

Computadores, Softwares e Patentes

Pedro Antônio Dourado de Rezende *

Hudson Flávio Meneses Lacerda **

II Conferência Latino-americana e do Caribe sobre
Desenvolvimento e Uso de Software Livre da UNESCO ***

Setembro de 2005

Índice

Começo

Introdução

Evolução

Radicalização

Meio

Avaliação

O cenário Brasileiro

A natureza da relação entre hardware e software

Final

A relação semiológica entre hardware e software

Arquitetura de Von Neumann

A Santa Inquisição

Bibliografia

Autoria e direitos autorais

Começo

Introdução

As discussões em torno da validade do patenteamento de "invenções implementadas por computador", ou seja, das chamadas patentes "de software", têm ganhado importância estratégica não só na sua dimensão econômica, mas também social e geopolítica. Não somente mas sobretudo na Europa, palco de disputas em torno da "Diretiva sobre a [patenteabilidade](#) de invenções [implementadas por computadores](#)" [1, 2], recentemente rejeitada pelo Parlamento Europeu.

Conceito comum aos regimes jurídicos que hoje abarcam processos produtivos, patentes são consideradas aplicáveis a invenções novas, não óbvias e úteis para aplicações industriais. Excluem-se do conceito, isto é, da patenteabilidade, as criações artísticas, as teorias científicas e as fórmulas e métodos matemáticos, entre outros. Nessas fórmulas e métodos é que se assentam as estruturas operativas das tecnologias

de informação e comunicação (TIC). Daí, os efeitos das chamadas patentes "de software" afetam a função sócio-econômica dessas tecnologias, inclusive através de outros regimes jurídicos.

As práticas observadas na concessão de patentes, cada vez mais questionáveis, reflete, de forma cada vez mais direta, pressões protecionistas de importantes atores no regime produtivo de bens simbólicos dito proprietário, pressões associáveis à [crescente ineficiência](#) deste regime frente ao regime [alternativo emergente](#), dito livre ou de código aberto. No processo evolutivo das TIC, regimes produtivos têm exibido ciclos de eficácia ou de predominância bastante regulares, e atualmente, o crescimento da conectividade e da informatização marca o movimento tectônico das fronteiras de eficiência entre esses [3, 4].

O regime proprietário é assim chamado por tratar cada cópia de um bem simbólico (por exemplo, software) como [propriedade do fornecedor](#), e, nos seus modelos de desenvolvimento e licenciamento, sua matriz geradora (código fonte, no caso do software) como segredo de negócio ou sob forte controle do conhecimento de seu funcionamento interno [5]. O regime livre ou de código aberto é assim chamado (pelo acrônimo FOSS, no caso do software) por tratar o licenciamento de cada cópia como ato benéfico, e sua matriz geradora como linguagem técnica, acessível à competência de potenciais interessados, concentrando seus modelos negociais em serviços. Embora o conhecimento seja vantagem competitiva em ambos os regimes, no segundo ele é livre, e esta liberdade, fundamental.

Evolução

Observa-se que a relação custo/benefício no [regime proprietário se deteriora](#), frente à alternativa, na medida em que se expandem a conectividade, a informatização e os serviços possibilitados pela Internet, e, no caso do software, também a base de código fonte disponível para reuso sob regime FOSS [6]. Os processos produtivos e modelos negociais baseados no regime FOSS independem e prescindem de proteção patentária, mas estão sujeitos à [influência deletéria](#) do seu [uso abusivo](#), cada vez mais praticado por agentes monopolistas ou aspirantes cujas estratégias dependem, por pressão competitiva intra-regime, do acúmulo de patentes desse tipo [7, 8].

Isso porque os modelos negociais baseados no regime proprietário dependem da escassez de bens simbólicos que produzem, escassez que precisa ser artificialmente sustentada frente à crescente abundância e qualidade dos que são concebidos sob regime livre, desenvolvidos e distribuídos sob modelos alternativos, como os do FOSS (software) e os do [Creative Commons](#) [9] (outros tipos de bens simbólicos). Entender a evolução das TIC exige reconhecer a importância estratégica desses modelos, e a tensão entre os de ambos regimes. Para isso, examinemos o processo produtivo do software, por ser ele pioneiro na prova de conceitos do regime livre, e foco desse trabalho.

Boa parte do software hoje produzido se destina a uso específico, caso em que o interessado empreende a produção, e o produto, por isso, não se destina à distribuição. Nesses casos, o desenvolvimento é interno (*in-house*), sob contrato e não sob licenciamento. Portanto, o software assim produzido, sendo de desenvolvimento e uso privado, não é nem livre nem proprietário. Ocorre que, devido à sua natureza, os processos produtivos de software ocorrem sob efeitos de estratificação e dependência a padrões, enquanto os negócios a eles vinculados competem sob o efeito de

externalidades positivas, do chamado "efeito rede".

Por esses efeitos, a interoperabilidade e a instrumentalidade de softwares de uso mais geral e distribuição ampla pautam seus modelos de desenvolvimento. Tanto mais quanto mais genérica for a sua funcionalidade, intermediadora de recursos computacionais e comunicacionais para outros softwares, com os quais interage através de padrões e formatos de conhecimento mútuo dos desenvolvedores. No estágio atual dessa estratificação, software não é apenas produto; é também, em muitos casos, meio de produção. Por isso, os modelos de desenvolvimento e licenciamento sob os quais são produzidos os sistemas operacionais tendem a se impor para as camadas de software que, sobre eles, acumulam funcionalidades [10] (neutralizar essa tendência é um dos desafios das chamadas "máquinas virtuais").

Para enfrentar a deterioração da eficiência do regime proprietário na base dessa estratificação, a estratégia dos agentes -- monopolistas, aspirantes e parasitas -- que apostam na sobrevivência predominante desse regime se direciona para o acirramento de restrições sobre o uso de processos elementares e essenciais à produção e utilização de bens simbólicos, especialmente em relação a software, por intermediar os demais. Esses apostadores encontram seu caminho no regime patentário. Por enquanto, lhes basta que uma nuvem de incerteza paira sobre a questão de quais desses usos poderiam violar quantas das dezenas de milhares de cartas patentárias hermética e obscuramente lavradas, que eles acumulam (supostamente para proteger produtos sob forte controle do conhecimento do seu funcionamento interno, p.e., Windows XP [11]).

Mas essa nuvem tem efeito temporário; a estratégia precisa se radicalizar, forçando uma escalada abusiva do regime patentário. A escalada abusiva desse regime jurídico, concebido para incentivar a criatividade humana, desvirtua sua meta ao transformar seus instrumentos, cartas patentárias que outorgam exclusividade para a exploração comercial de processos simbólicos, em barreira de entrada a novos atores nos mercados afetados, em armas de intimidação e chantagem contra concorrentes, em moeda podre para barganhar alianças entre ou intra-cartéis já estabelecidos [12]. Esses efeitos deletérios, associados à conseqüente amplificação do "efeito rede" em mercados naturalmente monopolizantes, provocam uma verdadeira "guerra fria" que inflaciona o custo social das TIC [13], faturado a uma sociedade cada vez mais delas dependente [14, 15, 16].

