A Mechanized Proof Reconstruction for SCNP Termination

Alexander Krauss[*] Armin Heller

Institut für Informatik
Technische Universität München

1 Introduction

Ben-Amram and Codish described SCNP [2], a subclass of the size-change termination criterion SCT [8], which permits efficient certificate checking. Termination problems in this class have a global ranking function of a certain form, which can be found using SAT solving.

This note describes an automated proof reconstruction for this certificate scheme, implemented in the theorem prover Isabelle/HOL [9]. In previous work [6], we have shown how to use the full size-change principle for termination proofs of recursive function definitions in Isabelle. Although the certificate-based approach is less powerful in theory, it has practical advantages:

- The transitive closure computation in [8] is an efficiency bottleneck, and optimizing it is hard since the code must be proved correct and executed within the logical system.
- Certificates can be stored, which makes proof checking easier when the proof script is re-run.
- Much less logical infrastructure is necessary. In particular, no formalization of Ramsey's theorem is required, which makes the approach portable to theorem provers with a constructive foundation, such as Coq [3].

Our method is included in the recent release of Isabelle 2009 (`http://isabelle.in.tum.de`).

2 Termination Goals

Isabelle termination goals arise from recursive function definitions and express the wellfoundedness of the call relation extracted from the definition. The induction principle encoded in this relation is then used internally to construct an explicit model of the function.

Termination goals in Isabelle have the following form:

$$wf\ (\{(r_1, l_1) \mid \Gamma_1\} \cup \ldots \cup \{(r_n, l_n) \mid \Gamma_n\})$$

Here, each recursive call in the function definition is expressed as a relation comprehension $\{(r_i, l_i) \mid \Gamma_i\}$, where l_i is the argument from the left hand side of the definition, r_i is the argument of the recursive call, and Γ_i is the condition that leads to the recursive call. The goal is to prove that the union of these relations is wellfounded.[1]

Example. The function *perm* below is taken from [8]. The corresponding termination goal consists of two calls:

$$perm\ (m, n, r) = (\text{if } 0 < r \text{ then } perm\ (m, r - 1, n) \text{ else if } 0 < n \text{ then } perm\ (r, n - 1, m) \text{ else } m)$$

$$wf\ (\{((m, r - 1, n), (m, n, r)) \mid 0 < r\} \cup \{((r, n - 1, m), (m, n, r)) \mid \neg\ 0 < r \wedge 0 < n\})$$

This kind of proof obligation is typical for the tradition of modelling programs as functions in higher-order logic. In this shallow embedding, where the syntax of programs is not modelled explicitly, our setting differs from that of the CoLoR [4] and A3PAT [5] projects and the recent CeTA tool [10]. However, our proof reconstruction should be applicable in that context, too.

For this presentation, we ignore many issues that tend to complicate things, like mutual and nested recursion, and just focus on the reconstruction.

[*] `http://www.in.tum.de/~krauss`

[1] The wellfoundedness predicate *wf* is defined in such a way that the "smaller" element is written left, contrary to conventions in rewriting.

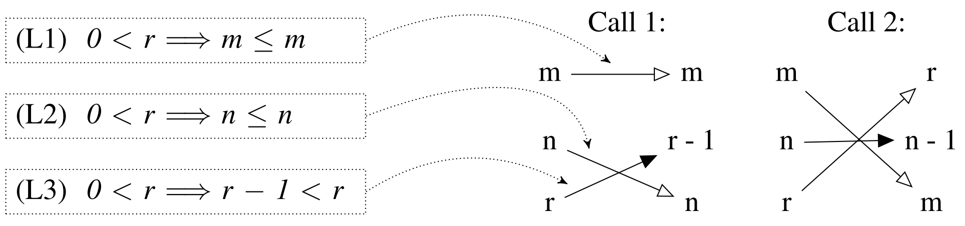
(L1) $0 < r \Longrightarrow m \leq m$
(L2) $0 < r \Longrightarrow n \leq n$
(L3) $0 < r \Longrightarrow r - 1 < r$
Call 1:
m
m
n
r - 1
r
n
Call 2:
m
r
n
n - 1
r
m

Figure 1: The size-change graphs generated for *perm* and the underlying theorems

3 SCNP = Size-Change Termination in NP

The SCNP criterion uses the abstraction known from size-change termination: A termination problem is represented as an abstract control-flow graph, whose edges carry information about the relative sizes of data in the transitions, depicted as size-change graphs. In Isabelle, this information is derived using off-the-shelf theorem proving methods, producing a theorem for each edge in a size-change graph. Figure 1 shows the size-change graphs generated for the function *perm* and the underlying theorems.

