Classical Proof Forestry

Willem Heijltjes

LFCS
School of Informatics
University of Edinburgh

SD'09 Bordeaux

This talk...

- The ingredients of first-order classical proof
- Classical proof forests
- Cut
- Cut-elimination
- Normalisation (through pruning)

First-order classical proof

Herbrand's Theorem [1]

- Witnesses to the quantifiers
- Multiple instantiations for existential ones
- An ordering on instantiations (prenexification)

[1] See: [Buss: On Herbrand's Theorem (1995)]

First-order classical proof

Backtracking games [2]

- Backtracking for ∃loise (or verifier)
- Order of moves in a game

Sequent calculus

- Contraction
- Eigenvariable restriction on universal quantifier introduction

[2] See: [Coquand: A Semantics of Evidence for Classical Arithmetic (1995)]

First-order classical proof

The necessary ingredients

- Witness assignment
- Multiple witnesses to existential quantifiers
- An ordering to reflect available information

Important: a partial order suffices.

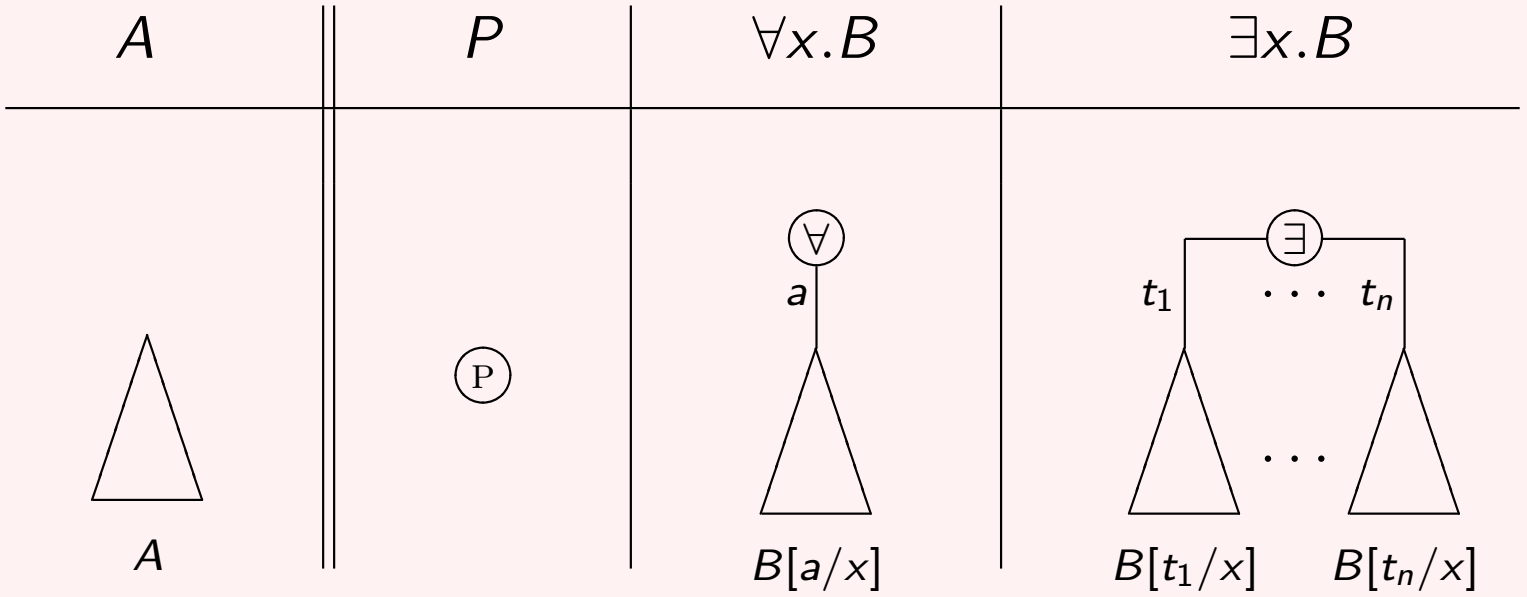
Dale Miller's expansion tree proofs[3]

In this presentation we will look at cut-elimination for this system.

[3] [Miller: A Compact Representation of Proofs (1987)]

Classical proof forests

An assignment tree for a prenex formula A is built as follows:

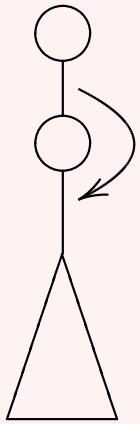


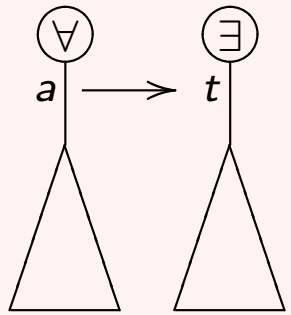
where P is propositional and $n \geq 0$; a is an eigenvariable.

Classical proof forests

A classical proof forest proves a sequent $A_1, \ldots, A_n$.

