

2

Abstract Reduction Systems

This chapter is concerned with the abstract treatment of reduction, where reduction is synonymous with the traversal of some directed graph, the stepwise execution of some computation, the gradual transformation of some object (e.g. a term), or any similar step by step activity. Mathematically this means we are simply talking about binary relations. An **abstract reduction system** is a pair (A, \rightarrow) , where the **reduction** \rightarrow is a binary relation on the set A , i.e. $\rightarrow \subseteq A \times A$. Instead of $(a, b) \in \rightarrow$ we write $a \rightarrow b$.

The term “reduction” has been chosen because in many applications something decreases with each reduction step, but cannot decrease forever. Yet this need not be the case, as witnessed by the reduction $0 \rightarrow 1 \rightarrow 2 \rightarrow \dots$.

Unless noted otherwise, all our discussions take place in the context of some arbitrary but fixed abstract reduction system (A, \rightarrow) .

2.1 Equivalence and reduction

We can view reduction in two ways: the first is as a directed computation, which, starting from some point a_0 , tries to reach a normal form by following the reduction $a_0 \rightarrow a_1 \rightarrow \dots$. This corresponds to the idea of program evaluation. Or we may consider \rightarrow merely as a description of $\overset{*}{\leftrightarrow}$, where $a \overset{*}{\leftrightarrow} b$ means that there is a path between a and b where the arrows can be traversed in both directions, for example, as in $a_0 \leftarrow a_1 \rightarrow a_2 \leftarrow a_3$. This corresponds to the idea of identities which can be used in both directions. The key question here is to decide if two elements a and b are **equivalent**, i.e. if $a \overset{*}{\leftrightarrow} b$ holds. Settling this question by an undirected search along both \rightarrow and \leftarrow is bound to be expensive. Wouldn't it be nice if we could decide equivalence by reducing both a and b to their normal forms and testing if the normal forms are identical? As explained in the first chapter, this idea is only going to work if reduction terminates and normal forms are unique.

Formally, we talk about *termination* and *confluence* of reduction, and the study of these two notions is one of the central themes of this book.

2.1.1 Basic definitions

In the sequel, we define a great many symbols, not all of which will be put to immediate use. Therefore you may treat these definitions as a table of relevant notions which can be consulted when necessary.

Given two relations $R \subseteq A \times B$ and $S \subseteq B \times C$, their **composition** is defined by

$$R \circ S := \{(x, z) \in A \times C \mid \exists y \in B. (x, y) \in R \wedge (y, z) \in S\}$$

Definition 2.1.1 We are particularly interested in composing a reduction with itself and define the following notions:

$\xrightarrow{0}$	$:= \{(x, x) \mid x \in A\}$	identity
$\xrightarrow{i+1}$	$:= \xrightarrow{i} \circ \rightarrow$	$(i+1)$-fold composition, $i \geq 0$
$\xrightarrow{+}$	$:= \bigcup_{i>0} \xrightarrow{i}$	transitive closure
$\xrightarrow{*}$	$:= \xrightarrow{+} \cup \xrightarrow{0}$	reflexive transitive closure
$\xrightarrow{\equiv}$	$:= \rightarrow \cup \xrightarrow{0}$	reflexive closure
$\xrightarrow{-1}$	$:= \{(y, x) \mid x \rightarrow y\}$	inverse
\leftarrow	$:= \xrightarrow{-1}$	inverse
\leftrightarrow	$:= \rightarrow \cup \leftarrow$	symmetric closure
$\xleftrightarrow{+}$	$:= (\leftrightarrow)^+$	transitive symmetric closure
$\xleftrightarrow{*}$	$:= (\leftrightarrow)^*$	reflexive transitive symmetric closure

Some remarks are in order:

1. Notations like $\xrightarrow{*}$ and \leftarrow only work for arrow-like symbols. In the case of arbitrary relations $R \subseteq A \times A$ we write R^* , R^{-1} etc.
2. Some of the constructions can also be expressed nicely in terms of *paths*:
 $x \xrightarrow{n} y$ if there is a path of length n from x to y ,
 $x \xrightarrow{*} y$ if there is some finite path from x to y ,
 $x \xrightarrow{+} y$ if there is some finite nonempty path from x to y .
3. The word **closure** has a precise meaning: the P closure of R is the least set with property P which contains R . For example, $\xrightarrow{*}$, the reflexive transitive closure of \rightarrow , is the least reflexive and transitive relation which contains \rightarrow . Note that for arbitrary P and R , the P closure of R need not exist, but in the above cases they always do because reflexivity, transitivity and symmetry are closed under arbitrary intersections. In

such cases the P closure of R can be defined directly as the intersection of all sets with property P which contain R .

4. It is easy to show that $\overset{*}{\leftrightarrow}$ is the least equivalence relation containing \rightarrow .

Let us add some terminology to this notation:

1. x is **reducible** iff there is a y such that $x \rightarrow y$.
2. x is in **normal form (irreducible)** iff it is not reducible.
3. y is a **normal form of x** iff $x \overset{*}{\rightarrow} y$ and y is in normal form. If x has a uniquely determined normal form, the latter is denoted by $x \downarrow$.
4. y is a **direct successor** of x iff $x \rightarrow y$.
5. y is a **successor** of x iff $x \overset{\pm}{\rightarrow} y$.
6. x and y are **joinable** iff there is a z such that $x \overset{*}{\rightarrow} z \overset{*}{\leftarrow} y$, in which case we write $x \downarrow y$.

Example 2.1.2

1. Let $A := \mathbb{N} - \{0, 1\}$ and $\rightarrow := \{(m, n) \mid m > n \text{ and } n \text{ divides } m\}$. Then
 - (a) m is in normal form iff m is prime.
 - (b) p is a normal form of m iff p is a prime factor of m .
 - (c) $m \downarrow n$ iff m and n are not relatively prime.
 - (d) $\overset{\pm}{\rightarrow} = \rightarrow$ because $>$ and “divides” are already transitive.
 - (e) $\overset{*}{\leftrightarrow} = A \times A$.
2. Let $A := \{a, b\}^*$ (the set of words over the alphabet $\{a, b\}$) and $\rightarrow := \{(ubar.v.uabv) \mid u, v \in A\}$. Then
 - (a) w is in normal form iff w is sorted, i.e. of the form a^*b^* .
 - (b) Every w has a unique normal form $w \downarrow$, the result of sorting w .
 - (c) $w_1 \downarrow w_2$ iff $w_1 \overset{*}{\leftrightarrow} w_2$ iff w_1 and w_2 contain the same number of as and bs .

