Loed files for testims intervel stuff
Use with UTIL
Alan Burne 15.6 .81 */
:- - commilect. int. 'extres:sportr'
1).
$+\cdots$
test, $\%$ Test exambles for fefer
Frobs,
'arithiodos',
aik.
].
\% Ms intervel stuff
\% General Fortras

$$
2
$$

\% Current problems
\% ond stuff from evel
\% Quick seve and restore

Alen Burrie UFdeted: 25 June 81

```
Alen Buride 19.12.79
vet/2, Fositive/1, nesetive/1, rion ries/1, non...Fos/1. nontzero/1, ecute/i, obtuse/1, nonimeflew/i,
```

finding/2, % Exported for convenience

```
```

finding/2, % Exported for convenience

```
\% Exported for convenience
```

from UTIL:TEACE
from UTIL:SETFOU
from LoNG
evel/1
evel/2
meesure/2 from notionel mecho detebese

Quentity/1
ensle/3
inclime/3
concevitu/2
slope/2
Fertition/2
*/

* MOMES */
* -.. mode

```
im(ty+);
sub\ldots...jut(ty+)%
below(t,+)g
diswoint(ty+),
overlem(t,+),
marter_flif(?,T)
    defeult.interval(?)g
fing(..int(t,r)"
    firim...int2(t, -.),
    findmint_merss(t, - - - )
    firid..simFle_int(ty-),
    merenzssumftiom_mositive(t),
int..#F%1:(t,+,-),
```



```
    311.-*re.-conteimed(t,+),
    meke_resions(t,t,-);
        EFlit(+,+%+,-);
                sFliti(+,+,*)
            cartesimri...fromuct(t,t,#,T),
                cert..Frod(ty+g+,-,%)
    firid..limite(tg+g+, - ),
            cleen!...uF(t,--),
            limits(+y+y+y+g?)
            set...brios(t,t,t,-),
                    uFgown...flife(t,+, - )
                    set..nrio(t,t%--),
orger(ty+g?, %),
less...them(t,+),
celc(t,t,?),
    breskuF,..brose(+,#,--),
como(t,T),
mono(t,T,T),
<1"scifu(t,-)
    iritervel(t, -.,--),
    collect inmtervels(ty+,m),
    Qusd(ty+g+%%)*
```

mete structures

| cintervals | hes form | i (LMerkerg Eottom, Tow, Fimerter) |
| :---: | :---: | :---: |
| Gbouriderss | hes form | b(NyMarker) |

where *

$$
\text { Bottom, Tof, } N \text { Ere riumbers }
$$

LMerter, Fimertery Merker sere one of fofenyclosedy
Ari intervel ranses between Eottom ano Top and is ofen or closed et the ends defenidis ori LMerter (for Eottom) eriof fimerier (for Tow).

A bounderu js an erio of en intervel. There are oferstions defined over these bounderies which ere then used to help defire the oferetions over intervels. Note thet the notion of a bounders does NOT imvolve ans specific end of an interval (ie Tof/Eottom) Thes

```
*/
%% 0Qe -.- merteer (tow of code)
```

/****************************************/

* Use intervel informetion - top level */
/****************************************/
\% Check thet solution is edmissible

```
vet(truegtrue)*
vet(f%lsegf=1se)
vet(A&B,AI&BI) %-- vet(AgAI); vet(B,BI);
Vet(A#BgAI#BI) ;-- vet(AgA1), vet(BgBI).
    C(A=B,A=E) #--
        firmo..int(A,IntA), fing_..int(E,IntE),
        overleF(IntA,IntE),
        !
vet(A=B,f%lse)
```

    \% \(\times\) is Fositive, riesetive, zouteg ete,
    



$r$ -
noni... sero(x) -
find...int (X, i (L, E, Ty Fi))

!
soute(X) :-
finci...int (X, i(L. E, T, Fi))
1ess...then(b(0,ofen), h(E,L)),
1ess...than(b(T,f) gh(90,open)).
ontuse(X) :-
fingi...nt (X, i (L, E, TyF)) ,

less..then(b(T, Fi) $\operatorname{th}(180,0 \mathrm{Fem})$ ).
ronn reflex (x) :-
firionint (X, i (L, E, T, Fi))
1ess ..then(b(O,ofen) $b(E, L))$,


```
/必****************************************/
/* Menimulatins Intervelse */
/*****************************************/
```

```
% Comoine a list of intervels os sweepirs list mad
% zcoumuletins the comoined intervels.
```

```
semi.nombime([FjretIntifestInts],Fesult)
    *-. ser...combirue(festInts,FirstInt,fesult),
```



```
#en\ldotscombine([Imt,FiestTmts],Acc,Fiesult)
    *-- combine(Int, Aco,NewAco)"
        seri...combirie(FiestImts,NewAcogFesult) *
```

    \% Combine \(x\) and 4 intervels
    


\% Number N is conteined in intervel

subnint(i(closedyN,N,closed),i(L,EyT,R)),
$\%$ * intervel is contaimed in second iritervei



\% x intervel is wholls below y iritervel


$\%$ \% and s intervels zre disioint
disioint (IntX, IntY) *-- below(IntXeInty) !

$\%$ \% and $\%$ intervels overlzf
$\% \%$ overlaf(IntX, IntY) ;- mot diswoint(IntX,IntY)

overlaf(...'...).
merker finf(ofen, closed) :- ! marker.flif(closedgopen).

## /束**************************************/

/* X lies in closed or ofen interval */

\% Worst cese defeult for intervels
defsultanterval(i(opengnesinfinituginfinitugopen)),

```
% Lets try to do better..
```

Find int(X,Intervel)
;- fimoninte(X,Fesult), \% suerentee mode (t,-) Intervel = Fesult.

```
% Cetch veriables (shoulon't be there!)
```

```
fimo..int2(U,..)
    :- ver(U),
        !,
        error('Intervel Fackase siven varizble: %w',[U],fail).
                            % Bese ceses
                            % Numbers have foint intervals
                            % Sumbols (etoms) heve verious specizl ceses
```

find int2(X,i(closed, $X, X, c l o s e d)):-\quad$ mmber(X), !
find int2(X, Intervel) t-- etom(X), ! findsimple int(X, Intervel),
\% Speciel cese nommelisetion
\% Convert $-(-1)$ to $1 /$
find inte (X"(-1), Int) t- !,
find inte(1/X, Int).
\% Inesl with exponentizls to even fower
find int2(XN, i(L, E,T,F)) :-
ever(N), !


celc(", $[b(T x, F x), b(N, C l o s e d)], b(T, R))$,
\% Convert cosecent to sine

\% Convert secent to cosine

```
fimonint2(sec(X), Int) %- !, firomint2(1/cos(X), Int),
% Comvert cotensent to tensent
Finominto(cot(X), Int):- !, fingoint2(1/tar(X), Int),
% Gerierel cese
% Fecursivels firid intervels for arsumerte eno
% then int_ewfly to sort this out. This will use
% monotonicits of F to csloulste intervel of Term
% from zrsuments
```


firon....rt...erss (TermeFyIntList),
int
!.
\% If the senerel cese feils

fimonimt2(cos(X), i(closedy(-1) $1, \operatorname{colosed})$;- !

