context of the speech but of its excess, where context is lost or turns in on itself recursively.[72]

Confronting the notion that machines merely do what they are programmed to do, the understanding of programming as a performative speech act extends the unstable relation between the activities of writing, compiling, and running of code as a set of interconnected actions (as the practice of live coding demonstrates so well). Indeed, saying words or running code or simply understanding how they work is not enough in itself. What is important is the relation to the consequences of that action, and that is why the analogy of speech works especially well—although ultimately the phrase "speech act" may not be sufficient for the purpose.[73]

The various attempts to capture the voice in speech machines and software reveal the futility of the endeavor. In resisting the forces of rationality, it challenges normative communication and synthetic technologies of the general economy, resonating more with the notion of excess.[74] That the voice cannot be fully captured by computation is very much the point, as it provides inspiration for new ways of working with code and examining "code acts," to reveal other possibilities and motivations.

2 Code Working

The phrase "code working" draws attention to the work involved in writing code, as well as the work that code does once executed. If physical, intellectual, and machine labor have become less and less differentiated, as labor has become more immaterial, collective, and communicative,[1] then labor's performative and linguistic (speechlike) qualities also require new emphasis. The workings of code encapsulate these qualities, not least in recognition that work is increasingly practiced at the behest of scripts.

The chapter emphasizes the cultural materiality of working with code, insofar as all codework necessarily carries with it the labor that has been invested in its production, as well as the labor invested in the broader apparatuses through which it is served. The code artist Alan Sondheim summarizes the relations of production in coding and its potential to reveal hidden layers of operation: "Every more or less traditional text is codework with invisible residue; every computer harbors the machinic, the ideology of capital in the construction of its components, the oppression of underdevelopment in its reliance on cheap labor."[2] Indeed, one should not underestimate how much software development is outsourced to software houses in parts of the world where labor comes cheap. The trick, on both global and local levels, is to hide the code content and coding labor under a deceptive rhetorical form, rather than reveal the contradictions of value and hence divisions of labor involved in production. Therefore the discussion of work in this chapter is allied to coding in a deliberately ambiguous way, to indicate the work involved in making software as well as the work that software does itself (in parallel to the way that speech both says and does something at the same time). These comments apply to cultural and artistic work in particular, inasmuch as this is where the software and the work of the programmer combine with hardware in performative ways. This indicates a conceptual turn in the consideration and practice of coding work, as work has become more identified with intellectual endeavor and linguistic competence. The point is that the very contradictions of labor value are encoded in ever more complex ways, through "machinic" social relations that involve humans *and* machines, and activities related to the networked extensions of the body and the intellect.[3]

A familiar line of argument describes how commercial software closes off the source code that the programmer works with, and thus makes its workings relatively inaccessible to further use or indeed criticism. The proprietary software compiles the code into an executable version that locks down the source, offering a useful analogy to the ways in which capital protects its interests. In response to this, many software producers have made the source code an integral part of the work, quite literally in their production of codeworks (as for instance with Harwood's *Perl Routines to Manipulate London*, cited in the previous chapter). Thus it becomes possible for codeworks to reveal some of the contradictions over production involved in working with code, in parallel to labor conditions and class struggle more broadly. Harwood's codework *Class Library* (2008) plays on these inherent antagonisms, along with the double coding of the term "class."[4] Below is a short extract:

```
package DONT::CARE;
use strict; use warnings;
sub aspire {
my $class{tab}          = POOR;
my $requested_type      = GET_RICHER;
my $aspiration{tab}     = "$requested_type.pm";
my $class{tab}          = "POOR::$requested_type";
require $aspiration;
return $class->new(@_);
}
1;
```

Evidently an understanding of work can be developed that takes account of the dynamic character of social processes, using secondary notation to form arguments as well as the internal contradictions that are evident in code itself. New contradictions are expressed in the complexity of coding labor relations, characterized not least by the production of free software and open licenses, which rely on the sharing of source code and expertise. The chapter elaborates on these issues and the paradoxes that arise, to demonstrate how the practice of coding extends social relations from the basic interaction of workers to their interactions with machines and code. It addresses the question of how working with code relates to work and action, firstly by discussing the ways that new ideas emerge through the loops of historical processes.

The dynamic relation between code and the actions that arise from it are an indication of wider recursive processes. The second section of the chapter considers how work has incorporated the communicative and linguistic dimension, facilitated by networked technologies and collective formations of work that together can be understood through the concept of "general intellect" (taken from the "Fragment on Machines" in Marx's *Grundrisse*).[5] The concept indicates the productive connection between networked machines and collective human intelligence, or what might be more usefully referred to here as networked intelligence, materialized in software practices that share code. The final section develops ideas of codework more specifically in relation to Arendt's distinctions between work and action,[6] to dispute deterministic ideas that program code is simply a means to an end (its functional application), or an end without means (necessarily closed off, locked down by commercial imperatives). Something far more paradoxical is offered, informed by artistic and other idiosyncratic interventions. Underlying the chapter as a whole is an exploration of the conditions on which labor relations are based, speculating on the possibility of work and code operating for-themselves in a speechlike manner.

Code in-itself

If coding is simultaneously saying something and doing it, what came before? Clearly someone conceives the program, as indicated by the etymology of the word "program" from the Greek *programma*, a written notice to the public. It indicates a procedural way of doing things, which is important for understanding computational processes more broadly: the logic of "*if* something *then* something else," for instance. As a noun, "program" is a description of a future event and a set of instructions used to execute a specified task. As a verb, it describes the process of the arrangement of the program as the expression but also the operation of the computer or machine in executing the instructions. The procedure is also ideological, as computational processes operate like other rhetorical strategies, something that Ian Bogost's phrase "procedural rhetoric" makes clear in describing how computational processes (like good speeches) model persuasion in systems involving the interpretation of any symbolic system that governs thinking and action[7]—between sender and receiver, speaker and listener, writer and reader, programmer and user.

In such ways, analogies can be drawn between the temporal operations of programming and historical processes more broadly.[8] For example, the codework *Repeating History* (2009) by Pall Thayer emphasizes that historical processes are not linear but cyclic. Significantly, it is also released for further modification and comes with the warning that if the script is run, it could cause damage to the user's system.[9]

```
#!/usr/bin/perl
sub relive {$command = shift;print `$command`;}
$bash_history = $ENV{ HOME }."/.bash_history";
while(1){
  open(HISTORY, $bash_history);
  while($moment = <HISTORY>){
    relive($moment);
  }
}
```

As evident if it is run, the source code expresses both what it will do and what it can do at the same time—like history, its operations can be thought of as relays between what exists and what is possible. Stretching the analogy further, it can be argued that the programmer is not simply doing work that becomes the object of history but intervening in the very processes of history. This evokes a position informed by historical materialism, that there are social forces that intervene in the process of history, and that humans and their ideas take an active role in this process. This is what Marx was referring to in "The Eighteenth Brumaire of Louis Bonaparte" (1852) when he claimed: "Men make their own history, but they do not make it as they please; they do not make it under self-selected circumstances, but under circumstances existing already, given and transmitted from the past. The tradition of all dead generations weighs like a nightmare on the brains of the living."[10] In historical materialist thinking, this fact underpins antagonistic relations (like the class anger expressed in Harwood's codework) as the subject gains self-knowledge of his or her role in history. However, the view that human subjects make their own history, even under particular circumstances, is now considered far too deterministic (and unfortunately seems to have been disproven by history), and agency is widely recognized as far more contingent on external conditions than simply preprogrammed in human subjectivity.[11]

Does this mean we are free agents only in the sense that we do not recognize the determining factors that control us? This goes some way to describe the gap between our lived sense of reality and what precedes it, forever bound into paradoxical relations. To Hegel, this logic indicated the passage from being "in-itself" (in its essence) to "for-itself" (in actuality), later adapted by Marx to describe the passage to class consciousness.[12] Acting for-itself seems to suggest something close to the operations of speech acts and program code, which are both preprogrammed to act and simply need to be executed. The question remains how to reconcile this rather deterministic view with broader contingencies.

Emergence

Historical processes can be understood as phenomena that are analogous to the inner workings of wider systems; they express ongoing processes of development and complexity, beyond the reach of a linear narrative of progress or the straightforward accumulation of knowledge.[13] Those keen to defend Hegel from the accusation of teleology (such as Žižek)[14] would suggest that his idea of the end of history has been misunderstood and that his approach is predicated on contingency and adaptive logic. The passage from in-itself to for-itself describes a developmental process in which consciousness of conditions is derived recursively, generating a consciousness of consciousness (as mind shapes the perception of the mind, echoing one of the principles of second-order cybernetics, or "cybernetics of cybernetics").[15] What appears is not true knowledge but what appears to be known, adding another level of consciousness, and so on, in an ongoing generative process. Moreover, the subject does not simply recognize false conditions nor is driven by a concept of preprogrammed subjectivity, but there is an antagonistic interaction of the two—combining both internal and external factors, oscillating between what is possible and what actually exists, but crucially without losing the antagonism between them. Žižek refers to this as "tarrying with the negative," the retention of contradiction rather than the creation of false totalities.[16] His point is that Hegel's conception, even though it tells a story about universal freedom, still allows for new historical departures, and thus remains useful.

