A language-independent ecosystem for program verification

#### Amélie LEDEIN

#### supervised by Catherine DUBOIS and Valentin BLOT





# A way to do program verification



Amélie LEDEIN

WG3 meeting - Timisoara

Thursday 9th February 2023 1 / 12

# A way to do program verification



Amélie LEDEIN

WG3 meeting - Timisoara

Thursday 9th February 2023 1 / 12

# A way to do program verification



Amélie LEDEIN

WG3 meeting - Timisoara

Thursday 9th February 2023 1 / 12

• Objectives:

#### • Objectives:

1. Formalize the translation from  $\mathbb K$  to  $_{{\rm KORE}}$ 

#### • Objectives:

- 1. Formalize the translation from  $\mathbb K$  to  $_{{\rm KORE}}$
- 2. Translate a  $\mathbb{K}$  semantics into DEDUKTI, to execute it.

#### • Milestones:

1. Test the semantics.



#### • Objectives:

- 1. Formalize the translation from  $\mathbb K$  to  $_{\rm KORE}$
- Translate a K semantics into DEDUKTI, to execute it.
- 3. Translate a K semantics into DEDUKTI, to check proofs.

- 1. Test the semantics.
- 2. Recheck  $\operatorname{KPROVER}$  proofs.



#### • Objectives:

- 1. Formalize the translation from  $\mathbb K$  to  $_{\rm KORE}$
- Translate a K semantics into DEDUKTI, to execute it.
- 3. Translate a K semantics into DEDUKTI, to check proofs.

- 1. Test the semantics.
- 2. Recheck KPROVER proofs.
- Prove properties about a language L described in K.



#### • Objectives:

- 1. Formalize the translation from  $\mathbb K$  to  $_{{\rm KORE}}$
- Translate a K semantics into DEDUKTI, to execute it.
- 3. Translate a K semantics into DEDUKTI, to check proofs.
- Export K semantics into CoQ, AGDA, etc. thanks to DEDUKTI.

- 1. Test the semantics.
- 2. Recheck KPROVER proofs.
- Prove properties about a language L described in K.



#### • Objectives:

- 1. Formalize the translation from  $\mathbb K$  to  $_{{\rm KORE}}$
- Translate a K semantics into DEDUKTI, to execute it.
- 3. Translate a K semantics into DEDUKTI, to check proofs.
- Export K semantics into CoQ, AGDA, etc. thanks to DEDUKTI.

- 1. Test the semantics.
- 2. Recheck KPROVER proofs.
- Prove properties about a language L described in K.
- 4. Allow multi-formalism semantics.



