ABSTRACTS
2nd International Workshop on Termination

La Bresse, France

May 29-31, 1995

INRIA-Lorraine
and
Centre de Recherche en Informatique de Nancy

1 A new technique to prove termination of constructor systems.

Thomas Arts, Hans Zantema

We present a new technique to prove termination of a subclass of term rewrite systems: the constructor systems (CSs). In earlier work we introduced a transformation of logic programs into CSs, such that termination of the logic program follows from termination of the obtained CS. The technique to prove termination of CSs described in this paper is in particular suitable, but not limited to, CSs that are obtained by this transformation of logic programs. Parts of the technique can be automized very easily. Other parts can be automized for subclasses of CSs. We even have an implementation in progress that is able to prove termination of some CSs that are not simply terminating, but these examples are rather trivial. For more realistic CSs a part of the technique has to be carried out by hand.

Consider the following CS (note that it is not simply terminating)

$$\begin{array}{lcl} h(x) & \rightarrow & s(x) \\ f(x,p(x)) & \rightarrow & g(h(x)) \\ g(s(s(x))) & \rightarrow & f(x,x) \\ f(x,x) & \rightarrow & f(x,p(x)) \end{array}$$

with the following typical reduction

$$f(s(x),p(x)) \rightarrow g(h(s(x))) \rightarrow g(s(s(x))) \rightarrow f(x,x) \rightarrow f(x,p(x))$$

In reductions with this kind of cyclic behaviour the first argument of f decreases. Our technique is a formalization of the intuition that in any reduction the same defined symbol reappears with decreased arguments. With our technique we give a systematical approach to collect all the reductions with this kind of cyclic behaviour. From this collection we infer a number of conditions that have to be fulfilled by a suitable well-founded order on terms. Thus, we translate the termination problem into finding a suitable order fulfilling some conditions. We proved soundness of this transformation, *i.e.*, termination of the CS follows from the existence of the demanded order. Applying the technique on the CS mentioned above, automatically results in the demand that there has to be a well-founded order $>$, closed under substitution, such that $s(x) > x$. Such an order obviously exists, and therefore, the CS is terminating.

2 Lambda-Upsilon, a calculus of explicit substitution which preserves strong normalisation

Zine-El-Abidine Benaissa, Daniel Briaud, Pierre Lescanne, Jocelyne Rouyer-Degli

Calculus of explicit substitution is a nice formalism which includes the mechanism of substitution inside the lambda calculus itself and which allows formalising beta-reduction.

Several calculi of explicit substitutions have been proposed, but Mellies has shown that some of them do not preserve beta strong normalisation of lambda terms. In the talk we will present a proof that lambda-upsilon preserves strong normalisation. This proof is rather original in the world of lambda-calculus since it is based on a least counter-example as Nash William's proof of Kruskal theorem.

3 Modular Orientation of Equations into Rewrite-Rules for Modular Algebraic Specifications

M. Bidoit, F. Voisin

The work we present here is part of an ongoing research on "modular proofs" for modular algebraic specification languages. We consider the classical problem of orienting (conditional) equations into finitely terminating rewrite systems, when the equations belong to modules written in a modular algebraic specification language. Assuming that for each module (a subset of) the equations have been turned into a finitely terminating rewrite system, we want to be able to merge the individual rewrite systems associated with a collection of modules into a single, finitely terminating, rewrite system. The actual collection of modules to be taken into consideration will depend on the proof to be performed.

Rather than relying on syntactical criterion to allow the merging of independent systems known to be independently finitely terminating - a problem known as the "modular" termination of rewriting systems -

we investigate a different approach to "modularity" where the equations in a given module are not oriented independently but with constraints that reflect the structure of the specification.

Restricting the way in which equations in a single module can be oriented ensures the smooth inclusion of the resulting rewrite system into any collection of rewrite systems in which it can latter appear, without loosing too much on the orientation efficiency but simply postponing the orientation of some equations. Equations in a single module are statically partitioned into three sets: the equations that can be oriented in a context-independent way, the equations known to be unorientable, and the set of remaining equations whose orientation is postponed until the precise context of their use is known. The equations in the latter set are the ones that have potential, inter-modules, orientation conflicts when modules are gathered: a premature orientation of an equation in a module can prevent the orientation of the equations in the other modules of the proof environment. Therefore, the method allows to statically orient part of the equations of a module while keeping to a dynamic stage the orientation of the equations that are more context dependent.

The method is currently defined for precedence methods, like the recursive path order, which are probably the most practical methods for general purpose theorem provers.

4 Preservation of Strong Normalisation for some Explicit Substitution Calculi

Roel Bloo

We study a simple named calculus of explicit substitutions called $\lambda\mathsf{exp}$ and some extensions. Using recursive path orderings (based on Kruskal's tree theorem) we show that two of these possess the PSN-property. This means that every term that is strongly normalising with respect to β-reduction is also strongly normalising with respect to explicit substitution reduction. Until 1995 it was an open question whether a calculus of explicit substitution could have the PSN-property.

$\lambda\mathsf{exp}$ can be seen as a named version of the calculus $\lambda\sigma$ of Abadi et al.[1] with restrictions on interaction between substitutions, but was independently proposed by Kamareddine and Nederpelt inspired by the Automath-languages [4]. The notation we use here is due to [8].

The system $\lambda\mathsf{exp}$ has the following terms: $T ::= x \mid TT \mid \lambda x.T \mid T\langle x := T\rangle$. The explicit substitution reduction relation $\rightarrow_{\mathsf{exp}}$ is the union of two subrelations $\rightarrow_{\mathsf{expg}}$ and $\rightarrow_{\mathsf{exps}}$ which are generated by the rules $(\lambda x.A)B \rightarrow_{\mathsf{expg}} A\langle x := B\rangle$, $x\langle x := A\rangle \rightarrow_{\mathsf{exps}} A$, $x\langle y := A\rangle \rightarrow_{\mathsf{exps}} x$, $(AB)\langle x := C\rangle \rightarrow_{\mathsf{exps}} (A\langle x := C\rangle)(B\langle x := C\rangle)$ and $(\lambda x.A)\langle y := B\rangle \rightarrow_{\mathsf{exps}} \lambda x.(A\langle y := B\rangle)$ (here $x \not\equiv y$ and the Barendregt convention is assumed, that is, free variables in terms are distinct from bound ones and all bound variables are distinct).

Properties that can be easily proved for $\lambda\mathsf{exp}$ are the confluence of the reduction relations $\rightarrow_{\mathsf{exps}}$, $\rightarrow_{\mathsf{expg}}$ and $\rightarrow_{\mathsf{exp}}$, and termination of $\rightarrow_{\mathsf{exps}}$.

