

# Namespace Aplib.Core

## Classes

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

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

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

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

### [Metadata](#)

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

## Interfaces

### [IAgent](#)

Defines an agent that can play a game.

### [ICompletable](#)

Defines an object that can be completed.

## Enums

### [CompletionStatus](#)

Represents the state of a completable object.

# Class BdiAgent<TBeliefSet>

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

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

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

## Type Parameters

**TBeliefSet**

## Inheritance

[object](#)  ← BdiAgent<TBeliefSet>

## Implements

[IAgent](#)

## 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 status of the agent.

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

## Property Value

[CompletionStatus](#)

## Remarks

This reflects whether the agent has achieved or failed its goals.

# 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.

# Class CircularArray<T>

Namespace: [Aplib.Core](#)

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

# 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 IAgent

Namespace: [Aplib.Core](#)

Assembly: Aplib.Core.dll

Defines an agent that can play a game.

```
public interface IAgent
```

## Properties

### Status

Gets the status of the agent.

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

### Property Value

[CompletionStatus](#)

### Remarks

This reflects whether the agent has achieved or failed its goals.

## Methods

### Update()

Updates the agent's state and goals.

```
void Update()
```

### Remarks

This method will get called every frame of the game.

# 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](#)

# 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
```

## Inheritance

[object](#)  ← Metadata

## 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 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 component during debugging, logging, or general overviews.

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

### Property Value

[string](#)

## Id

Gets the unique identifier of the component.

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

### Property Value

[Guid](#)

## Name

Gets the name used to display the component during debugging, logging, or general overviews.

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

### Property Value

[string](#)

# Namespace Aplib.Core.Belief

## 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\(\)](#).

### [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).

### [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.

## Interfaces

### [IBelief](#)

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

### [IBeliefSet](#)

A belief set defines beliefs for an agent.

# Class BeliefSet

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

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()
```

# Class Belief<TReference, TObservation>

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

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
```

## Type Parameters

### TReference

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

### TObservation

The type of the observation that the belief represents.

## Inheritance

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

## Implements

[IBelief](#)

## Derived

[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 implements the [IBelief](#) interface. It supports implicit conversion to **TObservation**.

## Constructors



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

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

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

### Parameters

**reference** TReference

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

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

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

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

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, Func<bool> shouldUpdate)
```

### Parameters

**reference** TReference

The object reference used to generate/update the observation.

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

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

**shouldUpdate** [Func](#)<[bool](#)>

A condition on when the observation should be updated.

# 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()
```

# 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](#)

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()
```

# Interface IBeliefSet

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

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()
```

# Class MemoryBelief<TReference, TObservation>

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

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
```

## Type Parameters

### TReference

The type of the reference used to generate/update the observation.

### TObservation

The type of the observation the belief represents.

## Inheritance

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

## Implements

[IBelief](#)

## Inherited Members

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

## Remarks

It implements the [IBelief](#) interface. It supports implicit conversion to `TObservation`.

# Constructors

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

Initializes a new instance of the [MemoryBelief<TReference, TObservation>](#) class with an object reference, and a function to generate/update the observation using the object reference. 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.

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

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

**framesToRemember** [int](#)

The number of frames to remember back.

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

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, Func<bool> shouldUpdate)
```

### Parameters

**reference** TReference

The reference used to generate/update the observation.

`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` [Func](#) <[bool](#)>

A function that sets a condition on when the observation should be updated.

## 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. If the index is out of bounds, returns the element closest to the index that is in bounds.

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

#### Parameters

`index` [int](#)

`clamp` [bool](#)

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()
```



# Namespace Aplib.Core.Desire

## Classes

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

Supports all classes in the .NET class hierarchy and provides low-level services to derived classes. This is the ultimate base class of all .NET classes; it is the root of the type hierarchy.

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

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

### [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.

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

Represents a sequential goal structure.

## 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.

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

Represents a goal structure.

# Class DesireSet<TBeliefSet>

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

Assembly: Aplib.Core.dll

Supports all classes in the .NET class hierarchy and provides low-level services to derived classes. This is the ultimate base class of all .NET classes; it is the root of the type hierarchy.

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

## Type Parameters

**TBeliefSet**

### Inheritance

[object](#)  ← DesireSet<TBeliefSet>

### Implements

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

### 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>)

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

```
public DesireSet(IGoalStructure<TBeliefSet> mainGoal)
```

## Parameters

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

The main goal structure that the agent needs to complete.