It consists of n trees plus a dependency ordering:





$(a \in FV(t))$

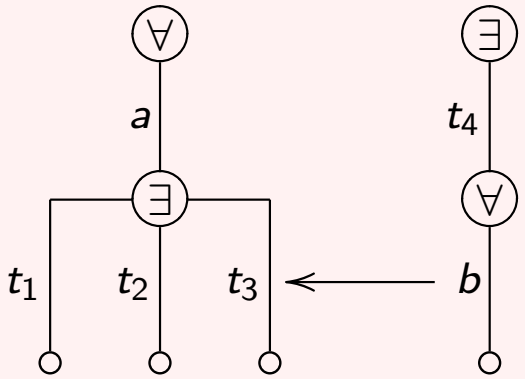
Correctness and validity criteria are:

- eigenvariables are unique,
- the dependency is a partial order, and
- the disjunction over the propositional leaves is a tautology.

Classical proof forests

Games interpretation

A forest gives a strategy for ∃loise in a backtracking game:



- The edges of the trees represent moves.
- The dependency gives the prerequisites of each move.

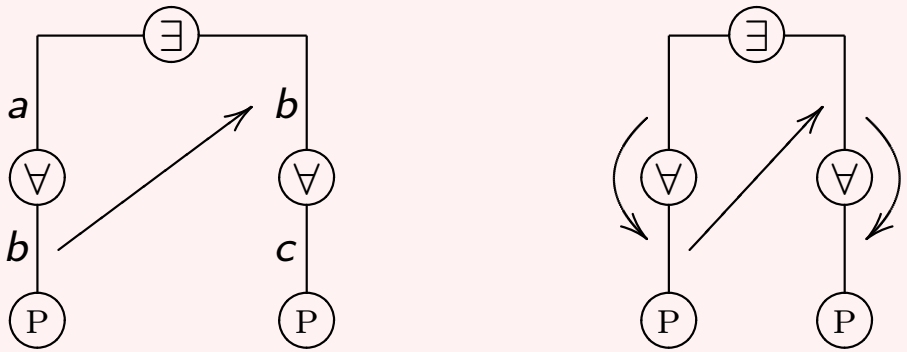
The edges plus dependency are an event structure (later: conflict).

Classical proof forests

Example: the drinker's formula

("There is a man in a bar, and if he drinks, everyone drinks.")

$$\exists x \forall y.\ (Px \to Py)$$

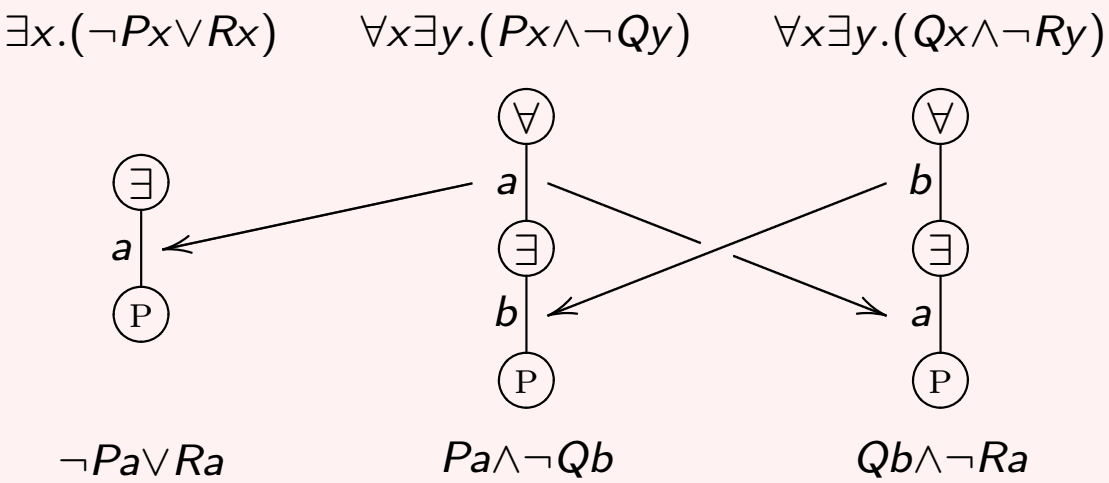


$Pa \to Pb$ $\qquad$ $Pb \to Pc$

On the right: the isolated dependency of the proof.

