Selected research overview

SAT/SMT

Automated Reasoning GrOup – ARGO

Department of Computer Science
Faculty of Mathematics
University of Belgrade, Serbia

Outline

Formalization and Verification of SAT/SMT Solvers

Implementation of SAT/SMT solvers

Applications of SAT/SMT Solvers
- Software verification
- Scheduling and timetabling
- SAT Encodings
- SAT / SMT solvers in Proof Assistants
- Combinatorial enumeration
- Coherent Logic and Larus

Formalization and Verification of SAT/SMT Solvers

Formal verification of SAT solvers

Old result

- Early ARGO work in the SAT/SMT area
- Formalized modern SAT solvers as abstract state transition systems in Isabelle/HOL
- Covered the classical DPLL procedure and later solver features used in modern SAT engines

Main contribution

- Proved soundness, termination, and completeness for the formalized systems
- Provided machine-verifiable proofs that can serve as a building block for verified SAT solvers

References

- Filip Marić. *Formalization and Implementation of Modern SAT Solvers. Journal of Automated Reasoning*, 2009.
- Filip Marić, Predrag Janičić. *Formalization of Abstract State Transition Systems for SAT. Logical Methods in Computer Science*, 2011.

Formal verification of incremental simplex

Old result

- Early work on verifying the incremental simplex algorithm described by Duterte and de Moura.
- Formalized the algorithm by stepwise refinement
- Focused on proving correctness of the incremental linear arithmetic core used in SMT solving

Main contribution

- Provided a machine-checked account of the algorithm's correctness arguments
- Helped establish the trust base for verified arithmetic reasoning inside SMT solvers

Reference

- Mirko Spasić, Filip Marić. *Formalization of Incremental Simplex Algorithm by Stepwise Refinement*. *FM 2012*, LNCS 7436, Springer, 2012.

Incremental simplex, extended

AFP extension

- Later Archive of Formal Proofs entry: *An Incremental Simplex Algorithm with Unsatisfiable Core Generation*
- Extends the verified simplex core with unsatisfiable core generation, incremental assertion, and backtracking

Applications

- Strengthens the verified arithmetic layer used in SMT solvers
- Enables later formal developments that rely on linear programming reasoning and Farkas-style arguments

Reference

- Filip Marić, Mirko Spasić, René Thiemann. *An Incremental Simplex Algorithm with Unsatisfiable Core Generation*. Archive of Formal Proofs, 2018.

Downstream reuse

Verified linear programming

- The simplex formalization is reused in *Farkas' Lemma and Motzkin's Transposition Theorem*
- *Linear Programming* later builds on that verified LP infrastructure

Termination certification context

- Certified termination tools such as CeTA and CPF rely on trusted proof certificates
- Linear ranking functions make LP-based reasoning relevant for termination certification

References

- Ralph Bottesch, Max W. Haslbeck, René Thiemann. *Farkas' Lemma and Motzkin's Transposition Theorem*. Archive of Formal Proofs, 2019.
- Christian Sternagel, René Thiemann, Sarah Winkler, Harald Zankl. *CeTA - A Tool for Certified Termination Analysis*. 2012.

Algorithm certification

Graph isomorphism problem

- Extending McKay's graph canonical labeling algorithm to produce certificates
  - developed a dedicated derivation system that formalizes the operation of the algorithm
  - certificates of the obtained labelings are exported as proofs in that derivation system
  - a simple proof-checker verifies the correctness of the produced certificates
- Such certificates are useful in proving graph (non)-isomorphism, and they help with efficient enumeration of non-isomorphic graphs

Publications

- Milan Banković, Ivan Drecun, Filip Marić. *A proof system for graph (non)-isomorphism verification*. *Logical Methods in Computer Science*, 2023.

Implementation of SAT/SMT solvers

Early SAT/SMT solver infrastructure

ArgoLib and ArgoSAT

- ArgoLib – pre-SMT implementation of decision procedures,
- ArgoSAT – a modular implementation of a SAT solver, that closely follows Isabelle/HOL formalization

References

- Filip Marić, Predrag Janičić. *argo-lib: A Generic Platform for Decision Procedures. IJCAR 2004*, LNCS 3097, Springer, 2004.
- Filip Marić. *Formalization, Implementation and Verification of SAT solvers*. PhD Thesis, University of Belgrade, 2009.