Finally we come to some of the central notions of this book.

Definition 2.1.3 A reduction \rightarrow is called

- Church-Rosser**† iff $x \overset{*}{\leftrightarrow} y \Rightarrow x \downarrow y$ (see Fig. 2.1).
confluent iff $y_1 \overset{*}{\leftarrow} x \overset{*}{\rightarrow} y_2 \Rightarrow y_1 \downarrow y_2$ (see Fig. 2.1).
terminating iff there is no infinite descending chain $a_0 \rightarrow a_1 \rightarrow \dots$.
normalizing iff every element has a normal form.
convergent iff it is both confluent and terminating.

Both reductions in Example 2.1.2 terminate, but only the second one is Church-Rosser and confluent.

† Alonzo Church and J. Barkley Rosser proved that the λ -calculus has this property [51].

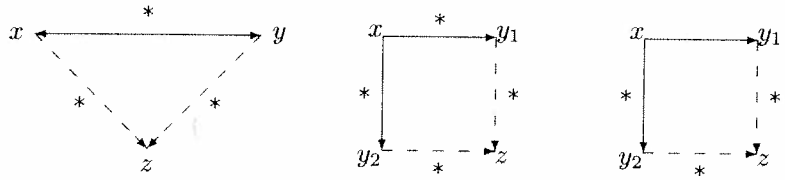


Fig. 2.1. Church-Rosser property, confluence and semi-confluence.

Remarks:

1. The diagrams in Fig. 2.1 have a precise meaning and are used throughout the book in this manner: solid arrows represent universal and dashed arrows existential quantification; the whole diagram is an implication of the form $\forall \bar{x}. P(\bar{x}) \Rightarrow \exists \bar{y}. Q(\bar{x}, \bar{y})$. For example, the confluence diagram becomes $\forall x, y_1, y_2. y_1 \xrightarrow{*} x \xrightarrow{*} y_2 \Rightarrow \exists z. y_1 \xrightarrow{*} z \xrightarrow{*} y_2$.
2. Because $x \downarrow y$ implies $x \xleftrightarrow{*} y$, the Church-Rosser property can also be phrased as an equivalence: $x \xleftrightarrow{*} y \Leftrightarrow x \downarrow y$.
3. Any terminating relation is normalizing, but the converse is not true, as the example in Fig 2.2 shows.

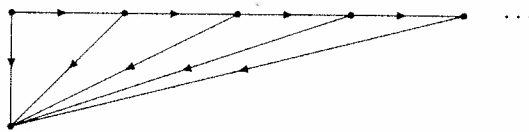


Fig. 2.2. Confluent, normalizing and acyclic but not terminating.

Thus we have come back to our initial motivation: the Church-Rosser property is exactly what we were looking for, namely the ability to test equivalence by the search for a common successor. We will now see how it relates to termination and confluence.

2.1.2 Basic results

It turns out that the Church-Rosser property and confluence coincide. The fact that any Church-Rosser relation is confluent is almost immediate, and the reverse implication has a beautiful diagrammatic proof which is shown in Fig. 2.3. It is based on the observation that any equivalence $x \xleftrightarrow{*} y$ can be

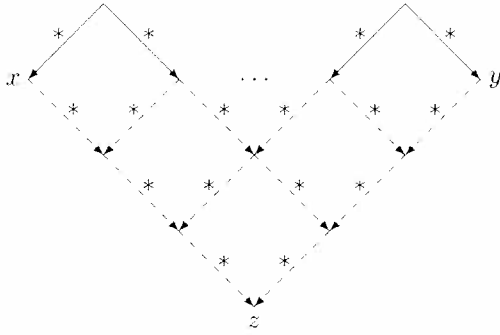


Fig. 2.3. Confluence implies the Church-Rosser property.

written as a series of peaks as in the top of the diagram. Now you can use confluence to complete the diagram from the top to the bottom. The formal proof below yields some additional information by involving an intermediate property:

Definition 2.1.4 A relation \rightarrow is **semi-confluent** (Fig. 2.1) iff

$$y_1 \leftarrow x \xrightarrow{*} y_2 \Rightarrow y_1 \downarrow y_2.$$

Although semi-confluence looks weaker than confluence, it turns out to be equivalent:

Theorem 2.1.5 *The following conditions are equivalent:*

1. \rightarrow has the Church-Rosser property.
2. \rightarrow is confluent.
3. \rightarrow is semi-confluent.

Proof We show that the implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$ hold.

(1 \Rightarrow 2) If \rightarrow has the Church-Rosser property and $y_1 \leftarrow x \xrightarrow{*} y_2$ then $y_1 \leftrightarrow y_2$ and hence, by the Church-Rosser property, $y_1 \downarrow y_2$, i.e. \rightarrow is confluent.

(2 \Rightarrow 3) Obviously any confluent relation is semi-confluent.

(3 \Rightarrow 1) If \rightarrow is semi-confluent and $x \leftrightarrow y$ then we show $x \downarrow y$, i.e. the Church-Rosser property, by induction on the length of the chain $x \leftrightarrow y$. If $x = y$, this is trivial. If $x \xrightarrow{*} y \leftrightarrow y'$ we know $x \downarrow y$ by induction hypothesis, i.e. $x \xrightarrow{*} z \xleftarrow{*} y$ for some suitable z . We show $x \downarrow y'$ by case distinction:

$y \leftarrow y'$: $x \downarrow y'$ follows directly from $x \downarrow y$.

$y \rightarrow y'$: semi-confluence implies $z \downarrow y'$ and hence $x \downarrow y'$.

The reasoning is displayed graphically in Fig. 2.4. □

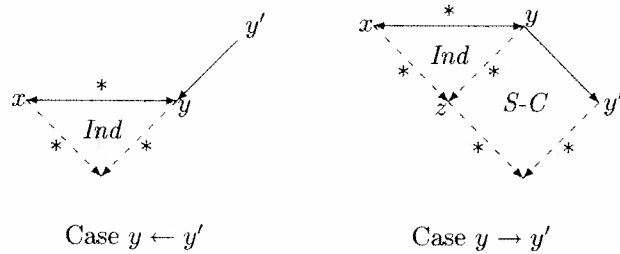


Fig. 2.4. Semi-confluence implies the Church-Rosser property.

