Knuth-Bendix Completion with Modern Termination Checking

Ian Wehrman
ACL2 Seminar
August 20, 2006

Equational Automated Theorem Proving

- Want to solve the **word problem** automatically.
- Does a finite set of identities (a **theory**) entail another identity?

Example Theory: Groups

- For example, the theory of **groups** (G) is axiomatized by three identities:

$$x * 1 \approx x \quad x * x^{-1} \approx 1 \quad x * (y * z) \approx (x * y) * z$$

Word Problem for Groups

- The **word problem** for G: is an identity a consequence of the axioms of group theory?
- E.g., a left-inverse lemma:

$$\boxed{G \models x^{-1} * x \overset{?}{\approx} 1}$$

Group Theory Proof

- Yes, there is a left inverse lemma! Here's the proof:

$$
\begin{aligned}
x^{-1} * x &\approx x^{-1} * (x * 1) && (1)\\
&\approx x^{-1} * (x * (x^{-1} * (x^{-1})^{-1})) && (1)\\
&\approx x^{-1} * ((x * x^{-1}) * (x^{-1})^{-1}) && (3)\\
&\approx x^{-1} * (1 * (x^{-1})^{-1}) && (2)\\
&\approx (x^{-1} * 1) * (x^{-1})^{-1} && (3)\\
&\approx x^{-1} * (x^{-1})^{-1} && (1)\\
&\approx 1 && (2)
\end{aligned}
$$

Automating Group Theory Proofs

- Found proof fully automatically using a tool called **Waldmeister**.
- Implements an algorithm called **completion**.
  - Input: theory* (finite set of identities).
  - Output: equivalent rewriting system (also called a **completion**) used to decide whether or not an identity holds.

Group Theory Completion

$$1 * x \approx x \quad x^{-1} * x \approx 1 \quad (x * y) * z \approx x * (y * z)$$

- Input: G
- Output: rewriting system equivalent to G.
- To **prove** an identity holds, rewrite both sides, then test for syntactic equality.

$$\begin{array}{lll}
1 * x \to x & x * 1 \to x & 1^{-1} \to 1 \\
(x^{-1})^{-1} \to x & (x * y)^{-1} \to y^{-1} * x^{-1} & (x * y) * z \to x * (y * z) \\
x * x^{-1} \to 1 & x^{-1} * x \to 1 & \\
x * (x^{-1} * y) \to y & x^{-1} * (x * y) \to y &
\end{array}$$

Group Theory Proofs Made Easy

- With a completion, it's easy to solve the word problem. Works every time.

$$
\begin{array}{lcl}
(y * x) * (x * y)^{-1} & \rightarrow & (y * x) * (x^{-1} * y^{1}) \\
 & \rightarrow & y * (x * (x^{-1} * y^{-1})) \\
 & \rightarrow & y * y^{-1} \\
 & \rightarrow & 1 \\
(y * x)^{-1} * (x * y) & \rightarrow & (y^{-1} * x^{-1}) * (x * y) \\
 & \rightarrow & y^{-1} * (x^{-1} * (x * y)) \\
 & \rightarrow & y^{-1} * y \\
 & \rightarrow & 1
\end{array}
$$

Another Completion

$$
\begin{array}{ll}
1 * x \approx x & (x * y) * z \approx x * (y * z) \\
x^{-1} * x \approx 1 & h(x * y) \approx h(x) * h(y)
\end{array}
$$

- Input: groups + one endomorphism (GE_1).
- Output: completion for GE_1. Use this to solve the word problem for GE_1. Easy!

$$
\begin{array}{ll}
x * 1 \to x & x * (y * z) \to (x * y) * z \\
1 * x \to x & (x * y)^{-1} \to x^{-1} * y^{-1} \\
x * x^{-1} \to 1 & (x * y) * y^{-1} \to x \\
x^{-1} * x \to 1 & (x * y^{-1}) * y \to x \\
1^{-1} \to 1 & h(x)^{-1} \to h(x^{-1}) \\
h(1) \to 1 & h(x) * h(y) \to h(x * y) \\
(x^{-1})^{-1} \to x & (x * h(y)) * h(z) \to x * h(y * z)
\end{array}
$$

Completion Fails!

$$
\begin{array}{lll}
1 * x \approx x & x^{-1} * x \approx 1 & (x * y) * z \approx x * (y * z) \\
f(x * y) \approx f(x) * f(y) & g(x * y) \approx g(x) * g(y) & f(x) * g(y) \approx g(y) * f(x)
\end{array}
$$

- Input: theory of groups + two commuting endomorphisms (CGE_2).
- Output: ... **not a completion!**

What Went Wrong?