Taking this adaptive logic a step further, critical realism (associated with Roy Bhaskar) aims to extend Hegelian thinking and to recognize the "emergent properties of the social realm."[17] Social phenomena can be seen to contain emergent properties as part of an open system that generates new possibilities. An understanding of adaptive systems informs this view, and also undermines any teleological or linear understanding of history (associated with Hegel and aspects of Marx in particular), making it far more recursive.[18] However, the dialectical concept of negation is not rejected but redefined as something more transformative, dynamic, and contingent.[19] Drawing on second-order cybernetics, Bhaskar describes the world as an open system, or more exactly as an "open-systemic entropic totality, in which results . . . are neither autogenetically produced nor even constellationally closed, but the provisional outcome of a heterogeneous multiplicity of changing mechanisms, agencies and circumstances."[20] He conceptualizes agency similarly, in terms of incompleteness or insufficient totality that drives transformation. This is characterized by the concept "transformative agency," to stress the importance of ideas and the active role of people in historical development; and also to recognize that people do not simply make their own history nor are determined by history, but both: they both act and are acted upon. In other words, there is a feedback loop that describes the way the historical subject can reassemble itself or self-organize, in the passage from in-itself to for-itself.

Computation

What does an understanding of computational processes further contribute to this understanding of the relations between speech and code, if there is a recognition that the historical subject can both program and be programmed? In a history of computing, for instance, Charles Babbage's "Analytical Engine" (first described in 1837) extended the earlier "Difference Engine" (first conceived in 1786) by employing the principle of the "strange loop" or "tangled hierarchy," a mathematical theory describing the capability of a computer to alter its own stored program.[21] This logic is taken to an extreme in the case of code that is self-referential or recursive, such as a "quine," a term used to describe a program whose output is exactly the same as its complete

source code. The idea is summarized by Hofstadter using quine as a verb, meaning, "to write (a sentence fragment) a first time, and then to write it a second time, but with quotation marks around it." For example, if we quine "say," we get "say 'say'."[22]

The "while loop" in programming allows sections of code to be repeated until a specified condition is met. In Alex McLean's artwork *forkbomb.pl* (2002),[23] one while loop is embedded in another, a technique often used for manipulating two-dimensional data. However, this construct is twisted in on itself by the final "goto" instruction which jumps back inside the loop, so that the two while statements are notionally inside each other. Stranger still, the "fork" instructions cause the process to split into two every time a while condition is tested, so that the condition simultaneously succeeds and fails in different processes. This also means that the number of processes doubles in every loop, quickly reaching the operating system's limits and possibly causing it to crash.

```
#!/usr/bin/perl

no warnings;

my $strength = $ARGV[0] + 1;
while (not fork) {
  exit unless --$strength;
  print 0;
  twist: while (fork) {
    exit unless --$strength;
    print 1;
  }
}
goto 'twist' if --$strength;
```

The script prints out zeros in the outer and ones in the inner while loop, which are shown in figure 2.1 as white and black pixels. Because *forkbomb.pl* pushes the system to its limits, it becomes sensitive to subtleties of timing and state in the operating system. For this reason the output is different every time it is run and the output is an artistic impression of your system under strain.

The way endless loops work on themselves—as if in self-recognition of wider conditions—neatly corresponds with a dialectical understanding of the inherent antagonism between internal and external factors, oscillating between what is possible and what actually exists. That the script "bombs" the system is also evocative in terms of wider critical expression, such as the adaptation to US foreign policy in *forkwar.pl* (2003), by deprogramming.us.[24]

Clearly, computational processes execute a very particular view of history, and the operations of memory and storage are key to this. In solving a given problem, the

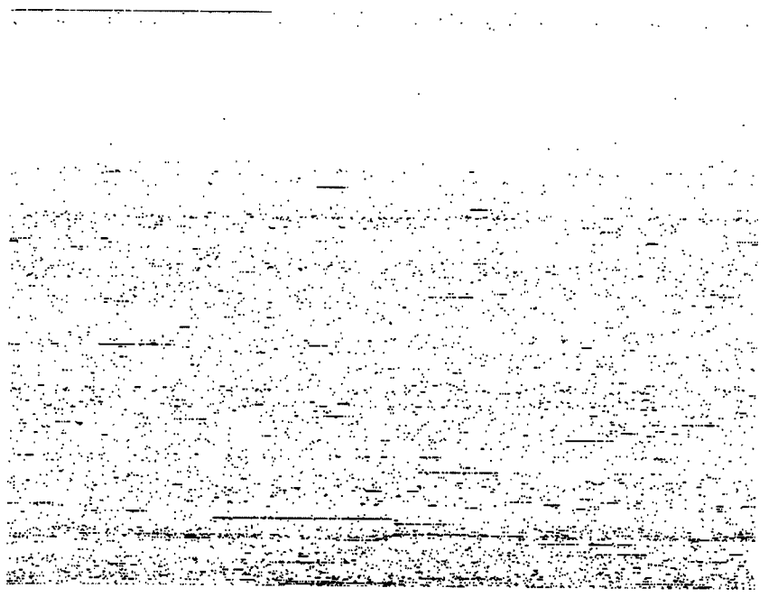

Figure 2.1
Alex McLean, output of *forkbomb.pl* (2002).

central processor takes symbols from memory, combines or compares them with other symbols, and then restores them to memory. "Memory" here is random-access memory (RAM), where programs are created, loaded, and run in temporary storage; whether these are written to hard memory becomes an intriguing analogy for the ways in which memory is loaded into history (and how this process is ideological in terms of what becomes official history).[25] Perhaps the oral history tradition seems a less distorted version of events, as it involves less-mediated forms of history like the subjective realm of memory itself. Nevertheless, however inscribed, data is selected, stored, processed, and also deleted in all systems.[26]

History is full of such examples and can be understood in similar terms of storage and deletion, turning human intelligence into artificial intelligence and producing artificial history. With the shortcomings of artificial intelligence in mind,[27] an important and recurring reference is the machine built by Wolfgang von Kempelen in 1769—a chess-playing automaton dressed in Turkish attire that wins every time it plays—which introduces Walter Benjamin's essay "On the Concept of History."[28] In the terms of the essay, the example of Kempelen's automaton, well known in Benjamin's time, demonstrates that the dynamic of history (like that of the machine) is

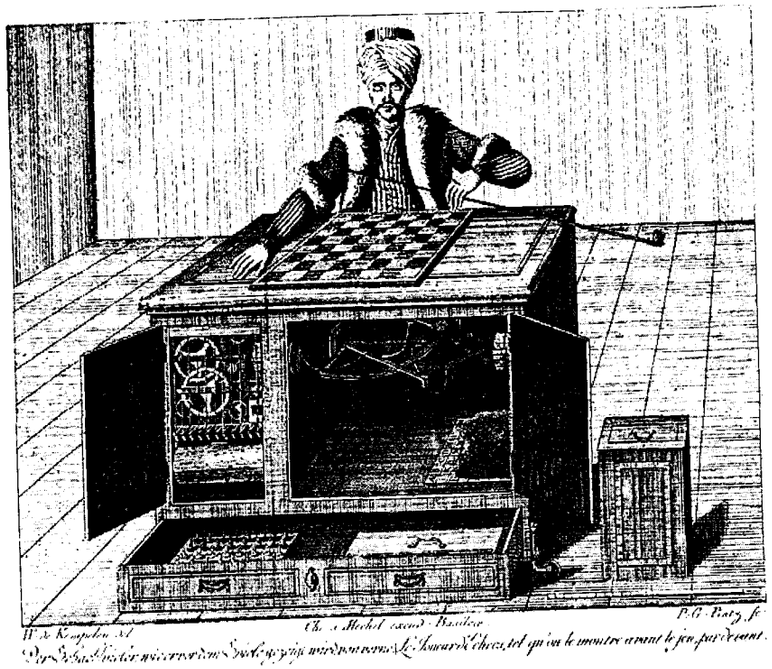

Figure 2.2
Wolfgang von Kempelen, *The Turk*, also known as the *Mechanical Turk* or *Automaton Chess Player* (1769). Copper engraving from Karl Gottlieb von Windisch's *Briefe über den Schachspieler des Hrn. von Kempelen, nebst drei Kupferstichen die diese berühmte Maschine vorstellen* (1783). Image in public domain.

fake; the task of the historical materialist is to reveal the inner workings as ideological constructions, so that they can be further modified.

