*Electronic Notes in Theoretical Computer Science 58 No. 2 (2001)*

*URL:* [*http://www.elsevier.nl/locate/entcs/volume58.htm*](http://www.elsevier.nl/locate/entcs/volume58.html)*l 34 pages*

*Induction for termination* with local strategies

*Olivier* [*Fissore*](#_bookmark0) *a, Isabelle* [*Gnaedig*](#_bookmark1) *b and H el ene Kirc*[*hner*](#_bookmark2) *c a LORIA-CNRS, P 239 F-54506 Vand uvre-l es-Nancy Cedex,*

*e-mail:* [*Olivier.Fissore@loria.fr*](mailto:Olivier.Fissore@loria.fr)

*b LORIA-INRIA, P 239 F-54506 Vand uvre-l es-Nancy Cedex, e-mail:* [*Isabelle.Gnaedig@loria.fr*](mailto:Isabelle.Gnaedig@loria.fr)

*c LORIA-INRIA & LORIA-CNRS, P 239 F-54506 Vand uvre-l es-Nancy Cedex, e-mail:* [*Helene.Kirchner@loria.fr*](mailto:Helene.Kirchner@loria.fr)

*Abstract*

*In this paper, we propose a method for speci cally proving termination of rewrit- ing with particular strategies: local strategies on operators. An inductive proof procedure is proposed, based on an explicit induction on the termination property. Given a term, the proof principle relies on alternatively applying the induction hy- pothesis on its subterms, by abstracting the subterms with induction variables, and narrowing the obtained terms in one step, according to the strategy. The induction relation, an F -stable ordering having the subterm property, is not given a priori, but its existence is checked along the proof, by testing satis ability of ordering constraints.*

*Keywords: Rewriting, termination, local strategy on operators, rule-based lan- guages, induction, narrowing, ordering constraints.*

# *1 Introduction*

*Termination of rewriting is a crucial problem in automated deduction, for* equational logic, as well as in programming, for rule-based languages. As it is undecidable in general, it is ensured in particular contexts with suÆ- cient conditions. A lot of termination proof techniques have been proposed, most of them using noetherian orderings on terms. But they usually tackle the property for the standard rewriting relation and essentially work on free term algebras. In the context of rule-based languages such as ASF+SDF [14], OBJ3 [13], Maude [5], CafeOBJ [10], Stratego [20], or ELAN [3], where pro- grams are sets of rules and executions consist in rewriting ground expressions, it would be useful to have more speci c termination proof tools: methods allowing to prove termination under speci c reduction strategies, or to prove

*c 2001 Published by Elsevier Science B. V. Open access under* [*CC BY-NC-ND license*](http://creativecommons.org/licenses/by-nc-nd/3.0/)*.*

*termination on the ground term algebra, for term rewriting systems (TRSs* in short) that are not terminating on the free term one. The proof method we propose here, based on an explicit induction on the termination property, enables us to tackle these problems.

*In the context of programming, there are sets of rules, that lead to di-* vergent computations when all derivations are considered, but that terminate for particular strategies. A famous example is the evaluation of a recursive function de ned with an if\_then\_else\_ expression, which can diverge if the

*rst argument is not evaluated rst.*

*Local strategies on operators are used in this context, in* [*particular*](#_bookmark18) *to force* the evaluation of expressions to terminate. This kind of strategy is allowed by languages such that OBJ3, CafeOBJ or Maude, and studied in [7] and [18]. It is de ned in the following way: to any operator f is attached an ordered list of integers, giving the positions of the subterms to be evaluated in a given term, whose top operator is f . For example, the TRS

*f (i(x)) ! if then else(zero(x); g(x);f (h(x)))*

*zero(0) ! true*

*zero(s(x)) ! f alse if then else(true; x; y) ! x*

*if then else(f alse; x; y) ! y*

*h(0) ! i(0)*

*h(x) ! s(i(x))*

*using the conditional expression, does not terminate for the standard* rewriting relation, but does with the following strategy: LS(ite) = [1; 0], LS(f ) = LS(zero) = LS(h) = [1; 0] and LS(g) = LS(i) = [1], where if then else is denoted ite for short.

*As far as we know, speci c termination proof tools for rewriting with strate-* gies have only been given for the innermost case [1] and for the context sen- sitive rewriting [17,16,21,11] on free term algebras, and for the innermost and the outermost cases on ground term ones [12]. Here, we propose a termina- tion proof method for the case of local strategies on operators, following the induction proof principle proposed in [12]. Note that with our approach we handle the leftmost innermost and the innermost strategies: the leftmost in- nermost strategy is a particular case of local strategy, and as proved in [15], termination of rewriting is equivalent for the leftmost innermost and the in- nermost strategies. As said above, there are also termination results for a kind of rewriting called context-sensitive rewriting. In this context, rewriting is allowed only at some speci ed position in the terms, which is di erent from local strategies, that are more speci c: in the second case, not only allowed rewriting positions are speci ed, but also the order to consider them. Except for particular cases of local strategies, the two kinds of strategy are di erent. The main idea of our proof method is to use explicit induction on the termination property in order to prove that any element t of a given set of

*terms T terminates i.e. there is no in nite derivation chain starting from t.* Our induction principle uses an ordering on ground terms having the subterm property. It is based on the simple idea that if reducing a term t according to a given strategy rst requires to normalize a subterm t0 of t, we can suppose, by induction hypothesis, that t0 terminates for the same strategy. If we replace t0 by an induction variable X representing any of its normal forms, it then remains to prove that the term u obtained by replacement of t0 by X in t is terminating, to prove that t is terminating. A rewriting step is then performed on u following the di erent possible values of X: it is computed by narrowing. This process is iterated until obtaining a non narrowable term, or a term the induction hypothesis applies on. Note that the induction ordering is not given a priori but constrained during the proof by setting ordering constraints. Applying the induction hypothesis then lies on testing whether these constraints are satis able.

*On the previous example, our method consists in proving termination of* the constants, and of the terms of the form f (T ); zero(T ); ite(T1; T2; T3); h(T ); i(T ); g(T ); s(T ), for the previously given strategy, whatever the values of the ground terms T; T1; T2; T3. Obviously, 0; true; f alse are in normal form and then terminating. For i(T ) (like for s(T ) and h(T )), using an induction order- ing such that i(T ) T , by induction hypothesis, we can suppose that T is terminating. So is i(T ), since i is a constructor (i.e. i is not a top symbol of left-hand side of rule).

*By de nition of the strategy, normalizing zero(T ) rst consists in normal-* izing T , into any of its normal forms T # if it exists, and then zero(T #) at the top position. For the same ordering as previously, we have zero(T ) T . Then, by induction hypothesis, T terminates. Let T # be any of its normal forms (there can be several normal forms if the system is not con uent). The termination of zero(T ) is then reduced to the termination of zero(T #), which can only reduce into true or f alse.

*Normalizing h(T ) also rst consists in normalizing T . In a similar way as* previously, the induction hypothesis can be applied to T . The termination of h(T ) is then reduced to the termination of h(T #), which can only reduce into i(0), that is in normal form, or into s(i(T #)), that is also in normal form.

*Normalizing ite(T1; T2; T3) rst consists in normalizing T1. As previously,* T1 can be supposed to be terminating, and the termination of ite(T1; T2; T3) reduced to the termination of ite(T1#; T2; T3). According to the strategy, ite(T1#; T2; T3) is then reduced into T2 or T3, that are terminating by induction hypothesis.

*We proceed in the same way for studying how f (T ) normalizes.*

*Our goal here is to provide a procedure implementing such a reasoning.* In Section 2, the background is presented. Section 3 introduces the basic notions formalizing our induction principle. In Section 4, a rule-based algo- rithm mechanizing the proof principle is given, its correctness is established and examples are given.

# *2 The background*

*We assume that the reader is familiar with the basic de nitions and notations* of term rewriting given for instance in [6]. T (F; X ) is the set of terms built from a given nite set F of function symbols having an arity n 2 N , and a set X of variables denoted x;y :: :. T (F) is the set of ground terms (without variables). The terms composed by a symbol of arity 0 are called constants; C is the set of constants of F. Positions in a term are represented as sequences of integers; denote the empty sequence. The top position of a term t is , and the symbol at the top position of t is written top(t). Let p and p0 be two positions. The position p is said to be pre x of p0 (and p0 suÆx of p) if p0 = p , where is a non empty sequence of integers. Given a term t, O(t) is the set of positions in t, inductively de ned as follows: O(t) = f g if t 2 X ; O(t) = f g[ fi:p j

*1 i n and p 2 O(ti)g if t = f (t1;::: ; tn). This set is partitioned into*

*O(t) = fp 2 O(t) j tjp 62 Xg and OV (t) = fp 2 O(t) j tjp 2 Xg where the* notation tjp stands for the subterm of t at position p. If p 2 O(t), then t[t0]p denotes the term obtained from t by replacing the subterm at position p by the term t0.

*A substitution is an assignment from X to T (F; X ), written = (x 7!* t) ::: (y 7! u). It uniquely extends to an endomorphism of T (F; X ). We identify a substitution = (x 7! t) ::: (y 7! u) with the nite set of equations (x = t) ^ ::: ^ (y = u). The result of applying to a term t 2 T (F; X ) is written (t) or t. The domain of , denoted Dom( ) is the nite subset of X such that x 6= x. The range of , denoted Ran( ), is de ned by Ran( ) =

*Sx2Dom( ) V ar( x). A ground substitution or instantiation is an assignment*

*from X to T (F). Id denotes the identity substitution. The composition of*

*substitutions 1 followed by 2 is denoted 2 1. Given two substitutions 1 and 2, we write 1 2 i 9 such that 2 = 1. Given a subset X1 of X , we note X1 for the restriction of to the variables of X1, i.e. the substitution such that Dom( X1 ) X1 and 8x 2 Dom( X1 ) : X1 x = x:*

*Given a set R of rewrite rules or term rewriting system on T (F; X ), a* function symbol in F is called a constructor if it does not occur in R at the top position of the left-hand side of a rule, and is called a de ned function symbol otherwise. The set of constructors of F for R is denoted by ConsR, the set of de ned function symbols of F for R is denoted by DefR (R is omitted when there is no ambiguity). The rewriting relation induced by R is called standard rewriting relation and is noted !R (! if there is no ambiguity on R). We note s !p;l!r; t (or s !p;l!r; t where either p or l ! r or may be omitted) if s rewrites into t at position p with the rule l ! r and the substitution . The term sjp is called a redex, the position p a redex position and the symbol in s at position p is called a redex symbol. The transitive (resp. re exive transitive) closure of the rewriting relation induced by R is

*denoted by !+ (resp. ! ). If it exists, the last term of a nite derivation*

*R R*

*starting from t is said to be in normal form, and is denoted by t#.*

*An ordering on T (F; X ) is said to be noetherian (or well-founded) i there is no in nite decreasing chain for this ordering. It is F-stable i for* any pair of terms t; t0 of T (F; X ), for any context f (::: :: :), t t0 implies f (::: t:: :) f (::: t0 :: :). It has the subterm property i for any t of T (F; X ), f (::: t:: :) t. Note that if is F-stable and has the subterm property, then it is noetherian. If, in addition, is stable by substitution (for any substitution , any pair of terms t; t0 2 T (F; X ); t t0 implies t t0), then it is called a simpli cation ordering. Let t be a term of T (F); let us recall that t terminates if and only if any rewriting derivation (or derivation chain) starting from t is nite.

