Non-$\omega$-overlappings TRSs are UN

Stefan Kahrs and Connor Smith
University of Kent
This is about:

• When is the equational theory of a TRS consistent (CON), when does it have unique normal forms (UN),...
• How can we prove it?
• How are these issues related to ω-substitutions, substitutions with infinitary terms in their range ...

Ultimately, we got:

• The result mentioned in the title.
• A new proof technique for proving consistency.
In this talk...

• ...I will gloss over the straightforward parts of the proof (and skip quickly over slides with that content)
• ...and focus mostly on the trickier areas
The issue:

• We are generally interested in the consistency of equational theories, i.e. that not everything is equal to everything (on open terms)
• This is normally proved via confluence (CR): A confluent system is consistent, because:
  • Variables are normal forms
  • Distinct variables are distinct normal forms
• Other kinds of consistency proofs?
  • One can create a non-trivial equational model, but...
  • ...that is hard to “bootstrap” in this case (no CPO structure).
Standard TRS confluence criteria

• For terminating systems:
  • weak confluence, all critical pairs between rewrite rules have common reducts

• For non-terminating systems:
  • There are no overlaps (rules that give rise to critical pairs) in the first place, and...
  • The system is left-linear.
Not just for Confluence, UN & CON too:

\[
F(x,x) \rightarrow A \\
F(C(x),x) \rightarrow B \\
E \rightarrow C(E)
\]

F(E, E) has distinct normal forms A and B. So this is not UN, despite having no overlaps. Moreover, add the rules:

\[
G(A,x,y) \rightarrow x \\
G(B,x,y) \rightarrow y
\]

Now, \(x = G(A,x,y) = G(F(E,E),x,y) = G(F(C(E),E),x,y) = G(B,x,y) = y\). The modified system still has no overlaps but is not CON.
What is going on?

• The system actually did have overlaps, but the standard definition of overlap does not acknowledge them:
  • If we allow for substitutions to replace variables with infinitary terms then the first two rules overlap, i.e. can be applied to the same term
  • And we had a finite term that was “semantical equal” to such an infinite term
• So if we move from LL & non-overlapping to non-ω-overlapping then this counter-example goes away, but is that it?
  • We will not regain confluence, but UN and/or CON?
  • Open problem #79 since 1989.
Reducing the problem (i)

It suffices to look at **CON**:

• (On open terms) **UN** implies **CON**

• Suppose we had a non-$\omega$-overlapping TRS that was not **UN**. Then
  • There are distinct but equivalent normal forms $t$ and $u$.
  • We can get non-$\omega$-unifiable but equivalent ground normal forms $t'$ and $u'$ from $t$ and $u$, possibly via signature extension.
  • We add rules $G(t',x,y) \rightarrow x$, $G(u',x,y) \rightarrow y$, with new ternary symbol $G$.
  • The resulting system remains non-$\omega$-overlapping but it fails **CON** too.
Reducing the problem (ii)

• We can reduce the CON problem of a TRS to the CON problem of its constructor translation.

• The constructor translation of a TRS T is:
  • A constructor TRS (first-order functional program) T’, with...
  • Back-and-forth translations between the terms of T and T’, which preserves variables and...
  • preserves equations either way. (So this preserves and reflects CON.)
  • Aside: our translation also preserves and reflects SN (and WN).
Constructor Translation of a TRS

• Step 1: **duplicate** the signature; the new constructor TRS has a destructor $F_d$ and a constructor $F_c$ for every symbol $F$ of the original signature

• Step 2: for every old rule $F(p_1,\ldots,p_n) \rightarrow r$ we get a new rule: $F_d([p_1], \ldots, [p_n]) \rightarrow [r]$

• Step 3: for every non-variable pattern $G(q_1,\ldots,q_k)$ (strict subterm of a left-hand side of an old rule) we get a new rule: $G_d([q_1], \ldots, [q_k]) \rightarrow G_c([q_1], \ldots, [q_k])$
Example

• Take Combinatory Logic: \( K \ x \ y \to x, S \ x \ y \ z \to x \ z \ (y \ z) \).
• As a TRS this really is: \( A(A(K, x), y) \to x, A(A(A(S, x), y), z) \to A(A(x, z), A(y, z)) \).
• Constructor translation (slightly abbreviated) for this:
  \[
  \begin{align*}
  A_d(A_c(K, x), y) & \to x \\
  A_d(K, x) & \to A_c(K, x) \\
  A_d(A_c(A_c(S, x), y), z) & \to A_d(A_d(x, z), A_d(y, z)) \\
  A_d(A_c(S, x), y) & \to A_c(A_c(S, x), y) \\
  A_d(S, x) & \to A_c(S, x)
  \end{align*}
  \]
What does the translation do to overlaps?

