(Micro)economics of libre and copylefted Software

Pietro Battiston

April 28, 2011

1 Introduction to Free/Open Source/Libre Software

1.1 Terminology and historical pointers

The term "Open Source" in origin - and literally - means "a software of which the *source* is released". Usually, softwares are constituted of a *source* (or *sourcecode*) which is written by programmers and which, when *compiled*, produces the final form of the software, ready to be distributed and used:[1] the source itself is not indispensible to *use* the software.

However, the source is indeed needed to *modify* the software, or to take some parts of it for use in some other application: it is written in some "programming language" which a programmer can understand, while the compiled form (usually referred to as "*binary* form") is intended to be "understood" only by the device - tipically the computer - on which the software will run, and to a human eye it may resemble random data.[2] Hence a programmer, a team of programmers or a company may choose to (not) release to the public also the *source* of some software they created - the reasons why they may choose to do so are one of the main subjects of this essay and will be investigated later.

Historically, information science stemmed from research departments in the universities, and in the first days *naturally* followed practices that mimicked the functioning of the scientific environment: in particular, cross-institutional and international collaborations were in the beginning limited by the deep differences between the several machines, but when possible *resharing* of results was felt as almost natural. Moreover, computers were still the domain of researchers or other expert individuals, who were accustomed with modifying, adapting and compiling some sourcecode for the particular machines they were working with, or the particular problems they targeted at solving: in these conditions, which lasted grossly

[1] *Today*, more and more softwares are written in so called *interpreted* languages, for which the distinction between source and binary forms basically doesn't hold; however, this observation doesn't interfere substantially with this brief *historical* introduction.

[2] While it may technically be possible to *reverse engineer* a software, which means reconstructing the internal structure starting from the binary form, it can be a difficult and in some cases illegal operation.

speaking from the 50s up to the 70s, the natural form in which to distribute a software was hence the source.[3]

1.2 The birth and raise of the software industry

Later, with the diffusion of softwares which were commercially produced and sold (sometimes, bundled with some hardware) by programmers or software companies, this started to change: the producer of a commercial software tipically receives an amount of money for each copy sold, and has no incentive to publicly release informations about how the software works, or to ease its modification or partial embedding in a new - potentially concurrent - product, which is what may happen by releasing the sourcecode. This observation, coupled with the fact that in the years computers have standardized a lot - and hence a single binary version can fit the needs of many customers[4] - shows clearly that the preferred form in which a commercial software is released is the *binary* one. In fact, this profound change can be detected even in the USA legal system, which extended copyright laws to the protection of software only in 1980.

The software industry has been a substantial engine of development for the field of computers, and in particular for their very rapid diffusion among non-professionals: this is quite natural, since the goal of a producer consists mainly in satisfying the customer, providing exactly the application she needs, while the goals of an academic institutions may consist in solving some problems, or exploring some techniques, without particular attention to the aspect of *usability* for a generic, not specifically skilled, individual. Still, it is fair to say that the form in which the industry develops - with only final products (binaries) being distributed, while the "raw material" from which they come from (the sourcecode) remains secret - causes some evident inefficiencies and wastes, if seen from the point of view of social welfare:

- *replication of research*: solutions adopted in a given software have to be reimplemented and possibly rediscovered by competitor softwares performing similar tasks; while in most areas of technology patents avoid this particular problem - a patent is always publicly released, and hence reduces the risk of duplicate efforts to attain the same goal[5] - a closed source software limits also the *knowledge*, not just the adoptability and/or its economic effectiveness;

[3]In this extreme synthesis of the history of computing, I am omitting some particulars, such as the fact that for a long time the distinction between hardware and software was not sharp as it is today, and in the first days of computing even less sharp was the distinction between source and binary. Suffices to say that the "art of computing" did take shape in the academic environment, and that this imprinting was still absolutely lively when some standardization made it possible to effectively *distribute* a software to users in different parts of the world.

[4]And more importantly, the vast majority of customers can be reached by providing a small number of different binary versions.

[5]I am not here assessing that patents are a solution for the software industry, and will discuss this issue in more detail later.

- *unadaptability*: computers are used today in an enormous variety of environments and for an enormous variety of goals, and hence it is natural to expect a high level of differentiation in softwares; while by sharing the development process even small niches of use cases can put in place synergies, with the majority of some software being developed in common and relatively simple modifications being coded *ad hoc*, a company may judge economically ineffective to customize a successful product specifically for some small segment of the market, and on the other hands smaller companies may not have the human power to create an equivalently powerful product,

- *unsecurity*: no software is perfect, and it is quite common that *bugs* - misbehaviours due to programming errors - are found, that make it behave differently than what was planned and possibly also represent a security risk for computer systems exposed to the public. Open source software can be audited by anyone with the needed programming skills by just getting the sources and reading through them: while this may represent an already evident advantage, detractors of this view point to the fact that malevolent programmers (aiming for instance at intruding in public systems), have too full access to the source, and hence can discover bugs in it and try to exploit them, as an added security risk.[6] While it is non trivial to get a clear cut answer to this debate, what is evident is that while institutional clients can choose to submit some open source software to arbitrarily deep audit processes to try to detect flaws, with closed source software they will have to blindly trust the software company or programmer that released it (and that may have misaligned interests, causing in the worst cases a *voluntary* introduction of flaws[7]).

I will not here analyze if the benefits brought by a higher degree of investments and competition[8] outweight those disadvantages, but not because I consider it a minor issue: vice versa, it is probably the most interesting and natural question that comes out when talking about open source software, but unfortunately treating it would mean diving into a discussion which is made complicated by (apparently) mixed evidence and by the vast eterogeneity of contributions already made, which vary both in the status of contributors - most doubts may possibly be dissipated if research in this field was conduced mainly by "super partes" researchers, in a world which is instead more often than not object of studies realized on demand for commissioners already knowing *what answer* they want to receive - and in the kind of argumentations - since the issue can be approached from a "traditionally utilitaristic" point of view, but also from the point of view of sociology, philosophy of sciences and many others: this small study is simply not the right place for the discussion.

Given that I don't want to prove, or assume, the thesis that open source is "good" in any possible sense, in order to justify the interest in this study I will just mention another aspect: open source is *considered* good by many (most?) experts, and has *certainly* become

[6]Those positions are syntetized in the "*Security through obscurity*" approach, which sees in the secrecy of software mechanisms a guarantee of security that adds up to the quality of the mechanisms themselves.

