# ekstrakto, a tool to reconstruct Dedukti proofs from TSTP files

Mohamed Yacine EL HADDAD, Guillaume BUREL, Frédéric BLANQUI

26-08-2019

CNRS, LSV, ENS Paris-Saclay, CNRS, INRIA, University of Paris-Saclay, DigiCosme

# Table of contents

# Introduction

Theorem provers:

# Introduction

Theorem provers:

- Interactive: **Dedukti**, Coq, Matita, Isabelle...

# Introduction

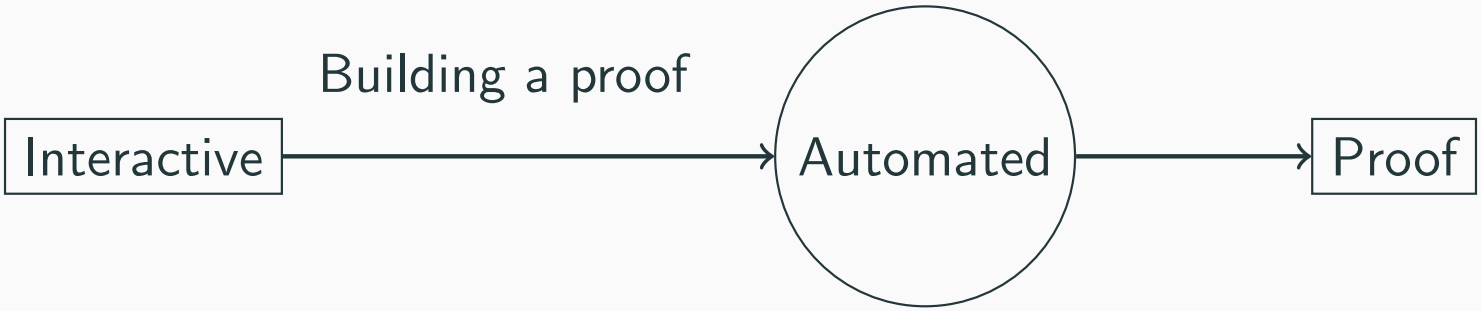
Theorem provers:

- Interactive: **Dedukti**, Coq, Matita, Isabelle...
- Automated: Vampire, Z3, Zenon, **ZenonModulo**, **ArchSAT**, E Prover, iProverModulo...

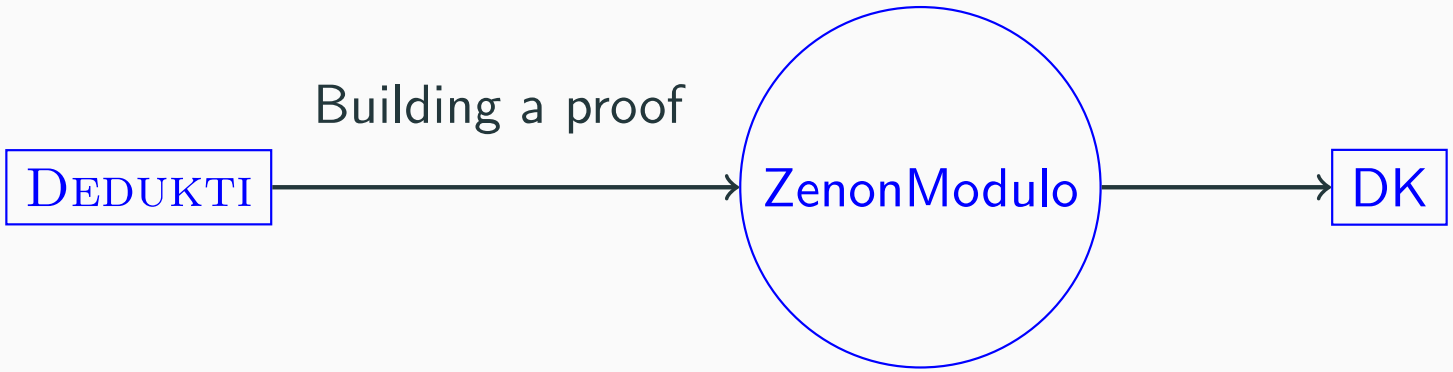
# Introduction

Theorem provers:

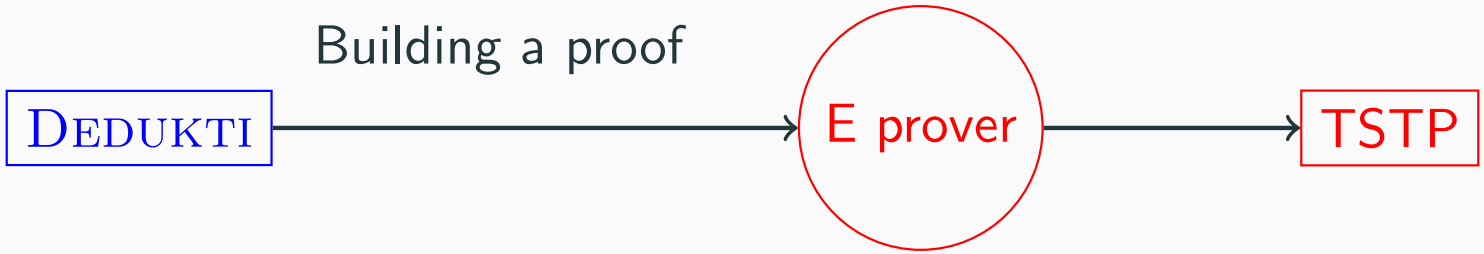
- Interactive: **Dedukti**, Coq, Matita, Isabelle...
- Automated: Vampire, Z3, Zenon, **ZenonModulo**, **ArchSAT**, E Prover, iProverModulo...



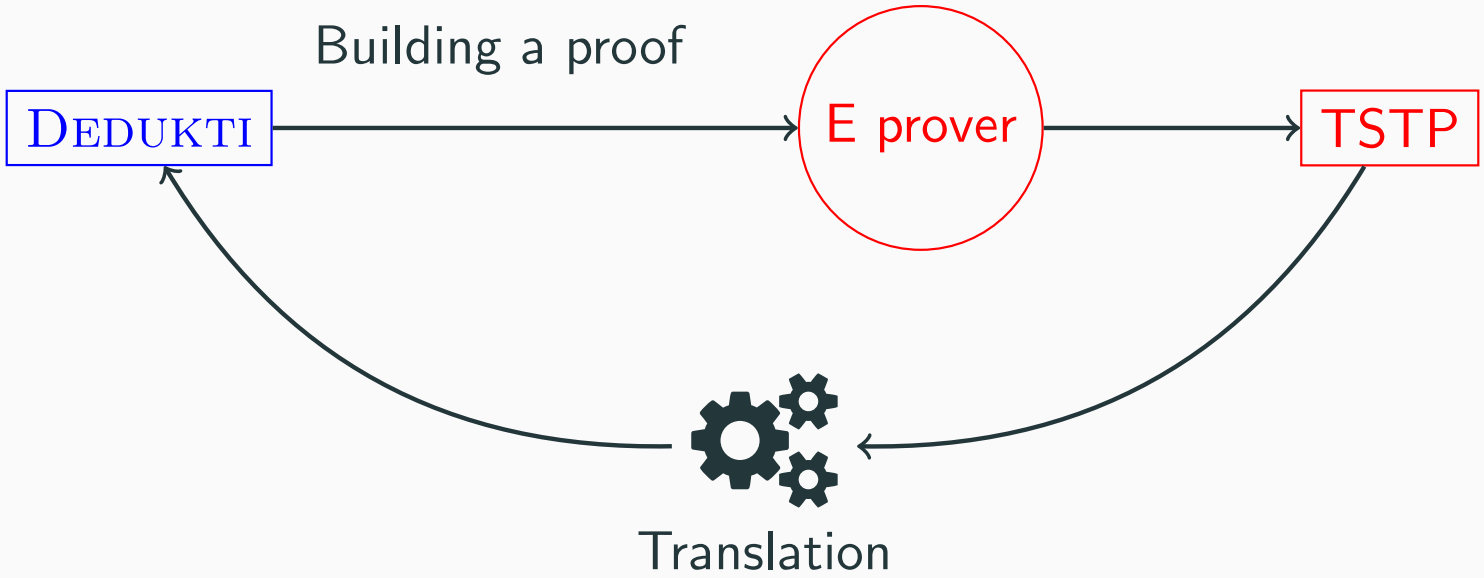
# Introduction



# Introduction



# Introduction



# TPTP/TSTP

# TPTP and TSTP

TPTP is a standard library to test automated theorem provers.

```
cnf(name, role, formula, source).
```

TSTP is a library of solutions for TPTP problems.

```
cnf(name, role, formula, name).
cnf(name, role, formula,inference(name,infos,[P0, P1,..., Pn])).
```

# TPTP and TSTP

```
<source>              :== <dag_source> | [ <sources> ] | ...
<dag_source>          :== <name>
                        |inference(..., ..., <inference_parents>)
<inference_parents>   :== [] | [ <sources> ]
<sources>             :== <source> (, <source>)*
```

