Lições aprendidas e uso da matemática durante o desenvolvimento e implantação de um software inovador na área estadual da saúde.
Motivação teórica e exemplos práticos em Ruby (Rails).

Apresentação para a Semana Municipal de Ciência, Tecnologia e Inovação

Prof. Marcelo Eduardo Redoschi

São Paulo, 25 de Outubro de 2017

Objetivo da apresentação

* Compartilhar experiência e lições aprendidas sobre o projeto e desenvolvimento de um sistema web inovador na esfera pública estadual.

* Em sentido amplo, realiza a gestão da transferência de recursos financeiros, nesta implementação específica trata-se de bolsistas de um programa de pós-graduação.

Apanhado geral ("overview") do sistema

* Desenvolvimento baseado em tecnologia previamente utilizada quando estive no Isitec (onde trabalhei em 2013).

* Em funcionamento desde Abril de 2016.

* Modernizou o processamento da matrícula ao melhorar a gestão e controle dos pagamentos bancários aos bolsistas.

* Matemática utilizada durante o desenvolvimento e controle de qualidade.

    * Otimização de processos (exemplo: registro mensal de frequência dos alunos).

Premissas fundamentais

- Eliminar ao máximo papel e antigos controles manuais (uso inapropriado de planilhas como 'bases de dados').
  - Eficiência (princípio consagrado na Constituição Federal).
  - Transparência e controle (acesso compartimentalizado, inclusive com permissões 'read-only').
- Interface de navegação web intuitiva, reativa ("responsive"), c/ links.
  - Telas amigáveis em computadores quanto em celulares e tablets, de vários fabricantes.
- Projetado para ser internacional desde o princípio (desenvolvido em inglês, textos traduzidos ao ortuguês).

Premissas (continua)

* Desenvolvimento sustentável (de software). Remete à arquitetura/engenharia física.
    * Utiliza software livre "de ponta", robusto, sem custos de licença ao erário.
    * Treinamento e atualizações mais factíveis (informação disponibilizada na Internet pela comunidade, desenvolvedor consegue replicar ambiente em seu laptop ou na produção sem custo).
    * "Leve", deixa pequeno rastro (footprint) dentro do servidor. Roda em ambiente compartilhado.
    * Disseminação local do conhecimento ("local capacity building").

Tecnologias utilizadas (bem resumidamente)

- ✓ Telas modernas ("responsive"): Zurb Foundation
- ✓ Framework: Ruby on Rails (software livre)

devise, i18n, cancan, yard, railroady, rspec, ransack

- ✓ Banco de dados: Postgres
- ✓ Controle de versão: git
- ✓ Conferência externa: R(Studio)

Público alvo do sistema

* Âmbito é sem fins lucrativos, do setor público.
* Mais de cem usuários institucionais cadastrados, de 40 universidades, faculdades, institutos de pesquisa públicos e privados atuantes no estado de São Paulo.
* Este contato com as instituições é gerenciado por uma pequena equipe (três pessoas). Inspirado em metodologias ágeis, realizo reuniões periódicas.
    * Um "sprint" geralmente leva de uma semana a um mês, a depender de minha carga de trabalho (pois sou o único desenvolvedor). Tenho múltiplas responsabilidades dentro do setor. Princípio norteador: "local capacity building", disseminação do conhecimento localmente dentro do quadro permanente (i.e. de funcionários concursados).
    * Minha expectativa é que futuramente os próprios bolsistas poderão consultar diretamente suas informações, tais como frequência e demonstrativos bancários.

Lições aprendidas ("Lessons Learned")

* Em 2017, os usuários adquiriram a cultura dos "smartphones e tablets", portanto presupõem que a navegação buscas e relatórios serão intuitivos, sempre.

* Informações importantes, tais como aquelas que impactam pagamento de bolsas (ou eventualmente benefícios sociais, etc), que necessitem de comprovação por parte da unidade central, inicialmente entrarão como "pendentes" (sem confirmação). Somente após verificação pelaunidade central é que passariam a ser assinaladas como autorizadas (confirmadas).

    * Um corolário disto é que o sistema, por padrão, aceitará anexos de documentação comprobatória, dentro de um limite razoável de tamanho (exemplo: 2 megabytes).

* Haverá um menu de "definições", ou como queira chamá-lo, onde constarão informações do tipo:

    * Geográficas (países, estados, municípios, eventualmente macro e/ou micro regiões estaduais).

    * Agências bancárias (periodicamente atualizadas, com base na informação oficial do banco central, em se tratando do Brasil). Linguagem R auxilia nisto.

    * Classificação brasileira das ocupações (CBO).

    * Etc...