[7]This is the case of closed source softwares which bundle hidden components - "*spywares*" are the most typical example.

[8]Not that the *libre* software world is free of competition: I'll illustrate more in depth the phenomenon in 3.2.2.

an economically very relevant phenomenon.

In this view - maintaining an "agnostic" point of view on the effects of open source, but assuming as evident that the open source movement *is* important today - it is interesting to provide an insight of how the whole movement was born, by presenting one of its most preminent and historically important figures: Richard M. Stallman.

1.3 Stallman and GNU

In the Seventies, Stallman was a researcher and software developer at the MIT in Boston. The Artificial Intelligence laboratories he worked in had a printer of which he had modified the *driver*[9] in order to alert users when their print jobs were completed, and every time there was a problem (such as a paper jam): the feature was quite necessary, since several users of the printer worked at different floors than the one where the printer operated. When a new printer was bought, in 1980, Stallman asked Xerox, the producer, to receive the sources of the new driver, in order to add this feature back in, but was refused access to them.

This experience was received by Stallman as an important signal of the fact that software, which was certainly to be gaining in the following years a central importance in the development of technology and society itself, was not only a "technical matter", but had very profound economic, social and possibly philosophic implications, in terms of the freedom that it gave or denied to users and actors of the information technology world. Three years later, he started the *GNU[10] project* to create a complete operative system that would guarantee to any user the *freedom* to *"share it with other people who like it"*. The quoted words are taken from the "*GNU manifesto*", which was released in 1983.[**?**]

1.3.1 Copyleft and the GPL

The GNU manifesto did not limit itself at emphasizing that the battle of Stallman was not just for a printer, but for a principle of *freedom*: it also put in evidence that (in his opinion, which is today shared by many participants and observers of the open source world) in order to get that freedom, simply releasing freely the sourcecode was not the best solution, as can be read:

"GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but no distributor will be allowed to restrict its further redistribution."

This is a condition that holds a central importance in my analisis, and its rationale and consequences will be devoted the necessary attention later. For the moment, I will just highlight

[9]The software needed for the computer to make it work.

[10]The name is a *recursive acronym* for "*GNU's Not Unix*" - Unix was at the time probably the most widespread type of operating system in professional environments, and GNU would represent a compatible but alternative one.

how the GNU movement coped with this condition, by formalizing it in the *GNU GPL* license.

"GPL" is an acronym for "General Public License": it is a license that was firstly written for some GNU softwares, but as the name suggests is intended for use by any developer, for any software. It is founded (as any software license is) on the legal basis of the traditional *copyright* law, but exploits its rules in a totally innovative sense, as is synthetized in the preamble of its text ([**?**]):
"The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users."

This is the core of the concept of *copyleft*: the GPL technically *restricts* the rights of recipients of the software, but it does so in order to guarantee the freedom of the software itself, since it restricts the "rights of restricting rights".[11]

To understand the importance of this aspect, it is interesting to make a brief mention of the *second* most widespread license for open source software (the first, as of today, is by large the GPL), which is the so called *MIT license*.[12] This license restricts in almost *no way* the "immediate" liberties of the user.[13] The differences in fortune of the two licenses and of the softwares which adopted them, as well as the philosophic differences between the views of the adopters and sustainers, reside in this apparently technical difference. The MIT license is defined a *permissive* license, while the GPL is a *viral* one.

1.3.2 Different concepts of freedom

The qualification "viral" contains in itself a first hint of why the concept of copyleft is relevant: it implicitly brings the issue of freedom at a *community* level. While a permissive license just takes the most immediate definition of freedom at the individidual level - do what you desire with the software - a viral license aims at *spreading* and *keeping alive* freedom.
A parallel can be drawn between this "addendum" to "bare liberty" and the theory of "positive freedom" (as opposite to *negative* freedom, considered as just the lack of restraints) as it is declined by Amartya Sen ([**?**]), with the difference that while the Indian economist correctly points out a list of "*ingredients*" (such as education and health care) which correspond to precise needs of citizens in order for them to gain this *real* freedom and which governments should make sure are granted, the GPL can, thanks to the possibility of applying desired conditions on redistribution of software, tackle the issue directly at a community level by just

[11]More precisely, the GPL establishes rules to which *redistributors* must abide when publishing some software, with or without modifications; no binding is set to what can be privately be done with it.

[12]Though according to the Free Software Foundation it should be called "X11 license", this is the name it is usually referred to as.

[13]Basically, the only condition that the license imposes is that the text of the license itself is always bundled in the software and in *"substantial portions"* of it ([**?**]).

relying on the underlying copyright law.

The GPL targets and configures a *sustainable* freedom, with increasing returns to scale, in the sense that every new software released under it represents at the same time a gain for the community by itself and a potential gain in the fact that derived software will have to be free as well. So far, we outlined the *motivations* of the writers and adopters of this license; now it is natural to wonder if this mechanism worked.

One of the most successful softwares released under the GPL is the *Linux kernel*, which, though initially developed independently from the GNU project by the Finnish student Linus Torvalds, ended up being its natural complement: the *GNU/Linux* systems are today used in a huge number of applications, from cellular phones to research devoted supercomputers, from desktop applications to web servers. The last mentioned is maybe the segment that best shows the increase of popularity of the system:

- *1969*: ARPANET, the first ancestor of Internet, is born in the USA, and soon after the first international network, with UK, is realized,
- *1976*: Queen Elisabeth II sends an email to the *Royal Signals and Radar Establishment*,
- *1989*: the first commercial Internet Service Providers are born,
- *1990*: at CERN, the first web server runs the first web site,
- *1991*: the first version of Linux is released, though it will take at least another year to have a really usable version,
- *1992*: version 0.12 of Linux is the first to be released covered by the GPL,
- *2008*: "forty percent of servers run Windows, 60 percent run Linux" - as recognized by Steve Ballmer, the CEO of Microsoft, the company producing Windows itself.[**?**]

Those figures clearly show that though Linux started somehow *late*, it had a very important growth.[14]

1.3.3 Different words for freedom?

Today, the GNU project indicates 4 different *freedoms* which a software should in their view respect ([**?**]):

0: *The freedom to run the program, for any purpose.*

1: *The freedom to study how the program works, and change it to make it do what you wish. Access to the source code is a precondition for this.*

[14] Just for reference, the total number of websites is over 200 millions ([**?**]) and the total number of servers is estimated as being higher than that - frequently visited sites can be hosted on thousands of web servers, and there are many servers not dedicated to websites hosting.

