

AUTOMATED REASONING

SLIDES 15:

TERMINATION OF REWRITE SYSTEMS

Properties for termination

Two useful partial orders: kbo and rpo

The ERUDIO tool (for interest only)

KB - AR - 2013

Termination of Rewrite Systems (Overview)

15ai

Q: Why should we want terminating rewrite systems?

A: We've seen that to be useful a set of rewrite rules should be complete; to check completeness uses the Knuth Bendix procedure; this requires to apply rewriting to critical terms ==> termination

(Fact D) Given a set of rewrite rules R, if $>$ is a *monotonic* and *well-founded ordering* on terms such that $l\sigma > r\sigma$ for each rule $l \rightarrow r$ and each ground substitution σ , then R will be terminating

Informally,

an ordering is *monotonic* if rewriting a term by a specific rule always makes it smaller, wherever the match to the rule appears
e.g. $h(x) \rightarrow x$ can be applied to $h(a)$, to $h(f(a))$, $f(h(a))$, $g(g(h(x)), h(x))$ etc.

an ordering is *well-founded* if there's no infinite sequence of decreasing terms

Examples: (1) $f(e, x) \Rightarrow x$ (2) $f(i(x), x) \Rightarrow e$ (3) $f(x, g(y)) \Rightarrow f(g(y), x)$
(4) $g(g(h(x))) \Rightarrow h(g(x))$ (5) $h(h(x)) \Rightarrow h(g(h(x)))$

Would you guess any of these rules (on its own) is terminating?

These slides will give some foolproof methods to determine termination

Termination of Rewrite Systems (2)

15aii

Overview of partial orders and well-foundedness

- A **partial order relation** " $>$ " is a transitive and irreflexive relation
i.e. $\forall x, y, z [x > y \text{ and } y > z \rightarrow x > z]$, $\forall x. \neg(x > x)$
 - It is also non-symmetric - i.e. $\forall x, y [x > y \rightarrow \neg(y > x)]$ (derivable)
 - A partial order $>$ is usually written in infix notation - $x > y$ rather than $>(x, y)$
 - Most relations we consider are *total* on ground terms (ie $x > y$ or $y > x$)
 - A partial order $>$ is well founded on a set of terms S
if there is no infinite descending chain $t_1 > t_1 > \dots > t_i > \dots$
eg1 S is the set of integers > -10 and $>$ is the ordinary "greater-than" relation
eg2 Any relation on a finite set S
- Note:** $s < t$ is the same as $t > s$; $s \geq t$ means $s > t$ or $s = t$; if $>$ is a partial order then \geq is reflexive ($\forall x. x \geq x$) and antisymmetric ($\forall x, y [x \geq y \text{ and } y \geq x \rightarrow x = y]$)

Exercise (ppt): $s > t$ if $\#(s) > \#(t)$ – for ground term s $\#(s)$ is the number of symbols (constants or functions) in s

Is $>$ a partial order (is it transitive? irreflexive? non-symmetric?)
Is $>$ a total order? Is it well-founded?

Termination of Rewrite Systems (based on Dershowitz, JSC, 3, 87)

15ai

The above paper by Dershowitz contains a wealth of information about orderings for rewrite rules. He introduces many ad hoc orderings, as well as some specific orderings, of which we'll cover two: "recursive path ordering" (rpo) and "Knuth Bendix ordering" (kbo). In the optional material of these slides is a third ordering, "lexicographic path ordering" (lpo), as well as some ad hoc orderings; as further examples. For our purposes kbo and rpo should be enough, though lpo is sometimes useful, if you're interested.

We will not include proofs that kbo and rpo are indeed partial orders - but assume that they are. If you're interested, you can check in the paper.

There is an excellent tool written by an MSc student Andrei Dvornik, called ERUDIO, which allows you to see the different orderings in practice; it gives justifications for ordering a rule in a particular direction, if that direction leads to a termination order. Some notes on ERUDIO are in the optional material for these slides. We'll see it in action later.

