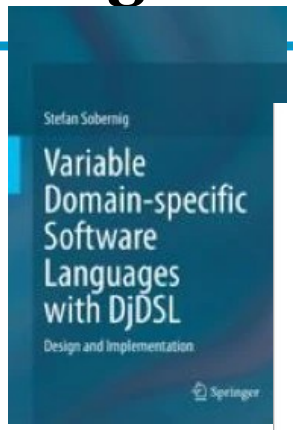
# Designing, implementing, ~~and using~~ UVL as a language-kernel extension

**Stefan Sobernig**

FOSD-Treffen 2023, Ulm

# Commonalities vs. specificities in (problem-space) variability modelling




VaMoS'21


- Variability modelling
  - **general-purpose** modelling, e.g.: Universal Variability Language (UVL), T extual Variability Language (TVL);
  - **special-purpose** modelling, e.g.:
    - modelling video variants (VM; Alférez et al., 2019);
    - modelling variants of measurement devices ("automotive testbeds");
    - conditional-branching sub-language embedded into a Questionnaire Language (QL);
- As a community, we are regularly confronted with the **known clash** (a.k.a. "**tools conundrum**") between
  - making available modelling-language tools targeting a maximally large (industry, scientific) audience (also, across domains) *and*
  - the well documented need for highly specialised modelling-language tools, esp. in industry, down to the levels of single organisational units or software engineers;

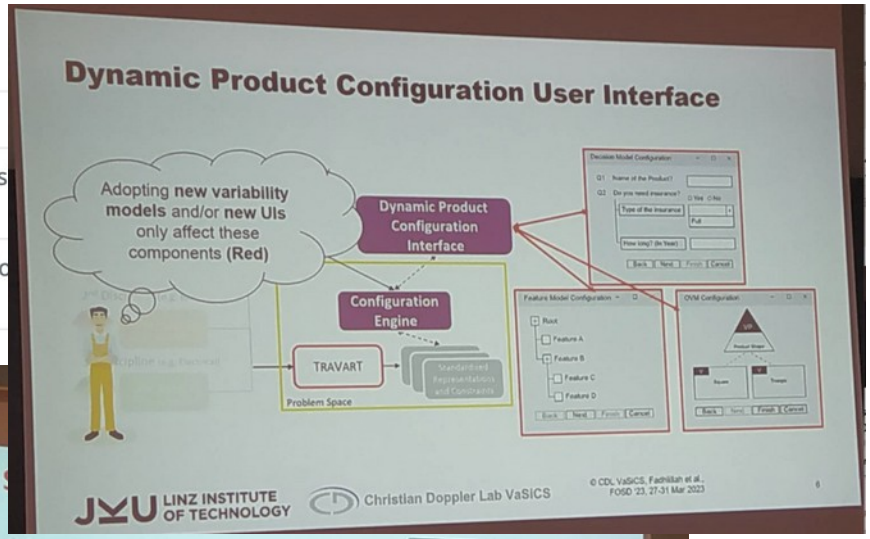
see Selic (2012), MODEVAR workshop series

# UVL, TVL, „Not Invented Here“, …
## *Heard that before at FOSD 2023?*

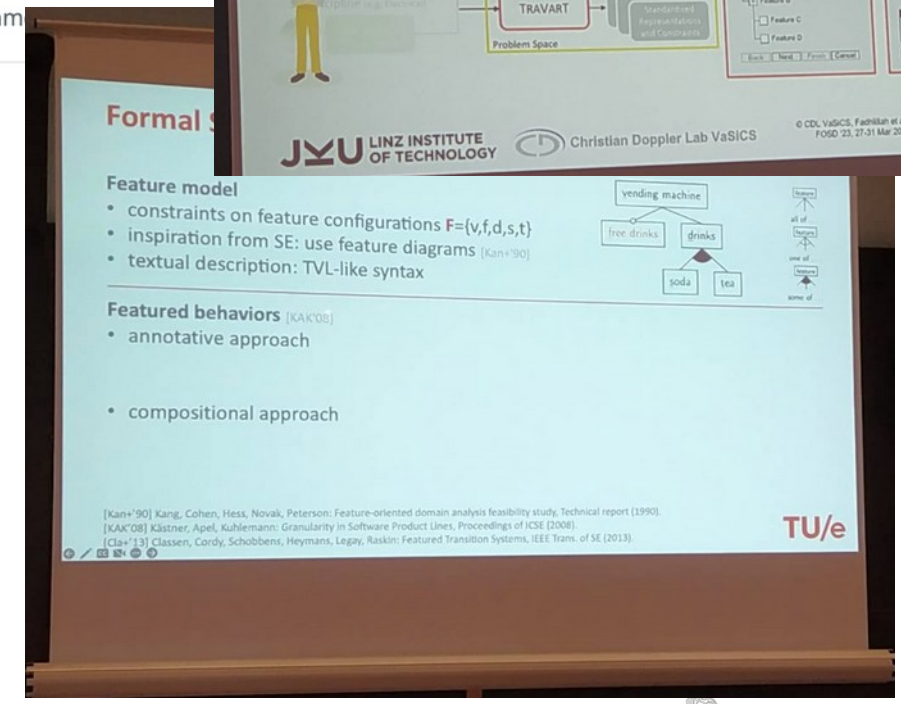
WU WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