Radicalização

Nos EUA, sob pressão de *lobbies* de poderosos agentes da indústria, da especulação e da acumulação financeira, a interpretação do que seja invenção patenteável -- isto é, idéias úteis, não óbvias e inéditas com aplicações industriais -- vem se ampliando e já abarca não apenas "processos implementáveis por programas de computador" (software), mas também métodos comerciais e até seres vivos, seguindo a orwelliana palavra de ordem pela avareza regulatória: "qualquer coisa sob o sol criada por humano deve ser patenteável" [17].

Ademais, tal radicalização concentra poder não apenas em torno de questões relativas à patenteabilidade. Também em questões jurisdicionais, com os apostadores buscando ultrapassar as fronteiras norte-americanas e impor globalmente sua esfera de influência [18], através de acordos internacionais de "livre comércio" [19, 20], marqueteados com a cenoura neoliberal da prosperidade presa à ponta da vara da concentração financeira.

Já no outro lado do Atlântico, sob tal pressão o Parlamento Europeu decidiu, em 06.06.05, pela [rejeição](#) de uma "[Diretiva](#) sobre a patenteabilidade de invenções implementadas por computador", por ampla maioria: 648 (cerca de 95%) dos 680 votos. Houve apenas 14 votos a favor da Diretiva e 18 abstenções [21, 22]. Esses números, entretanto, não refletem o fato de que tal decisão foi o desfecho de uma batalha política acirrada, repleta de dramatismo, abusos de autoridade, marchas e contra-marchas, como numa versão câmera-lenta, setorializada e parlamentarizada da "batalha de Seattle", travada em torno de espaço midiático por ocasião da assembléia mundial da Organização Mundial do Comércio em 1999.

Na versão apresentada na [Primeira Leitura](#) pelo Parlamento Europeu, em setembro de 2003 [23], esta Diretiva explicitava restrições que claramente excetuavam da patenteabilidade as tais "invenções implementáveis por computador", em harmonia com interesses de [pequenas e médias empresas](#) [24], [economistas](#) [25], [cientistas da computação](#) [2] e [desenvolvedores de software](#) [26]. No entanto, em maio de 2004 a Comissão e o Conselho de Ministros da União Européia [alteraram o texto](#) visando a atingir o [fim diametralmente oposto](#): o de fornecer sustentação legal para esse tipo de patente [27, 28].

Em 07.03.05 o Conselho Europeu, desrespeitando as regras de procedimento e desconsiderando a falta de apoio da maioria qualificada dos estados-membro, [tentou forçar](#) a adoção do texto da Diretiva [29] e evitar a submissão do seu texto a Segunda Leitura pelo Parlamento. *Lobbies* de grandes empresas (como Microsoft, IBM, Nokia, SAP e Siemens) impuseram dificuldades à reedição de emendas contra as patentes "de software", o que acabou levando o Parlamento, em Segunda Leitura, a rejeitar a Diretiva em 06.06.05. Com isso, findou-se o processo legislativo em torno da Diretiva, continuando a vigor a legislação anterior.

Meio

Avaliação

Proponentes da Diretiva alegam, agora, que seu objetivo não envolvia a ampliação dos critérios de patenteabilidade, mas simplesmente esclarecer a lei atual e harmonizar sua aplicação na Europa. No fundo, intentavam dar reconhecimento à [prática questionável, mas recorrente](#) no Escritório de Patentes Europeu (EPO), de -- hoje sem qualquer base legal sólida -- conceder patentes "de software em si", a pretexto de se tratarem de "invenções envolvendo o uso de programas de computador, (...) que apresentem uma 'contribuição técnica' e preencham os requisitos normais de patenteabilidade (...)" [30]. Doutra parte, organizações como a [Fundação para uma Infraestrutura de Informação Livre](#) [31, 32] mantém catálogos de patentes abusivas desse tipo, e de [estatísticas relacionadas](#) a seus efeitos deletérios para a sociedade [33].

Fosse o objetivo dos seus proponentes o de reforçar e esclarecer a atual situação legal, a Diretiva certamente não teria sido rejeitada pelo Parlamento. Sua rejeição sugere outra avaliação: a de que a citada prática do Escritório Europeu de Patentes (EPO) é indesejável e não reconhecida pelo Parlamento. Essa explicação é consistente com a posição do Parlamento em Primeira Leitura -- claramente contrário às chamadas patentes "de software" -- e com a representativa rejeição da última redação da Diretiva

em Segunda Leitura, na votação de 06.06.05.

Uma leitura objetiva mostra que, em sua última redação, a Diretiva se voltava claramente para legitimar supranacionalmente as ações do EPO, que vem concedendo patentes "de software" em detrimento do texto do [artigo 52 da EPC](#) (Convenção Européia de Patentes) [34], tornando-as, em contextos litigiosos na Europa, tão frágeis como nos EUA, onde cerca de [metade dos litígios](#) envolvendo patentes "de software" que vão a julgamento as invalidam [35, 36]. A aprovação da Diretiva ampliaria o âmbito das invenções patenteáveis para abarcar também "invenções implementadas por computador", colocando todos os países de União Européia sob os efeitos de tal regime, bem como o de patentes "de software" já concedidas ou por conceder.

Após a votação no Parlamento Europeu, o discurso dos que apostaram na radicalização passou a expor uma de duas posições antagônicas, conforme a conveniência. De um lado comemoram, quando lhes convém, a rejeição da Diretiva, como se a omissão da instância legislativa em manifestar-se sobre as práticas atuais significasse a validação das patentes irregulares ou abusivas já concedidas; doutro lado lamentam, quando lhes convém, a oportunidade perdida para se "harmonizar" tais práticas (os escritórios de patentes não ganharam autoridade adjudicadora), já que suas crescentes inconsistências e absurdos mantêm elevado o risco dessas patentes serem invalidadas em ações litigiosas, e portanto, elevado o tal "custo da proteção da PI".

O caminho apontado pelos que apostam na proposta de radicalização global dos regimes patentários, à guisa de uma alegada necessidade de se legitimar o *status quo*, a reboque da "harmonização" desses regimes, a pretexto da necessidade de se "racionalizar o custo da proteção" (como se esse fosse o único meio), nos levaria a uma posição onde a instância legitimadora da proposta seriam aquelas internas ao seus próprios aparatos cartoriais ("[Boards of Appeal](#)" [37, 38]). Porém, ou talvez por isso, esse caminho é [minado por sofismas](#) armados em [vagas expressões](#) e clichês, do tipo "invenções implementadas por computador" e "contribuição técnica" [39, 40]).

Caminho que traria poder de monopólio hermenêutico aos escritórios de patentes e seus agentes (sob o modelo do EPO/Boards of Appeal), isto é, poder para interpretar cartas patentárias e, indiretamente, as leis que as regem e o debate legislativo que cria essas leis. Poder de suprimir e censurar interpretações que indiquem irregularidades e abusos, poder para legitimar argumentos autoritários, que invertem o ônus da prova sobre possíveis efeitos de propostas, ou do tipo *ad hominem*: "devido à complexidade e especificidade, só quem entende e pode falar de patentes e de leis de patentes são advogados de patentes". Poder, enfim, de promover a ideologização do Estado como servo apenas dos interesses do capital, o ideário fascista.