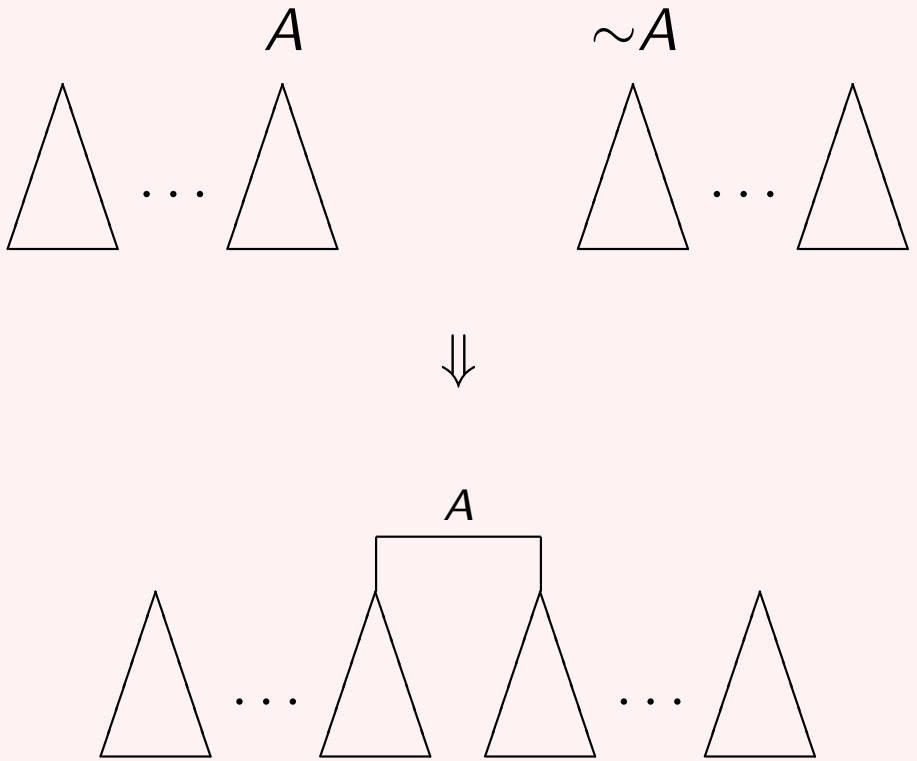
Classical proof forests

Another example.



Cut

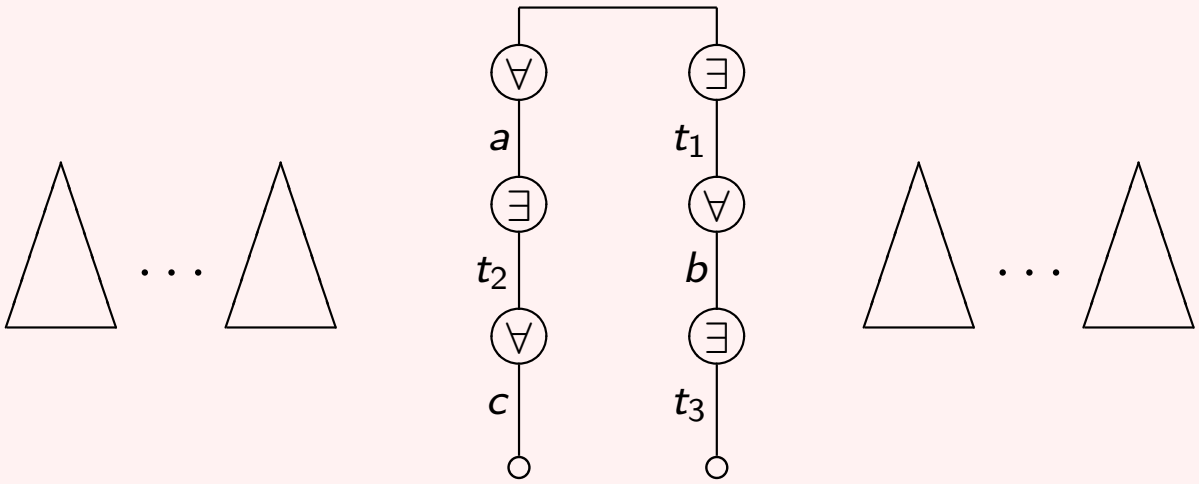
Composition



Cut

Interpretation (1)

Cut as communicating strategies: ∀belard mimicks ∃loise's moves on the other side.



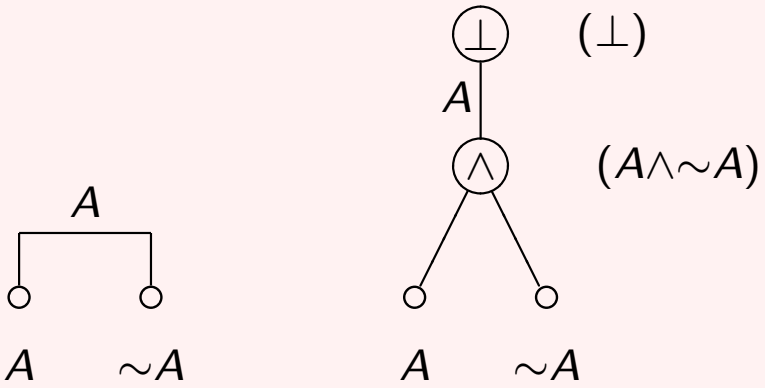
This gives an approach to normalisation.

But what about backtracking?

Cut

Interpretation (2)

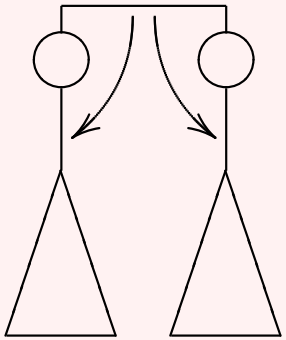
Cut as a combination of moves:

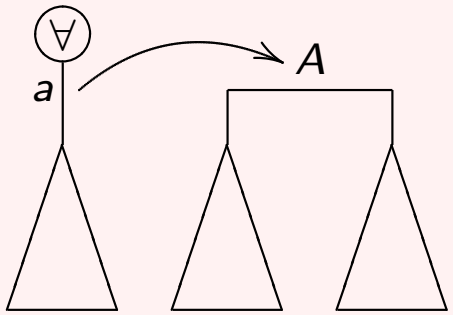


- Conjunctions can be modelled as a choice by ∀belard
- if A contains eigenvariables, ∃loise's move must participate in the dependency.