* Beyond the theory, completion also needs an **order**, used to orient each identity.

- Waldmeister accepts an order as input, or makes a rough guess before starting.
- Works for simple theories; fails on CGE_2.
- Why? Waldmeister only looks for orders of a certain class, and no suitable order in that class exists for CGE_2.

Life without a Completion

- Without a completion, we must resort to other methods to solve word problem, e.g. paramodulation, or our heads.
- Other methods generally less efficient, and don't yield completions.
- Completions preferred because they can lead to highly efficient decision procedures (e.g., for SMT).

Our Mission

- **Revise** the algorithm used by Waldmeister.
- Use it to **find a completion** for CGE_2.
- Solve the word problem for CGE_2 (without using our heads).

But first...

- Waldmeister's algorithm relies on results in the exciting field of **term rewriting**.
- Today's agenda:
  - Cover important details from term rewriting theory (particularly **proving termination**.)
  - Discuss **completion** (Waldmeister's algorithm).
  - See why completion fails and then **fix it**.

All About the Word Problem

$$u_1 \approx v_1, u_2 \approx v_2, \ldots, u_n \approx v_n \models t_1 \approx t_n$$

- It's **undecidable** (in general).
- Can decide the word problem for some theories, but not all.

Word Problem Proofs

- How do we know an identity **holds** in a theory? Find a proof.
- Proof is a sequence of terms: starting with one side of the identity and ending with the other side.
- Successive terms created by replacing instances of one side of the theory axioms with instances of the other.
- Easy to check, but hard to find.

Proof Construction by Rewriting

- <u>Idea</u>: **orient** axioms – now called **rules**.
- Replace instances of lhs with instances of rhs – called **rewriting**.
- Rewrite terms to **normal form.**
- Two sides of identity have same normal form iff identity holds.

$$\boxed{s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_n \stackrel{?}{=} t_n \longleftarrow \cdots \longleftarrow t_2 \longleftarrow t_1}$$

Rewriting to Normal Form

- To solve the word problem like this, normal forms must:

  1. require only finitely many steps to compute,
  2. be unique – same end result regardless of reduction sequence.

Properties of Rewriting Systems

- These requirements correspond to two important properties of rewriting systems:
  1. **Termination**: no infinitely long reduction sequences.
  2. **Confluence**: if a term is rewritten to distinct terms, then those terms can be rewritten to a common term (**joined**).
- Termination + confluence = **convergence**.

Rewriting Example 1

- The non-confluent, terminating system

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

applied to term *f(x,g(x))* could yield either reduction sequence:

$$\begin{array}{ll} 1. & f(x,g(x)) \to x \\ 2. & f(x,g(x)) \to f(x,x) \to h(x) \end{array}$$

Rewriting Example 2

- The confluent, nonterminating system

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

applied to term *f(x)* yields this looping reduction sequence:

$$\begin{aligned} & f(x) \to g(h(x)) \to \\ & f(h(x)) \to g(h(h(x))) \to \\ & f(h(h(x))) \to g(h(h(h(x)))) \to \\ & f(h(h(h(x)))) \to g(h(h(h(h(x))))) \to \cdots \end{aligned}$$

Rewriting Example 3

- The convergent system

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

applied to term *ack(3,3)* yields this long reduction sequence:

$$
\begin{aligned}
&ack(3,3) \to ack(2,ack(3,2)) \to ack(2,(ack(2,(ack(3,1))))) \to \\
&ack(2,(ack(2,(ack(2,ack(3,0)))))) \to ack(2,(ack(2,(ack(2,ack(2,1)))))) \to \\
&ack(2,(ack(2,(ack(2,ack(1,ack(2,0))))))) \to \cdots \to 61
\end{aligned}
$$

Proving Rewriting Properties

- To solve the word problem with rewriting, systems must be terminating and confluent.
  - How do we prove these properties?
  - What if we can't?

