Chain Derivations

The Birkhoff system has five rules; in fact we can collapse them all into just one proof rule with the format called chain derivations.

This has the disadvantage that the one proof rule is a bit complicated to state, but also the following advantages:

- Proofs are briefer to state;
- Proofs are more uniform;
- The process of changing one term into another gives something of the flavour of term rewriting.

Chain Derivations ctd.

An elementary derivation over a theory $T$ is an equation

$$t(\cdots r' \cdots) \approx t(\cdots s' \cdots)$$

Where $r' \approx s'$ is a substitution instance of $r \approx s$, and either

- $r \approx s$ is literally the same term - we write this $r \equiv s$;
- $s \approx r$ is an axiom of $T$.

A chain derivation of $t \approx t'$ is simply a sequence of elementary derivations $t \approx s_1, s_1 \approx s_2, \ldots, s_n \approx t'$.

Note where the rules [REFL], [SYMM], [TRANS], [SUBST] and [REP] are lurking in this definition!

With a bit of practice it is easy to extract chain derivations from Birkhoff derivations and vice versa, e.g.

$$\begin{align*}
(x \cdot y) \cdot x^{-1} &\approx x \cdot (y \cdot x^{-1}) & \text{[AX]} \\
(y \cdot x) \cdot x^{-1} &\approx y \cdot (x \cdot x^{-1}) & \text{[SUBST]} \\
x \cdot x^{-1} &\equiv e & \text{[AX]} \\
y \cdot x^{-1} \equiv y \cdot e &\approx y \cdot e & \text{[REP]} \\
(x \cdot x^{-1}) \equiv y \cdot e &\approx y \cdot e & \text{[TRANS]} \\
(y \cdot x) \cdot x^{-1} &\equiv y \cdot e & \text{[REP]} \\
y \cdot x^{-1} &\equiv y & \text{[SUBST]}
\end{align*}$$

is equivalent to the chain derivation

$$y \cdot x \cdot x^{-1} \approx y \cdot (x \cdot x^{-1}) \approx y \cdot e \approx y.$$
**Chain Derivations ctd.**

A derivation of \( e \cdot x \approx x \), given the axioms \( x \cdot x^{-1} \approx e \) and \((y \cdot x) \cdot x^{-1} \approx y\):

\[
\begin{align*}
e \cdot x & \approx ((e \cdot x) \cdot x^{-1}) \cdot (x^{-1})^{-1} \\
& \approx e \cdot (x^{-1})^{-1} \\
& \approx (x \cdot x^{-1}) \cdot (x^{-1})^{-1} \\
& \approx x
\end{align*}
\]

Exercise: find the chain derivation of \( e \cdot x \approx x \) using the three group axioms.

**Reflecting on Chain Derivations**

Chain Derivations have a somewhat algorithmic flavour, as terms are 'rewritten' till they become the term we're looking for.

However, they are not actually an algorithm - it's not clear at all that in rewriting \( e \cdot x \) to \( x \) the first step is to go to \(((e \cdot x) \cdot x^{-1}) \cdot (x^{-1})^{-1}\)!

We need stronger guidance on what rewritings we should do when.

**Key idea**: Do these rewrites only when they are simplifications.

An example: \( x \cdot e \rightarrow x \) is a simplification; \( x \rightarrow x \cdot e \) is not.

**Proof by Simplification**

If our simplification rules are set up 'properly', each term will have a unique simplest form. This is called a normal form.

The test for equality on two terms is to reduce both to their normal forms ('simplify as far as possible'), then look to see to see if these normal forms are literally equal!

For example, \( x \cdot e \approx e \cdot x \), because both should reduce to the normal form \( x \).

There are lots of questions raised by this sketch. When does a rewriting step count as a simplification? How do we ensure that each term has a unique normal form? And what are these 'rewriting steps' all about, anyway?

This last question is the one we will address first.

**Term Rewrite System**

A term rewrite rule, or directed equation, over a signature \( \Sigma \) is a pair of \( \Sigma \)-terms with an arrow symbol between them:

\( t \rightarrow t' \)

A term rewrite system (TRS) is a set of term rewrite rules.

