**Equational Logic and Term Rewriting: Lecture V**

Logic Summer School

Ranald Clouston
Australian National University
5–9 December 2011

---

**Confluence and Local Confluence**

Start with any term \( t \), and say it rewrites to two different terms \( u, u' \). \( R \) is **confluent** if there always exists a term \( v \) such that \( u, u' \) both rewrite to \( v \).

**Theorem.** If \( R \) is terminating, it is **normal form** iff confluent (easy).

Now say we have **elementary** rewrites from \( t \) to \( u, u' \). \( R \) is **locally confluent** if there always exists a term \( v \) such that \( u, u' \) both rewrite to \( v \).

---

**Critical Pairs Lemma**

From the last two slides, all we have to do to confirm that our terminating TRS is normal form is to check local confluence.

This still gives us a massive search space (every possible way every term could ‘split’ into two different ones). It turns out we can cut our search space down further by only looking at certain such splits.

**Critical Pairs Lemma.** If \( R \) is terminating then it is normal form (and confluent and locally confluent) iff it is **confluent on critical pairs**.

I’m afraid the proof of this is a bit involved to get through this week, but once you see what critical pairs are you should find the lemma quite plausible.

To define critical pairs we’ll need first to master **unification**.
**Unifiers**

Let’s put equational logic and term rewriting to the side for the moment, and think only about $\Sigma$-terms and their substitutions.

Our question is this: given terms $t$ and $t'$, is there a substitution $\sigma$ such that $t\sigma \equiv t'\sigma$?

If there is, we call $\sigma$ the **unifier** of $t$ and $t'$.

Sometimes there will be lots of unifiers: $x$ and $y$ are unified by the substitution $\{x \leftarrow t, y \leftarrow t\}$ for any $t$ at all.

Other pairs of terms cannot be unified at all, e.g. $x$ and $x \cdot x$.

---

**Most General Unifiers**

The most general unifier (mgu) of two terms is a unifier that is more general than all all other unifiers.

Mgu’s are not unique. When we are unifying $x \cdot y$ and $y \cdot x$, the unifiers

$$\{x \leftarrow y, y \leftarrow y\}, \quad \{x \leftarrow x, y \leftarrow x\}, \quad \{x \leftarrow z, y \leftarrow z\}$$

are all more general than each other, and anything else also. They are all mgu’s!

However mgu’s are unique up to variable renamings, and if any unifiers exist then there exist most general ones.

Further, we have an easy algorithm that finds most general unifiers, or tells you that no unifiers exist.

---

**The Critical Subterm Condition**

Take two terms written in Polish (prefix) notation.

Scan through through the terms until you find a disagreement. The subterms that begin with that disagreement are called **critical subterms**.

E.g. $\cdot \cdot \cdot e e x e$ (i.e., $(e \cdot x) \cdot e$) and $\cdot \cdot e \cdot y e e$ (i.e., $(e \cdot (y \cdot e)) \cdot y$) have the critical subterms $x$ and $\cdot y e$.

The **Critical Subterm Condition (CSC)** holds if one of the terms is a variable (say, $x$) and the other is a term that does not contain $x$.

The example above obeys CSC.

Non-examples. $e$ and $\cdot e y$; $x$ and $\cdot x y$. Such examples are not unifiable.
Unification Algorithm

let σ be the identity substitution.

while s ≠ t
    find the critical subterms s', t'
    if CSC fails return NOT UNIFIABLE
    else, where \{s', t'\} = \{x, u\},
        set s := s{t ← u}, t := t{x ← u}, and σ := \{x ← u\}σ
    return σ.

Example. s = e e e e e and t = e e e e e first produces
σ = \{x ← e\}, s = e e e, t unchanged.
and then
σ = \{y ← e, x ← e e\}, s = t = e e e e e.

Critical pairs

What happens when one rule acts on the left hand side of another?

- Let s1 → t1, s2 → t2 be two, or perhaps the same, rewrite rule. Rename the variables in one of them so that they have no variables in common.

  e.g. Take (x · y) · z → x · (y · z) and x · e → x. Rename the latter x' · e → x'.

- Choose a non-variable subterm s1' of s1 so that mgu σ unifies s1', s2.

  e.g. x · y is a non-variable subterm of (x · y) · z; it unifies with x' · e via mgu \{x' ← x, y ← e\}.

- Apply each rewrite rule in turn to s1.σ. This gives our critical pair.

  e.g. (x · e) · z → x · (e · z), and (x · e) · z → x · z, so (x · (e · z), x · z) is a critical pair.

Syntactic vs. Semantic Unification

You might wonder why we want to find a substitution σ making t and t' literally equal, rather than equal in some theory:

\[ T \vdash t \approx t' \sigma, \text{ or } T \vdash t \approx t' \]

This problem is called \( T \)-unification, or semantic unification, and is undecidable (but, or therefore, interesting!).