The intuition behind our proof of PSN is that the worst reductions are those that take place inside a substitution which is 'void'. A substitution $\langle x := B\rangle$ is void in $A\langle x := B\rangle$ if x is not free in A. Special care has to be paid to the notion *x is free in A*.

We define for all terms A a finite tree $T(A)$ reflecting this idea (more precisely, $T(A)$ is the tree with topmost label the pair $(\tilde{\beta}(A), \tilde{\sigma}(A))$ and subtrees $T(B)$ for all outermost void substitutions $\langle x := B\rangle$ in A, here $\tilde{\beta}(A)$ is the maximal length of β-reduction paths starting with the $\rightarrow_{\mathsf{exps}}$-normal form of A and $\tilde{\sigma}(A)$ is the maximal number of $\rightarrow_{\mathsf{exps}}$-reductions possible outside all the outermost void substitutions $\langle x := B\rangle$ in A). We then show that the recursive path ordering on these trees is in a certain sense a refinement of the reduction relation $\rightarrow_{\mathsf{exp}}$, which shall give us PSN.

Other recently discovered methods to prove the PSN-property are the notion of minimal derivation [2],[7] and nested induction on the maximal length of the β-reduction paths of the $\rightarrow_{\mathsf{exps}}$-normal forms of a term and the arguments of its substitutions [3]. These two methods however involve much technical reasoning about the position and order of β- and substitution redexes to be contracted. Our method using recursive path orderings was inspired by a similar proof by Herman Geuvers for the most restrictive system (see [3]) and is much less technical.

By giving some counterexamples inspired by P.A. Melliès' counterexample to PSN for $\lambda\sigma$ [6] we show that it is not possible to allow much interaction between substitutions and still have the PSN property, however a slight extension of the rules of $\lambda\mathsf{exp}$ is possible.

References

[1] Abadi, M., Cardelli, L., Curien, P.-L., and Lévy, J.-J., Explicit substitutions, POPL '90 San Francisco, California, Jan. 1990, pp. 31–46.

[2] Benaissa, Z.-E.-A., Briaud, D., Lescanne, P. and Rouyer-Degli, J., λv, a calculus of explicit substitutions which preserves strong normalisation (submitted), 1995, personal communication.

[3] Bloo, R., and Geuvers, J. H., Explicit Substitution: on the Edge of Strong Normalisation (in preparation).

[4] Kamareddine, F., and Nederpelt, R.P., On stepwise explicit substitution. *International Journal of Foundations of Computer Science 4 (3)*, 197-240, 1993.

[5] Klop, J. W., Term rewrite systems, in: Abramsky, S., Gabbay, D. M., and Maibaum, T. S. E. (eds.) (1992), *Handbook of Logic in Computer Science*, Vol. II, Oxford University Press, pp. 1–116.

[6] Mellies, P.-A., Typed λ-calculi with explicit substitutions may not terminate, personal communication, 1995.

[7] Ríos, A., and Kamareddine, F., A λ-calculus à la de Bruijn with explicit substitutions (draft), 1995.

[8] Rose, K. H., Explicit cyclic substitutions, CTRS '92—3rd International Workshop on Conditional Term Rewriting Systems (Pont-a-Mousson, France, July 1992), M. Rusinowitch and J.-L. Rémy, Eds., no. 656 in LNCS, Springer-Verlag, 1992, pp. 36–50.

5 Cut Elimination and rewriting: Termination proofs

E.A.Cichon, M. Rusinowitch, S. Selhab

We establish a strong form of Gentzen's Cut elimination theorem for the classical sequent calculus extending this result to intuitionistic and linear calculi. Our approach consists in the use of syntactic rewriting techniques to prove that our Cut elimination procedure terminates for any strategy.
We construct two rewriting systems (R and S) realising the normalisation of proofs, when considered as terms, starting from two different procedures of Cut elimination. We then prove termination of those rewrite systems. The use of the Contraction rule leads to splitting R into two terminating subsystems. We establish termination in a modular fashion by the quasi-commutation principle. On the other hand the termination proof for S is direct. We go on to show that the same approach can be applied successfully to the intuitionistic calculus with slight differences due to the form of intuitionistic sequents. Moreover in the case of linear sequent calculus without modalities the termination proof of the extracted rewriting system is direct because of Contraction rule dereliction. Finally using rewriting methods, we show how to extract bounds for lengths of rewrite sequences.

6 A Direct Limit Approach to Subrecursive Hierarchies

Adam Cichon, Hélène Touzet

We present a syntactic approach to ordinal recursion below ϵ_0.
We consider *ordinal terms* over the signature $\{0, s, +, *, exp\}$ with an operator Δ and we provide this structure with a well-founded ordering based on the Kruskal embedding, extending the usual definition of $<$ on $CNF(\epsilon_0)$. The main difference with set-theoretical ordinals is that limit ordinals are built up via the direct limit operator Δ, which is a more precise and more dynamic way of viewing the problem. For instance, the predecessor associated with this system exactly reflects each derivation step of the usual hydra battle and so a single *ordinal term* can encode the whole derivation. We believe this notion is more convenient and appropriate for proving termination with subrecursive hierarchies of functions.

7 A survey of the proof of the graph minor theorem

Bruno Courcelle

Robertson and Seymour have proved the analogous of Kruskal's theorem for finite graphs of various types and even hypergraphs. This result is known as the "Graph Minor Theorem".

The proof is not completely published but I can give an overview of it. The applications do not concern (yet) the termination of graph rewriting rules as for Kruskal, but rather the definition of classes of graphs by forbidden configurations. The extension of Kruskal's theorem by Friedman is useful for proving the GMT for graphs of bounded tree width.

8 Polynomial and Knuth-Bendix Orderings on Monadic Terms

Nick Cropper

The polynomial orderings on terms in two unary function symbols are fully resolved into simpler orderings. Thus it is shown that most of the complexity of polynomial orderings is redundant. The order type (logical invariant), either τ or λ (numeric invariant), and precedence is calculated for each polynomial ordering. The invariants correspond in a natural way to the parameters of the orderings, and so the tabulated results can be used to convert easily between polynomial orderings and more tangible orderings.

The orderings of order type ω^ω are two of the recursive path orderings. All of the other polynomial orderings are of order type ω or ω^2 and each can be expressed as a lexicographic combination of τ (weight), λ (matrix), and lexicographic (dictionary) orderings.

