Namespace LogicEngine

Classes

CompiledCatalog<T>

Represents a compiled catalog of rules that can be applied to items of type `T`.

CompiledRule<T>

Represents a compiled rule that can be applied to an object of type `T`.

CompiledRulesSet<T>

Represents a compiled set of rules that can be applied to items of type `T`.

Class CompiledCatalog<T>

Namespace: LogicEngine

Assembly: LogicEngine.dll

Represents a compiled catalog of rules that can be applied to items of type `T`.

```
public record CompiledCatalog<T> : IAppliable<T>, IDetailedAppliable<T,
IEnumerable<string>>, IAppliedSelector<T, string>, IEquatable<CompiledCatalog<T>>
where T : new()
```

Type Parameters

`T`

The type of items to which the catalog's rules are applied. Must have a parameterless constructor.

Inheritance

object ← CompiledCatalog<T>

Implements

IAppliable<T>, IDetailedAppliable<T, IEnumerable<string>>, IAppliedSelector<T, string>, IEquatable<CompiledCatalog<T>>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This catalog provides functionality to apply rules to an item, retrieve detailed results of rule application, and identify the first matching rule for a given item. It is constructed from a set of compiled rule sets.

Constructors

CompiledCatalog(CompiledRulesSet<T>[], string)

Represents a compiled catalog of rules, providing functionality to apply rules, retrieve detailed results, and find the first matching rule.

```
public CompiledCatalog(CompiledRulesSet<T>[] ruleSets, string name)
```

Parameters

`ruleSets` CompiledRulesSet<T>[]

An array of compiled rule sets. Each rule set is evaluated and included in the catalog if it contains valid rules.

`name` string

The name of the catalog.

Remarks

The catalog is initialized with the provided rule sets, filtering out any invalid or empty rule sets. If no valid rule sets are provided, default functions are used for applying rules, retrieving detailed results, and finding the first matching rule.

Properties

Name

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

Property Value

string

Methods

Apply(T)

Applies the specified operation to the given item and returns the result.

```
public bool Apply(T item)
```

Parameters

`item` T

The item to which the operation is applied.

Returns

bool

true if the operation succeeds; otherwise, false.

DetailedApply(T)

Applies the specified item and returns either a collection of error messages or a success indicator.

```
public Either<IEnumerable<string>, Unit> DetailedApply(T item)
```

Parameters

`item` T

The item to be applied. Cannot be null.

Returns

Either<IEnumerable<string>, Unit>

An TinyFp.Either<L, R> containing a collection of error messages if the operation fails, or a TinyFp.Unit value if the operation succeeds.

Remarks

Use this method to apply an item and handle potential errors in a detailed manner. The left side of the result contains error messages describing the issues encountered, while the right side indicates a successful operation.

FirstMatching(T)

Finds the first matching string for the specified item.

```
public Option<string> FirstMatching(T item)
```

Parameters

`item` T

The item to match against.

Returns

Option<string>

An TinyFp.Option<A> containing the first matching string if a match is found; otherwise, an empty TinyFp.Option<A>.

Remarks

The method uses the provided item to determine a match and returns the first result. The behavior of the matching logic depends on the implementation of the underlying `_firstMatching` function.

Class CompiledRule<T>

Namespace: LogicEngine

Assembly: LogicEngine.dll

Represents a compiled rule that can be applied to an object of type `T`.

```
public record CompiledRule<T> : IAppliable<T>, IDetailedAppliable<T, string>,
IEquatable<CompiledRule<T>> where T : new()
```

Type Parameters

`T`

The type of object to which the rule can be applied. Must have a parameterless constructor.

Inheritance

object ← CompiledRule<T>

Implements

IAppliable<T>, IDetailedAppliable<T, string>, IEquatable<CompiledRule<T>>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This class provides functionality to evaluate a rule against an object of type `T` and optionally retrieve detailed information about the evaluation result.