However we only need our simpler problem, sometimes called syntactic unification, to define critical pairs and hence get the normal form results we want.

Critical pairs ctd.

Recall that a terminating TRS is normal form iff it is confluent on critical pairs.

Our monoid TRS has produced the critical pair \( (x · (e · z), x · z) \); it is indeed confluent on this pair via the rule \( e · x → x \).

What other critical pairs does this TRS produce?

Consider the associativity rule and itself: \( x' · y' \) is a subterm of the renamed \( (x' · y') · z' \), and it unifies with \( (x · y) · z \) via \{x' ← x · y, y' ← z\}. Applying the rules to \( ((x · y) · z) · z' \) gives us the critical pair

\[ ((x · y) · (z · z'), (x · (y · z)) · z') \]

Keep using associativity to push the brackets to the right and you'll see we're confluent on this pair. There are only three more critical pairs to consider (can you find it?) and we're confluent on that too. The monoid TRS is normal form!
**Critical pairs ctd.**

One thing that might be perplexing you is why we need to find critical pairs produced by two copies of the same rule.

Say we have unary $f$ and a constant $c$, and one rewrite rule $f f x \rightarrow c$.

Consider the subterm $f x'$ of the renamed $f f x'$. This subterm unifies with $f f x$ via the mgu $\{x' \leftarrow f x\}$.

(This is why we have to rename our variables; $f x$ and $f f x$ do not unify).

We can now apply the rewrite rule to $f f f x$ in two different ways:

\[
\begin{align*}
\text{LHS:} & \quad f f f x \\
\text{RHS:} & \quad R_c \\
& \quad f f f x \\
& \quad R_f f c
\end{align*}
\]

And our rule is not confluent on the critical pair $(c, f c)$!

---

**Knuth-Bendix Completion**

Assume we have a reduction order; if we ever face a pair of terms that are incomparable we fail (more sophisticated algorithms can somewhat cope with this problem, as we’ll briefly see).

Direct all our axioms according to the reduction order to get our initial rules.

If we ever find that every critical pair can be joined we succeed (we managed this with monoids with no further work at all).

Otherwise, take a critical pair we are not yet confluent on. Reduce each side of the pair as much as possible, then **add this pair as a new rewrite rule**, using our reduction order to decide left and right. We’re now confluent on this critical pair - although we now have another rule to check!

Some of our new rules might generalise old ones, so we can ‘housekeep’ by deleting old rules, although this is not essential.

---

**What do we do if we’re not normal form?**

Given the three axiom presentation of groups, we can direct our equations in a way that is clearly terminating:

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

We’ve already seen that associativity and right identity produce the critical pair $(x \cdot (e \cdot z), x \cdot z)$. There are no rewrite rules we can apply to either of this pair - our rules are not confluent on critical pairs, and so are not normal form.

Do we just give up? Not necessarily. We have a technique that isn’t guaranteed success, but is often useful - **Knuth-Bendix completion**.

---

**Knuth-Bendix Completion ctd.**

Consider the TRS

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

We’ve already seen we’re not confluent on $(x \cdot (e \cdot z), x \cdot z)$, so add the rule

\[
x \cdot (e \cdot y) \rightarrow x \cdot y
\]

The subterm $e \cdot y$ of the left hand side of the above unifies with $x' \cdot e$ via $\{x' \leftarrow e, y \leftarrow e\}$. Then $x \cdot (e \cdot e)$ produces the critical pair $(x \cdot e, x \cdot e)$ - this is trivially joinable!

There are a few more critical pairs to check (can you find them?) and our rules are confluent on all of them. Our two axiom theory has produced a three rule normal form TRS!
Knuth-Bendix Completion ctd.

From an terminating TRS $\mathcal{R}$, based on an equational theory $\mathcal{T}$, Knuth-Bendix completion constructs a (generally bigger) TRS $\mathcal{R}'$ that is normal form.

$\mathcal{T}$ and $\mathcal{E}(\mathcal{R}')$ are different theories, but they have the same logical / semantic consequences, because of how they were constructed. Therefore

$$\mathcal{T} \vdash t \approx t' \iff n_{\mathcal{R}'}(t) = n_{\mathcal{R}'}(t')$$

$\mathcal{R}'$ is not deterministic step-by-step but it does terminate with a deterministic answer, so we’ve constructed a decision procedure for $\mathcal{T}$ that we could feed to a computer.

What’s the catch? Knuth-Bendix completion might fail (hitting an undirectable equation) or might not terminate! In practice, making a clever / lucky choice of reduction ordering is very important.

What about groups?

What happens when we apply Knuth-Bendix completion to group theory?

$$(x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z)$$

$$x \cdot e \rightarrow x$$

$$x \cdot x^{-1} \rightarrow e$$

That depends on what order you look at the critical pairs, and even more on your choice of reduction ordering (the most obvious one is not necessarily the best!). But one correct answer, with ten rules, is...