This theorem has some easy consequences:

Corollary 2.1.6 *If \rightarrow is confluent and $x \leftrightarrow^* y$ then*

1. $x \xrightarrow{*} y$ if y is in normal form, and
2. $x = y$ if both x and y are in normal form.

Now we know that for confluent relations, two elements are equivalent iff they are joinable. Of course the test for joinability can be difficult (and even undecidable) if the relation does not terminate: given two elements which are not joinable, when should you stop the search for a common successor in case of an infinite reduction starting from one of the two elements, as in the following example?

$$\begin{aligned} a_0 &\rightarrow a_1 \rightarrow a_2 \rightarrow \dots, \\ b_0 &\rightarrow b_1 \rightarrow b_2 \rightarrow \dots. \end{aligned}$$

It turns out that normalization suffices for determining joinability. To see this, let us explore the relationship between termination, normalization, confluence and the uniqueness of normal forms.

Fact 2.1.7 *If \rightarrow is confluent, every element has at most one normal form.*

Since every element has at least one normal form if \rightarrow is normalizing, it follows that for confluent and normalizing relations every element x has exactly one normal form which we write $x\downarrow$:

Lemma 2.1.8 *If \rightarrow is normalizing and confluent, every element has a unique normal form.*

Having established under what conditions the notation $x\downarrow$ is well-defined, we immediately obtain our main theorem:

Theorem 2.1.9 *If \rightarrow is confluent and normalizing then $x \leftrightarrow^* y \Leftrightarrow x\downarrow = y\downarrow$.*

Proof The \Leftarrow -direction is trivial. Conversely, if $x \stackrel{*}{\leftrightarrow} y$ then $x \downarrow \stackrel{*}{\leftrightarrow} y \downarrow$ and hence $x \downarrow = y \downarrow$ by Corollary 2.1.6. \square

Thus we have finally arrived at a very goal-directed equivalence test: simply check if the normal forms of both elements are identical. Provided normal forms are computable and identity is decidable, equivalence also becomes decidable.

Many authors prefer to work with termination instead of normalization and state Theorem 2.1.9 with “convergent” instead of “confluent and normalizing”. Although normalization suffices for finding normal forms, it means that breadth-first rather than depth-first search may be required, for example in Fig. 2.2. For this reason we will also concentrate on termination rather than normalization in the sequel.

Exercises

- 2.1 Which closure operations commute? Find a proof or counterexample:
- Is the reflexive closure of the transitive closure the same as the transitive closure of the reflexive closure, i.e. are $(\overset{\pm}{\rightarrow})^{\pm}$ and $(\overset{\pm}{\rightarrow})^{\pm}$ the same and do they coincide with $\overset{*}{\rightarrow}$?
 - What about the transitive and the symmetric closure? Do $(\leftrightarrow)^+$ and $(\overset{\pm}{\rightarrow}) \cup (\overset{\pm}{\rightarrow})^{-1}$ coincide?
- 2.2 Show that \rightarrow is confluent and normalizing iff every element has a unique normal form.
- 2.3 Find a reduction \rightarrow on \mathbb{N} such that \rightarrow is decidable but it is undecidable if some n is in normal form.

2.2 Well-founded induction

This section introduces the important proof principle of well-founded induction and shows that it is enjoyed by all terminating relations. As a motivation, recall the principle of induction for natural numbers: a property $P(n)$ holds for all natural numbers n if we can show that $P(n)$ holds under the (induction) hypothesis that $P(m)$ holds for all $m < n$. Why is this proof principle sound? Because there is no infinitely descending chain $m_0 > m_1 > \dots$ of natural numbers. The principle of **well-founded induction** is a generalization of induction from $(\mathbb{N}, >)$ to any terminating reduction system (A, \rightarrow) . Formally, it is expressed by the following infe-

rence rule:

$$\frac{\forall x \in A. (\forall y \in A. x \overset{+}{\rightarrow} y \Rightarrow P(y)) \Rightarrow P(x)}{\forall x \in A. P(x)} \quad (\text{WFI})$$

where P is some property of elements of A . The horizontal line is simply another symbol for implication.

In words: to prove $P(x)$ for all x , it suffices to prove $P(x)$ under the assumption that $P(y)$ holds for all successors y of x .

It may come as a bit of a surprise to see an induction schema without explicit base case. The solution to this puzzle is that the premise of WFI subsumes the base case. If \rightarrow is terminating, the “base case” of the induction consists of showing that $P(x)$ holds for all elements without successor, i.e. all normal forms. Hence the assumption $(\forall y \in A. x \overset{+}{\rightarrow} y \Rightarrow P(y))$ is trivially true and the premise of WFI degenerates to $P(x)$, just as expected.

WFI is not correct for arbitrary \rightarrow , but for terminating ones it is:

Theorem 2.2.1 *If \rightarrow terminates then WFI holds.*

Proof by contraposition. Assume that WFI does not hold for \rightarrow , i.e. there is some P such that the premise of WFI holds but the conclusion does not, i.e. $\neg P(a_0)$ for some $a_0 \in A$. But then the premise of WFI implies that there must exist some a_1 such that $a_0 \overset{+}{\rightarrow} a_1$ and $\neg P(a_1)$. By the same argument, there must exist some a_2 such that $a_1 \overset{+}{\rightarrow} a_2$ and $\neg P(a_2)$. Hence there is an infinite chain $a_0 \overset{+}{\rightarrow} a_1 \overset{+}{\rightarrow} a_2 \overset{+}{\rightarrow} \dots$, i.e. \rightarrow does not terminate. \square

As a first application of WFI, we can prove the converse of this theorem:

Theorem 2.2.2 *If WFI holds, then \rightarrow terminates.*

Proof by WFI where $P(x) :=$ “there is no infinite chain starting from x ”. The induction step is simple: if there is no infinite chain starting from any successor of x , then there is no infinite chain starting from x either. Hence the premise of WFI holds and we can conclude that $P(x)$ holds for all x , i.e. \rightarrow terminates. \square

A few words on terminology. Terminating relations are usually called **well-founded** in the mathematical literature. Hence the term “well-founded induction”. In the computer science literature the terms **Noetherian**[†] and **Noetherian induction** are sometimes used instead. Strictly speaking, a reduction system (A, \rightarrow) is well-founded if every nonempty $B \subseteq A$ has a minimal element, i.e. some $b \in B$ such that $b \rightarrow b'$ for no $b' \in B$. With

[†] In honour of the mathematician Emmy Noether.

the help of the Axiom of Choice it can be shown that well-foundedness and termination are equivalent.

