Christian Stein

Code Poetry.

Wortkunst zwischen künstlicher und natürlicher Sprache

In: Abecedarium der Sprache / Constanze Fröhlich, Martin Grötschel, Wolfgang Klein (Hg.). – ISBN: 978-3-86599-416-5. – Berlin: Kulturverlag Kadmos, 2019. S. 39-46

Persistent Identifier: urn:nbn:de:kobv:b4-opus4-30184

C – *Konditorei Café Richter, Berlin Charlottenburg*

Code Poetry. Wortkunst zwischen künstlicher und natürlicher Sprache

Christian Stein

Die Frage nach dem Verhältnis von Sprache und Wirklichkeit und damit auch von Sprache und Denken beschäftigt die Philosophie seit der Antike. Sprache ist die eigentümliche Form großer Teile des Denkens selbst, das Medium jeder Erkenntnis und als solche die Struktur unserer kulturellen wie individuellen Identität. Sprache ist nichts nachträglich Hinzugekommenes, sie ist der »Leib des Denkens« (Hegel 1955: 328) oder das »Haus des Seins« (Heidegger 2003: 313). Wilhelm von Humboldts Überlegungen zur Sprache bringen dies markant auf den Punkt: Die Sprache »ist ein eignes und selbstständiges Wesen, ein Individuum, die Summe aller Wörter, […] eine Welt, die zwischen der erscheinenden außen, und der wirkenden in uns in der Mitte liegt« (Humboldt 1995: 7) [→ *Humboldts Projekt*].

Dieses selbständige Wesen Sprache allerdings kann auch ein Biest sein – eines, das sich nicht packen lassen will, sich immer und immer wieder entwindet und jedem unserer hilflosen Bezähmungsversuche einen Satz voraus ist. Bei aller Erkenntniskraft ist die Sprache nämlich auch »die Quelle aller Missverständnisse« (Saint-Exupéry 1950: 69), denn die Beziehungen zwischen ihr und der Welt sind keineswegs eindeutig. Insbesondere die analytische Philosophie des 20. Jahrhunderts hatte sich zur Aufgabe gesetzt, dieses Biest dingfest zu machen. Ludwig Wittgenstein, einer der prominentesten Vertreter dieser Richtung, stellt dazu fest: »Die Grenzen meiner Sprache bedeuten die Grenzen meiner Welt« (Wittgenstein 1990a: 67) und folgert daraus später die

zentrale Bestimmung der Philosophie selbst: »Die Philosophie ist ein Kampf gegen die Verhexung unsres Verstandes durch die Mittel unserer Sprache« (Wittgenstein 1990b: 299).

Wittgenstein stellt sich diesem Kampf innerhalb seines Denkens auf sehr unterschiedliche Weise. Im *Tractatus logico-philosophicus* von 1921 bemüht er sich, eine Reihe von aufeinander aufbauenden Definitionen aufzustellen, die die Ambiguität der Sprache festsetzen und somit zu einer stringenten und eindeutigen Sprache als Voraussetzung aller Erkenntnis führen sollen. Denn, so behauptet er in seinem Vorwort programmatisch, »was sich überhaupt sagen läßt, läßt sich klar sagen« (Wittgenstein 1990a: 9). Es ist die Idee einer idealen Sprache, einer Sprache ohne Missverständnisse und mit einer gerade dadurch umso größeren Wirk- und Erklärmacht.

Wittgensteins Versuch, eine solche Sprache zu erschaffen, ist ebenso monumental wie gescheitert. Seine Sätze sind philosophisch hochinteressant, aber sie verwickeln, verzetteln und verunklaren sich gerade zum Ende des *Tractatus* zunehmend. Es ist ein Sich-Verwickeln in Sprache, die immer nur mit sich selbst erklärt werden kann, so dass er schließlich mit einem Gebot der Begrenzung der Sprache schließt, um diese nicht unsinnig werden zu lassen: »Wovon man nicht sprechen kann, darüber muß man schweigen« (Wittgenstein 1990a: 85). Fortan widmet sich Wittgenstein der Sprache über die Sprachspiele und Familienähnlichkeiten, um sich der Bedeutung der Ambiguitäten und Anwendungsformen der Sprache für unser Denken zu nähern, anstatt sie aufzulösen.

