)<TBeliefSet>

The action of the primitive tactic.

## Returns

[PrimitiveTactic](#)<TBeliefSet>

## Type Parameters

TBeliefSet

## Primitive<TBeliefSet>(IMetadata, IAction<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>(IMetadata  
metadata, IAction<TBeliefSet> action, Predicate<TBeliefSet> guard) where TBeliefSet  
: IBeliefSet
```

### Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

action [IAction](#)<TBeliefSet>

The action of the primitive tactic.

guard [Predicate](#)<TBeliefSet>

The guard of the primitive tactic.

### Returns

[PrimitiveTactic](#)<TBeliefSet>

### Type Parameters

TBeliefSet

## Primitive<TBeliefSet>(IMetadata, IQueryable<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.



```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>(IMetadata metadata,
IQueryable<TBeliefSet> query) where TBeliefSet : IBeliefSet
```

## Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

query [IQueryable](#)<TBeliefSet>

## Returns

[PrimitiveTactic](#)<TBeliefSet>

## Type Parameters

TBeliefSet

# Primitive<TBeliefSet>(IMetadata, IQueryable<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>(IMetadata metadata,
IQueryable<TBeliefSet> query, Predicate<TBeliefSet> guard) where TBeliefSet
: IBeliefSet
```

## Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

query [IQueryable](#)<TBeliefSet>

guard [Predicate](#)<TBeliefSet>

The guard of the primitive tactic.

Returns

[PrimitiveTactic](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Primitive<TBeliefSet>(IAction<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>(IAction<TBeliefSet>  
action) where TBeliefSet : IBeliefSet
```

Parameters

action [IAction](#)<TBeliefSet>

The action of the primitive tactic.

Returns

[PrimitiveTactic](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Primitive<TBeliefSet>(IAction<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>(IAction<TBeliefSet>  
action, Predicate<TBeliefSet> guard) where TBeliefSet : IBeliefSet
```

## Parameters

**action** [IAction](#)<TBeliefSet>

The action of the primitive tactic.

**guard** [Predicate](#)<TBeliefSet>

The guard of the primitive tactic.

## Returns

[PrimitiveTactic](#)<TBeliefSet>

## Type Parameters

**TBeliefSet**

# Primitive<TBeliefSet>(IQueryable<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>  
(IQueryable<TBeliefSet> query) where TBeliefSet : IBeliefSet
```

## Parameters

**query** [IQueryable](#)<TBeliefSet>

## Returns

[PrimitiveTactic](#)<TBeliefSet>

## Type Parameters

**TBeliefSet**

## Primitive<TBeliefSet>(IQueryable<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public static PrimitiveTactic<TBeliefSet> Primitive<TBeliefSet>  
(IQueryable<TBeliefSet> query, Predicate<TBeliefSet> guard) where TBeliefSet  
: IBeliefSet
```

### Parameters

**query** [IQueryable](#)<TBeliefSet>

**guard** [Predicate](#)<TBeliefSet>

The guard of the primitive tactic.

### Returns

[PrimitiveTactic](#)<TBeliefSet>

### Type Parameters

**TBeliefSet**

## Random<TBeliefSet>(IMetadata, params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public static RandomTactic<TBeliefSet> Random<TBeliefSet>(IMetadata metadata, params  
ITactic<TBeliefSet>[] subTactics) where TBeliefSet : IBeliefSet
```

### Parameters

**metadata** [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

subTactics [ITactic](#)<TBeliefSet>[]

Returns

[RandomTactic](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Random<TBeliefSet>(IMetadata, Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public static RandomTactic<TBeliefSet> Random<TBeliefSet>(IMetadata metadata,  
    Predicate<TBeliefSet> guard, params ITactic<TBeliefSet>[] subTactics) where  
    TBeliefSet : IBeliefSet
```

Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

guard [Predicate](#)  <TBeliefSet>

The guard condition.

subTactics [ITactic](#)<TBeliefSet>[]

Returns

[RandomTactic](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Random<TBeliefSet>(params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public static RandomTactic<TBeliefSet> Random<TBeliefSet>(params ITactic<TBeliefSet>
[] subTactics) where TBeliefSet : IBeliefSet
```

### Parameters

subTactics [ITactic](#)<TBeliefSet>[]

### Returns

[RandomTactic](#)<TBeliefSet>

### Type Parameters

TBeliefSet

## Random<TBeliefSet>(Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public static RandomTactic<TBeliefSet> Random<TBeliefSet>(Predicate<TBeliefSet>
guard, params ITactic<TBeliefSet>[] subTactics) where TBeliefSet : IBeliefSet
```

### Parameters

guard [Predicate](#)<TBeliefSet>

The guard condition.

subTactics [ITactic](#)<TBeliefSet>[]

### Returns

[RandomTactic](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Repeat<TBeliefSet>(IGoalStructure<TBeliefSet>)

```
public static RepeatGoalStructure<TBeliefSet> Repeat<TBeliefSet>  
(IGoalStructure<TBeliefSet> goalStructure) where TBeliefSet : IBeliefSet
```

Parameters

goalStructure [IGoalStructure](#)<TBeliefSet>

Returns

[RepeatGoalStructure](#)<TBeliefSet>

Type Parameters

TBeliefSet

## Repeat<TBeliefSet>(IMetadata, IGoalStructure<TBeliefSet>)

```
public static RepeatGoalStructure<TBeliefSet> Repeat<TBeliefSet>(IMetadata metadata,  
IGoalStructure<TBeliefSet> goalStructure) where TBeliefSet : IBeliefSet
```

Parameters

metadata [IMetadata](#)

goalStructure [IGoalStructure](#)<TBeliefSet>

Returns

[RepeatGoalStructure](#)<TBeliefSet>

## Type Parameters

TBeliefSet

## Seq<TBeliefSet>(params IGoalStructure<TBeliefSet>[])

Initializes a new instance of the [SequentialGoalStructure<TBeliefSet>](#) class.

```
public static SequentialGoalStructure<TBeliefSet> Seq<TBeliefSet>(params  
IGoalStructure<TBeliefSet>[] children) where TBeliefSet : IBeliefSet
```

## Parameters

children [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

## Returns

[SequentialGoalStructure](#)<TBeliefSet>

## Type Parameters

TBeliefSet

## Seq<TBeliefSet>(IMetadata, params IGoalStructure<TBeliefSet>[])

Initializes a new instance of the [SequentialGoalStructure<TBeliefSet>](#) class.

```
public static SequentialGoalStructure<TBeliefSet> Seq<TBeliefSet>(IMetadata  
metadata, params IGoalStructure<TBeliefSet>[] children) where TBeliefSet  
: IBeliefSet
```

## Parameters

metadata [IMetadata](#)



Metadata about this GoalStructure, used to quickly display the goal in several contexts.

**children** [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

Returns

[SequentialGoalStructure](#)<TBeliefSet>

Type Parameters

**TBeliefSet**

# Enum CompletionStatus

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

Represents the state of a completable object.

```
public enum CompletionStatus
```

## Fields

**Failure** = 2

Represents the status of a completable object that has failed to complete.

**Success** = 1

Represents the status of a completable object that has been successfully completed.

**Unfinished** = 0

Represents the status of a completable object that is not yet completed.

# Interface ICompletable

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

Defines an object that can be completed.

```
public interface ICompletable
```

## Properties

### Status

Gets the completion status of the object.

```
CompletionStatus Status { get; }
```

### Property Value

[CompletionStatus](#)

# Interface IDocumented

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

Represents an object that contains general information on an instance, such as [IMetadata](#).

```
public interface IDocumented
```

## Properties

### Metadata

Gets the metadata of the instance.

```
IMetadata Metadata { get; }
```

### Property Value

[IMetadata](#)

# Interface IMetadata

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

A collection of generic metadata for unique instances which should help visualise the instance with human-readable information.

```
public interface IMetadata
```

## Properties

### Description

Gets the description used to describe the instance.

```
string? Description { get; }
```

### Property Value

[string](#)

### Id

Gets the unique identifier of the instance.

```
Guid Id { get; }
```

### Property Value

[Guid](#)

### Name

Gets the name used to display the instance.

```
string? Name { get; }
```

Property Value

[string](#)

# Class LiftingExtensionMethods

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

Contains extension methods for lifting BDI cycle components into higher-order components.

```
public static class LiftingExtensionMethods
```

## Inheritance

[object](#)  ← LiftingExtensionMethods

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Methods

### Lift<TBeliefSet>(IGoalStructure<TBeliefSet>)

Wraps a goal structure into a desire set.

```
public static DesireSet<TBeliefSet> Lift<TBeliefSet>(this IGoalStructure<TBeliefSet>  
goalStructure) where TBeliefSet : IBeliefSet
```

## Parameters

**goalStructure** [IGoalStructure](#)<TBeliefSet>

The goal structure which on its own can function as a desire set. Meaning, the desire set consists of just a single goal structure.

## Returns

[DesireSet](#)<TBeliefSet>

A desire set.

## Type Parameters

**TBeliefSet**

## Lift<TBeliefSet>(IGoalStructure<TBeliefSet>, IMetadata)

Wraps a goal structure into a desire set.

```
public static DesireSet<TBeliefSet> Lift<TBeliefSet>(this IGoalStructure<TBeliefSet> goalStructure, IMetadata metadata) where TBeliefSet : IBeliefSet
```

## Parameters

**goalStructure** [IGoalStructure](#)<TBeliefSet>

The goal structure which on its own can function as a desire set. Meaning, the desire set consists of just a single goal structure.

**metadata** [IMetadata](#)

Optional metadata to be assigned to the desire set.

## Returns

[DesireSet](#)<TBeliefSet>

A desire set.

## Type Parameters

**TBeliefSet**

## Lift<TBeliefSet>(IGoal<TBeliefSet>)

Wraps a goal into a goal structure.

```
public static PrimitiveGoalStructure<TBeliefSet> Lift<TBeliefSet>(this IGoal<TBeliefSet> goal) where TBeliefSet : IBeliefSet
```



## Parameters

**goal** [IGoal](#)<TBeliefSet>

The goal which on its own can function as a goal structure. Meaning, the goal structure consists of just a single goal.

## Returns

[PrimitiveGoalStructure](#)<TBeliefSet>

A primitive goal structure.

## Type Parameters

**TBeliefSet**

# Lift<TBeliefSet>(IGoal<TBeliefSet>, IMetadata)

Wraps a goal into a goal structure.

```
public static PrimitiveGoalStructure<TBeliefSet> Lift<TBeliefSet>(this  
    IGoal<TBeliefSet> goal, IMetadata metadata) where TBeliefSet : IBeliefSet
```

## Parameters

**goal** [IGoal](#)<TBeliefSet>

The goal which on its own can function as a goal structure. Meaning, the goal structure consists of just a single goal.

**metadata** [IMetadata](#)

Optional metadata to be assigned to the goal structure.

## Returns

[PrimitiveGoalStructure](#)<TBeliefSet>

A primitive goal structure.

## Type Parameters

TBeliefSet

### Lift<TBeliefSet>(IAction<TBeliefSet>)

Wraps a normal action into a tactic.

```
public static PrimitiveTactic<TBeliefSet> Lift<TBeliefSet>(this IAction<TBeliefSet>  
action) where TBeliefSet : IBeliefSet
```

## Parameters

action [IAction](#)<TBeliefSet>

The action which on its own can function as a tactic. Meaning, the tactic consists of just a single action.

## Returns

[PrimitiveTactic](#)<TBeliefSet>

A primitive tactic, whose guard always returns true.

## Type Parameters

TBeliefSet

### Lift<TBeliefSet>(IAction<TBeliefSet>, IMetadata)

Wraps a normal action into a tactic.

```
public static PrimitiveTactic<TBeliefSet> Lift<TBeliefSet>(this IAction<TBeliefSet>  
action, IMetadata metadata) where TBeliefSet : IBeliefSet
```

## Parameters

action [IAction](#)<TBeliefSet>

The action which on its own can function as a tactic. Meaning, the tactic consists of just a single action.

**metadata** [IMetadata](#)

Optional metadata to be assigned to the tactic.

Returns

[PrimitiveTactic](#)<TBeliefSet>

A primitive tactic, whose guard always returns true.

Type Parameters

**TBeliefSet**

## Lift<TBeliefSet>(IQueryable<TBeliefSet>)

Wraps a queryable action into a tactic.

```
public static PrimitiveTactic<TBeliefSet> Lift<TBeliefSet>(this  
IQueryable<TBeliefSet> action) where TBeliefSet : IBeliefSet
```

Parameters

**action** [IQueryable](#)<TBeliefSet>

The action which on its own can function as a tactic. Meaning, the tactic consists of just a single action.

Returns

[PrimitiveTactic](#)<TBeliefSet>

A primitive tactic, whose guard always returns true.

Type Parameters

**TBeliefSet**

# Lift<TBeliefSet>(IQueryable<TBeliefSet>, IMetadata)

Wraps a queryable action into a tactic.

```
public static PrimitiveTactic<TBeliefSet> Lift<TBeliefSet>(this  
IQueryable<TBeliefSet> action, IMetadata metadata) where TBeliefSet : IBeliefSet
```

## Parameters

**action** [IQueryable](#)<TBeliefSet>

The action which on its own can function as a tactic. Meaning, the tactic consists of just a single action.

**metadata** [IMetadata](#)

Optional metadata to be assigned to the tactic.

## Returns

[PrimitiveTactic](#)<TBeliefSet>

A primitive tactic, whose guard always returns true.

## Type Parameters

**TBeliefSet**

# Class Metadata

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

Data structure to store information about a component which may be useful for debugging or logging.

```
public class Metadata : IMetadata
```








## Inheritance

[object](#)  ← Metadata

## Implements

[IMetadata](#)

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

### Metadata(string?, string?)


Store information about a BDI cycle component which may be useful for debugging or logging or general overviews.

```
public Metadata(string? name = null, string? description = null)
```

## Parameters

name [string](#) 

The name used to display the component.

description [string](#) 

The description used to describe the component.

# Properties

## Description

Gets the description used to describe the instance.

```
public string? Description { get; }
```

### Property Value

[string](#)

## Id

Gets the unique identifier of the instance.

```
public Guid Id { get; }
```

### Property Value

[Guid](#)

## Name

Gets the name used to display the instance.

```
public string? Name { get; }
```

### Property Value

[string](#)

# Namespace Aplib.Core.Agents

## Classes

### [BdiAgent<TBeliefSet>](#)

Represents an agent that performs actions based on goals and beliefs.