\% Find e list of intervels corresporidins to the
$\%$ ersuments of Term, Also return the furnotor.
find.int...arss (TermyFmgIntinst)
*-. functor(TermgFrigrits),
fimd...imt...arss(I,Arits, Term,IntList)


```
Fimdmint mres(NgMes,Term:[Int!IntFest])
    * ... arg(N,Term,Arg),
        fimol.int2(Ars,Imt),
        Nj i=N+1.
        final..jnt...erss(N1,Mex,TermgIntFest).
```

\% Find the intervel for $e$ simele stmbol
\% This involves lookins to see it we know
$\%$ znsthins speciel zbout the sembol which will
\% helf us.
\% Ad hoo fetoh for srevitu - frofer solution meariz
\% allowirs equetions between euentities and definire

Otherwise try to classitts sumbol (if it is en ansle)
\% Otherwise assume all auentities ere Fositive
$\% \quad$ (fossible extremer)
\% If there is no useful into we must use the defeult.



```
fimansimfle_int(M, i(ofem,oninfinitu,ofen)) *-
    mezsure(Q,M); Quentitu(0);
    !,
    meke_zessumptionmpositive(m):
```



```
    % Mate sho remember essumwtiom
marenessumftiom...Fositive(X) ;-. Essumed_mositive(X), !
malemessumption_mositive(x)
    #-- assert( assumec..wositive(X) ),
        trace('I Escume %t Fositive, \m', [X],1),
```



``` /来 Fimg intervel of furiction from intervels of its ersumerits */
```



```
% SimFle cese
```

int awn Is(F, FesiongIrt) -
morio(F, Is, Morio),

!
find...j.mjts(F, FesiongMonon Int).
\% Complex Cese
int amplu(F, FesiongTrt) *-
mono (Fy MFesion, Mono)
make resesions (Fiesion, MFesion, Newfesione);

!
seri...combine(Intervelset, Int)
\% int...effly ell intervels in eset (1ist)




\% All the ersument intervels ere sun irtervels of
\% the corresponoins monotomic intervels for the
$\%$ furiction (from morio), (ie meflist subint down
$\%$ the two "ersumenta lists),

```
#11...巴re_wonteined([],[])
```



```
    *-. sub...int(ArsIrt,FInt)*
        #1|...erencortejred(Arsfest,FFest),
```

\% Given the 1 ist of ectuel intervele erio the inst
$\%$ of monotonic intervals for the furiction build
$\%$ " set of similar intervel lists, derived from the
$\%$ ectuel intervel list, but such thet esch element
$\%$ of esch list in the set is wholly irisioe or outside
\% its corresforidins monotomic fumetion intervel.
This emoumts to case splittins the sctuel intervel
list into s set of iritervels for more treotoble
(sub) resions in the nil sface.
Implemerited bs sflittime lists to form a list of
sets and tatins the mill ertesian wroduct. Note
\% thet both sflit/4 ario eartesiani..Froduct/4 ferform
$\%$ order reversels - which emmel esch other out.
make resioms (Fiesion, MFesion, Newtesions)

cartesient...fronuct (Listofsets,[].Newfesions, []),

| \% Given the list of ectuel intervels erio the list ot |  |
| :---: | :---: |
| \% | morotomic intervels for the fumetions we muilo |
| \% | \% list of $n$ sets, where H is the erits of the |
| \% | function (ie the lensth of the lists) and where |
| $\%$ | each set conteirs iritervels which are wholly iriside |
| $\%$ | or outside the corresporidims monotoric furiotion |
| \% | iritervels, such thet the intervels iriemeh set |
| $\%$ | would combire to form the correspondins eotuel |
| \% | interval |
| \% | The combinims properts follows from the wss we split |
| \% | uF the sctuel interyels. |
| \% | The sets froduced at the momert will onls ever have |
| \% | number of members m such thet; $1=4$ m $=3$, |
| \% | The followins speoiel represertetions sre used for |
| \% | these cases; |
| \% | sirssetom(A) |
| \% | Feir (A, E) |
| \% | trifle(A)E, C ) |
| \% | In fect the code will currentle rever froduce sets |
| \% | of 3 elemerits (trifles) but I (Lewrerice) thirk |
| $\%$ | this is frobebls a bus so heve left the options mno |
| \% | this comment, eroumo til we cee. |
| \% | Note thet the list of sets built will be iri reverse |
| $\%$ | order compered with the "arsument" lists. This is |
| \% | is implementeg bs en extre ecoumuletor ersument |
| \% | (should be [] to stert with) onto which eech set |
|  | is fushed |


swlit ([ArsImtiArsfest]g[FIntiFFest]:SofergFesult)
*-- sflitl (ArsIrtoflrity Set)
\% Inte wholls withirn Irit

```
swliti(Tntesmnt,sinsleton(Ints)) *-
sumonnt(Intx, lnt),
!.
\% Inte arid Int overlep with Irto leftmost
```



``` merker...flif(Figfi), merker_flif(L, (LI),
```



``` correct(E,E1), 1ess..then(b(Tx,F以), b(T,F1)), rot less...them(b(Tx, F世) \(b(E, L))\),
```



```
% Giveri a list of n sets froduce the e set of the
% elemerits from the rill certesier: Froduct of the sets.
% The incomins sets zre represented with crecind
% furictors zs there zre onlu e few sfeciel ceses (see
% sflit): The resultins Froduct set is represented es
% e list. Eech element will itself be e list (of m
% iritervels) where the oroer of this element list will
% be the reverse of the order in which the items
% were foumo in the orisimel list of sets.
% The immlemeritation involves en ecoumulator for the
% (Fertiel) element beims built ano uses the
% differerce list techrigue to builo the firmel set
% of elemerts (refri sc a list),
```

©ertesiari... Foduct ([]:E1ement. [Element:Z],Z),
certesien...Froduct ([Firstifest],FertielElement,FroductSetgZ)
*- cert...Frod(First,Fest,FertialElemerit,Froductset,Z)
cart...Frod(sinsleton(A), fest,FertielElement,FSet, Z)


 certesiani...Froduct (fest, [BiFartizlElement],FSeti,Z);

 certesien...Froduct (fest, [B;FertielElement],FSeti,FSetz), certesieni. Froduct (fest, [CiFertizlElement],FSetz, Z),

## \% Caloulete Eottom ario Tof of Iritervel

find Iimits (F"fiesiongMonog Tnt) ;-
1jmits(bottompryfesionyMonoyb(E,L))



## \％Hack to cleer uF verious furnies




```
cleer,um(i(L, E,O,R); i(L,G,W(O),R)) %- !,
clegri..um(Int, Int);
correct(0, -(0)) %-.. !.
correct(B,B) :-. !.
```

    \% Celoulate limit for e Ferticuler bouriory
    I imits (TofBot,F, Fesiomy Morio, Eouridery)

celc(F, BouridersList, Bourioers).
\% Form e bounders 1 is . from an iritervel list
$\%$ siven various deteils - uFtrown $x$ tof+bottom:
set...brios ([]. $\ldots,[],[])$

\$-- uF downi..tlif (Tof Eot, Mono. Newmono),
set...brio(Newmoriog Irit. Erio) ,
set...bride (MFest, TofBot, IFiest, BFest) *
uFGowni..tlif(toF UII, UTi) *
uFgowninflif(bottom, uF, down) :- !
uFdowri..flif(bottom, dowrigm)
set...brid(uF, i(L,E,T,F), b(T,F)),


## ／＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊／

／米 Marimuletine Bourioaries＊／
／＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊⿻＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊／
\％Fut boumoeries ir：order
order（Brid．Brid．Briog．Brio）$\%-\quad$ ．
\％Boumieries ere identicel
\％Orie of Mis 1 e closed

\％Numbers ere differemt，N1 Emellest．

evol（N1 亿 N2），！

$$
\% N 2 i \leq s m e l l e s t
$$

```
orger(b(N1,mL)gb(N2,m2),g(N2gma)gb(N1,M1)),
```

```
% Oroerims of boumozries
% (sssumes iritervals are consecutive)
```

```
less tham(b(X,Mx)gb(Y,M&)) #--
    comb([Mx,Ms],M),
    less..tmeri._evel(MoXyY),
Iess then, eval(operigX,Y) :- evel( X = % Y ),
less..thenmevel(closedyX,Y) %- evel( X < Y ),
```

```
% AFFly Fumotion F to s boumriery list
% llo this bs combinires the bounoers merkers eno
% efFlyiris F to the rumbers.
```


breakuF hrids (BourdersList, Markers. Numbers)
comb(markers, M)
Term =: [FINumbers],
evel(Term; X ),
!
breakum, מrios ([.],[].].])

*-.. breatum hrios (fest, MFest, NFest).
\% Combine bouriders merkers
$\%$ Fesult $=$ oferi if aris of the iriputs is ofer
comb(markerlist, fesult) *- memberchk (oferimerterlist), ! Fesult = ofert.
comin(.nclosed).
mono(-, fi(closeg, mesinfinituginfinitu,closed)]. [oown]),
/* zodition */
mono(tyicolosed, mesinfinitusinfimjts, closed)
i (closed, nesinfimitusinfinitugolosed)]. [uF,uF])
/* binery mirus */
mono(-
i (closedynesinfinits,imfinitugclosed)], [uFgown]) *

```
/束 absolute value */
moro(ehs,[i(c]osec,mesimfimitug-(0),closed)], [dowri]).
mono(zbs,[id(closed,0,infimity,closed)], [um]).
* muldti&licetion */
momo(*)[i(closedgogimfinitugclosed), i(closed,0,imfinitugclosed)],
    [u*,u*]).
```



```
    [\mp@code{OWmiguF]).}
```



```
    [.,_F, dowm]) *
```



```
    [.dowm, down]).
```

/束 division */
mono(/,
[. 1 \% gown])

[. downy downil).

