Kalkulus I.

Járai Antal

Ezek a programok csak szemléltetésre szolgálnak

1. Halmazok

```
> restart;
```

1.1. Halmazelméleti alapfogalmak

1.1.1. Halmazelmélet.

```
> A:={a,b,c}; member(b,A); member(d,A); b in A; evalb(%); d
  in A; evalb(%);
```

$$A := \{b, c, a\}$$
$$true$$
$$false$$
$$b \in (\{b, c, a\})$$
$$true$$
$$d \in (\{b, c, a\})$$
$$false \tag{1.1.1.1}$$

```
> whattype(A); whattype(a); whattype(2); whattype
  (krikszkraksz); whattype("krikszkraksz");
```

$$set$$
$$symbol$$
$$integer$$
$$symbol$$
$$string \tag{1.1.1.2}$$

1.1.2. Meghatározottság.

```
> {a,a,b,b,a}; {b,a};
```

$$\{b, a\}$$
$$\{b, a\} \tag{1.1.2.1}$$

1.1.3. Részhalmazok.

```
> {a,b} subset {a,b,d};
```

$$true \tag{1.1.3.1}$$

```
> {a,c} subset {a,b,d};
```

$$false \tag{1.1.3.2}$$

```
> {a,b} subset {a,b};
```

$$true \tag{1.1.3.3}$$

```
> X subset X; X subset Y;
```

$$true$$
$$X \subseteq Y \tag{1.1.3.4}$$

```
> A:={1,2,3,4,5,6,7}; select(x->isprime(x),A);
```

$$A := \{1, 2, 3, 4, 5, 6, 7\}$$
$$\{2, 3, 5, 7\} \tag{1.1.3.5}$$

```
> {}; {a,b}; {a,a}; {a,b,c};
```

$$\{\}$$
$$\{b, a\}$$
$$\{a\}$$
$$\{b, c, a\} \tag{1.1.3.6}$$

```
> a:=b; {a,b};
```

$$a := b$$
$$\{b\} \tag{1.1.3.7}$$

```
> a:='a'; {a,b};
```

$$a := a$$
$$\{b, a\} \tag{1.1.3.8}$$

```
> {} subset X;
```

$$true \tag{1.1.3.9}$$

1.1.4. Unió.

```
> {a, b} union {b, c};      {a, b} union {b, c};
```

$$\{b, c, a\}$$
$$\{b, c, a\} \tag{1.1.4.1}$$

```
> A := {{a}, {b, c}, {1, 2, b}, {}, {a, b}}; op(A); `union`(op(A));
  `union`({a}, {b, c}, {1, 2, b}, {}, {a, b});
```

$$\mathcal{A} := \{\{\}, \{b, a\}, \{a\}, \{1, 2, b\}, \{b, c\}\}$$
$$\{\}, \{b, a\}, \{a\}, \{1, 2, b\}, \{b, c\}$$

$$\{1, 2, b, c, a\}$$
$$\{1, 2, b, c, a\} \tag{1.1.4.2}$$

```
> {a,b} union {b,c}; `union`({a},{b,c},{1,2,b},{},{a,b});
```

$$\{b, c, a\}$$
$$\{1, 2, b, c, a\} \tag{1.1.4.3}$$

```
> A:={a,b}; B:={b,c}; `union`(A);`union`(A,B);
```

$$A := \{b, a\}$$
$$B := \{b, c\}$$
$$\{b, a\}$$
$$\{b, c, a\} \tag{1.1.4.4}$$

```
> `union`();
```

$$\{\ \} \tag{1.1.4.5}$$

* 1.1.5. Axiómatikus halmazelmélet.

1.1.6. Állítás: az unió tulajdonságai.

```
> X union {};
```

$$X \tag{1.1.6.1}$$

```
> X union Y; Y union X;
```

$$X \cup Y$$
$$X \cup Y \tag{1.1.6.2}$$

```
> (X union Y) union Z; X union (Y union Z);
```

$$union(X, Y, Z)$$
$$union(X, Y, Z) \tag{1.1.6.3}$$

```
> X union X;
```

$$X \tag{1.1.6.4}$$

1.1.7. Metszet.

```
> {a,b,1} intersect {a,c,2,1};
```

$$\{1, a\} \tag{1.1.7.1}$$

```
> `intersect`({a,b,c,d},{a,b,c,1},{a,b,1,2});
```

$$\{b, a\} \tag{1.1.7.2}$$

```
> A:={a,b}; B:={b,c}; C:={c,a};
  A intersect B; B intersect C; C intersect A; `intersect`(A,
  B);
  `intersect`(A,B,C);
```

$$A := \{b, a\}$$
$$B := \{b, c\}$$
$$C := \{c, a\}$$
$$\{b\}$$
$$\{c\}$$
$$\{a\}$$
$$\{b\}$$
$$\{\ \}$$ (1.1.7.3)

1.1.8. Állítás: a metszet tulajdonságai.

Az első négy tulajdonságot a Maple is ismeri:

```
> X intersect {};
```
$$\{\ \}$$ (1.1.8.1)

```
> X intersect Y; Y intersect X;
```
$$X \cap Y$$
$$X \cap Y$$ (1.1.8.2)

```
> X intersect (Y intersect Z); (X intersect Y) intersect Z;
```
$$intersect(X, Y, Z)$$
$$intersect(X, Y, Z)$$ (1.1.8.3)

```
> X intersect X;
```
$$X$$ (1.1.8.4)

1.1.9. Állítás: disztributivitási szabályok.

```
> X intersect (Y union Z); expand(%);
```
$$X \cap (Y \cup Z)$$
$$X \cap Y \cup X \cap Z$$ (1.1.9.1)

```
> X union (Y intersect Z); A:={a,b,c}; B:={b,c,d}; C:={c,d,e}
  ;
  A union (B intersect C); (A union B) intersect (A union C);
```
$$X \cup Y \cap Z$$
$$A := \{b, c, a\}$$
$$B := \{b, c, d\}$$
$$C := \{c, d, e\}$$
$$\{b, c, a, d\}$$

$$\{b, c, a, d\} \tag{1.1.9.2}$$

1.1.10. Különbség és komplementer.

```
> A:={a,b}; B:={b,c}; C:={a,b,c,d}; A minus B;
  symmdiff(A,B); symmdiff(A,B,C);
```

$$A := \{b, a\}$$
$$B := \{b, c\}$$
$$C := \{b, c, a, d\}$$
$$\{a\}$$
$$\{c, a\}$$
$$\{b, d\} \tag{1.1.10.1}$$

1.1.11. Állítás.

```
> A; C minus (C minus A);
```

$$\{b, a\}$$
$$\{b, a\} \tag{1.1.11.1}$$

```
> C; C minus {};
```

$$\{b, c, a, d\}$$
$$\{b, c, a, d\} \tag{1.1.11.2}$$

```
> {}, C minus C;
```

$$\{\}, \{\} \tag{1.1.11.3}$$

```
> {}; A intersect (C minus A);
```

$$\{\}$$
$$\{\} \tag{1.1.11.4}$$

```
> C; A union (C minus A);
```

$$\{b, c, a, d\}$$
$$\{b, c, a, d\} \tag{1.1.11.5}$$

```
> B:={a,b,d}; A; C minus B; C minus A;
```

$$B := \{b, a, d\}$$
$$\{b, a\}$$
$$\{c\}$$
$$\{c, d\} \tag{1.1.11.6}$$

```
> A; B:={b,c}; C minus (A union B); (C minus A) intersect (C
  minus B);
```

$$\{b, a\}$$

$$B := \{b, c\}$$
$$\{d\}$$
$$\{d\} \tag{1.1.11.7}$$

```
> C minus (A intersect B); (C minus A) union (C minus B);
```

$$\{c, a, d\}$$
$$\{c, a, d\} \tag{1.1.11.8}$$

1.1.12. Hatványhalmaz.

```
> with(combinat,powerset):
  powerset({a,b,c}); powerset({a,b}); powerset({a}); powerset
  ({});
```

$$\{\{\}, \{b, c, a\}, \{b, a\}, \{a\}, \{b, c\}, \{c, a\}, \{b\}, \{c\}\}$$
$$\{\{\}, \{b, a\}, \{a\}, \{b\}\}$$
$$\{\{\}, \{a\}\}$$
$$\{\{\ \}\} \tag{1.1.12.1}$$

