Is Software Malfunction an Oxymoron?

Jesse Hughes

July 25, 2007
An introduction to malfunction

Is Software Malfunction an Oxymoron?
Outline

1. An introduction to malfunction
2. Token and type malfunction
Outline

1. An introduction to malfunction
2. Token and type malfunction
3. Misfunction
Outline

1. An introduction to malfunction
2. Token and type malfunction
3. Misfunction
Function-bearers

Some things have functions. We can ask, “What is it for?”
Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
Function-bearers

Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
- “That switch mutes the television.”
Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
- “That switch mutes the television.”
- “The magician’s assistant is for distracting the audience.”
Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
- “That switch mutes the television.”
- “The magician’s assistant is for distracting the audience.”
- “The subroutine ensures that the user is authorized.”
Function-bearers

Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
- “That switch mutes the television.”
- “The magician’s assistant is for distracting the audience.”
- “The subroutine ensures that the user is authorized.”

We ascribe functions to biological stuff,
Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
- “That switch mutes the television.”
- “The magician’s assistant is for distracting the audience.”
- “The subroutine ensures that the user is authorized.”

We ascribe functions to biological stuff, artifacts,
Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
- “That switch mutes the television.”
- “The magician’s assistant is for distracting the audience.”
- “The subroutine ensures that the user is authorized.”

We ascribe functions to biological stuff, artifacts, personal roles, ...
Function-bearers

Some things have functions. We can ask, “What is it for?”

- “The function of the heart is to pump blood.”
- “That switch mutes the television.”
- “The magician’s assistant is for distracting the audience.”
- “The subroutine ensures that the user is authorized.”

We ascribe functions to biological stuff, artifacts, personal roles, software...
Malfunction

A truism: *Things don’t always work like they should.*
Malfunction

A truism: *Things don’t always work like they should.*

- “An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)
A truism: *Things don’t always work like they should.*

- “An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)
- “a biological part functions properly when it can do what it was selected for and malfunctions when it cannot.” (Neander, 1995)
A truism: *Things don’t always work like they should.*

- “An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)
- “a biological part functions properly when it can do what it was selected for and malfunctions when it cannot.” (Neander, 1995)
- “If you can say what a thing is supposed to do, then you can also say when it is failing to do something that it is supposed to do, that is, malfunctioning.” (Preston, 1998)
Malfunction

A truism: *Things don’t always work like they should.*

- “An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)
- “a biological part functions properly when it can do what it was selected for and malfunctions when it cannot.” (Neander, 1995)
- “If you can say what a thing is supposed to do, then you can also say when it is failing to do something that it is supposed to do, that is, malfunctioning.” (Preston, 1998)
- “It is of the essence of purposes and intentions [and hence, of functions] that they are not always fulfilled.” (Millikan, 1989)
A truism: *Things don’t always work like they should.*

- “An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)
- “a biological part functions properly when it can do what it was selected for and malfunctions when it cannot.” (Neander, 1995)
- “If you can say what a thing is supposed to do, then you can also say when it is failing to do something that it is supposed to do, that is, malfunctioning.” (Preston, 1998)
- “It is of the essence of purposes and intentions [and hence, of functions] that they are not always fulfilled.” (Millikan, 1989)

**In sum:** Function-bearers are capable of malfunction.
Outline

1. An introduction to malfunction
2. Token and type malfunction
3. Misfunction
Token malfunction

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)
An introduction to malfunction
Token and type malfunction
Misfunction

"An obvious fact about function categories is that their members can always be defective…" (Millikan, 1989)

Token malfunction

"An obvious fact about function categories is that their members can always be defective…" (Millikan, 1989)

Token dysfunction (strong)

A token **dysfunctions** if it cannot perform its function.

Hughes

Is Software Malfunction an Oxymoron?
“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

**Token dysfunction (strong)**

A token *dysfunctions* if it cannot perform its function.

But what about a TV with poor reception?
An introduction to malfunction
Token and type malfunction
Misfunction

Token malfunction

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

Token dysfunction (weak)

A token **dysfunctions** if it cannot perform its function **reliably** or **effectively**.

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)
An introduction to malfunction
Token and type malfunction
Misfunction

Token malfunction

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

Token dysfunction (weak)
A token **dysfunctions** if it cannot perform its function **reliably** or **effectively**.

