## Synthesis2AMN ATL Transformation

* string представляется типом seq(0..255), но строковые константы не разрешены (в примерах спецификаций по возможности используем enum)
* для каждого типа создается абстрактная машина (без операций-методов) и уточнение
* инвариант в инициализацию и set-операции (инварианта из одного текущего типа достаточно)
  + аргументы не вставлять эти инварианты: для инициализации и так понятно, что если вставить, то po будет доказано. Инициализация вообще только для того, чтобы пройти синтаксическую проверку. Set-операции также можно не проверять, смысла в этом нет. А когда они используются внутри методов, то спецификация метода должна отвечать за выполнение инварианта.
* В уточнении set-операций нет
* enum – только contained тип (встроен в атрибут)
* типы атрибутов: ADT, string, integer, Boolean, enum, Set(ADT, integer, Boolean, string)
* АТД только объектные
* instance type класса должен быть в секции типов
* параметр функции может быть либо входным, либо выходным
* типы параметров и подкванторных переменных: ADT, string, integer, Boolean, Set(ADT, integer, Boolean, string)
* однобуквенный идентификатор (тип, переменная, атрибут, параметр и тд) превращается в двухбуквенный (одинаковые буквы)
* В спецификации функции не должно быть неявных эффектов (это отображение не сделать matched правилами, нужно делать lazy с перечислениями разных видов формул)
* Предикаты <, <= считаются предикатами на множествах. В качестве арифметических следует использовать lt, le. Предикаты, которые можно использовать: '<' | '<=' | '>=' | '>' | '<>' | '=' | 'lt' | 'le' | 'ge' | 'gt' | 'ne' | 'eq'
* Значения могут быть типов integer, Boolean, string, set, sequence, enum
* Встроенные операторы и функции +, -, \*, %, uni, intersect, differ, cardinal
* Функции только как методы типов
* Путь имеет вид v.a1. … .an или v.a1. … .an~ (переменная и имена атрибутов)
* Одна машина может появиться в SEES и INCLUDES (сначала в SEES добавлена, потом в INCLUDES) – это нужно убирать руками
* Не проверяется, если уже импортирована машина некоторого подтипа (тогда машину типа не нужно импортировать)
* instance type метакласса ассоциаций должен быть определен в самом метаклассе, не может быть двух атрибутов с одним метаклассом ассоциаций (если нужно, можно сделать общий суперкласс). Это делается потому, что переменная метакласса ассоциаций кладется в машину типа атрибута. Поэтому же для пары инверсных ассоциаций следует определять лишь один метакласс, и инверсию указывать в одном атрибуте (иначе будут кольцевые ссылки).

### Лишние элементы в AMN-модели

* для каждой переменной создается переменная, не все используются (в путях создаются выражения, переменные не все используются)
* для параметров создаются подстановки, нужны только для выходных
* если эффект представляется вызовом операции, то голова пути не используется
* если нет выходных параметров у метода, то не нужна подстановка ANY в уточняемой машине, только skip (если есть – не нужен skip)
* инварианты типа экземпляров метакласса ассоциаций трансформируются lazy-правилом; предикаты, сформированные по умолчанию остаются невостребованными