1.2. Relációk

1.2.1. Descartes-szorzat és reláció.

```
> evalb([x,y]=[y,x]);evalb([x,x,y]=[x,y,x]);
```

$$false$$
$$false \tag{1.2.1.1}$$

```
> p:=[x,y,z]; p[1]; p[2]; p[3];
```

$$p := [x, y, z]$$
$$x$$
$$y$$
$$z \tag{1.2.1.2}$$

```
> descartesprod:=proc(L::list(set)) local x,y,i,S,SS;
  if nops(L)=0 then return {} fi; S:=map(x->[x],L[1]);
  for i from 2 to nops(L) do SS:={};
    for x in S do for y in L[i] do SS:=SS union {[op(x),y]}
  od;
  od; S:=SS od; S; end;

  descartesprod([]); descartesprod([{1,2}]);
```

```
descartesprod([{1,2},{a,b}]);descartesprod([{1,2},{}]);
descartesprod([{1,2},{a},{x,y}]);descartesprod([{1,2},{a,b,
c},{x,y}]);
```

```
descartesprod := proc(L::(list(set)))
    local x, y, i, S, SS;
    if nops(L) = 0 then
        return {}
    end if;
    S := map(proc(x)
        option operator, arrow;
        [x]
    end proc, L[1]);
    for i from 2 to nops(L) do
        SS := {};
        for x in S do
            for y in L[i] do
                SS := union(SS, {[op(x), y]})
            end do
        end do;
        S := SS
    end do;
    S
end proc
```

$$\{\}$$

$$\{[1], [2]\}$$

$$\{[1, b], [1, a], [2, b], [2, a]\}$$

$$\{\}$$

$$\{[1, a, x], [1, a, y], [2, a, x], [2, a, y]\}$$

$$\{[1, a, x], [1, a, y], [2, a, x], [2, a, y], [1, c, x], [1, c, y], [2, c, x], [2, c, y], [1, b, x], [1, b, y], [2, b, x], [2, b, y]\} \quad (1.2.1.3)$$

```
> agent:={[D209,"Peti"],[KISZ1,"Fleto"],[Puf3,"Gyula"]};

  event:={[KISZ1,"Balaton",19930706],[Puf3,"Nyugati",
  19561108],[KISZ1,"Motim",19961231],[D209,"Paks",20000103],
  [KISZ1,"Fittelina",19980320],[D209,"Gresham",20010908],
  [KISZ1,"Nomentana",19951122]};

  descartesprod([agent,event]):select(x->x[1][1]=x[2][1],%)
```

```
:map(x->[x[1][2],x[2][2],x[2][3],x[2][1]],%):active:=select
(x->x[3]>19891023,%);
```

$$agent := \{[D209, \text{"Peti"}], [KISZ1, \text{"Fleto"}], [Puf3, \text{"Gyula"}]\}$$
$$event := \{[Puf3, \text{"Nyugati"}, 19561108], [KISZ1, \text{"Balaton"}, 19930706], [KISZ1, \text{"Motim"}, 19961231], [D209, \text{"Paks"}, 20000103], [KISZ1, \text{"Fittelina"}, 19980320], [D209, \text{"Gresham"}, 20010908], [KISZ1, \text{"Nomentana"}, 19951122]\}$$
$$active := \{[\text{"Fleto"}, \text{"Balaton"}, 19930706, KISZ1], [\text{"Fleto"}, \text{"Motim"}, 19961231, KISZ1], [\text{"Fleto"}, \text{"Fittelina"}, 19980320, KISZ1], [\text{"Fleto"}, \text{"Nomentana"}, 19951122, KISZ1], [\text{"Peti"}, \text{"Paks"}, 20000103, D209], [\text{"Peti"}, \text{"Gresham"}, 20010908, D209]\} \quad (1.2.1.4)$$

1.2.2. Megjegyzés.

1.2.3. Binér relációk.

```
> bincartprod:=proc(X::set,Y::set) local x,y,Z; Z:={};
  for x in X do for y in Y do Z:=Z union {[x,y]}; od; od; Z;
  end;
```

```
bincartprod := proc(X::set, Y::set)                                    (1.2.3.1)
    local x, y, Z;
    Z := {};
    for x in X do
        for y in Y do
            Z := union(Z, {[x, y]})
        end do
    end do;
    Z
end proc
```

```
> bincartprod({1,2,3},{a,b});
```

$$\{[3, a], [3, b], [1, b], [1, a], [2, b], [2, a]\} \quad (1.2.3.2)$$

```
> `type/ordpair`:=proc(x) type(x,list) and nops(x)=2 end;

  type([a,b],ordpair); type([1,2,3],ordpair);
```

```
type/ordpair := proc(x) type(x, list) and nops(x) = 2 end proc
                                true
                                false                                  (1.2.3.3)
```

```
> binrel:=proc(R::set(ordpair),X::set,Y::set) local r;
  if nargs<1 or nargs>3 then error ": needs 1..3 arguments"
  fi;
  for r in R do
    if nargs=2 and not (member(r[1],X) and member(r[2],X))
      then return false;
    elif nargs=3 and not (member(r[1],X) and member(r[2],Y))
      then return false
    fi;
  od; true; end;
```

$$binrel := \mathbf{proc}(R::(set(ordpair)), X::set, Y::set) \tag{1.2.3.4}$$

```
    local r;
    if nargs < 1 or 3 < nargs then
        error ": needs 1..3 arguments"
    end if;
    for r in R do
        if nargs = 2 and not (member(r[1],
        X) and member(r[2], X)) then
            return false
        elif nargs = 3 and not (member(r[1],
        X) and member(r[2], Y)) then
            return false
        end if
    end do;
    true
end proc
```

```
> binrel({});R:=bincartprod({1,2,3},{a,b});binrel(R);
  binrel(R,{0,1,2,3},{a,b,c});binrel(R,{a,b},{1,2,3});binrel
  (R,{1,2,3,a,b});
```

$$\begin{gathered} true \\ R := \{[3, a], [3, b], [1, b], [1, a], [2, b], [2, a]\} \\ true \\ true \\ false \\ true \end{gathered} \tag{1.2.3.5}$$

```
> id:=proc(X) local x; map(x->[x,x],X); end;id({1,3,a});
```

$$id := \mathbf{proc}(X)$$

```
    local x;
    map(proc(x)
```

```
        option operator, arrow;
        [x, x]
    end proc, X)
end proc
```

$$\{[1, 1], [3, 3], [a, a]\} \tag{1.2.3.6}$$

```
> F:={{1},{2},{1,2}}; FF:=bincartprod(F,F);select(x->x[1]
  subset x[2],FF);select(x->x[1] subset x[2] and not x[1]=x
  [2],FF);
```

$$F := \{\{1\}, \{2\}, \{1, 2\}\}$$

$$FF := \{[\{1, 2\}, \{1, 2\}], [\{1\}, \{2\}], [\{1\}, \{1, 2\}], [\{2\}, \{1\}], [\{2\}, \{2\}], [\{2\}, \{1, 2\}], [\{1, 2\}, \{1\}], [\{1, 2\}, \{2\}], [\{1\}, \{1\}]\}$$

$$\{[\{1, 2\}, \{1, 2\}], [\{1\}, \{1, 2\}], [\{2\}, \{2\}], [\{2\}, \{1, 2\}], [\{1\}, \{1\}]\}$$

$$\{[\{1\}, \{1, 2\}], [\{2\}, \{1, 2\}]\} \tag{1.2.3.7}$$

```
> irem(13,5);X:={1,2,3,4,5,6};XX:=bincartprod(X,X):
  R:=select(x->irem(x[2],x[1])=0,XX);
```

$$3$$

$$X := \{1, 2, 3, 4, 5, 6\}$$

$$R := \{[1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [2, 2], [3, 6], [4, 4], [2, 4], [2, 6], [5, 5], [1, 1], [3, 3], [6, 6]\} \tag{1.2.3.8}$$

▶ *1.2.4. Relációk gráfja.*

▼ *1.2.5. Értelmezési tartomány, értékkészlet.*

```
> R:={[1,a],[1,b],[2,b],[3,d],[2,d],[4,e]};
  dmn:=proc(R::set(ordpair)) map(x->x[1],R);end; dmn(R);
  rng:=proc(R::set(ordpair)) map(x->x[2],R); end; rng(R);
```

$$R := \{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$

```
dmn := proc(R::(set(ordpair)))
    map(proc(x)
        option operator,
        arrow;
        x[1]
    end proc, R)
end proc
```

$$\{1, 2, 3, 4\}$$

```
rng := proc(R::(set(ordpair)))
    map(proc(x)
```

```
        option operator,
       arrow;
        x[2]
    end proc, R)
end proc
```

$$\{b, a, d, e\} \tag{1.2.5.1}$$

1.2.6. Kiterjesztés, leszűkítés.

```
> R;select(x->(x[1]>1 and x[2]<>b),R);
  restrict:=proc(R::set(ordpair),X::set) select(x->(x[1] in
  X),R); end;
  X:={2,3};restrict(R,X);
```

$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$
$$\{[3, d], [2, d], [4, e]\}$$

```
restrict := proc(R::(set(ordpair)), X::set)
    select(proc(x)
        option operator, arrow;
        in(x[1], X)
    end proc, R)
end proc
```

$$X := \{2, 3\}$$
$$\{[2, b], [3, d], [2, d]\} \tag{1.2.6.1}$$

1.2.7. Inverz.

```
> relinv:=proc(R::set(ordpair)) map(x->[x[2],x[1]],R); end;

  R;dmn(R);rng(R);S:=relinv(R);dmn(S);rng(S);relinv(S);

relinv := proc(R::(set(ordpair)))
    map(proc(x)
        option operator,
       arrow;
        [x[2], x[1]]
    end proc, R)
end proc
```