Termination of Rewrite Systems (3)

15aiv

Some basic properties relevant for termination

- **Monotonicity:** if $t > u$ then $f(\dots t \dots) > f(\dots u \dots)$.
i.e. reducing a subterm reduces any superterm of it.
e.g. would like to be sure that if $a < b$ then $g(h(a)) < g(h(b))$.
- **Simplification:** A monotonic ordering $>$ is called a *simplification* ordering if for all ground terms $t, f(\dots t \dots) > t$.
Most standard orderings used to prove termination are simplification orderings.
- **Stability:** if $t > u$ then $t\sigma > u\sigma$ for all ground substitutions σ .
i.e. enables $>$ to be applied between non-ground terms.

Example: $s > t$ if $\#(s) > \#(t)$ – for ground term s $\#(s)$ is the number of symbols (constants or functions) in s

Monotonic? – yes: if s has more symbols than t , then $f(\dots, s, \dots)$ has more symbols than $f(\dots, t, \dots)$ for any functor f

Simplification? – yes: $f(\dots, s, \dots)$ has more symbols than s

Stable? – depends:

eg we can say $f(x, x) > g(x)$ – whatever ground term x is $\#f(x, x) > \#g(x)$
but not $f(x, y) > h(x, x)$ – if x is bound to a longer term than y , $\#f(x, y) < \#h(x, x)$

Checking a ruleset R for termination

15av

(Fact D) Given a set of rewrite rules R , if $>$ is a *monotonic* and *well-founded ordering* on terms such that $l\sigma > r\sigma$ for each rule $l \rightarrow r$ and each ground substitution σ , then R will be terminating

Examples: (1) $f(e, x) \Rightarrow x$ (2) $f(i(x), x) \Rightarrow e$ (3) $f(x, g(y)) \Rightarrow f(g(y), x)$
(4) $g(g(h(x))) \Rightarrow h(g(x))$ (5) $h(h(x)) \Rightarrow h(g(h(x)))$

Use $s > t$ if $\#(s) > \#(t)$

For each ground substitution for x , clearly $LHS(2) > RHS(2)$

Also $LHS(1) > RHS(1)$: $\#f(e, x) = 2 + \#x > \#x$ ($\#x$ means number symbols in x)

Similarly for (4)

The order is monotonic: if $s < t$ then $f(s, z) < f(t, z)$, $f(z, s) < f(z, t)$, $i(s) < i(t)$ (for any z), etc.

The order is well-founded as $\#s$ is ≥ 0 .

What about (5)? (This is a bit harder!) See optional material for an answer

It would be much easier if we could apply a standard set of simple tests -
it turns out we can, though we have to generalise the notion of ordering a little

15avii

Well-founded Ordering:

An order $<$ is a well-founded ordering on a set of terms if there is no infinite descending sequence of terms $s_0 > s_1 > s_2 > \dots$. eg $<$ is well-founded on $\{\text{integers} > k\}$ for any particular choice of k , but not on the set of integers. For our purposes we assume the s_i are derived by rewriting:
s.t. $s_0 \Rightarrow s_1, \dots, s_i \Rightarrow s_{i+1}$

Proof of Fact D:

Let R be a set of rewrite rules and $<$ be a well-founded monotonic term order. Assume $l\sigma > r\sigma$ for each rule and each ground substitution σ . Suppose that $s_0 \Rightarrow s_1 \Rightarrow \dots \Rightarrow s_n \dots$ is a non-terminating ground rewrite sequence using R , then, by assumption $s_0 > s_1 > \dots > s_n > \dots$ (since each rewrite uses a rule and $<$ is monotonic). But as $>$ is well-founded the sequence cannot continue forever, so the original rewrites cannot do so either. This is a contradiction, so the original assumption is false.

In case the rewrite sequence includes variables, instantiate to obtain a non-terminating ground rewrite sequence and use the above.

A little bit more notation (Dershowitz):

15bi

Recall:

A *standard partial order* $<$ is irreflexive and transitive (and $s < t$ implies $\neg(t < s)$ WHY?)

The relation \leq defined by $s \leq t$ iff $s < t$ or $s = t$, where $<$ is a standard partial order, is reflexive, transitive and anti-symmetric (i.e. $s \leq t$ and $t \leq s$ implies $s = t$)

Sometimes we want to relax anti-symmetry to allow $R(s,t)$ and $R(t,s)$ even if $s \neq t$.

A *quasi-partial order* \approx is reflexive and transitive but need not be anti-symmetric. i.e. if $s \approx t$ and $t \approx s$, then we say $s \approx t$

For a quasi-order \approx , we define $s < t$ iff $s \approx t$ and not $(t \approx s)$.