We will now use well-founded induction to study some further properties of reductions which are related to termination.

Definition 2.2.3 A relation \rightarrow is called

finitely branching if each element has only finitely many direct successors,

globally finite if each element has only finitely many successors,

acyclic if there is no element a such that $a \xrightarrow{+} a$.

Note that \rightarrow is globally finite iff $\xrightarrow{+}$ is finitely branching.

Lemma 2.2.4 *A finitely branching relation is globally finite if it is terminating.*

Proof Let \rightarrow be finitely branching and terminating. We use well-founded induction to prove that for every element the set of all its successors is finite. Since this is true for all its direct successors (by induction hypothesis), of which there are only finitely many, it is also true for the element itself. \square

It is not true that a finitely branching relation is terminating if it is globally finite. The reason is cycles. However, we have the following weaker implication:

Lemma 2.2.5 *Any acyclic relation is terminating if it is globally finite.*

The combination of the last two lemmas says that a finitely branching and acyclic relation is globally finite iff it is terminating. The special case of an acyclic relation induced by a tree is known as **König's Lemma**:

A finitely branching tree is infinite iff it contains an infinite path.

Exercises

- 2.4 Show that $\xrightarrow{+}$ is terminating iff \rightarrow is.
- 2.5 Show that $\xrightarrow{+}$ is a strict partial order iff \rightarrow is acyclic.
- 2.6 A relation \rightarrow is called **bounded** iff for each element the length of all paths starting from it is bounded: $\forall x. \exists n. \nexists y. x \xrightarrow{n} y$.
- (a) Is every terminating relation bounded?
- (b) Show that a finitely branching relation terminates iff it is bounded.
- 2.7 Prove Lemma 2.2.5.

2.3 Proving termination

The importance of termination hardly needs emphasizing: it is essential not just for programmers but also for theoreticians, as the previous sections, in particular the connection with well-founded induction, have shown. We will now examine a number of constructions for proving termination, a hard (because undecidable) task, as computer scientists well know. These constructions are on the level of relations and are applicable to termination proofs of programs as well as to purely mathematical questions, for example from the realm of group theory.

In connection with termination, it frequently pays to work with transitive relations or even partial orders. One reason is that there is a vast body of mathematical literature on partial orders. Another is that some of our constructions (e.g. the multiset order) are simpler for partial orders than for arbitrary relations. Fortunately, the transition to partial orders is without loss of generality: $\overset{\pm}{\rightarrow}$ terminates iff \rightarrow does, in which case $\overset{\pm}{\rightarrow}$ is a strict order (Exercises 2.4 and 2.5).

The most basic method for proving termination of some (A, \rightarrow) is to embed it into another abstract reduction system $(B, >)$ which is known to terminate. This requires a monotone mapping $\varphi : A \rightarrow B$, where **monotone** means that $x \rightarrow x'$ implies $\varphi(x) > \varphi(x')$. Now \rightarrow terminates because an infinite chain $x_0 \rightarrow x_1 \rightarrow \dots$ would induce an infinite chain $\varphi(x_0) > \varphi(x_1) > \dots$. The mapping φ is often called a **measure function** and the whole construction is known as the **inverse image** construction (because $\rightarrow \subseteq \varphi^{-1}(>) := \{(x, x') \mid \varphi(x) > \varphi(x')\}$). Note that if φ is the identity, this yields that any subset of a terminating relation is terminating.

Example 2.3.1 The most popular choice for termination proofs is an embedding into $(\mathbb{N}, >)$, which is known to terminate. For strings, i.e. $A := X^*$ for some set X , there are two natural choices:

1. Length. φ is defined by $\varphi(w) := |w|$. This proves termination of all length-decreasing reductions like $uabbbv \rightarrow_1 uaav$, where $u, v \in A$ are arbitrary and $a, b \in X$ are fixed.
2. Letters. For each $a \in X$ define $\varphi_a(w) :=$ “the number of occurrences of a in w ”. This can cope with reductions like $uav \rightarrow_2 vbu$ where $u, v \in A$ are arbitrary and $a, b \in X$, $a \neq b$, are fixed.

How about $\rightarrow_1 \cup \rightarrow_2$? We claim it also terminates, in which case Lemma 2.3.3 below tells us that there exists a measure function into \mathbb{N} . Can you find one?

Many program termination proofs follow the same schema by showing that

every computation step (e.g. loop iteration or recursive call) decreases the value of some expression $\varphi(\bar{x})$ in terms of the program variables \bar{x} .

Example 2.3.2 Assume all variables in the following program range over natural numbers:

```
while  $ub > lb + 1$  do
begin  $r := (ub + lb) \text{ div } 2$ ;
    if  $\Phi$  then  $ub := r$  else  $lb := r$ 
end
```

Termination is independent of the test Φ (provided Φ terminates and has no side effect) and can be proved with the measure function $\varphi(ub, lb) := ub - lb$ which decreases with every loop iteration.

The popularity of measure functions into \mathbb{N} is in part explained by the following completeness result:

Lemma 2.3.3 *A finitely branching reduction terminates iff there is a monotone embedding into $(\mathbb{N}, >)$.*

Proof The \Leftarrow -direction follows from the soundness of the measure function approach. For the other direction, let \rightarrow be a terminating and finitely branching reduction. Define $\varphi(x)$ as the number of successors of x which, by Lemma 2.2.4, must be finite. Since \rightarrow is terminating and hence acyclic, $x \rightarrow x'$ implies that x' has strictly fewer successors than x . Alternatively, $\varphi(x)$ can be defined as the length of the longest reduction starting from x . Since \rightarrow terminates, Exercise 2.6 implies that $\varphi(x)$ is well-defined. \square

The restriction to finitely branching relations is necessary, as the following example shows.