$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$

$$\{1, 2, 3, 4\}$$
$$\{b, a, d, e\}$$
$$S := \{[b, 1], [a, 1], [b, 2], [d, 3], [d, 2], [e, 4]\}$$
$$\{b, a, d, e\}$$
$$\{1, 2, 3, 4\}$$
$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\} \qquad (1.2.7.1)$$

1.2.8. Halmaz képe és inverz képe.

```
> mapset:=proc(R::set(ordpair),A::set) rng(restrict(R,A))
  end;
  invmapset:=proc(R::set(ordpair),A::set) rng(restrict(relinv
  (R),A)) end;

  R;S:=relinv(R);A:={1,4};B:={b,e}; mapset(R,A);mapset(R,B);
  invmapset(R,B);mapset(S,B);
```

$mapset := \mathbf{proc}(R::(set(ordpair)), A::set)$
$rng(restrict(R, A))$
end proc
$invmapset := \mathbf{proc}(R::(set(ordpair)), A::set)$
$rng(restrict(relinv(R),$
$A))$
end proc

$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$
$$S := \{[b, 1], [a, 1], [b, 2], [d, 3], [d, 2], [e, 4]\}$$
$$A := \{1, 4\}$$
$$B := \{b, e\}$$
$$\{b, a, e\}$$
$$\{\ \}$$
$$\{1, 2, 4\}$$
$$\{1, 2, 4\} \qquad (1.2.8.1)$$

1.2.9. Kompozíció.

```
> relcomp:=proc(R::set(ordpair),S::set(ordpair)) local r,s,T;
  T:={};
  for r in R do for s in S do
    if s[2]=r[1] then T:=T union {[s[1],r[2]]}; fi;
  od; od; T; end;
```

```
R;S:={[aa,1],[bb,3],[cc,5]};relcomp(R,S);
```

```
relcomp := proc(R::(set(ordpair)), S::(set(ordpair)))
    local r, s, T;
    T := {};
    for r in R do
        for s in S do
            if s[2] = r[1] then
                T := union(T,
                {[s[1], r[2]]})
            end if
        end do
    end do;
    T
end proc
```

$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$
$$S := \{[aa, 1], [bb, 3], [cc, 5]\}$$
$$\{[aa, b], [aa, a], [bb, d]\} \tag{1.2.9.1}$$

1.2.10. Állítás.

```
> R;S:={[aa,1],[aa,2],[bb,3],[cc,4],[dd,5]};
  relcomp(R,S);rng(R);rng(%%);
```

$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$
$$S := \{[aa, 2], [cc, 4], [dd, 5], [aa, 1], [bb, 3]\}$$
$$\{[aa, b], [aa, a], [bb, d], [aa, d], [cc, e]\}$$
$$\{b, a, d, e\}$$
$$\{b, a, d, e\} \tag{1.2.10.1}$$

```
> T:={[xx,aa],[xx,cc]};relcomp(R,relcomp(S,T));relcomp
  (relcomp(R,S),T);
```

$$T := \{[xx, aa], [xx, cc]\}$$
$$\{[xx, b], [xx, a], [xx, d], [xx, e]\}$$
$$\{[xx, b], [xx, a], [xx, d], [xx, e]\} \tag{1.2.10.2}$$

```
> relinv(relcomp(R,S));relcomp(relinv(S),relinv(R));
```

$$\{[b, aa], [a, aa], [d, bb], [d, aa], [e, cc]\}$$
$$\{[b, aa], [a, aa], [d, bb], [d, aa], [e, cc]\} \tag{1.2.10.3}$$

1.2.11. Állítás.

```
> R;IX:=id({1,2,3,4,5});relcomp(R,IX);IY:=id({a,b,c,d,e});
  relcomp(IY,R);
```

$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$
$$IX := \{[2, 2], [4, 4], [5, 5], [1, 1], [3, 3]\}$$
$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\}$$
$$IY := \{[a, a], [b, b], [c, c], [d, d], [e, e]\}$$
$$\{[1, b], [1, a], [2, b], [3, d], [2, d], [4, e]\} \qquad (1.2.11.1)$$

1.2.12. Rendezés.

```
> istransitive:=proc(R::set(ordpair)) local r,s;
  for r in R do for s in R do
    if r[2]=s[1] and not [r[1],s[2]] in R then return false
  fi;
  od; od; true; end;

  X:={1,2,3,4,5,6};XX:=bincartprod(X,X):R:=select(x->irem(x
  [2],x[1])=0,XX);istransitive(R);
```

```
istransitive := proc(R::(set(ordpair)))
    local r, s;
    for r in R do
        for s in R do
            if r[2] = s[1] and not in([r[1], s[2]], R) then
                return false
            end if
        end do
    end do;
    true
end proc
```

$$X := \{1, 2, 3, 4, 5, 6\}$$
$$R := \{[1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [2, 2], [3, 6], [4, 4], [2, 4], [2, 6], [5, 5], [1, 1], [3, 3], [6, 6]\}$$
$$true \qquad (1.2.12.1)$$

```
> istrichotom:=proc(X::set,R::set(ordpair)) local x,y;
  if not binrel(R,X) then return FAIL fi;
  for x in X do for y in X do
    if x<>y then if ([x,y] in R and [y,x] in R) or ((not [x,
  y] in R) and
```

```
      (not [y,x] in R)) then return false fi;fi;
   od;od; true; end;

   istrichotom(X,R);
```

istrichotom := **proc**(*X*::*set*, *R*::(*set*(*ordpair*)))
local *x*, *y*;
if not *binrel*(*R*, *X*) **then**
return *FAIL*
end if;
for *x* **in** *X* **do**
for *y* **in** *X* **do**
if *x* <> *y* **then**
if *in*([*x*, *y*], *R*) **and** *in*([*y*, *x*], *R*) **or not** (*in*([*x*, *y*], *R*) **or** *in*([*y*, *x*], *R*)) **then**
return *false*
end if
end if
end do
end do;
true
end proc

false (1.2.12.2)

```
> isordering:=proc(X::set,R::set(ordpair)) local x;
   if not binrel(R,X) then return FAIL fi;
   istransitive(R) and istrichotom(X,R); end;

   X:={a,u,i,e,o};R:={[a,u],[a,i],[a,e],[a,o],[u,i],[u,e],[u,
   o],[i,e],[i,o],[e,o]};isordering(X,R);
```

isordering := **proc**(*X*::*set*, *R*::(*set*(*ordpair*)))
local *x*;
if not *binrel*(*R*, *X*) **then**
return *FAIL*
end if;
istransitive(*R*) **and** *istrichotom*(*X*, *R*)
end proc

$X := \{i, a, u, o, e\}$

$$R := \{[a, u], [a, i], [a, e], [a, o], [u, i], [u, e], [u, o], [i, e], [i, o], [e, o]\}$$

$$true \tag{1.2.12.3}$$

1.2.13. Példa.

1.2.14. Intervallumok.

```
> int_o_o:=proc(X::set,R::set(ordpair),x,y) local S,z; S:={};
  for z in X do
    if [x,z] in R and [z,y] in R then S:=S union {z}; fi;
  od; S; end;

  int_o_c:=proc(X::set,R::set(ordpair),x,y) local S,z; S:={};
  for z in X do
    if [x,z] in R and ([z,y] in R or z=y) then S:=S union {z}
  ; fi;
  od; S; end;

  int_i_o:=proc(X::set,R::set(ordpair),x) local S,y; S:={};
  for y in X do
    if [y,x] in R then S:=S union {y}; fi;
  od; S; end;
```

```
int_o_o := proc(X::set, R::(set(ordpair)), x, y)
    local S, z;
    S := {};
    for z in X do
        if in([x, z], R) and in([z, y], R) then
            S := union(S,
            {z})
        end if
    end do;
    S
end proc
int_o_c := proc(X::set, R::(set(ordpair)), x, y)
    local S, z;
    S := {};
    for z in X do
        if in([x, z], R) and (in([z, y], R) or z = y) then
            S := union(S, {z})
        end if
```

```
    end do;
    S
end proc
```

$$int\_i\_o := \mathbf{proc}(X::set,\ R::(set(ordpair)),\ x) \tag{1.2.14.1}$$

```
    local S, y;
    S := {};
    for y in X do
        if in([y, x], R) then
            S := union(S, {y})
        end if
    end do;
    S
end proc
```

```
> X;R;int_o_o(X,R,u,o);int_o_c(X,R,a,o);int_i_o(X,R,o);
```

$$\{i, a, u, o, e\}$$
$$\{[a, u], [a, i], [a, e], [a, o], [u, i], [u, e], [u, o], [i, e], [i, o], [e, o]\}$$
$$\{i, e\}$$
$$\{i, u, o, e\}$$
$$\{i, a, u, e\} \tag{1.2.14.2}$$

```
> $ 2..6;$ 1..9;
```

$$2, 3, 4, 5, 6$$
$$1, 2, 3, 4, 5, 6, 7, 8, 9 \tag{1.2.14.3}$$

1.2.15. Legkisebb és legnagyobb elem.

```
> minimal:=proc(X::set,R::set(ordpair)) local x,y,f;
  for x in X do f:=true;
    for y in X do if x<>y and not [x,y] in R then f:=false;
  break; fi; od;
    if f then return x fi;
  od; NULL; end;

  maximal:=proc(X::set,R::set(ordpair)) local x,y,f;
  for x in X do f:=true;
    for y in X do if x<>y and not [y,x] in R then f:=false;
  break; fi; od;
    if f then return x fi;
  od; NULL; end;
```

$$minimal := \mathbf{proc}(X::set,\ R::(set(ordpair)))$$

```
    local x, y, f;
    for x in X do
        f := true;
        for y in X do
            if x <> y and not in([x, y], R) then
                f := false;
                break
            end if
        end do;
        if f then
            return x
        end if
    end do;
    NULL
end proc
```

```
maximal := proc(X::set, R::(set(ordpair)))
    local x, y, f;
    for x in X do
        f := true;
        for y in X do
            if x <> y and not in([y, x], R) then
                f := false;
                break
            end if
        end do;
        if f then
            return x
        end if
    end do;
    NULL
end proc
```