Lições aprendidas 2

* Inspirado na metodologia australiana, incluí informação de gênero (facultativa, informada em caso de diversidade) e as opções F, M, X.
* Documentação de projeto é essencial! Se estiver em inglês, melhor ainda.
* Dentro do código fonte, commits (desenvolvedor)
* Testes (tarefas rake, Rspec, watir)
* Validações (consistência) com mensagens 'auto-explicativas' de erro.
* Balões de ajuda em campos importantes (Zurb Foundation 'tooltips')
* Tutoriais dentro do sistema (páginas "html", leves). Se estiverem personalizados de acordo com a permissão do usuário, melhor ainda.

Lições aprendidas 3

* O sistema ou aplicativo implementará uma solução específica, (em meu caso gestão de um programa de bolsas), mas poderá ser adaptado para outras situações mais gerais (e.g. programas de assistência social)
    * Mecanismo de criação de usuários e entrada dados de contato consistentes já existe, bem como o mecanismo de geração de arquivos bancários texto.

* A matemática pode (e deve!) ser utilizada durante o desenvolvimento e aperfeiçoamento do sistema.

* Uma lógica correta é que o computador espera, (independentemente de linguagem de programação ou plataforma utilizada).

* (Potencialmente) melhora a comunicação. Recomenda-se utilizar conceitos do nível médio, quando possível, de modo a tornar o conteúdo compreensível a um público interno provavelmente leigo. Carência (por razões estruturais históricas) enorme de matemáticos, engenheiros, físicos, cientistas da computação atuantes na esfera pública. Razões estruturais, que estão além do escopo desta apresentação, porém que devem ser levadas em consideração.

Teoria matemática

Definition (syntax of propositional logic)

An *atomic formula* has the form A_i where $i = 1, 2, 3, \ldots$. *Formulas* are defined by the following inductive process:

1. All atomic formulas are formulas.
2. For every formula F, $\neg F$ is a formula.
3. For all formulas F and G, also $(F \vee G)$ and $(F \wedge G)$ are formulas.

A formula of the form $\neg F$ is called *negation* of F. A formula of the form $(F \vee G)$ is called *disjunction* of F and G, and $(F \wedge G)$ is the *conjunction* of F and G. Any formula F which occurs in another formula G is called a *subformula* of G.

Example: $F = \neg((A_5 \wedge A_6) \vee \neg A_3)$ is a formula, and all subformulas of F are:

$$F,\ ((A_5 \wedge A_6) \vee A_3),\ (A_5 \wedge A_6),\ A_5,\ A_6,\ \neg A_3,\ A_3$$

We introduce the following abbreviations which allow a more succinct representation of formulas:

$A, B, C, \ldots$	instead of	$A_1, A_2, A_3, \ldots$
$(F_1 \rightarrow F_2)$	instead of	$(\neg F_1 \vee F_2)$
$(F_1 \leftrightarrow F_2)$	instead of	$((F_1 \wedge F_2) \vee (\neg F_1 \wedge \neg F_2))$

Teoria matemática 2

$$(\bigvee_{i=1}^{n} F_i) \quad \text{instead of} \quad (\ldots((F_1 \vee F_2) \vee F_3) \vee \cdots \vee F_n)$$

$$(\bigwedge_{i=1}^{n} F_i) \quad \text{instead of} \quad (\ldots((F_1 \wedge F_2) \wedge F_3) \wedge \cdots \wedge F_n)$$

Here, $F_1, F_2, \ldots$ can be arbitrary formulas. In particular, that means that $(A \leftrightarrow E)$ is an abbreviation for the formula

$$((A \wedge E) \vee (\neg A \wedge \neg E))$$

which, again, is an abbreviation for

$$((A_1 \wedge A_5) \vee (\neg A_1 \wedge \neg A_5)).$$

Notice that formulas are nothing else but strings of symbols (i.e. syntactical objects). They do not have a "content" or "meaning" at the moment. Therefore, it would be incorrect (or premature) to read $\wedge$ as "and", and $\vee$ as "or". Better would be, say, "wedge" and "vee".

Formulas – and the components occuring in formulas – obtain an associated "meaning" by the following definition.

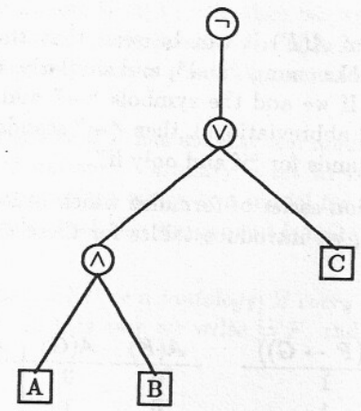
Definition (semantics of propositional logic)

The elements of the set $\{0,1\}$ are called *truth values*. An *assignment* is a function $\mathcal{A} : \mathbf{D} \to \{0,1\}$, where $\mathbf{D}$ is any subset of the atomic formulas. Given an assignment $\mathcal{A}$, we extend it to a function $\mathcal{A}' : \mathbf{E} \to \{0,1\}$, where $\mathbf{E} \supseteq \mathbf{D}$ is the set of formulas that can be built up using only the atomic formulas from $\mathbf{D}$.