9 A note on rewriting with real numbers

Fer-Jan de Vries and Junnosuke Yamada

We will introduce a term rewriting system for a fragment of real number arithmetic in the usual redex expansion notation. The fragment deals with addition, subtraction and multiplication of real numbers. The rewriting system contains a minimal number of extra symbols. Besides the digits and $+, \times, -$ we only need two concatenation symbols:

$$x : y \equiv 10x + y \text{ and } x \colon y \equiv x + \frac{1}{10}y$$

with bracket association to the left and to the right respectively. E.g., the number 123.456 is represented by the term $((1:2):3)\colon(4\colon(5\colon6))$.

Rewriting systems that describe arithmetic in such a concrete way seems little studied. Yet these systems can be fairly elegant with puzzling properties. Cohen and Watson (RTA91) seem to have been the first to propose a TRS for integer arithmetic, and posed the problem whether their system was terminating. Their proposal appears to be rather ad hoc. Walters (1994) defined a very elegant system for integer aritmetic, and proved the termination for the fragment involving addition and substraction. We extended Walters proposal to real number arithmetic and proved termination of the fragment with addition and multiplication using Zantema's semantic labeling technique. Walters and Zantema (RTA95) finally proved termination for Walters full integer arithmetic TRS for $+, \times, -$ with help of the semantic labeling technique and an elegant decomposition argument.

In this note we show that the decomposition argument can be extended to our TRS for real number aritmetic: restricted to finite terms real number arithmetic is terminating. We also show that modulo associativity and commutativity of addition the system of the positive reals for $+, \times$ can be completed. For both full systems (integers and reals) with $+, \times, -$, it is still open whether or not it can be completed modulo $AC(-)$ and the equation $-(x+y) = -x + -y$.

If we allow infinite terms in a context of infinite rewriting things get rather wild. Terms like $1+1+1+\ldots$ and $1\colon((1:1)\colon((1:1:1)\colon\ldots))$ have infinite semantics. However we can show that any term $C[t_1, \ldots, t_n]$ has a strongly converging reduction to normal form, where $C[\ldots\ldots]$ is a finite context built from $+, \times, -$ and $t_1, \ldots, t_n$ are closed normal forms.

10 ?

Nachum Dershowitz

?

11 Normalisation of sequent calculus derivations

Roy Dyckhoff, Luis Pinto

Derivations in a cut-free intuitionistic sequent calculus such as Gentzen's LJ with the same interpretation as natural deductions are said to be "equivalent" or "permutable". Proof search which finds just one member of each equivalence class is preferable to a search which finds permutation variants of already found derivations.

We have studied some equational systems which generate the equivalence classes of this relationship (details written out for the connectives for implication and disjunction): these differ from some 1970s work of Zucker in considering systems without CUT. We are now exploring ways of converting this system into a confluent and terminating rewriting system and the problems of proving termination. We are also exploring the connection with some unpublished work of Mints on the same topic.

12 Dummy elimination in term rewriting

Maria Ferreira, Hans Zantema

Suppose we want to prove termination of the following system

$$f(g(x)) \rightarrow f(a(g(g(f(x))), g(f(x))))$$

Intuitively, the function symbol a is created but seems not to have any influence on the reductions. Taking that into account, we can eliminate it and transform the given rule into

$$\begin{array}{lcl} f(g(x)) & \rightarrow & f(\diamond) \\ f(g(x)) & \rightarrow & g(g(f(x))) \\ f(g(x)) & \rightarrow & g(f(x)) \end{array}$$

where $\diamond$ is a fresh constant. Termination of the first system is not easy to prove (since the system is self-embedding orders like *recursive path order (rpo)* cannot be used) while termination of the second system is trivially proven with *rpo* by choosing the precedence $\triangleright$ satisfying $f \triangleright g \triangleright \diamond$. Now if the transformation is sound, i. e., termination of the original system can be inferred from termination of the transformed one, our task is done. In this talk we formally describe this transformation and others and prove their soundness with respect to termination.

In general, we are interested on simplifying the process of proving termination of term rewriting systems (TRS's). A possible approach to this goal is to devise sound transformations on TRS's such that the transformed systems are somehow easier to deal with, with respect to termination proofs, than the original ones.

We present a family of transformations, based on elimination of "useless" function symbols, and show that when the transformations can be applied they are sound with respect to termination, i. e., termination of the original system can be inferred from termination of the transformed one; we also discuss how the transformations are related among themselves. The transformations are simple and can be fully automatized. The technique used to prove these results relies on a particular kind of order which is a proper subset of the recursive path order.

13 A comparison between two semantic termination proof methods

Alfons Geser

To prove that a rewrite system R terminates, one may show that

1) R is ordered by some semantic path order (with status), or 2) some semantically labelled version of R is ordered by recursive path order (with status).

The two methods are deeply correlated. Both use a monotonic interpretation, and a labelling function. Both require that R is a quasimodel for the induced order. We will show under which circumstances the two methods provide equivalent termination proofs.

14 Proving Termination of Recursively Defined Algorithms

Jürgen Giesl

We develop a method for automated termination proofs of *recursively defined algorithms* [Gie95b]. For every algorithm a well-founded ordering relating inputs and corresponding arguments of recursive calls has to be generated.

For termination proofs of *term rewriting systems* several approaches to synthesize suited well-founded *term orderings* have been suggested. But unfortunately term orderings cannot be directly used for termination proofs of *algorithms*, because term orderings may conflict with the algorithmic definitions of *defined* function symbols.

A straightforward solution is the restriction to term orderings which respect the *semantics* of the called algorithms. Another obvious solution is to transform the algorithms into a term rewriting system and to prove the system's termination instead. But with both of these solutions most termination proofs will not succeed with any of the commonly used term orderings, i.e. these solutions result in a *too weak* termination criterion.

Therefore we have developed a calculus to transform inequalities between inputs and arguments of recursive calls into inequalities *without defined function symbols*. This transformation is an *abduction*, i.e. the resulting inequalities define a relation which *contains* the relation defined by the original inequalities as a subset. Consequently well-foundedness of this new relation is sufficient for the termination of the algorithm.

As the resulting inequalities contain no algorithmically defined function symbols, they define a relation whose well-foundedness can be directly proved with term orderings. For this purpose we use a procedure for the generation of *polynomial orderings* [Gie95a].

Hence our transformation enables the application of *term orderings* for termination proofs of *algorithms*. This results in a more powerful technique than previous procedures for automated termination proofs of algorithms. We are implementing our method within the induction theorem proving system INKA.

References

[Gie95a] J. Giesl. Generating Polynomial Orderings for Termination Proofs. In *Proceedings of the 6th International Conference on Rewriting Techniques and Applications*, Kaiserslautern, Germany, 1995. To appear.