[uF, !f:])



```
** exFonentiztion */
```



```
        [.|F,uF]).
```



```
        [cown, uF].] *
```

* loserithm */
mono(losg[i(closedgoginfinits,closed) i(closedyogimfinitugclosed)]g
[downymF]).
/* sime */

..ono(singli(closedi90,270,closed)],[down]),

/* cosine */


/* tensent */
morio(tang [i(oferig (-90),90, ofen)],[um]).


/米 imverse sime */
morio(ercsirig[i(closed, (-1), 1, closed)], [uF])
/* inverse cosine */

/* inverse tensent */


```
* irverse cosecant */
mono(arccsc,[i(closedymesintimits,(-1), closed)],[down]),
mono(erccso,[i(closed,\,imfinitu,c]osed)],[down]),
/* inverse secerit */
mono(arcsec,\mp@code{i(closed,mesinfinitus(-1), closeg)],[uF]),}
moro(srcsec,[i.(closed,l,infirijts,closed)],[uF]),
/* imverse coteriserit */
mono(erccot,[i(closed,mesimfinits, -(0),ofen)],[cown]),
```



```
/*************************************************/
/* Celoulate Intervel of Armle from Curve Twfe */
*************************************************/
```


Cleseifu(Ansley Int) *-
messure(0. Ansle) , \% detebese
arsle(Foimt, Q. Curve) $\quad$ ! \% detebese
intervel (ansleg Curveg Int )
clessifs(Ansles Int ) :-
meesure(0, Amsle) \% \% סetzbese
incline(Gurveg Qg Foint ) ! \% \% detebese
interval(inclineg Curveg Int ).
\% Find intervel from curve shefe
\% For simble curves
intervel(AT, Curveg Irit) F-
concevits(Qurve, Conv): $\%$ detebese
slofe(Curve, Slofe ), ! \% detebese
Quad(AI. Slofe, Goriv. Int ).
\% For comblex curves
intervel(AI. Curveg Int ) :-
Fertition(Curve, Clist), ! \% detebese
collect. intervels(Clist, AI, FIist):
Eeni..combirie(Filist, Irt ).

```
collect...intervols([],...[]).
collect.intervels([FirstiFest],AI,[FirstIntifestInt])
    * -. interval(AI,First,FirstInt),
        collect...irtervels(fest,AT, FestInt).
```

\% Informetion about frowerties of $s i m p l e$ eurues
$\%$ The intervel defends on both the slofe enim the
$\%$ concevits.
ausd(enslegleft, rishtgi(closed.0,90.closed)) *- !


qued(imclimegrisht, risht.i(closed.180.270.closed)) (- !



eusd(inclimegrishtylefty $(c l o s e d, 0,90,010 \leq e d)$ ) ;- !




Qued(ansleghorystlinegi(closed.270,270.closed)) ;- !
Qued(iriclineghorgstlinegi(closedrogonclosed)) *- !


$\cdot$ JOBS TO LO
write ssmbolic version for findins mex/mins
use monotonicjts in $\gamma=$ eto Isoletion rules
*/
/* Frobs.
CUFFENT FROELEMS*/
/* interval ano eval froblems */

Fbh *- acute(x*2).
monnzero(a)

Fbo *-- nori... zero( $\left.\left.73^{\cdots}(1 / 2)+9\right) *(1 / 2)\right)$.

```
* TEST.
est Exemwles for Fefer
1].0n Bumos 15.6.81 */
* Test rum with timinse */
```



```
                            tests,test97).
* Tests */
```



```
,est2(T) ;-. int.##Fls(sim, [i(ofeng 30.150,closed)],I), % I = [1/2,1]
(estZ(T) %-. int.epfls( zbs, [i(ofen:(-1),1,closed)], I), % I = [0.1]
,\mp@code{t4(I) %-" %/4 I = [-oo,-2/3)}
    int..жF%ls(% [i(ofer, 2,3,closed), i(closed,(-3),-(0),olosed)], I),
:estw(I) %- % I = [-3,3]
```



```
est6(I) %-w I = [1/2.2]
```



```
Est7(I) %- firmanimt( (-mI)*E/(m1+m2), I), % I = (-00,0)
```





```
** In Froblem with stetistics*/
#tzts(Nzme) *-- Froblem= + [NEme,Ars], stetistics(rumtimeg..),
                        cell(Froblem), !, Etatistics(rumtimeg[ .., Time]),
                        trace('\r%t took %t milliseconds end froduced enswer %t\rivi'g
                            [Name,TimegArs], O).
#tats(Name):- statistics(runtime,[ -." Time]),
    trece('\rsorrs I coulod not Frove %t erio I sperit %t rot doins it \ri\r'*
        [\mp@code{Nme, Time]! O).}
```

```
105
    T-- ruri.
.estj took g milliseconds amo froduced ariswer i(oferig (1/2), 1, closed)
est2 took 6% mil1iseconis and froduced answer i(closed, (1/2), 1, closed)
estz took 38 milliseconds arid froduced emswer i(closed, 0, 1. closed)
,est4 took 2, milljsecorids erod froduced eriswer i(olosed, riesimfiritty, (-2/3), oferi.
sest; took 104 milliseconis eno froduced enswer i(closed, -3, 3, closed)
est6 took 57 milliseconds erid Froduced erswer i(closed, (1/2), 2, closed)
est7 took 59 milliseconds end eroduced enswer i(ofen, mesinitinity, -. o, ofer,)
:esto tool 123 milliseconds mnd fromuced snswer i(ofeng mesinfinitug intinitug arei
estg took 36 mil]iseconds smo wroduced enswer i(oferig negiritinity, - o, ofen,
```

```
16:
```

16:
T-- core 68096 (38912 10-5es + 29184 ni-ses)
T-- core 68096 (38912 10-5es + 29184 ni-ses)
7es% 33792= 31227 in use t 2565 free
7es% 33792= 31227 in use t 2565 free
slobel. . 11%% = 16 in use + 1159 free
slobel. . 11%% = 16 in use + 1159 free
locel 1024 = 16 in use t 1008 free
locel 1024 = 16 in use t 1008 free
\thereforereil Sjl = 0 in use + 5ij free
\thereforereil Sjl = 0 in use + 5ij free
0.05 se%, for 2 GCs seinins 1213 worde
0.05 se%, for 2 GCs seinins 1213 worde
0,12 sec, for 20 locel shifts smo 2l trail shifts
0,12 sec, for 20 locel shifts smo 2l trail shifts
9.66 sec: rumtime

```
    9.66 sec: rumtime
```

```
yes
| T-- ruri.
testi toot 31 millisecomds erg Froduced enswer
XI
    where :
        X1 = i(owen, (1/2), 1. closed)
test2 took 145 milliseconds snd froduced enswer
X1
    where %
        X1 = i(closec, (1/2), 1, (10sed)
```

```
test3 took 1J5 milliseconge mrg Froduced erswer
```

test3 took 1J5 milliseconge mrg Froduced erswer
X1.
X1.
where :
where :
X1 = i(closeng 0% ebs(- 1)% ofen)
X1 = i(closeng 0% ebs(- 1)% ofen)
test4 took 65 milliseconds end produced enswer
X1
where:
X1 = i(closed, resirifimits, (-2/3), ofer,
testw took 2;G milliseconds end froduced erswer
X1.
where *
X1= i(closed, - 3, 3, closed)
-t6 took 1.75 millisecomos and Froduced enswer
where *
X1 =: i(closed, (1/2), 2, closed)
test7 took 211 mil1iseconde mmo froduced answer
X1.
where :
XI =: i(ofem, resinfinitu, -- o, oferi)
teste took ze3 milliseconms ano Froduced erswer
XI
where *
XI == i(ofeng mesinfimits, imfinitug ofen)

```
```