Whether this machine was driven by magic or by some other (Orientalist) trickery that made it able to play chess so well was a subject of much public discussion. Despite the apparatus being "stripped naked" as part of the presentation, the accepted historical view is that a small person was hidden under the automaton's table and operated the chess pieces through the use of magnets.[29] The controversy continued when Edgar Allan Poe compared the chess automaton to Babbage's calculating machine, questioning the authenticity of machines "without any immediate human agency."[30] For Benjamin this is exactly the point, and moreover that the illusions are challenged by the expertise embedded within; the forces that guide the puppet's hand by means of strings, like its counterpart historical materialism, are such that it always wins.

Voice

To Dolar, there is a further aspect of the ghost in the machine, namely the voice.[31] He refers to another invention by Kempelen, his speech machine of 1780 (mentioned in the previous chapter), and the way the voice is tied to subjectivity and to the unconscious, whereas with the chess-playing machine the link is between subjectivity and the appearance of intelligence (or consciousness). Actually the two examples are even more connected, as, after Kempelen's death, Johan Nepomuk Maelsel added some improvements to the chess-playing machine including speech—the announcement of "échec" (check) by means of bellows—to further authenticate its apparent "intelligence." Jonathan Rée also comments on the chess-playing machine, suggesting a media-archaeological approach to the role of artificial speech in relation to speech as the source of human society and history. Rée cites examples of how the analysis of speech informed the development of other media forms, such as the early description of cinema as "speech photography"; many of the examples from the previous chapter might also support such an approach that associates speech with authenticity.[32]

The distinction between the two Kempelen machines is quite precise for Dolar, who applies the Hegelian formulation: we move from "in-itself" in the case of the speaking machine to "for-itself" in the thinking machine.[33] Speech is the hidden mechanism behind thought, like the puppet in the chess-playing automaton or the ghost in the machine. Dolar proposes that Benjamin's conclusion be modified, such that "if the puppet called historical materialism is to win, it should enlist the services of the voice."[34] In similar ways, recognition of the layers of code could also be of service in disputing surface appearances, and in this way human subjects may be in a better position to modify their circumstances if they enlist the services of technical expertise and collective labor related to code. Thus it might become possible to conceive of software operating for-itself, in recognition of the complexities of the work, action, and intellect embedded in its forms.

Coding work

How has work itself transformed and incorporated the communicative dimension, making all work more like codework? The general claim is that the living contradiction of labor relations in the factory has been extended by collectivity and connectivity to the "social factory."[35] The figure of the factory remains useful to reinforce the view that all social relationships lie in social production and relationships between people in production, but these conditions have been extended more widely into society. In orthodox Marxism, the capitalist mode of production simultaneously produces and reproduces the antagonistic social relations between labor and capital contained within the site of production. But labor is no longer simply contained by

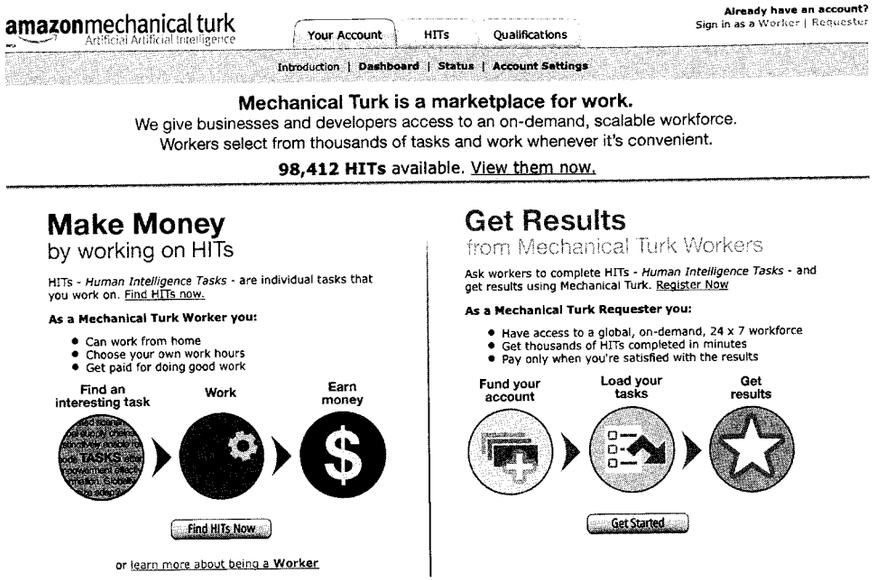

Figure 2.3
Amazon Mechanical Turk's homepage (2011). Screenshot.

the factory walls, as part of a process that Marx termed "real subsumption" (in *Grundrisse*) to conceptualize the way class exploitation is subsumed into the wider social realm.[36] This is clearly more evident under contemporary conditions than it was in the mid-nineteenth century, as the process of subsumption is now assisted by informational technologies and networked intelligence.

Amazon's Mechanical Turk is a case in point. Intriguingly named after Kempelen's automaton, it is an online marketplace for precarious work, running since 2005. It does not hide the labor of the worker but celebrates its immaterial character. Using so-called "artificial artificial intelligence,"[37] it claims to provide a marketplace for work, "to give businesses and developers access to an on-demand, scalable workforce," so that "Human Intelligence Tasks" can be executed (although some tasks, it acknowledges, are performed better by humans than machines). At the same time, and for obvious reasons, labor relations are downplayed through carefully chosen language, describing "requesters" rather than employers, perhaps to avoid questions of labor value and subsequent antagonisms. Even more depressing is that so-called "Turkers" in the US use this "crowdsourcing marketplace" as a form of entertainment. As Trebor Scholz puts it, "The biggest trick that Mechanical Turk ever pulled off was to make

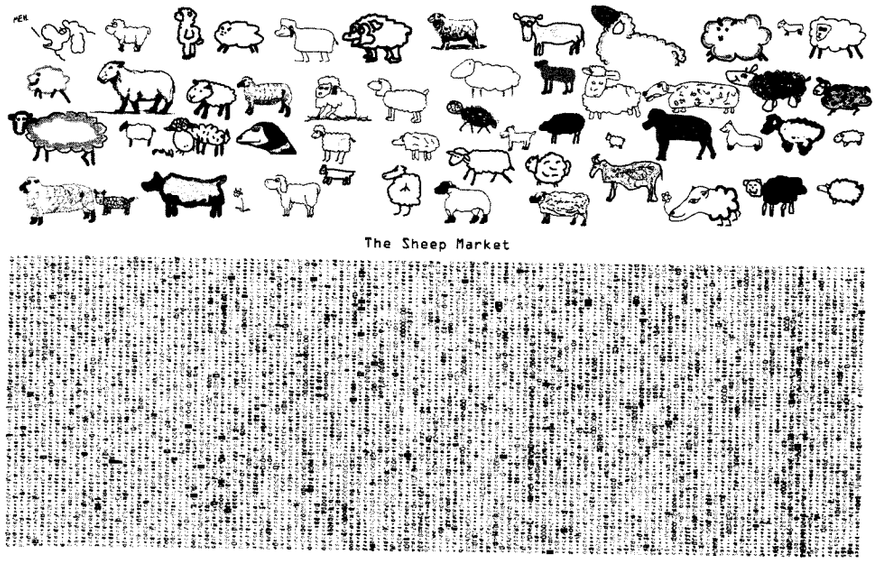

Figure 2.4
Aaron Koblin, *The Sheep Market* (2006). Image courtesy of Aaron Koblin.

workers believe that what they do is not really work. That, at least, is somewhat suggested by the slogan on MTurk's coffee mugs: 'Why work if you can turk?'"[38]

Artists have also enthusiastically responded with experiments in crowdsourcing, such as Aaron Koblin's *The Sheep Market* (2006), a collection of 10,000 sheep made by workers using Mechanical Turk, each paid US$0.02 to "draw a sheep facing to the left."[39] Whether effective irony or not, such examples reflect how labor is becoming ever more informational and communicative, leading to a situation in which all activities seem to have been turned into production.[40] *The Sheep Market* is also a good example of how capital tries to capture the creative and communicative capacity of the socialized labor force and turn it into information that can be marketized.

Consequently, the site of antagonisms has been extended to include the control of communications and the labor related to communications; the struggle is no longer simply between workers and capitalists but also between communities and platforms (which the following chapter will discuss in more detail through an engagement with more public forms). Class antagonism is transformed into a broader dynamic, as a consequence of the collapse of the management of production and the action of production itself, between conception and execution, exploding the site of conflict to society as a whole. Following this line of thought, neither labor time nor wage is

considered to be the central issue under critique but technological skill and organizational forms defined by cultural, informational factors and knowledge. Productive labor is upgraded by intellectual, immaterial, and communicative labor, disrupting orthodox conceptions of labor, value, and agency.[41] Work is reconnected to and extracted from the body, mind, and soul.