Surdos ao sentimento que a sociedade européia acaba de expressar em sua instância legislativa, e cegos aos efeitos sociais negativos de sua própria ação nos mercados, os apostadores nessa radicalização se mostram inclinados a dobrar suas fichas. A harmonizar ainda mais a sua conduta com o papel ilegítimo que as patentes sobre processos simbólicos (como software) cada vez mais exerce, em oposição ao de sua alegada função social. Diante da resistência encontrada, sua proposta levaria, ainda à guisa de "mais estímulo à inovação", a maiores [abusos](#) do poder econômico por acumuladores de patentes esotéricas [41], maior eficácia no uso destas como arma de intimidação e chantagem, num jogo ardiloso e obsessivo pela reedição da corrida ao ouro alquímico, até aos intestinos da galinha Internet, poedeira que inovou como nunca dantes inovado, sem precisar ou contar com tais "estímulos" mas correndo sério risco de com eles se decompor.

A Europa [pôde se livrar](#), por algum tempo, da [radicalização do regime patentário](#) mas ainda tem por cumprir a tarefa de extirpar-se do efeito deletério das patentes abusivas já concedidas ou por conceder [42, 43], e de se preparar para a próxima batalha.

O cenário Brasileiro

O Brasil vem se destacando internacionalmente por ações governamentais em favor do regime FOSS, bem como por ter em conta do interesse público a determinação de uma política de propriedade intelectual [equilibrada](#) e sensível aos direitos de [acesso e expressão do conhecimento](#), conquistados no Iluminismo [44, 45]. Não obstante, [reações](#) a favor do *status quo* monopolista nas TICs e noutros setores de bens simbólicos, e da radicalização do regime patentário por eles promovida, pululam na mídia corporativa [46, [47](#), [48](#)]. Em conseqüência, a questão das patentes "de software" cresce em importância, merecendo abordagem analítica que possa se contrapor à retórica quase sempre vazia de sustentação empírica, a municiar essa radicalização com não mais que [conjecturas econômicas intestáveis](#). [49, [50](#)]

Em 04.05.05, o Sr. Antonio Carlos Souza de Abrantes, examinador de patentes do INPI, postou na lista pública de e-mails PI-Brasil (Propriedade Intelectual no Brasil) um artigo que busca incentivar o patenteamento de processos implementáveis por programas de computador. Publicado também na revista eletrônica [ComCiência](#) [51], o [debate por ele gerado](#) na lista PI-Brasil [52] motiva, por sua rarefação semiológica, este trabalho. O argumento central usado pelo Sr. Abrantes, em defesa do patenteamento de programas de computador, é encontrado nestes parágrafos:

"No debate sobre patentes de software muitos críticos tem colocado como argumento uma suposta inadequação do sistema de propriedade industrial. O programa de computador comporta dois aspectos. O primeiro é o relativo às expressões literais da idéia, como o programa fonte ou objeto e o programa executável. Esta criação expressa literalmente a idéia através de um conjunto de instruções. A proteção de tais elementos do software, na medida em que atendam ao quesito de originalidade e não sejam ditadas por questões de funcionalidade ou por outras limitações operacionais são objeto de proteção pelo direito autoral. O segundo aspecto comporta os elementos não literais do programa de computador, ou seja, seus aspectos funcionais, suas características técnicas operacionais expressas por métodos e sistemas que são passíveis de proteção por patentes. Uma criação industrial relativa a programa de computador será considerada invenção desde que a criação como um todo apresente um efeito técnico, isto é, venha a resolver um problema encontrado na técnica, que não diga respeito unicamente à forma como este programa de computador é escrito, isto é, ao programa de computador em si.

O INPI tem considerado portanto como patenteáveis os programas de computador que evidenciem um efeito técnico novo, e que portanto não podem ser considerados como programas em si. Diretrizes de exame do início da década de 90 já estabeleciam tal conceito: "A concessão de patentes de invenção que incluem programas de computador para realização de um processo ou que integram equipamentos que realizam tais processos tem sido admitidos pelo INPI há longos anos. Isto porque não pode uma

invenção ser excluída de proteção legal, desde que atendidos os requisitos convencionais de patenteabilidade, meramente pelo fato de que para sua implementação utilizem programas de computador. Assim o programa de computador em si é excluído de proteção patentária, todavia, se o programa controla a operação de um computador mesmo convencional, de modo a alterar tecnicamente o seu funcionamento, a unidade resultante do programa e do computador combinados pode ser uma invenção patenteável como método ou dispositivo".

Tal argumentação segue a mesma linha de uma interpretação pela qual a Suprema Corte dos EUA teria decidido, no caso [Diamond versus Diehr](#) em 1981 [53] e em [subsequentes reinterpretações](#) [71], que o uso de um programa de computador num processo físico era insuficiente para tornar sua essência impatenteável. Uma reinterpretação distorcida por sofismas dessa decisão -- o argumento de que bastaria a um método matemático (ou essência lógica de um trecho de programa de computador) agregar "algo mais" a um processo para tornar tal método (ou essência lógica) patenteável -- acabou por abrir as portas ao patenteamento de idéias que perfazem a essência lógica de programas de computador -- algoritmos --, disfarçados de novos processos físicos implementáveis por computador.

A repetição desse vício, reforçado por *lobbies* de grandes escritórios de patentes e agentes com interesse mútuo no poder que esse tipo de relativismo jurídico lhes acumula, conduziu, começando pelos EUA, a jurisprudências e retoques regulatórios que acabaram por validar, na prática, o chamado [patenteamento "de software"](#) [54]. Isso tem levado à concessão de patentes de qualquer coisa que se disfarce de "contribuição técnica", e, através delas, ao uso abusivo e deletério dessas cartas patentárias esotéricas, cujo uso primordial -- moeda podre de barganha -- leva a pressões jurisdicionais, que os marqueteiros desse jogo chamam de "harmonização", à guisa de "racionalização do custo da proteção da PI", como acima citado.

Tais pressões surtem efeitos. Da mesma maneira que na tortuosa interpretação do caso *Diamond versus Diehr*, e se chocando com a posição diplomática brasileira em negociações internacionais sobre o tema, o Sr. Abrantes omite o passo essencial de avaliar a função específica que um programa de computador exerce em um processo cuja patente é reivindicada, em sua apologia à radicalização patentária:

"...Uma criação industrial relativa a programa de computador será considerada invenção desde que a criação como um todo apresente um efeito técnico, isto é, venha a resolver um problema encontrado na técnica, que não diga respeito unicamente à forma como este programa de computador é escrito, isto é, ao programa de computador em si...."

E resume, citando o que seria diretriz de exame do INPI:

"[N]ão pode uma invenção ser excluída de proteção legal, desde que atendidos os requisitos convencionais de patenteabilidade, meramente pelo fato de que para sua implementação utilizem programas de computador."

Destarte o Sr. Abrantes confunde ou ignora, nesse argumento, a diferença entre:

- Uma invenção que se utiliza, em algum aspecto específico, de um programa de computador como mecanismo de execução e/ou controle,
- A essência lógica de um programa ou trecho de programa de computador

(algoritmo) que implemente tal execução e/ou controle.