### Session 3 (Tuesday 13:00) Chair: Norbert Siegmund ↑

| Name | Title |
|---|---|
| Sandro Schulze | Will we ever agree? An eye-tracking study on program comprehens… CPP-based variability |
| Stefan Vill | Language Levels for the Universal Variability Language: An Extensio… Mechanism and Conversion Strategies |
| Daniel Jesús Muñoz Guerra | Extended Variability Models, Algebra, and Arithm… |

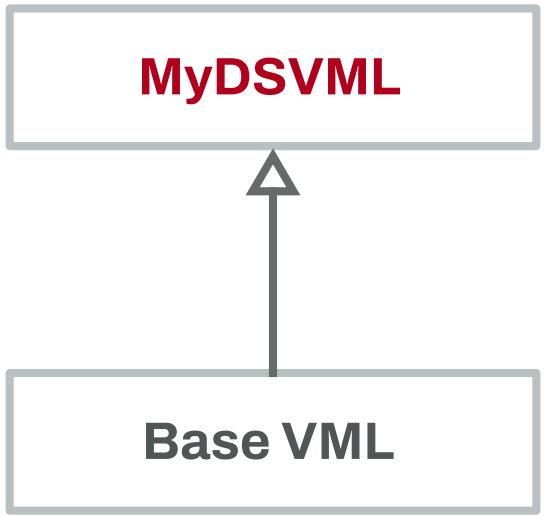


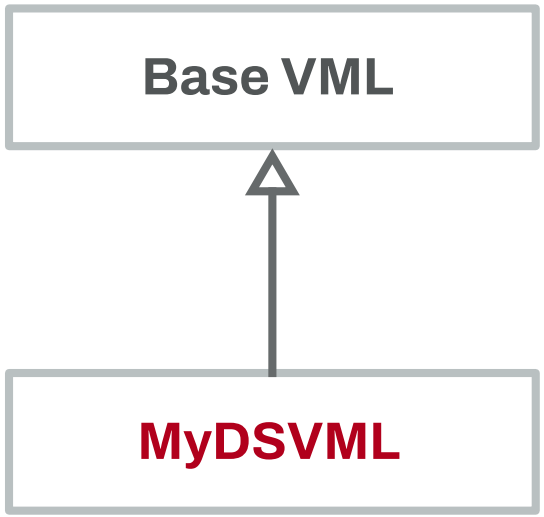





EQUIS ACCREDITED AACSB ACCREDITED ASSOCIATION AMBA ACCREDITED

# Language specialisation vs. extension?



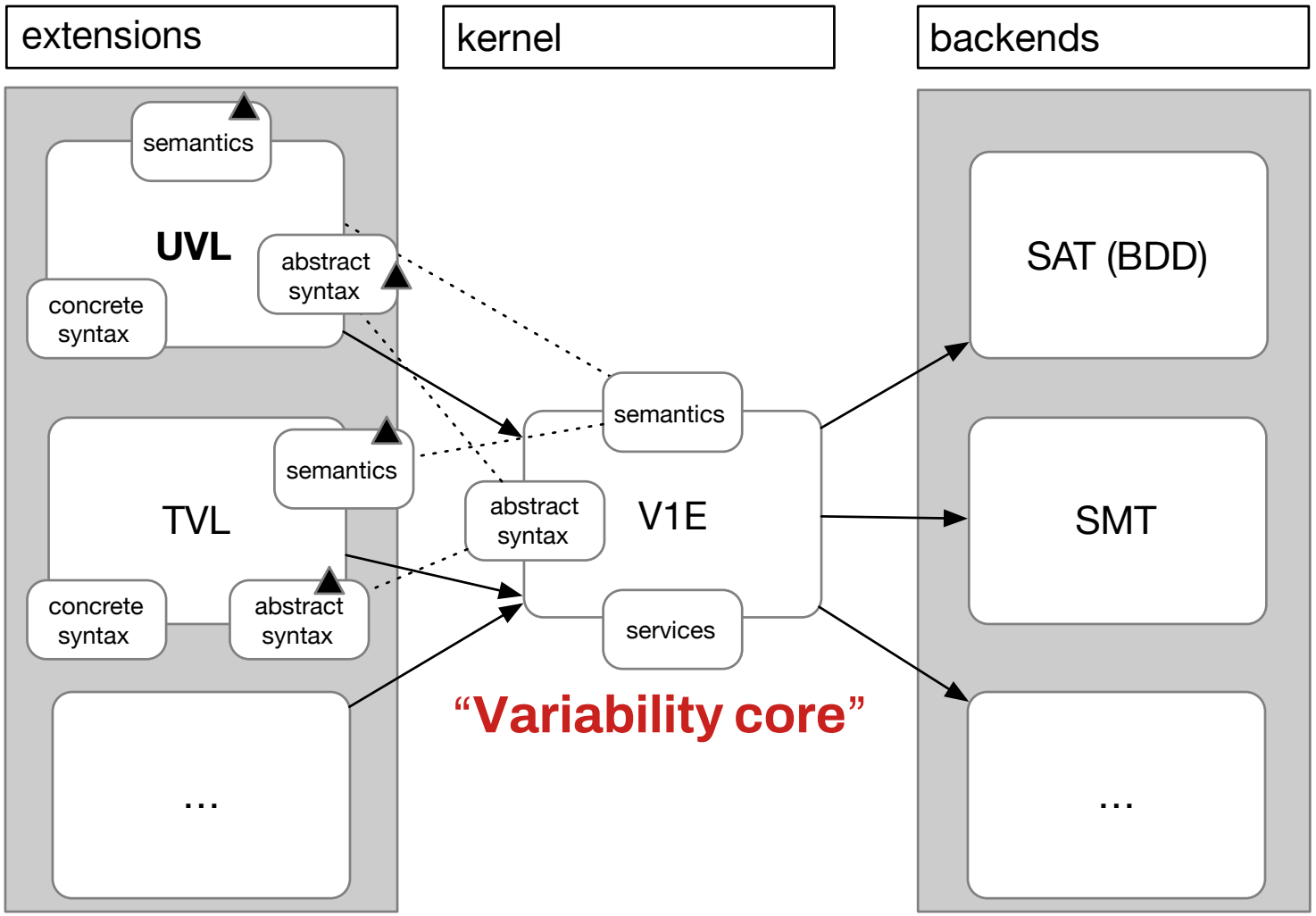




- ✔ Increase **integratability** into existing modelling tools/ toolchains
- ✔ Help meet **safety** requirements (e.g., via increasing verifiability)
- ⚠ Allows for **planned, restricted reuse!**

- ✔ Serve a **new need** in modelling (or analysis).
- ✔ Facilitate building domain-specific textual **system front-ends**;
- ⚠ Allows for **unplanned reuse open to extension**;