## Interfaces

### [IAgent](#)

Defines an agent that can play a game.

# Class BdiAgent<TBeliefSet>

Namespace: [Aplib.Core.Agents](#)

Assembly: Aplib.Core.dll

Represents an agent that performs actions based on goals and beliefs.

```
public class BdiAgent<TBeliefSet> : IAgent, ICompletable where TBeliefSet
    : IBeliefSet
```

## Type Parameters

**TBeliefSet**








### Inheritance

[object](#)  ← BdiAgent<TBeliefSet>

### Implements

[IAgent](#), [ICompletable](#)

### Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Constructors

### BdiAgent(TBeliefSet, IDesireSet<TBeliefSet>)

Initializes a new instance of the [BdiAgent<TBeliefSet>](#) class.

```
public BdiAgent(TBeliefSet beliefSet, IDesireSet<TBeliefSet> desireSet)
```

## Parameters

**beliefSet** TBeliefSet

The beliefset of the agent.



desireSet [IDesireSet](#)<TBeliefSet>

## Properties

### Status

Gets the completion status of the object.

```
public CompletionStatus Status { get; }
```

Property Value

[CompletionStatus](#)

## Methods

### Update()

Performs a single BDI cycle, in which the agent updates its beliefs, selects a concrete goal, chooses a concrete action to achieve the selected goal, and executes the chosen action.

```
public void Update()
```

### Remarks

This method will get called every frame of the game.

# Interface IAgent

Namespace: [Aplib.Core.Agents](#)

Assembly: Aplib.Core.dll

Defines an agent that can play a game.

```
public interface IAgent : ICompletable
```

## Inherited Members

[ICompletable.Status](#)

## Methods

### Update()

Updates the agent's state and goals.

```
void Update()
```

## Remarks

This method will get called every frame of the game.

# Namespace Aplib.Core.Belief.BeliefSets

## Classes

### [BeliefSet](#)

The [BeliefSet](#) class can be inherited to define a set of beliefs for an agent. All *public fields* of type [IBelief](#) that are defined in the inheriting class are automatically updated when calling [UpdateBeliefs\(\)](#).

## Interfaces

### [IBeliefSet](#)

A belief set defines beliefs for an agent.

# Class BeliefSet


Namespace: [Aplib.Core.Belief.BeliefSets](#)

Assembly: Aplib.Core.dll

The [BeliefSet](#) class can be inherited to define a set of beliefs for an agent. All *public fields* of type [IBelief](#) that are defined in the inheriting class are automatically updated when calling [UpdateBeliefs\(\)](#).

```
public abstract class BeliefSet : IBeliefSet
```








## Inheritance

[object](#)  ← BeliefSet

## Implements

[IBeliefSet](#)

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

### BeliefSet()

Initializes a new instance of the [BeliefSet](#) class, and stores all *public fields* of type [IBelief](#) (that have been defined in the inheriting class) in an array. All public [IBelief](#) fields are then automatically updated when calling [UpdateBeliefs\(\)](#).

```
protected BeliefSet()
```

## Methods

### UpdateBeliefs()

Updates all objects of type [IBelief](#) that are defined as *public fields* in the inheriting class.

```
public void UpdateBeliefs()
```

# Interface IBeliefSet

Namespace: [Aplib.Core.Belief.BeliefSets](#)

Assembly: Aplib.Core.dll

A belief set defines beliefs for an agent.

```
public interface IBeliefSet
```

## Methods

### UpdateBeliefs()

Updates all beliefs in the belief set.

```
void UpdateBeliefs()
```

# Namespace Aplib.Core.Belief.Beliefs

## Classes

### [Belief<TReference, TObservation>](#)

The [Belief<TReference, TObservation>](#) class represents the agent's belief of a single object. Some *object reference* is used to generate/update an *observation* (i.e., some piece of information of the game state as perceived by an agent).

### [ListBelief<TReference, TObservation>](#)

A convenience variant of [Belief<TReference, TObservation>](#) to track multiple references in one belief. Both the collection storing the references and the references themselves can be changed after the [ListBelief<TReference, TObservation>](#) has been created.

### [MemoryBelief<TReference, TObservation>](#)

The [MemoryBelief<TReference, TObservation>](#) class represents the agent's belief of a single object, but with additional "memory" of previous observations. Some *object reference* is used to generate/update an *observation* (i.e., some piece of information on the game state as perceived by an agent). This belief also stores a limited amount of previous observations in memory.

### [SampledMemoryBelief<TReference, TObservation>](#)

The [SampledMemoryBelief<TReference, TObservation>](#) class represents the agent's belief of a single object, but with additional "memory" of previous observations. These observations are sampled at a fixed rate. Some *object reference* is used to generate/update an *observation* (i.e., some piece of information on the game state as perceived by an agent). This belief also stores a limited amount of previous observation samples in memory. Optionally, the belief can always store the most recent observation, regardless of the sample rate.

## Interfaces

### [IBelief](#)

A belief represents/encapsulates an observation (i.e., piece of information of the game state as perceived by an agent).

## Enums

### [UpdateMode](#)

Specifies the update mode of a sampled memory belief.

# Class Belief<TReference, TObservation>

Namespace: [Aplib.Core.Belief.Beliefs](#)

Assembly: Aplib.Core.dll

The [Belief<TReference, TObservation>](#) class represents the agent's belief of a single object. Some *object reference* is used to generate/update an *observation* (i.e., some piece of information of the game state as perceived by an agent).

```
public class Belief<TReference, TObservation> : IBelief where TReference : class
```

## Type Parameters

### TReference

The type of the object reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if **TReference** is an interface.

### TObservation

The type of the observation that the belief represents.

## Inheritance

[object](#)  ← [Belief<TReference, TObservation>](#)








## Implements

[IBelief](#)

## Derived

[ListBelief<TReference, TObservation>](#), [MemoryBelief<TReference, TObservation>](#)

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Remarks

It supports implicit conversion to **TObservation**.



# Constructors

## Belief(Metadata, TReference, Func<TReference, TObservation>)

Initializes a new instance of the [Belief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated.

```
public Belief(Metadata metadata, TReference reference, Func<TReference, TObservation> getObservationFromReference)
```

### Parameters

**metadata** [Metadata](#)

Metadata about this Belief, used to quickly display the goal in several contexts.

**reference** TReference

The object reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if TReference is an interface.

**getObservationFromReference** [Func](#)<TReference, TObservation>

A function that takes an object reference and generates/updates an observation.

### Exceptions

[ArgumentException](#)

Thrown when **reference** is not a reference type.

## Belief(Metadata, TReference, Func<TReference, TObservation>, Predicate<TReference>)

Initializes a new instance of the [Belief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated.

```
public Belief(Metadata metadata, TReference reference, Func<TReference,
TObservation> getObservationFromReference, Predicate<TReference> shouldUpdate)
```

## Parameters

**metadata** [Metadata](#)

Metadata about this Belief, used to quickly display the goal in several contexts.

**reference** TReference

The object reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if TReference is an interface.

**getObservationFromReference** [Func](#) <TReference, TObservation>

A function that takes an object reference and generates/updates an observation.

**shouldUpdate** [Predicate](#) <TReference>

A condition on when the observation should be updated. Takes the object reference of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when **reference** is not a reference type.

## Belief(TReference, Func<TReference, TObservation>)

Initializes a new instance of the [Belief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated.

```
public Belief(TReference reference, Func<TReference, TObservation>
getObservationFromReference)
```

## Parameters

**reference** TReference

The object reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if `TReference` is an interface.

`getObservationFromReference` [Func](#) <TReference, TObservation>

A function that takes an object reference and generates/updates an observation.

## Exceptions

[ArgumentException](#)

Thrown when `reference` is not a reference type.

## Belief(TReference, Func<TReference, TObservation>, Predicate<TReference>)

Initializes a new instance of the [Belief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated.

```
public Belief(TReference reference, Func<TReference, TObservation>
getObservationFromReference, Predicate<TReference> shouldUpdate)
```

## Parameters

`reference` TReference

The object reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if `TReference` is an interface.

`getObservationFromReference` [Func](#) <TReference, TObservation>

A function that takes an object reference and generates/updates an observation.

`shouldUpdate` [Predicate](#) <TReference>

A condition on when the observation should be updated. Takes the object reference of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when `reference` is not a reference type.

## Fields

### `_getObservationFromReference`

A function that takes an object reference and generates/updates an observation.

```
protected readonly Func<TReference, TObservation> _getObservationFromReference
```

Field Value

[Func](#)  <TReference, TObservation>

### `_reference`

The object reference used to generate/update the observation.

```
protected readonly TReference _reference
```

Field Value

TReference

### `_shouldUpdate`

A condition on when the observation should be updated. Takes the object reference of the belief as a parameter for the predicate.

```
protected readonly Predicate<TReference> _shouldUpdate
```

Field Value

[Predicate](#)  <TReference>

# Properties

## Metadata

Gets the metadata of the Belief.

```
public Metadata Metadata { get; }
```

## Property Value

[Metadata](#)

## Observation

The observation represented by the belief (i.e., some piece of information of the game state as perceived by an agent).

```
public TObservation Observation { get; protected set; }
```

## Property Value

TObservation

## Methods

### UpdateBelief()

Generates/updates the observation if the shouldUpdate condition is satisfied. The observation is then updated by calling the getObservationFromReference function.

```
public virtual void UpdateBelief()
```

### UpdateObservation()

Generates/updates the observation.

```
protected void UpdateObservation()
```

## Operators

### implicit operator TObservation(Belief<TReference, TObservation>)

Implicit conversion operator to allow a [Belief<TReference, TObservation>](#) object to be used where a `TObservation` is expected.

```
public static implicit operator TObservation(Belief<TReference,  
TObservation> belief)
```

## Parameters

`belief` [Belief<TReference, TObservation>](#)

The [Belief<TReference, TObservation>](#) object to convert.

## Returns

`TObservation`

# Interface IBelief

Namespace: [Aplib.Core.Belief.Beliefs](#)

Assembly: Aplib.Core.dll

A belief represents/encapsulates an observation (i.e., piece of information of the game state as perceived by an agent).

```
public interface IBelief
```

## Methods

### UpdateBelief()

Updates the belief based on information of the game state.

```
void UpdateBelief()
```

# Class ListBelief<TReference, TObservation>

Namespace: [Aplib.Core.Belief.Beliefs](#)

Assembly: Aplib.Core.dll

A convenience variant of [Belief<TReference, TObservation>](#) to track multiple references in one belief. Both the collection storing the references and the references themselves can be changed after the [ListBelief<TReference, TObservation>](#) has been created.

```
public class ListBelief<TReference, TObservation> : Belief<IEnumerable<TReference>,
List<TObservation>>, IBelief
```

## Type Parameters





### TReference

The type of the object references used to generate/update the observation.

### TObservation

The type of the observations that the belief represents.

## Inheritance

[object](#)  ← [Belief](#)  [IEnumerable](#)  <TReference>, [List](#)  <TObservation>> ←  
ListBelief<TReference, TObservation>

## Implements

[IBelief](#)

## Inherited Members

[Belief<IEnumerable<TReference>, List<TObservation>>.\\_reference](#) ,  
[Belief<IEnumerable<TReference>, List<TObservation>>.\\_getObservationFromReference](#) ,  
[Belief<IEnumerable<TReference>, List<TObservation>>.\\_shouldUpdate](#) ,  
[Belief<IEnumerable<TReference>, List<TObservation>>.Metadata](#) ,  
[Belief<IEnumerable<TReference>, List<TObservation>>.Observation](#) ,  
[Belief<IEnumerable<TReference>, List<TObservation>>.UpdateBelief\(\)](#) ,  
[Belief<IEnumerable<TReference>, List<TObservation>>.UpdateObservation\(\)](#) ,  
[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,



[object.GetType\(\)](#), [object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

## Remarks

A [ListBelief<TReference, TObservation>](#) can be implicitly converted to a [List<T>](#), which will have the same size as the reference collection the last time that [UpdateBelief\(\)](#) was called, and contain the observation results for each element in the collection.

## Constructors

### ListBelief(Metadata, IEnumerable<TReference>, Func<TReference, TObservation>)

Initializes a new instance of the [ListBelief<TReference, TObservation>](#) class from an object reference collection, a function to generate an observation from an object reference, and optionally an update guard.

```
public ListBelief(Metadata metadata, IEnumerable<TReference> references,
    Func<TReference, TObservation> getObservationFromReference)
```

## Parameters

**metadata** [Metadata](#)

Metadata about this Belief, used to quickly display the goal in several contexts.

**references** [IEnumerable](#)<TReference>

The collection of reference objects. The underlying type implementing [IEnumerable<T>](#) must be a reference type, note that this is not enforced by C#.

**getObservationFromReference** [Func](#)<TReference, TObservation>

A function that takes an object reference and generates an observation.

## Exceptions

[ArgumentException](#)

Thrown when **references** is not a reference type.

# ListBelief(Metadata, IEnumerable<TReference>, Func<TReference, TObservation>, Predicate<IEnumerable<TReference>>)

Initializes a new instance of the [ListBelief<TReference, TObservation>](#) class from an object reference collection, a function to generate an observation from an object reference, and optionally an update guard.

```
public ListBelief(Metadata metadata, IEnumerable<TReference> references,
    Func<TReference, TObservation> getObservationFromReference,
    Predicate<IEnumerable<TReference>> shouldUpdate)
```

## Parameters

**metadata** [Metadata](#)

Metadata about this Belief, used to quickly display the goal in several contexts.

**references** [IEnumerable](#) <TReference>

The collection of reference objects. The underlying type implementing [IEnumerable<T>](#) *must* be a reference type, note that this is not enforced by C#.

**getObservationFromReference** [Func](#) <TReference, TObservation>

A function that takes an object reference and generates an observation.

**shouldUpdate** [Predicate](#) <[IEnumerable](#) <TReference>>

A condition on when the observation should be updated. Takes the object references enumerable of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when **references** is not a reference type.

