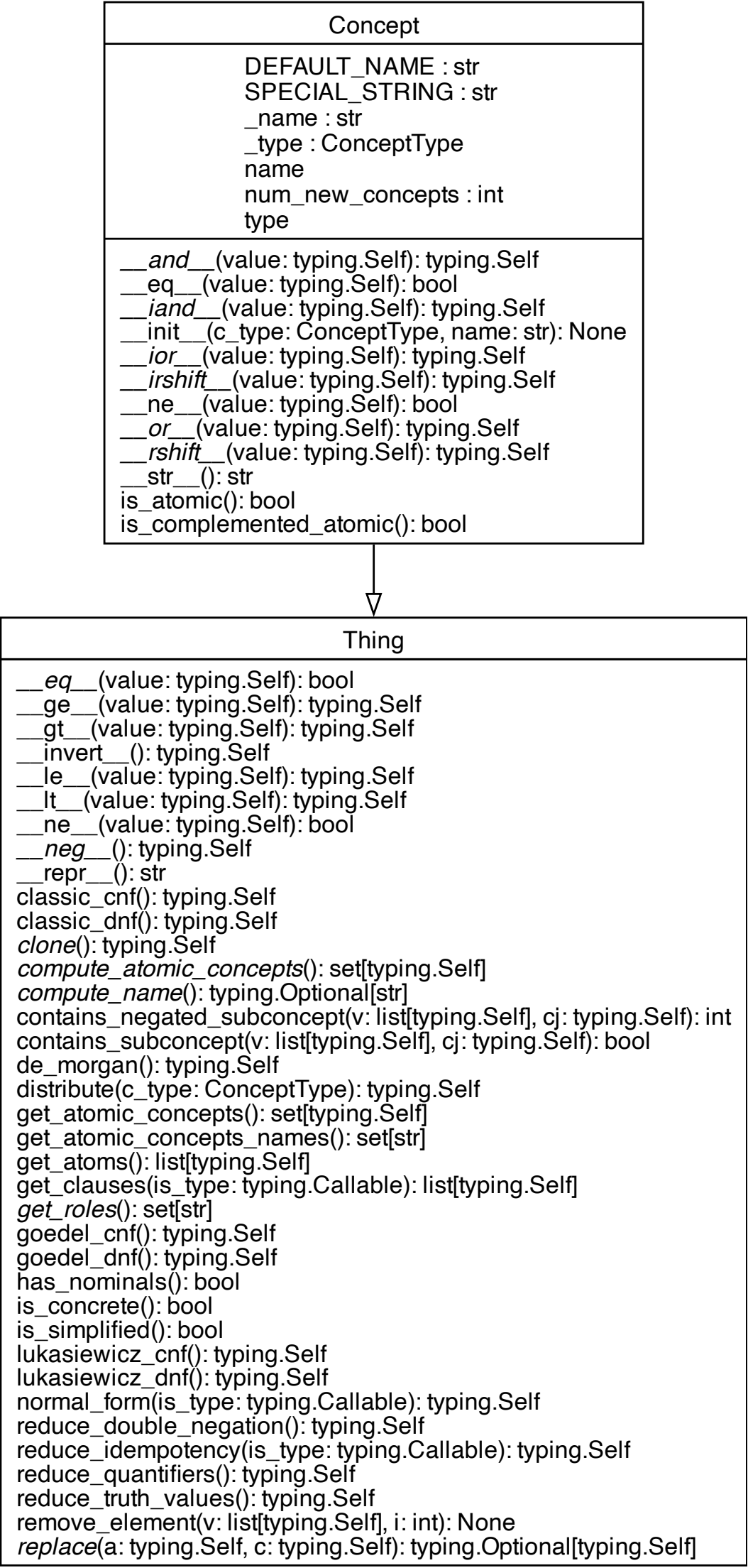
Concept
DEFAULT_NAME : str
SPECIAL_STRING : str
_name : str
_type : ConceptType
name
num_new_concepts : int
type
__and__(value: typing.Self): typing.Self
__eq__(value: typing.Self): bool
__iand__(value: typing.Self): typing.Self
__init__(c_type: ConceptType, name: str): None
__ior__(value: typing.Self): typing.Self
__irshift__(value: typing.Self): typing.Self
__ne__(value: typing.Self): bool
__or__(value: typing.Self): typing.Self
__rshift__(value: typing.Self): typing.Self
__str__(): str
is_atomic(): bool
is_complemented_atomic(): bool
Thing
__eq__(value: typing.Self): bool
__ge__(value: typing.Self): typing.Self
__gt__(value: typing.Self): typing.Self
__invert__(): typing.Self
__le__(value: typing.Self): typing.Self
__lt__(value: typing.Self): typing.Self
__ne__(value: typing.Self): bool
__neg__(): typing.Self
__repr__(): str
classic_cnf(): typing.Self
classic_dnf(): typing.Self
clone(): typing.Self
compute_atomic_concepts(): set[typing.Self]
compute_name(): typing.Optional[str]
contains_negated_subconcept(v: list[typing.Self], cj: typing.Self): int
contains_subconcept(v: list[typing.Self], cj: typing.Self): bool
de_morgan(): typing.Self
distribute(c_type: ConceptType): typing.Self
get_atomic_concepts(): set[typing.Self]
get_atomic_concepts_names(): set[str]
get_atoms(): list[typing.Self]
get_clauses(is_type: typing.Callable): list[typing.Self]
get_roles(): set[str]
goedel_cnf(): typing.Self
goedel_dnf(): typing.Self
has_nominals(): bool
is_concrete(): bool
is_simplified(): bool
lukasiewicz_cnf(): typing.Self
lukasiewicz_dnf(): typing.Self
normal_form(is_type: typing.Callable): typing.Self
reduce_double_negation(): typing.Self
reduce_idempotency(is_type: typing.Callable): typing.Self
reduce_quantifiers(): typing.Self
reduce_truth_values(): typing.Self
remove_element(v: list[typing.Self], i: int): None
replace(a: typing.Self, c: typing.Self): typing.Optional[typing.Self]