Adapted from Diomidis Spinellis (2001): „Notable design patterns for domain-specific languages“. JSS, 56(1):91–99, DOI: 10.1016/S0164-1212(00)00089-3)

# V1E

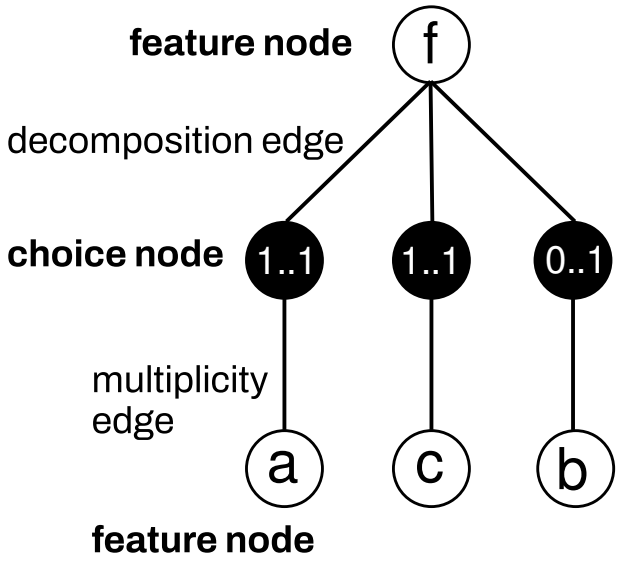

extensions
kernel
backends
semantics
UVL
abstract syntax
concrete syntax
semantics
TVL
concrete syntax
abstract syntax
...
semantics
abstract syntax
V1E
services
"Variability core"
SAT (BDD)
SMT
...

# UVL as a V1E extension (ex.)

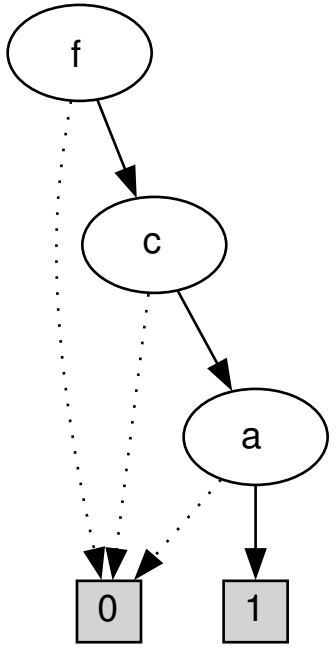
**Concrete syntax** (UVL)

```
namespace ns
features
        f
            mandatory
                a
                c
            optional
                b
```

**Abstract syntax** (V1E)




**Backend representation** (BDD)



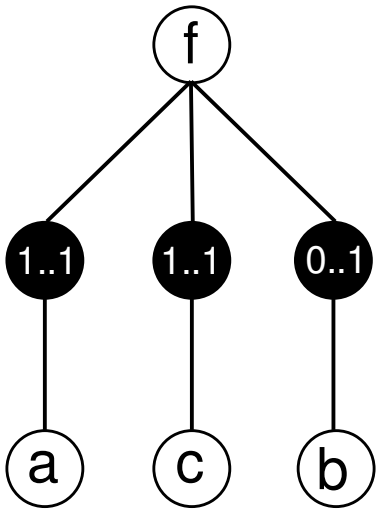

Schobbens et al. (2007): Generic semantics of feature diagrams. Computer Networks 51(2):456–479

# TVL as a V1E extension (ex.)

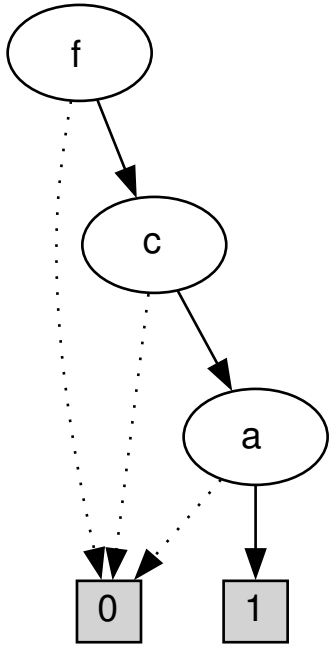
**Concrete syntax** (TVL)

```
root f {
  group [3..3] {
    a, c, opt b
  }
}
```

