Quick index
main
eev
maths
blogme
dednat4
littlelangs
PURO
(MD, GA,
BE, etc)
(Chapa 1)

emacs
lua
(la)tex
fvwm
tcl
forth
icon
debian
irc
contact
☿

Matemática Discreta - 2012.1

```
Horários do curso em 2012.2:
  4ªs e 6ªs 16-18, sala 11.
  Veja: http://angg.twu.net/2012.2.html
Material dos semestres anteriores:
  http://angg.twu.net/2011.2-MD.html          (página principal)
  http://angg.twu.net/MD/MD-2011.2.pdf        (versão para impressão)
  http://angg.twu.net/MD/MD-2011.2-tudo.pdf   (folhas manuscritas)
  http://angg.twu.net/MD/MD-2011.2-tudo.djvu  (idem, em djvu)
  http://angg.twu.net/2011.1-MD.html          (página principal)
  http://angg.twu.net/MD/MD-2011.1.pdf        (versão para impressão)
  http://angg.twu.net/MD/MD-2011.1-tudo.pdf   (folhas manuscritas)
  http://angg.twu.net/MD/MD-2011.1-tudo.djvu  (idem, em djvu)


Versão para impressão desta página:
  http://angg.twu.net/MD/2012.1-MD.pdf
(Pode estar desatualizada!)


Vamos usar o livro do Scheinerman,
alguns capítulos do livro da Judith Gersting,
e o primeiro capítulo do Hopcroft/Ullman/Motwani.
```

O código-fonte da Calculadora da Gabriela está aqui:

```
http://angg.twu.net/dednat5/gabriela.lua.html
```

A Ana Isabel está dando o curso dela de modo bem diferente, mas vale muito a pena olhar o material do curso dela (no Moodle):

```
http://softwarelivre.uff.br/esl/
http://softwarelivre.uff.br/esl/course/view.php?id=51  MD Bel 2011.2
http://softwarelivre.uff.br/esl/mod/forum/discuss.php?d=209 Gabriela
```

Plano de aulas / resumo do que já aconteceu:

```
1ª aula (07/mar): não teve aula.
2ª aula (09/mar): Avisei que este curso tem um problema (e os livros
  também!): a gente precisa aprender a usar uma linguagem - que vamos
  chamar de "matematiquês formal" - só que os livros raramente
  formalizam ela o suficiente.
    Vamos definir _precisamente_ um "fragmento" do matematiquês
  formal... Isto quer dizer que vamos saber _exatamente_ o que é uma
  expressão válida, o que é uma definição coreta, etc... e vamos ver
  uma _implementação_ deste fragmento.
    Um exemplo de expressão:
      ∀ x ∈ {2,3,4}. x <= 3
  como "pronunciar" isto?
  "Para todo x no conjunto {2,3,4} é verdade que x<=3."
      ∃ x ∈ {2,3,4}. x <= 3
  "Existe x no conjunto {2,3,4} tal que x <= 3."
  Obs: o "." tem uma pronuncia diferente no "∀" e no "∃".
  Fiz uma enquete entre os alunos pra ver que valor eles achavam que
  uma expressão tinha:
    (∀x∈{2,3,4}. x<=3) =? {2,3}  (5 votos)
    (∀x∈{2,3,4}. x<=3) =? F      (<- esse era o certo)
```