Decidability of Termination

- If we can't prove termination, we're stuck.
- Unfortunately, termination isn't just undecidable, it's also hard.
  - Undecidable classes are vast: 3-rule monadic systems, single-rule systems.
  - Decidable classes are relatively modest: right-ground.

Termination Warm-up

1. $\{f(g(x)) \to f(h(f(x))) \qquad g(g(x)) \to f(g(h(x)))\}$
2. $\{f(f(x)) \to g(x) \qquad g(g(x)) \to f(x)\}$
3. $\{f(f(x,x),y) \to f(y,y)\}$

- Do the following TRS's terminate? Why?
- (Remember: a counter-example requires exhibiting a term that starts an infinte reduction sequence)

Reduction Orders

- The foundation of most termination proofs is a reduction order.
- <u>Def</u>: A **rewrite order** $>$ is stable under:
  1. contexts: $s > t \rightarrow C[s] > C[t]$,
  2. substitutions: $s > t \rightarrow s\sigma > t\sigma$.
- <u>Def</u>: A **reduction order** is a well-founded rewrite order.

Characterizing Termination

- Reduction orders characterize termination of term rewriting systems.
- Thm: a TRS is terminating iff it is compatible with some reduction order.
- An order > is **compatible** with a TRS if $l > r$ for all rules $l \rightarrow r$.

Proving Termination

- Three ways to show termination:
  1. Semantical – reduction order defined w.r.t. a well-founded algebra.
  2. Syntactical – reduction order defined using the subterm property.
  3. Transformational – apply a termination-preserving map μ, then repeat with μ(R).
- Examples of 1 and 2 today.

Semantical Methods

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

- Idea: interpret ground terms as natural numbers N.
- Want interpretation s.t. rules' rhs are always less than lhs with the usual order <.
- Each rewrite yields smaller interpretation.
- Because (N,<) is well-founded, no infinite reduction sequences are possible.

Example Interpretation

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

- Interpret function symbols as functions in N:
  - $f^N : N \to N$, $g^N : N \to N$, $h^N : N \to N$.
  - $f^N(x) = x$, $g^N(x) = 1+x$, $h^N(x) = x$.
- Show lhs is greater than rhs:

1. $[f(g(x))]^N = 1+x > x = [f(h(f(x)))]^N$
2. $[g(g(x))]^N = 2+x > 1+x = [f(g(h(x)))]^N$

Σ-Algebras

- A **signature** $\Sigma = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- A Σ-**algebra** $\mathcal{A} = (A, \Sigma^{\mathcal{A}})$ s.t.
  $\Sigma^{\mathcal{A}} = \{f^{\mathcal{A}}: A^n \to A \mid f \in \Sigma^n\}$
- $(\mathcal{A},<)$ is a **well-founded** Σ-algebra if $<$ is well-founded on A.
- A **monotone** well-founded Σ-algebra has, $\forall f^{\mathcal{A}} \in \Sigma^{\mathcal{A}}, \forall a,b \in A$:

  $a < b \to f^{\mathcal{A}}(\ldots,a,\ldots) < f^{\mathcal{A}}(\ldots,b,\ldots)$.

Semantical Methods

- <u>Def</u>: Let $\mathcal{A}$ be a well-founded monotone Σ-algebra. Then $t <_{\mathcal{A}} t'$ iff $[t]^{\mathcal{A}} < [t']^{\mathcal{A}}$ for every variable assignment.
- <u>Lem</u>: $<_{\mathcal{A}}$ is a reduction order.
- <u>Thm</u>: A TRS terminates iff it admits a compatible well-founded monotone Σ-algebra.

Polynomial Orders

- Most common algebras used with semantic interpretation are polynomials with coefficients in N over N^+.
- Induces the class of reduction orders called **polynomial orders**.
- Decent heuristics exist for finding polynomials to associate with each function symbol.

