Contents

An Improved General Path Order

Alfons GESER
Universität Passau, Lehrstuhl für Programmiersysteme, D-94030 Passau
Phone: +49 851 509 3094, E-mail: geser@fmi.uni-passau.de

July 7, 1995

Abstract

Dershowitz/Hoot's "general path order" is a powerful method to prove termination of term rewriting systems. We simplify and strengthen the general path order, and demonstrate the power of the improved version by a proof of termination of a medium-scale term rewriting system that models a lift control.

List of Keywords: Term rewriting system, Termination, semantic path ordering, general path ordering.

1 Introduction

A term rewriting system R is called terminating if there is no infinite derivation

$$t \to_R t' \to_R t'' \to_R \cdots \ .$$

Termination of term rewriting systems is the key to effective term rewriting. The following properties do not hold for an arbitrary rewrite system, but hold for every finite, terminating term rewriting system R.

- every term can safely be rewritten into a R-normal form,
- the *reachability* problem (given t, t', does $t \to_R^* t'$ hold?) is decidable,
- the principle of *rewriting induction* is valid, i.e. Noetherian induction with the transitive closure $\to_R^+$ of the rewrite relation as inductive order. Newman's proof of confluence by local confluence for terminating rewrite systems admits a short presentation as a proof by rewriting induction [21]. Rewriting induction is an elegant proof technique, encoded in the "proof by consistency" method.
- If moreover confluence holds then the simple word problem is solvable, and normal forms are unique. For the class of finite, terminating rewrite systems, confluence is decidable by local confluence of critical pairs [28].

As is known, a rewrite system terminates if and only if, there is a termination order for it, i.e. if there is a transitive and terminating binary relation, closed under substitution and contexts, that contains ("orders") each rule of the rewrite system. Termination orders are an essential ingredient in the Knuth/Bendix completion procedure [28] which tries to convert a given a set of equations into a terminating, confluent rewrite system.

Termination of rewrite systems is known to be undecidable [22, 6]. So the challenge is to design termination orders that are powerful enough for practical application.

Besides the interpretation orders, the path orders are most widely used to prove termination of term rewriting systems. The *multiset path order* [8] to begin with, uses a *precedence*, i.e. a wellfounded quasiorder on function symbols, to split cases where arguments are compared recursively, or collections of arguments are recursively compared as multisets.

Collections of arguments may be compared recursively also in other ways, an observation that has led to the notion of status mapping. A *status* is a functional STAT that maps binary relations $>$ on ground terms to binary relations $\mathrm{STAT}(>)$ on ground terms, and satisfies a number of technical properties: It has to preserve orders, be monotonic and continuous w.r.t. the subset relation, and satisfy

$$s >_{spo} t \quad \Longrightarrow \quad f(\ldots, s, \ldots) \;\; \mathrm{STAT}(>_{spo}) \;\; f(\ldots, t, \ldots) \ . \tag{1}$$

For instance lexicographic comparison of the tuples of arguments, permuted according to the top function symbol, is a status. Thus Kamin/Lévy defined the *recursive path order (with status)* [25].

If the subterms of a term are compared recursively, two equal subterms cannot be distinguished. As soon as the contexts of the subterms are considered, this restriction does no longer hold. Following this idea, the recursive decomposition order [24, 31], the path of subterms order [33], the KNS path order [26], have been designed, to name a few.

An independent strain of generalization has arisen from the desire to involve semantic arguments into the comparison. The basic idea is that a strict subterm of a term t, i.e. a term *syntactically* smaller than t, may though be *semantically* greater than t. For instance in a recursive definition of the factorial function `fac` one would like to employ the property $n+1 > n+1-1$ on natural numbers to compare $\mathtt{fac}(s(x)) > \mathtt{fac}(p(s(x)))$. The semantics is expressed by a *model*, i.e. a value-preserving congruence $\sim_{[\_]}$ of R. Here a congruence is an equivalence closed under contexts; value-preserving means $R \subseteq \sim_{[\_]}$. Such a congruence is typically defined by a homomorphic interpretation $[\_]$ of ground terms into a domain $\mathcal{D}$ of values.

Plaisted (as mentioned in [11]) defined the *value-preserving path order* to compare two terms first by precedence and then lexicographically by a wellfounded order on the interpretations of the arguments. Kamin/Lévy [25] extended this non-recursive order towards an order where after the precedence and the comparison of the interpretations of arguments there may be still a recursive comparison of the arguments according to a status map. This became known as the *semantic path order* $>_{spo}$. It was Kamin/Lévy's great contribution to show that although $>_{spo}$ is usually *not* closed under contexts, it is so when accompanied by a rewrite step, i.e. the relation $>_{spo} \cap \rightarrow_R$ is closed under contexts. To this end, they require the condition[1]

$$s \rightarrow_R t \quad \Longrightarrow \quad f(\ldots, s, \ldots) \gtrsim_\phi f(\ldots, t, \ldots) \ . \tag{C}$$

Formally similar to the semantic path order, in the *Knuth/Bendix order* [28] two terms are compared first by their weights (i.e. weighted sums of nonnegative numbers assigned to each function symbol) then by the precedence, and then recursively their arguments in a lexicographic way. Lankford [29] replaced the weights by strictly monotonic polynomials with positive integer coefficients; Dershowitz [8] extended the Knuth/Bendix order by allowing instead of weights any monotonic interpretation that has the (weak) subterm property.

The *general path order* $>_{gpo}$ [10, 11] deserves its name for its ability to cover all known path orders and all extensions of the Knuth/Bendix order. In contrast to the semantic path order, the general path order has no fixed order of comparisons. Rather, recursive comparisons, precedences, and semantic comparisons may be mixed. Mimicking the Knuth/Bendix order, it may first use a monotonic interpretation that has the subterm property. This interpretation may be expressed also as a tuple, lexicographically ordered, of monotonic interpretations, all except the last one strictly monotonic, which have the strict subterm property.

[1] Originally, "$\sim_\phi$". An unknown reader of Kamin/Lévy's manuscript has remarked that it can be relaxed to "$\gtrsim_\phi$".

This is the spot where we can offer three basic improvements which both simplify the framework and strengthen the capabilities of the general path order. We call the new order the *improved general path order* [19].

Two changes are drawn from Zantema's "semantic labelling" approach [36].

First, the requirement of a model can be relaxed to that of a quasimodel. A *quasimodel* is a quasiorder $\gtrsim_{[\_]}$ on ground terms, closed under contexts, such that $R \subseteq \gtrsim_{[\_]}$. Again such a quasiorder is typically induced by a monotonic interpretation $[\_]$ to a quasi-ordered domain of values $(\mathcal{D}, \gtrsim_{\mathcal{D}})$.

Second, Zantema introduces the notion of a labelling function. A *labelling function* is a function π that maps a ground term into a terminating, quasi-ordered set $(\mathcal{E}, \gtrsim_{\mathcal{E}})$ of labels. The top function symbol of a term t is now decorated with its label $\pi(t)$, the purpose being that a precedence can be tuned much finer on labelled function symbols than on symbols that are not labelled. Most noticeable, labelling functions are based on monotonic interpretations but need not be interpretations themselves.

As semantic labelling and semantic path order are closely related, the formalism of labelling function can be carried over to the semantic path order. Rather than labellings we speak of *measures* here, and use the letter ϕ. The semantic comparison is done by $\gtrsim_\phi$.

Like the interpretation functions, the measure functions have to be monotonic. In effect, a measure ϕ induces a quasiorder $\gtrsim_\phi$ on ground terms that satisfies

$$s \gtrsim_{[\_]} t \quad \Longrightarrow \quad f(\ldots, s, \ldots) \gtrsim_\phi f(\ldots, t, \ldots)\ . \tag{2}$$

Admittedly this is nothing but a concretisation of the semantic path order. However it is also an instruction to build effective instances of the semantic path order. To see that semantic path order is an instance, choose $\to_R^+$ for $\gtrsim_{[\_]}$. Then Condition (2) follows from Condition (C). Conversely, if $l\sigma >_{spo} r\sigma$ and $l\sigma \gtrsim_{[\_]} r\sigma$ hold for every rule $l \to r$ in a given rewrite system R, and every ground substitution σ, which is nothing but a proof that R is ordered by $>_{spo}$ and that the interpretation is a quasimodel of R, respectively, then Condition (C) is satisfied. Hence Kamin/Lévy's proof method is exactly an encoding of a proof that R is ordered by the terminating order $>_{spo} \cap \gtrsim_{[\_]}$ that is closed under contexts [18]. The semantic path order may be treated technically as simple as other termination orders.

A straightforward consequence, the Extended Knuth/Bendix order is a special case of the semantic path order (Theorem 5 in [18]). Dershowitz/Hoot's *natural path order* [11] is another special case of the semantic path order, where the quasimodel is even a model and the status is empty.

We take the improved general path order as a further simplified form of semantic path order where the semantic comparison $\gtrsim_\phi$ is no longer a distinguished part of the definition, but instead becomes some component of the status functional. For this however Condition (1) is too restrictive. And here is the third spot where we put an essential change.

Our proof that $>_{gpo} \cap \gtrsim_{[\_]}$ is closed under contexts is general and simple. It uses the sufficient condition

$$s >_{gpo} t \wedge s \gtrsim_{[\_]} t \quad \Longrightarrow \quad f(\ldots, s, \ldots) \quad \mathrm{STAT}(>_{gpo}) \quad f(\ldots, t, \ldots) \tag{3}$$

which is clearly weaker than (1). It is weak enough above all, to enable, as components in a status functional, any number of measure comparisons $\gtrsim_{\phi_1}, \ldots, \gtrsim_{\phi_n}$ that satisfy Condition (2) each. This capability is definitely not shared by the semantic path order, and is not present in this generality in Dershowitz/Hoot's general path order.

As we can demonstrate at some examples, the distinction between measure functions and interpretation is an essential progress from Dershowitz/Hoot's order. Monotonicity of the interpretation and of the measure functions are natural conditions easy to satisfy; even trivial when $\gtrsim_{\mathcal{D}}$ is the equality on $\mathcal{D}$. This relieves the designer from the hardest monotonicity proof obligations.

After the preliminaries (Section 2), we define the notions of interpretation, of status, and of the improved general path order and prove a general abstract theorem (Section 3). Thereafter we outline a toolbox for status components (Section 4). Section 5 gives a check-list for developers of improved general path orders and enumerates some small examples for illustration, including a special application for conditional term rewriting systems. In Appendix B we demonstrate that improved general path order can handle a realistic 41-rule term rewriting system that formalizes a simple lift control [16].

2 Preliminaries

We assume that the reader is familiar with term rewriting, in particular with termination orders. For surveys on term rewriting see [23, 2, 27, 12, 32]. For notation see [13]. A comprehensive survey on termination of term rewriting systems is [9].

Let $\mathcal{D}$ be any countable set, and let $\rightarrow \subseteq \mathcal{D}^2$ be a binary relation on $\mathcal{D}$. A $\rightarrow$-derivation is a sequence of steps $t_1 \rightarrow t_2 \rightarrow \dots$, that may be finite or infinite. If there is no infinite $\rightarrow$-derivation then $\rightarrow$ is said to *terminate* or to be *terminating*.

A binary relation on $\mathcal{D}$ is called an *order* if it is irreflexive and transitive, and a *quasiorder* if it is reflexive and transitive. A quasiorder $\gtrsim$ defines its *strict part*, $>$, an order, by $s > t$, if $s \gtrsim t$ and $t \not\gtrsim s$, and its *equivalence kernel*, $\sim$, an equivalence relation, by $s \sim t$, if $s \gtrsim t$ and $t \gtrsim s$. We say that $\gtrsim$ *strictly* satisfies a property P if both $\gtrsim$ and $>$ satisfy P. We call a quasiorder $\gtrsim$ terminating if $>$ is terminating.

The reflexive, transitive closure of $\rightarrow$ will be denoted by $\rightarrow^*$.

Let two disjoint sets $\mathcal{F}$ of *function symbols* and $\mathcal{X}$ of *variables* be preassumed together with a function $\mathsf{arity} : \mathcal{F} \rightarrow \mathbb{N}$ which assigns each function symbol its fixed number of arguments. The set $\mathcal{T}$ of *terms* upon $\mathcal{F}$ and $\mathcal{X}$ is defined to be the smallest set containing $\mathcal{X}$ and satisfying

$$\mathsf{arity}(f) = n \quad \text{and} \quad t_1, \dots, t_n \in \mathcal{T} \quad \text{implies} \quad (f, t_1, \dots, t_n) \in \mathcal{T}$$

Function symbols f may also be seen as term constructing functions $f : \mathcal{T}^n \rightarrow \mathcal{T}$ by $f(t_1, \dots, t_n) = (f, t_1, \dots, t_n)$, a fact that allows to scrap the tuple notation. A *ground* term is a term that contains no variable. $\mathcal{GT}$ denotes the set of all ground terms.

A *substitution* σ is a function $\sigma : \mathcal{T} \rightarrow \mathcal{T}$ that satisfies $f(t_1, \dots, t_n)\sigma = f(t_1\sigma, \dots, t_n\sigma)$ for every function symbol $f \in \mathcal{F}$. Application of a substitution is denoted by postfixing the substitution. Because it is a homomorphism, σ is uniquely given by its restriction to the mapping $\sigma : \mathcal{X} \rightarrow \mathcal{T}$. A ground substitution is a substitution that maps every variable to a ground term.

A binary relation, $\rightarrow \subseteq \mathcal{T}^2$, on terms is called *closed under contexts*, if $s \rightarrow t \Longrightarrow c(s) \rightarrow c(t)$ holds for all terms s, t, and contexts $c(\_)$. If $\rightarrow$ is transitive then it is sufficient to show $s \rightarrow t \Longrightarrow f(\dots, s, \dots) \rightarrow f(\dots, t, \dots)$ for all terms s and t, for every function symbol $f \in \mathcal{F}$, and for every argument position i, $1 \le i \le \mathsf{arity}(f)$ of $\_$ in $f(\dots, \_, \dots)$. The dots are a convenient abbreviation for a sequence of terms $u_1, \dots, u_{i-1}, u_{i+1}, \dots, u_n$. The relation $\rightarrow$ is called *closed under substitution*, if $s \rightarrow t \Longrightarrow s\sigma \rightarrow t\sigma$ holds for every substitution σ, and for all terms s and t.

Given two binary relations $\gtrsim_1$, $\gtrsim_2$, their *lexicographic combination*, $(\gtrsim_1, \gtrsim_2)_{lex}$, is the binary relation $>_1 \cup (\sim_1 \cap \gtrsim_2)$. It is known that lexicographic combination preserves reflexivity, transitivity, termination, and strict closure under contexts and under substitution.

A quasiorder $\gtrsim$ which is closed under contexts, by symmetry entails that its equivalence kernel $\sim$ is closed under contexts: $s \sim t \Longrightarrow f(\dots, s, \dots) \sim f(\dots, t, \dots)$. Such an equivalence is also called a *congruence*. Be aware that there exist quasiorders who are closed under contexts, but not strictly. Recall that this means that their strict part, $>$, is not closed under contexts. For this reason lexicographic combination does not preserve closure under contexts, unless strict. To witness, let $a >_1 b$, $f^n(a) \sim_1 f^n(b)$, and $f^n(b) >_2 f^n(a)$ for all $n \geq 1$. Then, $a > b$, but $f(b) > f(a)$. The same holds for closure under substitutions.

A *term rewriting system* R (*rewrite system* for short) is any (usually finite) binary relation on terms. Its elements are written $l \to r$, and are also called *(rewrite) rules.* The *R-rewrite relation* $\to_R \subseteq \mathcal{T}^2$ is defined as the closure of R under contexts and under substitution. It is well-known that the rewrite step (given R, t, wanted some t' such that $t \to_R t'$ holds) is computable. R is called a *terminating rewrite system*, if its rewrite relation $\to_R$ terminates. An equational rewrite system is a pair (R, E) of rewrite systems where E is symmetric. The elements of E are written $l \equiv r$, and are also called *equations.* R is called E-terminating, if $\to_E^* \to_R \to_E^*$ terminates.

A *termination quasiorder* $\gtrsim$ is a terminating quasiorder on terms which is strictly closed under contexts and under substitution. A termination quasiorder is the basis to prove (equational) termination of a rewrite system: R is E-terminating, if and only if, there is some termination quasiorder $\gtrsim$ such that both $R \subseteq >$ and $E \subseteq \sim$ hold. I.e. one has to prove $l > r$ for all rules $(l \to r) \in R$ and $l \sim r$ for all equations $(l \equiv r) \in E$.

Termination quasiorders are useful even if E is empty, as the strictorder may profit from the equivalence kernel.

Example 1. *Let f and g have multiset status. Then*

$$f(g(x,y), f(y,y)) >_{gpo} f(g(y,x), y)$$

follows from

$$(g(x,y), f(y,y)) >_{\mathrm{gpo,mult}} (g(y,x), y) \quad .$$

To show this, $g(x,y) \sim_{gpo} g(y,x)$ is essential. The same conclusion applies when f instead has lexicographic status "left-to-right".

A term s is said to be a *superterm* of a term t, formally $s \trianglerighteq t$, or equivalently, t is a *subterm* of s, ($t \trianglelefteq s$), if s is of the form $c(t)$ for some context c. The superterm relation is a quasiorder. A binary relation on terms $\to \supseteq \triangleright$ is also said to have the *subterm property.* A quasiorder $\gtrsim$ that has the subterm property and is closed under contexts is called a *simplification quasiorder*[2]. Each simplification quasiorder by definition extends the relation $\to_{\triangleright}^*$, the converse of the *embedding* relation. By Kruskal's tree theorem, every simplification quasiorder terminates if F is finite.

A rewrite system that terminates by a simplification order is called *simply terminating.* If it terminates by a termination order that is total on ground terms, then it is called *totally terminating.*

3 Architecture and Ingredients

We are going to define an enhanced version of Dershowitz/Hoot's general path order as a liberalized form of Kamin/Lévy's semantic path order. The order is given as the strict part $>_{gpo} =_{\mathrm{def}} \gtrsim_{gpo} \setminus \lesssim_{gpo}$ of a quasiorder $\gtrsim_{gpo}$. We will establish the quasiorder $\gtrsim_{gpo}$ on the set of ground terms, and lift it to a quasiorder on terms with variables that is strictly closed under substitution.

For the quasiorder $\gtrsim_{gpo}$ on ground terms we will give a recursive definition scheme which is based on a scheme parameter, STAT. The functional

$$\mathrm{STAT} : \mathfrak{P}(\mathcal{GT}^2) \to \mathfrak{P}(\mathcal{GT}^2)$$

maps each binary relation $\gtrsim \subseteq \mathcal{GT}^2$ on ground terms towards a binary relation $\mathrm{STAT}(\gtrsim) \subseteq \mathcal{GT}^2$ on ground terms. The functional STAT is there to express how two terms may be

[2] quasi-simplification ordering in [9].

compared, given the results of comparisons of any pair of their proper subterms. This is the most general way to see the path order idea.

From the semantic path order approach we adopt the following plan. To construct an instance of $\gtrsim_{gpo}$, one has to define a *monotonic interpretation*, $[\_]$, and a *status*, STAT, i.e. a functional that satisfies a number of conditions. Unlike in the semantic path order, semantic comparisons will not be an extra ingredient but part of the functional.

3.1 Monotonic Interpretations

It is well-known how to establish a quasiorder which is closed under contexts. Let a set $\mathcal{D}$ be given, provided with a quasiorder $\gtrsim_{\mathcal{D}}$, and a homomorphism

$$[\_] : \mathcal{GT} \to \mathcal{D}$$

which assigns to each ground term t its interpretation $[t]$. The homomorphism $[\_]$ is conveniently given by a $\mathcal{F}$-sorted family of functions $[f] : \mathcal{D}^{\mathsf{arity}(f)} \to \mathcal{D}$, via

$$[f(s_1, \ldots, s_m)] =_{\text{def}} [f]([s_1], \ldots, [s_m]) \ .$$

The homomorphism $[\_]$ induces a quasiorder, $\gtrsim_{[\_]} \subseteq \mathcal{T}^2$, on terms, defined by

$$s \gtrsim_{[\_]} t \quad \Longleftrightarrow_{\text{def}} \quad [s\sigma] \gtrsim_{\mathcal{D}} [t\sigma] \text{ for all ground substitutions } \sigma \ .$$

If each of the $[f]$ is $\gtrsim_{\mathcal{D}}$-monotonic (in every argument), then $\gtrsim_{[\_]}$ is closed under contexts. In this case, $(\mathcal{D}, \gtrsim_{\mathcal{D}}, [\_])$ is also called a *monotonic interpretation*. A monotonic interpretation that satisfies $R \subseteq \gtrsim_{[\_]}$ is called a *quasimodel* of R [36].