[Gie95b] J. Giesl. *Automatisierung von Terminierungsbeweisen für rekursiv definierte Algorithmen*. Doctoral Dissertation, TH Darmstadt, Germany, 1995. To appear.

15 Termination by Divide and Conquer: Some New Preservation Results

Bernhard Gramlich

This research was supported by the 'Deutsche Forschungsgemeinschaft, SFB 314 (D4-Projekt)'

Incremental and modular techniques for proving termination of rewrite systems have attracted an increasing attention within the last years. The basic idea follows the *divide and conquer* paradigm: Decompose a given system into some components, prove termination of the latter systems, and infer termination of the original system by some general preservation result. It is well-known that termination is in general not preserved under arbitrary combinations, not even under (signature) disjoint unions of rewrite systems. However, various sufficient preservation criteria for termination of combined systems have been discovered and developed in the meanwhile, for disjoint unions as well as certain restricted classes of non-disjoint unions of rewrite systems. Here we shall present and discuss some recently obtained new preservation results.

16 Complete term rewrite systems for decimal arithmetic and other total recursive functions

Richard Kennaway

We present a term rewrite system which represents addition, subtraction, and multiplication of positive and negative integers in decimal representation (or any other base). A class of term rewrite systems including this one is easily proved to be strongly normalising and confluent. The time complexity of this implementation of arithmetic is comparable to the complexity of the usual manual algorithms.

Previous authors (*) who have studied this problem have produced systems which either describe fewer arithmetic operations, or are only ground confluent rather than confluent, or do not have known strong normalisation proofs.

The translation can be extended to all recursive functions. All the resulting rewrite systems are confluent, and every total recursive function has such a representation as a terminating rewrite system. The representation of total recursive functions, and the proof of termination of the TRSs, display a modularity property, in that if a function f is defined in terms of functions $g_1, \ldots, g_n$, then the representation of f and the proof that it is terminating do not depend on the representations of $g_1, \ldots, g_n$ or on their termination proofs. (*)

Cohen and Watson, "An efficient representation of arithmetic for term rewriting" (RTA conference, 1991)

Walters "A complete term rewriting system for decimal integer arithmetic" (Report CS-9435, CWI, 1994)

Walters and Zantema "Rewrite systems for integer arithmetic" (unpublished, 1994)

De Vries and Yamada "On termination and confluence of rewriting with real numbers" (unpublished, 1994)

17 Applications of the theory of termination orderings on strings in group theory

Ursula Martin

The theory of termination ordering on strings has sometimes seemed to be of marginal interest when compared the complexities of general program termination.

However fast implementations of string completion systems have been used in group theory, and the underlying ordering affects both the performance and the outcome.

In this talk we explain how the Scott classification of division orderings on strings may be used to assign numeric and logical invariants to complete string rewriting systems, and hence to classify the the complete systems which may arise from a given group or semigroup presentation.

18 A new proof of strong normalization for an extension of the Jouannaud Okada schema

Ralph Matthes

In their contribution to LICS '91, Jean-Pierre Jouannaud and Mitsuhiro Okada gave "A Computation Model for Executable Higher-Order Algebraic Specification Languages". The main result is the strong normalization of term rewrite systems which mix β-reduction, some strongly normalizing conservative first-order rewrite system and higher-order rewrite rules obeying the restrictions of the "general schema". The idea of the proof has been successfully applied to more powerful type disciplines and type assignment systems by F. Barbanera, M. Fernández, H. Geuvers and S. van Bakel. In 1994 Fernández and Jouannaud generalized the recursive schema to lexicographic comparison for the first-order case. An update of the Jouannaud Okada article is in preparation some stages of which I had the chance to read in advance. One example there reads:

$$\begin{aligned}
\mathsf{append}(\mathsf{nil}, l) &\mapsto l, \\
\mathsf{append}(\mathsf{cons}(x, l), l') &\mapsto \mathsf{cons}(x, \mathsf{append}(l, l')), \\
\mathsf{append}(\mathsf{append}(l, l'), l'') &\mapsto \mathsf{append}(l, \mathsf{append}(l', l'')), \\
\mathsf{map}(X, \mathsf{nil}) &\mapsto \mathsf{nil}, \\
\mathsf{map}(X, \mathsf{cons}(x, l)) &\mapsto \mathsf{cons}(X(x), \mathsf{map}(X, l)).
\end{aligned}$$

The first three rules are regarded as the first-order rewrite system, the strong normalization of which has to be shown by hand, and the two last rules obey to the recursive schema. Obviously, even the arguments of append on the right-hand-sides are always lexicographically smaller (in the sense of subterms) than the corresponding leading left-hand-side occurrences of append. The Jouannaud Okada schema doesn't allow for the associativity rule because of the nesting of append on the left-hand-side. The generalization to be presented overcomes this restriction. In the example above this implies that strong normalization of the subsystem of the first three rules needs no proof by hand anymore, and that additional first-order rules may be studied without complications possibly coming from the rules for append.

The proof is **not** an extension of the existing proof, but proceeds totally different. The computability predicate is defined by induction on types and not by general positive induction. The algebraic decomposition is a priori effectively computable. The main induction is over the inductive definition of terms. Not only the first-order rules are dealt with in a "principal case", but also the higher-order rules (obeying a schema which allows all combinations of multiset and lexicographic ordering over the natural ordering of terms by the subterm relation and doesn't restrict nesting or even any ingredient of the rules to be first-order). Hence, the proof of modularity of strong normalization is itself modular: It decomposes into the β-reduction part which is proved in a standard way, the first-order part whose proof is only slightly differing from Jouannaud's, and the higher-order part which is essentially new.

The main problem is to prove the strong computability of the recursive function calls, whose arguments are smaller in the well-founded ordering. Especially the lexicographic ordering gives rise to newly appearing terms in a reduction whose computability is not given by the induction hypothesis.

The solution is to follow the idea of the proof of strong normalization for Gödel's System T. The recursion operators have their step terms as arguments, hence the right-hand-sides of the rules are built up with the syntactic material of the left-hand-sides, and therefore no "new" terms pop up.

Following this idea the original (finitely many!) rules are syntactically analyzed and the resulting syntactic material is incorporated into transformed rules. The resulting system can be shown to be strongly normalizing in the modular way stated above, which immediately implies the same result for the source system.

Incorporating η-reduction does not need but one additional line in the proof, simultaneous recursion only blows up the notation, and the extension to type quantification (in the simplest case) should work by replacing the standard computability method by the candidate method as Jouannaud and Okada claim and sketch.

19 Simple Termination of Rewrite Systems

Aart Middeldorp, Hans Zantema.