2: *The freedom to redistribute copies so you can help your neighbor.*

3: *The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.*

Though all those points can be seen from a purely technical point of view, it is evident that the authors tend to stress the *ethical* aspect of them. By the way, those "freedoms" are not guaranteed only by GPL covered software, but by a wider spectrum of software licences, among which figures the already mentioned MIT license: all software products respecting them are called by the GNU project *"free software"*. Unfortunately, the term "free" is at risk of ambiguity, since it also has the more reductive meaning "free of charge": instead "free software" (the software respecting those liberties) is not the same thing as "freeware" (the software that can be obtained and used at no cost).[15] This is the reason why the term "open source" - which originally simply designed the softwares of which the sourcecode was publicly disclosed, with no reference at all to the usage conditions covering them - started to be used as a synonim of (the Stallman definition of) "free software". As of today, the term "open source" is probably the most widespread, even considering that it is very atypical for a proprietary (under this category, we simply regroup the softwares which do not respect the four freedoms) software to be open source.[16]

The fact that the two terms may be considered as synonims doesn't mean that the choice of one over the others is necessariliy *neutral*:[17] the Free Software Foundation, derived from the GNU project, has made a battle about using the "free" word, which puts more importance on the philosophic aspect of the movement, whereas in the industry the term *open source*, which more specifically refers to the mechanisms regulating the development of the software, is often preferred (Linus Torvalds, the autor of Linux, is one of the most strong sustainers of the idea that GPL software is good because of the positive effects it has on the growth and the quality of a software, not for the "philosophic" consequences which - he and others say - should not be the ultimate motivations of a software developer).

More recently, the Free Software Foundation suggested the use of the neologism "libre" (from the latin "liber", which means "free" *in the sense of freedom*) to unambiguously refer to free software without the confusion with freeware: though this is probably a smart solution, the term has still not entered the common language of professionals and, due to the diffuse adversion to the view of software as a "political instrument", it may never do.

[15]Stallman and others often emphasize the difference between two meanings of the word "free" referring to them respectively as "free as in *freedom*" and "free as in *free beer*".

[16]It would represent a loss of intellectual property without the "synergy" benefits which free software aims at reaching.

[17]That may be the reason why even in languages in which the corresponding word for "free" represents no ambiguity, the two different terms are used: that's the case for instance for French ("logiciel libre" doesn't bring any risk of misunderstanding, but "source ouverte" is used nevertheless) and Italian ("software libero" is again not at risk of equivocation, but the English "Open Source" is often used *verbatim*).

1.4 Other free operating systems

One may think that the popularity of Linux as an operating system for servers comes basically only from the fact that it is *free* in the sense that a copy can be obtained *at no cost*, while getting a copy of its main competitor product - Windows Server - can cost from 400 US dollars up to 4000. This path of reasoning certainly makes sense, but to ascertain to what extent it explains the phenomenon, it is useful to make some references to other free operating system: the main free alternatives to Linux are the ones deriving from the *BSD system*. The Berkeley Software Distribution was, similarly to Linux, a derivative of Unix, but with some important differences:

- it was born in 1977, almost 15 years before Linux, and was free (*libre*) since 1989,
- since that date, it has always been covered by a MIT-style license (to such an extent that today the MIT and other permissive licenses are often called “BSD style” licences),
- it has originated from an important university, not just from the quasi-hobbistic work of a student and a group of other developers (initially) missing any instituitional organization and coordination.

Nevertheless, the importance and presence of BSD derivatives[18] is today absolutely not comparable to the diffusion of GNU/Linux operating systems.

That doesn’t mean that BSD software didn’t have some spectacular successes: the core of the modern OS X operating system (which is shipped with all Apple computers since 2002, and in a stripped down version also in any IPhone), is the so called *Darwin kernel*, which contains relevant portions of code derived from BSD systems; similarly, Windows, by far the most widespread computer operating system, contained some BSD software at least until very recent times.[19]
The unfortunate particularity of those “successes” however is that those extremely popular final products which incorporated libre software *are not free*, in any possible sense of the word: they are closed-source, and are not even free of charge.[20]

This is the best way to really taste the profound difference between copyleft and permissive licences: the former tend to bind subsequent users and distributors at *contributing back* to the “common good”, in the form of of additions and fixes which a company may end up doing *just for the sake of getting a better software for itself*, while the latter are certainly more interesting for “second hand” commercial software producers - which can use them without giving anything back - but they don’t incentivate any virtuous circle to help the software grow and improve *while remaining free*.

[18]The most famous ones are OpenBSD, NetBSD and FreeBSD

[19]An example is the “rsh” utility in Windows XP (released in 2001, and still by far the most widespread operating system in the world).

[20]The fact that Windows and OS X are sold bundled with copmuters more often than as separate components should not mislead the reader into thinking that they are *free* - their cost is, explicitly or implicitly, added up to get the final cost of the machine.

I already wrote that this difference is crucial for the topic of this paper, and the last words already suggested why: we can hence now move to the tractation of the psychological and microeconomic aspect of those issues, in particular by comparing them with some standard assumptions of the microeconomic field of research.

It would be misleading to conclude this section, however, without mentioning some softwares covered by permissive licences that did become extremely popular while remaining free: two examples are *Apache*[21] and *Openssh*[22], which both run on a huge percentage of servers (running Linux, and not only) in the world. In the end, almost any working Linux-based system contains several crucial components which are released under non-GPL licences. However, those are often *very specialized* components, while permissive licences seem to be less appropriate for something as big and complex, and hence *necessarily* requiring a huge collaboration effort, as an operating system.

2 The analysis

In this section I will finally compare the evidence coming from the libre software world with some mainstream microeconomics paradigms.

Some honesty prevents me from attributing to any "mainstream economist" the real belief that every man and woman on earth behaves as a *homo œconomicus*, since it is evident to anyone that this sociologic thesis is simply unsustainable if confronted with evidence (we have *convincing proofs* that Mother Theresa of Calcutta did exist). The standard assumption is simply that deviations from "rationality" are negligeable. And still to not attack a straw man, I will interpret this "negligeable" not on the most natural scale for an analisis of microeconomic behaviour - the individual - but on a level more typical of social choice theory: the society. That is because the thesis "*nobody* has *non negligeable* deviations from rationality" (where by "rationality" I will for practicality denote the interpretation of it in purely utilitaristic models) is again clearly undefendeable, while the thesis "the set of individuals *affected* by *relevant* deviations from rationality is *negligeable*" is more interesting - so it is on that that I will focus.

