AUTOMATED REASONING

SLIDES 17:

ORIENTED PARAMODULATION
Using Orientation to control Paramodulation
Failure in Knuth-Bendix Procedure
Knuth Bendix and Theorem Proving

KB - AR - 2009

Oriented Paramodulation

- We can use the idea of ordering an equation to control paramodulation steps:
- Restrict paramodulation by requiring the replacing term to be **definitely not greater** than the one being replaced.
- In case an equation can be orientated (ie every instance satisfies LHS>RHS) then the restriction allows to order the equation LHS ==> RHS.

Oriented Paramodulation: l= r ∨ C paramodulates into s[u], u not a variable
if lθ =uθ and ¬ (lθ ≤ rθ), where
≤ is a stable monotonic simplification ordering (eg rpo, kbo).
(Method due to Hsiang and Rusinowitch CADE 8, 1986)

Example:
n(x,x)=n(M,x) and the kbo: n(x,x)<n(M,x) if x is bound to t<M;
n(x,x)>n(M,x) if x is bound to s>M;
Can apply oriented paramodulation into P(n(u,v)):
use L to R to give: P(n(M,v)) or R to L to give: P(n(v,v))
Thus θ may be u==v and must check ¬(n(v,v) ≤ n(M,v)) (OK)
Or θ may be u==M and must check ¬(n(M,v) ≤ n(v,v)) (OK)

Oriented Paramodulation (2)

- In case an equation can be orientated (ie every instance satisfies LHS>RHS) then the restriction allows to order the equation LHS ==> RHS.
- l= r ∨ C paramodulates into s[u] (u not a variable) **if** lθ =uθ and ¬(lθ ≤ rθ)

Example: 1. x=a ∨ x=b 2. ¬ g(x) = g(y) ∨ x=y 3. ¬ (g(g(a)) = a) 4. g(a) =>b
Choose a < b < g and ≥rpo (so a<b <g(a) < g(b) < g(g(a)) , …)
5 g(g(a))=a ∨ ¬b=a [P (1+3)] (put x=g(g(a)), and check ¬(g(g(a)) ≤ b)
use x=b for paramodulation)
6 ¬ b=a [R (3+5)]
7 ¬ g(b) = g(a) [R (6+ 2)]
8 ¬ g(b) = a [P (4+3)] (OK g(a) >b)
9 g(b) => b [R (1 + 8)]
10 ¬ b = g(a) [P (9+7)] (OK g(b) >b)
11 [] [R (10+ 4)] (use symmetry)

Notes:
(i) can replace u=v by u=>v if u>v for every instance of u,v
– so g(a)=b becomes g(a)=>b and g(b)=b becomes g(b)=>b
(ii) ¬(lθ ≤ rθ) means "it is not true that for every ground substitution σ, lθσ ≤ rθσ"
i.e. "there is some ground substiution σ, lθσ > rθσ"

Oriented Paramodulation and Predicate Ordering

- Oriented paramodulation can be combined with an ordering on predicate symbols (note the **largest** predicate symbol has highest priority here):
- ≤ is extended to literals as well as terms such that "=" ≤ all predicates

- C1: s=t ∨ D1 can paramodulate by oriented paramodulation into literal with largest predicate in C2 if D1 consists of predicates equal in the order to "="
- C1: E1∨ D1 and C2: ¬E2 ∨ D2 can be resolved if E1σ and E2σ are unifiable and no predicate in D1 is > E1 and no predicate in D2 is > E2.
i.e. E1/E2 use the largest predicates in C1/C2

Example: the Aunt Agatha problem

1. K(d,a), 2. d=a ∨ d=b ∨ d=c, 3. H(b,d), 4. x=b ∨ H(b,x), 5. ¬a =b
6. ¬K(x,y) ∨ H(x,y), 7. ¬H(c,x) ∨ x=b, 8. ¬K(a,a), 9. ¬H(x,f(x))

Order functors as f>d>a>b>c and predicates K>H>'=' (K has highest priority).