One of the main problems in the theory of term rewriting is the detection of termination: for a fixed system of rewrite rules, determine whether there exist infinite reduction sequences or not. Huet and Lankford showed that this problem is undecidable in general. However, there are several methods for proving termination that are successful for many special cases. Virtually all these methods yield, when applicable, simple termination. A term rewriting system is called simply terminating if there is a simplification order that orients the rewrite rules from left to right.

The basic ingredient of a simplification order is the subterm property, but in the literature two different definitions are given: one based on (strict) partial orders and another one based on preorders. In the first part of the talk we argue that there are no compelling reasons for chosing the second one. We prove (constructively) that every term rewriting system that can be shown to be terminating by means of a simplification order based on preorders, can be shown to be terminating by means of a simplification order based on partial orders. Since basing the notion of simplification order on preorders is more susceptible to mistakes and results in stronger proof obligations, simplification orders should be based on partial orders.

Simplification orders are known to be well-founded orders on terms over a finite signature. This important result no longer holds if we consider infinite signatures. Nevertheless, well-known simplification orders like the recursive path order are also well-founded on terms over infinite signatures, provided the underlying precedence is well-founded. In the second part of the talk we propose a definition of simplification order that matches exactly the requirements of Kruskal's Tree Theorem, since that is the basic motivation for the notion of simplification order. According to this new definition all simplification orders are well-founded, both over finite and infinite signatures. For finite signatures the new and the old notion of simplification order coincide. It is immediate from the definition that every recursive path order over a well-founded precedence can be extended to a simplification order, and hence it is well-founded.

In the third part of the talk we compare our class of simply terminating term rewriting systems with other proposals. It turns out that our class has a better modular behaviour: We show that simple termination is a modular property of constructor-sharing term rewriting systems.

20 The Term Rewriting Roller Coaster

Cyrus F. Nourani

A technique for generating normal forms subtree replacement systems is defined by defining model-theoretic forcing properties on algebraic trees. A forced termination rewrite computation theory is defined. Positive forcing [1] conditions are defined on algebraic trees such that infinitary models could be generated for algebraic subtree replacement systems by a Universal Inductive Closure, abbreviated by UIC, which defines a positive generic model. Models are essential for solving term rewriting problems and we present methods that show how to define them. Given a presentation, the UIC yields a generic set G. The generic set G generates a model in which the original group, for example A, is embedded. Every universal group sentence consistent over A can be embedded in some condition p in G; and every existential sentence has an instance in a condition p in G; therefore, they are satisfied by the generic model M generated by UIC. We can get specific and define an algebraically closed group by forcing . For example, let R be the set of terms over the group signature. Define $t1 == t2$ iff there is a condition p in _()_, where _()_ is the generic filter UIC . $p|| - +t1 = t2$. == is easily seen to be an equivalence relation by the definition of the positive forcing property, where $|| - +$ is the $-$force relation. It is also a congruence relation with respect to the operations of the signature -1 and . Now, let B be the canonical term algebra isomorphic to the quotient of R with respect to ==.

Theorem $B = (B.. . -1, \epsilon)$ is an algebraically closed group.

The models are generated by infinitary conditions and sets, to model a theory. The are models generated by infinitary forcing conditions. We are not solving word problems in the proof theoretic sense. We present and generate models, that could have many new Skolem functions added on such that a set of axioms is modeled. Term rewriting to normal forms feels like a roller coaster ride from the pint of view of the intended normal form. Some methods are a longer ride, some methods could appear as if the computer time is never enough, and some could not terminate and the normal form lost in the forest of trees. The theorem has the following consequent- by picking a generating set for a G-diagram we can define a canonical initial model- - which is by the results of the sections above an algebraically closed group. Nourani, C.F.,"The Connection Between Positive Forcing and Tree Rewriting," Proc. Logics In Computer Science Conference (LICS) and ASL, Stanford University, July 1985, Proc. JSL.

21 Termination of Graph Rewriting and Forward Closures

Detlef Plump

Forward closures are introduced for the algebraic approach to graph rewriting (also known as double pushout approach). Similar to the case of term rewriting, forward closures are restricted derivations in which each step depends on previous steps. It is shown that a graph rewriting system is terminating if and only if there is no infinite forward closure. This differs from the corresponding result for term rewriting [Dershowitz 1981] which has to require right-linear rules.

Applied to term graph rewriting, the result characterizes termination of term rewriting systems under graph rewriting. In particular, it provides a method to show termination for systems that are non-terminating with tree rewriting but terminating with graph rewriting. As an example, the graph rules for the system

```
f(a,b,x)  →  f(x,x,x)
       b  →  a
```

admit only finite forward closures, proving that graph rewriting is terminating (although term rewriting is not).

22 On Normalisation in λ-calculus

Femke van Raamsdonk, Paula Severi.

This work represents an effort to shed some more light on various results concerning normalisation in λ-calculus. We deal with λ-calculus with only β-reduction (defined by $(\lambda x.M)N \rightarrow M[x := N]$).

To give a characterisation of the weakly normalising terms is straightforward: it is the set of all normal forms closed under expansion. A specialisation of this idea yields a characterisation of the strongly normalising terms in the form of an inductively defined set $\mathcal{SN}$.

This set can be viewed in different ways. It is the largest saturated set. Further, more in the spirit or rewrite theory, it is the set of normal forms closed under expansion, where expansion is subject to two restrictions: first, the expansion step should yield a new outermost redex, and second, the argument of the redex introduced by the expansion step should be in the set of strongly normalising terms.

The interesting thing of the definition of $\mathcal{SN}$ is that it permits to give new proofs of important results concerning normalisation in λ-calculus. In most cases the new proofs are essentially simpler than existing ones. Moreover, we feel that it is important to have different proofs of important results because they may help us to understand not only *the mechanics* of the proofs but also *the reasons* for their validity.

Using the characterisation of strongly normalising terms we obtain new proofs of the following.

1. We give a short and simple proof of finiteness of developments. It is different from the proof using a decreasing labelling and different from the proof given by De Vrijer.

2. An extended notion of developments, called superdevelopments, is proven to be finite.

3. The strategy F_{bk} defined by Bergstra and Klop, is perpetual (meaning that it yields an infinite rewrite sequence whenever possible).

4. The strategy F_∞ defined by Barendregt, Bergsta, Klop and Volken is perpetual.

5. The strategy F_∞ is moreover maximal, meaning that it yields the longest possible reduction to normal form whenever the initial term is strongly normalising.

6. Simply typed λ-calculus is strongly normalising. We do not have to introduce the notion of saturated sets since we in fact work with the largest saturated set.