ArGO SmArT — Simple algorithm portfolio for SAT

Application

- Developed an open-source SAT solver portfolio that is intentionally simple
- Uses k-nearest neighbors on a training set to choose the solver for a new instance
- The selection relies only on a few similar instances, rather than a complex meta-model

Results

- Shows how solver performance can be improved by solver selection around a SAT backend
- Demonstrates that a lightweight portfolio can compete with more elaborate systems
- Connects practical solver engineering with instance-based learning

Reference

- Mladen Nikolić, Filip Marić, Predrag Janičić. *Simple algorithm portfolio for SAT*. *Artificial Intelligence Review*, 40:457–465, 2013. DOI: 10.1007/s10462-011-9290-2.

Implementation of SMT solvers

ArgoSMT

- CDCL($\mathcal{T}$)-based open-source SMT solver developed as an experimental platform
- Supported theories: EUF, LRA, and LIA
- Used to test solver techniques and theory-combination ideas in practice

Parallelization

- Implemented simplex parallelization inside the LRA theory solver
- Explored how arithmetic reasoning can benefit from multi-core execution

Reference

- Milan Banković. *Parallelizing simplex within SMT solvers*. *Artificial Intelligence Review*, 2016.

Integrating SMT with finite-domain constraint solving

Results

- Support for reasoning with finite-domain constraints within the CDCL($\mathcal{T}$) framework
  - a separate *CSP theory* assuming finite integer intervals as variable domains
- Implementation of a propagator for the `alldifferent` constraint, based on Régin's filtering algorithm — a well known algorithm in the CSP community
  - Producing smart explanations for `alldifferent`-induced propagations
- Support for finite-domain linear integer constraints
- Smart cooperation between the `alldifferent` and linear constraints

Publications

- Milan Banković. *Extending SMT solvers with support for finite domain alldifferent constraint*. *Constraints*, 2015.
- Milan Banković. *Solving finite-domain linear constraints in presence of the alldifferent*. *Logical Methods in Computer Science*, 2016.

Applications of SAT/SMT Solvers

Applications of SAT/SMT Solvers

Software verification

Formalization and Verification of SAT/SMT Solvers
Implementation of SAT/SMT solvers
Applications of SAT/SMT Solvers
- Software verification
- Scheduling and timetabling
- SAT Encodings
- SAT / SMT solvers in Proof Assistants
- Combinatorial enumeration
- Coherent Logic and Larus

Using Simplex method in verifying software safety

Application

- Early work on checking software safety
- Targets automated detection of buffer overflows in C programs
- Generates software correctness conditions and discharges them with a Simplex-based theorem prover

Approach

- Uses decision procedures for linear arithmetic inside verification
- Shows how SMT-style reasoning can find concrete safety violations
- Demonstrates an early bridge between solver technology and software verification

Reference

- Milena Vujošević-Janičić, Filip Marić, Dušan Tošić. *Using Simplex Method in Verifying Software Safety*. *Yugoslav Journal of Operations Research*, 19(1):133–148, 2009. DOI: 10.2298/YUJOR0901133V.

Regression verification for student programs

Application

- Work on regression verification for automated evaluation of student programs
- Combines verification with control-flow graph similarity and automated testing
- Goal: detect whether a revised student submission still preserves the intended behavior

Approach

- Verification conditions are discharged by automated reasoning
- Similarity and testing reduce the search space before solver-based checking
- The approach shows how SAT/SMT methods can be used beyond pure theorem-proving benchmarks

Reference

- Milena Vujošević Janičić, Filip Marić. *Regression verification for automated evaluation of students programs*. *Computer Science and Information Systems*, 17(1):205–227, 2020.

Applications of SAT/SMT Solvers

Scheduling and timetabling

Formalization and Verification of SAT/SMT Solvers
Implementation of SAT/SMT solvers
Applications of SAT/SMT Solvers
- Software verification
- Scheduling and timetabling
- SAT Encodings
- SAT / SMT solvers in Proof Assistants
- Combinatorial enumeration
- Coherent Logic and Larus

Timetabling based on SAT encoding

Application

- Course timetabling encoded as propositional satisfiability
- Lessons, rooms, teachers, and student groups are represented by Boolean variables and clauses
- The encoding supports many hard and soft requirements, including forbidden hours, idle periods, consecutive days, and room allocation

Approach

- Timetables are finite combinatorial objects, so SAT gives complete search over a large constraint space
- The encoding is flexible enough to capture institution-specific rules without redesigning the solver
- The approach was tested on one high school and two university departments, with encouraging results

Reference

- Filip Marić. *Timetabling Based on SAT Encoding: a Case Study*. Argo. 2008.