Cut

Cuts participate in the dependency in the following way:

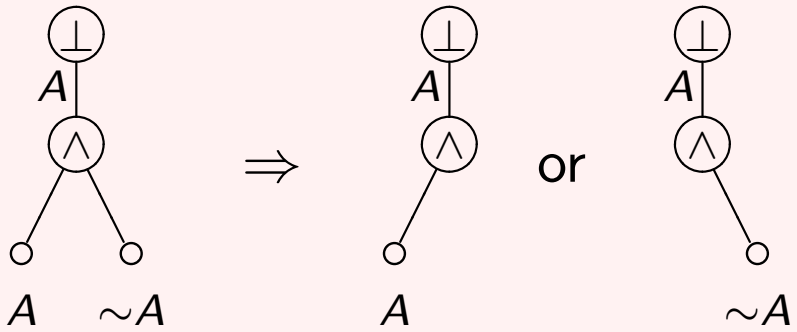




$(a \in FV(A))$

Cut

Validity



- A switching chooses one side of each cut.
  It represents a possible strategy for ∀belard on conjunctions.
- If ∀belard plays a branch, the other's dependants are removed.
- For any switching, this must be valid.
- Cut-free validity via a propositional tautology check.

Note this is different from Miller's validity criterion!

Cut

Conflict

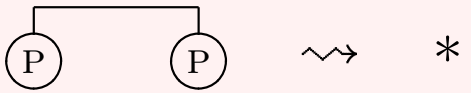
Another way of describing validity:

- Moves by ∀belard on a conjunction are in conflict (mutually exclusive in a single play).
- Any two moves depending on conflicting moves are in conflict.
- Any maximal conflict-free subforest must be winning.

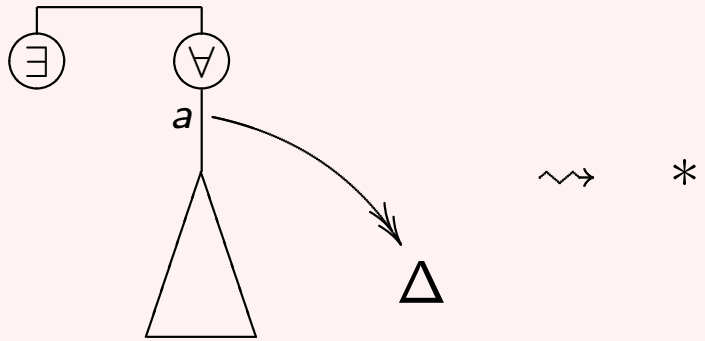
This gives the conflict relation of the event structure.

Reductions

A propositional step



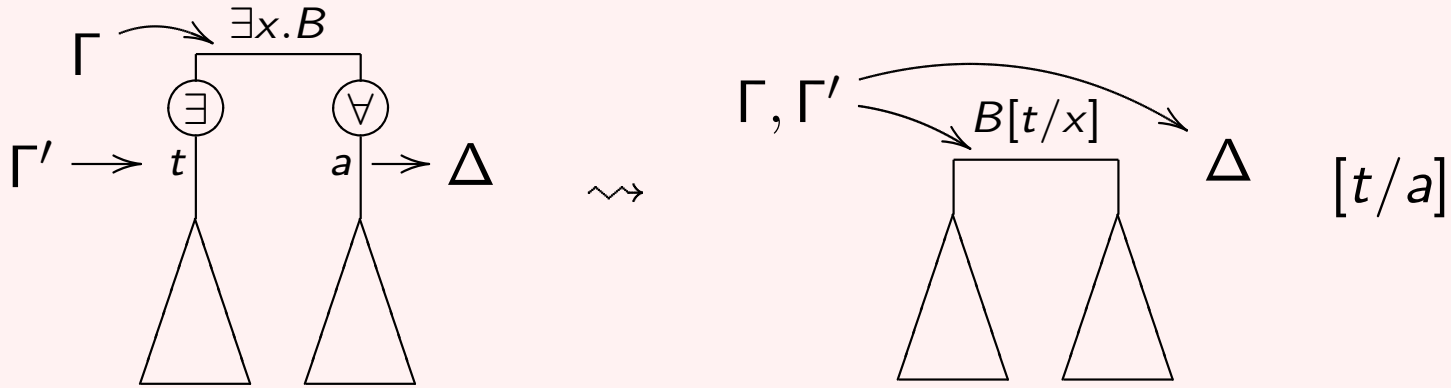
A disposal step



In this case also the dependants (Δ) are removed.

Reductions

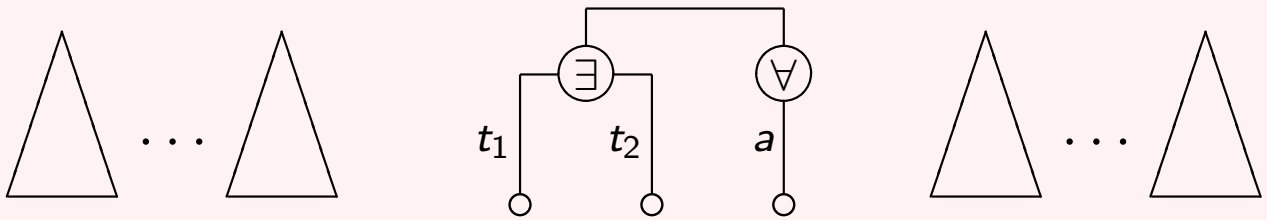
A logical step



- The eigenvariable a is substituted with the opposing term t.
- The dependency is updated accordingly.