7. We present a structural proof of the fact that all terms that are typable using intersection types are strongly normalising and vice versa.

Currently under investigation are strong normalisation of the the Calculus of Constructions and the complexity of proofs of strong normalisation of simply typed λ-calculus.

23 Extension Orderings and Modularity of Termination of TRS

Albert Rubio

Partially supported by the Esprit Working Group CCL, ref. 6028.

A well-known related problem in rewrite theory is to prove the modularity of termination of *term rewrite systems* (TRS's), that is, given two terminating rewrite systems R_1 and R_2, to show that their union $R_1 \cup R_2$ is also terminating. This property was shown to be false in general, even when the two rewrite systems do not share any function symbols (Toyama 1987). Therefore many efforts have been devoted to finding sufficient conditions under which termination is modular.

Most existing results impose syntactic (or abstract) conditions on (one of) the two TRS's, and some other ones impose restrictions on the method used to show the termination of the involved TRS's (e.g. simple termination).

In this context it is quite surprising that, although usually termination of TRS's is proved by actually building a *reduction ordering* in which the rewrite relation is contained, in most modularity proofs for termination such an ordering is not explicitly built, that is, in this sense these proofs are not constructive. Hence, for simplicity and to improve the understanding of the problem, it is worth to look at it from a more constructive point of view. This can give information about the weakest conditions ensuring modularity and why they are needed. As a good example for this viewpoint, *total termination* (i.e. the rewrite relation is included in some total reduction ordering) was proved modular provided that one of the TRS's is conservative.

The reason for this requirement seems to be rather technical, and the proof does provide no intuition of its need. Here we prove the same result in a simple way, actually building a total ordering containing the combined system. This gives some intuition about the restriction.

Our main idea is to generalize the combination of TRS's to the combination of orderings. Instead of proving that the ordering induced by the combined rewrite system is well-founded, we build a well-founded ordering and prove that it includes the combined TRS. This is done by means of *extension orderings*. By an extension ordering of a collection of orderings $\succ_1, \ldots, \succ_n$ on disjoint signatures, we mean another ordering $\succ$ s.t. $\succ_i \subseteq \succ$ for all i. Actually, an extension ordering is interesting when it preserves (part of) the properties satisfied by the initial orderings.

With this approach new constructive proofs for e.g. modularity of simple termination are obtained.

24 Classifying Total Division Orderings On Strings

Elizabeth Scott

Division orderings are well founded and hence are suitable for use in termination proofs. Recent work has greatly increased our understanding of the class of total division orderings.

Let $\mathcal{A} = \{a, b\}$, let $\mathcal{A}^*$ denote the set of strings of letters from $\mathcal{A}$ and let $\succ$ be a total division ordering on $\mathcal{A}^*$. It is known that $(\mathcal{A}^*, \succ)$ must have order type ω, ω^2 or ω^ω. In addition, there exist *weights* for the letters a, b such that if the weight of a string u is greater than the weight of a string v then $u \succ v$.

The order type of the ordering is ω if and only if the weights of both letters are non-zero. In this case there are only finitely many strings of a given weight. Thus, for all but finitely many strings, the relation of a string to a given string u is determined by the weights.

In the case where the weight of one of the letters is zero the weights tell us much less about the ordering. This case corresponds exactly to the case where the order type is ω^2 or ω^ω.

When the order type is ω^ω then $\succ$ is one of the four essentialy equivalent recursive path orderings.

There is less complete information about the orderings of order type ω^2. In this talk I will consider the following conjecture:

CONJECTURE *If $\succ$ is a total division ordering on $\{a, b\}^*$ of order type ω^2 then there exists a positive real number λ such that,*

$$t_0 + t_1\lambda + \ldots + t_n\lambda^n > r_0 + r_1\lambda + \ldots + r_n\lambda^n \quad \Rightarrow \quad a^{t_0}ba^{t_1}\ldots ba^{t_n} \succ a^{r_0}ba^{r_1}\ldots ba^{r_n},$$

where $b \succ a^i$, for all i.

If the conjecture is true then, since $\lambda > 0$ and the coefficients t_i are non-negative, there would only be finitely many strings with n occurrences of the letter b and a given value for $t_0 + t_1\lambda + \ldots + t_n\lambda^n$. It is known that if $(\mathcal{A}^*, \succ)$ has order type ω^2 and if the number of occurrences of b in u is greater than the number of occurrences of b in v then $u \succ v$. Thus the above conjecture would provide the same degree of information for the cases of orderings on two letters of type ω^2 as the weights result provides for orderings (on any finite number of letters) of type ω, i.e. there are only finitely many strings whose relation to u with respect to $\succ$ is not determined by the relative numbers of occurrences of b or by λ.

25 Embeddings and Infinite Reduction Paths in Untyped λ-Calculus

Morten Heine Sørensen.

In the untyped λ-calculus there are terms which have an infinite reduction path. The most obvious example is $\Omega \equiv \omega\,\omega$, where $\omega \equiv \lambda x.x\,x$. It has an infinite reduction path where in every step the term reduces to itself:

$$\sigma: \quad \Omega \to_\beta \Omega \to_\beta \ldots$$

Not all infinite reduction paths have this form, but other infinite reduction paths seem to share a similar but weaker property. For instance the term $\Psi \equiv v\,v$, where $v \equiv \lambda x.x\,x\,y$, has the following infinite reduction path:

$$\tau: \quad \Psi \to_\beta \Psi\,y \to_\beta \Psi\,y\,y \to_\beta \ldots$$

In every step the redex Ψ appears as a *subterm*, and the context of the redex is extended with an application $\bullet\, y$.

As a slightly more complicated example consider the term $v\,y\,v$, where $v \equiv \lambda a x.x\,(a\,y)\,x$. This term has the following infinite reduction path:

$$\rho: \quad v\,y\,v \to_\beta (\lambda x.x\,(y\,y)\,x)\,v \to_\beta v\,(y\,y)\,v \to_\beta (\lambda x.x\,(y\,y\,y)\,x)\,v \to_\beta v\,(y\,y\,y)\,v \to_\beta \ldots$$

The path ρ is similar to τ, but the extra application $\bullet\, y$ is not added to the context: it is added *inside* the redex.

In all three reduction paths above it happens infinitely often that a term is followed (after a number of steps) by another term in which the first is embedded (the latter arises from the former by addition of a number of subterms). In the first part of the paper we formalize this idea and show that it holds for all infinite reduction paths for all notions of reduction on λ-terms. The result is a translation into λ-calculus of Kruskal's Tree Theorem.