Não se pode confundir meio e fim. Antes, não pode uma invenção ser confundida com o meio digital de se implementar seu mecanismo de controle ou execução. Nesse contexto, tal confusão ou ignorância promove das duas uma. Ou repetidas e fictícias reinvenções do computador, para grilagem de idéias de uso geral através da concessão de patentes esotéricas ou abusivas, ou afrontas à [Lei 9279 de 1996](#) (Lei de Propriedade Industrial), que expressamente excetua de patenteabilidade (no seu artigo 10º) os programas de computador em si (inciso V), concepções puramente abstratas (inciso II) e métodos matemáticos (inciso I), como os algoritmos (tal qual são conhecidos na na Ciência da Computação [55]), como pretendemos mostrar.

Diz ainda o Sr. Abrantes, continuando a citar o texto que faria parte de uma diretriz de exame do INPI:

"[...] Assim o programa de computador em si é excluído de proteção patentária, todavia, se o programa controla a operação de um computador mesmo convencional, de modo a alterar tecnicamente o seu funcionamento, a unidade resultante do programa e do computador combinados pode ser uma invenção patenteável como método ou dispositivo."

Nesse ponto, onde o Sr. Abrantes enlaça a natureza técnica dos computadores com a dos programas que neles executam, torna-se necessária uma abordagem menos tosca e pueril sobre o que seja "funcionamento técnico" desses artefatos, e sobre a natureza da relação que embasa a compreensão desse funcionamento, a saber, a relação entre hardware e software. Noutras palavras, há que se ater à relação entre os componentes lógico e físico dos computadores para se alcançar os possíveis sentidos, lacunas e falácias no argumento oferecido pelo Sr. Abrantes.

A natureza da relação entre hardware e software

Para abordar tal relação, necessitamos antes observar dois fatos esclarecedores:

1- Nenhum programa que faça uso de hardware, ou seja, que não esteja imerso (*embedded*) no hardware, altera tecnicamente o funcionamento do hardware onde executa. Tais programas apenas manipulam e interpretam símbolos representados por sinais binários (bits) que trafegam por e/ou se armazenam nesse hardware.

2- A função técnica do hardware é processar sinais. Sinais *per se* nada significam, porque não são símbolos. Sinais veiculam símbolos através de códigos. Portanto, o significado das seqüências de bits manipuladas por esses programas é função semiológica das camadas de código que a lógica e ontologia dos demais programas envolvidos permitem aos seus autores representar.

O funcionamento técnico de um hardware é determinado na sua fabricação, explorável através do domínio de sucessivas camadas de codificação, por programadores que podem criar programas independentemente. As simbolizações que dão funcionalidade a programas de computador só se estabelecem pelo e no contexto dos demais programas envolvidos, inclusive "programas cognitivos" imersos na cultura dos usuários. Assim, o significado das seqüências de bits manipuladas por esses programas não só não altera,



como também não pode ser entendido como parte da função técnica do hardware onde executa. Da mesma forma que um disco não altera o funcionamento técnico de um toca-disco, ou, uma nova partitura não altera o funcionamento técnico de um violino. Quem se dispuser a refutar tal assertiva terá para si a tarefa de analisar os argumentos desenvolvidos no restante deste artigo, a começar pelos 16 pontos a seguir:

3- Para entender o funcionamento técnico de um computador, há que se começar pelo entendimento do que seja "técnico", e pelo funcionamento técnico do que há de mais essencial nos computadores, a saber, o componente de hardware conhecido por *processador* (de sinais digitais). Se quisermos o adjetivo "técnico" lastreado na filosofia do Direito, ao invés de enganchado na sofística do fascínio coletivo contemporâneo, devemos ter em mente, seguindo Kant, que o termo em sua origem, *techné*, tem por essência um processo causal, no sentido aristotélico ou tomístico de *causa eficiente*. Assim, o funcionamento técnico de um processador consiste na interpretação de seqüências de sinais. Nos computadores de hoje esses sinais são binários, denominados bits (*binary digits*), fisicamente representados pela presença ou ausência de voltagem. Uma seqüência de sinais que seja reconhecível para interpretação representa uma *instrução*.

4- A cada instrução corresponde um ação que o processador executa, ao reagir eletronicamente à presença dos sinais que a representam no *registrador de instruções* (grupo de circuitos internos para esta finalidade). A essa atuação do processador dá-se o nome de *interpretação da instrução*. A interpretação de uma instrução reage também a sinais presentes em outros circuitos internos do processador. Uma instância dessas presenças é normalmente chamada de *estado*, e os circuitos onde eles normalmente ocorrem são chamados de *registradores de dados*. Os sinais binários podem ser entendidos como representações de símbolos, normalmente *zero* e *um*, de sorte que cada instrução se faz corresponder a uma função simbólica. Conseqüentemente, cada interpretação de instrução pode ser entendida como o cálculo da respectiva função simbólica, para a instância de dados representada pelo estado corrente do processador.

5- Ao interpretar uma cadeia de instruções, o estado em que o processador inicia a interpretação de uma instrução é o que resulta das interpretações anteriores. A isso, chamemos *comportamento* do processador. O conjunto das instruções reconhecíveis, e suas respectivas interpretações, constitui uma espécie de "dicionário" chamado *código de máquina* do processador. Por sua vez um encadeamento de instruções, constituindo uma espécie de "frase", é dito um *programa*. A interpretação de uma cadeia de instruções, em código de máquina e perante uma instância de dados, constitui uma *execução* do respectivo programa. As possíveis interpretações dessa cadeia, que dependem dos dados, formam o *comportamento do processador perante o programa*; ou, pela ênfase, *comportamento do programa no processador*.

6- Nas camadas de codificação que venham a se sobrepor a esse comportamento, cada estado ou função simbólica pode ser entendida como elemento do processamento de dados (numéricos, lógicos ou literais, por exemplo) que o programa executa. Essas codificações são escolhas combinatórias e representacionais que agregam valor semiológico (portanto utilitário) a programas executáveis. Embora esses programas possam interpretar esses códigos (para cores, por exemplo) ao executar, o código de máquina não pressupõe, produz ou determina nenhum deles. Por isso, códigos que se sobreponham ao do processador não influem no seu funcionamento técnico (como causa eficiente). Na verdade, o código de máquina nem mesmo determina um programa em si: se o encadeamento de instruções do programa for abstraído da sua representação em código de máquina, tem-se o programa (em código) fonte. O programa fonte é composto de algoritmo (essência lógica do encadeamento) e estrutura de dados, expressáveis em linguagem mais próxima às humanas coloquiais.

7- Devido ao fato da coleção dos programas executáveis, e do repertório dos seus comportamentos, serem determinados pelo processador, e devido à natureza desses conceitos, o código de máquina do processador é causa material dos possíveis programas nele executáveis. E cada programa executável é causa eficiente do seu comportamento no processador. Assim, mesmo que não seja possível registrá-las em burocracias cartoriais do Direito industrial, especialmente a 30 centavos por folha de papel impressa, tal coleção e tal repertório, infinitos em potência, têm como causa formal o código de máquina do processador. Ou seja, traduzindo Aristóteles e Tomás de Aquino para informatiquês, a coleção de programas executáveis e o repertório dos seus comportamentos, isto é, o funcionamento técnico do processador, são fixos e determinados quando a matriz dos circuitos eletrônicos do *chip* do processador é determinada, e por ela.