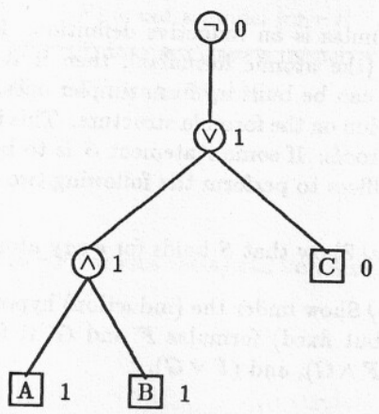
1. For every atomic formula $A_i \in \mathbf{D}$, $\mathcal{A}'(A_i) = \mathcal{A}(A_i)$.
2. $\mathcal{A}'((F \wedge G)) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 1 \text{ and } \mathcal{A}'(G) = 1 \\ 0, & \text{otherwise} \end{cases}$
3. $\mathcal{A}'((F \vee G)) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 1 \text{ or } \mathcal{A}'(G) = 1 \\ 0, & \text{otherwise} \end{cases}$
4. $\mathcal{A}'(\neg F) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 0 \\ 0, & \text{otherwise} \end{cases}$

Teoria matemática 3

Using these tables, it is easy to determine the truth value of a formula F, once an assignment of the variables occuring in F is given. As an example, we consider again the formula $F = \neg((A \wedge B) \vee C)$, and we represent the way F is built up by its subformulas as a tree:



The truth value of F is obtainable by marking all leaves of this tree with the truth values given by the assignment $\mathcal{A}$, and then determining the values of the inner nodes according to the above tables. The mark at the root gives the truth value of F under the given assignment $\mathcal{A}$.