Constructors

CompiledRule(Func<T, bool>, string)

Represents a compiled rule that can be executed against a specified input.

```
public CompiledRule(Func<T, bool> executable, string code)
```

Parameters

`executable` Func<T, bool>

A function that defines the rule logic and determines whether the input satisfies the rule.

`code` string

A string representing the unique code or identifier for the rule.

Properties

Code

The code that represents the rule

```
public string Code { get; }
```

Property Value

string

Methods

Apply(T)

Applies the specified condition or operation to the given item.

```
public bool Apply(T item)
```

Parameters

`item` T

The item to which the condition or operation is applied.

Returns

bool

true if the condition or operation succeeds; otherwise, false.

DetailedApply(T)

Applies the specified operation to the given item and returns the result as an TinyFp.Either<L, R>.

```
public Either<string, Unit> DetailedApply(T item)
```

Parameters

`item` T

The item to which the operation will be applied.

Returns

Either<string, Unit>

An TinyFp.Either<L, R> containing a string with an error message if the operation fails, or a TinyFp.Unit value if the operation succeeds.

Class CompiledRulesSet<T>

Namespace: LogicEngine

Assembly: LogicEngine.dll

Represents a compiled set of rules that can be applied to items of type `T`.

```
public record CompiledRulesSet<T> : IAppliable<T>, IDetailedAppliable<T,
IEnumerable<string>>, IAppliedSelector<T, string>, IEquatable<CompiledRulesSet<T>>
where T : new()
```

Type Parameters

`T`

The type of items to which the rules are applied. Must have a parameterless constructor.

Inheritance

object ← CompiledRulesSet<T>

Implements

IAppliable<T>, IDetailedAppliable<T, IEnumerable<string>>,
IAppliedSelector<T, string>, IEquatable<CompiledRulesSet<T>>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() ,
object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) ,
object.ToString()

Remarks

This class provides functionality to apply a set of compiled rules to an item, check detailed results of rule application, and retrieve the first matching rule. It implements the IAppliable<T>, IDetailedAppliable<T, TOut>, and IAppliedSelector<TIn, TOut> interfaces.

Constructors

CompiledRulesSet(CompiledRule<T>[], string)

Represents a compiled set of rules that can be applied to evaluate conditions and retrieve results.

```
public CompiledRulesSet(CompiledRule<T>[] rules, string name)
```

Parameters

`rules` CompiledRule<T>[]

An array of compiled rules to include in the set. Must not be empty. If empty, default functions are used.

`name` string

The name of the compiled rules set. This is used to identify the set.

Remarks

The CompiledRulesSet<T> class initializes a set of rules that can be applied to input data. If the provided `rules` array is empty, default functions are used for rule evaluation.

Properties

Name

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

Property Value

string

Methods

Apply(T)

Applies the specified operation to the given item and returns the result.

```
public bool Apply(T item)
```

Parameters

`item` T

The item to which the operation is applied.

Returns

bool

true if the operation succeeds; otherwise, false.

DetailedApply(T)

Applies the specified operation to the given item and returns either a collection of error messages or a success indicator.

```
public Either<IEnumerable<string>, Unit> DetailedApply(T item)
```

Parameters

`item` T

The item to which the operation is applied. Cannot be null.

Returns

Either<IEnumerable<string>, Unit>

An TinyFp.Either<L, R> containing a collection of error messages if the operation fails, or a TinyFp.Unit value indicating success if the operation completes successfully.

Remarks

Use this method to perform an operation on the provided item while capturing detailed error information in case of failure. The returned TinyFp.Either<L, R> allows callers to handle success and failure cases explicitly.

FirstMatching(T)

Finds the first matching string for the specified item.

```
public Option<string> FirstMatching(T item)
```

Parameters

`item` T

The item to match against.

Returns

Option<string>

An TinyFp.Option<A> containing the first matching string if a match is found; otherwise, an empty TinyFp.Option<A>.

Remarks

The method uses the provided item to determine a match and returns the first result. If no match is found, the returned TinyFp.Option<A> will be empty.