Weaknesses of Polynomial Orders

- Existence of suitable algebra is undecidable.
- Worse, proving compatibility is undecidable.
- Proof-theoretic strength is limited. Can't prove termination if TRS either:

  1. has reduction sequences of length > $2^{2^{c|t|}}$,
  2. computes a super-polynomial number-theoretic function.

Syntactical Methods

- We can define reduction orders without relying on a well-founded algebra.
- But any order must be well-founded, and this is just what having a semantic interpretation helped with.

Subterm Ordering

- Idea: $t_1 < t_2$ iff t_1 is a proper subterm of t_2.
- Unfortunately, weak – length of reduction sequences limited by depth of term.
- Can we find an **extension** of this order that remains well-founded?

Simplification Orders

- Thm: **any** strict rewrite order with the subterm property is well-founded.
- Reduction orders with subterm property are called **simplification orders**.
- Important because they (typically):

1. are relatively powerful,
2. have simple recursive definitions,
3. are easily tested for compatibility,
4. are decidable.

Recursive Path Orders

- Common orders: recursive path orders (RPO) and Knuth-Bendix order (KBO).
- Idea behind RPO's:
  - define partial **precedence order** on Σ;
  - decide order by comparing root symbols then recursively comparing arguments.
- RPO's differ in how the arguments are compared: e.g., lexicographically (LPO), as multisets (MPO).

Definition of LPO

Let Σ be a finite signature and $>$ be a strict order on Σ.
Then $s >_{\mathrm{lpo}} t$ iff

LPO1 $t = Var(s)$ and $s \neq t$; or

LPO2 $s = f(s_1, \ldots, s_m)$, $t = g(t_1, \ldots, t_n)$; and

- **LPO2a** $\exists 1 \leq i \leq m : s_i \geq_{\mathrm{lpo}} t$; or
- **LPO2b** $f > g$ and $\forall 1 \leq j \leq n : s >_{\mathrm{lpo}} t_j$; or
- **LPO2c** $f = g$, $\forall 1 \leq j \leq n : s >_{\mathrm{lpo}} t_j$,
  and $\exists 1 \leq i \leq m : s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$
  and $s_i >_{\mathrm{lpo}} t_i$.

LPO Example 1

$$x + 0 \to x \qquad x + s(y) \to s(x + y)$$

- Above TRS shown LPO-terminating with precedence + > s > 0:

1. (LPO1) because rhs is a variable in lhs
2. (LPO2b) + > s in precedence, so check that lhs $>_{lpo}$ of rhs subterm. Yes: (LPO2c) then (LPO1) three times.

LPO Example 2

$$
\begin{array}{ll}
x + 0 \rightarrow x & x + s(y) \rightarrow s(x + y) \\
x * 0 \rightarrow 0 & x * s(y) \rightarrow x + (x * y) \\
exp(x, 0) \rightarrow s(0) & exp(x, s(y)) \rightarrow x * exp(x, y)
\end{array}
$$

- Above TRS shown LPO-terminating with precedence exp > * > + > s > 0.
- Multiplication rules:

1. (LPO2b) * > 0, and 0 has no subterms.
2. (LPO2b) * > +, so check that lhs $>_{lpo}$ of rhs subterms; (LPO1) and (LPO2c), and (LPO1) three times.

LPO Example 3

$$ack(0, x) \to s(x)$$
$$ack(s(x), 0) \to ack(x, s(0))$$
$$ack(s(x), s(y)) \to ack(x, ack(s(x), y))$$

- Above TRS is shown terminating with precedence ack > s > 0

1. (LPO2b) ack > s, then (LPO1).
2. (LPO2c), (LPO1), (LPO2b), (LPO2a), (LPO1).
3. (LPO2c), (LPO1), (LPO2c), (LPO1), ...

LPO is pretty great

- Much stronger than polynomial orders – proves termination of functions that grow faster than any primitive recursive function.
- Compatibility easily checked in poly time.
- Existence is NP-hard – just try all possible total orders on Σ – but decidable.

LPO is not perfect

$$1 * x \approx x \qquad x^{-1} * x \approx 1 \qquad (x * y) * z \approx x * (y * z)$$
$$f(x * y) \approx f(x) * f(y) \qquad g(x * y) \approx g(x) * g(y) \qquad f(x) * g(y) \approx g(y) * f(x)$$

- Recall the theory CGE_2 mentioned earlier.
- Waldmeister fails to complete this system because there is no compatible LPO.
- Why? Stuck on commutativity of *f* and *g*.

