60 Years of SAT Solving

– Applications to Automotive Configuration –

21 February 2020

Prof. Dr. Wolfgang Küchlin

Dipl.-Inform., Dr. sc. techn. (ETH)

Symbolic Computation Group
Wilhelm-Schickard-Institute of Informatics
Faculty of Mathematics and Sciences

Universität Tübingen

Steinbeis Technology Transfer Centre
Object- und Internet-Technologies (STZ OIT)

Wolfgang.Kuechlin@uni-tuebingen.de
http://www-sr.informatik.uni-tuebingen.de

SR

Variant-rich automotive configuration

- Variant-rich „individualized mass production“
  - Configure 1 out of ~10^{30} cars, observing Boolean constraints
- 2 Levels: Product description (PD) + parts list (BoM)
  - High Level PD: Which configurations can be built?
    - Boolean constraints system, ~500KB, 1000s constraints, ~10^{30} solutions
    - Daimler: PÜ; VW: MBT; BMW: VRM; GM: VDS; Renault, Peugeot, …
  - Low Level BoM: Which parts go into each configuration?
    - List of all (10,000s) parts for each model line (e.g. C Class, A4, Golf)
    - Boolean conditions (*if* cond *then* part) select the parts for each order
- **Mechanical Theorem Proving → Formal Verification**
  - verify properties, detect defects, answer queries, optimize
  - with respect to the full theoretical variance (~10^{30} orders)

Mercedes High-Level Product Description

Example: E-Class

- approx. 1.500 codes (options, countries, ..)
- approx. 3.000 rules in product overview (PÜ / PD)
- rule-based BOM with approx. 35,000 parts.
- B(code): condition for presence of code in order
- Z(code): condition for automatic addition of code

B(P09) = 297+540+543;

B(610) = (512 / 527 / 528)+608+ -978;

Z(P09) = (M271 + -M013 / M272) + 830 /
Z04+ -(M273 + Z27)

Z(682) = 623 / 830 / 513L

B(450) = L+965+ 670+837+ -P34+
((M271/M651)+(953/955)+(100A/200A)+(334/335)+(301/336/337) /
/ M642+(2XXL/557L/571L)+(953/955)+(100A/200A)+(334/335)+ (301/337));

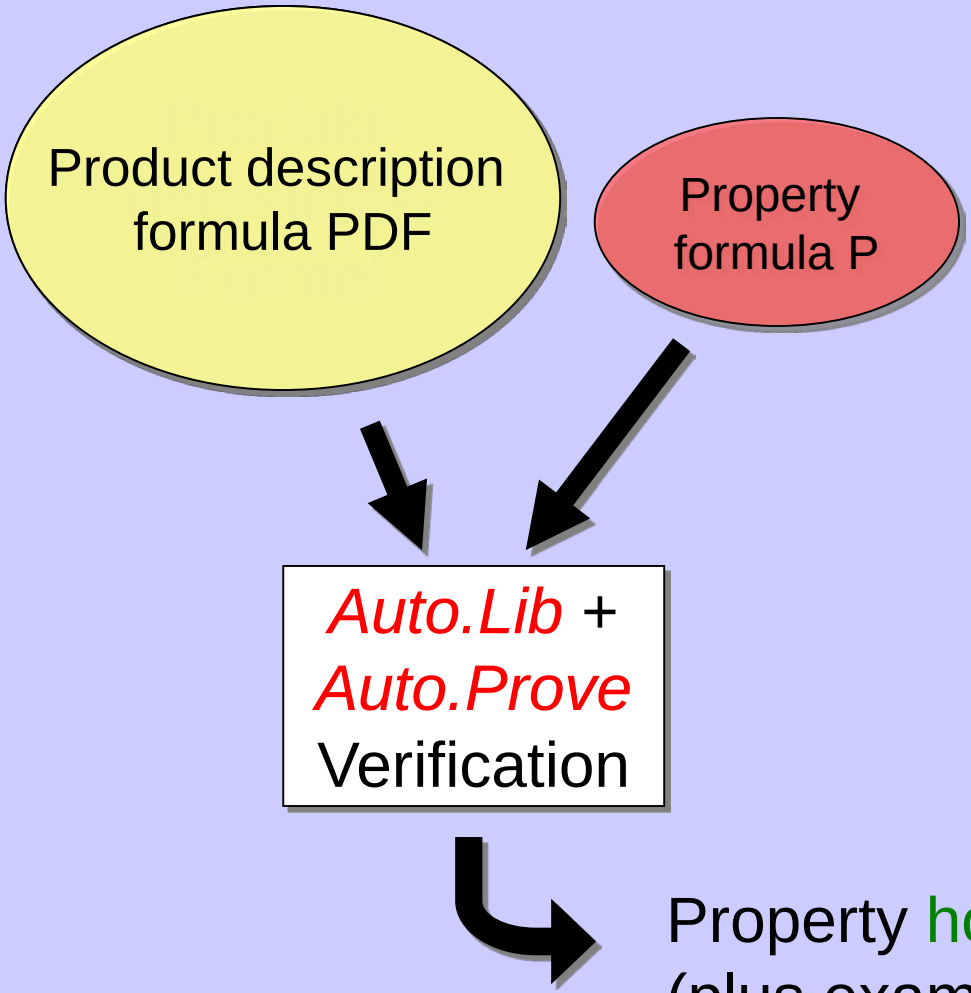
Low-Level Product Description (BOM)

➢ BOM – Bill-of-Materials

- List of Materials (parts, software, colors) for entire car model line
- Grouped by Functionality (e.g. steering wheels, headlights,…)
- Within group: List of alternative parts (materials)
- List of Tupels: <Material, Boolean Selection Condition>

ZB	Benennung	Codebedingung
POS	ZB RAD 34R	
ZB	ZBU RAD MIT REIFEN 34R	
ZB	RAD / 7J X 16H2 + 205/55 R16 91V	34R+R01+V22;
ZB	RAD / 7J X 16H2 + 205/55 R16 91V	34R+R01+V22+(470/475);
ZB	RAD / 7J X 16H2 + 205/55 R16 91W	34R+R01+V50;
ZB	RAD / 7J X 16H2 + 205/55 R16 91W	34R+R01+V50+(470/475);
ZB	RAD / 7J X 16H2 + 205/55 R16 91H	34R+R01+V21;
ZB	RAD / 7J X 16H2 + 205/55 R16 91H	34R+R01+V21+(470/475);
ZB	RAD / 7J X 16H2 + 205/55 R16 91W	34R+R01+V50+M272+M35+M012;
ZB	RAD / 7J X 16H2 + 205/55 R16 91H AL...	34R+R02+V21;
ZB	RAD / 7J X 16H2 + 205/55 R16 91H M...	34R+645+V21;
ZB	RAD / 7J X 16H2 + 205/55 R16 91H M...	34R+645+V21+(470/475);
ZB	RAD / 7J X 16H2 + 205/55 R16 91H M...	34R+645+V21+M272+M35+M012;