Example 2.3.4 Let $A := \mathbb{N} \times \mathbb{N}$ and let \rightarrow be defined by the two rules $(i+1, j) \rightarrow (i, k)$ and $(i, j+1) \rightarrow (i, j)$ for all $i, j, k \geq 0$. This reduction is not finitely branching because the value of k in the first rule is not constrained by the left-hand side. Termination of \rightarrow can be shown by a simple lexicographic construction (see Section 2.4). Yet there is no monotone function φ from $(\mathbb{N} \times \mathbb{N}, \rightarrow)$ into $(\mathbb{N}, >)$. For if there were such a function φ , observe that monotonicity implies $k := \varphi(1, 1) > \varphi(0, k) > \varphi(0, k-1) > \dots > \varphi(0, 0)$. This is a contradiction because there are only k natural numbers below k and yet the chain $\varphi(0, k) > \dots > \varphi(0, 0)$ has length $k+1$.

Even in the context of finitely branching reductions, an embedding into \mathbb{N} can be tricky to find.

Example 2.3.5 Let $A = \mathbb{N} \times \mathbb{N}$ and define the reduction by $(i, j+1) \rightarrow (i, j)$ and $(i+1, j) \rightarrow (i, i)$. This reduction terminates at $(0, 0)$ for every start point. It is also finitely branching. Hence there is a measure function into \mathbb{N} . In this particular case $\varphi(i, j) = i^2 + j$ does the job, but it takes a moment to find this function and prove that it is monotone.

We will now discuss how to get around the above problems with measure functions into \mathbb{N} by building complex orders from simpler ones using fixed constructions which preserve termination.

Exercises

- 2.8 Find a measure function into \mathbb{N} which proves termination of \rightarrow in Example 2.1.2, part 2.
- 2.9 Find a measure function into \mathbb{N} which proves termination of $\rightarrow_1 \cup \rightarrow_2$ in Example 2.3.1.

2.4 Lexicographic orders

Given two strict orders $(A, >_A)$ and $(B, >_B)$, the **lexicographic product** $>_{A \times B}$ on $A \times B$ is defined by

$$(x, y) >_{A \times B} (x', y') \iff (x >_A x') \vee (x = x' \wedge y >_B y').$$

If A and B are obvious from the context we write $>$ instead of $>_{A \times B}$. Sometimes we also write $>_A \times_{lex} >_B$.

The following property is routine to prove:

Lemma 2.4.1 *The lexicographic product of two strict orders is again a strict order.*

More interestingly we have

Theorem 2.4.2 *The lexicographic product of two terminating relations is again terminating.*

Proof by contradiction. Assume there is an infinitely descending chain $(a_0, b_0) > (a_1, b_1) > \dots$. This implies $a_0 \geq_A a_1 \geq_A \dots$. Since $>_A$ terminates this chain cannot contain an infinite number of strict steps $a_i >_A a_{i+1}$. Hence there is a k such that $a_i = a_{i+1}$ for all $i \geq k$. But this implies $b_i >_B b_{i+1}$ for all $i \geq k$, which contradicts the termination of $>_B$. \square

This theorem proves termination of \rightarrow on $\mathbb{N} \times \mathbb{N}$ in Examples 2.3.4 and 2.3.5: $(i, j) \rightarrow (i', j')$ is defined such that (i, j) is lexicographically greater

than (i', j') , i.e. \rightarrow is a subset of the terminating relation $>_{\mathbb{N} \times \mathbb{N}}$. It also proves termination of $\rightarrow_1 \cup \rightarrow_2$ in Example 2.3.1: \rightarrow_1 decreases the length whereas \rightarrow_2 leaves the length invariant but decreases the number of a 's.

Lexicographic products are essential in building up more complex orders from simpler ones. By iteration, we can form lexicographic products over any number of strict orders $(A_i, >_i)$, $i = 1, \dots, n$: $>_{1\dots n}$, where $n > 1$, is the lexicographic product of $>_1$ and $>_{2\dots n}$. Unwinding the recursion and writing $>$ instead of $>_{1\dots n}$ we get

$$(x_1, \dots, x_n) > (y_1, \dots, y_n) \Leftrightarrow \exists k \leq n. (\forall i < k. x_i = y_i) \wedge x_k >_k y_k. \quad (2.1)$$

If all $(A_i, >_i)$ are the same we write $>_{lex}^n$ for the n -fold lexicographic product.

The above results for the binary lexicographic product carry over to n -fold products: $>$ is again a strict order and it terminates if all the $>_i$ terminate. The proofs are by induction on n .

Instead of tuples of fixed length, we can also consider strings of arbitrary but finite length: given a strict order $(A, >)$, the **lexicographic order** $>_{lex}^*$ on A^* is defined as

$$u >_{lex}^* v \Leftrightarrow (|u| > |v|) \vee (|u| = |v| \wedge u >_{lex}^{|u|} v)$$

where $|w|$ is the length of w and $>_{lex}^{|w|}$ is the order on $A^{|w|}$ defined in (2.1) above. More concisely, we can define $>_{lex}^*$ as the lexicographic product of $>_{len}$ and $\bigcup_{i \in \mathbb{N}} >_{lex}^i$, where $u >_{len} v \Leftrightarrow |u| > |v|$. Since A^i and A^j are disjoint if $i \neq j$, the second component of this product is a union of orders over disjoint sets. Since such unions (this is easy to see) and lexicographic products (as shown above) preserve strict orders and termination, we have

Lemma 2.4.3 *If $>$ is a strict order, so is $>_{lex}^*$. If $>$ terminates, so does $>_{lex}^*$.*

Despite its name, $>_{lex}^*$ is not the order used in dictionaries. The latter does not terminate: $a >_{dict} aa >_{dict} aaa >_{dict} \dots$

Yet another interesting variation on lexicographic orders compares strings from left to right as follows: $w_1 >_{Lex} w_2$ if w_2 is a proper prefix of w_1 or if $w_1 = uav$, $w_2 = ubw$ and $a > b$, where $>$ is the underlying strict order. For example, if $a > b$, then $aaaa >_{Lex} aaa >_{Lex} abba$. Unfortunately, $>_{Lex}$ need not terminate either, even if $>$ does (exercise!). Nevertheless, $>_{Lex}$ can be a useful component in more complicated orders.

Lemma 2.4.4 *If $>$ is a strict order, so is $>_{Lex}$.*

The proof, a simple case analysis, is left as an exercise.