Valorization

In *The Soul at Work*, Berardi examines these contemporary forms that put the soul to work, taking his cue from Spinoza (taking the soul, in other words, in a materialist sense, not related to religion as such).[42] Berardi insists on a shift of attention to the way contemporary modes of production convert mind, language, and creativity (together taken to be attributes of the soul) into value. Therefore these combined territories of the intellect, language, and imagination become core concerns for understanding contemporary forms of alienation, as the soul is no longer ignored but directly engaged, with alienation now extended to the whole of life. To understand alienation, he charts its passage from the Hegelian-Marxist tradition—where it is measured between human essence and the perversion of this into work activity, the split between life and labor (close to Arendt's distinctions to be taken up later in the chapter)—to the Italian workerist tradition where it is defined in relation to labor time and the creation of value, "the reification of body and soul," as Berardi puts it.[43]

The significance of the break represented by workerism (and later Autonomia) is the rejection of the passivity of the worker and insistence on the possibilities of active "refusal of work," as a strategy of exit from the capitalist valorization process.[44] This tactic is based on the paradox that Mario Tronti's essay "The Strategy of Refusal" identified in 1965: that the logic of capital "seeks to use the worker's antagonistic will-to-struggle as a motor for its own development."[45] The paradox is that capital does not wish to destroy critique but tame it through processes of subsumption. Berardi's genealogy of alienation demonstrates how antagonism in the refusal of work has moved to a situation where work increasingly seems to define subjectivities, all aspects and every detail of life, the very essence of the living thing (and this is what constitutes the soul according to Aristotle).[46] In integrating intellect, language, and imagination, labor power produces new and more totalizing kinds of subjectivities. What Berardi refers to as the soul—and for the argument here, the voice—are now also deeply implicated in these processes.

Networked communication technologies play a significant role in these processes of subjectification as they relate to general intellect, forming networked intelligence.[47] With this concept (an elaboration of subsumption), already Marx had predicted that the productive forces of the intellect, of human knowledge and skills, would be incorporated into capital itself. The crucial issue, both then and now, is that general intellect

unleashes contradictions by combining technical knowledge and the social cooperation of bodies. The concept is somewhat materialized in the procedures of Amazon's Mechanical Turk and more generally in coding cultures, where connectivity is reflected in the open-source and free software movements as an example of how technical expertise and socialized labor can be shared and recombined effectively. Such ways of working with software are arguably more robust and less bug-ridden as a result of collective development, but the contradictions are also evident, as this approach becomes the orthodoxy in the release of proprietary software development too. Indeed free and open software principles both contest and affirm capital investment on human and economic levels, as labor expands to involve cultural activities not traditionally considered to constitute work, including intellectual labor and artistic practices.[48] Indeed, the once straightforward distinction between waged and unwaged work, or work and nonwork (play), becomes ever harder to establish.

If the complexities of contemporary labor are exemplified by the ways in which waged and free labor have become harder to differentiate, much is also now practiced outside of traditional production processes, on computers and across telecommunication networks. The former distinction between leisure time and work time is now further confused by the ways physical labor, intellectual labor, and machine labor have collapsed in on themselves. Consequently, a traditional view of labor value appears inadequate because of the difficulty in calculating working time related to signification, as opposed to the relative ease of calculating working time in making traditional material goods. To Tiziana Terranova, the complexities of contemporary labor value are characterized by free labor within capitalism, rather than by gift economies and systems of exchange in societies outside its reach.[49] Therefore the production of free and open-source software cannot be considered to be an alternative to capitalism but an expression of new forms of labor within capitalism, "part of the process of economic experimentation with the creation of monetary value out of knowledge/culture/affect."[50]

What is considered "free," whether labor, software, or speech, is clearly based upon market infrastructures that use intellectual property rights to further vested interests. Free speech operates with similar paradoxes (discussed in more detail in the following chapter), since ideas are no longer free once made tangible. This is a by-product of neoliberalism that tries to capture all forms of labor, including intellectual labor and protest. But although capital tries to treat ideas as it would any other goods, it does not succeed in commodifying them altogether, because intellectual work is not simply reducible to market principles. This is a point that Maurizio Lazzarato makes in relation to books, drawing on the work of the sociologist Gabriel Tarde to explain that the market can only determine a book's exchange value as a product, not its value as knowledge.[51] In a similar way, the publishing of books can be understood as an attempt to commodify the oral tradition of storytelling.

Property

Similar ambiguities apply to artistic works in general, and especially to those that follow a conceptual tradition or are performed live or involve speech, as the legal apparatus struggles to adapt to informational logic and remains locked into frame-works that tend to consider tangible objects rather than intangible ideas. So too with codework, and perhaps especially so when it is in the public domain.

An example that played on these ambiguities was Robert Luxembourg's *The Conceptual Crisis of Private Property as a Crisis in Practice* (2003), which consisted of program code (crisis.php), an explanatory text file (crisis.txt), and a screenshot (crisis.png).[52] If the program was run, it parsed the screenshot into the full text of the novel *Cryptonomicon* by Neal Stephenson (of 1999). The project operated as a conceptual puzzle that contained a number of interconnected parts all found in the screenshot itself. These included a still from the film *The Matrix*,[53] and a passage from Hardt and Negri's *Empire* (published in 2000) that questioned the limits of the legal apparatus that underpins contemporary power structures[54] (and this passage from Hardt and Negri also supplies the phrase that Robert Luxembourg uses for his work's title). The project thus formed a neat conceptual loop about encryption and intellectual property rights, allowing readers to gain access to the Stephenson book and at the same time for the author of the software to notionally circumvent the legal constraints imposed by the publishers. The conceit was that the script does not infringe copyright in itself, but only if executed by the user.[55]

Apart from being a novel about encryption, it is also interesting to note that *Cryptonomicon* demonstrates the interactions of code and written text languages, as Stephenson is both a print author and a computer programmer. It follows that it can be read as both written text and a program to be executed in its reading, evoking his nonfiction work *In the Beginning Was the Command Line* that charts his migration from Macintosh to UNIX-based operating systems (written 1995–1998, simultaneous to *Cryptonomicon*). The fact that the GNU General Public License agreement emerged from a UNIX culture of sharing also sets the release of the crisis.php script in sharp contrast to the constrained conditions of distribution of the book it de-encypts.

The various licenses on offer to software producers reflect the ways that code as an intangible object is being forced into a property regime by those who wish to consider it to be like any other material goods that can be traded in the open market. The perversity of the logic is the basis for Stallman's argument in "Why Software Should Not Have Owners," and his position that to think of software in these terms and to adopt their legal protections is anachronistic.[56] The legal apparatus remains poorly suited to dealing with the results of immaterial labor, as it still tends to view property in terms of end products and is unable to deal with other modes of production, such as commons-based production (more on this issue will be developed in the following chapter). Open license agreements, such as those provided by the Creative Commons

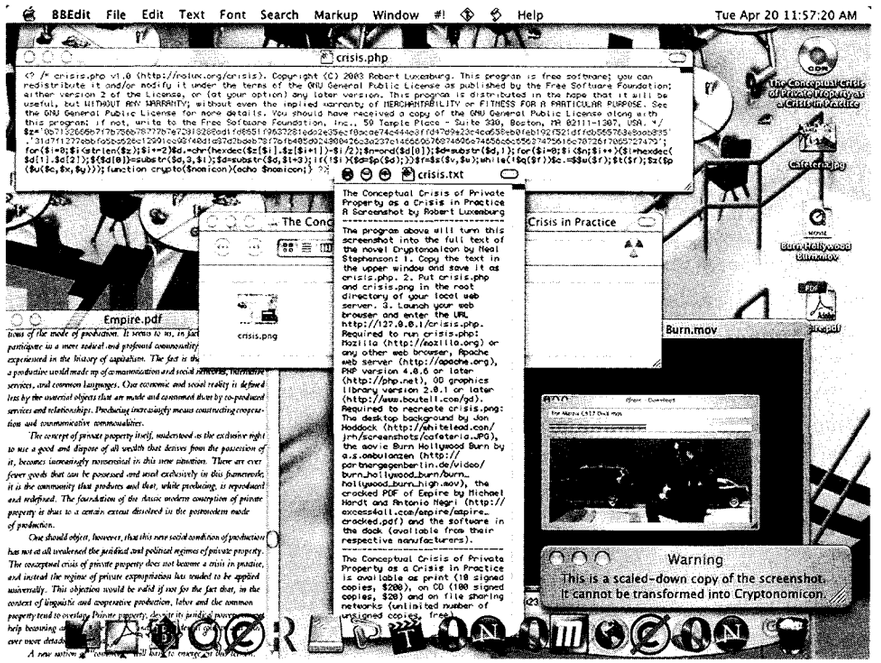

Figure 2.5
Robert Luxembourg, *The Conceptual Crisis of Private Property as a Crisis in Practice* (2003). Screenshot.

