JOHANNES KEPLER
UNIVERSITY LINZ

From Decision Models To User-Guiding Configurators Using SMT

Maximilian Heisinger, Florian Piminger, Martina Seidl
Institute for Symbolic AI

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Introduction & Background

Industry partner has a big legacy system we call *PropDM* that gives us exports in `DMX` (proprietary XML schema).

Introduction & Background

Industry partner has a big legacy system we call *PropDM* that gives us exports in `DMX` (proprietary XML schema).

They want to shift their CTO to ETO ratio to be more effective

Introduction & Background

Industry partner has a big legacy system we call *PropDM* that gives us exports in `DMX` (proprietary XML schema).

They want to shift their CTO to ETO ratio to be more effective

Couldn't do it, as PropDM is more similar to a programming environment than to a variability model.

Introduction & Background

Industry partner has a big legacy system we call *PropDM* that gives us exports in `DMX` (proprietary XML schema).

They want to shift their CTO to ETO ratio to be more effective

Couldn’t do it, as PropDM is more similar to a programming environment than to a variability model.

Configuring this is more like generating test data for software.

What are we dealing with?

Var1	Var2	Action 1	Action 2
> 10	CP 12*	Var1 = 5	
= 10	CP 13*	Var3 = 20	
= 10		Var2 = 20	Var3 = 40

```
def dg1(v):
    if v['Var1'] > 10 and re.match('12.*', v['Var2']): v['Var1'] = 5
    elif v['Var1'] == 10 and re.match('13.*', v['Var2']): v['Var3'] = 20
    elif v['Var1'] == 10:
        v['Var2'] = 20
        v['Var3'] = 40
```

Mapping to Decision Models and to our Contribution

We describe PropDM in terms of *Decision Models*, as this is the closest from the literature. Differences to the literature:

- No decision variables, instead all variables global.
- Every variable may be overridden at any point.
- Peculiarities of PropDM: GOTOs, special variables, different types of DGs.

Mapping to Decision Models and to our Contribution

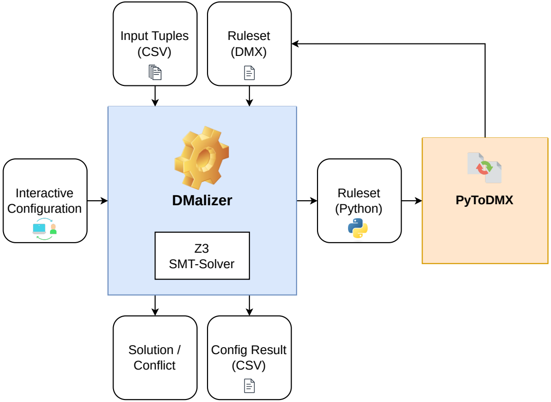
We describe PropDM in terms of *Decision Models*, as this is the closest from the literature. Differences to the literature:

- No decision variables, instead all variables global.
- Every variable may be overridden at any point.
- Peculiarities of PropDM: GOTOs, special variables, different types of DGs.

We built a user guiding non-linear configurator for this system using our contributions:

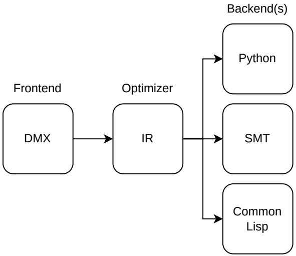
- An *SMT-Encoding* of these semantics and
- a method of using it to build such a configurator.

Our DMalizer Framework



Processing and Transpilation

1. Parsing a DMX file into our IR
2. Optimizing the IR and deducting sorts
3. Transpiling to different languages



DMalizer Execution Modes

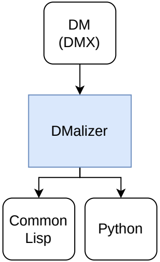


Figure 1: Transpiling a DM into Common Lisp or Python

DMalizer Execution Modes

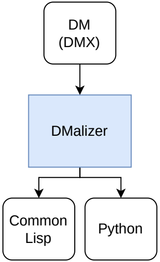


Figure 1: Transpiling a DM into Common Lisp or Python

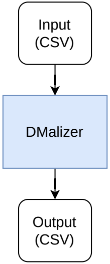


Figure 2: Emulating a run of PropDM

DMalizer Execution Modes

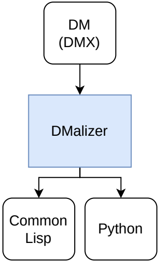


Figure 1: Transpiling a DM into Common Lisp or Python

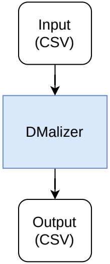


Figure 2: Emulating a run of PropDM

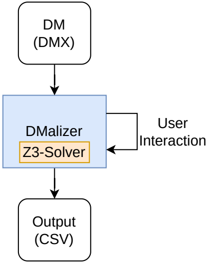


Figure 3: Interactive Artifact Configuration

Decision Models as SMT Formulas

Encoding structure:

- Declaring variables in single-static-assignment form (SSA)
- Asserting active decision groups
- Decisions in `ite` structure
- Linking variables (inside and between decision groups)
- Inactive implication chain
- Asserting values
- Calling the solver

Decision Models as SMT Formulas

Decisions

$(\texttt{ite}\ (\texttt{and}\ \alpha^i_j\ p^i_j)$
$\quad (\texttt{and}\ \tau(a^i_j)\ (\texttt{not}\ \alpha^i_{j+1})\ (=\ S^{i+1}\ a^i_j(S^i))$
$\quad (\texttt{=>}\ \alpha^i_j\ \alpha^i_{j+1})))$

Decision Groups

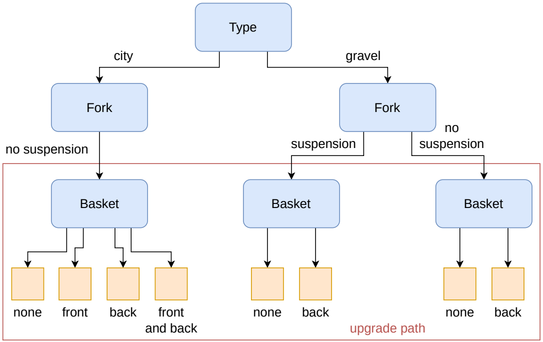
$(\texttt{and}\ (\texttt{ite}\ \alpha^i_0\ \alpha^i_1\ (\texttt{not}\ \alpha^i_1))$
$\quad \texttt{D[i,1] D[i,2] ... D[i,}|DG^i|\texttt{]}$
$\quad (\texttt{=>}\ (\texttt{not}\ \alpha^i_1)\ (\texttt{not}\ \alpha^i_2))\ (\texttt{=>}\ (\texttt{not}\ \alpha^i_2)\ (\texttt{not}\ \alpha^i_3))$
$\quad \texttt{...}\ (\texttt{=>}\ (\texttt{not}\ \alpha^i_{|DG^i|})\ (\texttt{not}\ \alpha^i_{|DG^i|+1}))$
$\quad (\texttt{=>}\ (\texttt{or}\ (\texttt{not}\ \alpha^i_0)\ \alpha^i_{|DG^i|+1})\ (=\ S^{i+1}\ S^i)))$

Bikeshop Decision Model

```
def RUL_TYPE(v):
    if v['TYPE'] == 'city': v['FORK'] = 'no suspension'

def RUL_UPGRADE(v):
    if v['TYPE'] == 'gravel' and v['BASKET'] == 'front':
        v['ERROR'] = 1
    elif v['TYPE'] == 'gravel' and v['BASKET'] == 'front and back':
        v['ERROR'] = 1
```

- Variables with only one choice are implied directly
- Gravel choice is not in the Python code because it implies no fork choice



Decision Models as SMT Formulas

Decisions in ite structure

```
(ite (and ACTIVE_1 (and (= PRE_TYPE "city")))
  (and (not ACTIVE_2)
   (and (= PRE_TYPE POST_TYPE)
    (= POST_FORK "no suspension")))
  (=> ACTIVE_1 ACTIVE_2))
```

- If decision is active and predicate is true, execute action
- Else imply next decision (active implication chain)

Decision Models as SMT Formulas

Linking variables inside decision groups

```
(=> (or (not ACTIVE) ACTIVE_2)
  (and (= PRE_TYPE POST_TYPE)
   (= PRE_FORK POST_FORK)))
```

- If decision group is inactive or no matching decision applied, link variables from last to next state

Inactive implication chain

```
(and (=> (not ACTIVE_1) (not ACTIVE_2)))
```

- If first decision is inactive, next decisions are inactive

Check it out!

- SMT encoding for linear decision models
- Enhance decision models with non-linear and user-guiding configuration
- Scales to (at least) hundreds of decisions

maximaximal.pages.sai.jku.at/vamos24

JKU

JOHANNES KEPLER
UNIVERSITY LINZ