8- O funcionamento prático do processador, isto é, a utilidade do repertório de comportamentos de programas nele executáveis, limitado por fatores como tempo de execução, capacidade de armazenagem e de fluxo dos demais dispositivos do computador, e codificações que se sobreponham para os dados manipulados, em nada afeta a relação causal entre o projeto do processador e seu funcionamento técnico, entre a matriz dos seus circuitos eletrônicos e o repertório de comportamentos dos programas nele executáveis. Doutra feita, esta relação causal é estímulo, muito mais do que qualquer proteção patentária, à criatividade inovadora na produção de programas úteis.

9- A obra intelectual de se produzir (e testar) um programa útil, inédito na forma dentre os executáveis já em uso, no Brasil e pela convenção de Berna é regida pelo direito autoral, que protege o autor da obra contra uso indevido de suas representações (em código fonte ou executável). E não, *per se*, pelo direito industrial, que daria a um titular de patente exclusividade para exploração comercial de alguma suposta funcionalidade inédita, contra programas cujo comportamento indevida e alegadamente exibam tal funcionalidade, qualquer que seja a sua utilidade, forma ou código em que esteja representado.

10- Doutra feita, dizer que o funcionamento técnico de um processador é alterado por um programa nele executável, ou que um tal programa altera esse funcionamento técnico, não faz sentido exceto no ilusionismo, já que a autoria, representação ou conhecimento de um tal programa em nada altera a coleção dos executáveis à qual ele já pertence, por definição do projeto do *chip* do processador. Tampouco faz sentido dizer que a funcionalidade de um tal programa altera o repertório dos possíveis comportamentos do processador. Por outro lado, se a finalidade (causa final) de um tal programa altera a utilidade do processador onde executa, isso em nada modifica seu funcionamento técnico, pois em nada afeta as causas eficientes, aqui explicadas, que o processador é capaz de instrumentar.

11- Pode-se perguntar, então, que programas poderiam alterar o funcionamento técnico de um processador. Somente as instruções propriamente ditas, fisicamente expressas por circuitos eletrônicos no desenho do *chip* e nele imersas na fabricação, e não as seqüências de suas representações binárias, pode-se admitir que alterem esse funcionamento: mas apenas no estrito sentido em que determinam o código de máquina do processador, através do código subjacente dos circuitos eletrônicos, quando inseridos no desenho do seu *chip* durante a etapa de projeto.

12- Mas já a partir da imersão da matriz desses circuitos no *chip*, durante a etapa de fabricação, o comportamento do processador (que contiver o *chip*) perante as seqüências de instruções do seu código de máquina, isto é, o comportamento dos programas nele executáveis, ou, noutras palavras, o funcionamento técnico do processador -- no sentido tomístico-kantiano e não mágico de "técnico" --, estará fixo e determinado, só se alterando se o *chip* for danificado. O mesmo se pode dizer do funcionamento técnico dos dispositivos de memória, cujo comportamento corresponde à armazenagem de seqüências de bits, limitadas em comprimento à sua capacidade de armazenamento. O mesmo, também, do funcionamento técnico de um hardware formado por processador, memória e placa integradora, ou do funcionamento técnico de um computador de uso genérico, ou de uma rede de computadores, etc. A diretiz do INPI inverte, portanto, uma relação de causa que não é inversível.

Final

A relação semiológica entre hardware e software

A relação entre hardware e software só se define integralmente na esfera semiológica. A investigação desta relação começa pela seguinte pergunta: Como pode a escolha de um conjunto de instruções, durante o projeto de um processador, determinar o repertório de seus comportamentos diante dos programas nele executáveis, e diante das possíveis codificações que possam a eles se sobrepor? A resposta mais abrangente e completa possível é apontada não por sofismas, mas pelo método científico:

13- Pela [tese de Church](#) (ou Church-Turing) [56], se o processador for minimamente capaz de executar operações aritméticas, e se o hardware

incluir dispositivos de entrada e saída de dados digitais (como por exemplo, um computador ou uma rede de computadores como hoje conhecidos), o repertório dos possíveis comportamentos deste hardware, relativo às funções simbólicas que representam, será o mesmo qualquer que seja o restante do código de máquina do processador, e qualquer que seja a utilidade de qualquer programa nele executável. Noutras palavras, a coleção dos programas executáveis num computador, ou, o seu funcionamento técnico, se limita e se fixa maximalmente por uma ocorrência mínima de instruções aritméticas no código de máquina do(s) processador(es) e de dispositivos de entrada e saída de dados digitais compondo o hardware.

14- O formalismo matemático pioneiro na literatura científica para descrever tal coleção de programas computáveis, possíveis de serem executados por qualquer computador ou rede de computadores habilitados à aritmética elementar e à adequada manipulação de dados, foi desenvolvido e apresentado por [Alan Turing](#) em 1936 [57]. O dispositivo teórico pelo qual Turing descreve funcionalmente esses programas é hoje chamado [Máquina de Turing](#) [58], e o repertório de seus comportamentos, pela chamada máquina universal de Turing.

15- A idéia básica do dispositivo universal de Turing é a de permitir a representação e o armazenamento concomitante de programas e dados. Qualquer programa executável é nele representado por meio de sinais indistinguíveis daqueles que representam os dados a serem manipulados. Donde o termo computador *programável* (oposto a *programado*), esse *fiat lux* que permite a sobreposição de códigos e a estratificação de funcionalidades simbólicas. Em linguagem contemporânea, podemos dizer que a máquina universal de Turing plantou a semente do conceito de sistema operacional, programa básico cuja função é interpretar qualquer programa executável no processador, em cujo desenho ele (sistema operacional) seja representável ou esteja imerso.

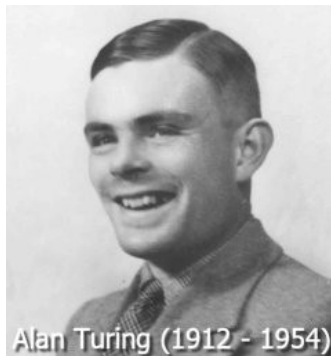
16- A tese de Church surge quando outros formalismos lógico-matemáticos propondo descrever coleções maximais de funções computáveis se revelam, através do lambda-cálculo, rigorosamente equivalentes à descrição funcional de Turing. Quando formalismos tais como o das funções recursivamente enumeráveis, o do cálculo de predicados de primeira ordem e o sistema de reescrita de Post, propostos como novas abordagens ao conceito de computabilidade, acabam por definir como computáveis as mesmas funções simbólicas correspondentes aos programas executáveis pela máquina universal de Turing.

