Boolean Quantifier Elimination for Automotive Configuration — A Case Study —

Christoph Zengler and Wolfgang Küchlin

Symbolic Computation Group
Informatics Department
Universität Tübingen, Germany
`www-sr.informatik.uni-tuebingen.de`

23-09-2013

Overview of the Talk

SAT and Related Problems

Definition (SAT Problem)

Is there an assignment β such that a given formula φ in Propositional Logic evaluates to `true`?

SAT and Related Problems

Definition (SAT Problem)

Is there an assignment β such that a given formula φ in Propositional Logic evaluates to `true`?

Definition (QBF/QSAT Problem)

Does a fully quantified Boolean formula φ evaluate to `true` or `false`?

SAT and Related Problems

Definition (SAT Problem)

Is there an assignment β such that a given formula φ in Propositional Logic evaluates to `true`?

Definition (QBF/QSAT Problem)

Does a fully quantified Boolean formula φ evaluate to `true` or `false`?

Definition (Open QBF/PQSAT Problem)

Compute a Boolean formula which establishes necessary and sufficient conditions on the free variables of a Boolean formula φ for the existence of a satisfying assignment.

SAT and Related Problems

Definition (SAT Problem)

Is there an assignment β such that a given formula φ in Propositional Logic evaluates to `true`?

Definition (QBF/QSAT Problem)

Does a fully quantified Boolean formula φ evaluate to `true` or `false`?

Definition (Open QBF/PQSAT Problem)

Compute a Boolean formula which establishes necessary and sufficient conditions on the free variables of a Boolean formula φ for the existence of a satisfying assignment.

Analogies to Quantifier Elimination

- SAT Solver: QE procedure for existential Boolean sentences
- QSAT Solver: QE procedure for Boolean sentences
- PQSAT Solver: QE procedure for arbitrary Boolean formulas

Examples

Example (SAT, QSAT and PQSAT)

$$\varphi = (x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w) \wedge (u \vee w)$$

- $\mathtt{sat}(\varphi) = \mathtt{qsat}(\exists (x, y, u, w)\ \varphi) = \mathtt{true}$

Examples

Example (SAT, QSAT and PQSAT)

$$\varphi = (x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w) \wedge (u \vee w)$$

- $\mathtt{sat}(\varphi) = \mathtt{qsat}(\exists (x, y, u, w)\ \varphi) = \mathtt{true}$
- $\mathtt{qsat}(\exists (x, y) \forall u \exists w\ \varphi) = \mathtt{true}$

Examples

Example (SAT, QSAT and PQSAT)

$$\varphi = (x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w) \wedge (u \vee w)$$

- $\mathtt{sat}(\varphi) = \mathtt{qsat}(\exists(x,y,u,w)\ \varphi) = \mathtt{true}$
- $\mathtt{qsat}(\exists(x,y)\forall u \exists w\ \varphi) = \mathtt{true}$
- $\mathtt{qsat}(\exists(x,y)\forall(u,w)\ \varphi) = \mathtt{false}$

Examples

Example (SAT, QSAT and PQSAT)

$$\varphi = (x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w) \wedge (u \vee w)$$

- $\mathtt{sat}(\varphi) = \mathtt{qsat}(\exists(x, y, u, w)\ \varphi) = \mathtt{true}$
- $\mathtt{qsat}(\exists(x, y)\forall u \exists w\ \varphi) = \mathtt{true}$
- $\mathtt{qsat}(\exists(x, y)\forall(u, w)\ \varphi) = \mathtt{false}$
- $\mathtt{pqsat}(\exists x \forall y\ \varphi) = (u \wedge w) \vee (\neg u \wedge w) \equiv w$

Examples

Example (SAT, QSAT and PQSAT)

$$\varphi = (x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w) \wedge (u \vee w)$$

- $\mathtt{sat}(\varphi) = \mathtt{qsat}(\exists(x,y,u,w)\ \varphi) = \mathtt{true}$
- $\mathtt{qsat}(\exists(x,y)\forall u \exists w\ \varphi) = \mathtt{true}$
- $\mathtt{qsat}(\exists(x,y)\forall(u,w)\ \varphi) = \mathtt{false}$
- $\mathtt{pqsat}(\exists x \forall y\ \varphi) = (u \wedge w) \vee (\neg u \wedge w) \equiv w$

In this talk...