Reductions

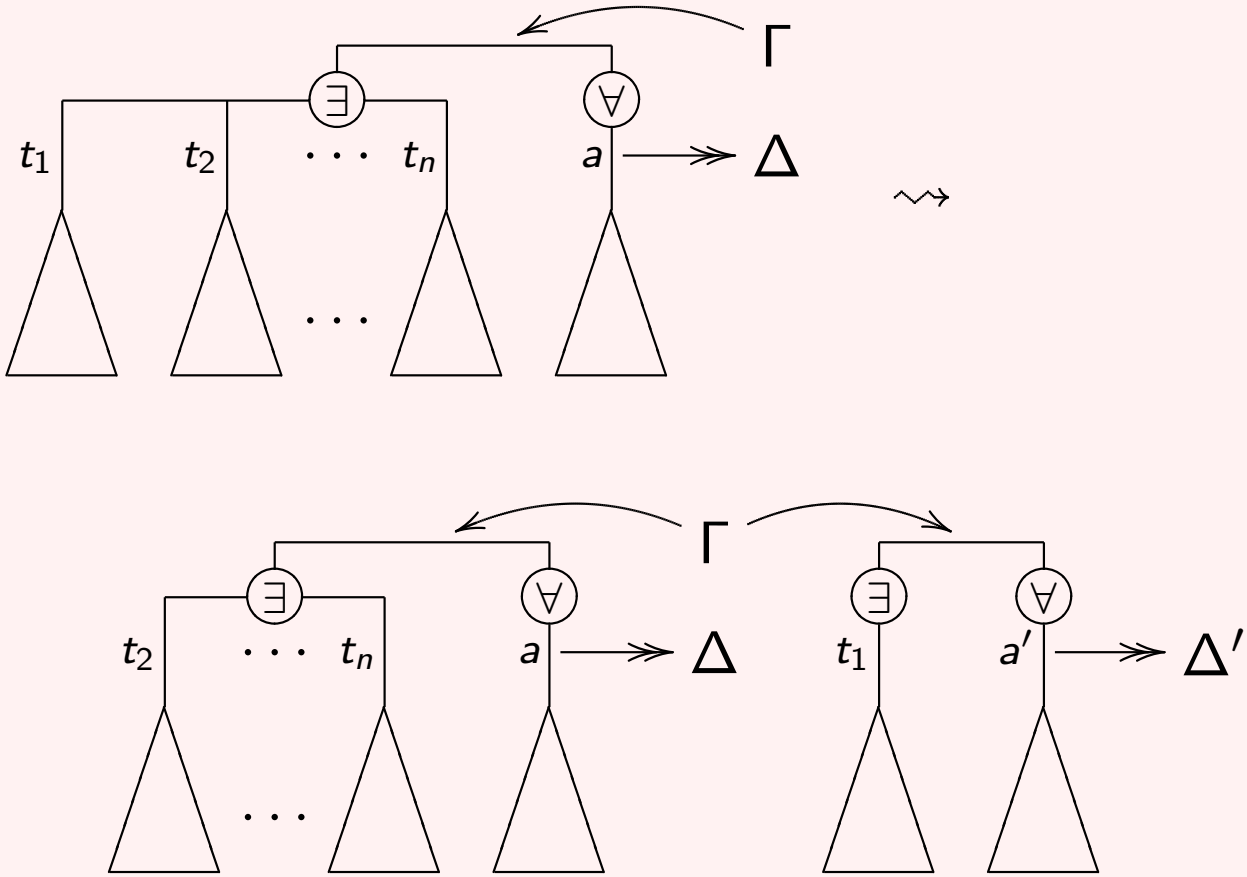
Backtracking



- Two ways to instantiate ∀belard's choice a.
- The moves responding to it are the dependants.
- These must be duplicated to accommodate both choices.