17- [Alonzo Church](#) [59], orientador de Turing no seu doutorado em matemática em Princeton, com base na equivalência dos formalismos até ali investigados postula, então, que todo e qualquer formalismo matemático, conhecido ou não, que se proponha a definir o que seja computável equivalerá à definição funcional da máquina universal de Turing. Trata-se de uma conjectura que veio coroar o esforço da escola filosófica formalista, no sentido de depurar e consolidar os fundamentos da

lógica e da matemática diante de questionamentos metafísicos sobre a consistência desses fundamentos, fecundados pela revolução científica. Surgida no início do século XX sob a liderança do famoso e talvez mais profícuo matemático dos últimos tempos, [David Hilbert](#) [60], a escola formalista revelou os mais fundamentais limites ao positivismo científico, com a publicação dos teoremas de incompletude por [Kurt Gödel](#) [61] em 1931, enquanto dava origem à Teoria da Computação.

18- No caso *Diamond vs. Diher* (de 1981), em que se inspiram os sofistas que propugnam pela legalidade do patenteamento de algoritmos sob os regimes jurídicos vigentes, o dispositivo que teve seu funcionamento técnico "alterado" pelo software em questão não era nenhum computador, que realiza a máquina universal de Turing sob as limitações práticas contingentes; era o equipamento de vulcanização de borracha, objeto da patente, controlado pelo software em questão.

A obra de Turing e Church delimita os sentidos nos quais software e matemática podem se distinguir. Ao mesmo tempo, [fornece a base](#) para uma divisão clara entre, de um lado, objetos e processos físicos que implementam máquinas de estados -- invenções que podem ser patenteáveis -- e, de outro, as informações (programas e dados) que essas máquinas processam -- que deveriam permanecer impatenteáveis [62].



Talvez de maior importância hoje, a obra de Turing também revela a natureza da relação entre programas de computador e procedimentos matemáticos. Programas do computador só se distinguem de processos e operações matemáticas na hermenêutica de subseqüentes e independentes camadas de codificação. A impossibilidade de distingui-los sintaticamente causa, quando ignorada, efeitos colaterais nas jurisdições onde são aceitas patentes "de software". Onde se admite que a descrição do objeto da patente misture algoritmo e hermenêutica do processo a que este se aplica, mas onde o algoritmo, sendo conceito científico e abstrato, é impatenteável se esse processo for simbólico -- doutro algoritmo -- e não físico -- invento em si.

Turing não tentou patentear o dispositivo teórico que ganhou seu nome, até porque a Ciência e as Leis eram então respeitadas pelos escritórios de patentes. Não havia implementação e, além disso, sendo o dispositivo puramente algorítmico (os processos a que se aplica são simbólicos), não era (nem é) patenteável. Mas isso não significa que Turing tenha desprezado sua realização prática ou sua exploração utilitária. Assim que teve oportunidade, envolveu-se no esforço secreto do governo britânico para construir e operar o [Colossus](#) [63], um pioneiro computador programável para criptoanálise da cifra usada pela Alemanha na IIa. Guerra.

Arquitetura de Von Neumann

Todavia, os motivos pelos quais a realização da magna idéia de Turing (sobre como projetar computadores universais programáveis) se tornou impatenteável, seja uma ou muitas vezes, não se assentam nesses fatos, mas em outros dois. Assentam-se na tese de

Church-Turing nunca ter sido refutada, e na trajetória intelectual de um gênio contemporâneo a ambos: o matemático [John Von Neumann](#) [64]. Citamos do [artigo do prof. Tomasz Kowaltowski](#), que homenageia esse fundador da computação moderna [65]:

Para uma grande parte dos praticantes da Computação, o nome de Von Neumann está associado à idéia de arquitetura de Von Neumann, ou seja, à estrutura, hoje considerada clássica, de computadores digitais com programa armazenado na própria memória...O envolvimento direto de Von Neumann com a Computação teve início na década de 1930, quando a disponibilidade de dispositivos eletro-mecânicos (relés) e eletrônicos (válvulas) aceleravam o desenvolvimento de máquinas automáticas de cálculo. Matemático de reputação mundial [também Químico e Físico, criou vários novos ramos da Ciência, como a teoria dos jogos e a teoria ergódica], ... desde 1933 no prestigioso Instituto de Estudos Avançados (IAS) de Princeton, consultor científico de várias agências governamentais ligadas às forças armadas, incluindo o Laboratório de Pesquisas Balísticas de Aberdeen (Maryland) e o Laboratório Científico de Los Alamos (New Mexico), este último responsável pelo desenvolvimento da primeira bomba atômica.

...O contato mais importante e mais frutífero foi com o trabalho de construção do computador chamado ENIAC, desenvolvido por J. Presper Eckert e John Mauchly, na Escola Moore da Universidade de Pensilvânia, sob contrato do Laboratório de Pesquisas Balísticas. O encontro de Von Neumann com a equipe do ENIAC materializou-se em 1944. Na mesma época, a Universidade de Pensilvânia ganhou um contrato suplementar para a construção de uma nova máquina, denominada EDVAC, proposta pouco antes por Eckert e Mauchly, mas cujas características ainda eram muito vagas. O novo projeto despertou enorme interesse de Von Neumann que iniciou uma série de visitas regulares à Escola Moore, participando de reuniões relativas ao projeto, juntamente com Eckert, Mauchly, Goldstine e outros.

Como resultado das reuniões com a equipe de projeto e da freqüente troca de correspondência, Von Neumann ficou encarregado de produzir um documento descrevendo os detalhes da organização da nova máquina. Como indica o próprio título, "First Draft of a Report on the EDVAC", este documento nunca passou da fase de rascunho, tendo sido publicado na íntegra somente vários anos mais tarde, sob forma ligeiramente editada [66].

Existe bastante controvérsia quanto a quem teria sido o primeiro a propor o conceito de programa armazenado. O trabalho teórico de Alan Turing, com o qual Von Neumann estava familiarizado, já indicava esta possibilidade. Por outro lado, existem algumas referências a este assunto, bastante obscuras e ambíguas, em algumas fontes anteriores ao documento produzido por Von Neumann, além das afirmações posteriores de Eckert, Mauchly e outros. Não há dúvida de que a idéia de programa armazenado estava "no ar" e é bastante provável que tenha sido sugerida por mais de

uma pessoa ou nascido no meio de discussões sobre o novo projeto.

... Independentemente de quem tenha sido primeiro a sugerir a idéia de programa armazenado na memória, o fato é que o documento redigido por Von Neumann é a primeira descrição minuciosa e quase completa de uma arquitetura de computador deste tipo, com repertório de operações que permitiriam utilização plena dos seus recursos. O documento é resultado, sem dúvida, das várias reuniões e das trocas de correspondência, mas o próprio fato de ter sido Von Neumann, consultor do projeto, encarregado da sua redação indica a importância e o grau da sua contribuição. De acordo com depoimentos de alguns dos seus colaboradores, o projeto lógico do computador deve-se principalmente a Von Neumann, enquanto que Eckert e Mauchly foram os principais responsáveis pelo projeto de circuitos de alta velocidade, linhas de atraso e outros detalhes físicos.