Namespace LogicEngine.Compilers

Classes

RuleCompiler

RulesCatalogCompiler

RulesSetCompiler

Class RuleCompiler

Namespace: LogicEngine.Compilers

Assembly: LogicEngine.dll

```
public class RuleCompiler : IRuleCompiler
```

Inheritance

object ← RuleCompiler

Implements

IRuleCompiler

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Compile<T>(Rule)

Compiles a rule into a strongly-typed, executable representation.

```
public Option<CompiledRule<T>> Compile<T>(Rule rule) where T : new()
```

Parameters

`rule` Rule

The rule to compile. This must be a valid rule that can be transformed into a compiled representation.

Returns

Option<CompiledRule<T>>

An TinyFp.Option<A> containing a CompiledRule<T> if the compilation succeeds; otherwise, an empty option if the rule cannot be compiled.

Type Parameters

T

The type of the object that the compiled rule will evaluate. Must have a parameterless constructor.

Remarks

The compiled rule includes both a delegate for evaluating the rule and the original code representation.

Class RulesCatalogCompiler

Namespace: LogicEngine.Compilers

Assembly: LogicEngine.dll

```
public class RulesCatalogCompiler : IRulesCatalogCompiler
```

Inheritance

object ← RulesCatalogCompiler

Implements

IRulesCatalogCompiler

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesCatalogCompiler(IRulesSetCompiler)

```
public RulesCatalogCompiler(IRulesSetCompiler rulesSetCompiler)
```

Parameters

`rulesSetCompiler` IRulesSetCompiler

Methods

Compile<T>(RulesCatalog)

Compiles a rules catalog into a compiled catalog for the specified type.

```
public Option<CompiledCatalog<T>> Compile<T>(RulesCatalog catalog) where T : new()
```

Parameters

`catalog` RulesCatalog

The rules catalog to compile. Must not be null and must contain valid rule sets.

Returns

Option<CompiledCatalog<T>>

An TinyFp.Option<A> containing a CompiledCatalog<T> if the compilation is successful; otherwise, an empty option if no valid rule sets are found.

Type Parameters

`T`

The type of the objects that the compiled catalog will operate on. Must have a parameterless constructor.

Remarks

This method filters and compiles the rule sets in the provided catalog, producing a compiled catalog that can be used for rule evaluation. If no valid rule sets are found, the method returns an empty option.

Class RulesSetCompiler

Namespace: LogicEngine.Compilers

Assembly: LogicEngine.dll

```
public class RulesSetCompiler : IRulesSetCompiler
```

Inheritance

object ← RulesSetCompiler

Implements

IRulesSetCompiler

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesSetCompiler(IRuleCompiler)

```
public RulesSetCompiler(IRuleCompiler singleRuleCompiler)
```

Parameters

`singleRuleCompiler` IRuleCompiler

Methods

Compile<T>(RulesSet)

Compiles a set of rules into a compiled rules set for the specified type.

```
public Option<CompiledRulesSet<T>> Compile<T>(RulesSet set) where T : new()
```

Parameters

`set` RulesSet

The set of rules to compile. Cannot be null.

Returns

Option<CompiledRulesSet<T>>

An TinyFp.Option<A> containing a CompiledRulesSet<T> if the compilation is successful, or an empty option if no rules are compiled.

Type Parameters

`T`

The type of object the rules will operate on. Must have a parameterless constructor.

Remarks

This method processes the rules in the provided `set` by compiling each rule individually, filtering out invalid or uncompiled rules, and then aggregating the results into a compiled rules set.

Namespace LogicEngine.Extensions

Classes

EnumerableExtensions

Class EnumerableExtensions

Namespace: LogicEngine.Extensions

Assembly: LogicEngine.dll

```
public static class EnumerableExtensions
```

Inheritance

object ← EnumerableExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Filter<T>(IEnumerable<T>, IAppliable<T>)

Filters the elements of the specified enumerable based on a condition defined by the provided IAppliable<T> implementation.

```
public static IEnumerable<T> Filter<T>(this IEnumerable<T> enumerable, IAppliable<T>
app) where T : new()
```

Parameters

`enumerable` IEnumerable<T>

The source enumerable to filter. Cannot be null.