Verification by Auto.Prove Mechanical Prover



Standard Checks on PD

- Codes which CANNOT be selected
- Codes which MUST be selected
- Per Model line, model, country …

Standard Checks on BOM

- No-Hits (part is missing for some car)
- Double-Hits (some car gets 2 parts)
- Orphaned parts (no car gets the part)

Individual (free) Queries

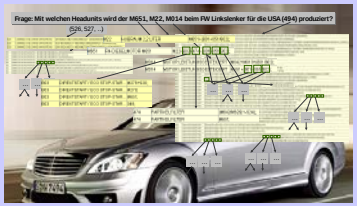
- Codes optional in US, ….
- Parts deliverable to Japan, …
- → **Configurator**

Property holds always / does not hold always
(plus example of car / proof for no car)

Query Example: Explaining PD Inconsistency

Ex: Country code 817L „Taiwan“ impossible for station wagon, model type Cxxx

Analysis of product description which yields contradiction and pinpoints the problem:



Auto.Prove

Code rule for 817L (Taiwan):	L+M272+M30
Code rule for M30 (displacement):	M642 / R
Code rule for M642 (engine):	-M272

Conclusion:
Taiwan (817L) requires L+M272+M30. Since engine displacement M30 with motor M272 requires right-hand steering R, no orders for Taiwan (817L) can be manufactured.
→ The product overview is defective / incomplete and the wagon is impossible for Taiwan!

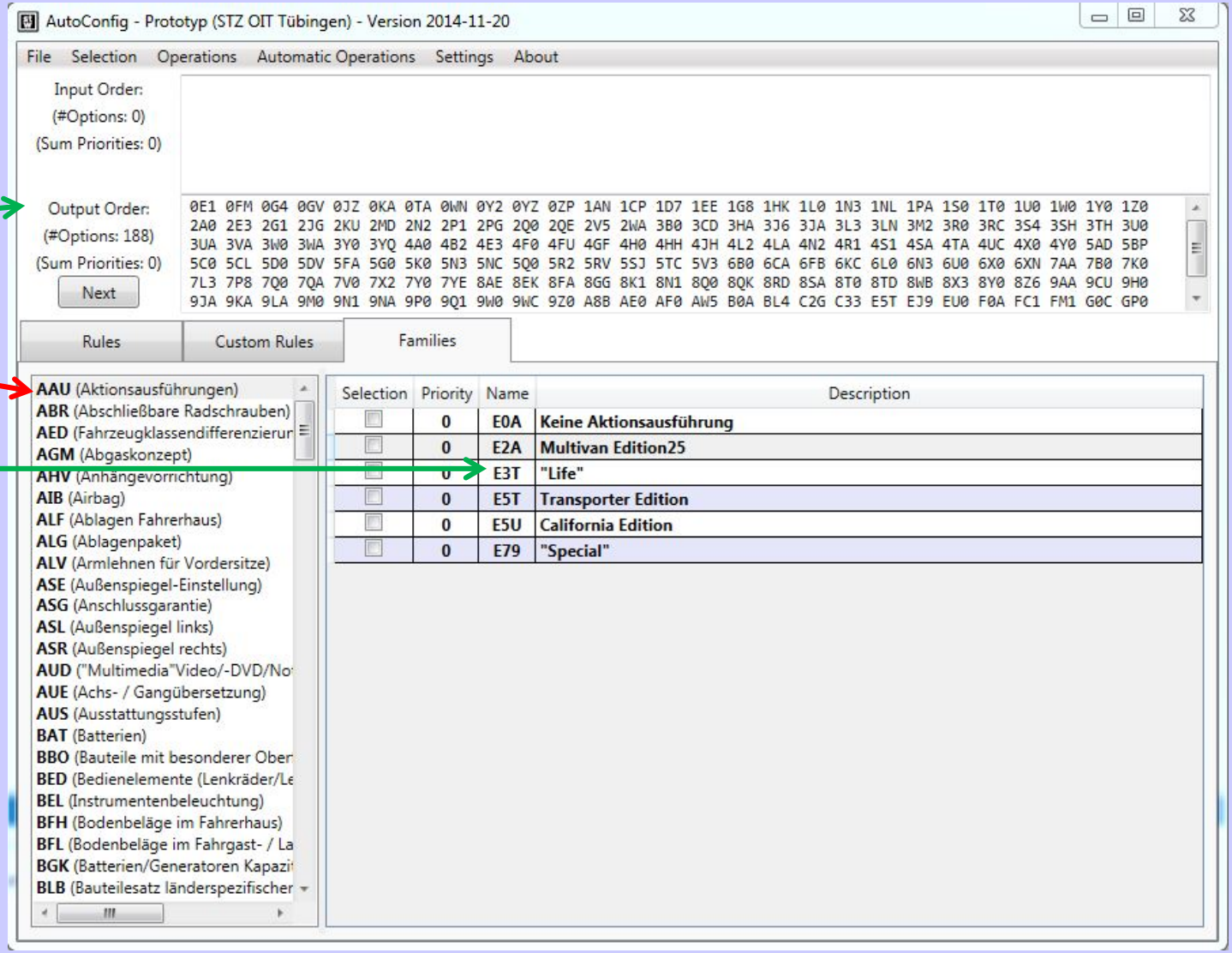
Defects in the product overview can potentially cause significant damage (Mio Euros):
- **Delivery of non certified Wheel/Tyre combinations to Asia**
- **Delivery of non certified aggregate combinations to China**
- **Delivery of headunits to Asia which can only display Latin characters**

Our SAT based configurator framework Auto.Config

- **Car configuration task**
  - User enters some option combination A+B+…
  - User selects any option family F in any order (e.g. motors, seats, wheels ..)
  - *Auto.Config* computes paths to success (SAT Solving the config rules)
    - which options in F can still be selected (which are implied / which are impossible)
    - and which BoM parts are already implied / which are impossible / still possible
  - User selects an option O in F and repeats the process with A+B+O
- **Re-configuration task**
  - User selects an impossible option O
  - *Auto.Config* computes (using a MaxSAT algorithm)
    - **minimal change** to order (undo previous selections) so O can be selected
- **Weak learning AI?**
  - we program only the Config framework, then load (learn?) PD and BoM **rules**
  - If Config is wrong, **error** can be in **programming**, or in the **rules**