## Properties

### Status

Gets the completion status of the object.

```
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.

### UpdateStatus(TBeliefSet)

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

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

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Class FirstOfGoalStructure<TBeliefSet>

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

Assembly: Aplib.Core.dll

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

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

## Type Parameters

**TBeliefSet**

The beliefset of the agent.








## Inheritance

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

## Implements

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

## Inherited Members

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

## 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.

## 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.

## UpdateStatus(TBeliefSet)

Updates the state of the goal structure.

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

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Class GoalStructure<TBeliefSet>

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

Assembly: Aplib.Core.dll

Describes a structure of goals that need to be fulfilled.

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

## Type Parameters

**TBeliefSet**

### Inheritance

[object](#)  ← GoalStructure<TBeliefSet>

### Implements

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

### 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\(\)](#) 

## Constructors

### 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

### 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.

### UpdateStatus(TBeliefSet)

Updates the state of the goal structure.

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

#### Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Interface IDesireSet<TBeliefSet>

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

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.

# UpdateStatus(TBeliefSet)

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

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

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Interface IGoalStructure<TBeliefSet>

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

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](#)

## 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.

## 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](#)

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 where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The beliefset of the agent.

## Inheritance

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

## Implements

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

## Inherited Members

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

## 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.

## 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.

### UpdateStatus(TBeliefSet)

Updates the state of the goal structure.

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

## Parameters

**beliefSet** TBeliefSet



The belief set of the agent.

# Class RepeatGoalStructure<TBeliefSet>

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

Assembly: Aplib.Core.dll

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

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

## Type Parameters

**TBeliefSet**

The beliefset of the agent.








## Inheritance

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

## Implements

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

## Inherited Members

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

## Remarks

This structure will repeatedly execute the goal it was created with until the goal is finished.

## Constructors

### RepeatGoalStructure(IGoalStructure<TBeliefSet>)

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

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

## Parameters

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

The goalstructure to repeat

## 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.

### UpdateStatus(TBeliefSet)

Updates the state of the goal structure.

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

## Parameters

`beliefSet` TBeliefSet

The belief set of the agent.

# Class

## SequentialGoalStructure<TBeliefSet>

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

Assembly: Aplib.Core.dll

Represents a sequential goal structure.

```
public class SequentialGoalStructure<TBeliefSet> : GoalStructure<TBeliefSet>,
    IGoalStructure<TBeliefSet>, ICompletable, 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](#), [IDisposable](#) 

### Inherited Members

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

### Remarks

This class is a specific type of goal structure where goals are processed sequentially. All goals must be completed in order for the goal structure to be completed.

### 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.

## 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.

## UpdateStatus(TBeliefSet)

Updates the state of the goal structure.

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

## Parameters

**beliefSet** TBeliefSet

The belief set of the agent.

# Namespace Aplib.Core.Desire.Goals

## Classes

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

Contains helper methods to generate commonly used heuristic functions.

### [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.

### [Heuristics](#)

Contains all information on how close the associated state is to its goal. Can be used to optimise search algorithms.

## Interfaces

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

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

## Delegates

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

The abstract definition of what it means to test the Goal's heuristic function. Returns [Heuristics](#), as they represent how close we are to matching the heuristic function, and if the goal is completed.

# Class

## CommonHeuristicFunctions<TBeliefSet>

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

Assembly: Aplib.Core.dll

Contains helper methods to generate commonly used heuristic functions.

```
public static class CommonHeuristicFunctions<TBeliefSet> where TBeliefSet
: IBeliefSet
```

### Type Parameters

**TBeliefSet**

### Inheritance

[object](#)  ← CommonHeuristicFunctions<TBeliefSet>

### Inherited Members

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

## Methods

### Boolean(Func<TBeliefSet, bool>)

Converts a boolean-based heuristic function to a [Goal<TBeliefSet>.HeuristicFunction](#).

```
public static Goal<TBeliefSet>.HeuristicFunction Boolean(Func<TBeliefSet,
bool> heuristicFunction)
```

### Parameters

**heuristicFunction** [Func](#)  <TBeliefSet, [bool](#)  >

A heuristic function which returns true only when the state is considered completed.



Returns

[Goal<TBeliefSet>.HeuristicFunction](#)

A heuristic function which wraps around the boolean-based heuristic function.

