Formalizing Cut Elimination of Coalgebraic Logics in Coq

Hendrik Tews

Technische Universität Dresden

Tableaux, September 17, 2013

Summary

- in Coq, formalize ⅔ of

Cut Elimination in Coalgebraic Logics

Dirk Pattinson*, Dept. of Computing, Imperial College London

Lutz Schröder†, DFKI Bremen
and Dept. of Comput. Sci., Univ. Bremen

Abstract

We give two generic proofs for cut elimination in propositional modal logics, interpreted over coalgebras. We first investigate semantic coherence conditions between the axiomatisation of a particular logic and its coalgebraic semantics that guarantee that the cut-rule is admissi-

- formalisation of syntax, semantics and 2 cut-elimination theorems for (generic) propositional multi-modal logic
- **K** as example, (work in progress on coalition logic)
- revealed only 4 errors (which were easy to correct)
- see `http://askra.de/science/coalgebraic-cut`

Motivation

Verified Cut Elimination

- Cut elimination is an important meta property of a logic
- . . . but is tricky to prove
- . . . and proofs are rarely ever spelled out

Generic Nature of Coalgebraic Modal Logics

- results apply to every logic that fits into the framework
- formalising the preconditions suffices
  to obtain formalised soundness, completeness and cut-elimination results

This work is the basis for

- certified validity checkers extracted from the completeness proof

Cut Elimination

Semantic: Given a proof for Γ

- soundness shows validity of Γ
- cut-free completeness shows the existence of a cut-free proof

Syntactic: Shift cut upwards, replacing, for instance,

$$\dfrac{(\neg\wedge)\,\dfrac{\vdash \neg A, \neg B, C}{\vdash \neg(A \wedge B), C} \qquad \dfrac{\vdash A \qquad \vdash B}{\vdash A \wedge B}\,(\wedge)}{\vdash C}\,(\text{cut})$$

by

$$(\text{cut})\,\dfrac{\dfrac{\vdash \neg A, \neg B, C \qquad \vdash A}{\vdash \neg B, A} \qquad \vdash B}{\vdash C}\,(\text{cut})$$

Outline

- Introduction
- Formalization in Coq
  - syntax
  - proofs
  - ~~semantics~~
- Selection of Major Results
- Some Interesting Bits
  - classical vs. intuitionistic logic
  - 1 of the 4 problems found during the formalisation
- Conclusion

Coalgebraic Modal Logics: Formulas

Multi-modal Propositional Modal Logic

- parametric on modal similarity type Λ
  which provides the set of modal operators and their arity
- formulas: p, $f \wedge g$, $\neg f$, $\heartsuit(f_1, \ldots, f_n)$
  for some set of propositional variables V, $p \in V$ and $\heartsuit$ of arity n

```
Record modal_operators : Type := { operator : Type; arity : operator → nat }.
Variable (V : Type) (L : modal_operators).

Inductive lambda_formula : Type :=
  | lf_prop : V → lambda_formula
  | lf_neg : lambda_formula → lambda_formula
  | lf_and : lambda_formula → lambda_formula → lambda_formula
  | lf_modal : forall(op : operator L),
      counted_list lambda_formula (arity L op) → lambda_formula.
```

- counted_list A n are lists over A of length n

Coalgebraic Modal Logics: Formulas

Multi-modal Propositional Modal Logic

- parametric on modal similarity type Λ
  which provides the set of modal operators and their arity
- formulas: p, $f \wedge g$, $\neg f$, $\heartsuit(f_1, \ldots, f_n)$
  for some set of propositional variables V, $p \in V$ and $\heartsuit$ of arity n

Record modal_operators : **Type** := { operator : **Type**; arity : operator → nat }.
Variable (V : **Type**) (L : modal_operators).

Inductive lambda_formula : **Type** :=
  | lf_prop : V → lambda_formula
  | lf_neg : lambda_formula → lambda_formula
  | lf_and : lambda_formula → lambda_formula → lambda_formula
  | lf_modal : **forall**(op : operator L),
      counted_list lambda_formula (arity L op) → lambda_formula.

- counted_list A n are lists over A of length n

Coalgebraic Modal Logics: Formulas

Multi-modal Propositional Modal Logic

- parametric on modal similarity type Λ
  which provides the set of modal operators and their arity
- formulas: p, $f \wedge g$, $\neg f$, $\heartsuit(f_1, \ldots, f_n)$
  for some set of propositional variables V, $p \in V$ and $\heartsuit$ of arity n

Record modal_operators : **Type** := { operator : **Type**; arity : operator → nat }.
Variable (V : **Type**) (L : modal_operators).

Inductive lambda_formula : **Type** :=
  | lf_prop : V → lambda_formula
  | lf_neg : lambda_formula → lambda_formula
  | lf_and : lambda_formula → lambda_formula → lambda_formula
  | lf_modal : **forall**(op : operator L),
      counted_list lambda_formula (arity L op) → lambda_formula.