... we only need PQSAT with a purely existential prefix

Existential Boolean Quantifier Elimination (EBQE)

Overview: Some Methods for EBQE

- **Substitute & Simplify**: intuitive, based on Shannon.
- **Model Enumeration and Projection**: intuitive, for small model counts.
- **Clause Distribution**: [Davis, Putnam] resolution on eliminated variable, creates (many) new clauses.
- **Knowledge Compilation and Projection**: OBDD [Briant] based, up to mid-size formulas.
- **Knowledge Compilation and Projection**: DNNF [Darwiche] based, relaxation of OBDDs, large(r) formulas.
- **Dependency Sequents**: improvement [Goldberg, Manolios] over Clause Distribution, creates fewer new constraints.

Overview: Some Methods for EBQE

- **Substitute & Simplify**: intuitive, based on Shannon.
- **Model Enumeration and Projection**: intuitive, for small model counts.
- **Clause Distribution**: [Davis, Putnam] resolution on eliminated variable, creates (many) new clauses.
- **Knowledge Compilation and Projection**: OBDD [Briant] based, up to mid-size formulas.
- **Knowledge Compilation and Projection**: DNNF [Darwiche] based, relaxation of OBDDs, large(r) formulas.
- **Dependency Sequents**: improvement [Goldberg, Manolios] over Clause Distribution, creates fewer new constraints.

	Method	**Input Format**	**Output**
(MEP)	model enum. & proj.	arbitrary (CNF)	DNF
(MEPI)	MEP & prime implicants	arbitrary (CNF)	CNF
(CD)	clause distribution	CNF	CNF
(SuSi)	substitute & simplify	arbitrary	arbitrary
(DNNF)	DNNF compilation & proj.	arbitrary (CNF)	DNNF
(DDS)	dependency sequents	CNF	CNF

Substitute & Simplify

- Goes back at least to Boole and Shannon

Elimination of a Single Quantifier

$$\exists x \varphi \equiv \varphi[\top/x] \vee \varphi[\bot/x] \qquad (1)$$

Algorithm: $\texttt{SuSi}(\varphi)$

Input: An existential Boolean formula φ
Output: ψ with $\psi \equiv \varphi$ and ψ quantifier-free

1. Eliminate quantifiers successively with (1)
2. simplify the formula after each elimination step

Example

Example (Substitute & Simplify)

$$\varphi = \exists x \exists y\ (x \wedge z) \wedge (\neg y \wedge (x \vee w))$$

Example

Example (Substitute & Simplify)

$$\varphi = \exists x \exists y \ (x \wedge z) \wedge (\neg y \wedge (x \vee w))$$

1. Substitute x:

$$\exists y \ [(\top \wedge z) \wedge (\neg y \wedge (\top \vee w))] \vee [(\bot \wedge z) \wedge (\neg y \wedge (\bot \vee w))]$$

Example

Example (Substitute & Simplify)

$$\varphi = \exists x \exists y \; (x \wedge z) \wedge (\neg y \wedge (x \vee w))$$

1. Substitute x:

$$\exists y \; [(\top \wedge z) \wedge (\neg y \wedge (\top \vee w))] \vee [(\bot \wedge z) \wedge (\neg y \wedge (\bot \vee w))]$$

2. Simplify

$$\exists y \; [z \wedge \neg y]$$

Example

Example (Substitute & Simplify)

$$\varphi = \exists x \exists y \ (x \wedge z) \wedge (\neg y \wedge (x \vee w))$$

1. Substitute x:

$$\exists y \ [(\top \wedge z) \wedge (\neg y \wedge (\top \vee w))] \vee [(\bot \wedge z) \wedge (\neg y \wedge (\bot \vee w))]$$