2.1 The perfect license for the Homo Œconomicus

It may be useful to now take a look at the decisions which any individual which wants to produce software will face, and their apparent relative convenience under the purely utilitaristic view.

Let us assume that she has already written a good software, and must now decide the terms under which to release it. She will have basically three options:

1. release the software as a proprietary product, without releasing the sources,

[21]The *web server*, the component responsible of making web pages available to visitors of some website.
[22]A program to securely connect to remote servers.

2. release the software as a proprietary product, but disclosing the sources,
3. release the software under a *libre* license (for the moment, we will not make a distinction between copyleft and more permissive licences).

What can the (expected) payoffs of each choice be?

In the first case, the software producer can extract some price from each copy of the software that she is able to sell. If the software is good - in particular, if there are some potential clients for which it has the most convenient features/price ratio - or if anyway she is able to put in place a successful marketing strategy to support it - she will extract a profit which is proportional to the number of copies sold, *as long as* her product isn't superseded by a competitor which gives more features and/or costs less.

In the second case, she will have the same expected payoff just described, *with the difference* that any competitor will be able to start from her ideas: though they (legally) will not be able to directly reuse the code, they will - to a higher or lower extent, depending on the law about software patents which holds in the country of operation - have the possibility of reusing the same ideas and solutions.
She will have the thoretical benefit that every one reading the sourcecode will be able to find problems and to propose a fix, but on the other hand she will have no reason to expect that many people will read the source (competitors presumably will, but won't help her by pointing at problems and fixes).

In the third case, she will *not* have the possibility to charge a price on several copies of the software - not that it's forbidden, but if she did so, the first buyer would be able to resell at a lower price, or even to freely share, the copy bought with other potential clients - but on the other hand she may expect that:

- her software gains a rapid adoption, since it is free to use and in particular to test,
- the projects gains the attention of other developers (this happens naturally for softwares that are on the edge of innovation and that have a consistent user base - I will analize later why),
- users and developers help her in finding bugs, providing fixes and adding features,
- being the leader of a popular software project, she can ask for donations from users, or be payed for ad hoc customizations by companies adopting it.

Once the second option is ruled out,[23] since it provides no advantage over the first, the decision problem is restricted to choosing between the first and the third. In a static framework, there doesn't seem to be any competition: in one case, there is the certainty of the fees that *each user* will pay, while in the other there is the *possibility* that users make donations, or that some company in need of particular modifications directly asks them to the original

[23]This is not just a comfortable assumption: the cases of open source non-*libre* softwares are *really* rare.

developer.

What is very peculiar to the software world, however, is that it is *very dynamic and non-linear*, in the sense that it is characterized by increasing returns (of quality, at least) to scale in time: if a software is recognized as being better than the competitors, it will gain momentum and automatically get even better, causing a feedback loop particularly effective in a market where conquering a relevant share of the potential user base can be a matter of few months.[24] By consequence, the advantage brought by *freeness* - namely, the "attention" of users and developers - can become more and more important over time.[25]

2.1.1 Resharing - a different development model

There's more: so far, I have imagined the situation of an individual/company that has *just* wrote the software, and is considering the possible redistribution solutions *ex post*. But the first *waste* of the commercial software industry that I had indicated in 1.2, namely the *duplication of efforts*, can have a very important *ex ante* effect in affecting the decision *before* writing the software. More precisely, if portions of some already existing *copylefted* software are useful for an application I want to develop, this may influence my decision of developing a free (copylefted, in turn) software in order to use them.[26]
Far from being a purely theoretical reasonment, this is also an important reason in determining the typical organization of free software in *libraries* - that is, portions of code that have no "final user" utility by themselves but which can be important components of applications - and *applications* themselves. So it is *very* typical that the possibility of starting from a good (and tested) library performing a core task represents a *substantial* advantage in development, and hence can be the discriminant causing the developer/company to *develop*, and release, a libre software (this last paragraph starts to put in evidence the fact that libre software is not just a *marketing* strategy: thanks to copyleft, it can become a - cost effective - *development* strategy).

2.1.2 Who plays first?

The reason why resharing was presented *after* other prospected advantages of *libre* software is a sort of "chicken and the egg" problem: that possibility is for me an incentive to develop free software *if and only if* there *already* exists some other *libre* (and copylefted) software, but in order for *libre* software to exist at first, there must have been someone who firstly chose to develop (and copyleft) it *without* the (direct) incentive of resharing: she may have

[24]This period is particularly short for *some types* of softwares, while for instance migrating core functions of some firms' workflow from a software to a new product can be a very complex and time-consuming process.

[25]The attention of potential employers can be another factor, studied for instance in [4].

[26]Technically, several non-copyleft *libre* licences are *compatible* with GPL, in the sense that for instance a software covered by a MIT license can incorporate GPL covered components. However, the final software won't be covered by a global MIT-license, and it is unlikely that the non-GPL parts will be independently reused.

hoped that resharing would stimulate contributions to her own software, but she certainly couldn't use previously existing code.

That's the reason why the space devoted to the figure of Richard M. Stallman in the introduction is not just a *folkloristic* concession of the paper; on the contrary, the "first mover disadvantage" is a phenomenon which reflects in a *very important* dependence of results on some particular kind of initial conditions: the fact that someone *indipendently* chooses to release copylefted software.

Doesn't this kind of reasoning have nothing to do with the "homo œconomicus" concept this section started from?! What has Stallman - who evidently and explicitly acted based on *ideals* (maybe utopy?) and not on convenience - to do with the reasonments about a purely egoistic agent?

My main motivation comes from the idea that an effective reply to a oversimplicistic model - such as the *homo œconomicus* - cannot be an oversimplicistic "reversed" model - such would be the assumption that all agents (including companies) are altruistic, or at least coordinated and "solidal", attracted by some form of reciprocity - but that there is a *mixed* environment. The reason why I'm confident that this *is* an innovative approach is that evidence suggests how an even *small* number of "idealistic" agents can have a *very strong* influence on a world of mainly *egoistic* agents.