Automated Termination Checkers

- There are nifty tools to **automatically** prove termination using previous methods.
- Works for systems that are compatible with any one (or a combination of) a variety of reduction orders.
- E.g., **AProVE**: fast, effective and produces human-readable proofs.
- Will be useful later...

Proving Confluence

- Confluence is undecidable in general, but decidable for rewriting systems that are terminating.
- Proceed by showing joinability of a particular (finite) set of pairs of terms.
- Finite computation that implies confluence.

Joining Critical Pairs

- Start with a common instance of two rules' lhs: s_1.
- Rewrite with both rules to two different terms: $t_2 \leftarrow s_1 \rightarrow t_1$.
- Try to join those terms to a common term: $t_1 \rightarrow s_2 \leftarrow t_2$.
- (t_1, t_2) called a **critical pair**.
- Thm: joinability of all critical pairs implies confluence **for terminating systems**.

Critical Pair Example 1

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

- Common instances of rules' lhs rewrites two ways:

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

Non-Confluent Systems

- But simply orienting the axioms of many theories yields a non-confluent TRS.
- If a system is not confluent, sometimes we can find an **equivalent** system that is.
- Systems are equivalent if an identity holds in one system iff it holds in the other.

Equivalent Confluent Systems

- Start with a terminating system, compatible with reduction order >.
- Compute a non-joinable critical pair (t_1,t_2).
- If $t_1 > t_2$, then **add rule** $t_1 \rightarrow t_2$ to system.
- Now (t_1,t_2) is joinable.
- Continue until all critical pairs are joinable.

Critical Pair Example 2

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

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

- Add unjoinable critical pair as rewrite rule. New, equivalent system:

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

Completion

- Called **completion procedure**, invented by Knuth in '70.
- Completion can help **solve the word problem**.
  - Use the equivalent, convergent rewrite system (the **completion**) to normalize both sides of any identity.
  - If normal forms are the same, identity holds, otherwise it doesn't.

Limits of Completion

- Completion doesn't always work:
  - An unorientable critical pair could be generated (completion **fails**);
  - Critical pair generation might not terminate.
- Fails **only if** reduction order is incompatible with the new rule.
- (Can show that "infinite" executions lead to semidecision procedure.)

Completion Specified Formally

- Completion typically specified as an **inference system**.
- Operates on pairs (E,R) – set of identities and set of rewrite rules.
- Start with $(E_0,\emptyset)$ and finish with $(\emptyset,R_\infty)$.
- E_0 is the theory and R_∞ is an equivalent convergent system (the completion).

Completion as an Inference System

$$
\begin{array}{lcl}
\textsc{Orient:} & \dfrac{(E \cup \{s \dot{\approx} t\}, R)}{(E, R \cup \{s \to t\})} & \text{if } s > t \\[2ex]
\textsc{Deduce:} & \dfrac{(E, R)}{(E \cup \{s \approx t\}, R)} & \text{if } s \leftarrow_R u \to_R t \\[2ex]
\textsc{Delete:} & \dfrac{(E \cup \{s \approx s\}, R)}{(E, R)} & \\[2ex]
\textsc{Simplify:} & \dfrac{(E \cup \{s \dot{\approx} t\}, R)}{(E \cup \{u \dot{\approx} t\}, R)} & \text{if } s \to_R u \\[2ex]
\textsc{Compose:} & \dfrac{(E, R \cup \{s \to t\})}{(E, R \cup \{s \to u\})} & \text{if } t \to_R u \\[2ex]
\textsc{Collapse:} & \dfrac{(E, R \cup \{s \to t\})}{(E \cup \{v \approx t\}, R)} & \text{if } s \overset{\sqsupset}{\to}_R v
\end{array}
$$

Correctness of Completion

- If executions eventually consider all critical pairs (are **fair**) and can orient every identity (is **non-failing**), completion succeeds.
- Theorem: a non-failing, fair execution with identities *E* yields a convergent, equivalent rewriting system *R*, which can be used to solve the word problem for *E*.