test9 took 102 millisecorids and froduced enswer
XI
where ;
Xl = i (oweng nesimfinits, - O, ofen)

```


\section*{UNIVERSITY OF EIINEURGH}


Subject: A Generalized Interval Fackese and its Use for Semantic Checking Author: Alan Ends

Abstract

We describe an interval arithmetic peckese, INT, which generalises previous interval packeses bu using information about the monotonicity of functions. INT rms been used in en elsebreic manipulation feckeseg fess to check the editions of rewrite rules end to vet the solutions to eouetions.

\section*{1. Introduction}

Ir i this moper we describe a semeral interval fackese; that is g e computer Frosrem, celled INT, in which arithmetic functions ere extended so that they baffles not just to numbers, but to intervals of the reel line, If \(f\) is an misery function then it cen be extended to intervals with the definition
so that
\[
[1.2]+[3,5]=[4,7]
\]
where \([\) gb] \(=\{x ;\) es \(x \quad b\}\) is the closed interval from to \(b\). Such feckeses are in common use for providing suerenteed error bounds in arithmetic ese e es. [Good \& London 70, Yonne 79]).

INT was built for a different purpose: namely, for checking the conditions of rewrite rules and vetting the solutions to enuetions, in en elsebreic manipulation frosrem, FRESS, [burns and welham 81]. For instance, the rewrite rule
\[
凸 * V \equiv W \quad=\quad 4 \geq W / V
\]

hes the coridition thet \(v\) be fositive, ise lie in the oper intervel (o.oo), other turicel comoitions are that a variable be non zerog zn acute ansleg etc.
 еre known to be an acceleretions mess, mess end eccelerstion due to srevits, respeotivelu) then we went to be eble to vet, zno rejeot, the equetion
\[
==-m \omega * s /(m\rfloor+m 2)
\]
:s F Foseible solution for \(=\) in terms of min m2 and e, since it woulo implu thet a were mesetive.

The JNT Feckase differs from Frevious feckeses in the followins respectes
-- INT cen deel with intervels whose bounderies include the infinite rumbers, 士oo.
-- INT can deal with intervals with ofer or closed bounderies. These are denoted bs rourid end souere breckets. resfectively, e.s. \([0,5)=\{x ; 0 \leq x<5\}\)
-- INT can deal with ens functiong frovided onls thet informetion is Frovided ebout where thet function is monotonicelly incressins end decreasine, Frevious Feckeses have defined interuel arithmetic for onls e few sfeoific fumctions, ess, [Good g Lomoon 70\(]\) sives

 inverse trisomometric fumctions: ario sbsolute velues
... INT cen deel with intervels which stradole severel monotonio resions of efuriction \(\quad\) In Ferticulerg it cen deal with i/j when i contziris o.
-. INT cen use informetion ebout z Fertiouler constemt, winch sfecifies in whet intervel it lies. This hes been used in conwumtion with \(e\) Frosram for solvims mechamics Froblems. [Euriset al 79]. to ellow semantic imformetion about Fhssicel euentities, ess, thet mis Fositiveg \(\phi\) is ontuse, etc, to influerice the zleebreic menifuletion:

\section*{2. Monotonicits and Unary Functions}

The kes ides benino INT is to use the monotoricits of e furiction to decice which boumozries of its ersuments to use to celoulete ite uffer end lower bounds. For instence, sin is monotomicells increasins on the intervel © 30.907 , so the lower bourid of sin (30, 907 should be celouleted from ( 30 shd the uffer
 Note thet the tuee of the bounders (open or closed) is inherited elons with its velue. We will ses thet sin is maffed across the intervel (30.90] to froduce

the intervel (1/2,17.

Now comsider the sewlicetion of sin to sq0.jol. ein is momotonicelly decressims on this intervel, so the lower bourid of eiri (90. 50\(]\) Ehoulo be celoulated from 1507 and the uffer boumi from so, giving the intervel \([1 / 2,1)\), Note ssein thet both the tume ena velue of the intervel vie inherited. We can summerise this bs sesins that the intervel (90. 150\(]\) is inverted to the fseudo-interval [150; 90 ) * \(*\) siris meffed ecross it. to froduce Lsin \(150,5 i n 90\) ) and the bourieries are evelueted to froduce \([1 / 2,1\) ),
sin is simme monotomic on both the intervels (30.90] ano (90.150]. To celculate the velue of sin on an interval, on which it is rot simply monotonic. is more comFliceted. Consider the intervel (30.150]. To celoulete sin ( 30.150 g INT first divioes the irterval into sub-intervelsy om eech of which sin is simbls monotonic. In this cese (30. 50 ] would be divided into (30,90] and (90.150]. sin is then afflied separetels to each sub-interval. to Froduce (1/2,1] and [1/2,1), erig these results are then combined into the intervel [1/2,1].

\section*{3. Generalized Monotonicity and Non-Unary Functions}

To deal with nommuers fumotions we heve to senerelize the motion of monotonicits to a tuple velued furiction. This is beceuse a rion-uriery furiotion mas have different moriotoricits behaviour on different ersumerite, For instemce, biners minus is monotonicells increesins on its first ersument end monotomically decreseins on its second. Thet is x-y incresees es x iriereeses. but decreases \(e s\) increeses. we refresent this bs sesins thet - hes

 monotomicits sdowns on resion < [90.270]\%.

We cen formalize this motetion as follows;
nefinition 1: An nill resion is \(\quad\) n moturle of intervels. If every intervel is \([-\infty 0\) too] the the resion is celled the whole nill space.

Inefinition 2: Ari nugrs functiong fo is monontonically increasins on

\(1 \quad n_{1}\)

米 \(1.50,90\) ) would be the empts intervel by the rormel defirition. Howevery the Fhrese 'Feeurowintervel' is mesnt to imply thet we will rot reserd it es e Frofer intervel" but merelu zs a surtectic gevice.


Mefinition 3: A functiong f; is monontonicalls decreasins on its jth arsument in resion ii \(, \ldots, i \quad i \quad i f f\)
```

                        |
    ```


Mefinition 4: Ari n- ers fumction is simfls monotomic in resion r iff it is monotomicells incressins or monotonicelly decreesins on eech of its arsmments in resion \(r\), Its monotonicits is siven bs an n-turle in which the ith element is 'uf' or 'oown ecoordins as the function is monotonicells incressins or decressins on the ith ersument int \(r\).

Armed with this notationg we cen row tackle the froblem of exteridins intervel srithmetic to mon-mners functions, Consider the apelicetion of -. to the
 ersument erio momotomicells decreesins on its second we shoulo celculete the lower hourio of [3, 5\(]-[1,2)\) from \([3\) ari 2 ) to sield (i, Similerly, the upfer boumd should be celouleted from 5\(]\) sho [1 to sield 4\(]\) This cen be summerised bs sesins thet [1,2) is inverted to the fseudo-intervel (2, 1\(]\) aro - is theri maffed ecross the pseuromresion \([3,5], 2,1]\) to froduce (3-2. 5-1]. The rule is thet the ith intervel is inverted iff the furiction is monotomicelle decressins on the jth arsument, Note thet the tuFe of s bouriory is ofen umless ell of the bounderies it is celculated from ere closeds sirice efumction cen onls ettein e houmbers iff all its ersuments do.

Both + and \(-\cdots\) re simpls monotorice throushout the whole 2 m sfece, * ario / howevery are rot simplu morotoric throushout the segee thes heve four resions of simple monotomicitu, For instences / hes the monotonicities siven in fieure
```

4F,uF% in
<Cown,mFy iri

```
UF: 万owns ir:
\([-\infty 0,-0],[+0,+\infty 0] \% \quad[+0,+00],[+0,+\infty 0]\)
down, downs in
< \([+0,+\infty 0],[-\infty,-0]\).

Fisure 3-1: The Monotoricities of /
 the seferete apwlicetions of \(/\) to \([2,3],[-1,-0) \%\) aro \([2,3],[+0,1] \%\) ir, which resioms / is simpls monotonic, Celouletins these efolicetions from the monotoricits information siven ebove sields the followins frocess,
```

To celoulete ( 2FFlied to [[2,3], [-1, -0)%
jmvert both ersmments and mef / zoross them to froduce
[3/-0.2/-1]
Eveluate the boumozries to froduce
[-00%-2]
To awFly ( to <1.,3], 1.+0,1]%
invert the second ersument end mef / zerose the result to froduce
[2/1,3/+0]
Evelumte the bounderies to froduce
[2%+00]

```

The two resultims jntervels are then combined into mew intervel bhose umper bound is their maximum uffer bourid and whose lower boumg is their minimum lower boumg This sielos the intervel [-oogtoo]. Note thet this intervel is too Fermissive, in the semse thet it includes the intervel ( -2.2 ), which should be excluded.