# *3 Induction for termination with local strategies*

*We now tackle the termination problem for rewriting with local strategies on* operators, as expressed in [13] and studied in [7]. A local strategy is de ned in the following way.

*De nition 3.1 An LS rewriting strategy (or LS-strategy) on terms of T (F; X )* (resp. of T (F)) is a function LS from F to the set of lists of integers L(N ), de ning a rewriting strategy as follows.

*Given a LS-strategy such that LS(f ) = [p1;::: ; pk], pi 2 [0::arity(f )] for all i 2 [1::k], for some symbol f 2 F, normalizing a term t = f (t1;::: ; tm) 2 T (F; X ) (resp 2 T (F)) with respect to LS(f ) = [p1;::: ; pk], consists in normalizing all subterms of t at positions p1;::: ; pk successively, according to the strategy. If there exists i 2 [1::k] such that p1;::: ; pi 1 6= 0 and pi = 0 (0 is the top position), then*

*if the current term t0 obtained after normalizing tjp ; : : : ; tjp is reducible at the top position into a term g(u1; ::: ; un), then g(u1;::: ; un) is normalized with respect to LS(g) and the rest of the strategy [pi+1;::: ; pk] is ignored,*

*1 i 1*

*if t0 is not reducible at the top position, then t0 is normalized with respect to pi+1;::: ; pk.*

*At each rewriting step, the term t is said to LS-rewrite into a term t0. If* t does not rewrite for the LS-strategy, it is said to be in LS-normal form (or in normal form if there is no ambiguity). If any LS-rewriting chain starting from t leads to an LS-normal form then t is said to be LS-terminating (or to LS-terminate). If the evaluation strategy of a term t0 is the empty list, then t0 is in LS-normal form.

*In the following, we will use a notion expressing the possible reducible* positions of any instantiation of t, with respect to the LS-strategy.

*De nition 3.2 A position p of a term t 2 T (F; X ) is an LS-position in t if* the LS-strategy allows to rewrite t at position p, or if the LS-strategy allows to rewrite any ground instance of t at position p or at a suÆx position of p.

*The set LS P OS(t) of LS-positions of a term t can be computed in the* following way.

*LS P OS(f (u1;:::; un)) =*

*<8 f g Si2LS(f )n0fi:p j p 2 LS P OS(ui)g if 0 2 LS(f );*

*: Si2LS(f )fi:p j p 2 LS P OS(ui)g if 0 62 LS(f );*

*LS P OS(x*

*) = 8< f g if x 2 X ;*

*: ; if x 2 N :*

*3.1 Induction for local strategies*

*For proving that a term t of T (F) LS-terminates, we proceed by induction* on T (F) with a noetherian ordering (more precisely, an F-stable ordering having the subterm property), assuming that for any t0 such that t t0, t0 LS-terminates. We rst prove that a basic set of minimal elements for LS- terminates. As the subterm property for is required, the set of minimal elements is a subset of the set of constants of F.

*We then consider the case of any term t of T (F). For that, we observe* the rewriting derivation tree for the LS-strategy starting from a term tref = g(x1;::: ; xm), for any g 2 F, where x1;::: ; xm are induction variables that can be instantiated by any ground term. The LS rewriting relation on ground terms is simulated by the two mechanisms below to follow the derivation tree starting from tref , and whose current term is t. Let LS(top(t)) = [p1;::: ; pn], and pk the rst element of [p1;::: ; pn] such that pk = 0.

*First, the subterms tjp1 ;::: ; tjpk 1 of t have to be LS-normalized, by def-* inition of the above LS-strategy. If tref tjp1 ;::: ; tjpk 1 we can suppose, by induction hypothesis, that these subterms are LS-terminating. We then replace them in t by abstraction variables Xi representing respectively any of their normal forms ti#: these variables will only be instantiated by terms in normal form. Reasoning by induction allows us to only suppose the ex- istence of the ti# without explicitly computing them; this step will be called abstraction step or abstraction of the subterms of t. We also say that t is abstracted into a term v.

*Second, rewriting the resulting term v at position , following all possible* ground instantiations of v. This is computed by a narrowing step on v. Two cases may happen:

*if v is not narrowable at the top position, the subterms vjpk+1 ;::: ; vjpn of* v then have to be LS-normalized, and we try to abstract them like above;

*if v is narrowable at the top position, the narrowing step is computed* with all possible rules and all possible substitutions 1;::: ; l to give terms w1;::: ; wl, that have to be considered respectively with the strate- gies LS(top(w1));::: ; LS(top(wl)). So the two mechanisms above are again applied on the terms w1;::: ; wl. In addition, instances of v that are not considered by the narrowing have to be reduced at the positions

*pk+1;::: ; pn. So the two mechanisms described above are also applied on v at positions pk+1;::: ; pn for the instances of v that are not instances of*

*iv; i 2 [1::l].*

*The process stops on the current terms t having an empty LS-strategy or* on current terms the induction hypothesis can be applied on (i.e. such that tref t; in this case, t is supposed to be LS-terminating).

*Note that if there does not exist pk in fp1;::: ; png such that pk = 0, then* only the rst point is processed, abstracting every subterm tjpi of t; i 2 [1::n].

*3.2 Abstraction*

*We now give some new de nitions to formalize the above mechanisms. Ab-* straction needs the use of special variables representing LS-normal forms.

*De nition 3.3 Let N be a set of new variables disjoint from X . Symbols of* N are called NF-variables. Substitutions and instantiations are extended to T (F; X [ N ) in the following way. Let X 2 N ; for any substitution (resp. instantiation ) such that X 2 Dom( ), X (resp. X) is in normal form.

*Note that for abstracting the current term f (u1;::: ; um), it is not useful* to introduce an abstraction variable for the uj that are ground terms already in normal form, nor for the uj that are already NF-variables.

*De nition 3.4 The term f (u1;::: ; um) is abstracted into f (U1;::: ; Um) at positions fi1;::: ; ipg [1::m] if :*

*fi1;::: ; ipg are the positions of [1::m] such that ui1 ;::: ; uip are neither ground terms in normal form, nor NF-variables,*

*Uj = Xj where Xj is a fresh NF-variable, if j 2 fi1;::: ; ipg, Ui = ui otherwise.*

*We will prove LS-termination on T (F), reasoning on terms with abstrac-* tion variables, i.e. on terms of T (F; X [ N ).

*3.3 Constraints*

*Let us now de ne the di erent constraints needed by our proof process. Unlike* in classical approaches using induction, the induction ordering is not given a priori. Constraints are set along the proof, following the requirements appear- ing when induction hypotheses have to be applied. Such ordering constraints are cumulated in a set C and the satis ability of C is tested any time the induction hypotheses have to be applied.

*We now formally de ne the satis ability of ordering constraints.*

*De nition 3.5 An ordering constraint (t > t0) on terms of T (F; X [N ) is satis able if there exists an ordering and at least one instantiation such that t t0. We say that and satisfy (t > t0).*

*A conjunction C of ordering constraints is satis able if there exists an* ordering and an instantiation satisfying all conjuncts. The empty conjunction, always satis ed, is denoted by >.

*Along our induction process, when abstracting subterms ti by Xi, we state* constraints on NF-variables to express that their instances can only be the normal forms of the corresponding instances of the ti. They are of the form t# = X where t 2 T (F; X ), and X 2 N , or more generally of the form t# = t0 where t; t0 2 T (F; X [ N ). Let us call such a constraint an abstraction constraint.

*De nition 3.6 An abstraction constraint (t# = t0) where t; t0 2 T (F; X [ N ) is satis able if there exists at least one instantiation such that t# = t0. We say that satis es (t# = t0).*

*A constraint formula A is a formula of the form V (ti# = t0 ) V (W (xk 6=*

*i i j kj j*

*i*

*ukj )), xkj*

*2 X [ N ; ukj*

*2 T (F; X [N ) and the (ti# = t0 ) are abstraction*

*constraints. The empty formula is denoted >. A formula A is satis able if*

*there exists at least one instantiation such that V ( ti# = t0 ) V (W ( xk 6=*

*ukj )). We say that satis es A.*

*i i j kj j*

*In this paper, we consider constraint problems composed of 2-tuples (A; C)* where A is constraint formula and C is a conjunction of ordering constraints.

*De nition 3.7 Let A be a constraint formula and C a conjunction of ordering* constraints. The constraint problem (A; C) is satis ed by an ordering if A is satis able, and for all instantiations satisfying A, and satisfy C. (A; C) is satis able if A is satis able and there exists an ordering as above.

*Deciding the satis ability of (A; C) would require to express all instantia-* tions satisfying A. As we will see later, an interesting point of our method is that we do not need to characterize all those instantiations. It is enough to exhibit one of them to prove the satis ability of A. In such a case, a suÆcient condition for an ordering to satisfy (A; C) is that is stable by substitution (the induction ordering is then a simpli cation ordering) and t t0 for any inequality t > t0 of C.

*3.4 Narrowing*

*After the abstraction of the term f (u1;::: ; um) into f (U1; ::: ; Um) at po-* sitions fi1;::: ; ipg, where the uij are supposed to have a normal form uij #, and are replaced by abstraction variables Xij , we test whether the ground instances of f (U1;::: ; Um) are reducible with a case study on the syntactic form of the possible instantiations of the Xij . This test consists in narrowing f (U1;::: ; Um) at position with all possible substitutions instantiating the Xi only with irreducible terms, and all possible rewrite rules.