  (∀x∈{2,3,4}. x<=3) =? V
A gente costuma marcar com "?" um "=" que estamos discutindo se é
verdade ou não (ou que ainda não temos certeza).
Valores
=======
Booleanos: V e F (em negrito, exceto quando estivermos com pressa)
Números: 0, 1, 2, 3, ... ∈ N
         -1, -2, -3, ... ∈ Z
         2/3, -7/4, ... ∈ Q
         pi, e, sqrt(3)... ∈ R   (varios "tipos")
Neste curso quase so' vamos usar inteiros.
Repare que escrevemos "5∈N".
Temos alguns conjuntos predefinidos: N, Z, Q, R, C, vazio.
Cada um tem um símbolo e uma pronúncia - o conjuntos dos naturais, o
dos inteiros, o dos racionais, o dos reais, o dos complexos, o
conjunto vazio.
Obs ***MUITO*** importante: se dois objetos tem tipos diferentes,
eles sao diferentes. 2 e' um numero, {2} e' um conjunto, o "tipo" do
2 é numero, o "tipo" do {2} é conjunto, e 2!={2}.
Considerar 2={2} é ***erro conceitual gravissimo***, e isto está até
mencionado na folha de regras:
  http://angg.twu.net/LATEX/2011-1-GA-regras.pdf
[---- falta passar o resto a limpo ----]
Como escrever contas, respostas, etc?
Existe uma folha - tem link pra ela na pagina do curso - que tem uma
serie de regras sobre "como escrever"
Comparacoes entre matematiques formal e outras linguages Em C os
booleanos sao representados por numeros - 0 e' falso e qqer outro
valor e' verdadeiro. (o verdadeiro tipico é o -1).
Em matematiques formal 0 e 1 sao numeros e nao sao booleanos.
0=F e' faso.
Revisao do ops booleanas - pronuncia das ops
Exercicios de revisao
Quando P=V e Q=F, qual e' o valor de P&Q ou (Q->(P->Q))?
Calcule a tabela para as expressoes:
P Q (P&Q)->P P->(P&Q)
P Q R (P&Q)vR P&(QvR)
Truque: se pensarmos que F=0 e V=1 então P&Q=P·Q=min(P,Q), ..., P->Q =
<=
As operações &, v, ->, nt são "operações" no mesmo sentido que +, -,
·, /, ... são operações, só que _ainda_ não temos muita familiaridade
com &, v, ->, nt...
Um objetivo _básico_ do curso é a gente aprender muito sobre &, v, ex,
fa, ...
O objetivo _real_ (mais avançado) do curso é a gente aprender a
definir operacoes novas e a descobrir que propriedades elas tem.
Algebra:
  33·237·(44-40-4) = 0
Se x in {-3,-2,-1,0,1,2,3} e y in {-3,-2,-1,2,3,4,5,6}, será que
3(x+2) <= y+4 é sempre verdade?
Terminologia
============
Podemos tentar responder essa pergunta pelo "metodo burro" - que
corresponde ao programa de computador mais simples - ou a gente pode
tentar algo mais esperto. "Metodo burro" = "Forca bruta" - when in
doubt use brute force
A gente muita vezes vai começar usando forca bruta - pra ver se
aparece uma ideia melhor.
Fizemos uma tabelona
Estamos procurando o pior caso - em que 3(X+2) é o maior possivel e
y+4 é o menor possivel
Quando x=3 o 3(x+2) vai assumir o maior valor possivel, e 3(x+2) = 3·5
= 15
Quando y=-3 o y+4 vai ser o menor possível, i.e., 1. (... -> F).
exercicio (pra agora E para casa): para cada uma das expressoes

```
  logicas abaixo tente descobrir se ela é sempre verdadeira ou nao - e
  pra procurar um caso no qual ela seja falsa use a idéia de "procurar o
  pior caso".
    a) P&Q -> PvQ
    b) PvQ -> P&Q
    c) (P->Q)->(Q->R)
    d) P->((P&Q)&(PvQ))
  Fiz um desenhao pro caso do c, avisando que não há uma notação padrão
  para aquilo e que normalmente os livros fazem isso em portugues.

3ª aula (14/mar):
  estavamos vendo algumas operacoes booleanas: e, ou, ->, nt
  agora vamos ver os quantificadores: fa, ex.
  obs: os livros comecam com quantificacao sobre conjuntos _infinitos_.
  Pra gente este tipo de quantificacao vai ser mais basico:
    Fa a in A. P(a)
         \--/
      o "in A" e' a parte extra que os livros demoram pra introduzir.
  Exemplo: Fa a in {2,3,4}
  Existe um modo explicito de calcular o valor de uma expressao
  destas - e o melhor modo da gente conseguir visualizar ("direto")
  quando uma sentença destas é verdade é ter prática em fazer estas
  contas explicitamente.
  Entao: Fa a in {2,3,4}. a<=3
         isto e uma espe  esta expr vai ser "executada"
         cie de loop      com a=2, a=3, a=4...
  Truque: expressoes com "Fa" viram varias copias da expressao depois do
  ".", conectadas por "&".
   ...  -^^^^-> (2<=3)&(3<=3)&(4<=3)
   Vamos usar setas sinuosas ("squiggly arrows") pra indicar
   "reducao"... Obs: reducao e' um termo tecnico, que mais tarde vamos
   formalizar... Vamos ter varias regras de reducao, e (teorema!)
   todas os "caminhos de redução" chegam ao mesmo resultado final.
   Nossa 1ª regra de redução: Fa a in {a1,...,an}.P(a) -> P(a1)&...P(an)
   (obs: isto é o modo formal e quase ilegível de generalizar o
   exemplo acima)
   A 2a regra de reducao é : toda expressão da forma a op b, onde a e
   b sao numeros ou booleanos, pode ser trocada pelo seu resultado
   Exemplo: caminhos de reducao em Fa a in {2,3,4}. a<=3
   (exemplo ruim)
   [cav] formalizar todas estas regras de reducao em todo detalhe dá
   um trabalhao, e só vamos fazer isto _bem_ mais tarde...
   Por enquanto a gente vai improvisar um pouco e fazer contas até ter
   um pouco de prática.
   Exercicios (segustao: facam em grupo) calculem:
    a) fa a in {-4,5}. a<=a^2
    b) fa a in {-1,0,1,2}. a=a^2
    c) fa a in {2,3}. fa b in {3,4}. a<b
  c) caminhos de redução; podemos reduzir subexpressoes; exemplo
  Obs: normalmente existem _muitos_ caminhos de redução, e - teorema
  dificil - todos dão o mesmo resultado!
  A regra de reducao para o Ex é a mesma que para o Fa, só que com v
  no lugar de &...
  Exercicio: calculem Ex y in {5,6}. 3<y --> ()v()
  Mais regras de reducao
  ======================
    Se a e b sao inteiros e a<=b entao {a,..,b}-{a,a+1,...,b}
    exemplo: {2,...5}->...      [obs: esta regra é temporaria]
    Exercico: calcule
      ex k in {0,..5}. 2k=5
      ex a in {2,...,4}. ex k in {0,..5}. ak=5
      ex a in {2,...,5}. ex k in {0,..6}. ak=6
  obs: em geral "..."s são ambiguos.
  Exemplo: o que {2/3, ..., 4} quer dizer?
  Ideia 1: {2/3, 1, 2, 3, 4}
```

```
  Ideia 2: {2/3, 1, 4/3, 5/3, ..., 4}
  Ideia 3: {2/3, 5/3, 8/3, ..., 4}
     ex a in {2,...,n-1}. ex k in {0,..n}. ak=n
    é verdade para n composto e falso para n primo.
  para casa: ler a secao 1 do cap 1 do scheinerman.
  a divide b ---> ex k in {0,...,b}.ka=b
  ex a in 2,...,4. a div 5
4ª aula (16/mar):
  No fim da aula passada nós vimos que
    C(n) := ∃a∈{2,...,n-1}.∃b∈{0,...,n}.ab=n
    \-----/
      def
  é uma expressão que testa de n é composto (i.e., não-primo)...
  Sabemos fazer uma tabela:
     n  C(n)
    --------
     3   F
     4   V
     5   F
     6   V
     7   F
     8   V
     9   V
    10   V
  Como calcular C(4)?
  Pela definição,
    C(4) = ∃a∈{2,...,4-1}.∃b∈{0,...,4}.ab=4
  (Repare que pegamos a definição e trocamos todos os "n"s por 4 -
  isso é um "caso particular" da definição)...
    C(4) = ... = V
  Seria bom voces conseguirem visualizar este tipo de expressao
  mesmo para n grande...
  Exemplo:
    C(15) = V (fiz todas as contas, com reticencias)
    C(17) = F (direto)
  Definicoes
  **********
    Quando dizemos C(n) := ...
    a gente está atribuindo um significado preciso pra expressoes
    como C(3), C(4),...
    Ou (vendo as coisas por um ponto de vista computacional)
    estamos definindo uma regra de "reducao" para expressoes como
    C(3), C(4), ..., que nos permitem calcular valores.
    A regra de reducao nos permite dar um passo a partir de
    (por exemplo) C(17) - mas não garante que a gente vá chegar
    num resultado final.
      Exemplos: C(2/3) = ?
                C(-5)  = ?
      O que é {2, ..., 2/3}?
      O que é {0, ..., -5}?
      AInda não definimos!
    (1) Matemática é _toda_ feita de definições.
    (2) Programação também.
    (3) Várias definições vão ser feitas parcialmente em portugues.
  Exercicio (resto da aula)
    De definicoes formais de "par", "impar", "divisivel" e "primo"
    e teste-as.
  Obs: faca definicoes sem conjuntos infinitos.
  (Dica: se voces estiverem em duvida _escrevam alguma ideia_ e discutam
  com os colegas).
  Obs: voces sabem testar definicoes.
  Def: A(n) := (2n=6)
  Será que A(n) é verdade exatamente quando n é par?
  Será que A(n) é verdade exatamente quando n é ímpar?
  Será que A(n) é verdade exatamente quando n é primo?
```

```
  Dica: se vocês fizerem uma definição formal _qualquer_ vocês sabem
  fazer uma tabela pra ela.
  Defs:
    A(n) := (2n=6)
    F(n) := (n∈{0,3,5,10}∧n>=10)
    B(n) := (n∈{0,1,2,3,...,10}∧2n∈{0,4,8,12,16,20})
    P(n) := (∃c∈{1,...,10}.2c=n)
    Q(n) := (∃c∈{1,...,n}.2c=n)
    Fizemos uma tabelona para n=0..24 para n, A(n), n é par, n é impar,
    n é primo, F(n), B(n), P(n), Q(n).
    Pra casa: escrevam várias definições que pareçam razoáveis para "n é
    ímpar".
      I_1(n) := (∃c∈{1,...,n}. 2c=n+1)
      I_2(n) := (∃c∈{1,...,n}. 2c+1=n)
      I_3(n) := (∃c∈{0,...,n}. 2c+1=n)
      I_4(n) := (∃c∈{-n,...,n}. 2c+1=n)

5ª aula (21/mar): Na aula anterior chegamos a estes 4 candidatos a
  definições formais pra "ímpar":
      I_1(n) := (∃c∈{1,...,n}. 2c=n+1)
      I_2(n) := (∃c∈{1,...,n}. 2c+1=n)
      I_3(n) := (∃c∈{0,...,n}. 2c+1=n)
      I_4(n) := (∃c∈{-n,...,n}. 2c+1=n)
  Problema: será que I_1, I_2, I_3 e I_4 são equivalentes?
  Isto é, será que ∀k∈Z.I_1(k)=I_2(k)=I_3(k)=I_4(k)?
  Duas expressões _não são equivalentes_ quando existe alguma situação
  na qual elas dão resultados diferentes - mas ainda não temos métodos
  pra mostrar que duas expressões lógicas são equivalentes! No colégio
  nós aprendemos que (x+1)(x-1) e (x^2-1) são equivalentes... como
  conseguir algo parecido para expressões lógicas?
    Fizemos uma tabela para I_1, I_2, I_3, I_4; vimos que como ainda
  não sabemos o que é, por exemplo, {0,...,-2} nós não sabemos
  calcular I_1(k), ..., I_4(k) para alguns valores de k∈Z; fazendo
  umas contas explícitas vimos que I_1(1)=I_3(1)=I_4(1)=V, I_2(1)=F.
    Ainda não _definimos_ o que é {0,...,-2}, mas temos alguma noção
  (intuitiva) do que são definições razoáveis - todo mundo achou que
  {0,...,-2} = {22,33,44} não era uma definição razoável.
  O que faz uma definição ser "razoável"?
  Passei um exercício pra gente _começar_ a entender o que seria uma
  resposta pra isto. Suponha que A={1,2,3} e B={2,3,4}.
  Sejam E_1 = ∃x∈A∩B.P(x),
        E_2 = ∃x∈A∪B.P(x),
        E_3 = ∀x∈A∩B.P(x),
        E_4 = ∀x∈A∪B.P(x),
  Exercício (em grupo): tentem "calcular" E_1, E_2, E_3, E_3 o máximo
  possível _sem usar a definição de P(x)_. Depois tentem calcular:
        E'_1 = (∃x∈A.P(x))∨(∃x∈B.P(x))
        E'_2 = (∃x∈A.P(x))∧(∃x∈B.P(x))
        E'_3 = (∀x∈A.P(x))∨(∀x∈B.P(x))
        E'_4 = (∀x∈A.P(x))∧(∀x∈B.P(x))
  Quais destas 8 expressões são equivalentes?
  (A resposta disto vai nos dar uma noção de que regras de
  simplificação devem valer...)
  Resp (parcial - primeiros passos):
        E_1 =      P(2)∨P(3)
        E_2 = P(1)∨P(2)∨P(3)∨P(4)
        E_3 =      P(2)∧P(3)
        E_4 = P(1)∧P(2)∧P(3)∧P(4)
  e se soubermos P - isto é, se tivermos uma definição para P -
  podemos calcular E_1, E_2, E_3 e E_4... Podemos fazer uma tabela com
  os "valores de P" (definições para P) e os resultados de E_1, E_2,
  E_3, E_4. Exemplo: se P(k) = (k>=2) então
        E_1 =       (2>=2)∨(3>=2)
        E_2 = (1>=2)∨(2>=2)∨(3>=2)∨(4>=2)
        E_3 =       (2>=2)∧(3>=2)
```