10. (1+2, P) K(a,a) ∨ d=b∨ d=c
11. (10+8, R) d=b ∨ d=c
12. (4+9, R) f(b) =b
13. (12+9, P) ¬H(b,b)
14. (11+3, P) H(b,b) ∨ d=c
15. (13+14, R) d=c
16. (1+6, R) H(d,a)
17. (16+15, P) H(c,a)
18. (7+17, R) a=b
19. (18+5, R) []

Combining Oriented Paramodulation and Predicate Ordering: 17aiv

Oriented Paramodulation allows to control the use of paramodulation. It can also be combined with predicate ordering if we treat predicates as functors for the purpose of ordering. It is easiest to make the greatest predicate have the highest priority (in contrast to what we did in Slides 7, but like Otter does), and to give the = predicate lowest priority. In case paramodulation is explicitly simulated by resolution, this behaves similarly to locking the equality axioms as we suggested in Slides 12. We can extend the use of quasi-orderings to other refinements, even if paramodulation is not involved, such as atom ordering and hyper-resolution. Some examples of using these ideas are given on Slide 17av.

An example of an ordering of terms that's combined with a predicate ordering was given in slides 7 (the lexicographic ordering). However, once orderings are combined also with paramodulation steps, we require that the order be a simplification order. For instance, kbo or rpo. If < is such an order, then we can compare two atoms thus: s=P(s1,...,sn)>t=Q(t1,...,tm) if (i) P>Q in the predicate order, or (ii) P=Q, P is not "=" and [s1,...,sn]>*[t1,...,tm], where >* is the lexicographic order based on <. or (iii) P=Q, P is "=" and {s1,s1}>>{t1,t2} (multi-set order because = is symmetric).

Note about Oriented Paramodulation: Since ≤ is stable, ¬(lθ ≤ rθ) means that "it is not the case that lθσ ≤ rθσ for every substitution σ". Hence ¬(lθ ≤ rθ) → lθσ > rθσ for <u>some ground substitution</u> σ. Hence it is possible to have both ¬(lθ1 ≤ rθ1) and ¬(rθ2 ≤ lθ2) (for different substitutions θ1 and θ2) and the equation l=r could be used in both directions but at different times.

Further Examples: (Extension to atom ordering) 17av

1) P(0) 2) ¬P(x) ∨ <u>P(s(x))</u>

P(s(x)) > P(x) because s(x) > x (using any simplification ordering)
so P(s(x)) is the literal that must be selected in (2).
There are then no ordered resolvents between these clauses.

Group Theory problem:

1. f(a,b) = c 2. ¬f(b,a) = c 3. f(x,x) = e 4. f(x,e) = x
5. f(e,x) = x 6. f(f(x,y),z) = f(x, f(y,z))

Use kbo based on length of terms.

7. (1+6, P)	f(a, f(b,z)) = f(c,z)	10. (9+6, P)	f(a, f(c,z)) = f(b,z)
8. (3+6+5, P)	f(x,f(x,z)) = z	11. (10+ 3+4, P)	f(b,c) = a
9. (1+8, P)	f(a,c) = b	12. (8+11, P)	f(b,a) = c
		13. (2+12, R)	[ ]

Completeness of the method is shown in Hsiang and Rusinowitch, CADE- 8.

When Knuth Bendix Completion Fails 17bi

The Knuth Bendix procedure **fails** if an equation cannot be orientated

- eg x+y = y+x, or f(x, g(z)) = f(g(z), x)
  leads to circular rewriting as in 2+3 => 3+2 => 2+3 ...

- One way to avoid failure is to allow superposition to/from either side of an non-orientable equation and use the ideas of oriented paramodulation.

