Open Source as a Signalling Device - An Economic Analysis*

Samuel Lee Nina Moisa Marco Weiss†

This version: March, 21st 2003

Abstract

Open source projects produce goods or standards that do not allow for the appropriation of private returns by those who contribute to these projects. In this paper we analyze why programmers will nevertheless invest their time and effort to code open source software. We argue that the particular way in which open source projects are managed and especially how contributions are attributed to individual agents, allows the best programmers to create a signal that more mediocre programmers cannot achieve. Through setting themselves apart they can turn this signal into monetary rewards that correspond to their superior capabilities. With this incentive they will forgo the immediate rewards they could earn in software companies producing proprietary software by restricting the access to the source code of their product. Whenever institutional arrangements are in place that enable the acquisition of such a signal and the subsequent substitution into monetary rewards, the contribution to open source projects and the resulting public good is a feasible outcome that can be explained by standard economic theory.

JEL Classification: *D82, L14, L86, O31*
Keywords: *open source software, signalling, career concerns, economics of organization*

*We are grateful for valuable discussions with our colleagues at the Department of Finance of the Goethe-University, Frankfurt/Main. We especially thank Christian Laux and Marcel Tyrell for their thorough review.

†Address for correspondence: Wilhelm Merton-Chair for International Banking and Finance; Johann Wolfgang Goethe-University, Frankfurt/Main; http://www.finance.uni-frankfurt.de; Phone: ++49(0)69 798-28268; M@il: maweiss@wiwi.uni-frankfurt.de

Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute for free?[a]

OSS poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.[b]

[a]Open Letter to Hobbyists by Bill Gates, Feb. 3^{rd} 1976
[b]Halloween Memorandum I by Microsoft, Aug. 11^{th} 1998

1 Introduction

All over the world, computers run on a variety of programs and communicate over networks linked by protocols that are generated in the open source domain. Ever more electronic devices like mobile phones rely on open source products and thrive on the standards that are established by a community that not only includes single hackers working in their leisure time but also the giants of the commercial world like IBM and Motorola. Linux looms as its flagship among other prominent success stories such as sendmail or Apache. Linux develops operating software for almost every electronic device and is especially successful in the market for server software: In 2001 it had a market share of 25% compared to Microsoft Windows with a share of 49%.[1] The Linux program was originally developed by Linus Thorvalds in 1991, then a student at Helsinki University. Its kernel was based on Unix which came in half a dozen proprietary versions at that time. Instead of generating proprietary software, Thorvalds made his program code accessible for other programmers and invited them to contribute on a voluntary basis.[2]
That leads us to the most striking characteristic of open source software: free access to the product and to its source code. This characteristic is legally embodied in what is called the 'General Public License' (GPL)[3] - sometimes also referred to as 'copyleft'. Ensuring that the source code of a software program remains open, it states that everybody may run, copy, modify and distribute the program under the terms of the original license. Though the sale of modifications is not prohibited, the public shall be free to access and use the original source code. Therefore the prices of open source products beat those of their proprietary counterparts and whoever wishes to do so can download the latest version of the product for free.[4]
Since no one can exercise ownership of the original product in the sense of excluding others from the right to use it under the GPL, revenues from transferring or licensing this right prove elusive. In turn, this implies that the community of involved code contributors and debuggers can not claim any monetary compensation for their time and effort! Even more astonishing is the fact that yet both quantity and quality of the contributions have such an extent that these products seriously compete with those of software giants like Microsoft.[5] Not surprisingly the phenomenon of open source has generated a growing interest in the academic community.

[1]See Deutsche Bank Research (2002, p.7).
[2]For a more extensive description of the history of Linux, see http://www.linux.org/info.
[3]For the GNU General Public License see http://www.opensource.org/licenses/gpl-license.html. A good discussion can also be found in Kaisla (2001).
[4]Additional services such as manuals or related service packages are sold by different distributors who compete mainly on service, support and training.
[5]See for increasing evidence Economist (2000), Economist (2001) and Economist (2002).

At first glance the existence and the success of open source systems (OSS) is equally puzzling for economists and surprising for their closed source system (CSS) competitors. They particularly marvel at the eagerness of obviously highly skilled agents to work for free: Why would rational programmers grant their time and skills to a non-profit OSS-project instead of taking up a career in a CSS-firm like Microsoft where they would get paid for their work?
To answer this question, the public opinion often alludes to an ideological rebellion against commercialism and the reign of near-monopolists such as Microsoft. In the same manner, a considerable strand of research has taken recourse to psychological motives such as altruism and dogma. The main idea behind these propositions is that there exists some intrinsic motivation, some sort of emotional satisfaction harvested from unselfish behavior.[6] But there are also less idealistic theories, those which refer to external rewards. Lakhani and von Hippel (2000) emphasize the advantages of user-to-user based feedback systems and the intangible utility user-developers extract from combining both activities. OSS programmers profit both by learning on the production side and by obtaining a better product on the consumption side. The more people join this community, the higher are the individual benefits. Rapid feedback by the very best in each specific field or project allows for very rapid advancement along the learning curve by individual programmers.[7]
Another group of ideas resorts to the value of peer recognition among software programmers. This line of reasoning argues that programmers behave as to be appreciated by their fellows and for this purpose like to show off their abilities. Even hobbyists, if seriously devoted to their pastime, are habitually embedded in a community where performance is compared and acknowledged and reputations for expertise can be earned among the like-minded.
Our approach is similar in spirit to the last one, but different insofar as it adds material compensations. We join works such as Lerner and Tirole (2002), Johnson (2002) and Mustonen (2003) who claim that this kind of recognition can be transferred to the outside and moreover be translated into monetary rewards. In this view, contributions to the program are not so much unselfish donations or the pursuit of vain self-gratification, but rather future-oriented investments which are based on career concerns. In the words of established economic theory, ventures in the world of open-source are undertaken for the sake of a credible job market signal as described by Spence (1973). How does this work?

A close inspection reveals that the OSS is organized such that every significant contribution can be traced back to the original author. In one of the biggest OSS-projects, the Linux kernel, there exists a public changelog file which lists all those programmers who have contributed to the official source and their specific inputs.[8] Naturally, not everyone makes it onto the list. Each proposal to modify the code undergoes a peer review process and only those modifications sanctioned by the referees make their creators legitimate authors. The authors' names and contributions are recorded in the changelog file which is an honoring and a sign of expertise among the programmers.[9] This is where the theory of peer recognition stops, but not the one on career concerns. For, if peer recognition theory presumes that infor-

[6] See Hars and Ou (2002) for some data on motivations to work for OSS-projects.

[7] The systemic features of the OSS are described in Raymond (2000a) where he likens the processes of the OSS to a bazaar in contrast to the 'cathedrals' that are crafted by proprietary software firms with their products.

[8] See e.g. Moon and Sproull (2000) and Raymond (2000b).

[9] In the case of Linux, the changelog file portrays a pyramid-like hierarchy among the contributors. See http://www.kernel.org/pub/linux/kernel/v2.4/ChangeLog-2.4.20. The changelog file classifies different programmer types just as there are different kinds of contributions ranging from documentation over debugging to more complex developing tasks. We interpret this as evidence that any kind of programmer has the potential to signal his level of skill, but none which is higher.

mation is revealed inside the community, why not admit that the same signal could also reduce the information deficit of people from the outside? Prerequisite for this is the existence of some suitable and convenient mechanism to transfer the signal beyond the domain and thus educate outsiders about the superior ability associated with it.
In general, it seems reasonable to assume that the knowledge about skills is less uneven among programmers than between them and outsiders. A spot in the credits thus serves as a valuable signal on a job market characterized by asymmetric information. Imagine a personnel manager faced with two candidates A and B who claim to deserve excellent pay. Suppose both certify basic programming skills, yet B in addition proves that he has contributed important modules to the Linux code. Who of them is more likely to get the higher salary? If the Linux graduate is indeed rewarded a premium, it pays off for him to have spent the effort on OSS programming.[10] The *ex ante* expected value of the deferred pay-off makes striving for the signal worthwhile since the unrestricted access to the Linux kernel code and its changelog file allows for the right interpretation and honoring even by outsiders *ex post.*

The signal can cross the borders of the OSS community, precisely because the source code is open. To function well, however, it must be sufficiently visible and credible. Otherwise, potential employers will either not receive the signal or will not (fully) rely on it. Consider what a signal means to them: its quality corresponds with their willingness to pay its bearer a wage premium. In other words, the value of the deferred pay-off depends on the properties of the emitted signal. *Visibility* is achieved by a broad distribution of the product and a well-known brand. We argue that the effect of the number of OSS programmers on this criterion is significant: First, the number of developers raises the number of users directly. Secondly, a higher number of developers augments the quality of the product, the acceptance of which will therefore rise among the less sophisticated users. A possible third effect, an inter-linkage of the first two, would be conceivable, if user-developers were mainly avantgarde-users and industry trendsetters. The *credibility* of a signal grows with the superiority of the refereeing process and the total number of proposed modifications. To understand this, recall the information that a signal carries: "This programmer has met the standards set by the referees and has prevailed among x modification proposals to earn this spot in the changelog file." Though the level of the standards and the number of competitors are not directly observable, they are usually implied by the quality of the product[11], which in turn affects the distribution and the visibility of the product. We propose that this visibility and therefore the credibility of the signal rise with the number of programmers.[12]
In the previous example, a good signal would therefore evoke the following conclusions in our personnel manager: "Candidate B has drafted vital modules for the Linux program. Linux is a prominent brand. (This is the reason why I would know of it.) Since the product is widely known and used (especially by software experts), it must be good. Apparently, Linux has high quality requirements and a lot of good programmers involved. (Otherwise, the product would not be so successful.) *So, if this guy has made it into the changelog file, he must be very skilled.* We should offer him an adequate salary." On the whole, visibility and credibility, which rise with the number of programmers in the OSS, positively affect the deferred pay-offs and

[10] Lerner and Tirole (2002) speak of a deferred pay-off.