Instead of computing the transitive closure of the set of size-change graphs, the SCNP approach is based on orderings. It uses the following extensions of wellfounded relations to finite multisets:

Definition 1 (Max-, min-, and multiset extension).
$\textit{max-ext}\, R = \{(X, Y) \mid Y \neq \emptyset \wedge (\forall x{\in}X.\ \exists y{\in}Y.\ (x, y) \in R)\}$
$\textit{min-ext}\, R = \{(X, Y) \mid X \neq \emptyset \wedge (\forall y{\in}Y.\ \exists x{\in}X.\ (x, y) \in R)\}$
$\textit{ms-ext}\, R = \{(X, Y) \mid \exists a\, Z\, K.\ Y = Z \cup \{a\} \wedge X = Z \cup K \wedge (\forall x{\in}K.\ (x, a) \in R)\}^+$

Intuitively, *max-ext* and *min-ext* compare sets by their maximum or minimum, and *ms-ext* is the standard multiset order [1]. Ben-Amram and Codish also use a fourth extension called *dual multiset order*, but experiments show that it contributes very little to the power of the approach, and we decided not to include it. The extensions above are applied to the lexicographic order on pairs of natural numbers. Formally:

$$\begin{array}{lll} \textit{lex}_< \;=\; \{((a, b), (c, d)) \mid a < c \vee a \leq c \wedge b < d\}, & & \textit{lex}_\leq = \textit{lex}_< \cup \textit{Id}, \\ \textit{max}_< = \textit{max-ext}\ \textit{lex}_<, & \textit{min}_< = \textit{min-ext}\ \textit{lex}_<, & \textit{ms}_< = \textit{ms-ext}\ \textit{lex}_<, \\ \textit{max}_\leq = \textit{max-ext}\ \textit{lex}_\leq \cup \{(\emptyset, \emptyset)\}, & \textit{min}_\leq = \textit{min-ext}\ \textit{lex}_\leq \cup \{(\emptyset, \emptyset)\}, & \textit{ms}_\leq = \textit{ms}_< \cup \textit{Id}. \end{array}$$

Using our comprehension notation, a call $\{(r_i, l_i) \mid \Gamma_i\}$ is said to be *R-decreasing under a map f* if $\{(f\, r_i, f\, l_i) \mid \Gamma_i\} \subseteq R$ or, equivalently, $\Gamma_i \Longrightarrow (f\, r_i, f\, l_i) \in R$. We are looking for a map under which some calls are $\mu_<$-decreasing and the remaining calls are $\mu_\leq$-decreasing for some $\mu \in \{\textit{max}, \textit{min}, \textit{ms}\}$. Then the $\mu_<$-decreasing calls can be removed from the termination problem and the process is repeated (possibly with a different ordering) until no calls are left.

The SCNP approach uses maps which assign a multiset of pairs to each argument vector. The first component of the pairs is the size of some argument[2], and the second is a fixed natural number called a *tag*. A function of this kind is called a *level mapping*. There is no intuitive motivation for the tags, but they add an extra degree of freedom to the analysis, thus making it slightly more powerful in some cases.

Example (cont'd). The termination proof for the function *perm* above uses the level mapping $\lambda(m, n, r).\ \{(m, 0), (n, 0), (r, 0)\}$. Both calls are $\textit{ms}_<$-decreasing under that level mapping.

[2]Isabelle generates size measures heuristically based on the types. The size of a natural number is the number itself.

$max_<$:	(1)	$Y \neq \emptyset \Longrightarrow (\emptyset, Y) \in max_<$
	(2)	$y \in Y \Longrightarrow (x, y) \in lex_< \Longrightarrow (X, Y) \in max_< \Longrightarrow (\{x\} \cup X, Y) \in max_<$
$ms_<$:	(3)	$(Z, Z') \in pairwise_\leq \Longrightarrow (A, B) \in max_< \Longrightarrow (Z \cup A, Z' \cup B) \in ms_<$
$pairwise_\leq$:	(4)	$(\emptyset, \emptyset) \in pairwise_\leq$
	(5)	$(x, y) \in lex_\leq \Longrightarrow (X, Y) \in pairwise_\leq \Longrightarrow (\{x\} \cup X, \{y\} \cup Y) \in pairwise_\leq$
$lex_{</\leq}$:	(6)	$a < b \Longrightarrow ((a, s), (b, t)) \in lex_<$
	(7)	$a \leq b \Longrightarrow s < t \Longrightarrow ((a, s), (b, t)) \in lex_<$
	(8)	$a < b \Longrightarrow ((a, s), (b, t)) \in lex_\leq$
	(9)	$a \leq b \Longrightarrow s \leq t \Longrightarrow ((a, s), (b, t)) \in lex_\leq$

Figure 2: Introduction rules for proof reconstruction

Certificates A certificate is the justification for removing some calls from a termination problem. It contains the following information:

1. The ordering $\mu \in \{max, min, ms\}$,
2. the level mapping f, which maps the argument vectors to multisets,
3. the calls that are $\mu_<$-decreasing under f, and
4. a *covering function* for each call, which contains information required in the proof that the call is decreasing. If $\mu \in \{max, min\}$, the function provides the instantiations of the existential quantifiers in Def. 1. If $\mu = ms$, it provides the decomposition of the multisets used in rule (3) below.

To construct certificates, we use the SAT-encoding given by Ben-Amram and Codish [2]. By calling an external SAT solver, we obtain a satisfying assignment from which we obtain the information above.