Applications of SAT/SMT Solvers

SAT Encodings

Formalization and Verification of SAT/SMT Solvers
Implementation of SAT/SMT solvers
Applications of SAT/SMT Solvers
- Software verification
- Scheduling and timetabling
- SAT Encodings
- SAT / SMT solvers in Proof Assistants
- Combinatorial enumeration
- Coherent Logic and Larus

URSA: a system for uniform reduction to SAT

System

- Uniform way to solve a wide range of finite-domain problems by reducing them to SAT
- Introduces the URSA specification language, combining imperative and declarative programming
- Uses symbolic execution to turn problem specifications into SAT instances

Approach

- The SAT translation is defined precisely by the language semantics
- Multiple NP-complete and constraint problems can be modeled in a single framework
- The approach makes encoding reusable, systematic, and easier to verify

Reference

- Predrag Janičić. *URSA: A System for Uniform Reduction to SAT*. *Logical Methods in Computer Science*, 8(3):1–39, 2012.

URSA example: pseudorandom generator

Example 3.2 from the paper

- The specification checks whether the 100th value of a linear congruential generator matches a given target
- The same imperative code can be used as a precise SAT-oriented problem specification

Specification

```
nx=nseed;
for(ni=1;ni<=100;ni++)
  nx=nx*1664525+1013904223;
assert(nx==3998113695);
```

Why it matters

- Shows how URSA turns imperative programs into SAT instances by symbolic execution
- The approach avoids hand-building encodings for each intermediate state in the sequence

URBiVA: uniform reduction to bit-vector arithmetic

System

- System inspired by UrSA to reduce to SMT (bit-vector arithmetic, linear arithmetic, bit-blasting to SAT)
- Uses a C-like high-level specification language with arrays, loops, and user-defined functions
- Symbolic execution turns specifications into quantifier-free SMT formulas

Approach

- Provides a uniform interface to multiple SMT solvers
- Supports bit blasting to SAT as one backend option
- Shows how solver choice can be separated from problem specification

Reference

- Filip Marić, Predrag Janičić. *URBiVA: Uniform Reduction to Bit-Vector Arithmetic. IJCAR 2010.*

meSAT: multiple encodings of CSP to SAT

Application

- Work by Mirko Stojadinović and Filip Marić on translating finite linear CSP instances to SAT
- Supports several encodings and combinations of encodings instead of relying on a single translation scheme
- Uses syntactic features of the CSP instance to select a suitable encoding automatically

Approach

- Shows SAT as a backend for solving richer constraint problems
- Highlights how encoding choice strongly affects solver performance
- Connects solver technology with constraint solving practice

Reference

- Mirko Stojadinović, Filip Marić. *meSAT: multiple encodings of CSP to SAT*. *Constraints*, 19:380–403, 2014. DOI: 10.1007/s10601-014-9165-7.

Verifying translations between different representations

Results

- Verifying FlatZinc-to-SAT translation
  - Developed a tool that translates a FlatZinc model to SAT and exports proof obligations
  - Proving the proof obligations verifies the correctness of the obtained SAT translation
  - Proof obligations are expressed in the SMT-LIB language and proved by an SMT solver

Publications (in progress)

- Matija Lojović, Milan Banković. *A Certificate-Producing FlatZinc-to-SAT Reduction*. Under review, 2026.

