

# Main

Camilo Correa Restrepo

July 1, 2022

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Function for transforming the templates</b>	<b>1</b>

This document describes the VariaMos Semantic Translator. It is an interactive notebook written in Org-Mode with python literate programming features. It is similar to but more powerful than Jupyter notebooks since it is fully customizable and it provides a superior code editing experience.

## 1 Introduction

The purpose of this software is to take in VariaMos-formatted JSON with a model of some type, and, using a set of transformation rules: (a) generate a machine-interpretable representation of the model; (b) execute the model in the chosen language; (c) and, return any diagnostic info to VariaMos.

We now turn to defining where we will find our files for processing:

```
FILE = "/home/kaiser185/workspace/semantic_translator/json/vmosfm.json"
RULES = "/home/kaiser185/workspace/semantic_translator/json/fmrules.json"
```

## 2 Function for transforming the templates

Now that we have our imports and files, we shall create a set of functions that will perform the transformation of a given constraint template into the necessary text:

```
import json
import re
```

```

from minizinc import Instance, Model, Solver

def replaceWithPattern(pattern, string, occ, v):
    if type(v) is not str and string is not None:
        print(v.items())
        print(string)
        [string := string.replace(occ, str(val)) for (k, val) in v.items()]
        print('OK')
        return string

def replaceExpr(bundle, elems, rels, cons, p):
    regex = r"\w?" + re.escape(p) + r"\w?"
    occs = re.findall(regex, cons)
    f = [iden for (k, r) in rels.items() for ((iden, _), _) in elems.items() if (str(r[
    fs = [
        iden
        for ((iden, _), elem)
        in elems.items()
        if ([rel for (_, rel) in rels.items() if rel["targetId"] == iden])
    ]
    fs = ["uuid_" + ef.replace("-", "_") for ef in fs if (ef not in f)]
    print(fs)
    pattern = {
        "F": f[0],
        "Xs": {
            "sum": "+".join(fs),
        }
    }
    # p = "Fs"
    [
        cons := (str(cons).replace(occ, v)
        if (pat == p and type(v) == str)
        else replaceWithPattern(pat, cons, occ, v))
        for (pat, v) in pattern.items()
        for occ in occs
    ]
    return cons

```

```

def bundleCons(bundle, elems, rels, rules):
    # get constraint rule
    rule = rules["elementTranslationRules"][0]["Bundle"]
    cons = rule["constraint"][bundle["properties"][1]["value"]]
    [cons := cs for cs in [
        replaceExpr(bundle, elems, rels, cons, p)
        for p in rule["param"]
    ] if cs is not None]
    return cons

def mapBundles(elems, rels, rules):
    return [
        bundleCons(bs, elems, rels, rules)
        for bs in [
            e if e["type"] == "Bundle" else None for ((iden, typ), e) in elems.items()
        ]
        if bs is not None
    ]

def mapVar(element, rule):
    # return rule
    if bool(rule):
        constraint = (
            rule["constraint"].replace(
                rule["param"], str(element["id"]).replace("-", "_")
            )
            + f'% {element["type"]} -> {element["id"]}'
        )
        return constraint
    # If not bool(rule) then return None

def mapVars(elems, rules):
    return [
        cs
        for cs in [
            mapVar(element, rules["elementTranslationRules"][0][typ])

```

```

        if (typ in rules["elementTypes"])
        else None
        for ((ident, typ), element) in elems.items()
    ]
    if cs is not None
]

def mapCons(relation, rule):
    if bool(rule):
        acc = rule["constraint"]
        [
            acc := acc.replace(
                p,
                str(
                    relation[("source" if p == rule["params"][0] else "target") + "Id"]
                ).replace("-", "_"),
            )
            for p in rule["params"]
        ]
        return acc

def mapRels(relations, rules):
    return [
        rs
        for rs in [
            mapCons(
                v, rules["relationTranslationRules"][0][v["properties"][0]["value"]]
            )
            for (k, v) in [
                (k, rel) for (k, rel) in relations.items() if rel["properties"]
            ]
            if (v["properties"][0]["value"] in rules["relationTypes"])
        ]
        if rs is not None
    ]

```

Next we need to construct our result:

*# Load file*

```

with open(FILE, "r") as f:
    # Load json as obj
    array = json.load(f)
    # Get the feature model @ /productLines[0]/domainEngineering/models[0]
    fm = array["productLines"][0]["domainEngineering"]["models"][0]
    # Get the elements
    elementsMap = {(e["id"], e["type"]): e for e in fm["elements"]}
    # Get the relationships
    relationsMap = {r["id"]: r for r in fm["relationships"]}
    # Create the rules
    with open(RULES, "r") as r:
        rules = json.load(r)
        # Map the constraints for the vars
        constraints = (
            mapVars(elementsMap, rules)
            + mapRels(relationsMap, rules)
            + mapBundles(elementsMap, relationsMap, rules)
            + ["solve satisfy;"]
        )
    print(constraints)
    print("-----")
    print("\n".join([c for c in constraints]))
    # Add model and solver
    gecode = Solver.lookup("gecode")
    model = Model()
    model.add_string("\n".join([c for c in constraints]))
    instance = Instance(gecode, model)
    result = instance.solve()
    print(result)

['uuid_ac0d2916_749b_4146_ad32_37622e2aeef0', 'uuid_9e5a250c_9ee7_4d7b_9486_40563a1e9a']
dict_items([('sum', 'uuid_ac0d2916_749b_4146_ad32_37622e2aeef0+uuid_9e5a250c_9ee7_4d7b_9486_40563a1e9a')])
constraint :: "bf3ab018-6304-4e84-a11f-80f3f5d1d80f XOR Xs" (bf3ab018-6304-4e84-a11f-80f3f5d1d80f XOR Xs)
OK
dict_items([('sum', 'uuid_ac0d2916_749b_4146_ad32_37622e2aeef0+uuid_9e5a250c_9ee7_4d7b_9486_40563a1e9a')])
constraint :: "bf3ab018-6304-4e84-a11f-80f3f5d1d80f XOR Xs" (bf3ab018-6304-4e84-a11f-80f3f5d1d80f XOR Xs)
OK
dict_items([('sum', 'uuid_ac0d2916_749b_4146_ad32_37622e2aeef0+uuid_9e5a250c_9ee7_4d7b_9486_40563a1e9a')])
constraint :: "bf3ab018-6304-4e84-a11f-80f3f5d1d80f XOR Xs" (bf3ab018-6304-4e84-a11f-80f3f5d1d80f XOR Xs)
OK

```

```

['uuid_ac0d2916_749b_4146_ad32_37622e2aeef0', 'uuid_9e5a250c_9ee7_4d7b_9486_40563a1e9ab8']
["var 1..1:'uuid_69784178_c589_4447_bbe5_7b51b97f4918';% RootFeature -> 69784178-c589-4447-bbe5-7b51b97f4918"]
-----
var 1..1:'uuid_69784178_c589_4447_bbe5_7b51b97f4918';% RootFeature -> 69784178-c589-4447-bbe5-7b51b97f4918
var 0..1:'uuid_bf3ab018_6304_4e84_a11f_80f3f5d1d80f';% AbstractFeature -> bf3ab018-6304-4e84-a11f-80f3f5d1d80f
var 0..1:'uuid_ac0d2916_749b_4146_ad32_37622e2aeef0';% AbstractFeature -> ac0d2916-749b-4146-ad32-37622e2aeef0
var 0..1:'uuid_9e5a250c_9ee7_4d7b_9486_40563a1e9ab8';% ConcreteFeature -> 9e5a250c-9ee7-4d7b-9486-40563a1e9ab8
var 0..1:'uuid_43634fef_d816_4cc4_bbde_02cb7865afef';% ConcreteFeature -> 43634fef-d816-4cc4-bbde-02cb7865afef
var 0..1:'uuid_87b866ef_e358_4797_829c_d3fcac43a21f';% ConcreteFeature -> 87b866ef-e358-4797-829c-d3fcac43a21f
var 0..1:'uuid_e51771f2_b0cc_433a_bfee_8e106bb8d17e';% AbstractFeature -> e51771f2-b0cc-433a-bfee-8e106bb8d17e
var 0..1:'uuid_1cb2b338_f05e_4ccb_9df2_2bc76894336a';% ConcreteFeature -> 1cb2b338-f05e-4ccb-9df2-2bc76894336a
var 0..1:'uuid_b2f0093c_60b1_40a0_98d6_ab392dcc74cc';% ConcreteFeature -> b2f0093c-60b1-40a0-98d6-ab392dcc74cc
constraint :: "69784178_c589_4447_bbe5_7b51b97f4918 mandatory bf3ab018_6304_4e84_a11f_80f3f5d1d80f"
constraint :: "69784178_c589_4447_bbe5_7b51b97f4918 mandatory ac0d2916_749b_4146_ad32_37622e2aeef0"
constraint :: "bf3ab018_6304_4e84_a11f_80f3f5d1d80f optional 9e5a250c_9ee7_4d7b_9486_40563a1e9ab8"
constraint :: "ac0d2916_749b_4146_ad32_37622e2aeef0 mandatory e51771f2_b0cc_433a_bfee_8e106bb8d17e"
constraint :: "e51771f2_b0cc_433a_bfee_8e106bb8d17e mandatory 1cb2b338_f05e_4ccb_9df2_2bc76894336a"
constraint :: "e51771f2_b0cc_433a_bfee_8e106bb8d17e optional b2f0093c_60b1_40a0_98d6_ab392dcc74cc"
constraint :: "b2f0093c_60b1_40a0_98d6_ab392dcc74cc excludes 87b866ef_e358_4797_829c_d3fcac43a21f"
constraint :: "9e5a250c_9ee7_4d7b_9486_40563a1e9ab8 includes 43634fef_d816_4cc4_bbde_02cb7865afef"
constraint :: "bf3ab018-6304-4e84-a11f-80f3f5d1d80f XOR Xs" (bf3ab018-6304-4e84-a11f-80f3f5d1d80f
solve satisfy;

```