On the

Generation of Proofs

from the

Clausal Normal Form Transformation

Hans de Nivelle
Max Planck Institut für Informatik
Saarbrücken Germany

Background Information

We want to generate proof objects from a resolution based theorem prover.

A resolution prover:

- Transforms a first-order formula into clausal normal form.
- Applies resolution/paramodulation on the resulting clauses until it derives a contradiction, or until it reaches a fixed point.

Example:

- Formula $\forall x \exists y \; P(x, y) \wedge \exists z \forall t \; \neg P(z, t)$ is transformed into clauses $\{P(x, f(x))\}, \; \{\neg P(c, y)\}$.
- The clauses resolve into a contradiction through the instances $\{P(c, f(c))\}, \; \{\neg \; P(c, f(y))\}$.

Generating Proof Objects

We want to generate the proof objects in Type Theory, (COQ) because it is well-defined and easy to check.

At present, we can generate proof terms for the second part $C \rightarrow \perp$ from our theorem prover Bliksem. Still missing are the proofterms for $F \rightarrow C$.

There are 2 approaches:

- Verify the CNF-transformer.

$$\mathrm{Correct}(\Phi) \wedge \Phi(F) = C.$$

- Let the CNF-transformer generate proof objects.

1st approach has the advantage, that once the check is complete, there is no remaining computational cost.

2nd approach has the advantage that you can use partial algorithms, and that you can compute directly on the meta logic.

Overview of the Talk

- Technical problems with generation of type theory proofs.
- Solution: The Replacement Calculus.
- How to handle Skolemization in a unified way.
- Conclusions.

Technical Problems with Proofs

Problem: Naively constructed proofs become too BIG.

Example: Transformation to NNF.

$\neg\neg P \Rightarrow P$
$\neg(P \vee Q) \Rightarrow \neg P \wedge \neg Q,$
$\neg(P \wedge Q) \Rightarrow \neg P \vee \neg Q,$
$P \rightarrow Q \Rightarrow \neg P \vee Q.$
(some more rules omitted)

Obvious approach: Implement the algorithm the way you would do otherwise, but have it construct a proof by using a justifying axiom for each replacement rule, and a context axiom for each logical operator.

Axioms needed for NNF-transformation

$$\begin{array}{ll}
\text{tnd}: & \Pi P{:}\,\mathrm{Prop}\ \neg\neg P \to P \\
\text{orneg}: & \Pi P, Q{:}\,\mathrm{Prop}\ (\neg(P \vee Q)) \leftrightarrow \neg P \wedge \neg Q \\
\text{andneg}: & \Pi P, Q{:}\,\mathrm{Prop}\ (\neg(P \wedge Q)) \leftrightarrow \neg P \vee \neg Q \\
\text{arrow}: & \Pi P, Q{:}\,\mathrm{Prop}\ (P \to Q) \leftrightarrow (\neg P \vee Q)
\end{array}$$

$$\begin{array}{ll}
\text{inand}: & \Pi P_1, P_2, Q_1, Q_2{:}\,\mathrm{Prop}\ (P_1 \leftrightarrow P_2) \to (Q_1 \leftrightarrow Q_2) \to \\
& \qquad P_1 \wedge Q_1 \leftrightarrow P_2 \wedge Q_2 \\
\text{inor}: & \Pi P_1, P_2, Q_1, Q_2{:}\,\mathrm{Prop}\ (P_1 \leftrightarrow P_2) \to (Q_1 \leftrightarrow Q_2) \to \\
& \qquad P_1 \vee Q_1 \leftrightarrow P_2 \vee Q_2
\end{array}$$

$$\begin{array}{ll}
\text{refl}: & \Pi P{:}\,\mathrm{Prop}\ \ P \leftrightarrow P \\
\text{trans}: & \Pi P, Q, R{:}\,\mathrm{Prop}\ \ (P \leftrightarrow Q) \to (Q \leftrightarrow R) \to (P \leftrightarrow R)
\end{array}$$

- It is easily checked that this method results in a proof of size $O(n^3)$ (both in the input and in the result) That is too big for practical purposes.
- The cost is dominated by the cost of building up contexts. $O(n^2)$ in the size of the context.