(1.2.15.1)

```
> X;R;minimal(X,R);maximal(X,R);minimal({},R);maximal({},R);
```

$$\{i, a, u, o, e\}$$

$$\{[a, u], [a, i], [a, e], [a, o], [u, i], [u, e], [u, o], [i, e], [i, o], [e, o]\}$$

$$a$$

$$o$$

(1.2.15.2)

1.2.16. Korlátok.

```
> islowerbound:=proc(X::set,R::set(ordpair),Y::set,x) local
  y;
  for y in Y do if x<>y and not [x,y] in R then return false
  fi; od;
  true; end;

  isupperbound:=proc(X::set,R::set(ordpair),Y::set,x) local
  y;
  for y in Y do if x<>y and not [y,x] in R then return false
  fi; od;
  true; end;
```

```
islowerbound := proc(X::set, R::(set(ordpair)), Y::set, x)
    local y;
    for y in Y do
        if x <> y and not in([x, y], R) then
            return false
        end if
    end do;
    true
end proc
isupperbound := proc(X::set, R::(set(ordpair)), Y::set, x)
    local y;
    for y in Y do
        if x <> y and not in([y, x], R) then
            return false
        end if
    end do;
    true
end proc
```

(1.2.16.1)

```
> X;R;islowerbound(X,R,{u,i,e},a);isupperbound(X,R,{a,u,i},i)
  ;
```

$$\{i, a, u, o, e\}$$

$$\{[a, u], [a, i], [a, e], [a, o], [u, i], [u, e], [u, o], [i, e], [i, o], [e, o]\}$$

$$true$$

$$true$$

(1.2.16.2)

```
> lowerbounds:=proc(X::set,R::set(ordpair),Y::set) local S,x;
  S:={};
  for x in X do
    if islowerbound(X,R,Y,x) then S:=S  union {x} fi;
  od; S; end;
```

```
lowerbounds := proc(X::set, R::(set(ordpair)), Y::set)                    (1.2.16.3)
    local S, x;
    S := {};
    for x in X do
        if islowerbound(X, R, Y, x) then
            S := union(S, {x})
        end if
    end do;
    S
end proc
```

```
> upperbounds:=proc(X::set,R::set(ordpair),Y::set) local S,x;
  S:={};
  for x in X do
    if isupperbound(X,R,Y,x) then S:=S  union {x} fi;
  od; S; end;
```

```
upperbounds := proc(X::set, R::(set(ordpair)), Y::set)                    (1.2.16.4)
    local S, x;
    S := {};
    for x in X do
        if isupperbound(X, R, Y, x) then
            S := union(S, {x})
        end if
    end do;
    S
end proc
```

```
> X;R;lowerbounds(X,R,{i,e}); upperbounds(X,R,{i,e});
                                {i, a, u, o, e}
  {[a, u], [a, i], [a, e], [a, o], [u, i], [u, e], [u, o], [i, e], [i, o], [e, o]}
                                {i, a, u}
                                 {o, e}                                   (1.2.16.5)
```

```
> inf:=proc(X::set,R::set(ordpair),Y::set)
  maximal(lowerbounds(X,R,Y),R); end;

  sup:=proc(X::set,R::set(ordpair),Y::set)
  minimal(upperbounds(X,R,Y),R); end;

  inf(X,R,{i,e});sup(X,R,{i,e});
```

```
inf := proc(X::set, R::(set(ordpair)), Y::set)
```

```
    maximal(lowerbounds(X, R, Y), R)
end proc
sup := proc(X::set, R::(set(ordpair)), Y::set)
    minimal(upperbounds(X, R, Y), R)
end proc
```

$$\begin{matrix} i \\ e \end{matrix} \tag{1.2.16.6}$$

1.2.17. Jólrendezés.

```
> iswellordering:=proc(X::set,R::set(ordpair)) local f,Y,P;
  f:=isordering(X,R); if not f then return f fi;
  P:=powerset(X);
  for Y in P do
    if Y<>{} then f:=minimal(Y,R); if f=NULL then return f;
  fi; fi;
  od; true; end;
```

```
iswellordering := proc(X::set, R::(set(ordpair)))
    local f, Y, P;
    f := isordering(X, R);
    if not f then
        return f
    end if;
    P := combinat:-powerset(X);
    for Y in P do
        if Y <> {} then
            f := minimal(Y, R);
            if f = NULL then
                return f
            end if
        end if
    end do;
    true
end proc
```

(1.2.17.1)

```
> X;R;iswellordering(X,R);
```

$$\{i, a, u, o, e\}$$

$$\{[a, u], [a, i], [a, e], [a, o], [u, i], [u, e], [u, o], [i, e], [i, o], [e, o]\}$$

$$true \tag{1.2.17.2}$$

1.2.18. Példa.

1.3. Függvények

1.3.1. Függvény.

```
> isfunction:=proc(f::set(ordpair),X::set,Y::set) local x,y,
  S;
  if not binrel(args) then return false fi;
  for x in dmn(f) do S:={};
    for y in rng(f) do
      if [x,y] in f then S:=S union {y} fi;
    od; if nops(S)>1 then return false fi;
  od; true; end;
```

```
isfunction := proc(f::(set(ordpair)), X::set, Y::set)                (1.3.1.1)
    local x, y, S;
    if not binrel(args) then
        return false
    end if;
    for x in dmn(f) do
        S := {};
        for y in rng(f) do
            if in([x, y], f) then
                S := union(S, {y})
            end if
        end do;
        if 1 < nops(S) then
            return false
        end if
    end do;
    true
end proc
```

```
> f:=id({a,b});isfunction(f);isfunction(f,{a,b,1});
  isfunction(f,{a,b},{1,2});f:={[a,1],[b,2],[a,2]};isfunction
  (f);
```

$$f := \{[a, a], [b, b]\}$$

$$true$$

$$true$$

$$false$$

$$f := \{[a, 2], [a, 1], [b, 2]\}$$

$$false \tag{1.3.1.2}$$

```
> isinjective:=proc(f::set(ordpair))
  isfunction(f) and isfunction(relinv(f));
  end;

  isinjective({[a,1],[b,2]});isinjective({[a,1],[b,1]});
```

$$isinjective := \mathbf{proc}(f::(set(ordpair)))$$
$$\quad isfunction(f)\ \mathbf{and}\ isfunction(relinv(f))$$
$$\mathbf{end\ proc}$$

$$true$$

$$false \tag{1.3.1.3}$$

```
> issurjective:=proc(f::set(ordpair),Y::set)
  isfunction(f) and rng(f)=Y; end;

  issurjective({[a,1],[b,1]},{1,2});
  issurjective({[a,1],[b,2]},{1,2});
```

$$issurjective := \mathbf{proc}(f::(set(ordpair)),\ Y::set)$$
$$\quad isfunction(f)\ \mathbf{and}\ rng(f) = Y$$
$$\mathbf{end\ proc}$$

$$false$$

$$true \tag{1.3.1.4}$$

```
> isbijective:=proc(f::set(ordpair),Y::set)
  isinjective(f) and issurjective(f,Y); end;

  isbijective(id({1,2,3}),{1,2,3});
  isbijective({[a,1],[b,2]},{1,2,3});
```

$$isbijective := \mathbf{proc}(f::(set(ordpair)),\ Y::set)$$
$$\quad isinjective(f)\ \mathbf{and}\ issurjective(f,\ Y)$$
$$\mathbf{end\ proc}$$

$$true$$

$$false \tag{1.3.1.5}$$

```
> f:=x->x^2;f(1);f(2);f(3);eval(f);type(f,procedure);
```

$$f := x \rightarrow x^2$$

$$1$$

$$4$$

$$9$$

$$x \to x^2$$

$$true \tag{1.3.1.6}$$

```
> f:='f';eval(f);whattype(f);
  f(1):=1;eval(f);
  f(2):=4;f(3):=8;
  f(1);f(2);f(3);f(4);
```

