Diagram Report

CAT_Lib

Version 1.0.1 •

Date Generated:	4/29/2022
Author:	Davide Basile (ISTI CNR Italy)

Table of Contents

automaton diagram

Class diagram in package 'automaton'

automaton

Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

```
S1
L1
S > State<S1>
T > Transition<S1,L1,S,? extends Label<L1>>
```

Automaton

- tra: Set<T> {readOnly}

+ Automaton(Set<T>)
+ getBasicStates(): Map<Integer,Set<BasicState<S1>>>
+ getForwardStar(AbstractState<?>): Set<T>
+ getInitial(): S
+ getNumStates(): int
+ getRank(): Integer
+ getStates(): Set<S>
+ getTransition(): Set<T>
+ toString(): String
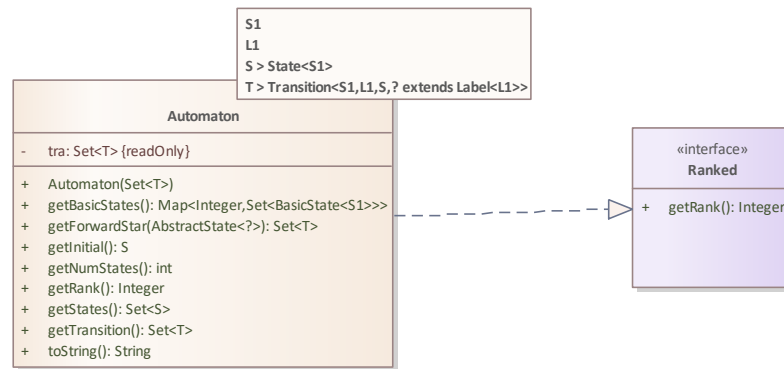
«interface»
Ranked

+ getRank(): Integer

Figure 1: automaton

label diagram

Class diagram in package 'label'

label

Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

CALabel

+	CALabel(Integer, Integer, Action)
+	CALabel(List<Action>)
+	getAction(): Action
+	getCoAction(): Action
+	getOfferer(): Integer
-	getOffererIfAny(): Integer
+	getOffererOrRequester(): Integer
+	getRequester(): Integer
-	getRequesterIfAny(): Integer
+	isMatch(): boolean
+	isOffer(): boolean
+	isRequest(): boolean
+	match(Label<Action>): boolean

< T->Action >

T

Label

-	content: List<T> {readOnly}
+	equals(Object): boolean
+	getAction(): Action
+	getContent(): List<T>
+	getRank(): Integer
+	hashCode(): int
+	Label(List<T>)
+	match(Label<T>): boolean
+	toString(): String

< T->Label<T> >

T

«interface»
Matchable
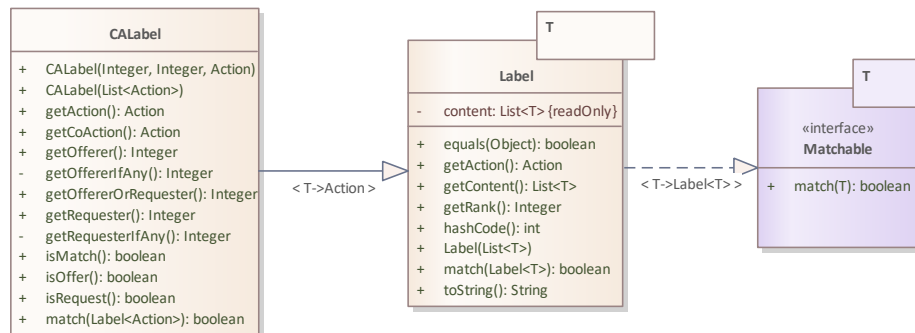
+	match(T): boolean