There are some famous quasi-orders: \leq_{kbo} and \leq_{rpo} etc. (see slides 15ci - 15civ).

To show termination of R using a simplification quasi-order \approx such as \leq_{kbo} or \leq_{rpo} show each rule in R satisfies $l \sigma > r \sigma$ for all substitutions σ .

Quasi-orderings can be simplification orderings, monotonic, stable, etc. The definitions of those are simply given by using the quasi-order \approx in place of the standard partial order $<$ properties.

Knuth - Bendix ordering (kbo) (ppt)

15ci

$s = (f(s_1 \dots s_m)) \geq_{kbo} t = (g(t_1 \dots t_n))$

- if $s > t$ (where $>$ is a simplification quasi-ordering on ground terms) (definitely $>$)
- or $s \approx t$ and $f >_1 g$ ($>_1$ applies to functors here) (definitely $>$)
- or $s \approx t$, $f = g$ and $(s_1 \dots s_m) \geq^* (t_1 \dots t_n)$

\geq^* is the lexicographic ordering induced by \geq_{kbo}

To use kbo to show termination of R: show each rule in R satisfies $l \sigma > r \sigma$ for all substitutions σ .

- | | |
|--------------------------------------|---|
| 1. $0+x \Rightarrow x$ | $s \approx t$
if # occurrences of $+/-$ in $s \leq$
occurrences of $+/-$ in t ; |
| 2. $(-x) + x \Rightarrow 0$ | |
| 3. $(x+y) + z \Rightarrow x + (y+z)$ | |

1. is ok since $\# +/-$ in LHS $>$ $\# +/-$ in RHS.
2. is ok since $\# +/-$ in LHS $\geq 2 >$ $\# +/-$ in RHS = 0.
3. is ok since $\# +/-$ in LHS = $\# +/-$ in RHS, both terms have outer $+$, and $((x+y), z) \geq^*_{kbo} (x, (y+z))$ as $\# +/-$ in $x+y >$ $\# +/-$ in x ; (i.e. lexicographic order based on kbo)

Also, \approx is a simplification ordering: eg $x < y \Rightarrow -x < -y$ and $-x > x$.

Recursive Path Ordering (rpo)

15cii

$s = (f(s_1 \dots s_m)) \geq_{rpo} t = (g(t_1 \dots t_n))$

- if $s_i \geq_{rpo} t$ for some $i = 1 \dots m$ (definitely $>$)
- or $f >_1 g$ and $s >_{rpo} t_j$ for all $j = 1 \dots n$ ($>_1$ orders functors) (definitely $>$)
- or $f = g$ and $\{s_1 \dots s_m\} \gg \{t_1 \dots t_n\}$ (\gg is a multi-set ordering)

Multi-set Ordering (ppt)

- A *multi-set* over a set of terms E with order $>$ is a mapping m from E to \mathbb{N} . e.g. $S = \{3,3,4,0\} = \{3 \rightarrow 2, 4 \rightarrow 1, 0 \rightarrow 1\}$ or $S(3) = 2, S(4) = 1, S(0) = 1$.
- If S is a multi-set then $d(S) = \{\text{elements in S (as a set)}\}$
- If e in E and not(e in $d(S)$) then $S(e) = 0$
- $S \gg T$ iff $\forall e: e \text{ in } d(T) [S(e) \geq T(e) \vee \exists g [g \text{ in } d(S) \wedge g > e \wedge S(g) > T(g)]]$

$\{3,3,4,0\} \gg \{3,3,2,2,1,1\}$ if $\{4,0\} \gg \{2,2,1,1\}$ (remove occurrences of = elements)
 $\{4,0\} \gg \{2,2,1,1\}$ each element in $\{2,2,1,1\}$ is dominated by an element in $\{4,0\}$

(Here 4 dominates all of 2,2,1,1.)