- Reliably: How likely the goal is achieved.
“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

Token dysfunction (weak)
A token dysfunctions if it cannot perform its function reliably or effectively.

- Reliably: How likely the goal is achieved.
- Effectively: The degree to which the goal is achieved.
An introduction to malfunction
Token and type malfunction
Misfunction

Token malfunction

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

Token dysfunction (weak)

A token **dysfunctions** if it cannot perform its function **reliably** or **effectively**.

Reliably or effectively **compared to what**?
An introduction to malfunction
Token and type malfunction
Misfunction

Token malfunction

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

Token dysfunction (weak)

A token **dysfunctions** if it cannot perform its function **reliably** or **effectively**.

Reliably or effectively **compared to what**?

Compared to “normal” tokens of the same type.
An introduction to malfunction
Token and type malfunction
Misfunction

Token malfunction

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

Token dysfunction (weak)

A token **dysfunctions** if it cannot perform its function **reliably** or **effectively**.

Reliably or effectively **compared to what?**

Compared to **“normal”** tokens of the same type.

(Needs some discussion)
Software tokens

What is a software token?
Software tokens

What is a software token?

**Type:** Defined by code.
What is a software token?

**Type:** Defined by code.

**Token (roughly):** a copy of code, ready for execution.
Software tokens

What is a software token?

**Type:** Defined by code.

**Token (roughly):** a copy of code, ready for execution.

**Consequence:** Two software tokens of the same type behave indistinguishably.
Software tokens

What is a software token?

**Type:** Defined by code.

**Token (roughly):** a copy of code, ready for execution.

**Consequence:** Two software tokens of the same type behave indistinguishably.

Software does not malfunction?

```perl
sub filter {
    my ($self) = @_;
    my ($status);
    tr/n-zs-mN-ZA-N/a-zA-Z/
    if ($status = filter_read()) > 0;
    $status;
}
```
Type dysfunction

Need: a notion of type dysfunction (i.e. bad design).
An introduction to malfunction
Token and type malfunction
Misfunction

Need: a notion of **type dysfunction** (i.e. bad design).

**Type dysfunction**
A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

**Hughes**

Is Software Malfunction an Oxymoron?
An introduction to malfunction
Token and type malfunction
Misfunction

Type dysfunction

**Need:** a notion of **type dysfunction** (i.e. bad design).

A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

- A bad cutter: wires bend easily.

**Hughes**

Is Software Malfunction an Oxymoron?
An introduction to malfunction
Token and type malfunction
Misfunction

Need: a notion of type dysfunction (i.e. bad design).

Type dysfunction
A type dysfunctions if it does not perform its function as effectively or reliably as other available, comparable types.

- A bad cutter: wires bend easily.
- An okay cutter: sturdy, useful.

Hughes
Is Software Malfunction an Oxymoron?
Type dysfunction

Need: a notion of **type dysfunction** (i.e. bad design).

A type **dysfunctions** if it does not perform its function as effectively or reliably as other available, comparable types.

- A bad cutter: wires bend easily.
- An okay cutter: sturdy, useful.
- A *great* cutter: has thumb rest!
Type dysfunction

Need: a notion of **type dysfunction** (i.e. bad design).

**Type dysfunction**

A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

- A bad cutter: wires bend easily.
- An okay cutter: sturdy, useful.
- A great cutter: has thumb rest!

**Note:** A bit vague when mediocrity becomes dysfunction!
Type dysfunction

Need: a notion of **type dysfunction** (i.e. bad design).

**Type dysfunction**

A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

Available: *within state-of-the-art capabilities*
An introduction to malfunction
Token and type malfunction
Misfunction

Need: a notion of **type dysfunction** (i.e. bad design).

**Type dysfunction**

A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, *comparable* types.

**Available:** within state-of-the-art capabilities

**Comparable:** satisfying similar function with similar costs, lifespan, etc.

Hughes

Is Software Malfunction an Oxymoron?
Need: a notion of **type dysfunction** (i.e. bad design).

**Type dysfunction**

A type **dysfunctions** if it does not perform its function as effectively or reliably as other available, comparable types.

**Available:** within state-of-the-art capabilities

**Comparable:** satisfying similar function with similar costs, lifespan, etc.

Strong version?
Software dysfunction and specifications

Most software comes with a specification.