initiative, may be useful in providing flexible copyright licenses for creative works, but they do not adequately address the problem at the site of production.[57] Indeed, as many commentators have pointed out, they arguably represent the potential for the further commodification of the creative work invested in the first place.[58] Even the more radical Creative Commons "Public Domain Dedication, No Copyright" license speaks the language of law: "The person who associated a work with this deed has *dedicated* the work to the public domain by waiving all of his or her rights to the work worldwide under copyright law, including all related and neighboring rights, to the extent allowed by law. You can copy, modify, distribute and perform the work, even for commercial purposes, all without asking permission."[59]

If the legal apparatus, ironically itself a performative language, sets out to turn information into commodities, then alternatives are required that seek to free ideas from their complicity with these constraints. An extreme position is to reject copyright altogether, following the logic that any form of property rights over the

commons is unethical. This is the position taken by Piratbyrån (the bureau of piracy), the founders of the peer-to-peer bittorrent site Pirate Bay, when they reject copyright with the slogan "No copyright. No license,"[60] or by using the imperative "Copy Me!"[61] The more pragmatic and alternative critique provided by copyleft licenses (GPL being the earliest example, in 1989) maintains the right to copy and share enshrined in the legal apparatus; although such licenses may go further than Creative Commons, they still serve to reinforce the system of copyright, even through its apparent negation.

In the "copyfarleft" license, Dmytri Kleiner takes copyleft further by addressing some of the problems associated with the expropriation of free labor, in arguing for licenses combined with class consciousness. He states: "While copyleft is very effective in creating a commons of software, to achieve a commons of cultural works requires copyfarleft, a form of free licensing that denies free access to organizations that hold their own assets outside the commons."[62] His position, in keeping with the title of his book *The Telekommunist Manifesto,* is that the rejection of property regimes is only possible once a classless society has been achieved; for the system of property itself needs to be further negated if one follows the Hegelian notion of "negation of negation" (which would regard an initial negation of copyleft as still operating with the symbolic confines of property, and hence its further negation as necessary to account for external conditions).[63]

One of the political challenges is to apply copyleft thinking more generally to tangible goods, not the other way around, as the legal apparatus attempts. Kleiner develops analogies between technical and social systems of organization in a section of his book called "Peer-to-Peer Communism vs The Client-Server Capitalist State," where distributed forms like the commons are contrasted with centralized statist forms.[64] He identifies the contradiction at the core of free culture with its connection to the term "free market," which is clearly not free but a description of the market economy that follows "unfree" principles. The concept of economic rent is used to indicate the income that owners earn simply from the act of ownership itself,[65] like the earnings derived by "landlords" letting private property (the name comes from the enclosure of common land by lords). Both capitalist and landlord expropriate in this way. What is developed in terms of this critique is that property remains a core issue, enshrined in the legal apparatus to protect class divisions and to legitimize rent.

If the law is a speech act, then perhaps the whole legal system needs to be negated through language, destabilizing its codes. Virno's nondialectical understanding of negation refers to the inherent paradoxes of language as it both does negation (by identifying what something is not) and is negation (inasmuch as it can only signify something).[66] More will be said of this in the last chapter, but for now it suggests possibilities for new "nonrepresentational" forms of political organization that are based on speech acts that adhere to distributed forms and common ownership.

Self-organization

The political point is that ownership should apply to labor, not property. "Venture communism" is the ironic term Kleiner uses to evoke workers' self-organization, to address the way that class conflict is conceived across telecommunications networks through the pervasive use of Internet and mobile technologies. This becomes the foundation for a critique of the rise of social media in particular as a project of venture capitalism that expropriates developments in the free software movement and commons-based peer-to-peer technologies.[67] The point remains that ownership and property are core issues in these platforms; they are organized in ways that follow the logic of rent, to secure profit from immaterial assets. The owners profit through centralized control, whereas the distinctiveness of peer production lies in the relative independence of workers to control common productive assets and share the benefits. The idea of venture communism registers the need to develop an alternative business model that supports these principles, and at the same time rejects the way that value has been stolen from the commons in the first place.

An example of software that attempts to apply this logic is *Thimbl* (2010–2011), a free, open-source, distributed microblogging platform developed by the Telekommunisten collective (of which Kleiner is a founding member). The publicity on the website confirms: "The most significant challenge the open web will need to overcome is not technical, it is political."[68] *Thimbl* uses a common server software called Finger, adapted to the principles of the open web, and positions it in relation to the legacy of peer-to-peer organizational forms such as UseNet (the distributed messaging system, operating since 1979 although since subsumed by the Google monopoly).[69] This reflects one of Kleiner's main arguments, that social networking with its participatory ethic has been largely stolen from free software development, a theft interpreted as "capitalism's pre-emptive attack against peer-to-peer systems."[70]

The history of *Thimbl*'s technical development is important of course, and political in terms of its negation of dominant infrastructure: unlike a platform like Twitter, where the service is provided through a client-server architecture, with *Thimbl* clients are encouraged to become active in their co-ownership of the service with other "workers." Yet this is largely a symbolic gesture, not a viable alternative to Twitter as such but a performative artwork, or "performative science fiction" as Kleiner puts it.[71] He also clarifies that the speculation is not based on technical viability, as it does work, but more on economic viability, exemplifying the concept of venture communism: "Thimbl is an economic fiction."[72] Based on an understanding of the political economy of social media, its intention is to uncover uneven social relations embedded in communication technology, creating a platform that works in unexpected ways, like other projects of Telekommunisten's Miscommunication Technologies series.[73]

In this way, the ability to use technology differently can still be seen to rest with those who have the expertise to operate it; not the platform owners or landlords, but

Figure 2.6
Telekommunisten, *Thimbl* (2010–2011). Screenshot of homepage. Image courtesy of Telekommunisten.

the software developers, systems operators, programmers, technicians, office workers, designers, artists, and so on. Given this emphasis, it is work itself that needs to be transformed and made more autonomous. It follows that new collective characterizations are required that respond to both social cooperation and technical knowledge, and the threat posed by market forces in striving to capture networked intelligence.

The unstable relations within sociotechnical systems between human and machine labor indicate some of the possibilities for rethinking the social relations of production in working with code and being worked on by code. Another precedent for this line of argument lies in Félix Guattari's comment in *Chaosophy* that Marx was mistaken in thinking social relations lie outside of the tool and the machine.[74] The distinction between tool and machine is much misunderstood, and often further confused in connection with software and hardware. To clarify, Marx stressed that machines are a factor of communication, whereas tools merely extend control through direct contact: "the tool is a simple machine and the machine is a complex tool."[75] The machine becomes more and more independent of the worker, extending the limits of human

effort, and becomes part of a wider scheme of machines working together collectively: "Here we have, in place of the isolated machine, a mechanical monster whose body fills whole factories, and whose demonic power, at first hidden by the slow and measured motions of its gigantic members, finally bursts forth in the fast and feverish whirl of its countless working organs."[76]

The worker too becomes ever more like a machine, prefigured in the development of automata and "second-order robots" programming their sequences of work. But the complex relations of worker and machine, prefigured in the rhetorical descriptions of factories as giant automata or workers as robots,[77] requires further elaboration, as worker and tool are part of ever more expansive machinic assemblages.[78] To explain: the writing, compiling, and running code are part of software that, along with hardware and with the labor invested in both, makes the computer-machine assemblage as a whole. So rather than the dead labor of machines replacing human labor, or other simplifications, Guattari states: "On the contrary, I think that machines must be used—and all kinds of machines, whether concrete or abstract, technical, scientific or artistic. Machines do more than revolutionize the world, they completely recreate it."[79]

The argument of this section has been to consider work in these machinic terms, to look beyond the emancipatory rhetoric of free software to a more detailed understanding of labor and agency across a plurality of social movements, worker self-organization, and coding cultures, and the ability to write, compile, and run code in unexpected ways. *Ungovernable.patch* details a small change to the Linux kernel, in the standard "patch" format that programmers use to share modifications to software. The change alters the working behavior of the "governor," the operating system component that changes the speed of the computer processor. Instead of making the computer operate faster when there is more work to do, it works faster when there is *less* to do. The human is forced to work in cooperation with the machine, finding that the computer quickly tires if too much is asked of it but works very well in response to less demanding tasks.

```
# ungovernable.patch (c) 2011, released under the GPL version 2
--- vanilla/drivers/cpufreq/cpufreq_ondemand.c
+++ new/drivers/cpufreq/cpufreq_ondemand.c
@@ -494,7 +494,7 @@
    }

    /* Check for frequency increase */
-   if (max_load_freq > dbs_tuners_ins.up_threshold * policy->cur) {
+   if (max_load_freq <= dbs_tuners_ins.up_threshold * policy->cur) {
      /* If switching to max speed, apply sampling_down_factor */
      if (policy->cur < policy->max)
          this_dbs_info->rate_mult =
@@ -513,7 +513,7 @@
```

```
    * can support the current CPU usage without triggering the up
    * policy. To be safe, we focus 10 points under the threshold.
    */
-   if (max_load_freq <
+   if (max_load_freq >=
      (dbs_tuners_ins.up_threshold - dbs_tuners_ins.down_differential) *
       policy->cur) {
        unsigned int freq_next;
```

Code action

If the concept of speech acts is one way of characterizing the breakdown of distinctions across the different articulations of code working, then more detail is required on the matter of execution (as the above example demonstrates to some degree). This last section of the chapter elaborates on the relationship of work to action, in order to respond to the changed relationship between conception and execution of work, enacted materially by instructions.