# Dedukti

# Introduction

Dedukti[1] is a **proof checker** based on $\lambda\Pi^{\equiv}$ Theory, an extension of the classical $\lambda\Pi$-*calculus*.
Properties:

- Very expressive (dependent types, rewrite rules...)
- Fast
- Higher Order Rewriting
- Small kernel
- Could be translated to other systems

[1] https://github.com/Deducteam/Dedukti.git

# First order logic in Dedukti

| $Form : Type$ | $term : Type$ |
|---|---|
| $x : term$ | $\varphi(x) = x$ |
| $f : term \to ... \to term \to term$ | $\varphi(f\ t_1 ... t_n) := f\, \varphi(t_1)\ ...\ \varphi(t_n)$ |
| $\dot{\bot} : Form$ | $\varphi(\bot) := \dot{\bot}$ |
| $\dot{\top} : Form$ | $\varphi(\top) := \dot{\top}$ |
| $\dot{\neg} : Form \to Form$ | $\varphi(\neg A) := \dot{\neg}\varphi(A)$ |
| $\dot{\wedge} : Form \to Form \to Form$ | $\varphi(A \wedge B) := \varphi(A)\ \dot{\wedge}\ \varphi(B)$ |
| $\dot{\vee} : Form \to Form \to Form$ | $\varphi(A \vee B) := \varphi(A)\ \dot{\vee}\ \varphi(B)$ |
| $\dot{\Rightarrow} : Form \to Form \to Form$ | $\varphi(A \Rightarrow B) := \varphi(A) \dot{\Rightarrow} \varphi(B)$ |
| $\dot{\forall} : (term \to Form) \to Form$ | $\varphi(\forall x A) := \dot{\forall}(\lambda x : term.\ \varphi(A))$ |
| $\dot{\exists} : (term \to Form) \to Form$ | $\varphi(\exists x A) := \dot{\exists}(\lambda x : term.\ \varphi(A))$ |
| $\dot{=} : term \to term \to Form$ | $\varphi(x = y) := \varphi(x) \dot{=} \varphi(y)$ |
| $P : term \to ... \to term \to Form$ | $\varphi(P t_1 ... t_n) := P\ \varphi(t_1) ...\ \varphi(t_n)$ |

$$\varphi(A) : Form$$

**Figure 1:** Encoding of first order logic in DEDUKTI

# Curry-Howard correspondence

We define a new interpretation symbol *prf* to interpret a formula as type:

$$prf \quad : \quad Form \rightarrow Type$$

A proof of a formula $F$ is a DEDUKTI term $t$ that has the type *prf (F)*
**Goal**: find the term $t$ from a TSTP file.

# Zenon Modulo & ArchSAT

Zenon Modulo:

- Build easily **checkable** proofs
- Use the **Tableau** method
- Accept **TPTP** format
- Generate a **low level** format of proofs

ArchSAT:

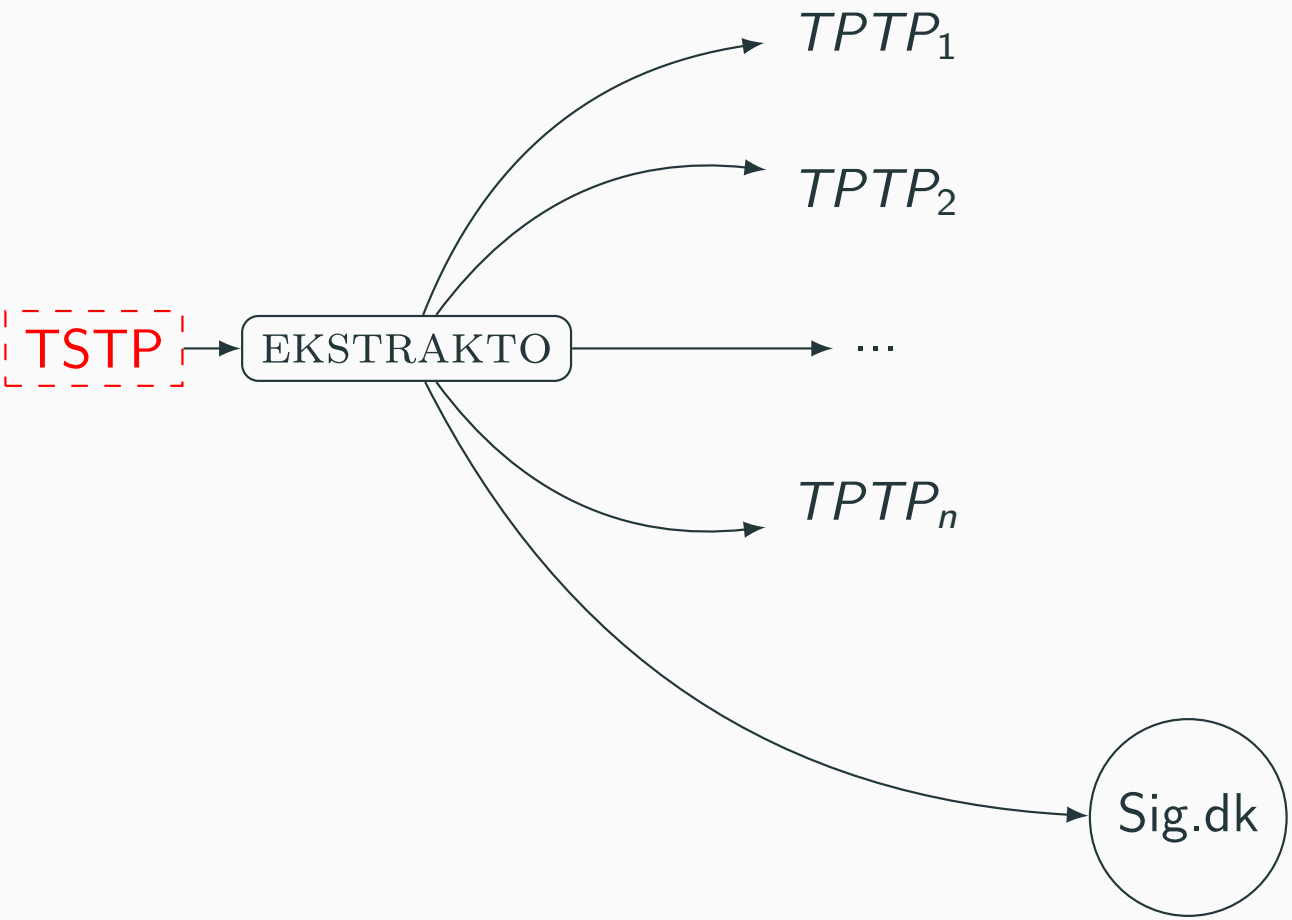
- Use **Tableau** and **Superposition**.
- Accept **TPTP**, **SMTLib** formats.
- Generate Dedukti proofs.