• The translation does not introduce overlaps, except between pattern rules.

• If a TRS is non-\(\omega\)-overlapping then its constructor-translation is “strongly almost non-\(\omega\)-overlapping”. This means: whenever two rules overlap then they are substitution instances of a common generalisation rule.

• For rules derived for patterns with root \(G\) we always have the generalisation \(G_d(x_1, ..., x_n) \rightarrow G_c(x_1, ..., x_n)\)

• This implies that all \(\omega\)-overlaps between rules are trivial (“almost non-\(\omega\)-overlapping”), but is even stronger than that.
Intermission: a tool for reasoning about terms

• Given a relation $R$ between terms, we write $\tilde{R}$ for the relation on terms defined as:

$$t \tilde{R} u \equiv \exists F \in \Sigma, t_1, ..., t_n, u_1, ..., u_n. t = F(t_1, ..., t_n) \land u = F(u_1, ..., u_n) \land \forall i. t_i R u_i$$

• Similarly, $t\hat{R}u$ and $t\bar{R}u$ express the corresponding relations when the shared symbol $F$ is requested to be constructor (in $\Sigma_c$) or destructor (in $\Sigma_d$), respectively.

• A relation $R$ is called $\Sigma$-closed iff $\tilde{R} \subseteq R$. 
Observation: confluence vs. consistency

Why does confluence give us consistency?
• A system is confluent iff the joinability relation \( \downarrow \) is transitive.
• The joinability relation can be defined like this:
  \[
  \downarrow \equiv \mu x. \text{id} \cup x \cup x^{-1} \cup \tilde{x} \cup \rightarrow_R x
  \]
• Thus: joinability is by construction reflexive, symmetric and \( \Sigma \)-closed, and contains rewrite steps. It is just short of transitivity from being a congruence.
• It is also by construction consistent.

Note: there are other relations that share these properties with \( \downarrow \), so they could take its part in consistency proofs.
Computational invariants

- We typically prove confluence by showing that $\Downarrow$ is some kind of computational invariant. For this it needs to “survive” pattern matching. Relation-algebraically, it is this property:
  - A relation $R$ between terms is called constructor-compatible iff we have $\hat{id} \cdot R \cdot \hat{id} \subseteq \hat{R}$
  - In long form: if two constructor-topped terms are related by $R$ then they are topped by the same constructor and their direct subterms are pairwise related by $R$
  - $\Downarrow$ is always constructor-compatible
Pattern Matching; Rule Application

• Let $p$ be a constructor term.
• Let $t = \sigma(p)$ and $u = \theta(p)$ be two substitution instances of $p$.
• If $t R u$ and $R$ is constructor-compatible then $\sigma$ and $\theta$ are pointwise related by $R$. (on variables occurring in $p$)
• If in addition $R$ is $\Sigma$-closed then it must survive parallel rule application with the same rule.
We need more though...

• How can we make sure though that parallel rule applications are with the same rule?

• We have this result: whenever two redexes $t$ and $u$ are related by $\equiv_c$, where $=c$ is a constructor-compatible equivalence, then $t$ and $u$ are instances of two $\omega$-unifiable left-hand sides.

• Why? Informal reason: when we do $\omega$-unification of we perform some equational transformations. If the terms we unify are constructor terms then each transformational step is sound for any constructor-compatible equivalence.
Another invariant

• The semi-joinability relation can be defined like this:

\[ \Downarrow \equiv \mu x. \text{id} \cup x \cup x^{-1} \cup \overline{x} \cup \overline{x} \cdot x \cup \varepsilon \cdot x \]

• So, this relation \( \Downarrow \) is reflexive, symmetric, \( \Sigma \)-closed, closed under prefixing with root-rewrite-steps, and it is closed under prefixing with itself on subterms of destructor-topped terms.

• Regardless of TRS, this relation is also constructor-compatible (and therefore consistent – when we view variables as constructors).

• So, this gives us a more relaxed invariant for consistency proofs than joinability. So, if \( \Downarrow \) is transitive then we are home and dry.
One key difference to joinability

• Joinability is closed under prefixing with $\rightarrow_R$, semi-joinability is closed under prefixing with $\downarrow$ - which is a symmetric relation.
• This gives extra flexibility when trying to construct a common “semi-reduct”.

Term-coalgebras

• $\Sigma$-coalgebras are sets whose elements (nodes) are term-like objects.
  • We may have additional structure, e.g. node labels.
  • The terms associated with nodes could be infinitary, and we may have the same term associated with more than one node.

• Term-coalgebras (for $\Sigma$) is the special case of sets of finite terms, closed under subterms.