Figure 2: label

action diagram

Class diagram in package 'action'

action

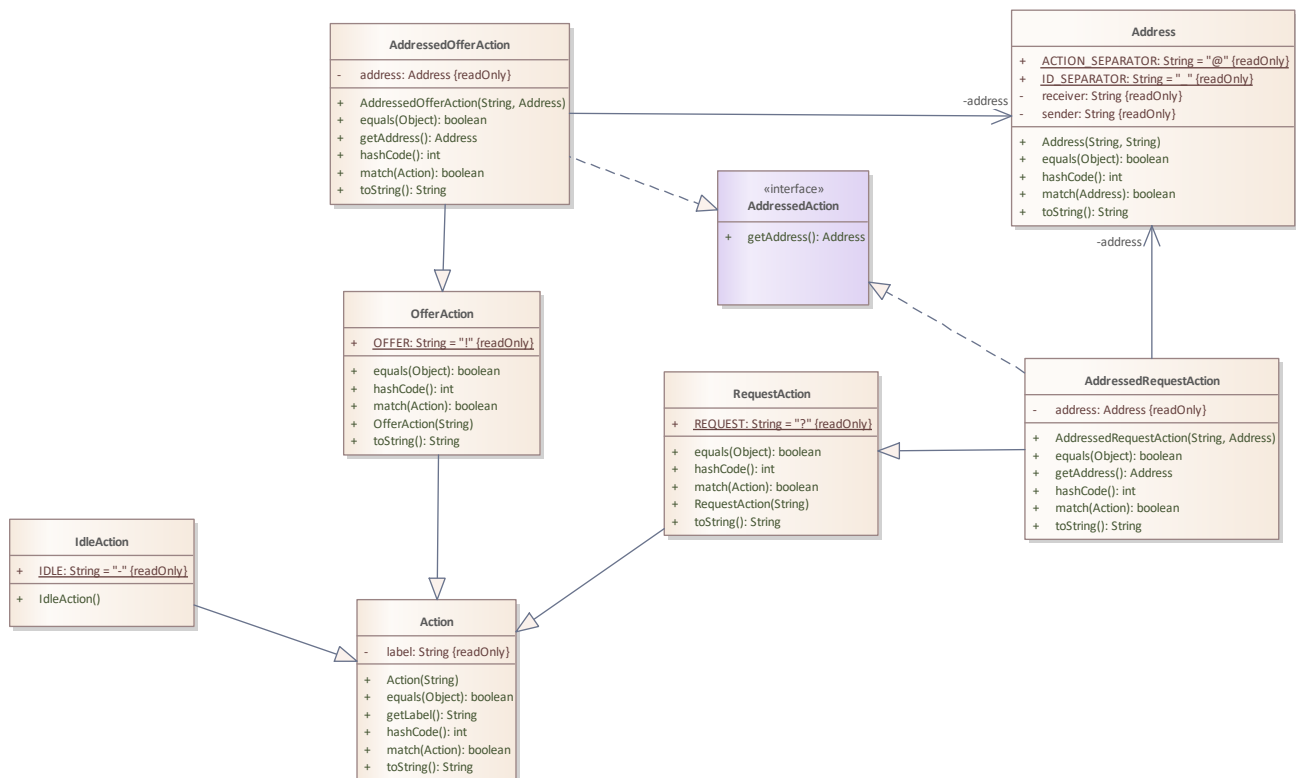Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

Figure 3: action

state diagram

Class diagram in package 'state'

state

Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

T

BasicState

- fin: boolean {readOnly}
- init: boolean {readOnly}

+ BasicState(T, Boolean, Boolean)
+ getRank(): Integer
+ isFinalState(): boolean
+ isInitial(): boolean
+ toString(): String

T

State

+ getRank(): Integer
+ getState(): List<BasicState<T>>
+ isFinalState(): boolean
+ isInitial(): boolean
+ State(List<BasicState<T>>)
+ toString(): String