2. Simplify

$$\exists y \ [z \wedge \neg y]$$

3. Substitue y:

$$[z \wedge \bot] \vee [z \wedge \top]$$

Example

Example (Substitute & Simplify)

$$\varphi = \exists x \exists y \; (x \wedge z) \wedge (\neg y \wedge (x \vee w))$$

1. Substitute x:

$$\exists y \; [(\top \wedge z) \wedge (\neg y \wedge (\top \vee w))] \vee [(\bot \wedge z) \wedge (\neg y \wedge (\bot \vee w))]$$

2. Simplify

$$\exists y \; [z \wedge \neg y]$$

3. Substitue y:

$$[z \wedge \bot] \vee [z \wedge \top]$$

4. Simplify

$$z$$

Model Enumeration with Projection

Model Enumeration

- Enumerate and output all models of a given propositional formula
- Variants of DPLL can be used.
  Naive: Add a blocking clause for each solution found by the solver.

Combine Model Elimination and Projections

1. Enumerate all models
2. Project onto the free variables

Model Enumeration with Projection

Model Enumeration

- Enumerate and output all models of a given propositional formula
- Variants of DPLL can be used.
  Naive: Add a blocking clause for each solution found by the solver.

Combine Model Elimination and Projections

1. Enumerate all models
2. Project onto the free variables

Example (Model Enumeration with Projection)

$\varphi = \exists x \; \exists y \; (x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)$

Model Enumeration with Projection

Model Enumeration

- Enumerate and output all models of a given propositional formula
- Variants of DPLL can be used.
  Naive: Add a blocking clause for each solution found by the solver.

Combine Model Elimination and Projections

1. Enumerate all models
2. Project onto the free variables

Example (Model Enumeration with Projection)

$\varphi = \exists x \; \exists y \; (x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)$

1. Enumerate all models of $\mathrm{mat}(\varphi)$:

$$\{(\neg x, y, \neg w, z), (x, y, \neg w, z), (x, y, w, z)\}$$

Model Enumeration with Projection

Model Enumeration

- Enumerate and output all models of a given propositional formula
- Variants of DPLL can be used.
  Naive: Add a blocking clause for each solution found by the solver.

Combine Model Elimination and Projections

1. Enumerate all models
2. Project onto the free variables

Example (Model Enumeration with Projection)

$\varphi = \exists x\ \exists y\ (x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)$

1. Enumerate all models of $\mathrm{mat}(\varphi)$:

$$\{(\neg x, y, \neg w, z), (x, y, \neg w, z), (x, y, w, z)\}$$

2. Projection to free variables: $\{(\neg w, z), (w, z)\}$

Model Enumeration with Projection

Model Enumeration

- Enumerate and output all models of a given propositional formula
- Variants of DPLL can be used.
  Naive: Add a blocking clause for each solution found by the solver.

Combine Model Elimination and Projections

1. Enumerate all models
2. Project onto the free variables

Example (Model Enumeration with Projection)

$\varphi = \exists x\ \exists y\ (x \lor \neg w) \land (\neg x \lor z) \land (\neg z \lor y) \land (w \lor z)$

1. Enumerate all models of mat(φ):

$$\{(\neg x, y, \neg w, z), (x, y, \neg w, z), (x, y, w, z)\}$$

2. Projection to free variables: $\{(\neg w, z), (w, z)\}$
3. Quantifier-free equivalent to φ: $(\neg w \land z) \lor (w \land z) \equiv z$

(High Variance) Automotive Product Configuration

Used e.g. by car manufacturers (OEMs) Daimler, BMW, AUDI/VW, OPEL/GM

- Goal: personalized products at mass production price
- Only every 30,000th Mercedes C-Class car is identical
- Every Mercedes truck is built identically only 1.5 times a year (of 120,000 trucks, 70,000 are different)

(High Variance) Automotive Product Configuration

Used e.g. by car manufacturers (OEMs) Daimler, BMW, AUDI/VW, OPEL/GM

- Goal: personalized products at mass production price
- Only every 30,000th Mercedes C-Class car is identical
- Every Mercedes truck is built identically only 1.5 times a year (of 120,000 trucks, 70,000 are different)

The order process

1. Every equipment option is represented by a *true/false* code
2. A customer's car order is a list of *true* option codes
3. Order completion: Extend the customer's order by additional codes
   - implied by packages, country of delivery, . . .
   - used for production control (factory codes)
4. Complete order (configuration): all order codes are *true*, all other codes are *false*
5. . . . but not all *true/false* combinations (configurations) yield constructible orders (*valid* configurations)

Automotive Product Documentation

Product Documentation = Product Description + Product Structure.