It begins from Arendt's essay of 1964, "Labor, Work, Action,"[80] in which she addresses the distinctions between these three terms and what she considered to be a problem in orthodox Marxist thinking, that labor was tied too firmly to work at the expense of action. Arendt's understanding of action is a critique of the Platonic separation of knowing and doing, in which knowledge is identified with command and action with execution. These ideas are evident in the process of fabrication, "first perceiving the image or shape (*eidos*) of the produce-to-be, and then organizing the means and starting the execution."[81] According to Arendt, the mistake involves substituting making for action (persistent in political theory in general, in the centrality of "production"), and this leads to a line of thought in which any means appear justifiable to pursue a recognized end. This is the instrumental logic of "you can't make an omelette without breaking eggs," as she puts it, in which ethical considerations are suspended.[82] In contrast, she insists that human action lies in the realm of uncertainty, as something that cannot be fully known and as part of an ongoing process of setting something in motion without an end in mind. Ideas are not simply executed but continually interpreted in real time.

This line of thought holds a number of implications for coding practices, and certainly the description of continual interpretation is consistent with computer processing, but first the terms "work" and "labor" require more definition if we are to understand making activities more generally.[83] According to Arendt, a basic distinction can be made between *homo faber*, who makes and works upon something or fabricates it, and *animal laborans*, which produces labor at the level of life itself (and provides salvation for the soul).[84] In *animal laborans*, the human body is given over to labor, evident in the reproductive processes in their broadest sense (the pains of birth are

considered to be labor, for instance). Whereas labor combines production and consumption like life itself (and speech of course), the fabricated thing (as a result of work) is an end product, derived from a production process entirely separate from its possible uses (such as a book or software). These are categories of means and ends, and the significance is that certain "works" do not fit into the deterministic "means-end" chain of events.

Virtuosity

To Arendt, the work of art is both the most enduring and most useless fabricated object that human hands can produce: "the proper intercourse with a work of art is certainly not 'using' it; on the contrary, it must be removed carefully from the whole context of ordinary use objects to attain its proper place in the world."[85] This is a general observation, of course, and certainly much software production sits uneasily among her distinctions, but her main point is that the performative arts are particularly resistant to reification as the least materialistic of the arts.[86] Poetry too (especially if spoken) demonstrates procedures that are close to thought processes that have neither an end nor specific aims as such.

In addition, in Arendt's view a positive attribute of thinking is that it is entirely useless, "as useless, indeed, as the works of art it inspires,"[87] for thinking inspires the highest of human achievements such as poetry or conceptual art or other intellectual pursuits. The example of poetry (think of Schwitters's *Ursonate* or other useless examples of software art), in combining speech and action, demonstrates separation from the drudgery of instrumentalized human existence and a world driven by algorithms and instructions with predetermined outcomes. Furthermore, if more complex formulations of work are applied and the worker and tool combine as machinic assemblages, this comes close to Arendt's description of the tools becoming part of the laboring process in tune with the body, or replicating the body's movements and rhythms (her example being the deployment of labor-saving gadgets in the kitchen).[88] These machines draw labor and action ever closer together, not simply as labor-saving devices but as ways to recreate conditions. And yet since the time of her writing instrumentalism appears to have taken hold, and machines have become far too useful.

Reworking Arendt's distinctions in 2004, Virno thinks the separations of labor, action, and intellect have further dissolved. Whereas Arendt argued that politics imitated labor (as good action), Virno believes the opposite, that labor imitates politics—or indeed, that poiesis has taken on the appearance of praxis, as he puts it.[89] Since labor increasingly takes on the forms of action or indeed has served to depoliticize action, this explains the current "crisis of politics, the sense of scorn surrounding political praxis today, the disrepute into which action has fallen."[90] The problem lies in what Arendt already identified as the separations of making and action and knowing,

and that one should not determine the other in a procedural means-end chain of events. Like Arendt, Virno cites Aristotle's famous passage from the *Nicomachean Ethics*: "For while making has an end other than itself, action cannot; for good action [understood both as ethical conduct and as political action] itself is its end."[91]

Reworking the same quote from Aristotle, Giorgio Agamben extends the understanding of action to gesture (referring to the distinction the Roman scholar Varro makes between *facere* and *agere*): "For production [poiesis] has an end other than itself, but action [praxis] does not; good action is itself an end."[92] The claim is that gesture not only disrupts the false distinction between means and ends but also occupies mediality itself, and in this way opens up the ethical dimension: "Politics is the sphere neither of an end in itself nor of means subordinated to an end; rather, it is the sphere of a pure mediality without end intended as the field of human action and of human thought."[93]

The ethical dimension is addressed by Virno through the idea of "virtuosity," a concept also derived from Aristotle who identifies two kinds of virtue, one intellectual and one ethical.[94] Ethical virtue for Aristotle is more than simply an intellectual decision on how to best act in a given situation, but also relies on ingrained habits of action negotiated through excess and deficiency, making virtue a disposition concerned with choice. Interestingly, Aristotle regards virtue as similar to musical skill, improved with practice. One who plays a musical instrument, not just playing something but playing it well, is like a virtuous human agent not just acting in the world but doing this well too. Both actions are learned by acting, as Aristotle explains: "The causes and means whereby every virtue is cultivated or destroyed are the same, just as in the case of all the arts. . . . If this were not the case there would be no need for teachers and everyone would be born good or bad. It is just like this with the virtues."[95]

To Arendt, similarly, certain activities demonstrate excellence in the public realm, constituted by the presence of peers and the formality that is required for a performance.[96] As an audience is required for the virtuoso performance, this also emphasizes the political sphere. Both politics and performances require a "publicly organized space," as does labor to be productive, as it involves communicative action.[97] Virno continues: "It is enough to say, for now, that contemporary production becomes 'virtuosic' (and thus political) precisely because it includes within itself linguistic experience as such."[98] Extending Arendt's definition of the public defined through speech, he links this to virtuosity, as a phenomenon that does not produce an end product independent of the act of speech itself, and that operates in public. The pressing issue for Virno is to ask: "What is the *score* which the virtuosos-workers perform? What is the script of their linguistic-communicative *performances*?"[99] The suggestion is that general intellect provides the "know-how on which social productivity relies,"[100] itself virtuosic in the sense that it combines technical skill, networked machines, and collective human intelligence. However, this should not have specific aims or an end in

sight, as Arendt insisted too, but rather "virtuosity without a script, or rather, based on the premise of a script that coincides with pure and simple *dynamis*, with pure and simple potential."[101]

Performativity

Herein lies one of the challenges for those making program scripts that underscore various dynamic procedures of machines. As already established, a computer program undermines the distinction between its function as a score and the performance of the score, through its speechlike qualities. This is exemplified in the practice of live coding, where programmers make music in keeping with the expressive qualities of live performance, by coding in real time (using a command line interface, for example). To explain: "Live coding is where computer programming becomes performance art; improvising a performance using a programming language. This is made possible by the dynamic programming languages at the heart of all live coding environments. Sections of code are interpreted and added to a running program, so that a musician may enact thematic changes without a break in the sound. Thus the usual design-implement-test development cycle is discarded in favour of immediate actions and reactions more suited to creative exploration."[102]

Feedback.pl is a text editor written by Alex McLean with an unusual form of feedback. The programmer edits the code, which runs live while it is modified. However, the running code is also able to edit its own source code, so that the code is able to make fundamental self-modifications. Below is a simple example of some code modifying a comment, each time doubling up the letter "e" in the word "Feedback." That these self-modifications happen directly to the code being edited in real time puts the code visibly on the same level as the programmer. In a sense, the code embeds both action in-itself and action for-itself. Self-modifying code blatantly breaks the determinism of code and makes it explicitly performative. It demonstrates how programs operate together with the programmer, both relaying instructions and acting upon them in an uncertain relation.[103]

```
sub bang {
my $self = shift;

# Feedback
$self->code->[3]=~ s/e/ee/;
$self->modified;
}
sub bang {
my $self = shift;

# Feeeeeeedback
$self->code->[3]=~ s/e/ee/;
```

```
$self->modified;
}
sub bang {
my $self = shift;
# Feeeeeeeeeeeeeeeeeeeeeeedback
$self->code->[3] =~ s/e/ee/;
$self->modified;
}
```