  If $A_1 \leftrightarrow A_2$ has a proof of size c, then
  $A_1 \wedge B_1 \wedge \cdots \wedge B_n \leftrightarrow A_2 \wedge B_1 \wedge \cdots \wedge B_n$ receives a proof of size $O(n^2) + c$.
  (This is due to the fact that one has to sum up formula chains
  $A_1,\ A_1 \wedge B_1,\ A_1 \wedge B_1 \wedge B_2, \ldots, A_1 \wedge B_1 \wedge \cdots \wedge B_n$ and
  $A_2,\ A_2 \wedge B_1,\ A_2 \wedge B_1 \wedge B_2, \ldots, A_2 \wedge B_1 \wedge \cdots \wedge B_n$)
- One can try to obtain sharing of contexts, but then the order of replacements becomes important. That is no problem for NNF, but for some transformations, the order is hard to predict or control.

Solution: Replacement Calculus

We define a calculus for representing proofs with the following features:

- Repeated build up of same context can be normalized away. The programmer need not worry about the order of the replacements.
- Drops an order in proof size, (because formulas on which a rule is applied need not be mentioned)
- Can be translated into type theory in either quadratic time, or in linear time (when using definitions).

The calculus is intended for representing the intermediate proofs, not for the final result.

The replacement calculus proves equivalences of form $A \equiv B$. We try to be as general as possible, so concrete instances of $\equiv$ can be:

- $(\approx\ D)$.
  Equality on some domain D.

- $\leftrightarrow$.
  Logical equivalence in the meta logic.

- $\lambda p_1, p_2{:}\,\mathrm{form}\ (T\ p_1) \leftrightarrow (T\ p_2)$.
  Logical equivalence in some object logic.

- $\lambda P_1, P_2{:}\, D \to \mathrm{Prop}\ \ \Pi d{:}\, D\ (P_1\ d) \leftrightarrow (P_2\ d)$.
  Extensional equivalence of predicates on domain D.

Reflexivity:

$$\frac{}{A \equiv A}\ \text{refl}$$

Transitivity:

$$\frac{A \equiv B \qquad B \equiv C}{A \equiv C}\ \text{trans}$$

Application:

$$\frac{A_1 \equiv B_1 \qquad A_2 \equiv B_2}{A_1 \cdot A_2 \equiv B_1 \cdot B_2}\ \text{appl}$$

Abstraction:

$$\frac{(P\ x) \equiv (Q\ x)}{(\lambda x{:}\, X\ \ P) \equiv (\lambda x{:}\, X\ \ Q)}\ \text{abstr}$$

x has to be a fresh variable.

Axiom:

$$\frac{}{A \equiv B}\ \text{axiom,}$$

$(A \equiv B)$ has to be justified from the axioms.

Removal of reflexivity in contexts 1

$$\dfrac{\dfrac{}{A_1 \equiv A_1}\,\text{refl} \qquad \dfrac{}{A_2 \equiv A_2}\,\text{refl}}{A_1 \cdot A_2 \equiv A_1 \cdot A_2.}\,\text{appl}$$

reduces into

$$\dfrac{}{A_1 \cdot A_2 \equiv A_1 \cdot A_2}\,\text{refl}$$

Removal of reflexivity in contexts 2

$$\dfrac{\dfrac{}{A \equiv A}\ \text{refl}}{\lambda x{:}X\ \ A \equiv \lambda x{:}X\ \ A}\ \text{abstr}$$

reduces into

$$\dfrac{}{\lambda x{:}X\ \ A \equiv \lambda x{:}X\ \ A}\ \text{refl}$$

Removal of repeated context construction 1

$$\dfrac{\dfrac{A_1 \equiv B_1 \quad A_2 \equiv B_2}{A_1 \cdot A_2 \equiv B_1 \cdot B_2}\ \text{appl} \qquad \dfrac{B_1 \equiv C_1 \quad B_2 \equiv C_2}{B_1 \cdot B_2 \equiv C_1 \cdot C_2}\ \text{appl}}{A_1 \cdot A_2 \equiv C_1 \cdot C_2}\ \text{trans}$$