\section*{4. The Alsorithm}

After the informel introduction of the fremeedins sections we row turrito \(\%\) formel sccount of the semersl, intervel-erithmetic elsorithmy which ue uil cell Int-AFFls,




```

        ith sremmerit.
        Mef f eqross r', eveluete the bourigeries of the result erig
        returmit,
    (b) Tf f í: riot sjm; = moriotonio ori r theri
SFlit r jrto \# muburesion ori which f is simFlu monotomio
H
\#пи the z-1 comflementare subwresiors (some of which mzu
boemFty)
Cal| the fromedure reoursivelu on emoh of the 2
suh-resioris which sre rot emftu,
Gombjre the resultims intervels irito ore iritervel.

```

The above descriftion leaves verious sub-mpocedures undefined. nemely the wrocesses of 'maprins ecross', 'invertins', 'eveluetins the bounderies', selittins e resion into sub-resions and conbinins several resions into one. we now proceed to define these processes.
-- Intervals. Intervels are refresented as gusdrufles, in which the first and last elements ere either 'open' or 'closed', to represent the tuse of bounders, and the second and third elements are mumbers representins the value of the bounderies, ess, \{90.150] is rewresented by open,90,150, closed.
-- Invertins. The inverse of intervel L, E,T,F\% is \&R,T,E,L>.
-. Mapeins Across. The mappins of \(f\) across

i. 5
where \(c(S e t)=0\) en iff open \(e\) Set and c(Set)=closed otherwise.
-- Evaluation As can be seen from frevious seotions the normel arithmetic fumctions must be exterimed to deel with the infinite rumbers -oo ano too. It is also necessers to distinsuish betweer -0 and +0 , since \(3 /-0=-00\) ano \(3 /+0=+00\) For eveluetins the boumperies of intervels INT uses en Erbitreremprecisioris retionel-rumberg erithmetic Fectese developed bu ficherd okeefe. Ir this Feckase rumbers ere refresented bs trifles of

 oferetions recuire orly trivizl soaftetion to return the correct answers for infinite mumbers. In inoeterminete ceses, e, s, 0/0, the answer 'umdefined' is returned. Such an enswer ceuses Int-Amely to return the defeult iritervel (-ooytoo).
 so throush the simply monotonic resions of f uritil one umis * ming is founc with the froferts thet each I.j is a disuoint union of intervals I.j" zho Ij", where Ij" is monmemfts, shd e sub-intervel of r


 cemrot be divided into \(\boldsymbol{f}\) inite set of simply momotomic sub-resions then this splittins frocess mes mot terminete.
-- Combiniris. To combine eset of intervels, forme rew intervel whose lower boumd is the minimum of the lower bourids of the set erio whose uffer boung is the maximum of the set, Note thet the minimum of two bounderies with the same velueg but different tufes, is the closed boumoers. Similerly with the msximum. The comoimetion of two intervals is the smallest intervel contaimins their union but it mes not be equal to their uriong ess combinirs \([-\infty 0,2]\) end \([2,+00]\) Froduces [--oogtoo].

\section*{5. Semantic Checkins}
he sbove sections describe an irtervel erithmetic elsorithm we now ectimiri the efflicetions of this elsorithm in the alsebreic menifuletion feckeseg FFESS, [Eunds and Welham 81].

As describe in the introduction, the sfflicetions zre twofolds chectims the comidtions of rewrite rules eno vettins the solutions to equetions. Eoth these EFFlications mate use of a common sub-frocedurey Find-Int. Fino-Int tekes en slébreic term erio returns the intervel within whioh it lies, For iristarices if
 …mi*e/(mltm2) will return (-oogo) from which it cen be deduced thet the term is resetive and thet \(e=-m 1 * s /(m i+m 2)\) is felse,

The Frocedure, Find-Irt, worke bu cell bs velue, Afrlied to orm torm
```

f(t,...gt ), Find-lnt is celled recursively on eech t gno returns i, *i is

```
 1 п
Th then the interval [ngri is returned, e.s. [2.2], If Find-Int is apficed to e ssmbolic constent or verieble, e, s, mi, then sementic informetion is used to trs to determine the result. If this is not successful then the defeult interval, (-oo, +oo), is returned.

The sementic informetion is frovided by the mECHO prosrem, [Bunds et el 79], for solvins Mechanics problems. Informetion is frovided ebout two sorts of constent: fhssicel ouantities (e.s, messes, ecceleretions, etc) and ansles. All Fhssicel Quantities are essumed to be fositive, thet is, to lie in the intervel (O,too), Frovision exists for frovidins more sophisticeted information about esch king of euantits, but this hes not been exploited. In the cese of ensles en ettemft is made to infer in which ousorant(s) of the cirele the ensle lies, for instance, \([0,90]\), [180, 360\(]\), etc.
nsles are defined in MECHO either as the inclinetion or the normel to a 2 dimensional curve, where the nommal is towards the conver side of the curve and the inclination is 90 desrees sreeter. For simple curves, i, monotonic curves with monotonic first derivatives, both the inclination and the normal lie wholls within e sinsle auedrant, Which euedrent this is depends on the sisn of the first and second derivatives. We cell the first derivative the slope and the second derivative the concavity of the curve. For instance, if the slofe is Fositive and the concevits is fositive then the inclination lies in the eusorent [0,90] end the normel lies in the ousorent [270.360]. This and the other seven ceses are illustreted in fisure 5-1,

If Find-Int is efwlied to an ansle defined on a nonsimple curve then the curve is first broken into simple curves, the ebove frocess is efrlied to esch of these and the resultins Quedrants are combined, usins the combinetion Frocedure described in section 4,

\section*{6. Fiesults}

The INT intervel erithmetic feckese hes been imelemented in FRoLog cfereire et al 797 on \(e\) necio. It occupies \(39 \mathrm{k}, 36\) bit words and the FROLOE eystem occufies \(e\) further 2gk. The former fisure could probebly be reduced substantially by deletins various utility frocedures reauired by fRESS but not by INT, Table 6-i summerises the results of apflyins Int-Affly to some tupicel functions and resions. Teble \(6-2\) summerises the results of efflyins find-Int to some turicel formulae.

\section*{7. Limitations}

The INT Feckese uses informetion about the simply monotonie resions of \(z\) function to extend its definition on mubers to one on intervels. In this wey
normel [270,360] mormel [90,180] normel (270.360) incline \([0,90]\) iriclirie \([180.270]\) iriclirie (0.90)
\begin{tabular}{|c|c|c|}
\hline normel [180,270] & normel [0.90] & rornel (180.270) \\
\hline irncline [270,360] & incline [90,180] & incline (270.360) \\
\hline \multirow[t]{2}{*}{imposeible} & immostiole & \\
\hline & & \[
\begin{aligned}
& \text { normel=270 } \\
& \text { incline=0 }
\end{aligned}
\] \\
\hline immossible & impossible & \\
\hline & & \[
\begin{aligned}
& \text { norme1 }=180 \\
& \text { incline }=270
\end{aligned}
\] \\
\hline
\end{tabular}

Fisure 5-1; C1essificetion of the Aneles of simple Curves

\section*{Interval \\ Calculation}
\(\sin (30,150]\)
\(8 \mathrm{bs}(-1,1]\)
(2, 3] \(14-3,-0]\)
\([-3, \cdots)\) ( \(-[-1,1]\)
\([-1.1]\)
(1.2.