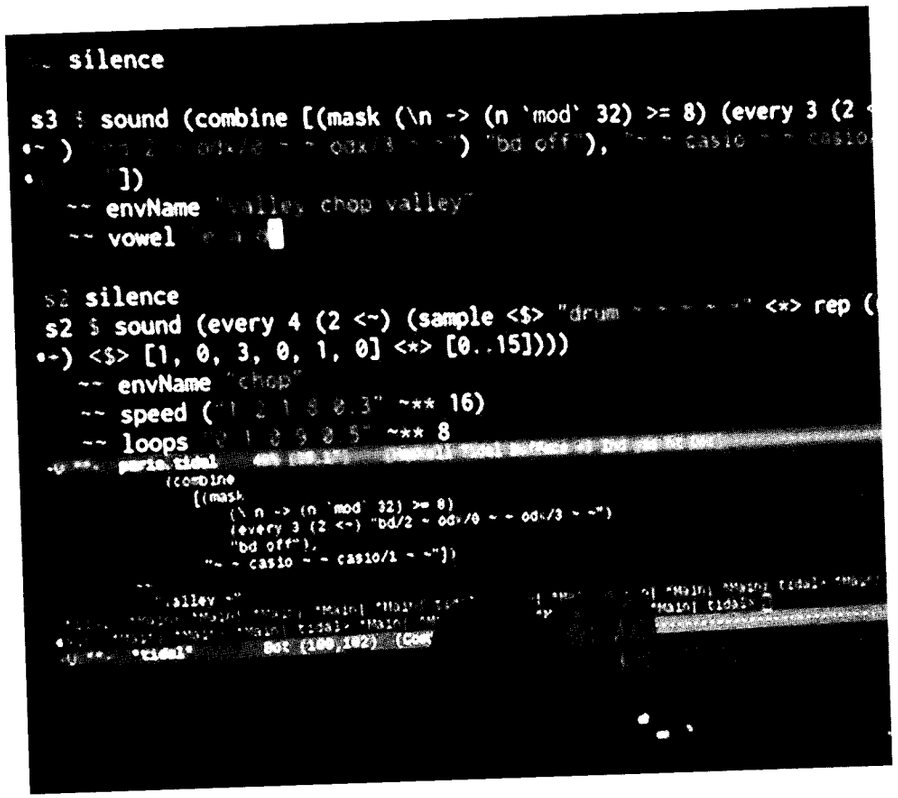

Figure 2.7
Live-coding performance by slub, at Maison Rouge, Paris, 30 August 2011. Image courtesy of slub.

It follows that many of the attributes associated with virtuosity could also be applied to the work of programmers, who demonstrate their technical and cultural agility through performances in public spaces. In this sense, the performativity of code breaks out of the means-end chain of following instructions to a predetermined end. The intervention of the programmer allows for an even more indeterminate approach and openness to other transformative possibilities, such as the possible and often unpredictable actions that result when a program runs, including the return of errors. The program performs the music with the programmer and vice versa, both relaying instructions and acting upon them.

In "Real DJs Code Live," McLean explains his approach to practice, emphasizing the active role of the programmer: "By describing a musical idea in code, we're describing it at a higher level than if we're entering notes into a sequencer. . . . I've tried sequencers and found it a slow, difficult, maddening way of doing music. There's an atmosphere of musicians being subservient to software. It really limits the kind of music that can be made. . . . Live coding places the human right back in the creative process so you can't really call it 'computer-generated' any more."[104]

The last part of the quote emphasizes the machinic and performative qualities and undermines the usual relations between coder and code for something far more dynamic. Similarly stressing the "performativity of code," Adrian Mackenzie clarifies that, "in terms of the contestations of agency associated with software, the primacy of coding can be seen as asserting the identity of programmers as the originators of software."[105] The agency of code workers may be often overlooked, but so too is a fuller description of the multiple processes and agents at work. In this way, the practice of live coding allows for an expanded sense in which plural aspects of coding work—writing, compiling, and running code—come to represent software as a whole.

Once software is defined as not only the program code but also the other materials required for the program to run, programmers become an integral part of the action. The act of coding becomes a prototype for action in broader terms, which includes a critique of the commercial imperative of software development and also the normative social relations associated with this. Like source code, these relations themselves become open to further modification, as is the case with the development of the Linux operating system. Mackenzie cites the Linux kernel (the core component of the GNU/Linux operating system), which he understands to have a particularly unstable relation to commodified software and hardware as the most pervasive example of free/open-source software development, by virtue of the file and process of the UNIX-like operating system and the enforcement of its GNU General Public License. He identifies the collaborative laboring process that underpins its efficiency: "The way in which the Linux kernel is produced and continually changed cannot be separated from its structure as a coding project. The performance of Linux as a contemporary operating system

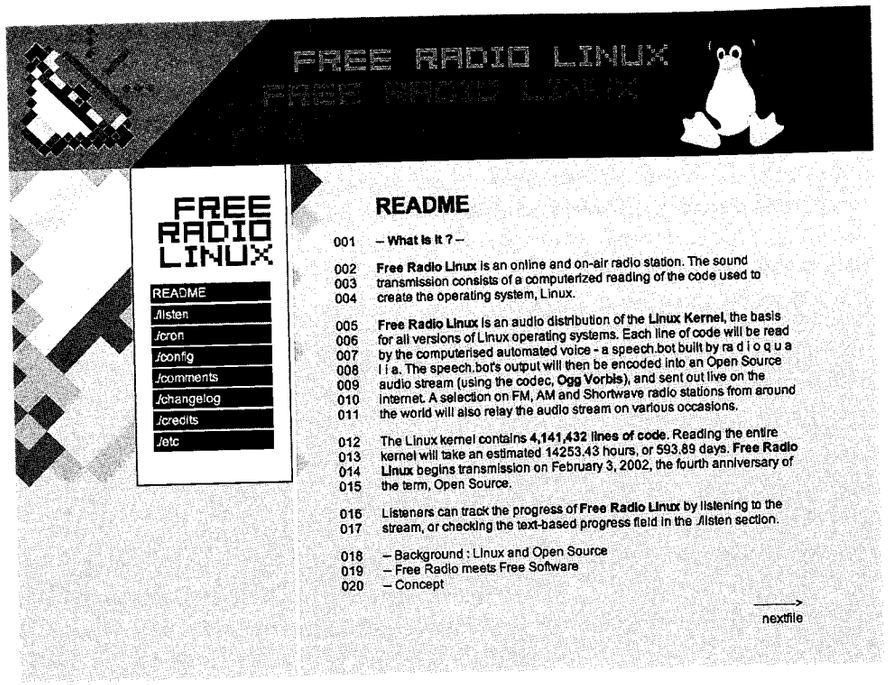

Figure 2.8
radioqualia, *Free Radio Linux* (2001). Screenshot of homepage. Image courtesy of radioqualia.

cannot be detached from the circulation of Linux kernel code through code repositories and software distributions."[106] Its development is enacted through an ongoing collective and collaborative working process that Mackenzie considers to be a speech act. He refers to radioqualia's *Free Radio Linux* (2001), in which the source code of the Linux kernel was webcast over the Internet using a speech synthesizer to convert the 4,141,432 lines of code into speech (taking an estimated 593.89 days to read).[107]

The performance encapsulates the way that coding works in its fuller sense, as already described in relation to live-coding performances, and underpins the tensions around contemporary labor and coding practices. It produces an uncertain relation between the code object and code subject—the program and the programmers—and thus challenges property relations bound to the development and distribution of code. Like live coding, it is an example of machinic expression at the level of embodied communication or speech.

Referring both to live coding and to Virno's scores of virtuosos-workers, Simon Yuill further contextualizes coding practices through the example of the Scratch Orchestra

of the late 1960s.[108] The historical parallel to improvisation serves to demonstrate how innovative forms of notation and performance techniques seek to free themselves from aesthetic and social conventions that are devised in relation to wider modes of production. Live coding can be seen to reflect present conditions, in which our lives seem to be increasingly determined by various scores and scripts but the possibility exists for more expansive conceptions of collective action, or more positive implementation of general intellect, and as indeterminate as one of the events itself with all its flaws and uncertainties of live performance.

Recomposition

If the economy is understood more and more in terms of scores and scripts, it is because new forms of labor are performative and linguistic. To demonstrate how work is now bound to speaking, Virno quotes Austin's *How to Do Things with Words*: "In the assertion 'I speak,' I *do* something by *saying* these words; moreover, I declare what it is that I do while I do it."[109] There is a biopolitical dimension to this, in the human capacity for speech and to make relations with others, based in what Virno calls "the faculty to speak; not work actually done but the generic capacity to produce."[110] Underlying this is the distinction that Arendt makes between the production process of work and labor that is more closely tied to life itself. The concept of general intellect is important to this too in emphasizing linguistic communication and social cooperation, enhanced by networks of machines and workers.