- **Product description (high-level configuration):**
  Which configurations are *valid*?
- **Product structure (low-level configuration):**
  Which parts (materials) belong to a valid configuration?

Automotive Product Documentation

Product Documentation = Product Description + Product Structure.

- **Product description (high-level configuration):** Which configurations are *valid*?
- **Product structure (low-level configuration):** Which parts (materials) belong to a valid configuration?

Product Description: The formula PDF

- Set of (boolean) constraints between equipment codes
  - Order with equipment code c requires (boolean) *constructibility condition* C: $c \rightarrow C$
  - Order satisfying (boolean) *supplementing condition* A implies equipment code c: $A \rightarrow c$
  - Group (family) conditions: exactly one radio
  - . . . plus OEM specific tricks . . .
- PDF [Küchlin,Sinz 2000]: Compile all constraints into a single *Product Description Formula* PDF
- All valid orders are represented as the solutions of the PDF
  - Implicit representation: PDF approx. 500KB, solution count is from 10^{20} to 10^{100}.

Example of a product description formula - 1

Example (Mercedes)

- about 600 order options (+ 350 country codes)
- about 3,300 rules (constructibility, additional codes)
- parts list with about 35,000 parts

Example of a product description formula - 1

Example (Mercedes)

- about 600 order options (+ 350 country codes)
- about 3,300 rules (constructibility, additional codes)
- parts list with about 35,000 parts

Example (Some rules...)

Equipment codes: 675 = sport suspension, Mxxx = different motors, 769 = AMG tires, 956 sport package
C(x) = Constructibility condition for x, i.e. $x \rightarrow C(x)$
A(x) = Supplementing condition for x, i.e. $A(x) \rightarrow x$

- $C(675) = M111 \vee M112 \vee M605 \vee M611$
- $A(675) = 769 \vee 956$

The Product Documentation ctd.

Product structure: The parts list BOM (*Bill-of-Materials*)

- List of all parts ever needed for an entire car *line* (all AUDI A4, ...)
- Structured by geometric mounting positions (e.g. front door left)
- Each position lists all *variants* of a part (e.g. different radios)
- Each part T has a selection condition $S(\mathrm{T})$, and T is selected if $\mathrm{O} \models S(\mathrm{T})$

The Product Documentation ctd.

Product structure: The parts list BOM (*Bill-of-Materials*)

- List of all parts ever needed for an entire car *line* (all AUDI A4, ...)
- Structured by geometric mounting positions (e.g. front door left)
- Each position lists all *variants* of a part (e.g. different radios)
- Each part T has a selection condition $S(T)$, and T is selected if $O \models S(T)$

Example

Pos	Var	Part no	Description	Code Constraint
2549	10	A211...	automatic interior light	;
2555	10	N072...	bulb IEC60809-C5W-12V	;
1620	10	N000...	screw N 14112-M 6x12	Z04;
2000	10	A000...	fastener plug door	Z04;
150	10	A211...	mirror case with glass	-249+-623+-835+-494;
150	20	A212...	mirror case with dimmed glass	249+-623+-835+-494;

Analyzing Automotive Product Documentation

Traditional order dependent questions

- **Constructibility check**: Is order O valid, i.e. $\mathrm{O} \models \mathrm{PDF}$?
- **Parts list generation**: Compute the parts for a valid order. Part T is needed if $\mathrm{O} \models \mathrm{S}(\mathrm{T})$.