**Example.** A TRS for monoids:

\[
\begin{align*}
(x \cdot y) \cdot z & \rightarrow x \cdot (y \cdot z) \\
e \cdot x & \rightarrow x \\
x \cdot e & \rightarrow x
\end{align*}
\]

If we swapped the left and right hand sides above, that would also be a TRS. The union of these two TRSs would give yet another one, etc...
### Elementary rewrites and rewrite relations

An elementary rewrite obtained from the rule \( t \rightarrow t' \) is a rewrite
\[
s(\cdots t\sigma \cdots) \rightarrow s(\cdots t'\sigma \cdots)
\]
where \( \sigma \) is a substitution.

**Example.** \((x \cdot y) \cdot e \rightarrow (x \cdot y) \cdot x)\) is an elementary rewrite obtained from
the rule \( x \cdot e \rightarrow x \) with the substitution \( \{ x \leftarrow x \cdot y \} \).

Given a TRS \( R \), we write \( t \rightarrow_R t' \) if \( t \rightarrow t' \) is an elementary rewrite obtained
from a rule in \( R \).

We write \( \rightarrow^*_R \) for the reflexive-transitive closure of \( \rightarrow_R \). That is, \( t \rightarrow^*_R t' \) if either \( t \equiv t' \) or \( t \rightarrow_R t_1 \rightarrow_R t_2 \rightarrow_R \cdots \rightarrow_R t_n \rightarrow_R t' \).

We write \( \leftrightarrow^*_R \) for the reflexive-transitive-symmetric closure of \( \rightarrow_R \).

### From term rewrite systems to equational theories

Given a TRS \( R \), we write \( E(R) \) for the equational logic theory obtained by
replacing all the arrows \( \rightarrow \) with equalities \( \approx \).

E.g. our ‘TRS for monoids’, \( \{ (x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z), e \cdot x \rightarrow x, x \cdot e \rightarrow x \} \),
gives rise to our familiar equational theory for monoids.

It then holds that
\[
E(R) \vdash t \approx t' \iff t \leftrightarrow^*_R t'
\]

**Proof sketch** – the definition of \( \leftrightarrow^*_R \) is essentially identical to the definition of
chain derivations for \( E(R) \).

It is by no means as simple to go from equational theories to TRSs!

### Termination

A sequence of elementary rewrites obtained from \( R \)
\[
t_1 \rightarrow_R t_2 \rightarrow_R \cdots t_{n-1} \rightarrow_R t_n
\]
is terminating, or Noetherian (after Emmy Noether (1882-1935)),
if no \( t_n \rightarrow_R t_{n+1} \) is possible.

A term rewrite system \( R \) is terminating if all sequences of elementary
rewrites with \( R \) are finite.

**Example.** The TRS \( \{ x \cdot e \rightarrow x \} \) is terminating - it halts after it has
removed all \( e \)'s in the correct position. The TRS \( \{ x \rightarrow x \cdot e \} \) is not
terminating - we can add \( e \)'s to the right all day long.

### Termination ctd.

How do we prove that a TRS is terminating?

In general this is undecidable (we can encode Turing Machines as TRSs).

However one method that often works is to show that every rewrite rule in our
TRS is a simplification - the right hand side is strictly simpler than the left
hand side.

To do this we need to be precise about what ‘simpler’ means, of course.
Reduction orderings

A reduction ordering for a signature Σ is a binary relation > on the set of Σ-terms such that

- > is irreflexive: \( \neg (t > t) \);
- > is transitive: \( t > u \) and \( u > v \) implies \( t > v \);
- > is compatible with Σ’s function symbols: if \( t_i > t'_i \) then \( f t_1 \cdots t_n > f t'_1 \cdots t'_n \);
- > is fully invariant: if \( t > t' \) then \( t \sigma > t' \sigma \) for any substitution \( \sigma \);
- > is well-founded: chains \( t_1 > t_2 > \cdots \) are always finite.

You could think of such a reduction ordering as a giant TRS. More intuitively, read \( t > u \) as ‘\( u \) is simpler than \( t \)’.

Reduction orderings ctd.

It’s pretty clear why we need irreflexivity, transitivity and well-foundedness.