$\{3,3,4,0\} \gg \{3,3,3\}$ if $\{4,0\} \gg \{3\}$. (4 dominates 3 so OK)

$\{4,3,3\} \gg \{4,3\}$ if $\{3\} \gg \emptyset$. (OK)

$\{4,1,1\} \gg \{4,2\}$ if $\{1,1\} \gg \{2\}$. (Not OK as 1 doesn't dominate 2)

Example of using recursive path ordering (ppt)

15ciii

1. $\neg \neg x \Rightarrow x$
2. $\neg(x \wedge y) \Rightarrow \neg x \vee \neg y$
3. $\neg(x \vee y) \Rightarrow \neg x \wedge \neg y$
4. $x \wedge (y \vee z) \Rightarrow (x \wedge y) \vee (x \wedge z)$
5. $(y \vee z) \wedge x \Rightarrow (y \wedge x) \vee (z \wedge x)$

- 1. $\neg \neg x \geq_{rpo} x$ {x is a subterm} (use $n(x)$ for $\neg x$ if you prefer)
- 2. $\neg(x \wedge y) \geq_{rpo} \neg x \vee \neg y$ if $\neg(x \wedge y) >_{rpo} \neg x$ and $>_{rpo} \neg y$ (choose $\neg >_1 \vee$)
 i.e. if $\{x \wedge y\} \gg \{x\}$ (and $\gg \{y\}$) which it is as x/y are subterms of $x \wedge y$ (use $a(x,y)$ for $x \wedge y$, and $o(x,y)$ for $x \vee y$ if you prefer)
- 3 similar
- 4. $x \wedge (y \vee z) \geq_{rpo} (x \wedge y) \vee (x \wedge z)$ if $x \wedge (y \vee z) >_{rpo} x \wedge y$ and $>_{rpo} x \wedge z$ (choose $\wedge >_1 \vee$)
 i.e. if $\{x, (y \vee z)\} \gg \{x, y\}$ and $\gg \{x, z\}$ which they are.
- 5 similar

Exercise: Suppose that s and t are terms, and t is a subterm of s, or of a subterm of s. When comparing s and t by rpo, explain why the first case will always hold (though it may have to be applied more than once).

The two orderings kbo, and rpo

The **Knuth Bendix Ordering (kbo)** (on 15ci) is the easiest to use. To apply it you need an order on functors that you choose and a quasi-simplification order $<=>$ on ground terms. A standard choice for $<=>$ is the number of symbols, but others are possible. If you use another order, you just need to show it is a simplification order (ie if $x <=> y$ then $f(\dots x \dots) <=> f(\dots y \dots)$ for all functors f , and that $f(\dots, x, \dots) > x$.) In Case 3 the lex ordering is a dictionary ordering based on the underlying kbo. That is, to compare two lists of terms, compare the lists as you would compare words in a dictionary.

The **Recursive Path Ordering (rpo)** (on 15cii) is next easiest. There are 3 cases. To show $s \geq_{rpo} t$, Case 1 checks if an argument of $s \geq_{rpo} t$. This will be true, for example, if t occurs as a subterm of s , for you can recursively apply this case until you have extracted t as a subterm of s . If Case 1 doesn't hold, then look at the outer functors of s and t . Note that the second condition requires s to be definitely greater than t . In Case 3 a multi-set ordering is used. Despite the complicated definition it is easy to check. First strike out identical terms from the two lists. Next, take each element e left in t and check there is an element left in s that is larger than e . (**Exercise:** Show this procedure satisfies the given definition of $>>$.)

A third ordering, the **Lexicographic Path Ordering (lpo)** is given in the optional part of these slides.

All three orderings were proved by Dershowitz to be well-founded simplification orderings.

Summary of Slides 15

1. Rewrite systems are most useful when they are terminating. There are several ad hoc methods to show termination, the most useful of which is stated in Fact D.
2. Important properties of term orderings are *stability*: if $s < t$ then also $s\theta < t\theta$; *monotonicity*: if $s < t$ then $f(\dots s \dots) < f(\dots t \dots)$; and *simplification*: $t < f(\dots t \dots)$.
3. $s \leq t$ is defined as $s < t$ or $s = t$ (for a partial order \leq).
4. The more useful orders are based on *quasi-orderings*, which are (partial) orders that are not anti-symmetric. That is, it is possible for two terms s and t to satisfy $s <=> t$ and $t <=> s$ and yet for $s \neq t$. $s < t$ iff $s <=> t$ and not $(t <=> s)$.
5. The two quasi-orderings considered here are *knuth bendix ordering* (kbo) and *recursive path ordering* (rpo). Both orders depend also on an ordering of function symbols, which can be chosen by the user. The knuth bendix ordering depends also on a total order on ground terms that is also a simplification ordering. The recursive path ordering uses the concept of multi-set ordering.