A final word of warning about our definition of the lexicographic product. Although we assume the component relations to be strict orders, the definition works just as well for arbitrary relations. In fact, Theorem 2.4.2 depends on termination only. Nevertheless, the lexicographic product of two arbitrary relations may not be what you expect. For example $\geq_{\mathbb{N}} \times_{lex} \geq_{\mathbb{N}}$ relates all (i, j) and (i, k) , simply because $i \geq_{\mathbb{N}} i$. Hence you should not use \times_{lex} directly with reflexive relations. Given two partial orders \geq_A and \geq_B , their lexicographic product should be defined as the reflexive closure of $>_A \times_{lex} >_B$. (Remember that the strict part of a partial order \geq is written $>$.) Of course this can be written more succinctly, if slightly ambiguously, as $\geq_{A \times B}$. Alternatively, we can define the lexicographic product directly for partial orders:

$$(x, y) \geq_{A \times B} (x', y') \quad :\Leftrightarrow \quad (x >_A x') \vee (x = x' \wedge y \geq_B y').$$

It is easy to show that these two definitions of $\geq_{A \times B}$ are equivalent and that $\geq_{A \times B}$ is a partial order if \geq_A and \geq_B are partial orders (exercise!).

Exercises

- 2.10 Prove Theorem 2.4.2 by well-founded induction.
- 2.11 Show that the following process always terminates. There is a box full of black and white balls. Each step consists of removing an arbitrary ball from the box. If it happens to be a black ball, one also adds an arbitrary (but finite) number of white balls to the box.
- 2.12 Show that $v_1 >_{lex}^* v_2$ implies $uv_1w >_{lex}^* uv_2w$.
- 2.13 Show that $>_{A \times B}$ is linear if both $>_A$ and $>_B$ are.
- 2.14 Show that $>_{lex}^*$ is linear if $>$ is.
- 2.15 Why do the following two programs terminate, provided all variables range over positive natural numbers?

```
while m ≠ n do
  if m > n then m := m - n else n := n - m
```

```
while m ≠ n do
  if m > n then m := m - n
  else begin h := m; m := n; n := h end
```

What if the variables range over positive rational numbers?

- 2.16 Show that the evaluation of the following recursively defined function, also known as **Ackermann's function**, terminates for all $m, n \in \mathbb{N}$:

$$\begin{aligned} \text{ack}(0, n) &= n + 1, \\ \text{ack}(m + 1, 0) &= \text{ack}(m, 1), \\ \text{ack}(m + 1, n + 1) &= \text{ack}(m, \text{ack}(m + 1, n)). \end{aligned}$$

- 2.17 Does termination of $>$ imply termination of $>_{Lex}$?
 2.18 Prove Lemma 2.4.4.
 2.19 Show that $>_{Lex}$ is linear if $>$ is.
 2.20 Formalize the order used in dictionaries.
 2.21 The lexicographic product of two quasi-orders \succsim_A and \succsim_B is defined as follows:

$$(x, y) \succsim (x', y') \Leftrightarrow x >_A x' \vee (x \sim_A x' \wedge y \succsim_B y').$$

- (a) Show that \succsim is a quasi-order if both \succsim_A and \succsim_B are.
 (b) Show that $>$, the strict part of \succsim , terminates if $>_A$ and $>_B$ do.

2.5 Multiset orders

Consider the following reduction on \mathbb{N}^* : $u(i+1)v \rightarrow uiiv$ for all $u, v \in \mathbb{N}^*$ and $i \in \mathbb{N}$. It turns out that \rightarrow terminates, and because it is finitely branching, there should also exist a measure function into \mathbb{N} . If you want to spare yourself the torture of finding that function, you should read on.

One of the most powerful ways of building terminating orders is *multisets*. They are usually defined as “sets with repeated elements”, which the purist will find a contradiction in terms, but which conveys their nature quite well. Examples are $\{a, a, b\}$ and $\{a, b, a\}$, which are identical, and $\{a, b, b\}$, which is distinct from them. Of course, we can also be more formal:

Definition 2.5.1 A **multiset** M over a set A is a function $M : A \rightarrow \mathbb{N}$. Intuitively, $M(x)$ is the number of copies of $x \in A$ in M .

A multiset M is **finite** if there are only finitely many x such that $M(x) > 0$. Let $\mathcal{M}(A)$ denote the set of all finite multisets over A .

Although multisets can be infinite, and much of the theory works for infinite multisets, the bit that is crucial for us fails: termination. Therefore all our multisets are assumed to be finite unless stated otherwise.

We use standard set notation like $\{a, a, b\}$ as an abbreviation of the function $\{a \mapsto 2, b \mapsto 1, c \mapsto 0\}$ over the base set $A = \{a, b, c\}$. It will be obvious from the context if we refer to a set or a multiset.

Most set operations are easily generalized to multisets by replacing the underlying Boolean operations by similar ones on \mathbb{N} .

Definition 2.5.2 Some basic operations and relations on $\mathcal{M}(A)$ are:

Element : $x \in M \Leftrightarrow M(x) > 0$.

Inclusion : $M \subseteq N \Leftrightarrow \forall x \in A. M(x) \leq N(x)$.

Union : $(M \cup N)(x) := M(x) + N(x)$.

Difference : $(M - N)(x) := M(x) \dot{-} N(x)$

where $m \dot{-} n$ is $m - n$ if $m \geq n$ and is 0 otherwise.

Some typical examples: $\emptyset \subseteq \{a, a\} \subseteq \{a, a, a\}$, $\{a, b\} \cup \{b, a\} = \{a, a, b, b\}$ and $\{a, b, b, b\} - \{a, a, b, c\} = \{b, b\}$.

Now we come to the central concept of this section, an order on multisets: the smaller multiset is obtained from the larger one by removing a nonempty subset X and adding only elements which are smaller than some element in X .

Definition 2.5.3 Given a strict order $>$ on a set A , we define the corresponding **multiset order** $>_{mul}$ on $\mathcal{M}(A)$ as follows:

$$M >_{mul} N \text{ iff there exist } X, Y \in \mathcal{M}(A) \text{ such that}$$

$$\emptyset \neq X \subseteq M \text{ and}$$

$$N = (M - X) \cup Y \text{ and}$$

$$\forall y \in Y. \exists x \in X. x > y.$$

For example, $\{5, 3, 1, 1\} >_{mul} \{4, 3, 3, 1\}$ is verified by replacing $X = \{5, 1\}$ by $Y = \{4, 3\}$. Note that X and Y are not uniquely determined: $X = \{5, 3, 1, 1\}$ and $Y = \{4, 3, 3, 1\}$ work just as well.