E_4 = (1>=2)∧(2>=2)∧(3>=2)∧(4>=2)
Exercício (cav/2): mostre que E_1, E_2, E_3 e E_4 _não são
equivalentes_ - por exemplo, para mostrar que E_1 e E_2 não são
equivalentes precisamos encontrar um P (uma definição para P!) tal
que E_1 != E_2.
Tabela:

P(k)	P(1)	P(2)	P(3)	P(4)	E_1	E_2	E_3	E_4
(k>=2)	F	V	V	V	V	V	V	F
(2k=2)	V	F	F	F	F	V	F	F
(k=1)	V	F	F	F	F	V	F	F
(k<1)	V	F	F	F	F	V	F	F
(k!=2)	V	F	V	V	V	V	F	F
(k!=3)	V	V	V	V	V	V	F	F
(k!=2)∧(k!=3)	V	F	V	V	F	V	F	F
(k!=0)	V	V	V	V	V	V	V	V

E_1 e E_2 não são equivalentes porque quando P(k) = (2k=2) temos
E_1=F e E_2=V;
E_1 e E_3 não são equivalentes porque quando P(k) = (k!=2) temos
E_1=V e E_3=F; etc...
Pra casa: estendam esta tabela pra incluir uma coluna por E'_1, uma
pro E'_2, uma pro E'_3, uma pro E'_4.

6ª aula (23/mar): estamos continuando a tentar entender o que são
"expressões equivalentes"... usamos as mesmas expressões E_1, ...,
E'_4 da aula passada,
E_1 = ∃x∈A∩B.P(x),
E_2 = ∃x∈A∪B.P(x),
E_3 = ∀x∈A∩B.P(x),
E_4 = ∀x∈A∪B.P(x),
E'_1 = (∃x∈A.P(x))∨(∃x∈A.P(x))
E'_2 = (∃x∈A.P(x))∧(∃x∈A.P(x))
E'_3 = (∀x∈A.P(x))∨(∀x∈A.P(x))
E'_4 = (∀x∈A.P(x))∧(∀x∈A.P(x))
Estamos usando:
A = {1,2,3}
B = {2,3,4}
mas isto pode mudar depois.
Pedi pros alunos simplificarem as expressões E_1, ..., E'_4.
Usamos P1:=P(1), P2:=P(2), etc, pra notação ficar menos pesada,
E_1 = P2∨P3
E_2 = P1∨P2∨P3∨P4
E_3 = P2∧P3
E_4 = P1∧P2∧P3∧P4
E'_1 =
Pra mostrar que E_1 e E_2 nao sao equivalentes temos que encontrar
OU uma situacao na qual E1 =V e E2=F
Ou uma situacao na qual E1=F e E2=V.
Mas uma destas alternativas vai ser impossivel. Qual? Porque?
Repare que "E1=V e E2=F" é a mesma coisa que E_1->E_2 = F.
Ou seja, queremos encontrar
OU uma situacao na qual E1->... = F
Ou uma situacao na qual E@->... =F.
Dá pra _provar_ que E1 -> E2 é sempre verdade -
ou seja, quando E_1=V temos E_2=V,
ou "E2 é mais verdade que E1",
ou "E1<=E2" (se vemos E1, E2 como 0, 1).
A idéia "->"vai ser importantíssima em lógica - podemos "ordenar"
todas as expressões com "->"!
Exercicio (cav*4, em grupo):
acabamos de nos convencer de que E1->E2 (argumento do Pedro Paulo)
e que E2-/->E1 (por um contra-exemplo - P(k) = k=1)
Temos 8 expressões - E_1, ..., E'_4. Tentem ordená-las - algo como:
F ---> A ... V
<-/-
Obs: "A->B" quer dizer "A->B é sempre verdade"

```
     "A-/->B" quer dizer "A->B nem sempre é verdade"
Hip do grupo 1:
  F -> E3' E4 E3 E1 E2' E4' E2 E1' V
Grupo 2:
  F -> E4 E3' E2' E1 E1' E4' E2 V
Pra casa: tentem ordenar as 8 expressões (e mais o V e o F),
tentem provar as setas -/-> (fácil) e as setas -> (difícil).
Dica: alguns "->"s são equivalentes!
```

7ª aula (28/mar):

```
No problema da aula passada, as definições eram:
    A   = {1,2,3}
    B   =   {2,3,4}
   E_1  = ∃x∈A∩B.P(x)              =    P2∨P3
   E_2  = ∃x∈A∪B.P(x)              = P1∨P2∨P3∨P4
   E_3  = ∀x∈A∩B.P(x)              =    P2∧P3
   E_4  = ∀x∈A∪B.P(x)              = P1∧P2∧P3∧P4
   E'_1 = (∃x∈A.P(x))∨(∃x∈A.P(x))  = (P1∨P2∨P3)∨(P2∨P3∨P4)
   E'_2 = (∃x∈A.P(x))∧(∃x∈A.P(x))  = (P1∨P2∨P3)∧(P2∨P3∨P4)
   E'_3 = (∀x∈A.P(x))∨(∀x∈A.P(x))  = (P1∧P2∧P3)∨(P2∧P3∧P4)
   E'_4 = (∀x∈A.P(x))∧(∀x∈A.P(x))  = (P1∧P2∧P3)∧(P2∧P3∧P4)
(a coluna da direita tem simplificações)
E a solução é:
       E4'                              E1'
       ||                               ||
  F -> E4 -> E3' -> E3 -> E1 -> E2' -> E2 -> V
Uma tabela (feita pela calculadora da Gabriela):
```