reduces into

$$\dfrac{\dfrac{A_1 \equiv B_1 \quad B_1 \equiv C_1}{A_1 \equiv C_1}\ \text{trans} \qquad \dfrac{A_2 \equiv B_2 \quad B_2 \equiv C_2}{A_2 \equiv C_2}\ \text{trans}}{A_1 \cdot A_2 \equiv C_1 \cdot C_2}\ \text{appl}$$

Removal of repeated context construction 2

$$\dfrac{\dfrac{A_1 \equiv A_2}{\lambda x{:}X\ \ A_1 \equiv \lambda x{:}X\ \ A_2}\ \text{abstr} \qquad \dfrac{A_2 \equiv A_3}{\lambda x{:}X\ \ A_2 \equiv \lambda x{:}X\ \ A_3}\ \text{abstr}}{\lambda x{:}X\ \ A_1 \equiv \lambda x{:}X\ \ A_3}\ \text{trans}$$

reduces into

$$\dfrac{\dfrac{A_1 \equiv A_2 \qquad A_2 \equiv A_3}{A_1 \equiv A_3}\ \text{trans}}{\lambda x{:}X\ \ A_1 \equiv \lambda x{:}X\ \ A_3}\ \text{abstr}$$

Removal of reflexivity after/before trans

$$\dfrac{A \equiv B \qquad \dfrac{}{B \equiv B}\,\text{refl}}{A \equiv B}\,\text{trans}$$

and

$$\dfrac{\dfrac{}{A \equiv A}\,\text{refl} \qquad A \equiv B}{}\,\text{trans}$$

both reduce into

$$A \equiv B$$

Making trans leftassociative

$$\dfrac{A \equiv B \qquad \dfrac{B \equiv C \qquad C \equiv D}{B \equiv D}\,\text{trans}}{A \equiv D.}\,\text{trans}$$

reduces into

$$\dfrac{\dfrac{A \equiv B \qquad B \equiv C}{A \equiv C}\,\text{trans} \qquad C \equiv D}{A \equiv D.}\,\text{trans}$$

- Every proof has a unique normal form, which has minimal size.
- Proofs can be translated into typetheory, resulting in a proof with size quadratic in the depth.
- Proofs be translated into a proof of linear size, by introducing definitions for subformulas. This is done by using definitions in the formula chains as follows:
  $\alpha_1 := A_1 \wedge B_1, \quad \alpha_2 := \alpha_1 \wedge B_2, \ldots, \alpha_n := \alpha_{n-1} \wedge B_n,$
  $\beta_1 := A_2 \wedge B_1, \quad \beta_2 := \beta_1 \wedge B_2, \ldots, \beta_n := \beta_{n-1} \wedge B_n.$
  Such proofs will be checked in linear time, because definitions are lazily expanded.

Generating Proofs for improved Skolemization

Standard Skolemization is the replacement of existential quantifiers by fresh function symbols. $F[\exists y\ P(\overline{x}, y)]$ is replaced by $F[P(\overline{x}, f(\overline{x}))\ ]$, where f is a new function symbol, the variables $\overline{x}$ are the free variables of P (without y).

There are two ways to transform a refutation of a Skolemized formula into a refutation of the original formula:

- Use a choice (or ϵ) axiom to prove the Skolemized formula from the original formula.
- Eliminate the Skolem function from the proof.

(1) is efficient but ugly. (2) is elegant but inefficent.

Whatever method is chosen, there is the problem of handling optimized and strong Skolemization.

Strong Skolemization can be applied on formulas of the following form:

$$F[\exists y\ [P_1(\overline{x}_1, y) \wedge P_2(\overline{x}_2, y) \wedge \cdots \wedge P_n(\overline{x}_n, y)]\ ],$$

where $\overline{x}_1 \subseteq \overline{x}_2 \subseteq \cdots \subseteq \overline{x}_n \subseteq \overline{x}$.