# ListBelief(IEnumerable<TReference>, Func<TReference, TObservation>)

Initializes a new instance of the [ListBelief<TReference, TObservation>](#) class from an object reference collection, a function to generate an observation from an object reference, and optionally an update guard.

```
public ListBelief(IEnumerable<TReference> references, Func<TReference, TObservation> getObservationFromReference)
```

## Parameters

**references** [IEnumerable](#) <TReference>

The collection of reference objects. The underlying type implementing [IEnumerable<T>](#) *must* be a reference type, note that this is not enforced by C#.

**getObservationFromReference** [Func](#) <TReference, TObservation>

A function that takes an object reference and generates an observation.

## Exceptions

[ArgumentException](#)

Thrown when **references** is not a reference type.

## ListBelief(IEnumerable<TReference>, Func<TReference, TObservation>, Predicate<IEnumerable<TReference>>)

Initializes a new instance of the [ListBelief<TReference, TObservation>](#) class from an object reference collection, a function to generate an observation from an object reference, and optionally an update guard.

```
public ListBelief(IEnumerable<TReference> references, Func<TReference, TObservation> getObservationFromReference, Predicate<IEnumerable<TReference>> shouldUpdate)
```

## Parameters

**references** [IEnumerable](#) <TReference>

The collection of reference objects. The underlying type implementing [IEnumerable<T>](#) must be a reference type, note that this is not enforced by C#.

`getObservationFromReference` [Func](#) <TReference, TObservation>

A function that takes an object reference and generates an observation.

`shouldUpdate` [Predicate](#) <[IEnumerable](#) <TReference>>

A condition on when the observation should be updated. Takes the object references enumerable of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when `references` is not a reference type.

# Class MemoryBelief<TReference, TObservation>

Namespace: [Aplib.Core.Belief.Beliefs](#)

Assembly: Aplib.Core.dll

The [MemoryBelief<TReference, TObservation>](#) class represents the agent's belief of a single object, but with additional "memory" of previous observations. Some *object reference* is used to generate/update an *observation* (i.e., some piece of information on the game state as perceived by an agent). This belief also stores a limited amount of previous observations in memory.

```
public class MemoryBelief<TReference, TObservation> : Belief<TReference, TObservation>, IBelief where TReference : class
```

## Type Parameters

### TReference

The type of the reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if **TReference** is an interface.

### TObservation

The type of the observation the belief represents.

## Inheritance

[object](#)  [Belief](#)<TReference, TObservation>  MemoryBelief<TReference, TObservation>

## Implements

[IBelief](#)

## Derived

[SampledMemoryBelief<TReference, TObservation>](#)

## Inherited Members

[Belief<TReference, TObservation>.\\_reference](#) ,  
[Belief<TReference, TObservation>.\\_getObservationFromReference](#) ,  
[Belief<TReference, TObservation>.\\_shouldUpdate](#) ,  
[Belief<TReference, TObservation>.Metadata](#) ,

[Belief<TReference, TObservation>.Observation](#) ,  
[Belief<TReference, TObservation>.UpdateBelief\(\)](#) ,  
[Belief<TReference, TObservation>.UpdateObservation\(\)](#) , [object.Equals\(object\)](#)<sup>↗</sup> ,  
[object.Equals\(object, object\)](#)<sup>↗</sup> , [object.GetHashCode\(\)](#)<sup>↗</sup> , [object.GetType\(\)](#)<sup>↗</sup> ,  
[object.MemberwiseClone\(\)](#)<sup>↗</sup> , [object.ReferenceEquals\(object, object\)](#)<sup>↗</sup> , [object.ToString\(\)](#)<sup>↗</sup>

## Remarks

It supports implicit conversion to `TObservation`.

## Constructors

### MemoryBelief(Metadata, TReference, Func<TReference, TObservation>, int)

Initializes a new instance of the [MemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. Also initializes the memory array with a specified number of slots.

```
public MemoryBelief(Metadata metadata, TReference reference, Func<TReference,  
TObservation> getObservationFromReference, int framesToRemember)
```

## Parameters

`metadata` [Metadata](#)

Metadata about this Belief, used to quickly display the goal in several contexts.

`reference` `TReference`

The reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if `TReference` is an interface.

`getObservationFromReference` [Func](#)<sup>↗</sup> <TReference, TObservation>

A function that takes a reference and generates/updates a observation.

`framesToRemember` [int](#)<sup>↗</sup>

The number of frames to remember back.

# Exceptions

## [ArgumentException](#)

Thrown when `reference` is not a reference type.

## MemoryBelief(Metadata, TReference, Func<TReference, TObservation>, int, Predicate<TReference>)

Initializes a new instance of the [MemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. Also initializes the memory array with a specified number of slots.

```
public MemoryBelief(Metadata metadata, TReference reference, Func<TReference, TObservation> getObservationFromReference, int framesToRemember, Predicate<TReference> shouldUpdate)
```

## Parameters

`metadata` [Metadata](#)

Metadata about this Belief, used to quickly display the goal in several contexts.

`reference` TReference

The reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if TReference is an interface.

`getObservationFromReference` [Func](#) <TReference, TObservation>

A function that takes a reference and generates/updates a observation.

`framesToRemember` [int](#)

The number of frames to remember back.

`shouldUpdate` [Predicate](#) <TReference>

A condition on when the observation should be updated. Takes the object reference of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when `reference` is not a reference type.

## MemoryBelief(TReference, Func<TReference, TObservation>, int)

Initializes a new instance of the [MemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. Also initializes the memory array with a specified number of slots.

```
public MemoryBelief(TReference reference, Func<TReference, TObservation>  
getObservationFromReference, int framesToRemember)
```

## Parameters

`reference` TReference

The reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if `TReference` is an interface.

`getObservationFromReference` [Func](#)<TReference, TObservation>

A function that takes a reference and generates/updates a observation.

`framesToRemember` [int](#)

The number of frames to remember back.

## Exceptions

[ArgumentException](#)

Thrown when `reference` is not a reference type.

## MemoryBelief(TReference, Func<TReference, TObservation>, int, Predicate<TReference>)



Initializes a new instance of the [MemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. Also initializes the memory array with a specified number of slots.

```
public MemoryBelief(TReference reference, Func<TReference, TObservation>  
getObservationFromReference, int framesToRemember, Predicate<TReference>  
shouldUpdate)
```

## Parameters

**reference** TReference

The reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if TReference is an interface.

**getObservationFromReference** [Func](#) <TReference, TObservation>

A function that takes a reference and generates/updates a observation.

**framesToRemember** [int](#)

The number of frames to remember back.

**shouldUpdate** [Predicate](#) <TReference>

A condition on when the observation should be updated. Takes the object reference of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when **reference** is not a reference type.

## Fields

**\_memorizedObservations**

A "memorized" resource, from the last time the belief was updated.

```
protected readonly ExposedQueue<TObservation> _memorizedObservations
```

Field Value

[ExposedQueue](#)<TObservation>

## Methods

### GetAllMemories()

Gets all the memorized observations. The first element is the newest memory.

```
public TObservation[] GetAllMemories()
```

Returns

TObservation[]

An array of all the memorized observations.

### GetMemoryAt(int, bool)

Gets the memorized observation at a specific index. A higher index means a memory further back in time.

```
public TObservation GetMemoryAt(int index, bool clamp = false)
```

Parameters

index [int](#)

The index of the memory to get.

clamp [bool](#)

If true, the index will be clamped between 0 and the last memory index.

Returns

TObservation

The memory of the observation at the specified index.

## GetMostRecentMemory()

Gets the most recently memorized observation.

```
public TObservation GetMostRecentMemory()
```

Returns

TObservation

The most recent memory of the observation.

## UpdateBelief()

Generates/updates the observation. Also stores the previous observation in memory.

```
public override void UpdateBelief()
```

# Class SampledMemoryBelief<TReference, TObservation>

Namespace: [Aplib.Core.Belief.Beliefs](#)

Assembly: Aplib.Core.dll

The [SampledMemoryBelief<TReference, TObservation>](#) class represents the agent's belief of a single object, but with additional "memory" of previous observations. These observations are sampled at a fixed rate. Some *object reference* is used to generate/update an *observation* (i.e., some piece of information on the game state as perceived by an agent). This belief also stores a limited amount of previous observation samples in memory. Optionally, the belief can always store the most recent observation, regardless of the sample rate.

```
public class SampledMemoryBelief<TReference, TObservation> :  
    MemoryBelief<TReference, TObservation>, IBelief where TReference : class
```

## Type Parameters


### TReference

The type of the reference used to generate/update the observation. This *must* be a reference type, be aware that this is not enforced by C# if **TReference** is an interface.

### TObservation

The type of the observation the belief represents.

## Inheritance

[object](#)  ← [Belief](#)<TReference, TObservation> ←  
[MemoryBelief](#)<TReference, TObservation> ←  
SampledMemoryBelief<TReference, TObservation>

## Implements

[IBelief](#)

## Inherited Members

[MemoryBelief<TReference, TObservation>.\\_memorizedObservations](#) ,  
[MemoryBelief<TReference, TObservation>.UpdateBelief\(\)](#) ,  
[MemoryBelief<TReference, TObservation>.GetMostRecentMemory\(\)](#) ,

[MemoryBelief<TReference, TObservation>.GetMemoryAt\(int, bool\)](#) ,  
[MemoryBelief<TReference, TObservation>.GetAllMemories\(\)](#) ,  
[Belief<TReference, TObservation>.\\_reference](#) ,  
[Belief<TReference, TObservation>.\\_getObservationFromReference](#) ,  
[Belief<TReference, TObservation>.\\_shouldUpdate](#) ,  
[Belief<TReference, TObservation>.Metadata](#) ,  
[Belief<TReference, TObservation>.Observation](#) ,  
[Belief<TReference, TObservation>.UpdateBelief\(\)](#) ,  
[Belief<TReference, TObservation>.UpdateObservation\(\)](#) , [object.Equals\(object\)](#)<sup>↗</sup> ,  
[object.Equals\(object, object\)](#)<sup>↗</sup> , [object.GetHashCode\(\)](#)<sup>↗</sup> , [object.GetType\(\)](#)<sup>↗</sup> ,  
[object.MemberwiseClone\(\)](#)<sup>↗</sup> , [object.ReferenceEquals\(object, object\)](#)<sup>↗</sup> , [object.ToString\(\)](#)<sup>↗</sup>

## Remarks

It supports implicit conversion to `TObservation`.

## Constructors

### SampledMemoryBelief(Metadata, TReference, Func<TReference, TObservation>, int, UpdateMode, int)

Initializes a new instance of the [SampledMemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. This belief also stores a limited amount of previous observation samples in memory. Optionally, the belief can always store the most recent observation, regardless of the sample rate.

```
public SampledMemoryBelief(Metadata metadata, TReference reference, Func<TReference, TObservation> getObservationFromReference, int sampleInterval, UpdateMode updateMode, int framesToRemember)
```

## Parameters

`metadata` [Metadata](#)

Metadata about this goal, used to quickly display the goal in several contexts.

`reference` `TReference`

The reference used to generate/update the observation. This *must* be a reference type.

`getObservationFromReference` [Func](#) <TReference, TObservation>

A function that takes a reference and generates/updates an observation.

`sampleInterval` [int](#)

The sample interval of the memory. One observation memory (i.e., snapshot) is stored every `sampleInterval`-th cycle.

`updateMode` [UpdateMode](#)

Specifies how this sampled memory belief should be updated.

`framesToRemember` [int](#)

The number of frames to remember back.

## Exceptions

[ArgumentException](#)

Thrown when `reference` is not a reference type.

## SampledMemoryBelief(Metadata, TReference, Func<TReference, TObservation>, int, UpdateMode, int, Predicate<TReference>)

Initializes a new instance of the [SampledMemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. This belief also stores a limited amount of previous observation samples in memory. Optionally, the belief can always store the most recent observation, regardless of the sample rate.

```
public SampledMemoryBelief(Metadata metadata, TReference reference, Func<TReference, TObservation> getObservationFromReference, int sampleInterval, UpdateMode updateMode, int framesToRemember, Predicate<TReference> shouldUpdate)
```

## Parameters

`metadata` [Metadata](#)

Metadata about this goal, used to quickly display the goal in several contexts.

`reference` [TReference](#)

The reference used to generate/update the observation. This *must* be a reference type.

`getObservationFromReference` [Func](#) [<TReference, TObservation>](#)

A function that takes a reference and generates/updates an observation.

`sampleInterval` [int](#)

The sample interval of the memory. One observation memory (i.e., snapshot) is stored every `sampleInterval`-th cycle.

`updateMode` [UpdateMode](#)

Specifies how this sampled memory belief should be updated.

`framesToRemember` [int](#)

The number of frames to remember back.

`shouldUpdate` [Predicate](#) [<TReference>](#)

A condition on when the observation should be updated. Takes the object reference of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when `reference` is not a reference type.

## SampledMemoryBelief(TReference, Func<TReference, TObservation>, int, UpdateMode, int)

Initializes a new instance of the [SampledMemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. This belief also stores a limited amount of previous observation samples in memory. Optionally, the belief can always store the most recent observation, regardless of the sample rate.

```
public SampledMemoryBelief(TReference reference, Func<TReference, TObservation>
getObservationFromReference, int sampleInterval, UpdateMode updateMode,
```

`int framesToRemember)`

## Parameters

`reference` [TReference](#)

The reference used to generate/update the observation. This *must* be a reference type.

`getObservationFromReference` [Func](#) [<TReference, TObservation>](#)

A function that takes a reference and generates/updates an observation.

`sampleInterval` [int](#)

The sample interval of the memory. One observation memory (i.e., snapshot) is stored every `sampleInterval`-th cycle.

`updateMode` [UpdateMode](#)

Specifies how this sampled memory belief should be updated.

`framesToRemember` [int](#)

The number of frames to remember back.

## Exceptions

[ArgumentException](#)

Thrown when `reference` is not a reference type.

## SampledMemoryBelief(TReference, Func<TReference, TObservation>, int, UpdateMode, int, Predicate<TReference>)

Initializes a new instance of the [SampledMemoryBelief<TReference, TObservation>](#) class with an object reference, a function to generate/update the observation using the object reference, and a condition on when the observation should be updated. This belief also stores a limited amount of previous observation samples in memory. Optionally, the belief can always store the most recent observation, regardless of the sample rate.



```
public SampledMemoryBelief(TReference reference, Func<TReference, TObservation>
getObservationFromReference, int sampleInterval, UpdateMode updateMode, int
framesToRemember, Predicate<TReference> shouldUpdate)
```

## Parameters

**reference** [TReference](#)

The reference used to generate/update the observation. This *must* be a reference type.