P(k)		P(1)	P(2)	P(3)	P(4)	F	E_4	E'_3	E_3	E_1	E'_2	E_2	T
F	:	F	F	F	F	F	F	F	F	F	F	F	T
k>=4	:	F	F	F	T	F	F	F	F	F	F	T	T
k==1|k==4	:	T	F	F	T	F	F	F	F	F	T	T	T
k>=3	:	F	F	T	T	F	F	F	F	T	T	T	T
k==2|k==3	:	F	T	T	F	F	F	F	T	T	T	T	T
k>=2	:	F	T	T	T	F	F	T	T	T	T	T	T
T	:	T	T	T	T	F	T	T	T	T	T	T	T

```
Obs: o código está em:
  http://angg.twu.net/dednat5/gabriela-app.lua.html#grids-tests
                 (find-dn5 "gabriela-app.lua"    "grids-tests")
Vi que os alunos sabiam completar tabelas como a acima, e escrevi no
quadro estas duas afirmações:
   (I) se 6|k então k é par
  (II) se 3|k então k é par
Obs: "a|b" é notação para "a divide b", e a definição formal que
vamos usar pra a|b (por enquanto) é:
  (a|b) := (∃k∈Z. ak=b)
Em matemática a gente vê sempre afirmações como "6|k->2|k", que só
fazem sentido quando sabemos o valor de k... Quando o valor de k
está "livre" isto quer dizer:
  ∀k∈Z. 6|k->2|k
  \---/
  implícito pelo contexto
Repare que se k fosse um valor fixo, p.ex. k=30, "6|k->2|k" quereria
dizer "6|30->2|30", que sabemos calcular - dá V->V, que é V.
Mas
  ∀k∈Z. 6|k->2|k
quer dizer
  ∀k∈{...,-2,-1,0,1,2,3,4,5,6,...}. 6|k->2|k
que poderíamos tentar expandir para:
  ...∧(6|-2->2|-2)∧
      (6|-1->2|-1)∧
       (6|0->2|0)∧
       (6|1->2|1)∧
       (6|2->2|2)∧
       (6|3->2|3)∧...
```

Como calcular isto?
Resposta (honesta): NÃO DÁ PRA CALCULAR!
A fórmula usual para calcular o valor de um "∀" não é "efetiva"
neste caso... "efetivo" é um termo técnico: um procedimento
"efetivo" é um que um computador poderia executar e chegar a um
resultado.
Truque: nós vamos ter _alguns_ métodos indiretos pra calcular
valores de expressões - e alguns destes métodos vão _atribuir
valores_ pra expressões que computadores não conseguiriam calcular
(!!!).
Um exemplo de "método indireto" (em álgebra): 2^100-2^99=...=2^99.
Da mesma forma que temos alguns truques pra fazer algumas contas com
expressões simbólicas e números grandes vamos ter truques pra fazer
algumas contas com expressões infinitas.
Vamos começar vendo _contra-exemplo_ e _exaustão_.
Voltando ao exercício: será que E'_3->E_4?
Aparentemente isto é uma pergunta "infinita"...

```
  ∀P∈?. E'_3->E_4
     ^
     |
     \--- obs: ainda não sabemos o que pôr aqui!
```

Mini-exercício: expanda isto.
Exercício: convença o seu vizinho de que isto é falso.
Exercícios:

```
  a) mostre que "2|k -> 3|k" é falso.
  b) mostre que "3|k -> 2|k" é falso.
  c) mostre que   "V -> E_2" é falso.
  d) mostre que "E_2 -> E_1" é falso.
  e) mostre que "E_1 -> E_3" é falso.
  f) mostre que  "E'_3 -> F" é falso.
```

Sugestão de modo de escrever as respostas:

```
  a) Quando k=2 temos (2|k->3|k) = (V->F) = F,
     portanto 2|k->3|k não é sempre verdade.
  d) Quando P(k) := (k>=2) temos...
```

Vimos que vários alunos estavam com muita dificuldade na hora de
definir funções, e passamos o resto da aula vendo como definir
funções em linguagens de programação e na sintaxe de MD. Em C os
booleanos são representados como números, e podemos escrever coisas
como:

```
  int quadrado   (int x) { return x*x; }
  int menorque4  (int x) { return x<4; }
  int constante2 (int x) { return 2; }
```

e:

```
  main () {
    printf("%d\n", 2<3);   /* -1 */
    printf("%d\n", 2>3);   /* 0  */
  }
```

[Falta pôr aqui a sintaxe do Pascal - que eu chutei um pouco na
aula]
Em MD a gente usa esta sintaxe,

```
  P(x) := (x<=2)
```

ou, ainda mais explicitamente,

```
  P := (λx.x<=2)
```

8ª aula (30/mar): Emergência!

Descobrimos que quase ninguém sabia definir funções bem, então vamos
fazer uma revisão de definição de funções, e vamos ver uma idéia
extra (bonus!): notação lambda (λ).
Temos duas notações pra definir funções em MD:

```
  f(x) := x+2      (notação usual)
  f := λx.x+2      (notação λ)
```