Completion and CGE_2

$$
\begin{array}{lll}
1 * x \approx x & x^{-1} * x \approx 1 & (x * y) * z \approx x * (y * z) \\
f(x * y) \approx f(x) * f(y) & g(x * y) \approx g(x) * g(y) & f(x) * g(y) \approx g(y) * f(x)
\end{array}
$$

- Recall: completion didn't work with the two commuting endomorphisms (CGE_2) theory.
- Doesn't fail *per se*, because it never starts.
- How to orient identities? What reduction order to use?

The Reduction Order Requirement

- Completion requires compatible reduction order, usually provided by the user.
- Can't find one in usual classes. We've looked.
- Even if we found one, we couldn't specify it – no orders supported by tools (e.g. Waldmeister) are compatible.
- Without an order, completion is useless.

Issues with Completion

- We'll to solve the following problems:
  1. Compatible orders hard for the user to find.
  2. Implementations of completion only implement a few classes of orders. (Even if a compatible order exists, user can't specify it.)

The Orient Rule

$$\text{ORIENT:} \qquad \frac{(E \cup \{s \stackrel{.}{\approx} t\}, R)}{(E, R \cup \{s \rightarrow t\})} \qquad \text{if } s > t$$

- Trouble is with the user-supplied order.
- Manifested in the **orient** rule – only place the order is needed.
- Completion would work for **more theories** if the user didn't have to explicitly provide an order at the start.

A New Orient Rule

- Idea: what if we **use a termination checker** instead?
- New orient precondition: require that adding $s \rightarrow t$ preserves termination of the rewriting system.
- Implies the **existence** of a compatible reduction order.

New Precondition

- Tentative change to **orient** rule precondition:

$$\text{ORIENT:} \quad \frac{(E \cup \{s \doteq t\}, R)}{(E, R \cup \{s \to t\})} \quad \text{if } s > t$$

Original

Modified

$$\text{ORIENT:} \quad \frac{(E \cup \{s \doteq t\}, R)}{(E, R \cup \{s \to t\})} \quad \text{if } R \cup \{s \to t\} \text{ terminates}$$

Correctness of the New Orient Rule

- Different from standard completion in an important way –
- Termination implies the existence of a compatible order, but the **order could be different** each time the orient rule is applied.
- Like performing completion with **multiple orders**.

Completion with Multiple Orders

- Completion with multiple orders was used for years (without correctness proof) in some systems.
- Useful – if an unorientable identity is encountered, just find another compatible order and keep going.

Multiple Orders Not Correct

- Correctness an open problem for years.
- Settled by Sattler-Klein in ‘94 – not correct.
- Multiple orders with completion can yield non-confluent, non-terminating systems.

A Correct Special Case

- But Sattler-Klein also proved that one kind of multi-ordered completion is correct:
- For finite executions without **compose** or **collapse**, completion works with multiple orders.

Compose and Collapse

COMPOSE:	$\dfrac{(E, R \cup \{s \to t\})}{(E, R \cup \{s \to u\})}$	if $t \to_R u$
COLLAPSE:	$\dfrac{(E, R \cup \{s \to t\})}{(E \cup \{v \approx t\}, R)}$	if $s \overset{\sqsupset}{\to}_R v$

- Why? These are the only rules that change or remove rules from the current rewriting system.
- Without these, the intermediate rewrite systems form an **increasing chain**.
- The **final** order could have been used from the start without failure.

Constraint System

- Could use new orient rule without compose and collapse, but performance suffers.
- Instead: check termination of a **constraint** rewriting system not affected by compose and collapse.
- Lem: Termination of constraint system implies termination of rewriting system and existence of increasing chain of reduction orders.