## Completed()

Returns a heuristic function which always, at all times, and forever, returns a value indicating the state can be seen as completed.

```
public static Goal<TBeliefSet>.HeuristicFunction Completed()
```

Returns

[Goal<TBeliefSet>.HeuristicFunction](#)

Said heuristic function.

## Constant(float)

A [Goal<TBeliefSet>.HeuristicFunction](#) which always returns [Heuristics](#) with the same distance.

```
public static Goal<TBeliefSet>.HeuristicFunction Constant(float distance)
```

Parameters

distance [float](#)

The distance which the heuristic function must always return.

Returns

[Goal<TBeliefSet>.HeuristicFunction](#)

## Uncompleted()

Returns a heuristic function which always, at all times, and forever, returns a value indicating the state can be seen as NOT completed.

```
public static Goal<TBeliefSet>.HeuristicFunction Uncompleted()
```

Returns

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

Said heuristic function.

# 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 where TBeliefSet
    : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.

## Inheritance

[object](#)  ← Goal<TBeliefSet>

## Implements

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

## Inherited Members

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

## Constructors

Goal(ITactic<TBeliefSet>, HeuristicFunction, double, Metadata?)

Creates a new goal which works with [Heuristics](#).

```
public Goal(ITactic<TBeliefSet> tactic, Goal<TBeliefSet>.HeuristicFunction
    heuristicFunction, double epsilon = 0.005, Metadata? metadata = null)
```

## Parameters

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

The tactic used to approach this goal.

**heuristicFunction** [Goal](#)<TBeliefSet>.[HeuristicFunction](#)

The heuristic function which defines whether a goal is reached

**epsilon** [double](#) 

The goal is considered to be completed, when the distance of the [DetermineCurrentHeuristics\(TBeliefSet\)](#) is below this value.

**metadata** [Metadata](#)

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

## Goal(ITactic<TBeliefSet>, Func<TBeliefSet, bool>, double, Metadata?)

Creates a new goal which works with boolean-based [Heuristics](#).

```
public Goal(ITactic<TBeliefSet> tactic, Func<TBeliefSet, bool> predicate, double  
epsilon = 0.005, Metadata? metadata = null)
```

## Parameters

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

The tactic used to approach this goal.

**predicate** [Func](#)  <TBeliefSet, [bool](#)  >

The heuristic function (or specifically predicate) which defines whether a goal is reached

**epsilon** [double](#) 

The goal is considered to be completed, when the distance of the [DetermineCurrentHeuristics\(TBeliefSet\)](#) is below this value.

**metadata** [Metadata](#)

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

## Fields

### heuristicFunction

The concrete implementation of this Goal's [Goal<TBeliefSet>.HeuristicFunction](#). Used to test whether this goal is completed.

```
protected Goal<TBeliefSet>.HeuristicFunction _heuristicFunction
```

### Field Value

[Goal<TBeliefSet>.HeuristicFunction](#)

### See Also

[GetStatus\(TBeliefSet\)](#)

## Properties

### Metadata

Gets the metadata of the goal.

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

### Property Value

[Metadata](#)

## Status

Gets the completion status of the object.

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

Property Value

[CompletionStatus](#)

## Tactic

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

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

Property Value

[ITactic<TBeliefSet>](#)

## \_epsilon

The goal is considered to be completed, when the distance of the [DetermineCurrentHeuristics\(TBeliefSet\)](#) is below this value.

```
protected double _epsilon { get; }
```

Property Value

[double](#)

## Methods

### DetermineCurrentHeuristics(TBeliefSet)

Gets the [Heuristics](#) of the current state of the game.

```
public virtual Heuristics DetermineCurrentHeuristics(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

Returns

[Heuristics](#)

Remarks

If no heuristics have been calculated yet, they will be calculated first.

## GetStatus(TBeliefSet)

Tests whether the goal has been achieved, bases on the [\\_heuristicFunction](#) and the [DetermineCurrentHeuristics\(TBeliefSet\)](#). When the distance of the heuristics is smaller than [\\_epsilon](#), the goal is considered to be completed.

```
public virtual CompletionStatus GetStatus(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

Returns

[CompletionStatus](#)

An enum representing whether the goal is complete and if so, with what result.