Reductions

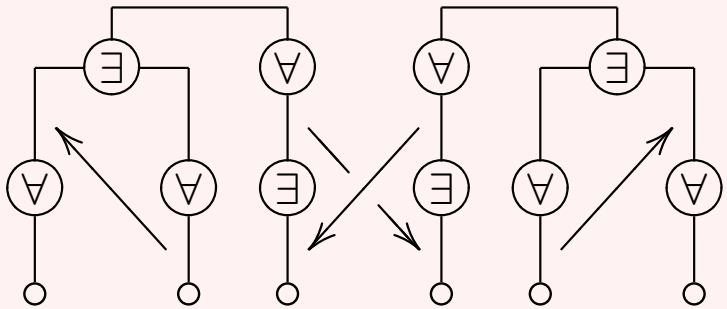
A structural step



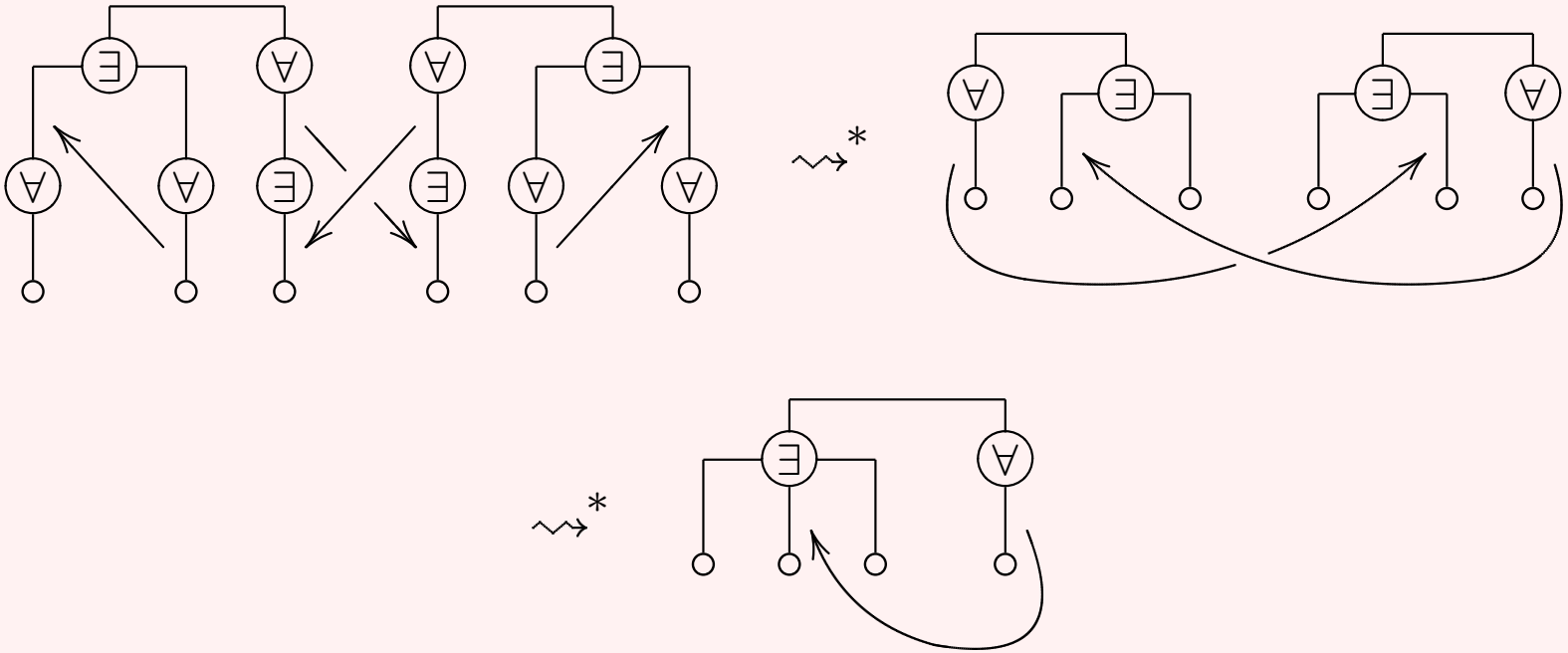
Reductions

Properties

- The reduction relation preserves validity.
- It is weakly normalising.
- It is non-confluent.
- Naively, the reduction relation exhibits non-termination.

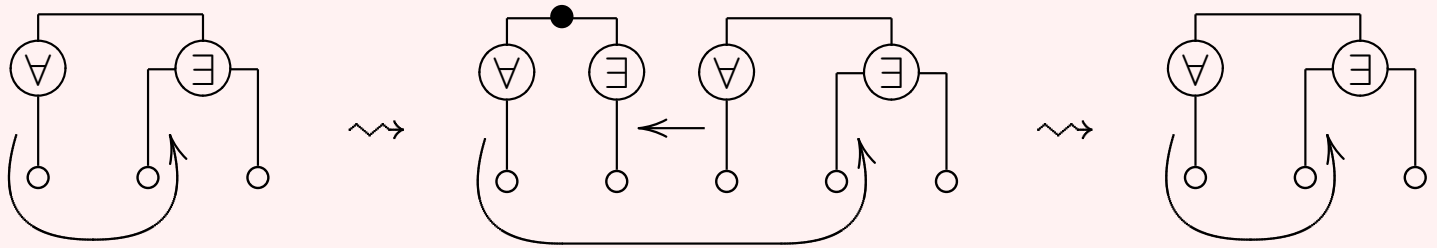


Non-termination



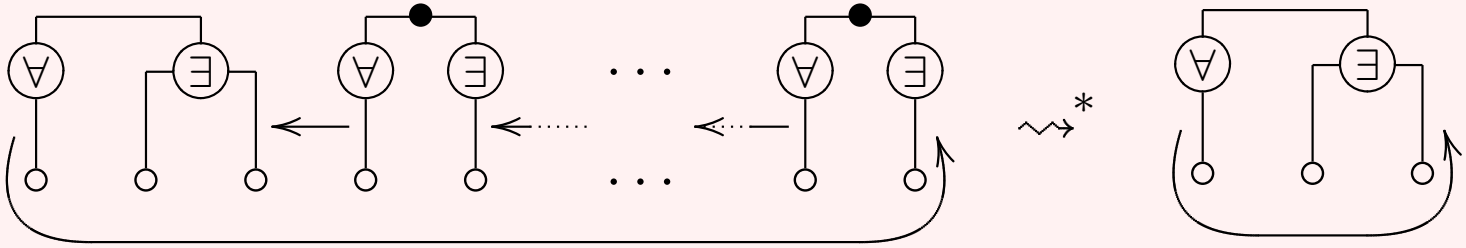
Non-termination

A dependency ‘crossing’ a cut generates infinite reductions.