It means no restriction to require $\gtrsim_{\mathcal{D}}$ antisymmetric since $\mathcal{D}$ can be partitioned into $\sim_{\mathcal{D}}$-equivalence classes without any change for the termination proof. If $\gtrsim_{\mathcal{D}} = =$ is the equality on $\mathcal{D}$ then monotonicity becomes trivially satisfied by the homomorphism property. In this case a quasimodel is called a *model*.

3.2 Statuses

In order to ensure that $\gtrsim_{gpo}$ satisfies the properties of termination and quasiorder, the functional STAT should essentially preserves these properties. Kamin/Lévy [25] gave a list of conditions for STAT to satisfy. Pierre Lescanne [30] coined the notion of *status* for a functional that satisfies the conditions.

In contrast to Kamin/Lévy, we set up the notion of status for quasiorders. Exceptionally, we will write $\mathrm{STAT}(>)$ not for $\mathrm{STAT}(\gtrsim \setminus \lesssim)$ for which we have no need, but for the relation $\mathrm{STAT}(\gtrsim) \setminus \mathrm{STAT}(\lesssim)$. Likewise $\mathrm{STAT}(\sim)$ denotes $\mathrm{STAT}(\gtrsim) \cap \mathrm{STAT}(\lesssim)$.

Definition 3.1 (Status, STAT, [25], [18]). *Let $\gtrsim_{[\_]} \subseteq \mathcal{GT}^2$ be a quasiorder on ground terms, closed under contexts. A functional*

$$\mathrm{STAT} : \mathfrak{P}(\mathcal{GT}^2) \to \mathfrak{P}(\mathcal{GT}^2)$$

is called a status*, if it satisfies the following conditions:*

- STAT *preserves quasiorders,*
- STAT *is* subterm-founded*, i.e. for every pair s, t, of ground terms, and every binary relation $\gtrsim$ on ground terms,*

$$s \ \ \mathrm{STAT}(\gtrsim) \ \ t \quad \Longleftrightarrow \quad s \ \ \mathrm{STAT}(\gtrsim') \ \ t$$

where $\gtrsim'$ is defined by

$$s' \gtrsim' t' \quad \Longleftrightarrow_{\text{def}} \quad s' \gtrsim t' \wedge (s', t') \lhd_{mult} (s, t) \ .$$

- STAT *is* strictly prepared for contexts*:*

$$\begin{array}{lclr} s \gtrsim t \wedge s \gtrsim_{[\_]} t & \Longrightarrow & f(\dots,s,\dots) \ \ \mathrm{STAT}(\gtrsim) \ \ f(\dots,t,\dots) & \textit{and} \\ s > t \wedge s \gtrsim_{[\_]} t & \Longrightarrow & f(\dots,s,\dots) \ \ \mathrm{STAT}(>) \ \ f(\dots,t,\dots) & (*) \end{array}$$

- STAT decreases infinite derivations*, i.e. for every infinite derivation*

$$t^1 \ \ \mathrm{STAT}(>) \ \ t^2 \ \ \mathrm{STAT}(>) \ \ \cdots$$

there is an infinite derivation $u^1 > u^2 > \cdots$ *such that* $t^j \rhd u^1$ *holds for some index* j*.*

In view of a compositional notion of status, we put an extra definition to distinguish *components* of a status. Status components are expected to satisfy all conditions for a status except $(*)$, the strict part of prepredness for contexts. This is similar to Dershowitz/Hoot's policy. But in contrast, we will care that the final assembly of a number of status components is strictly prepared for contexts. For this purpose we retain where a component is already strictly prepared.

Definition 3.2 (Status component; strictly prepared for (f,i)**).** *Let* $\gtrsim_{[\_]} \subseteq \mathcal{GT}^2$ *be a quasiorder, closed under contexts, and* STAT : $\mathfrak{P}(\mathcal{GT}^2) \to \mathfrak{P}(\mathcal{GT}^2)$ *a status except for the condition* $(*)$*. Then* STAT *is called a* status component*. Moreover* STAT *is called* strictly prepared for the pair (f,i)*, if*

$$s > t \wedge s \gtrsim_{[\_]} t \quad \Longrightarrow \quad f(\dots,s,\dots) \ \ \mathrm{STAT}(>) \ \ f(\dots,t,\dots)$$

holds, where $f \in \mathcal{F}$ *and* i *denotes the position of* $\_$ *within* $f(\dots,\_,\dots)$*. If* S *is a set of pairs* (f,i)*, then* STAT *is called* strictly prepared for S*, if it is strictly prepared for each element of* S*.*

Now if STAT is a status component which is strictly prepared for every pair (f,i), $f \in \mathcal{F}$, $1 \le i \le \mathsf{arity}(f)$, then STAT satisfies $(*)$, i.e. is a status.

The following property is useful for reasoning in proofs below.

Lemma 3.1. *If* STAT *is subterm-founded, then for all terms* s*,* t*, and binary relations* $\gtrsim$ *on terms,*

$$\begin{array}{lcll} s \ \ \mathrm{STAT}(>) \ \ t & \Longleftrightarrow & s \ \ \mathrm{STAT}(>') \ \ t & \textit{and} \\ s \ \ \mathrm{STAT}(\sim) \ \ t & \Longleftrightarrow & s \ \ \mathrm{STAT}(\sim') \ \ t \ , & \end{array}$$

where $\gtrsim'$ *is defined by*

$$s' \gtrsim' t' \quad \Longleftrightarrow_{\mathrm{def}} \quad s' \gtrsim t' \wedge (s',t') \lhd_{mult} (s,t) \ .$$

The **proof** is easy; we just remark that $(s,t) \ =_{mult} \ (t,s)$ holds. □

Kamin/Lévy put the condition that STAT is monotonic and continuous w.r.t. $\subseteq$. Continuity means informally that for each comparison s $\mathrm{STAT}(\gtrsim)$ t, only a finite number of pairs $s' \gtrsim t'$ need to be examined. We drop monotonicity, and instead of continuity we require a condition that we call *subterm foundedness*. In contrast to continuity, subterm foundedness restricts the set of term pairs not only to a finite set, but to the set of terms which are smaller w.r.t. the multiset extension of the subterm relation. This set is certainly finite.

Though we prefer subterm foundedness for the following reasons.

- Subterm foundedness allows for structural inductive proofs of properties of $\gtrsim_{gpo}$, whereas otherwise one has to employ fixpoint induction.

- With subterm foundedness one is guaranteed that $\gtrsim_{gpo}$ is defined everywhere. Technically convenient, this implies the well-definedness of $>_{gpo}$ as the strict part of $\gtrsim_{gpo}$.
- With terminating algorithms for $\gtrsim_{[\_]}$ and STAT given, $\gtrsim_{gpo}$ becomes a terminating algorithm for ground terms.
- In all known applications subterm foundedness is satisfied. Indeed subterm foundedness can still be relaxed further, replacing in its definition $\trianglerighteq$ by any terminating quasiorder which has the subterm property.

3.3 The Improved General Path Order

Now we are ready to express $\gtrsim_{gpo}$ as unique fixed point of a recursive scheme with the status component STAT as scheme parameter.

Definition 3.3 (Improved general path order, $\gtrsim_{gpo}$, [10, 11]). *For each functional* STAT : $\mathfrak{P}(\mathcal{GT}^2) \to \mathfrak{P}(\mathcal{GT}^2)$, *its induced* improved general path quasiorder $\gtrsim_{gpo} \subseteq \mathcal{GT}^2$ *is defined as follows.*

$s \gtrsim_{gpo} t$, *if* $s = f(s_1, \ldots, s_m)$, $\quad t = g(t_1, \ldots, t_n)$, *and*

1. $\forall i.\ s >_{gpo} t_i, \quad s \ \mathrm{STAT}(\gtrsim_{gpo})\ t$, *or*
2. $\exists i.\ s_i \gtrsim_{gpo} t$

The improved general path order is lifted to terms that may contain variables by

$s \gtrsim_{gpo} t \quad \Longleftrightarrow_{\mathrm{def}}$

- $s\sigma >_{gpo} t\sigma$ for all ground substitutions σ, or
- $s\sigma \sim_{gpo} t\sigma$ for all ground substitutions σ.

Theorem 1 (Main theorem). *If* $\gtrsim_{[\_]} \subseteq \mathcal{T}^2$ *is a quasiorder on terms, closed under contexts and under substitution, and* STAT : $\mathfrak{P}(\mathcal{GT}^2) \to \mathfrak{P}(\mathcal{GT}^2)$ *a status, then*

1. $\gtrsim_{gpo}$ *is a terminating quasiorder that has the strict subterm property, and*
2. $(>_{gpo} \cap \gtrsim_{[\_]}) \cup (\sim_{gpo} \cap \sim_{[\_]})$ *is a termination quasiorder.*

Proof. *The lifting of both* $\gtrsim_{gpo}$ *and* $(>_{gpo} \cap \gtrsim_{[\_]}) \cup (\sim_{gpo} \cap \sim_{[\_]})$ *from ground terms to terms that may contain variables is by construction strictly closed under substitution. Furthermore it preserves quasiorder, termination, the strict subterm property, and strict closure under contexts. We may therefore resort to the case of ground terms.*

The proof of (1) *is rather technical and less interesting. See Appendix A.1.*

For (2), *quasiorder and termination follow immediately from* (1). *Notice that to maintain termination from* $>_{gpo}$, *the problematic part* $\sim_{gpo} \cap >_{[\_]}$ *had to be cut off* $\gtrsim_{gpo} \cap \gtrsim_{[\_]}$.

We still have to prove that $\gtrsim_{gpo} \cap \gtrsim_{[\_]}$ *and* $>_{gpo} \cap \gtrsim_{[\_]}$ *are closed under contexts. The proof is similar to the one in [18] for a variant of the semantic path order. It is surprisingly simple.*

To show that $\gtrsim_{gpo} \cap \gtrsim_{[\_]}$ *is closed under contexts let* $s \gtrsim_{gpo} t$ *and* $s \gtrsim_{[\_]} t$ *where* $s, t \in \mathcal{GT}$. *Since* $\gtrsim_{[\_]}$ *is closed under contexts,* $f(\ldots, s, \ldots) \gtrsim_{[\_]} f(\ldots, t, \ldots)$ *holds. This leaves to show* $f(\ldots, s, \ldots) \gtrsim_{gpo} f(\ldots, t, \ldots)$. *We do it using Case* (1) *of the definition of* $\gtrsim_{gpo}$.

1. *We have* $f(\ldots, s, \ldots) >_{gpo} s \gtrsim_{gpo} t$, *by the strict subterm property of* $\gtrsim_{gpo}$. *Transitivity of* $\gtrsim_{gpo}$ *yields* $f(\ldots, s, \ldots) >_{gpo} t$.
2. $f(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_n) >_{gpo} u_j$ *for all* $j \neq i$ *holds by the strict subterm property of* $\gtrsim_{gpo}$.

3. $f(\ldots, s, \ldots)$ $\mathrm{STAT}(\gtrsim_{gpo})$ $f(\ldots, t, \ldots)$ *follows from the premises* $s \gtrsim_{gpo} t$ *and* $s \gtrsim_{[\_]} t$ *by the fact that the status component* STAT *is prepared for contexts.*

Likewise, $s >_{gpo} t$ *and* $s \gtrsim_{[\_]} t$ *implies* $f(\ldots, s, \ldots) >_{gpo} f(\ldots, t, \ldots)$. *To show that* $f(\ldots, s, \ldots)$ $\mathrm{STAT}(>_{gpo})$ $f(\ldots, t, \ldots)$, *one uses the fact that* STAT *is strictly prepared for contexts.*

This finishes the proof that $\gtrsim_{gpo} \cap \gtrsim_{[\_]}$ *and* $>_{gpo} \cap \gtrsim_{[\_]}$ *are closed under contexts. Together,* $(>_{gpo} \cap \gtrsim_{[\_]}) \cup (\sim_{gpo} \cap \sim_{[\_]})$ *is strictly closed under contexts. So* $(>_{gpo} \cap \gtrsim_{[\_]}) \cup (\sim_{gpo} \cap \sim_{[\_]})$ *is a termination quasiorder, and the proof of* (2) *is finished.*

4 A Toolbox for Status Components

A decisive question for the strength the improved general path order is: Which forms of status components are available? Among the various ways to define status components, we pick a few which we consider most important. We are going to introduce measures as constant status components; lexicographic composition of status components; intersection of status components; the selection of an argument at a specified position; and restriction to a set of function symbols. Dershowitz/Hoot [11] have investigated multisets of specified arguments, and multisets of arguments of a specified rank w.r.t. $>_{gpo}$.

4.1 Measures

A particular status component is one where there is no reference to its parameter at all: Constant status components. Measure comparisons $\gtrsim_\phi$ form such status components. Let $\mathcal{E}$ be a set, ordered by the terminating quasiorder $\gtrsim_{\mathcal{E}}$. A *measure* is then expressed by a function

$$\phi : \mathcal{GT} \to \mathcal{E}$$

that resorts to the arguments of a term only via their interpretation. Such a function is conveniently given by a $\mathcal{F}$-sorted family of functions $\phi^f : \mathcal{D}^{\mathsf{arity}(f)} \to \mathcal{E}$, by defining

$$\phi(f(s_1, \ldots, s_m)) =_{\mathrm{def}} \phi^f([s_1], \ldots, [s_m])$$

The induced quasiorder, $\gtrsim_\phi \subseteq \mathcal{T}^2$, on terms, defined by

$$s \gtrsim_\phi t \iff_{\mathrm{def}} \phi(s) \gtrsim_{\mathcal{E}} \phi(t) \text{ for all ground substitutions } \sigma$$

is a terminating quasiorder, as can easily be verified. If every ϕ^f is $\gtrsim_{\mathcal{D}}$-monotonic, more precisely if

$$d \gtrsim_{\mathcal{D}} d' \implies \phi^f(\ldots, d, \ldots) \gtrsim_{\mathcal{E}} \phi^f(\ldots, d', \ldots) ,$$

then $\gtrsim_\phi$ satisfies the condition

$$s \gtrsim_{[\_]} t \implies f(\ldots, s, \ldots) \gtrsim_\phi f(\ldots, t, \ldots) .$$

By the following result, $\gtrsim_\phi$ is then a constant status component.

Proposition 4.1. *Let* $\gtrsim_\phi \subseteq \mathcal{GT}^2$ *be a terminating quasiorder such that*

$$s \gtrsim_{[\_]} t \implies f(\ldots, s, \ldots) \gtrsim_\phi f(\ldots, t, \ldots) \tag{2}$$

holds for every function symbol $f \in \mathcal{F}$, *and every position* i *of* s *in* $f(\ldots, s, \ldots)$. *Then the constant mapping* $\mathrm{STAT} : \mathfrak{P}(\mathcal{GT}^2) \to \mathfrak{P}(\mathcal{GT}^2)$, *defined by* $\mathrm{STAT}(\gtrapprox) =_{\mathrm{def}} \gtrsim_\phi$ *for every binary relation* $\gtrapprox$ *on terms, is a status component.*

Proof. *As* STAT($\gtrapprox$) *does not depend on* $\gtrapprox$, *subterm foundedness is trivial. All other required properties of* STAT *(quasiorder, termination,* (2)*) are given.*

It is safe to assume that the set of strict preparedness of STAT is empty. Of course, it may be convenient to reason ad-hoc in favour of a non-empty set of strict preparedness.

We stipulate that $\gtrsim_{[\_]}$ and $\gtrsim_{\phi}$ need not coincide. Indeed they may be induced by different functions, $[\_] \neq \phi$. Therefore $\gtrsim_{\mathcal{D}}$, and so $\gtrsim_{[\_]}$, need not terminate, and ϕ need not be a homomorphism. $\gtrsim_{\phi}$ can be chosen as strong as wanted, without having to strengthen $\gtrsim_{[\_]}$ at the same time. To underline this distinction, we will call $[\_]$ an *interpretation*, as opposed to each ϕ which we will call a *measure*. Kamin/Lévy [25] seem to have recognized the potential of this distinction, but there was no method to exploit it until Zantema [36] expressed the concept formally by the notion of labelling function π in his "semantic labelling" approach.

The following special forms of measure functions are worth mentioning. Constant functions ϕ are measure functions. The interpretation $[\_]$ itself may be used as a measure function. The function that takes the the top function symbol, $\phi^f(d_1, \ldots, d_n) = f$, to be compared by a precedence, i.e. a terminating quasiorder on function symbols, $\gtrsim_{prec} \subseteq \mathcal{F}^2$, defines a measure: Choose $\mathcal{E} = \mathcal{F}$, with $\gtrsim_{\mathcal{E}} = \gtrsim_{prec}$, and $\phi^f(d_1, \ldots, d_n) = f$. In that case, the underlying interpretation is irrelevant.

4.2 Compositions of Status Components

In the previous section, we encountered the probably most important form of status component, *measures*. Next we show now that (pointwise) *lexicographic combination* preserves status components.

Definition 4.1 (Lexicographic combination of status components). *The* lexicographic combination *of status components* STAT_1, STAT_2, *is a functional* $\mathrm{STAT} = (\mathrm{STAT}_1, \mathrm{STAT}_2)_{lex}$ *defined by*

$$\mathrm{STAT}(\gtrsim) \quad =_{\mathrm{def}} \quad (\ \mathrm{STAT}_1(\gtrsim)\ ,\ \mathrm{STAT}_2(\gtrsim)\ )_{lex}$$

for each binary relation $\gtrsim$ *on terms.*

Proposition 4.2.

1. *Lexicographic combination preserves status components.*
2. *If* STAT_1, STAT_2 *are strictly prepared for the sets* S_1, S_2, *respectively, then* STAT *is strictly prepared for the set* $S_1 \cup S_2$.

Part 2 of the claim witnesses that the notion of set of strict preparedness is useful. For the proof see Appendix A.2.

In a similar way, pointwise intersection ($\mathrm{STAT}(\gtrsim) =_{\mathrm{def}} \mathrm{STAT}_1(\gtrsim) \cap \mathrm{STAT}_2(\gtrsim)$) can be shown to preserve status components. The set of strict preparedness is again the union of the sets of strict preparedness of the components. Pointwise intersection, as opposed to pointwise lexicographic combination, is interesting for automatic tools where it is advantageous to delay the decision whether STAT_1 or STAT_2 should receive more weight in the comparison.

4.3 Selection of Arguments

Third we are going to show that comparison of an *argument at a specified position* is a status component. It may depend on the top function symbol which argument position is meant. This is expressed by a function π.

Definition 4.2 (Selection of an argument, $\_{\pi}$). *An* (argument) selector *is a function* $\pi : \mathcal{F} \to \mathbb{N} \setminus \{0\}$. *It induces a functional* $\_{\pi}$ *defined by*

$f(s_1, \ldots, s_m) \gtrsim_\pi g(t_1, \ldots, t_n)$, *if*

1. $\pi(f) > m$ *and* $\pi(g) > n$, *or*
2. $\pi(f) \leq m$ *and* $\pi(g) \leq n$ *and* $s_{\pi(f)} \gtrsim t_{\pi(g)}$

The case analysis is needed because $\pi(f) > m$ may happen, e.g. if f is a constant. In this case $s_{\pi(f)}$ is undefined. Case 1 is necessary to maintain reflexivity of $\gtrsim_\pi$. We will use $\_{i}$ where $i \in \mathbb{N}$ to abbreviate the selector $\_{\pi}$ where $\pi(f) = i$ holds for every $f \in \mathcal{F}$.

Proposition 4.3.

1. $\_{\pi}$ *is a status component for every* π.
2. $\_{\pi}$ *is strictly prepared for the set* $\{(f, \pi(f)) \mid f \in \mathcal{F} \wedge \pi(f) \leq \mathsf{arity}(f)\}$.

For the proof see Appendix A.2.

4.4 Restriction

Yet another operator for status components proves useful. One may collapse all terms whose top function symbol is not in a given set. We will argue below that together with intersection it enables writing a conditional for status comparisons.