The result is:

$$F[\quad [\ \forall(\overline{x} \backslash \overline{x}_1)\ P_1(\overline{x}, f(\overline{x}))\ ] \quad \wedge \quad [\ \forall(\overline{x} \backslash \overline{x}_2)\ P_2(\overline{x}, f(\overline{x}))\ ] \quad \wedge \quad \cdots \quad \wedge \quad [\ \forall(\overline{x} \backslash \overline{x}_n)\ P_n(\overline{x}, f(\overline{x}))\ ] \quad].$$

Optimized Skolemization can be applied on formulas of the form:

$$F[\exists y\ P_1(\overline{x}, y) \wedge P_2(\overline{x}, y)] \quad \wedge \quad \exists y\ P_1(\overline{x}, y).$$

The result is $F[\ P_2(\overline{x}, f(\overline{x}))\ ]\ \wedge\ P_1(\overline{x}, f(\overline{x}))$.

Problem: What kind of choice functions does one need for this? How many? (or alternatively, how can one delete such Skolemizations from proofs?)

Solution: They can be reduced to standard Skolemization.

This is done through (yet) another type of Skolemization, called stratified Skolemization.

Stratified Skolemization is defined on formulas of the following form

$$\forall \overline{x}\ C_1(\overline{x}) \rightarrow \exists y[\ P_1(\overline{x}, y)\ ]\ \ \wedge$$
$$\forall \overline{x}\ C_2(\overline{x}) \rightarrow \exists y[\ P_1(\overline{x}, y) \wedge P_2(\overline{x}, y)\ ]\ \ \wedge$$
$$\dots$$
$$\forall \overline{x}\ C_n(\overline{x}) \rightarrow \exists y[\ P_1(\overline{x}, y) \wedge P_2(\overline{x}, y) \wedge \cdots \wedge P_n(\overline{x}, y)\ ],$$

in case that

$$\forall \overline{x}\ C_n(\overline{x}) \rightarrow C_{n-1}(\overline{x}),\ \ C_{n-1}(\overline{x}) \rightarrow C_{n-2}(\overline{x}),\ \cdots,\ C_2(\overline{x}) \rightarrow C_1(\overline{x}).$$

The result is:

$$\forall \overline{x}\ C_1(\overline{x}) \rightarrow P_1(\overline{x}, f(\overline{x}))\ \ \wedge$$
$$\forall \overline{x}\ C_2(\overline{x}) \rightarrow P_2(\overline{x}, f(\overline{x}))\ \ \wedge$$
$$\dots$$
$$\forall \overline{x}\ C_n(\overline{x}) \rightarrow P_n(\overline{x}, f(\overline{x})).$$

Theorem Stratified Skolemization can be reduced to standard Skolemization in first-order Logic.

precise formulation: For every formula F on which stratified Skolemization is applicable, there exists a formula G that can be constructed in linear time, that is logically equivalent to F, such that the ordinary Skolemization of G equals the stratified Skolemization of F.

Proof:

Take $G := \forall \overline{x} \; \exists y \; (C_1(\overline{x}) \rightarrow P_1(\overline{x}, y)) \wedge \cdots \wedge (C_n(\overline{x}) \rightarrow P_n(\overline{x}, y))$.

Theorem Both optimized and strong Skolemization can be reduced to stratified Skolemization.

Consequence No additional choice axioms are needed. (or alternatively: No new proof transformation techniques are needed)

Question Does stratified Skolemization have any usefulness on its own?

Conclusions

- We introduced techniques that make explicit proof generation for the CNF-transformation realistic. The techniques are general. They can be used also for other automated proof search procedures.
- We reduced the improved Skolemization techniques to standard Skolemization.
- Keep an eye on `http://www.mpi-sb.mpg.de/~bliksem` to see how implementation proceeds.