University of Leicester

Mining State-Based Models from Proof Corpora

Thomas Gransden Neil Walkinshaw Rajeev Raman

Department of Computer Science
University of Leicester
tg75@student.le.ac.uk

CICM 2014, Coimbra - Portugal
MKM Track

Motivation

Interactive Theorem Proving

The process of interacting with a computer to complete proofs.

User completes proof by entering a sequence of *tactics*.

Same task for novice and experienced users.

How to proceed?

When manual intuition hasn't led to a proof:

- Automated tactics - `auto`, `firstorder`, `tauto`...
- Outsource to ATPs (`why3` in Coq, `sledgehammer` in Isabelle).
- Utilise existing proofs.
    - `search`, `searchAbout`, `searchPattern`.
    - ML4PG (by Heras and Komendantskaya)

Existing Proofs

Examples where user has entered a correct sequence of tactics.

The Coq Proof Assistant

The Coq Standard Library

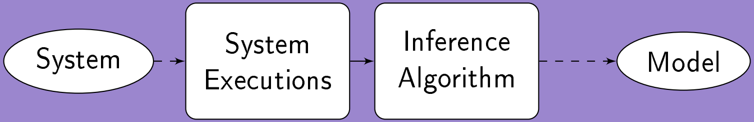
Here is a short description of the Coq standard library, which is distributed with the system. It provides a set of modules directly available through the `Require Import` command.

The standard library is composed of the following subdirectories:

Init: The core library (automatically loaded when starting Coq)
Notations Datatypes Logic Logic_Type Peano Specif Tactics Wf (Prelude)
Logic: Classical logic and dependent equality
SetIsType Classical_Pred_Set Classical_Pred_Type Classical_Prop Classical_Type (Classical) ClassicalFacts Decidable Eqdep_dec EqdepFacts Eqdep JMeq ChoiceFacts RelationalChoice ClassicalChoice ClassicalDescription ClassicalEpsilon ClassicalUniqueChoice Berardi Diaconescu Hurkens ProofIrrelevance ProofIrrelevanceFacts ConstructiveEpsilon Description Epsilon IndefiniteDescription FunctionalExtensionality ExtensionalityFacts
Structures: Algebraic structures (types with equality, with order, ...). DecidableType* and OrderedType* are there only for compatibility.
Equalities EqualitiesFacts Orders OrdersTac OrdersAlt OrdersEx OrdersFacts OrdersLists GenericMinMax DecidableType DecidableTypeEx OrderedType OrderedTypeAlt OrderedTypeEx
Bool: Booleans (basic functions and results)
Bool BoolEq DecBool IfProp Sumbool Zerob Bvector
Arith: Basic Peano arithmetic
Arith_base Le Lt Plus Minus Mult Gt Between Peano_dec Compare_dec (Arith) Min Max Compare Div2 EqNat Euclid Even Bool_nat Factorial Wf_nat
PArith: Binary positive integers
BinPosDef BinPos Pnat POrderedType (PArith)
NArith: Binary natural numbers
BinNatDef BinNat Nnat Ndigits Ndist Ndec Ndiv_def Ngcd_def Nsqrt_def (NArith)

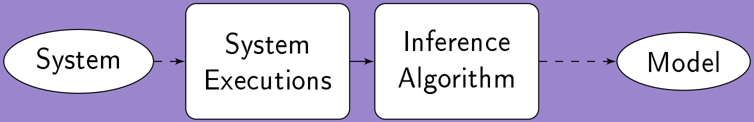
Model Inference

Software Engineering

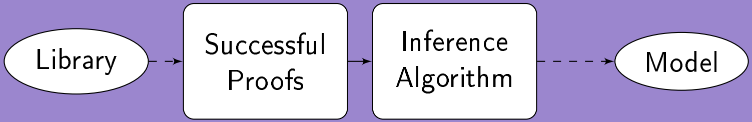


Model Inference

Software Engineering

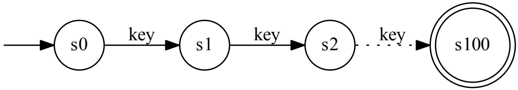


Theorem Proving



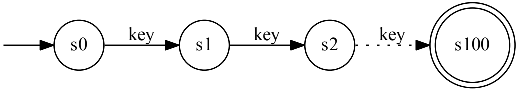
FSMs and EFSMs Example

Finite State Machine

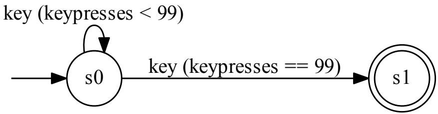


FSMs and EFSMs Example

Finite State Machine



Extended Finite State Machine



Modelling Proofs with State Machines - 1

Given the following examples:

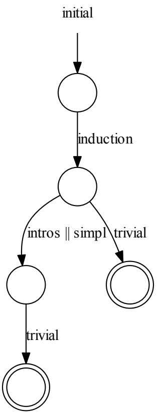
```
induction n. simpl. trivial.
induction a. intros. trivial.
induction l. trivial.
induction m. trivial.
induction n. trivial.
induction l. simpl. trivial.
```

Modelling Proofs with State Machines - 1

Remove the parameters:

induction. simpl. trivial.
induction. intros. trivial.
induction. trivial.
induction. trivial.
induction. trivial.
induction. simpl. trivial.

Inferred FSM

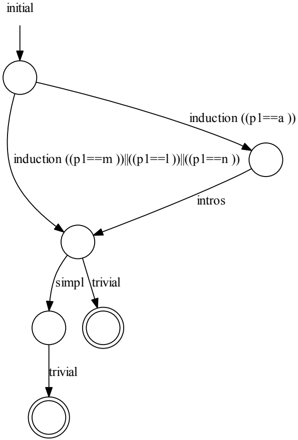


Modelling Proofs with State Machines - 2

Given the following examples:

induction n. simpl. trivial.
induction a. intros. trivial.
induction l. trivial.
induction m. trivial.
induction n. trivial.
induction l. simpl. trivial.

Inferred EFSM



Evaluation Process - Accuracy

Sensitivity

Proportion of times a model correctly accept a valid sequence of tactics.

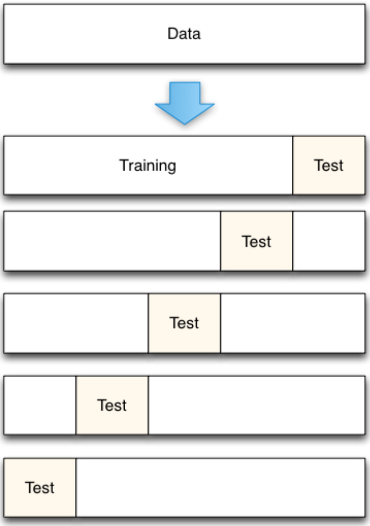
Specificity

Proportion of times a model correctly rejects an invalid sequences of tactics.

Negative examples generated by:

- Randomizing valid tactic sequences
- Using proofs from different theories to the dataset

Evaluation Process - k-folds cross validation



Results

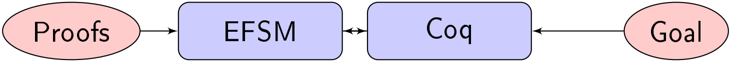
Data Set	Proofs	Sensitivity	Specificity
ListNat	70	0.84	0.81
Bool	100	0.95	0.55
Coqlib	100	0.22	0.96
Values	85	0.24	0.98

Qualitative Value of Models

Can an inferred model be useful in proof development?

- Provides a visual interpretation of proofs
- Manually inspect the model.
- Automated application of EFSMs

Automated Application



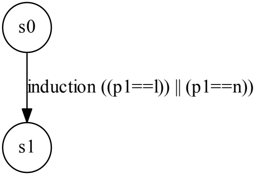
Proof attempt is made by search through inferred EFSM.

Preliminary Results from Automated Application

Data Set	EFSM Success
ListNat	67%
Bool	30%
ConstructiveGeometry	35%
RegExp	25%
Float	48%

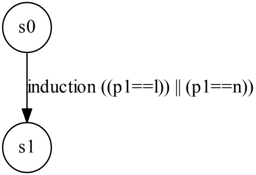
Abstraction of labels

Without Types

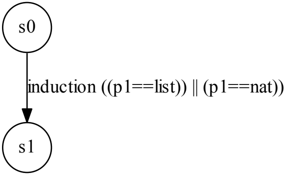


Abstraction of labels

Without Types



With Types



Incorporate Negative Informations

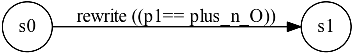
Currently, models are inferred from successful examples.

During a proof attempt, there may be a lot of negative examples - failed derivations.

Can we include this information in the model?

Filtering of Existing Proofs

Combine our tool with clustering tool ML4PG (Heras and Komendantskaya).



If this fails, try lemmas in the same cluster as plus_n_O

Conclusions

We have shown that:

- Model Inference can be applied to theorem proving
- Inferred models can be useful in proof development
- Many ways in which we can improve the models