$$f := f$$

$$f$$

$$symbol$$

$$f(1) := 1$$

$$\mathbf{proc}()\ \mathbf{option}\ remember,\ 'procname(args)'\ \mathbf{end\ proc}$$

$$f(2) := 4$$

$$f(3) := 8$$

$$1$$

$$4$$

$$8$$

$$f(4) \tag{1.3.1.7}$$

```
> isarrowfromto:=proc(f::procedure,X::set,Y::set) local x;
  for x in X do if not f(x) in Y then return false fi; od;
  true; end;

  isarrowfromto(f,{1,2,3},{1,4,8,10});
  isarrowfromto(f,{1,2},{1,8});
```

$$isarrowfromto := \mathbf{proc}(f::procedure,\ X::set,\ Y::set)$$
$$\quad \mathbf{local}\ x;$$
$$\quad \mathbf{for}\ x\ \mathbf{in}\ X\ \mathbf{do}$$
$$\quad\quad \mathbf{if\ not}\ in(f(x),\ Y)\ \mathbf{then}$$
$$\quad\quad\quad \mathbf{return}\ false$$
$$\quad\quad \mathbf{end\ if}$$
$$\quad \mathbf{end\ do};$$
$$\quad true$$
$$\mathbf{end\ proc}$$

$$true$$

$$false \tag{1.3.1.8}$$

```
> makefunction:=proc(R::set(ordpair)) local x,y,f;
  if not isfunction(R) then return fi;
  for x in dmn(R) do
    for y in rng(R) do if [x,y] in R then f(x):=y fi; od;
```

```
   od; eval(f); end;

   f:='f';R:={[1,1],[2,4],[3,9]};f(1);f(2);f(3);f(4);
   f:=makefunction(R);f(1);f(2);f(3);f(4);
```

```
makefunction := proc(R::(set(ordpair)))
    local x, y, f;
    if not isfunction(R) then
        return
    end if;
    for x in dmn(R) do
        for y in rng(R) do
            if in([x, y], R) then
                f(x) := y
            end if
        end do
    end do;
    eval(f)
end proc
```