What about groups? ctd

All this work relies on equations being directable according to a reduction order. Without that we don’t have termination, let alone normal forms.

Some equations can never be directed; the most pervasive is commutativity.

Suppose we had this rewrite rule:

$$xy \rightarrow yx$$

We could use this to go $t \cdot u \rightarrow_{\mathcal{R}} u \cdot t \rightarrow_{\mathcal{R}} t \cdot u \ldots$ all day long!

One solution is congruence class rewriting, where we have both rewrite rules and undirected equations, on which we can do normal equational proofs.

We can then (sometimes) design an algorithm to solve the equational side of our system. Commutativity, for example, is a solved problem.

Advanced topic I - Undirectable equations

All this work relies on equations being directable according to a reduction order. Without that we don’t have termination, let alone normal forms.

Some equations can never be directed; the most pervasive is commutativity.

Suppose we had this rewrite rule:

$$xy \rightarrow yx$$

We could use this to go $t \cdot u \rightarrow_{\mathcal{R}} u \cdot t \rightarrow_{\mathcal{R}} t \cdot u \ldots$ all day long!

One solution is congruence class rewriting, where we have both rewrite rules and undirected equations, on which we can do normal equational proofs.

We can then (sometimes) design an algorithm to solve the equational side of our system. Commutativity, for example, is a solved problem.
Advanced topic II - Partial algebra

Terms can generally be constructed by giving function symbols any terms at all (as long as their number matches the arity).

But this isn’t always appropriate - if we are trying to axiomatise certain aspects of arithmetic, we might have a 0 constant and symbols for division (as with fields) or exponentiation. These should sometimes be undefined!

This leads us to partial algebra which extends equational logic with a convergence operator \( \downarrow \), so e.g. we cannot prove \((x/0)\downarrow\).

Adding new operators to equational logic adds a lot of work, of course, as e.g. a notion of term rewriting must be reconstructed.

Advanced topic III - Dealing with names

We’ve shown that propositional logic fits nicely into the equational framework. First order logic sits less well. The start of the problem is that we have two types of variables:

\[
\neg \forall a. x \approx \exists a. \neg x
\]

Here \( x \) is like the variables we’ve handled thus far, as it can be substituted for any term. It is sometimes called a metavariable.

However you can’t substitute \( a \) for any term - \( \forall(\exists \neg z). x \) is ill-formed. \( a \) is called an object-level variable, or name.

Things get worse though, because our equations often have side conditions about names appended to them:

\[
(\forall a. x) \land y \approx \forall a. (x \land y) \text{ if } a \text{ does not appear free in } y
\]

Advanced topic III - Dealing with names ctd.

There are several ways to deal with names and their side conditions.

One is higher order rewriting, which uses the \( \lambda \)-calculus to encode names and binding. Alwen Tiu will show you some higher order logic next week, although not specifically in this context.

Another is nominal logic. In this course we’ve defined structures with sets and functions. It turns out if we’re a bit more mathematically adventurous we can define structures in a way that lets us deal with names directly.

Nominal Equational Logic is equational logic merged with nominal logic, and is my major research area: users.cecs.anu.edu.au/~rclouston/.

References

  – Chapter 3 was the ‘textbook’ for this course.
- Term Rewriting and All That, Franz Baader and Tobias Nipkow (1998)
  – Accessible textbook with an emphasis on rewriting.
- Rewrite Systems, Nachum Dershowitz and Jean-Pierre Jouannaud (1990)
  – Chapter 6 of Handbook of Theoretical Computer Science, Volume B: Formal Methods and Semantics, J. van Leeuwen (ed).
  – A survey article on rewriting with lots of pointers to the literature.
New Research

Research on equational logic and term rewriting gets published in any number of logic, theoretical computer science and abstract algebra outlets, but there are two worth noting in particular:

**Algebra universalis** [www.springer.com/birkhauser/mathematics/journal/12](http://www.springer.com/birkhauser/mathematics/journal/12) publishes papers in “universal algebra, lattice theory, and closely related areas”.

The **International Conference on Rewriting Techniques and Applications (RTA)** [rewriting.loria.fr/rta](http://rewriting.loria.fr/rta) “solicits original papers on all aspects of rewriting, including (but not limited to) applications, foundations, frameworks, implementation [and] semantics”.

Student Experience of Teaching

By now all of you (other than the three whose addresses we had wrong) should have been e-mailed a link from ANU’s stats department to fill out a **Student Experience of Teaching** survey.

This survey is to be filled out on my five lectures, not the summer school as a whole.

Remember, words-based feedback is more useful than numbers-based feedback!

Thank you for taking a small amount of time to help me improve my teaching.

And... that's all!

Thanks for listening so attentively, and I'll see you all at dinner tonight!