- counted_list A n are lists over A of length n

Coalgebraic Modal Logics: Rules I

Fixed Propositional Rules

$$\frac{}{\vdash \Gamma, p, \neg p}\ (\mathrm{Ax}) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \wedge B}\ (\wedge) \qquad \frac{\vdash \Gamma, \neg A, \neg B}{\vdash \Gamma, \neg(A \wedge B)}\ (\neg\wedge)$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \neg\neg A}\ (\neg\neg) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, \neg A}{\vdash \Gamma, \Delta}\ (\mathrm{cut})$$

```
Definition sequent : Type := list lambda_formula. (* modulo reordering *)
Record sequent_rule : Type := {assumptions: list sequent; conclusion: sequent}.
```

Coalgebraic Modal Logics: Rules I

Fixed Propositional Rules

$$\frac{}{\vdash \Gamma, p, \neg p}\,(\mathrm{Ax}) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \wedge B}\,(\wedge) \qquad \frac{\vdash \Gamma, \neg A, \neg B}{\vdash \Gamma, \neg(A \wedge B)}\,(\neg\wedge)$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \neg\neg A}\,(\neg\neg) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, \neg A}{\vdash \Gamma, \Delta}\,(\mathrm{cut})$$

Definition sequent : **Type** := list lambda_formula. *(* modulo reordering *)*
Record sequent_rule : **Type** := {assumptions: list sequent; conclusion: sequent}.

Coalgebraic Modal Logics: Rules II

Logic Specific 1-Step Rules for Modalities

$$\frac{\vdash a_1^1, \ldots, \neg b_1^1, \ldots \qquad \cdots \qquad \vdash a_1^k, \ldots, \neg b_1^k, \ldots}{\vdash \heartsuit_1(\ldots), \ldots, \neg \heartsuit_1'(\ldots), \ldots}$$

Subject to Additional Conditions

- non-empty conclusion
- arguments for the modal operators in the conclusion are unnegated propositional variables
- all variables in the assumptions appear in the conclusion
- proofs may contain substitution instances of 1-step rules

Coalgebraic Modal Logics: Proofs

Proofs are finite trees build from rules and assumptions

```
Inductive proof(rules : set sequent_rule)(hypotheses : set sequent)
                                                  : sequent → Type :=
  | assume : forall(gamma : sequent),
      hypotheses gamma → proof rules hypotheses gamma
  | rule : forall(r : sequent_rule), rules r →
      dep_list sequent (proof rules hypotheses) (assumptions r) →
        proof rules hypotheses (conclusion r).
```

- proof R H G is the type of proof trees for sequent G using rules R and hypotheses H
- dep_list A T $[a_1; \ldots; a_n]$ is a inhomogeneous list of n elements where the i-th element has type T a_i
- very concise formalisation relying on dependent types

Coalgebraic Modal Logics: Proofs

Proofs are finite trees build from rules and assumptions

```
Inductive proof(rules : set sequent_rule)(hypotheses : set sequent)
                                          : sequent → Type :=
  | assume : forall(gamma : sequent),
      hypotheses gamma → proof rules hypotheses gamma
  | rule : forall(r : sequent_rule), rules r →
      dep_list sequent (proof rules hypotheses) (assumptions r) →
        proof rules hypotheses (conclusion r).
```

- proof R H G is the type of proof trees for sequent G using rules R and hypotheses H
- dep_list A T $[a_1; \ldots; a_n]$ is a inhomogeneous list of n elements where the i-th element has type T a_i
- very concise formalisation relying on dependent types

Outline

Introduction

Formalization in Coq

Selection of Major Results

Some Interesting Bits

Conclusion

Formalized Results

```
Variable T : functor.

Lemma cut_free_completeness :
  forall(enum_V : enumerator V)(LS : lambda_structure)
       (rules : set sequent_rule)(osr : one_step_rule_set rules)(s : sequent),
    classical_logic →
    non_trivial_functor T →
    one_step_cut_free_complete (enum_elem enum_V) LS rules osr →
    valid_all_models (enum_elem enum_V) LS s →
      provable (GR_set rules) empty_sequent_set s.
```

Formalized Results II

```
Variable op_eq : eq_type (operator L).
Variable v_eq : eq_type V.

Theorem syntactic_admissible_cut :
  forall(rules : set sequent_rule),
    countably_infinite V →
    one_step_rule_set rules →
    absorbs_congruence rules →
    absorbs_contraction op_eq v_eq rules →
    absorbs_cut op_eq v_eq rules →
      admissible_rule_set (GR_set rules) empty_sequent_set is_cut_rule.
```

Application to K

using the rule set $\dfrac{\vdash \neg p_1, \ldots, \neg p_n, p_0}{\vdash \neg\Box p_1, \ldots \neg\Box p_n, \Box p_0}$

Theorem k_semantic_cut :
  classical_logic →
    admissible_rule_set (GR_set k_rules) (empty_sequent_set VN KL) is_cut_rule.