Einige Jahrzehnte nach Wittgensteins *Tractatus* entwickelt sich jedoch eine völlig neue Sprachfamilie, die diesen Versuch von einer ganz anderen Seite wieder aufnimmt: die Programmiersprachen. Im Jahr 1945 veröffentlicht Konrad Zuse mit Plankalkül die erste anwendbare Programmiersprache für Computer. Ihr folgten in den fünfziger Jahren mit Fortran, Lisp, COBOL und Algol die ersten modernen Programmiersprachen, die die Welt erobern sollten. Ging es Wittgenstein zuerst noch um ein Bändigen der natürlichen Sprache, entstehen hier gänzlich künstliche Sprachen, die anfangs sehr den Formeln der Mathematik gleichen [→ *Bantu education,*

Lungu & Tucholsky]. Ihr Ziel ist es nicht, universell zwischen Menschen zu kommunizieren, sondern Menschen und Maschinen miteinander sprechen zu lassen. Die Eindeutigkeit ist dabei nicht erst ihr Ziel, sondern ihre unumgängliche Voraussetzung. Was sich überhaupt sagen lässt, könnte man im Anschluss an Wittgenstein behaupten, lässt sich codieren. Missverständnisse und Interpretationen sind hier keine zu überwindenden Verständnisschwierigkeiten mehr, sondern einfach *Bugs*, Programmfehler. Was der Compiler als höchste Interpretationsinstanz nicht versteht, ist schlicht falsch. (Ein Compiler ist ein Computerprogramm, das den Programmiercode in direkt ausführbare Maschinenbefehle übersetzt.)

Natürlich können solche künstlichen Programmiersprachen nicht alles aussagen, was natürliche Sprachen jahrtausendelang an Sagkraft aufgebaut haben. Die damit einhergehende Begrenzung der Aussagemöglichkeiten entfesselt auf der anderen Seite aber eine ungeheure Wirkmächtigkeit. Das erste Mal bewirkt eine Sprache unvermittelt durch menschliche (oder tierische) Interpretation eine Aktion in der Welt. Diese eigentümliche Aktivität des Programmierens bildet Zusammenhänge unserer Welt nicht nur in Sprache ab, sie verändert sie grundlegend. Heute ist unsere Welt so digital, dass kaum ein Aspekt unserer Gesellschaft ohne diese künstlichen Sprachen verstanden werden kann. Wirtschaft, Politik, Kunst oder Wissenschaft sind unumkehrbar durchzogen von Operationslogiken künstlicher Sprachen und wirken durch diese genauso, wie sie ihnen unterworfen sind.

War es jedoch am Anfang noch die Logik der Rechenmaschine, die die Struktur der Programmiersprachen maßgeblich bestimmt hatte, hielten zusehends mehr menschliche und natürlichsprachliche Logiken Einzug in diese Kommunikation. 1964 beispielsweise wurde mit BASIC der »Beginner's Allpurpose Symbolic Instruction Code« vorgestellt, der gezielt auf Einsteiger abzielte: Der Übergang von der natürlichen Sprache zur künstlichen sollte so einfach wie möglich sein. Seitdem richten sich Programmiersprachen mehr und mehr an den Menschen aus und sehen immer weniger aus wie technische Formeln. Ziel bei der Entwicklung einer neuen

Programmiersprache ist nicht mehr nur deren Performanz, sondern die Lesbarkeit, Schreibbarkeit und Verständlichkeit durch den Menschen. Mitte der 1990er Jahre entwickelte Yukihiro Matsumoto die Programmiersprache Ruby, die sich stärker als je zuvor an menschlicher Sprache sowie den Annahmen und der Logik menschlichen Denkens orientiert. Über sie schreibt er Unerwartetes: »Rubys wesentliches Ziel ist ›Freude‹. Meines Wissens gibt es keine andere Sprache, die sich so sehr auf die Freude konzentriert. Rubys eigentliches Ziel ist es zu erfreuen – Sprachdesigner, Anwender, Sprachlerner, jeden« (Matsumoto 2000, zit. nach der Seite »Ruby (Programmiersprache)« 2018).

Aber nicht nur die Programmiersprachen nähern sich immer mehr den natürlichen Sprachen an, umgekehrt ist dies auch der Fall. Bereits heute lernen Kinder und Teenager Programmiersprachen ganz natürlich parallel zur ersten Fremdsprache. Diesen *Digital Natives* ist der Computer kein zu erlernendes Werkzeug mehr, dem man seine Logik aufzwingen muss, er ist eine Erweiterung ihrer selbst und eine fundamentale Verbindung zur Welt. Im Dialog mit aufkommenden Künstlichen Intelligenzen wie Siri oder Alexa, die über mündliche Sprachbefehle bedient werden, verschränken sich natürliche und künstliche Sprache bereits umfassend. In diesem Sinne gliedern sich die Strukturen der Programmiersprachen auch in unsere Alltagssprache und unser Denken ein – wir lernen so zu sprechen, dass die Maschine uns versteht [→ *Diagnose Dr. Online*].