# ekstrakto

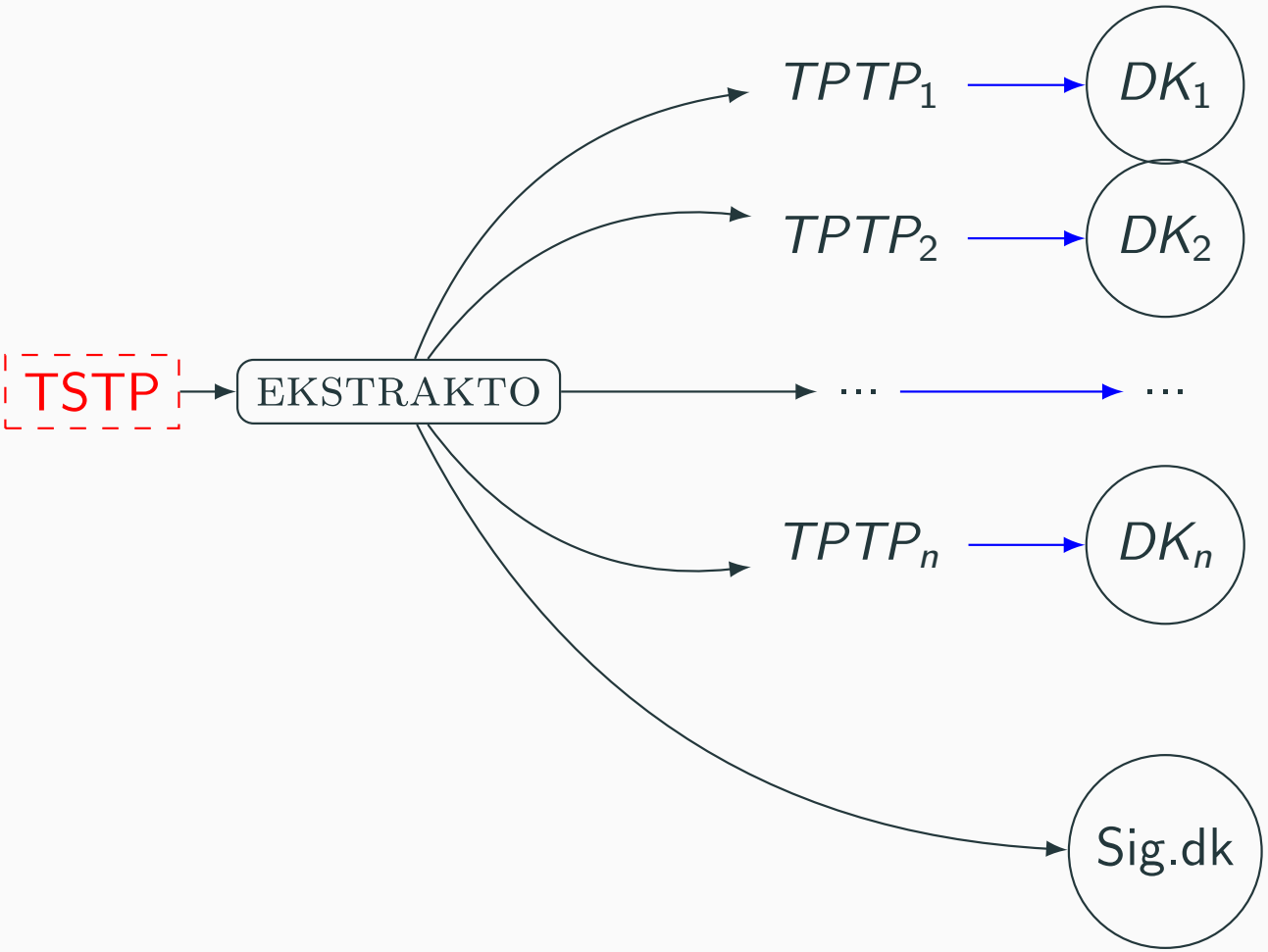
# ekstrakto

# ekstrakto

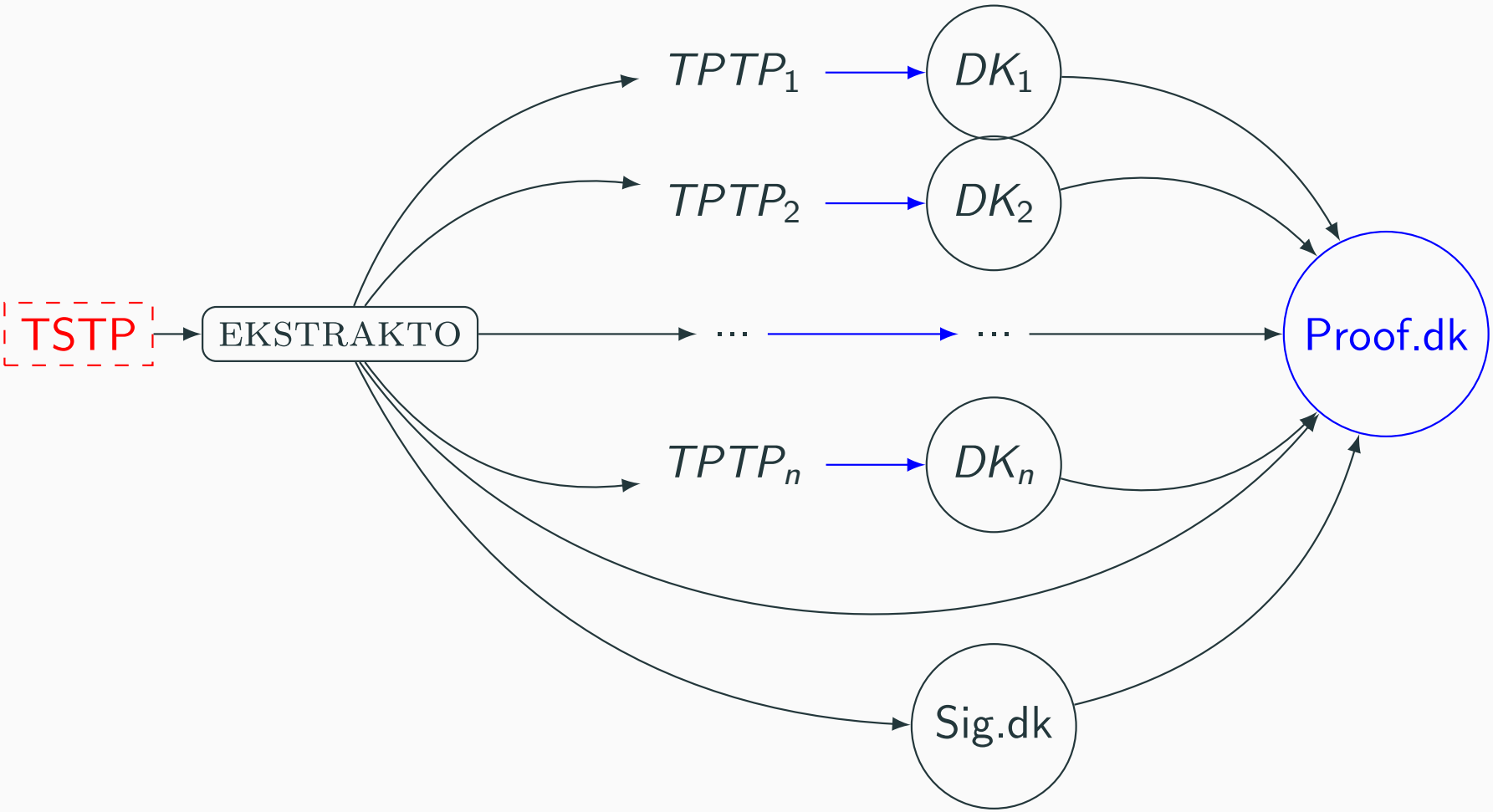
# ekstrakto



# ekstrakto

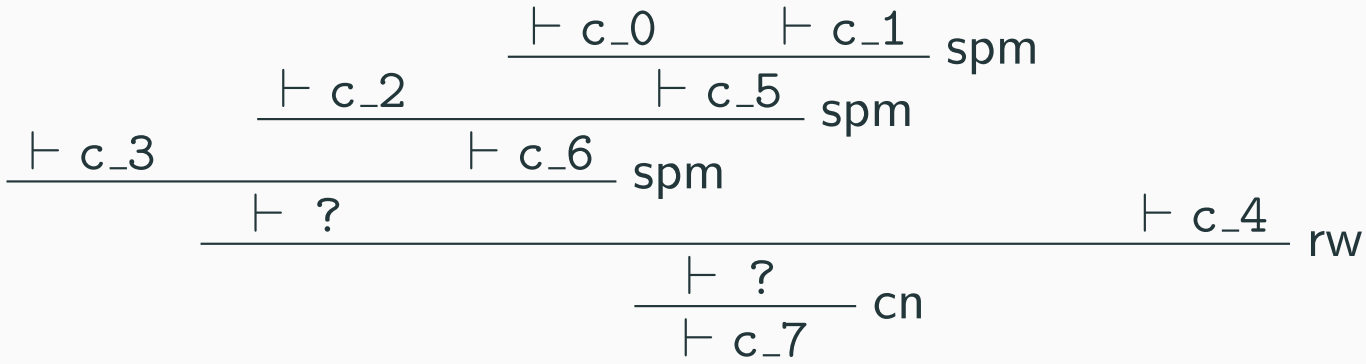


# ekstrakto



# TSTP

```
cnf(c_0,axiom,
        ( subset(X1,X2)
        | ~ equal_sets(X1,X2) )).
cnf(c_1,hypothesis,
        ( equal_sets(b,bb) )).
cnf(c_2,axiom,
        ( member(X1,X3)
        | ~ member(X1,X2)
        | ~ subset(X2,X3) )).
cnf(c_3,negated_conjecture,
        ( ~ member(element_of_b,bb) )).
cnf(c_4,hypothesis,
        ( member(element_of_b,b) )).
cnf(c_5,hypothesis,
        ( subset(b,bb) ),
        inference(spm,[status(thm)],[c_0,c_1])).
cnf(c_6,hypothesis,
        ( member(X1,bb)
        | ~ member(X1,b) ),
        inference(spm,[status(thm)],[c_2,c_5])).
cnf(c_7,negated_conjecture,
        ( $false ),
        inference(cn,[status(thm)],[inference(rw,[status(thm)],
        [inference(spm,[status(thm)],[c_3,c_6]),c_4])]), [proof]).
```


⊢ c_0
⊢ c_1
spm
⊢ c_2
⊢ c_5
spm
⊢ c_3
⊢ c_6
spm
⊢ ?
⊢ c_4
rw
⊢ ?
cn
⊢ c_7

# TSTP

The TSTP file give informations about how the proof is generated. It contains all axioms and hypothesis used in the proof.