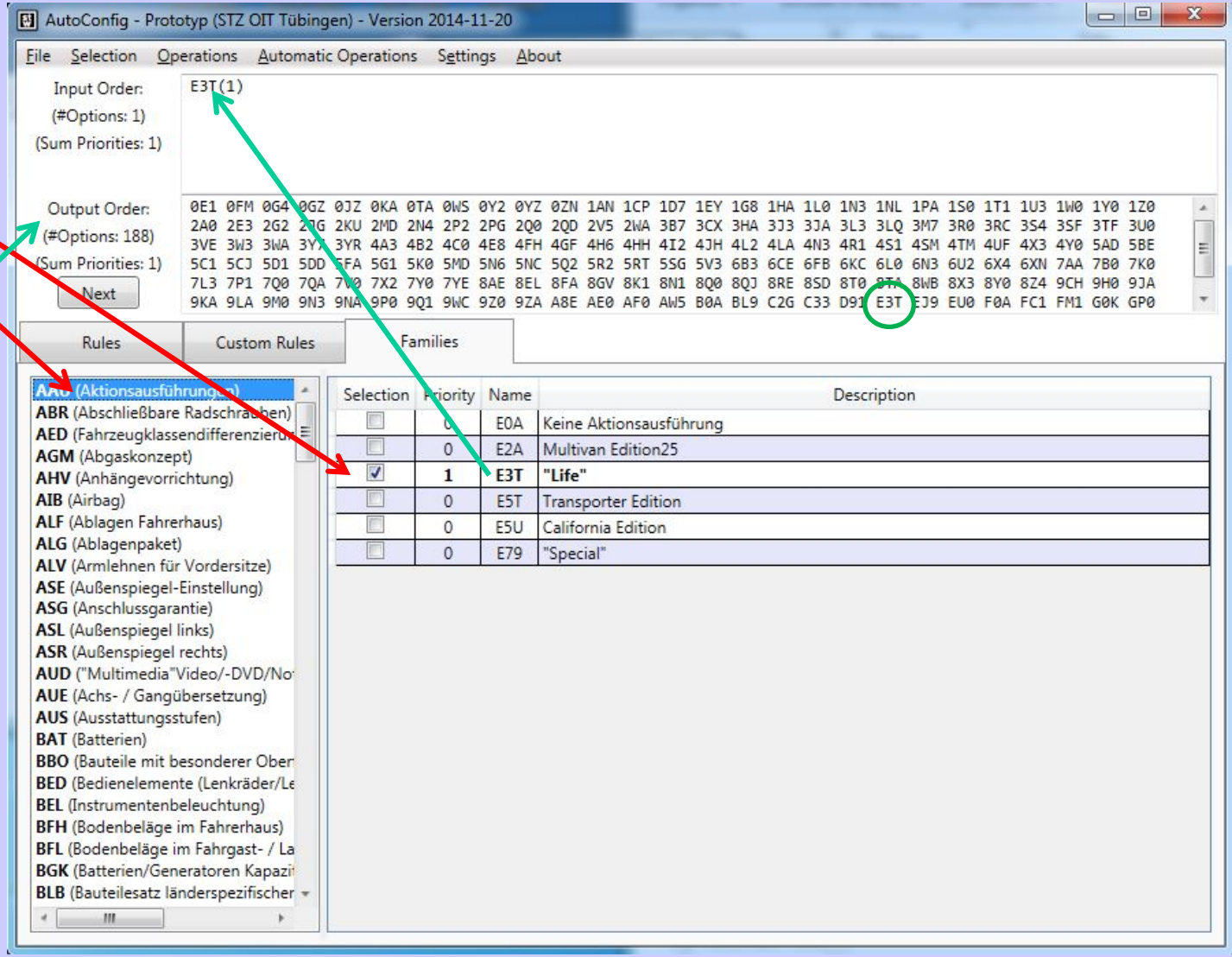
Example: Configuration with Auto.Config

- Start with a model
- *Auto.Config* computes arbitrary PD-valid order
- Select any option family
- **Bold:** *Auto.Config* computes options available for this model
- Grey: *Auto.Config* computes options not available for this model