[11] Consider, analogously, the influence of the competence of scouts and the number of aspiring athletes on the level of the game e.g. in pro-basketball.

[12] This is obvious for the quantity of proposed modifications, but not so for the quality of the referees. For intuition, consider how the competence of scouts will probably rise with the overall popularity of basketball.

thus make the signal more valuable.

By the preceding account, good programmers should altogether work and acquire a signal at the OSS. If the number of programmers had only positive effects, it would be beneficial to have as many OSS colleagues as possible. But additional programmers can also have negative effects on their co-workers acquisition of a signal: The story we have told is not yet complete, and the point we missed boils down to the fact that aspirants must compete for (the limited number of) signals. To make this case, let us first revisit the requirement that the product be widely visible. Let us add the observation that the product needs to be complex such that contributions to it have the character of innovations. Otherwise there would be a limit to the skill levels that can be demonstrated by delivering a share of the program, for simple products require relatively simple craft. Naturally, these properties put constraints on the nature of products suited for OSS fabrication: they characteristically need to be sophisticated products and of wide use. Furthermore, improvements in thus suitable products will be restricted by natural boundaries to innovation. E.g. the number of problems one encounters in the making of operating system software is finite in the short-term.[13] Thus, given these boundaries in the set of products and innovation, the overall stack of problems inventive programmers can pile into via OSS is limited. As the number of programmers in OSS increases, more and more will start to cross paths and embark on the same technical questions. Since only one solution will make its way into the official code, they will begin to compete for spots in the changelog file. The winner-takes-it-all principle driven by the peer review process will make it increasingly difficult for any one of them to stand out. Thus, in sum, each programmer in the OSS exerts a positive and a negative externality on his peers. His participation increases the value of the signal, but at the same time decreases the probability of obtaining one! We will therefore model the *expected value of the deferred payoff* such that it first rises and then falls in the number of OSS programmers.[14]

Evidently, the hitherto sketched mechanism would only induce programmers to an OSS career, if the discounted expected deferred pay-off were higher than the expected wage in a CSS-firm. But why would we presume that CSS-firms do not differentiate wages after the first year of employment or so, for, if they did, little would remain of the OSS' attractiveness? One important reason is that, in the absence of an OSS, there is no incentive to do so. Why should the CSS-firm concede to its employees a signal, which enables them to market themselves to outside firms? Once they have it, they could threaten to leave the firm in order to renegotiate their salary. Should there be a supply shortage on the labor market, the firm would have to give in, lest it would lose its most able programmers. Thus, the CSS-firms' best strategy would be not to grant such a signal in the first place. In the presence of an OSS, a CSS-firm can react in two ways: On the one hand, if the provoked drain of programmers were negligible, so might the incentive to change the wage structure. On the other hand, if the threat were considerable, more effort might be undertaken to differentiate wages. There is nowadays indicative evidence of CSS-firms' attempts to emulate OSS-like production and reward structures.[15] It is not clear, however, where this will take them.

[13]Even if additional programmers were to discover new problems, the marginal rate of growth in the number of problems would certainly be declining in the number of programmers.

[14]See Mustonen (2003) for a different model of the expected payoff function.

[15]Microsoft introduced a philosophy called 'Shared Source' under which terms it grants different users some insights into the source code of its Windows operating system. HewlettPackard introduced 'Corporate Source' to reap some benefits of the processes at work in the OSS.

In essence, we argue that a signalling mechanism is at work for those who contribute to an open-source product that is distributed for free. In contrast to that, the restriction of an undisclosed source-code in a traditional software firm necessarily limits the transparency concerning individual contributions, resulting in a more levelled wage for programmers of different productivity. As a consequence, under certain circumstances some high potentials might prefer to invest their resources in an OSS-project. We aim to show the conditions for such a result in a model in this article.

The paper proceeds as follows: Section 2 sets up the basic model. An equilibrium analysis is undertaken in section 3 before in section 4 the equilibria are further analyzed in a comparative statics way for the effect that changes in important parameters can have. Implications of our model are briefly discussed in section 5, whereas section 6 offers some possible extensions to our model.

2 The model

2.1 The programmers

We assume

1. a total population of n programmers consisting of n_A programmers of type A and n_B programmers of type B. That is,

$$n = n_A + n_B.$$

2. that type A generates an output of q_A, type B produces q_B, while effort levels are constant and costs of effort are equal for both types, so that there is no moral hazard. By assumption,

$$q_A > q_B.$$

3. the information regarding his type is private knowledge for each agent. Thus, the labor market is subject to asymmetric information. It is also characterized by excess demand, i.e. in principle, its demand-side institutions can and would absorb all programmers.

4. the programmers are risk-neutral and completely patient, i.e. they have a discount factor of one.

2.2 The institutions

In our model we distinguish two types of institutional arrangements that software production can take and between whom all programmers have to make career choice: an open-source system (OSS) and a closed-source system (CSS). Both possible systems are stylized as having an identical production function and a different remuneration method.

2.2.1 The production function

We assume

1. the two institutional arrangements possess identical production functions which are specified as the sum of the individual productivities of all programmers working for a representative firm or project in the respective system. This

additive production technology employs human capital as the sole production factor and is a function which is homogenous of degree one. Therefore, the marginal return of one additional programmer always equates his individual productivity q_i (q_A or q_B). Formally, this production function can be stated as follows

$$Q^j = \sum_{i=1}^{n^j} q_i = n_A^j q_A + n_B^j q_B \qquad for \quad j = OSS, CSS \tag{2.1}$$

where n_A^j and n_B^j represent the number of type A and B programmers in the respective system. If we denote the fraction of n_A working for the CSS with α (where $0 \leq \alpha \leq 1$) and the fraction of n_B working for the CSS with β (where $0 \leq \beta \leq 1$), then the production functions for the respective system can be rewritten as

$$Q^{CSS} = \alpha n_A q_A + \beta n_B q_B$$
$$Q^{OSS} = (1-\alpha) n_A q_A + (1-\beta) n_B q_B.$$

2. Q^j denotes the aggregate output in a representative firm in the CSS or a particular project in the OSS. It can be interpreted as the quality of the software product and its thereof derived degree of distribution among users. In the case of the CSS, we equate Q^{CSS} to the revenues earned by the sale of its products. In the case of the OSS, where no proceeds are reapt, Q^{OSS} is a proxy for its *visibility* and *credibility*.

2.2.2 The wage function (CSS)

While having the same production function, the two institutions are substantially different in terms of remuneration. The CSS uses the proceeds to pay wages to the programmers, whereas the OSS lacks those proceeds. At the OSS, programmers work for free and receive - if anything at all - a signal which is rewarded in monetary terms only in later periods.

We assume

1. a CSS-firm cannot (or does not want to) distinguish between the two types of programmers as it has no access to a sufficiently effective or inexpensive screening technology.

2. outstanding performance during a CSS career does not lead to higher wages. The closed-source technology implies certain limits to the transparency on different programmers' contributions so that the individual output is not publicly observable or verifiable. Though a type A programmer can demonstrate his programming skills within a particular company, the CSS-firm initially has no incentive to grant him a signal which could be used to seek a better paid job somewhere else. With no such signal at hand, any outside company would at best offer him some pooling wage, leaving him no better off. Therefore he is not in the position to threaten termination of his contract. Knowing this, the firm has no incentive to increase his wage.

3. the CSS chooses the wage level as to realize zero profits.

Our assumptions imply that the earnings equal to Q^{CSS} are shared evenly among all CSS programmers. Everyone gets the same wage

$$w(\alpha, \beta) = \frac{Q^{CSS}}{n^{CSS}} = \frac{\alpha n_A q_A + \beta n_B q_B}{\alpha n_A + \beta n_B} \tag{2.2}$$

with n^{CSS} denoting the total number of programmers working for the representative CSS-firm. The sum of the wages always equals the total output Q^{CSS} for any given level of α or β. The following figure illustrates the relationship between the fraction of type A and type B which join the CSS and the wage they get consequently.

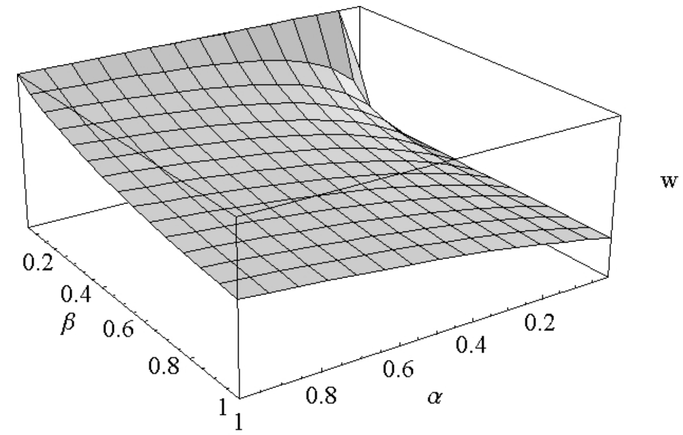

Figure 1: The wage function

2.2.3 The deferred pay-off function (OSS)

Programmers at the OSS are not paid wages, but can earn signals which indicate their productivity. The value of such a signal is the discounted value of the resulting deferred pay-off. Since it cannot be earned with certainty, programmers must calculate their benefits on the basis of expected values: namely the value of the associated deferred pay-off weighted with the probability of obtaining a signal.[16] We assume that

1. the *value of the deferred pay-off* associated with a signal is positively related to the signal's visibility and credibility, which Q^{OSS} is a proxy for. According to equation 2.1, Q^{OSS} rises with n^{OSS}. The logic is as follows: The more programmers cooperate, the better will the joint product be. The better the product is, the larger will its distribution be. And the more prominent the product is, the more visible and credible will the signal be.