*Let us now recall the de nition of narrowing.*

*De nition 3.8 Let R be a TRS on T (F; X ). A term t is narrowed into t0,* at the non variable position p, using the rewrite rule l ! r of R and the substitution , when is a most general uni er of tjp and l, t0 = (t[r]p). This is denoted t ;p;l!r; t0 where either p, or l ! r or may be omitted. It is always assumed that there is no variable in common between the rule and the term, i.e. that V ar(l) \ V ar(t) = ;.

*R*

*The requirement of disjoint variables is easily ful lled by an appropriate* renaming of variables in the rules when narrowing is performed. Note that for the most general uni er used in the above de nition, Dom( ) V ar(l) [ V ar(t) and we can choose Ran( )\(V ar(l)[V ar(t)) = ;, thus introducing in the range of only fresh variables. Thus V ar(t) \ V ar(t0) = ; if in addition, variables of Var (t) Dom( ) are renamed through a substitution denoted

*ren.*

*As we will see below, in our proof process, we will also have to consider* the negation of a substitution.

*De nition 3.9 Let be a substitution on T (F; X [ N ) de ned by Vi(xi =*

*ti) xi 2 X [N , ti 2 T (F; X [N ). The negation of , denoted is the formula Wi(xi 6= ti).*

# *4 A rule-based algorithm*

*4.1 The inference rules*

*Inference rules describing our termination proof mechanism for local strategies* work on sets of 4-tuples T = (fug; [p1;::: ; pm]; A; C), where:

*fug is a set of terms of T (F; X [ N ), containing the current term u whose* ground instances have to be proved LS-terminating. This is either a single- ton or the empty set.

*[p1;::: ; pm] is the list of positions with respect to whom the current term u* has to be evaluated. This is a sublist of LS(top(u)).

*A is a constraint formula memorizing the abstractions and narrowing sub-* stitutions performed on the current term u. The sub-formulas of the form u# = X; u 2 T (F; X [N );X 2 N are stated each time a subterm u of the current term is abstracted by a new NF-variable X.

*C is a conjunction of ordering constraints completed by the abstraction* steps.

*Let us now present the inference rules.*

*The rule Abstract processes the abstracting step. It applies on (ff (u1;::: ; um)g; [p1;::: ; pn]; A; C), when there exists k 2 [2::n]; pk = 0 and p1;::: ; pk 1 6= 0. The term u = f (u1;::: ; um) is abstracted at positions i1;::: ; ip 2*

*fp1;::: ; pk 1g if there exists an F-stable ordering having the subterm prop-*

*erty and such that (A; C ^ tref > ui1 ;::: ; uip ) is satis able. Indeed, by*

*Abstract:*

*T [ f(ff (u1;:::; um)g; [p1;::: ; pn]; A; C)g*

*T [ f(ff (U1;:::; Um)g; [0; pk+1;::: ; pn]; A ^(ui# = Xi); C ^tref > ui)g*

*i i*

*where f (u1;:::; um) is abstracted by f (U1;::: ; Um) at the pos: i 2 fi1;:::; ipg fp1;:::; pk 1g*

*if 9k 2 [2::n] : p1;:::; pk 1 6 0; pk = 0 and (A; C*

*^*

*i2fi1 ;:::;ipg*

*tref > ui) is satis able*

*Abstract-Stop:*

*T [ f(ff (u1;::: ; um)g; [p1;::: ; pn]; A; C)g*

*T [ f(;; []; A; C*

*^*

*i2fi1 ;:::;ipg*

*tref > ui)g*

*where f (u1;:::; um) can be abstracted by f (U1;:::; Um) at the pos: i1;:::; ip 2 fp1;::: ; png*

*if p1;:::; pn 6= 0 and (A; C*

*^*

*i2fi1 ;:::;ipg*

*tref > ui) is satis able*

*Narrow-Y:*

*T [ f(ff (u1;:::; um)g; [0; p1;::: ; pn]; A; C)g*

*T [i2[1::l] f(fwig;*

*LS(top(wi)); A ^ i; iC)g[ COM P L*

*if 9 such that f (u1;:::; um)  ; w and A ^ is satis able*

*where wi;i 2 [1::l]; are all terms such that f (u1;:::; um)  ; i wi and A ^ i is satis able;*

*COM P L =*

*f(ff (u ;::: ;u )g; [p1::p ]; A*

*1*

*m*

*n*

*V*

*l*

*i=1*

*i; C)g if (A*

*V*

*l*

*i=1*

*) satis able*

*i*

*: ;*

*otherwise:*

*Narrow-N:*

*T [ f(ff (u1;::: ; um)g; [0; p1;:::; pn]; A; C)g*

*T [ f(ff (u1;::: ; um)g; [p1;:::; pn]; A; C)g*

*if f (u1;:::; um) is not narrowable at the top position*

*or 8 narrowing substitution of f (u1;::: ; um) at the top position ;A ^ is not satis able:*

*Stop-Ind:*

*T [ f(fug; [p1;:::; pn]; A; C)g*

*T [ f(;; []; A; C ^ tref > u)*

*if (A; C ^ tref > u) is satis able*

*Stop-A:*

*T [ f(ff (u1;::: ; um)g; [p1;::: ; pn]; A; C)g*

*T [ f(f]g; []; A; C)g*

*if p1 6= 0 and neither Abstract nor Abstract Stop applies:*

*Stop:*

*T [ f(ff (u1;::: ; um)g; []; A; C)g*

*T [ f(;; []; A; C)g*

*Table 1*

*Inference rules for tref LS-termination*

*induction hypothesis, all ground instances of ui1 ;::: ; uip LS-terminate. So ff (u1;::: ; um)g is replaced by f (U1;::: ; Um). The list of positions then becomes [0; pk+1;::: ; pn].*

*The rule Abstract Stop processes the abstracting step as above, when* there is no position 0 in the strategy of the current term. Any ground instance of the term obtained after abstraction is irreducible, by de nition of the LS-strategy, which ends the proof on the current derivation chain.

*The set containing the current term is then replaced by the empty set.*

*The rule Narrow Y processes the narrowing step at position 0 of the cur-* rent term u. If u is narrowable with a substitution satisfying the current constraint formula A, then u is narrowed in all possible ways in one step, with all possible rewrite rules of the rewrite system R, and all possible sub- stitutions i, into wi;i 2 [1::l]. Then (fug; [0; p1;::: ; pn]; A; C) is replaced by f(fwig; LS(top(wi));A ^ i; iC); i 2 [1::l]g, where i is the most general substitution allowing narrowing of u into terms wi. Moreover, since in A, we only memorize the abstractions and narrowing substitutions performed on the current term u, and since Var (u) is disjoint from the set of variables occuring in the rewrite rules of the TRS, we can restrict i to Var (u) in adding i to A. Thus, in the following, we will write i for iVar (u).

*This narrowing step means that 1u;::: ; lu are all instances of u that*

*are reducible at the top position. It involves that if = 1 ^ ::: ^*

*l is satis able, for each substitution satisfying , u is not reducible* at the top position. Then, as these u have to be reduced at positions [p1;::: ; pn], if is satis able, to the previous set we must add the set :

*(fug; [p1;::: ;p ];A Vl ; C)g. Note that if 9i such that*

*is just a re-*

*n i=1 i i*

*naming of variables, then = ;.*

*Let us also precise that if wi is a variable x 2 X , we cannot conclude* anything about termination of ground instances of x. So we force the proof process to stop in setting LS(x) to a particular symbol ]. However, if wi = X 2 N , LS(X) is set to [], which is coherent with the fact that any ground instance of X is in normal form.

*The rule Narrow N handles the case where u is not narrowable at position*

*0 or is narrowable with a substitution that does not satisfy the current* constraint formula A. Then no narrowing is processed and the current term is evaluated at positions following the top position in the strategy. The list of positions then becomes [p1;::: ; pn].

*We also can test for the current term whether there exists an ordering having* the subterm property such that (A; C ^ tref > u) is satis able. Then, by in- duction hypothesis, any ground instance of u terminates for the LS-strategy, which ends the proof on the current derivation chain. The Stop Ind rule then replaces the set containing the current term by the empty set.

*The rule Stop A allows to stop the inference process when neither Abstract* nor Abstract Stop applies, replacing u by the particular symbol ].

*The rule Stop allows to stop the inference process when the list of positions* is empty.

*The set of inference rules is given in Table 1.*

*Once Abstract is applied, the evaluation list's rst element is 0, so the* only rule that applies then is one of fNarrow Y, Narrow Ng. When Narrow Y does not apply, Narrow N applies. When Abstract does not apply, and the evaluation list's rst element is not 0, either Abstract Stop

*or Stop A applies, and then no rule applies anymore. The strategy for* applying these rules is:

*repeat\**

*((Abstract/ Abstract Stop/ Stop A);* (Narrow Y/ Narrow N);

*Stop; Stop Ind)*

*where ";" expresses the sequential application of the rules, r1/:: :/rn expresses* that the rules r1;::: ; rn are mutually exclusive and that one of them will be applied, and repeat\* (r1; :: :; rn) stops if none of the ri applies anymore.

*We write SU CCESS(g; ) if application of the inference rules on (fg(x1;*

*::: ; xm)g; LS(g); >; >), whose conditions are satis ed by , gives a state of* the form (;; []; A; C) on every branch of the derivation tree.

*Theorem 4.1 Let R be a TRS on T (F; X ), and LS : F 7! L(N ) a LS- strategy such that the constants of F LS-terminate. If there exists an F-stable ordering having the subterm property, such that for every non constant de ned symbol g, SU CCESS(g; ), then every term of T (F) LS-terminates.*

*Remark the important point that the ordering has to be the same for* all g(x1;::: ; xm) 2 Def . Remark also that the noetherian property of is implied by F-stability and the subterm property.

*For the proof of Theorem 4.1, as well as for the next lemmas, proposition* and theorems, see the Appendix.

*In the proof process, the information that variables are NF-variables can be* very important to conclude: if the current term is a NF-variable, its strategy is set to [] and the rule Stop applies. This information can be easily deduced when new variables are introduced: the abstracting process directly introduces NF-variables, by de nition.

*For the narrowing process, the narrowing substitution , whose range only* contains new variables of X , can be transformed in a new substitution NF by replacing some of these variables by NF-variables. Let us consider an equality of the form X = u, introduced by the narrowing substitution , where X is an NF-variable, and u 2 T (F; X ). As X is an NF-variable, any ground instance of u must be in normal form. So the variables in u that can be replaced by NF-variables are the variables that occur at an LS-position in u.