- eg x +0 = 0+x <u>can</u> be oriented if 0 is smallest element and x ≠ 0

- Can superpose l = r and s => t as long as ¬(lθ ≤ rθ);
  (θ is either mgu of l and a subterm of s, or of s and a subterm of l.)
  i.e. there are some instances for which lθ > rθ (else lθ ≤ rθ).
- > must be total on ground terms;
- when *rewriting* using l = r, must have lθ >rθ. (Ideas due to Bachmair)

• eg j(f(x),y) = j(y,g(x)) and j(v,v) => v, using kbo based on counting terms
<u>Can</u> superpose j(v,v) and j(f(x),y) if ¬(j(f(x),f(x)) ≤ j(f(x),g(x)))
In fact, j(f(x)),f(x)) > j(f(x),g(x)) exactly if f>g giving j(f(x),g(x)) => f(x)
<u>Can</u> superpose j(v,v) and j(y,g(x)) if ¬(j(g(x),g(x)) ≤ j(f(x),g(x)))
In fact, if f>g then j(f(x),g(x)) > j(g(x),g(x)).

When Knuth Bendix Completion Fails (2) 17bii

- Can superpose l = r and s => t as long as ¬(lθ ≤ rθ);
  (θ is either mgu of l and a subterm of s, or of s and a subterm of l.)
  i.e. there are some instances for which lθ > rθ (else lθ ≤ rθ).
- > must be total on ground terms;
- when *rewriting* using l = r, must have lθ >rθ. (Ideas due to Bachmair)

(4) x+ -x => 0 (5) x + 0 => x (6) u+v = v+u

Use kbo: s >≈ t iff # functors in s ≤ # functors in t, and 0 $<_1$ all other terms.

(5)+(6) give 0+x => x;
check OK: ¬(x+0 ≤0+x) since for some x (i.e. x≠0) x+0>0+x

(4)+(6) give -x+x =>0; not OK since ¬(x+-x≤-x+x) is false

This method works because the transformation steps applied to any ground proof (using equations) to turn it into a rewrite proof by critical pair formation can be lifted to the general level. The lifted proof will not have been excluded by the restrictions:

• if lθ ≤ rθ (i.e. an excluded step) then all instances of it would lead to excluded steps too; these excluded steps could not have been part of the transformation process of the original ground derivation, a contradiction.

Examples using the orientation restriction

17biii

(1) n(x,x) = n(M,x) (2) n(g(u,v),x) => n(u,n(v,x))

Use kbo: s $\geq_{kbo}$ t if #symbols in s ≥ #symbols in t.

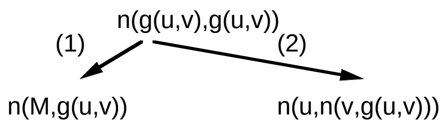
Cannot order (1): x>M==>n(x,x)>n(M,x) and x<M ==>n(M,x)>n(x,x)

Form critical pair – unify LHS of 1 with LHS 2

Check: ¬(n(g(u,v),g(u,v)) ≤n(M,g(u,v)))

In fact, n(g(u,v),g(u,v)) > n(M,g(u,v)) (so OK)

gives new rule
n(u,n(v,g(u,v))) => n(M,g(u,v))



Example of a special case (this slide is not examinable)

17biv

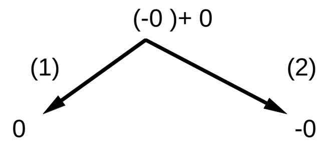
For a commutative and associative operator (eg +, or set union), there is a special unification algorithm called *AC-unification*, which takes these properties into account during superposition. The result is always a finite number of unifiers (possibly none). (See Bundy.)