2. the *probability of obtaining a signal* is negatively correlated with the degree of competition among programmers. We assume competition to be a function also of n^{OSS}. As more programmers join, it gets increasingly difficult to earn a place in the changelog file. The likelihood that a software problem draws more than one solver rises, while chances to be in the ranks of those whose solutions have worked their way into the official source code drop.[17]

[16]Note that the assumption of risk-neutrality and complete patience on the part of our agents allows us to treat discounted expected deferred pay-off as if it were neither uncertain nor deferred. For reasons of brevity, we will often omit the attributes 'expected' and 'discounted', while keeping the 'deferred' to indicate that programmers work for free during their time at an OSS-project. Dropping the assumption of risk-neutrality would alter our results to the extent that *ceteris paribus* only the less risk-averse type A programmers would ponder and possibly embark upon an OSS career. Leaving aside the assumption of complete patience would decrease the present value of the expected signal, thus making the OSS career less attractive.

[17]See Appendix A.1 for an exemplary formal treatment of the competition effect.

3. combining the two effects described in (1) and (2) results in the following shape of the deferred pay-off function: its value first rises and then falls over n^{OSS}. Gradually, the competition effect offsets and later outgrows the visibility effect. At sufficiently high levels of n^{OSS}, the value will approach zero. When approximating the function, we will assume a level k of n^{OSS} at which the value actually is zero. k can then be understood as a proxy for the innovative potential of the industry.[18]

4. the value of the deferred pay-off is also dependant on exogenous mechanisms that allow for the transfer of information and the willingness of outside commercial firms to pay a premium for the signal. Risk-averse principals faced with asymmetric information on the labor market are prepared to pay a premium for the revelation of the type.[19] Furthermore, we assume that such a premium will potentially only be paid for type A whose productivity is higher and that the maximum pay-off obtainable for type B is thus bounded by his productivity. We subsume these elements into the non-negative parameter v_i, whereby $v_B \leq q_B$.

5. the deferred pay-off functions are separate ones for type A and type B, because the changelog file effectively classifies the programmers' productivity. That is, the peer review process of the OSS prevents type B from imitating type A. Therefore, for a given programmer, competition is dependent only on the number of rival programmers from the same type.[20]

Assumptions (1)-(4) imply that there exists a unique maximum at which the value of the deferred pay-off equals v_i. On the basis of (5) we specify separate, independent functions for type A and B. We approximate them by using quadratic functions of the following form:[21]

$$r_A(\alpha) = -\frac{4n_A^2 v_A}{k_A^2}[\alpha - (1 - \frac{k_A}{2n_A})]^2 + v_A \tag{2.3}$$

$$r_B(\beta) = -\frac{4n_B^2 v_B}{k_B^2}[\beta - (1 - \frac{k_B}{2n_B})]^2 + v_B \tag{2.4}$$

In 2.3 and 2.4, α and β are the independent variables. The parameter k_i denotes the absolute number of type i programmers at the OSS for which the relevant functions assume the value of zero due to excessive competition. E.g. the expected deferred pay-off value for type A equals zero if $n_A^{OSS} = n_A(1-\alpha) = k_A$ (equivalent to $\alpha = 1 - \frac{k_A}{n_A}$). Obviously, for $\alpha = 1$ function 2.3 is also zero. The same analysis applies to k_B and β.

The parameter v_i is related to the individual productivity q_i of the programmers and furthermore captures the existence and quality of the surrounding markets and the willingness of commercial firms to honor the acquired signal. v_i determines the maximum value of the achievable deferred pay-off for the two types, whereas $(1 - \frac{k_i}{2n_i})$ are the α and β values for which the functions reach this maximum. Figure 2 illustrates the functions 2.3 and 2.4.

[18]The logic behind this is as follows: The innovative potential can be interpreted as the number of problems to be solved by innovations. The smaller this number, the more likely will programmers cross paths and competition will be higher. Thus, if the innovative potential is low, the competition effect sets in faster and will eliminate the visibility effect earlier. Hence, k will be low. Notice that the dimension of k is equal to that of the absolute number of programmers n. See also Appendix A.1 and the impact of the parameter m therein.

[19]The booming headhunting business shows that this is not an unrealistic assumption.

[20]Although the competition effects for the two types are independent (and therefore imply separate functions), the visibility effect is not. Thus, the deferred pay-off functions would be separate, but not wholly independent. We will, however, model independent functions for reasons of simplicity.

[21]See Appendix A.2 for the derivation of the function.

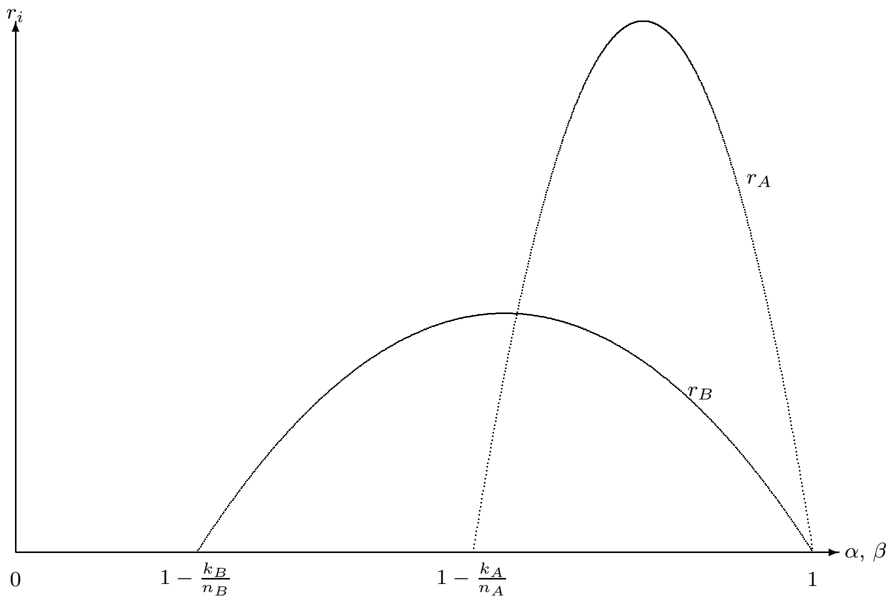

Figure 2: The deferred pay-off function

So far we have outlined the basic framework of our model. Its elements are the two types of programmers and the two institutional settings with identical production but different remuneration functions, one paying wage and the other yielding an expected deferred pay-off. Together, these elements sketch the decision problem which lies at the heart of our model. In the following section, we will formally analyze this decision problem for each type.

3 Equilibrium analysis

3.1 Conditions for the existence and stability of equilibria

We will not undertake an elaborate study of all possible coordination games between programmers in this paper and leave all the dynamic aspects that these provoke to follow-up works. In this paper, we assume that α and β are known with certainty. We proceed from the assumption that the effects of individual programmers on the given α or β are infinitesimal. A single agent therefore presumes that his decision will not affect the overall distribution and thus act as a price-taker with regard to w or r. An assumption that would lead to the same outcome is myopia by the agents such that they have no information about either of the global population parameters n_A, n_B, α or β. They neither know the complete shape of the remuneration functions nor their position on it, but observe only the locally given w and r. Both sets of assumptions sensibly rule out strategic considerations and result in a static optimization problem where agents act solely upon the observed values of w and r. We define a static equilibrium as follows:

Definition 1 *A static equilibrium is a situation that is characterized by no inherent tendency for change. Whenever small deviations in the variables occur, equilibrium is restored.*

Equilibrium is the aggregate outcome of individual decisions of type A and type B agents. To analyze the decision of the individual programmers we use the following difference functions:

$$\begin{aligned}\Delta_A(\alpha) &= w(\alpha,\beta) - r_A(\alpha) \\ \Delta_B(\beta) &= w(\alpha,\beta) - r_B(\beta)\end{aligned} \tag{3.1}$$

They represent the rationale of a single programmer of the respective type choosing between an OSS and a CSS career taking the values of α and β as given. A programmer will opt for the CSS, if the value of the difference function is positive, thus, whenever $w > r$. Conversely, the OSS will be preferred, if the value of the difference function is negative, i.e. if $w < r$. Programmers are indifferent between the two career paths whenever the value of the difference function is zero.

Even though the influence of a single agent is just minuscule, the overall distribution de facto results from the sum of all individual choices. For further analysis, we define a gravitation field as follows:

Definition 2 *A positive (negative) Δ_A establishes a positive (negative) gravitation field and α will tend to increase (decrease). The same holds true for β and Δ_B.*

Interior solutions For an interior solution to be an equilibrium two conditions must be fulfilled: Naturally, the necessary condition is the existence of an interior solution. For this, the programmers must be indifferent between the two career paths for a certain value of α^* and β^*. This implies that the difference function must have a value of zero, i.e. that the following conditions are met:

$$\begin{aligned}w(\alpha,\beta) &= r_A(\alpha) \\ w(\alpha,\beta) &= r_B(\beta)\end{aligned} \tag{3.2}$$

In addition to that, there must be a tendency to restore α^* or β^* in case of small deviations, which requires the following, sufficient condition, to be met: There must be a positive gravitation field to the left and a negative gravitation field to the right of α^* or β^*. In this case 3.2 satisfies our definition of an equilibrium.