A representação de funções por conjuntos,

```
  f := {(x,x+2) | x∈R},
```

vai ser deixada pra depois.
Exercícios:
(1) Defina funções - uma para cada item, nas duas notações quando

```
      possível -, que:
        a) receba x e retorne x^2-x,
        b) receba w e retorne 2w+3,
        c) receba t e retorne at+b,
        d) receba a e b e retorne a^2-ab,
        e) receba um número de horas, um de minutos e um de segundos,
           e retorne o total de segundos correspondente.
    (2) Sejam:
        g(x) := x^2+2,
        h(y) := y+y,
        P := λt.t<=10
      Calcule:
        2a) g(3)
        2b) h(9)
        2c) g(h(4))
        2d) h(a+b)
        2e) P(g(2))
        2f) P(g(a+b))
        2g) g(x+3)
        2h) h(4y-1)
    (3) Calcule (mostre todos os caminhos):
      (λx.x+x)((λy.y-2)5)
    (4) Sejam w := (λy.y-2) e e h como na 2.
       Calcule h(w(5)).
  Mais uma operação: if
  =====================
  A notação usual em matematiquês é:
    |x| := /  x, quando x>=0,
           \ -x, quando x<0.
  Em notações mais parecidas com C, isto fica:
    |x| := (if x>=0 then x else -x)
  ou:
    |x| := (x>=0 ? x : -x)
  Nós vamos usar, _temporariamente_, esta notação:
    |x| := (if x>=0 then x else -x)
  Exemplo:
    abs := λx.(if x>=0 then x else -x)
  Outra def:
    fact := λn.(if n<=1 then 1 else n*fact(n-1))
  Exercícios:
    (5) Calcule:
      5a) abs(4)
      5b) abs(-5)
      5c) fact(1)
      5d) fact(2)
      5e) fact(3)
  Obs: (if V then alfa else beta) ---> alfa
       (if F then alfa else beta) ---> beta
  Pedi pros alunos refazerem todos os exercícios em casa pra se
  familiarizarem mais com a notação, e avisei que esse pedido era
  sério - a parte mais difícil aqui é a notação, e depois que a gente
  tem prática com a notação o resto fica conceitualmente fácil (apesar
  de trabalhoso).

9ª aula (04/abr):
  Tinha ficado faltando um modo de definir funções...
  Exemplo: f = {(2,4),(3,9),(4,16)}.
  Digamos que B:N->N e que:
     ∀k∈N.B(2k)=10B(k)
  e  ∀k∈N.B(2k+1)=10B(k)+1.
  Por incrível que pareça estas duas sentenças definem uma função...
  Exercício: especialize as duas sentenças acima para k=0,1,2,3
  depois encontre B(0), B(1), ..., B(7).
  Pra quem estiver com dificuldade, expandam isto ao invés:
     ∀k∈{0,1,2,3}.B(2k)=10B(k)
```

```
     ∀k∈{0,1,2,3}.B(2k+1)=10B(k)+1.
  Aí fizemos este sub-exercício, porque os alunos estavam com
  dificuldade de calcular os "B(n)"s...
  Defina funções com os comportamentos abaixo, usando "λ" e "if":
    n   C(n)   D(n)   E(n)
    ----------------------
    0     2      4      3
    1     2      4      3
    2    99      4      2
    3    99      4      2
    4    99      4      1
    5    99      5      1
    6    99      6      1
    7    99      8      1
    9    99      9      1
            (...)
  Todo mundo conseguiu, e eu propus o seguinte problema: digamos que
  G:N->N e que G(0)=2, G(1)=3, G(2)=4. Quanto vale G(200)? Resposta:
  não sabemos! Ele está "livre" (="indefinido") e pode ser qualquer um
  - dá pra definir a G de forma que G(202) seja 202, e dá pra
  definí-la de forma que G(202)=0...
  Na função B que definimos acima, para qualquer n∈N o valor de B(n)
  está "amarrado" ("bem-definido")!
  Agora digamos que F:N->N e que ∀n∈N.f(n)+f(n+1)=f(n+2).
    Se f(0)=2 e f(1)=3, quanto vale f(3)?
    Se f(200)=4 e f(201)=5, quanto valem f(202) e f(203)?
  Digamos que H:N->N e que ∀n∈N.H(n+1)=-H(n).
    Então H(1002)=-H(1001)=H(1000)...
  Repare que por enquanto estamos encontrando relações entre os
  H(0),H(1),H(2),..., G(0),G(1),G(2),...
  Uma _função_ em MD vai ser um conjunto de pares ordenados com a
  seguinte propriedade: se (a,b)∈f e (a,b')∈f então b=b'.
  Exemplo: f = {(2,3),(3,9),(3,10)} não é função.
  Def formal (parcial!):
    f é função se:
    ∀a,b,b'.(a,b)∈f∧(a,b')∈f->b=b'.
  Como estes quantificadores não são limitados a tabela pra "∀" é
  infinita, mas numa linha dela - a que tem a=3, b=9, b'=9 - temos
  ((a,b)∈f∧(a,b')∈f->b=b') falso.
10ª aula (06/abr): Semana Santa.

11ª aula (11/abr): Lembre que tínhamos definido (!)
  uma função B:N->N por:
       ∀k∈N.  B(2k)=10B(k),
       ∀k∈N.B(2k+1)=10B(k)+1...
  O que é "demonstrar P(n) para todo n"?
  Exemplo: digamos que A:N->N obedece:
    (I)  A(0)=0,
    (II) ∀n∈N.A(n+1)=2A(n)+1.
  Exercícios: 1) calcule A(1), A(2), A(3),
              2) mostre que *se* A(1000)=3      <- (III)
                         *então* A(1002)=15.
  O (2) pode ser resolvido fazendo uma lista de "fatos",
    1) A(1000)=3               (por (III))
    2) A(1001)=2A(1000)+1      (por (II), com n=1000)
      ...
    n) A(1002)=15          ***completar***
  Uma das grandes dificuldades de MD é lidar com os "se"s!!!!
  Fizemos a tabela dos valores de A(n) e apareceu uma hipótese:
  ∀n∈N.A(n)=2^n-1. Vamos definir:
    P(n) = (A(n)=2^n-1).
  Como provar que P(1000000)=V?
  [*cav*] NÃO RESPONDAM "É ÓBVIO QUE SIM"!!!!
  Def: A':N->N obedece:
    (I')  A'(0)=1
```

```
    (II') ∀n∈N.A'(n+1)=2A'(n)+1
  Exercício: prove que
    (IV) Se a,k∈N
          e A(k)=2^a-1
         então A(k+1)=2^(a+1)-1.
    Def: Q(k,a) = (A(k)=2^a-1)
         R(k,a) = (Q(k,a)->Q(k+1,a+1))
    Exerc: encontre alguns pares (k,a) pros quais Q(k,a) é verdade e
    alguns pros quais Q(k,a) é falso).
  Truque: podemos calcular o valor de R(k,a) de outra forma -
  e provar que R(k,a) é sempre V sem precisar fazer as tabelas!
  Exercício: simplifique R(k,a).
    R(k,a) = (Q(k,a) -> Q(k+1,a+1))
           = (A(k)=2^a-1 -> A(k+1)=2^(a+1)-1)
  Exercício: prove que se A(k)=2^a-1        <- (I)
                  então A(k+1)=2^(a+1)-1
  e **arrume a demonstração de um modo claro**.
  Discutimos a demonstração no quadro e chegamos a:
    1) A(k) = 2^a-1            (hipótese (I))
    2) 2A(k) - 2(2^a-1)        (por (1) e álgebra)
    3) 2^(m+n) = 2^m 2^n       (regra algébrica conhecida)
    4) 2(2^a-1) =   2·2^a - 2
                = 2^1·2^a - 2
                = 2^(a+1) - 2  (por álgebra)
    5) 2A(k)   = 2^(a+1)-2     (por (2) e (4))
    6) 2A(k)+1 = 2^(a+1)-1     (por (5) e álgebra)
  Exercícios (pra sexta):
    O primeiro exercício da p.84 do livro da Judith é:
    "prove que 2+6+...+(4n-2)=2n^2".
    Defina formalmente uma seqüência A(0), A(1), ... tal que
      A(0) = 0,
      A(1) = 2,
      A(2) = 2+6,
      A(n) = 2+6+...+(4n-2);
    Dê esta definição a) usando somatório, b) usando uma sentença
    da forma "A(0)=...∧∀n∈N.A(n+1)=f(A(n),n)".
    (Deixei a parte de provar a implicação pra aula que vem).
12ª aula (13/abr):
  Voltamos ao exercício da aula passada (Judith, p.84):
    Prove que 2+6+...+(4n-2) = 2n^2.
  O que vamos fazer hoje: nas págs 84 e 85 do livro da Judith tem 22
  problemas de indução que são _contas_ (obs: estamos usando a 5ª
  edição - alguns alunos conseguiram baixar da rede um PDF da 3ª
  edição, e alguns destes problemas aparecem na p.64 da 3ª edição -
  exs 1 a 15). Para cada um destes problemas:
    a) Defina PRECISAMENTE (sem reticências) uma função cujo resultado
       é a expressão à equerda da igualdade do problema (que é sempre
       um somatório ou produtório).
    b) Idem para o lado direito (fácil).
    c) Se chamamos estas funções de A_k(n) e B_k(n), onde k é o número
       do problema (k=1,2,...,22), cada problema consiste um provar
       que:
         ∀n∈N. A_k(n)=B_k(n)
       Prove que:
         A_k(1000)=B_k(1000) -> A_k(1001)=B_k(1001)
       e que:
         A_k(n)=B_k(n) -> A_k(n+1)=B_k(n+1)    (para n∈N)
  Aí vi que os alunos estavam com MUITA dificuldade no (a), e passei
  dois exercícios mais básicos:
    a^-) Encontre o valor (numérico!) de:
         A_1(2), A_1(3), A_1(4),
         A_2(2), A_2(3), A_2(4),
         A_3(2), A_3(3), A_3(4),
         ...
         A_22(2), A_22(3), A_22(4).
```

a^--) Escreva as igualdades das questões 1-11 da Judith para n=2, n=3, n=4.

Fizemos uma tabela no quadro:

k	A_k(2)	A_k(3)	A_k(4)
1	2+6	2+6+10	2+6+10+14
2	2+4	2+4+6	2+4+6+8
3	1+5	1+5+9	1+5+9+13
	(...)		
7	1^2+2^2	1^2+2^2^3^2	1^2+2^2^3^2+4^2

e dei alguns exemplos de definições formais em matematiquês sem reticências (algumas indutivas): a função B que retorna a expansão binária de cada natural, a sequência de Fibonacci, e B_4 (o lado direito da igualdade do problema 4), em várias sintaxes:

```
      B_4(n)  :=   n(n+1)(n+2)/6
      B_4  := λn∈N.n(n+1)(n+2)/6
        B_4: N --> N
             n |-> n(n+1)(n+2)/6
      ∀n∈N. B_4(n)=n(n+1)(n+2)/6
```

As dicas principais pra como resolver estes problemas são as mesmas de sempre (e que valem pra todos os cursos de matemática):

```
******************************
** 1) Escrevam suas hipóteses **
** 2) Testem-nas              **
******************************
```

13ª aula (18/abr): Nossa única aula sobre "contas" (no sentido usual de "contas")!

Primeiro voltamos pros problemas da aula passada, e pedi pros alunos definirem formalmente A_1(n), A_2(n), A_3(n).