`app` IAppliable<T>

An implementation of IAppliable<T> that defines the condition to apply to each element. Cannot be null.

Returns

IEnumerable<T>

An IEnumerable<T> containing elements from the source enumerable that satisfy the condition defined by `app`.

Type Parameters

`T`

The type of elements in the enumerable. Must have a parameterless constructor.

Remarks

This method uses the Apply(T) method to determine whether each element in the source enumerable should be included in the result.

FirstOrDefault<T>(IEnumerable<T>, IAppliable<T>)

Returns the first element in the sequence that satisfies the specified condition, or the default value if no such element is found.

```
public static T FirstOrDefault<T>(this IEnumerable<T> @this, IAppliable<T> app)
where T : new()
```

Parameters

`this` IEnumerable<T>

The sequence to search.

`app` IAppliable<T>

An object implementing IAppliable<T> that defines the condition to apply to each element.

Returns

T

The first element in the sequence that satisfies the condition defined by `app`. If no such element is found, a new instance of `T` is returned.

Type Parameters

T

The type of the elements in the sequence. Must have a parameterless constructor.

Namespace LogicEngine.Interfaces

Interfaces

IAppliable<T>

IAppliedSelector<TIn, TOut>

IDetailedAppliable<T, TOut>

Interface IAppliable<T>

Namespace: LogicEngine.Interfaces

Assembly: LogicEngine.dll

```
public interface IAppliable<in T> where T : new()
```

Type Parameters

T

Methods

Apply(T)

Applies the item to the appliable

```
bool Apply(T item)
```

Parameters

item T

Returns

bool

Interface IAppliedSelector<TIn, TOut>

Namespace: LogicEngine.Interfaces

Assembly: LogicEngine.dll

```
public interface IAppliedSelector<TIn, TOut> where TIn : new()
```

Type Parameters

`TIn`

`TOut`

Methods

FirstMatching(TIn)

Returns the first matching item

```
Option<TOut> FirstMatching(TIn item)
```

Parameters

`item` TIn

Returns

Option<TOut>

Interface IDetailedAppliable<T, TOut>

Namespace: LogicEngine.Interfaces

Assembly: LogicEngine.dll

```
public interface IDetailedAppliable<T, TOut> where T : new()
```

Type Parameters

`T`

`TOut`

Methods

DetailedApply(T)

Applies the item to the appliable

```
Either<TOut, Unit> DetailedApply(T item)
```

Parameters

`item` T

Returns

Either<TOut, Unit>

Namespace LogicEngine.Interfaces.Compilers

Interfaces

IRuleCompiler

IRulesCatalogCompiler

IRulesSetCompiler

Interface IRuleCompiler

Namespace: LogicEngine.Interfaces.Compilers

Assembly: LogicEngine.dll

```
public interface IRuleCompiler
```

Methods

Compile<T>(Rule)

Compiles a rule into a compiled rule, None if the rule is invalid

```
Option<CompiledRule<T>> Compile<T>(Rule rule) where T : new()
```

Parameters

`rule` Rule

Returns

Option<CompiledRule<T>>

Type Parameters

`T`

Interface IRulesCatalogCompiler

Namespace: LogicEngine.Interfaces.Compilers

Assembly: LogicEngine.dll

```
public interface IRulesCatalogCompiler
```

Methods

Compile<T>(RulesCatalog)

Compiles a catalog of rules into a compiled catalog, None if the catalog is invalid

```
Option<CompiledCatalog<T>> Compile<T>(RulesCatalog catalog) where T : new()
```