Analyzing Automotive Product Documentation

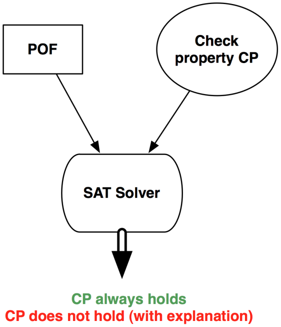
Traditional order dependent questions

- **Constructibility check**: Is order O valid, i.e. $\mathrm{O} \models \mathrm{PDF}$?
- **Parts list generation**: Compute the parts for a valid order. Part T is needed if $\mathrm{O} \models S(\mathrm{T})$.

New Order Independent Analysis of Product Description

- **Model counting** [Kübler,Zengler,Küchlin 2010] (non-CNF model counting): Count the solutions of the PDF (as a conjunction of arbitrary boolean formulas).
- **Dead codes in PDF** [Küchlin,Sinz 2000]: Can every option code c appear in at least one valid order, i.e. SAT($c \wedge \mathrm{PDF}$)?
- **Forced codes in PDF** [Küchlin,Sinz 2000]: Can every option code c *not* appear in at least one valid order, i.e. SAT($\neg c \wedge \mathrm{PDF}$)?
- **Car configurator**: Can partial order O be extended by code c? i.e. SAT($\mathrm{O} \wedge c \wedge \mathrm{PDF}$)?

Verification of the PDF



Examples of checkable properties:

- Customer order (navigation system, radio, color,...) constructible?
- Code combination possible? (radio + navigation system)?
- Code present in any possible configuration?
- Code constructible in a specific country?
- ...

Analyzing Automotive Product Documentation ctd.

New Abstract Analysis of the BOM

Abstract = independent of concrete orders (Before Start of Production)

- **Dead Parts in BOM** [Küchlin,Sinz 2000]: Can every part T appear in at least one car? SAT($S(\mathrm{T}) \wedge \mathrm{PDF}$)?
- **Missing parts in BOM** [Küchlin,Sinz 2000]: Does every car have a part in all positions? SAT(...).
- **Double hits in BOM** [Küchlin,Sinz 2000]: Ist there an order for which two alternative parts are selected simultaneously? SAT($S(\mathrm{T}_1) \wedge S(\mathrm{T}_2) \wedge \mathrm{PDF}$)?

Analyzing Automotive Product Documentation ctd.

New Abstract Analysis of the BOM

Abstract = independent of concrete orders (Before Start of Production)

- **Dead Parts in BOM** [Küchlin,Sinz 2000]: Can every part T appear in at least one car? $\mathrm{SAT}(S(\mathrm{T}) \wedge \mathrm{PDF})$?
- **Missing parts in BOM** [Küchlin,Sinz 2000]: Does every car have a part in all positions? SAT(...).
- **Double hits in BOM** [Küchlin,Sinz 2000]: Ist there an order for which two alternative parts are selected simultaneously? $\mathrm{SAT}(S(\mathrm{T}_1) \wedge S(\mathrm{T}_2) \wedge \mathrm{PDF})$?

Example (Superfluous part)

41	40	A211...	...	Z04 + 573;

- ... but according to the PDF, Z04 and 573 cannot occur together in a valid configuration

Advanced Abstract Analysis (Order Independent)

Focus on Subsets [Küchlin, Sinz 2000]

- Repeat all analyses for subsets of orders given by specific code values (Markets, Motor+Steering+DriveTrain combinations, factories, . . .)
- this is far more interesting in real life . . .

Advanced Abstract Analysis (Order Independent)

Focus on Subsets [Küchlin, Sinz 2000]

- Repeat all analyses for subsets of orders given by specific code values (Markets, Motor+Steering+DriveTrain combinations, factories, . . .)
- this is far more interesting in real life . . .

New: Abstract from (any values of) Manufacturer Codes

- Model counting *modulo* factory settings
  - If O is a customer order and F_1 and F_2 indicate factories, then orders $O \cup F_1$ and $O \cup F_2$ count as two different cars. But the cars have identical equipment options O.
- Projection of BOM constraints
  - Parts selection formulas often contain both customer selectable codes and manufacturer control codes.
  - We *project* the formulas to the customer codes.
  - The projected $S'(T)$ tells us: If $S'(T) = true$ for a customer order O, then it is possible to set the manufacturer control codes such that $S(T) = true$ and the part will be selected.