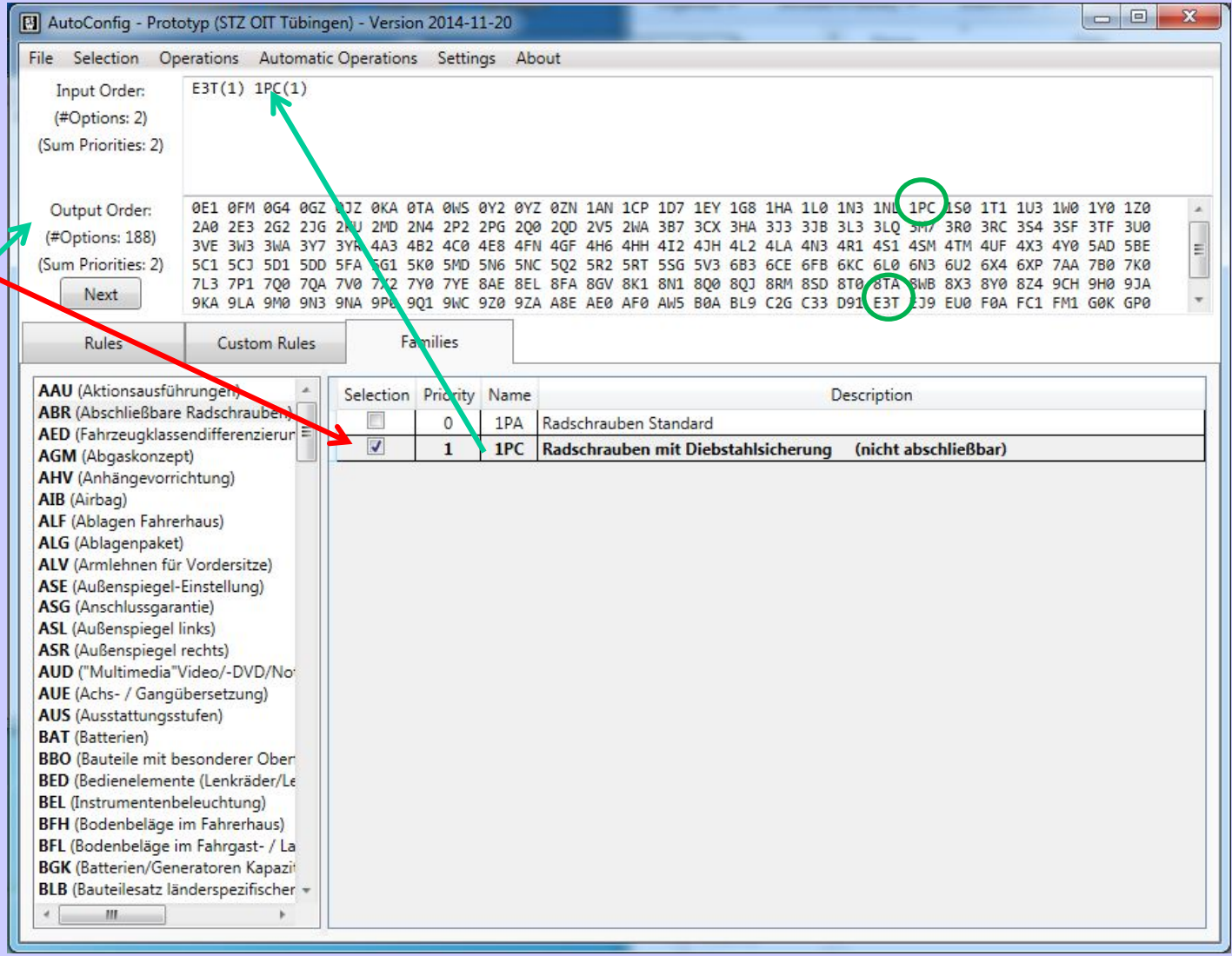
Example: Configuration with Auto.Config

- Select any option family
- Select any available option
- *Auto.Config* re-computes PD-valid order with selected option
- **Bold:** *Auto.Config* computes options available for this input order
- Grey: *Auto.Config* computes options not available for this input order



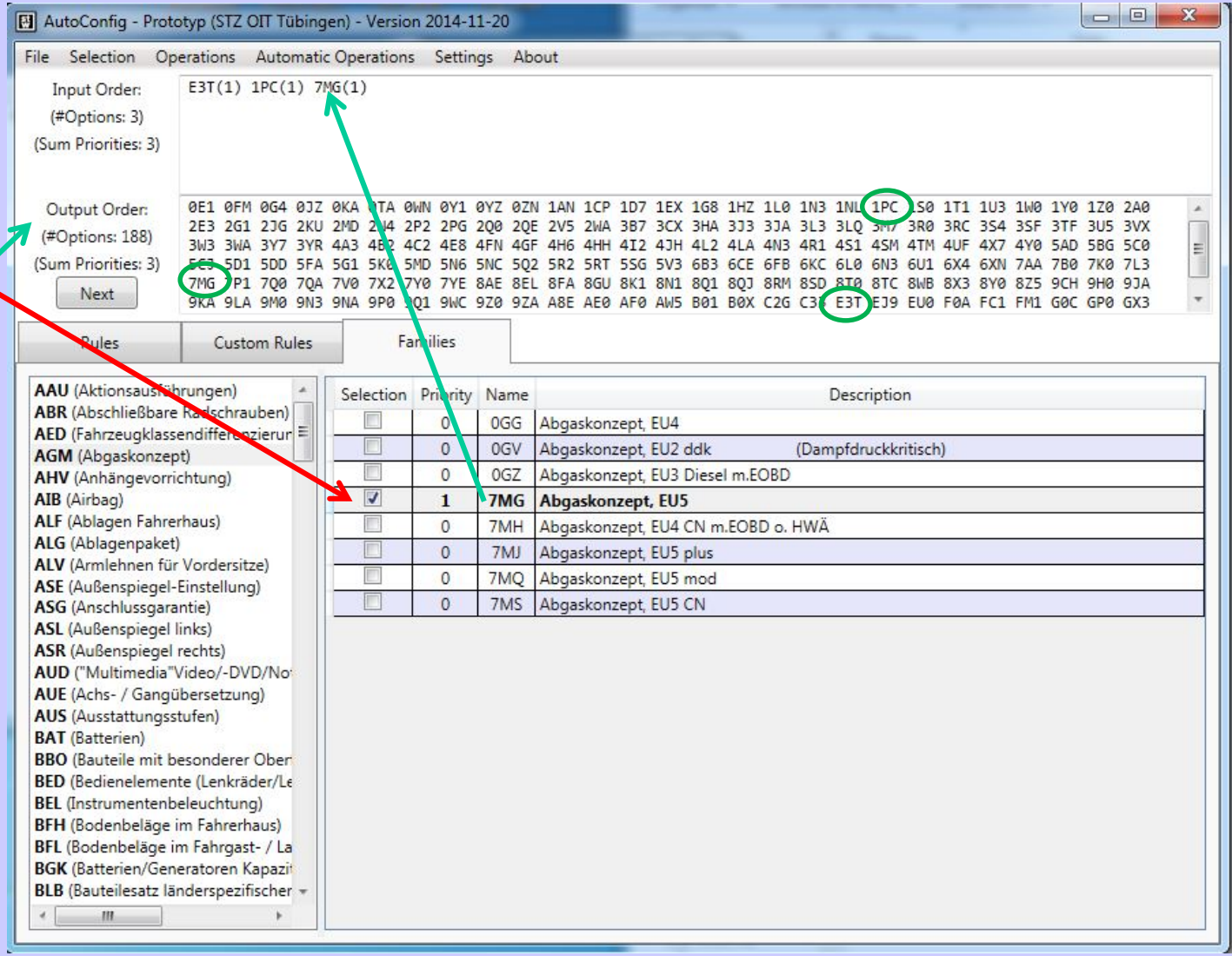
Example: Configuration with Auto.Config

- Select any option family
- Select any available option
- *Auto.Config* re-computes PD-valid order with selected option
- **Bold:** *Auto.Config* computes options available for this input order
- Grey: *Auto.Config* computes options not available for this input order