4 Proof Reconstruction

The main task in the proof reconstruction is to formally derive the inclusions $\{(f\,r_i, f\,l_i) \mid \Gamma_i\} \subseteq \mu_{</\leq}$ that are postulated by the certificate. For this purpose we derive a set of simple facts about $max_{</\leq}$, $min_{</\leq}$, $ms_{</\leq}$, and $lex_{</\leq}$, which serve as introduction rules for the respective relations. Figure 2 shows some of the rules; the complete list can be found in [7]. The actual reconstruction consists of applying these rules in a controlled manner. To express intermediate subgoals, we must introduce the auxiliary relation $(X, Y) \in pairwise_\leq$ which expresses that there is a one-to-one correspondence between the respective elements of X and Y, such that corresponding elements $(x, y) \in lex_\leq$.

Proving that two given sets are $max_<$-decreasing amounts to showing that for each $x \in X$ there is a $y \in Y$ such that $x < y$ using rules (1) and (2). At each application of rule (2), the variable y must be instantiated with an element of Y, which is given by the covering function in the certificate. Multiset descent can be reduced to max-descent using rule (3), which decomposes the multisets into two parts. The decomposition is part of the certificate.

Example (cont'd). To illustrate the automated proof reconstruction, we show a detailed proof that the first call of *perm* is $ms_<$-decreasing. Thus we start with the following goal, which we will refine gradually:

$$0 < r \Longrightarrow (\{(m, 0), (r-1, 0), (n, 0)\}, \{(m, 0), (r, 0), (n, 0)\}) \in ms_<$$

We rewrite the multisets using associativity and commutativity to split them into two parts, as in rule (3):

$$0 < r \Longrightarrow (\{(m, 0), (n, 0)\} \cup \{(r-1, 0)\}, \{(m, 0), (n, 0)\} \cup \{(r, 0)\}) \in ms_<$$

After applying rule (3), we get two subgoals:

$$0 < r \Longrightarrow (\{(m, 0), (n, 0)\}, \{(m, 0), (n, 0)\}) \in pairwise_\leq$$

$$0 < r \Longrightarrow (\{(r-1, 0)\}, \{(r, 0)\}) \in max_<$$

We apply the introduction rules for $\mathit{pairwise}_{\leq}$ to the first subgoal, obtaining the following goal state:

$$0 < r \Longrightarrow ((m, 0), (m, 0)) \in \mathit{lex}_{\leq}$$

$$0 < r \Longrightarrow ((n, 0), (n, 0)) \in \mathit{lex}_{\leq}$$

$$0 < r \Longrightarrow (\{(r - 1, 0)\}, \{(r, 0)\}) \in \mathit{max}_{<}$$

Then we apply the introduction rules for $\mathit{lex}_{\leq}$:

$$0 < r \Longrightarrow m \leq m$$

$$0 < r \Longrightarrow 0 \leq 0$$

$$0 < r \Longrightarrow n \leq n$$

$$0 < r \Longrightarrow 0 \leq 0$$

$$0 < r \Longrightarrow (\{(r - 1, 0)\}, \{(r, 0)\}) \in \mathit{max}_{<}$$

Now, subgoals 1 and 3 are precisely the local descent properties (L1) and (L2) that we have already proved (cf. Fig. 1). Subgoals 2 and 4 are trivial inequalities between number literals (the tags) and are easily discharged. For the last subgoal, we apply rule (2), instantiating y with $(r, 0)$. This yields

$$0 < r \Longrightarrow ((r - 1, 0), (r, 0)) \in \mathit{lex}_{<}$$

$$0 < r \Longrightarrow (\emptyset, \{(r, 0)\}) \in \mathit{max}_{<}$$

Then, using rule (6), we can apply the local descent property (L3). Rule (1) solves the last subgoal. Fortunately, these tedious proofs are fully automated and invisible to the user.

Acknowledgements We thank Amir Ben-Amram and Michael Codish for answering all our questions on SCNP details.

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] A. Ben-Amram and M. Codish. A SAT-based approach to size change termination with global ranking functions. In C. R. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*, volume 4963 of *LNCS*, pages 218–232. Springer, March 2008.

[3] Y. Bertot and P. Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions.* Texts in theoretical computer science. Springer, 2004.

[4] F. Blanqui, S. Coupet-Grimal, W. Delobel, S. Hinderer, and A. Koprowski. CoLoR, a Coq library on rewriting and termination. In A. Geser and H. Söndergaard, editors, *International Workshop on Termination (WST'06)*, 2006.

[5] E. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. In B. Konev and F. Wolter, editors, *Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *LNAI*, pages 148–162. Springer, 2007.

[6] A. Krauss. Certified size-change termination. In F. Pfenning, editor, *Automated Deduction (CADE-21)*, volume 4603 of *LNCS*, pages 460–476. Springer, 2007.

[7] A. Krauss. *Automating Recursive Definitions and Termination Proofs in Higher-Order Logic.* PhD thesis, Institut für Informatik, Technische Universität München, Germany, January 2009. Submitted.

[8] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Principles of Programming Languages (PoPL 2001)*, pages 81–92, 2001.

[9] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[10] R. Thiemann and C. Sternagel. Certification of termination proofs with CeTA. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics (TPHOLs '09)*, LNCS. Springer, 2009.