Definition 4.3 (Restriction, $\restriction f$). *Let* STAT *be a status component and* S *a set of function symbols. The functional* STAT $\restriction S$ *is defined by*

$f(s_1, \ldots, s_m) \quad (\mathrm{STAT} \restriction S)(\gtrsim) \quad g(t_1, \ldots, t_n) \quad$, *if*

1. $f, g \notin S$, *or*
2. $f, g \in S$ *and* s $\mathrm{STAT}(\gtrsim)$ t .

Proposition 4.4.

1. $\_ \restriction S$ *preserves status components.*
2. *If* STAT *is strictly prepared for the set* S' *then* STAT $\restriction S$ *is strictly prepared for the set* $\{(f, i) \mid f \in S \wedge (f, i) \in S'\}$.

For the proof see Appendix A.2.

4.5 A Proof Method

We propose the following method to prove termination.

1. Provide a *monotonic interpretation* $(\mathcal{D}, \gtrsim_\mathcal{D}, [\_])$, i.e. a quasiordered set $(\mathcal{D}, \gtrsim_\mathcal{D})$, together with a $\mathcal{F}$-sorted family of $\gtrsim_\mathcal{D}$-monotonic functions $[f] : \mathcal{D}^{\mathsf{arity}(f)} \to \mathcal{D}$.

2. Design a number of *measures* $(\mathcal{E}, \gtrsim_\mathcal{E}, \phi)$ (among which there may be constant functions, the interpretation itself, precedences, etc.); each ϕ as a $\mathcal{F}$-sorted family $\phi^f : \mathcal{D}^{\mathsf{arity}(f)} \to \mathcal{E}$ of $\gtrsim_\mathcal{D}$-monotonic functions to a quasiordered set $(\mathcal{E}, \gtrsim_\mathcal{E})$ where $\gtrsim_\mathcal{E}$ terminates.

3. Design a *status component* STAT $= (\gtrsim_{prec}, \bigcap_{f\in\mathcal{F}} \mathrm{STAT}_f)_{lex}$ where for each function symbol[3] f, there is a status component $\mathrm{STAT}_f = (\mathrm{STAT}_{1(f)}, \dots, \mathrm{STAT}_{n(f)})_{lex} \restriction f$ composed of status components $\mathrm{STAT}_{j(f)}$ which may be measures or selectors. The restriction operator $\restriction f$ cares that only terms whose top symbol is f are affected; all other terms are equivalent. Establish strict preparedness for contexts in all arguments; as necessary by adding $\__i$ as a lexicographic component at the end of STAT_f. The intersection puts all STAT_f together. This way one has a case analysis of status components dependent on the top function symbol f.

4. Show $l\sigma \gtrsim_{[\_]} r\sigma$ and $l\sigma >_{gpo} r\sigma$ for all $(l \to r) \in R$ and ground substitutions σ.

One may even extend the proof of termination of R towards a proof of equational termination of the equational term rewriting system (R, E), by

5. Show $l\sigma \sim_{[\_]} r\sigma$ and $l\sigma \sim_{gpo} r\sigma$ for all $(l \equiv r) \in E$ and ground substitutions σ.

However this extension seems not promising; $\sim_{gpo}$ is too weak.

5 Applications

5.1 Examples

Next some examples follow where we demonstrate how the improved general path order works. Let the ordered sets $(\mathcal{D}, \gtrsim_{\mathcal{D}}) =_{\mathrm{def}} (\{0,1\}, =)$ and $(\mathcal{E}, \gtrsim_{\mathcal{E}}) =_{\mathrm{def}} (\{0,1\}, \geq)$ be presupposed.

Example 2. *The following one-rule system of Dershowitz's [7] is classic for a terminating, not simply terminating rewrite system.*

$$f(f(x)) \to f(g(f(x)))$$

To prove its termination by the improved general path order let the interpretation $[\_]$ *be given by*

$$[f](x) = 1, \quad [g](x) = 0 \ ,$$

a measure ϕ *by*

$$\phi^f(x) = x \ ,$$

and a precedence $\gtrsim_{prec}$ *by*

$$f >_{prec} g \ .$$

The improved general path order $\gtrsim_{gpo}$ *induced by the functional*

$$\mathrm{STAT} =_{\mathrm{def}} (\gtrsim_{prec}, \gtrsim_{\phi}, \_1)_{lex}$$

is able to prove termination. First, STAT *is indeed a status: It is composed of status components, and contains* $\_1$ *as a component whence it is strictly prepared for contexts. To prove* $R \subseteq >_{gpo} \cap \gtrsim_{[\_]}$ *let* σ *be a ground substitution and* $t =_{\mathrm{def}} \sigma(x)$*. We have to show* $f(f(t)) \gtrsim_{[\_]} f(g(f(t)))$ *and* $f(f(t)) >_{gpo} f(g(f(t)))$*. To prove the former,*

$$[f(f(t))] = [f]([f]([t])) = 1 = [f]([g]([f]([t]))) = [f(g(f(t)))] \ .$$

The proof of $f(f(t)) >_{gpo} f(g(f(t)))$ *is given in a compact Gentzen-style representation.*

$$\dfrac{\dfrac{\dfrac{-}{f(f(t)) >_{gpo} f(t)}\,\triangleright}{f(f(t)) >_{gpo} g(f(t))}\,{\scriptstyle f >_{prec} g}}{f(f(t)) >_{gpo} f(g(f(t)))}\,{\scriptstyle f \sim_{prec} f \text{ and } f(f(t)) >_{\phi} f(g(f(t)))}$$

[3] more generally, for every $\sim_{prec}$-equivalence class

This schema is to say: By the subterm property of $>_{spo}$, $f(f(t)) >_{gpo} f(t)$ holds. From this, $f(f(t)) >_{gpo} g(f(t))$ follows by Case 1 of the definition of $\gtrsim_{gpo}$, due to $f >_{prec} g$. From that, $f(f(t)) >_{gpo} g(f(t))$ by Case 1, due to $f \sim_{prec} f$ and $f(f(t)) >_{\phi} f(g(f(t)))$.

The claim $f(f(t)) >_{\phi} f(g(f(t)))$ is proven by the following derivation.

$$\phi(f(f(t))) = \phi^f([f]([t])) = \phi^f(1) = 1 \gtrsim_{\mathcal{E}} 0 = \phi^f(0) = \phi^f([g]([f]([t]))) = \phi(f(g(f(t)))) \ .$$

So $f(f(t)) >_{gpo} f(g(f(t)))$ and $f(f(t)) \gtrsim_{[\_]} f(g(f(t)))$ hold. This proves $R \subseteq >_{gpo} \cup \gtrsim_{[\_]}$, and by Theorem 1 the proof of termination of R is finished.

Example 3. *Toyama's one-rule rewriting system [34]*

$$f(a, b, x) \to f(x, x, x)$$

is known as terminating but not simply terminating. A termination proof by the improved general path order is as follows. Define an interpretation by

$$[a] = 0, \quad [b] = 1, \quad [f](x, y, z) = 1$$

A useful measure function is ϕ^f defined by

$$\phi^f(x, y, z) =_{\mathrm{def}} |x - y| \ .$$

With the status STAT $=_{\mathrm{def}} (\gtrsim_{\phi}, \text{-}1)_{lex}$, *the rewrite rule is ordered by $>_{gpo} \cap \gtrsim_{[\_]}$. Let σ be a ground substitution and $t =_{\mathrm{def}} \sigma(x)$. First, $f(a, b, t) \gtrsim_{[\_]} f(t, t, t)$, because*

$$[f(a, b, t)] = 1 = [f(t, t, t)] \ .$$

Now for $f(a, b, t) >_{gpo} f(t, t, t)$. Trivially $f(a, b, t) >_{gpo} t$. The following derivation proves that $f(a, b, t) >_{\phi} f(t, t, t)$.

$$\phi(f(a, b, t)) = \phi^f(0, 1, [t]) = 1 \gtrsim_{\mathcal{E}} 0 = \phi^f([t], [t], [t]) = \phi(f(t, t, t))$$

Example 4. *Huet/Oppen [23] gave a rewrite system known to be simply terminating, but not totally terminating. We extend it by a third rule such that the system is still simply terminating, not totally terminating.*

$$\begin{aligned} f(a) &\to f(b), \\ g(b) &\to g(a), \\ f(x) &\to g(x) \end{aligned}$$

Improved general path order proves its termination. Define an interpretation $[\_]$ by

$$[a] = 0, \quad [b] = [f](x) = [g](x) = 1$$

Choose STAT $=_{\mathrm{def}} (\gtrsim_{prec}, \gtrsim_{\phi}, \text{-}1)_{lex}$ *where the precedence $\gtrsim_{prec}$ is given by*

$$f >_{prec} b; \qquad f >_{prec} g >_{prec} a$$

and the measure ϕ is given by

$$\phi^f(x) =_{\mathrm{def}} 1 - x, \quad \phi^g(x) =_{\mathrm{def}} x$$

With these settings, $f(a) >_{gpo} f(b)$ holds. For, $f(a) >_{gpo} b$ holds by the precedence $f >_{prec} b$. And $f(a) >_{\phi} f(b)$ by the derivation

$$\phi(f(a)) = \phi^f(0) = 1 \gtrsim_{\mathcal{E}} 0 = \phi^f(1) = \phi(f(b)) \ .$$

This finishes the proof for the first rule. The second rule is done analogously. For the third rule, we have $f >_{prec} g$ which leaves to show $f(t) \gtrsim_{gpo} t$. The latter is by the strict subterm property.

Example 5. *The following is an extract (rules (9), (38), (39), resp.) of the* `LIFT` *example in appendix B.*

$$g(b) \to f(b)$$
$$f(a) \to g(a)$$
$$b \to a$$

Despite its similarity to example 4 this system is totally terminating. Though the recursive path order cannot prove termination[4]*. To prove termination by the improved general path order, we use the same interpretation* $[\_]$ *as in example 4, however for the ordered set* $(\mathcal{D}, \gtrsim_{\mathcal{D}}) =_{\text{def}} (\{0,1\}, \geq)$. *Let the precedence* $\gtrsim_{prec}$ *be defined by*

$$f \sim_{prec} g >_{prec} b >_{prec} a$$

and the monotonic measure function ϕ *to* $(\mathcal{E}, \gtrsim_{\mathcal{E}}) =_{\text{def}} (\{0,1,2,3\}, \geq)$ *by*

$$\phi^f(x) =_{\text{def}} 1, \quad \phi^g(x) =_{\text{def}} 2x$$

Then $>_{gpo}$ *using status* $\mathrm{STAT} =_{\text{def}} (\gtrsim_{prec}, {}_{-1}, \gtrsim_{\phi})_{lex}$ *orders the three rules. Moreover, each rule* $l \to r$ *satisfies* $l \gtrsim_{[\_]} r$.

Here observe that were $[b] \sim_{\mathcal{D}} [a]$ *then* $f(b) \sim_{\phi} f(a)$ *as well as* $g(b) \sim_{\phi} g(a)$ *by monotonicity of* ϕ^f *and* ϕ^g, *respectively. But that would make any semantic comparison useless. For this reason it is essential to order* $[b] = 1 >_{\mathcal{D}} 0 = [a]$ *non-trivially whence* $x \mapsto 1 - x$ *is no longer monotonic.*

Example 6. *The idea to the following example comes from Dershowitz' example 18 [9]. Dershowitz demonstrates that the recursive path order cannot prove termination but a lexicographic combination of recursive path orders can. We insert an* f *symbol on the right hand side of his second rule, by which we get a system where even a lexicographic combination of recursive path orders fails.*

$$g(f(x), y) \to f(h(x,y))$$
$$h(x,y) \to g(x, f(y))$$

The same pattern occurs when logic programs are transformed to term rewriting systems in order to prove their termination [1]. Unlike Arts/Zantema, here we need no interpretation, i.e. everything is interpreted equal. Thus in fact we deal with a "syntactic" path order. We choose as the status functional $\mathrm{STAT} =_{\text{def}} (\gtrsim_{prec}, {}_{-1}, \gtrsim'_{prec}, {}_{-2})_{lex}$ *where the two precedences* $\gtrsim_{prec}$, *and* $\gtrsim'_{prec}$ *are defined by*

$$g \sim_{prec} h >_{prec} f \qquad \text{and} \qquad h >'_{prec} g \ ,$$

respectively. Now let σ *be a ground substitution and* $t =_{\text{def}} \sigma(x)$, $u =_{\text{def}} \sigma(y)$. *With that the two rules are ordered as follows.*

$$\dfrac{\dfrac{\dfrac{-}{g(f(t),u) >_{gpo} t}\rhd \quad \dfrac{-}{g(f(t),u) >_{gpo} u}\rhd \quad \dfrac{-}{f(t) >_{gpo} t}\rhd}{g(f(t),u) >_{gpo} h(t,u)}\,{\scriptstyle g\sim_{prec}h}}{g(f(t),u) >_{gpo} f(h(t,u))}\,{\scriptstyle g>_{prec}f}$$

$$\dfrac{\dfrac{-}{t \sim_{gpo} t} = \quad \dfrac{-}{f(t) >_{gpo} t}\rhd}{h(t,u) >_{gpo} g(t, f(u))}\,{\scriptstyle h\sim_{prec}g \text{ and } h>'_{prec}g}$$

[4] The polynomial interpretation order can: $[a] = 2$, $[b] = 3$, $[f](x) = x + 5$, $[g](x) = 3x$.

Example 7. *The improved general path order is able to improve upon the Knuth/Bendix order, as we can demonstrate at example 17 of Dershowitz [9] extended by an additional rule for function symbol f.*

$$\begin{array}{rcl} --x & \to & x \\ -(x+y) & \to & ---x * ---y \\ -(x*y) & \to & ---x + ---y \\ f(-x) & \to & ---f(x) \end{array}$$

Choose $(\mathcal{D}, \gtrsim_{\mathcal{D}}) =_{\text{def}} (\mathbb{N}, \geq)$. The interpretation counts the number of symbols "$+$" and "$*$" in a term.

$$[f](x) = [-](x) = x, \quad [+](x) = [*](x) = x+1$$

Let the status functional be $\text{STAT} =_{\text{def}} (\gtrsim_{[\_]}, \gtrsim_{prec}, {}_{-1}, {}_{-2})_{lex}$*, where the precedence $\gtrsim_{prec}$ is defined by*

$$f >_{prec} - >_{prec} + \sim_{prec} * \ .$$

Note that the proof obligation $l \gtrsim_{[\_]} r$ for every rewrite rule $l \to r$ is void because it already follows from $l \gtrsim_{gpo} r$, by the subterm property of $\gtrsim_{[\_]}$ (see also Lemma 5.3). Note also that an essential technical condition for the Knuth/Bendix order is not satisfied: that "$-$" has to be the greatest symbol in the precedence.

Example 8. *Two weird functions, f and g, on the natural numbers are specified by the following rewrite rules [20].*

$$\begin{array}{ccc} x - 0 \to x & f(0) \to 0 & g(0) \to s(0) \\ 0 - s(y) \to 0 & f(s(x)) \to s(x) - g(f(x)) & g(s(x)) \to s(x) - f(g(x)) \\ s(x) - s(y) \to x - y & & \end{array}$$

We do not know the semantics of f and g, but it is fairly obvious that $[f](x) \leq x$ and $[g](x) \leq x+1$ hold provided that $[f]$ and $[g]$ are total. We can profit from this knowledge by defining an interpretation $[\_]$ by

$$\begin{array}{lll} [0] =_{\text{def}} 0, & [s](x) =_{\text{def}} x+1, & [-](x,y) =_{\text{def}} \max\{x,y\}, \\ [f](x) =_{\text{def}} x, & [g](x) =_{\text{def}} x+1 & \end{array}$$

to the natural numbers, naturally ordered. "$-$" is interpreted as maximum because subtraction is not monotonic under $\geq$. Now $l \gtrsim_{[\_]} r$ holds for every rewrite rule $l \to r$. To witness,

$$\begin{gathered} [t-0] = \max\{[t], 0\} = [t], \\ [0 - s(u)] = \max\{0, [u]+1\} > 0 = [0], \\ [s(t) - s(u)] = \max\{[t]+1, [u]+1\} > \max\{[t],[u]\} = [t-u], \\ [f(0)] = 0 = [0], \\ [f(s(t))] = [t]+1 = \max\{[t]+1, [t]+1\} = [s(t) - g(f(t))], \\ [g(0)] = 1 = [s(0)], \\ [g(s(t))] = [t]+2 > \max\{[t]+1, [t]+1\} = [s(t) - f(g(t))] \end{gathered}$$

Let us now choose $\text{STAT} =_{\text{def}} (\gtrsim_{[\_]}, \gtrsim_{prec}, {}_{-1}, {}_{-2})_{lex}$ *as a status, where the precedence is given by*

$$f \sim_{prec} g >_{prec} - >_{prec} s \gtrsim_{prec} 0$$

Then the induced general path order proves termination. The decisive inequalities are

$$[s(t)] = [t] + 1 > [t] = [f(t)] \quad \textit{and}$$
$$[g(s(t))] = [t] + 2 > [t] + 1 = [f(g(x))] \ .$$

Example 9. *And finally an example where it pays to have measures different from the interpretations. Let x/y denote the integer quotient of natural numbers x, y, specified as follows.*

$$\begin{array}{cccc}
x - 0 \to x & x < 0 \to \texttt{false} & \texttt{if}(\texttt{true}, x, y) \to x & x/0 \to 0 \\
0 - s(y) \to x & 0 < s(y) \to \texttt{true} & \texttt{if}(\texttt{false}, x, y) \to y & 0/y \to 0 \\
s(x) - s(y) \to x - y & s(x) < s(y) \to x < y & \multicolumn{2}{l}{s(x)/s(y) \to \texttt{if}(x < y, 0, s((x-y)/s(y)))}
\end{array}$$

The last rule of this rewrite system has a self-embedding derivation. The re-occurrences of the slices of the left hand side are enclosed by a frame box and by a hollow frame box, respectively.

$$s(x)/s(s(x)) \quad \to_R \quad \texttt{if}(x < s(x), 0, s(\boxed{\boxed{(x - \boxed{s(x)})}/s(s(x))}))$$

The improved general path order with the following settings proves termination. Natural interpretation to $\mathcal{D} = \mathbb{N}$ ordered by equality; precedence $/ >_{prec}$ $\texttt{if}, <, s >_{prec} \texttt{true}, \texttt{false}, 0$; status STAT $= (\gtrsim_{prec}, \gtrsim_\phi, {}_{\text{-}1}, {}_{\text{-}2})$*; measure $\phi^/(x, y) = x$ to $\mathcal{D} = \mathbb{N}$ ordered by $\geq$. The decisive inequation is*

$$\phi(s(t)/s(u)) = [s(t)] = [t] + 1 > [t] - [u] = [t - u] = \phi((t - u)/s(u)) \ .$$

This quite naive design solves the following conflicting goals: Whereas $[/]$ is bound to be integer division *in order to have $[\_]$ a (quasi-)model, $\phi^/$ should express that* the first parameter *decreases semantically.* $\gtrsim_{[\_]}$ may not *compare $s(t)$ and t; otherwise our chosen interpretations for "$-$" and "$/$" were not monotonic. But* $\gtrsim_\phi$ must *compare $s(t)$ and t so as to help order the last rule.*

5.2 Termination Pairs

The orders $>_{gpo}$ and $>_{gpo} \cap \gtrsim_{[\_]}$ correspond to each other in a remarkable way. $>_{gpo}$ has the subterm property whereas $>_{gpo} \cap \gtrsim_{[\_]}$ is closed under contexts. Precisely such a *pair* of orders is needed in the framework of *conditional rewriting.*

To have the rewrite relation $\to_R$ of a conditional rewrite system R computable, one needs to show that for each application of a conditional rewrite rule

$$p \implies l \to r$$

the recursive descent into the premise p as well as the successful application of the rule decreases the term. To simplify the presentation, we consider the premise to be a term. A rule is *enabled* if its premise is reducible to the constant `true`.

Wirth/Gramlich defined the notion of termination pair.

