The structure of the solution space in algorithmic cut-introduction

Daniel Weller
joint work with S. Hetzl, A. Leitsch and G. Reis

September 16, 2013

Table of Contents

Cut-Introduction (reminder)

- Aim: Shorten a proof by introducing a cut (i.e. a *lemma*).
- Method (previous talk):
  1. Represent a (large) set of terms by a (small) grammar.
  2. Compute an appropriate cut-formula.
- This talk: find a good cut-formula.

The problem

- There always exists an appropriate cut-formula: *canonical solution* C.
- C is optimal w.r.t. *quantifier-complexity* of generated proofs.
- In practice: Propositional part of proofs important as well.

Better solutions

- First approach: find a *small* cut-formula/solution for a given grammar.
- Towards this, we investigate the structure of the set of solutions.

Solutions (reminder)

Definition

Given a formula $F(x)$ and sets of terms U, S, a *solution* is formula $G(x)$ such that the sequent

$$\bigwedge_{u \in U} F(u), G(\alpha) \supset \bigwedge_{s \in S} G(s) \rightarrow$$

is valid (u may contain α).

- Induced by a proof of $\forall x F(x) \rightarrow$.
- U, S represents a grammar introducing one Π_1-cut.
- We may assume: $\bigwedge_{t \in T} F(t) \rightarrow$ valid for $T = \{u[\alpha \backslash s] \mid u \in U, s \in S\}$.

The structure of the solution space

- Since α is an eigenvariable/constant, the problem is purely propositional.
- We denote by $\models$ the propositional consequence relation.

The structure of the solution space

- Consider the boolean algebra $\mathcal{F}$ of formulas (modulo equivalence) and
- the set $\mathcal{S}$ of solutions.
- How is $\mathcal{S}$ situated in $\mathcal{F}$?

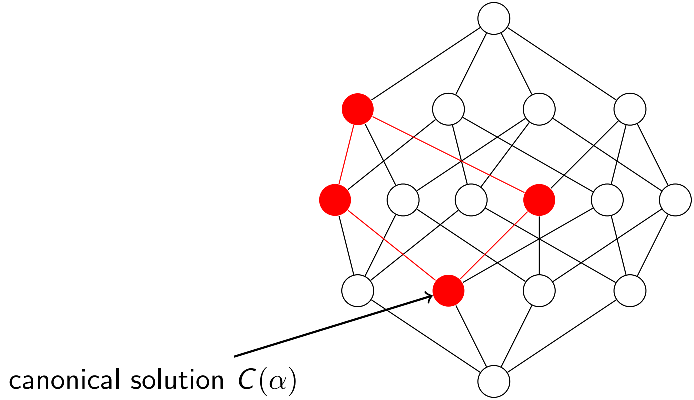
The structure of the solution space

Theorem

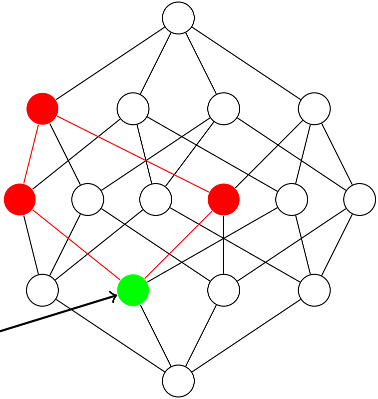
- *If C is the canonical solution and A an arbitrary solution then $C \models A$.*
- *If A, B are solutions and $A \models D \models B$, then D is a solution.*
- *If A, B are solutions then $A \circ B$ is a solution for $\circ \in \{\wedge, \vee\}$.*
- *If $A(x)$ is a solution in CNF and $A'(x)$ is obtained from $A(x)$ by removing all clauses that do not contain x, then A' is a solution.*

The structure of the solution space

- $\mathcal{S}$ is a bounded convex sublattice with $\perp = C(\alpha)$!

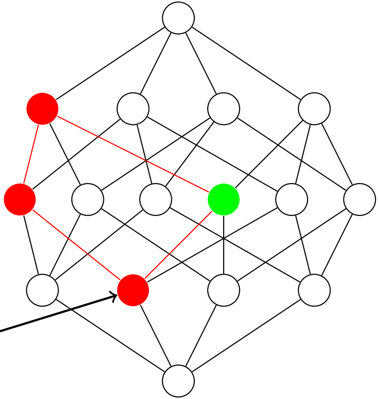


Looking for solutions



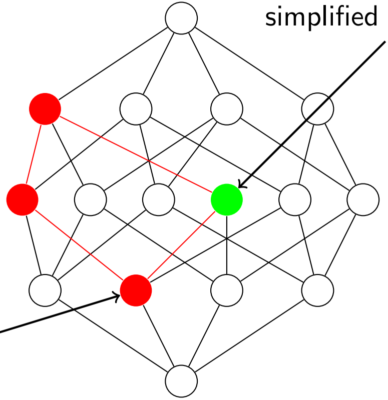
canonical solution $C(\alpha)$

Looking for solutions



canonical solution $C(\alpha)$

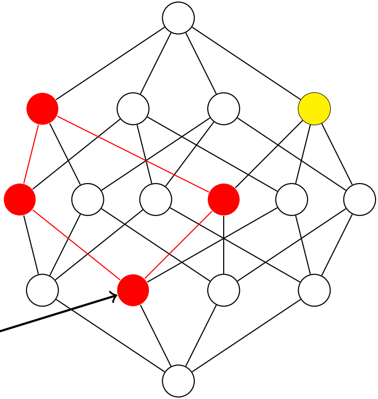
Looking for solutions



simplified TAUT-check!