**getObservationFromReference** [Func](#) <TReference, TObservation>

A function that takes a reference and generates/updates an observation.

**sampleInterval** [int](#)

The sample interval of the memory. One observation memory (i.e., snapshot) is stored every **sampleInterval**-th cycle.

**updateMode** [UpdateMode](#)

Specifies how this sampled memory belief should be updated.

**framesToRemember** [int](#)

The number of frames to remember back.

**shouldUpdate** [Predicate](#) <TReference>

A condition on when the observation should be updated. Takes the object reference of the belief as a parameter for the predicate.

## Exceptions

[ArgumentException](#)

Thrown when **reference** is not a reference type.

## Methods

### UpdateBelief()

Generates/updates the observation if applicable. Also stores the previous observation in memory every `sampleInterval`-th cycle.

```
public override void UpdateBelief()
```

# Enum UpdateMode

Namespace: [Aplib.Core.Belief.Beliefs](#)

Assembly: Aplib.Core.dll

Specifies the update mode of a sampled memory belief.

```
public enum UpdateMode
```

## Fields

`AlwaysUpdate = 0`

Update the observation every cycle.

`UpdateWhenSampled = 1`

Update the observation whenever a memory sample is stored.

# Namespace Aplib.Core.Collections

## Classes

### [CircularArray<T>](#)

An array that wraps around when it reaches its end. Functionally works like a queue with indexing.

### [ExposedQueue<T>](#)

A queue with all elements exposed. Functionally works like a queue with indexing. It has a MaxCount and Count. MaxCount being the maximal length of the queue, and Count being the actual number of elements in the queue.

### [OptimizedActivationStack<T>](#)

A stack that has a predefined set of items that can be *activated* (i.e., pushed on top of the stack). When an item that is already on the stack is activated, it is *reactivated* (i.e., moved to the top of the stack).

### [OptimizedActivationStack<T>.StackItem](#)

Represents (i.e., encapsulates) an item on the activation stack.

# Class CircularArray<T>

Namespace: [Aplib.Core.Collections](#)

Assembly: Aplib.Core.dll

An array that wraps around when it reaches its end. Functionally works like a queue with indexing.

```
public class CircularArray<T>
```








## Type Parameters

**T**

### Inheritance

[object](#)  ← CircularArray<T>

### Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Constructors

### CircularArray(int)

Initializes a new instance of the [CircularArray<T>](#) class.

```
public CircularArray(int size)
```

### Parameters

size [int](#) 

The size of the array.

### CircularArray(T[])

Initializes a new instance of the [CircularArray<T>](#) class.

```
public CircularArray(T[] array)
```

## Parameters

**array** T[]

An array to use as the circular array.

## Properties

### this[int]

Gets the element at the specified index.

```
public T this[int index] { get; set; }
```

## Parameters

**index** [int](#)

The index of the element to get.

## Property Value

T

The element at the specified index.

## Length

The length of the array.

```
public int Length { get; }
```

## Property Value

## Methods

### GetFirst()

Gets the first element of the array.

```
public T GetFirst()
```

Returns

T

The last element of the array

### GetHead()

Gets the element at the head of the array.

```
public T GetHead()
```

Returns

T

The element at the head of the array

### Put(T)

Puts an element at the start of the array.

```
public void Put(T value)
```

Parameters

value T

The element to add to the array

## ToArray(int, int)

Converts the circular array to an array. The head should be the last element of the array. Copies from start to end inclusive.

```
public T[] ToArray(int start = 0, int end = -1)
```

### Parameters

start [int](#)

The start index of the range to copy.

end [int](#)

The end index of the range to copy.

### Returns

T[]

The circular array as a normal array



# Class ExposedQueue<T>

Namespace: [Aplib.Core.Collections](#)

Assembly: Aplib.Core.dll


A queue with all elements exposed. Functionally works like a queue with indexing. It has a MaxCount and Count. MaxCount being the maximal length of the queue, and Count being the actual number of elements in the queue.

```
public class ExposedQueue<T> : ICollection<T>, IEnumerable<T>, IEnumerable
```

## Type Parameters

**T**








### Inheritance

[object](#)  ← ExposedQueue<T>

### Implements

[ICollection](#)  <T>, [IEnumerable](#)  <T>, [IEnumerable](#) 

### Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Remarks

When adding an element to a full queue, all other elements are shifted one place like so:  
[4, 3, 2, 1], Put(5) => [5, 4, 3, 2]

## Constructors

### ExposedQueue(int)

Initializes a new empty instance of the [ExposedQueue<T>](#) class.

```
public ExposedQueue(int size)
```

## Parameters

size [int](#)

The maximum size of the queue.

## ExposedQueue(T[])

Initializes a new instance of the [ExposedQueue<T>](#) class with an array.

```
public ExposedQueue(T[] array)
```

## Parameters

array T[]

The array to initialize the queue with.

## Remarks

The array will be copied to the queue, and the head will be set to the last element of the array. This method expects the array to be filled.

## Exceptions

[ArgumentOutOfRangeException](#)

Thrown when the max count is negative.

## ExposedQueue(T[], int)

Initializes a new instance of the [ExposedQueue<T>](#) class with an array.

```
public ExposedQueue(T[] array, int maxCount)
```

## Parameters

array T[]

The array to initialize the queue with.

`maxCount` [int](#)

The maximum count of the queue.

## Remarks

The array will be copied to the queue, and the head will be set to the last element of the array. This method expects the array to be filled.

## Exceptions

[ArgumentOutOfRangeException](#)

Thrown when the max count is negative.

# Properties

## Count

Actual number of elements in the array.

```
public int Count { get; }
```

## Property Value

[int](#)

## IsReadOnly

Gets a value indicating whether the [ICollection<T>](#) is read-only.

```
public bool IsReadOnly { get; }
```

## Property Value

[bool](#)

[true](#) if the [ICollection<T>](#) is read-only; otherwise, [false](#).

## this[int]

Gets the element at the specified index. Throws an exception if the index is out of bounds.

```
public T this[int index] { get; }
```

### Parameters

index [int](#)

The index of the element to get.

### Property Value

T

The element at the specified index.

### Exceptions

[ArgumentOutOfRangeException](#)

Thrown when the index is out of range.

## MaxCount

The length of the array.

```
public int MaxCount { get; }
```

### Property Value

[int](#)

## Methods

### Add(T)

Adds an item to the [ICollection<T>](#).

```
public void Add(T item)
```

## Parameters

**item** T

The object to add to the [ICollection<T>](#).

## Exceptions

[NotSupportedException](#)

The [ICollection<T>](#) is read-only.

## Clear()

Removes all items from the [ICollection<T>](#).

```
public void Clear()
```

## Exceptions

[NotSupportedException](#)

The [ICollection<T>](#) is read-only.

## Contains(T)

Determines whether the [ICollection<T>](#) contains a specific value.

```
public bool Contains(T item)
```

## Parameters

**item** T

The object to locate in the [ICollection<T>](#).

## Returns

[bool](#)

[true](#) if `item` is found in the [ICollection<T>](#); otherwise, [false](#).

## CopyTo(T[], int)

Copies the elements of the [ICollection<T>](#) to an [Array](#), starting at a particular [Array](#) index.

```
public void CopyTo(T[] array, int arrayIndex)
```

## Parameters

`array` `T[]`

The one-dimensional [Array](#) that is the destination of the elements copied from [ICollection<T>](#). The [Array](#) must have zero-based indexing.

`arrayIndex` [int](#)

The zero-based index in `array` at which copying begins.

## Exceptions

[ArgumentNullException](#)

`array` is [null](#).

[ArgumentOutOfRangeException](#)

`arrayIndex` is less than 0.

[ArgumentException](#)

The number of elements in the source [ICollection<T>](#) is greater than the available space from `arrayIndex` to the end of the destination `array`.

## CopyTo(T[], int, int)

Copies the ExposedQueue to an array. The head should be the last element of the array.  
Copies from start to end inclusive.

```
public void CopyTo(T[] array, int arrayIndex, int endIndex)
```

## Parameters

array T[]

The array to copy to."

arrayIndex [int](#)

The start index of the range to copy.

endIndex [int](#)

The end index of the range to copy.

## GetEnumerator()

Returns an enumerator that iterates through the collection.

```
public IEnumerator<T> GetEnumerator()
```

## Returns

[IEnumerator](#) <T>

An enumerator that can be used to iterate through the collection.

## GetFirst()

Gets the first element of the queue.

```
public T GetFirst()
```

## Returns

T

The first element of the queue.

## GetLast()

Gets the element at the end of the queue.

```
public T GetLast()
```

Returns

T

The element at the end of the queue.

## Put(T)

Puts an element at the start of the queue.

```
public void Put(T value)
```

Parameters

value T

The element to add to the queue.

## Remove(T)

Removes the specified item from the queue and shifts remaining elements to the left. For example, given the queue [4, 3, 2, 1], if you call Remove(3), the resulting queue will be [4, 2, 1].

```
public bool Remove(T item)
```

Parameters



**item** `T`

The item to remove.

Returns

`bool` [↗](#)

True if the item was successfully removed; otherwise, false.

Remarks

The MaxCount will not change, but the Count will decrease by one.

## ToArray()

Converts the ExposedQueue to an array. Only returns the used slots.

```
public T[] ToArray()
```

Returns

`T[]`

An array containing the elements within the specified range.

## ToArray(int, int)

Converts the ExposedQueue to an array.

```
public T[] ToArray(int start, int end)
```

Parameters

**start** `int` [↗](#)

The start index of the range to convert.

**end** `int` [↗](#)

The end index of the range to convert.

Returns

`T[]`

An array containing the elements within the specified range.

# Class OptimizedActivationStack<T>

Namespace: [Aplib.Core.Collections](#)

Assembly: Aplib.Core.dll

A stack that has a predefined set of items that can be *activated* (i.e., pushed on top of the stack). When an item that is already on the stack is activated, it is *reactivated* (i.e., moved to the top of the stack).


```
public class OptimizedActivationStack<T>
```

## Type Parameters








**T**

The type of the items that are put on the stack.

## Inheritance

[object](#) ← OptimizedActivationStack<T>

## Inherited Members

[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#), [object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

## Remarks

The [OptimizedActivationStack<T>](#) allows for O(1) activation and reactivation of an arbitrary stack item.

## Constructors

### OptimizedActivationStack(T[])

Initializes an optimized activation stack with a set of activatable data.

```
public OptimizedActivationStack(T[] activatables)
```

## Parameters

**activatables** T[]

A set of activatable items that could be pushed on the stack.

## Properties

### ActivatableStackItems

Gets the activatable stack items.

```
public IEnumerable<OptimizedActivationStack<T>.StackItem> ActivatableStackItems {  
    get; }
```

## Property Value

[IEnumerable](#)  <[OptimizedActivationStack](#)<T>.[StackItem](#)>

## Remarks

The stack items are exposed, since they should be accessible from the outside to provide O(1) activation of a stack item with [Activate\(StackItem\)](#).

## Count

Gets the number of items that are currently activated (i.e., on the stack).

```
public int Count { get; }
```

## Property Value

[int](#) 

## Exceptions

[InvalidOperationException](#) 

Thrown when the stack count is negative.

# Methods

## Activate(StackItem)

Activates an item (i.e., pushes an item on top of the stack). If the pushed item is already on the stack, it is extracted from the stack before it is put on top again.

```
public void Activate(OptimizedActivationStack<T>.StackItem item)
```

### Parameters

item [OptimizedActivationStack<T>.StackItem](#)

The stack item that is pushed on top of the stack (i.e., it is activated).

### Exceptions

[ArgumentException](#) 

Thrown when an item is pushed that belongs to a different stack.

## Peek()

Peeks the top item from the stack.

```
public T Peek()
```

### Returns

T

The top item.

### Exceptions

[InvalidOperationException](#) 

Thrown when the stack is empty.

# Pop()

Pops the top item from the stack.

```
public T Pop()
```

## Returns

T

The popped item.

## Exceptions

[InvalidOperationException](#)

Thrown when the stack is empty.

# Class

## OptimizedActivationStack<T>.StackItem

Namespace: [Aplib.Core.Collections](#)

Assembly: Aplib.Core.dll







Represents (i.e., encapsulates) an item on the activation stack.

```
public sealed class OptimizedActivationStack<T>.StackItem
```

### Inheritance

[object](#)  ← OptimizedActivationStack<T>.StackItem

### Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Remarks

This class is public, because the whole stack item should be accessible from the outside to provide O(1) activation of a stack item with [Activate\(StackItem\)](#).

## Constructors

### StackItem(T, OptimizedActivationStack<T>)

Creates a stack item for the [OptimizedActivationStack<T>](#) class.

```
public StackItem(T data, OptimizedActivationStack<T> activationStack)
```

### Parameters

**data** T

The data to put on the stack.

**activationStack** [OptimizedActivationStack<T>](#)

The activation stack instance that this stack item belongs to.

# Properties

## ActivationStack

Gets the activation stack instance that this stack item belongs to.

```
public OptimizedActivationStack<T> ActivationStack { get; }
```

## Property Value

[OptimizedActivationStack<T>](#)

## Data

Gets the data that this stack item represents.

```
public T Data { get; }
```

## Property Value

T

## IsActive

Gets or sets a value indicating whether the item is currently on the stack.

```
public bool IsActive { get; set; }
```

## Property Value

[bool](#)

## Next

Gets or sets the next (above) item on the stack.



```
public OptimizedActivationStack<T>.StackItem? Next { get; set; }
```

Property Value

[OptimizedActivationStack<T>.](#)[StackItem](#)

## Previous

Gets or sets the previous (below) item on the stack.

```
public OptimizedActivationStack<T>.StackItem? Previous { get; set; }
```

Property Value

[OptimizedActivationStack<T>.](#)[StackItem](#)

## Methods

### PushOnStackAfter(StackItem)

Pushes an item that is not on the stack yet after another item that is already on the stack.

```
public void PushOnStackAfter(OptimizedActivationStack<T>.StackItem item)
```

Parameters

**item** [OptimizedActivationStack<T>.](#)[StackItem](#)

An item that is already on the stack.

