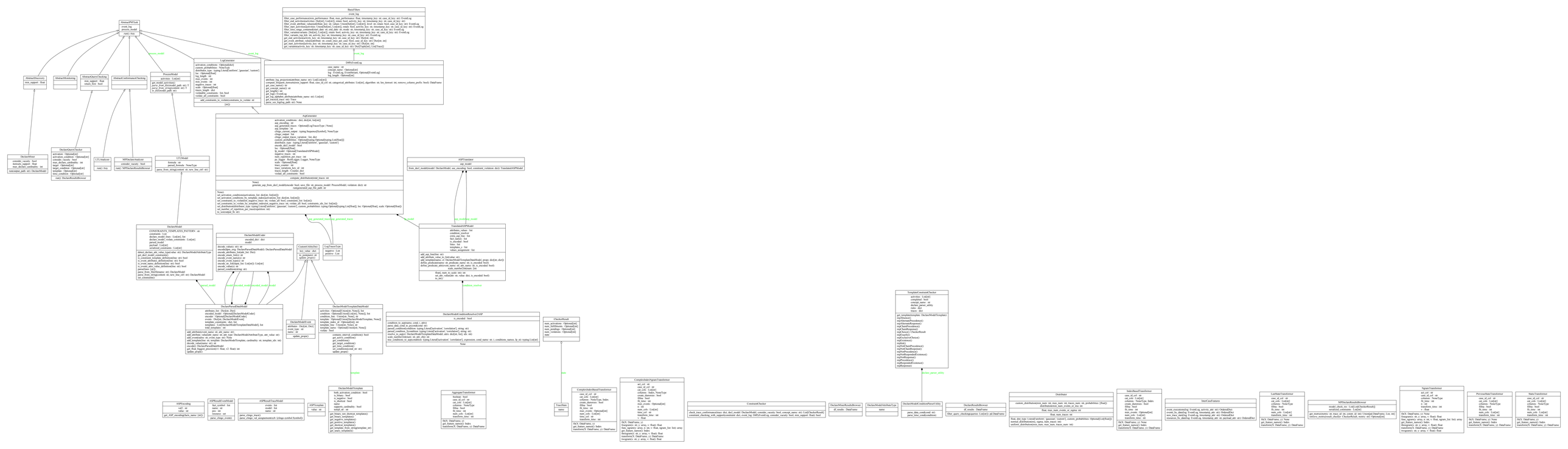
AbstractPMTask
event_log
process_model
run() -> Any
BaseEfficiency
event_log
filter_case_performance(min_performance: float, max_performance: float, timestamp_key: str, case_id_key: str) -> EventLog
filter_event_attribute_values(attribute_key: str, values: List[str], level: str, retain: bool, case_id_key: str) -> EventLog
filter_event_attribute_values(attribute_key: str, values: Union[Set[str], List[str]], level: str, retain: bool, case_id_key: str) -> EventLog
filter_start_activities(activities: Union[Set[str], List[str]], retain: bool, activity_key: str, timestamp_key: str, case_id_key: str) -> EventLog
filter_time_range(min_time: str, max_time: str, timestamp_key: str, case_id_key: str) -> EventLog
filter_variants(variants: Union[Set[str], List[str]], retain: bool, activity_key: str, timestamp_key: str, case_id_key: str) -> EventLog
filter_variants_top_k(k: int, activity_key: str, timestamp_key: str, case_id_key: str) -> EventLog
get_event_attribute_values(attribute_key: str) -> Dict[str, int]
get_event_attribute_values(attribute: str, count_once_per_case: bool, case_id_key: str) -> Dict[str, int]
get_start_activities(activity_key: str, timestamp_key: str, case_id_key: str) -> Dict[str, int]
get_variants(activity_key: str, timestamp_key: str, case_id_key: str) -> Dict[Tuple[str], List[Trace]]
process_model
event_log
event_log
AbstractDiscovery
mm_support : float
AbstractMonitoring
AbstractQueryChecking
mm_support : float
return_live : bool
AbstractConformanceChecking
LogGenerator
activation_conditions : Optional[dict]
custom_probabilities : Optional[dict]
distribution_type : typing.Literal['uniform', 'gaussian', 'custom']
log_mean : Optional[int]
log_length : int
max_events : int
no_traces : int
negative_traces : int
scale : Optional[float]
trace_length : int
violation_constraints : Set_Sed
violate_all_constraints : bool
add_constraints_to_violate(constraints_to_violate: str [set])
DMPyEventLog
case_name : str
concept_name : Optional[str]
case_id_key : Optional[str]
log_length : Optional[int]
attribute_log_projection(attribute_name: str) -> List[List[str]]
compute_frequent_items(min_support: float, case_id_col: str, categorical_attributes: List[str], algorithm: str, low_itemset: int, remove_column_prefix: bool): DataFrame
get_case_name() -> str
get_concept_name() -> str
get_length() -> int
get_log() -> EventLog
get_log_projection(attribute_name: str) -> List[str]
get_trace(id_trace: int) Trace
parse_xes_log(log_path: str) -> None
DeclareModel
consider_vacuity : bool
itemsets_support : float
min_declare_cardinality : int
max_declare_cardinality : int
discover() -> DeclareModel
DeclareQuasiChecker
activation : Optional[str]
target_condition : Optional[str]
template : Optional[str]
time_condition : Optional[str]
run() -> DeclareRoundRobbrowser
LTLAnalyzer
run() -> Any
MPPDeclareAnalyzer
consider_vacuity : bool
run() -> MPPDeclareRoundRobbrowser
LTLModel
formula : str
used_formula : NoneType
parse_from_string(content: str, new_line_ctrl: str)
AspGenerator
activation_conditions : dict, dict[str, list[str]]
no_traces : int
opt_generated_traces : Optional[Literal['one'] | 'max'] | None