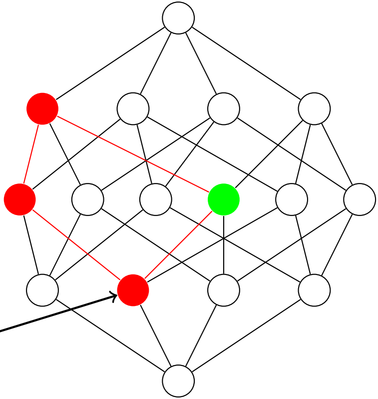
canonical solution $C(\alpha)$

Looking for solutions



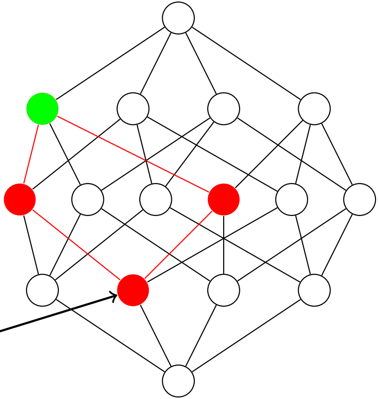
canonical solution $C(\alpha)$

Looking for solutions



canonical solution $C(\alpha)$

Looking for solutions



canonical solution $C(\alpha)$

Two concrete algorithms

- We search for solutions that are *implied* by other solutions.
- How do we look for consequences algorithmically?
- Two algorithms based on *resolution*, starting with the canonical solution.

Two concrete algorithms

- $\mathrm{SF}_{\mathcal{S}}$ is based on
  - computing the deductive closure $\mathcal{D}(\cdot)$ of a CNF via resolution and
  - checking subsets of this closure.

Theorem

$F \in \mathrm{SF}_{\mathcal{S}}(\mathcal{D}(C))$ for all minimal solutions F (in CNF), where C is the canonical solution in CNF.

Proof.

By a completeness theorem of (Lee 1967). □

Two concrete algorithms

- $\mathrm{SF}_{\mathcal{F}}$ is based on *forgetful resolution*.
- Forgetful resolution deletes the parent clauses of every resolution step.
- This is of course incomplete — but fast.
- Interestingly, this produces nice cut-formulas on some examples.[1]

[1] See G. Reis' talk after the coffee break.

Example (forgetful resolution)

Example

Consider

$$S : Pa, \forall x\, (Px \supset Pfx) \rightarrow Pf^4 a$$

with the grammar $\{\alpha, f\alpha\} \circ \{a, f^2 a\}$. The canonical solution is

$$C(x) : Pa \wedge (Px \supset Pfx) \wedge (Pfx \supset Pf^2 x) \wedge \neg Pf^4 a.$$

By deletion of x-free clauses we obtain

$$C'(x) : (Px \supset Pfx) \wedge (Pfx \supset Pf^2 x).$$

We have $\mathcal{F}(C'(x)) = \{Px \supset Pf^2 x\}$. It suffices to check whether

$$Pa, Pa \supset Pf^2 a, Pf^2 a \supset Pf^4 a \rightarrow Pf^4 a$$

is valid, which is the case. Search terminates since $\mathcal{F}(Px \supset Pf^2 x) = \emptyset$.

Going to multiple cuts

- To introduce n cuts, we have to find formulas $F_1, \ldots, F_n$.
- Finding optimal solutions seems to be very hard.
- But: it is possible to lift the algorithms from the 1-cut case by iteration (but optimality is not guaranteed).
- First, F_1 is generated by 1-CI,
- then F_2 is generated by 1-CI (from a problem based on F_1), ...

Table of Contents

An exponential speed-up

- Consider again the sequence from the previous talk:

$$S_n\colon Pa, \forall x(Px \supset Pfx) \rightarrow Pf^{2^{n+1}}a.$$

- Its shortest cut-free proofs admit a grammar

$$G_n\colon \{\alpha_1, f\alpha_1\} \circ_{\alpha_1} \{\alpha_2, f^2\alpha_2\} \circ_{\alpha_2} \cdots \circ \{\alpha_n, f^{2^{n-1}}\alpha_n\} \circ_{\alpha_n} \{a, f^{2^n}a\}.$$

inducing n cuts.

Going to multiple cuts

- Both algorithms generate the solution

$$[X_1 \backslash \lambda x.Px \supset Pf^2 x, \ldots, X_n \backslash \lambda x.Px \supset Pf^{2^n} x].$$

- From this solution, a proof with n cuts is constructed.
- This proof: linear vs. cut-free: exponential length!

Conclusion

- Several properties of the solution space help to guide search.
- Two concrete algorithms: one complete, one incomplete but faster.
- Even the incomplete one may yield an exponential compression.
- Future work: Empirical comparison of the algorithms.