Revised Completion

$$\begin{array}{lll}
\textsc{Orient}: & \dfrac{(E \cup \{s \mathrel{\dot{\approx}} t\}, R, C)}{(E, R \cup \{s \to t\}, C \cup \{s \to t\})} & \text{if } C \cup \{s \to t\} \text{ terminates} \\
\textsc{Deduce}: & \dfrac{(E, R, C)}{(E \cup \{s \approx t\}, R, C)} & \text{if } s \leftarrow_R u \to_R t \\
\textsc{Delete}: & \dfrac{(E \cup \{s \approx s\}, R, C)}{(E, R, C)} & \\
\textsc{Simplify}: & \dfrac{(E \cup \{s \mathrel{\dot{\approx}} t\}, R, C)}{(E \cup \{u \mathrel{\dot{\approx}} t\}, R, C)} & \text{if } s \to_R u \\
\textsc{Compose}: & \dfrac{(E, R \cup \{s \to t\}, C)}{(E, R \cup \{s \to u\}, C)} & \text{if } t \to_R u \\
\textsc{Collapse}: & \dfrac{(E, R \cup \{s \to t\}, C)}{(E \cup \{v \approx t\}, R, C)} & \text{if } s \overset{\sqsupset}{\to}_R v
\end{array}$$

- Key differences: constraint system *C* and termination predicate in orient precondition.

Completion Search

- What if a if a rule can be oriented two different ways?
- Just try both. **Search** for a correct completion.
- (Search avoids problems with pesky infinite executions mentioned earlier.)
- Breadth-first search guarantees that we will eventually find a completion.

Revised Completion

- Revised method is **correct**.
- Order is **discovered**, not provided.
- With perfect termination-checking ability, the method completes any theory compatible with some reduction order.
- With real termination-checking program that decides a class of orders *O*, then revised method completes any theory compatible with an order in *O*.

Slothrop

- Implementation of revised procedure: Slothrop.
- 3500-line Ocaml program.
- Integrates with AProVE termination checker, and recently with TPA.
- Applications of orient rule preceded with calls to checker to verify termination of constraint system.
- Best heuristic: size(C + E + cp(R))

Completion of CGE_2

- Slothrop completes a variety of theories (e.g., groups and other algebraic structures).
- Completed CGE_2 – **first ever** automatic completion!

$(x * y) * z \to x * (y * z)$	$f(1) \to 1$
$x^{-1} * x \to 1$	$(f(x))^{-1} \to f(x^{-1})$
$x * x^{-1} \to 1$	$f(x) * f(y) \to f(x * y)$
$x * (x^{-1} * y) \to y$	$f(x) * (f(y) * z) \to f(x * y) * z$
$x^{-1} * (x * y) \to y$	$g(1) \to 1$
$(x * y)^{-1} \to y^{-1} * x^{-1}$	$(g(x))^{-1} \to g(x^{-1})$
$1 * x \to x$	$g(x) * g(y) \to g(x * y)$
$x * 1 \to x$	$g(x) * (g(y) * z) \to g(x * y) * z$
$1^{-1} \to 1$	$f(x) * g(y) \to g(y) * f(x)$
$(x^{-1})^{-1} \to x$	$f(x) * (g(y) * z) \to g(y) * (f(x) * z)$

Performance

- Time: 6s to find G completion, 30s for GE_1, 15m for CGE_2.
- Calls to AProVE: 30 calls to complete G, 100 for GE_1, 2500 for CGE_2.
- Most of runtime spent in AProVE, but most calls return in < 0.5s.

Evaluation of Slothrop

- Efficiency is a limitation, but with patience hard completions are found.
- Works well on small-to-medium theories, slow on large theories.
- Not yet able to find completion of CGE_3.

Future Work

- Improved termination checking:
  - Stronger – increase class of theories that can be completed.
  - Faster – find completions faster. Need **incremental** termination checking.
- Improved search heuristic:
  - **Hard question**: how can diverging completion instances be identified?

Conclusion

- Read more: system description in RTA 2006, details in WUCSE-2006-42.
- Download Slothrop online: cl.cse.wustl.edu
- Thanks for sitting through two long talks!

Conclusion

- Fin.

Confluence Details

Local confluence: $(\rightarrow)^{-1} \cdot \rightarrow \subseteq \rightarrow^* \cdot (\rightarrow^*)^{-1}$

Confluence: $(\rightarrow^*)^{-1} \cdot \rightarrow^* \subseteq \rightarrow^* \cdot (\rightarrow^*)^{-1}$

- Critical pair lemma: joinability of critical pairs implies local confluence.
- Newman's lemma: local confluence + termination implies confluence.