Depois de muitas tabelas e muito sofrimento chegamos a estas expressões para as "diferenças" (def: C_k(n) = A_k(n+1)-A_k(n)),

```
   ∀n∈N.C_1(n)=4n+2
   ∀n∈N.C_2(n)=2n+2
   ∀n∈N.C_3(n)=4n+1
```

a estas sentenças para os "lados esquerdos das igualdades",

```
   A_1(0)=0 ∧ ∀n∈N. A_1(n+1)=A_1(n)+4n+2
   A_2(0)=0 ∧ ∀n∈N. A_2(n+1)=A_2(n)+2n+2         (*)
   A_3(0)=0 ∧ ∀n∈N. A_3(n+1)=A_3(n)+4n+1
```

e a estas para os "lados direitos":

```
   ∀n∈N.B_1(n)=2n^2
   ∀n∈N.B_2(n)=n(n+1)                            (**)
   ∀n∈N.B_3(n)=n(2n-1)
```

Os exercícios de indução da Judith pedem pra gente provar que:

```
   ∀n∈N.A_1(n)=B_1(n),
   ∀n∈N.A_2(n)=B_2(n),
   ∀n∈N.A_3(n)=B_3(n),
```

é mais fácil a gente começar provando que:

```
   A_1(0)=B_1(0) ∧ ∀n∈N.(A_1(n)=B_1(n) -> A_1(n+1)=B_1(n+1))
   A_2(0)=B_2(0) ∧ ∀n∈N.(A_2(n)=B_2(n) -> A_2(n+1)=B_2(n+1))
   A_3(0)=B_3(0) ∧ ∀n∈N.(A_3(n)=B_3(n) -> A_3(n+1)=B_3(n+1))
```

A minha idéia inicial era que a gente passaria a maior parte da aula discutindo isto aqui: como provar _claramente_ algo como

```
                  ∀n∈N.(A_1(n)=B_1(n) -> A_1(n+1)=B_1(n+1)) ?
```

Queremos nos livrar do "∀n∈N" e do "A_1(n)=B_1(n) ->"...

Truque: "suponha"!

```
  Suponha que                                n∈N
  e que                                A_2(n) = B_2(n).
  Queremos provar que                A_2(n+1) = B_2(n+1).    (***)
  Como                           ∀n∈N. B_2(n) = n(n+1)
  temos                              B_2(n+1) = (n+1)(n+2)
  e portanto (***) é equivalente a   A_2(n+1) = (n+1)(n+2)   (****)
  e como sabemos que                 A_2(n+1) = A_2(n)+2n+2
  temos que (****) é equivalente a A_2(n)+2n+1 = (n+1)(n+2)   (*****)
  Como supusemos que                   A_2(n) = B_2(n) = n(n+1)
```

```
    basta provar que                 n(n+1)+2n+1 = (n+1)(n+2)
    que é uma conta...
    (Obs: vamos entender mais sobre estes "suponha"s na aula que vem!)
    ***Note que nós não começamos suponde que (***) era verdade!!!***
14ª aula (20/abr): Dei pros alunos cópias das pags 84 a 86 da Judith e
  passamos a aula discutindo o problema 13 (da p.85).
  No fim da aula os alunos estavam sabendo definir as funções A e B
  bastante bem, e ficaram de treinar em casa provar a parte
  "∀n∈N.A(n)=B(n)->A(n+1)=B(n+1)" de cada um dos problemas das pags 84
  a 86 da Judith, e ficaram de tentar fazer o que desse dos outros
  problemas.

15ª aula (25/abr): Não íamos ter muito tempo de aula, porque um debate
  no saguão sobre as mobilizações para uma provável greve começaria às
  16:30 ou pouco depois. Apresentei a calculadora da Gabriela e pedi
  pros alunos começarem a fazer experiências com ela no computador do
  Pedro Antonellini e no meu.
16ª aula (27/abr):
  Seja A:N->N a função que obedece:
    A(0)=0,
    ∀n∈N. A(n+1) = (if (A(n)=n)
                    then 10A(n)+9
                    else A(n))
  Calcule A(0), A(1), ..., A(1002).
  Seja B: N×N  -> N
          x,y |-> x(A(y)+1)+y.
  Calculem: B(1, 2),
            B(12, 345),
            B(10, 20),
            B(10, 200),
            B(0, 23),
            B(23, 0),
            B(0, 0).
  Vamos definir a operação ++ ("concatenação")
  _para argumentos numéricos_ desta forma:
    quando x,y∈N, x++y := B(x,y).
  Quando os argumentos do ++ forem strings, x++y vai ser definido
  como a concatenação usual.
  Def:       E_0 := {"a"},
   ∀n∈N. E_(n+1) := {x++"+"++y | x,y∈E_n}
                  ∪ {"("++x++")" | x∈E_n}
                  ∪ E_n
  Calculem E_1 e E_2.
  (Passamos um tempão tirando dúvidas sobre este exercício, e vendo
  truques pra verificar a sintaxe).
  [*CAV*] Vocês vão precisar ter prática suficiente com este tipo de
  definição!
  Moral: dá pra definir _precisamente_ o conjunto das expressões
  válidas (de uma linguagem).
  O detalhe que falta é:
         ∞
    E =  ∪  E_i.
        i=0
  Árvores (formalmente)
  =====================
  A expressão 2·3+4·5 pode ser representada como:
         +            +_____.
        / \           |     |
      *     *         *__.  *__.
     / \   / \        |  |  |  |
    2   3 4   5       2  3  4  5
  Formalmente:
    ("+", ("*", 2, 3), ("*", 4, 5))
  Vamos definir:
              F_0 := {2}
```

```
    ∀n∈N. F_(n+1) := {("+", x, y) | x,y∈F_n}
                   ∪ {("*", x, y) | x,y∈F_n}
                   ∪ F_n
  Exercício:
  Calculem F_1 e representem cada termo de F_1 como árvore.
  Idem para F_2.
  Sub-exercício: represente "como lista" e "como árvore" isto aqui,
  que está numa notação híbrida:
    ("+", *__. , 5)
          |  |
          3  4

  Fato [*cav*]: expressões matemáticas são escritas _mais ou menos_
  como strings... obs:
     ∞   1
     Σ  ---   é um string????...
    n=1 2^n
  mas elas vão ter várias representações formais diferentes - algumas
  delas em árvores - e a noção de _subexpressão_ sempre está
  bem-definida. Por exemplo, na expressão 9-4-3, temos:
         isto _não é_ uma subexpressão,
         /---\
     9 - 4 - 3
     \---/
     mas isto é,
  já que 9-4-3 = (9-4)-3... em árvore:
         -
        / \
       -   3
      / \
     9   4
  outro exemplo que talvez seja mais claro:
    isto é subexpressão
    /---\   e isto também,
            /---\
    2 * 3 + 4 * 5
        \---/
        mas isto não,
  e se a gente tenta calcular 2*3+4*5 transformando-o em 2*7*5 a gente
  se ferra. Subexpressões _em geral_ podem ser substituídas pelos seus
  resultados...
  Dever de casa: olhem para *todas* as expressões que vocês já viram,
  encontre _algum modo_ de representá-las em árvore, e aprenda a
  reconhecer subexpressões.
  Dever extra: identifique as (poucas) situações nas quais
  subexpressões não podem ser substituídas pelos seus valores. Por
  exemplo:
    ∀x∈N.x+2>0
     \-/
     aqui.

17ª aula (02/mai): Hoje: revisão de relações! (Scheinerman, seções 11-)
  Lembre que uma _relação de A em B_
        (ou: uma "relação de A para B)
  é um subconjunto R \subseteq A×B.
  Vamos começar com um exemplo:
    A={1,...,10}
    R={(a,b)∈A^2 | ∃p primo. pa=b}
  Às vezes representamos relações graficamente, interpretando cada par
  (a,b) como uma seta a->b. Exercício: faça a representação gráfica de R.
  Notação "infix" para relações: aRb quer dizer (a,b)∈R.
  Composição de relações:
           se R \subseteq A×B
            e S \subseteq B×C
    então (R¢S) \subseteq A×C é definida por:
```

a(R¢S)b se e só se ∃b∈B. aRb∧bSc.
Exercício: calcule R¢R para R={(a,b)∈A^2 | ∃p primo. pa=b}.
Exercício: calcule R'¢S', onde
A = {1,2,3,4},
B = {5,6,7,8},
C = {9,10,11,12},
R' = {(1,5),(2,5),(3,6),(3,7)}
S' = {(5,9),(6,9),(6,10),(7,11),(8,12)}
(fizemos graficamente)
Def: R^2=R¢R, R^3=R¢R¢R, ..., e o _fecho transitivo de R_, R^+,
é definido como:

```
 +    ∞   n
R  =  ∪  R
     n=1
```