Definition 5.1 ([35]). *A* termination pair *is a pair $(>_{sub}, >_{mon})$ of orders, closed under substitution, such that*

1. $>_{mon} \subseteq >_{sub}$,
2. $>_{sub}$ *terminates,*

3. $>_{sub}$ *has the subterm property,*

4. $>_{mon}$ *is closed under contexts.*

A rule $p \implies l \to r$ is called *aligned with* the termination pair $(>_{sub}, >_{mon})$, if $l >_{sub} p$ and $l\sigma >_{mon} r\sigma$ for all substitutions σ such that $p\sigma \to_R^* \texttt{true}$. Dershowitz/Okada/Sivakumar's notion of decreasing conditional rewrite system [15] is related: $p \implies l \to r$ is called *decreasing* if there is a terminating order $>_{sub}$, closed under substitution, such that $l >_{sub} p$ and $\to_R \subseteq >_{sub}$. Obviously a decreasing system is aligned with the termination pair $(>_{sub}, \to_R^+)$.

Theorem 2 ([35]). *If* $(>_{sub}, >_{mon})$ *is a termination pair, and every rule of a system R of conditional rewrite rules is aligned with it, then every term has effectively an R-normal form.*

If $>_{mon}$ is a simplification order then $>_{mon}$ and $>_{sub}$ coincide. The case where $>_{mon}$ is not a simplification order has not yet been investigated. The following example, due to Dershowitz/Okada/Sivakumar, needs such a pair.

Example 10 ([15]).

$$b \to \texttt{true} \tag{4}$$

$$f(b) \to f(a) \tag{5}$$

$$b \implies a \to \texttt{true} \tag{6}$$

Rule (6) requires $a >_{sub} b$ and Rule (5) requires $f(b) >_{mon} f(a)$ to hold. Were $a >_{mon} b$ then by closure under contexts $f(a) >_{mon} f(b)$, a contradiction. So $>_{mon}$ and $>_{sub}$ must differ.

Theorem 3. *The conditional rewriting system of example* (10) *is aligned with some termination pair* $(>_{gpo}, >_{gpo} \cap \gtrsim_{[\_]})$.

Proof. *We choose* $(\mathcal{D}, \gtrsim_{\mathcal{D}}) =_{\text{def}} (\mathcal{E}, \gtrsim_{\mathcal{E}}) =_{\text{def}} (\{0,1\}, \geq)$. *The interpretation,* $[\_]$, *is given by*

$$[b] = 1, \quad [a] = [\texttt{true}] = [f](x) = 0 \ .$$

The general path order $>_{gpo}$ *is induced by the status* $\text{STAT}(\gtrsim) =_{\text{def}} (\gtrsim_{prec}, \gtrsim_{\phi}, \text{-}1)_{lex}$, *with the precedence,* $\gtrsim_{prec}$, *given by*

$$f >_{prec} a >_{prec} b >_{prec} \texttt{true}$$

and the measure, ϕ, *by* $\phi^f(x) = x$. *We have to satisfy the following proof obligations.*

1. $[f]$ *and* ϕ^f *are monotonic: Trivial.*
2. $f(b) \gtrsim_{[\_]} f(a)$: $[f(b)] = 0 = [f(a)]$.
3. $f(b) >_{gpo} f(a)$: *By the precedence* $f >_{prec} a$, $f(b) >_{gpo} a$ *holds. Next,* $f \sim_{prec} f$ *and* $f(b) >_{\phi} f(a)$ *by*

$$\phi(f(b)) = \phi^f(1) = 1 \gtrsim_{\mathcal{E}} 0 = \phi^f(0) = \phi(f(a)) \ .$$

4. $b \gtrsim_{[\_]} \texttt{true}$: $[b] = 1 >_{\mathcal{D}} 0 = [\texttt{true}]$.
5. $b >_{gpo} \texttt{true}$: *By precedence.*
6. $a \gtrsim_{[\_]} \texttt{true}$: $[a] = 0 = [\texttt{true}]$.

7. *$a >_{gpo}$ `true`: By precedence.*

8. *$a >_{gpo} b$: By precedence.*

Note that for Case (4) actually $b >_{[\_]}$ `true` was essential. For, if $b \sim_{[\_]} a$ then $f(b) \sim_{\phi} f(a)$ by monotonicity, which fails to handle Rule (5). This finishes the proof.

5.3 Improved?

We claim that Dershowitz/Hoot's general path order is an instance of our improved general path order. This statement can be put precisely and the proof is instructive.

Theorem 4. *Every valid termination proof by Dershowitz/Hoot's general path order where the functional*

- *does not use proper multisets nor ranked multisets of arguments, and*
- *contains selections for each argument position,*

is a valid termination proof by improved general path order.

I conjecture that with some technical effort the multiset cases can be added as well. Furthermore I hold that the second requirement is weak. The proof will use the following two standard results from universal algebra.

Lemma 5.1. *If $I_i =_{\text{def}} (\mathcal{D}_i, \gtrsim_{\mathcal{D}_i}, [\_]_i)$ is a monotonic $\mathcal{F}$-interpretation for every $1 \le i \le k$, then their product*

$$I = I_1 \times \cdots \times I_k =_{\text{def}} (\mathcal{D}, \gtrsim_{\mathcal{D}}, [\_]) \ ,$$

defined by

$$\begin{aligned} \mathcal{D} \quad &=_{\text{def}} \quad \mathcal{D}_1 \times \cdots \times \mathcal{D}_k, \\ d \gtrsim_{\mathcal{D}} d' \quad &\iff_{\text{def}} \quad d_1 \gtrsim_{\mathcal{D}_1} d'_1 \wedge \cdots \wedge d_k \gtrsim_{\mathcal{D}_k} d'_k, \\ [f](d^1, \ldots, d^m) \quad &=_{\text{def}} \quad ([f]_1(d^1_1, \ldots, d^m_1), \ldots, [f]_k(d^1_k, \ldots, d^m_k)) \end{aligned}$$

is a monotonic $\mathcal{F}$-interpretation. Here d_i denotes the i-th component of the k-tuple d.

Lemma 5.2. *If $I_i =_{\text{def}} (\mathcal{D}_i, \gtrsim_{\mathcal{D}_i}, [\_]_i)$ is a monotonic $\mathcal{F}$-interpretation for every $1 \le i \le k$, strictly monotonic if $i \ne k$, then their lexicographic product*

$$I = I_1 \times_{lex} \cdots \times_{lex} I_k =_{\text{def}} (\mathcal{D}, \gtrsim_{\mathcal{D}}, [\_]) \ ,$$

defined by

$$\begin{aligned} \mathcal{D} \quad &=_{\text{def}} \quad \mathcal{D}_1 \times \cdots \times \mathcal{D}_k, \\ d \gtrsim_{\mathcal{D}} d' \quad &\iff_{\text{def}} \quad d_1 >_{\mathcal{D}_1} d'_1 \vee \\ &\qquad d_1 \sim_{\mathcal{D}_1} d'_1 \wedge (d_2 >_{\mathcal{D}_2} d'_2 \vee \\ &\qquad\qquad d_2 \sim_{\mathcal{D}_2} d'_2 \wedge (\cdots \\ &\qquad\qquad\qquad \wedge d_k \gtrsim_{\mathcal{D}_k} d'_k) \cdots), \\ [f](d^1, \ldots, d^m) \quad &=_{\text{def}} \quad ([f]_1(d^1_1, \ldots, d^m_1), \ldots, [f]_k(d^1_k, \ldots, d^m_k)) \end{aligned}$$

is a monotonic $\mathcal{F}$-interpretation. Here d_i denotes the i-th component of the k-tuple d.

The subterm property is useful to get rid of the obligation to prove $l \gtrsim_{[\_]} r$ as the following lemma shows.

Lemma 5.3 ([18]). *If $\gtrsim_{[\_]}$ has the subterm property and* $\mathrm{STAT} = (\gtrsim_\phi, \mathrm{STAT}_1)_{lex}$ *has as first component a quasiorder $\gtrsim_\phi \subseteq \gtrsim_{[\_]}$, then $\gtrsim_{gpo} \subseteq \gtrsim_{[\_]}$.*

Proof. *Let $s, t \in \mathcal{GT}$ and let $s >_{gpo} t$. To prove $s \gtrsim_{[\_]} t$ we employ induction on s ordered by $\rhd$. For $s >_{gpo} t$ we distinguish cases along the definition of $\gtrsim_{gpo}$. Case (1): s $\mathrm{STAT}(\gtrsim_{gpo})$ t and $s >_{gpo} t_i$ for all i. By the architecture of* STAT *chosen, $s \gtrsim_\phi t$ follows immediately. So $s \gtrsim_{[\_]} t$ by premise. Case (2): $s_i \gtrsim_{gpo} t$ for some i. Then $s \gtrsim_{[\_]} s_i \gtrsim_{[\_]} t$ holds by the subterm property of $\gtrsim_{[\_]}$, and the inductive hypothesis for s_i, respectively. By transitivity of $\gtrsim_{[\_]}$ the claim follows.*

Proof (of Theorem 4). *Let w.l.o.g. a finite or infinite ground term rewriting system R be given, together with a proof of termination by Dershowitz/Hoot's general path order obeying the mentioned restrictions. In other words, there is $0 \le i \le k \le n$; for each $1 \le j \le k$ there are interpretations $I_j = (D_j, \gtrsim_{D_j}, [\_]_j)$ such that*

1. *$I_1, \ldots, I_{i-1}$ are strictly monotonic and satisfy the strict subterm property,*

2. *I_i is monotonic and satisfies the strict subterm property, and*

3. *$I_{i+1}, \ldots, I_k$ are value-preserving congruences, i.e. $l \sim_{[\_]_j} r$ and f is $\sim_{[\_]_j}$-monotonic for every f, $i+1 \le j \le k$;*

there is a functional

$$\mathrm{STAT} =_{\mathrm{def}} (\gtrsim_{[\_]_1}, \ldots, \gtrsim_{[\_]_i}, \mathrm{STAT}_{i+1}, \ldots, \mathrm{STAT}_n)$$

such that each of $\mathrm{STAT}_{i+1}, \ldots, \mathrm{STAT}_n$ is either

1. *a precedence, or*

2. *an argument at a specified position, or*

3. *a quasiorder $\gtrsim_{[\_]_j}$ induced by one of the interpretations I_j, $i+1 \le j \le k$;*

and $R \subseteq >_{gpo}$ where $>_{gpo}$ is the general path order induced by STAT.

First we claim that $I_{lex} =_{\mathrm{def}} I_1 \times_{lex} \cdots \times_{lex} I_i$ is a quasimodel for R. If $i = 0$ then this is the trivial model; so assume $i > 0$. Given that I_j is monotonic, if $j \ne i$ even strictly, we obtain that I_{lex} is a monotonic interpretation by Lemma 5.2. By construction the quasiorder on terms induced by the interpretation I_{lex} admits the representation

$$\gtrsim_{[\_]_{lex}} =_{\mathrm{def}} (\gtrsim_{[\_]_1}, \ldots, \gtrsim_{[\_]_i})_{lex}$$

whence by the subterm property of each I_j, $1 \le j \le i$, also I_{lex} has the subterm property. So in fact w.l.o.g. $i = 1$ may be assumed. The property $R \subseteq \gtrsim_{[\_]_{lex}}$ follows from $R \subseteq >_{gpo}$ by Lemma 5.3. So I_{lex} is a quasimodel for R.

Second we state that for every $i+1 \le j \le k$ the interpretation $\bar{I}_j =_{\mathrm{def}} (\mathcal{D}_j, \sim_{\mathcal{D}_j}, [\_]_j)$ derived from I_j is a quasimodel of R. This follows immediately from the premises put on I_j.

Now by Lemma 5.1 the product $I =_{\mathrm{def}} I_{lex} \times \bar{I}_{i+1} \times \cdots \times \bar{I}_k$ is a monotonic interpretation as well. It is even a quasimodel of R as every component is. So we may use I as the interpretation underlying our improved general path order.

With that, STAT *turns out to be a status component: As proven in Subsection 4.2 pointwise lexicographic combination preserves status components and each precedence and each selector of an argument at a specified position are status components. Each $\gtrsim_{[\_]_j}$ finally, $1 \le j \le k$, can be expressed as a quasiorder $\gtrsim_{\phi_j}$ induced by a measure ϕ_j on I. To this end define $\phi_j : \mathcal{GT} \to \mathcal{D}_j$ by*

$$\phi_j^f(d^1, \ldots, d^n) =_{\mathrm{def}} ([f](d^1, \ldots, d^n))_j = [f]_j(d_j^1, \ldots, d_j^n)$$

where d_j selects the j-th component from d. It is obvious that $\gtrsim_{\phi_j} = \gtrsim_{[\_]_j}$. Observe that the monotonicity conditions are satisfied. (For $i+1 \leq j \leq k$, the property $\sim_{[\_]_j} \subseteq \gtrsim_{[\_]_j}$ is employed.) Therefore the functional STAT *is a status component.*

By the premise that STAT *contains every argument position, it is strictly prepared for contexts, so* STAT *is a status. Theorem 1 is applicable. Hence R terminates by improved general path order. This finishes the proof.*

According to Theorem 4 the improved general path order can do all termination proofs Dershowitz/Hoot's general path order can do, except for the multiset ones, with at most the same effort. In addition to the impressive list of cases covered by Dershowitz/Hoot's method [11], our method handles some cases beyond.

- Example 9 and our `LIFT` case study (see the conclusion to Appendix B) need a measure $(\mathcal{E}, \gtrsim_{\mathcal{E}}, \phi)$ that cannot serve as interpretation $(\mathcal{D}, \gtrsim_{\mathcal{D}}, [\_])$, neither can its equivalence kernel $(\mathcal{E}, \sim_{\mathcal{E}}, \phi)$. Dershowitz/Hoot's general path order does not support that.
- To model the semantic path order, in Dershowitz/Hoot's approach Condition (C) (of Kamin/Lévy's) is an essential *extra* condition. The improved general path order cares for this condition itself and so includes the semantic path order properly.
- The improved general path order can handle the case of Knuth/Bendix order where symbols of weight 0 need not be unary and maximal in precedence (Example 7), and the case of polynomial path order where the polynomials are monotonic, but not strictly monotonic. In these cases the order $\gtrsim_{[\_]}$ no longer has the strict subterm property or is strictly monotonic, respectively.

Concluding Remarks

We have introduced the *improved general path order*, an extension both of the *general path order* of Dershowitz/Hoot and of the *semantic path order* of Kamin/Lévy. These orders are suitable to prove termination of a rewriting system whenever the termination proof needs semantic arguments. We introduced the notion of "improved general path order" in an abstract form, based on a monotonic interpretation $[\_]$ and a status STAT. Under weak premises then $>_{gpo} \cap \gtrsim_{[\_]}$ is a termination order (Thereom 1). This is another good justification of the attribute "general".

Statuses can be *composed lexicographically* from components such as *measures* and *argument selectors*. Statuses may be made depend on the top function symbol by a *restriction operator*, and joined by *intersection*. We have given a design guideline for the user. Several small examples demonstrate power, versatility and ease of use of the improved general path order. In the appendix we report on a realistic, medium-size application where we encounter some of the typical features. We have not included selectors of multisets of arguments; however it seems possible to do.

The pair $(>_{gpo}, >_{gpo} \cap \gtrsim_{[\_]})$ is to my knowledge the first nontrivial instance of a *termination pair*. A termination pair is useful to prove computability of normal forms for a conditional term rewriting system.

By Theorem 4 we have given a precise account inhowfar our method covers Dershowitz/Hoot's version. Our method can completely dispose of the technical conditions that the tuple of status components has to start with a number of interpretations which are strictly monotonic and satisfy the strict subterm property. This is due to three relaxations: a more liberal condition for the parameter STAT of the order; the conceptual separation of the interpretation, $[\_]$, from measures, ϕ; and the relaxation from models to quasimodels. Though by Lemma 5.3 the subterm property may be technically useful to get rid of a proof obligation.

A particularly important application of termination is in the Knuth/Bendix completion procedure. As is known, this procedure can be turned into one that does not stop with failure (unfailing Knuth/Bendix procedure, UKB, [3]) provided the procedure uses a termination quasiorder that can be extended to a total one on ground terms. The restriction to totally terminating rewrite systems is unsatisfactory. To overcome it we suggest the following method. The procedure obviously satisfies $s \leftrightarrow^*_{R \cup E} t$ for every critical pair (s,t). To order all ground instances of (s,t) then, $\gtrsim$ need not be total on all ground terms. It is sufficient if $\gtrsim$ is total on $\leftrightarrow^*_{R \cup E}$-equivalence classes of ground terms, i.e. it orders those pairs (s,t) of ground terms where $s \leftrightarrow^*_{R \cup E} t$.

Definition 5.2. *$\gtrsim \subseteq \mathcal{GT}^2$ is called* semantically total, *if for all ground terms $s \leftrightarrow^*_{R \cup E} t$, either $s \gtrsim t$ or $t \gtrsim s$ holds.*

If $\sim_{[\_]}$ is a model then a quasiorder is semantically total if it is total on $\sim_{[\_]}$-equivalence classes of ground terms. I conjecture that general path orders which are based on models can be made total on $\sim_{[\_]}$-equivalence classes. This narrows a conjecture of Rusinowitch (Problem 85 in [14]).

Conjecture. *Let $\sim_{[\_]}$ be a congruence on terms, closed under substitution. Every general path quasiorder $\gtrsim_{gpo}$ based on $\sim_{[\_]}$ can be extended to a general path quasiorder $\gtrsim'_{gpo}$ based on $\sim_{[\_]}$ which is total on $\sim_{[\_]}$-equivalence classes of ground terms.*

Acknowledgements

I am grateful to Ulrich Fraus for explaining me his lift rewrite system. I also wish to thank Nachum Dershowitz for a number of discussions about the general path order, and Joachim Steinbach for pointing to interesting examples.

References

[1] T. Arts and H. Zantema. Termination of logic programs via labelled term rewriting systems. Technical Report UU-CS-1994-20, Universiteit Utrecht, The Netherlands, 1994.

[2] J. Avenhaus and K. Madlener. Term rewriting and equational reasoning. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence: A Source-Book*. Elsevier Science Publishers, North-Holland, 1990.

[3] L. Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In H. Ait-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures 2: Rewriting Techniques*, pages 1–30. Academic Press, 1989.

[4] F. Bellegarde and P. Lescanne. Termination by completion. *AAECC*, 1:79–96, 1990.

[5] M. Broy. An example for the design of distributed systems in a formal setting: The lift problem. Technical Report MIP-8804, Universität Passau, Germany, 1988.

[6] M. Dauchet. Simulation of Turing machines by a regular rule. *Theoret. Comput. Sci.*, 103:409–420, 1992.

[7] N. Dershowitz. A note on simplification orderings. *Inform. Process. Lett.*, 9:212–215, 1979.

[8] N. Dershowitz. Orderings for term rewriting systems. *Theoret. Comput. Sci.*, 17(3):279–301, March 1982.

[9] N. Dershowitz. Termination of rewriting. *J. Symb. Comput.*, 3(1&2):69–115, Feb./April 1987. Corrigendum: 4, 3, Dec. 1987, 409-410.

[10] N. Dershowitz and C. Hoot. Topics in termination. In C. Kirchner, editor, *5th Int. Conf. Rewriting Techniques and Applications*, pages 198–212. Springer LNCS 690, 1993.

[11] N. Dershowitz and C. Hoot. Natural termination. *Theoret. Comput. Sci.*, 1994. submitted.

[12] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Formal Models and Semantics, Handbook of Theoretical Computer Science, Vol. B*, pages 243–320. Elsevier - The MIT Press, 1990.

[13] N. Dershowitz and J.-P. Jouannaud. Notations for rewriting. *Bulletin of the EATCS*, 43:162–172, 1991.

[14] N. Dershowitz, J.-P. Jouannaud, and J. W. Klop. Problems in rewriting III. In J. Hsiang, editor, *6th Int. Conf. Rewriting Techniques and Applications*, pages 457–471. Springer LNCS 914, 1995.

[15] N. Dershowitz, M. Okada, and G. Sivakumar. Canonical conditional rewrite systems. In *9th Int. Conf. Automated Deduction*, pages 538–549. Springer LNCS 310, 1988.

[16] U. Fraus. Verifying the specification of a technical software system by induction. Research report FORWISS, Universität Passau, Germany, 1992.

[17] U. Fraus. Inductive theorem proving for algebraic specifications — TIP system user's manual. Technical Report MIP-9401, Universität Passau, Germany, Feb. 1994.