< T->T >

< T->List<BasicState<T>> >

T

AbstractState

- label: T {readOnly}

AbstractState(T)
+ getState(): T
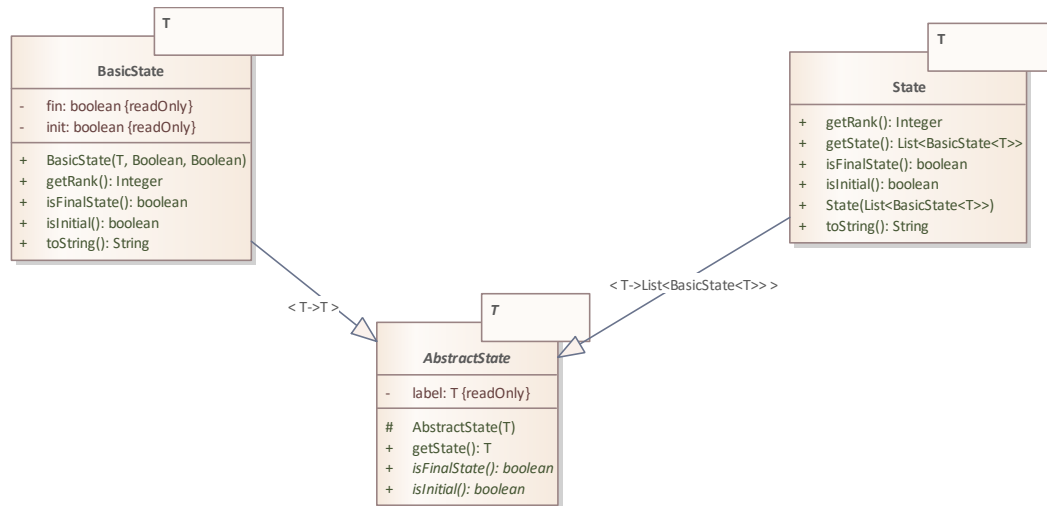+ *isFinalState(): boolean*
+ *isInitial(): boolean*

Figure 4: state

transition diagram

Class diagram in package 'transition'

transition

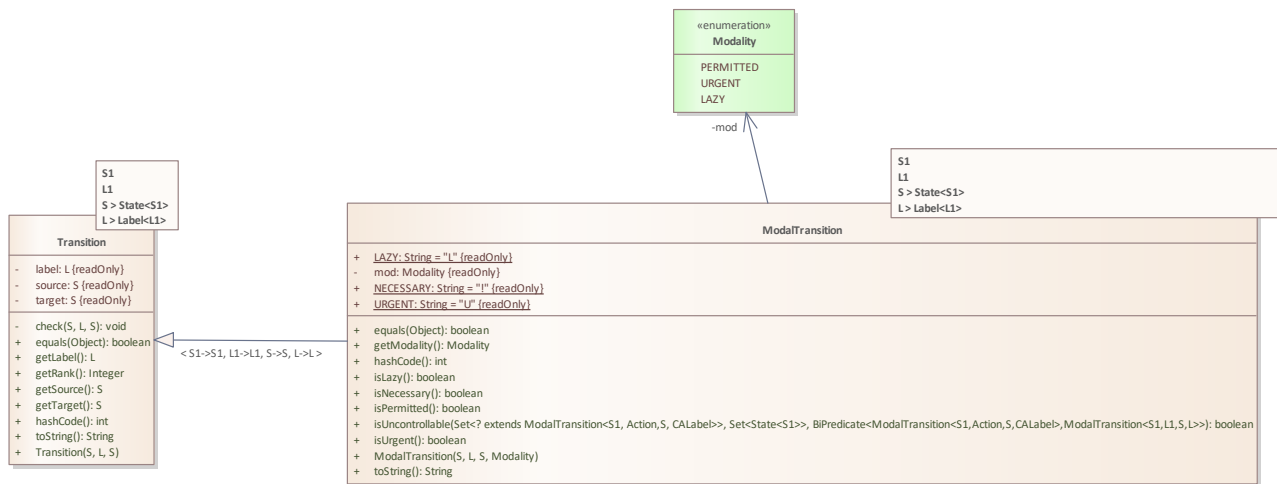Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

Figure 5: transition

converters diagram

Class diagram in package 'converters'

converters

Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

A2 > Automaton<?,?,?,?>

«interface»
AutConverter

+ exportMSCA(String, A2): void
+ importMSCA(String): A1

«default»
+ parseAction(String): Action

< A1->Automaton<String,Action, State<String>, A2->ModalTransition<String, Action,State<String> >

L > Label<Action>

AutDataConverter

- createLabel: Function<List<Action>,L> {readOnly}
- EMPTYMSG: String = "Empty file name" {readOnly}
- SUFFIX: String = ".data" {readOnly}

+ AutDataConverter(Function<List<Action>, L>)
- createLabel(String[][]): L
- createOrLoadState(Set<State<String>>, Map<Integer,Set<BasicState<String>>>, String[], String[], String[][]): State<String>
+ exportMSCA(String, Automaton<?,?,?,?>): void
+ importMSCA(String): Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,L>>
- loadTransition(String, int, ModalTransition.Modality, Set<State<String>>, Map<Integer,Set<BasicState<String>>>, String[], String[][]): ModalTransition<String,Action,State<String>,L>
- readFinalState(String, int): String[]
- readInitialState(String, int): String[]
- readModality(String): ModalTransition.Modality