*Let now be the substitution (xi = Xi; 8xi 2 V ar(u); xi occurs at an LS position in u; f or all equation X = u of ; X 2 N ; u 2 T (F; X )). Then*

*NF = .*

*4.2 Extending the induction principle*

*When the induction hypothesis cannot be applied on a term u, the inductive* reasoning can be completed as follows. It can sometimes be possible to prove termination of any ground instance of ui (resp. u) by another way. Let TERM IN (u) be a predicate that is true i any ground instance of u LS-

*terminates. In Abstract, Abstract Stop and Stop Ind, we can then* replace the condition t > ui for some i (resp. t > u) by the alternative predicate TERM IN (ui) (resp. TERM IN (u)). Obviously, in this case, the ordering constraint t > ui (resp. t > u) is not added to C.

*As in [12], for establishing that TERM IN (u) is true, in some cases, the* notion of usable rules can be used. Given a TRS R on T (F; X ) and a term t 2 T (F; X [ N ), we determine the only rewrite rules that are likely to apply to any of its ground instances, for the standard rewriting relation, until its ground normal form is reached, if it exists. Then we try to nd a simpli cation ordering N so that these rules are oriented. Thus any ground instance t is bound to terminate for the standard rewriting relation: indeed, if t ! t1 ! t2 ! :: :, then, thanks to the previous hypotheses, t N t1 N t2 N ::: and, since the ordering N is noetherian, the rewriting chain cannot be in nite.

*More formally, given a TRS R, we call usable rules of a term t 2 T [N , as* in [2], a calculable superset U (t) of the set of rules of R used in all possible LS- derivations starting from any t, and de ned as follows. When t is a variable of X , then the usable rules of t are R itself. Likewise, the set of usable rules associated to a NF-variable is empty, since the only possible instances of sucha variable are ground terms in normal form. When t is of the form f (u1;::: ; un), then the usable rules of t are the usable rules of the ui, for i 2 LS(f );i 6= 0 and, if 0 2 LS(f ), all the rules l ! r having the symbol f as top symbol of lhs l, altogether with sets U (r) of the usable rules of the terms r.

*Since in general r contains variables of X , the evaluation of U (r) and then* of U (t) is likely to result in the whole set of rules R. To compute a smaller set of usable rules, like for the narrowing substitutions, we will replace variables by NF-variables as much as we can. The idea is that the usable rules of some variables in r can be omitted since they are included in the usable rules of the ui.

*More precisely, rewriting t with l ! r at the top position with the ground* substitution leads to the term r. Let x be a variable occuring in the lhs of l ! r at an LS-position p. By de nition of rewriting, we have x = tjp. Moreover, p is an LS-position in t, and if x occurs in r at position p0, we have

*rjp0 = tjp. In addition, we need U (x)( U (r)), only to consider the rules*

*that can apply in the derivations of rjp0 . But as rjp0 = tjp, and the rules* that can apply in the the derivations of tjp are included in U (ui) for some i, we can suppress the computation of the U (x) in the computation of U (r).

*A formal justi cation of these facts can be found in the proof of Lemma 4.3.*

*So, for computing the usable rules of t, we transform each rule l ! r of R into a rule NF (l ! r) = l ! r0 as follows : r0 = r where is the substitution (xi = Xi; xi 2 X ; Xi 2 N ; 8xi 2 V ar(r) and xi is at an LS position in l). We note NF (R) the set fNF (l ! r)jl ! r 2 Rg.*

*The formal de nition of the usable rules for LS-rewriting is then the fol-* lowing.

*De nition 4.2 Let R be a TRS on a set F of symbols. Let Rls(f ) = fl !* r 2 R j top(l) = f g, and Rls0(f ) = fl ! r0 2 NF (R) j top(l) = f g. For any t 2 T (F; X [ N ), the set of usable rules of t, denoted U (t), is de ned by:

*U (t) = R if t 2 X ,*

*U (t) = ; if t 2 N ,*

*>8 Rls(f ) Si2LS(f );i6=0 U (u ) Sl!r 2Rls (f ) U (r ) if 0 2 LS(f )*

*0*

*U (f (u1;::: ; un)) = <>*

*i 0 0*

*:>>*

*Si2LS(f );i6=0 U (ui) otherwise:*

*Lemma 4.3 Let R be a TRS on a set F of symbols, R0 = NF (R) and for each l ! r0 2 R0, no variable of X occurs in r0 at an LS-position. Let t 2*

*T (F; X [ N ). Whatever t ground instance of t and t !p1;l1!r1 t1 !p2;l2!r2 t2 ! ::: !pn;ln!rn tn LS-rewrite chain starting from t, then 8i 2 [1::n] : li ! ri 2 U (t).*

*We then can give a suÆcient criterion for ensuring termination for the* standard rewriting relation (and then LS-termination) of any ground instance of a term t.

*Proposition 4.4 Let R be a TRS on a set F of symbols, R0 = NF (R) and for each l ! r0 2 R0, no variable of X occurs in r0 at an LS-position. Let t 2 T (F; X [ N ). If there exists a simpli cation ordering such that 8l ! r 2 U (t) : l r, then any ground instance of t is terminating.*

*Remark that if there exists a simpli cation ordering such that l r for* any rewrite rule l ! r of a TRS R, then we have SU CCESS(g; 0), for any de ned symbol g 2 Def R and any F-stable ordering 0 having the subterm property. Therefore Theorem 4.1 applies. Indeed, for any tref = g(x1;::: ; xm), we have U (tref ) = R, with every rule oriented by . Finally, thanks to

*Proposition 4.4, we have TERM IN (tref ), and then Stop Ind applies.*

*An interesting point of this method is that the ordering that can be used* to orient the usable rules is completely independent of the induction ordering

*0.*

*Let us now illustrate our complete method on the example given in the* Introduction.

*Example 4.5 Recall the rules are:*

*f (i(x)) ! ite(zero(x); g(x);f (h(x))) zero(0) ! true*

*zero(s(x)) ! f alse*

*ite(true; x; y) ! x*

*ite(f alse; x; y) ! y*

*h(0) ! i(0)*

*h(x) ! s(i(x))*

*The LS-strategy is the following :*

*LS(ite) = [1; 0],*

*LS(f ) = LS(zero) = LS(h) = [1; 0] and*

*LS(g) = LS(i) = [1].*

*Let us prove the termination of this system on the signature F = ff : 1; zero : 1; ite : 3;h : 1;s : 1;i : 1;g : 1; 0 : 0g.*

*Obviously, the constant 0 LS-terminates. Applying the inference rules on* f (x1), we get :

*f (x1) [1; 0]*

*A = > C = >*

*Abstract*

*f (X1) [0]*

*A = (x1# = X1) C = (f (x1) > x1)*

*Abstract applies, since C is satis able by any ordering having the subterm* property. A is satis able with any instantiation such that x1 = X1 = 0.

*Narrow Y*

*ite(zero(X2); g(X2);f (h(X2))) = (X1 = i(x2) ^ x0 = x2)*

*NF = (X1 = i(X2) ^ x0 = X2) [1; 0]*

*A = (x1# = X1 ^ X1 = i(X2)) C = (f (x1) > x1)*

*f (X1) []*

*A = (x1# = X1) ^ (X1 6= i(X2)) C = (f (x1) > x1)*

*Note that x0 comes from the renaming of x in the rst rule. The renaming* of x2 into X2 comes from the fact that x2 occurs in i(x2) at an LS-position.

*Here, the rst constraint formula A is satis able by any instantiation such that X2 = 0 and x1 = i(0). The second constraint formula is satis ed* by any instantiation such that x1 = X1 = X2 = 0.

*Narrow Y expresses the fact that f (X1) is reducible if is such that*

*X1 = i(X2), and that the other instances ( 0f (X1) with 0X1 6= i(X2))*

*cannot be reduced.*

*Stop*

*ite(zero(X2); g(X2);f (h(X2))) [1; 0]*

*A = (x1# = X1 ^ X1 = i(X2)) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1) ^ (X1 6= i(X2)) C = (f (x1) > x1)*

*Stop applies, ending the second branch.*

*Abstract*

*ite(X3; g(X2);f (h(X2))) [0]*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1) ^ (X1 6= i(X2)) C = (f (x1) > x1)*

*The constraint formula A is satis able with any instantiation such that*

*X1 = i(0), X2 = 0, X3 = true and x1 = i(0).*

*Abstract applies here, since zero(X2) can be abstracted, thanks to Propo- sition 4.4. Indeed, U (zero(X2)) = fzero(0) ! true; zero(s(x)) ! f alseg, and both rules can be oriented by a LPO with the precedence zero F true and zero F f alse. Then we have TERM IN (zero(X2)).*

*Narrow Y*

*g(X4) = (X3 = true ^ X2 = x4 ^ x00 = g(x4) ^ y00 = f (h(x4)))*

*NF = (X3 = true ^ X2 = X4 ^ x00 = g(X4) ^ y00 = f (h(X4))) [1]*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = true*

*^ X2 = X4) C = (f (x1) > x1)*

*f (h(X4)) = (X3 = f alse ^ X2 = x4 ^ x00 = g(x4) ^ y00 = f (h(x4)))*

*NF = (X3 = f alse ^ X2 = X4 ^ x00 = g(X4) ^ y00 = f (h(X4))) [1; 0]*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3*

*^ X3 = f alse ^ X2 = X4) C = (f (x1) > x1)*

*ite(X3; g(X2);f (h(X2)) []*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3)*

*^ (X3 6= true ^ X3 6= f alse) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1) ^ (X1 6= i(X2)) C = (f (x1) > x1)*

*The rst constraint formula A is satis able by any instantiation such that*

*X4 = 0 and x1 = i(0). The second one is satis able by any instantiation such that X4 = s(0) and x1 = i(s(0)). The third one is satis able by any* instantiation such that X3 = zero(i(0)), X2 = i(0) and x1 = i(i(0)).

*The following step ends the third branch, whose strategy evaluation list is* empty.

*Stop*

*g(X4) [1]*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = true ^ X2 = X4) C = (f (x1) > x1)*

*f (h(X4)) [1; 0]*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = f alse ^ X2 = X4) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3) ^ (X3 6= true ^ X3 6= f alse) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1) ^ (X1 6= i(X2)) C = (f (x1) > x1)*

*Abstract (twice) g(X4) []*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = true*

*^ X2 = X4) C = (f (x1) > x1)*

*f (X5) [0]*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = f alse*

*^ X2 = X4 ^ h(X4)# = X5) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3) ^ (X3 6= true*

*^ X3 6= f alse) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1) ^ (X1 6= i(X2)) C = (f (x1) > x1)*

*Abstract trivially applies on g(X4) : since X4 is an NF-variable, there is* no need to abstract it.