Commutativity and associativity can also be included when rewriting

eg x+-x =>0 together with commutativity of + allows -(b+a)+(a+b) => 0:

-(b+a)+(a+b) can be unified with x+-x with substitution x==(b+a) or x==(a+b) by using commutativity (twice) => 0 (by the rule)

(Note that adding x+y=y+x explicitly will not necessarily allow to derive -x+x=>0 from x+-x =>0 by oriented paramodulation because of the oriented restriction: -x+x ≥ x+-x is likely the case; eg if kbo is used.)



eg (1) -x+x => 0 (2) 0+y => y
yield the critical pair -0 = 0.

x+-x = -x+x (by commutativity)
then obtain substitutions x==0 and y==-0;

adding x+y=y+x explicitly here allows to derive x +-x => 0, but not y +0=>y, since 0 +y ≤ y+0

Using Knuth Bendix Completion as a Theorem Prover

Recall that *narrowing* is like rewriting except variables can be bound in the term that is to be rewritten.

• eg g(a,z) can be narrowed by g(a,b)=>c into c (bind z==b)

• Consider goals of the form ∃x[t1[x] = t2[x]] and data restricted to equations.
• The negated goal is ∀x[¬(t1[x]= t2[x])] ==> ¬(t1[x1]=t2[x1]) (using free variable rule) and then the two sides can be *narrowed* until a substitution is found that makes both sides equal;
• The resulting inequation can then be resolved with x=x.
• The Knuth Bendix procedure can also be applied incrementally to the rewrite rules and the constrained form (of Slides 17b) is applied to equations that cannot be oriented. This copes both with failure and divergence.

Example: (1) g(a,b) => a (2) g(g(x,y),y) => h(y,x)

Superposition yields g(g(a,b),b) =>* a (use (1) twice) and => h(b,a) (by (2)) giving (3) h(b,a) => a

Suppose the goal is ∃z[g(a,z)= h(z,a)]. Negated, this is ∀z [¬(g(a,z)=h(z,a))].

Using the rules (1) and (3) we get ¬(a = h(b,a)) (by (1) and binding z==b) and then ¬(a = a) (by (3)), which resolves with x=x.

17ci

Example

17cii

1 n(x,x) = n(M,x)
2 n(g(u,v),x)=>n(u,n(v,x))
3 n(z,z) ≠ z
4 x= x

Use kbo: s$\geq_{kbo}$t if #symbols in s ≥ #symbols in t (similar to slide 17bii)

(5) (1+3) n(M,z) ≠ z
(Check: ¬ (n(x,x) ≤ n(M,x)), True - if x>M then n(x,x)>n(M,x)

(6) (1+2) n(u,n(v,g(u,v))) => n(M,g(u,v)) (see 17bii for details of this step)

(7) (5+6) n(M,g(M,v1)) ≠ n(v1,g(M,v1)) (u==M and z==n(v1,g(M,v1)))

(8) (7+4) (v1 ==M) [] {z== n(v1,g(M,v1)) == n(M,g(M,M)) }

Question: Are there any other solutions?

Summary of Slides 17

1. The Knuth Bendix procedure normally has three outcomes: success (a confluent and terminating set of rules is produced), failure (some rule cannot be oriented) and divergence (there are an infinite number of rules). Oriented paramodulation gives ideas on how to deal with failure.

2. Oriented paramodulation restricts paramodulation according to some term ordering. It can be combined with resolution restricted by atom ordering. An equation $l=r$ may be used for paramodulation from l to r as long as there are some instances such that $l\theta>r\theta$. Otherwise, $r\geq l$ and it must be used in that direction.

3. In the KB procedure similarly, superposition is allowed between $l=r$ and $s=>t$ if $l\theta\sigma>r\theta\sigma$ is for some substitution σ, where θ is the unifying substitution of the superposition step.

4. There are special procedures for the particular case of an associative and commutative operator, eg +, in which the properties are built into the unification.

5.The Knuth Bendix procedure can be used as a theorem prover. The goal (often of the form $\exists x[\ t1[x] = t2[x]\ ]$) is negated to give $\forall x[\ t1[x] \neq t2[x]\ ]$. Knuth Bendix is applied to generate rewrite rules and they are used in narrowing steps to reduce both sides of the inequality to a common term. Resolution with $x=x$ then gives [ ]. Even if the KB procedure diverges, interleaving of rule generation with narrowing can give a solution.