# KORE: A MATCHING LOGIC theory

axiom{R} \exists{R} (Val:Sortint{}, \equals{Sortint{}, R} (Val:Sortint{}, Lbl'UndsPlus'Int'Unds'{}(K0:Sortint{}, K1:Sortint{}))) [functional{}()] // functional axion(R) \exists(R) (Val:SortString{), \equals{SortString{}, R} (Val:SortString{}, Lbl'UndsPlus'String'UndsUnds'STRING-COWWON'Unds'String'Unds'String{}, K1:SortString{}))) [f axiom(R} \exists(R) (Val:SortAExp{}, k] (Val:S axion{}\inplies{SortAExp{}} (\and{SortAExp{}} (Lbl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}(X0:SortAExp{}, X1:SortAExp{}), Lbl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp axlom(]\not{SortAExp{}} (\and{SortAExp}) (Lbl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}, X1:SortAExp{}, Lbl'Unds'-'UndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp axion{\\not{SortAExp{}} (\and{SortAExp{}} (\bl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}, X1:SortAExp{}, Lbl'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{} axion(R) \exists(R) (Val;SortIds(), \equals(SortIds(), R) (Val;SortIds(), Lbl'UndsCormUndsUnds'IMP-SYNTAX'Unds'Ids'Unds'Id'Unds'Ids()(K0;SortId(), K1:SortIds()))) [functional()()] // functional axion{}\inplies{SortIds{}} (\bl'UndsComMUndsUnds'IMP-SYNTAX'Unds'Ids'Unds'Ids'Id'Unds'Ids{}(X0:SortId{}, X1:SortIds{}), Lbl'UndsComMUndsUnds'IMP-SYNTAX'Unds'Ids'Unds'Ids[}(Y0:SortId axion(R} \exists(R} (Val:SortInt{}, \equals{SortInt{}, R} (Val:SortInt{}, Lbl'Unds'-Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional axiom(R) \exists(R) (Val:SortMap(), \equals(SortMap(), R) (Val:SortMap(), Lbl'Unds'-Map'UndsUnds'MAP'Unds'Map'Unds'Map()(K0:SortMap(), K1:SortMap()))) [functional()()] // functional axiom{R} \exists{R} (Val:SortAExp{}, k] (Val:SortAExp{}, R} (Val:SortAExp{}, R) (Val:SortAExp{}, Lbl'Unds'-IUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}, K1:SortAExp{}, K1:SortAExp{}))) [functional{}()] // functional axlom{}\inplies{SortAExp{}} (\and{SortAExp{}} (Lbl'Unds'-'UndsUnds'AExp'Unds'AExp'Unds'AExp[)(X0:SortAExp{}, X1:SortAExp{}), Lbl'Unds'-'UndsUnds'AExp'Unds'AExp'Unds'AExp'Unds'AExp axiom(\\not{SortAExp{}} (\and{SortAExp{}} (Lb1'Unds'-'UndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp[)(X0:SortAExp{}, X1:SortAExp{}), Lb1'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp'Unds'AExp axion(R} \exists(R) (Val:SortAExp{}, k] (Val:SortAExp{}, R} (Val:SortAExp{}, R) (Val:S axion(R} \exists(R} (Val:SortBool{}, \equals(SortBool{}, R) (Val:SortBool{}, Lbl'Unds-LT-Eqls'Int'Unds'()(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional axion(R) \exists(R) (Val:SortBool(). \equals(SortBool(). R) (Val:SortBool(). Lbl'Unds-LT-Eols'Map'UndsUnds'MAP'Unds'Bool'Unds'Map(). Ki:SortMap(). Ki:SortMap())) [functional()()] // functional axion(R} \exists(R} (Val:SortBool(), \equals(SortBool(), R} (Val:SortBool(), Lbl'Unds-LT-Eqls'Set'UndsUnds'SET'Unds'Bool'Unds'Set{}(K0:SortSet{}, K1:SortSet{}))) [functional()] // functional axiom(R) \exists(R) (Val:SortBool(), \equals(SortBool(), R) (Val:SortBool(), Lbl'Unds-LT-Eqls'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String'Unds'String()(K0:SortBtring(), K1:SortString())) [funct axion(R) \exists(R) (Val:SortBExp(), R) (Val:SortBExp(), R) (Val:SortBExp(), Lbl'Unds-LT-EqlSUndsUnds'INP-SYNTAX'Unds'BExp'Unds'AExp()(K0:SortAExp(), K1:SortAExp())) [functional()] // functional() axion{}\inplies{SortBExp{}} (\and{SortBExp{}} (Lbl'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp{}(X0:SortAExp{}, X1:SortAExp{}), Lbl'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp' axtom()\not{SortBExp{}} (\and{SortBExp{}} (Lbl'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp'Unds'AExp(), X1:SortAExp{}, X1:SortAExp{}), Lbl'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp axion{\\not{SortBExp{}} (\and{SortBExp{}} (\bl\'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp{}(X8:SortAExp{}), Lb\'Unds'and'UndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp'\) axion{}\not{Sort8Exp{}} (Lbl'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp'Unds'AExp{}, X1:SortAExp{}, Lblnot'UndsUnds'IMP-SYNTAX'Unds'BExp{Unds'BExp{}} axion(R) \exists(R) (Val:SortBool(), \equals(SortBool(), R) (Val:SortBool(), Lbl'Unds-LT-'Int'Unds'()(K0:SortInt(), K1:SortInt()))) [functional()()] // functional axiom/R} \exists{R} (Val:SortBool{}, lequals{SortBool{}, R} (Val:SortBool{}, Lbl'Unds-LT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String'Unds'String{}, K1:SortBool{}, Lbl'Unds-LT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String'Unds'String{}, K1:SortBool{}, Lbl'Unds-LT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String'Unds'String{}, K1:SortBool{}, Lbl'Unds-LT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String{}, K1:SortBool{}, K1:SortBool{}, Lbl'Unds-LT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String{}, K1:SortBool{}, K1:SortBool{}, Lbl'Unds-LT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String{}, K1:SortBool{}, K1:SortBoo