$$f := f$$

$$R := \{[2, 4], [3, 9], [1, 1]\}$$

$$f(1)$$

$$f(2)$$

$$f(3)$$

$$f(4)$$

```
f := proc() option remember, 'procname(args)' end proc
```

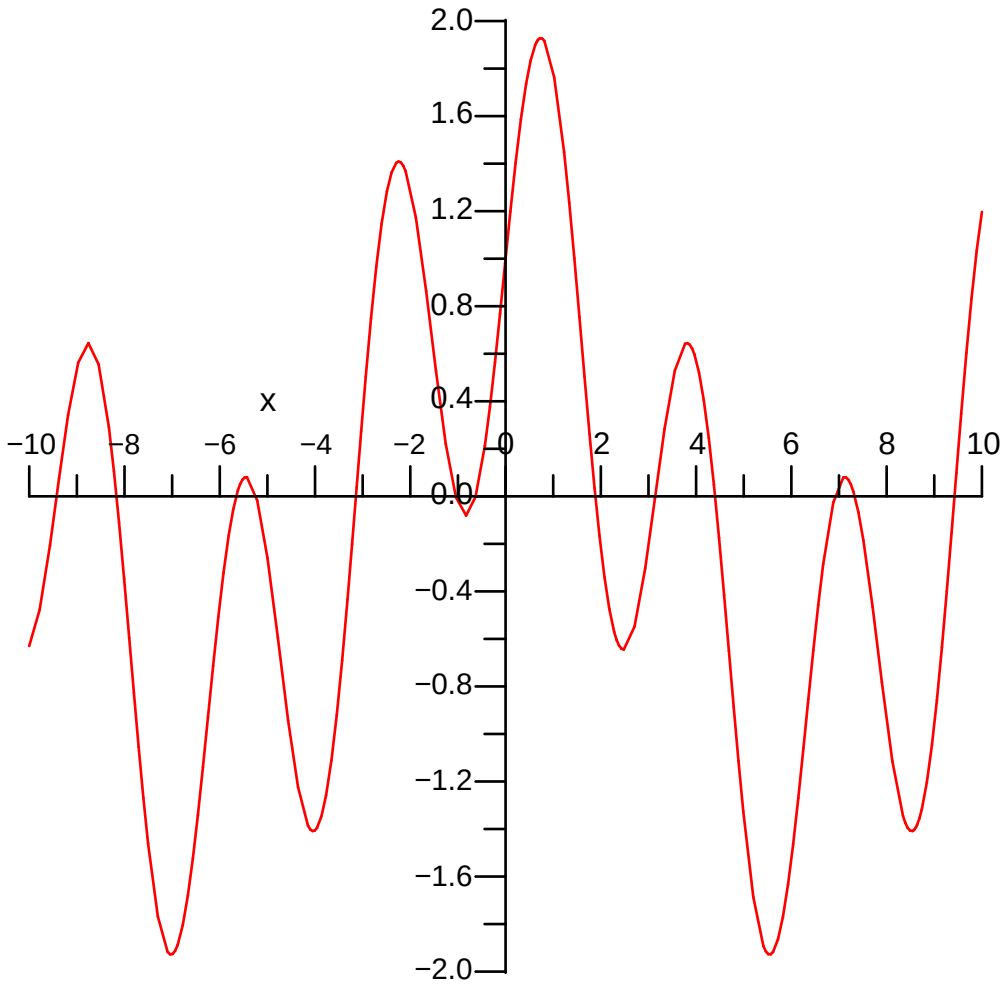
$$1$$

$$4$$

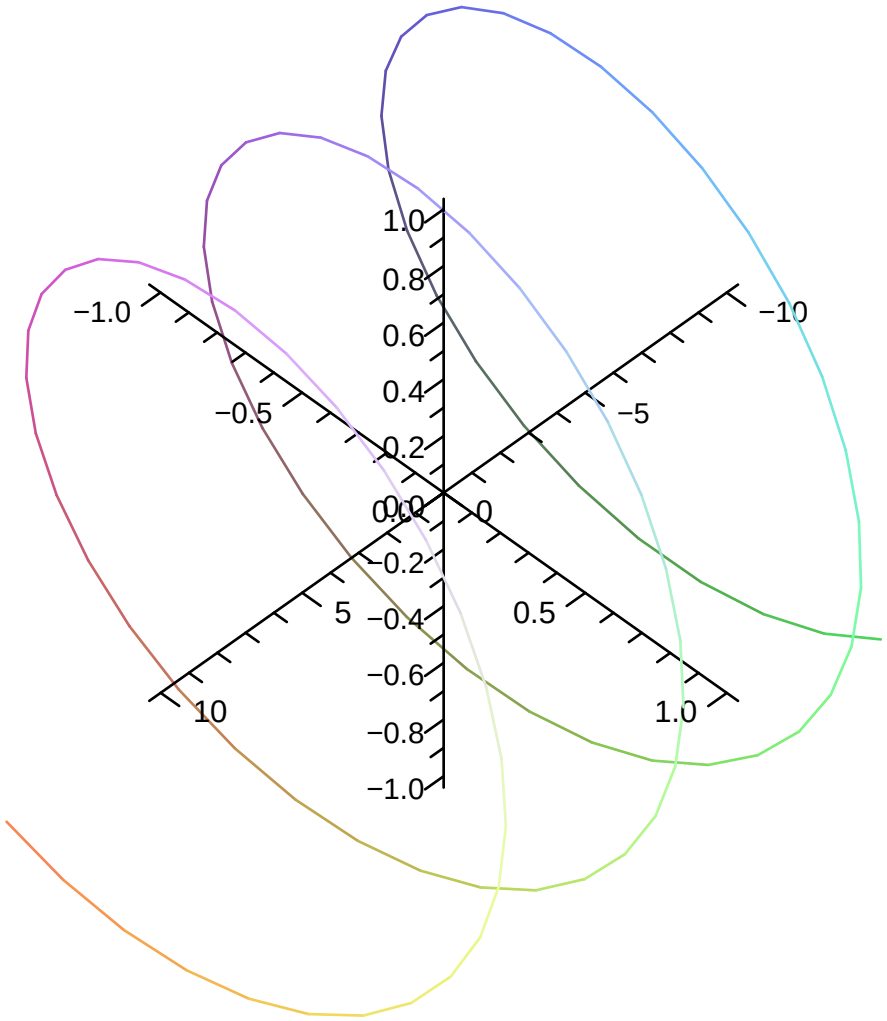
$$9$$

$$f(4) \qquad (1.3.1.9)$$

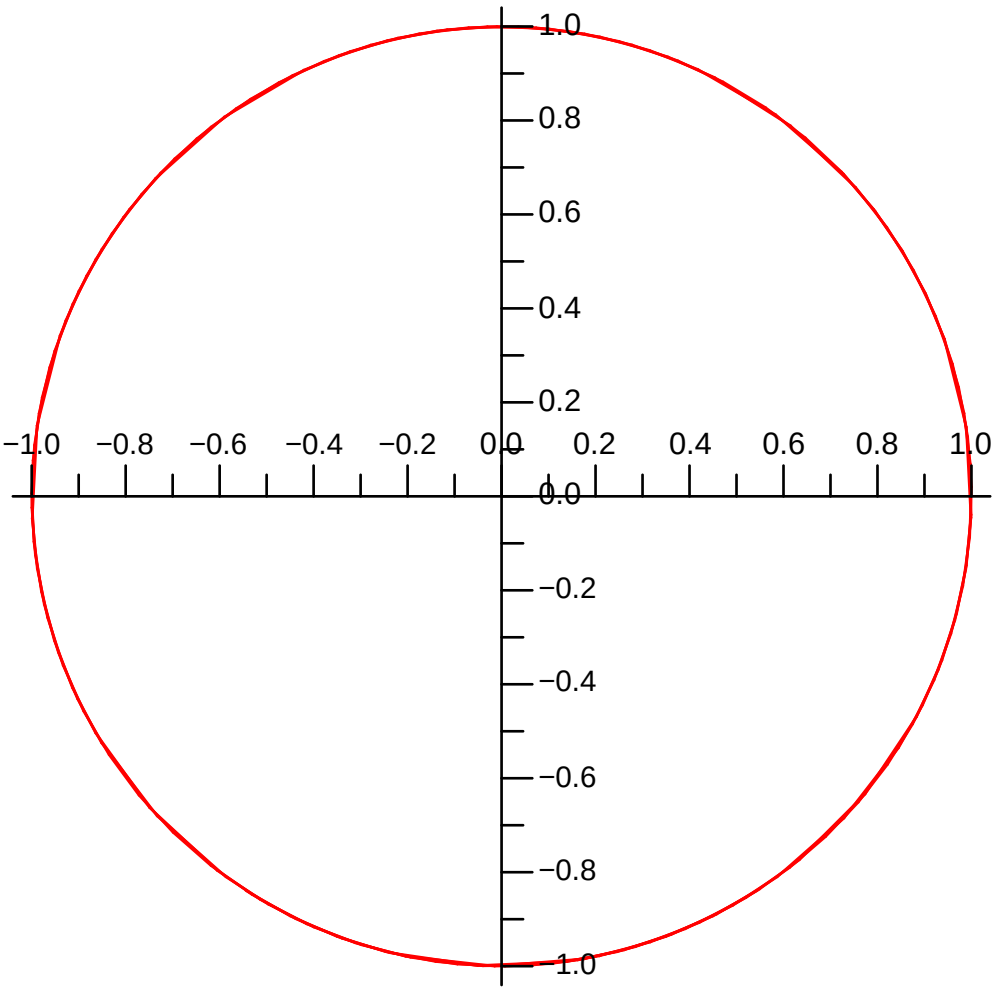
```
> plot(sin(2*x)+cos(x/2),x=-10..10);
```




```
> plots[spacecurve]([x,sin(x),cos(x)],x=-10..10,numpoints=
  100,axes=normal);
```




```
> plot([sin(x),cos(x),x=-10..10]);
```



1.3.2. Állítás.

1.3.3. Monoton függvények.

Az alábbi két függvényben R rendezés X-en, S pedig rendezés Y-on.

```
> isincreasing:=proc(f::procedure,X::set,R::set(ordpair),
  Y::set,S::set(ordpair)) local x,y;
  if not isarrowfromto(f,X,Y) then return FAIL fi;
  if not isordering(X,R) or not isordering(Y,S) then return
  FAIL fi;
  for x in X do for y in X do
    if [x,y] in R and not ([f(x),f(y)] in S or f(x)=f(y))
  then return false    fi;
  od; od; true end;

  X:={1,2,3}; XX:=bincartprod(X,X):
```

```
R:=select(x->x[1]<x[2],XX);Y:=X;S:=select(x->x[1]>x[2],XX);

f:=x->x; isincreasing(f,X,R,Y,S);

f:=x->4-x; isincreasing(f,Y,S,X,R);
```

```
isincreasing := proc(f::procedure, X::set, R::(set(ordpair)), Y::set,
    S::(set(ordpair)))
    local x, y;
    if not isarrowfromto(f, X, Y) then
        return FAIL
    end if;
    if not (isordering(X, R) and isordering(Y, S)) then
        return FAIL
    end if;
    for x in X do
        for y in X do
            if in([x, y], R) and not (in([f(x), f(y)],
            S) or f(x) = f(y)) then
                return false
            end if
        end do
    end do;
    true
end proc
```

$$X := \{1, 2, 3\}$$
$$R := \{[1, 2], [1, 3], [2, 3]\}$$
$$Y := \{1, 2, 3\}$$
$$S := \{[2, 1], [3, 1], [3, 2]\}$$
$$f := x \to x$$
$$false$$
$$f := x \to 4 - x$$
$$true \qquad (1.3.3.1)$$

1.3.4. Családok.

```
> issetfamily:=proc(Iset::set,f::procedure) local i;
  for i in Iset do if not type(f(i),set) then return false
  fi;
  od; true; end;
```

```
f:='f';f(1):={a,b};f(2):={b,c,d};
issetfamily({1,2},f);issetfamily({1,2,3},f);
```

```
issetfamily := proc(Iset::set, f::procedure)
    local i;
    for i in Iset do
        if not type(f(i), set) then
            return false
        end if
    end do;
    true
end proc
```

$$f := f$$
$$f(1) := \{a, b\}$$
$$f(2) := \{b, c, d\}$$
$$true$$
$$false \qquad (1.3.4.1)$$

1.3.5. De Morgan-szabályok.

1.3.6. Megjegyzés.

```
> `union`();
```

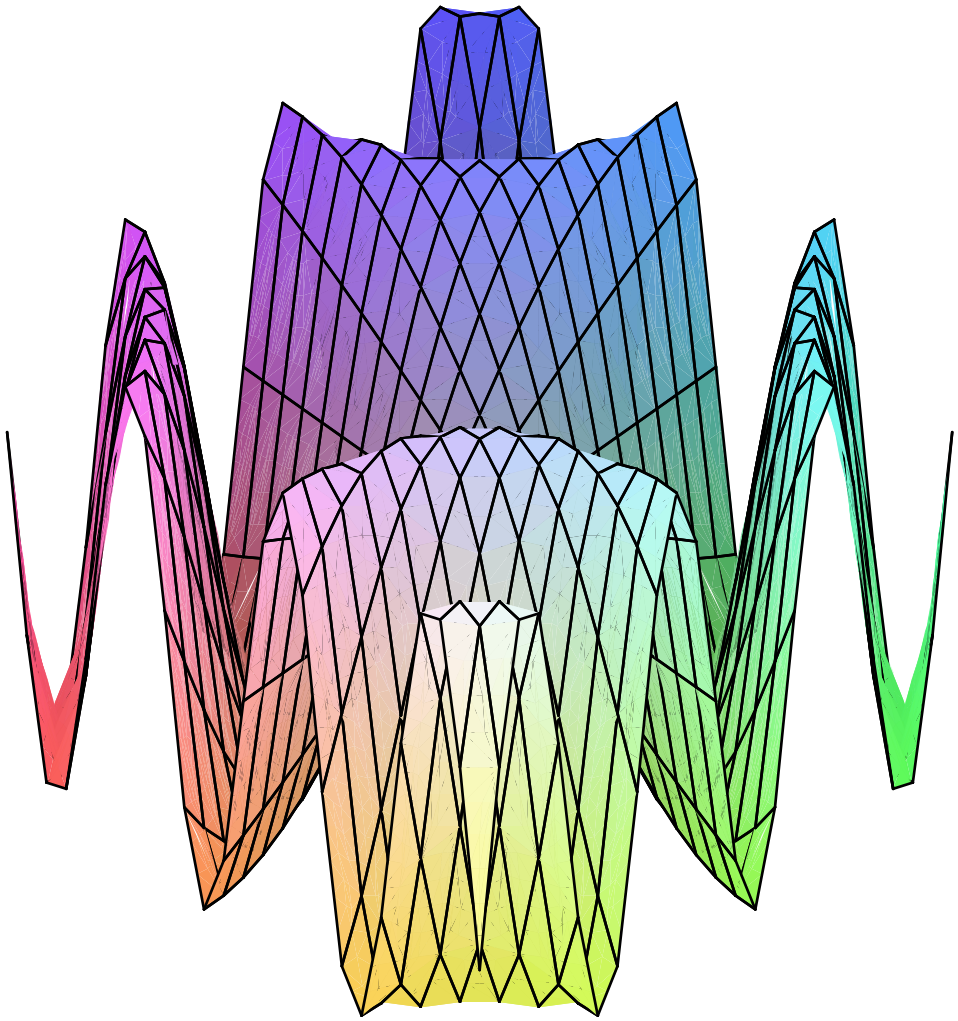
$$\{\} \qquad (1.3.6.1)$$

```
> `intersect`();
Error, invalid input: `intersect` expects 1 or more arguments,
but received 0
```