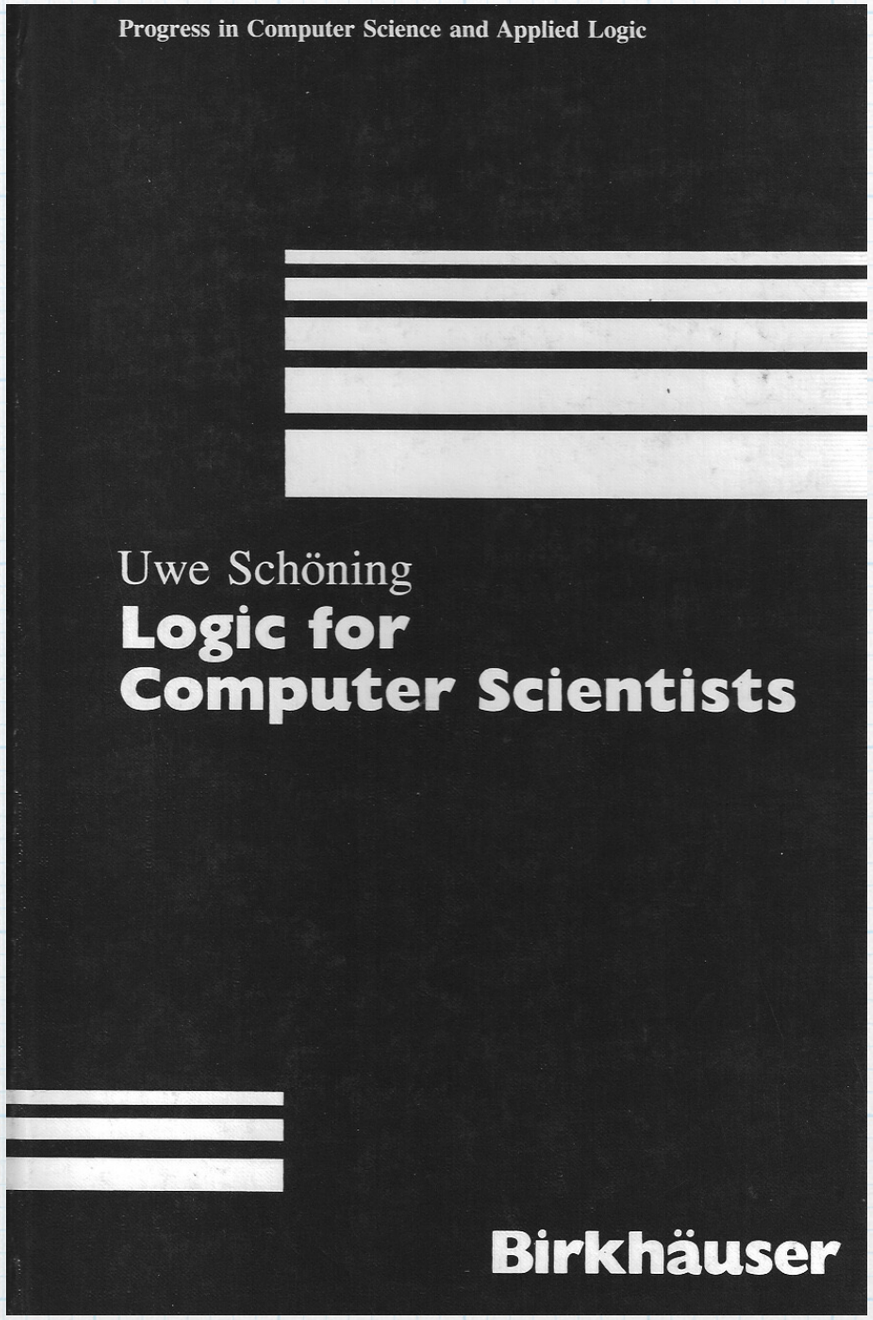


Objetivo: se ter fórmulas válidas (lógica consistente)

all formulas in propositional logic

valid formulas	satisfiable, but non-valid formulas	unsatis-fiable formulas
$\neg G$	F $\neg F$	G

Base teórica (livro de referência)



Exemplos práticos de uso da matemática

O sistema foi projetado levando-se em conta conceitos da matemática, tais como árvores binárias e propriedades dos conjuntos. Esta abordagem possibilitou-me definir de maneira mais precisa, primeiro em notação matemática e em um segundo momento, expressas na linguagem de programação Ruby, conceitos relativamente complexos de maneira precisa e eficiente ("código elegante").

Melhoria do controle de frequência dos matriculados (aplicação prática)

* Como exemplo, poderia citar uma regra (definição, conceito) importante que adotei, referente ao controle de frequência dos bolsistas:

* "Supõe-se que todos os alunos estejam frequentes a menos que algo tenha sido lançado em seu prontuário naquele mês."

* Ou seja, inverti a lógica comumente utilizada (que supunha que ninguém estava frequente a menos que estivesse marcado como tal), porém preservei o resultado final.

* Esta medida simples gerou, de imediato, uma redução do retrabalho da ordem de cerca de 90 a 95% (pois a grande maioria dos alunos estará frequente durante o mes).

Frequência dos alunos (refinamentos)

* Ao continuar a análise foi possível realizar mais alguns refinamentos. Por exemplo, daqueles alunos do mês onde ocorreu algum lançamento, haveriam duas possibilidades

* Todo lançamento de frequência que não for assinalado especificamente como falta, será uma licença. Isto, dentro da programação Ruby (on Rails) ficou assim:

```ruby
def self.leave

  where.not(:id => self.absence)

end
```

Detalhamento

* Remunerada (paga)

```
def self.paid_leave

  leave.joins(:leavetype).merge(Leavetype.paid)

end
```

* Pendência

```
# Not confirmed - management authorization pending (e.g. paid leave)
def self.pending
     where.not(id: self.confirmed)
end
```

Segundo exemplo de aplicação da matemática (conferência externa)

* Dados de frequência e matrícula dos alunos e também de pagamento são reconferidos externamente, utilizando-se a linguagem R (programa RStudio, em sua versão comunitária).

* Exemplo de inovação: já são registradas dentro do sistema as matrículas ditas "especiais", de reposição e reingresso. Esta funcionalidade está sendo aperfeiçoada. Aqui foi utilizada a teoria dos conjuntos. Sabemos com 100% de certeza que todas as matrículas ou são normais ou especiais (subconjuntos mutuamente exclusivos).

Agradecimentos

* Prefeitura de São Paulo, Secretaria de Desenvolvimento, Trabalho e Empreendedorismo, pelo convite.

* Sindicato dos Engenheiros no Estado de São Paulo.

Sobre o palestrante

* Matemático, arquiteto de sistemas/desenvolvedor, tradutor.
* Bachelor of Science (Mathematics), University of Maryland at College Park, 1993
* Master of Business Administration (Information Systems), The American University, 1997
* Mestre em Letras (Italiano), Universidade de São Paulo, 2008

Contatos (e-mail)

institucional

mredoschi@saude.sp.gov.br

Executivo Público

pessoal

mredoschi@uol.com.br