But once we admit a mixed environment, the definition of "homo œconomicus" must be refined to distinguish it from a simple model of *stupidity*:

- a purely egoistic agent which sees altruistic agents will certainly internalize this information in its decisions,
- a purely egoistic agent which internalizes this information may reasonably assume that *other* (purely egoistic) *agents* are internalizing it as well,
- more in general, that same agent will know that all his collegues may want to reason to the same "depth" as him.[27]

There is no obvious reason why these effects should all operate in the direction of software freedom - that can also be observed by looking at the controversial effects of permissive licences, mentioned in 1.4 - but in the case of copyleft the *legal* binding seems to "align" them in the direction of making it more convenient to work on free software.

This is the reason why the private initiative of an (at the time) unknown *altruistic* computer programmer *is* very relevant for the actions of many firms today, even for the ones whose CEOs may share none of the goals of GNU project.

[27] A focused analysis of the concept of *depth of thinking* is present in the literature regarding the variations of Keines's *"beauty contest"* - see for instance [3], or [2] for an interesting cross-discilinary approach - but has rarely been taken into account by standard microeconomics and games theory studies.

2.1.3 Compulsory reciprocity?

By the way, the word "altruistic" may not be the most appropriate to describe Stallman, since in the end his initiative just helped the software world in becoming - or remaining - more similar to what responded to some *ideals* and *principles* that a from a purely utilitaristic point should be viewed simply as... his *tastes*!

In the same way, the word "reciprocity", which is gaining popularity in the literature about *civil economy*, may seem out of place when talking about the effects of the GPL, which *obliges* software producers to use it in order to "recycle" some existing GPL-ed sourcecode, and to gain the popularity and momentum that may help them in achieving the top.

This is a subtle issue, which, though almost irrelevant from a purely economic point of view, can have deep ripercussions in the sociologic and psychologic analysis of the free software phenomenon: my impression is that the bindings imposed by the GPL are more a *shield* protecting "genuine" reciprocity - in other words, it seems that the world of programmers *tends to spontaneously resemble*, at least in part, *to the scientific academic environment* it rose from - than a way to *force* it. The important mass of software "protected" by permissive licences - its still relevant importance in the software industry is at least a witness of its quality - is already an evidence that cannot just be neglected, but in the light of that evidence, it is very presumable that also in the *huge* mass of GPLed software there is an important component of *want of gratuity* for which the GPL represents simply an "assurance" against "selfish" counterparts.

In 2.1.2, I tried to differentiate - in the specific environment of software development and distribution - the concept of *egoism* from the one of *stupidity*: maybe a specular operation could be made on the *reciprocity* side, by distinguishing different levels of cooperation, based on their being symptoms of *ethical principles* ("tastes", from a reductionist point of view) or *capacity* of collective coordination:

- a purely altruistic behaviour, in the sense that it is only motivated by the want to help others,[28]
- a *coordination game* in which a group of individuals is able to risk something in order to develop *synergies*,
- there is certainly a set of people which takes part in the game ex post, still representing a gain for the system, but without taking any particular risk.

As already said, people from the first group are simply not considered by mainstream microeconomics, and this is not completely surprising, considered that assuming a world made

[28]An altruist is here not defined as such based on what he will get as a consequence of his actions, but just based on the fact that this "payoff" doesn't motivate them - for instance because the reward can come with a small probability, or after a long time, or just because it is not considered at all. Alternatively (in a purely utilitaristic setup), it could be said that an altruist is just an individual having a utility function increasing in *other agents*' bundles of commodities: this definition just doesn't change the substance of what I'm referring to.

only of "good citizens" *can obviously* be very misleading,[29] in particular for policy decisions, and the real challenge is in case understanding the important apport of a *limited* set of altruistic individuals in some *mixed* environments.

In the strategic interaction theory, the classical solution to a competitive coordination game is the *Nash Equilibrium*, which can be seen as a sort of "mirrored minmax strategy": *I want to avoid the worse possible solution*, but I think that the other player will want to avoid it too, and if the fact that he wants to cause himself no harm can represent for me a partial assurance, *internalizing it* in my reasonments can yield a better solution.

Can the current level investment of time and capitals in libre software be the result of a Nash equilibrium, at least for that share of population that is not purely altruistic but is "ready" for cooperation (basically because it trusts in the readiness of others)? The answer to this question is not clear-cut, since the Nash Equilibrium is fundamentally a *static* concept: it is a configuration which should be stable, but which yields no hint on its *attainability*. It is tipically assumed that a static equilibrium is attractive for players and hence may be the result of "mental convergence" of their strategies, but this is exactly where the coordination problem comes in: what I want to express is that

- even if *currently*, the libre software path could seem a viable alternative to profit oriented individuals and companies, it certainly has not been for years.
- by "viable alternative", I mean that libre software has a high ratio quality/costs, but not that this aspect alone justifies all the contributions given back to the community.

Given the detail that I dedicated to the mechanism of "copyleft", this last point may seem contradictory: aren't contributions *compulsory*?! The situation is not so simple: though when releasing softwares based on GPLed components, the license establishes that source-code must be released as well, there are a number of even big companies (Google is probably the most known example) that tend to release to the public even *internally used* software utilities and libraries, under free licences, without any legal binding obliging them to do so.

The choice of those companies can certainly be partly justified by the seek of *reputational capital* (recall that many libre projects, known for being the top in their field, can be powerful "marketing platforms" to put in evidence the quality of a programmer or team). But again, this doesn't solve the question from the "mainstream" point of view: why are libre softwares often able to catch the attention, in particular of experts, more than commercial ones?

Moreover, the contribution of firms, which has been observed even in the years in which the forces *leading* the free software "market" and development where mostly researchers from the academic sector or hobbists, may be the best evidence supporting the efficiency of

[29] I don't want to dive in any antropologic analysis about what share of the apparently widespread selfishness is a characteristic of the *Homo sapiens sapiens* and which instead comes from the conditioning of vicious circles in the modern society: suffices to say that for a well disposed citizen, trustful in the goodness of the human being, it is not uncommon to lock up the door when leaving home.

the development model, but from an (homo o)economic(us) point of view, the most puzzling aspect remains the contribution of individuals, (often working in companies producing proprietary software, and) devoting their free time with apparently no reward expected, getting sometimes as "payoff" some effective reputation which however could hardly be *monetized* in their professional life. The simple appeal represented by being able to use the newly written software can be a partial explanation, but it is not less puzzling, if we think that in the first days *economies of scale* were very limited by the lower number of participants.