Aber die Verschränkung macht dort nicht halt, sondern geht weiter. In der Kunst, die seit jeher die Reflexion über unsere Welt und Gesellschaft betrieben hat, steht neben Literatur und bildender Kunst nun die *Code Art*. Spätestens seit der *Ars Electronica* ist digitale Kunst überall präsent. Mit Programmiersprachen lassen sich völlig neue Formen der Kunst schaffen. Aber nicht nur das. Der programmierte Code selbst – nicht nur seine Ausführung – ist zur Kunst geworden. Diese Kunstform nennt man *Code Poetry*.

In *Code Poetry* verschränken sich natürliche Sprache und Programmiersprache zu einer Form der Poesie, die sowohl

für den Menschen verständlich als auch für die Maschine interpretierbar ist. Der Compiler ist dabei in der Lage, das Gedicht als Programmcode zu verstehen und entsprechend zu verarbeiten. Für den Menschen entfalten sich dagegen Bedeutungsdimensionen, die der Maschine verborgen bleiben. Die Bedeutung eines *Code-Poetry*-Gedichts ist somit nur im Dazwischen zu erschließen, dem Verständnis sowohl aus Menschen- als auch aus Maschinensicht: Zwischen den Sprachen findet hier eine neue Form der Kunst statt, die die Ambiguität mit der Eindeutigkeit verbindet.

Das erste berühmt gewordene *Code-Poetry*-Gedicht ist »Black Perl« von Larry Wall aus dem Jahr 1990. Wall ist der Erfinder der Programmiersprache Perl und schreibt ein Gedicht, das aussieht wie natürliche Sprache, allerdings gleichzeitig syntaktisch korrekten Perl-Code darstellt. Gibt man dieses Gedicht in einen Computer ein, läuft es als Programm ab. In diesem frühen Versuch berechnet das Programm zwar nicht allzu viel und bricht de facto bei der ersten Verwendung des Befehles »exit« ab – aber allein die Tatsache, dass ein Gedicht ausführbar ist, ändert Bedeutung und Lesart fundamental:

Das Gedicht liest sich wie ein Zauberspruch und verweist bereits damit auf die besondere Fähigkeit der Programmiersprachen, Worte gleichsam magisch in Aktionen münden zu lassen. In diesem »Zauberspruch« ist nun beispielsweise eine Schleife eingebaut (die Wiederholung der gleichen Code-Segmente bis zum Erreichen einer Abbruchbedingung): »redo ritual until ›all the spirits are pleased‹;«. Als Abbruchbedingung dieser Schleife wird allerdings eine (in sich nicht als Befehl zu lesende) Zeichenkette, ein sogenannter String angegeben (erkennbar an den umgebenden Anführungszeichen), der dann kurz darauf folgt, so dass die Methode »ritual« in Wirklichkeit niemals ausgeführt wird. Man könnte sagen, »all spirits are pleased« verhindert das Ritual. Ein anderes Beispiel: »shift moralities« bedeutet, dass der erste Eintrag des Feldes, in dem das Wort »moralities« steht, gelöscht wird und somit keine einfache Moralverschiebung stattfindet, wie man meinen könnte, sondern stattdessen die erste Moral entfernt wird – welche auch immer das ist.

Das *Code-Poetry*-Gedicht »Black Perl« von Jerry Wall:

```
BEFOREHAND: close door, each window & exit; wait until time.
     open spellbook, study, select it, confess, tell, deny;
write it, print the hex while each watches,
     reverse "its length", write again;
     kill spiders, pop them, chop, split, kill them.
          unlink arms, shift, wait & listen (listening, wait),
sort the flock (then, warn "the goats". kill "the sheep");
     kill them, dump qualms, shift moralities,
     values aside, each one;
          die sheep, die, reverse system
          you accept (reject, respect);
next step,
     kill next sacrifice, each sacrifice,
     wait, redo ritual until "all the spirits are pleased";
     do it ("as they say").
do it(*everyone***must***participate***in***forbidden**s*e*x*).
return last victim; package body;
     exit crypt (time, times & "half a time") & close it,
     select (quickly) & warn next victim;
AFTERWORDS: tell nobody.
     wait, wait until time;
     wait until next year, next decade;
          sleep, sleep, die yourself and
          rest at last
```

Diese Ebenen des Gedichtes sind nur verständlich, wenn man die Perl-Syntax versteht. Nicht der ausgegebene Text – das Resultat des Programms – ist interessant, sondern sein Quellcode. Nicht Englisch ist die Sprache des Gedichts, sondern Perl. Damit führt Wall vor Augen, dass der Code selbst eine eigene Logik, Ästhetik und Aussagekraft hat. Hier verschwimmt die Grenze von Programm-Code und Literatur-Code.