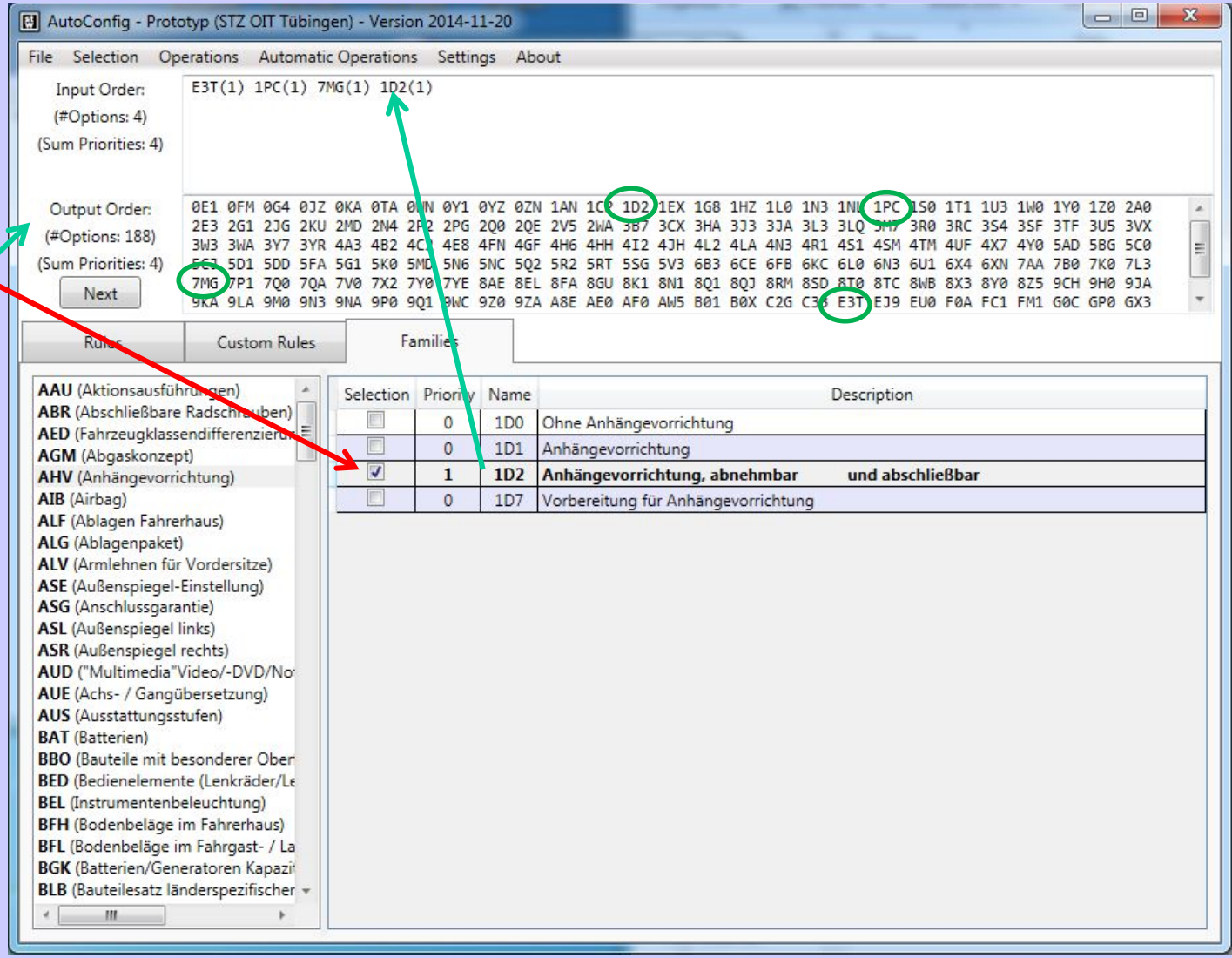
Example: Configuration with Auto.Config

- Select any option family
- Select any available option
- *Auto.Config* re-computes PD-valid order with selected option
- **Bold:** *Auto.Config* computes options available for this input order
- Grey: *Auto.Config* computes options not available for this input order