Ariswer
(1/2.1.]
\([1 / 2,1]\)
\([0.1]\)
38

18

120

58
64

CFU/Time Is Fiesion Simply in msecs Monotonic?
\(13 \quad\) yes
no
no
yes no 10

Tahle 6-1: Fiesults of AFFluins Int-AFFly
\begin{tabular}{|c|c|c|}
\hline Formula & \begin{tabular}{l}
Interval \\
it lies in
\end{tabular} & CFU Time in msecs \\
\hline \(\cdots \mathrm{m} .1\) 米 \(\mathrm{s} /(\mathrm{m} 1 .+\mathrm{m} 2)\) & (-00, -0) & 63 \\
\hline \(\sin (\theta)+2 / \cos (\omega)\) & (-00.00) & 128 \\
\hline \[
\operatorname{los}_{2} \sin (\theta)
\] & (-00:-0) & 37 \\
\hline
\end{tabular}

Where mig m2 ard o are fositive mumbers, \(\theta\) is ari acute arele arid of is an obtuse ansle,

Table 6-2: Fesults of AFFluing Fimg-Int
it sereralises frevious intervel peckeses, which could deel with orise
-specified number of functions, However, INT onls works well or: fumetions which are well beheved monotonicells, i, e, divide the whole space into g firite set of contimums simply monotonic resions, some of the situetions in which InT joes not beheve ineelly ere listed below,
- INT ©ennot deel with the situetion where e furiction must be spelieg to a resion which onls divides into en infinite set of simplut momotonic sub-resions, beceuse Int-AFels will mot terminetes for instance, sin awelied to (-ootoo) s should return [-1.1], hist ( - ootoo) cerinot be divided into efinite set of simplsmonotonic suburesions, so lnt-Affly will mot terminete this ferticuler cese is deelt with hy a Fetch to the INT Feckese to teke irito ecoourit the boumbedriess of siris cos. ete.
… Trit-AFFly is constrained to return a simsle continuous interyel es its result To returris ses, eset of disioint intervels would elter the whole besis of the alsorithm with e consequent loss of simplioits ario efficiemcs, The Frice to be Feio for this is thet the result of Int-AFFlu will sometimes be too inclusive. Ari exemple of this wes siven in section 3 where the result of \([2,3] /[-1,1]\) inclured the interval (-2, 2), 米 Sometimes the best descriftion of the monotonicite
* B Ceflet, iri erivete commuricetiorig hes sussester thet pllowirs 'extervels' to be returned bs Int-AFFls, would accourit for the mejority of zounterexemples which erise in Frectice, for only e smell imeresee in the zomplexits of the slsorithm. An extervel is eset of real mumbers obteiried bu subtrectins en intervel from the reel line, \(I\) neve rot explored this -ossibilitu.
```

of e fumction involves monmcontinuous sets of mumbers. For instence,
y masotonicite wowr,ur` when x is nesative and is an even

* hes monotonicity dowmpufs when x is nesetive end y is en even
rationel. Unfortumetely, the even rationels canmot be refresented as
an intervel, In fect exponentiztion hes no simply monotonic
sub-resions in <[-0o:0],[-oo,+oo]>. There is e fetch ir, INT to deel
with this perticular cese, but in senerel such situetions are outside
i.ts scofe.

```

\section*{3. Conclusion}

We have seen thet the INT interval arithmetic fackese senerelises frevious interval fackases bs the use of the monotonicits of functions. It cen deal with more fumctions than frevious Feckeses and cen deel correctly with the afflicetion of a furmetion to e resion which stradoles e finite muber of simply monotonic resions. It breaks down only when \(e\) furiction hes e ferticulerly comflex monotonic beheviour.
"he INT Fackase hes been afflied in unusual wass, to check the conditions of rewrite rules and to vet the solutions to equations, es pert of an elsebreic menifulation fackese,

\section*{3eferences}

Wunds and Welham 81]
Bunds, \(A\), and Welhem, \(B\),
Usins mete-level inference for selective zfflicetion of multifle rewrite rules in alsebraic manifuletion,
Artificial Iatelliseace 16(2), 1981.
[Burios et al 79]

Felmer, M.
Solvins Mechanics Froblems Usins Meta-Level Inference.
In Enocs of the sixth. IJCAI, Tokso, 1979,
Also available from Edinbursh es llal Reseerch Fefer No, i. 12,
[Good \& London 70]
Good, II. I. and London, R.L.
Computer intervel erithmetict definition and froof of correct
imelementetion.
Jecm 17(4):603-612, 1970.
[.Fereire et el 79]

Usex's suide to DECsystem-10 EEOLOG.
Occesionel Faper 15, Iept, of Artificiel Intellisence, Edinbursh, 1979.

Yohe, J.m.
Softwere for intervel erithmetic: A ressoneoly forteble peckese. AC苗 Irans, Math. Software 5(1):pp50-63. Merch. 1979.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

\section*{UNIVERSITY OF EDINBURGH}

DAI Working Paper No: 86
27 March 1981
Subject: A Generalized Interval Package and its Use for Semantic Checking Author: Alan Bund

\section*{Abstract}

We describe an interval arithmetic package, INT, which generalises previous interval packages by using information about the monotonicity of functions. INT has been used in an algebraic manipulation package, PRESS, to check the conditions of rewrite rules and to vet the solutions to equations.

\section*{1. Introduction}

In this paper we describe a general interval package; that is, a computer program, called INT, in which arithmetic functions are extended so that they apply, not just to numbers, but to intervals of the real line. If \(f\) is an \(n\)-ary function then it can be extended to intervals with the definition
\[
f\left(i_{1}, \ldots, i_{n}\right)=\left\{f\left(x_{1}, \ldots, x_{n}\right): x_{j} \in i_{j} \text { for all } j\right\}
\]
so that
\[
[1,2]+[3,5]=[4,7]
\]
where \([a, b]=\{x: a \leq x \leq b\}\) is the closed interval from \(a\) to \(b\). Such packages are in common use for providing guaranteed error bounds in arithmetic (see [Good \& London 701). furs?

INT was built for a different purpose: namely, for checking the conditions of rewrite rules and vetting the solutions to equations, in an algebraic manipulation program, PRESS, [Bundy and Welham 81]. For instance, the rewrite rule
```

u*v \geqw => u \geqw/v

```
has the condition that \(v\) be positive, ice lie in the open interval ( \(0, \infty\) ). Other typical conditions are that a variable be non zero, an acute angle, etc. If \(a, m 1\), \(m 2\) and \(g\) are all known to be a positive quantities (egg. because they are known to be an acceleration, mass, mass and acceleration due to gravity, respectively) then we want to be able to vet, and reject, the equation
\[
a=-m 1^{*} g /(m 1+m 2)
\]
as a possible solution for \(a\) in terms of m 1 , m 2 and g , since it would imply that a were negative.

The INT package differs from previous packages in the following respects.
- INT can deal with intervals whose boundaries include the infinite numbers, \(\pm \infty\).
- INT can deal with intervals with open or closed boundaries. These are denoted by round and square brackets, respectively, e.g.
```

[0,5) = {x: 0 < x < 5}

```
- INT can deal with any function, provided only that information is provided about where that function is monotonically increasing and decreasing. Previous packages have defined interval arithmetic for only a few specific functions, e.g. [Good \& London 70] gives definitions only for,,\(+- *\) and /. Currently, the INT package can deal with +, -, *, /, exponentiation, logarithms, trigonometric and inverse trigonometric functions, and absolute value.
- INT can deal with intervals which straddle several monotonic regions of a function. In particular, it can deal with \(i / j\) when \(j\) contains 0 .
- INT can use information about a particular constant, which specifies in what interval it lies. This has been used in conjunction with a program for solving Mechanics problems, [Bundy et al 79], to allow semantic information about physical quantities, e.g. that \(m\) is positive, \(\phi\) is obtuse, etc, to influence the algebraic manipulation.

\section*{2. Monotonicity and Unary Functions}

The key idea behind INT is to use the monotonicity of a function to decide which boundaries of its arguments to use to calculate its upper and lower bounds. For instance, sin is monotonically increasing on the interval (30,90], so the lower bound of \(\sin (30,90]\) should be calculated from ( 30 and the upper bound from 90], giving the interval ( \(\sin 30\), \(\sin 90]\), which equals (1/2,1]. Note that the type of the boundary (open or closed) is inherited along with its value. We will say that sin is mapped across the interval ( 30,90 ] to produce the interval \((\sin 30, \sin 90\) ] and then the boundaries are evaluated to produce the interval \((1 / 2,1]\).

Now consider the application of \(\sin\) to \((90,150]\). \(\sin\) is monotonically decreasing on this interval, so the lower bound of \(\sin (90,150]\) should be calculated from 150] and the upper bound from (90, giving the interval [1/20w ,1). Note again that both the type and value of the intervalas inherited. We can summarise this by saying that the interval ( 90,150 ] is inverted to the pseudo-interval \([150,90) . * \sin\) is mapped across it, to produce \([\sin 150\), \(\sin (1)\) \(90)\), and the boundaries are evaluated to produce \([1 / 2,1)\).
\(\sin\) is simply monotonic on both the intervals \((30,90]\) and \((90,150]\). To calculate the value of sin on an interval, on which it is not simply monotonic, is more complicated. Consider the interval (30,150]. To calculate \(\sin (30,150]\), INT first divides the interval into sub-intervals, on each of which sin is simply monotonic. In this case \((30,150]\) would be divided into ( 30,90 ] and \((90,150]\). sin is then applied separately to each sub-interval, to produce \((1 / 2,1]\) and \([1 / 2,1)\), and these results are then combined into the interval [1/2,1].