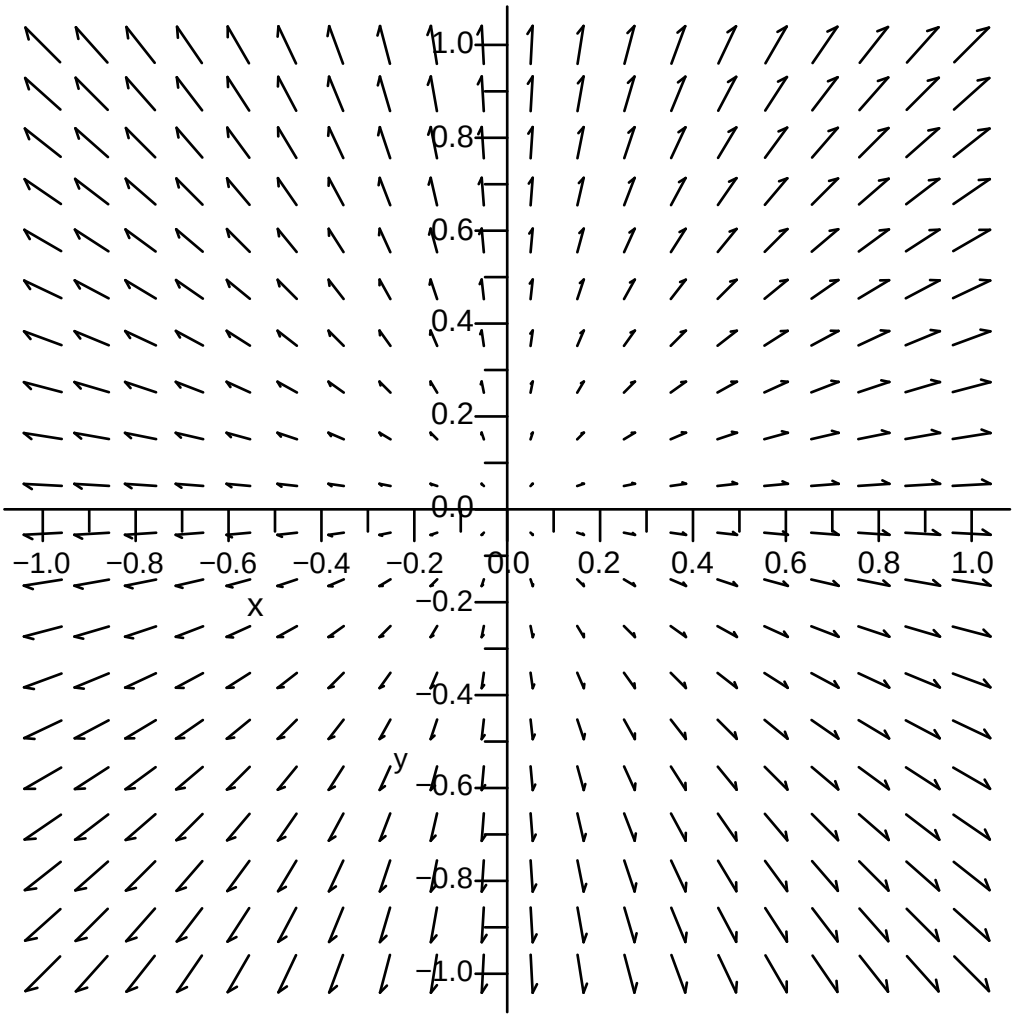
1.3.7. Tétel.

1.3.8. Többváltozós függvények.

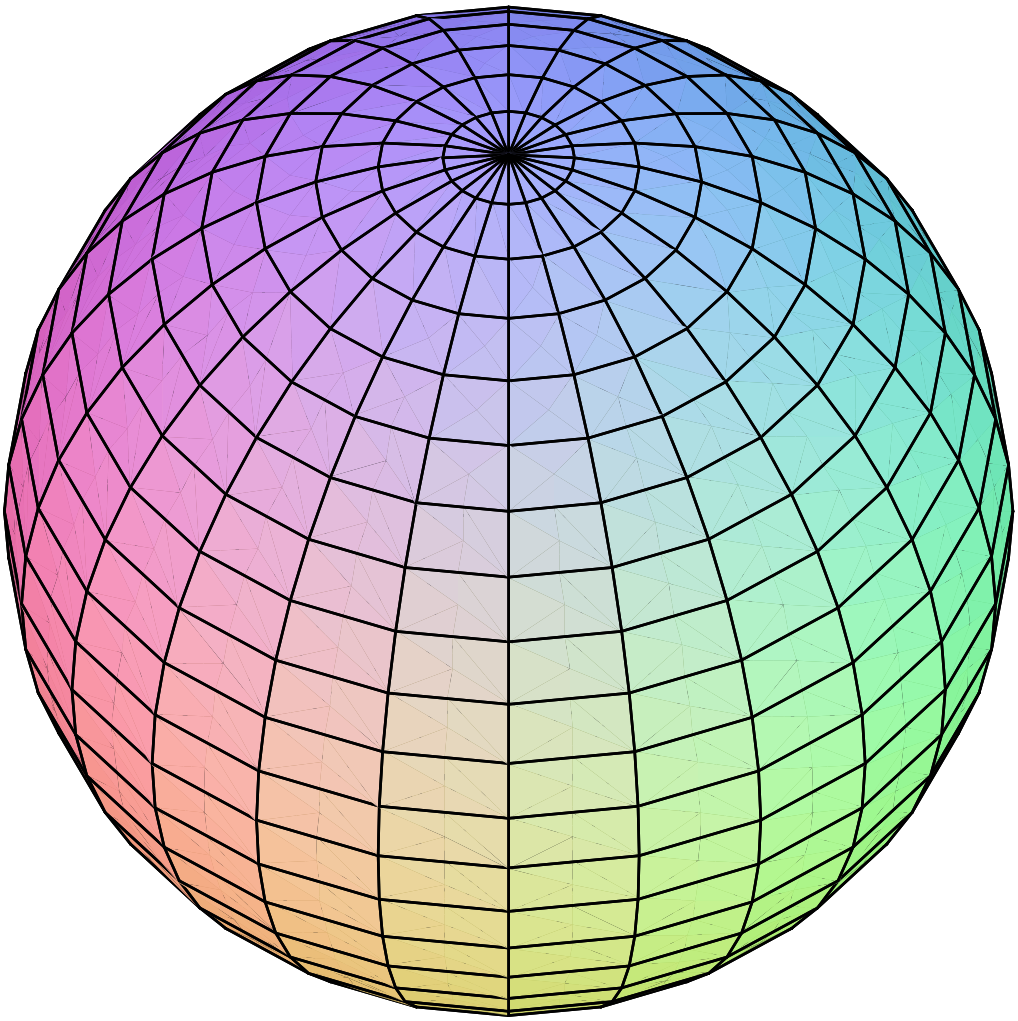
```
> plot3d(sin(x*y),x=-Pi..Pi,y=-Pi..Pi);
```



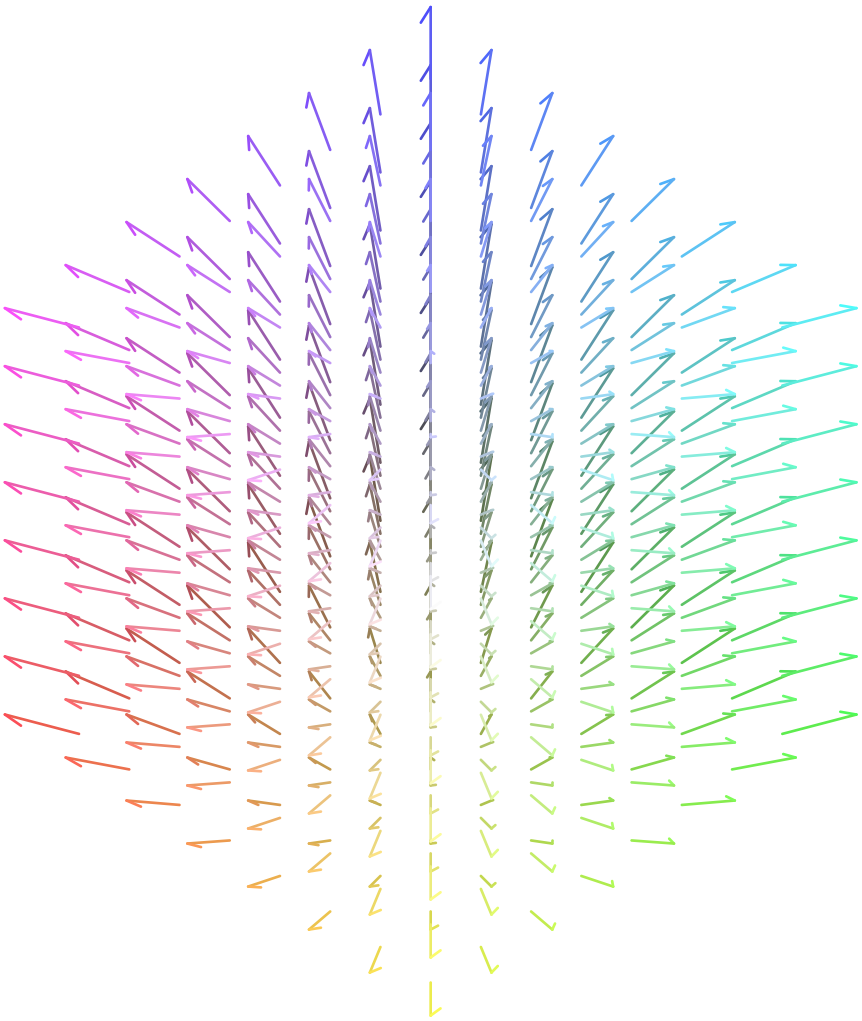
```
> plots[fieldplot]([x/(x^2+y^2+4)^(1/2),y/(x^2+y^2+4)^(1/2)],
  x=-1..1,y=-1..1);
```




```
> plot3d([cos(t)*cos(s),cos(t)*sin(s),sin(t)],t=-Pi/2..Pi/2,
  s=0..2*Pi);
```



```
> plots[fieldplot3d]([2*x,2*y,1],x=-1..1,y=-1..1,z=-1..1);
```