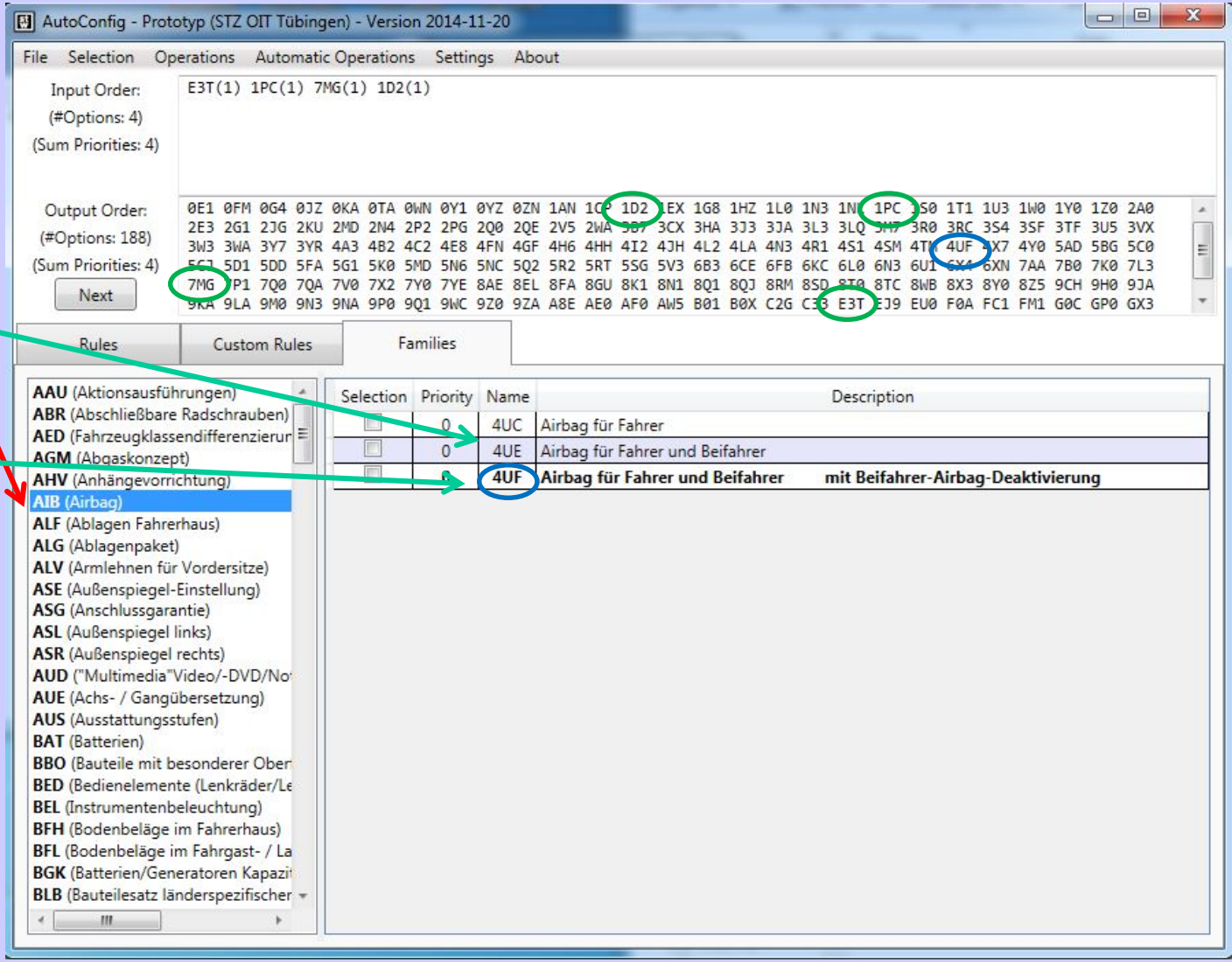
Example: Configuration with Auto.Config

- Select any option family
- Select any available option
- *Auto.Config* re-computes PD-valid order with selected option
- **Bold:** *Auto.Config* computes options available for this input order
- Grey: *Auto.Config* computes options not available for this input order



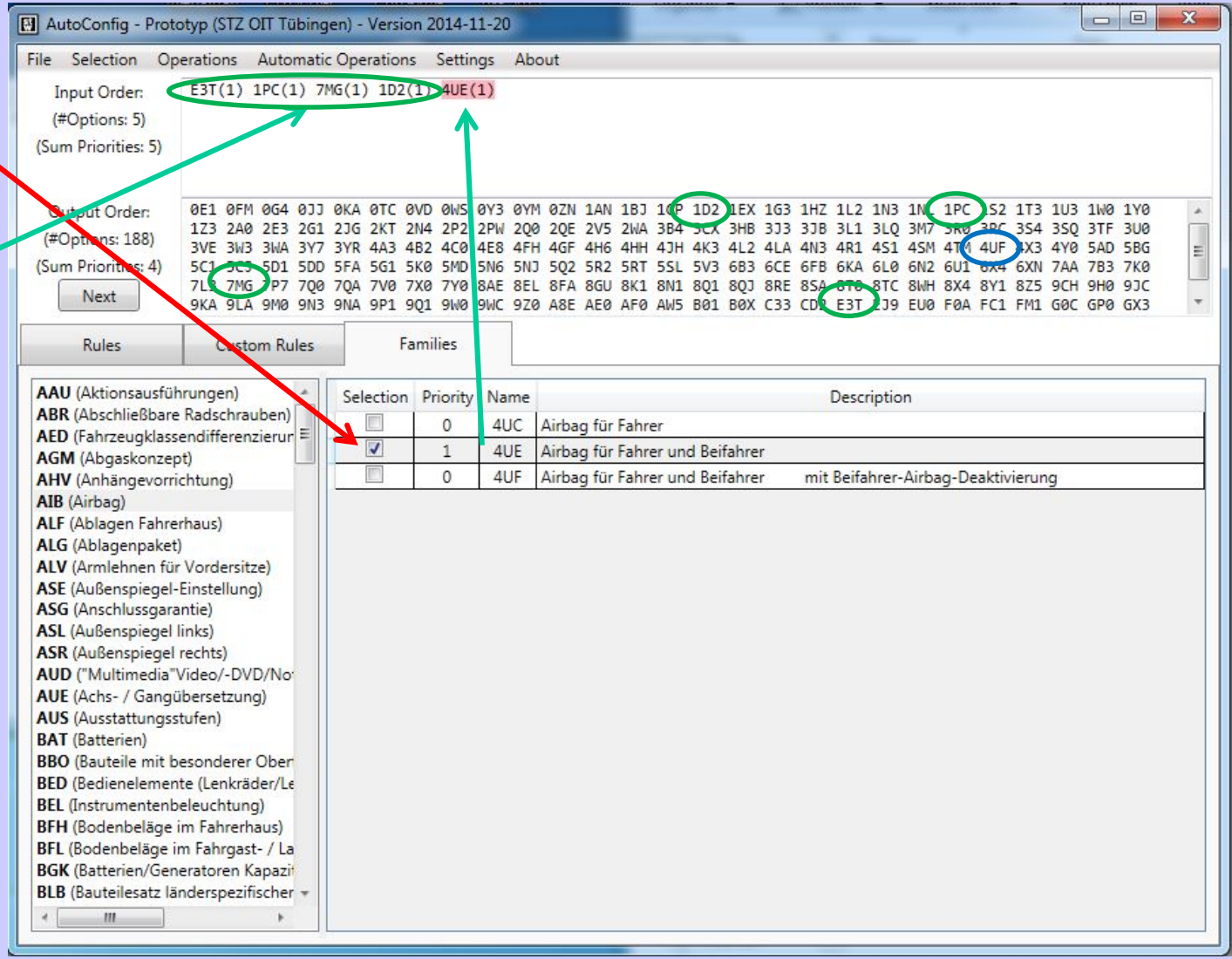
Example: Configuration with Auto.Config

- Select any option family
- Grey: options not available for this input order
- **Bold**: available options in family for this input order