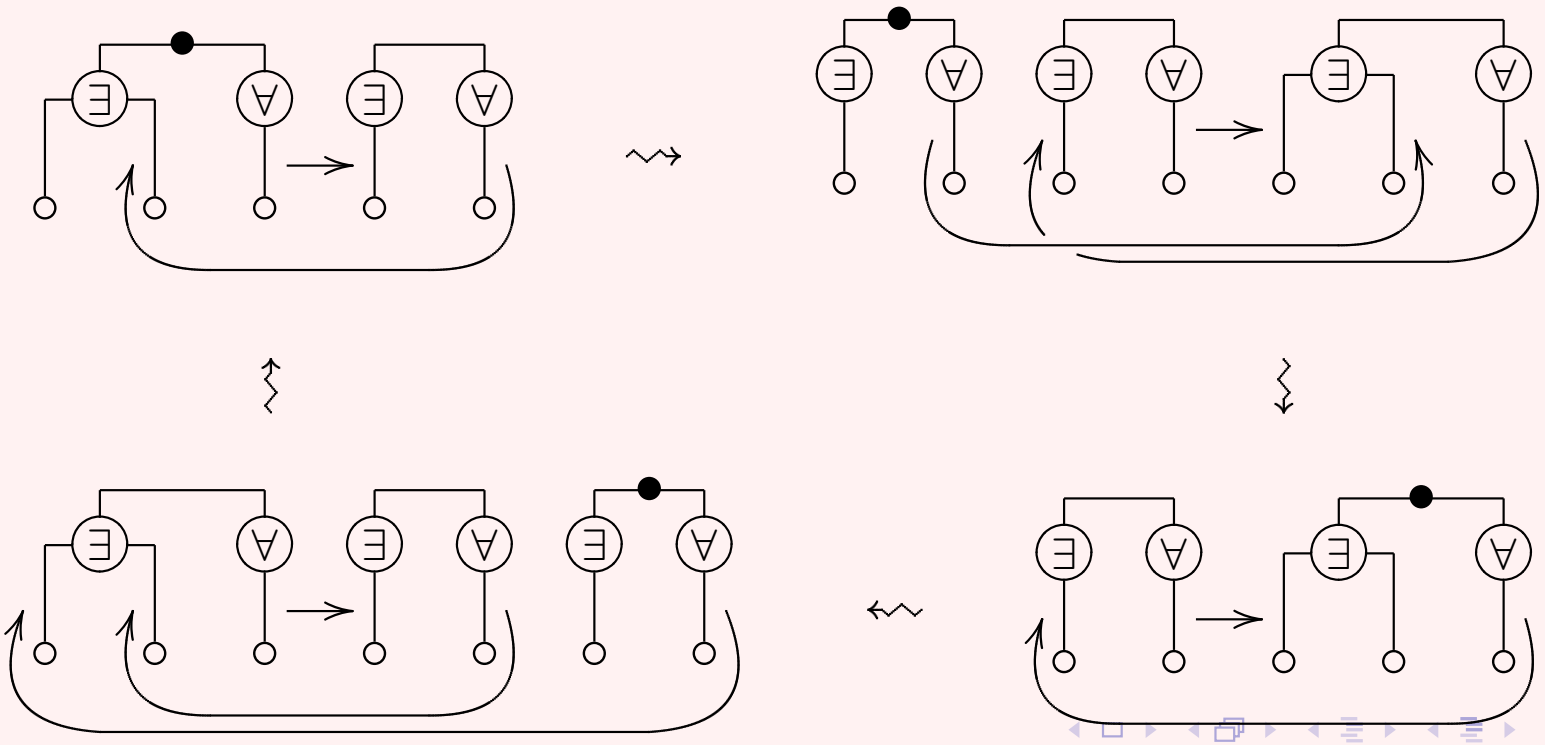


This example has no normalizing reduction paths.

A ‘cycle of cuts’ can reduce to a dependency across a cut. . .

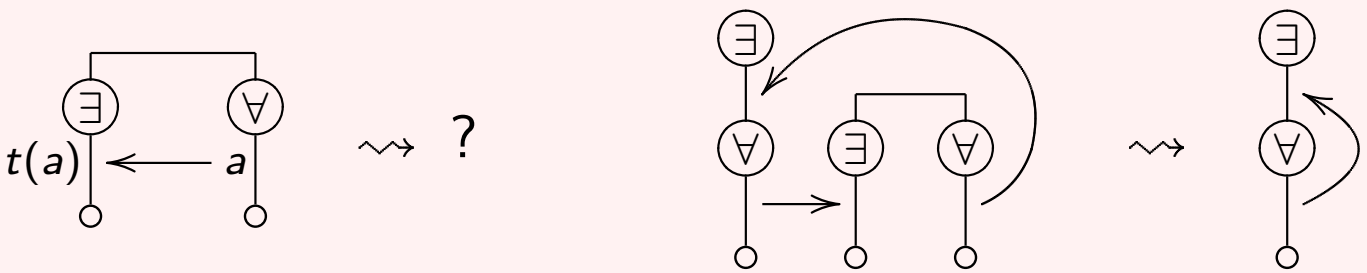


. . . or it can pass branches around indefinitely (example by McKinley).