Parameters

`catalog` RulesCatalog

Returns

Option<CompiledCatalog<T>>

Type Parameters

`T`

Interface IRulesSetCompiler

Namespace: LogicEngine.Interfaces.Compilers

Assembly: LogicEngine.dll

```
public interface IRulesSetCompiler
```

Methods

Compile<T>(RulesSet)

Compiles a rule set into a compiled rule set, None if the rule set is invalid

```
Option<CompiledRulesSet<T>> Compile<T>(RulesSet set) where T : new()
```

Parameters

`set` RulesSet

Returns

Option<CompiledRulesSet<T>>

Type Parameters

`T`

Namespace LogicEngine.Internals

Enums

OperatorType

Enum OperatorType

Namespace: LogicEngine.Internals

Assembly: LogicEngine.dll

```
public enum OperatorType
```

Fields

`Contains = 11`

Item parameter contains specified constant

`ContainsKey = 15`

Item parameter dictionary property contains the given key

`ContainsValue = 17`

Item parameter dictionary property contains the given value

`Equal = 1`

Item parameter is equal to specified constant

`GreaterThan = 6`

Item parameter greater than specified constant

`GreaterThanOrEqual = 7`

Item parameter greater or equal than specified constant

`InnerContains = 29`

Item parameter array contains another item parameter

`InnerEqual = 23`

Item parameter equal to another item parameter

`InnerGreaterThan = 24`

Item parameter greater than another item parameter

`InnerGreaterThanOrEqual = 25`

Item parameter greater or equal than another item parameter

`InnerLessThan = 26`

Item parameter less than another item parameter

`InnerLessThanOrEqual = 27`

Item parameter less or equal than another item parameter

`InnerNotContains = 30`

Item parameter array does not contain another item parameter

`InnerNotEqual = 28`

Item parameter not equal to another item parameter

`InnerNotOverlaps = 32`

Item parameter array does not overlap another array parameter

`InnerOverlaps = 31`

Item parameter array overlaps another array parameter

`IsContained = 21`

Item parameter is contained into specified constant array

`IsNotContained = 22`

Item parameter is not contained into specified constant array

`KeyContainsValue = 19`

Item parameter dictionary has the given value for the specified key

`LessThan = 8`

Item parameter less than specified constant

`LessThanOrEqual = 9`

Item parameter less or equal than specified constant

`None = 0`

`NotContains = 12`

Item parameter not contains specified constant

`NotContainsKey = 16`

Item parameter dictionary property does not contain the given key

`NotContainsValue = 18`

Item parameter dictionary property does not contain the given value

`NotEqual = 10`

Item parameter not equal to specified constant

`NotKeyContainsValue = 20`

Item parameter dictionary has not the given value for the specified key

`NotOverlaps = 14`

Item parameter does not have intersections with specified enumerable constant

`Overlaps = 13`

Item parameter has intersection with specified enumerable constant

`StringContains = 4`

Item parameter string contains specified constant

`StringEndsWith = 3`

Item parameter string ends with specified constant

`StringRegexIsMatch = 5`

Item parameter string matches specified regex

`StringStartsWith = 2`

Item parameter string starts with specified constant

Namespace LogicEngine.Models

Classes

Rule

RulesCatalog

RulesSet

Class Rule

Namespace: LogicEngine.Models

Assembly: LogicEngine.dll

```
public record Rule : IEquatable<Rule>
```

Inheritance

object ← Rule

Implements

IEquatable<Rule>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Rule(string, OperatorType, string, string)

```
public Rule(string property, OperatorType type, string value, string code)
```

Parameters

`property` string

`type` OperatorType

`value` string

`code` string

Properties

Code

Code to return if the rule is not satisfied

```
[DataMember(Name = "code")]
[Required]
public string Code { get; init; }
```

Property Value

string

Operator

Operator to apply to the property

```
[DataMember(Name = "operator")]
[Required]
public OperatorType Operator { get; init; }
```

Property Value

OperatorType

Property

Name of the property the rule is applied to

```
[DataMember(Name = "property")]
[Required]
public string Property { get; init; }
```

Property Value

string

Value

Value to compare the property to or name of the property to compare with Property

```
[DataMember(Name = "value")]
[Required]
public string Value { get; init; }
```

Property Value

string

Methods

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

string

A string that represents the current object.