Exceptions

[ArgumentException](#)

Thrown when an item is pushed after an item that is not on the same stack, when an item is already on the stack, or when an item is pushed after an item that is not on the stack.

# RemoveFromStack()

Safely remove the item from the stack.

```
public void RemoveFromStack()
```

# Namespace Aplib.Core.Desire.DesireSets

## Classes

[DesireSet<TBeliefSet>](#)

## Interfaces

[IDesireSet<TBeliefSet>](#)

Represents a set of goals that the agent has. This is the main structure that the agent will use to determine what it should do next.

# Class DesireSet<TBeliefSet>

Namespace: [Aplib.Core.Desire.DesireSets](#)

Assembly: Aplib.Core.dll

```
public class DesireSet<TBeliefSet> : IDesireSet<TBeliefSet>, ICompletable,  
    ILoggable, IDocumented where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**








## Inheritance

[object](#)  ← DesireSet<TBeliefSet>

## Implements

[IDesireSet](#)<TBeliefSet>, [ICompletable](#), [ILoggable](#), [IDocumented](#)

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Constructors

DesireSet(IGoalStructure<TBeliefSet>, params  
(IGoalStructure<TBeliefSet> goalStructure,  
Predicate<TBeliefSet> guard)[])

```
public DesireSet(IGoalStructure<TBeliefSet> mainGoal, params  
    (IGoalStructure<TBeliefSet> goalStructure, Predicate<TBeliefSet> guard)[] sideGoals)
```

## Parameters

**mainGoal** [IGoalStructure](#)<TBeliefSet>

```
sideGoals (IGoalStructure<TBeliefSet> goalStructure, Predicate<TBeliefSet> guard)
[]
```

## DesireSet(IMetadata, IGoalStructure<TBeliefSet>, params (IGoalStructure<TBeliefSet> goalStructure, Predicate<TBeliefSet> guard)[])

Initializes a new instance of the [DesireSet<TBeliefSet>](#) class.

```
public DesireSet(IMetadata metadata, IGoalStructure<TBeliefSet> mainGoal, params
(IGoalStructure<TBeliefSet> goalStructure, Predicate<TBeliefSet> guard)[] sideGoals)
```

### Parameters

**metadata** [IMetadata](#)

Metadata about this GoalStructure, used to quickly display the goal in several contexts.

**mainGoal** [IGoalStructure](#)<TBeliefSet>

The main goal structure that the agent needs to complete.

```
sideGoals (IGoalStructure<TBeliefSet> goalStructure, Predicate<TBeliefSet> guard)
[]
```

The side goal structures that could be activated during the agent playthrough.

## Properties

### Metadata

Gets the metadata of the instance.

```
public IMetadata Metadata { get; }
```

### Property Value

[IMetadata](#)

# Status

If there are no goal structures left to be completed, the status of this desire set is set to the main goal status.

```
public CompletionStatus Status { get; }
```

## Property Value

[CompletionStatus](#)

# Methods

## GetCurrentGoal(TBeliefSet)

Gets the current goal using the given [IBeliefSet](#).

```
public IGoal<TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

## Returns

[IGoal](#)<TBeliefSet>

The current goal to be fulfilled.

## GetLogChildren()

Gets the children of the loggable object.

```
public IEnumerable<ILoggable> GetLogChildren()
```

## Returns

[IEnumerable](#)  <[ILoggable](#)>

The children of the loggable object.

## Update(TBeliefSet)

Activates side goal structures when their guard is satisfied, and updates the activation stack by popping goal structures from the top of the stack when they are finished.

```
public void Update(TBeliefSet beliefSet)
```

### Parameters

**beliefSet** [TBeliefSet](#)

The belief set of the agent.

## Operators

### implicit operator DesireSet<TBeliefSet> (GoalStructure<TBeliefSet>)

Implicitly lifts a goal structure a desire set.

```
public static implicit operator DesireSet<TBeliefSet>(GoalStructure<TBeliefSet>  
goalStructure)
```

### Parameters

**goalStructure** [GoalStructure](#)<[TBeliefSet](#)>

The goal structure which on its own can function as a desire set. Meaning, the desire set consists of just a single goal structure.

### Returns

[DesireSet](#)<[TBeliefSet](#)>

The most logically matching desire set, wrapping around **goalStructure**.

# implicit operator DesireSet<TBeliefSet> (Goal<TBeliefSet>)

Implicitly lifts a goal into a desire set.

```
public static implicit operator DesireSet<TBeliefSet>(Goal<TBeliefSet> goal)
```

## Parameters

**goal** [Goal](#)<TBeliefSet>

The goal which on its own can function as a goal structure. Meaning, the goal structure consists of just a single goal.

## Returns

[DesireSet](#)<TBeliefSet>

The most logically matching desire set, wrapping around **goal**.



# Interface IDesireSet<TBeliefSet>

Namespace: [Aplib.Core.Desire.DesireSets](#)

Assembly: Aplib.Core.dll

Represents a set of goals that the agent has. This is the main structure that the agent will use to determine what it should do next.

```
public interface IDesireSet<in TBeliefSet> : ICompletable where TBeliefSet
: IBeliefSet
```

## Type Parameters

**TBeliefSet**

## Inherited Members

[ICompletable.Status](#)

## Methods

### GetCurrentGoal(TBeliefSet)

Gets the current goal using the given [IBeliefSet](#).

```
IGoal<in TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

## Returns

[IGoal](#)<TBeliefSet>

The current goal to be fulfilled.

# Update(TBeliefSet)

Updates the status of this [IDesireSet<TBeliefSet>](#).

```
void Update(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Namespace Aplib.Core.Desire.Goal Structures

## Classes

### [FirstOfGoalStructure<TBeliefSet>](#)

Represents a goal structure that will complete if any one of its children completes.

### [GoalStructure<TBeliefSet>](#)

Describes a structure of goals that need to be fulfilled.

### [PrimitiveGoalStructure<TBeliefSet>](#)

Represents a goal structure that will complete if any of its children complete.

### [RepeatGoalStructure<TBeliefSet>](#)

Represents a goal structure that will complete if any of its children complete. This structure will repeatedly execute the goal it was created with until the goal is finished, or the maximum number of retries is reached.

### [SequentialGoalStructure<TBeliefSet>](#)

Represents a goal structure that will complete if all of its children complete.

## Interfaces

### [IGoalStructure<TBeliefSet>](#)

Represents a goal structure.

# Class FirstOfGoalStructure<TBeliefSet>

Namespace: [Aplib.Core.Desire.GoalStructures](#)

Assembly: Aplib.Core.dll

Represents a goal structure that will complete if any one of its children completes.

```
public class FirstOfGoalStructure<TBeliefSet> : GoalStructure<TBeliefSet>,
    IGoalStructure<TBeliefSet>, ICompletable, ILoggable, IDocumented, IDisposable where
    TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The beliefset of the agent.








## Inheritance

[object](#)  ← [GoalStructure](#)<TBeliefSet> ← FirstOfGoalStructure<TBeliefSet>

## Implements

[IGoalStructure](#)<TBeliefSet>, [ICompletable](#), [ILoggable](#), [IDocumented](#), [IDisposable](#) 

## Inherited Members

[GoalStructure<TBeliefSet>.Metadata](#), [GoalStructure<TBeliefSet>.\\_children](#),  
[GoalStructure<TBeliefSet>.Status](#), [GoalStructure<TBeliefSet>.\\_currentGoalStructure](#),  
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,  
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,  
[object.ToString\(\)](#) 

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>\)](#),  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>, IMetadata\)](#)

## Remarks

The children of this goal structure will be executed in the order they are given.

## Constructors

## FirstOfGoalStructure(params IGoalStructure<TBeliefSet>[])

Initializes a new instance of the [FirstOfGoalStructure<TBeliefSet>](#) class.

```
public FirstOfGoalStructure(params IGoalStructure<TBeliefSet>[] children)
```

### Parameters

**children** [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

## FirstOfGoalStructure(IMetadata, params IGoalStructure<TBeliefSet>[])

Initializes a new instance of the [FirstOfGoalStructure<TBeliefSet>](#) class.

```
public FirstOfGoalStructure(IMetadata metadata, params IGoalStructure<TBeliefSet>  
[] children)
```

### Parameters

**metadata** [IMetadata](#)

Metadata about this GoalStructure, used to quickly display the goal in several contexts.

**children** [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

## Methods

### Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.


```
public void Dispose()
```

## Dispose(bool)

Disposes of the goal structure.

```
protected virtual void Dispose(bool disposing)
```

### Parameters

disposing [bool](#)

Whether we are actually disposing.

## GetCurrentGoal(TBeliefSet)

Gets the current goal using the given [IBeliefSet](#).

```
public override IGoal<TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```

### Parameters

beliefSet [TBeliefSet](#)

The belief set of the agent.

### Returns

[IGoal](#)<[TBeliefSet](#)>

The current goal to be fulfilled.

## GetLogChildren()

Gets the children of the loggable object.

```
public override IEnumerable<ILoggable> GetLogChildren()
```

Returns

[IEnumerable](#)  <[ILoggable](#)>

The children of the loggable object.

## Reset()

Resets the goal structure to its initial state.

```
public override void Reset()
```

## UpdateStatus(TBeliefSet)

Updates the status of the [FirstOfGoalStructure<TBeliefSet>](#). The goal structure status is set to:

- [Success](#) - When any one of its children is successful.
- [Failure](#) - When all children fail.
- [Unfinished](#) - Otherwise.

```
public override void UpdateStatus(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Class GoalStructure<TBeliefSet>

Namespace: [Aplib.Core.Desire.GoalStructures](#)

Assembly: Aplib.Core.dll

Describes a structure of goals that need to be fulfilled.

```
public abstract class GoalStructure<TBeliefSet> : IGoalStructure<TBeliefSet>,
    ICompletable, ILoggable, IDocumented where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

## Inheritance

[object](#)  ← GoalStructure<TBeliefSet>

## Implements

[IGoalStructure](#)<TBeliefSet>, [ICompletable](#), [ILoggable](#), [IDocumented](#)

## Derived

[FirstOfGoalStructure](#)<TBeliefSet>, [PrimitiveGoalStructure](#)<TBeliefSet>, [RepeatGoalStructure](#)<TBeliefSet>, [SequentialGoalStructure](#)<TBeliefSet>

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Extension Methods

[LiftingExtensionMethods.Lift](#)<TBeliefSet>(IGoalStructure<TBeliefSet>), [LiftingExtensionMethods.Lift](#)<TBeliefSet>(IGoalStructure<TBeliefSet>, IMetadata)

## Constructors

GoalStructure(IMetadata,  
IEnumerable<IGoalStructure<TBeliefSet>>)

Initializes a new instance of the [GoalStructure<TBeliefSet>](#) class.



```
protected GoalStructure(IMetadata metadata, IEnumerable<IGoalStructure<TBeliefSet>>
children)
```

## Parameters

metadata [IMetadata](#)

Metadata about this GoalStructure, used to quickly display the goal in several contexts.

children [IEnumerable](#) <[IGoalStructure](#)<TBeliefSet>>

The children of the goal structure.

## GoalStructure(IEnumerable<IGoalStructure<TBeliefSet>>)

Initializes a new instance of the [GoalStructure<TBeliefSet>](#) class.

```
protected GoalStructure(IEnumerable<IGoalStructure<TBeliefSet>> children)
```

## Parameters

children [IEnumerable](#) <[IGoalStructure](#)<TBeliefSet>>

The children of the goal structure.

## Fields

### \_children

The children of the goal structure.

```
protected readonly IEnumerable<IGoalStructure<TBeliefSet>> _children
```

## Field Value

[IEnumerable](#) <[IGoalStructure](#)<TBeliefSet>>

# **\_currentGoalStructure**

The goal structure that is currently being fulfilled.

```
protected IGoalStructure<TBeliefSet>? _currentGoalStructure
```

Field Value

[IGoalStructure](#)<TBeliefSet>

## **Properties**

### **Metadata**

Gets the metadata of the instance.

```
public IMetadata Metadata { get; }
```

Property Value

[IMetadata](#)

### **Status**

Gets the completion status of the object.

```
public CompletionStatus Status { get; protected set; }
```

Property Value

[CompletionStatus](#)

## **Methods**

### **GetCurrentGoal(TBeliefSet)**

Gets the current goal using the given [IBeliefSet](#).

```
public abstract IGoal<TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

## Returns

[IGoal](#)<TBeliefSet>

The current goal to be fulfilled.

## GetLogChildren()

Gets the children of the loggable object.

```
public abstract IEnumerable<ILoggable> GetLogChildren()
```

## Returns

[IEnumerable](#)<sup>↗</sup><[ILoggable](#)>

The children of the loggable object.

## Reset()

Resets the goal structure to its initial state.

```
public virtual void Reset()
```

## UpdateStatus(TBeliefSet)

Updates the state of the goal structure.

```
public abstract void UpdateStatus(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

## Operators

implicit operator GoalStructure<TBeliefSet>  
(Goal<TBeliefSet>)

Implicitly lifts a goal into a goal structure.

```
public static implicit operator GoalStructure<TBeliefSet>(Goal<TBeliefSet> goal)
```

## Parameters

**goal** [Goal](#)<TBeliefSet>

The goal which on its own can function as a goal structure. Meaning, the goal structure consists of just a single goal.

## Returns

[GoalStructure](#)<TBeliefSet>

The most logically matching goal structure, wrapping around **goal**.

# Interface IGoalStructure<TBeliefSet>

Namespace: [Aplib.Core.Desire.GoalStructures](#)

Assembly: Aplib.Core.dll

Represents a goal structure.

```
public interface IGoalStructure<in TBeliefSet> : ICompletable where TBeliefSet
: IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.

## Inherited Members

[ICompletable.Status](#)

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>, IMetadata\)](#)

## Remarks

A goal structure is a structure of predicates that must be fulfilled in order to complete a test.

## Methods

### GetCurrentGoal(TBeliefSet)

Gets the current goal using the given [IBeliefSet](#).

```
IGoal<in TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

Returns

[IGoal](#)<TBeliefSet>

The current goal to be fulfilled.