[18] A. Geser. On a monotonic semantic path ordering. Technical Report 92-13, Ulmer Informatik-Berichte, Universität Ulm, Germany, 1992.

[19] A. Geser. An improved general path order. Technical Report MIP-9407, Universität Passau, Germany, June 1994.

[20] D. R. Hofstadter. *Gödel, Escher, Bach: An eternal golden braid.* Basic Books, New York, 1979.

[21] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *J. Assoc. Comput. Mach.*, 27:797–821, 1980.

[22] G. Huet and D. Lankford. On the uniform halting problem for term rewriting systems. Technical Report 283, INRIA, 1978.

[23] G. Huet and D. C. Oppen. Equations and rewrite rules — a survey. In R. Book, editor, *Formal Languages: Perspectives and Open Problems*, pages 349–405. Academic Press, 1980.

[24] J.-P. Jouannaud, P. Lescanne, and F. Reinig. Recursive decomposition ordering. In D. Bjørner, editor, *Formal description of programming concepts 2*, pages 331–348. North-Holland, 1982.

[25] S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path orderings. Manuscript; copy available at Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Feb. 1980.

[26] D. Kapur, P. Narendran, and G. Sivakumar. A path ordering for proving termination of term rewriting systems. In *10th Colloquium on Trees in Algebra and Programming*, pages 173–185. Springer LNCS 185, 1985.

[27] J. W. Klop. Term rewrite systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 1–116. Oxford University Press, 1992.

[28] D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[29] D. S. Lankford. On proving term rewriting systems are noetherian. Technical Report MTP-3, Louisiana Technical University, Math. Dept., Ruston, LA, 1979.

[30] P. Lescanne. Uniform termination of term rewriting systems: Recursive decomposition ordering with status. In B. Courcelle, editor, *6th Colloquium on Trees in Algebra and Programming*, pages 181–194, Bordeaux, France, Mar. 1984.

[31] P. Lescanne. On the recursive decomposition ordering with lexicographic status and other related orderings. *J. Automated Reasoning*, 6:39–49, 1990.

[32] D. A. Plaisted. Term rewriting systems. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, chapter 2. Oxford University Press, 1993. To appear.

[33] M. Rusinowitch. Path of subterms ordering and recursive decomposition ordering revisited. *J. Symb. Comput.*, 3(1&2):117–131, 1987.

[34] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Inform. Process. Lett.*, 25:141–143, 1987.

[35] C.-P. Wirth and B. Gramlich. A constructor-based inductive validity in positive/negative-conditional equational specifications. *J. Symb. Comput.*, 11, 1994.

[36] H. Zantema. Termination of rewriting by semantic labelling. Technical Report RUU-CS-92-38, Utrecht University, Dec. 1992. Extended and revised version appeared as RUU-CS-93-24, July 1993, accepted for special issue on term rewriting of Fundamenta Informaticae.

A Technical Proofs

A.1 Proof of Theorem 1, Part (1)

Let $\gtrsim_{[\_]} \subseteq \mathcal{GT}^2$ be a quasiorder, closed under contexts, and STAT : $\mathfrak{P}(\mathcal{GT}^2) \to \mathfrak{P}(\mathcal{GT}^2)$ a status. We now prove Part (1) of Theorem 1.

First we prove two simple technical properties. See also [10].

Lemma A.1. *$s \gtrsim_{gpo} t \rhd t' \implies s >_{gpo} t'$ holds for all $s, t, t' \in \mathcal{GT}$.*

Proof. *We need only consider the special case*

$$s \gtrsim_{gpo} g(t_1, \ldots, t_n) \quad \Longrightarrow \quad s >_{gpo} t_i$$

from which the claim follows by a simple structural induction on t.

Let $t = g(t_1, \ldots, t_n)$. We prove the claim by induction on pairs (s, t), ordered by the multiset extension $\rhd_{mult}$ of the subterm relation. Let $s \gtrsim_{gpo} t$, and $1 \leq i \leq n$. If Case (1) of the definition of $\gtrsim_{gpo}$ has been used then immediately $s >_{gpo} t_i$. Else Case (2) must have been used, so $s = f(s_1, \ldots, s_m)$ and $s_j \gtrsim_{gpo} t$ holds for some $1 \leq j \leq m$. By inductive hypothesis for (s_j, t) we get $s_j >_{gpo} t_i$, so by Case (2) of the definition of $\gtrsim_{gpo}$, it follows $s \gtrsim_{gpo} t_i$. To show ">", assume $t_i \gtrsim_{gpo} s$. By inductive hypothesis for (t_i, s), we get $t_i >_{gpo} s_j$, a contradiction to $s_j >_{gpo} t_i$. So $t_i \not\gtrsim_{gpo} s$, hence $s >_{gpo} t_i$.

Lemma A.2. *$s' \rhd s \gtrsim_{gpo} t \implies s' >_{gpo} t$ holds for all $s, s', t \in \mathcal{GT}$.*

Proof. *We need only consider the special case*

$$s_i \gtrsim_{gpo} t \quad \Longrightarrow \quad f(s_1, \ldots, s_m) >_{gpo} t$$

from which the claim follows by structural induction on s.

Let $s = f(s_1, \ldots, s_m)$, and $s_i \gtrsim_{gpo} t$ for some $1 \leq i \leq m$. Using Case (2) *of the definition of $\gtrsim_{gpo}$, we get $s \gtrsim_{gpo} t$. To show "$>$", assume $t \gtrsim_{gpo} s$. By Lemma A.1, $t >_{gpo} s_i$ holds, a contradiction to the premise $s_i \gtrsim_{gpo} t$. So $s >_{gpo} t$.*

Lemma A.3. *$\gtrsim_{gpo}$ is reflexive.*

Proof. *We prove $s \gtrsim_{gpo} s$ by induction on s, ordered by $\rhd$. Let $s = f(s_1, \ldots, s_m)$. By inductive hypothesis, $s_i \gtrsim_{gpo} s_i$ whence $s >_{gpo} s_i$ by Lemma A.2. By reflexivity of $\gtrsim_{[\_]}$, we get $s_i \gtrsim_{[\_]} s_i$. Preparedness for contexts of* STAT *yields s* $\mathrm{STAT}(\gtrsim_{gpo})$ *s. The claim follows by Case* (1) *of the definition of $\gtrsim_{gpo}$.*

By Lemma A.1, the *strict subterm property* follows from reflexivity of $\gtrsim_{gpo}$. Moreover, it follows immediately that $>_{gpo}$ is *irreflexive.*

To prove transitivity, we need two technical lemmas about the multiset extension of the subterm order.

Lemma A.4. *For all terms s, s', t, t', u, and u', if*

$$\forall x \in \{s', t', u'\}\ \exists y \in \{s, t, u\}.\ x \trianglelefteq y \tag{1}$$

then one of the following holds.

$$(s', t', u') \trianglelefteq_{mult} (s, t, u) \qquad or \tag{2}$$

$$\exists r, r', r''.\ (s', t', u') =_{mult} (r, r', r'') \wedge r \trianglelefteq r' \tag{3}$$

Proof. *Let* (1) *hold. Case 1: Even $\forall x \in \{s', t', u'\} \exists y \in \{s, t, u\}.\ x \lhd y$ holds. Then* (2) *holds, as can easily be checked. Case 2: Case 1 does not apply, whence $\exists x \in \{s', t', u'\}\ \exists y \in \{s, t, u\}.\ x = y$. W.l.o.g. let $s' = s$ hold. Case 2.1: $\exists x \in \{t', u'\}.\ x \trianglelefteq s$, then* (3) *holds with the settings $r = x$, $r' = s'$. Case 2.2: Case 2.1 does not apply, whence $\forall x \in \{t', u'\}\ \exists y \in \{t, u\}.\ x \trianglelefteq y$. Case 2.2.1: If even $\forall x \in \{t', u'\}\ \exists y \in \{t, u\}.\ x \lhd y$, then* (2) *holds. Case 2.2.2: If Case 2.2.1 does not apply then $\exists x \in \{t', u'\}\ \exists y \in \{t, u\}.\ x = y$. W.l.o.g. let $t' = t$. If now $u' \trianglelefteq t$, then* (3) *holds with the settings $r = u'$, $r' = t$. Otherwise $u' \trianglelefteq u$ whence* (2) *must hold.*

Lemma A.5. *For all terms s, s', t, t', u, and u', if $(s', u') \lhd_{mult} (s, u)$ and $(s', t', u') =_{mult} (s, t, u)$ then $t \lhd s$ or $t \lhd u$ holds.*

Proof. *Let $(s', u') \lhd_{mult} (s, u)$ and $(s', t', u') =_{mult} (s, t, u)$. Obviously t' must be one of s, t, u, otherwise $(s', t', u') =_{mult} (s, t, u)$ would not hold. Case 1: $t' = t$ then $(s', u') =_{mult} (s, u)$, a contradiction to $(s', u') \lhd_{mult} (s, u)$. Case 2: $t' = s$. Then $(t, u) =_{mult} (s', u') \lhd_{mult} (s, u)$, whence $t \lhd s$. Case 3: $t' = u$. Then $(s, t) =_{mult} (s', u') \lhd_{mult} (s, u)$, whence $t \lhd u$.*

We remark that it should be possible to extend these two lemmas towards tuples of arbitrary but fixed length, and to multiset extensions of any quasiorder.

Lemma A.6. *$\gtrsim_{gpo}$ is transitive.*

Proof. *We claim that for all terms s, t, and u, $s \gtrsim_{gpo} t \gtrsim_{gpo} u$ implies $s \gtrsim_{gpo} u$. The proof is done by induction on triples (s,t,u), ordered by the multiset order $\rhd_{mult}$. If $s \trianglerighteq t$, then $s \gtrsim_{gpo} u$ follows from Lemma A.2. If $t \trianglerighteq u$, then $s \gtrsim_{gpo} u$ follows from Lemma A.1. If $s \trianglerighteq u$ then $s \gtrsim_{gpo} u$ follows by the subterm property of $\gtrsim_{gpo}$. If $s \lhd t$ or $t \lhd u$, then we use Lemma A.1 to get a contradiction to irreflexivity of $>_{gpo}$. If $u \lhd s$ then we use Lemma A.1 to obtain a contradiction to the premise $s \gtrsim_{gpo} t$. This settles the case where (at least) two of s, t, u are in the subterm relation. Henceforth we may exclude that case.*

Let $s = f(s_1, \ldots, s_m)$, $t = g(t_1, \ldots, t_n)$, $u = h(u_1, \ldots, u_p)$. We distinguish cases along the definition of $\gtrsim_{gpo}$ for $s \gtrsim_{gpo} t$ and $t \gtrsim_{gpo} u$, respectively. Case 1: $s_i \gtrsim_{gpo} t$ holds for some $1 \le i \le m$. Then $s_i \gtrsim_{gpo} u$ by inductive hypothesis for the triple (s_i, t, u), so $s \gtrsim_{gpo} u$ by Case (2) of the definition of $\gtrsim_{gpo}$. Case 2: s $\mathrm{STAT}(\gtrsim_{gpo})$ t, $s >_{gpo} t_i$ for all $1 \le i \le n$, and $t_i \gtrsim_{gpo} u$ for some $1 \le i \le n$. Then $s \gtrsim_{gpo} u$ by inductive hypothesis for the triple (s, t_i, u). Case 3: s $\mathrm{STAT}(\gtrsim_{gpo})$ t $\mathrm{STAT}(\gtrsim_{gpo})$ u, $s >_{gpo} t_i$ for all $1 \le i \le n$, and $t >_{gpo} u_j$ for all $1 \le j \le p$. From $s \gtrsim_{gpo} t \gtrsim_{gpo} u_j$ by inductive hypothesis for the triple (s, t, u_j) we get $s \gtrsim_{gpo} u_j$. To show ">", assume $u_i \gtrsim_{gpo} s$. Then $u_i \gtrsim_{gpo} t$ by inductive hypothesis for (u_i, s, t), a contradiction to $t >_{gpo} u_i$. So $s >_{gpo} u_i$ holds.

The remainder of this proof is devoted to the proof of s $\mathrm{STAT}(\gtrsim_{gpo})$ u. For convenience, let $\mathsf{fin}(s,t)$ denote the set of pairs of ground terms smaller than the pair (s,t),

$$\mathsf{fin}(s,t) =_{\mathrm{def}} \{(s',t') \mid (s',t') \lhd_{mult} (s,t)\} \ ,$$

a notation that allows us to reformulate subterm foundedness as

$$s \quad \mathrm{STAT}(\gtrsim) \quad t \quad \Longleftrightarrow \quad s \quad \mathrm{STAT}(\gtrsim \cap \mathsf{fin}(s,t)) \quad t \ .$$

Let moreover

$$\mathsf{Fin} =_{\mathrm{def}} \mathsf{fin}(s,t) \cup \mathsf{fin}(t,u) \cup \mathsf{fin}(s,u) \ .$$

Now consider the relation

$$\gtrsim_{trans} =_{\mathrm{def}} (\gtrsim_{gpo} \cap \mathsf{Fin})^* \ ,$$

which is a quasiorder by definition. We claim that

$$s' \gtrsim_{trans} u' \Longleftrightarrow s' \gtrsim_{gpo} u' \tag{*}$$

holds for all $(s',u') \in \mathsf{Fin}$. Provided that (*) holds, we can finish the proof of transitivity by the following derivation.

$$\begin{array}{lll}
 & s \quad \mathrm{STAT}(\gtrsim_{gpo}) \quad t \quad \mathrm{STAT}(\gtrsim_{gpo}) \quad u & \\
\Longleftrightarrow & s \quad \mathrm{STAT}(\gtrsim_{gpo} \cap \mathsf{fin}(s,t)) \quad t \quad \mathrm{STAT}(\gtrsim_{gpo} \cap \mathsf{fin}(t,u)) \quad u & \textit{(subt. found.)} \\
\Longleftrightarrow & s \quad \mathrm{STAT}(\gtrsim_{trans} \cap \mathsf{fin}(s,t)) \quad t \quad \mathrm{STAT}(\gtrsim_{trans} \cap \mathsf{fin}(t,u)) \quad u & (*) \\
\Longleftrightarrow & s \quad \mathrm{STAT}(\gtrsim_{trans}) \quad t \quad \mathrm{STAT}(\gtrsim_{trans}) \quad u & \textit{(subt. found.)} \\
\Longrightarrow & s \quad \mathrm{STAT}(\gtrsim_{trans}) \quad u & \textit{(qu. ord. pres.)} \\
\Longleftrightarrow & s \quad \mathrm{STAT}(\gtrsim_{trans} \cap \mathsf{fin}(s,u)) \quad u & \textit{(subt. found.)} \\
\Longleftrightarrow & s \quad \mathrm{STAT}(\gtrsim_{gpo} \cap \mathsf{fin}(s,u)) \quad u & (*) \\
\Longleftrightarrow & s \quad \mathrm{STAT}(\gtrsim_{gpo}) \quad u & \textit{(subt. found.)}
\end{array}$$

Finally we have only left to prove the claim (*). This is the most complicated part of the proof. "$\Longleftarrow$" is by definition. For "$\Longrightarrow$", let a derivation

$$s' = r^0 \gtrsim_{gpo} r^1 \gtrsim_{gpo} \cdots \gtrsim_{gpo} r^k = t'$$

be given where $(r^i, r^{i+1}) \in \mathsf{Fin}$ for each $0 \le i < k$. By induction on k, we prove $r^0 \gtrsim_{gpo} r^k$. If $k = 0$ or $k = 1$ then the claim holds trivially. So let $k \ge 2$.

First we handle the case where two terms in the given derivation are in the subterm relation. To this end let $0 \le i < j \le k$ and $r^i \trianglerighteq r^j$ or $r^i \triangleleft r^j$. Case 1: $r^i \trianglerighteq r^j$ for $i \neq 0$. Then $r^{i-1} \gtrsim_{gpo} r^j$ using Lemma A.2. Furthermore, due to $(r^{i-1}, r^j) \trianglelefteq_{mult} (r^{i-1}, r^i) \in \mathsf{Fin}$, the property $(r^{i-1}, r^j) \in \mathsf{Fin}$ is maintained. So the inductive hypothesis of (*) for $k - j + i$ applies to the derivation

$$r^0 \gtrsim_{gpo} \cdots \gtrsim_{gpo} r^{i-1} \gtrsim_{gpo} r^j \gtrsim_{gpo} \cdots \gtrsim_{gpo} r^k$$

and yields the claim. Case 2: $r^i \trianglerighteq r^j$ for $i = 0$ holds. Then $(r^j, r^k) \trianglelefteq_{mult} (r^0, r^k) \in \mathsf{Fin}$ by premise, and so $(r^j, r^k) \in \mathsf{Fin}$. Hence by inductive hypothesis of (*) for $k - j$ applied to the derivation

$$r^j \gtrsim_{gpo} \cdots \gtrsim_{gpo} r^k$$

*we get $r^j \gtrsim_{gpo} r^k$, from which the claim $r^0 \gtrsim_{gpo} r^k$ is obtained by Lemma A.2. Case 3: $r^i \triangleleft r^j$ holds. Then $r^{j-1} \gtrsim_{gpo} r^j \triangleright r^i$, so $r^{j-1} >_{gpo} r^i$ by Lemma A.1. On the other hand, $(r^{j-1}, r^i) \trianglelefteq_{mult} (r^{j-1}, r^j)$, so $(r^{j-1}, r^i) \in \mathsf{Fin}$. Hence the inductive hypothesis of * for the derivation*

$$r^i \gtrsim_{gpo} \cdots \gtrsim_{gpo} r^{j-1}$$

applies and yields $r^i \gtrsim_{gpo} r^{j-1}$, a contradiction. So henceforth we may assume that no pair of terms in the given derivation is in the subterm relation.

Next we claim that there is an index $0 < j < k$ such that $r^0 \gtrsim_{gpo} r^j \gtrsim_{gpo} r^k$. Case 1: There is an index $0 < j < k$ such that both $(r^0, r^j) \in \mathsf{Fin}$ and $(r^j, r^k) \in \mathsf{Fin}$. Since the derivations

$$r^0 \gtrsim_{gpo} \cdots \gtrsim_{gpo} r^j \qquad \text{and} \qquad r^j \gtrsim_{gpo} \cdots \gtrsim_{gpo} r^k$$

are each strictly shorter than k, $r^0 \gtrsim_{gpo} r^j \gtrsim_{gpo} r^k$ holds by inductive hypothesis of (*). Case 2: For every $0 < j < k$ either (r^0, r^j) or (r^j, r^k) is not in Fin. Since each r^0, r^j, r^k is a subterm of one of s, t, u, this can happen only if one of r^0, r^k is element of (s, t, u). The other must then be a subterm of another element of (s, t, u) to satisfy $(r^0, r^k) \in \mathsf{Fin}$. But then r^j must be the remaining third term of the multiset (s, t, u). This means that r^j is the same for all j in question. As we may rely on the absence of duplicates, we may conclude that $k = 2$ holds, and our claim follows immediately.

By definition of $\gtrsim_{trans}$, we know that each r^0, r^j, r^k is a subterm of one of s, t, u. Applying Lemma A.4 to (r^0, r^j, r^k), we have $(r^0, r^j, r^k) \trianglelefteq_{mult} (s, t, u)$. By Lemma A.5, the comparison even is strict. So the inductive hypothesis of our transitivity claim applies to this triple, and we may infer $r^0 \gtrsim_{gpo} r^k$.

This finishes the proof of (*).