Christian Marazzi develops these ideas in *Capital and Language*, citing Austin as well to cast the powerful conventions of financial markets in terms of performative utterances, like "saying something makes that something true."[111] Marazzi cites Searle who refers to printed currency that is not simply a description of a fact but creates the fact; the "fact-that-one-speaks" produces the fact merely by the fact that it has been said. The recognition of the reliance of financial markets on collective speech acts underpins the viral operations of what Berardi calls "semiocapitalism" (he uses this term to signal the fusion of mind, language, and creativity).[112] This also establishes how the immaterial forces of language and money act on the soul in real terms, not at the level of metaphor, producing the fact of the present sickness of the economy.

According to Berardi, the soul has been broken and subjectivity made effectively soulless, as both desire and the soul have been colonized by capital and cast into depressed states.[113] He calls the situation in which the soul is put to work the "factory of unhappiness,"[114] and the control of happiness has become key to the health of the economy. The point is that unhappiness is encouraged to bolster productivity (as with so-called shopping therapy) but is carefully managed, for "the masters of the world do not want humanity to be happy, because a happy humanity would not let itself

be caught up in productivity. . . . However, they try out useful techniques to make unhappiness moderate and tolerable, for postponing or preventing a suicidal explosion, for inducing consumption."[115]

Yet in the factories of unhappiness, things are beginning to spiral out of control, for example with the emergence of Facebook suicide groups such as the Facebook Mass Suicide Club or Hong Kong Facebook "suicide" group (sharing suicide methods and urging members to kill themselves on the same day in 2009).[116] Indeed, Berardi claims that "suicide is the decisive political act of our times," partly typified by 9/11 but more so by the example of the Finnish youngster Pekka Auvinen, who in 2007 turned up at school and shot eight people including himself, wearing a T-shirt with the sentence "Humanity is overrated."[117] What need to be rediscovered are forms of happiness tied to collective formations, such as the commons, and not to proprietary platforms like Facebook or Twitter (and more will be said on this in the following chapter, with examples of virtual suicides from these platforms). Alienation is accelerated as communication breaks down and subjects are left to babble, as in the case of *prozac.pl* cited in the introduction where "Hello world" is endlessly repeated.

The spread and speed of communications contributes to "psychopathology" according to Berardi, as collective intelligence is no longer able to adequately process the complexity of information being generated and we have forgotten the lyrics and rhythms that once bound communities together (as in the singing of folk songs). Happiness is exemplified by the public performance of songs. "For Marx, the privileged example of really free working—happiness itself—is 'composition,' the construction of the communist score . . . communism whose song will free the space in which it resonates, and spreads."[118]

A further example is Aaron Koblin and Daniel Massey's artwork *Bicycle Built for Two Thousand* (2009), using Amazon's Mechanical Turk crowdsourcing web service as mentioned earlier in the chapter. This time over two thousand recorded voices are collected and assembled into the song "Daisy Bell," the same song that was used in the first example of musical speech synthesis in 1962 (which made the IBM 7094 the first computer to sing a song).[119] Yet the comparison between the computer-synthesized vocals and the one created with distributed humans is not very encouraging, as both have been effectively synthesized. Despite payment (US$0.06), their voices like their souls appear to have been stolen at source. Interestingly, in Kubrick's *2001: A Space Odyssey*, HAL, the computer capable of reasoning, sings the "Daisy Bell" song as it descends into suicidal depression.[120]

The synthesized voice machine called the mobile phone is Berardi's example to demonstrate the network dependency and depressive tendencies that underpin contemporary production. With smart phones, voices operate as if in an echo chamber, for they have been effectively muted in the nullification of networked intelligence

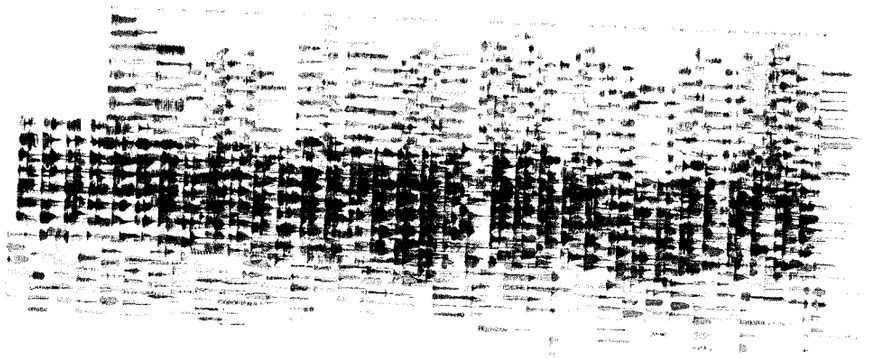

Figure 2.9
Aaron Koblin and Daniel Massey, *Bicycle Built for Two Thousand* (2009). Image courtesy of Aaron Koblin.

(replaced by so-called "smartness"). This is what Berardi means when he says that the soul has been put to work, such that information workers continue to work even when seemingly not working. Available for the iPhone, the parody game *iCapitalism* (2011) created by Crotch Zombie Productions provides an example in this connection, rejected from Apple's App Store for no good reason despite fulfilling their published criteria of acceptance.[121] The nature of the content was clearly the issue, despite the seemingly indiscriminate inclusion of endlessly trivial apps that are made widely available. The game is simple and ironic: the person who pays the most wins: "iCapitalism is the world's first game entirely driven by microtransactions. There is literally no gameplay outside of the ability to upgrade your character using real money."[122] It removes aspects of games like the amount of time required and skill involved, and without even the need for outsourcing development. As with all games, there are winners and losers, so in this case the platform owners win every time, but explicitly. Added interest lies in the fact that the time invested in the production of the app is wasted, as Apple developers invest their labor before its acceptance in the marketplace.

The speech act in this case is aggressively locked down, underlining how subjectivity in the form of the voice is somehow captured and fragmented in the use of these telecommunications devices. The digital network more generally facilitates the spatial and temporal globalization and precaritization of labor, but it is the "cellular" qualities of this that recombine semiotic fragments endlessly to produce semiocapitalism, according to Berardi.[123] The most important commodity of late capitalism, the mobile phone, is the instrument through which this takes place, melting our brains both literally and metaphorically through its use of microwave radio frequencies.

But all is not lost. Berardi reads the present financial crisis as a precondition for the return of the soul, like the return of the repressed. If neoliberalism attempted to capture the very essence of life, it will always ultimately fail, as it wrongly assumes "that the soul can be reduced to mere rationality."[124] The soul is even more unpredictable than the mind or body. It is something far more unknown and complex, its potentialities irreducible to the market or even language. It is something closer to poetry or the voice,[125] and to the indeterminate code acts that have been described thus far that demonstrate the potential for the recomposition of collective action, using improvised scripts and esoteric programs.

3 Coding Publics

There has been much recent interest in revisiting Arendt's ideas, in particular in relation to a reconceptualization of publicness.[1] As outlined in the previous chapter, Arendt identifies the centrality of action to politics, as distinct from other activities related to work and labor, because action is necessarily linked to plurality, to performative actions in public. Although all human activities are conditioned by society,[2] only action and its relation to politics are entirely dependent upon a "paradoxical plurality of unique beings," who with actions and speech introduce themselves to the world.[3] In Virno's work, further recognizing the communicative dimension of capitalism, the reference to Arendt is emphasized because of what he considers the relative ineffectiveness of political action today.[4]

Yet the human capacity to act in the world is enduring, even under the most difficult circumstances according to Arendt, a view underpinned by her support of grassroots political movements at her time of writing (such as the citizen councils that arose during the Hungarian Revolution of 1956). It is interesting to note that at the time of writing now (summer and autumn of 2011), the enduring power of social movements and public action has been proved again, as witnessed by the various "pro-democracy" campaigns in North Africa and the Near East (so-called "Arab Spring"), movements opposing state budget cuts to the public sector, protests against the marketization of education, and the political agenda around Internet freedom and the controversies surrounding WikiLeaks.[5] An example of the latter is the recent "denial of service" attacks by the loosely organized group of "hacktivists" called Anonymous.[6] Emerging from the online message forum 4chan,[7] the group coordinated various distributed denial of service (DDoS) attacks using forums and social media websites, where instructions were disseminated on how to download attack software to bombard websites with data to try to throw them offline, and target sites were publicized such as the organizations that had cut ties with WikiLeaks (such as MasterCard, Visa, and PayPal, through "operation payback"). Their slogan reflects their constitution as a public: "We are Anonymous. We are Legion."[8] Most recently (since 17 September 2011), the Occupy Wall