*The second Abstract applies onf (h(X4)), thanks to Proposition 4.4. In-* deed, U (h(X4)) = fh(0) ! i(0); h(x) ! s(i(x))g, and both rules can be ori- ented by the same LPO as previously with the additional precedence h F i and h F s. Then we have TERM IN (h(X4)).

*The rst constraint formula has not changed, while the second one is now* satis able by any instantiation such that X5 = s(i(s(0))), X4 = s(0) and

*x1 = i(s(0)).*

*Narrow N (on the second branch) g(X4) []*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = true*

*^ X2 = X4) C = (f (x1) > x1)*

*f (X5) []*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = f alse*

*^ X2 = X4 ^ h(X4)# = X5) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3) ^ (X3 6= true*

*^ X3 6= f alse) C = (f (x1) > x1)*

*; []*

*A = (x1# = X1) ^ (X1 6= i(X2)) C = (f (x1) > x1)*

*One could have tried to narrow f (X5), by using the rst rule and the narrowing substitution NF = (X5 = i(X6) ^ x000 = X6). But then A ^ NF would be (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = f alse ^ X2 = X4 ^ h(X4)# = X5 ^ X5 = i(X6)). For any satisfying A ^ NF , must be such that h(X4)# = h( X4#)# = i( X6). If X4# 6= 0, then, according to R, h( X4#) ! s(i( X4#)), where s is a constructor. Then we cannot have h( X4#)# = i( X6), so must be such that X4# = 0. But then zero(X4)# = true, which makes A ^ NF unsatis ed. Therefore there is no narrowing.*

*The process is ended by a double application of Stop.*

*Stop (twice)*

|  |  |
| --- | --- |
| *;* | *[]* |
| *;* | *A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = true*  *^ X2 = X4) C = (f (x1) > x1)*  *[]* |
| *;* | *A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3 ^ X3 = f alse*  *^ X2 = X4 ^ h(X4)# = X5) C = (f (x1) > x1)*  *[]* |
|  | *A = (x1# = X1 ^ X1 = i(X2) ^ zero(X2)# = X3) ^ (X3 6= true*  *^ X3 6= f alse)* |
| *;* | *C = (f (x1) > x1)*  *[]* |
|  | *A = (x1# = X1) ^ (X1 6= i(X2))*  *C = (f (x1) > x1)* |

*Like for the de ned symbols ite; zero; h, the inference rules apply success-* fully through one Abstract, Narrow Y, Abstract without abstraction, Narrow N and Stop application. Therefore R is LS-terminating.

*Let us now give an example that cannot be handled with the context-* sensitive approach.

*Example 4.6 Let R be the following TRS*

*f (a; g(x)) ! f (a; h(x)) h(x) ! g(x)*

*with the LS-strategy : LS(f ) = [0; 1; 2]; LS(h) = [0] and LS(g) = [1].*

*Applying the rules on f (x1; x2), we get:*

*f (x1; x2) [0; 1; 2] A = > C = >*

*Narrow Y*

*f (a; h(x3)) [0; 1; 2]*

*= (x1 = a ^ x2 = g(x3) ^ x0 = x3) A = (x1 = a ^ x2 = g(x3))*

*C = >*

*f (x1; x2) [1; 2]*

*A = (x1 6= a \_ x2 6= g(x3)) C = >*

*Abstract Stop*

*f (a; h(x3)) [0; 1; 2]*

*A = (x1 = a ^ x2 = g(x3)) C = >*

*; []*

*A = (x1 6= a \_ x2 6= g(x3)) C = (f (x1; x2) > x1; x2)*

*Narrow N*

*f (a; h(x3)) [1; 2]*

*A = (x1 = a ^ x2 = g(x3)) C = >*

*; []*

*A = (x1 6= a \_ x2 6= g(x3)) C = (f (x1; x2) > x1; x2)*

*Abstract Stop*

*; []*

*A = (x1 = a ^ x2 = g(x3)) C = (f (x1; x2) > a; h(x3))*

*; []*

*A = (x1 6= a \_ x2 6= g(x3)) C = (f (x1; x2) > x1; x2)*

*Applying the rules on h(x1), we get:*

*h(x1) [0]*

*A = > C = >*

*Narrow Y*

*g(x2) [1]*

*= (x1 = x2 ^ x0 = x2)*

*A = (x1 = x2) C = >*

*Abstract*

*g(X) []*

*A = (x1 = x2 ^ x2# = X) C = (h(x1) > x2)*

*Stop*

*; []*

*A = (x1 = x2 ^ x2# = X) C = (h(x1) > x2)*

*Let us nally give another example, that cannot be be handled with* the context-sensitive approach:the TRS ff (b) ! c; g(x) ! h(x); h(c) ! g(f (a));a ! bg with the LS-strategy LS(f ) = [0; 1]; LS(g) = LS(h) = [1; 0]; LS(a) = [0], and two examples of innermost rewriting: the well-known Toyamas'example ff (0; 1; x) ! f (x; x; x); g(x; y) ! x; g(x; y) ! yg with the

*LS-strategy LS(f ) = [1; 2; 3; 0]; LS(g) = [1; 2; 0]; LS(0) = LS(1) = [0], and*

*ff (f (x)) ! f (f (x));f (a) ! ag with the LS-strategy LS(f ) = [1; 0]; LS(a) = [0], that is innermost terminating on T (F), but is not on T (F; X ). The* complete development of these examples can be found in the [9].

# *5 Conclusion*

*In this paper, we have proposed a method to prove termination of term rewrit-* ing with local strategies on operators by explicit induction on the termination property. Our method works on the ground term algebra using as induction relation an F-stable ordering having the subterm property. The general proof principle relies on the simple idea that for establishing termination of a ground term t, it is enough to suppose that terms smaller than t for this ordering are terminating, and that rewriting the context leads to terminating chains. It-

*erating this process until obtaining a context which is not reducible anymore* establishes the termination of t.

*More precisely, the method is applied on terms of the form g(x1;::: ; xm),* where g is a de ned symbol, and consists in iterating the application of two steps: an abstraction step, replacing immediate subterms by NF -variables, representing any of their normalized instances, and a narrowing step, reducing the resulting term according to the di erent possible instances of its variables. These two steps are iterated until getting a term for which one can easily say that all ground instances are terminating. The important point to automate our proof principle is the satisfaction of the ordering constraints for Abstract, Abstract Stop and Stop Ind.

*On many examples as those given in the paper, this is immediate since* they are ensured by the subterm property. In many other cases, a LPO is suÆcient to satisfy these constraints. Note that such an LPO does not suÆce when it is used in the classical way (any left-hand side of rule is greater than any corresponding right-hand side) since we can handle systems that are not terminating for standard rewriting. Testing satis ability of a constraint formula A remains simple to handle in practice.

*We now have a semi-automatic implementation of the inference rules and* the strategy for the leftmost innermost case, that can be expressed by a local strategy on operators. It has been implemented in ELAN [4], which is a logical environment for specifying and prototyping deduction systems in a rule based language with strategies. In this case, as any list of reduction positions ends with the top position, the formula A never contains disequations. SuÆcient conditions are implemented to detect unsatis ability of A, by identifying the reducible right-hand sides. The subterm property of the induction ordering to be found is also implemented, allowing the rst application of the rule Abstract to be completely automatic. Given a TRS, the program interacts with the user and builds the derivation tree resulting from the application of the inference rules according to the strategy we have de ned in this paper [8]. Execution examples are available. 1 We also have proposed a variant of the previous implementation, reducing the interaction with the user, by ignoring the satis ability problem of A. In this case, the obtained proof derivation tree contains the tree we would obtain in using A: states for which A is not satis-

*able just correspond to empty sets of ground terms. We thus have in general*

*more computations, but with considerably less user interactions. Moreover,* for many examples, the subterm property is the only required property on the induction ordering, so there are no user interactions to test the satis ability of C and the algorithm is completely automatic.

*Our process can also be extended to other strategies. We recently have* proposed inference rules for the outermost strategy [12]. Moreover, since our induction principle is based on the rewriting relation itself, the extension to

*1* [*http://www.loria.fr/~fissore/Demo/description.html*](http://www.loria.fr/~fissore/Demo/description.html)

*equational rewriting as well as to typed rewriting seems to be easy.*

*Acknowledgments: We would like to thank Salvador Lucas for his help-* ful comments on this work.

# *References*

*[1] T. Arts and J. Giesl. Proving innermost normalisation automatically. In Proceedings 8th Conference on Rewriting Techniques and Applications, Sitges (Spain), volume 1232 of Lecture Notes in Computer Science, pages 157{171. Springer-Verlag, 1997.*

*[2] T. Arts and Giesl. J. Proving innermost normalization automatically. Technical Report 96/39, Technische Hochschule Darmstadt, Germany, 1996.*

*[3] P. Borovansky , C. Kirchner, H. Kirchner, P.-E. Moreau, and Ch. Ringeissen. An Overview of ELAN. In C. Kirchner and H. Kirchner, editors, Proc. Second Intl. Workshop on Rewriting Logic and its Applications, Electronic Notes* *in Theoretical Computer Science, Pont- a-Mousson (France), September 1998.*

*Elsevier.*

*[4] Peter Borovansky , Claude Kirchner, H el ene Kirchner, Pierre-Etienne Moreau, and Christophe Ringeissen. An overview of ELAN. In Claude Kirchner and* *H el ene Kirchner, editors, Proceedings of the second International Workshop on Rewriting Logic and Applications, volume 15,* [*http://www.elsevier.nl/locate/entcs/volume15.html, Pont-*](http://www.elsevier.nl/locate/entcs/volume15.html%2CPont-) *a-* *Mousson (France), September 1998. Electronic Notes in Theoretical Computer Science. Report LORIA 98-R-316.*

*[5] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In J. Meseguer, editor, Proceedings of the 1st International Workshop on Rewriting Logic and its Applications, volume 5 of Electronic Notes in Theoretical Computer Science, Asilomar, Paci c Grove, CA, USA, September 1996. North Holland.*

*[6] Nachum Dershowitz and Jean-Pierre Jouannaud. Handbook of Theoretical Computer Science, volume B, chapter 6: Rewrite Systems, pages 244{320. Elsevier Science Publishers B. V. (North-Holland), 1990. Also as: Research report 478, LRI.*