START of OPTIONAL MATERIAL (SLIDES 15)

A third useful partial order: lpo
Extra examples of ad hoc orders
ERUDIO

More Examples (solutions on 15eii)

(Fact E) Even if $>$ is not monotonic, Fact D can be relaxed:

R will be terminating if $>$ is a well-founded ordering on terms, $l\sigma > r\sigma$ for each rule and each ground substitution σ , and $s \Rightarrow^* t$ and $s > t$ implies $f(\dots s \dots) > f(\dots t \dots)$ – i.e. R is monotonic at least on terms that rewrite to each other.

This example uses Fact E $f(f(x)) \Rightarrow f(g(f(x)))$?

(Try $ffgfgfa \Rightarrow ?$ - does it terminate?)

(i) Find a well-founded order (not necessarily monotonic) and show $LHS > RHS$ for all x

(ii) Show the order is monotonic on terms that rewrite to each other.

Hint: consider counting adjacent occurrences of f

This example can also be shown to be terminating by Fact E, but it needs a more complicated ordering:

$f(g(x)) \Rightarrow g(g(f(x)))$

(Try $fgfggfa \Rightarrow ?$ does it terminate?)

Hint: What happens to the occurrences of f to the left of an occurrence of g ?

$f(f(x)) \Rightarrow f(g(f(x)))$.

Count #pairs of adjacent fs. It is clear that for any x the number of adjacent pairs of f is reduced by 1 after applying the rule. As counts are ≥ 0 the ordering is well-founded.

The ordering is not monotonic though: $g(f(f(a)))$ has 1 pair of adjacent f and $f(a)$ has none. So $g(f(f(a))) > f(a)$. But $f(g(f(f(a))))$ is not $> f(f(a))$ as both have 1 pair of adjacent f . However, $g(f(f(a)))$ does not rewrite to $f(a)$. If this is a general property, then we can apply FACT E:

Comparing $f(s)$ and $f(t)$ it's clear this property **is** maintained (and also for $g(s)$ and $g(t)$). Assume $s \Rightarrow t$, then s has the form $\langle \dots \rangle f f \langle \dots \rangle$ and t has the form $\langle \dots \rangle f g f \langle \dots \rangle$. If the number of pairs of f in $s >$ number of pairs in t , then it's easy to argue, by considering cases and counting, that the same holds for $f(s)$ (i.e. $f \langle \dots \rangle f f \langle \dots \rangle$) and $f(t)$ (i.e. $f \langle \dots \rangle f g f \langle \dots \rangle$).

$f(g(x)) \Rightarrow g(g(f(x)))$.

Count #gs to right of each f . Note #fs remains fixed for rewriting a given term – let #fs = n . Let (a_i) be the number of gs to right of i 'th f from the left.

Define $(a_1, a_2, \dots, a_n) > (b_1, b_2, \dots, b_n)$ iff $a_i > b_i$ and $\forall j : i+1 \leq j \leq n \rightarrow (a_j = b_j)$

ie i is first position from right at which $a_i \neq b_i$

e.g. $f g f g f a \Rightarrow f g g f g f a$ and the counts are $(3, 2, 0)$ and $(4, 1, 0)$; Notice $(3, 2, 0) > (4, 1, 0)$

Check $>$ is well-founded and LHS $>$ RHS for all substitutions of x

Well founded: minimal counts $= (0 \dots 0)$ ($n \times 0$ for n fs). eg $\text{count}(g \dots g f f a) = (0, 0, 0)$

Suppose $f(g(x))$ has k occurrences of f with count $= (c_k, \dots, c_1)$.

Then $g(g(f(x)))$ has count $= (c_k - 1, \dots, c_2, c_1)$ which is $< (c_k, \dots, c_1)$.

Also, check that if $s \Rightarrow^* t$ and $s > t$ then $f(s) > f(t)$ and $g(s) > g(t)$ (do this in a similar way to above – notice $g(s)$ has the same count value as s). Then use Fact E.