Lemma A.7. *$>_{gpo}$ terminates.*

Proof. *Termination is proven by "minimal counterexample" [25, 10]. Assume that there is an infinite derivation $t^1 >_{gpo} t^2 >_{gpo} \cdots$. Given a nonempty set of infinite derivations one can approximate a minimal infinite derivation in the following sense: For all $i \in \mathbb{N}$, every derivation that starts with $t^1 >_{gpo} \ldots >_{gpo} t^{i-1} >_{gpo} t'$, satisfies $t' \ntriangleleft t^i$. The infinite sequence is approximated by successively constructing finite prefixes $t^1 >_{gpo} \cdots >_{gpo} t^i$. The prefix is trivial for $i = 0$ and is prolonged from i to $i + 1$ by choosing t^{i+1} minimal w.r.t. $\trianglelefteq$ among all infinite derivations that start with $t^1 >_{gpo} \ldots >_{gpo} t^i$. We are going to demonstrate by a case analysis on the structure of $>_{gpo}$ that the existence of such a minimal counterexample leads to a contradiction, so that the set of infinite derivations must be empty, and so that $>_{gpo}$ terminates.*

Case 1: Some step $t^i >_{gpo} t^{i+1}$ in the given derivation is due to Case (2) of the definition of $\gtrsim_{gpo}$, say $t^i_j \gtrsim_{gpo} t^{i+1}$. Then the infinite derivation

$$t^1 >_{gpo} \cdots >_{gpo} t^{i-1} >_{gpo} t^i_j >_{gpo} t^{i+2} >_{gpo} \cdots$$

is smaller at i than the given one, a contradiction to the assumption that the given derivation was minimal. Case 2: Case (1) *of the definition of* $\gtrsim_{gpo}$ *is used at each step. We have thus an infinite derivation*

$$t^1 \ \ \mathrm{STAT}(>_{gpo}) \ \ t^2 \ \ \mathrm{STAT}(>_{gpo}) \ \ \cdots \ .$$

The fact that the functional STAT *decreases infinite derivations, gives us an infinite derivation* $u^1 >_{gpo} u^2 >_{gpo} \cdots$ *where for some* $j \geq 1$, $t^j \rhd u^1$ *holds. This is sufficient to construct the infinite derivation*

$$t^1 >_{gpo} \cdots >_{gpo} t^{j-1} >_{gpo} u^1 >_{gpo} u^2 >_{gpo} \cdots$$

which is smaller at j than the given infinite derivation. This again is a contradiction to the assumption that the given derivation is minimal.

Summarizing, $\gtrsim_{gpo}$ is reflexive (Lemma A.3), transitive (Lemma A.6), terminating (Lemma A.7), and has the strict subterm property (remark below Lemma A.3). This finishes the proof of part (1) of Theorem 1.

A.2 Proofs of Section 4

Proof (of Proposition 4.2). *Claim:*

1. *Lexicographic combination preserves status components.*
2. *If* STAT_1, STAT_2 *are strictly prepared for the sets* S_1, S_2, *respectively, then* STAT *is strictly prepared for the set* $S_1 \cup S_2$.

For (1), *let* STAT_1 *and* STAT_2 *be status components, and let* $\mathrm{STAT} = (\mathrm{STAT}_1, \mathrm{STAT}_2)_{lex}$ *be their lexicographic combination.*

If $\gtrsim$ *is a quasiorder then so are* $\mathrm{STAT}_1(\gtrsim)$ *and* $\mathrm{STAT}_2(\gtrsim)$. *From* $s \ \mathrm{STAT}_1(\gtrsim) \ s$ *and* $s \ \mathrm{STAT}_2(\gtrsim) \ s$, *by symmetry* $s \ \mathrm{STAT}_1(\sim) \ s$ *and* $s \ \mathrm{STAT}_2(\sim) \ s$ *follows which is* $s \ \mathrm{STAT}(\sim) \ s$ *by definition of* STAT. *Hence* $\mathrm{STAT}(\gtrsim)$ *is reflexive. To show that* $\mathrm{STAT}(\gtrsim)$ *is transitive, assume* $s \ \mathrm{STAT}(\gtrsim) \ t \ \mathrm{STAT}(\gtrsim) \ u$. *By definition of* STAT, $s \ \mathrm{STAT}_1(\gtrsim) \ t \ \mathrm{STAT}_1(\gtrsim) \ u$ *holds, and so* $s \ \mathrm{STAT}_1(\gtrsim) \ u$. *Case 1:* $s \ \mathrm{STAT}_1(>) \ u$. *Then* $s \ \mathrm{STAT}(\gtrsim) \ t$ *by definition of* STAT. *Case 2:* $s \ \mathrm{STAT}_1(\sim) \ u$. *Then* $s \ \mathrm{STAT}_1(\sim) \ t \ \mathrm{STAT}_1(\sim) \ u$, *and* $s \ \mathrm{STAT}_2(\gtrsim) \ t \ \mathrm{STAT}_2(\gtrsim) \ u$ *according to the definition of* STAT. *So* $s \ \mathrm{STAT}_2(\gtrsim) \ u$ *whence* $s \ \mathrm{STAT}(\gtrsim) \ u$. *So* $\mathrm{STAT}(\gtrsim)$ *is transitive. Thus we have shown that* STAT *preserves quasiorders.*

If $\gtrsim'$ *is defined by*

$$s' \gtrsim' t' \quad \Longleftrightarrow_{\mathrm{def}} \quad s' \gtrsim t' \wedge (s', t') \lhd_{mult} (s, t)$$

then subterm foundedness is shown by the following derivation.

$$\begin{array}{lllll}
 & s \ \ \mathrm{STAT}(\gtrsim) \ \ t & & & \\
\Longleftrightarrow & s \ \ \mathrm{STAT}_1(>) \ \ t \ \ \vee & & & \\
 & s \ \ \mathrm{STAT}_1(\sim) \ \ t & \wedge & s \ \ \mathrm{STAT}_2(\gtrsim) \ \ t & \textit{(defn. } \mathrm{STAT}\textit{)} \\
\Longleftrightarrow & s \ \ \mathrm{STAT}_1(>') \ \ t \ \ \vee & & & \\
 & s \ \ \mathrm{STAT}_1(\sim') \ \ t & \wedge & s \ \ \mathrm{STAT}_2(\gtrsim') \ \ t & \textit{(subt. found. of } \mathrm{STAT}_1, \mathrm{STAT}_2\textit{; Lemma 3.1)} \\
\Longleftrightarrow & s \ \ \mathrm{STAT}(\gtrsim') \ \ t & & & \textit{(defn. } \mathrm{STAT}\textit{)}
\end{array}$$

To show that STAT *is prepared for contexts, let* $s \gtrsim t$ *and* $s \gtrsim_{[\_]} t$ *hold. By preparedness of* STAT_1 *and* STAT_2, *we get* $f(\ldots, s, \ldots) \ \mathrm{STAT}_1(\gtrsim) \ f(\ldots, t, \ldots)$ *as well as*

$f(\ldots,s,\ldots)$ $\mathrm{STAT}_2(\gtrsim)$ $f(\ldots,t,\ldots)$, so $f(\ldots,s,\ldots)$ $\mathrm{STAT}(\gtrsim)$ $f(\ldots,t,\ldots)$, by definition of lexicographic combination.

For the proof of decrease of infinite derivations, assume the infinite derivation t^1 $\mathrm{STAT}(>)$ t^2 $\mathrm{STAT}(>)$ $\cdots$ be given. Case 1: $\mathrm{STAT}_1(>)$ occurs infinitely often in this derivation. Then there is a subsequence t^{k_1} $\mathrm{STAT}_1(>)$ t^{k_2} $\mathrm{STAT}_1(>)$ $\cdots$ with $1 = k_1 < k_2 < \cdots$. By decrease of STAT_1, there is a derivation $u^1 > u^2 > \cdots$ such that $t^j \rhd u^1$ holds for some index j. Case 2: $\mathrm{STAT}_1(>)$ occurs only finitely often. Then there is a subsequence t^N $\mathrm{STAT}_2(>)$ t^{N+1} $\mathrm{STAT}_2(>)$ $\cdots$, and by decrease of STAT_2, again a derivation $u^1 > u^2 > \cdots$ with $t^j \rhd u^1$ for some index j.

For (2), let $(f,i) \in S_1 \cup S_2$, $s > t$, $s \gtrsim_{[\_]} t$. From preparedness for contexts, we already got $f(\ldots,s,\ldots)$ $\mathrm{STAT}(\gtrsim)$ $f(\ldots,t,\ldots)$. If $f(\ldots,s,\ldots)$ $\mathrm{STAT}_1(>)$ $f(\ldots,t,\ldots)$ holds, then we are finished. Otherwise $(f,i) \in S_2$, so $f(\ldots,s,\ldots)$ $\mathrm{STAT}_2(>)$ $f(\ldots,t,\ldots)$.

Proof (of Proposition 4.3). *Claim:*

1. *$\_\pi$ is a status component for every π.*
2. *$\_\pi$ is strictly prepared for the set $\{(f,\pi(f)) \mid f \in \mathcal{F} \wedge \pi(f) \leq \mathsf{arity}(f)\}$.*

Let $\gtrsim$ be a quasiorder. If $\pi(f) < \mathsf{arity}(f)$ then $s_{\pi(f)} \gtrsim s_{\pi(f)}$ by reflexivity of $\gtrsim$. So $f(s_1,\ldots,s_m) \gtrsim_\pi f(s_1,\ldots,s_m)$ by definition of $\gtrsim_\pi$. Hence $\gtrsim_\pi$ is reflexive. To show that it is transitive, let $f(s_1,\ldots,s_m) \gtrsim_\pi g(t_1,\ldots,t_n) \gtrsim_\pi h(u_1,\ldots,u_p)$ be given. If $\pi(f) > m$, $\pi(g) > n$, and $\pi(h) > p$ then clearly $f(s_1,\ldots,s_m) \gtrsim_\pi h(u_1,\ldots,u_p)$. Else $\pi(f) \leq m$, $\pi(g) \leq n$, and $\pi(h) < p$, together with $s_{\pi(f)} \gtrsim t_{\pi(g)} \gtrsim u_{\pi(h)}$ hold. By transitivity of $\gtrsim$ then $s_{\pi(f)} \gtrsim u_{\pi(h)}$, so $f(s_1,\ldots,s_m) \gtrsim_\pi h(u_1,\ldots,u_p)$. Hence $\gtrsim_\pi$ is transitive. This finishes the proof that $\_\pi$ preserves quasiorders.

The subterm foundedness property of $\_\pi$ follows from the fact that $s_{\pi(f)}$ and $t_{\pi(g)}$ are proper subterms of s and t, respectively.

Preparedness for contexts: Assume $s \gtrsim t$. To show $f(\ldots,s,\ldots) \gtrsim_\pi f(\ldots,t,\ldots)$, we distinguish cases. Let the argument position of s in $f(\ldots,s,\ldots)$ be i. Case 1: $i = \pi(f)$. Then the claim is equivalent to $s \gtrsim t$, by definition of $\gtrsim_\pi$, and so true by premise. Moreover we observe strict preparedness in this case: If $s > t$ then $f(\ldots,s,\ldots) >_\pi f(\ldots,t,\ldots)$. Case 2: $i \neq \pi(f)$. Then by definition of $\gtrsim_\pi$ the claim is true if $\pi(f) > \mathsf{arity}(f)$; and equivalent to $u_{\pi(f)} = u_{\pi(f)}$ otherwise.

To prove decrease of infinite derivations, let

$$t^1 >_\pi t^2 >_\pi \cdots$$

be given where each term t^i is of the form $f^i(t^i_1,\ldots,t^i_{\mathsf{arity}(f^i)})$. By definition of $\gtrsim_\pi$, this is equivalent to the derivation

$$t^1_{\pi(f^1)} > t^2_{\pi(f^2)} > \cdots$$

where clearly $t^1_{\pi(f^1)}$ is a proper subterm of t^1.

Proof (of Proposition 4.4). *Claim:*

1. *$\_ \restriction S$ preserves status components.*
2. *If STAT is strictly prepared for the set S' then $\mathrm{STAT} \restriction S$ is strictly prepared for the set $\{(f,i) \mid f \in S \wedge (f,i) \in S'\}$.*

Let STAT a status component. Let $\gtrsim$ be a quasiorder. Then $\mathrm{STAT}(\gtrsim)$ is reflexive and transitive by premise. We are going to prove that $\mathrm{STAT} \restriction S$ is reflexive and transitive as well. s $\mathrm{STAT}(\gtrsim)$ s holds by reflexivity of $\mathrm{STAT}(\gtrsim)$. Then s $(\mathrm{STAT} \restriction S)(\gtrsim)$ s obviously holds. To prove transitivity, let

$$f(s_1,\ldots,s_m) \quad (\mathrm{STAT} \restriction S)(\gtrsim) \quad g(t_1,\ldots,t_n) \quad (\mathrm{STAT} \restriction S)(\gtrsim) \quad h(u_1,\ldots,u_p) \ .$$

Either $f, g, h \notin S$, in which case $f(s_1, \ldots, s_m)$ $(\mathrm{STAT} \upharpoonright S)(\gtrsim)$ $h(u_1, \ldots, u_p)$ immediately follows. Or $f, g, h \in S$ and $f(s_1, \ldots, s_m)$ $\mathrm{STAT}(\gtrsim)$ $g(t_1, \ldots, t_n)$ $\mathrm{STAT}(\gtrsim)$ $h(u_1, \ldots, u_p)$ hold. In that case, $f(s_1, \ldots, s_m)$ $\mathrm{STAT}(\gtrsim)$ $h(u_1, \ldots, u_p)$ follows by transitivity of $\mathrm{STAT}(\gtrsim)$, and $f(s_1, \ldots, s_m)$ $(\mathrm{STAT} \upharpoonright S)(\gtrsim)$ $h(u_1, \ldots, u_p)$ holds as well. So $\mathrm{STAT}(\gtrsim)$ S is a quasiorder which finishes the proof that $\mathrm{STAT} \upharpoonright S$ preserves quasiorders.

Subterm foundedness of $\mathrm{STAT} \upharpoonright S$ follows immediately from subterm foundedness of STAT.

To show preparedness for contexts, assume $s \gtrsim t$. Case 1: $f \in S$. Then by preparedness of STAT, *we have $f(\ldots, s, \ldots)$ $\mathrm{STAT}(\gtrsim)$ $f(\ldots, t, \ldots)$, and so $f(\ldots, s, \ldots)$ $(\mathrm{STAT} \upharpoonright S)(\gtrsim)$ $f(\ldots, t, \ldots)$. Case 2: $f \notin S$. Then $f(\ldots, s, \ldots)$ $(\mathrm{STAT} \upharpoonright S)(\gtrsim)$ $f(\ldots, t, \ldots)$ immediatly by definition of $\upharpoonright$.*

Observe that s $(\mathrm{STAT} \upharpoonright S)(>)$ t is equivalent to: s and t have top symbols in S and s $\mathrm{STAT}(>)$ t. Decrease of infinite derivations is now proven as follows. Suppose given an infinite derivation

$$t^1 \quad (\mathrm{STAT} \upharpoonright S)(>) \quad t^2 \quad (\mathrm{STAT} \upharpoonright S)(>) \quad \cdots \ .$$

This can only happen if every t^i has top function symbol in S and t^1 $\mathrm{STAT}(>)$ t^2 $\mathrm{STAT}(>)$ $\cdots$ holds. The claim follows by decrease of infinite derivations of STAT.

If $f \in S$ and $(f, i) \in S'$ then we have strict preparedness. For, in the case $f \in S$, if $s > t$ then by strict preparedness of STAT, *$f(\ldots, s, \ldots)$ $\mathrm{STAT}(>)$ $f(\ldots, t, \ldots)$ holds; this is equivalent to $f(\ldots, s, \ldots)$ $(\mathrm{STAT} \upharpoonright S)(>)$ $f(\ldots, t, \ldots)$. This finishes the proof.*

B `LIFT`: A Medium-scale Example

In order to demonstrate the descriptive power and the line of reasoning of the improved general path order, we will perform a realistic, medium-scale proof of termination of the term rewriting system `LIFT`. The example rewrite system models a toy lift control, one of the favourite examples in the area of formalization of distributed systems [5]. The rewrite system is, apart from minor changes, the same as Ulrich Fraus's [16]. Unusually, the rewrite rules do not express an equational theory but a state transition relation.

The lift control is informally specified as follows. Imagine a building having three floors. A lift cabin moves up and down between these floors, and sometimes stops at a floor. At each floor, there is a button to call the lift cabin, and a light to indicate that the button has been pressed, but the cabin has not visited the floor since. The task of the lift control is to indicate by the button lights that a request has been recognized, and to send the lift cabin to a floor and open the cabin door there at least once after its button has been pressed.

The lift control is modelled by a simply typed rewrite system. We do not model infinite behaviour. First we give the signature, i.e. the types and typed function symbols, of the lift control.

B.1 The Signature

Let there be the type `Bool` of Boolean values, with constants `true`, `false`, and a binary function `or`. We will use Booleans to indicate states of lights (`true` means "on", and `false` means "off"), as well as states of buttons (then `true` means "pressed", and `false` means "not pressed").

Histories of inputs to the lift are modelled by the type `Input`, with a constant `empty` for "no input", and a function symbol

$$\texttt{newbuttons} : \texttt{Bool}^3 \times \texttt{Input} \rightarrow \texttt{Input}$$

where $\texttt{newbuttons}(i1, i2, i3, i)$ tells which buttons are pressed at the moment, $i1$ for the basement button, $i2$, for the first floor button, and $i3$ for the second floor button. What happens after, is recorded in i.

The modelling of simultaneously pressed buttons is realistic for the following reasons. For the lift, requests come in packages. This is so because input must be buffered as long as the lift is busy. Each time the lift has finished a step, and is aware of further input, several buttons may have been pressed in the meantime. It does not matter how many times buttons have been pressed, or in which order. So the input relevant to the lift control is the set of buttons pressed, encoded in the triple $(i1, i2, i3)$ of truth values.

The floors where the lift may be are modelled as constants of type `Floor`,

$$\mathtt{B}, \mathtt{BF}, \mathtt{F}, \mathtt{FS}, \mathtt{S} : \mathtt{Floor}$$

with `B` for basement, `F` for first floor, and `S` for second floor, and the intermediate locations, `BF`, between basement and first floor, and `FS`, between first and second floor. The latter two are necessary to model the situation when the lift is during a move from one (proper) floor to another.

The door of the lift cabin can be `open` or `closed`. These are constants of type `Door`. The lift can move `up`, `down`, or it can `stop`. These are constants of type `Move`.

Lift states are objects of type `State`. There are constants `correct`, `incorrect`, for correct and incorrect termination of the lift, respectively. Here correct termination means that the lift has successfully treated each request. Incorrect termination means that the lift has become unreliable and is for safety reasons set out of order. As long as the lift has not yet terminated, it switches between two modes: Working (`busy`), and waiting for input (`idle`). Both are modelled as function symbols of type

$$\mathtt{Floor} \times \mathtt{Door} \times \mathtt{Move} \times \mathtt{Bool}^3 \times \mathtt{Input} \to \mathtt{State}$$

where the first argument denotes the floor where the lift currently is, the second, whether the door is open or closed, the third, how the lift is moving. The three Boolean arguments indicate the state of the floor lights: Whether the light is on for the basement, the first floor, the second floor, respectively. The last argument tells which pressed button triples the lift still has to face. A function symbol $\mathtt{start} : \mathtt{Input} \to \mathtt{State}$ gives an initialized lift.