*[7] S. Eker. Term rewriting with operator evaluation strategies. In C Kirchner and H. Kirchner, editors, Proceedings of the 2nd International Workshop on Rewriting Logic and its Applications, Pont-a-Mousson, France, September 1998.*

*[8] O. Fissore. Terminaison par induction. M emoire de DEA, Universit e Henri Poincar e { Nancy 1, June 2000.*

*[9] O. Fissore, I. Gnaedig, and H. Kirchner. Termination of rewriting with local strategies. Technical Report A01-R-177, LORIA, Nancy, France, 2001.*

*[10] K. Futatsugi and A. Nakagawa. An overview of CAFE speci cation environment { an algebraic approach for creating, verifying, and maintaining formal speci cations over networks. In Proceedings of the 1st IEEE Int.* *Conference on Formal Engineering Methods, 1997.*

*[11] J. Giesl and Middeldorp A. Transforming Context-Sensitive Rewrite Systems.* *In Proceedings of the 10th International Conference on Rewriting Techniques and Applications, volume 1631 of Lecture Notes in Computer Science, pages 271{285, Trento, Italy, 1999. Springer-Verlag.*

*[12] I. Gnaedig, H. Kirchner, and O. Fissore. Induction for innermost and outermost* *ground termination. Technical Report A01-R-178, LORIA, Nancy, France, 2001.*

*[13] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.P. Jouannaud.* *Introducing OBJ3. Technical report, Computer Science Laboratory, SRI International, march 1992.*

*[14] P. Klint. A meta-environment for generating programming environments. ACM Transactions on Software Engineering and Methodology, 2:176{201, 1993.*

*[15] M.R.K. Krishna Rao. Some characteristics of strong normalization. Theoretical Computer Science, 239:141{164, 2000.*

*[16] S. Lucas. Termination of context-sensitive rewriting by rewriting. In* *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, volume 1099 of Lecture Notes in Computer Science, pages 122{*

*133. Springer-Verlag, 1996.*

*[17] S. Lucas. Context-sensitive rewriting strategies. Technical Report DSIC-* *II/7/00, Departamento de Sistemas Informaticos y Computaci on, Universidad Polit ecnica de Valencia, Spain, 2000.*

*[18] Nakamura M. and Ogata K. The evaluation strategy for head normal form with and without on-demand ags. In K. Futatsugi, editor, Proceedings of the Third International Workshop on Rewriting Logic and its Applications, WRLA'2000, pages 211{227, Kanazawa City Cultural Halt, Kanazawa, Japan, September 2000. Electronic Notes in Theoretical Computer Science.*

*[19] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. Applicable Algebra in Engineering, Communication and Computation, 5(3 & 4):213{253, 1994.*

*[20] E. Visser. Stratego: A Language for Program Transformation based on Rewriting Strategies. System Description for Stratego 0.5. In Aart Middeldorp, editor, Proceedings of the 12th International Conference on Rewriting Techniques and Applications. Springer-Verlag, 2001. To appear.*

*[21] H. Zantema. Termination of context-sensitive rewriting. In Proceedings of the 8th International Conference on Rewriting Techniques and Applications, volume 1232 of Lecture Notes in Computer Science, pages 172{186. Springer-Verlag, 1997.*

# *Appendix*

*A The* [*usable*](#_bookmark9) *rules*

*To prove Lemma 4.3, we need the next ve lemmas. The rst one (Lemma A.1)* highlights the relationship between LS-positions and usable rules. Lemmas A.2 and A.3 are corollaries of this lemma.

*Lemma A.4 expresses the relationship between a rewrite rule and the usable*

*rules of the rhs, and the key Lemma A.5 explains where the symbols of the* redex positions come from.

*Lemma A.1 Whatever t 2 T (F; X [N ) and p 2 LS P OS(t), the usable rules of t contain the usable rules of tjp.*

*8t 2 T (F; X [N ); 8p 2 LS P OS(t) : U (tjp) U (t)*

*Proof.*

*We proceed by structural induction on t.*

*If t 2 N , then LS P OS(t) = ;, and the property is trivial.*

*If t 2 X [ C, then LS P OS(t) = f g and then the property is trivially* satis ed.

*If t = f (u1;::: ; un), we then have two cases :*

*either 0 2 LS(f ), and then LS P OS(t) = f g Si2LS(f )n0fi:p j p 2*

*LS P OS(ui)g.*

*if p = , then the property is trivial ;*

*if p 2 Si2LS(f )n0fi:p j p 2 LS P OS(ui)g, then 9j 2 LS(f );j 6=*

*0, such that p 2 LS P OS(uj), that is 9p0 2 LS P OS(uj) such*

*that p = j:p0. By induction hypothesis on uj, strict subterm of t, we* have the property : 8p 2 LS P OS(uj) : U (ujjp) U (uj). Then, since p 2 LS P OS(uj), we have U (ujjp0 ) U (uj), that is U (tjp) U (uj). Moreover, by de nition of the usable rules, we have : U (t) =

*0*

*Rls(f ) S U (u ) S U (r0), henceforth U (u ) U (t).*

*i2LS(f );i6=0 i l!r0 2Rls0 (f ) j*

*Therefore we get U (tjp) U (t).*

*or 0 62 LS(f ), and then LS P OS(t) = Si2LS(f )fi:p j p 2 LS P OS(ui)g. Since p 2 Si2LS(f )fi:p j p 2 LS P OS(ui)g, then 9j 2 LS(f ) such that*

*p 2 LS P OS(uj), that is 9p0 2 LS P OS(uj) such that p = j:p0. By*

*induction hypothesis on uj, strict subterm of t, we have the property :* 8p 2 LS P OS(uj) : U (ujjp) U (uj). Then, since p0 2 LS P OS(uj), we have U (ujjp0 ) U (uj), that is U (tjp) U (uj). Moreover, by de nition of

*the usable rules, we have : U (t) = Si2LS(f ) U (ui), henceforth U (uj) U (t).*

*Therefore we get U (tjp) U (t).*

*2*

*Lemma A.2 Let R be a TRS and t 2 T (F; X ). If a variable occurs in t at an LS-position, then U (t) = R.*

*Proof. This lemma is a corollary of the Lemma A.1. Indeed, if x occurs at* an LS-position p in t then, according to Lemma A.1, U (tjp) U (t). Since tjp = x 2 X , by de nition of the usable rules, we have U (tjp) = R, therefore R U (t). Since we also have U (t) R, we get U (t) = R. 2

*Lemma A.3 Let R be a TRS on a set F of symbols and t 2 T (F; X [ N ). Then, every symbol f 2 F occuring in t at a redex position is such that Rls(f ) U (t).*

*Proof. This lemma is also a corollary of Lemma A.1. Indeed, if f occurs in* t at a redex position p, then p 2 LS P OS(t). According to Lemma A.1, U (tjp) U (t). Since f is a redex symbol, we have 0 2 LS(f ), and then

*U (tj ) = Rls(f ) S U (u ) S U (r0). Consequently, we have*

*p i2LS(f );i6=0 i l!r0 2Rls0 (f )*

*Rls(f ) U (t). 2*

*Lemma A.4 Let R be a TRS on a set F of symbols, R0 = NF (R) and t 2 T (F; X [N ). For each l ! r 2 R, we note l ! r0 the rule NF (l ! r). Then l ! r 2 U (t) ) U (r0)* [*U*](#_bookmark32) *(t).*

*Proof. According to Lemma A.2, if a variable of X occurs in t at an LS-* position, then U (t) = R, and then the property is trivially true. We will then suppose in the following that t does not contain any variable of X at an LS-position.

*Let l ! r 2 U (t). By de nition of U (t), among all recursive applications* of the de nition of U in U (t), there is an application U (t0) of U to some

*term t0 such that U (t0) = Rls(g) S*

*U (t0j ) S*

*U (r0), with*

*i2LS(g);i6=0 i*

*U (t0) U (t), and l ! r 2 Rls(g), with g = top(l).*

*l!r0 2Rls0 (g)*

*Then we have l ! r0 2 Rls0(g), and then U (r0) 2 [l!r0 2Rls0 (g)U (r0), and then l ! r0 2 U (t0), and therefore l ! r0 2 U (t). 2*

*Lemma A.5 Let R be a TRS on a set F of symbols and R0 = NF (R), so that for each l ! r0 2 R0, no variable of X occurs in r0 at an LS-position. Let t 2 T (F; X [ N ) such that no variable of X occurs in t at an LS-position. Whatever ground substitution such that Var (t) Dom( ), and t !p1;l1!r1 t1 !p2;l2 !r2 t2 ! ::: !pn;ln !rn tn LS-rewrite chain starting from t, the symbol of tk; 1 k n at a redex position of tk is either a redex symbol of t or one of the ri;i 2 [1::k].*

*Proof. We proceed by induction on the length of the derivation.*

*For an empty derivation, the property has to be checked on t. In this case,* let us show that whatever x 2 X [ N variable of t, x does not introduce redex symbol in t. Let x be a variable of t occuring at position p.

*For x 2 X , by hypothesis, p is not an LS-position of t. Therefore t cannot*

*reduce at a suÆx position of p.*

*For x 2 N , by de nition of the NF-variables, X is normalized. Therefore*

*tjp has no redex symbol. Hence, the property is true for t.*

*Let us show the property for the rst rewriting step t !p1;l1!r1 t1. By de nition of the rewriting, 9 : l1 = tjp1 and t1 = t[ r1]p1 . Let f be a redex symbol of t1 :*

*either it is a symbol of t, and then we showed that f occurs in t at an* LS-position ;

*or it is a symbol of r1, and then :*

*either it belongs to r1, and the property is satis ed ;*

*or it belongs to x, for x 2 Var (r1). Since Var (r1) Var (l1), we have x 2 Var (l1).*

* *if x does not occur at an LS-position in l1, then x 2 Var (r01), which is in contradiction with the hypothesis of the current lemma ;*
* *if x occurs at an LS-position in l1, since l1 = tjp1 , then x is a subterm of t at an LS-position. Then, for a redex symbol f of x, f is a redex symbol of t. Then, as shown before, f is a redex symbol of t.*

*Let us now suppose the property true for any term of the LS-rewrite chain t0 = t !p1;l1!r1 t1 ! ::: !pk;lk !rk tk, and let us consider tk !pk+1;lk+1!rk+1 tk+1. We then have to show that every de ned symbol of tk+1 at a redex position comes either from t or from one of the ri;i 2 [1::k +1]. By hypothesis, there exists a substitution such that lk+1 = tkjpk+1 and tk+1 = tk[ rk+1]pk+1 .*

*>From the rst equality, we can deduce that 8x 2 Dom( ) \Var (lk+1); x is a*

*subterm of tkjpk+1 . Then, by induction hypothesis on the ground term tk, we* can infer that 8x 2 Dom( ) \Var (lk+1), the symbols at redex positions of x come either from t or from the ri;i 2 [1::k]. Henceforth, since Var (rk+1)

*Var (lk+1), the symbols at redex positions of rk+1 come either from t or from the ri;i 2 [1::k], or from rk+1. Then, by induction hypothesis on the ground term tk, the property is true at the* [*order*](#_bookmark9) *k +1 on tk+1 = tk[ rk+1]pk+1 . 2*

*We are now able to prove Lemma 4.3.*

*Lemma 4.3 Let R be a TRS on a set F of symbols, R0 = NF (R) and for each l ! r0 2 R0, no variable of X occurs in r0 at an LS-position. Let t 2 T (F; X [ N ). Whatever t ground instance of t and t !p1;l1!r1 t1 !p2;l2!r2 t2 ! ::: !pn;ln!rn tn LS-rewrite chain starting from t, then 8i 2 [1::n] : li ! ri 2 U (t).*

*Proof. If a variable x 2 X occurs in t at an LS-position then, thanks to* Lemma A.2, we have U (t) = R and the property is trivially true. We then consider in the following that the variables of X in t do not occur at an LS- position. We proceed by induction on the length of the derivation.