Figure 6: converters

family diagram

Class diagram in package 'family'

family

Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

Family

- areComparable: BiPredicate<Product,Product> {readOnly}
- compare: BiFunction<Product, Product, Integer> {readOnly}
- po: Map<Product, Map<Boolean, Set<Product>>> {readOnly}
- products: Set<Product> {readOnly}

+ equals(Object): boolean
+ Family(Set<Product>)
+ Family(Set<Product>, BiPredicate<Product,Product>, BiFunction<Product, Product, Integer>)
+ getMaximalProducts(): Set<Product>
+ getMaximumDepth(): int
+ getPo(): Map<Product, Map<Boolean, Set<Product>>>
+ getProducts(): Set<Product>
+ getSubProductsNotClosedTransitively(Product): Set<Product>
+ getSubProductsOfProduct(Product): Set<Product>
+ getSuperProductsOfProduct(Product): Set<Product>
+ hashCode(): int
+ toString(): String

-family

Product

- forbidden: Set<Feature> {readOnly}
- required: Set<Feature> {readOnly}

+ checkForbidden(Set<? extends ModalTransition<String,Action,State<String>,CALabel>>): boolean
+ checkRequired(Set<? extends ModalTransition<S1, Action, State<S1>, CALabel>>): boolean
+ equals(Object): boolean
+ getForbidden(): Set<Feature>
+ getForbiddenAndRequiredNumber(): int
+ getRequired(): Set<Feature>
+ hashCode(): int
+ isForbidden(CALabel): boolean
+ isValid(Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>): boolean
+ Product(Set<Feature>, Set<Feature>)
+ Product(String[], String[])
+ removeFeatures(Set<Feature>): Product
+ toString(): String

FMCA

- aut: Automaton<String, Action, State<String>, ModalTransition<String,Action,State<String>, CALabel>> {readOnly}
- family: Family {readOnly}

+ FMCA(Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>, Family)
+ FMCA(Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>, Set<Product>)
+ getAut(): Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>
+ getCanonicalProducts(): Map<Product,Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>>
+ getFamily(): Family
+ getOrchestrationOfFamily(): Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>
+ getOrchestrationOfFamilyEnumerative(): Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>
+ getTotalProductsWithNonemptyOrchestration(): Map<Product,Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>>
+ productsRespectingValidity(): Set<Product>
- productsRespectingValidity(Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>): Set<Product>
+ productsWithNonEmptyOrchestration(): Set<Product>
- productsWithNonEmptyOrchestration(Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>): Set<Product>
- selectProductsSatisfyingPredicateUsingPO(Automaton<String,Action,State<String>,ModalTransition<String,Action,State<String>,CALabel>>, Predicate<Product>): Set<Product>

Feature

- name: String {readOnly}

+ equals(Object): boolean
+ Feature(String)
+ getName(): String
+ hashCode(): int
+ toString(): String

UnaryOperator
PartialProductGenerator

+ apply(Set<Product>): Set<Product>

Figure 7: family

converters diagram

Class diagram in package 'converters'