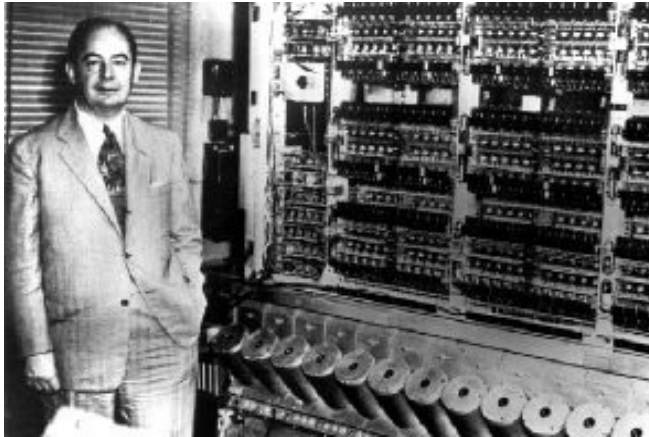
...Nota-se que não existia naquela época uma linguagem adequada para descrever muitos dos conceitos que estavam sendo introduzidos, o que dá ao texto um certo sabor "medieval" sob o ponto de vista da Computação. Por outro lado, é surpreendente a riqueza de idéias, muitas das quais continuam válidas até hoje. Von Neumann separa claramente o conceito de arquitetura lógica do computador da sua implementação física. ... A própria divisão do projeto em unidades de controle, aritmética, memória e de entrada e saída é precursora de todos os projetos posteriores. Na realidade, quase todos os conceitos ainda nos parecem familiares.

O relatório de Von Neumann, apesar de incompleto, teve uma divulgação muito grande e tornou-se um paradigma de projeto para muitas máquinas de primeira geração. O interesse despertado entre instituições de pesquisa e empresas foi tão grande que a Escola Moore organizou, em 1946, um curso sobre a arquitetura do EDVAC (vide [67]). A maior parte das palestras foi apresentada por membros originais da equipe de projeto (Eckert, Mauchly, Goldstine e Von Neumann) apesar de eles não participarem mais da construção da máquina. Um exemplo da importância da influência exercida pelo projeto é a construção da máquina EDSAC na Universidade de Cambridge, por Maurice Wilkes que participou do curso. O EDSAC foi o primeiro computador controlado por programa armazenado a entrar em funcionamento, em 1949.

...Ao desenvolver os projetos lógicos do EDVAC e da máquina do IAS, Von Neumann tinha também uma preocupação muito grande com a sua programação. No caso do primeiro projeto, seu plano original previa a inclusão de exemplos de programação no próprio relatório, que ficou inacabado. Entretanto, existe um manuscrito de Von Neumann que contém o que é quase certamente o primeiro programa escrito para um computador com programa armazenado na memória. Uma [análise detalhada deste manuscrito e da sua história](#) foi feita em 1970 por Donald E. Knuth [68].

...No fundo, apesar de contarmos com uma grande diversificação tecnológica que inclui conceitos como microprocessadores, computação

paralela e distribuída, redes de computadores, interfaces gráficas e outros, os princípios básicos de sua arquitetura e programação ainda são os mesmos derivados das descrições do EDVAC e da máquina do IAS [na imagem, com Von Neumann].



Esse depoimento do prof. Kowaltowski esclarece como a principal idéia de Turing, sobre a possibilidade teórica de computadores universais programáveis, foi realizada por inventos que, sob a direção do gênio de Von Neumann, exploravam os recursos tecnológicos disponíveis a partir do pós-guerra. Inventos postos em domínio público pela forma colaborativa e aberta com que

foram desenvolvidos, como deveria ser pela compreensão vigente sobre a natureza do empreendimento científico.

A radicalização do regime patentário, promovida pelos atuais interesses monopolistas, sofisma em torno da expressão "inovação do funcionamento técnico" desses inventos. Sofismar em torno de qualquer variante dessa expressão, com qualquer das formas utilitariamente evoluídas desses inventos, pode levar a apenas um de dois resultados finais: ou a violação sistemática dos critérios vigentes sobre patenteabilidade, ou a grilagem do legado científico de Turing, Church e Neumann no acervo platônico das formas e idéias perfeitas reveladas.

A Santa Inquisição

Convém lembrar que programas de computador são protegidos no Brasil por [legislação própria](#) [69], com base no [Direito autoral](#) [70], sendo que tanto o regime proprietário quanto o FOSS surgiram com base nesta proteção. A maior prova de que tal proteção, sob escassez de código fonte, sempre foi suficiente e eficaz é a meteórica trajetória evolutiva das TIC até a Internet. E a maior prova de que assim permanece, mesmo sob abundância de código fonte disponível para reuso, é a trajetória de sucesso do regime FOSS.

Doutra feita, o recurso à proteção adicional patentária expõe todo e qualquer desenvolvedor de software -- seja livre, proprietário ou privado -- ao risco artificial de litígio, em se presumir seu produto infrator de quantas das dezenas de milhares de patentes sobre algoritmos, já concedidas ou por se conceder, em situações que, doutra forma, se caracterizariam como livre concorrência comercial e cognitiva.

Idéias, recursos e técnicas que constituem a produção de software são processos mentais, puramente informacionais e cognitivos. Algoritmos são construções lógico-matemáticas que, quando expressos em programas executáveis, ganham dimensão simbólica e potencial valor de uso, no papel de intermediadores da inteligência humana numa sociedade informatizada. Cercar esses bens, naturalmente comunais, com regimes de exclusão, a pretexto de se perseguir a cenoura da prosperidade na ponta da vara da

acumulação monopolista, fará ressurgir na sociedade, antes que seja saboreada, os nefastos efeitos da censura ao saber.

Atribuir critério de patenteabilidade, de forma ilegal ou não, sobre processos simbólicos prejudica mais do que estimula a evolução científica, cultural e o bem comum da sociedade informacional em que vivemos. Assim prejudica ao aviltar a meta que justificou a introdução do regime patentário e ao servir, antes e mais, a um papel ilegítimo: o de proteger práticas monopolistas predatórias nos mercados das TIC, inclusive contra outras jurisprudências que lhe fulcram pelo equilíbrio. Prejudica através da criação artificial de escassez de código disponível como fonte à criatividade humana, escassez útil apenas à sobrevivência de regimes negociais que se obsoletam e perdem eficiência com o crescimento da conectividade, da base de código livre, dos meios de produção dessa base e da versatilidade desses meios, com e pela Internet.

Se um círculo esotérico de operadores do Direito pretende redefinir, pela via autoritária e apoiados em leituras jurídicas toscas e pueris, o que seja Ciência, sofismando em conluio com interesses monopolistas e com governos cooptados, para que se produza escassez artificial de conhecimento disponível em código programável, que tentem. Mas que se lembrem: legalidade e legitimidade são conceitos distintos.

Tanto a Ciência quanto o Direito são construtos sociais. O primeiro, apesar de mais recente, tem se mostrado metodologicamente melhor equipado que o segundo para conduzir seus conceitos e instrumentos pela trilha do tempo. Com agenda semelhante à da radicalização patentária, a Santa Inquisição foi instalada: também sob o guante do segundo, armado de dogmas, para tentar controlar o primeiro. E a História nos mostra as conseqüências.