*The property is trivially true if t is in normal form. For any t !p*[*1;l1*](#_bookmark33)*!r1* t1, since no variable of t occurs at an LS-position, p1 is a non variable position of t. Let f be the symbol at position p1 in t. Then, thanks to Lemma A.3, Rls(f ) U (t). Moreover, since f is the symbol at the redex position, l1 ! r1 2 Rls(f ). Therefore we get l1 ! r1 2 U (t).

*Let us now suppose the property is true for any LS-derivation chain starting*

*from t whose length is less or equal to k, and consider the chain: t !p1;l1!r1 t1 !p2;l2 !r2 t2 ! ::: !pk;lk !rk tk !pk+1;lk+1!rk+1 tk+1. By Lemma A.5 with a derivation of length k, we have two cases:*

*either the symbol f at position pk+1 in tk is a symbol of t occuring at an LS-* position; then, thanks to Lemma A.3 on t, we get Rls(f ) U (t); moreover, since f is the symbol at the redex position, lk+1 ! rk+1 2 Rls(f ), henceforth lk+1 ! rk+1 2 U (t);

*or the symbol f at position pk+1 in tk is a symbol of a r0i;i 2 [1::k], occuring at an LS-position ; then, thanks to Lemma A.3 on r0i, we get Rls(f ) U (r0i); moreover, since f is the symbol at the redex position, lk+1 ! rk+1 2 Rls(f ), henceforth lk+1 ! rk+1 2 U (r0i); by induction hypothesis we have li ! ri 2 U (t) and, thanks to Lemma A.4, we have U (r0i) U (t). Henceforth lk+1 ! rk+1 2 U (t).*

*2*

*Proposition 4.4 Let R be a TRS on a set F of symbols, R0 = NF (R) and for each l ! r0 2 R0, no variable of X occurs in r0 at an LS-position. Let t 2 T (F; X [ N ). If there exists a simpli cation ordering such that 8l ! r 2 U (t) : l r, then any ground instance of t is terminating.*

*Proof. First, let us prove the following property:*

*8t; t0 2 T (F); [(9l ! r such that l r and a position p : t !p;l!r t0) ) t t0].*

*If t !p;l!r t0, then there exists a ground substitution such that l = tjp and* t0 = t[ r]p. Then comparing t with t0 comes to comparing t[tjp]p = t[ l]p with t[ r]p. Since l r and the ordering is stable by substitution, then we have

*l r. Then, since the ordering is F-stable, we get t[ l]p t[ r]p, that is* t t0.

*Let us then suppose 9 such that there exists an in nite rewrite chain*

*t ! t1 ! t2 ! :::*

*According to the hypotheses of the proposition and Lemma 4.3, is such that*

*t t1 t2 :: :, which is in contradiction with the noetherian property of* the ordering . 2

# *B The lifting* [*lemma*](#_bookmark7)

*In order to prove Theorem 4.1, we need a lifting lemma for LS-narrowing,* lying the following two propositions (the rst one is obvious).

*Proposition B.1 Let t 2 T (F; X ) and a substitution of T (F; X ). Then V ar( t) = (V ar(t) Dom( )) [ Ran( V ar(t)).*

*Given a set of variables V and two substitutions and , we write = [V ]* i 8x 2 V : x = x.

*Proposition B.2 Suppose we have substitutions ; ; and sets A; B of vari- ables such that (B Dom( )) [ Ran( ) A. If = [A] then = [B].*

*Proof.*

*Let us consider ( )B, which can be divided as follows : ( )B = ( )B\Dom( ) [ ( )B Dom( ).*

*For x 2 B \ Dom( ), we have Var ( x) Ran( ), and then ( )x = ( x) =*

*Ran( )( x) = ( Ran( ) )x. Therefore ( )B\Dom( ) = ( Ran( ) )B\Dom( ).*

*For x 2 B Dom( ), we have x = x, and then ( )x = ( x) = x.*

*Therefore we have ( )B Dom( ) = B Dom( ). Henceforth we get ( )B = ( Ran( ) )B\Dom( ) [ B Dom( ).*

*By a similar reasoning, we get ( )B = ( Ran( ) )B\Dom( ) [ B Dom( ).*

*By hypothesis, we have Ran( ) A and = [A], then we can infer Ran( ) =*

*Ran( ). Likewise, since B Dom( ) A, we have BDom( ) = B Dom( ). Then we have ( )B = ( Ran( ) )B\Dom( ) [ B Dom( ) = ( Ran( ) )B\Dom( ) [*

*B Dom( ) = ( )B with the assumptions used in the second equality. There-*

*fore ( ) = ( )[B]. 2*

*Lemma B.3 Let R be a TRS. Let s 2 T (F; X ), a ground substitution such that s is reducible at a position p 2 O(s), and Y X fVar (l) j l ! r 2 Rg a set of variables such that V ar(s) [ Dom( ) Y. If s !p;l!r t0, then there exist a term s0 2 T (F; X ) and substitutions ; such that :*

*1: s ;p;l!r; s0; 2: s0 = t0;*

*3: = [Y]:*

*Proof. The proof is essentially borrowed (hardly adapted) from [19].*

*If s !p;l!r t0, then there exists a substitution such that Dom( )*

*Var (l) and ( s)jp = l. Moreover, since p 2 O(s), then ( s)jp = (sjp).*

*Let = [ . Then we have : (sjp) = (sjp) = l = l, therefore sjp and l are uni able. Let us note the most general uni er of sjp and l, and s0 = (s[r]p). Therefore, by de nition : s ;[p;l!r; ] s0, and then the point 1.*

*of the current lemma holds.*

*Since , there exists a substitution such that = . Let Y1 = (Y Dom( )) [ Ran( ). We de ne = Y1 . Clearly Dom( ) Y1.*

*We now want to show that Var (s0) Y1, by the following reasoning :*

*by de nition, s0 = (s[r]p), therefore we have Var (s0) = Var ( (s[r]p));*

*the rule l ! r is such that Var (r) Var (l), therefore we have Var ( (s[r]p))*

*Var ( (s[l]p)), and then, thanks to the previous point, Var (s0) Var ( (s[l]p));*

*by de nition, we have (s[l]p) = s[ l]p and, since uni es l and sjp, we get*

*(s[l]p) = s[ (sjp)]p = s[sjp]p = (s) and, thanks to the previous point :*

*Var (s0) Var ( (s));*

*according to Proposition B.1, we have Var ( (s)) = (Var (s) Dom( )) [ Ran( Var(s)); by de nition, Var (s) Y and Ran( Var(s)) Ran( ), there- fore Var ( (s)) (Y Dom( )) [ Ran( ), that is Var ( (s)) Y1. There- fore, with the previous point, we get V ar(s0) Y1.*

*>From Dom( ) Y1 and V ar(s0) Y1, we can infer Dom( ) [V ar(s0) Y1.*

*We are now going to demonstrate the point 2., that is s0 = t0.*

*By de nition, = Y , therefore = [Y1]. Since Dom( ) [ V ar(s0) Y1, we get s0 = s0. By de nition, s0 = (s[r]p) = (s[r]p) = s[ r]p.*

*1*

*By hypothesis, we have Dom( ) Var (l) and Y \ Var (l) = ;, then we have Y \ Dom( ) = ;. Therefore, from = [ , we get = [Y]. Since*

*Var (s) Y, we get s = s.*

*Likewise, by hypothesis we have Dom( ) Y, Var (r) Var (l) and Y \ Dom( ) = ;, then we get V ar(r) \ Dom( ) = ;, and then we have =*

*[V ar(r)], and therefore r = r.*

*>From s = s and r = r we get s[ r]p = s[ r]p. Since, by hypothesis,*

*s !p t0, with l = ( s)jp, then s[ r]p = t0. Finally, we get s0 = t0 (2).*

*Next we show that = [Y] (point 3. of the current lemma). Reminding* that Y1 = (Y Dom( )) [ Ran( ), Proposition B.2 (with the notations A for Y1, B for Y, for , for and for ) yields = [Y]. We already noticed that = [Y]. Linking these two equalities via the equation = yields = [Y] (3). 2

# *C The correctness theorem*

*Theorem 4.1 Let R be a TRS on T (F; X ), and LS : F 7! L(N ) a LS-strategy such that the constants of F LS-terminate. If there exists an F-stable ordering*

*having the subterm property, such that for every non constant de ned symbol g, SU CCESS(g; ), then every term of T (F) LS-terminates.*

*The proof of Theorem 4.1 is given for the rules extended with the TERM IN* predicate, as introduced in Section 4.2. These rules are given in Table C.1.

*Proof.*

*We prove by induction on T (F) that any ground instance f (x1;::: ; xm)* of any term f (x1;::: ; xm) 2 T (F; X ) LS-terminates. The induction ordering is constrained along the proof. At the beginning, it has at least to be F-stable and to have the subterm property, which ensures its well-foundedness. Such an ordering always exists on T (F) (for instance the embedding relation). Let us denote it .

*The minimal elements of T (F) for LS-terminate since, by hypothesis,* the constants of F LS-terminate, and for any ordering on ground terms having the subterm property, the set of minimal elements is a subset of the set of constants.