B.2 The Term Rewriting System LIFT

The 41 term rewriting rules in Table 1 form a rewrite system which we will call `LIFT`. The variables in these rules range as follows.

$$fl : \mathtt{Floor}, \quad d : \mathtt{Door}, \quad m : \mathtt{Move}, \quad i : \mathtt{Input}, \quad b, b1, b2, b3, i1, i2, i3 : \mathtt{Bool}$$

The lift starts at the first floor with its door closed. All lights are put off and the lift does not move (1). The lift behaves incorrectly if it stops between floors (2), (3), or moves while the door is open (4), (5). The lift finishes correctly as soon as the cabin stands at some floor with the door closed, and no more requests are pending (6), (7), (8). If the cabin stands at some floor with the door closed, and no lights are on, then the lift waits until new buttons are pressed (9), (10), (11). If the stopped lift is not requested at its current floor, it closes the door (12), (13), (14), else it opens the door (it might be open already) and clears the request (15), (16), (17). The lift does not try to go beyond the basement or the second floor (18), (19). Otherwise the lift stops as soon as it comes around a requested floor (20), ..., (23). At floors that are not requested (24), ..., (27) and between floors (28), ..., (31) the lift keeps on moving. A stopped lift with closed door and no request at the current floor starts going to satisfy some request at another floor (32), ..., (37). After each step the lift is looking for new input. If there is no more input it continues working (38), else it consumes the new input. If a button $i1$, $i2$, or $i3$ is pressed then the corresponding light $b1$, $b2$, $b3$ is set on (39). This is achieved by the `or` connective (40), (41).

$$\texttt{start}(i) \to \texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, i) \tag{1}$$

$$\texttt{busy}(\texttt{BF}, d, \texttt{stop}, b1, b2, b3, i) \to \texttt{incorrect} \tag{2}$$
$$\texttt{busy}(\texttt{FS}, d, \texttt{stop}, b1, b2, b3, i) \to \texttt{incorrect} \tag{3}$$

$$\texttt{busy}(fl, \texttt{open}, \texttt{up}, b1, b2, b3, i) \to \texttt{incorrect} \tag{4}$$
$$\texttt{busy}(fl, \texttt{open}, \texttt{down}, b1, b2, b3, i) \to \texttt{incorrect} \tag{5}$$

$$\texttt{busy}(\texttt{B}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{empty}) \to \texttt{correct} \tag{6}$$
$$\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{empty}) \to \texttt{correct} \tag{7}$$
$$\texttt{busy}(\texttt{S}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{empty}) \to \texttt{correct} \tag{8}$$

$$\begin{aligned}&\texttt{busy}(\texttt{B}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{newbuttons}(i1, i2, i3, i)) \to\\ &\quad\texttt{idle}(\texttt{B}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{newbuttons}(i1, i2, i3, i))\end{aligned} \tag{9}$$
$$\begin{aligned}&\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{newbuttons}(i1, i2, i3, i)) \to\\ &\quad\texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{newbuttons}(i1, i2, i3, i))\end{aligned} \tag{10}$$
$$\begin{aligned}&\texttt{busy}(\texttt{S}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{newbuttons}(i1, i2, i3, i)) \to\\ &\quad\texttt{idle}(\texttt{S}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{false}, \texttt{newbuttons}(i1, i2, i3, i))\end{aligned} \tag{11}$$

$$\texttt{busy}(\texttt{B}, \texttt{open}, \texttt{stop}, \texttt{false}, b2, b3, i) \to \texttt{idle}(\texttt{B}, \texttt{closed}, \texttt{stop}, \texttt{false}, b2, b3, i) \tag{12}$$
$$\texttt{busy}(\texttt{F}, \texttt{open}, \texttt{stop}, b1, \texttt{false}, b3, i) \to \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{stop}, b1, \texttt{false}, b3, i) \tag{13}$$
$$\texttt{busy}(\texttt{S}, \texttt{open}, \texttt{stop}, b1, b2, \texttt{false}, i) \to \texttt{idle}(\texttt{S}, \texttt{closed}, \texttt{stop}, b1, b2, \texttt{false}, i) \tag{14}$$

$$\texttt{busy}(\texttt{B}, d, \texttt{stop}, \texttt{true}, b2, b3, i) \to \texttt{idle}(\texttt{B}, \texttt{open}, \texttt{stop}, \texttt{false}, b2, b3, i) \tag{15}$$
$$\texttt{busy}(\texttt{F}, d, \texttt{stop}, b1, \texttt{true}, b3, i) \to \texttt{idle}(\texttt{F}, \texttt{open}, \texttt{stop}, b1, \texttt{false}, b3, i) \tag{16}$$
$$\texttt{busy}(\texttt{S}, d, \texttt{stop}, b1, b2, \texttt{true}, i) \to \texttt{idle}(\texttt{S}, \texttt{open}, \texttt{stop}, b1, b2, \texttt{false}, i) \tag{17}$$

$$\texttt{busy}(\texttt{B}, \texttt{closed}, \texttt{down}, b1, b2, b3, i) \to \texttt{idle}(\texttt{B}, \texttt{closed}, \texttt{stop}, b1, b2, b3, i) \tag{18}$$
$$\texttt{busy}(\texttt{S}, \texttt{closed}, \texttt{up}, b1, b2, b3, i) \to \texttt{idle}(\texttt{S}, \texttt{closed}, \texttt{stop}, b1, b2, b3, i) \tag{19}$$

$$\texttt{busy}(\texttt{B}, \texttt{closed}, \texttt{up}, \texttt{true}, b2, b3, i) \to \texttt{idle}(\texttt{B}, \texttt{closed}, \texttt{stop}, \texttt{true}, b2, b3, i) \tag{20}$$
$$\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{up}, b1, \texttt{true}, b3, i) \to \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{stop}, b1, \texttt{true}, b3, i) \tag{21}$$
$$\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{down}, b1, \texttt{true}, b3, i) \to \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{stop}, b1, \texttt{true}, b3, i) \tag{22}$$
$$\texttt{busy}(\texttt{S}, \texttt{closed}, \texttt{down}, b1, b2, \texttt{true}, i) \to \texttt{idle}(\texttt{S}, \texttt{closed}, \texttt{stop}, b1, b2, \texttt{true}, i) \tag{23}$$

$$\texttt{busy}(\texttt{B}, \texttt{closed}, \texttt{up}, \texttt{false}, b2, b3, i) \to \texttt{idle}(\texttt{BF}, \texttt{closed}, \texttt{up}, \texttt{false}, b2, b3, i) \tag{24}$$
$$\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{up}, b1, \texttt{false}, b3, i) \to \texttt{idle}(\texttt{FS}, \texttt{closed}, \texttt{up}, b1, \texttt{false}, b3, i) \tag{25}$$
$$\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{down}, b1, \texttt{false}, b3, i) \to \texttt{idle}(\texttt{BF}, \texttt{closed}, \texttt{down}, b1, \texttt{false}, b3, i) \tag{26}$$
$$\texttt{busy}(\texttt{S}, \texttt{closed}, \texttt{down}, b1, b2, \texttt{false}, i) \to \texttt{idle}(\texttt{FS}, \texttt{closed}, \texttt{down}, b1, b2, \texttt{false}, i) \tag{27}$$

$$\texttt{busy}(\texttt{BF}, \texttt{closed}, \texttt{up}, b1, b2, b3, i) \to \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{up}, b1, b2, b3, i) \tag{28}$$
$$\texttt{busy}(\texttt{BF}, \texttt{closed}, \texttt{down}, b1, b2, b3, i) \to \texttt{idle}(\texttt{B}, \texttt{closed}, \texttt{down}, b1, b2, b3, i) \tag{29}$$
$$\texttt{busy}(\texttt{FS}, \texttt{closed}, \texttt{up}, b1, b2, b3, i) \to \texttt{idle}(\texttt{S}, \texttt{closed}, \texttt{up}, b1, b2, b3, i) \tag{30}$$
$$\texttt{busy}(\texttt{FS}, \texttt{closed}, \texttt{down}, b1, b2, b3, i) \to \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{down}, b1, b2, b3, i) \tag{31}$$

$$\texttt{busy}(\texttt{B}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{true}, b3, i) \to \texttt{idle}(\texttt{B}, \texttt{closed}, \texttt{up}, \texttt{false}, \texttt{true}, b3, i) \tag{32}$$
$$\texttt{busy}(\texttt{B}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{true}, i) \to \texttt{idle}(\texttt{B}, \texttt{closed}, \texttt{up}, \texttt{false}, \texttt{false}, \texttt{true}, i) \tag{33}$$
$$\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{stop}, \texttt{true}, \texttt{false}, b3, i) \to \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{down}, \texttt{true}, \texttt{false}, b3, i) \tag{34}$$
$$\texttt{busy}(\texttt{F}, \texttt{closed}, \texttt{stop}, \texttt{false}, \texttt{false}, \texttt{true}, i) \to \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{up}, \texttt{false}, \texttt{false}, \texttt{true}, i) \tag{35}$$
$$\texttt{busy}(\texttt{S}, \texttt{closed}, \texttt{stop}, b1, \texttt{true}, \texttt{false}, i) \to \texttt{idle}(\texttt{S}, \texttt{closed}, \texttt{down}, b1, \texttt{true}, \texttt{false}, i) \tag{36}$$
$$\texttt{busy}(\texttt{S}, \texttt{closed}, \texttt{stop}, \texttt{true}, \texttt{false}, \texttt{false}, i) \to \texttt{idle}(\texttt{S}, \texttt{closed}, \texttt{down}, \texttt{true}, \texttt{false}, \texttt{false}, i) \tag{37}$$

$$\texttt{idle}(fl, d, m, b1, b2, b3, \texttt{empty}) \to \texttt{busy}(fl, d, m, b1, b2, b3, \texttt{empty}) \tag{38}$$

$$\begin{aligned}&\texttt{idle}(fl, d, m, b1, b2, b3, \texttt{newbuttons}(i1, i2, i3, i)) \to\\ &\quad\texttt{busy}(fl, d, m, \texttt{or}(b1, i1), \texttt{or}(b2, i2), \texttt{or}(b3, i3), i)\end{aligned} \tag{39}$$

$$\texttt{or}(\texttt{true}, b) \to \texttt{true} \tag{40}$$
$$\texttt{or}(\texttt{false}, b) \to b \tag{41}$$

Table 1: The rewrite system LIFT.

B.3 What is its Proof of Termination Good for?

Termination of LIFT entails the validity of the principle of *rewriting induction* for LIFT. By rewriting induction Ulrich Fraus [16] formally proved the claim

$$\texttt{LIFT} \vdash \forall i : \texttt{Input}\,.\ \texttt{start}(i) = \texttt{correct} \quad .$$

The proof was supported by the semi-automatic inductive prover TIP [17]. As LIFT has only trivial critical pairs it is confluent. Confluence entails that the normal forms correct and incorrect are semantically distinct, by which from Fraus's result the *safety property* follows that the lift never breaks.

$$\texttt{LIFT} \not\vdash \exists i : \texttt{Input}\,.\ \texttt{start}(i) = \texttt{incorrect}$$

So termination of LIFT is essential for the validity of a mechanically proven safety property.

B.4 Termination of LIFT is Difficult to Prove

We now show that none of the straightforward termination proof methods is able to prove that LIFT terminates. More precisely, we show that no termination order which is total on ground terms, can order LIFT.

Theorem 5. LIFT *is not totally terminating.*

Proof. *Let* $\gtrsim$ *be a termination quasiorder such that* LIFT $\subseteq >$ *holds. Assume* B $\gtrsim$ BF*. Then one gets the derivation*

$$\begin{array}{lll} & \texttt{busy}(\texttt{B},\texttt{closed},\texttt{down},\texttt{false},b2,b3,\texttt{empty}) & \\ \gtrsim & \texttt{busy}(\texttt{BF},\texttt{closed},\texttt{down},\texttt{false},b2,b3,\texttt{empty}) & (\gtrsim \textit{ cl. u. cont.}) \\ > & \texttt{idle}(\texttt{B},\texttt{closed},\texttt{down},\texttt{false},b2,b3,\texttt{empty}) & (\to_{(29)} \subseteq >) \\ > & \texttt{busy}(\texttt{B},\texttt{closed},\texttt{down},\texttt{false},b2,b3,\texttt{empty}) & (\to_{(38)} \subseteq >) \end{array}$$

which contradicts transitivity and reflexivity of $\gtrsim$*. In the same way, using Rule* (24) *instead of* (29)*, one can show that* BF $\gtrsim$ B *leads to a contradiction. So* $\gtrsim$ *cannot be total on ground terms as it cannot order the two constants* BF*,* B *in either orientation.*

Every precedence based order, and every interpretation to a totally ordered set can only prove total termination. For this reason, all these orders, in particular the path and decomposition orders with status, the Knuth/Bendix order, and the polynomial and elementary interpretation orders fail to prove LIFT terminating.

B.5 Construction of a Suitable Status Component

In order to prove termination of LIFT we are going to construct an interpretation, $[\_]$, and a status, STAT, such that the induced general path order, $>_{gpo}$, and the induced quasiorder $\gtrsim_{[\_]}$ satisfy $l >_{gpo} r$ as well as $l \gtrsim_{[\_]} r$ for each rewrite rule $(l \to r) \in$ LIFT.

First we fix the architecture and the syntactic components of the functional STAT. As is usual practice, we put a lexicographic combination, where the first component compares top symbols according to their precedence, and the second component is a status component dependent on the top function symbol.

$$\begin{array}{rcl} \mathrm{STAT} & =_{\mathrm{def}} & (\gtrsim_{prec},\ \mathrm{STAT}_{\texttt{start}} \cap \mathrm{STAT}_{\texttt{busy}} \cap \mathrm{STAT}_{\texttt{newbuttons}} \cap \mathrm{STAT}_{\texttt{or}})_{lex} \\ \mathrm{STAT}_{\texttt{start}} & =_{\mathrm{def}} & {}_{-1} \restriction \texttt{start} \\ \mathrm{STAT}_{\texttt{busy}} & =_{\mathrm{def}} & ({}_{-7}, {}_{-4}, {}_{-5}, {}_{-6}, {}_{-2}, \gtrsim_\phi, {}_{-1}, {}_{-3}, \gtrsim_\psi)_{lex} \restriction \{\texttt{busy}, \texttt{idle}\} \\ \mathrm{STAT}_{\texttt{newbuttons}} & =_{\mathrm{def}} & ({}_{-4}, {}_{-1}, {}_{-2}, {}_{-3})_{lex} \restriction \texttt{newbuttons} \\ \mathrm{STAT}_{\texttt{or}} & =_{\mathrm{def}} & ({}_{-1}, {}_{-2})_{lex} \restriction \texttt{or} \end{array}$$

As the precedence we choose

$\texttt{start} >_{prec} \texttt{F}, \texttt{closed}, \texttt{stop}, \texttt{false}$
$\texttt{start} >_{prec} \texttt{busy} \sim_{prec} \texttt{idle} >_{prec} \texttt{correct}, \texttt{incorrect}$
$\texttt{true} >_{prec} \texttt{false}$
$\texttt{open} >_{prec} \texttt{closed}.$

Below we give a motivation for this choice. I think it is possible to automate construction of all components but $\gtrsim_\phi$.

1. Rules (1), ..., (8) are easily ordered by the precedence. The `or`-rules (40), (41) are ordered by the strict subterm property of $\gtrsim_{gpo}$.

2. For the precedence, $\texttt{busy} \sim_{prec} \texttt{idle}$ is a good choice. Not to order `busy` and `idle` in the precedence would anyway block the comparison. To order $\texttt{idle} >_{prec} \texttt{busy}$ instead would order rules (9), (10), and (11) in their reversed direction, and to order $\texttt{busy} >_{prec} \texttt{idle}$ would do the same with rule (38). As rules (9), (10), and (11) and (38) do not change their argument tuple from the left to the right hand side we may delay ordering these rules until the last component of the status. There a measure ψ will order them.

3. The last argument of `busy` (and of `idle`), i, either remains constant, or decreases by the strict subterm property of $>_{gpo}$, along a rule application. So we may take as first component in the tuple of status components, the selector $\_7$ that selects the 7th argument, to be compared recursively by $\gtrsim_{gpo}$. This cares for Rule (39).

4. Next observe that in the remaining rules, each button state either remains constant, or changes from `true` to `false`. This change becomes a decrease as soon as $\texttt{true} >_{prec} \texttt{false}$ is added to the precedence and the arguments 4, 5, and 6 are selected in the status component. This handles rules (15), (16), and (17).

5. In the same way, adding $\texttt{open} >_{prec} \texttt{closed}$ to the precedence, and putting $\_2$ as next component of the status component, one gets rid of the rules (12), (13), and (14).

6. The rules (18), ..., (37) are difficult to order. A semantic component is necessary. We are going to construct the measure ϕ to satisfy $l >_\phi r$ for these.

7. The selectors $\_1$, $\_3$ are added to ensure that STAT is strictly prepared for the positions 1 and 3 of `busy` and `idle` each.

Summarizing, we have argued why every rule is ordered by the improved general path order. Table 2 gives an overview. Under the proviso that an interpretation $[\_]$ and two measures ϕ, ψ will be defined suitably, the following statements can be made.

Lemma B.1. STAT *is a status.*

Proof. *Particularly,* STAT *is strictly prepared for every argument position i of every function symbol f. Check that the corresponding argument selector $\_i$ occurs in the tuple restricted by $\restriction \{f, \dots\}$. For instance,* STAT *is strictly prepared for* $(\texttt{idle}, 1)$ *because* $\_1$ *occurs as last but two component in* $\mathrm{STAT}_{\texttt{busy}}$.

Lemma B.2. $\texttt{LIFT} \subseteq >_{gpo}$.

Rule	$\gtrsim_{prec}$	-7	-4	-5	-6	-2	$\gtrsim_{\phi}$	-1	-3	$\gtrsim_{\psi}$	comments
(1), ..., (8)	$>$										
(9), (10), (11)	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$>$	Lemma (B.6)
(12), (13), (14)	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$>$					`open` $>_{prec}$ `closed`
(15)	$\sim$	$\sim$	$>$								`true` $>_{prec}$ `false`
(16)	$\sim$	$\sim$	$\sim$	$>$							`true` $>_{prec}$ `false`
(17)	$\sim$	$\sim$	$\sim$	$\sim$	$>$						`true` $>_{prec}$ `false`
(18), ..., (37)	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$>$				Lemma (B.5)
(38)	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$>$	Lemma (B.6)
(39)	$\sim$	$>$									
(40)	$>$										
(41)											by $\rhd \subseteq \gtrsim_{gpo}$

Legend: The table summarizes the justifications of l STAT$(>_{gpo})$ r for every rule $l \to r$ of `LIFT`. Each rule $l \to r$ is represented by a row, and each component of STAT by a column. The symbols "$>$" and "$\sim$" denote the result of comparison of the component.

Table 2: Overview of the Status Comparison.

B.6 Choosing an Interpretation

The lift achieves a progress when it approaches its target. On this account the interpretation has to give enough information to determine this progress. Such information may be button states, movements, and floors. Values for inputs will be needed to define measure ψ later.

We partition the data domain $\mathcal{D}$ into carrier sets $\mathcal{D}_s$, indexed by types s, and an extra element $\perp$ to denote the value of ill-typed, undefined, or irrelevant terms.

$$\mathcal{D} =_{\text{def}} \mathcal{D}_{\texttt{Floor}} + \mathcal{D}_{\texttt{Door}} + \mathcal{D}_{\texttt{Input}} + \mathcal{D}_{\texttt{Move}} + \mathcal{D}_{\texttt{Bool}} + \mathcal{D}_{\texttt{State}} + \{\perp\}$$

We require that $[t] \in \mathcal{D}_s + \{\perp\}$ if t is a well-typed ground term of type s. As the types `Door` and `State` are irrelevant for the interpretations, we may define $\mathcal{D}_{\texttt{Door}} = \mathcal{D}_{\texttt{State}} = \emptyset$. For the sake of simplicity, we identify elements of $\mathcal{D}$ with function symbols, so we obtain

$$\begin{aligned}
\mathcal{D}_{\texttt{Floor}} &=_{\text{def}} \{\texttt{B}, \texttt{BF}, \texttt{F}, \texttt{FS}, \texttt{S}\}, \\
\mathcal{D}_{\texttt{Move}} &=_{\text{def}} \{\texttt{up}, \texttt{stop}, \texttt{down}\}, \\
\mathcal{D}_{\texttt{Bool}} &=_{\text{def}} \{\texttt{true}, \texttt{false}\}, \\
\mathcal{D}_{\texttt{Input}} &=_{\text{def}} \{\texttt{empty}, \texttt{newbuttons}\}
\end{aligned}$$

Accordingly we choose the interpretation

$$\begin{aligned}
&[c] =_{\text{def}} c \qquad \text{for every constant } c \text{ except } \texttt{open} \text{ and } \texttt{closed}, \\
&[\texttt{open}] =_{\text{def}} [\texttt{closed}] =_{\text{def}} \perp \\
&[\texttt{newbuttons}](i1, i2, i3, i) =_{\text{def}} \texttt{newbuttons}, \\
&[\texttt{start}](i) =_{\text{def}} [\texttt{busy}](\text{fl}, d, m, b1, b2, b3, i) =_{\text{def}} [\texttt{idle}](\text{fl}, d, m, b1, b2, b3, i) =_{\text{def}} \perp \\
&[\texttt{or}](y, x) =_{\text{def}} \begin{cases} \texttt{true}, & \text{if } y = \texttt{true}, \\ x, & \text{if } y = \texttt{false}, \\ \perp, & \text{else} \end{cases}
\end{aligned}$$

For simplicity we choose $\gtrsim_{\mathcal{D}} =_{\text{def}} =$, the equality on $\mathcal{D}$. Thus, $\gtrsim_{[\_]}$ is trivially closed under contexts, for $[\_]$ is a homomorphism. Now every rule $(l \to r) \in$ `LIFT` except the `or`-rules satisfies $[l\sigma] = \perp = [r\sigma]$ for every ground substitution σ. By the natural interpretation for `or`, the two remaining rules (40) and (41) satisfy $[l\sigma] = \texttt{true} = [r\sigma]$ and $[l\sigma] = [b\sigma] = [r\sigma]$, respectively. So we have `LIFT` $\subseteq \sim_{[\_]}$.