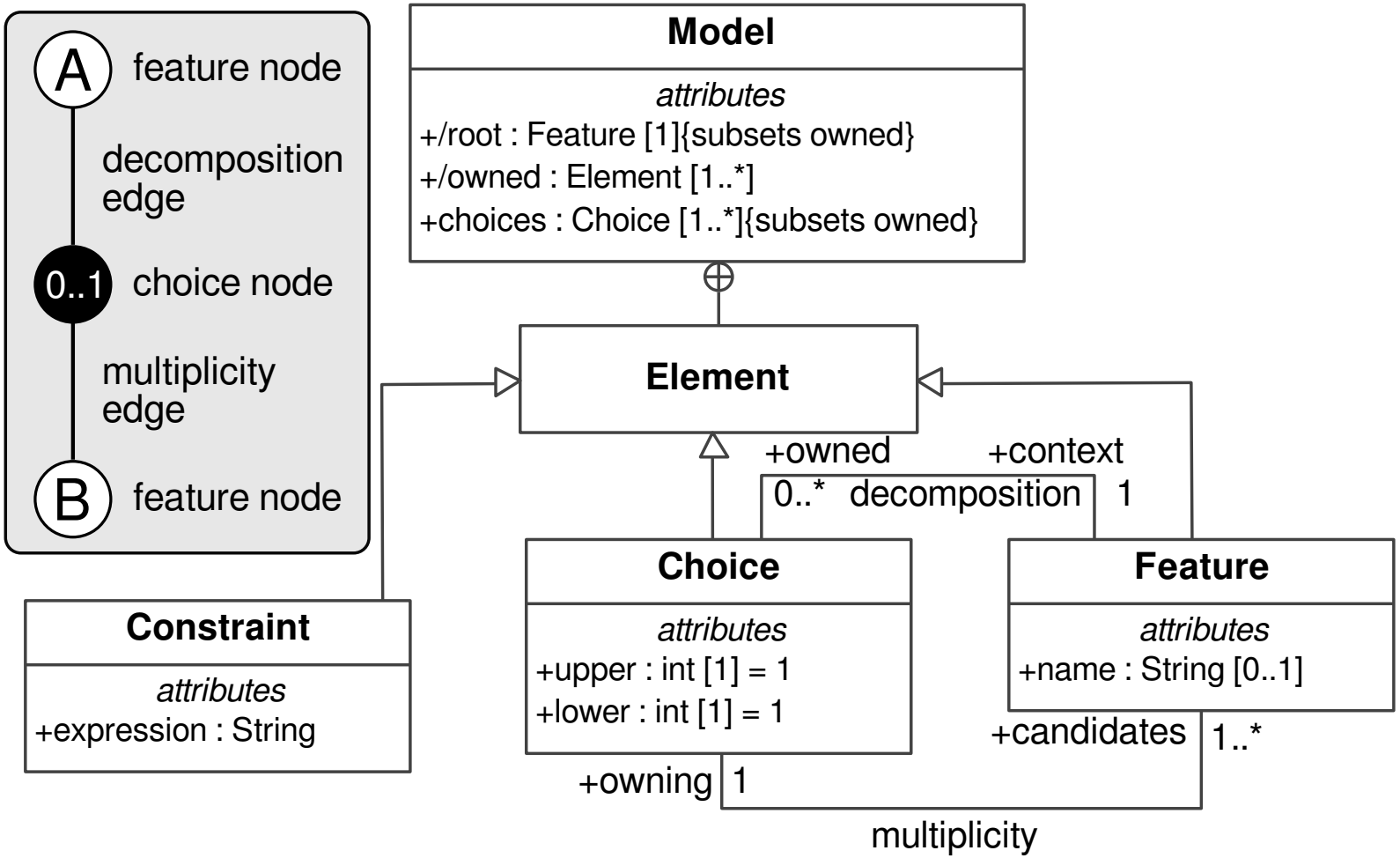
**Abstract syntax** (V1E)