asp_template : str
custom_probabilities : Optional[dict[str, typing.Sequence[float]]]
distribution_type : typing.Literal['uniform', 'gaussian', 'custom']
escode_decl_model : bool
log_mean : Optional[int]
ltc_model : Optional[TranslatedASPModel]
negative_traces : int
no_constraints_to_violate : int
pic_bigger : float(Logic_Logic), NoneType
scale : Optional[float]
trace_length : int
traces_variation_key_id : int
traces_length : Counter, dict
violate_all_constraints : bool
add_distribution(distribution_type: typing.Literal['uniform', 'gaussian', 'custom'], custom_probabilities: typing.Optional[typing.List[float]], loc: Optional[float], scale: Optional[float]) -> None
add_distribution_to_trace(trace_id: int, distribution_type: typing.Literal['uniform', 'gaussian', 'custom'], custom_probabilities: typing.Optional[typing.List[float]], loc: Optional[float], scale: Optional[float]) -> None
set_activation_conditions(activations: list dict[str, list[str]]) -> None
set_activation_conditions_by_template_index(activations: list dict[str, list[str]]) -> None
set_constraints_to_violate(constraints_to_violate: list[str]) -> None
set_constraints_to_violate_by_template_index(negative_trace: int, violate_all: bool, constraints_ids: list list[int]) -> None
set_distribution(distribution_type: typing.Literal['uniform', 'gaussian', 'custom'], custom_probabilities: typing.Optional[typing.List[float]], loc: Optional[float], scale: Optional[float]) -> None
set_duration_of_algorithm_per_trace(maxminute: int) -> None
set_one(output_fn: str) -> None
ASPTranslator
asp_model
from_dict_model(dictmodel: DeclareModel, use_encoding: bool, constraint_violation: dict): TranslatedASPModel
None()
generate_asp_from_dict_model(dictmodel: bool, save_file: str, process_model: ProcessModel, violations: dict): str
run(generated_asp_file_path: str
None()
set_activation_conditions(activations: list dict[str, list[str]]) -> None
set_activation_conditions_by_template_index(activations: list dict[str, list[str]]) -> None
set_constraints_to_violate(constraints_to_violate: list[str]) -> None
set_constraints_to_violate_by_template_index(negative_trace: int, violate_all: bool, constraints_ids: list list[int]) -> None
set_distribution(distribution_type: typing.Literal['uniform', 'gaussian', 'custom'], custom_probabilities: typing.Optional[typing.List[float]], loc: Optional[float], scale: Optional[float]) -> None
set_duration_of_algorithm_per_trace(maxminute: int) -> None
set_one(output_fn: str) -> None
asp_generated_traces
asp_generated_traces
model
asp_model
asp_model
DeclareModel
CONSTRAINTS_TEMPLATES_PATTERN : str
activities : list
declare_model_base : List[str], list
declare_model_violate_constraints : List[str]
max_declare_cardinality : int
parsload : List[str]
serialized_constraints : List[str]
detect_declare_attr_value_type(value: str) DeclareModelAttributeType
get_all_constraints() -> list
is_constraint_template_definition(line: str): bool
is_event_attribute_definition(line: str): bool
is_event_name_definition(line: str): bool
is_events_attrs_value_definition(line: str): bool
parse_from_file(filename: str) DeclareModel