Exercício: calcule R^2, R^3, R^4 e R^+ (fizemos graficamente).
Pra que serve isto?
===================
Um exemplo: seja E o conjunto das expressões válidas, e definimos:
e Red e' se e só se a expressão e se reduz à expressão e' "em um passo". Fizemos um exemplinho no quadro, começando com 2·3+4·5, e mostrei que a relação Red^+ diz que expressões podem ser reduzidas a outras.
Aí definimos:
A_k = {k,...,10},
R_k = {(a,b)∈A_k^2 | ∃p primo. ap=b}
Vocês sabem calcular, p.ex., R_3^+ (fizemos isso graficamente).
Def: a _inversa_ de uma relação R \subseteq A×B é:
R^-1 = {(b,a) | (a,b)∈R}
Calculem: (R_3^+)^-1
e S = R_3^+ ∪ (R_3^+)^-1.
A relação S é transitiva? Isto é, S = S^+?
(Calculamos S e S^+ graficamente; comecei a mostrar o que é a relação de equivalência gerada pela relação Red, mas ainda nem mencionei relações de equivalência pelo nome. Pedi pros alunos lerem as seções 11 e 12 do Scheinerman em casa e tentarem fazer os exercícios).

18ª aula (04/mai): Exercícios de revisão:

1) A função "floor" ("piso") pode ser definida por:
para qualquer x∈R, floor(x)=y se e só se y∈Z e x∈[y,y+1).   ]
Def: f(k) = floor(k/3).
Consiga uma "definição formal no sentido de MD", indutiva, para f:N->N. Lembre que isto quer dizer que f tem que ser definida por uma expressão da forma "f(0)=... ∧ ∀n∈N. ...", que não envolve números não-inteiros.
2) Consiga uma definição formal, indutiva, para:

```
g: N  -> N
   n |-> n - 10 floor(n/10).
```

3) Def: (n mod 10) := g(n)   (isto é só uma mudança de notação),
onde g é a função acima.

```
Seja h: {0,...,9} -> {0,...,9}
                n |-> 3n mod 10
```

Represente h e h^-1 como conjuntos.