2.2 The theoretic lesson

Summarizing, the lesson that evidence on libre software seems to give to microeconomy is double:

1. even admitting that people with "very significant" deviations from the "*homo œconomicus*" model are a small minority, their action can be fundamental not just in forming a "separate" environment - such as a "philantropic sector" which depends on the "standard economy" but has no effect on it - but in heavily influencing the directions that the main sector takes,

2. ideological and ethic preferences tend to favour the convergence in efficient, though at a first sight apparently *highly improbable*, solutions which may be seen *ex post* as Nash Equilibria, but *ex ante* simply expose too much uncertainty to be recognized as such - in particular because the number of players is unknown.[30]

3 More practical lessons?

So far, I tried to study the "microeconomic justifications" for the *libre* software macrophenomenon.

In the light of those, I will now try to sketch a couple of practical implications, that policy makers, in particular, should be taking into consideration.

3.1 A really relevant phenomenon?

First, a legitimate doubt may raise: does *libre* software really have such a relevance that governments should considered it as an important presence in the software industry?

The evidence shown about the servers market in 1.3.2 should apparently leave no doubts about it. Instead, the issue is very subtle, and has to do with the definition of "economy" we assume. As I reported, Linux ran on 60% (and rapidly growing) of active servers in 2008; nevertheless, it represents *only 13.6% of the market share* in monetary terms.[31]

The reason for this discrepancy is clear enough: Linux-based servers are usually cheaper to setup and run. Though paid options, provided by important companies such as Red Hat

[30] It may be worth noticing that the literature about games with two players is thousand of times more numerous than the literature about *games* with thousand of players.

[31] Data about fourth quarter of 2008 ([1]).

(with *Red Hat Enterprise Linux*) and Novell (with *SUSE Linux Enterprise*), and providing a bundle of software and support service, are probably still the most common option for business critical servers, unpaid alternatives, such as *Ubuntu* (which also has a company behind - Canonical Ltd. - providing however only an optional service support), are gaining momentum, significantly lowering the average cost of a running Linux server. Moreover, even paid alternatives often represent a significant gain over Windows Server, both because of the lower license costs and because they already provide the technical support which otherwise is usually bought from third parties.
Finally, Linux becomes particularly convenient as scale increases: for a company which needs many servers to perform a similar task, once the work of configuring a particular system for the purpose is done, it can be replicated at virtually no cost on any number of machines.

The phenomenon is not restricted to servers: in the desktop sector, where recent estimates position Linux at around 1% of the user base, the market share, in dollars, is certainly far smaller than that, even considering the support contracts that many companies sign. The situation is aggravated by the fact that the vast majority of personal computers sold on retail are shipped with an already installed version of a Windows operating system, and hence even those who choose to use Linux on those personal computers have in practice already contributed to Microsoft and more in general to the proprietary operating systems industry.

Similarly, the increasing number of devices (i.e. mobile phones) that are being shipped with a Linux-based operative system, usually chosen to replace proprietary operative systems used on similar older devices, doesn't allow analysts to conclude that Linux is in this way building up part of the gross domestic product of some nation: Linux doesn't represent a *new kind of product*, opening a new market, but a cheaper (and better) alternative to some already existing products.

In the end, reasoning in terms of GDP may be very confusing in order to assess the economic importance of libre software. Not only evaluating savings may be harder than evaluating gains: the technologies developed can also be fundamental for *other* sectors. An example is *paravirtualization*, a particularly cost-effective technology that is gaining a wide adoption in the web hosting industry. Another example is provided by softwares enabling advanced graphic effects in the ordinary desktop environment, which were first developed in Linux, with the first release in January 2006, and then imitated by Microsoft in Windows Vista, at the end of that same year.

3.2 Libre software and competition

3.2.1 A meritocratic industry

One evident peculiarity of the *libre* software industry is that *information* is widespread: understanding how a software performs a given tasks simply implies, for a programmer, getting the sourcecode, locating the portion of interest and reading through it. Moreover, it is considered a good development practice to provide documentation not only for users, but also targeting potential contributors. Hence, this is an industry with singularly low *entry barriers*: it is not uncommon that low budget projects started by single developers or small companies become the *de facto* standard after a couple of years, or even much less.

Not only entry barriers are low in terms of *information* needed to start working at top levels, but also in terms of penetration effort for a working product: even the biggest companies supporting and/or using Open Source software, such as HP, spend in marketing much less than Microsoft, who also has a long story of publicizing third-parties (Microsoft financed) comparative studies tipically aimed at convincing the reader of the insecurity of Open Source software solutions.[32] But most importantly, marketing of Open Source products is almost always directed at attracting potential migrations from proprietary software, while the resources dedicated to marketing targeting current customers of other libre platforms is virtually non-existant. This doesn't mean that competition doesn't exist: well on the contrary, due to the open nature of the market, a product which is not on the edge of innovation immediately risks to be obsoleted and forgotten.

Public availability of informations also lowers *geographic* entry barriers: it is often the case that even large, professional projects are developed by collaborators scattered all around the world, typically working from home, and the concept of high-tech hub of which the Silicon Valley is the most famous example is in many cases replaced by annual conventions of cooperators which vary in place from year to year and gather also many hobbists, in an environment which is occasion for taking part in productive discussions and seminars, but also for concluding mutual knowledge and friendship, and for participating in entertaining activities.
This decentralization is particularly important for some countries which have a good education system but miss an history of technologically advanced industries, those countries which would be by consequence usually excluded from the hi-tech sector, or would end up providing low-skilled labour.

Summarizing, the libre software world is very meritocratic: good ideas and skilled individuals have the possibility of showing off their potential with few obstacles.

3.2.2 Does competition kill competition?

After having underlined so boldly the strong competition that open source favours, there is one contradictory aspect on which it may be worth spending some words, before jumping to regulatory conclusions: the very important entry barrier that libre software imposes... to commercial software! This may seem a nonsense, and it certainly is *not* one of the typical arguments cited by Open Source critics and/or sustainers of the principles of competition. However, it *could* in line of principle be an interesting problem from the point of view of

[32]Showing sensibly different results from those obtained by more independent researches, such as some feasibility studies about migration to open source drawn up by public administrations around the world.

social welfare, so I'll analize it briefly here.