Corner solutions In addition to the interior solutions, the right-hand corner solutions $\alpha = 1$ and $\beta = 1$ as well as the left-hand corner solutions $\alpha = 0$ and $\beta = 0$ represent potential equilibria. The necessary and sufficient condition for the right-hand corner solutions to be equilibria is that there is a positive gravitation field to its left. Accordingly, the necessary and sufficient condition for the left-hand corner solutions to be equilibria is that there is a negative gravitation field to its right.

To summarize, we illustrate the point graphically with Figure 3 which shows the difference function for type A.

3.2 The decision of type B

Lemma 1 *All type B programmers join the CSS.*

Proof: We insert equations 2.2 and 2.4 into 3.1 and get

$$\Delta_B = \frac{\alpha n_A(q_A - v_B) + \beta n_B(q_B - v_B)}{\alpha n_A + \beta n_B} + \frac{4n_B^2 v_B}{k_B^2}[\beta - (1 - \frac{k_B}{2n_B})]^2 \geq 0 \tag{3.3}$$

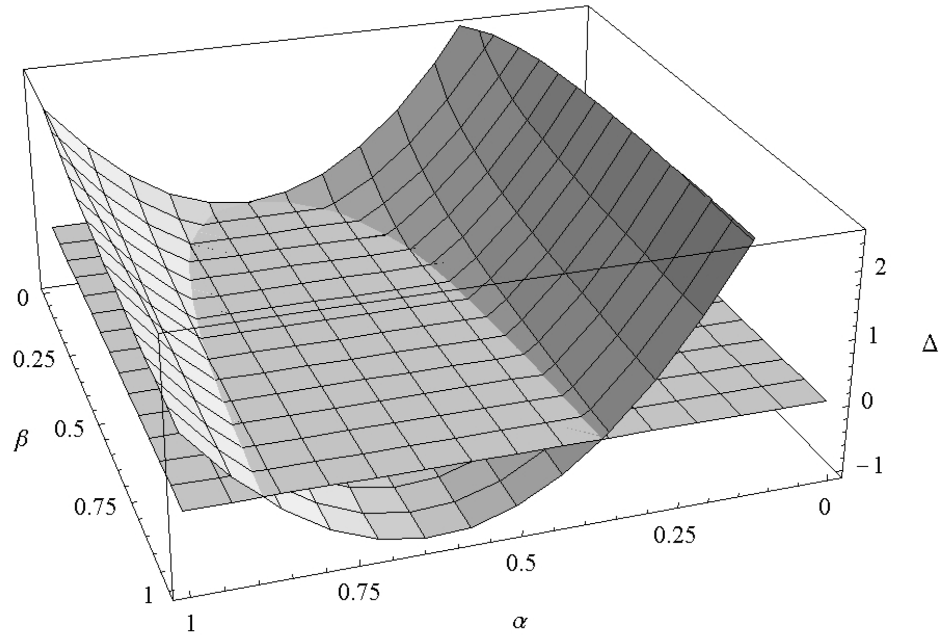

Figure 3: The difference function

> Since the productivity of type B programmers is the upper bound which the market is willing to attribute to them, the difference function can never be negative for type B programmers.[22] Although there is a special case in which the parameters have values that result in $\Delta_B = 0$ and the agents are indifferent between an OSS and a CSS career, this interior solution strictly features a positive gravitation field to its right and thereby violates our stability criterium. It is therefore not an equilibrium. The only viable equilibrium, which fulfils our stability conditions, is the right-hand corner solution. Consequently, type B programmers will always opt for the CSS. □

Whenever the possibility of an employment by the closed source system exists, type B programmers will join this system.[23] Therefore, we restrict ourselves in the following analysis to the case where $\beta = 1$. Equation 2.2 can be rewritten as

$$w(\alpha) = q_B + \frac{n_A(q_A - q_B)}{n_B + \alpha n_A} \cdot \alpha \quad for \quad \beta = 1 \tag{3.4}$$

The wage function for type A given that all type B programmers join the CSS can then be illustrated graphically as in Figure 4.[24]

[22]This statement holds true, even if we allow for non-independent deferred pay-off functions as explained under fn. 20.

[23]This result could be disputed on empirical grounds: There seem to be less skilled programmers involved e.g. at Linux. We suggest two possible responses. First, we believe that they are differently motivated than our career-concerned investors and that their presence does by no means have a negative effect on our investors. If anything, they are highly welcome since they provide valuable debugging. Franck and Jungwirth (2002), in fact, argue that in OSS both groups co-exist in symbiosis without crowding each other out. Secondly, we believe that, notwithstanding the first critique, in reality many low-end tasks with regard to open source products (e.g. documentation, maintenance, servicing) are performed in CSS-firms which accompany the OSS-project like satellites. It is very much as if the OSS were outsourcing unspectacular tasks, a phenomenon which

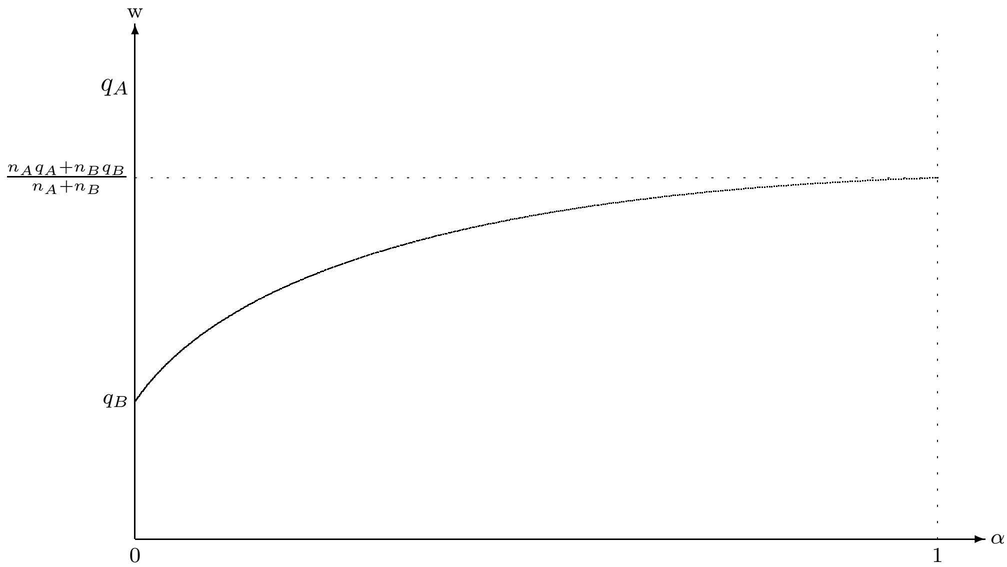

Figure 4: The wage function for $\beta = 1$

3.3 The decision of type A

In the following we will analyze the decision of type A, starting with their choice in the absence of any type B programmers before establishing the results for the case that both types are present.

Lemma 2 *If n_B were zero, all type A programmers would join the CSS.*

> *Proof:* If there were no type B programmers, the CSS wage would be q_A at all times. For $n_B = 0$, equation 2.2 yields $w = q_A$ and the difference function for type A reads
>
> $$\Delta_A = (q_A - v_A + \frac{4n_A^2 v_A}{k_A^2}[\alpha - (1 - \frac{k_A}{2n_A})]^2 \geq 0. \tag{3.5}$$
>
> In this special case with only one type of agent, firms are not at risk of employing a 'lemon'. No value is obtained by signalling and no rationally acting firm is ready to attribute a premium beyond the productivity of type A. In this special case v_A is bounded by q_A and Δ_A is always non-negative. From the non-negativity of the Δ-function it follows that $\alpha = 1$ is the only stable solution. □

Lemma 3 *For $n_B > 0$, the necessary condition for a type A programmer to join the OSS is that the premium attributed to type A, v_A, is higher than the productivity of type B, q_B, which is also the CSS minimum wage.*

---

could easily be explained by our result.

[24] One could argue that, if all type B programmers choose the CSS, participation in the OSS is already sufficient to signal the type to outside employers -without recourse on the changelog file. There are, however, two explanations which disqualify sole participation in an OSS as a signal: Firstly, it would enable bad programmers to imitate good ones. In fact, the separation in Lemma 1 would collapse. Secondly, in practice, any OSS programmer remains invisible until he is revealed by the changelog file.

Proof: After substituting 2.2 and 2.3 into 3.1 for type A, we obtain

$$\Delta_A = \frac{\alpha n_A(q_A - v_A) + \beta n_B(q_B - v_A)}{\alpha n_A + \beta n_B} + \frac{4n_A^2 v_A}{k_A^2}[\alpha - (1 - \frac{k_A}{2n_A})]^2$$

where the second addend is non-negative. The sign of the first addend is dependent on the relation between v_A and the productivities q_i. Since Lemma 1 established the fact that $\beta = 1$ the above equation can be rewritten to

$$\Delta_A = q_B - v_A + \frac{\alpha n_A(q_A - q_B)}{\alpha n_A + n_B} + \frac{4n_A^2 v_A}{k_A^2}[\alpha - (1 - \frac{k_A}{2n_A})]^2. \quad (3.6)$$

The last two addends are always non-negative and only the first can have a negative value within our definition space. Δ_A can only assume a negative sign if v_A exceeds q_B to a sufficient degree. q_B is the lower bound of the wage function and v_A represents the maximum value the deferred pay-off function can ever assume, $q_B < v_A$ is thus a minimum condition for the signalling mechanism to work. □

Lemma 4 *For $n_B > 0$ and $v_A > q_B$, there are parameter constellations which allow for a non-positive value of the difference function, which is the sufficient condition for a type A programmer to join the OSS.*

Proof: To prove Lemma 4, we choose the following parameter constellation:

$$k_A = 2n_A \qquad v_A = q_A$$

Equation 3.6 simplifies to

$$\Delta_A = q_B - v_A + \frac{\alpha n_A(q_A - q_B)}{\alpha n_A + n_B} + q_A\alpha^2$$

$\alpha = 0$ results in $\Delta(0) = q_B - v_A$ which is less than zero, if the necessary condition identified by Lemma 3 holds. □

3.4 Aggregate outcome

Lemmata 1 to 4 imply

Proposition 1 *The CSS exists in any case. The co-existence of an OSS is possible only under certain parameter constellations.*

The parameter-dependent structure and stability of the equilibria determine at what ratio the type A population can split up between the CSS and the OSS.