We need compatibility with all function symbols because elementary rewrites work on subterms of the term we’re rewriting, and full invariance because we use substitution instances of the rules. A non-example makes this clear:

Let \( t > t' \) if \( t \) is longer (has more variables and function symbols) than \( t' \).

> is irreflexive, transitive, well-founded, and compatible \( \checkmark \).

But say we had the rewrite rule \( (x \cdot y) \cdot z \rightarrow x \cdot x \cdot z \), which is contained in >.

Then we can do the elementary rewrite

\[
((x_1 \cdot x_2) \cdot x_3) \cdot y \rightarrow ((x_1 \cdot x_2) \cdot x_3) \cdot ((x_1 \cdot x_2) \cdot x_3)
\]

which is not contained in >. Therefore > is not fully invariant.

Reduction orderings ctd.

Considering a reduction ordering as a TRS, it is clearly terminating - that’s what the ‘well-founded’ condition tells us.

More interesting is what following a reduction ordering (in the sense made precise below) tells us about an ordinary TRS:

**Theorem.** Given a reduction ordering > and a TRS \( \mathcal{R} \), that is a subset of > (i.e. for all \( (t \rightarrow t') \in \mathcal{R} \) we have \( t > t' \)), \( \mathcal{R} \) is terminating.

This holds because each elementary rewrite produces a simpler term, and this can only happen finitely often.

Reduction orderings ctd.

A reduction ordering that does work is to let \( t > t' \) if \( t \) is longer than \( t' \), and \( t' \) contains no more occurrences of any variable than \( t \).

With our monoid TRS, \( e \cdot x \rightarrow x \) and \( x \cdot e \rightarrow x \) are in this ordering. But which of \( (x \cdot y) \cdot z \) and \( x \cdot (y \cdot z) \) is simpler?

The choice is arbitrary, but we can extend our > by saying that \( t > t' \) if the above situation holds, or if they have the same size and numbers of each variable, but the first subterm where they differ in length is longer in \( t \) than \( t' \). Hence \( (x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z) \) is a ‘simplification’.

Reduction orderings can be generated via a consistent ‘recipe’ with Knuth-Bendix Orderings (after Donald Knuth (1938-) and Peter Bendix), but we won’t go into the details of this.
Normal forms

The TRS $R = \{ x \cdot y \to x, x \cdot y \to y \}$ is terminating (it halts once all ‘dots’ are gone). However it will not do to help us decide an equational theory.

Why? Because $e \cdot x \to_R e$ is terminating, and so is $e \cdot x \to_R x$. So terms do not reduce to a unique normal form.

On the other hand, $R' = \{ e \cdot x \to x, e \cdot x \to x \}$ is fine even though it is non-deterministic:

A normal form TRS is a terminating TRS $R$ where every terminating chain of rewrites starting at some term $t$ has the same last term. We call this the normal form of $t$ and write it $n_R(t)$.

Normal forms ctd.

Recall the fact that $E(R) \vdash t \approx t'$ iff $t \leftrightarrow^*_R t'$

This isn’t that useful as is, because $\leftrightarrow^*_R$ cannot always easily be calculated.

If $R$ is a normal form TRS, then we get a rather better situation:

$E(R) \vdash t \approx t'$ iff $n_R(t) \equiv n_R(t')$

Proof Sketch. If $n_R(t) \equiv n_R(t')$ then we simply need to stitch together the chain derivations of $t \approx n_R(t)$ and $n_R(t) \approx t'$. The converse follows because any two terms that are related by $\leftrightarrow^*_R$ have the same normal form.

Normal forms ctd.

Our monoid TRS,

\[
\begin{align*}
(x \cdot y) \cdot z & \to x \cdot (y \cdot z) \\
e \cdot x & \to x \\
x \cdot e & \to x
\end{align*}
\]

is a normal form TRS - associativity pushes the brackets as far to the right as possible, and either every $e$ is eliminated, or the term reduces to $e$.

(We would also have a normal form TRS if we reversed the associativity rule; this does not apply to the other rules!)

This sort of hand waving is not good enough for larger TRSs; it’s hardly good enough for this one! We need techniques for proving TRSs have the normal form property. We do this via the concept of confluence.