Projection of BOM constraints

Example

- We have customer options $\{o_1, o_2, o_3, o_4\}$ and manufacturer options for motors M_1, M_2, M_3 and drivetrains A, F, R.
- We consider the part selection formula:

$$S = [(o_1 \wedge M_1 \wedge (\neg o_2 \vee o_3)) \vee ((M_2 \vee M_3) \wedge \neg o_1 \wedge o_2)] \wedge o_4 \wedge \neg F \wedge \neg R$$

- Even for this very short constraint it is not immediately obvious for which combinations of customer options this material can possibly be selected.
- We project the formula by existential quantification $S' = \exists(M_1, M_2, M_3, A, F, R)\ S$, and obtain

$$S' = o_4 \wedge ((o_1 \wedge \neg o_2) \vee (o_1 \wedge o_3) \vee (\neg o_1 \wedge o_2))$$

- S' is a necessary and sufficient condition on the customer options for the existence of vehicles which use this part.

Application: BMW product types

- BMW product types are subsets of car series, e.g. 320i Sedan, left steering, rear wheel drive is a subset of the BMW 3-series.
- $|\mathcal{O}|$ states the number of HLC options for the respective product type
- $|\mathcal{O}_C|$ states the number of control options
- $|\mathcal{R}|$ states the number of HLC rules
- #orig represents the model count for the original PDF
- #proj represents the model count for the PDF where the options of $\mathcal{O}_C$ were eliminated
- We show the computation times of the methods Clause Distribution (DD), DNNF compilation (DNNF), Substitute & Simplify (SuSi), and Dependeny Sequents (DDS) for eliminating all options of $\mathcal{O}_C$ from the respective PDF. MEP and MEPI did not finish due to our high model count.
- We also show the sizes of the resulting CNF formulas for CD and DDS, which introduce new clauses.

Benchmark results (excerpt)

Instance						*QE time in s*			**DDS**	
type	$\|\mathcal{O}\|$	$\|\mathcal{O}_C\|$	$\|\mathcal{R}\|$	**#orig**	**#proj**	**CD**	**DNNF**	**SuSi**	**stand.**	**opt**
S2T01	423	256	211	$9.95 \cdot 10^{48}$	$2.95 \cdot 10^{16}$	0.07	0.14	0.92	**0.01**	1.08
S2T02	403	237	190	$1.20 \cdot 10^{48}$	$1.18 \cdot 10^{17}$	0.07	0.12	0.60	**0.01**	1.11
S2T03	425	258	208	$9.64 \cdot 10^{48}$	$1.64 \cdot 10^{16}$	0.07	0.13	1.00	**0.02**	1.01
S2T04	408	242	215	$2.52 \cdot 10^{47}$	$1.81 \cdot 10^{16}$	0.06	0.12	0.81	**0.01**	1.07
S2T05	223	85	122	$1.59 \cdot 10^{33}$	$3.84 \cdot 10^{13}$	**0.01**	0.03	0.12	**0.01**	0.48
S2T06	441	272	220	$4.00 \cdot 10^{53}$	$6.35 \cdot 10^{16}$	0.07	0.14	1.14	**0.01**	1.12
S2T07	424	256	229	$5.43 \cdot 10^{52}$	$2.57 \cdot 10^{17}$	0.07	0.12	0.74	**0.02**	1.15
S2T08	220	83	122	$7.78 \cdot 10^{32}$	$1.99 \cdot 10^{13}$	**0.01**	0.02	0.10	**0.01**	0.50
S2T09	433	264	224	$5.75 \cdot 10^{49}$	$3.23 \cdot 10^{16}$	0.07	0.13	1.11	**0.02**	1.13
S2T10	417	249	236	$1.01 \cdot 10^{49}$	$6.54 \cdot 10^{16}$	0.07	0.13	0.98	**0.02**	1.12