Viele weitere Beispiele von *Code Poetry* sind seither geschrieben worden, auch wenn die Zahl der Autoren noch verhältnismäßig überschaubar ist. Teils sind es vollständige, ausführbare Programme, teils sind es Experimente mit der Syntax und Anordnung unterschiedlicher Programmiersprachen. Fast überall geht es aber um das Spiel mit den Perspektiven zwischen Mensch und Computer. Es ist die Frage nach dem Verhältnis von technischer und natürlicher Sprache oder, wie Friedrich Kittler es formuliert, »die dringliche Frage, was Wörter leisten und was sie nicht leisten, nach welchen

Regeln sie aufgeschrieben und gespeichert werden, nach welchen Regeln gelesen und ausgelegt« (Kittler 2012: 117).

Wahrscheinlich haben sich die meisten Sprachdenker vergangener Jahrhunderte in allen Bemühungen um eine Eindeutigkeit der Sprache nicht träumen lassen, dass es Maschinen bzw. die Kommunikation mit ihnen sein würden, die einen so wesentlichen und wirkmächtigen Beitrag zu einer solchen Eindeutigkeit leisten sollten. Interessanterweise aber zeigt gerade diese Entwicklung, dass sich in die zugrundeliegenden Fragen nach dem Verhältnis von Sprache und Welt niemals Eindeutigkeit bringen lässt. Aber sie sind um eine Dimension erweitert, die die Interpretation nicht mehr nur den Menschen überlässt. Das kann man als eine weitere Kränkung der Menschheit auffassen – oder aber als eine Chance für neue Fragen und neue Antworten.

Natürliche wie künstliche Sprachen codieren so oder so auf bedeutsame Weise unser Denken und Handeln. In diesem Sinne erscheint eine Kunstform wie *Code Poetry* besonders geeignet, die Bedeutung schaffenden Strukturen unserer gleichermaßen analogen wie digitalen Welt aufscheinen zu lassen – in ihrer mechanischen Wirkmacht wie in ihrer unzähmbaren, fantastischen Biestigkeit.

Literatur

Hegel, Georg Wilhelm Friedrich [1843] (1955): *Die Wissenschaft der Logik*. In: Ders.: *Sämtliche Werke*. Hg. v. Herrmann Glockner. Bd. 8. Stuttgart-Bad Cannstatt: Frommann.

Heidegger, Martin [1959] (2003): »Der Weg zur Sprache.« In: Ders.: *Unterwegs zur Sprache*. Stuttgart: Klett-Cotta. S. 239–268.

Humboldt, Wilhelm von [1806] (1995): »Über die Natur der Sprache im allgemeinen«. In: Ders.: *Schriften zur Sprache*. Hg. v. Michael Böhler. Ergänzte Ausgabe Stuttgart: Reclam.

Kittler, Friedrich (2012): »Aufschreibesysteme 1800/1900. Vorwort«. In: *Zeitschrift für Medienwissenschaft*. 6. S. 117–126.

Saint-Exupéry, Antoine de (1950): *Der kleine Prinz*. Übertrag. ins Dt. v. Grete und Josef Leitgeb. Bad Salzig: Rauch.

Seite »Ruby (Programmiersprache)«. In: *Wikipedia. Die freie Enzyklopädie*. Bearbeitungsstand: 15. September 2018, 18:50 UTC. URL: https://de.wikipedia.org/w/index.php?title=Ruby_(Programmiersprache)&oldid=180936349 (Abgerufen: 28.9.2018, 12:51 UTC).

Wall, Larry (1990): *Black Pearl*. Auffindbar unter der URL: https://www.perlmonks.org/?node_id=578707 (Abgerufen: 1.10.2018).
Wittgenstein, Ludwig [1921] (1990a): *Tractatus logico-philosophicus*. In: Ders.: *Werkausgabe*. Bd 1. Hg. v. Joachim Schulte. Frankfurt am Main: Suhrkamp. S. 7–86.
Wittgenstein, Ludwig [1953] (1990b): *Philosophische Untersuchungen*. In: Ders.: *Werkausgabe*. Bd 1. Hg. v. Joachim Schulte. Frankfurt am Main: Suhrkamp. S. 225–580.