Sometimes it can be useful to realize that $M >_{mul} N$ holds iff you can get from M to N by carrying out the following procedure one or more times: remove an element x and add a finite number of elements, all of which are smaller than x (see Exercise 2.22).

On finite multisets, the multiset order is again a strict order:

Lemma 2.5.4 *If $>$ is a strict order, so is $>_{mul}$.*

Proof Irreflexivity: if $M >_{mul} M$, there are X and Y such that $X \subseteq M$, $M = (M - X) \cup Y$, i.e. $X = Y$, and $\forall y \in Y. \exists x \in X. x > y$, which implies $\forall y \in X. \exists x \in X. x > y$. Since $>$ is a strict order this implies that X is infinite, a contradiction.

Transitivity is more involved. If $M_1 >_{mul} M_2 >_{mul} M_3$ then $M_2 = (M_1 - X_1) \cup Y_1$ and $M_3 = (M_2 - X_2) \cup Y_2$, for multisets X_i and Y_i satisfying the appropriate conditions in the definition of $>_{mul}$. We now claim that

$X := X_1 \cup (X_2 - Y_1)$ and $Y := (Y_1 - X_2) \cup Y_2$ prove $M_1 >_{mul} M_3$. Let us look at the required conditions in turn.

- $X \neq \emptyset$ is implied by $X_1 \neq \emptyset$.
- $X_2 \subseteq M_2 = (M_1 - X_1) \cup Y_1$ implies $X_2 - Y_1 \subseteq M_1 - X_1$ and hence, because $X_1 \subseteq M_1$, $X = X_1 \cup (X_2 - Y_1) \subseteq M_1$.
- We need to show that $M_3 = (M_1 - X) \cup Y =: M'_3$, which follows if we can show $M_3(a) = M'_3(a)$ for an arbitrary $a \in A$. We have $M'_3(a) = (M_1(a) \dot{-} (X_1(a) + (X_2(a) \dot{-} Y_1(a)))) + ((Y_1(a) \dot{-} X_2(a)) + Y_2(a))$. Because $X \subseteq M_1$, the first “ $\dot{-}$ ” in this expression can be replaced by an ordinary minus “ $-$ ”, which (after some arithmetic rearrangement) yields $M'_3(a) = (M_1(a) - X_1(a)) + ((Y_1(a) \dot{-} X_2(a)) - (X_2(a) \dot{-} Y_1(a))) + Y_2(a)$. Obviously, $(m \dot{-} n) - (n \dot{-} m) = m - n$, and thus we obtain $M'_3(a) = (((M_1(a) - X_1(a)) + Y_1(a)) - X_2(a)) + Y_2(a) = (M_2(a) - X_2(a)) + Y_2(a) = M_3(a)$.
- To prove $\forall y \in Y. \exists x \in X. x > y$ let $y \in Y$. If $y \in Y_1$, $M_1 >_{mul} M_2$ implies $x > y$ for some $x \in X_1 \subseteq X$. If $y \in Y_2$, $M_2 >_{mul} M_3$ implies $x > y$ for some $x \in X_2$. If $x \in X_2 - Y_1 \subseteq X$, we are done. Otherwise $x \in Y_1$, in which case $M_1 >_{mul} M_2$ implies $x_1 > x$ for some $x_1 \in X_1 \subseteq X$ and hence $x_1 > y$ by transitivity of $>$ on A . \square

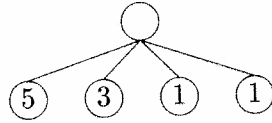
The really important nontrivial property of $>_{mul}$ is

Theorem 2.5.5 *The multiset order $>_{mul}$ is terminating iff $>$ is.*

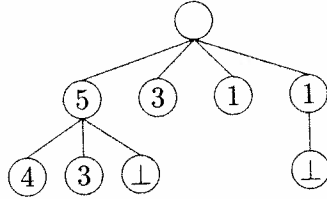
Proof If $>$ does not terminate, there is an infinite chain $a_0 > a_1 > \dots$ which induces an infinite chain $\{a_0\} >_{mul} \{a_1\} >_{mul} \dots$ of multisets. Hence $>_{mul}$ does not terminate either.

If $>$ terminates, we show by contradiction that $>_{mul}$ terminates. Assume there is an infinite chain $M_0 >_{mul} M_1 >_{mul} \dots$. We can then build a finitely branching but infinite tree where the nodes are labelled with elements of A such that along each path the labels decrease w.r.t. $>$. Using König's Lemma, it follows that this tree must have an infinite branch, which yields an infinitely descending sequence in A , the desired contradiction. It remains to be seen how to construct this tree.

Let \perp be an arbitrary element not in A , let $A_\perp := A \cup \{\perp\}$, and extend $>$ by defining $a > \perp$ for all $a \in A$. Obviously $(A_\perp, >)$ is still terminating. Now we grow the following tree whose nodes are labelled with elements of A_\perp . At stage i of the construction the non- \perp leaf nodes form the multiset M_i . The initial tree has a root with an arbitrary label and a successor node for each element of M_0 , e.g. $M_0 = \{5, 3, 1, 1\}$:



Since $M_0 >_{mul} M_1$, there are finite multisets X and Y with the properties stated in the definition of $>_{mul}$. For every $y \in Y$ add a new node labelled y and make it the child of some leaf node labelled x where $x > y$. By definition of $>_{mul}$ such an x must exist in $X \subseteq M_0$ and hence x is among the current leaf nodes. In addition we add a son labelled \perp to each $x \in X$. This ensures that even if Y is empty, the tree has grown. Example: $M_1 = \{4, 3, 3, 1\}$, $X = \{5, 1\}$ and $Y = \{4, 3\}$:



This process can be continued for M_2, M_3, \dots . Thus we are constructing a finitely branching (the M_i are finite) but infinite (for each M_i at least one node is added) tree. Ignoring the root node, the labels on each path are strictly decreasing by construction.

Note that the proof does not require $>$ to be a strict order but works for any relation.

It is now easy to see that the reduction $u(i+1)v \rightarrow uiiv$ considered at the beginning of this section terminates: the mapping $\varphi : \mathbb{N}^* \rightarrow \mathcal{M}(\mathbb{N})$ defined by $\varphi(i_1 \dots i_n) := \{i_1, \dots, i_n\}$ is obviously monotone ($\varphi(u(i+1)v) = \varphi(u) \cup \{i+1\} \cup \varphi(v) >_{mul} \varphi(u) \cup \{i, i\} \cup \varphi(v) = \varphi(uiiv)$) and $>_{mul}$ on $\mathcal{M}(\mathbb{N})$ terminates because $>$ on \mathbb{N} does.