parse_from_lines(lines: str) DeclareModel
parse_from_string(content: str, new_line_ctrl: str) DeclareModel
set_constraints(list
DeclareModelCode
encoded_dict : dict
model
decode(values): str) -> str
encode(dpm_orig: DeclareParsedDataModel): DeclareParsedDataModel
encode(dpm: str): DeclareModel
encode_event_label(s: str) -> str
encode_event_name(s): str
encode_label(s: str) -> str
encode_set_for(label_list: List[str]): List[str]
encode_value(s): str
get_encoding_mapping(): str
Constant.Attr(Enu
key_value : dict
to_ignore : str
update_project()
LogTraceType
negative : List
positive : List
DeclareModelEvent
attributes : Dict[str, Dict]
event_type : str
name : str
update_project()
DeclareModelTemplateDataModel
activities : Optional[Union[int, NoneType]] : list
condition : Optional[Union[List[List[str]], None]], list
condition_line : Union[int, None], int
condition_template : Optional[DeclareModelTemplate, None]]
template_index_of : Optional[int], int
template_line : Union[int, None], int
template_name : Optional[Union[str, None]]
visible : bool
contains_mirrored_condition(): bool
get_active_condition()
get_condition_line()
get_target_condition()
get_template_name()
set_condition_and_str(: str)
update_project()
DeclareModelConditionResolverDASP
is_encoded : bool
condition_to_asp(name_cond_i, attr)
parse_data_cond_to_pycond(cond: str)
parse_template_condition(activation: List[List[str]]activation', 'correlation'], string: str)
parsed_condition_2(condition: typing.List[List[str]activation', 'correlation'], string: str)
resolve_to_asp(t: DeclareModelTemplateDataModel, attrs: dict[str, list], ids: int)
to_asp_condition(cond: str) -> str
tree_conditions_to_asp(condition: typing.List[List[str]activation', 'correlation'], expression_cond_name: str, i, conditions_names_ls: tp.it) -> typing.List[str]
None
CheckerResult
state : str
template : str
DeclareModelTemplate
both_activation_condition : bool
id : NoneType
is_negative : bool
is_vbcheck : bool
name : str
supports_cardinality : bool
templ_str : str
get_binary_not_chain(not_template)
get_binary_not_chain() -> str
get_negative_template()
get_not_chain_template()
get_chain(not_template)
get_template_from_string(template_str)
get_unary_template()
DeclareModelTemplateDataModelBrowser
df_results : DataFrame
DeclareModelAttributeType
name : str
DeclareModelConditionParserUtility
parse_data_cond(cond: str)
parse_time_cond(condition)
DeclareResultsBrowser
filter_query_checks(arguments: List[str]): pd.DataFrame
TemplateConstraintChecker
activities : List[str]
completed : bool
log : EventLog
declare_parser_utility
index : dict
traces : dict
get_template(template: DeclareModelTemplate)
isEventually()
mpiAlternateProcedence()
mpiAlternateResponse()
mpiChainProcedence()
mpiChainResponse()
mpiCoExistenceCheckerResult
mpiEventually()
mpiExclusiveChoice()
mpiNotChain()
mpiNotCoExistence()
mpiPrecedence()
mpiPrecedenceChainResponse()
mpiPrecedenceResponse()
mpiPrecedenceResponseAlternate()
mpiProbResponse()
mpiProcedence()
mpiResponse()
mpiResponseEventually()
mpiResponseChain()
declare_parser_utility
DeclareModelTemplateDataModel
DeclareModelTemplateDataModel
DeclareModelTemplateDataModel