Bibliografia

- 1- <http://swpat.ffii.de/index.en.html>
- 2- <http://lpf.ai.mit.edu>
- 3- http://www.dwheeler.com/oss_fs_why.html
- 4- <http://perens.com/Articles/Economic.html>
- 5- <http://www.gnu.org/philosophy/categories.html>
- 6- <http://www.fsf.org/licensing/essays/microsoft-new-monopoly.html>
- 7- <http://swpat.ffii.de/patents/effects/index.en.html>
- 8- <http://swpat.ffii.de/pikta/xrani/voip/index.en.html>
- 9- <http://creativecommons.org>
- 10- <http://www.cic.unb.br/docentes/pedro/trabs/nassif1.html>
- 11- <http://www.microsoft.com/windowsxp/pro/eula.msp>
- 12- <http://swpat.ffii.de/vreji/quotes/index.en.html#reback0206>
- 13- <http://observatorio.ultimosegundo.ig.com.br/artigos/eno300520011.htm>
- 14- <http://www.nosoftwarepatents.com/>
- 15- <http://lpf.ai.mit.edu/Patents/industry-at-risk.html>
- 16- <http://www.spectrum.ieee.org/careers/careerstemplate.jsp?ArticleId=i080205>
- 17- <http://www.bio.org/speeches/speeches/041101.asp?p=yes>
- 18- http://www.technetra.com/writings/silicon_valley/innovation_or_patent_colonialism.html
- 19- <http://observatorio.ultimosegundo.ig.com.br/artigos.asp?cod=335OFC002>
- 20- http://news.com.com/Copyright+lobbyists+strike+again/2010-1071_3-5811025.html
- 21- <http://mail.fsfeurope.org/pipermail/press-release/2005q3/000109.html>
- 22- <http://www.eweek.com/article2/0,1895,1834392,00.asp>
- 23- <http://swpat.ffii.de/papers/europarl0309/index.en.html>
- 24- <http://swpat.ffii.de/vreji/cusku/index.en.html>

- 25- <http://www.economic-majority.com>
- 26- http://www.onlamp.com/pub/a/onlamp/2005/07/21/software_pricing.html
- 27- <http://register.consilium.eu.int/pdf/en/04/st09/st09713.en04.pdf>
- 28- <http://swpat.ffii.de/papers/europarl0309/cons0401/tab/index.en.html>
- 29- <http://wiki.ffii.de/Cons050307Pt>
- 30- <http://www.patent.gov.uk/media/pressrelease/2005/0607a.htm>
- 31- <http://webshop.ffii.de>
- 32- <http://www.base.com/software-patents/examples.html>
- 33- <http://swpat.ffii.de/patents/stats/index.en.html>
- 34- <http://www.european-patent-office.org/legal/epc/e/ar52.html>
- 35- <http://www.seriousliving.net/new-2904217-1179.html>
- 36- <http://yementimes.com/article.shtml?i=869&p=culture&a=2>
- 37- <http://legal.european-patent-office.org/dg3/biblio/t971173ex1.htm>
- 38- <http://legal.european-patent-office.org/dg3/biblio/t950333eu1.htm>
- 39- <http://www.ffii.de/~jmaebe/swpat/cii.html>
- 40- <http://swpat.ffii.de/papers/europarl0309/juri0504/ffiiopp050615.en.pdf>
- 41- <http://www.groklaw.net/staticpages/index.php?page=20050402193202442>
- 42- <http://www.guardian.co.uk/online/comment/story/0,12449,1540984,00.html>
- 43- <http://news.zdnet.co.uk/business/legal/0,39020651,39210094,00.htm>
- 44- <http://www.cic.unb.br/docentes/pedro/trabs/OMPI.html>
- 45- http://www.cic.unb.br/docentes/pedro/trabs/bresil_foss.pdf
- 46- <http://www.cic.unb.br/docentes/pedro/trabs/gandra.html>
- 47- <http://www.cic.unb.br/docentes/pedro/trabs/galmeida2.html>
- 48- <http://www.cic.unb.br/docentes/pedro/trabs/fisl2004.html>
- 49- <http://swpat.ffii.de/vreji/mirror/impact/index.en.html>
- 50- <http://wiki.ffii.org/CampVald05En>
- 51- <http://www.comciencia.br/presencadoleitor/artigo19.htm>
- 52- <http://br.groups.yahoo.com/group/pibrasil/messages/2748?threaded=1>
- 53- http://www.technologyreview.com/BizTech/wtr_12074,311,p1.html
- 54- <http://lpf.ai.mit.edu/Patents/against-software-patents.html>
- 55- http://www.mct.gov.br/legis/leis/9279_96.htm
- 56- http://en.wikipedia.org/wiki/Church-Turing_thesis
- 57- <http://www.abelard.org/turpap2/tp2-ie.asp>
- 58- <http://www.turing.org.uk/turing/scrapbook/machine.html>
- 59- <http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Church.html>
- 60- <http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Hilbert.html>
- 61- <http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Godel.html>
- 62- <http://www.spectrum.ieee.org/careers/careerstemplate.jsp?ArticleId=i070305>
- 63- http://en.wikipedia.org/wiki/Colossus_computer
- 64- http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Von_Neumann.html
- 65- <http://www.ic.unicamp.br/~tomasz/projects/vonneumann/>
- 66- John von Neumann: "*First Draft of a Report on the EDVAC*".
Annals of the History of Computing, 15(4):27-75, Outubro 1993.
- 67- <http://ieeexplore.ieee.org/iel4/85/5017/00194089.pdf?arnumber=194089>
- 68- http://portal.acm.org/ft_gateway.cfm?id=356581&type=pdf
- 69- http://www.mct.gov.br/legis/leis/9609_98.htm
- 70- http://www.mct.gov.br/legis/leis/9610_98.htm

Autoria e Direitos Autorais

* Pedro Antonio Dourado de Rezende

Matemático, Professor de Ciência da Computação da Universidade de Brasília (UnB), sendo uma de suas cadeiras favoritas a de Teoria Computação. Coordena o programa de Extensão Universitária em Criptografia e Segurança na Informática da UnB, representa a sociedade civil no Comitê Gestor da Infraestrutura de Chaves Públicas Brasileira, é conselheiro do Instituto Brasileiro de Direito e Política de Informática e membro do Projeto Software Livre Brasil. Portal: <http://www.cic.unb.br/docentes/pedro/sd.htm>

**** Hudson Flávio Meneses Lacerda**

Pós-Graduado em Música Brasileira - Práticas Interpretativas pela Escola de Música da Universidade do Estado de Minas Gerais (UEMG) e Bacharel em Música pela Universidade Federal de Minas Gerais (UFMG). Atualmente, leciona na Escola de Música da UEMG e na Fundação de Educação Artística (FEA), atento aos efeitos deletérios dos abusos do regime patentário em seus interesses artísticos e profissionais. Portal: <http://geocities.yahoo.com.br/hfmlacerda>

***** Conferência UNESCO - LACFREE 2005**

Portal: <http://www.lacfree2005.org/>

Alguns direitos reservados

Pedro Antônio Dourado de Rezende e Hudson Flávio Meneses Lacerda - Artigo sob licença Creative Commons (CC NC-ND-2.0): Livre para republicações com Atribuição, Uso Não-Comercial e Não Derivadas. (Republicação para uso comercial pode ser solicitada aos autores).

Termos da licença: <http://creativecommons.org/licenses/by-nc-nd/2.0/deed.pt>.