Applications of SAT/SMT Solvers

SAT / SMT solvers in Proof Assistants

Formalization and Verification of SAT/SMT Solvers
Implementation of SAT/SMT solvers
Applications of SAT/SMT Solvers
- Software verification
- Scheduling and timetabling
- SAT Encodings
- SAT / SMT solvers in Proof Assistants
- Combinatorial enumeration
- Coherent Logic and Larus

Erdős-Szekeres conjecture

Combinatorial geometry result

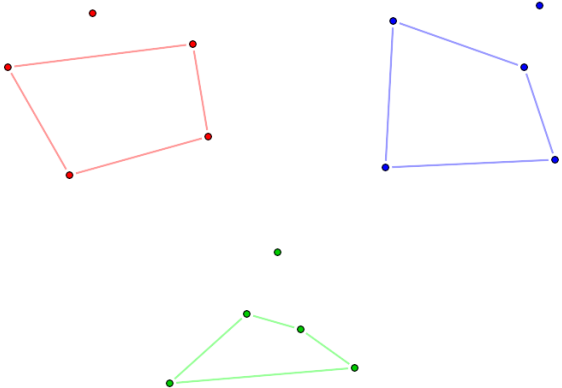
- Erdős-Szekeres "Happy Ending" conjecture for convex polygons
- The smallest number of points for which any general position arrangement contains a convex subset of n points is $2^{n-2} + 1$.
- Verified the case of convex polygons with at most 6 points
- Combined modern SAT solving with Isabelle/HOL formalization to make the proof reliable

Approach

- The proof relies on a finite combinatorial search space
- Symmetry breaking and SAT solving make the enumeration feasible
- The work is a good example of SAT as an engine for mathematical proof search

Reference

- Filip Marić. *Fast Formal Proof of the Erdős-Szekeres Conjecture for Convex Polygons with at Most 6 Points. Journal of Automated Reasoning*, 2019. DOI: 10.1007/s10817-017-9423-7.



KRK chess endgame in Isabelle/HOL

Application

- Formalized a King and Rook vs King chess endgame strategy in Isabelle/HOL
- Built an executable specification together with a total correctness proof
- Connected the strategy to the chess rules inside the theorem prover

Approach

- Central lemmas were proved automatically by the integrated Z3 SMT solver
- The proof obligations were expressed in linear integer arithmetic
- Shows the synergy of interactive theorem proving with push-button SMT automation

Reference

- Filip Marić, Predrag Janičić, Marko Maliković. *Proving Correctness of a KRK Chess Endgame Strategy by using Isabelle/HOL and Z3*. 2015.

Applications of SAT/SMT Solvers

Combinatorial enumeration

Formalization and Verification of SAT/SMT Solvers
Implementation of SAT/SMT solvers
Applications of SAT/SMT Solvers
- Software verification
- Scheduling and timetabling
- SAT Encodings
- SAT / SMT solvers in Proof Assistants
- Combinatorial enumeration
- Coherent Logic and Larus

Enumeration of non-isomorphic CC systems

Results

- Efficient enumeration of point configurations in a plane
- Related to order-type enumeration (mathematical classification of a finite set of points based on their relative spatial configurations, ignoring specific distances or scaling)
- Knuth's axioms for point orderings in a plane (D. Knuth, Axioms and Hulls, 1992.)
- Main techniques used:
  - Faradžev-Read enumeration algorithm scheme
  - SAT-based enumeration of augmented structures
  - Efficient permutation-space pruning algorithm

Publications

- Milan Banković, Filip Marić. *Faradžev Read-type enumeration of non-isomorphic CC systems*. *Computational Geometry*, 2021.

Applications of CP solving and planning

Results

- Triangle construction problems in geometry reduced to constraint solving and automated planning
  - construction problems expressed as automated planning problems
  - the obtained models solved by automated planners, as well as by reduction to finite-domain constraint solving

Publications

- Milan Banković. *Automation of triangle straightedge-and-compass constructions using automated planning*. *Annals of Mathematics and Artificial Intelligence*, 2025.

Applications of SAT/SMT Solvers

Coherent Logic and Larus

Formalization and Verification of SAT/SMT Solvers
Implementation of SAT/SMT solvers
Applications of SAT/SMT Solvers
- Software verification
- Scheduling and timetabling
- SAT Encodings
- SAT / SMT solvers in Proof Assistants
- Combinatorial enumeration
- Coherent Logic and Larus

Coherent Logic (CL)

- Coherent logic formula (universally closed):

$$A_0(\vec{x}) \wedge \ldots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x}, \vec{y}) \vee \ldots \vee \ B_{m-1}(\vec{x}, \vec{y}))$$

- No function symbols of arity > 0 nor negation symbols
- Many theories can be directly formulated in CL
- Every FOL theory can be translated into CL, possibly with additional predicate symbols