Lemma B.3. *$(\mathcal{D}, =, [\_])$ is a model of* LIFT.

B.7 How to Express "Progress" of the Lift

We are going to develop a measure ϕ that yields the "job effort", i.e. some number that decreases when a step is undertaken to complete the current job.

A good starting point is to think about the rules as an "operational semantics" of the lift: Each rule expresses a step the lift may do. The task of the lift is to solve requests. Each step has to mirror some progress towards solution of one of the requests pending. Intuitively, the lift turns towards a nearest floor (there may be more than one) whose button is lit, moves to this floor, and stops. If the nearest light is upstairs, then the lift must change from `stop` or `down` to `up`, and then pass the floors upstairs. If it is downstairs, then the lift must change its movement from `stop` or `up` to `down`, and then pass the floors downstairs. Thus the lift gets nearer to the completion of its next job. Ulrich Fraus [16] constructed ad-hoc a "job completion order" to formalize this idea, but did not succeed in proving that the job completion order terminates.

Let ϕ^f be the constant 0 function, unless $f \in \{\texttt{busy}, \texttt{idle}\}$. Now let us develop $\phi^{\texttt{busy}} = \phi^{\texttt{idle}}$. In order to calculate with distances, we use a function $\# : \mathcal{D}_{\texttt{Move}} + \mathcal{D}_{\texttt{Floor}} \to \mathbb{Z}$ to assign numeric values to movements and to floors.

$$\#\,\texttt{up} = 1, \quad \#\,\texttt{stop} = 0, \quad \#\,\texttt{down} = -1$$
$$\#\,\texttt{B} = 0, \quad \#\,\texttt{BF} = 1, \quad \#\,\texttt{F} = 2, \quad \#\,\texttt{FS} = 3, \quad \#\,\texttt{S} = 4$$

If any button is lit, i.e. if $(b1, b2, b3) \neq (\texttt{false}, \texttt{false}, \texttt{false})$ holds, then the lift should consider one of the requested floors as its next target. Otherwise, as we will argue below, the lift pretends as if it had a set of "virtual" targets. Anticipating this exceptional case, the set of targets to the lift is given by the function

$$\mathsf{targets} : \mathcal{D}^3_{\texttt{Bool}} \times \mathcal{D}_{\texttt{Move}} \to \mathfrak{P}(\mathcal{D}_{\texttt{Floor}})$$

defined by

$$\mathsf{targets}(b1, b2, b3, m) =_{\text{def}}$$
$$\begin{cases} \{\texttt{B}\}, & \text{if } m = \texttt{down} \text{ and } b1 = b2 = b3 = \texttt{false}, \\ \{\texttt{B}, \texttt{S}\}, & \text{if } m = \texttt{stop} \text{ and } b1 = b2 = b3 = \texttt{false}, \\ \{\texttt{S}\}, & \text{if } m = \texttt{up} \text{ and } b1 = b2 = b3 = \texttt{false}, \\ \{\texttt{B} \mid b1 = \texttt{true}\} \;\cup\; \{\texttt{F} \mid b2 = \texttt{true}\} \;\cup\; \{\texttt{S} \mid b3 = \texttt{true}\}, & \text{else} . \end{cases}$$

We may expect that the lift has a certain notion of distance to tell which floor it prefers. The lift should turn and then move towards some preferred floor in the target set. Next we are going to define such a distance.

Assume given the ordinary case $(b1, b2, b3) \neq (\texttt{false}, \texttt{false}, \texttt{false})$ for the moment. Let fl denote the floor where the lift cabin is, and $\text{fl}' \in \mathsf{targets}(b1, b2, b3, m)$ some target floor where the lift may try to go next. The lift gets nearer to this target if it decreases the distance to it. Naively the distance between floors is given by the expression $|\#\,\text{fl} - \#\,\text{fl}'\,|$. This is o.k. if the lift has stopped. However a view at rules (24), ..., (31) shows that a *moving* lift obeys quite a different metric.

Example 11. *We would like to validate the inequation*

$$\begin{array}{rl} & \texttt{busy}(\texttt{BF}, \texttt{closed}, \texttt{up}, \texttt{true}, \texttt{false}, \texttt{false}, i) \\ \gtrsim_\phi & \texttt{idle}(\texttt{F}, \texttt{closed}, \texttt{up}, \texttt{true}, \texttt{false}, \texttt{false}, i) \end{array}$$

for Rule (28). *According to our intuition only the basement can be the target of the lift. Strangely the lift does* not *turn downwards but continues moving up, although it thereby increases the naive distance* $|\#\,\mathtt{BF} - \#\,\mathtt{B}\,| = 1 < 2 = |\#\,\mathtt{F} - \#\,\mathtt{B}\,|$. *Finally arrived at the second floor, the lift will of course turn down and will then go back to visit the basement floor.*

As the rules impressively show, when the lift moves it always prefers to keep moving. A distance measure that decreases at each step of this travel essentially has to express the length of the travel. For this purpose it must consider floors in the back of the lift more distant than any in front. This leads to a move-dependent function

$$\mathsf{dist} : \mathcal{D}^2_{\mathtt{Floor}} \times \mathcal{D}_{\mathtt{Move}} \to \{0, \ldots, 7\},$$

defined by

$$\mathsf{dist}(\mathrm{fl}, \mathrm{fl}', m) =_{\mathrm{def}} \begin{cases} |\#\,\mathrm{fl} - \#\,\mathtt{S}\,| + |\#\,\mathtt{S} - \#\,\mathrm{fl}'\,|, & \text{if } m = \mathtt{up} \text{ and } \#\,\mathrm{fl}' < \#\,\mathrm{fl}, \\ |\#\,\mathrm{fl} - \#\,\mathtt{B}\,| + |\#\,\mathtt{B} - \#\,\mathrm{fl}'\,|, & \text{if } m = \mathtt{down} \text{ and } \#\,\mathrm{fl} < \#\,\mathrm{fl}', \\ |\#\,\mathrm{fl} - \#\,\mathrm{fl}'\,|, & \text{else} \end{cases}$$

Now indeed the distance to the nearest floor in the nonempty target set decreases for all rules (24), ..., (31) and all triples $(b1, b2, b3) \neq (\mathtt{false}, \mathtt{false}, \mathtt{false})$.

For the rules which handle a stopping (18), ..., (23) and starting (32), ..., (37) of the lift, the lights and the floor do not change. For these rules by definition the distance remains constant. It is the *change of movement* that decreases. If the expression $\#\,\mathrm{fl} - \#\,\mathrm{fl}'$ is positive, then the lift is located above its target, and it has to go down. If the expression is negative, then the lift is below, and it has to turn upwards. In the case of zero, it has reached the target floor, and it should stop. Now, consider the expression $\mathsf{sign}(\#\,\mathrm{fl} - \#\,\mathrm{fl}') + \#m$ where m denotes the movement of the lift, and sign is the function that returns 1 for a positive argument, 0 for zero, and -1 otherwise. The better the lift is oriented towards its target, the less is the absolute value of this expression. We model, therefore, a function

$$\mathsf{orient} : \mathcal{D}^2_{\mathtt{Floor}} \times \mathcal{D}_{\mathtt{Move}} \to \{0, 1, 2\},$$

defined by

$$\mathsf{orient}(\mathrm{fl}, \mathrm{fl}', m) =_{\mathrm{def}} |\,\mathsf{sign}(\#\,\mathrm{fl} - \#\,\mathrm{fl}') + \#m| \ .$$

As can be checked, the rules (18), ..., (23), and (32), ..., (37), decrease the expression $\mathsf{orient}(\mathrm{fl}, \mathrm{fl}', m)$ for the nearest floor fl'. In total, the pair

$$(\mathsf{dist}(\mathrm{fl}, \mathrm{fl}', m), \mathsf{orient}(\mathrm{fl}, \mathrm{fl}', m))$$

lexicographically decreases for some nearest floor fl', for all rules (18) ..., (37), and triples $(b1, b2, b3) \neq (\mathtt{false}, \mathtt{false}, \mathtt{false})$ of button lights. In order to work with $\mathcal{E} = \mathbb{N}$ and $\gtrsim_{\mathcal{E}} = \geq$, the pairs are replaced by the order-isomorphic expression

$$3 * \mathsf{dist}(\mathrm{fl}, \mathrm{fl}', m) + \mathsf{orient}(\mathrm{fl}, \mathrm{fl}', m) \ .$$

Now let us consider the exceptional case $b1 = b2 = b3 = \mathtt{false}$. This case can happen for the rules (18), (19), and (24), ..., (31). Although the lift is requested nowhere, it does still move, and this way undergoes some rewrite steps. We claim that these can be only finitely many. To explain the strange behaviour of the lift, we assume a "virtual" target, B if the lift moves down or stops, and S if it moves up or stops. So we adopt the same measure as above, but select other fl'.

m	fl	$(b1,b2,b3)$							
		000	001	010	011	100	101	110	111
	`B`	1	14	8	8	1	1	1	1
	`BF`	3	17	11	11	3	3	3	3
`down`	`F`	6	20	1	1	6	6	1	1
	`FS`	9	23	3	3	9	9	3	3
	`S`	12	1	6	1	12	1	6	1
	`B`	0	13	7	7	0	0	0	0
	`BF`	4	10	4	4	4	4	4	4
`stop`	`F`	7	7	0	0	7	7	0	0
	`FS`	4	4	4	4	10	4	4	4
	`S`	0	0	7	0	13	0	7	0
	`B`	12	12	6	6	1	1	1	1
	`BF`	9	9	3	3	23	9	3	3
`up`	`F`	6	6	1	1	20	6	1	1
	`FS`	3	3	11	3	17	3	11	3
	`S`	1	1	8	1	14	1	8	1

Legend: Table of values of

$$\min\{3 * \mathsf{dist}(\mathrm{fl},\mathrm{fl}',m) + \mathsf{orient}(\mathrm{fl},\mathrm{fl}',m) \mid \mathrm{fl}' \in \mathsf{targets}(b1,b2,b3,m)\}$$

listed for all well-typed fl, m, and triples $(b1,b2,b3)$. Triples are abbreviated as three-character strings where $0 = \texttt{false}$, $1 = \texttt{true}$. For example, 001 stands for $(\texttt{false},\texttt{false},\texttt{true})$.

Table 3: Table of Measure Values.

All this expertise about lifts can be coded in the measure ϕ^{busy} provided that all its arguments are well-typed, i.e. $\mathrm{fl} \in \mathcal{D}_{\texttt{Floor}}$, $d \in \mathcal{D}_{\texttt{Door}}$, $m \in \mathcal{D}_{\texttt{Move}}$, $b1,b2,b3 \in \mathcal{D}_{\texttt{Bool}}$ hold.

$$\begin{aligned} &\phi^{\mathsf{busy}}(\mathrm{fl},d,m,b1,b2,b3,i) =_{\mathrm{def}} \\ &\phi^{\mathsf{idle}}(\mathrm{fl},d,m,b1,b2,b3,i) =_{\mathrm{def}} \\ &\min\{3 * \mathsf{dist}(\mathrm{fl},\mathrm{fl}',m) + \mathsf{orient}(\mathrm{fl},\mathrm{fl}',m) \mid \mathrm{fl}' \in \mathsf{targets}(b1,b2,b3,m)\} \end{aligned}$$

Call a substitution $\sigma : \mathcal{X} \to \mathcal{T}$ *well-typed* if $[\mathrm{fl}\,\sigma] \in \mathcal{D}_{\texttt{Floor}}$, $[m\sigma] \in \mathcal{D}_{\texttt{Move}}$, and $[b1\sigma],[b2\sigma],[b3\sigma] \in \mathcal{D}_{\texttt{Bool}}$ hold. Then we have the following result.

Lemma B.4. *For every well-typed ground substitution $\sigma : \mathcal{X} \to \mathcal{GT}$, and every rewrite rule*

$$\texttt{busy}(l_1,\ldots,l_7) \quad \to \quad \texttt{idle}(r_1,\ldots,r_7)$$

in (18) *...,* (37) *of* LIFT*, the value of ϕ strictly decreases:*

$$\phi^{\mathsf{busy}}([l_1\sigma],\ldots,[l_7\sigma]) \quad \gtrsim_{\mathcal{E}} \quad \phi^{\mathsf{idle}}([r_1\sigma],\ldots,[r_7\sigma])$$

Proof. *This can easily be checked for each triple $([b1\sigma],[b2\sigma],[b3\sigma]) \in \mathcal{D}^3_{\texttt{Bool}}$, using Table 3.*

B.8 Typing Issues

To this point, we pretended as if terms were always well-typed. In fact one must be aware of ill-typed terms as well, because termination is claimed for *every* rewrite derivation, not only for well-typed ones. And ill-typed substitutions may introduce ill-typed values. While this is no problem for the syntactic components, the measure function defined above is not aware of ill-typed substitutions.

Example 12. *Let the ground substitution $\sigma : \mathcal{X} \to \mathcal{GT}$ be given where $[b2\sigma] = \mathtt{correct}$, $[b3\sigma] = \mathtt{up}$, $[i\sigma] = \mathtt{stop}$ hold. Obviously, σ is not well-typed. Now consider the proof obligation for Rule* (24)*.*

$$\begin{array}{rl} & \phi^{\mathtt{busy}}(\mathtt{B}, \mathtt{closed}, \mathtt{up}, \mathtt{false}, \mathtt{correct}, \mathtt{up}, \mathtt{stop}) \\ \gtrsim_{\mathcal{E}} & \phi^{\mathtt{idle}}(\mathtt{BF}, \mathtt{closed}, \mathtt{up}, \mathtt{false}, \mathtt{correct}, \mathtt{up}, \mathtt{stop}) \end{array}$$

The definition of ϕ^{busy} above is not prepared to handle this case: The crucial expression $\mathsf{targets}(\mathtt{false}, \mathtt{correct}, \mathtt{up}, \mathtt{up})$ *turns out meaningless.*

How can ϕ^{busy} be extended for ill-typed arguments? There is surprisingly a very simple answer. We only have to assume, for each domain D_s, a $\gtrsim_{\mathcal{D}}$-monotonic coercion function[5] $(\_ :: \mathcal{D}_s) : \mathcal{D} \to \mathcal{D}_s$ such that $(d :: \mathcal{D}_s) = d$ holds whenever $d \in \mathcal{D}_s$. Before the measure is applied, every object is mapped to the desired domain. Thus the case of ill-typed arguments is reduced to the case of well-typed ones. We need not even mention which coercion functions we actually use.

Definition B.1 (The measure function ϕ^{busy}). *Let $(\mathcal{E}, \gtrsim_{\mathcal{E}}) =_{\mathrm{def}} (\mathbb{N}, \geq)$. Then $\phi^{\mathsf{busy}}, \phi^{\mathtt{idle}} : \mathcal{D}^7 \to \mathcal{E}$ are defined by*

$$\begin{array}{l} \phi^{\mathsf{busy}}(d_1, \ldots, d_7) =_{\mathrm{def}} \\ \phi^{\mathtt{idle}}(d_1, \ldots, d_7) =_{\mathrm{def}} \\ \min\{3 * \mathsf{dist}(\mathrm{fl}, \mathrm{fl}', m) + \mathsf{orient}(\mathrm{fl}, \mathrm{fl}', m) \mid \mathrm{fl}' \in \mathsf{targets}(b1, b2, b3, m)\} \enspace , \end{array}$$

where

$$\begin{array}{c} \mathrm{fl} = (d_1 :: \mathcal{D}_{\mathtt{Floor}}), \quad m = (d_3 :: \mathcal{D}_{\mathtt{Move}}), \\ b1 = (d_4 :: \mathcal{D}_{\mathtt{Bool}}), \quad b2 = (d_5 :: \mathcal{D}_{\mathtt{Bool}}), \quad b3 = (d_6 :: \mathcal{D}_{\mathtt{Bool}}) \enspace . \end{array}$$

Example 12 (Continued). *Suppose we choose the coercion mappings such that* $(\mathtt{correct} :: \mathcal{D}_{\mathtt{Bool}}) = \mathtt{false}$ *and* $(\mathtt{up} :: \mathcal{D}_{\mathtt{Bool}}) = \mathtt{true}$*. Then we have $\phi(l\sigma) = 12 > 9 = \phi(r\sigma)$.*

By Lemma B.4, the following is immediate.

Lemma B.5. *$l >_{\phi} r$ holds for every rewrite rule $l \to r$ in* (18), ..., (37)*.*

B.9 Finish of the Proof

To order the remaining four rules (9), (10), (11), and (38), we define a measure function ψ by

$$\psi_{\mathtt{busy}}(d_1, \ldots, d_7) =_{\mathrm{def}} \begin{cases} 0, & \text{if} \quad (d_7 :: \mathcal{D}_{\mathtt{Input}}) = \mathtt{empty} \\ 1, & \text{else} \end{cases}$$

$$\psi_{\mathtt{idle}}(d_1, \ldots, d_7) =_{\mathrm{def}} \begin{cases} 1, & \text{if} \quad (d_7 :: \mathcal{D}_{\mathtt{Input}}) = \mathtt{empty} \\ 0, & \text{else} \end{cases}$$

As codomain for ψ we may take $1 > 0$. The induced quasiorder $\gtrsim_{\psi}$ then satisfies the following.

[5] If $\gtrsim_{\mathcal{D}}$ is wellfounded then such a coercion function always exists. Define $(d :: \mathcal{D}_s) = d'$ where d' is the smallest element in $\mathcal{D}_s$ for which $d \gtrsim d'$.

Lemma B.6. *$l >_\psi r$ holds for every rewrite rule $l \to r$ in* (9)*,* (10)*,* (11)*, and* (38)*.*

Now all obligations for the termination proof of `LIFT` are solved.

Theorem 6. `LIFT` *is a terminating term rewriting system.*

Proof. *We have proven that* `LIFT` $\subseteq >_{gpo} \cap \gtrsim_{[\_]}$ *(by Lemma B.2 and Lemma B.3, resp.), that* $(\mathcal{D}, =, [\_])$ *is a monotonic interpretation (by Lemma B.3), and that* STAT *is a status for* $\gtrsim_{[\_]}$ *(by Lemma B.1). By Theorem 1, the claim follows.*

Conclusion of the Case Study

We have introduced Ulrich Fraus's term rewriting system `LIFT`, consisting of 41 rewrite rules, modelling a simple lift control. We have proved that `LIFT` is a terminating rewrite system, and thus demonstrated the improved general path order introduced in the main part. `LIFT` is not totally terminating, so no precedence based order or interpretation to a totally ordered domain is able to order `LIFT`.

Although we have no evidence that our $\gtrsim_{gpo}$ is strictly more powerful than Dershowitz/Hoot's or Kamin/Lévy's orders, we feel that they are not able to support our proof idea. Kamin/Lévy's semantic path order fails because a semantic comparison, $\gtrsim_\phi$, is needed *after* the recursive call for the 7th argument. Dershowitz/Hoot's version of the general path order fails because Rule (39) satisfies $l\sigma <'_{\mathcal{D}} r\sigma$ for the ground substitution σ defined by $b1\sigma = b2\sigma = b3\sigma = i1\sigma = i2\sigma =$ `false`, $i3\sigma =$ `true`, $fl\sigma =$ `B`, $m\sigma =$ `down`, we have $\phi(l\sigma) = 1 <'_{\mathcal{D}} 14 = \phi(r\sigma)$. Hence neither $\gtrsim_\phi$ nor $\sim_\phi$ can be a quasimodel of `LIFT`.

We do not know whether `LIFT` can be ordered by any other known termination proof method for non-total termination, like e.g. transformation order [4] or semantic labelling [36]. Semantic labelling, followed by a recursive path order with status, lacks the same weakness as the semantic path order.

The ability to prove termination for term rewriting systems is of basic importance if one is interested in program verification based on rewriting methods. Termination is the access key to confluence and to automated inductive proofs of equalities. By these one can prove safety conditions of distributed systems. The `LIFT` case study is a clear example for this.