The above definition of $>_{mul}$ is quite intuitive but also a little cumbersome because of its many quantifiers and conditions. Therefore the following alternative characterization is useful:

Lemma 2.5.6 *If $>$ is a strict order and $M, N \in \mathcal{M}(A)$, then*

$$M >_{mul} N \Leftrightarrow M \neq N \wedge \forall n \in N - M. \exists m \in M - N. m > n.$$

Proof For the \Rightarrow -direction, assume $M >_{mul} N$, in which case there are X and Y as in the definition of $>_{mul}$. $M \neq N$ follows from irreflexivity of $>_{mul}$. For the second conjunct, let $y_1 \in N - M = ((M - X) \cup Y) - M = ((M \cup Y) - X) - M = ((M \cup Y) - M) - X = Y - X$. Hence there is $y_2 \in X$ such that $y_2 > y_1$. Either $y_2 \in X - Y = (M - (M - X)) - Y$

$M - ((M - X) \cup Y) = M - N$, in which case we are done, or $y_2 \in X \cap Y$ (where $(X \cap Y)(x) := \min(X(x), Y(x))$), in which case there is a $y_3 \in X$ such that $y_3 > y_2$. Because our multisets are finite and $>$ is a strict order, there is no infinite ascending chain $y_1 < y_2 < y_3 < \dots$ in $X \cap Y$, i.e. this process must always terminate with some $y_n \in X - Y = M - N$. Transitivity yields $y_n > y_1$.

The \Leftarrow -direction is left as an exercise. \square

It is worth noting that if $>$ is linear, then $M >_{mul} N$ can be computed quite efficiently: sort M and N into descending order (w.r.t. $>$) and compare the resulting lists lexicographically w.r.t. $>_{Lex}$. Let \vec{M} be the sorted version of M . It is easy to see that $\vec{M} >_{Lex} \vec{N}$ implies $M >_{mul} N$: either \vec{N} is a proper prefix of \vec{M} , in which case $M \supset N$ and hence $M >_{mul} N$; or $\vec{M} = umv$, $\vec{N} = unw$ such that $m > n$, in which case m is larger than all elements in w , which again implies $M >_{mul} N$. Conversely, if $\vec{M} \not>_{Lex} \vec{N}$ then either $M = N$ or $\vec{N} >_{Lex} \vec{M}$ (Exercise 2.19) and thus $N >_{mul} M$; since $>_{mul}$ is strict, this implies $M \not>_{mul} N$ in both cases. Thus we conclude that

$$M >_{mul} N \Leftrightarrow \vec{M} >_{Lex} \vec{N}. \quad (2.2)$$

Let us briefly look at the multiset extension of partial orders. As in the lexicographic case, we have to be a bit careful. If we simply replace $>$ by \geq we end up with $\{1\} \geq_{mul} \{1, 1\}$, which is not desirable. Instead, \geq_{mul} , the multiset extension of a partial order \geq , is defined as follows:

$$M \geq_{mul} N \Leftrightarrow M >_{mul} N \vee M = N.$$

Given a quasi-order (A, \succsim) , we define its multiset extension via the induced partial order \geq on A/\sim :

$$M \succsim N \Leftrightarrow M/\sim \geq_{mul} N/\sim$$

where $\{a_1, \dots, a_k\}/\sim := \{[a_1]_{\sim}, \dots, [a_k]_{\sim}\}$.

Exercises

2.22 Given a strict order $(A, >)$, define the following single-step relation on $\mathcal{M}(A)$:

$$M >_{mul}^1 N \Leftrightarrow \exists x \in M, Y \in \mathcal{M}(A). N = (M - \{x\}) \cup Y \wedge \forall y \in Y. x > y.$$

Show that $>_{mul}$ is the same as the transitive closure of $>_{mul}^1$. (*Hint*: show that each relation is contained in the other using appropriate inductions.) Conclude that $>_{mul}$ is transitive.

- 2.23 Show that X and Y in the definition of $>_{mul}$ can always be chosen such that they are disjoint.
- 2.24 Give a counterexample to Lemma 2.5.4 for infinite multisets. Show that Lemma 2.5.4 also holds for infinite multisets provided there is no infinitely ascending chain $x_0 < x_1 < \dots$.
- 2.25 Prove the \Leftarrow -direction of Lemma 2.5.6.
- 2.26 Show that if \geq is a partial order, so is \geq_{mul} , and that \succsim_{mul} is a quasi-order if \succsim is one.

2.6 Orders in ML

How should we implement strict/partial orders in general? The obvious implementation as a function $ord: \tau * \tau \rightarrow bool$ has its problems:

- If $ord(x, y)$ implements $x > y$, we cannot recover $x \geq y$ by writing $ord(x, y) \text{ or else } x = y$ because in general we cannot assume that the mathematical equality $=$ on the base set A coincides with the programming language equality $=$ on the type τ used to implement A . For example, if sets are implemented by lists, we do not have $[1, 2] = [2, 1]$ although they are equal as sets.
- If $ord(x, y)$ implements $x \geq y$, we can compute $x > y$ as $ord(x, y) \text{ and also } \text{not}(ord(y, x))$. This is mathematically correct but inefficient because of the two calls to ord . The performance penalty is exponential in the depth of the nesting of orders.
- Implementing both $>$ and \geq is likely to duplicate much of the code.

To overcome these problems we introduce

```
datatype order = GR | EQ | NGE;
```

which represents the three outcomes $>$, $=$ and $\not\geq$. We say that a function ord computes a strict/partial order $>/\geq$ if

$$ord(x, y) = \begin{cases} GR & \text{if } x > y, \\ EQ & \text{if } x = y, \\ NGE & \text{if } x \not\geq y. \end{cases}$$

Note that by $x = y$ we mean equality on the abstract, not the implementation level. The latter is $x = y$, which is too weak, as the set/list example demonstrates: on the implementation level, a partial order may turn into a quasi-order. The purpose of EQ instead of $=$ is to hide that fact. On the other hand, we may even start with a quasi-order \succsim , in which case GR , EQ and NGE represent $>$, \sim and $\not\geq$.