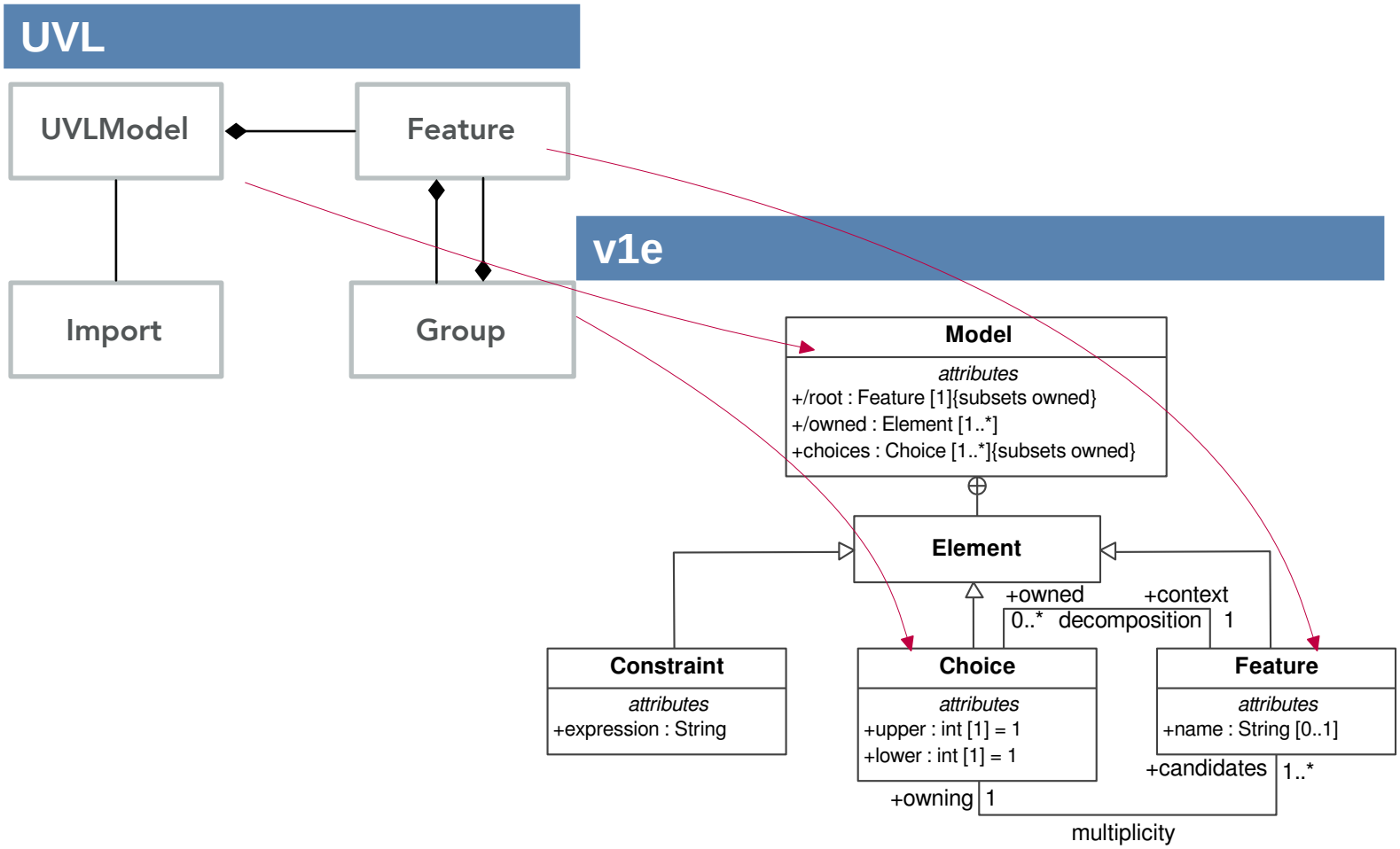

**Backend representation** (BDD)



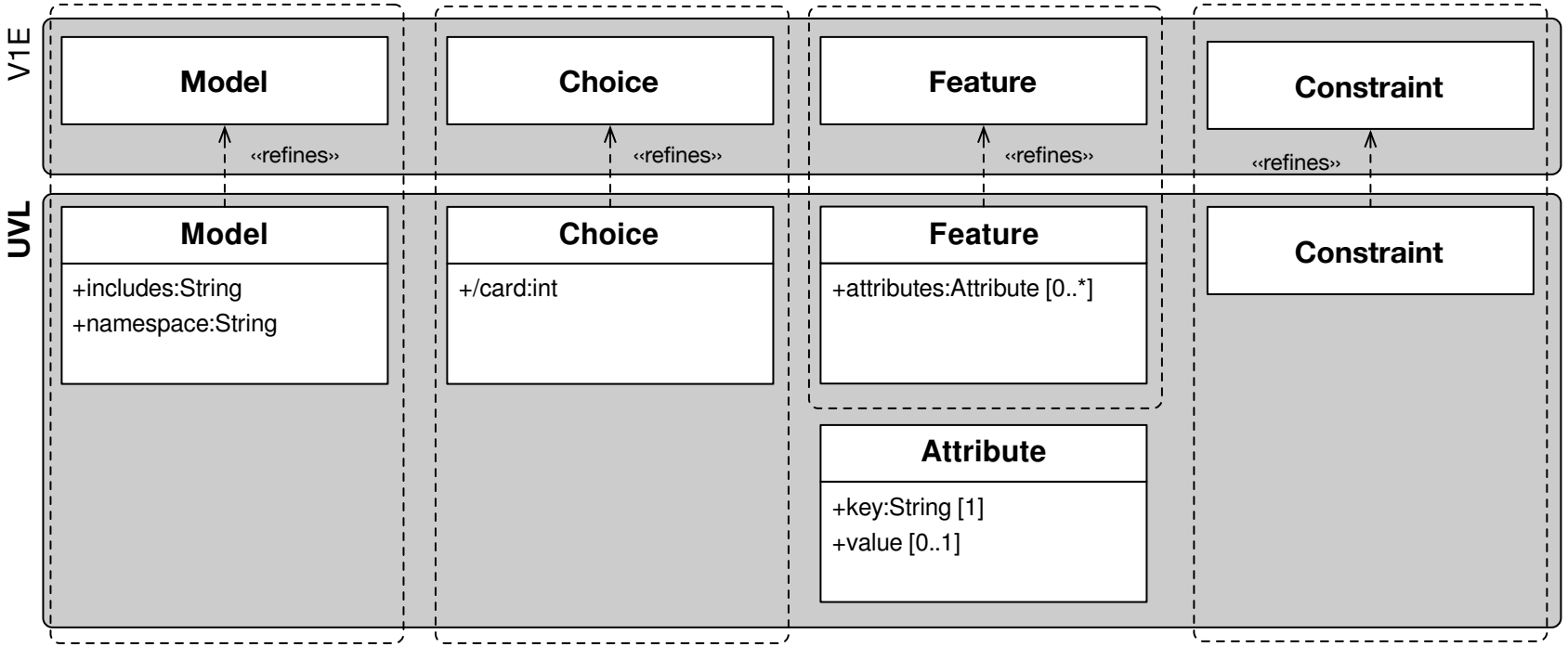
# V1E: Abstract syntax



# UVL abstract-syntax model (excerpt): Mapping to V1E



# V1E/UVL as an Abstract-Syntax Extension



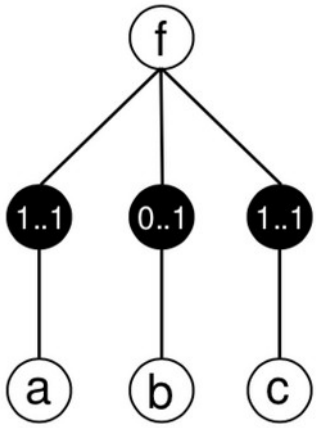

"**Collaboration**-based Design"

# Grammar-based syntax definition and processing (1)

```
namespace ns
features
        f
            mandatory
                a
                c
            optional
                b
```

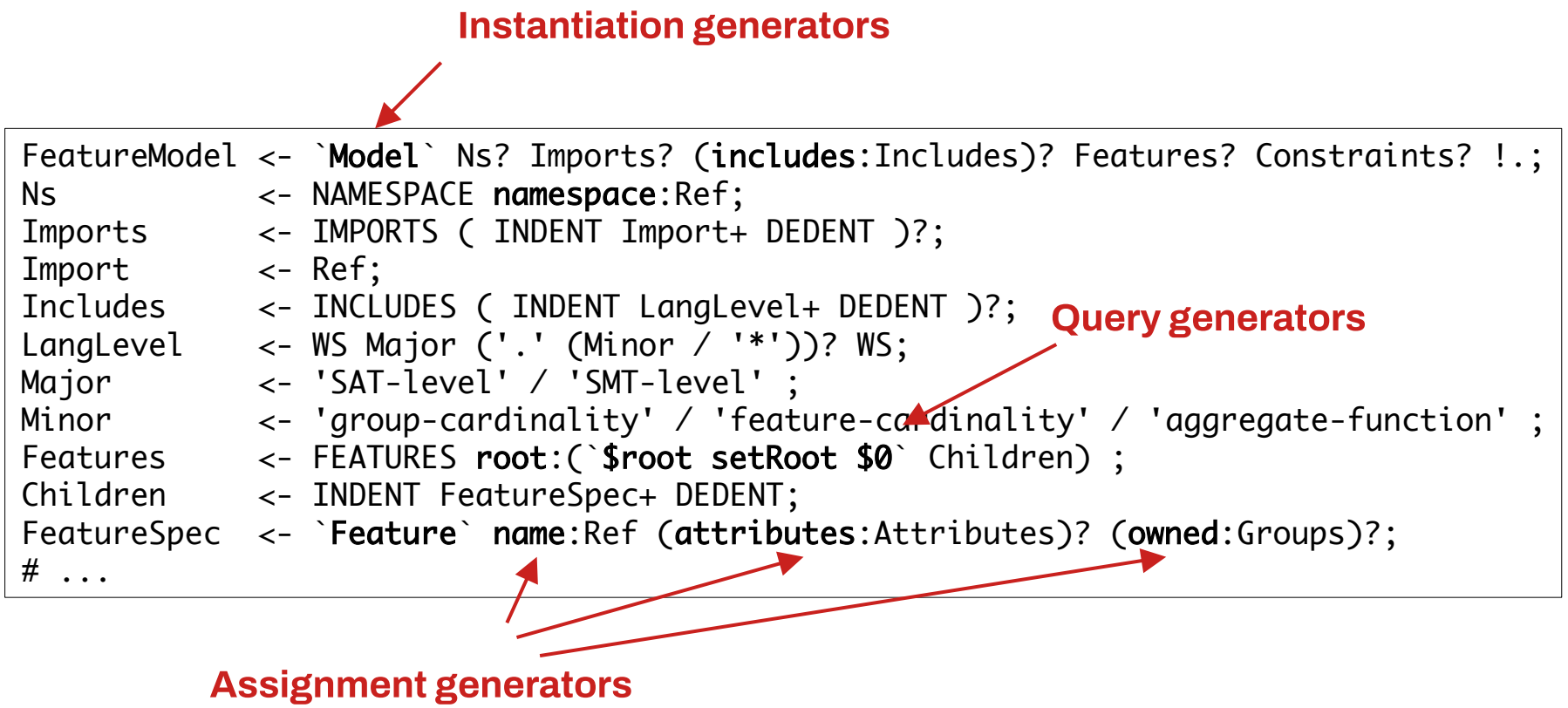
(1) Concrete-syntax definition

(2) Instantiation




**Two-in-one** grammar definition (a.k.a "object grammar")

# Grammar-based syntax definition and processing (2)



# State of V1E/UVL

| UVL | Concrete syntax | Abstract syntax | Semantics/ services |
|---|---|---|---|
| Feature hierarchy | ● | ● | ● |
| Feature groups* | ● | ● | ● |
| Attributes | ● | ● | ◒ |
| Composition** | ● | ● | ○ |
| Cross-tree constraints | ● | ◒ | ◒ |
| Group cardinalities | ● | ● | ● |
| Feature cardinalities | ○ | ○ | ○ |
| Levels*** | ● | ○ | ○ |