**Exemple**:

$$
\begin{array}{ll}
C_0, C_1 & \vdash C_5 \\
C_2, C_5 & \vdash C_6 \\
C_3, C_4, C_6 & \vdash C_7
\end{array}
$$

EKSTRAKTO will generate a sub problem for each deduction:

c_5.p

```
fof(c_5, conjecture, (
            (![X1, X2] : (s (X1, X2)|~equal_sets (X1, X2)))
        =>  ((equal_sets (b, bb))
        =>  (subset (b, bb))))).
```

c_6.p

```
fof(c_6, conjecture,(
        (![X1, X2, X3] : (member (X1, X3)|~member (X1, X2) |~subset
    (X2, X3)))
        =>  ((subset (b, bb))
        =>  (![X1] : (member (X1, bb)|~member (X1, b)))))).
```

c_7.p

```
fof(c_7, conjecture, (
            (~member (element_of_b, bb))
        =>  ((![X1] : (member (X1, bb)|~member (X1, b)))
        =>  ((member (element_of_b, b))
        =>  ($false))))).
```

We call ZenonModulo to prove the sub problem. The proof is not completed since we do not have any link between deductions.

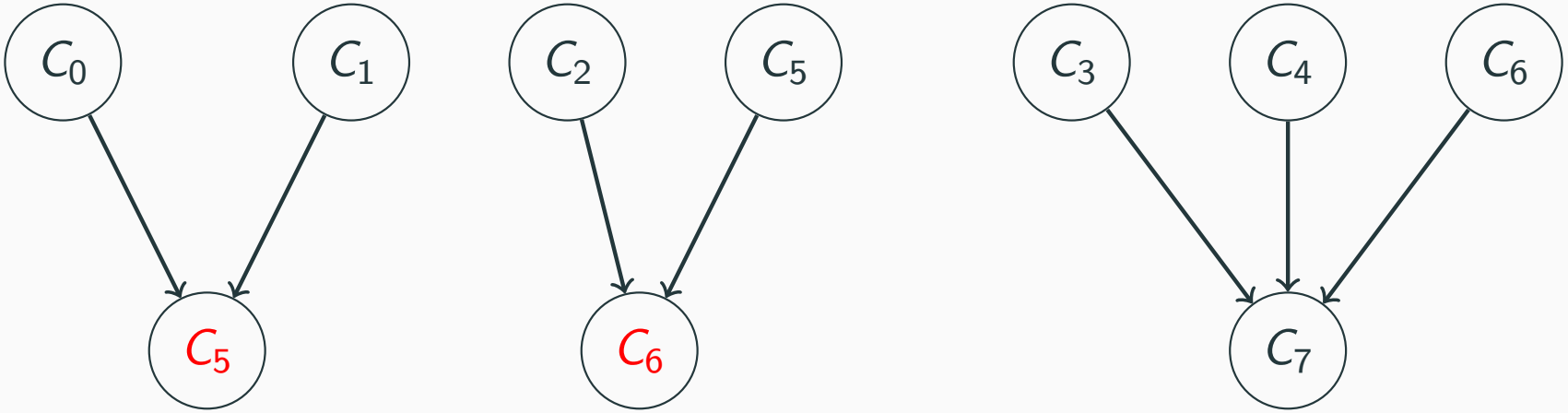



**Figure 2:** Diagram of proofs generated by ZenonModulo

We call ZenonModulo to prove the sub problem. The proof is not completed since we do not have any link between deductions.