Corollary 1 *Some parameter settings establish a separating equilibrium as feasible.*

Proof: This follows directly from Lemma 4. □

A separating equilibrium is instated, when some type A agents rationally choose to forgo a wage in search for a signal that sets them apart from type B. Some settings even constitute a separating equilibrium, in which all type A programmers join the OSS. The last case is shown by the fact that $\Delta(0) \leq 0$ can be obtained as a result for certain parameter constellations. Note that the existence of multiple equilibria is possible. However, the exact coordination by which an equilibrium is reached is not modelled here.

Corollary 2 *Even if a separating equilibrium is feasible, it is not necessarily established. A pooling equilibrium is always a rival option.*

Proof: For all parameter constellations $\Delta(1) > 0$ by definition. Recall that $r(1) = 0$ and $w(1) > 0$. That is, the right-hand corner solution is always a possible equilibrium outcome. □

In a pooling equilibrium, all programmers regardless of their type work for the CSS.

So far, we have shown that the number of equilibria depends on the parameter constellation. We pursue this line of thought in a comparative static analysis highlighting the effect of the parameters $\frac{n_A}{k_A}$ and v_A in particular. Furthermore, we will look at what happens if the zero-profit condition for the CSS is dropped allowing it to set its wage arbitrarily.

4 Comparative statics

4.1 Analysis with respect to the population parameters

In the following we will analyze how the outcome is affected if changes in the population parameters k and n occur. We assume that type A is rewarded with a premium by the market that exactly equals his productivity, i.e. in this section $v_A = q_A$ holds. As shown by Lemma 1, $\beta = 1$. Although the dimension of k indeed renders it a population parameter, we will refer to it as the proxy for the industry's innovative potential.[25]

4.1.1 Equilibria

Lemma 5 *The number of equilibria depends on the relation between the number of type A programmers and the absolute number of programmers for which the deferred pay-off function assumes a value of zero, i.e. on* $\frac{n_A}{k_A}$.

Proof: See Appendix A.3. □

We can distinguish three cases:

- CASE (A) If
$$\frac{n_A}{k_A} < \frac{1}{2}(1 - \sqrt{1 - \frac{q_B}{q_A}}),$$
only the right-hand corner solution is an equilibrium. Given a certain innovative potential in the industry, the number of type A programmers is yet insufficient to sustain an OSS project as its overall quality would not provide their signal with enough visibility and credibility. The CSS exists alone.

- CASE (B) If
$$\frac{1}{2}(1 - \sqrt{1 - \frac{q_B}{q_A}}) \leq \frac{n_A}{k_A} < \frac{1}{2}(1 + \sqrt{1 - \frac{q_B}{q_A}}),$$
both corner solutions represent possible equilibria. The number of type A programmers relative to the industry's innovative potential is so large that the deferred pay-offs would exceed the pooling wage, if enough type A programmers created an OSS. At the same time, the population is yet too small for an excessive competition effect to press the expected value of the deferred pay-off back below the wage level. An OSS is a feasible outcome.

[25] Compare fn. 18.

- CASE (C) If
$$\frac{1}{2}(1+\sqrt{1-\frac{q_B}{q_A}}) \leq \frac{n_A}{k_A},$$
the difference function has two roots of which the left hand one is an equilibrium. It joins the right-hand corner solution as a possible outcome. The type A population is now so large relative to the industry's innovative potential that with decreasing α, from a certain threshold on, the expected value of the deferred pay-off again slips beneath the wage offered at the CSS. Due to this competition effect, the two remuneration functions now intersect twice.

Figure 5 illustrates the three cases graphically.

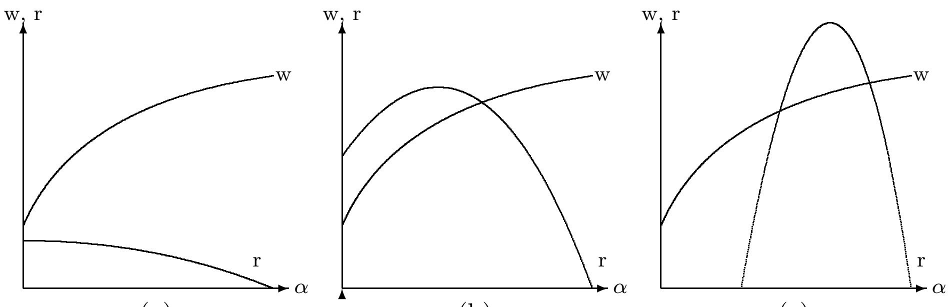

Figure 5: Changes in the population parameters

Note that the lower boundary condition of case (B) is the necessary condition for a forking of projects. It denotes a critical mass in the population of type A needed for an OSS-project to possibly subsist. Lemma 5 implies

Proposition 2 *Given a well-developed information transfer mechanism ($v_A = q_A$), an OSS can only emerge, if the type A population is sufficiently large relative to the industry's innovative potential.*

Note that the above proposition only denotes a necessary condition for the possibility and the sustainability of an OSS . Whether and when an OSS will actually come into existence remains unanswered by this analysis.

4.1.2 OSS threshold and trigger

Even though the unique intersection between wage and deferred pay-off in case (B) and the right intersection in case (C) yield a value of zero for the Δ-function, these points are no sustainable equilibria. Nevertheless, they play a significant role, since they represent a critical threshold. Once α falls below this threshold, the industry is drawn away from the monopoly situation where all work for the CSS. Instead a duopoly structure of the industry is established where type A programmers can distinguish themselves from the type B programmers and thereby signal their superior productivity. Any event or action that pushes α across this threshold, triggers off the establishment of a sustainable OSS. Any analysis of such trigger events aims at answering the above questions of whether and when an OSS emerges.[26] We propose that the closer the threshold is to $\alpha = 1$, the more likely is a trigger event.

[26]E.g. Franck and Jungwirth (2002) consider ideologically motivated *donators* as those who trigger off the emergence of OSS.

4.1.3 Discussion and practical relevance: evolutionary potential of industry

In our model, the effect of n_A and k_A should always be analyzed in combination. Consider e.g. the following extreme cases:

1. With k_A approaching zero in the limit, the deferred pay-off curve approximates the function $\alpha = 1$. This also makes sense intuitively: Given there were no room for innovation, who could set himself apart by innovation? No matter how large the population of workers in that industry were, an OSS would not emerge. For this reason, mature industries will seldom witness open source business models.

2. Conversely, consider k_A very high, but n_A verging on zero. Despite a high innovative potential, there could be too few good programmers in the industry to create a prestigious open source project. In the extreme case of $n_A = 0$, only one type of programmer would remain eliminating the need for signalling (and thus an OSS) regardless of k_A.

Different constellations of the population parameter $\frac{n_A}{k_A}$ can signify various evolutionary stages within one industry or characteristics of different industries. In principle, the emergence of an OSS is driven by the desire of the better-skilled programmers to emit a signal to set themselves apart from the less productive programmers. However, our analysis shows that such a development is contingent on the industry's room for innovation as well as on the number of people who can tap and exploit this potential.
The analysis of $\frac{n_A}{k_A}$ offers yet another interesting aspect. Let us assume n_A and k_A are sufficiently large. As the industry grows and a present OSS-project becomes too small to accommodate all good programmers, an extension of projects within an OSS may lead to a higher k_A or more OSS-projects may emerge and expand into other areas of software programming. These projects might even compete as to which one owns the better signalling mechanism. We perceive two balancing forces at work: On the one hand, an increasing population of good programmers will raise the need for differentiation and, in turn, the number, scale and scope of the OSS. On the other hand, whenever a good and successful OSS-project is established the incentive for the programmers to take this project private and place some copyright on its product gets stronger. Some projects start on a small scale in the OSS, but may at some point turn into private ventures.
Which structure will the industry settle on in the long-run? Our guess is that, even though a CSS mono-existence is always feasible, as long as credible and transferable job market signals can be gained by innovation, i.e. evolutionary potential is high, the industry will likely oscillate around equilibria with multiple systems - open and closed.

4.2 The importance of information transfer mechanisms

4.2.1 Equilibria

In the preceding paragraph the maximum deferred payoff obtainable for the type A programmers was assumed to equal their productivity q_A. Now we return to the analysis of the general case where the maximum deferred pay-off is not preset to the individual productivity. To concentrate on the effect of a change in v_A, we assume in this section with a slight loss in generality that the absolute number of programmers for which the deferred pay-off function has a value of zero - k_A - would equal the population of type A programmers, i.e. $k_A = n_A$. The reputation function for type A then simplifies to

$$r_A(\alpha) = -4v_A[\alpha - \frac{1}{2}]^2 + v_A.$$

With this function, deferred payoff will be zero for an α of either zero or one and the maximum remuneration value that can be gained by working for the OSS will be reached if the population of good programmers exactly splits up between the CSS and the OSS.
Depending on the value of v_A there exists either just one equilibrium for $\alpha = 1$ or a situation with two possible equilibria: the CSS-only outcome and a mixed outcome with both CSS and OSS in place. Starting with a low v_A any increase in v_A will first establish intersections between the wage curve and the deferred pay-off function and then shift the intersections to the extremes. The right-hand intersection of the wage and deferred pay-off function is an instable saddle point and represents the threshold separating the gravitation fields of the other two equilibria (the right-hand corner solution with solely CSS to the right and the mixed outcome to the left). Figure 6 illustrates three cases with an increasing v_A graphically.