## Reset()

Resets the goal structure to its initial state.

```
void Reset()
```

## UpdateStatus(TBeliefSet)

Updates the state of the goal structure.

```
void UpdateStatus(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Class PrimitiveGoalStructure<TBeliefSet>

Namespace: [Aplib.Core.Desire.GoalStructures](#)

Assembly: Aplib.Core.dll

Represents a goal structure that will complete if any of its children complete.

```
public class PrimitiveGoalStructure<TBeliefSet> : GoalStructure<TBeliefSet>,
    IGoalStructure<TBeliefSet>, ICompletable, ILoggable, IDocumented where TBeliefSet
    : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The beliefset of the agent.








## Inheritance

[object](#)  ← [GoalStructure](#)<TBeliefSet> ← PrimitiveGoalStructure<TBeliefSet>

## Implements

[IGoalStructure](#)<TBeliefSet>, [ICompletable](#), [ILoggable](#), [IDocumented](#)

## Inherited Members

[GoalStructure<TBeliefSet>.Metadata](#), [GoalStructure<TBeliefSet>.\\_children](#),  
[GoalStructure<TBeliefSet>.Status](#), [GoalStructure<TBeliefSet>.\\_currentGoalStructure](#),  
[GoalStructure<TBeliefSet>.Reset\(\)](#), [object.Equals\(object\)](#) ,  
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>\)](#),  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>, IMetadata\)](#)

## Remarks

This is the most primitive goal structure. It is used to represent a single goal that is not part of a larger structure. This goal structure will only return the goal it was created with if the goal is not yet finished.

# Constructors

## PrimitiveGoalStructure(IGoal<TBeliefSet>)

Initializes a new instance of the [PrimitiveGoalStructure<TBeliefSet>](#) class.

```
public PrimitiveGoalStructure(IGoal<TBeliefSet> goal)
```

### Parameters

goal [IGoal](#)<TBeliefSet>

The goal to fulfill.

## PrimitiveGoalStructure(IMetadata, IGoal<TBeliefSet>)

Initializes a new instance of the [PrimitiveGoalStructure<TBeliefSet>](#) class.

```
public PrimitiveGoalStructure(IMetadata metadata, IGoal<TBeliefSet> goal)
```

### Parameters

metadata [IMetadata](#)

Metadata about this GoalStructure, used to quickly display the goal in several contexts.

goal [IGoal](#)<TBeliefSet>

The goal to fulfill.

# Methods

## GetCurrentGoal(TBeliefSet)

Gets the current goal using the given [IBeliefSet](#).

```
public override IGoal<TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```



## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

## Returns

[IGoal](#)<TBeliefSet>

The current goal to be fulfilled.

## GetLogChildren()

Gets the children of the loggable object.

```
public override IEnumerable<ILoggable> GetLogChildren()
```

## Returns

[IEnumerable](#) <[ILoggable](#)>

The children of the loggable object.

## UpdateStatus(TBeliefSet)

Updates the status of the [FirstOfGoalStructure<TBeliefSet>](#). The goal structure status is set to the status of the underlying [IGoal<TBeliefSet>](#).

```
public override void UpdateStatus(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Class RepeatGoalStructure<TBeliefSet>

Namespace: [Aplib.Core.Desire.GoalStructures](#)

Assembly: Aplib.Core.dll

Represents a goal structure that will complete if any of its children complete. This structure will repeatedly execute the goal it was created with until the goal is finished, or the maximum number of retries is reached.

```
public class RepeatGoalStructure<TBeliefSet> : GoalStructure<TBeliefSet>,
    IGoalStructure<TBeliefSet>, ICompletable, ILoggable, IDocumented where TBeliefSet
    : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.








## Inheritance

[object](#)  ← [GoalStructure](#)<TBeliefSet> ← RepeatGoalStructure<TBeliefSet>

## Implements

[IGoalStructure](#)<TBeliefSet>, [ICompletable](#), [ILoggable](#), [IDocumented](#)

## Inherited Members

[GoalStructure<TBeliefSet>.Metadata](#) , [GoalStructure<TBeliefSet>.\\_children](#) ,  
[GoalStructure<TBeliefSet>.Status](#) , [GoalStructure<TBeliefSet>.\\_currentGoalStructure](#) ,  
[GoalStructure<TBeliefSet>.Reset\(\)](#) , [object.Equals\(object\)](#)  ,  
[object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  ,  
[object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>, IMetadata\)](#)

## Constructors

RepeatGoalStructure(IGoalStructure<TBeliefSet>)

```
public RepeatGoalStructure(IGoalStructure<TBeliefSet> goalStructure)
```

## Parameters

goalStructure [IGoalStructure](#)<TBeliefSet>

## RepeatGoalStructure(IGoalStructure<TBeliefSet>, int)

```
public RepeatGoalStructure(IGoalStructure<TBeliefSet> goalStructure, int maxRetries)
```

## Parameters

goalStructure [IGoalStructure](#)<TBeliefSet>

maxRetries [int](#)

## RepeatGoalStructure(IMetadata, IGoalStructure<TBeliefSet>)

```
public RepeatGoalStructure(IMetadata metadata, IGoalStructure<TBeliefSet>  
goalStructure)
```

## Parameters

metadata [IMetadata](#)

goalStructure [IGoalStructure](#)<TBeliefSet>

## RepeatGoalStructure(IMetadata, IGoalStructure<TBeliefSet>, int)

```
public RepeatGoalStructure(IMetadata metadata, IGoalStructure<TBeliefSet>  
goalStructure, int maxRetries)
```

## Parameters

**metadata** [IMetadata](#)

**goalStructure** [IGoalStructure](#)<TBeliefSet>

**maxRetries** [int](#)<sup>↗</sup>

## RepeatGoalStructure(IMetadata, IGoalStructure<TBeliefSet>, int?)

Initializes a new instance of the [RepeatGoalStructure<TBeliefSet>](#) class.

```
protected RepeatGoalStructure(IMetadata metadata, IGoalStructure<TBeliefSet>  
goalStructure, int? maxRetries)
```

## Parameters

**metadata** [IMetadata](#)

Metadata about this goal, used to quickly display the goal in several contexts.

**goalStructure** [IGoalStructure](#)<TBeliefSet>

The GoalStructure to repeat.

**maxRetries** [int](#)<sup>↗</sup>?

The maximum number of times to retry the goal after it has failed. If omitted, the goal will be retried indefinitely.

## Exceptions

[ArgumentOutOfRangeException](#)<sup>↗</sup>

If **maxRetries** is less than zero.

## Fields

**\_maxRetries**

The maximum number of times to retry the goal after it has failed. If this is `null`, the goal will be retried indefinitely.

```
protected readonly int? _maxRetries
```

Field Value

[int](#)?

## `_retryCount`

The number of times the goal has been retried so far.

```
protected int _retryCount
```

Field Value

[int](#)

## Methods

### GetCurrentGoal(TBeliefSet)

Gets the current goal using the given [IBeliefSet](#).

```
public override IGoal<TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```

Parameters

`beliefSet` [TBeliefSet](#)

The belief set of the agent.

Returns

[IGoal](#)<[TBeliefSet](#)>

The current goal to be fulfilled.

# GetLogChildren()

Gets the children of the loggable object.

```
public override IEnumerable<ILoggable> GetLogChildren()
```

Returns

[IEnumerable](#)  <[ILoggable](#)>

The children of the loggable object.

# UpdateStatus(TBeliefSet)

Updates the status of the [RepeatGoalStructure<TBeliefSet>](#). The goal structure status is set to:

- [Success](#) - When the underlying goal structure is successful.
- [Failure](#) - If the underlying goal structure fails when the maximum number of retries has been reached. But never if no maximum number of retries has been specified.
- [Unfinished](#) - When the underlying goal structure is unfinished. The underlying goal structure will be retried when it fails.

```
public override void UpdateStatus(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Class

## SequentialGoalStructure<TBeliefSet>

Namespace: [Aplib.Core.Desire.GoalStructures](#)

Assembly: Aplib.Core.dll

Represents a goal structure that will complete if all of its children complete.

```
public class SequentialGoalStructure<TBeliefSet> : GoalStructure<TBeliefSet>,
    IGoalStructure<TBeliefSet>, ICompletable, ILoggable, IDocumented, IDisposable where
    TBeliefSet : IBeliefSet
```

### Type Parameters

**TBeliefSet**

The type of belief set that this goal structure operates on.








### Inheritance

[object](#)  ← [GoalStructure](#)<TBeliefSet> ← SequentialGoalStructure<TBeliefSet>

### Implements

[IGoalStructure](#)<TBeliefSet>, [ICompletable](#), [ILoggable](#), [IDocumented](#), [IDisposable](#) 

### Inherited Members

[GoalStructure<TBeliefSet>.Metadata](#), [GoalStructure<TBeliefSet>.\\_children](#), [GoalStructure<TBeliefSet>.Status](#), [GoalStructure<TBeliefSet>.\\_currentGoalStructure](#), [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

### Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>\)](#), [LiftingExtensionMethods.Lift<TBeliefSet>\(IGoalStructure<TBeliefSet>, IMetadata\)](#)

### Remarks

The children of this goal structure will be executed in the order they are given.

# Constructors

## SequentialGoalStructure(params IGoalStructure<TBeliefSet>[])

Initializes a new instance of the [SequentialGoalStructure<TBeliefSet>](#) class.

```
public SequentialGoalStructure(params IGoalStructure<TBeliefSet>[] children)
```

### Parameters

**children** [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

## SequentialGoalStructure(IMetadata, params IGoalStructure<TBeliefSet>[])

Initializes a new instance of the [SequentialGoalStructure<TBeliefSet>](#) class.

```
public SequentialGoalStructure(IMetadata metadata, params IGoalStructure<TBeliefSet>[] children)
```

### Parameters

**metadata** [IMetadata](#)

Metadata about this GoalStructure, used to quickly display the goal in several contexts.

**children** [IGoalStructure](#)<TBeliefSet>[]

The children of the goal structure.

# Methods

## Dispose()



Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.


```
public void Dispose()
```

## Dispose(bool)

Disposes the enumerator.

```
protected virtual void Dispose(bool disposing)
```

### Parameters

disposing [bool](#)

Whether the object is being disposed.

## GetCurrentGoal(TBeliefSet)

Gets the current goal using the given [IBeliefSet](#).

```
public override IGoal<TBeliefSet> GetCurrentGoal(TBeliefSet beliefSet)
```

### Parameters

beliefSet [TBeliefSet](#)

The belief set of the agent.

### Returns

[IGoal](#)<[TBeliefSet](#)>

The current goal to be fulfilled.

## GetLogChildren()

Gets the children of the loggable object.

```
public override IEnumerable<ILoggable> GetLogChildren()
```

Returns

[IEnumerable](#)  <[ILoggable](#)>

The children of the loggable object.

## Reset()

Resets the goal structure to its initial state.

```
public override void Reset()
```

## UpdateStatus(TBeliefSet)

Updates the status of the [SequentialGoalStructure<TBeliefSet>](#). The goal structure status is set to:

- [Success](#) - When all children are successful.
- [Failure](#) - When any one of its children fails.
- [Unfinished](#) - Otherwise.

```
public override void UpdateStatus(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Namespace Aplib.Core.Desire.Goals

## Classes

### [Goal<TBeliefSet>](#)

A goal effectively combines a heuristic function with a tactic, and aims to meet the heuristic function by applying the tactic. Goals are combined in a

[GoalStructure<TBeliefSet>](#), and are used to prepare tests or do the testing.

## Interfaces

### [IGoal<TBeliefSet>](#)

Defines a goal that can be achieved by a [Tactic<TBeliefSet>](#).

# Class Goal<TBeliefSet>

Namespace: [Aplib.Core.Desire.Goals](#)

Assembly: Aplib.Core.dll

A goal effectively combines a heuristic function with a tactic, and aims to meet the heuristic function by applying the tactic. Goals are combined in a [GoalStructure<TBeliefSet>](#), and are used to prepare tests or do the testing.


```
public class Goal<TBeliefSet> : IGoal<TBeliefSet>, ICompletable, ILoggable,
IDocumented where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.








## Inheritance

[object](#)  ← Goal<TBeliefSet>

## Implements

[IGoal](#)<TBeliefSet>, [ICompletable](#), [ILoggable](#), [IDocumented](#)

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoal<TBeliefSet>\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoal<TBeliefSet>, IMetadata\)](#)

## Constructors

Goal(IMetadata, ITactic<TBeliefSet>, Predicate<TBeliefSet>)

```
public Goal(IMetadata metadata, ITactic<TBeliefSet> tactic,  
Predicate<TBeliefSet> predicate)
```

## Parameters

**metadata** [IMetadata](#)

**tactic** [ITactic](#)<TBeliefSet>

**predicate** [Predicate](#)<TBeliefSet>

## Goal(IMetadata, ITactic<TBeliefSet>, Predicate<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new goal from a given tactic and a success predicate, and an optional fail-guard.

```
public Goal(IMetadata metadata, ITactic<TBeliefSet> tactic, Predicate<TBeliefSet>  
predicate, Predicate<TBeliefSet> failGuard)
```

## Parameters

**metadata** [IMetadata](#)

Metadata about this goal, used to quickly display the goal in several contexts. If omitted, default metadata will be generated.

**tactic** [ITactic](#)<TBeliefSet>

The tactic used to approach this goal.

**predicate** [Predicate](#)<TBeliefSet>

A predicate that determines when the goal has succeeded.

**failGuard** [Predicate](#)<TBeliefSet>

A predicate that determines when the goal has failed. If the fail-guard is true, but the success predicate is also satisfied, the success predicate takes precedence. If omitted, the goal will never fail.

# Goal(ITactic<TBeliefSet>, Predicate<TBeliefSet>)

```
public Goal(ITactic<TBeliefSet> tactic, Predicate<TBeliefSet> predicate)
```

## Parameters

tactic [ITactic](#)<TBeliefSet>

predicate [Predicate](#)<TBeliefSet>

# Goal(ITactic<TBeliefSet>, Predicate<TBeliefSet>, Predicate<TBeliefSet>)

```
public Goal(ITactic<TBeliefSet> tactic, Predicate<TBeliefSet> predicate,  
Predicate<TBeliefSet> failGuard)
```

## Parameters

tactic [ITactic](#)<TBeliefSet>

predicate [Predicate](#)<TBeliefSet>

failGuard [Predicate](#)<TBeliefSet>

## Fields

### \_failGuard

An (optional) fail-guard for the goal's completion status. The fail-guard predicate is a condition that, when true, indicates that the goal has failed.