A System of Inference Rules for CL

$$\frac{\Gamma, ax, A_0(\vec{a}), \ldots, A_{n-1}(\vec{a}), \underline{B_0(\vec{a},\vec{b}) \vee \ldots \vee B_{m-1}(\vec{a},\vec{b})} \vdash P}{\Gamma, ax, A_0(\vec{a}), \ldots, A_{n-1}(\vec{a}) \vdash P}\ \text{MP}$$

$$\frac{\Gamma, \underline{B_0(\vec{c})} \vdash P \quad \ldots \quad \Gamma, \underline{B_{m-1}(\vec{c})} \vdash P}{\Gamma, B_0(\vec{c}) \vee \ldots \vee B_{m-1}(\vec{c}) \vdash P}\ \text{QEDcs (case split)}$$

$$\frac{}{\Gamma, \underline{B_i(\vec{a},\vec{b})} \vdash \exists \vec{y}(B_0(\vec{a},\vec{y}) \vee \ldots \vee\ B_{m-1}(\vec{a},\vec{y}))}\ \text{QEDas (assumption)}$$

$$\frac{}{\Gamma, \bot \vdash P}\ \text{QEDefq (ex falso quodlibet)}$$

$\underline{\Phi}$ denotes the list of conjuncts in Φ if Φ is a conjunction, and the formula itself otherwise. Rules are read bottom-up

The number of disjuncts can be reduced to two

CL Example

1. ax0 : $\forall X\ (p(X) \Rightarrow q(X)\ )$
2. ax1 : $\forall X\ (q(X) \Rightarrow r(X) \vee s(X)\ )$
3. ax2 : $\forall X\ (r(X) \Rightarrow \bot\ )$

Conjecture: $\forall X\ (p(X) \Rightarrow s(X)\ )$

Proof: Consider an arbitrary a such that: $p(a)$.
It should be proved that $s(a)$.

0. $q(a)$ (by MP, from $p(a)$ using axiom ax0; instantiation: $X \mapsto a$)
1. $r(a) \vee s(a)$ (by MP, from $q(a)$ using axiom ax1; instantiation: $X \mapsto a$)
  2. Case $r(a)$:
    3. $\bot$ (by MP, from $r(a)$ using axiom ax2; instantiation: $X \mapsto a$)
    4. Contradiction! (by QEDefq)
  5. Case $s(a)$:
    6. Proved by assumption! (by QEDas)
7. Proved by case split! (by QEDcs, by $r(a), s(a)$)

Theorem Proving as Constraint Solving

- In traditional ATP approaches, the search space consists of formulae, and the goal is to derive a specific formula
- In this approach, we search for a proof as a whole
- A proof can be encoded as a sequence of natural numbers that satisfy specific structural constraints
- A sought, unknown proof is represented by variables and constraints over a finite domain
- A suitable solver (e.g., SAT/SMT) finds a model for those constraints, from which the proof is reconstructed

Correctness

In a CL/CLF theory, there exists a proof of conjecture $\mathcal{T}$
of length less than k

if and only if

the described constraint has a model

Larus Prover

- Predrag Janičić and Julien Narboux: The Larus prover, implemented in C++ and is open-source: https://github.com/janicicpredrag/Larus
- Built for CL/CLF (but could be used for other logics)
- For finding a model, Larus uses SAT, SMT, and CSP solvers
- CL: without function symbols; CLF: with function symbols
- Both CL and CLF are as expressive as FOL.
- In Larus (for CLF), rather than encoding terms as trees, an SMT support for *inductive data types* is used

Reference

- Predrag Janičić, Julien Narboux. *Theorem Proving as Constraint Solving with Coherent Logic. Journal of Automated Reasoning*, 66(4):689–746, 2022.

CLF - Example

Axioms:

1. ax1 : $\forall X\ (f(f(f(f(f(X))))) = X\ )$
2. ax2 : $\forall X\ (f(f(f(X))) = X\ )$

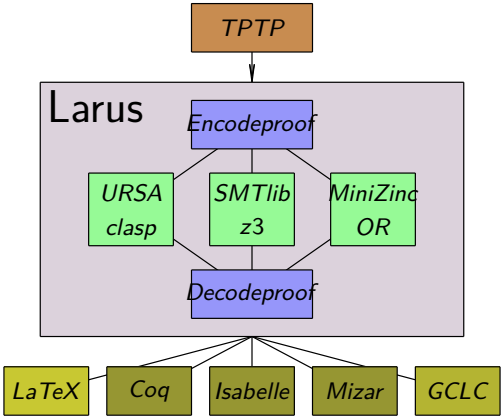
Theorem

fc : $\forall X\ (f(X) = X\ )$

Consider an arbitrary a. It should be proved that $f(a) = a$.