Example: A program is an **adder** if:

- **Given** appropriate input \( x, y \)
- **Output** \( x + y \).
Software dysfunction and specifications

Most software comes with a specification.

Example: A program is an adder if:

Given appropriate input $x, y$
Output $x + y$.

A program which fails to output $x + y$ is no adder at all.
Software dysfunction and specifications

Most software comes with a specification.

**Example:** A program is an **adder** if:

- **Given** appropriate input \( x, y \)
- **Output** \( x + y \).

A program which fails to output \( x + y \) is no adder at all.

**Specifications provide constitutive norms.**
Software dysfunction and specifications

Most software comes with a specification.

**Example:** A program is an *adder* if:

- **Given** appropriate input $x$, $y$
- **Output** $x + y$.

A program which fails to output $x + y$ is no adder at all.

Specifications provide **constitutive** norms.

Other examples of constitutive norms:
- Game rules
Software dysfunction and specifications

Most software comes with a specification.

**Example:** A program is an adder if:
- Given appropriate input $x, y$
- Output $x + y$.

A program which fails to output $x + y$ is no adder at all.

Specifications provide **constitutive** norms.

Other examples of constitutive norms:
- Game rules
- Axioms for mathematical models

Hughes

Is Software Malfunction an Oxymoron?
Software dysfunction

But some software is better than others.

Consider a chess program that
• plays chess correctly (satisfies spec.)
Software dysfunction

But some software is better than others.

Consider a chess program that
- plays chess correctly (satisfies spec.)
- values knights over rooks
Software dysfunction

But some software is better than others.

Consider a chess program that

- plays chess correctly (satisfies spec.)
- values knights over rooks

**Functional goal:** winning
Software dysfunction

But some software is better than others.

Consider a chess program that

- plays chess correctly (satisfies spec.)
- values knights over rooks

**Functional goal:** winning

This program does not reliably win.
But some software is better than others.

Consider a chess program that

- plays chess correctly (satisfies spec.)
- values knights over rooks

**Functional goal:** winning

This program does not reliably win.

It is badly designed—i.e., *dysfunctions.*
But some software is better than others.

Consider a chess program that
- plays chess correctly (satisfies spec.)
- values knights over rooks

**Functional goal:** winning

This program does not reliably win.

It is badly designed—i.e., *dysfunctions*.

Software is capable of *type dysfunction*. 
Outline

1. An introduction to malfunction
2. Token and type malfunction
3. Misfunction
Other kinds of malfunction?

Type dysfunction
A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

Example: Floor-scorching stove
A certain gas stove was constructed without a heat shield.
Other kinds of malfunction?

Type dysfunction

A type **dysfunctions** if it does not perform its function as effectively or reliably as other available, comparable types.

**Example:** Floor-scorching stove

A certain gas stove was constructed without a heat shield.

**Result:** Damaged floors
An introduction to malfunction
Token and type malfunction
Misfunction

Other kinds of malfunction?

Type dysfunction
A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

**Example:** Floor-scorching stove
A certain gas stove was constructed without a heat shield.

**Result:** Damaged floors
- Cooked food properly (no *dysfunction*)
Other kinds of malfunction?

Type dysfunction

A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

**Example:** Floor-scorching stove

A certain gas stove was constructed without a heat shield.

**Result:** Damaged floors

- Cooked food properly (no *dysfunction*)
- But malfunctioning nonetheless!
Other kinds of malfunction?

Type dysfunction
A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

Dysfunction: Not doing what it should.
Other kinds of malfunction?

**Type dysfunction**

A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

**Dysfunction:** Not doing what it should.

**Misfunction:** Doing what it shouldn’t.
An introduction to malfunction
Token and type malfunction

Misfunction

Type dysfunction

A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

Type malfunction

A type *misfunctions* if it produces negative effects that other available types do not produce.
An introduction to malfunction
Token and type malfunction

Misfunction

Type dysfunction
A type *dysfunctions* if it does not perform its function as effectively or reliably as other available, comparable types.

Type malfunction
A type *misfunctions* if it produces negative effects that other available types do not produce.

Token malfunction
A token *misfunctions* if it produces negative effects that other ("normal") tokens do not produce.

Hughes

Is Software Malfunction an Oxymoron?
An introduction to malfunction
Token and type malfunction

Misfunction

Misfunctioning software

Type malfunction

A type *misfunctions* if it produces negative effects that other available types do not produce.