In all three reduction paths above it may also be noted that all the terms have Ω embedded in them. In the second part of the paper we formalize this idea and show that it holds for all β-reduction paths. The proof is an application of the theory of so-called perpetual reductions, currently being developed by the author and others.

26 Termination Problems Requiring Semantic Proofs

Joachim Steinbach

There exist various techniques for proving termination of term rewriting systems (TRSs). Most of them are based on simplification orderings requiring the subterm property $C[t] \succ t$. This condition prevents termination proofs of a rule $l \to r$ where l is homeomorphically embedded in r (for example, $\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$). Especially, recursive function definitions can therefore be difficult to deal with. Thus, the development of adequate approaches has been forced since a few years. However, most of these (semantic) techniques cannot be applied without any knowledge on the TRS, the termination of which must be shown since it is very hard to choose adequate ordering parameters for a given TRS. Therefore, we want to present a catalogue of hard termination problems (special TRSs) together with appropriate orderings. For an impression of what kinds of problems we are interested in, see the following two examples.

Example 1 (Pseudo-Fibonacci Numbers, D.R. Hofstadter) *This specification is reminiscent of the Fibonacci definition in that each new value is a sum of the two previous values – but not of the* immediately *previous two values. Instead, the two immediately previous values tell* how far to count back *to obtain the numbers to be added to make the new value.*

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{f}(\mathsf{s}(0)) & \xrightarrow{1} & \mathsf{s}(0) & \mathsf{f}(\mathsf{s}(\mathsf{s}(0))) & \xrightarrow{2} & \mathsf{s}(0) \\
\mathsf{f}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))) & \xrightarrow{3} & \multicolumn{4}{l}{\mathsf{f}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))-\mathsf{f}(\mathsf{s}(\mathsf{s}(x)))) + \mathsf{f}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))-\mathsf{f}(\mathsf{s}(x)))}
\end{array}$$

The difficulty in proving termination is that the left-hand side of the third rule is homeomorphically embedded in even both arguments of + of the right-hand side.

Example 2 (List Reverse, G. Huet & J.-M. Hullot) *Consider an axiomatization of list structures where* nil *and* $\bullet$ *are constructors. Let the reverse* rev *be defined with the help of two auxiliary functions* rev1 *and* rev2 *such that* $\mathsf{rev1}(x.y) \equiv car(\mathsf{rev}(x \bullet y))$ *and* $\mathsf{rev2}(x.y) \equiv cdr(\mathsf{rev}(x \bullet y))$.

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{rev}(\mathsf{nil}) & \xrightarrow{1} & \mathsf{nil} & \mathsf{rev1}(x.\mathsf{nil}) & \xrightarrow{3} & x \\
\mathsf{rev}(x \bullet y) & \xrightarrow{2} & \mathsf{rev1}(x.y) \bullet \mathsf{rev2}(x.y) & \mathsf{rev1}(x.y \bullet z) & \xrightarrow{4} & \mathsf{rev1}(y.z) \\
\mathsf{rev2}(x.\mathsf{nil}) & \xrightarrow{5} & \mathsf{nil} & & & \\
\mathsf{rev2}(x.y \bullet z) & \xrightarrow{6} & \multicolumn{4}{l}{\mathsf{rev1}(x.\mathsf{rev}(\mathsf{rev2}(y.z))) \bullet \mathsf{rev2}(x.\mathsf{rev}(\mathsf{rev2}(y.z)))}
\end{array}$$

A termination proof for this TRS has to take into account the mutual recursion of rev *and* rev2 *(rules 2 & 6) by simultaneously requiring* $\bullet$ *to be minimal.*

For each example, (i) a detailed analysis wrt. the termination point of view will be given and (ii) parameters for semantic path orderings, transformation orderings and other semantic techniques will be proposed in order to prove termination, which is intended to help not only for evaluating the power of but also for developing new orderings.

27 On Proving the Termination of Algorithms by Machine

Christoph Walther

Proving the termination of a recursively defined algorithm requires a certain creativity of the (human or automated) reasoner for inventing a hypothesis whose truth implies that the algorithm terminates. We present a reasoning method for simulating this kind of creativity by machine. The proposed method works automatically. i.e. without any human support. We show. (1) how a termination hypothesis for an algorithm is synthesized by machine. (2) which knowledge about algorithms is required for an automated synthesis. and (3) how this knowledge is computed. Our method solves the problem for a relevant class of algorithms. including classical sorting algorithms and algorithms for standard arithmetical operations. which are given in a pure functional notation. The soundness of the method is proved and several examples are presented for illustrating the performance of the proposal. The method has been implemented and proved successful in practice.

References

[Walther. 94] Christoph Walther. *On Proving the Termination of Algorithms by Machine*. Artificial Intelligence. vol 71. no 1. 1994.

28 Some applications of monotone interpretations

Andreas Weiermann

We plan to include the following material.
1. Cichon & Weiermann 1995:
The derivation lengths of rewrite systems which model parameter recursion. simple nested recursion. unnested multiple recursion. etcetera are primitive recursive.
2. Weiermann 1995:
A natural rewrite system for the Kirby Paris hydra battle is defined an its derivation lengths are classified with appropriate Hardy functions [cf. Problem 23 LNCS 488].
3. Weiermann 1993:
We give some general conditions under which the following principle (due to Cichon) holds: The derivation lengths of terminating rewriting systems is related to the order type of its underlying termination ordering via the slow growing hierarchy. [This result generalizes prior results of Hofbauer and Weiermann concerning the multiset and lexicographic path ordering.]

29 Total termination of term rewriting is undecidable

Hans Zantema

Usually termination of term rewriting systems is proved by means of a monotonic well-founded order. It can be proved that most standard orders used for this purpose like recursive path order and Knuth-Bendix order are total on ground terms. or can be extended to an order that is total. monotonic and well-founded on ground terms. A term rewriting system is called totally terminating if its termination can be proved by means of a monotonic well-founded order that is total on ground terms. For example. a terminating systems like

$$\begin{aligned} f(a) &\rightarrow f(b) \\ g(b) &\rightarrow g(a) \end{aligned}$$

is not totally terminating since it is essentially impossible to orient a and b.

It is well-known that termination is an undecidable property of finite TRS's. Even simple termination is undecidable. even for single rewrite rules. On the other hand. compatibility with recursive path order or Knuth-Bendix order is decidable. The notion of total termination is stronger than the undecidable property simple termination, and weaker than the decidable property of path order termination. The question arises whether total termination is decidable or not. In September 1994 two conflicting conjectures arose on this topic: Albert Rubio conjectured that it was decidable while Jean-Pierre Jouannaud conjectured that it was undecidable.