axion(R) \exists(R) (Val:SortBExp(}, \equals(SortBExp{}, R) (Val:SortBExp{}, Lbl'Unds-LT-UndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp'Unds'AExp{}, K1:SortAExp{}, K1:SortAExp{}))) [functional{})] // functional axion{}\inplies{SortBExp{}} (\and{SortBExp{}} (Lbl'Unds-LT-UndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp'Unds'AExp{}, X1:SortAExp{}, Lbl'Unds-LT-UndsUnds'IMP-SYNTAX'Unds'AExp axion(}\not{Sort8Exp{}} (\and{Sort8Exp{}} (Lbl'Unds-LT-UndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp'\AsisortAExp{}, X1:SortAExp{}, Lbl'Unds'and'UndsUnds'IMP-SYNTAX'Unds'BExp'Unds'BExp'Unds'AExp{} axlon{\\not{SortBExo{}} (\and{SortBExo{} (Lbl'Unds-LT-UndsUnds'INP-SYNTAX'Unds'BExo'Unds'AExo{} X1:SortAExo{}, Lblnot'UndsUnds'INP-SYNTAX'Unds'BExo{}(Y0:SortBExo{})); axion{R} \exists{R} (Val:SortBool{}, kguals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEglsSlshEqls'Bool'Unds'{}(K0:SortBool{}, K1:SortBool{})) [functional{}()] // functional axion(R} \exists(R) (Val:SortBool(), \equals(SortBool(), R) (Val:SortBool(), Lbl'UndsEqlsSlshEqls'Int'Unds'()(K0:SortInt(), K1:SortInt()))) [functional()()] // functional axion{R} \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEalsSlshEals'K'Unds'{}(K0:SortK{}, K1:SortK{}))) [functional{}()] // functional axion{R} \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEqlsSlshEqls'String'UndsUnds'STRING-COMMON'Unds'Bool'Unds'String{}(K0:SortString{}, K1:SortString{}))) [fille] axion(R} \exists(R) (Val:SortBool{}, \equals(SortBool{}, R) (Val:SortBool{}, Lbl'UndsEqlsEqls'Bool'Unds'{}(K0:SortBool{}, K1:SortBool{})) [functional{}()] // functional axion{R} \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEqlsEqls'Int'Unds'{}(K0:SortInt{}))) [functional{}()] // functional axion(R} \exists(R) (Val:SortBool(), \equals(SortBool(), R) (Val:SortBool(), Lbl'UndsEqlsEqls'K'Unds'{)(K0:SortK(), K1:SortK()))) [functional()()] // functional axion{R} \exists{R} (Val:SortBool{}, \equals{SortBool}, R} (Val:SortBool{}, R) (Val:SortBool{}, Lbl'UndsEqlsEqls'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String'Unds'String{}, K1:SortString{}))) [funct: axiom(R) \exists(R) (Val:SortStnt{), \equals(SortStnt{), R} (Val:SortStnt{), Lbl'UndsEqlsUndsSClnUnds'INP-SYNTAX'Unds'Stnt'Unds'Id'Unds'AExp{}(K0:SortId{), K1:SortAExp{}))) [functional{}()] // functional axion{\\inpl\es{SortStmt{}} (\and{SortStmt{}} (Lbl'UndsEalsUndsSClnUnds'IMP-SYNTAX'Unds'Id'Unds'Id'Unds'AExof)(X0:SortIdf). X1:SortAExof)). Lbl'UndsEalsUndsSClnUnds'IMP-SYNTAX'Unds'Id'Unds'Id'Unds'A axion{}\not{SortStmt{}} (\and{SortStmt{}} (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stnt'Unds'Id'Unds'AExp{}(X0:SortId{}, X1:SortAExp{}), Lbl'UndsUndsUnds'IMP-SYNTAX'Unds'Stnt'Unds'Stnt{}(Y0:SintAExp{})) axlom()\not{SortStmt(}} (\and{SortStmt(}) (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stmt'Unds'AExp()(X0:SortId(), X1:SortAExp()), Lbltf'Unds'then'Unds'else'UndsUnds'IMP-SYNTAX'Unds'Stmt'Unds'BEx axion{}\not{SortStmt{}} (\and{SortStmt{}} (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stnt'Unds'Id'Unds'AExp{}(X0:SortId{}, X1:SortAExp{}), Lbl'LBraUndsRBraUnds'IMP-SYNTAX'Unds'Stnt{}(Y0:SortStmt axlon()\not{SortStmt()} (\and{SortStmt()} (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stmt'Unds'Id'Unds'AExp()(X0:SortId(), X1:SortAExp()), Lbl'LBraRBraUnds'IMP-SYNTAX'Unds'Stmt{())) [constructor{()] // axion{R} \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'Unds-GT-Eqls'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional axion(R) \exists(R) (Val:SortBool(), \equals(SortBool{), R} (Val:SortBool{), Lbl'Unds-GT-Eqls'String'UndsUnds'STRING-COMMON'Unds'Bool'Unds'String'Unds'String{} (K8:SortString{}, K1:SortString{}))) [funct axion{R} \exists{R} (Val:SortBool{), \equals{SortBool{}, R} (Val:SortBool{}, Lbl'Unds-GT-'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional axion(R) \exists(R) (Val:SortBool{), \equals(SortBool{), R) (Val:SortBool{), Lbl'Unds-GT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String'Unds'String()(K8:SortString{), K1:SortString{))) [functional axion(R) \equals(SortList(), R) (Lbl'Unds'List'Unds'()(List'Unds'()(K1:SortList(),K3:SortList()),Lbl'Unds'List'Unds'()(K1:SortList(),Lbl'Unds'List'Unds'()(K2:SortList(),K3:SortList()),Lbl'Unds'List'Unds'()(K2:SortList(),K3:SortList()),Lbl'Unds'List'Unds'()(K2:SortList(),K3:SortList()),Lbl'Unds'List'Unds'List'Unds'()(K2:SortList(),K3:SortList()),Lbl'Unds'List'Unds'List'Unds'()(K2:SortList(),K3:SortList()),Lbl'Unds'List'Unds'List'Unds'()(K2:SortList(),K3:SortList()),Lbl'Unds'List'Unds'List'Unds'()(K2:SortList()),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'(),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'()),Lbl'Unds'List'