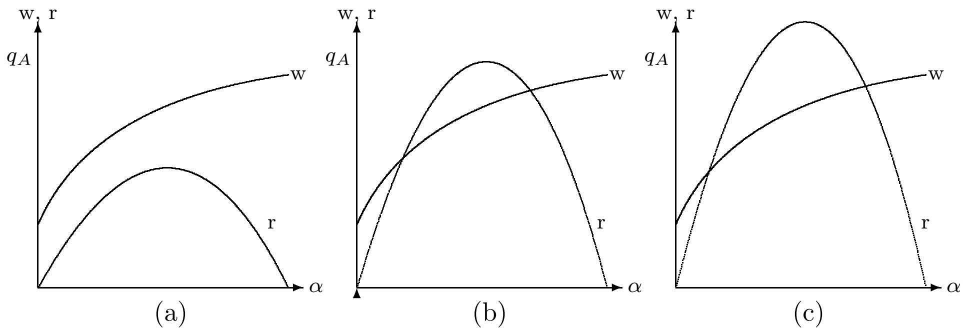

Figure 6: Changes in the institutional parameters

Our analysis leads us to

Proposition 3 *The co-existence of OSS and CSS is only a viable equilibrium if the institutional environs (or market surroundings) allow OSS programmers to credibly transfer a signal to the market. The higher the valuation of the market for outstanding performance in the OSS, the more likely is a shift from mono-existence of CSS to a mixed equilibrium.*

4.2.2 Discussion and practical relevance: the rise of information transfer mechanisms

A low value for v_A indicates a non-existent or insufficiently developed market mechanism to remunerate signals gained in the OSS. The markets in which the OSS is embedded need good information flows to substitute future rewards for such signals. The emergence of the internet provided more transparency, as a better transmission of information became possible. This effect is captured by an increase in v_A.
Graphically, this means that the parable shifts upwards and, at some point in time, intersects with the wage graph. The stable, left-hand equilibrium and the instable, right-hand intersection are established. Any further increase in the quality and accuracy with which the market takes the information about the work of type A programmers into account shifts the intersections to the extremes - lowering the threshold for leaving the CSS and facilitating the emergence of OSS.
Not only these benign effects of markets caused by the technological advances in information processing lead to a lower threshold, but also any mis-pricing that might

occur in times of a bubble economy: Akin to the dot.com bubble up to the year 2000 in the sector of Technology, Media and Telecommunications (TMT) and the subsequent over-investment in resources in these areas, the over-shooting of the market also led to an over-investment in open-source projects and possibly over-pricing of high-potential IT specialists. In our model, an overshooting of the valuation for OSS programmers represented by a value of $v_A > q_A$ can have two reasons: On the one hand, firms offering career opportunities for OSS programmers might have a specific production function which employs the signals of their employees as one production factor, resulting in additional profits generated by the mere fact that programmers of high reputation are associated with the company. An example for this is Linus Thorvalds who now works for Transmeta, an internet startup developing low-power microprocessors in an OSS-like development process.[27] On the other hand it may well be possible, that potential employers of OSS programmers form overshooting beliefs about the productivity of these agents. Such uncertainty or overshooting has the effect of triggering new OSS projects more easily as individual programmers try to exploit the trade-off between the visibility and the competition effect. As the hype and overshooting ebbed away, many of these promising projects were quietly cancelled.[28]

4.3 Strategic wage setting

We initially assumed that a representative firm in the CSS sets its wage level so that it makes no profits. In this section we drop this assumption to see what happens if the CSS can change the level of payment to its employees. We still stick with the assumption that there is a uniform wage for all programmers employed by the CSS. The pooling wage w may now be freely set and varied. We also assume a parameter constellation that allows for two intersections of the wage curve and the deferred pay-off function. To simplify, we assume $k_A = n_A$ in our formal analysis.
The former wage curve serves as a benchmark. With this wage function given by equation 2.2, profits for the CSS are always zero. We denote this case by w^0. In contrast to that, we define arbitrary wage as the actual wage that the CSS pays all of its employees and denote it by $\bar{w}$.
The minimum arbitrary wage that a CSS-firm can pay its employees equals the minimum value of the zero-profit wage: $\bar{w}_{min} = w^0_{min} = q_B$. Figure 7 shows us four different arbitrary wage levels ($\bar{w_{min}}$ to $\bar{w}_3$). For each wage level, we get different points of intersection between $\bar{w}_j$ and r_A. The lower the wage level is, the farther apart are the intersections other things being equal. $\bar{w}_3$ clearly shows that once $\bar{w}_j$ surpasses the maximal reputation v_A, there is only the CSS-only equilibrium for $\alpha = 1$. Increasing the wage level reduces the profit zone, i.e. the range between $\alpha = 1$ and the intersection between $\bar{w}$ and w^0.
The sequence of events is as follows: (1) The CSS arbitrarily chooses a wage $\bar{w}$. (2) This wage level determines the equilibria, (3) which in turn imply the respective level of profit for the CSS-firm. We follow this thread in our analysis.

4.3.1 Equilibria

Imagine the CSS sets an arbitrary wage $\bar{w}$. Programmers make their decision on the basis of this wage. The difference function therefore incorporates $\bar{w}$ instead of w^0. The same holds true for the condition for an interior solution which turns into:

$$\bar{w} = r_A.$$

[27]See http://www.transmeta.com for more details.
[28]See http://www.sourceforge.net for a listing of inactive projects.

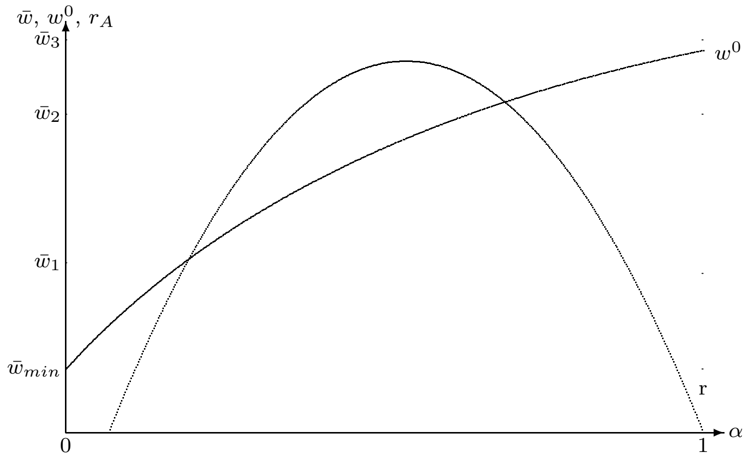

Figure 7: Strategic wage setting

Lemma 6 *Decreasing the wage by the CSS increases its profits. Any decrease in the wage level, however, moves the threshold closer to the CSS-only equilibrium, thereby increasing the chance for an OSS to be triggered.*

Proof: In the symmetric case of $k_A = n_A$, substituting 2.3 into the above equation yields:

$$\bar{w} = -4v_A[\alpha - \frac{1}{2}]^2 + v_A$$

Solving this equation for α gives us the following interior solutions for the redefined difference function:

$$\alpha_1^* = \frac{1}{2} - \sqrt{\frac{v_A - \bar{w}}{4v_A}}$$
$$\alpha_2^* = \frac{1}{2} + \sqrt{\frac{v_A - \bar{w}}{4v_A}}$$

As can clearly be seen, we get no solutions for $\bar{w} > v_A$. For $q_B \leq \bar{w} \leq v_A$, the effect which $\bar{w}$ has on the positions of the equilibria can be summarized by

$$\frac{\partial \alpha_1^*}{\partial \bar{w}} > 0 \quad and \quad \frac{\partial \alpha_2^*}{\partial \bar{w}} < 0$$

For increasing $\bar{w}$ the right-hand intersection travels to the left, while the left-hand intersection (and equilibrium) goes to the right. They meet comprising a tangency point for $\bar{w} = v_A$. After that, no intersection exists. Conversely, decreasing $\bar{w}$ augments the distance between α_1^* and α_2^*. □

The CSS is able to influence the position of the equilibrium outcome where both a CSS and an OSS exist. This allows a determination of the size of the gravitation fields of the two equilibria. E.g. decreasing the wage level enlarges the negative gravitation field between the two intersection points at the expense of the outer fields.

4.3.2 Equilibrium profit

Only the interior solution α_1^* is a possible equilibrium besides the CSS-only outcome for the corner solution of $\alpha = 1$. We therefore have two potential settings for any arbitrary wage level below v_A and can calculate the CSS profit per programmer for the two cases by

$$\pi_1(\bar{w}) = w^0(1) - \bar{w}$$
$$\pi_2(\bar{w}) = w^0(\alpha_1^*) - \bar{w}$$

Total profit is then calculated by

$$\Pi_1(\bar{w}) = [w^0(1) - \bar{w}] \cdot (n_A + n_B)$$
$$\Pi_2(\bar{w}) = [w^0(\alpha_1^*) - \bar{w}] \cdot (\alpha_1^* n_A + n_B)$$

Since α_1^* is itself a function of $\bar{w}$, the profit functions are solely dependent on $\bar{w}$.

4.3.3 Discussion and practical relevance: Playing fields and strategies

We have two possible outcomes. If we presume that CSS will sensibly avoid to create any long-term equilibria where it induces a loss, the possible outcomes will range somewhere either to the left of the left-hand interior solution or to the right of the right-hand interior solution in Figure 7. We call them the playing fields because these settings require different strategic considerations by the CSS-firm.