Theorem k_syntactic_cut :
  admissible_rule_set (GR_set k_rules) (empty_sequent_set VN KL) is_cut_rule.

Lemma k_nd_equiv : **forall**(s : sequent VN KL),
  provable (GRC_set k_rules) (empty_sequent_set VN KL) s ↔
    provable (GRC_set is_k_n_rule) k_d_axioms s.

Outline

Classical vs. Intuitionistic Logic

Classical object logic of Pattinson & Schröder

- rules $\dfrac{}{\vdash \Gamma, p, \neg p}\,(\mathrm{Ax})$ and $\dfrac{\vdash \Gamma, A}{\vdash \Gamma, \neg\neg A}\,(\neg\neg)$
- defined disjunction: $A \vee B \quad \stackrel{\text{def}}{=} \quad \neg(\neg A \wedge \neg B)$

Coq's intuitionistic meta logic

- $A \vee \neg A$ is not a tautology, but $\neg(\neg A \wedge \neg\neg A)$ is
- $\neg\neg A \to A$ is not a tautology, but $A \to \neg\neg A$ is

Expect, that some results of Pattinson & Schröder are not provable in Coq

- making Coq classical: **Require** Classical.
- I prefer

  Definition classical_logic : **Prop** := **forall**(P : **Prop**), ¬ ¬ P → P.

The need for classical reasoning

. . . depends on disjunction and the semantic of sequents

- disjunction is syntactic sugar: $A \vee B \stackrel{\text{def}}{=} \neg(\neg A \wedge \neg B)$ in the object logic
- semantic of sequents ($[\![-]\!]_S$) is defined via the semantic of formulas ($[\![-]\!]_F$)

$$[\![\Gamma]\!]_S \stackrel{\text{def}}{=} [\![\bigvee \Gamma]\!]_F$$

$$[\![A, B]\!]_S \stackrel{\text{def}}{=} [\![A \vee B]\!]_F = [\![\neg(\neg A \wedge \neg B)]\!]_F$$

Double negation translation has surprising effects

- $\dfrac{}{\vdash \Gamma, p, \neg p}$ (Ax) is sound, because $\neg(\neg p \wedge \neg\neg p)$ is tautological
- $\dfrac{\vdash \Gamma, A \qquad \vdash \Delta, \neg A}{\vdash \Gamma, \Delta}$ (cut) is only sound when assuming classical_logic,

  because $A \wedge \neg(\neg B \wedge \neg\neg A) \rightarrow B$ is not a tautology

Substitution Lemma

Lemma (original substitution lemma)

Assume

- Γ *is provable with rules of modal rank* n *(i.e.,* Γ *has rank* n*)*
- σ *is a substitution that maps to formulas of modal rank* k

Then $\Gamma\sigma$ *is provable with rules of modal rank* $n + k$*,*
using the additional assumptions Ax_k*, where*

$$\mathrm{Ax}_k \stackrel{\text{def}}{=} \{\Gamma, A, \neg A \mid \Gamma \text{ and } A \text{ of modal rank } k\}$$

Proof.

Take the original proof, substituting $\neg p\sigma, p\sigma, \Gamma$ from Ax_k for $\dfrac{}{\vdash \Gamma, p, \neg p}$ (Ax)

□

Wrong Substitution Lemma

Lemma (original substitution lemma)

Assume

- Γ *is provable with rules of modal rank* n *(i.e.,* Γ *has rank* n*)*
- σ *is a substitution that maps to formulas of modal rank* k

Then $\Gamma\sigma$ *is provable with rules of modal rank* $n + k$*,*
using the additional assumptions Ax_k*, where*

$$\mathrm{Ax}_k \stackrel{\mathrm{def}}{=} \{\Gamma, A, \neg A \mid \Gamma \textit{ and } A \textit{ of modal rank } k\}$$

Example

- $\Gamma = \heartsuit(p), p, \neg p$ of modal rank $n = 1$, provable by (Ax)
- $\sigma : p \mapsto \heartsuit(p)$ of modal rank $k = 1$
- but $\Gamma\sigma = \heartsuit(\heartsuit(p)), \heartsuit(p), \neg\heartsuit(p)$ of rank $n + k = 2$
  is not in Ax_1

Substitution Lemma II

Error seems to break the main theorems

- subst. lemma is used inside induction proofs on the modal rank
- Γ of rank 1, σ of rank k
- reduces $\Gamma\sigma$ of rank $k+1$ to Ax_k of rank k
- thus permitting the use of the induction hypothesis