Amélie LEDEIN

WG3 meeting - Timisoara

# KORE: A MATCHING LOGIC theory

axiom{R} \exists{R} (Val:SortInt{}, \equals{SortInt{}, R} (Val:SortInt{}, Lbl'UndsPlus'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) |functional{}()] // functional axion(R) \exists(R) (Val:SortString{), \equals{SortString{}, R} (Val:SortString{}, Lbl'UndsPlus'String'UndsUnds'STRING-COWWON'Unds'String'Unds'String{}, K1:SortString{}))) [f axiom(R} \exists(R} (Val:SortAExp{}, \equals(SortAExp{}, R) (Val:SortAExp{}, Lbl'UndsPlusUndsUnds'IMP-SYNTAK'Unds'AExp'Unds'AExp'Unds'AExp{}(K8:SortAExp{}, K1:SortAExp{}))) [functional{}()] // functional axion{}\inplies{SortAExp{}} (\and{SortAExp{}} (Lbl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}(X0:SortAExp{}, X1:SortAExp{}), Lbl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp axlom(]\not{SortAExp{}} (\and{SortAExp}) (Lbl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}, X1:SortAExp{}, Lbl'Unds'-'UndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp axion{\not{SortAExp{}} (\and{SortAExp{}} (\bl'UndsPlusUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp[}(X0:SortAExp{}, X1:SortAExp{}), Lbl'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp[}(X0:SortAExp{}), Lbl'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp[}(X0:SortAExp{}), Lbl'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp[}(X0:SortAExp{}), Lbl'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AE axion(R) \exists(R) (Val;SortIds(), \equals(SortIds(), R) (Val;SortIds(), Lbl'UndsCormUndsUnds'IMP-SYNTAX'Unds'Ids'Unds'Id'Unds'Ids()(K0;SortId(), K1:SortIds()))) [functional()()] // functional axion{}\inplies{SortIds{}} (\and{SortIds{}} (Lbl'UndsCommUndsUnds'IMP-SYNTAX'Unds'Ids'Unds'Ids'Ids{}(X0:SortId{}, X1:SortIds{}), Lbl'UndsCommUndsUnds'IMP-SYNTAX'Unds'Ids'[Y0:SortId \exists{R} (Val:SortInt{}, \equals{SortInt{}, R} (Val:SortInt{}, Lbl'Unds'-Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional axion(R} \exists(R} (Val;SortMap{}, keuals(SortMap{}, R} (Val;SortMap{}, Lbl'Unds'-Map'Unds'MAp'Unds'Map'Unds'Map{Unds'Map{}, K1;SortMap{}))) [functional{}()] // functional axiom{R} \exists{R} (Val:SortAExp{}, \equals{SortAExp{}, R} (Val:SortAExp{}, R) (Val:SortAExp{}, Lbl'Unds'-'UndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}(K0:SortAExp{}, K1:SortAExp{}))) [functional{}()] // functional axion{\\inplies{SortAExp{}} (\and{SortAExp{}} (Lbl'Unds'-'UndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp{\X8:SortAExp{}, X1:SortAExp{}, Lbl'Unds'-'UndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp{} axiom(\\not{SortAExp{}} (\and{SortAExp{}} (Lb1'Unds'-'UndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp[)(X0:SortAExp{}, X1:SortAExp{}), Lb1'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp'Unds'AExp axiom{R} \exists{R} (Val:SortAExp{}, \equals{SortAExp{}, R} (Val:SortAExp{}, Lbl'UndsSlshUndsUnds'IMP-SYNTAX'Unds'AExp'Unds'AExp'Unds'AExp{}(KB:SortAExp{}, K1:SortAExp{}))) [functional{}()] // functional \existsfR} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'Unds-LT-Eqls'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional A semantics which has 16 lines. axion()\inplies{Sort8Exp{}} (\andfSort8Exp{}) (Lb1'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'8Exp'Unds'AExp{\K0:SortAExp{}, X1:SortAExp{}), Lb1'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'8Exp'Unds'AExp axlom{}\not{SortBExp{}} (\and{SortBExp{}} (Lbl'Unds-LT-EqlsUndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp[\K0:SortAExp{}, X1:SortAExp{}), Lbl'Unds-LT-UndsUnds'IMP-SYNTAX'Unds'BExp'Unds'AExp