Benchmark results (excerpt)

Instance						*QE time in s*				
									DDS	
type	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{O}_C\rvert$	$\lvert\mathcal{R}\rvert$	**#orig**	**#proj**	**CD**	**DNNF**	**SuSi**	**stand.**	**opt**
S2T01	423	256	211	$9.95 \cdot 10^{48}$	$2.95 \cdot 10^{16}$	0.07	0.14	0.92	**0.01**	1.08
S2T02	403	237	190	$1.20 \cdot 10^{48}$	$1.18 \cdot 10^{17}$	0.07	0.12	0.60	**0.01**	1.11
S2T03	425	258	208	$9.64 \cdot 10^{48}$	$1.64 \cdot 10^{16}$	0.07	0.13	1.00	**0.02**	1.01
S2T04	408	242	215	$2.52 \cdot 10^{47}$	$1.81 \cdot 10^{16}$	0.06	0.12	0.81	**0.01**	1.07
S2T05	223	85	122	$1.59 \cdot 10^{33}$	$3.84 \cdot 10^{13}$	**0.01**	0.03	0.12	**0.01**	0.48
S2T06	441	272	220	$4.00 \cdot 10^{53}$	$6.35 \cdot 10^{16}$	0.07	0.14	1.14	**0.01**	1.12
S2T07	424	256	229	$5.43 \cdot 10^{52}$	$2.57 \cdot 10^{17}$	0.07	0.12	0.74	**0.02**	1.15
S2T08	220	83	122	$7.78 \cdot 10^{32}$	$1.99 \cdot 10^{13}$	**0.01**	0.02	0.10	**0.01**	0.50
S2T09	433	264	224	$5.75 \cdot 10^{49}$	$3.23 \cdot 10^{16}$	0.07	0.13	1.11	**0.02**	1.13
S2T10	417	249	236	$1.01 \cdot 10^{49}$	$6.54 \cdot 10^{16}$	0.07	0.13	0.98	**0.02**	1.12

	CD		**DDS**		**DDS (opt)**	
type	#vars	#clauses	#vars	#clauses	#vars	#clauses
S2T01	190	1338	221	1978	221	**828**
S2T02	191	1316	226	1964	226	**838**
S2T03	189	1297	218	1849	218	**771**
S2T04	190	1303	221	1844	221	**798**
S2T05	151	812	205	1193	205	**633**
S2T06	192	1345	224	1980	224	**834**
S2T07	193	1342	225	1978	225	**838**
S2T08	150	807	204	1192	204	**629**
S2T09	191	1356	222	1968	222	**863**
S2T10	191	1349	223	1943	223	**828**

Conclusion for these applications

Counting Projected Models

Three suitable approaches: **Clause distribution, substitute & simplify**, and **dependency sequents**. All three require an additional model counter. Here DNNF compilation and model counting proved to be a stable solution. We used `c2d` which was able to count the models in < 1 sec for each instance. Model enumeration with projection is not suitable because the model counts of our application instances are too large. DNNF computation with projection is not suitable because after variable elimination the output is no longer necessarily a deterministic DNNF and therefore model counting cannot be performed in linear time.

Conclusion for these applications

Counting Projected Models

Three suitable approaches: **Clause distribution, substitute & simplify**, and **dependency sequents**. All three require an additional model counter. Here DNNF compilation and model counting proved to be a stable solution. We used `c2d` which was able to count the models in < 1 sec for each instance. Model enumeration with projection is not suitable because the model counts of our application instances are too large. DNNF computation with projection is not suitable because after variable elimination the output is no longer necessarily a deterministic DNNF and therefore model counting cannot be performed in linear time.

Projection of BOM parts selection formulae

Two suitable approaches: **model enumeration with projection** and **substitute & simplify**. Model enumeration with projection is especially well-suited because of its output format DNF. Substitute & simplify yields a formula with a high resemblance to the original input formula. Clause distribution, DNNF computation and projection, and DDS are not very well-suited because their output formats are too distinct from the input formula. Therefore humans would have problems relating the projected constraints to the original ones.