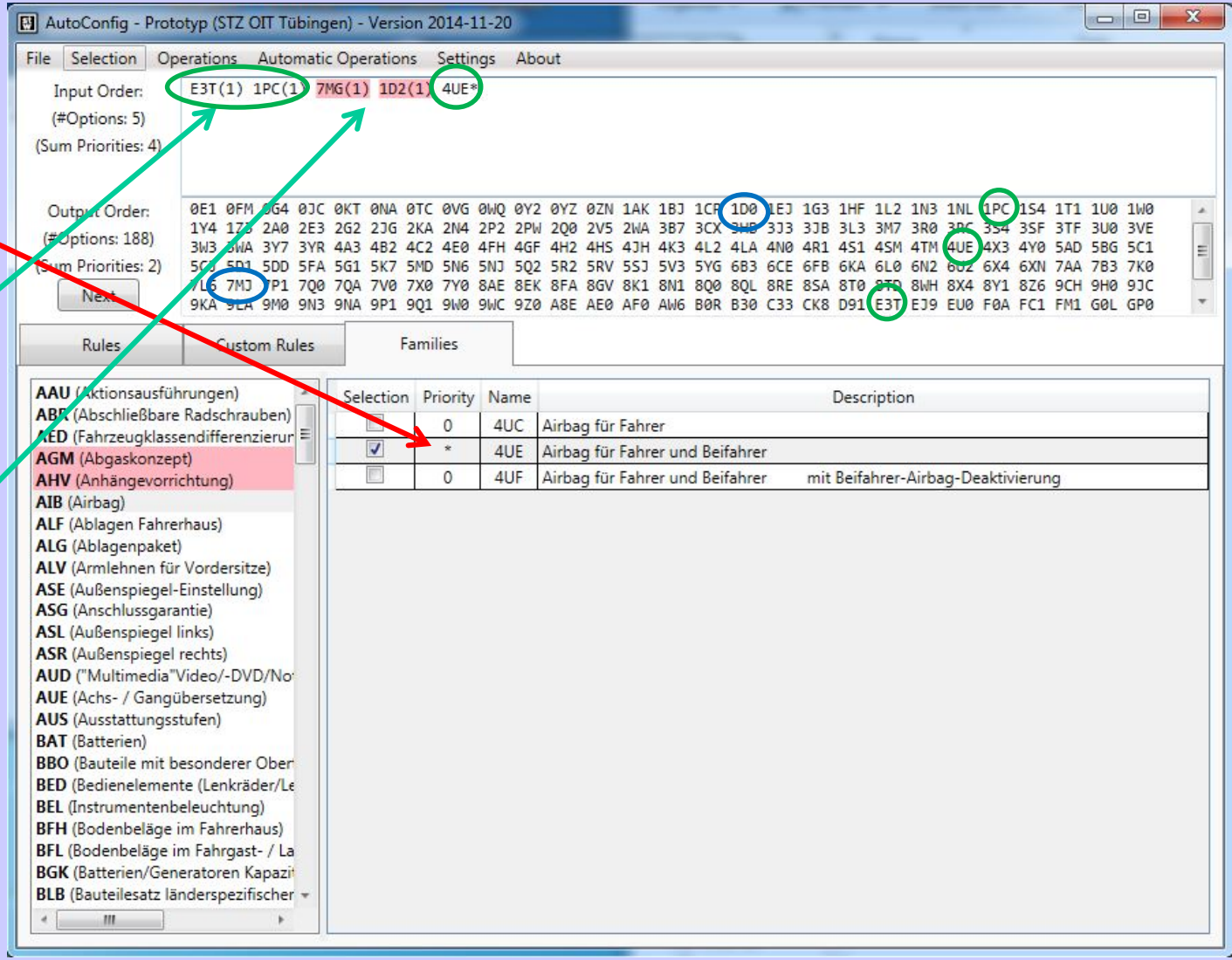
Optimization Ex.: Re-Configuration with Auto.Config

- Select an option **not** available
- *Auto.Config* computes **maximum** subset of PD-**legal** option selections
- *Auto.Config* computes **minimum** subset of **impossible** option selections
- *Auto.Config* computes new order containing the PD-legal options with **maximum sum of input weights** (here allways 1)



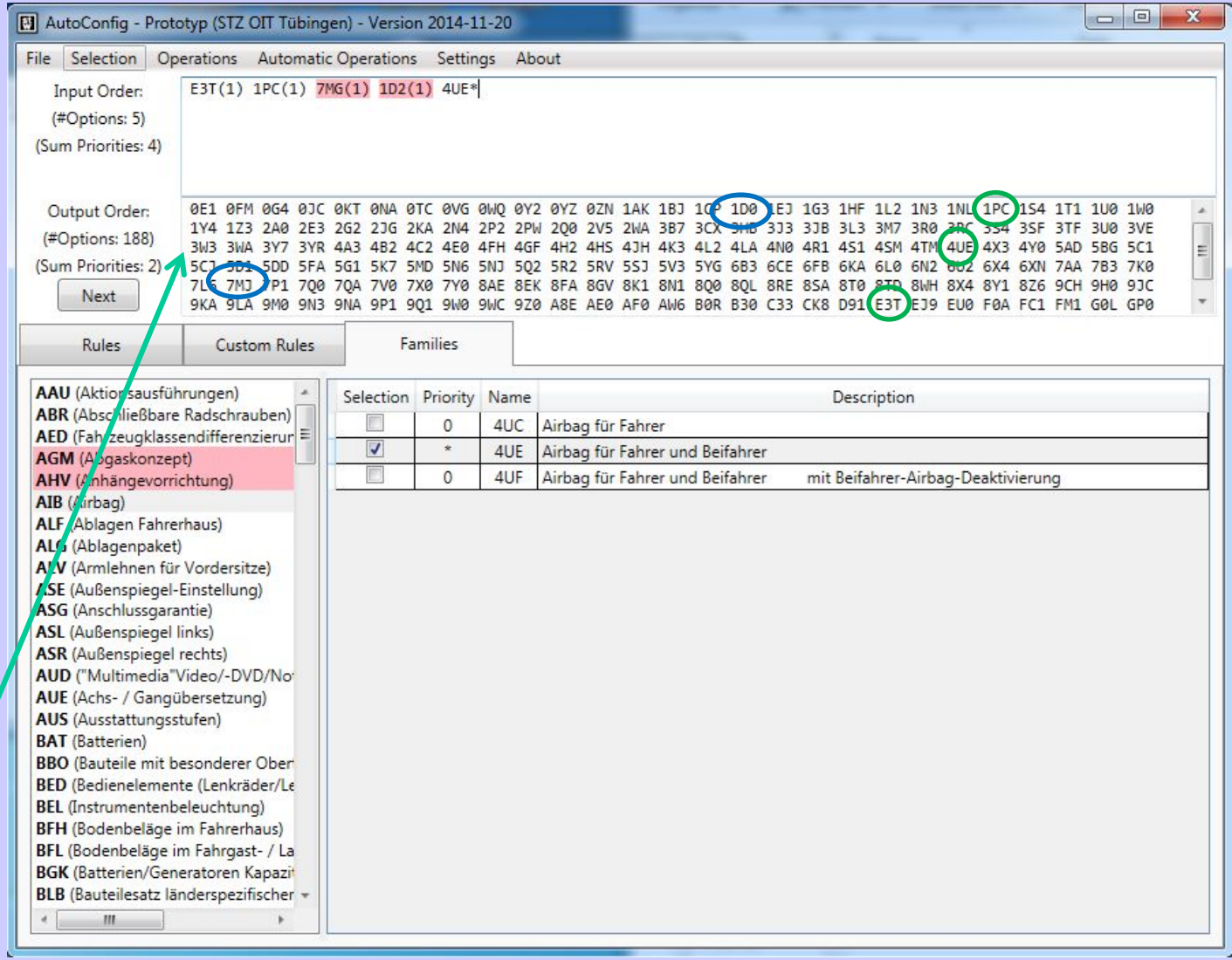
Optimization Ex.: Re-Configuration with Auto.Config