axum(b) (exist(v) (visionized)) (vala(strettep()), b) (visionized)), b) (visionized) (visionized) (visionized) (visionized) (visionized)), b) (visionized)), visionized)), visioni axiom(R \exists(R (Val:SortBool{}, lequals(SortBool{}, R) (Val:SortBool{}, Lbl'UndsEqlsSlshEqls'Bool'Unds'{}(K8:SortBool{}, K1:SortBool{}))) [functional{}()] // functional \exists{R} (Val:SortBool{}, \equals(SortBool{}, R) (Val:SortBool{}, Lbl'UndsEqlsSlshEqls'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEglsSlshEgls'K'Unds'{}(K0:SortK{}, K1:SortK{}))) [functional{}()] // functional \vexists(R) (Val:SortBool(), \vexists(R) (Val:SortBool(), R) (Val:SortBool(), Lbl'UndsEglsSlshEgls'String'UndsUnds'STRING-COWMON'Unds'String'Unds'String'Unds'String'(K0:SortString()))) (file) :SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEqlsEqls'Bool'Unds'{}(K0:SortBool{}, K1:SortBool{})) [functional{}()] // functional \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEqlsEqls'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEqlsEqls'K'Unds'{}(K0:SortK{}, K1:SortK{}))) [functional{}()] // functional \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'UndsEqlsEqls'String'UndsUnds'STRING-COMMON'Unds'Bool'Unds'String'Unds'String{}(K8:SortString{}, K1:SortString{}))) [funct: \exists{R} (Val:SortStnt{}, kequals[SortStnt{}, R] (Val:SortStnt{}, Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stnt'Unds'Id'Unds'AExp{}(K8:SortAExp{}))) [functional{}()] // functional axlom{\\inpl\es{SortStmt{}} (\and{SortStmt{}} (Lbl'UndsEalsUndsSClnUnds'IMP-SYNTAX'Unds'Stmt'Unds'Id'Unds'AExof}(X0:SortIdf). X1:SortAExof)). Lbl'UndsEalsUndsSClnUnds'IMP-SYNTAX'Unds'Id'Unds'Id'Unds'AExof} axion{}\not{SortStmt{}} (\and{SortStmt{}} (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stnt'Unds'Id'Unds'AExp{}(X0:SortId{}, X1:SortAExp{}), Lbl'UndsUndsUnds'IMP-SYNTAX'Unds'Stnt'Unds'Stnt{}(Y0:SintAExp{})) axlom()\not{SortStmt(}} (\and{SortStmt(}) (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stmt'Unds'AExp()(X0:SortId(), X1:SortAExp()), Lbltf'Unds'then'Unds'else'UndsUnds'IMP-SYNTAX'Unds axion{\\not{SortStmt{}} (\and{SortStmt{}} (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stmt'Unds'Id'Unds'AExp{}(X0:SortId{}, X1:SortAExp{}), Lblwhile'Unds'do'UndsUnds'IMP-SYNTAX'Unds'BExp'Unds'ISt axion{\\not{SortStmt}} (\and{SortStmt}} (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stmt'Unds'Id'Unds'AExp{}(X0:SortId{}, X1:SortAExp{}), Lbl'LBraUndsRBraUnds'IMP-SYNTAX'Unds'Stmt{}(Y0:SortStm axlom()\not{SortStmt(}} (\and{SortStmt(}) (Lbl'UndsEqlsUndsSClnUnds'IMP-SYNTAX'Unds'Stmt'Unds'Id'Unds'AExp()(X0:SortId(), X1:SortAExp()), Lbl'LBraRBraUnds'IMP-SYNTAX'Unds'Stmt{}))) [constructor{}()] // axion{R} \exists{R} (Val:SortBool{}, \equals{SortBool{}, R} (Val:SortBool{}, Lbl'Unds-GT-Eqls'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional axion(R) \exists(R) (Val:SortBool(). \equals(SortBool(). R) (Val:SortBool(). Lbl'Unds-GT-Eqls'String'UndsUnds'STRING-COMMON'Unds'Bool'Unds'String'Unds'String()(K0:SortBtring(), K1:SortBtring()))) [funct axion{R} \exists{R} (Val:SortBool{), \equals{SortBool{}, R} (Val:SortBool{}, Lbl'Unds-GT-'Int'Unds'{}(K0:SortInt{}, K1:SortInt{}))) [functional{}()] // functional axiom(R) \exists(R) (Val:SortBool{}, \equals(SortBool{}, R) (Val:SortBool{}, Lbl'Unds-GT-'String'UndsUnds'STRING-COWMON'Unds'Bool'Unds'String'Unds'String{}(K8:SortString{}, K1:SortString{}))) [functiona 