```
protected readonly Predicate<TBeliefSet> _failGuard
```

## Field Value


[Predicate](#)<TBeliefSet>

# `_predicate`

A predicate that determines whether the goal has succeeded. Intuitively, the predicate is the goal itself.

```
protected readonly Predicate<TBeliefSet> _predicate
```

Field Value

[Predicate](#)  <TBeliefSet>

## Properties

### Metadata

Gets the metadata of the instance.

```
public IMetadata Metadata { get; }
```

Property Value

[IMetadata](#)

### Status

Gets the completion status of the goal. This value may need to be updated first using the [UpdateStatus\(TBeliefSet\)](#) method.

```
public CompletionStatus Status { get; protected set; }
```

Property Value

[CompletionStatus](#)

### See Also

[UpdateStatus\(TBeliefSet\)](#)

# Tactic

The [Tactic<TBeliefSet>](#) used to achieve this [Goal<TBeliefSet>](#). It is executed once in every iteration of the BDI cycle while this goal is the active goal of the agent.

```
public ITactic<TBeliefSet> Tactic { get; }
```

## Property Value

[ITactic<TBeliefSet>](#)

## Methods

### GetLogChildren()

Gets the children of the loggable object.

```
public IEnumerable<ILoggable> GetLogChildren()
```

## Returns

[IEnumerable](#) [<ILoggable>](#)

The children of the loggable object.

### UpdateStatus(TBeliefSet)

Checks whether the goal has been achieved and stores the result in [Status](#).

If the predicate of the goal is satisfied, the goal is considered to have succeeded. If the fail-guard is satisfied, the goal is considered to have failed. If both are satisfied, the success predicate takes precedence. If neither are satisfied, the goal is considered unfinished. The table below summarizes the possible outcomes:

Predicate	Fail guard	Result
true	false	<a href="#">Success</a>
true	true	<a href="#">Success</a>



Predicate	Fail guard	Result
false	true	<a href="#">Failure</a>
false	false	<a href="#">Unfinished</a>

Use [Status](#) to get the updated value.

```
public virtual void UpdateStatus(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

## See Also

[GoalStructure](#)<TBeliefSet>

# Interface IGoal<TBeliefSet>

Namespace: [Aplib.Core.Desire.Goals](#)

Assembly: Aplib.Core.dll

Defines a goal that can be achieved by a [Tactic<TBeliefSet>](#).

```
public interface IGoal<in TBeliefSet> : ICompletable where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.

## Inherited Members

[ICompletable.Status](#)

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoal<TBeliefSet>\)](#) ,

[LiftingExtensionMethods.Lift<TBeliefSet>\(IGoal<TBeliefSet>, IMetadata\)](#)

## Properties

### Tactic

The [Tactic<TBeliefSet>](#) used to achieve this [Goal<TBeliefSet>](#), which is executed during every iteration of the BDI cycle.

```
ITactic<in TBeliefSet> Tactic { get; }
```

## Property Value

[ITactic<TBeliefSet>](#)

## Methods

# UpdateStatus(TBeliefSet)

Tests whether the goal has been achieved, based on the heuristic function of the goal. The new completion status can be accessed via the [Status](#) property.

```
void UpdateStatus(TBeliefSet beliefSet)
```

## Parameters

`beliefSet` TBeliefSet

## See Also

[Status](#)

# Namespace Aplib.Core.Intent.Actions

## Classes

### [Action<TBeliefSet>](#)

Describes an action that can be executed and guarded.

### [QueryAction<TBeliefSet, TQuery>](#)

Describes an action that can be executed and guarded with a query that stores a result. The result can be used in the effect.

## Interfaces

### [IAction<TBeliefSet>](#)

Represents an action that can be executed on a belief set.

### [IQueryable<TBeliefSet>](#)

Represents an interface for executing queries on a belief set.

# Class Action<TBeliefSet>

Namespace: [Aplib.Core.Intent.Actions](#)

Assembly: Aplib.Core.dll

Describes an action that can be executed and guarded.


```
public class Action<TBeliefSet> : IAction<TBeliefSet>, ILoggable, IDocumented where
    TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.

## Inheritance

[object](#)  ← Action<TBeliefSet>








## Implements

[IAction](#)<TBeliefSet>, [ILoggable](#), [IDocumented](#)

## Derived

[QueryAction](#)<TBeliefSet, TQuery>

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>, IMetadata\)](#)

## Constructors

Action(IMetadata, Action<TBeliefSet>)

Initializes a new instance of the [Action<TBeliefSet>](#) class.

```
public Action(IMetadata metadata, Action<TBeliefSet> effect)
```

## Parameters

metadata [IMetadata](#)

Metadata about this action, used to quickly display the action in several contexts.

effect [Action](#) <TBeliefSet>

The effect of the action.

## Action(Action<TBeliefSet>)

Initializes a new instance of the [Action<TBeliefSet>](#) class.

```
public Action(Action<TBeliefSet> effect)
```

## Parameters

effect [Action](#) <TBeliefSet>

The effect of the action.

## Fields

### \_effect

Gets or sets the effect of the action.

```
protected readonly Action<TBeliefSet> _effect
```

## Field Value

[Action](#) <TBeliefSet>

# Properties

## Metadata

Gets the metadata of the instance.

```
public IMetadata Metadata { get; }
```

## Property Value

[IMetadata](#)

# Methods

## Execute(TBeliefSet)

Executes the action on the specified belief set.

```
public virtual void Execute(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set on which the action is executed.

## GetLogChildren()

Actions do not have children, as they are the lowest level of the hierarchy.

```
public IEnumerable<ILoggable> GetLogChildren()
```

## Returns

[IEnumerable](#)  <[ILoggable](#)>

An empty enumerable.

# Interface IAction<TBeliefSet>

Namespace: [Aplib.Core.Intent.Actions](#)

Assembly: Aplib.Core.dll

Represents an action that can be executed on a belief set.

```
public interface IAction<in TBeliefSet> where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The type of the belief set that the action uses.

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>, IMetadata\)](#)

## Methods

### Execute(TBeliefSet)

Executes the action on the specified belief set.

```
void Execute(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set on which the action is executed.



# Interface IQueryable<TBeliefSet>

Namespace: [Aplib.Core.Intent.Actions](#)

Assembly: Aplib.Core.dll

Represents an interface for executing queries on a belief set.

```
public interface IQueryable<in TBeliefSet> : IAction<TBeliefSet> where TBeliefSet
: IBeliefSet
```

## Type Parameters

**TBeliefSet**

The type of the query object.

## Inherited Members

[IAction<TBeliefSet>.Execute\(TBeliefSet\)](#)

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>, IMetadata\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IQueryable<TBeliefSet>\)](#) ,  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IQueryable<TBeliefSet>, IMetadata\)](#)

## Methods

### Query(TBeliefSet)

Executes a query on the specified belief set.

```
bool Query(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set to query.

## Returns

[bool](#) 

A boolean value indicating whether the query executed successfully or not.

# Class QueryAction<TBeliefSet, TQuery>

Namespace: [Aplib.Core.Intent.Actions](#)

Assembly: Aplib.Core.dll

Describes an action that can be executed and guarded with a query that stores a result. The result can be used in the effect.

```
public class QueryAction<TBeliefSet, TQuery> : Action<TBeliefSet>, ILoggable,
IDocumented, IQueryable<TBeliefSet>, IAction<TBeliefSet> where TBeliefSet
: IBeliefSet
```

## Type Parameters

### TBeliefSet

The belief set of the agent.

### TQuery

The type of the query of the action








## Inheritance

[object](#)  ← [Action](#)<TBeliefSet> ← QueryAction<TBeliefSet, TQuery>

## Implements

[ILoggable](#), [IDocumented](#), [IQueryable](#)<TBeliefSet>, [IAction](#)<TBeliefSet>

## Inherited Members

[Action<TBeliefSet>.Metadata](#), [Action<TBeliefSet>.GetLogChildren\(\)](#),  
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,  
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,  
[object.ToString\(\)](#) 

## Extension Methods

[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>\)](#),  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IAction<TBeliefSet>, IMetadata\)](#),  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IQueryable<TBeliefSet>\)](#),  
[LiftingExtensionMethods.Lift<TBeliefSet>\(IQueryable<TBeliefSet>, IMetadata\)](#)

# Constructors

## QueryAction(IMetadata, Action<TBeliefSet, TQuery>, Func<TBeliefSet, TQuery?>)

Initializes a new instance of the [QueryAction<TBeliefSet, TQuery>](#) class.

```
public QueryAction(IMetadata metadata, Action<TBeliefSet, TQuery> effect,
    Func<TBeliefSet, TQuery?> query)
```

### Parameters

**metadata** [IMetadata](#)

Metadata about this action, used to quickly display the action in several contexts.

**effect** [Action](#) <TBeliefSet, TQuery>

The effect of the action.

**query** [Func](#) <TBeliefSet, TQuery>

The query of the action.

## QueryAction(Action<TBeliefSet, TQuery>, Func<TBeliefSet, TQuery?>)

Initializes a new instance of the [QueryAction<TBeliefSet, TQuery>](#) class.

```
public QueryAction(Action<TBeliefSet, TQuery> effect, Func<TBeliefSet, TQuery?>
    > query)
```

### Parameters

**effect** [Action](#) <TBeliefSet, TQuery>

The effect of the action.

**query** [Func](#) <TBeliefSet, TQuery>

The query of the action.

# Fields

## `_effect`

Gets or sets the effect of the action.

```
protected readonly Action<TBeliefSet, TQuery> _effect
```

Field Value

[Action](#)  <TBeliefSet, TQuery>

## `_query`

Gets or sets the query of the action.

```
protected readonly Func<TBeliefSet, TQuery?> _query
```

Field Value

[Func](#)  <TBeliefSet, TQuery>

## `_storedQueryResult`

Gets or sets the result of the query.

```
protected TQuery? _storedQueryResult
```

Field Value

TQuery

# Methods

## `Execute(TBeliefSet)`

Executes the action on the specified belief set.

```
public override void Execute(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set on which the action is executed.

## Query(TBeliefSet)

Queries the environment for the queried item and returns whether the query is not null.

```
public bool Query(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

## Returns

[bool](#) 

True if the query is not null; otherwise, false.

# Namespace Aplib.Core.Intent.Tactics

## Classes

### [FirstOfTactic<TBeliefSet>](#)

Represents a tactic that executes the first enabled action from a list of subtactics.

### [PrimitiveTactic<TBeliefSet>](#)

Represents a primitive tactic

### [RandomTactic<TBeliefSet>](#)

Represents a tactic that executes a random tactic from the provided subtactics.

### [Tactic<TBeliefSet>](#)

Tactics are the real meat of [Goal<TBeliefSet>](#)s, as they define how the agent can approach the goal in hopes of finding a solution which makes the Goal's heuristic function evaluate to being completed. A tactic represents a smart combination of [Action<TBeliefSet>](#)s, which are executed in a Belief Desire Intent Cycle.

## Interfaces

### [ITactic<TBeliefSet>](#)

Represents a tactic that an agent can use to achieve its goals. A tactic is a strategy for achieving a particular goal.

# Class FirstOfTactic<TBeliefSet>

Namespace: [Aplib.Core.Intent.Tactics](#)

Assembly: Aplib.Core.dll

Represents a tactic that executes the first enabled action from a list of subtactics.

```
public class FirstOfTactic<TBeliefSet> : Tactic<TBeliefSet>, ITactic<TBeliefSet>,
    ILoggable, IDocumented where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**








## Inheritance

[object](#)  ← [Tactic](#)<TBeliefSet> ← FirstOfTactic<TBeliefSet>

## Implements

[ITactic](#)<TBeliefSet>, [ILoggable](#), [IDocumented](#)

## Inherited Members

[Tactic<TBeliefSet>.\\_guard](#) , [Tactic<TBeliefSet>.Metadata](#) ,  
[Tactic<TBeliefSet>.IsActionable\(TBeliefSet\)](#) , [object.Equals\(object\)](#)  ,  
[object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  ,  
[object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

### FirstOfTactic(IMetadata, params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public FirstOfTactic(IMetadata metadata, params ITactic<TBeliefSet>[] subtactics)
```

## Parameters

**metadata** [IMetadata](#)



Metadata about this tactic, used to quickly display the tactic in several contexts.

**subtactics** [ITactic](#)<TBeliefSet>[]

The list of subtactics.

## FirstOfTactic(IMetadata, Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public FirstOfTactic(IMetadata metadata, Predicate<TBeliefSet> guard, params
ITactic<TBeliefSet>[] subtactics)
```

### Parameters

**metadata** [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

**guard** [Predicate](#)<TBeliefSet>

The guard condition.

**subtactics** [ITactic](#)<TBeliefSet>[]

The list of subtactics.

## FirstOfTactic(params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public FirstOfTactic(params ITactic<TBeliefSet>[] subtactics)
```

### Parameters

**subtactics** [ITactic](#)<TBeliefSet>[]

The list of subtactics.

## FirstOfTactic(Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified subtactics and guard condition.

```
public FirstOfTactic(Predicate<TBeliefSet> guard, params ITactic<TBeliefSet>
[] subtactics)
```

### Parameters

**guard** [Predicate](#) <TBeliefSet>

The guard condition.

**subtactics** [ITactic](#) <TBeliefSet>[]

The list of subtactics.

## Fields

### \_subtactics

Gets or sets the subtactics of the tactic.

```
protected readonly LinkedList<ITactic<TBeliefSet>> _subtactics
```

### Field Value

[LinkedList](#) <[ITactic](#) <TBeliefSet>>

## Methods

### GetAction(TBeliefSet)

Gets the first enabled action of the tactic.

```
public override IAction<TBeliefSet>? GetAction(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

## Returns

[IAction](#)<TBeliefSet>

A concrete [IAction<TBeliefSet>](#) that the tactic can perform, or null if no actions are enabled.

## GetLogChildren()

Gets the children of the loggable object.

