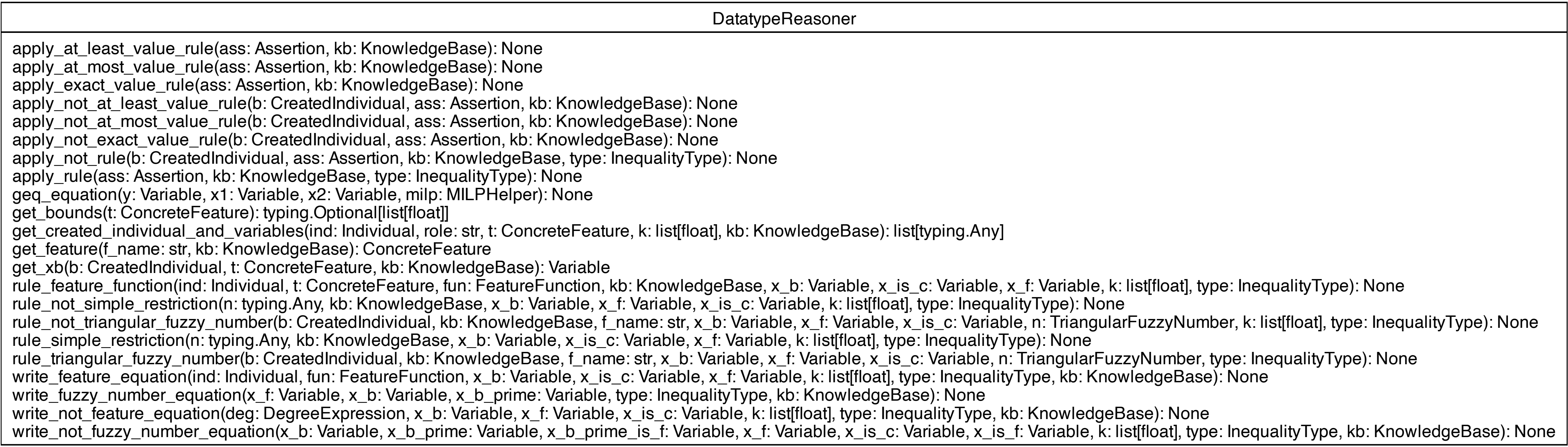

DatatypeReasoner

```
apply_at_least_value_rule(ass: Assertion, kb: KnowledgeBase): None
apply_at_most_value_rule(ass: Assertion, kb: KnowledgeBase): None
apply_exact_value_rule(ass: Assertion, kb: KnowledgeBase): None
apply_not_at_least_value_rule(b: CreatedIndividual, ass: Assertion, kb: KnowledgeBase): None
apply_not_at_most_value_rule(b: CreatedIndividual, ass: Assertion, kb: KnowledgeBase): None
apply_not_exact_value_rule(b: CreatedIndividual, ass: Assertion, kb: KnowledgeBase): None
apply_not_rule(b: CreatedIndividual, ass: Assertion, kb: KnowledgeBase, type: InequalityType): None
apply_rule(ass: Assertion, kb: KnowledgeBase, type: InequalityType): None
geq_equation(y: Variable, x1: Variable, x2: Variable, milp: MILPHelper): None
get_bounds(t: ConcreteFeature): typing.Optional[list[float]]
get_created_individual_and_variables(ind: Individual, role: str, t: ConcreteFeature, k: list[float], kb: KnowledgeBase): list[typing.Any]
get_feature(f_name: str, kb: KnowledgeBase): ConcreteFeature
get_xb(b: CreatedIndividual, t: ConcreteFeature, kb: KnowledgeBase): Variable
rule_feature_function(ind: Individual, t: ConcreteFeature, fun: FeatureFunction, kb: KnowledgeBase, x_b: Variable, x_is_c: Variable, x_f: Variable, k: list[float], type: InequalityType): None
rule_not_simple_restriction(n: typing.Any, kb: KnowledgeBase, x_b: Variable, x_f: Variable, x_is_c: Variable, k: list[float], type: InequalityType): None
rule_not_triangular_fuzzy_number(b: CreatedIndividual, kb: KnowledgeBase, f_name: str, x_b: Variable, x_f: Variable, x_is_c: Variable, n: TriangularFuzzyNumber, k: list[float], type: InequalityType): None
rule_simple_restriction(n: typing.Any, kb: KnowledgeBase, x_b: Variable, x_is_c: Variable, x_f: Variable, k: list[float], type: InequalityType): None
rule_triangular_fuzzy_number(b: CreatedIndividual, kb: KnowledgeBase, f_name: str, x_b: Variable, x_f: Variable, x_is_c: Variable, n: TriangularFuzzyNumber, type: InequalityType): None
write_feature_equation(ind: Individual, fun: FeatureFunction, x_b: Variable, x_is_c: Variable, x_f: Variable, k: list[float], type: InequalityType, kb: KnowledgeBase): None
write_fuzzy_number_equation(x_f: Variable, x_b: Variable, x_b_prime: Variable, type: InequalityType, kb: KnowledgeBase): None
write_not_feature_equation(deg: DegreeExpression, x_b: Variable, x_f: Variable, x_is_c: Variable, k: list[float], type: InequalityType, kb: KnowledgeBase): None
write_not_fuzzy_number_equation(x_b: Variable, x_b_prime: Variable, x_b_prime_is_f: Variable, x_f: Variable, x_is_c: Variable, x_is_f: Variable, k: list[float], type: InequalityType, kb: KnowledgeBase): None
```