Class RulesCatalog

Namespace: LogicEngine.Models

Assembly: LogicEngine.dll

```
public record RulesCatalog : IEquatable<RulesCatalog>
```

Inheritance

object ← RulesCatalog

Implements

IEquatable<RulesCatalog>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesCatalog(IEnumerable<RulesSet>, string)

Initializes a new instance of the RulesCatalog class with the specified rules sets and name.

```
public RulesCatalog(IEnumerable<RulesSet> rulesSets, string name)
```

Parameters

`rulesSets` IEnumerable<RulesSet>

A collection of RulesSet objects that define the rules contained in the catalog. This parameter cannot be null.

`name` string

The name of the rules catalog. This parameter cannot be null or empty.

Properties

Name

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

Property Value

string

RulesSets

Rules sets that make up the catalog

```
public IEnumerable<RulesSet> RulesSets { get; }
```

Property Value

IEnumerable<RulesSet>

Operators

operator +(RulesCatalog, RulesCatalog)

Combines two RulesCatalog instances into a single catalog containing the rules from both.

```
public static RulesCatalog operator +(RulesCatalog catalog1, RulesCatalog catalog2)
```

Parameters

`catalog1` RulesCatalog

The first RulesCatalog to combine.

`catalog2` RulesCatalog

The second RulesCatalog to combine.

Returns

RulesCatalog

A new RulesCatalog that contains the combined rule sets from both catalogs.

Remarks

The resulting catalog will include all rule sets from `catalog1` and `catalog2`. The name of the resulting catalog will be a combination of the names of the input catalogs, formatted as "(Name1 OR Name2)".

operator *(RulesCatalog, RulesCatalog)

Combines two RulesCatalog instances by applying a logical AND operation to their respective rule sets.

```
public static RulesCatalog operator *(RulesCatalog catalog1, RulesCatalog catalog2)
```

Parameters

`catalog1` RulesCatalog

The first RulesCatalog to combine.

`catalog2` RulesCatalog

The second RulesCatalog to combine.

Returns

RulesCatalog

A new RulesCatalog instance containing the combined rule sets of `catalog1` and `catalog2`, with a name indicating the logical AND operation.

Remarks

This operator creates a new RulesCatalog where each rule in `catalog1` is combined with each rule in `catalog2` using their respective multiplication logic. The resulting catalog's name reflects the combination of the two input catalogs.

Class RulesSet

Namespace: LogicEngine.Models

Assembly: LogicEngine.dll

```
public record RulesSet : IEquatable<RulesSet>
```

Inheritance

object ← RulesSet

Implements

IEquatable<RulesSet>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesSet(IEnumerable<Rule>, string)

Initializes a new instance of the RulesSet class with the specified rules and name.

```
public RulesSet(IEnumerable<Rule> rules, string name)
```

Parameters

`rules` IEnumerable<Rule>

A collection of Rule objects that define the rules for this set. Cannot be null.

`name` string

The name of the rules set. Cannot be null or empty.

Properties

Name

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

Property Value

string

Rules

Rules that make up the set

```
[DataMember(Name = "rules")]
[Required]
public IEnumerable<Rule> Rules { get; init; }
```

Property Value

IEnumerable<Rule>

Operators

operator *(RulesSet, RulesSet)

Combines two RulesSet instances into a new RulesSet that contains the concatenated rules and a combined name.

```
public static RulesSet operator *(RulesSet set1, RulesSet set2)
```

Parameters

`set1` RulesSet

The first RulesSet to combine.

`set2` RulesSet

The second RulesSet to combine.

Returns

RulesSet

A new RulesSet that contains the rules from both `set1` and `set2`, and a name representing their logical combination.