Use $\mathrm{Ax}_\sigma^{n+k} = \{\Gamma, p\sigma, \neg p\sigma \mid \Gamma \text{ of modal rank } n+k\}$

- "binding" of σ makes other proofs simpler
- need to use weakening before applying the induction hypothesis
- this way, original proofs remain valid

Substitution Lemma II

Error seems to break the main theorems

- subst. lemma is used inside induction proofs on the modal rank
- Γ of rank 1, σ of rank k
- reduces $\Gamma\sigma$ of rank $k+1$ to Ax_k of rank k
- thus permitting the use of the induction hypothesis

Use $\mathrm{Ax}_{\sigma}^{n+k} = \{\Gamma, p\sigma, \neg p\sigma \mid \Gamma \text{ of modal rank } n+k\}$

- "binding" of σ makes other proofs simpler
- need to use weakening before applying the induction hypothesis
- this way, original proofs remain valid

Outline

Conclusion I

Summary

- soundness, completeness, cut-elimination results
  for generic multi-modal propositional logic in Coq
- modal logic **K** as example
- very concise formalisation of syntax, semantics, proofs
  relying on dependent types (without predicates for well-formedness)
- only 4 non-trivial problems revealed (+1 for coalition logic)
- the usual peer-review process does not ensure correctness

Future Work

- coalition logic (work in progress) and other example logics
- remaining content of the paper,
  especially interpolation theorem and interpolants
- change formalisation to extract certified tautology checkers

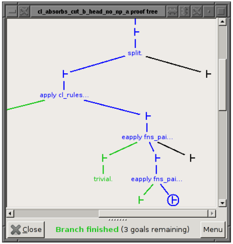
Conclusion II

Complexity

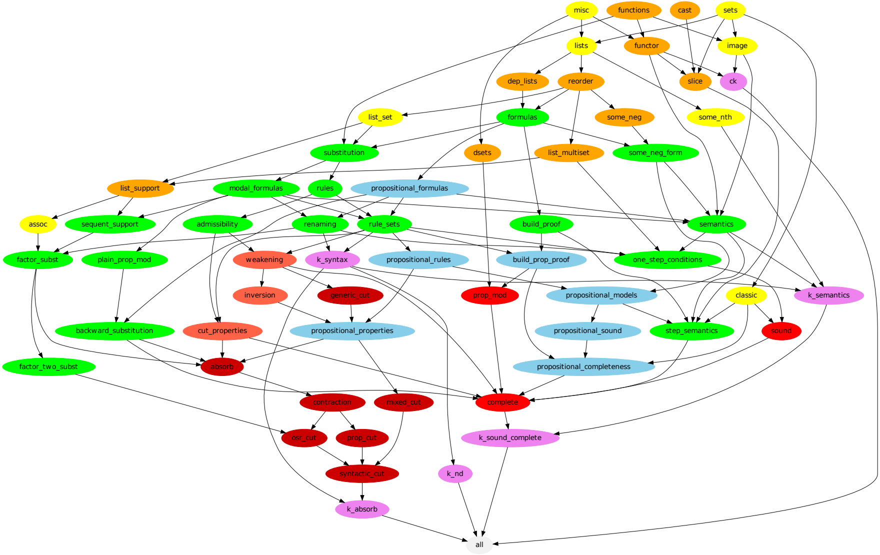
- 36,000 lines, 400 definitions, 1300 theorems in Coq
- for 19 propositions, 7 definitions, 3 examples on ≈ 31 pages

Side Effects

- parallel library compilation for Coq in Proof General
- proof tree visualisation



File Dependencies



Coalgebraic Modal Logics: Semantics

- a functor T describes the type of frames
- behaviour of modal operators is given by (fibred) predicate liftings: $[\![\heartsuit]\!] : \big((P_1 \subseteq X), \ldots, (P_n \subseteq X)\big) \mapsto (Q \subseteq TX)$
- a frame (model) is given by a coalgebra $\gamma : X \longrightarrow TX$ together with a valuation $\tau : V \longrightarrow \mathcal{P}(X)$
- formula semantics yields a subset of the state space $[\![-]\!]^c_\tau \subseteq X$:

$$\begin{array}{rcl} [\![p]\!]^c_\tau & = & \tau(p) \\ [\![A \wedge B]\!]^c_\tau & = & [\![A]\!]^c_\tau \cap [\![B]\!]^c_\tau \\ [\![\neg A]\!]^c_\tau & = & X \setminus [\![A]\!]^c_\tau \\ [\![\heartsuit(A_1, \ldots, A_n)]\!]^c_\tau & = & \gamma^{-1}\big([\![\heartsuit]\!]([\![A_1]\!]^c_\tau, \ldots, [\![A_n]\!]^c_\tau)\big) \end{array}$$