- Playing field 1 - Monopoly:
  Playing field 1 is relevant if the CSS exists alone, i.e. while the industry rests in the right-hand corner solution. On playing field 1, the CSS will want to maximize its profit $\Pi_1(\bar{w})$ by setting $\bar{w}$ to its minimum q_B. This will increase the chances of OSS triggering.[29] This danger will more or less hinder the CSS from fully exploiting the type A programmers. The CSS needs to trade off the profit maximization against safe-guarding its monopoly.

- Playing field 2 - Duopoly:
  Playing field 2 is relevant when the CSS and the OSS co-exist, i.e. while the industry rests in the left-hand equilibrium. On playing field 2, the CSS can pursue two strategies. It can try to reach playing field 1 again by setting a wage higher than r_A, thereby eliminating the negative gravitation field and thus tempting type A programmers away from the OSS. It would have to put up with losses with this foreclosure strategy until the feat is done. This only makes sense if the CSS expects to be compensated for these by future monopolistic rents. Otherwise, it could accept the duopoly situation, settle for the left-hand equilibrium and maximize its profit function $\Pi_2(\bar{w})$.

This implies

Proposition 4 *Given the other parameters render the existence of an OSS viable in the zero-profit case, the CSS - by uniformly varying the wage parameter - is not able to influence the number of possible equilibria without incurring losses. In such a monopoly situation the CSS can reduce the chance of a trigger event by raising its wage, but the market is always contestable by the OSS.*

Note that even if we relaxed the parameter assumptions at the start of this section, we would find that, for all constellations, the zero-profit case minimizes the number of equilibria. Thus, the CSS can never create a permanent situation with less equilibria, if its wage policy remains non-discriminatory.

[29] In fact, in some situations the CSS may itself create the possibility of an OSS. Consider case (a) in Figure 6. By setting its wage lower than v_A, it actually grants an OSS room to maneuver where there had been none before.

5 Discussion of the model

What motivates highly skilled people to commit valuable effort to an open-source product? Indeed, they devote time and resources without being able to recoup their personal investment by retrieving the ensuing rents. Put briefly, their labor creates a public good. Any economist would allege that private provision of a public good should therefore generally suffer from under-investment. Surely, beyond altruism, incentives to invest ought to be weakened by open access to a good. This is why private ownership plays such a prominent role in our societies. In fact, it is a *de rigueur* premise to any market-based economy.

We see institutions actually fulfil the function of helping investors claim their righteous rents when investments create a good which is subject to free-riding, plagiarism or imitation. In the case of literary, dramatic, musical or artistic works, sound records, films, television programs as inventions, the rights of the author as an inventor are protected by intellectual property rights, i.e. legal institutions such as copyrights or patents respectively. The protection of intellectual property rights is not only a matter of law but increasingly a matter of practicability as it has become visible in the way that the Internet has allowed its users to breach copyrights for sound recordings of the entertainment industries on a hitherto unknown scale. Fact is that a record whose sold copies are freely distributed on the Internet by its purchasers becomes a public good. Surely this would reduce the number of records bought and consequently impede the amount of investments made in the production of records. Which goes to show that property matters.

Generally, one would conclude that the more one is able to restrict ownership over a good, the more rents can be reaped by its use or sale. The software industry often establishes effective property rights by technically restricting the access to the source code or any modules of their products. The right to make alterations to the product is reserved for designated programmers who usually only have insight into parts of the code. For these firms, unrestricted access to their source code would mean abandoning their property rights and would amount to economic suicide. But that is exactly what the open source domain implies.

In the line of the above arguments, we should not observe the OSS to survive in economic reality. Conversely, it is a true surprise that we actually do. Our model tries to unravel this mystery. In our model, the mediocre programmers at the CSS, who earn more than their productivity, appropriate some of the better programmers' output. They actually free-ride.[30] I.e. while the CSS as a whole establishes ownership over its product, the better programmers within the CSS can only incompletely do so over their personal contributions. On the other hand, in an OSS, the institution as a whole exerts no property right over its product, but inside of it fosters the establishment of individual intellectual property for its contributors. Thus, this is what our analysis puts forth as a possible explanation for the existence of open source systems. The driving element is that the systems differ in terms of remuneration: The CSS pays a non-discriminatory, pooling wage, whereas the OSS by way of a separating deferred pay-off function offers better programmers the possibility to set themselves apart from the less skilled and reap the equivalent of their own product. Given this basic mechanism, we show the conditions required for a CSS to be sustainable. In our analysis, we conclude that

1. an open-source system will never exist alone, though a closed-source system can. The reasoning is quite simple: If high-end programmers feel a need for differentiation, there will be low-end programmers for whom it certainly is

[30] See Rajan and Zingales (2000) for an excellent model of a stylized firm with unequal endowments in resources between different stake-holders and their predictions about the allocation of property rights in such a setting.

not advisable to choose an OSS career. If no such need is felt, then everyone works at the CSS anyway;

2. the OSS needs a critical number of high-end programmers relative to its industry's innovative potential in order to reach a level of quality and visibility which makes their signals credible. Therefore, OSS becomes feasible only if the population of high-end programmers is relatively large;

3. a well-developed mechanism to transfer the signals to the surrounding environs is a prerequisite for OSS. The job markets must acknowledge the information discovery service of the OSS. As such, it may be that the OSS must establish a reputation for being a reliable signalling device;

4. provided the conditions in 2 and 3 hold, there is always a positive probability that an OSS may come into existence. Although the CSS may reduce this probability by strategically setting its wage, the possible emergence of an OSS can never be totally ruled out.

In sum, our analysis shows why and how an equilibrium is viable in which both, CSS and OSS, co-exist.

5.1 Empirical implications

To check the validity of our model, we propose that empiricists assess the following hypotheses in our model, which allude both to underlying assumptions as well as predictions.

1. There are ex-OSS programmers who cash in on their reputation. Their wages are substantially higher than what the best earn at a CSS-firm.

2. There is a visible differentiation within OSS-projects as to the skills of their programmers, e.g. a ranking or informal hierarchy.

3. There is no (established) industry in which only OSS-projects exist. Put differently, any considerable OSS is embedded in a CSS environment.[31]

4. OSS-projects are not encountered in industries with low innovation. Conversely, they are more common in highly innovative sectors.

5. As institutional parameters like better information transfer mechanisms and legal devices (e.g. the General Public Licence) develop, OSS-projects become increasingly feasible.

6. The success of OSS-projects has had some influence on the remuneration policy in CSS-firms resulting either in changes in the wage level or in attempts to imitate the selection processes of the OSS.

[31] The mitigating words 'established' and 'considerable' are deliberate. When innovative industries are in a embryonic state, small-scale OSS-projects might exist alone. Imagine a group of scientists freely sharing their knowledge and jointly developing a completely new technology. Over time and with more and more success of their product, the incentive to take the venture private rises. By establishing property rights and starting to sell the technology the OSS-project becomes a CSS-firm. The software industry as a whole has been born this way.

5.2 Analogy with academia

To back our arguments, we would like to draw a rather imperfect but nonetheless telling analogy. Why would a successful university graduate forego years of wage and even spend money to earn a Ph.D. degree? Little of what he produces can be appropriated by him alone. His research is openly published and particularly accessible to other scientists. He works in what strongly resembles an open source environment.[32] We argue that he seeks a signal to stand out from the common mass of college graduates. Should he be successful, he creates a signal via the academic degree which also functions on the basis of peer review and which he hopes will materialize in some way afterwards.[33] Imagine he has the choice between a certain university U and the firm F. In F he will receive a fixed wage. In U he must pay a tuition for an opportunity to earn his Ph.D. degree. According to our model, his choice of U will - given that F's wage is fixed - depend on the following two considerations: Firstly, the university will have to offer a promising research environment in general, i.e. other talented scientists and Ph.D. students in particular. They represent the reputation of U which in turn contributes to the value of a degree earned from that university. If their number is too low, the Ph.D. degree might prove wanting in value and later not yield a deferred pay-off as high as desired. Secondly, if there are too many Ph.D. candidates and the level of competition (or required quality) is very high, the challenge might appear too tough for the aspirant. In any case, only those who think they are talented enough will tread this path. Otherwise, they will choose F. In time, institutions such as U may develop a reputation for screening quality, as a consequence of which vital information transfer mechanisms (e.g. job fairs, student workshops, etc.) may evolve and establish themselves around them. Conceivable are models of competition and differentiation between such institutions and their environments. Our basic model simply suggests that there should be an equilibrium number of Ph.D. students for any such institution and the labor markets in which they are embedded.

6 Possible extensions

We believe that our model still holds some promise for research and consider several strands along which it should be enhanced and perhaps rectified. Firstly, our model is slightly more than simple comparative static analysis. We have implicitly assumed that all agents have perfect foresight about the equilibrium in which the industry will eventually settle in. We have neglected how our interpretation suggests that the OSS represents only a transitory stage in the career of an individual programmer wherefore the OSS should have to cope with a higher fluctuation than the CSS. In order to correct these shortcomings, one would have to build a more dynamic model in which at each point in time a proportion of the incumbent programmers graduates from the systems while newly arriving programmers with imperfect knowledge and foresight decide upon their careers. As a consequence, the model would have to incorporate expectation building and beliefs within the framework of a fully-fledged dynamic analysis. Another very important aspect is that the programmers might face a coordination failure. If we assume once-and-for-all career decisions without job switching, then each generation of newly arriving programmers plays a non-repeated coordination game, the working of which has to be explored in more depth. We say non-repeated, because it does not recur with the same individuals, though the game itself is repeated every generation. It might

[32] Interestingly, there is a bi-monthly magazine for university researchers in Germany entitled *opensource - the network magazine for research assistants.* For more information, visit www.opensource-online.de.