\footnotetext{
* \([150,90\) ) would bc the empty interval by the normal definition. However, the phrase 'pseudo-interval' is meant to imply that we will not regard it as a proper interval, but merely as a syntactic device.
}

\section*{3. Generalized Monotonicity and Non-Unary Functions}

To deal with non-unary functions we have to generalize the notion of monotonicity to a tuple valued function. This is because a non-unary function may have different monotonicity behaviour on different argumentsof for instance, binary minus is monotonically increasing on its first argument and monotonically decreasing on its second. That is \(x-y\) increases as \(x\) increases, but decreases as y increases. We represent this by saying that - has monotonicity <up,down> on the region \(\langle[-\infty,+\infty]\), \([-\infty, \infty]\) (i.e. everywhere). Using the same notation, sin has monotonicity <up> on region <[-90,90]>, and monotonicity <down> on region <[90,270]>.

We can formalize this notation as follows:
Definition 1: An \(n D\) region is an n-tuple of intervals. If every interval is \([-\infty,+\infty]\) the the region is called the whole nD space.

Definition 2: An n-ary function, f , is monontonically increasing on its \(j\) th argument in region \(\left\langle i_{1}, \ldots, i_{n}\right\rangle\) iff
\[
\begin{aligned}
& \left.x_{j_{r}}<y_{j} \quad->\quad \underset{x_{k}}{ } \quad \underset{i_{k}}{i_{k}} \text { and } y_{j}, \ldots, x_{j}, \ldots, x_{n}\right) \quad<f\left(x_{1}, \ldots, y_{j}, \ldots, x_{n}\right)
\end{aligned}
\]

Definition 3: A function, \(f\), is monontonically decreasing on its \(j\) th argument in region \(\left\langle i_{1}, \ldots, i_{n}\right\rangle\) iff
\[
\begin{aligned}
& x_{j_{j}}<y_{j} x_{k}->\underset{i_{k}}{ }\left(x_{1}, \ldots, x_{j}, \ldots, x_{n}\right)>f\left(x_{1}, \ldots, y_{j}, \ldots, x_{n}\right) \\
& \text { ford } \left.y_{j}\right)
\end{aligned}
\]

Definition 4: An n-ary function is simply monotonic in region \(r\) iff it is monotonically increasing or monotonically decreasing on each of its arguments in region \(r\). Its monotonicity is given by an n-tuple in which the jth element is 'up' or 'down' according as the function is monotonically increasing or decreasing on the jth argument in \(r\).

Armed with this notation, we can now tackle the problem of extending interval arithmetic to non-unary functions. Consider the application of - to the arguments \(\langle[3,5],[1,2)\rangle\). Since - is monotonically increasing on its first
- argument and monotonjcally decreasing on its second we should calculate the lower bound of \([3,5]-[1,2\) ) from [3 and 2) to yield (1. Similarily, the upper bound should be calculated from 5] and [1 to yield 4]. This can be summarised by saying that \([1,2)\) is inverted to the pseudo-interval \((2,1]\) and - is then mapped across the pseudo-region \(\langle[3,5],(2,1]\rangle\) to produce ( \(3-2,5-1]\). The rule is that the jth interval is inverted iff the function is monotonically decreasing on the jth argument. Note that the type of a boundary is open unless all of the boundaries it is calculated from are closed, since a function can only attain a boundary iff all its arguments do.

Both + and - are simply moriotonic throughout the/whole 2D space. * and /, however, are not simply monotonic throughout the space, they have four regions of simple monotonicity. For instance, / has the monotonicities given in figure 3-1.

Consider the application of / to \(\langle[2,3],[-1,1]\rangle\). INT breaks this down into the separate applications of / to \(\langle[2,3],[-1,-0)\rangle\) and \(\langle[2,3],[+0,1]\rangle\), in which regions / is simply monotonic. Calculating these applications from the monotonicity information given abcve yields the following process.

To calculate / applied to \(\langle[2,3],[-1,-0)\rangle\)
\begin{tabular}{|c|c|}
\hline \[
\begin{aligned}
& \langle u p, \text { up }\rangle \text { in } \\
& \langle[-\infty,-0],[+0,+\infty]\rangle
\end{aligned}
\] & \[
\begin{aligned}
& \text { <up, down> in } \\
& \langle[+0,+\infty],[+0,+\infty]\rangle
\end{aligned}
\] \\
\hline <down,up> in \(\langle[-\infty,-0],[-\infty,-0]\rangle\) & <down,down> in
\[
\langle[+0,+\infty],[-\infty,-0]\rangle
\] \\
\hline
\end{tabular}

Figure 3-1: The Monotonicities of /
invert both arguments and map \(/\) across them to produce
[3/-0,2/-1]
Evaluate the boundaries to produce
[->, -2]
To apply / to \(\langle[2,3],[+0,1]\rangle\)
invert the second argument and map / across the result to produce [2/1,3/+0]
Evaluate the boundaries to produce
\([2,+\infty]\)
The two resulting intervals are then combined into a new interval whose upper bound is their maximum upper bound and whose lower bound is their minimum lower bound. This yields the interval \([-\infty,+\infty]\). Note that this interval is too permissive, in the sense that it includes the interval ( \(-2,2\) ), which should be excluded.
4. The Algorithm

After the informal introduction of the preceeding sections we now turn to a formal account of the general, interval-arithmetic algorithm, which we will call Int-Apply.

To Int-Apply a function, \(f\), to a region, \(r\), we must consider two cases:
(a) If \(f\) is simply monotonic on \(r\) then

Form a pseudo-region \(r^{\prime}\) from \(r\) by replacing the jth element of \(r\) by its inverse iff \(f\) is monotonically decreasing on its jth argument.

Map \(f\) across \(r^{\prime}\), evaluate the boundaries of the result and return it.
(b) If \(f\) is not simply monotonic on \(r\) then

Split \(r\) intc a sub-region on which \(f\) is simply monotonic and the \(2^{n}-1\) complementary sub-regions (some of which may be empty).

Call the procedure recursively on each of the \(2^{n}\) sub-regions which are not empty.

Combine the resulting intervals into one interval.
The above description leaves various sub-procedures undefined, namely the
processes of＇mapping across＇，＇inverting＇，＇evaluating the boundaries＇， splitting a region into sub－regions and combining several regions into one．We now proceed to define these processes．
－Intervals．Intervals are represented as quadruples，in which the first and last elements are either＇open＇or＇closed＇，to represent the type of boundary，and the second and third elements are numbers representing the value of the boundaries，e．g．\((90,150]\) is represented by＜open， 90,150, closed＞．
－Inverting．The inverse of interval \(\langle L, B, T, R\rangle\) is \(\langle R, T, B, L\rangle\) ．
－Mapping Across．The mapping of \(f\) across
\[
\langle\langle L 1, B 1, T 1, R 1\rangle, \ldots,\langle L n, B n, T n, R n\rangle\rangle
\]
is
\[
\langle c(\{L 1, \ldots, L n\}), f(B 1, \ldots, B n), f(T 1, \ldots, T n), c(\{R 1, \ldots, R n\})\rangle
\]
where \(c(\) Set \()=0\) pen iff open 6 Set and \(c(\) Set \()=c l o s e d\) otherwise．
－Evaluation．As can be seen from previous sections the normal arithmetic functions must be extended to \(d \in a l\) with the infinite numbers \(-\infty 0\) and \(+\infty\) ．It is also necessary to distinguish between -0 and +0 ，since \(3 /-0=-\infty\) and \(3 /+0=+\infty\) ．For evaluating the boundaries ：of intervals INT uses an arbitrary－precision， rational－number，arithmetic package developed by Richard 0＇Keefe．In this package numbers are represented by triples of ＜sign，numerator，denominator〉，e．g．〈－，2，1＞．＋oo is represented by \(\langle+, 1,0\rangle\) and -0 by \(\langle-, 0,1\rangle\) ，etc．The standard rational arithmetic operations require only trivial adaptation to return the correct answers for infinite numbers．In indeterminate cases，e．g．0／0，the answer＇undefined＇is returned．Such an answer causes Int－Apply to return the default interval（ \(-\infty,+\infty\) ）．
－Splitting．To split region＜I1，．．．，In＞into appropriate sub－regions go through the simply monotonic regions of \(f\) until one，〈M1，．．．Min＞， is found with the property that each ij is a disjcint union or intervals \(I j^{\prime}\) and \(I j "\) ，where \(I j "\) is non－empty，and a sub－interval of Mj．Return the \(2^{n}\) sub－regions 〈I1＊，．．．，In＊〉，where \(I j^{*}\) is either \(I j^{\prime}\) or Ij＂．Note that f is simply monotonic on sub－region 〈I1＂，．．．，In＂＞． If \(I j^{\prime}\) is empty then so is any region containing it．If 〈I1，．．．，In＞ cannot be divided into a finite set of simply monotonic sub－regions then this splitting process may not terminate．
－Combining．To combine a set of intervals，form a new interval whose lower bound is the minimum of the lower bounds of the set and whose upper bound is the maximum of the set．Note that the minimum of two boundaries with the same value，but different tjpes，is the closed boundary．Similar？y with the maximum．The combination of two intervals is the smallest intervaj containing their union，but it may not be equal to their union，e．g．combining \([-\infty,-2]\) and \([2,+\infty]\) produces \([-\infty,+\infty]\) ．