0. $f(f(f(f(f(f(a)))))) = f(f(f(a)))$ (by MP, using axiom ax2)
1. $f(f(f(f(f(f(a)))))) = f(a)$ (by MP, using axiom ax1)
2. $f(a) = f(f(f(a)))$ (by MP, from $f(f(f(f(f(f(a)))))) = f(f(f(a)))$, $f(f(f(f(f(f(a)))))) = f(a)$ using axiom eqnativeEqSub0)
3. $f(f(f(a))) = a$ (by MP, using axiom ax2)
4. $f(a) = a$ (by MP, from $f(a) = f(f(f(a)))$, $f(f(f(a))) = a$ using axiom eqnativeEqSub1)
5. Proved by assumption! (by QEDas)

Generation of Readable and Machine-Verifiable Proofs



Additional Features: What can (only) Larus do (easily)?

- Generation of readable and machine-verifiable proofs
- Generation of shortest proofs
- Completion of incomplete proofs – using "hints" [Add constraints]
- Completion of incomplete premises (abduction) [Omit constraints]
- Completion of incomplete goals [Omit constraints]
- Automated generation of proof illustrations

Completion of incomplete premises/Abduction

1. ax0 : $\forall X\ (p(X) \Rightarrow q(X)\ )$
2. ax1 : $\forall X\ (q(X) \Rightarrow r(X) \vee s(X)\ )$
3. ax2 : $\forall X\ (r(X) \Rightarrow \bot\ )$

Theorem

$\forall X\ (\ p(X) \Rightarrow\ s(X)\ )$

The conjecture cannot be proven, but Larus will offer two abducts:

```
1. set:
  1. ((q(a)))
Abducts CONSISTENT!

2. set:
  1. ((p(a)))
Abducts CONSISTENT!
```

Completion of incomplete goals – Example

1. a0 : $cronus \neq zeus \wedge cronus \neq poseidon \wedge cronus \neq apolon \wedge zeus \neq poseidon \wedge zeus \neq apolon \wedge poseidon \neq apolon$
2. a1 : $father(zeus) = cronus$
3. a2 : $father(poseidon) = cronus$
4. a3 : $father(apolon) = zeus$
5. a4 : $\forall X\ \forall Y\ (father(X) = father(Y) \wedge X \neq Y \Rightarrow brother(X, Y))$
6. a5 : $\forall X\ \forall Y\ (brother(X, father(Y)) \Rightarrow uncle(X, Y))$

Theorem

$uncle(_, apolon)$

0. $cronus = father(zeus)$ (by MP, using axiom eqnativeEqSub0; instantiation: $A \mapsto father(zeus)$, $B \mapsto father(zeus)$, $X \mapsto cronus$)
1. $father(apolon) = zeus$ (by MP, using axiom eqnativeEqSub0; instantiation: $A \mapsto father(apolon)$, $B \mapsto zeus$, $X \mapsto father(apolon)$)
2. $father(poseidon) = father(zeus)$ (by MP, from $cronus = father(zeus)$ using axiom eqnativeEqSub1; instantiation: $A \mapsto father(poseidon)$, $B \mapsto cronus$, $X \mapsto father(zeus)$)
3. $brother(poseidon, zeus)$ (by MP, from $father(poseidon) = father(zeus)$ using axiom a4; instantiation: $X \mapsto poseidon$, $Y \mapsto zeus$)
4. $brother(poseidon, father(apolon))$ (by MP, from $brother(poseidon, zeus)$, $father(apolon) = zeus$ using axiom brotherEqSub1; instantiation: $A \mapsto poseidon$, $B \mapsto zeus$, $X \mapsto father(apolon)$)
5. $uncle(poseidon, apolon)$ (by MP, from $brother(poseidon, father(apolon))$ using axiom uncleEqSub0; instantiation: $A \mapsto poseidon$, $B \mapsto apolon$, $X \mapsto poseidon$)
6. Proved by assumption! (by QEDas)