Consumers tend to naturally favour higher quality *or* lower cost solutions. It *may* be that skilled programmer or companies, which *would*, in a world without "zero-cost" software, find it profitable to develop a good application and launch it on the market, don't jump into the initiative because they know that their potential userbase is/would be eroded by those free (of cost) alternatives and hence expect it to be not convenient. The aggregate effect of such decisions would be a lower level of research, of investments. . . of competition.

I don't consider this argument as merely theoretical: there are good reasons to think that such a mental process has passed through the mind of many players of the software industry, and conditioned their actions, limiting their investments (in the end, Microsoft is spending in marketing huge resources that it may have devoted to research and development, had it not felt the need to fight the "danger" represented by the more cost-effective competitor). There are however more than one reason why I think this it is a bearable effect: firstly, because in the last decades the software industry is one of those in which the academic sector has been most active, and in which the shift from "pure" research to applications, from a purely theoretical idea to a successful implementation, can happen faster - hence, attention must be payed not to cut out this environment by limiting the circulation of informations; secondly, because *anyway* freeware exists, and would have the same negative effect without providing in return a good environment for research cooperation to develop in; thirdly, because *as of today* many of the best minds in the industry found indeed their ideal job in companies that (also) produce open source applications - rumors that today Google stole to Microsoft the title of "best place to work" are widespread and very credible; fourthly, the market for programmers, in particular highly skilled and creative ones, doesn't really seem to be saturated or saturating at the moment - human resources not devoted to new commercial initiatives will hardly be wasted; finally, the proprietary software industry tends to produce many *very similar* products, which compete on (price and) marginal features, but have to be written each almost from scratch, because there is no sharing of sourcecode: this is a waste usually not justified by the features added (hence not socially favourable), though it may be convenient for the firm performing it, if with that move it is able to steal shares of market to rivals.

Another argument could be that protection of ideas and algorithms is better provided with a patents system than with the bare closeness of source, which blocks ideas circulation, and not just their commercial applicability. . . but this is an argument that I want to avoid using at all costs, for reasons that will be evident later.

It is important to notice in this section I tried to analize the social effect of this risk of low competition assuming that *competition is good because it brings toward a better service for the users, and hence for the society*. This is, grossly speaking, the rationale for the antitrust laws in the *civil law* systems. In common law systems, instead, the rationale for antitrust laws is more that *the rights of (small) firms to operate in the same conditions of the leader ones must be granted*. From this point of view, the "entry barrier" represented by freeness can represent a bigger reason to worry, but final results shouldn't change.

3.2.3 Software patents

I have mentioned so far the relation between policy makers and the world of libre software, but have not specified *what* can the formers really do that may influence the latter: indeed, apart from administrations which find in software freedom an *intrinsic* value because of philosophical and cultural reasons, or citizens that do, hence pretending administrations to recognize it,[33] there has been traditionally no particular intervention that the *libre* software world has asked to politics[34] (indeed, also because of its status of "novelty", many of its participants prefer to show its efficiency through a macroeconomic *proof of concept*, born and raised in "natural" autarchy to protect some apparently natural individual rights, than to pretend it should be an institutional measure to grant them).

There is one dramatic exception that got some public attention particularly in the last years: the *software patents* issue. In many countries, patents have never been considered applicable to *softwares*: more precisely, while *some device performing some particular task and being driven by some software* can be patented, an algorithm, or data structure, or software mechanism, can't: in summary, if there is *no specific hardware*, it is seen as not distinguishable from a "mathematical idea" - and mathematical ideas are luckily not patentable. However, in the USA courts the patentability of software algorithms and mechanisms is well established at least since a decade.

This is a very unfortunate situation for quite general reasons, but the damage it does is heavier for the free software movement - and it applies to an important share of the industry, as can be recognized even though committing to "agnosticity" regarding the *goodness* of the movement itself. It should hence have *very convincing justifications*, in the sense that it should bring important benefits to other sides of the market. I will try to show this is false; by the way, many more authoritative voices, not necessarily having to do with libre software, agree on this conclusion.

But first: how do software patents harm libre software? I have discussed in 3.2.1 about the very low entry barriers of this industry. This ceases to be true with software patents, for several reasons that I'll now outline.

- Before getting a project started, it will be necessary to look at existing patents to

[33]The debate on the adoption of free software in the public administrations - and about migrations to Open Source software in general - is very interesting, but opposes to some fundamental principles - resharing of sourcecode is not just a phenomenon instrumental to software development, but can be seen as an occasion for spreading knowledge and favouring a sort of "new culture" - a bunch of technical details which would require by themselves a dedicated study - and indeed, many have already been made.

[34]As I already said, the academic sector is a relevant engine of development for the *libre* software phenomenon, and in many countries it is basically publicly financed, so *there is* an evident contribution. However, assuming that the directions of research are not determined by the government itself, but by some researchers or academic bodies - as happens in virtually every democratic country - this cannot be seen as a *direct* mechanism of *libre* software public governance.

ascertain none of then is being infringed. This already adds a significant overhead to the cost of the operation: basically, it is something that a single programmer, or a small firm without a legal department, can not cope with. The problem is not peculiar to the software industry, but it is here aggravated by the fact that most softwares are closed source, so it may be at first glance difficult to understand what mechanisms they are based on - and hence which related patents a competitor product would risk to infringe. There are several cases of software projects which were started and subsequently aborted, after that a lot of work had already be spent in developing them, because the discovery of infringed patents made it "not convenient" to continue,[35] and many more are the examples of projects which *never started* for the same reason.

- The problem is particularly important when a new software must be *interoperable* with an existing one: if the format in which informations are stored or transmitted is patent protected, it will be *virtually impossible* for the newcomer to enter the market in a profitable position, though its entrance may for instance target a niche of users needing features not provided by already existing products.