• The $\tilde{R}$ notations carry over naturally to term-coalgebras (and indeed arbitrary $\Sigma$-coalgebras).
Transporting definitions

• We can view relations such as $\downarrow$ as being defined (in the same way), for a particular coalgebra $A$.

• However, $\downarrow_A$ is not just the restriction of $\downarrow$ to $A \times A$, because $A$ is not required to include all terms – a redex may lose redex-status.

• In any case, $\downarrow_A$ (on a term-coalgebra $A$) is a subrelation of $\downarrow$ - because of monotonicity of the construction.

• Generally, if $t \downarrow u$ holds then it is also the case that $t \downarrow_A u$ for some finite term-coalgebra $A$. 
Constructing an equivalence

• To prove that $\Downarrow_A$ is an equivalence for a finite term coalgebra $A$ we simply build an equivalence relation which:
  • is constructor-compatible,
  • is a subrelation of $\Downarrow_A$,
  • is $\Sigma$-closed and contains $\Downarrow_A^*$ as a subrelation, and which
  • includes “sufficiently many” redex contractions
How do we build it?

• As a union/find structure (with proof annotations).
• The node set of the structure is all of A.
• An edge from \( a \) to \( b \) requires that either
  \[ a \Downarrow b \text{ or } a \rightarrow^\epsilon b \text{ or } a \equiv_e b \]
  where \( =_e \) is the equivalence defined by the structure.
• We merge equivalence classes by adding an edge to a root of the structure that points to another class.
• We prioritise \( \Downarrow \) edges over redex edges.
Proof graphs...

• The co-algebra is a set of finite terms (closed under subterms).
• These terms are the nodes of our union/find-structure.
• Our invariant relation $\downarrow_A$ is reflexive (and symmetric), i.e. every term is related to itself.
• The edge relation $\rightarrow_e$ of the union/find structure preserves the invariant: $\rightarrow_e \cdot \downarrow_A \subseteq \downarrow_A$
• Therefore all elements in a connected component are $\downarrow_A$-related to each other.
• Overall: any proof graph defines a constructor-compatible equivalence, which is a subrelation of the invariant
Prioritisation

• In a proof graph we can connect at most one edge to a node.
• For terms with a destructor-root this could be a redex-contraction or an “inner”-step.
• We prioritise inner steps, so that all equivalence classes of the relation $\Downarrow_A^*$ are eventually connected in the graph.
• Consequence: the equivalence $\equiv_e$ defined by the proof graph is necessarily $\Sigma$-closed, because it is a subrelation of $\Downarrow_A$; overall it is a constructor-compatible congruence relation.
• But is it the same as $\equiv_R$?
Missing rewrite steps?

Let $a \xrightarrow{\epsilon} b$ be any rewrite step of the co-algebra that is not an edge in the completed proof graph. Then either:

- $a$ is the (local) root of its equivalence class of $\Downarrow_A^*$ and $b$ is also in that class (and therefore $a =_e b$), or...
- The local root of $a$ is some redex $c$, with $c \xrightarrow{\epsilon} d$, and $c =_e d$. Then $a$ and $c$ are connected with inner steps in the proof graph (and we can ensure that these steps lie within $\equiv_e$)
- Hence $a \equiv_e c$ and therefore $b$ and $d$ are related by $=_e$ (because it is a constructor-compatible congruence parallel steps stay within the invariant), and so $a =_e b$. 
Consequences

• $\downarrow_A$ is transitive (for finite A) for “well-behaved” TRSs
• $\downarrow$ is transitive for well-behaved TRSs too:
  • If $t \downarrow u$ and $u \downarrow v$ then for some finite term-coalgebras A and B we have
    $t \downarrow_A u$ and $u \downarrow_B v$.
  • But $C = A \cup B$ is then a term-coalgebra too, we get $t \downarrow_C u$ and $u \downarrow_C v$ by
    monotonicity, $t \downarrow_C v$ by transitivity of $\downarrow_C$, and $t \downarrow v$ by monotonicity.
• Thus, “well-behaved” TRSs are consistent.
• Strongly almost non-$\omega$-overlapping Constructor TRSs are “well-behaved”.
• Therefore: non-$\omega$-overlapping TRSs have unique normal forms.
Future Work

Almost non-$\omega$-overlapping Constructor TRSs

Parallel steps could be with different rules, but we should still have our consistency invariant property.

Relaxing the condition on constructor-compatible equivalences:

We currently require that for all constructor-compatible sub-equivalences $S$ of $\downarrow$ that $CT(S)$ holds for the contracta whenever $\bar{S}$ holds between redexes.

But: one does not need “all”, one only needs “all that are sufficiently large”.