4) Seja D: N×N -> N a função que "percorre N×N por diagonais"; se escrevermos sobre cada ponto de N×N o valor de D(x,y), temos:

```
  |
 14 19
  |
  9 13 18
  |
  5  8 12 17
  |
  2  4  7 11 16
  |
--0--1--3--6-10-15
```

```
      |
     defina formalmente a função D.
  5) Se f:A->B e A' \subseteq A,  a∈A,
                 B' \subseteq B,  b∈B,
     um abuso de linguagem comum em livros de Matemática é escrevermos
     simplesmente f(a,B'), f(A',b), f(A',B') para:
       f(a, B') := {f(a, b') | b'∈B'}           (*)
       f(A',b)  := {f(a',b ) | a'∈A'}           (**)
       f(A',B') := {f(a',b') | a'∈A', b'∈B'}    (***)
     A mesma idéia pode ser usada para operações binárias. P.ex.:
       {10, 20} + {3, 4} = {13, 14, 23, 24}
     Calcule "(" ++ {"a","b"} ++ ")".
     Primeiro mostre o resultado, depois mostre passo a passo como
     calculá-lo usando as regras (*), (**) e (***).
  Obs: floor(0.66)=0 <=> 0.66∈[0,0+1)
       floor(0.66)=1 <=> 0.66∈[1,1+1)
  Fizemos uma tabela para f(n), g(n) e h(n), mas ninguém estava
  conseguindo encontrar as definições formais para f.
  Começamos a discutir como resolver a (1). Tentei convencer os alunos
  a começarem com chutes ruins e irem testando eles e melhorando-os
  aos poucos. Discutimos estas hipóteses (erradas):
    f_1(n) = if n <= 3(n-1) then n else n+1
    f_2(n) = if n  = 0      then 0 else f_2(n-1)+2
    f_3(n) = if n <= 3      then 0 else 4 f_3(n-2)+5
  Todo o resto ficou pra casa.           ]]

19ª aula (09/mai): *** a P1 foi remarcada para 16/maio ***
  Hoje: revisão, e esclarecer a relação entre alguns pedaços do livro
  e como a matéria foi dada em sala:
    * Tautologias,                        (<= Scheinerman, p.30)
    * Substituição de iguais por iguais,  (<= Scheinerman, p.523)
    * alguns métodos de prova.
  1) Sejam P_1 e P_2 estas duas tentativas de definir "primo"
     formalmente:
       P_1(n) <=> ∀a,b∈{2,...,n-1}. ab!=n
       P_2(n) <=> ∀a,b∈{-n,...,n}. (ab=n -> (a=1∨b=1∨a=n∨b=n))
     Compare as definições de P_1 e P_2.
     Elas são equivalentes?
     Mostre como calcular P_1(5) e P_2(5).
     Faça isto de um modo que seja fácil de entender e de generalizar.
       (Fizemos uma tabelona para (a=1∨b=1∨a=n∨b=n), outra para
        (ab=n), outra para (ab=n -> (a=1∨b=1∨a=n∨b=n)).)
     Como mostrar pro leitor mais formalmente como estas tabelas que
     vocês construíram funcionam? Definam funções de a e b, e treine
     usar reticências. Sugestão:
       Q(a,b) = (ab=5),
       R(a,b) = (a=1∨b=1∨a=n∨b=n),
       S(a,b) = (ab=5-> (a=1∨b=1∨a=n∨b=n)).
     Outra notação possível: alfa_ab, beta_ab, gama_ab.
  2) Revisão de um detalhe sobre igualdade e conjuntos.
     Se encaramos A=B "computacionalmente", ele funciona assim:
       Se A e B têm tipos diferentes, retorna falso;
       se A e B são números, use a comparação usual;
       se A e B são booleanos, idem;
       se A e B são _conjuntos_, use estas definições:
                     A=B <=> A \subseteq B ∧ B \subseteq A
           A \subseteq B <=> ∀a∈A.a∈B
         a∈{b_1,...,b_n} <=> a=b_1 ∨ ... ∨ a=b_n.
     Exercício (5 mins, em sala): calcule {3,4}={4,3,4}.
     Mencionei quanto trabalho um computador (programado do jeito mais
     simples, sem otimizações) teria para calcular:
       {{3,4}, {2,3,4}} = {{4,3,4}, {2,2,3,3,4}, {4,3}}
  3) Tautologias e expressões equivalentes:
       P->Q é equivalente a ¬P∨Q
     Isto é o mesmo que:
```

```
      ∀P,Q∈{F,V}. ((P->Q)=(¬PvQ))
    (Podemos substituir o "=" por "<->").
    E:
      (P∧Q)->P é uma tautologia
    é o mesmo que:
      ∀P,Q∈{F,V}. (P∧Q)->P.
    Pra casa: faça estas contas, e leia as últimas frases da p.523 do
    Scheinerman - uma conseqüência delas é que podemos substituir
    (P∧Q)->P por V em _qualquer lugar_, e para quaisquer P e Q - que
    podem ser expressões!
    Pra casa: releia a matéria toda e identifique onde usamos esta
    idéia - ela já foi usada várias vezes, mas sem a nomearmos muito
    claramente.
20ª aula (11/mai): revisão.
  Voltamos ao problema da aula passada (o P_3 é novo):
     P_1(n) <=> ∀a,b∈{2,...,n-1}. ab!=n
     P_2(n) <=> ∀a,b∈{-n,...,n}. (ab=n -> (a=1vb=1va=nvb=n))
     P_3(n) <=> ∀a,b∈ {1,...,n}. (ab=n -> (a=1vb=1va=nvb=n))
  Encontre um modo de mostrar como calcular P_2(15) e P_2(17) -
  use definições e reticências.
  Dica: sabemos simplificar algumas expressões como P∧Q, PvQ, P->Q se
  sabemos o valor de P ou o de Q.
    Se P=V:  P∧Q ---> Q
             PvQ ---> V
            P->Q ---> Q
    Se P=F:  P∧Q ---> F
             PvQ ---> Q
            P->Q ---> V
    Se Q=V: P->Q ---> V
    Se Q=F: P->Q ---> ¬P
  Exercício: defina formalmente Q(a,b), R(a,b), S(a,b) em:
              S(a,b)
      /---------------------\
     Q(a,b)        R(a,b)
      /--\    /------------\
      ab=n -> a=1vb=1va=nvb=n
  Mostre que o valor de P_3(15) só depende dos resultados de R(1,15),
  R(3,5), R(5,3), R(15,1).
  Os alunos levaram um tempão pra chegarem a definições para Q, R, S
  com a sintaxe certa, como a da função P abaixo... primeiro tentamos
  definir Q e R em C:
    int n;
    int Q (int a, b) { if (a*b==n) return 1; else return 0; }
    int Q2(int a, b) { return a*b==n; }
    int R (int a, b) { return a==1 || b==1 || a==n || b==n; }
    int main () {
      n = 5;
      printf("%d\n", Q(2, 3));
    }
  Dica: se vocês entenderem isto direito vocês vão entender como é que
  o "->" pode ser usado pra formalizar a idéia de "quando".
  Seja T: N -> {F,V} um predicado sobre os naturais, e seja:
    P: N  -> {F,V}
       n |-> (n é primo)
  Mostre como simplificar:
    ∀k∈{2,...,20}. P(k)->T(k),
    ∀k∈{2,...,20}. T(k)->P(k),
    ∃k∈{2,...,20}. P(k)->T(k),
    ∃k∈{2,...,20}. T(k)->P(k),

21ª aula (16/mai): P1. Matéria: definições, construções indutivas,
  alguns tipos de demonstrações. As principais coisas que vocês vão
  ter que saber para a prova são:
    1) interpretar definições MUITO BEM,
    2) calcular valores de expressões passo a passo (por reduções),
```

```
      incluindo expressões das formas "∀a∈A.___" e "∃a∈A.___", onde A
      é um conjunto finito,
   3) mostrar que certas sentenças da forma "∀a∈A.___" são falsas,
   4) mostrar que certas sentenças da forma "∃a∈A.___" são
      verdadeiras,
   5) mostrar que certas sentenças são ou não são tautologias,
   6) entender funções definidas indutivamente (como fizemos em
      muitas das últimas aulas)
   7) encontrar definições formais para funções definidas
      informalmente,
   8) definir formalmente as sequências A(n) e B(n) dos problemas de
      indução das pags 84 a 86 da Judith e provar que
      A(n)=B(n)->A(n+1)=B(n+1).
  Lembre da folha de regras:
   http://angg.twu.net/LATEX/2011-1-GA-regras.pdf
  Scan da prova (ainda sem gabarito):
   http://angg.twu.net/MD/MD_P1_2012may16.pdf
   http://angg.twu.net/MD/MD_P1_2012may16.djvu
22ª aula (18/mai):

   --------------
23ª aula (26/set): 1ª aula depois da greve.
  Hoje: Aritmética modular - introdução.
  Comecei fazendo tabelas para duas funções que representavam a
  operação "módulo 3":
      m_3: Z -> {0,1,2}
   e  f_3: N -> {0,1,2}.
  Pra retormar o ritmo nós vamos tentar _definir formalmente_ essas
  funções, de modo que a definição formal coincida com o comportamento
  esperado (que todo mundo aprendeu intuitivamente preenchendo as
  tabelas). Avisei que vamos começar com a f_3, porque ela é bem mais
  fácil.
  Vimos que isto aqui é verdade:
    ∀n∈N.(f_3(n) = f_3(n+3))
  vamos usar isto como uma parte da definição da f_3.
  Exercício: digamos que g:N->N obedece:
    g(2) = 5,
    g(4) = 9,
    g(6) = 20,
    ∀n∈N.g(n)=g(n+3)
  Calcule g(10), g(11) e g(12) e explique como você os calculou. Dica,
  use algo como isto aqui pra escrever a parte em Português:
   Fazendo n=17,  ∀n∈N.g(n)=g(n+3)
           vira:      g(17)=g(20).
  Exercício (*importante*!):
   Seja h:N->N uma função que obedece
       ∀n∈N. h(n)=h(n+3)
    e  ∀n∈N. n<2->h(n)=n.
   a) Mostre como calcular h(7),
   b) mostre que h(6) não está definido, usando o seguinte truque:
      defina duas funções, h' e h'', que obedeçam as condições acima,
      mas tais que h'(6)=10 e h''(6)=20.
24ª aula (28/set):
  Ainda não transcrevi. Fotos do quadro:
   http://angg.twu.net/MD/quadro/2012-09-28_MD1.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD2.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD3.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD4.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD5.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD6.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD7.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD8.jpg
   http://angg.twu.net/MD/quadro/2012-09-28_MD9.jpg

25ª aula (03/out):
```

```
  Ainda não transcrevi. Fotos do quadro:
    http://angg.twu.net/MD/quadro/2012-10-03_MD1.jpg
    http://angg.twu.net/MD/quadro/2012-10-03_MD2.jpg
    http://angg.twu.net/MD/quadro/2012-10-03_MD3.jpg
    http://angg.twu.net/MD/quadro/2012-10-03_MD4.jpg
26ª aula (05/out):
  Ainda não transcrevi. Fotos do quadro:
    http://angg.twu.net/MD/quadro/2012-10-05_MD1.jpg
    http://angg.twu.net/MD/quadro/2012-10-05_MD2.jpg
    http://angg.twu.net/MD/quadro/2012-10-05_MD3.jpg
    http://angg.twu.net/MD/quadro/2012-10-05_MD4.jpg

27ª aula (10/out):
  Ainda não transcrevi. Fotos do quadro:
    http://angg.twu.net/MD/quadro/2012-10-10_MD1.jpg
    http://angg.twu.net/MD/quadro/2012-10-10_MD2.jpg
    http://angg.twu.net/MD/quadro/2012-10-10_MD3.jpg
    http://angg.twu.net/MD/quadro/2012-10-10_MD4.jpg
    http://angg.twu.net/MD/quadro/2012-10-10_MD5.jpg
    http://angg.twu.net/MD/quadro/2012-10-10_MD6.jpg
    http://angg.twu.net/MD/quadro/2012-10-10_MD7.jpg
    http://angg.twu.net/MD/quadro/2012-10-10_MD8.jpg
28ª aula (12/out): Feriado (Dia das Crianças).

29ª aula (17/out):
  Ainda não transcrevi. Fotos do quadro:
    http://angg.twu.net/MD/quadro/2012-10-17_MD1.jpg
    http://angg.twu.net/MD/quadro/2012-10-17_MD2.jpg
    http://angg.twu.net/MD/quadro/2012-10-17_MD3.jpg
30ª aula (19/out):

31ª aula (24/out):
32ª aula (26/out): P2

33ª aula (31/out): VR
34ª aula (01/nov): VS


Notas da P1:
  Cauã              7.0
  Felipe            4.0
  Francielle        3.0
  Ilídio            1.4
  Kelvin            3.4
  Leandro           3.2
  Nayara            8.2
  Pedro Antonellini 6.6
  Thomaz            4.0
  Valdenir          0.7
  Dayana             ?
  Leonardo           ?
  Pedro Paulo        ?
  Vinicius           ?
```