Lexicographic path ordering (lpo)

$s (= f(s_1 \dots s_m)) \geq_{lpo} t (= g(t_1 \dots t_n))$ if

- $s_i \geq_{lpo} t$ for some $i = 1 \dots m$, or
- $f >_1 g$ and $s >_{lpo} t_j$ for all $j = 1 \dots n$, or
- $f = g$ and $(s_1 \dots s_m) \geq_{lpo} (t_1 \dots t_n)$ and $s >_{lpo} t_j$ for all $j = 2 \dots n$, where \geq_{lpo}^* is the lexicographic ordering induced by \geq_{lpo}

Example of using lexicographic path ordering

$(x+y)+z \geq_{lpo} x+(y+z)$: main functor = + in both cases so case 3

- $((x+y), z) \geq_{lpo} (x, (y+z))$ since $(x+y) \geq_{lpo} x$ (because x is a subterm of $x+y$), and
- $(x+y)+z \geq_{lpo} (y+z)$: main functor = + in both, so case 3
 - $((x+y), z) \geq_{lpo} (y, z)$ since $(x+y) \geq_{lpo} y$, and
 - $(x+y)+z \geq_{lpo} z$ (because z is a subterm)

The **Lexicographic Path Ordering (lpo)** is similar to rpo. The two orders only differ when the two terms to be compared have the same top level functor, that is for Case 3. In Case 3 you must first check the arguments of s and t are pairwise lexicographically ordered (as in kbo) and then recursively check s is definitely $>_{lpo}$ than each argument of t (other than the first). You can show that if the first lexicographic condition finds (say) argument 2 of $s >_{lpo}$ argument 2 of t , then the second condition can start at argument 3, since Case 1 would hold for $s >_{lpo}$ argument 2. For example, to compare $f(x, b, y)$ and $f(x, a, y)$, where $b > a$, Case 3 says to compare $[x, b, y]$ and $[x, a, y]$ lexicographically, which holds as $b > a$. The second condition of Case 3 says to check $f(x, b, y) >_{lpo} a$ and $f(x, b, y) >_{lpo} y$. Both of these are true by Case 1.

Another occasionally useful fact about term orderings is the following.

(Fact F) If $<$ is well-founded on the set of ground terms, then R will be terminating if for ground terms s and t , if $s \Rightarrow^* t$ then $s > t$.

(Not very useful actually, as it is hard to consider all pairs of ground terms)

Proof of Fact F: Let $<$ be well-founded and $s \Rightarrow^* t$ imply $s > t$ for all terms s and t . (*). Suppose first that $s_0 \Rightarrow s_1 \Rightarrow \dots \Rightarrow s_n \dots$ is a non-terminating, ground rewrite sequence using R , then, by (*), $s_0 > s_1 > \dots > s_n > \dots$. But as $>$ is well-founded the sequence cannot continue forever. So neither can the original rewrites. This is a contradiction, so the original assumption is false.

For the general case, notice that no variables other than those in s_0 may appear in any s_i . Suppose s_0 is not ground and there is a non-terminating sequence $s_0 \Rightarrow s_1 \Rightarrow \dots \Rightarrow s_n \dots$. Consider some ground instance $s_0\theta$ of s_0 and hence of $\{s_i\}$ ($\{s_i\theta\}$). It is still the case that $s_0\theta \Rightarrow s_1\theta \Rightarrow \dots \Rightarrow$ and hence $s_0\theta > s_1\theta > \dots >$ and this sequence must terminate at some $s_k\theta$ as $<$ is well-founded. Hence $s_k\theta$ does not rewrite to $s_{k+1}\theta$. But then s_k could not rewrite to s_{k+1} either, a contradiction. So the original rewrite sequence must terminate.

ERUDIO

In 2010, MSc student Andrei Dvornik implemented ERUDIO, a tool for carrying out Knuth Bendix completion. It operates in two modes - automatic, or step-by-step. The latter is very useful for learning about the various orderings, finding critical pairs and so on. (I will arrange a lab session to show how it works.)

It has several built-in ways to order equations (namely kbo, rpo and lpo).

You can run the tool by typing erudio at the linux prompt. The first window allows you to add equations and save them, or to load previously saved equations. The next window allows you to perform ordering of the equations. Only after equations have been ordered consistently can you move on to the superposition window, which allows to find critical pairs and other useful simplification steps. (See slides 16).

A further feature is that the tool deals with equations that cannot be ordered.

Warning: There are one or two bugs in the tool, but I assure you that there are not very many (we only found one or two last year).

There are many improvements and extensions possible if anyone would like an interesting project.