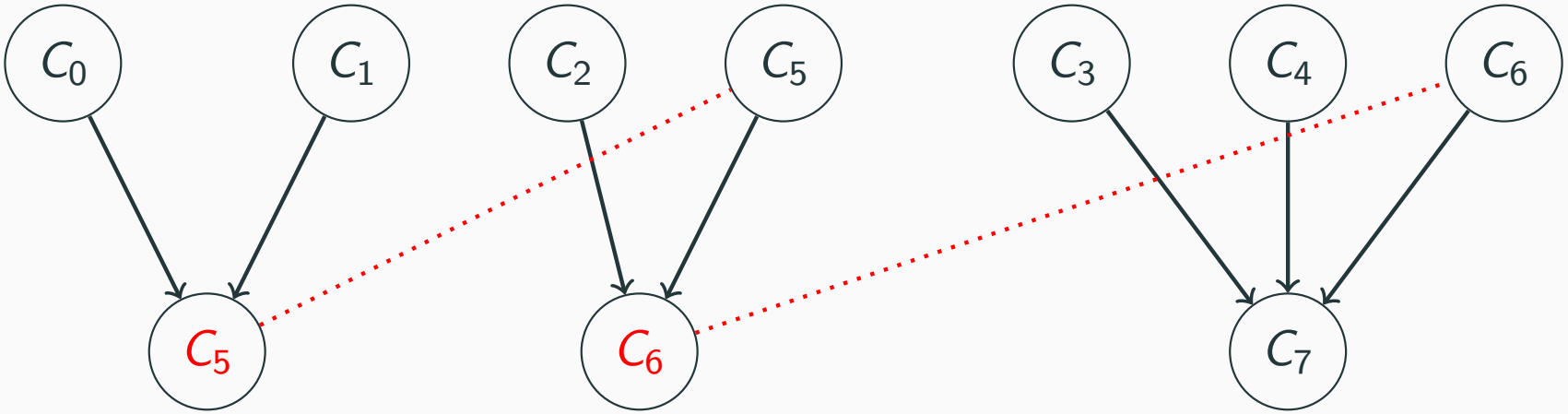



**Figure 2:** Diagram of proofs generated by ZenonModulo

$$
\begin{aligned}
\mathsf{c\_5.lp} \quad &: \mathit{prf}\,\varphi(C_0) \to \mathit{prf}\,\varphi(C_1) \to \mathit{prf}\,\varphi(C_5) \\
\mathsf{c\_6.lp} \quad &: \mathit{prf}\,\varphi(C_2) \to \mathit{prf}\,\varphi(C_5) \to \mathit{prf}\,\varphi(C_6) \\
\mathsf{c\_7.lp} \quad &: \mathit{prf}\,\varphi(C_3) \to \mathit{prf}\,\varphi(C_4) \to \mathit{prf}\,\varphi(C_6) \to \mathit{prf}\,\varphi(C_7)
\end{aligned}
$$

$$
\begin{aligned}
\text{c\_5.lp} \quad &: \mathit{prf}\,\varphi(C_0) \to \mathit{prf}\,\varphi(C_1) \to \mathit{prf}\,\varphi(C_5) \\
\text{c\_6.lp} \quad &: \mathit{prf}\,\varphi(C_2) \to \mathit{prf}\,\varphi(C_5) \to \mathit{prf}\,\varphi(C_6) \\
\text{c\_7.lp} \quad &: \mathit{prf}\,\varphi(C_3) \to \mathit{prf}\,\varphi(C_4) \to \mathit{prf}\,\varphi(C_6) \to \mathit{prf}\,\varphi(C_7)
\end{aligned}
$$

But what we want is just **one** term of type:

$$
\mathit{prf}\,\varphi(C_0) \to \mathit{prf}\,\varphi(C_1) \to \mathit{prf}\,\varphi(C_2) \to \mathit{prf}\,\varphi(C_3) \to \mathit{prf}\,\varphi(C_4) \to \mathit{prf}\,\varphi(C_7)
$$

Proof.lp:

```
definition proof_trace
(hyp_c_0 : zen.Proof(φ (c_0)))
(hyp_c_1 : zen.Proof(φ (c_1)))
(hyp_c_2 : zen.Proof(φ (c_2)))
(hyp_c_3 : zen.Proof(φ (c_3)))
(hyp_c_4 : zen.Proof(φ (c_4)))
: zen.seq
:=
let lemmas_c_5 = c_5.delta hyp_c_0 hyp_c_1 in
let lemmas_c_6 = c_6.delta hyp_c_2 lemmas_c_5 in
let lemmas_c_7 = c_7.delta hyp_c_3 lemmas_c_6
   hyp_c_4 in lemmas_c_7
```

# Experiments

**Table 1:** Percentage of DEDUKTI proofs on the 7922 TPTP files

| Prover | % TPTP |
|---|---|
| *ZenonModulo* | 13% |
| *ArchSAT* | 6% |
| *E prover* | No proof term |

From 7922 problems, **E prover** generated 4582 TSTP files.
From 4582 TSTP files, EKSTRAKTO generated 362556 sub-problems.

**Table 2:** Percentage of DEDUKTI sub-proofs on the 362556 extracted TPTP files

| Prover | % TPTP |
|---|---|
| ZenonModulo | 90% |
| ArchSAT | 96% |
| ZenonModulo ∪ ArchSAT | 97% |

**Table 3:** Percentage of DEDUKTI proofs on the 4582 TSTP files generated by *E prover*

| Prover | Global proofs | % TSTP |
|---|---|---|
| ArchSAT | 2793 | 61% |
| ZenonModulo | 2189 | 48% |
| ZenonModulo ∪ ArchSAT | 3285 | 71% |

# Further works

- Add a support for **Skolemisation**[2].
- Add arithmetic.
- Handle typing of TSTP files.

[2] [G.Dowek and B.Werner 2005] http://www.lsv.fr/∼dowek/Publi/skolem.pdf

Thank you !