**See Also**

[\\_epsilon](#)

## See Also

[GoalStructure](#)<TBeliefSet>

# Delegate Goal<TBeliefSet>.HeuristicFunction

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

Assembly: Aplib.Core.dll

The abstract definition of what is means to test the Goal's heuristic function. Returns [Heuristics](#), as they represent how close we are to matching the heuristic function, and if the goal is completed.

```
public delegate Heuristics Goal<TBeliefSet>.HeuristicFunction(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The abstract definition of what is means to test the Goal's heuristic function. Returns , as they represent how close we are to matching the heuristic function, and if the goal is completed.

## Returns

[Heuristics](#)

The abstract definition of what is means to test the Goal's heuristic function. Returns , as they represent how close we are to matching the heuristic function, and if the goal is completed.

## See Also

[GetStatus](#)(TBeliefSet)



# Class Heuristics

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

Assembly: Aplib.Core.dll








Contains all information on how close the associated state is to its goal. Can be used to optimise search algorithms.

```
public class Heuristics
```

## Inheritance

[object](#)  ← Heuristics

## Inherited Members

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

## Properties

### Distance

The logical distance the current state is to its goal.

```
public float Distance { get; set; }
```

### Property Value

[float](#) 

## Methods

### Boolean(bool)

Creates a heuristic value representing just a boolean. The heuristic value is considered '0' or 'done' when the boolean is true. Non-zero otherwise.

```
public static Heuristics Boolean(bool value)
```

## Parameters

value [bool](#) 

True if completed, False if not completed.

## Returns

[Heuristics](#)

# 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](#)

## 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

### DetermineCurrentHeuristics(TBeliefSet)

Gets the [Heuristics](#) of the current state of the game.

Heuristics `DetermineCurrentHeuristics`(TBeliefSet beliefSet)

## Parameters

`beliefSet` TBeliefSet

## Returns

[Heuristics](#)

## Remarks

If no heuristics have been calculated yet, they will be calculated first.

## GetStatus(TBeliefSet)

Tests whether the goal has been achieved, based on the [\\_heuristicFunction](#) and the [DetermineCurrentHeuristics\(TBeliefSet\)](#). When the distance of the heuristics is smaller than [\\_epsilon](#) , the goal is considered to be completed.

CompletionStatus `GetStatus`(TBeliefSet beliefSet)

## Parameters

`beliefSet` TBeliefSet

## Returns

[CompletionStatus](#)

An enum representing whether the goal is complete and if so, with what result.

## See Also

[\\_epsilon](#)

# Namespace Aplib.Core.Intent.Actions

## Classes

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

Describes an action that can be executed and guarded.

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

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

## Interfaces

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

Represents an action that can be executed 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> where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.

## Inheritance

[object](#)  ← Action<TBeliefSet>

## Implements

[IAction](#)<TBeliefSet>

## Derived

[GuardedAction](#)<TBeliefSet, TQuery>

## Inherited Members

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

## Constructors

### Action(Metadata?)

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

```
protected Action(Metadata? metadata)
```

## Parameters

metadata [Metadata](#)

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

## Remarks

Only meant for internal use

## Action(Action<TBeliefSet>, Metadata?)

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

```
public Action(Action<TBeliefSet> effect, Metadata? metadata = null)
```

## Parameters

effect [Action](#) <TBeliefSet>

The effect of the action.

metadata [Metadata](#)

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

## Action(Action<TBeliefSet>, Func<TBeliefSet, bool>, Metadata?)

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

```
public Action(Action<TBeliefSet> effect, Func<TBeliefSet, bool> guard, Metadata? metadata = null)
```

## Parameters

effect [Action](#) <TBeliefSet>

The effect of the action.

guard [Func](#) <TBeliefSet, [bool](#)>

The guard of the action.

metadata [Metadata](#)

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

## Properties

### Metadata

Gets the metadata of the action.

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

### Property Value

[Metadata](#)

### \_effect

Gets or sets the effect of the action.

```
protected Action<TBeliefSet> _effect { get; set; }
```

### Property Value

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

### \_guard

Gets or sets the guard of the action.

```
protected Func<TBeliefSet, bool> _guard { get; set; }
```

### Property Value

[Func](#) [<TBeliefSet, bool>](#)



## 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.

### IsActionable(TBeliefSet)

Guard the action against unwanted execution. The result is stored and can be used in the effect.

```
public virtual bool IsActionable(TBeliefSet beliefSet)
```

#### Parameters

**beliefSet** TBeliefSet

The belief set on which the action is executed.

#### Returns

[bool](#)

True if the action is actionable, false otherwise.

# Class GuardedAction<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 the result of the guard. The result can be used in the effect.