Examples:

- Misleading interfaces
  - Anna.Kournikova.jpg.vbs appears as Anna.Kournikova.jpg
Misfunctioning software

Type malfunction

A type *misfunctions* if it produces negative effects that other available types do not produce.

Examples:

- Misleading interfaces
  Anna.Kournikova.jpg.vbs appears as Anna.Kournikova.jpg

- Security flaws
  The ping of death
An introduction to malfunction
Token and type malfunction
Misfunction

Misfunctioning software

Type malfunction

A type *misfunctions* if it produces negative effects that other available types do not produce.

Examples:

- Misleading interfaces
  Anna.Kournikova.jpg.vbs appears as Anna.Kournikova.jpg

- Security flaws
  The ping of death...now dysfunction!
An introduction to malfunction
Token and type malfunction

Misfunction

Misfunctioning software

Type malfunction
A type *misfunctions* if it produces negative effects that other available types do not produce.

Examples:

- Misleading interfaces
  Anna.Kournikova.jpg.vbs appears as Anna.Kournikova.jpg
- Security flaws
  The ping of death... now dysfunction!
- Unrealistic game physics
  Strafe jumping

Hughes
Is Software Malfunction an Oxymoron?
An introduction to malfunction
Token and type malfunction
Misfunction

Misfunctioning software

Type malfunction
A type *misfunctions* if it produces negative effects that other available types do not produce.

Examples:

- Misleading interfaces
  Anna.Kournikova.jpg.vbs appears as Anna.Kournikova.jpg
- Security flaws
  The ping of death... now dysfunction!
- Unrealistic game physics
  Strafe jumping... now de rigueur!
“An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)
Malfunction: the old picture

“An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

- Applies to tokens only
Malfunction: the old picture

“An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)

- Applies to tokens only
  ... hence not to software.
Malfunction: the old picture

“An obvious fact about function categories is that their members can always be defective. . .” (Millikan, 1989)

- Applies to tokens only
  ... hence not to software.
- Strong dysfunction only
Malfunction: the old picture

“An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)

- Applies to tokens only
  ... hence not to software.
- Strong dysfunction only
  ... software dysfunction is weak.
Malfunction: the old picture

“An obvious fact about function categories is that their members can always be defective...” (Millikan, 1989)

- Applies to tokens only
  ...hence not to software.
- Strong dysfunction only
  ...software dysfunction is weak.
- No misfunction
An obvious fact about function categories is that their members can always be defective…” (Millikan, 1989)

- Applies to tokens only
  ... hence not to software.
- Strong dysfunction only
  ... software dysfunction is weak.
- No malfunction
  ... missing most software bugs.
Malfunction: the new picture

<table>
<thead>
<tr>
<th>Dysfunction</th>
<th>Misfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>token</strong></td>
<td><strong>type</strong></td>
</tr>
<tr>
<td>Artifact</td>
<td></td>
</tr>
<tr>
<td>Biological</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
</tbody>
</table>

Is Software Malfunction an Oxymoron?
Malfunction: the new picture

<table>
<thead>
<tr>
<th>Dysfunction</th>
<th>Misfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>token</td>
</tr>
<tr>
<td>type</td>
<td>type</td>
</tr>
<tr>
<td>Artifact</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Biological</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Outstanding issues with biological type malfunction!

```perl
sub filter {
  my ($self) = @_;
  my ($status);
  tr/a-zA-Z/;
  if ($status = filter_read()) > 0;
  $status;
}
```
An introduction to malfunction
Token and type malfunction
Misfunction

Malfunction: the new picture

<table>
<thead>
<tr>
<th>Dysfunction</th>
<th>Misfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>type</td>
</tr>
<tr>
<td>Artifact</td>
<td>yes</td>
</tr>
<tr>
<td>Biological</td>
<td>yes</td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
</tbody>
</table>

Note: Outstanding issues with biological type malfunction!
Malfunction: the new picture

<table>
<thead>
<tr>
<th>Dysfunction</th>
<th>Misfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>type</td>
</tr>
<tr>
<td>Artifact</td>
<td>yes</td>
</tr>
<tr>
<td>Biological</td>
<td>yes</td>
</tr>
<tr>
<td>Software</td>
<td>no</td>
</tr>
</tbody>
</table>

Note: Outstanding issues with biological type malfunction!
Thank you!