```
public override IEnumerable<ILoggable> GetLogChildren()
```

## Returns

[IEnumerable](#)<sup>↗</sup><[ILoggable](#)>

The children of the loggable object.

# Interface ITactic<TBeliefSet>

Namespace: [Aplib.Core.Intent.Tactics](#)

Assembly: Aplib.Core.dll

Represents a tactic that an agent can use to achieve its goals. A tactic is a strategy for achieving a particular goal.

```
public interface ITactic<in TBeliefSet> where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The type of the belief set that the tactic uses.

## Methods

### GetAction(TBeliefSet)

Gets the first enabled action of the tactic.

```
IAction<in TBeliefSet>? GetAction(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

## Returns

[IAction](#)<TBeliefSet>

A concrete [IAction](#)<TBeliefSet> that the tactic can perform, or null if no actions are enabled.

### IsActionable(TBeliefSet)

Determines whether the tactic is actionable.

```
bool IsActionable(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

## Returns

[bool](#) 

True if the tactic is actionable, false otherwise.

# Class PrimitiveTactic<TBeliefSet>

Namespace: [Aplib.Core.Intent.Tactics](#)

Assembly: Aplib.Core.dll

Represents a primitive tactic

```
public class PrimitiveTactic<TBeliefSet> : Tactic<TBeliefSet>, ITactic<TBeliefSet>,
    ILoggable, IDocumented where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.








## Inheritance

[object](#)  ← [Tactic](#)<TBeliefSet> ← PrimitiveTactic<TBeliefSet>

## Implements

[ITactic](#)<TBeliefSet>, [ILoggable](#), [IDocumented](#)

## Inherited Members

[Tactic<TBeliefSet>.\\_guard](#) , [Tactic<TBeliefSet>.Metadata](#) ,  
[Tactic<TBeliefSet>.IsActionable\(TBeliefSet\)](#) , [object.Equals\(object\)](#)  ,  
[object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  ,  
[object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

### PrimitiveTactic(IMetadata, IAction<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IMetadata metadata, IAction<TBeliefSet> action)
```

## Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

action [IAction](#)<TBeliefSet>

The action of the primitive tactic.

## PrimitiveTactic(IMetadata, IAction<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IMetadata metadata, IAction<TBeliefSet> action,
    Predicate<TBeliefSet> guard)
```

### Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

action [IAction](#)<TBeliefSet>

The action of the primitive tactic.

guard [Predicate](#) <TBeliefSet>

The guard of the primitive tactic.

## PrimitiveTactic(IMetadata, IQueryable<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IMetadata metadata, IQueryable<TBeliefSet> queryAction)
```

### Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

queryAction [IQueryable](#)<TBeliefSet>

The queryable action of the primitive tactic.

## PrimitiveTactic(IMetadata, IQueryable<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IMetadata metadata, IQueryable<TBeliefSet> queryAction,
    Predicate<TBeliefSet> guard)
```

### Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

queryAction [IQueryable](#)<TBeliefSet>

The queryable action of the primitive tactic.

guard [Predicate](#)<TBeliefSet>

The guard of the primitive tactic.

## PrimitiveTactic(IAction<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IAction<TBeliefSet> action)
```

### Parameters



**action** [IAction](#)<TBeliefSet>

The action of the primitive tactic.

## PrimitiveTactic(IAction<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IAction<TBeliefSet> action, Predicate<TBeliefSet> guard)
```

### Parameters

**action** [IAction](#)<TBeliefSet>

The action of the primitive tactic.

**guard** [Predicate](#)<TBeliefSet>

The guard of the primitive tactic.

## PrimitiveTactic(IQueryable<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IQueryable<TBeliefSet> queryAction)
```

### Parameters

**queryAction** [IQueryable](#)<TBeliefSet>

The queryable action of the primitive tactic.

## PrimitiveTactic(IQueryable<TBeliefSet>, Predicate<TBeliefSet>)

Initializes a new instance of the [PrimitiveTactic<TBeliefSet>](#) class with the specified action and guard.

```
public PrimitiveTactic(IQueryable<TBeliefSet> queryAction,  
    Predicate<TBeliefSet> guard)
```

## Parameters

**queryAction** [IQueryable](#)<TBeliefSet>

The queryable action of the primitive tactic.

**guard** [Predicate](#)<TBeliefSet>

The guard of the primitive tactic.

## Fields

### **\_action**

Gets the action of the primitive tactic.

```
protected readonly IAction<TBeliefSet> _action
```

## Field Value

[IAction](#)<TBeliefSet>

## Methods

### **GetAction(TBeliefSet)**

Gets the first enabled action of the tactic.

```
public override IAction<TBeliefSet>? GetAction(TBeliefSet beliefSet)
```

## Parameters

`beliefSet` TBeliefSet

Returns

[IAction](#)<TBeliefSet>

A concrete [IAction<TBeliefSet>](#) that the tactic can perform, or null if no actions are enabled.

## GetLogChildren()

Gets the children of the loggable object.

```
public override IEnumerable<ILoggable> GetLogChildren()
```

Returns

[IEnumerable](#)<sup>↗</sup><[ILoggable](#)>

The children of the loggable object.

# Class RandomTactic<TBeliefSet>

Namespace: [Aplib.Core.Intent.Tactics](#)

Assembly: Aplib.Core.dll

Represents a tactic that executes a random tactic from the provided subtactics.

```
public class RandomTactic<TBeliefSet> : Tactic<TBeliefSet>, ITactic<TBeliefSet>,
    ILoggable, IDocumented where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**








## Inheritance

[object](#)  ← [Tactic](#)<TBeliefSet> ← RandomTactic<TBeliefSet>

## Implements

[ITactic](#)<TBeliefSet>, [ILoggable](#), [IDocumented](#)

## Inherited Members

[Tactic](#)<TBeliefSet>.\_guard, [Tactic](#)<TBeliefSet>.Metadata, [Tactic](#)<TBeliefSet>.IsActionable(TBeliefSet), [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Constructors

### RandomTactic(IMetadata, params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public RandomTactic(IMetadata metadata, params ITactic<TBeliefSet>[] subtactics)
```

## Parameters

**metadata** [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

**subtactics** [ITactic](#)<TBeliefSet>[]

The list of subtactics.

## RandomTactic(IMetadata, Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public RandomTactic(IMetadata metadata, Predicate<TBeliefSet> guard, params  
ITactic<TBeliefSet>[] subtactics)
```

### Parameters

**metadata** [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

**guard** [Predicate](#)<TBeliefSet>

The guard condition.

**subtactics** [ITactic](#)<TBeliefSet>[]

The list of subtactics.

## RandomTactic(params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public RandomTactic(params ITactic<TBeliefSet>[] subtactics)
```

### Parameters

**subtactics** [ITactic](#)<TBeliefSet>[]

The list of subtactics.

## RandomTactic(Predicate<TBeliefSet>, params ITactic<TBeliefSet>[])

Initializes a new instance of the [RandomTactic<TBeliefSet>](#) class with the specified subtactics and an optional guard condition.

```
public RandomTactic(Predicate<TBeliefSet> guard, params ITactic<TBeliefSet>
[] subtactics)
```

### Parameters

**guard** [Predicate](#) <TBeliefSet>

The guard condition.

**subtactics** [ITactic](#) <TBeliefSet>[]

The list of subtactics.

## Fields

### \_subtactics

Gets or sets the subtactics of the tactic.

```
protected readonly LinkedList<ITactic<TBeliefSet>> _subtactics
```

### Field Value

[LinkedList](#) <[ITactic](#) <TBeliefSet>>

## Methods

### GetAction(TBeliefSet)

Gets the first enabled action of the tactic.

```
public override IAction<TBeliefSet>? GetAction(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

## Returns

[IAction](#)<TBeliefSet>

A concrete [IAction<TBeliefSet>](#) that the tactic can perform, or null if no actions are enabled.

## GetLogChildren()

Gets the children of the loggable object.

```
public override IEnumerable<ILoggable> GetLogChildren()
```

## Returns

[IEnumerable](#)<sup>↗</sup><[ILoggable](#)>

The children of the loggable object.

# Class `Tactic<TBeliefSet>`

Namespace: [Aplib.Core.Intent.Tactics](#)

Assembly: Aplib.Core.dll

Tactics are the real meat of [Goal<TBeliefSet>](#)s, as they define how the agent can approach the goal in hopes of finding a solution which makes the Goal's heuristic function evaluate to being completed. A tactic represents a smart combination of [Action<TBeliefSet>](#)s, which are executed in a Belief Desire Intent Cycle.

```
public abstract class Tactic<TBeliefSet> : ITactic<TBeliefSet>, ILoggable,
IDocumented where TBeliefSet : IBeliefSet
```

## Type Parameters

`TBeliefSet`

The belief set of the agent.

## Inheritance

[object](#)  ← `Tactic<TBeliefSet>`








## Implements

[ITactic<TBeliefSet>](#), [ILoggable](#), [IDocumented](#)

## Derived

[FirstOfTactic<TBeliefSet>](#), [PrimitiveTactic<TBeliefSet>](#), [RandomTactic<TBeliefSet>](#)

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Constructors

### `Tactic()`

Initializes a new instance of the [object](#)  class.



```
protected Tactic()
```

## Tactic(IMetadata)

```
protected Tactic(IMetadata metadata)
```

### Parameters

metadata [IMetadata](#)

## Tactic(IMetadata, Predicate<TBeliefSet>)

Initializes a new instance of the [Tactic<TBeliefSet>](#) class with a specified guard.

```
protected Tactic(IMetadata metadata, Predicate<TBeliefSet> guard)
```

### Parameters

metadata [IMetadata](#)

Metadata about this tactic, used to quickly display the tactic in several contexts.

guard [Predicate](#) <TBeliefSet>

The guard of the tactic.

## Tactic(Predicate<TBeliefSet>)

Initializes a new instance of the [Tactic<TBeliefSet>](#) class with a specified guard.

```
protected Tactic(Predicate<TBeliefSet> guard)
```

### Parameters

guard [Predicate](#) <TBeliefSet>

The guard of the tactic.

## Fields

### \_guard

Gets or sets the guard of the tactic.

```
protected readonly Predicate<TBeliefSet> _guard
```

Field Value

[Predicate](#) <TBeliefSet>

## Properties

### Metadata

Gets the metadata of the instance.

```
public IMetadata Metadata { get; }
```

Property Value

[IMetadata](#)

## Methods

### GetAction(TBeliefSet)

Gets the first enabled action of the tactic.

```
public abstract IAction<TBeliefSet>? GetAction(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

Returns

[IAction](#)<TBeliefSet>

A concrete [IAction<TBeliefSet>](#) that the tactic can perform, or null if no actions are enabled.

## GetLogChildren()

Gets the children of the loggable object.

```
public abstract IEnumerable<ILoggable> GetLogChildren()
```

Returns

[IEnumerable](#)<sup>↗</sup> <[ILoggable](#)>

The children of the loggable object.

## IsActionable(TBeliefSet)

Determines whether the tactic is actionable.

```
public virtual bool IsActionable(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

Returns

[bool](#)<sup>↗</sup>

True if the tactic is actionable, false otherwise.

# Operators

## implicit operator `Tactic<TBeliefSet>` `(Action<TBeliefSet>)`

Implicitly lifts an action into a tactic.

```
public static implicit operator Tactic<TBeliefSet>(Action<TBeliefSet> action)
```

## Parameters

`action` [Action](#)<TBeliefSet>

The action which on its own can function as a tactic. Meaning, the tactic consists of just a single action.

## Returns

[Tactic](#)<TBeliefSet>

The most logically matching tactic, wrapping around `action`.

## See Also

[Goal](#)<TBeliefSet>

[Action](#)<TBeliefSet>

# Namespace Aplib.Core.Logging

## Classes

### [LogNode](#)

Represents a node in the log tree.

## Interfaces

### [ILoggable](#)

An interface that allows defining the structure of loggable objects through a [LogNode](#) tree. The only method that needs to be implemented in loggable classes is the [GetLogChildren\(\)](#) method; the structure of the log tree is generated automatically when calling the [GetLogTree\(int\)](#) method.

# Interface ILoggable

Namespace: [Aplib.Core.Logging](#)

Assembly: Aplib.Core.dll

An interface that allows defining the structure of loggable objects through a [LogNode](#) tree. The only method that needs to be implemented in loggable classes is the [GetLogChildren\(\)](#) method; the structure of the log tree is generated automatically when calling the [GetLogTree\(int\)](#) method.

```
public interface ILoggable : IDocumented
```

## Inherited Members

[IDocumented.Metadata](#)

## Methods

### GetLogChildren()

Gets the children of the loggable object.

```
IEnumerable<ILoggable> GetLogChildren()
```

### Returns

[IEnumerable](#)  <[ILoggable](#)>

The children of the loggable object.

### GetLogTree(int)

Generates a log tree of the loggable object.

```
LogNode GetLogTree(int depth = 0)
```

## Parameters

depth [int](#)

The depth of this node in the log tree.

Returns

[LogNode](#)

The root node of the log tree.

# Class LogNode

Namespace: [Aplib.Core.Logging](#)

Assembly: Aplib.Core.dll








Represents a node in the log tree.

```
public class LogNode
```

## Inheritance

[object](#)  ← LogNode

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Constructors

### LogNode(ILoggable, int)

```
public LogNode(ILoggable loggable, int depth)
```

## Parameters

loggable [ILoggable](#)

depth [int](#) 

### LogNode(ILoggable, int, List<LogNode>)

Initialize a new [LogNode](#) from a given loggable object, the depth, and optionally a list of children.

```
public LogNode(ILoggable loggable, int depth, List<LogNode> children)
```



## Parameters

**loggable** [ILoggable](#)

The loggable object of the node.

**depth** [int](#)

The depth at which this node resides.

**children** [List](#) <[LogNode](#)>

The children of the node. It is assumed the children have a correct depth set. If omitted, an empty list will be used.

## Properties

### Children

The children of the node.

```
public List<LogNode> Children { get; }
```

### Property Value

[List](#) <[LogNode](#)>

### Depth

The depth at which this node resides.

```
public int Depth { get; }
```

### Property Value

[int](#)

## Loggable

The loggable object of the node.

```
public ILoggable Loggable { get; }
```

Property Value

[ILoggable](#)