Amélie LEDEIN

WG3 meeting - Timisoara











• Result: See https://hal.science/hal-03895834/

• Challenge: Gap between AML and  $\mu$ -ML

### Overview of DEDUKTI

• Logical framework based on  $\lambda\Pi$ -CALCULUS MODULO THEORY  $\rightarrow \lambda$ -calculus + dependent types + rewriting rules

• Example of an encoding with natural numbers:

```
constant symbol Nat : TYPE;
constant symbol 0 : Nat;
constant symbol S : Nat \rightarrow Nat;
symbol + : Nat \rightarrow Nat \rightarrow Nat;
rule $m + 0 \hookrightarrow $m;
rule $m + (S $n) \hookrightarrow S ($m + $n);
symbol add_10 : \Pi (_ : Nat), Nat := \lambda x, x + 10;
```













- Results
  - Implementation of a transformation  $^1$  from CTRS to TRS
  - Automatic translator from  $\mathbb K$  to  $\operatorname{Dedukti}$  to execute small programs
- Challenges
  - Scale up! (Threads, non-determinism, etc.)
  - Prove the preservation of confluence and termination

<sup>&</sup>lt;sup>1</sup>Inspired from Patrick Viry proposal



## Translation from $\mathbb K$ to $\operatorname{Dedukti}$ - Verification



### Translation from $\mathbb K$ to $\operatorname{Dedukti}$ - Verification



## Translation from $\mathbb K$ to $\operatorname{Dedukti}$ - Verification



Amélie LEDEIN

# A specific logic to recheck concrete executions

10/12

#### Results

- Partial shallow encoding of MATCHING LOGIC into DEDUKTI.
- Specific MATCHING LOGIC fragment to recheck concrete KPROVER executions.
- METAMATH translator to DEDUKTI.
- Challenges
  - Symbolic execution + Circular coinduction
  - Interoperability between the MATCHING LOGIC encodings in COQ, LEAN, METAMATH and DEDUKTI

# Conclusion and further works

- To recap, translators from:
  - $\mathbb{K}$  to Dedukti
  - Metamath to Dedukti
- To investigate:
  - Recheck symbolic KPROVER executions
  - Interoperability between the various MATCHING LOGIC encodings