```
public class GuardedAction<TBeliefSet, TQuery> : Action<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](#)   $\leftarrow$  [Action](#)  $\leftarrow$  GuardedAction<TBeliefSet, TQuery>

## Implements

[IAction](#)  $\leftarrow$  GuardedAction<TBeliefSet>

## Inherited Members

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

## Constructors

GuardedAction(Action<TBeliefSet, TQuery>, Func<TBeliefSet, TQuery?>, Metadata?)

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

```
public GuardedAction(Action<TBeliefSet, TQuery> effect, Func<TBeliefSet, TQuery?>  
guard, Metadata? metadata = null)
```

## Parameters

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

The effect of the action.

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

The guard of the action.

**metadata** [Metadata](#)

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

## Properties

### effect

Gets or sets the effect of the action.

```
protected Action<TBeliefSet, TQuery> _effect { get; set; }
```

### Property Value

[Action](#) <TBeliefSet, TQuery>

### guard

Gets or sets the guard of the action.

```
protected Func<TBeliefSet, TQuery?> _guard { get; set; }
```

### Property Value

[Func](#) <TBeliefSet, TQuery>

## \_storedGuardResult

Gets or sets the result of the guard.

```
protected TQuery? _storedGuardResult { get; set; }
```

Property 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.

### IsActionable(TBeliefSet)

Guard the action against unwanted execution. The result is stored and can be used in the effect.

```
public override bool IsActionable(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

The belief set on which the action is executed.

Returns

[bool](#) 

True if the action is actionable, false otherwise.

# 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.

## 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.

### IsActionable(TBeliefSet)

Guard the action against unwanted execution. The result is stored and can be used in the effect.

```
bool IsActionable(TBeliefSet beliefSet)
```

## Parameters

**beliefSet** TBeliefSet

The belief set on which the action is executed.

## Returns

[bool](#)

True if the action is actionable, false otherwise.

# Namespace Aplib.Core.Intent.Tactics

## Classes

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

Represents a tactic that executes any of the provided sub-tactics.

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

Represents a tactic that executes the first enabled action from a list of sub-tactics.

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

Represents a primitive tactic

### [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 Believe 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 AnyOfTactic<TBeliefSet>

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

Assembly: Aplib.Core.dll

Represents a tactic that executes any of the provided sub-tactics.

```
public class AnyOfTactic<TBeliefSet> : Tactic<TBeliefSet>, ITactic<TBeliefSet> where
    TBeliefSet : IBeliefSet
```

## Type Parameters

[TBeliefSet](#)

## Inheritance

[object](#)  ← [Tactic](#)<TBeliefSet> ← AnyOfTactic<TBeliefSet>

## Implements

[ITactic](#)<TBeliefSet>

## Derived

[FirstOfTactic](#)<TBeliefSet>

## Inherited Members

[Tactic](#)<TBeliefSet>.Metadata , [Tactic](#)<TBeliefSet>.\_guard ,  
[Tactic](#)<TBeliefSet>.IsActionable(TBeliefSet) , [object.Equals\(object\)](#)  ,  
[object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  ,  
[object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

### AnyOfTactic(Metadata?, params ITactic<TBeliefSet>[])

Initializes a new instance of the [AnyOfTactic<TBeliefSet>](#) class with the specified sub-tactics.

```
public AnyOfTactic(Metadata? metadata = null, params ITactic<TBeliefSet>
    [] subTactics)
```

## Parameters

**metadata** [Metadata](#)

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

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

The list of sub-tactics.

## AnyOfTactic(Func<TBeliefSet, bool>, Metadata?, params ITactic<TBeliefSet>[])

Initializes a new instance of the [AnyOfTactic<TBeliefSet>](#) class with the specified sub-tactics and guard condition.

```
public AnyOfTactic(Func<TBeliefSet, bool> guard, Metadata? metadata = null, params ITactic<TBeliefSet>[] subTactics)
```

## Parameters

**guard** [Func](#)<TBeliefSet, [bool](#)>

The guard condition.

**metadata** [Metadata](#)

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

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

The list of sub-tactics.

## Properties

### \_subTactics

Gets or sets the sub-tactics of the tactic.

```
protected LinkedList<ITactic<TBeliefSet>> _subTactics { get; set; }
```

Property 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.

# 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 sub-tactics.