As the title indicates, we prove that total termination is undecidable. Our basic tool for proving undecidability is Post's Correspondence Problem, which is a well-known undecidable problem. First we give a very simple new proof that simple termination is an undecidable property by means of a transformation of an arbitrary instance of Post's Correspondence Problem to a term rewriting system. Next we give a slight modification of this transformation for which the proof of undecidability of total termination can be given. The proof consists of the construction of a monotonic well-founded total order. For non-totally terminating systems the properties totality and monotonicity are conflicting: the main problem in the proof is to use the particular shape of the rewrite system to be able to combine totality and monotonicity.

Abstracts (Non-Participants)

30 Algorithms Used In Automated Natural Deduction Prover And Experiments

Li Dafa

The paper presents algorithms and some heuristics used to implement automated natural deduction prover (ANDP), the natural deduction was adapted from Gentzen system. Andrews' Challenge and Turing Halting Problem were solved using ANDP, but they have not been proved by other prover.

Keywords: Unification Algorithms, Gentzen system, heuristics, natural deduction

There are many rules in natural deduction system adapted from Gentzen system. The rules for introducing quantifiers are Universal Generalization and Existential Generalization. The rules for eliminating quantifiers are Universal Specialization and Existential Specialization.

We have two algorithms to handle quantifiers, one is for introducing quantifiers, the other is for eliminating quantifiers.

The paper presents some techniques used to implement the automated Gentzen system.

The experiments:

(1) The mechanical proof of Andrews' challenge in natural deduction style, but it is not solved by resolution.

(2) The mechanical proof of Halting Problem in natural deduction style.

It Failed To Prove Halting Problem By Resolution.

Massimo Bruschi reported the following facts in [2]:

"I was unable to obtain a mechanical proof of the theorem simply by applying EXprover". "Applications of OTTER, Argonne's theorem prover, also were unsuccesseful.

Prof. Li Dafa
Dept. of Applied Mathematics,
Tsinghua University,
Beijing 100084,
China.

e-mail: yanglp@bepc2.ihep.ac.cn
Fax: (861) 2562768

31 The Uniform Termination Problem for Certain Classes of Semi-Thue Systems

Robert McNaughton

(Announcement of results, May 13, 1994)

Where C is a class of semi-Thue systems, the uniform termination problem for C is: Given $S \epsilon C$, is there an infinite derivation in S? Of interest is this problem when C is the class of one-rule Thue systems. The question about the solvability of this problem, a live one for some years now, is the focus of the research reported here.

An *inhibitor* of a semi-Thue system is an alphabetic character that does not occur in the left side of any rule but occurs in every right side. Our first result is an algorithm for the uniform termination problem for the class of semi-Thue systems with an inhibitor (but with any finite number of rules). For example, the one-rule semi-Thue system with the single rule $cb \to bbdcc$ has the inhibitor d. This system has an infinite derivation whose first three lines are *cbb*, *bbdccb*, *bbdcbbdcc*.

We designate the one rule of a one-rule semi-Thue system as $w_1 \to w_2$. Some one-rule semi-Thue systems without inhibitors have derivations that are like derivations in systems with inhibitors. A derivation (finite or infinite) in a one-rule system is *well behaved* if, for every occurrence of w_1 in any line each of whose letter occurrences is introduced in that line or some previous line as a letter of w_2, the following is true: $w_1 = xy$, where (1) the occurrence of x comes totally from one occurrence of w_2 introduced in a line, (2) the occurrence of y comes totally from an occurrence of w_2 introduced in another line, and (3) exactly one of the following holds: (a) the occurrence of x is a suffix of its w_2, (b) the occurrence of y is a prefix of its w_2. Otherwise the derivation is *ill behaved.* (It is not difficult to prove that, in a one-rule system with an inhibitor, all derivations are well behaved.)

Our second result is an algorithm to tell whether a given one-rule semi-Thue system has an ill behaved finite derivation. Our third result is an algorithm to determine whether there exists an infinite well behaved derivation in any given one-rule semi-Thue system. Our fourth result is that a system that has an infinite well behaved derivation has a loop; i.e., there are strings x_1, x_2, x_3 such that $x_2 \to^+ x_1x_2x_3$. (In the example above, there is such a loop: $x_2 = cbb$, $x_1 = bbd$ and $x_3 = dcc$.) The question corresponding to this last result for systems all of whose infinite derivations are ill behaved is open.

For comparison, consider the system with the one rule $cb \to bbcc$, which has no inhibitor but is similar to the example above. It has an infinite well behaved derivation similar to that mentioned for the other system, whose first three lines are *cbb*, *bbccb*, *bbcbbcc*.

As a result of this research, the termination problem for one-rule Thue systems is reduced to the termination problem for one-rule systems without infinite well behaved derivations. The system $ccb \to bbccc$ is an example of one with no infinite well behaved derivation but has an infinite ill behaved derivation; one such is the infinite derivation whose first six lines are *ccccbb*, *ccbbcccb*, *bbcccbcccb*, *bbcbbccccccb*, *bbcbbccccbbccc*, *bbcbbccbbcccbccc*. (Note that the first line is embedded in the fifth and the second is embedded in the sixth. The unique occurrence of w_1 in the fifth line comes from occurrences of w_2 introduced in three distinct lines.)

References

[1] Winfried Kurth. *Termination und Konfluenz von Semi-Thue-systemen mit nur einer Regel.* Dissertation. Technischen Universität Clausthal. 1990.

[2] Winfried Kurth. "Explanations to the text." unpublished paper of five pages distributed by the author, summarizing [1].

[3] Robert McNaughton. "The uniform halting problem for one-rule semi-Thue systems: progress report." Report 94-18. Department of Computer Science. Rensselaer Polytechnic Institute. Troy. NY 12180. U.S.A.. 1994. (I would like to announce that Theorem 2.8 is not valid, but becomes valid with some changes in definitions. Also, Theorems 3.1 and 3.2 should have acknowledged the work of Zantema and Geser [4].)

[4] H. Zantema and A. Geser. "A complete characterization of termination of $0^p1^q \to 1^r0^s$." presented at a conference in 1995. Appeared previously as Report UU-CS-1994-44. Computer Science. Universiteit Utrecht. Utrecht. The Netherlands. 1994.

[5] Winfried Kurth. "One-rule semi-Thue systems with loops of length one, two or three." Report. Abteillung für forstliche Biometrie und Informatik. Universität Göttingen. Germany. January 1995.

Robert McNaughton
Department of Computer Science
Rensselaer Polytechnic Institute
Troy. NY 12180-3590. U.S.A.
mcnaught@cs.rpi.edu