# Main

Camilo Correa Restrepo

July 2, 2022

## Contents



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 represenation of the model; (b) execute the model in the chosen langugage; (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 replaceExprs(bundle, elems, rels, cons, params, complexT):
    """
    This function replaces the first and second expressions for a bundle's constraint.
    """
    f = [
        iden
        for (k, r) in rels.items()
        for ((iden,_), _) in elems.items()
        if (
            str(r["sourceId"]) == str(iden) and
            str(r["targetId"]) == str(bundle["id"])
        )
    ]
    # replace constraint for principal param
    fs = [
        iden
        for ((iden, _), elem)
        in elems.items()
        if (
            [
                rel
                for (_,rel) in rels.items()
                if rel["sourceId"] == bundle["id"] and
                    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),
            }
        }
        cons = str(cons).replace(
            params[0],
            "uuid_" + pattern[params[0]].replace("-","_")
        )
        funs = r"(" + r"|".join(complexT["functions"]) + r")"
        regex_paren = funs + r"\(" + re.escape(params[1]) + r"\)"
        occs = set([
            oc.group(0)
            for oc in re.finditer(regex_paren, cons)
        ])
        [
            cons := cons.replace(
                occ,
                pattern[params[1]][
                    re.compile(regex_paren).search(occ).group(1)
                ]
            )
            for occ in occs
        ]
        return cons



def bundleCons(bundle, elems, rels, rules):
    """
    This is an auxiliary function that builds the request to replaceExprs
    """
    # get constraint rule
    rule = rules["elementTranslationRules"][0]["Bundle"]
    cons = rule["constraint"][bundle["properties"][1]["value"]]
    complexTrans = rules["complexElemTranslations"]
    return replaceExprs(bundle, elems, rels, cons, rule["param"], complexTrans)


def mapBundles(elems, rels, rules):
```

```
    """
    This function collects all the strings related to the bundles
    (it is the only portion of this module that is custom to feature models)
    """
    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):
    """Maps an element into a constraint according to the rules"""
    # 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):
    """This function collects all strings related to a set of elements and translation
    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):
    """This function maps a relation into a constraint according to the rules"""
    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):
    """This function collects all strings related to a set of relations and translatio
    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; we define therefore a function that takes in the model and the rule file and both generates the constraints and gets a solution from the solver:

We also define a function that allows us to test things locally before exposing our code to the server.

```
def test():
    """Test function locally"""
```

```
    # Load file
    with open(FILE, "r") as f:
        # Load json as obj
        model = json.load(f)
        # Create the rules
        with open(RULES, "r") as r:
            rules = json.load(r)
            x = run(model, rules, 'minizinc')
            print("----------------------RESULTS------------------------------")
            print(x)
            print("----------------------/RESULTS-----------------------------")

test()

----------------------MODEL---------------------------------
var 1..1:'uuid_69784178_c589_4447_bbe5_7b51b97f4918';% RootFeature -> 69784178-c589-444
var 0..1:'uuid_bf3ab018_6304_4e84_a11f_80f3f5d1d80f';% AbstractFeature -> bf3ab018-6304
var 0..1:'uuid_ac0d2916_749b_4146_ad32_37622e2aeef0';% AbstractFeature -> ac0d2916-749
var 0..1:'uuid_9e5a250c_9ee7_4d7b_9486_40563a1e9ab8';% ConcreteFeature -> 9e5a250c-9ee
var 0..1:'uuid_43634fef_d816_4cc4_bbde_02cb7865afef';% ConcreteFeature -> 43634fef-d81
var 0..1:'uuid_87b866ef_e358_4797_829c_d3fcac43a21f';% ConcreteFeature -> 87b866ef-e358
var 0..1:'uuid_e51771f2_b0cc_433a_bfee_8e106bb8d17e';% AbstractFeature -> e51771f2-b0c
var 0..1:'uuid_1cb2b338_f05e_4ccb_9df2_2bc76894336a';% ConcreteFeature -> 1cb2b338-f05
var 0..1:'uuid_b2f0093c_60b1_40a0_98d6_ab392dcc74cc';% ConcreteFeature -> b2f0093c-60b
constraint :: "69784178_c589_4447_bbe5_7b51b97f4918 mandatory bf3ab018_6304_4e84_a11f_8
constraint :: "69784178_c589_4447_bbe5_7b51b97f4918 mandatory ac0d2916_749b_4146_ad32_3
constraint :: "bf3ab018_6304_4e84_a11f_80f3f5d1d80f optional 9e5a250c_9ee7_4d7b_9486_40
constraint :: "ac0d2916_749b_4146_ad32_37622e2aeef0 mandatory e51771f2_b0cc_433a_bfee_8
constraint :: "e51771f2_b0cc_433a_bfee_8e106bb8d17e mandatory 1cb2b338_f05e_4ccb_9df2_2
constraint :: "e51771f2_b0cc_433a_bfee_8e106bb8d17e optional b2f0093c_60b1_40a0_98d6_ab
constraint :: "b2f0093c_60b1_40a0_98d6_ab392dcc74cc excludes 87b866ef_e358_4797_829c_d3
constraint :: "9e5a250c_9ee7_4d7b_9486_40563a1e9ab8 includes 43634fef_d816_4cc4_bbde_02
constraint :: "uuid_bf3ab018_6304_4e84_a11f_80f3f5d1d80f XOR Xs" (uuid_bf3ab018_6304_4e
solve satisfy;
----------------------/MODEL--------------------------------
----------------------RESULTS-------------------------------
Solution(uuid_69784178_c589_4447_bbe5_7b51b97f4918=1, uuid_bf3ab018_6304_4e84_a11f_80f3
----------------------/RESULTS------------------------------
```