```
public class FirstOfTactic<TBeliefSet> : AnyOfTactic<TBeliefSet>,
ITactic<TBeliefSet> where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

## Inheritance

[object](#)   $\leftarrow$  [Tactic](#)<TBeliefSet>  $\leftarrow$  [AnyOfTactic](#)<TBeliefSet>  $\leftarrow$  FirstOfTactic<TBeliefSet>

## Implements

[ITactic](#)<TBeliefSet>

## Inherited Members

[AnyOfTactic<TBeliefSet>.\\_subTactics](#) , [Tactic<TBeliefSet>.Metadata](#) ,  
[Tactic<TBeliefSet>.\\_guard](#) , [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(Metadata?, params Tactic<TBeliefSet>[])

Initializes a new instance of the [FirstOfTactic<TBeliefSet>](#) class with the specified sub-tactics.

```
public FirstOfTactic(Metadata? metadata = null, params Tactic<TBeliefSet>
[] subTactics)
```

## Parameters

metadata [Metadata](#)

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

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

The list of sub-tactics.

## FirstOfTactic(Func<TBeliefSet, bool>, Metadata?, params Tactic<TBeliefSet>[])

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

```
public FirstOfTactic(Func<TBeliefSet, bool> guard, Metadata? metadata = null, params  
Tactic<TBeliefSet>[] subTactics)
```

### Parameters

guard [Func](#)<TBeliefSet, [bool](#)>

The guard condition.

metadata [Metadata](#)

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

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

The list of sub-tactics.

## 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.

# 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>  
where TBeliefSet : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.

## Inheritance

[object](#)  ← [Tactic](#)<TBeliefSet> ← PrimitiveTactic<TBeliefSet>

## Implements

[ITactic](#)<TBeliefSet>

## Inherited Members

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

## Constructors

### PrimitiveTactic(IAction<TBeliefSet>, Metadata?)

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

```
public PrimitiveTactic(IAction<TBeliefSet> action, Metadata? metadata = null)
```

## Parameters

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

The action of the primitive tactic.

metadata [Metadata](#)

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

## PrimitiveTactic(IAction<TBeliefSet>, Func<TBeliefSet, bool>, Metadata?)

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

```
public PrimitiveTactic(IAction<TBeliefSet> action, Func<TBeliefSet, bool> guard,
Metadata? metadata = null)
```

### Parameters

action [IAction](#)<TBeliefSet>

The action of the primitive tactic.

guard [Func](#)<TBeliefSet, [bool](#)>

The guard of the primitive tactic.

metadata [Metadata](#)

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

## 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.

### IsActionable(TBeliefSet)

Determines whether the tactic is actionable.

```
public override bool IsActionable(TBeliefSet beliefSet)
```

#### Parameters

**beliefSet** TBeliefSet

#### Returns

[bool](#)<sup>↗</sup>

True if the tactic is actionable, false otherwise.

# 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 Believe Desire Intent Cycle.


```
public abstract class Tactic<TBeliefSet> : ITactic<TBeliefSet> where TBeliefSet
    : IBeliefSet
```

## Type Parameters

**TBeliefSet**

The belief set of the agent.

## Inheritance

[object](#)  ← `Tactic<TBeliefSet>`

## Implements

[ITactic](#)  <TBeliefSet>

## Derived

[AnyOfTactic<TBeliefSet>](#), [PrimitiveTactic<TBeliefSet>](#)

## Inherited Members

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

## Constructors

### `Tactic`(Metadata?)

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

```
protected Tactic(Metadata? metadata)
```

## Parameters

**metadata** [Metadata](#)

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

## Tactic(Func<TBeliefSet, bool>, Metadata?)

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

```
protected Tactic(Func<TBeliefSet, bool> guard, Metadata? metadata = null)
```

## Parameters

**guard** [Func](#) <TBeliefSet, [bool](#)>

The guard of the tactic.

**metadata** [Metadata](#)

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

## Properties

### Metadata

Gets the metadata of the tactic.

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

## Property Value

[Metadata](#)

## \_guard

Gets or sets the guard of the tactic.

```
protected Func<TBeliefSet, bool> _guard { get; set; }
```

Property Value

[Func](#) <[TBeliefSet](#), [bool](#)>

## 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.

### IsActionable(TBeliefSet)

Determines whether the tactic is actionable.

```
public virtual bool IsActionable(TBeliefSet beliefSet)
```

Parameters

**beliefSet** TBeliefSet

## Returns

[bool](#) 

True if the tactic is actionable, false otherwise.

## See Also

[Goal](#)<TBeliefSet>

[Action](#)<TBeliefSet>