[33] See also Franck and Jungwirth (2002, fn.18).

be interesting to examine the possibility and effect of inter-generational learning in this context.
Secondly, we have so far not discussed the welfare aspects of the emergence of an OSS. One could do so both from an individual as well as public perspective. It would be especially interesting in this regard to vary the effort level of the programmers, which in our present model is constant, and allow thereby for moral hazard. Interesting welfare aspects would likewise be created, if we were to use synergetic instead of linear production functions. One could furthermore think of the reduction of monopoly costs, the mitigation of the public good under-investment problem, and the fact that the programmers themselves derive utility from the usage of the resulting product.
A further strand of analysis along this line is the inclusion of the demand side. In the case of operating systems this is basically a battle for standards with network effects and the interesting peculiarity that one product is offered for free. Many of these welfare, competition and demand-side consequences could be modelled within a tournament model to capture the rat race aspect of parallel development and high-powered incentives.

A Appendix

A.1 The competition effect

The following is a simple theoretical explanation for assumption 2 of the deferred pay-off function. Imagine

1. there are n programmers of the same type (productivity).
2. there is a total of m known problems to be solved.
3. a programmer can only work on one problem at a time.
4. only one solution is implemented in the actual product. I.e. if several programmers offer different solutions for the same problem, only one gets chosen. This is the "winner-takes-it-all" principle.
5. the programmer whose solution is implemented can henceforth credibly signal his productivity, because the open source makes the programmers' individual contributions transparent.

As long as $n \leq m$, each programmer may work on a problem alone. Competition can be avoided. Once $n \geq m$, at least some programmers are bound to contend over the same problem. On average, there will be

$$x = \frac{n}{m}$$

programmers working on one problem. If we assume that every programmer has the same chance of winning the signal, then the expected value for each of them is

$$E[r_i] = prob(x) \cdot q_i$$

where $prob(x)$ is the individual probability of success given a number of x competitors and q_i is the certainty value of the signal. The chances being evenly distributed, $prob(x)$ is defined as

$$prob(x) = \frac{1}{x}$$

Substituting the equations into each other yields

$$E[r_i] = \frac{m}{n} \cdot q_i$$

Clearly,

$$\frac{\partial E[r_i]}{\partial n} < 0$$

and

$$\lim E[r_i] = 0 \qquad as \quad n \to \infty$$

A.2 Derivation of the deferred pay-off function

To approximate the deferred pay-off function for type A, we use a quadratic function of the form

$$r_A(\alpha) = -c \cdot (\alpha - \alpha_{max})^2 + r_{max}$$

where c, α_{max} and r_{max} are the unknown parameters. From our assumptions, the following conditions can be postulated:

$$\begin{aligned} r_{max} &= v_A \\ r_A(1) &= 0 \\ r_A(1 - \frac{k}{n_A}) &= 0 \end{aligned}$$

This system is determined as we have three independent equations for three unknown parameters. From this we get

$$r_A(\alpha) = -\frac{4n_A^2 v_A}{k_A^2}[\alpha - (1 - \frac{k_A}{2n_A})]^2 + v_A$$

The procedure for type B is analogous.

A.3 Proof of lemma 5

The intercept of the wage function is always q_B which at the same time is its minimum value (see figure 4). The intercept of the deferred pay-off function varies dependent on $\frac{n_A}{k_A}$. We will look at the difference function of type A with $v_A = q_A$. For $\alpha = 0$, we get

$$\Delta_A(0) = (q_B - q_A) + \frac{4n_A^2 q_A}{k_A^2} \cdot (1 - \frac{k_a}{2n_A})^2.$$

Rearranging this leads to

$$\Delta_A(0) = \frac{4n_A^2 q_A}{k_A^2} - \frac{4n_A q_A}{k_A} + q_B. \tag{A.1}$$

In the limits $\Delta_A(0)$ behaves as follows

$$\begin{aligned} \Delta_A(0) = qB > 0 &\qquad as \quad \frac{n_A}{k_a} \to 0 \\ \Delta_A(0) = \infty > 0 &\qquad as \quad \frac{n_A}{k_a} \to \infty \\ \Delta_A(0) = q_B - q_A < 0 &\qquad as \quad \frac{n_A}{k_a} \to \frac{1}{2} \end{aligned}$$

Since this is positive for the corner values of $\frac{n_A}{k_A}$ and negative for a value inside this range, being monotonous the difference function must have two roots for $\alpha = 0$ dependent on $\frac{n_A}{k_A}$. Figure 8 shows us its behavior for a steadily increasing $\frac{n_A}{k_A}$. The sequence is to be viewed from left to right and top down. Note that the sign of $\Delta(0)$ changes in (b) and (d).

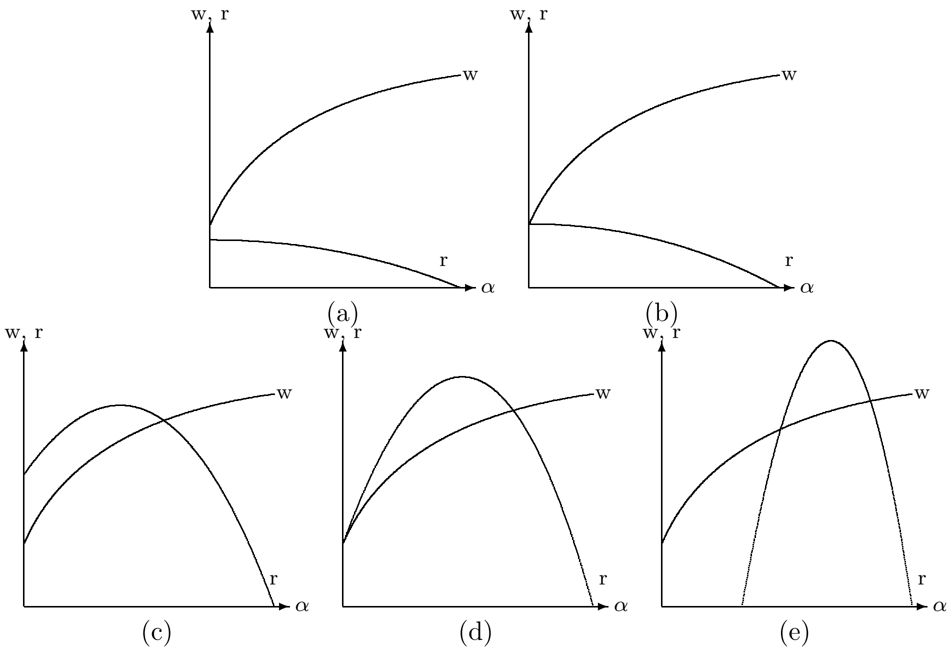

Figure 8: The effect of a changing $\frac{n_A}{k_A}$

It should also be noted that in

- (a) there is no intersection
- (c) there is one intersection
- (e) there are two intersections

between the wage and the deferred pay-off function. Thus (b) and (d) with $\Delta_A(0) = 0$ mark not only the change of the sign of $\Delta_A(0)$, but also the changes in the number of intersections between the two underlying functions. Next, we calculate the exact level of $\frac{n_A}{k_A}$ in (b) and (d). For this, we set $\Delta_A(0)$ to zero in equation A.1 which after some rearranging gives us

$$z^2 - 2z + \frac{q_B}{q_A} = 0 \qquad with \quad z = \frac{2n_A}{k}$$

Solving this quadratic equation for z yields the following solutions for $\frac{n_A}{k_A}$

$$\frac{n_A}{k_A} = \frac{1}{2} \cdot [1 \pm \sqrt{1 - \frac{q_B}{q_A}}].$$

Since the term under the root is between zero and one, we get two positive and therefore viable solutions.

References

Deutsche Bank Research (2002): "Free software, big business?," *http://www.dbresearch.com*, (32).

Economist (2000): "Open sesame," *The Economist*, Apr. 13^{th}.

——— (2001): "The penguin gets serious," *The Economist*, Jan. 25^{th}.

——— (2002): "Going hybrid," *The Economist*, Jul. 27^{th}.

Franck, E., and C. Jungwirth (2002): "Reconciling investors and donators - The governance structure of open source," *Universität Zürich, Working Paper No. 8.*

Hars, A., and S. Ou (2002): "Working for Free? Motivations for Participating in Open-Source Projects," *International Journal of Electronic Commerce*, 6(3), 25–39.

Johnson, J. P. (2002): "Economics of Open Source Software," *Journal of Economics & Management Strategy*, 11(4), 637–662.

Kaisla, J. (2001): "Constitutional Dynamics of the Open Source Software Development," .

Lakhani, K., and E. von Hippel (2000): "How Open Source software works: 'Free' user-to-user assistance," *MIT Sloan School of Management Working Paper*, (4117).

Lerner, J., and J. Tirole (2002): "Some Simple Economics of Open Source," *Journal of Industrial Economics*, 52, 197–234.

Moon, J. Y., and L. Sproull (2000): "Essence of Distributed Work: The Case of the Linux Kernel," *Firstmonday, http://www.firstmonday.dk/issues/issue5_11/moon/*, 5(11).

Mustonen, M. (2003): "Copyleft - the economics of Linux and other open source software," Forthcoming in: Information Economics and Policy.

Rajan, R., and L. Zingales (2000): "The tyranny of inequality," *Journal of Public Economics*, 76(3), 521–558.

Raymond, E. S. (2000a): "The Cathedral and the Bazaar," *http://www.tuxedo.org/~esr/writings/cathedral-bazaar.*

——— (2000b): "Homesteading the Noosphere," *http://www.tuxedo.org/~esr/writings/cathedral-bazaar/homesteading/.*

Spence, M. (1973): "Job market signaling," *Quarterly Journal of Economics*, 87(3), 355–374.