\section*{5. Semantic Checking}

The above sections describe an interval arithmetic algorithm. We now explain the applications of this algorithm in the algebraic manipulation package, PRESS, [Bund and Welham 81].

As describe in the introduction, the applications are twofold: checking the conditions of rewrite rules and vetting the solutions to equations. Both these applications make use of a common sub-procedure, Find-Int. Find-Int takes an algebraic term and returns the interval within which it lies. For instance, if \(a, m 1, m 2\) and \(g\) are all positive then Find-Int applied to the term \(-m 1 * g /(m 1+m 2)\) will return \((-\infty, 0)\), from which it can be deduced that the term is negative and that \(a=-m 1 * g /(m 1+m 2)\) is false.

The procedure, Find-Int, works by call by value. Applied to a term \(f\left(t_{1}, \ldots, t_{n}\right)\), Find-Int is called recursively on earn \(t_{j}\) and returns \(i_{j}\). \(f\) is then applied to \(\left\langle i_{1}, \ldots, i_{n}\right\rangle\) by Int-Apply. If Find-Int is applied to a number, \(n\), then the interval \([n, n]\) is returned, egg. [2,2]. If Find-Int is applied to a symbolic constant or variable, e.g. mi, then semantic information is used to try to determine the result. If this is not successful then the default interval, \((-\infty,+\infty)\), is returned.

The semantic information is provided by the MECHO program, [Bund et al 79], for solving Mechanics problems. Information is provided about two sorts of constant: physical quantities (e.g. masses, accelerations, etc) and angles. All physical quantities are assumed to be positive, that is, to lie in the interval \((0,+\infty)\). Provision exists for providing more sophisticated information about each kind of quantity, but this has not been exploited. In the case of angles an attempt is made to infer in which quadrant (s) of the circle the angle lies, for instance, \([0,90],[180,360]\), etc.

Angles are defined in MECHO either as the inclination or the normal to a 2 dimensional curve, where the normal is towards the convex side of the curve and the inclination is 90 degrees greater. For simple curves, i.e. monotonic curves with monotonic first derivatives, both the inclination and the normal lie wholly within a single quadrant. Which quadrant this is depends on the sign of the first and second derivatives. We call the first derivative the slope and the second derivative the concavity of the curve. For instance, if the slope is positive and the concavity is positive then the inclination lies in the quadrant \([0,90]\) and the normal lies in the quadrant \([270,360]\). This and the other seven cases are illustrated in figure 5-1.

If Find-Int is applied to an angle defined on a non-simple curve then the curve is first broken into simple curves. the above process ip applied to each of these and the resulting quadrants are combined, using the combination procedure described in section 4.

\section*{6. Limitations}


The INT package uses information about the simply monotonic regions of a function to extend its definition on numbers to one on intervals. In this way it generalises previous interval packages, which could deal with only a pre-specified number of functions. However, INT only works well on functions which are well behaved monotonically, ie. divide the whole space into a finite set of continuous simply monotonic regions. Some of the situations in which INT does not behave ideally are listed below.
\begin{tabular}{|c|c|c|c|}
\hline  & Positive & Negative & Zero \\
\hline Positive &  &  &  \\
\hline Negative &  &  & \[
\begin{aligned}
& \text { normal } \in(180,270) \\
& \text { incline } \in(270,360)
\end{aligned}
\] \\
\hline Zero & impossible & impossiole &  \\
\hline Infinity & impossible & impossible &  \\
\hline
\end{tabular}

Figure 5-1: Classification of the Angles of Simple Curves
- INT cannot deal with the situation where a function must be applied to a region which only divides into an infinite set of simply monotonic sub-regions, because Int-Apply will not terminate. For instance, sin applied to \(\langle(-\infty,+\infty)\rangle\) should return \([-1,1]\), but \(\langle(-\infty,+\infty)\rangle\) cannot be divided into a finite set of simply monotonic sub-regions, so Int-Apply will not terminate. This particular case is dealt with by a patch to the INT package to take into account the boundedness of sin, cos, etc.
- Int-Apply is constrained to return a single continuous interval as its result. To return, say, a set of disjoint intervals would alter the whole basis of the algorithm with a consequent loss of simplicity and efficiency. The price to be paid for this is that the result of Int-Apply will sometimes be too inclusive. An example of this was given in section 3 where the result of \([2,3] /[-1,1]\) included the interval (-2,2). \(\downarrow\)
- Scmetimes the best description of the monotonicity of a function involves non-continuous sets of numbers. For instance, \(x^{y}\) has monotonicity <down, up> when \(x\) is negative and \(y\) is an even rational. Unfortunately, the even rationals cannot be represented as an interval. In fact exponentiation has no simply monotonic sub-regions in \(\langle[-\infty, 0],[-\infty,+\infty]\rangle\). There is a patch in INT to deal with this particular case, but in general such situations are outside its scope.

\section*{7. Conclusion}

We have seen that the INT interval arithmetic package generalises previous interval packages by the use of the monotonicity of functions. It can deal with more functions than previous packages and can deal correctly with the application of a function to a region which straddies a finite number of simply monotonic regions. It breaks down only when a function has a particularly complex monotonis behaviour.

The INT package has been applied in unusual ways, to check the conditions of rewrite rules and to vet the solutions to equations, as part of an algebraic manipulation package.

\section*{References}
[Bundy and Welham 81]
Bundy, A, and Weiham, B.
Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulaijion.
Artificial Inteligence, in press 1981.
[Bundy st al 79]
Bundy, A., Byrd, L., Luger, G., Mellish, C., Milne, R. and
Palmer, M.
Solving Mechanics Problems Using Meta-Level Infererce.
In Procs of the sixth. IJCAI, Tokyo, i979.
Also available from Edinburgh as DAI Research Paper No. 112.
[Good \& London 70]
Good, D.I. and London, R.L.
Computer interval arithmetic: definjitifn and proof of correct
implementation.
JACM 17(4):603-612, 1970.
```

ALBERT - LUD W I GS - UN IVERS I TÄ T

```

INSTITUT FÜR ANGEWANDTE MATHEMATIK
Prof. Dr. K. Nickel

D-7800 FREIBURG I. BR., den May 5, 1981
Hermann-Herder-Strasse 10
Telefon (0761) 203-3062

Dr. Alan Bundy
University of Edinburgh
Department of Artificial Intelligence
Hope Park Square, Meadow Lane


Edinburgh EH8 9NW

Dear Doctor Bundy!

Thank you very much for your letter of April 20 and for your manuscript "A generalized interval package ..." I read it with great interest. We will include it in our "interval library" such that everybody interested in it can look at it. I enclose a list of the papers which are included in that library.

Some years ago \(I\) worked also in the field of the monotonicity of functions. A paper of mine is enclosed. I found your ideas very stimulating and \(I\) am sorry that \(I\) did not know them at that time.

I would very much like to see your results published. Probabely a Journal on Computer Science and/or on Artificial Intelligence would be the best. Perhaps you could also add some practical results? Unfortunately I do not know which Journal accepts such a paper. There is still much hesitation in accepting papers from Interval Mathematics.

Sincerely yours
Kor Viol

Morme R\& Interal Analysis

Rentise Hall 1966.```