converters

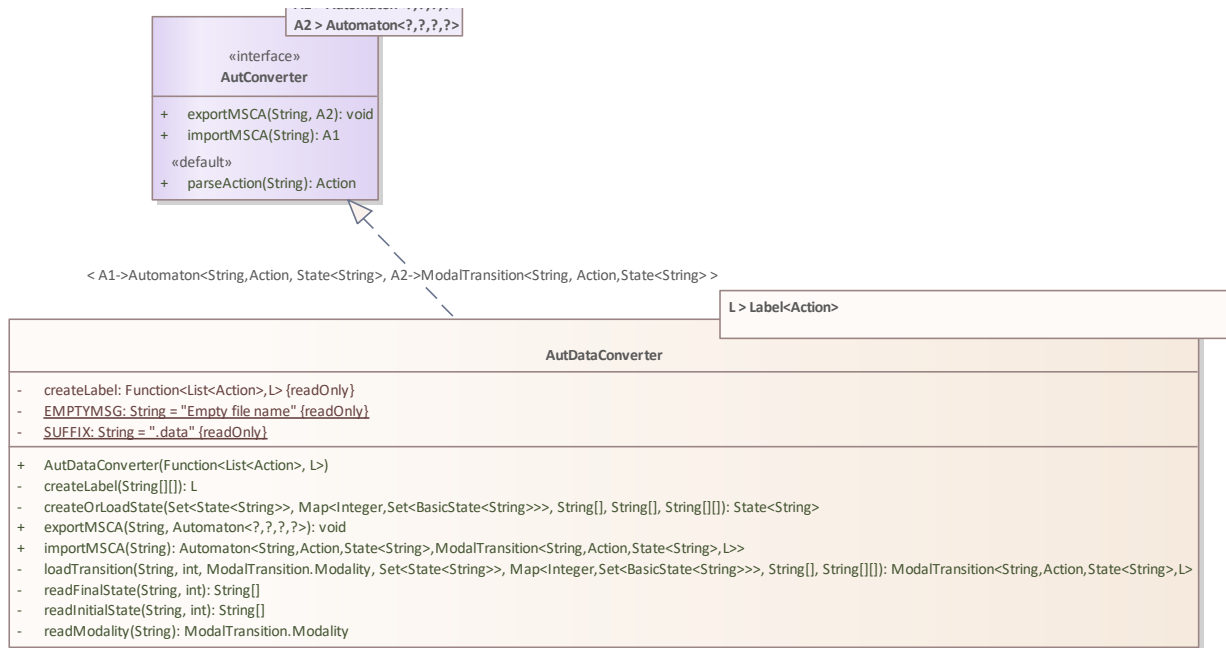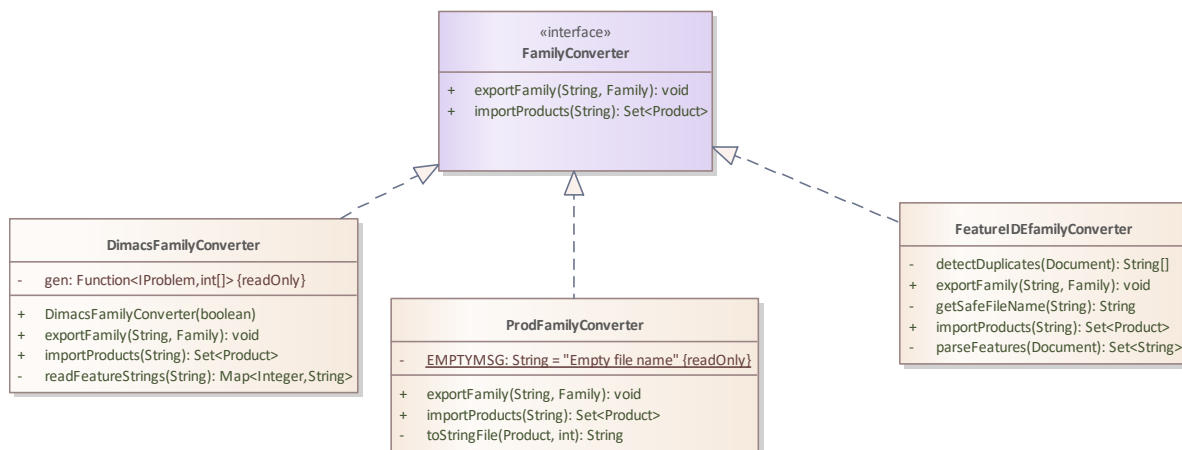Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

Figure 8: converters

operations diagram

Class diagram in package 'operations'