Non-termination

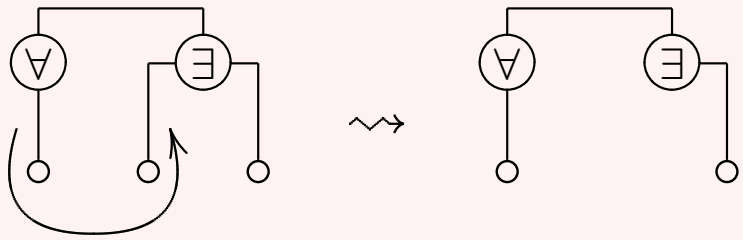
The following constructions do not arise from the example,



but their occurrence would break the reduction mechanism.

Reductions

Pruning



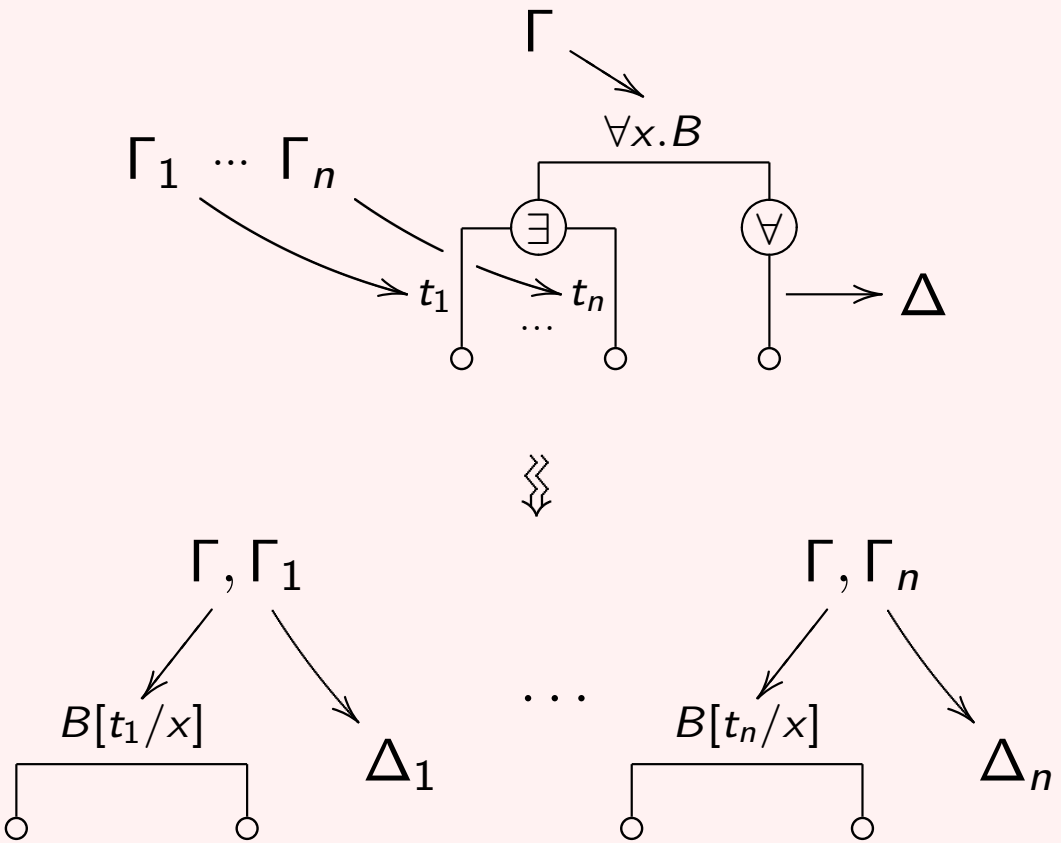
- The offending branch depends on both sides of the cut.
- In any switching, one side will be removed.
- In other words, the branch is in conflict with itself.

Pruning is the removal of such branches.

Reductions

Compound reduction steps

- The structural steps on a single cut,
- plus the logical steps on the resulting cuts.



Compound steps are unique and finite (without self-conflict).

Normalisation

Weak normalisation

The complexity of the cut-formula provides a measure.

- A compound step replaces a cut with a finite number of lower complexity.
- A step duplicates only cuts lower in the dependency.
- If all dependent cuts are of lower complexity a cut may be reduced.
- Antisymmetry of the dependency guarantees existence of a suitable cut.

Normalisation

Strong normalisation?

- The previously non-terminating example is now strongly normalising.
- The proof of weak normalisation allows reduction of ‘most’ cuts.
- The measure used is very ‘simple’.

This gives decent hope for strong normalisation.

Future work

- A strong normalisation proof
- More on non-confluence