* mandatory, alternative, or, optional
** imports, namespaces
*** major, minor

*Based on Stefan Vill, „Language Levels for the Universal Variability Language: An Extension Mechanism and Conversion Strategies“, 2022*

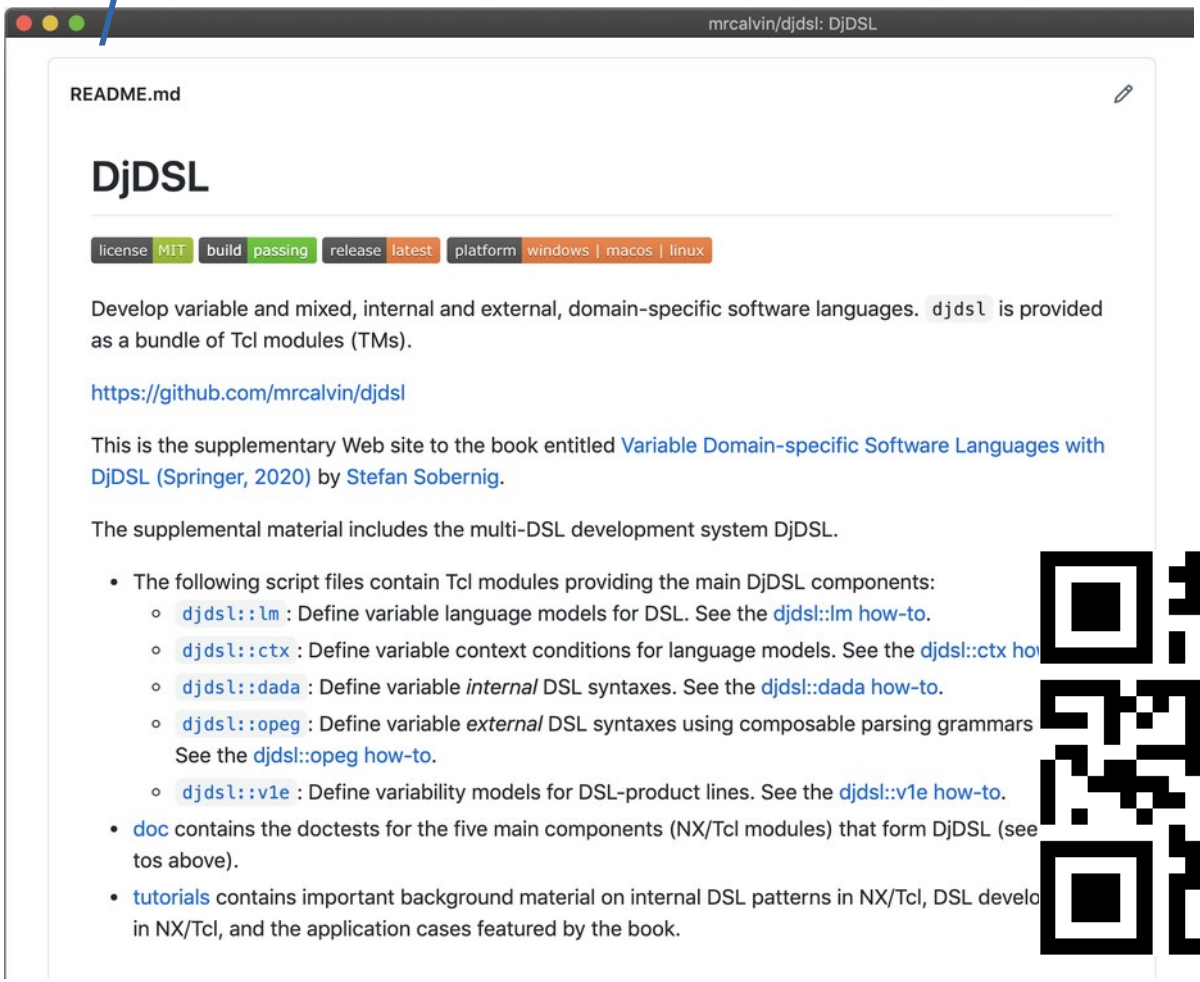
# Let's discuss

- **Contributions**:
  - Extensible abstract syntax: Extensible language-kernel model;
  - Extensible concrete syntax: Composable object parsing-expression grammars (OPEGs);
  - Semantics: Versatile multiplicity encoding and extensible constraint definitions;
  - One analysis backend: BDD representation (using tclbdd);
  - v1e/UVL proof-of-concept implementation (~400 SLOC, 66 E/BNF-like grammar)
- **Additional limitations**:
  - Just one backend (tclbdd);
  - Limited backend services (counting/ enumeration services only)
- **Next steps**
  - Complete v1e/UVL (e.g., feature cardinalities, composition)
  - Towards a "language product-line": Identify common assets between language implementations, ...
  - "Stresstesting" using the UVL model collection (What are the language-level affordances, exactly?)
  - Extended and new backend services, based on Sundermann et al. (2021): "Applications of #SAT Solvers on Feature Models". In: Proc. VaMoS '21.