operations

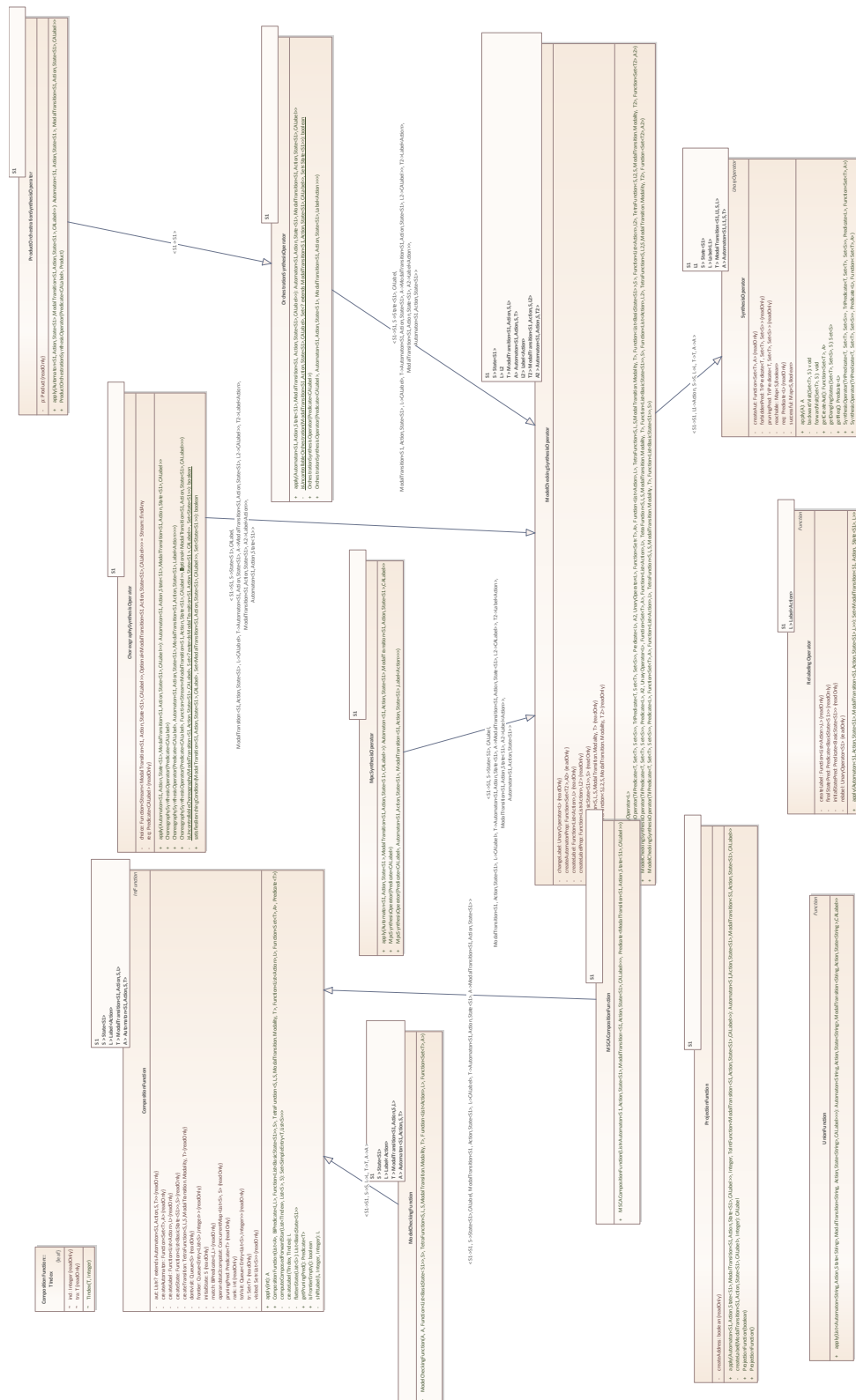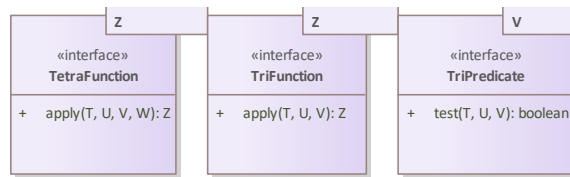Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

Figure 9: operations

interfaces diagram

Class diagram in package 'interfaces'

interfaces

Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/23/2022

Figure 10: interfaces

requirements diagram

Class diagram in package 'requirements'

requirements

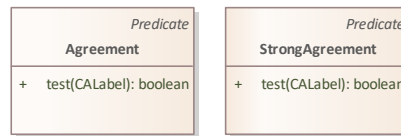Davide Basile (ISTI CNR Italy) created on 4/23/2022. Last modified 4/29/2022

Predicate		*Predicate*
Agreement		**StrongAgreement**
+ test(CALabel): boolean		+ test(CALabel): boolean

Figure 11: requirements

requirements diagram

Class diagram in package 'requirements'