- The patents system is intrinsecally *too slow* and burocratic for a sector which develops *very fast*: the release of some softwares may have to be delayed, waiting for the patent to be successfullly filed, and in the meanwhile other competitors may vice versa work on similar technologies without knowing that a patent is being filed.[36]

- Since the (only sustainable) philosophy of patent offices has always been of not devoting too many resources at establishing patents validity *ex ante*, but instead leaving the final word to courts in case of appeal: there is a long history of "innovations" of at least debatable patentability being accepted. However, in the software field, possibly because the new technologies are mastered worse by patents offices employees, or because they favour more vague descriptions and claims, simply *ridicule* patents, because of the negligible innovative apport or because of the blatant existence of "prior art", are known to have been accepted.[37]

- In theory, that last point should not be a problem: if a patent is unjustly accepted, it means that in the case in which it will be debated in court, it will be declared invalid. Unfortunately, taking part in such a debatement, in particular against an important company with powerful legal departments, can cost so much that not only small players

[35] The *Tux2 filesystem* was already gaining some popularity when its development stopped, due to the discovery of patents covering it: they had been granted *after* the first implementation of those ideas in Tux2, but due to the difficulty of proving the existence of thois *prior art*, or simply fearing an expensive legal battle with the company owning them, *Network Appliance*, the project was in the end dropped.

[36] The lapse between the first filing of a patent request and the date from which the patent is (retroactively) valid (if later approved) may be not very large - around one month - but in many software innovations what is more important is the *tuning* and testing phase, which tipically allows to understand the breadth of an innovation, and can requre a long period during which the producer must keep it secret.

[37] An example that gained some celebrity is the US patent 7,415,666 that Microsoft obtained, in 2008, on a *"Method and system for navigating paginated content in page-based increments"*: it consists in exactly the keys "Page Up" and "Page Down", existing since the beginning of the 80's and part of any generic keyboard since that time.

will be unable to bear the risk of loosing, but also bigger ones may find it convenient to reach out-of-court settlements, even in cases in which they seem to be in the right.[38]

What those findings should suggest is that software patents are not only a problem because they intrinsically limit resharing of ideas and discoveries, but also because *in the particular case* of the software industry, they are a mechanism too rigid to avoid severe secondary effects and abuses.
The *abuses* topic is particularly relevant: Microsoft, which alternates declarations of support to open source with campaigns trying to put it in bad light, has publicly declared in 2007 that Linux was violating as many as 235 of its patents, but has never released any detail about the statement, which was hence seen by many observers as a pure act of psychologic terrorism.[**?**]

That said, it would be unfair to consider the motivations exposed so far in disfavour of software patents without confronting with the standard argument used by intellectual property protection advocates, namely the risk of *no incentives for research* (if there is no hope to extract profits from an innovation because of the free competition, no investments will be made). Still, this argument immediately appears *extremely* weak if confronted with the evidence coming from the software world, where the cases of research and investments made independently from the guarantee of an exclusive profit stream do not limit to the (already by itself very vast and technologically productive) *libre* software world.

There are two possible explanations for this, and both are probably true to some extent: firstly, in the world of software *expensive inventions* are not common - ideas are often "cheap",[39] while their *implementation* can be more costly. Secondly, software is the perfect product for exhibiting economies of scale, where the marginal cost of production of one "item" is virtually zero: for this reason, just *copying solutions* is not a rewarding strategy, since it implies reimplementing all of them without then having the possibility to undercut production costs.[40]

What is happening in the software world is that companies are filing thousands of patents, but do not get any royalties from the vast majority of them: instead, those portfolios are just exhibited and possibly used as bargaining chips in intercoorporate contracts and "peace treaties"; in a 1991 memo, Bill Gates himself recognizes "defense" as the main justification for accumulating software patents.[41]
The libre software world is trying to counteract in different ways:

[38]This is what many observers think have happened in the dispute between TomTom, worldwide producer of navigation devices, and Microsoft, accusing the former, which was at the time in a difficult economic situation, of having infringed several patents (regarding the "FAT filesystem" developed by Microsoft). Though the general opinion was that TomTom would have won the case, the two companies reached an out of court settlement, and TomTom stopped using the FAT system in its devices.

[39]This is probably an oversimplification; however, I'm *voluntarily* neglecting the theoretical programming results, which can be very important but are not covered by patent laws, being "abstract ideas".

[40]Paying a high entry price to take part in a symmetric game of Bertrand competition doesn't really seem a motivating prospective.

[41]As reported by ArsTechnica: http://arstechnica.com/business/news/2007/03/analysis-microsofts-software-patent-flip-flop.ars

- the birth of the "Open Invention Network", a company funded by IBM, NEC, Novell, Philips, Red Hat and Sony with the goal of collecting a portfolio of software patents sufficient to make Linux users and investors "feel safe",
- the third version of the GPL adds the clause that anyone releasing sourcecode covered by it is *automatically* renouncing to the possibility of enforcing patents used in it.

Though thanks to those countermeasures the risk that libre software incurs because of patents may now be lower, the vast majority of its users and companies is strongly against the concept itself of software patents, and in particular has been very active in fighting their adoption in Europe: a directive aimed at regulating software patents, proposed in 2002 by the European Commission and sustained by the lobbying activity of some important hi-tech corporations (including Microsoft, HP and IBM) was, after a long and lively public debate, rejected in 2005 by the European Parliament with 648 votes against 14.

4 Conclusions

The particular structure of the libre software economy, which has given so far results that can be surprising if judged with canonical microeconomic models, provides not only sufficient evidence for innovative insights of choice theory, but also suggests that policy makers must pay attention to this new model (intrinsecally characterized by resharing of ideas, informations and software), considering that the low material costs and entry barriers can allow even private citizens to voluntarily help a movement that in the end has very benign (and appreciated) consequences even for important sectors of the "traditional" economy, but at the same time can be very fragile against legal measures limiting those freedoms.[42]

References

[1] Worldwide Server Market Contracts Sharply in Fourth Quarter as Market Revenues Decline to $53.3 Billion in 2008, According to IDC. *International Data Corporation*, February 2010. `http://www.idc.com/getdoc.jsp?containerId=prUS21703309`.

[2] G. Coricelli and R. Nagel. Neural correlates of depth of strategic reasoning in medial prefrontal cortex. *Proceedings of the National Academy of Sciences*, 106(23):9163, 2009.

[3] R. Nagel. Unraveling in guessing games: An experimental study. *The American Economic Review*, 85(5):1313–1326, 1995.

[4] J. Prüfer. Why do developers and firms contribute to the production of Open Source Software, 2004.

[5] Eric Von Hippel. Open Source Projects as Horizontal Innovation Networks - By and For Users. *SSRN eLibrary*, 2002.

[42] I have limited my analysis to the topic of patents; some more ample discussion on policy implications can be found in [5].