1.3.9. Műveletek.

```
> isbinop:=proc(X::set,f::procedure) local x,y;
  for x in X do for y in X do
    if not f(x,y) in X then return false fi;
  od; od; true end;

  f:='f';f(0,0):=0;f(0,1):=1;f(1,0):=1;f(1,1):=0; isbinop({0,
  1},f);

  isbinop({0,1,2},f);
```

```
isbinop := proc(X::set, f::procedure)
    local x, y;
    for x in X do
        for y in X do
            if not in(f(x, y), X) then
```

```
                return false
            end if
        end do
    end do;
    true
end proc
```

$$f := f$$
$$f(0, 0) := 0$$
$$f(0, 1) := 1$$
$$f(1, 0) := 1$$
$$f(1, 0) := 0$$
$$true$$
$$false \qquad (1.3.9.1)$$

```
> &+(0,0):=0;&+(0,1):=1;&+(1,0):=1;&+(1,1):=0;

  0&+1;1&+1;&+(0,1);isbinop({0,1},(x,y)->x &+ y);
```

$$0 \,\&+\, 0 := 0$$
$$0 \,\&+\, 1 := 1$$
$$1 \,\&+\, 0 := 1$$
$$1 \,\&+\, 1 := 0$$
$$1$$
$$0$$
$$1$$
$$true \qquad (1.3.9.2)$$

```
> isunop:=proc(X::set,f::procedure) local x;
  for x in X do if not f(x) in X then return false fi; od;
  true end;

  f:='f';f(0):=1;f(1):=0; isunop({0,1},f); isunop({0,1,2},f);
```

```
isunop := proc(X::set, f::procedure)
    local x;
    for x in X do
        if not in(f(x), X) then
            return false
        end if
    end do;
    true
end proc
```

$$f := f$$
$$f(0) := 1$$
$$f(1) := 0$$
$$true$$
$$false \qquad (1.3.9.3)$$

```
> &inc(0):=1;&inc(1):=0; &inc 0; &inc 1;isunop({0,1},x->&inc
  x);
```

$$\&inc(0) := 1$$
$$\&inc(1) := 0$$
$$1$$
$$0$$
$$true \qquad (1.3.9.4)$$

```
> 9();9(x);9(x,y);
```

$$9$$
$$9$$
$$9 \qquad (1.3.9.5)$$

```
> isnullop:=proc(X::set,f::procedure) f() in X end;

  f:='f';f():=1;eval(f);isnullop({0,1},f);isnullop({0,2},f);
```

$$isnullop := \mathbf{proc}(X{::}set,\ f{::}procedure)\ in(f(), X)\ \mathbf{end\ proc}$$
$$f := f$$
$$f() := 1$$
$$\mathbf{proc}()\ \mathbf{option}\ remember,\ 'procname(args)'\ \mathbf{end\ proc}$$
$$1 \in (\{0, 1\})$$
$$1 \in (\{0, 2\}) \qquad (1.3.9.6)$$

1.3.10. Példák.

```
> X:={1,2,3};P:=combinat[powerset](X);isbinop(P,(x,y)->x
  union y);
```

$$X := \{1, 2, 3\}$$
$$P := \{\{\}, \{1\}, \{2\}, \{1, 2\}, \{2, 3\}, \{3\}, \{1, 3\}, \{1, 2, 3\}\}$$
$$true \qquad (1.3.10.1)$$

1.3.11. Példák: művelet megadása táblázattal.

```
> true and true; true and false; false and true; false and
  false;
```

```
T:=table();T[true,true]:=true;T[true,false]:=false;T[false,
true]:=false;T[false,false]:=false;print(T);
```

$$true$$
$$false$$
$$false$$
$$false$$
$$T := table([\,])$$
$$T_{true,\ true} := true$$
$$T_{true,\ false} := false$$
$$T_{false,\ true} := false$$
$$T_{false,\ false} := false$$
$$table([(true, false) = false, (false, true) = false, (true, true) = true, (false, false) = false]) \quad (1.3.11.1)$$

```
> X:={true,false};XX:=bincartprod(X,X);

  map(x->cat("",x[1]," and ",x[2],"=",x[1] and x[2]),XX);
  map(x->cat("",x[1]," or ",x[2],"=",x[1] or x[2]),XX);
  map(x->cat("",x[1]," implies ",x[2],"=",x[1] implies x[2]),
  XX);
  map(x->cat("",x[1]," iff ",x[2],"=",evalb(x[1]=x[2])),XX);
  map(x->cat("",x[1]," xor ",x[2],"=",x[1] xor x[2]),XX);
```

$$X := \{false, true\}$$
$$XX := \{[false, true], [true, false], [true, true], [false, false]\}$$
{"false and true=false", "true and false=false",
"false and false=false", "true and true=true"}
{"true or true=true", "true or false=true", "false or false=false",
"false or true=true"}
{"false implies true=true", "true implies true=true",
"true implies false=false", "false implies false=true"}
{"false iff true=false", "true iff false=false", "true iff true=true",
"false iff false=true"}
{"true xor false=true", "false xor false=false", "false xor true=true", (1.3.11.2)
"true xor true=false"}

1.3.12. Műveletek függvényekkel.

```
> f:=x->x^2; g:=x->x^3; (f*g)(2);(f*g)(3);(f/g)(2);(f/g)(0);
  (g/f)(0);
```

$$f := x \to x^2$$

$$g := x \to x^3$$

$$32$$

$$243$$

$$\frac{1}{2}$$

```
Error, numeric exception: division by zero
Error, numeric exception: division by zero
```

1.3.13. Példák.

```
> f:=[true,true,false,false];g:=[true,false,true,false];zip(
  (x,y)->x and y,f,g);
```

$$f := [true, true, false, false]$$

$$g := [true, false, true, false]$$

$$[true, false, false, false] \tag{1.3.13.1}$$

1.3.14. Példák.

```
> f:=x->x^2; g:=x->x^3;
  (f+g)(2);(f+g)(3);
  (f*g)(2);(f*g)(3);
  (f/g)(2);(f/g)(0);(g/f)(0);
```

$$f := x \to x^2$$

$$g := x \to x^3$$

$$12$$

$$36$$

$$32$$

$$243$$

$$\frac{1}{2}$$

```
Error, numeric exception: division by zero
Error, numeric exception: division by zero
```

1.3.15. Művelettartó leképezések.

```
> ishom:=proc(phi::procedure,X::set,f::procedure,Y::set,
  g::procedure)
  local x,y;
  if not isarrowfromto(phi,X,Y) then return FAIL fi;
  if not isbinop(X,f) then return FAIL fi;
  if not isbinop(Y,g) then return FAIL fi;
```

```
  for x in X do for y in X do
    if phi(f(x,y))<>g(phi(x),phi(y)) then return false fi;
  od; od; true end;

  X:={true,false};Y:=X;ishom(x->x,X,(x,y)->x and y,Y,(x,y)->x
  or y);

  ishom(x-> not x,X,(x,y)->x and y,Y,(x,y)->x or y);
```

```
ishom := proc(phi::procedure, X::set, f::procedure, Y::set,
    g::procedure)
    local x, y;
    if not isarrowfromto(phi, X, Y) then
        return FAIL
    end if;
    if not isbinop(X, f) then
        return FAIL
    end if;
    if not isbinop(Y, g) then
        return FAIL
    end if;
    for x in X do
        for y in X do
            if phi(f(x, y)) <> g(phi(x), phi(y)) then
                return false
            end if
        end do
    end do;
    true
end proc
```

$$X := \{false, true\}$$
$$Y := \{false, true\}$$
$$false$$
$$true \qquad (1.3.15.1)$$

1.3.16. Példa.

```
> a:='a':x:='x':y:='y':a^(x+y);expand(%);
```

$$a^{x+y}$$
$$a^x a^y \qquad (1.3.16.1)$$