- **Re-Configuration**
- Fix the unavailable option 4UE(*)
- *Auto.Config* computes **maximum** subset of PD-**legal** option selections
- *Auto.Config* computes **minimum** subset of impossible option selections



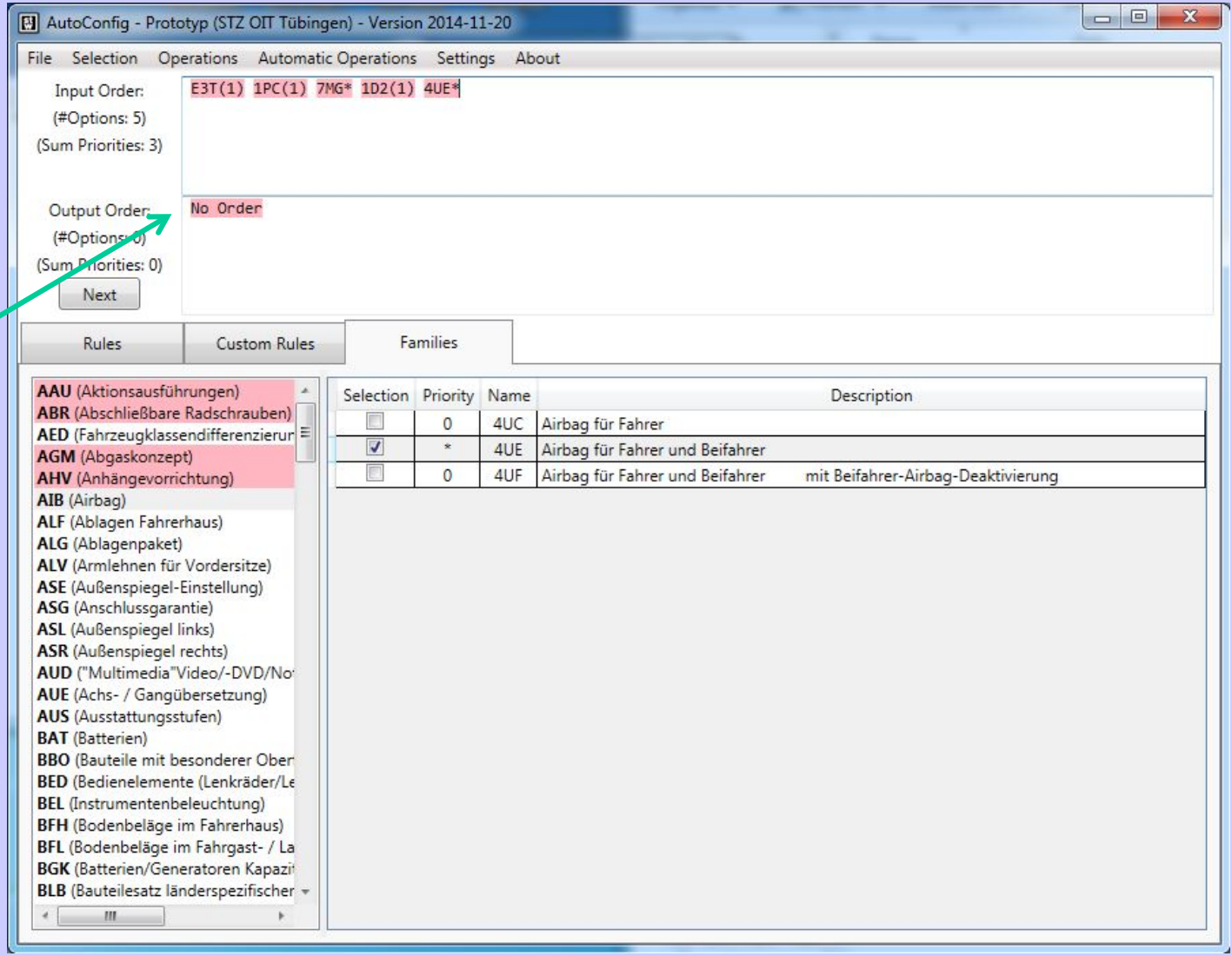
Optimization Ex.: Re-Configuration with Auto.Config

- **Re-Configuration**
- Fix the unavailable option 4UE(*)
- *Auto.Config* computes **maximum** subset of PD-**legal** option selections
- *Auto.Config* computes **minimum** subset of impossible option selections
- *Auto.Config* computes new order **maximizing the weight** of the PD-legal input options



Optimization Ex.: Re-Configuration with Auto.Config

- **Re-Configuration**
- Fix (*) the unavailable options 7MG, 4UE
- *Auto.Config* finds **contradiction:** no legal order with 7MG + 4UE



Other Problems

- **Compute all combinations of a set of materials**
  - given the materials in a subset of the BoM (e.g. driver´s seat, rear axle, …)
  - From each BoM position pick exactly one material variant
  - Compute all consistent combinations which can occur in configurable vehicles
  - Example: All seats or all axles which can occur in a model line
- **Compute software upgrades**
  - given software variants and their upgrade dependencies
  - compute possible upgrades / compute all cars with possible upgrades
- **Compute homologation (certification) requirements**
  - given a sales program for a market
  - compute all homologation relevant materials for that market
- **Change the documentation method**
  - convert the formulas for a car model line from one documentation method to another, preserving the set of legal configurations.

Literature

1. W. Küchlin, C. Sinz. Proving Consistency Assertions for Automotive Product Data Management. *J. Automated Reasoning* **24** (2000)
2. C. Sinz, A. Kaiser, W. Küchlin. Formal Methods for the Validation of Automotive Product Configuration Data. *AIEDAM J.* **17** (2003)
3. Ch. Zengler, W. Küchlin. Boolean Quantifier Elimination for Automotive Configuration. In: *Formal Methods for Industrial Critical Systems (FMICS)*, 2013.
4. A. Biere, M. Heule, H. van Maaren, T. Walsh (eds.). *Handbok of Satisfiability*. IOS Press 2009. (Comprehensive account of SAT based methods)
5. D. E. Knuth. Satisfiability. *The Art of Computer Programming* Vol **4** Fasc. 6. Addison Wesley, 2016