*Abstract:*

*T [ f(ff (u1;:::; um)g; [p1;:::; pn]; A; C)g*

*T [ f(ff (U1;:::; Um)g; [0; pk+1;::: ; pn]; A ^(ui# = Xi); C ^H(ui))g*

*i i*

*where f (u1;:::; um) is abstracted by f (U1;::: ; Um) at the pos: i 2 fi1;:::; ipg fp1;:::; pk 1g*

*if 9k 2 [2::n] : p1;:::; pk 1 6 0; pk = 0 and (A; C ^ tref > uk1 ;::: ; ukl ) is satis able for fk1;:::; klg fi1;:::; ipg and TERM IN (uj) for j 2 fi1;:::; ipgn fk1;::: klg*

*where H(ui) = true if TERM IN (ui); tref > ui otherwise:*

*Abstract-Stop:*

*T [ f(ff (u1;:::; um)g; [p1;:::; pn]; A; C)g*

*T [ f(;; []; A; C*

*^*

*i2fi1 ;:::;ipg*

*H(ui))g*

*where f (u1;:::; um) can be abstracted by f (U1;:::; Um) at the pos: i1;:::; ip 2 fp1;::: ; png*

*if p1;:::; pn 6= 0 and (A; C ^ tref > uk1 ;:::; ukl ) is satis able for fk1;::: ; klg fi1;::: ; ipg and TERM IN (ui) for i 2 fi1;:::; ipgn fk1;::: klg*

*where H(ui) = true if TERM IN (ui); tref > ui otherwise:*

*Narrow-Y:*

*T [ f(ff (u1;::: ; um)g; [0; p1;:::; pn]; A; C)g*

*T [i2[1::l] f(fwig;*

*LS(top(wi)); A ^ i; iC)g[ COM P L*

*if 9 such that f (u1;:::; um)  ; w and A ^ is satis able*

*where wi;i 2 [1::l]; are all terms such that f (u1;:::; um)  ; i wi and A ^ i is satis able:*

*COM P L =*

*f(ff (u ;::: ;u )g; [p1::p ]; A*

*1*

*m*

*n*

*V*

*l*

*i=1*

*i; C)g if (A*

*V*

*l*

*i=1*

*) satis able*

*i*

*: ;*

*otherwise:*

*Narrow-N:*

*T [ f(ff (u1;:::; um)g; [0; p1;::: ; pn]; A; C)g*

*T [ f(ff (u1;:::; um)g; [p1;::: ; pn]; A; C)g*

*if f (u1;:::; um) is not narrowable at the top position*

*or 8 narrowing substitution of f (u1;::: ; um) at the top position ;A ^ is not satis able:*

*Stop-Ind:*

*T [ f(fug; [p1;::: ; pn]; A; C)g*

*T [ f(;; []; A; C ^ H(u))g*

*if (A; C ^ tref > u) is satis able or TERM IN (u):*

*where H(u) = true if TERM IN (u); tref > u otherwise:*

*StopA:*

*T [ f(ff (u1;:::; um)g; [p1;:::; pn]; A; C)g*

*T [ f(f]g; []; A; C)g*

*if p1 6= 0 and neither Abstract nor Abstract Stop applies:*

*Stop:*

*T [ f(ff (u1;:::; um)g; []; A; C)g*

*T [ f(;; []; A; C)g*

*Table C.1*

*Inference rules for tref LS-termination*

*By subterm property of , we have f (x1;::: ; xm) = f ( x1;::: ; xm)*

*xi1 ;::: ; xip , with fi1;::: ; ipg = fijj ij 2 fp1;::: png; ij =6 0g where [p1;::: ; pn]*

*= LS(f ). Then, by induction hypothesis, let us suppose that xi1 ;::: ; xip*

*LS-terminate. Let xi1 #;::: ; xip # be respectively any of their normal forms.*

*If f is a constructor, then f ( x1;::: ; xm) is irreducible at the top position, as well as its reduced forms. Then f ( x1;::: ; xm)# = f ( x1;::: ; xm)[ xi1 #]i1*

*::: [ xip #]ip . Therefore f ( x1;::: ; xm) is terminating for the LS-rewriting*

*relation.*

*If f is not a constructor, let us denote it g and prove that g( x1;::: ; xm)* LS-terminates for any satisfying (A0 = >; C0 = >), if application of the inference rules on

*(fg(x1;::: ; xm)g; strat(g); >; >) terminates on states (;; []; Ap; Cp). Let us*

*denote g(x1;::: ; xm) by tref in the sequel of the proof.*

*To each step of the procedure characterized by (ftg; l; A; C), we associate* the set of ground terms G = f t j and satisfy (A; C)g. Inference rules Abstract, Narrow Y and Narrow N transform (ftg; l; A; C) into (ft0g; l0; A0; C0) to which is associated G0 = f t0 j and satisfy (A0; C0)g. We then prove the following result: if for all t0 2 G0, t0 LS-terminates wrt l0, then any t in G LS-terminates wrt l.

*Either Abstract is applied, so ff (u1;::: ; um)g becomes ff (U1;::: ; Um)g and l = [p1;::: ; pn] becomes l0 = [0; pk+1;::: ; pn]. For each such that t is in G, we prove that there exists a such that t0 is in G0 and such that LS-termination of t0 wrt l0 implies LS-termination of t wrt l.*

*According to the condition of Abstract, can be chosen to be such*

*that for any satisfying A, tref uk1 ;::: ; ukl for some k1;::: ; kl 2*

*fi1;::: ; ipg fp1; ::: ; pk 1g, where i1;::: ; ip are the positions at which*

*f (u1;::: ; um) is abstracted. So by induction hypothesis, the ukj LS- terminate. Moreover, for i 2 fi1;::: ; ipgnfk1;::: ; klg, we have TERM IN (ui) and then ui LS-terminates. Finally, for i 2 fp1;::: ; pk 1g n fi1;::: ; ipg, ui is either a ground term in normal form or an NF-variable, and then ui LS-terminates.*

*Then let us de ne = [ fXp1 = up1 #;::: ; Xpk 1 = upk 1 #g. Clearly*

*satis es (A0; C0). Moreover, f (U1; :::; Um) = f (u1; :::; um)[ up #]p :::*

*1 1*

*[ upk #]pk .*

*Therefore, LS-termination of the f (U1;::: ; Um) wrt [0; pk+1;::: ; pn], for*

*all possible normal forms up1 #;::: ; upk 1 # of up1 ;::: ; upk 1 , implies LS- termination of f (u1;::: ; um) wrt [p1;::: ; pk 1; 0; pk+1;::: ; pn].*

*Or Narrow Y is applied on ff (u1;::: ; um)g with l = [0; p1;::: ; pn]. For any satisfying (A; C),*

*either f (u1;::: ; um) is irreducible at the top position, but may be re- duced at the positions p1;::: ; pn. So termination of the term f (u1;::: ; um), where f (u1;::: ; um) is produced by Narrow Y together with the strat- egy [p1;::: ; pn] implies termination of f (u1;::: ; um) with the strategy [0; p1;::: ; pn].*

*Let us now prove that satis es (A Vl ; C). In this case, t0 = t.*

*i=1*

*i*

*Thus, since A Vl*

*i=1*

*i characterizes the set of instantiations such that*

*f (u ;::: ;u ) is irreducible, satis es A Vl . Finally, as t0 = t*

*1 m i=1 i*

*and satis es C, also satis es C.*

*or f (u1;::: ; um) is reducible at the top position. In this case, if f (u1;*

*::: ; um) ! t0, thanks to Lemma B.3, there exists a narrowing step* f (u1;::: ; um); v and a substitution such that t0 = v. Moreover, this narrowing derivation is e ectively produced by Narrow Y, which is applied in all possible ways on ff (u1;::: ; um)g. Then LS-termination of the v implies LS-termination of f (u1;::: ; um).

*Let us now prove that satis es (A ^ ; C). On variables of f (u1;::: ;*

*um), we have = . In addition, the domain of , that is the range* of , can be extended to the variables of A ^ and C by setting x =

*x for x 2 (Var (A) [ V ar(C)) n V ar(f (u1;::: ; um)). So = on V ar(A ^ ) [ V ar(C), and then (A ^ ) = A ^ = A ^ . As*

*satis es A, A is true. Moreover, as A only contains ground terms,*

*A ^ is true. So satis es A ^ . Finally, since C = C and satis es C, then satis es C.*

*Or Narrow N is applied on f (u1;::: ; um). For any satisfying (A; C),*

*f (u1;::: ; um) is irreducible at the top position. Then, by taking = (since A0 = A and C0 = C), LS-termination of f (u1;::: ; um) wrt [p1;::: ; pn] is equivalent to LS-termination of f (u1;::: ; um) wrt [0; p1;::: ; pn].*

*Let us now prove that the ground instances satisfying (A; C) of each term* t removed from T during the application of the rules LS-terminate. The rst rule removing terms from T is Stop. When Stop is applied and removes t from T , then the evaluation strategy of t is []. Then, for any ground substitution satisfying A and C, t LS-terminates.

*The second rule removing terms from T is Abstract Stop. According* to the condition of Abstract Stop, can be chosen to be such that for any

*satisfying A, tref uk1 ;::: ; ukl for some k1;::: ; kl 2 fi1;::: ; ipg fp1;*

*::: ; png, where i1;::: ; ip are the positions f (u1;::: ; um) is abstracted at. So by*

*induction hypothesis, the ukj LS-terminate. Moreover, for i 2 fi1;::: ; ipgn fk1;::: ; klg, we have TERM IN (ui) and then ui LS-terminates. Finally, for i 2 fp1;::: ; png n fi1;::: ; ipg, ui is either a ground term in normal form or an NF-variable, and then ui LS-terminates. Therefore, by de nition of the evaluation strategy of t, t LS-terminates.*

*The third rule removing terms from T is Stop Ind. When Stop Ind is* applied and removes t from T , then for any ground substitution satisfying A,

*either the ordering is such that tref t and then, by induction hy-* pothesis, t LS-terminates,

*or we have TERM IN (t), and then t LS-terminates too.*

*As the process is initialized with ftref g and the constraint problem (A; C) =*

*(>; >) satis able by any ground substitution, we get that g( x1;::: ; xm) is LS-terminating, for any tref = g(x1;::: ; xm), and any ground instance .*

*Moreover, as the terms f (x1;::: ; xm), where f is a constructor are also* LS-terminating for any ground instances x1;::: ; xm and the constants are LS-terminating, then any term of T (F) is LS-terminating. 2