# Thank your for your attention!

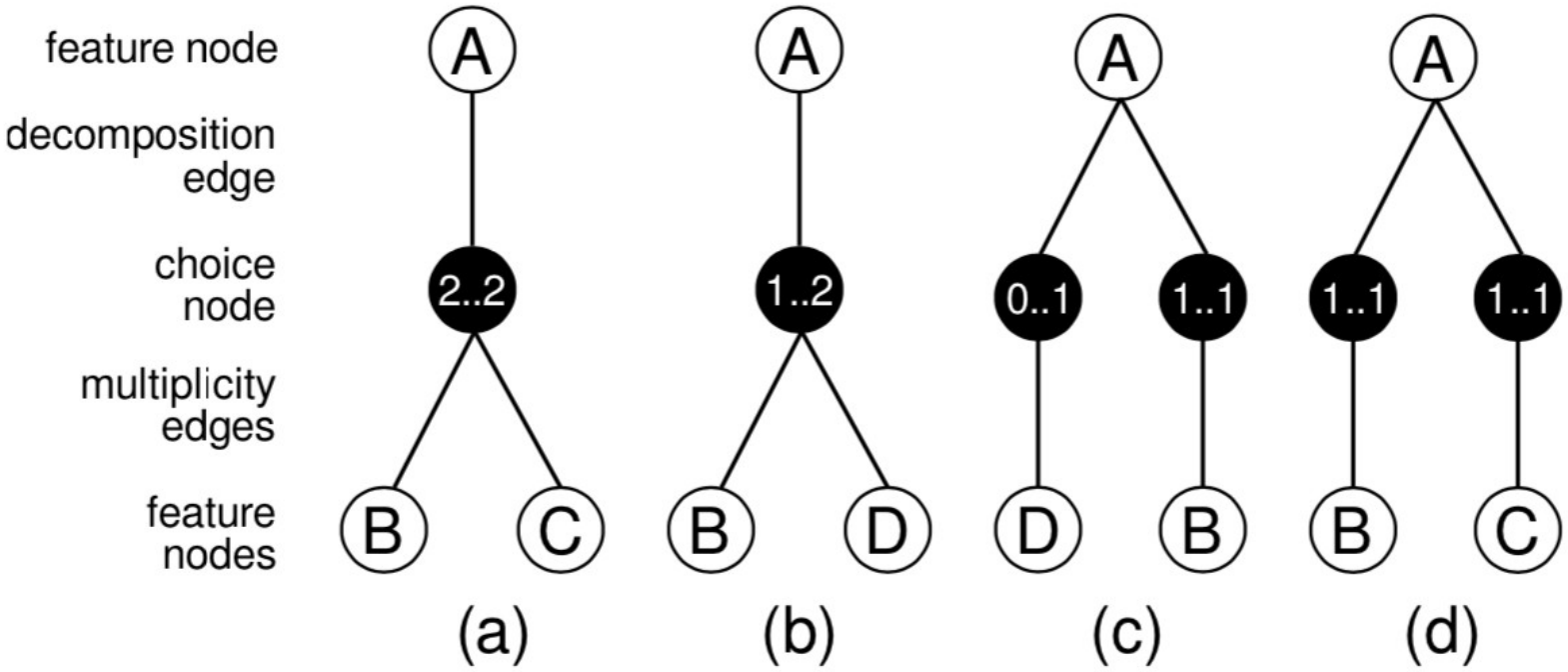
WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

## https://github.com/mrcalvin/djdsl




EQUIS ACCREDITED AACSB ACCREDITED ASSOCIATION AMBA ACCREDITED

# Multiplicity encoding of variation points(ex.)




**explicit and-** choice

**inclusive-or** choice

**optional** choice

**implicit** (“split”) **and**-choice

# Multiplicities and Boolean correspondences

| | | | |
|---|---|---|---|
| Optional sub-feature | 0..1 | $\|\text{CAND}\| = 1$ | implication ($\Leftarrow$) |
| Mandatory sub-feature | 1..1 | $\|\text{CAND}\| = 1$ | bi-implication ($\Leftrightarrow$) |
| Inclusive-or group of sub-features | 1..$n$ | $n = \|\text{CAND}\|, \|\text{CAND}\| > 1$ | disjunction ($\vee$) |
| Exclusive-or group of sub-features | 1..1 | $\|\text{CAND}\| > 1$ | at-most-one (e.g., binomial enc.) |
| And group of sub-features | $s$..$s$ | $s \in \mathbb{N}_{>0}, s = \|\text{CAND}\|$ | conjunction $\wedge$ |
| Absent (negated) sub-feature | 0..0 | (for constraints) | negation $\neg$ |
| Lower bound | $j$..$n$ | $n = \|\text{CAND}\|, j \in \mathbb{N}_0, j \leq n$ | at-least-j (e.g., binomial enc.) |
| Upper bound | 0..$k$ | $k \in \mathbb{N}, 1 \leq k \leq \|\text{CAND}\|$ | at-most-k (e.g., binomial enc.) |
| Lower and upper bound | $j$..$k$ | $j, k \in \mathbb{N}_0, j \leq k \leq \|\text{CAND}\|$ | at-least-j $\wedge$ at-most-k |

Multiplicity-based encoding of v1e model (choices) and the operators used in the corresponding Boolean formula.