Is Software Malfunction an Oxymoron?

Jesse Hughes

July 25, 2007

Outline

Outline

Outline

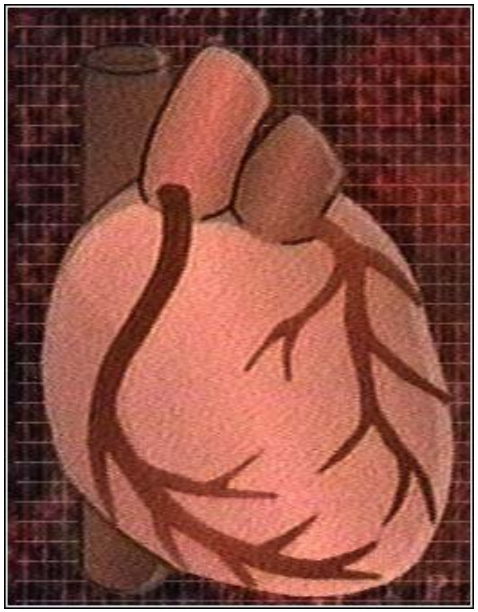
Outline

Function-bearers

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

Function-bearers

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."

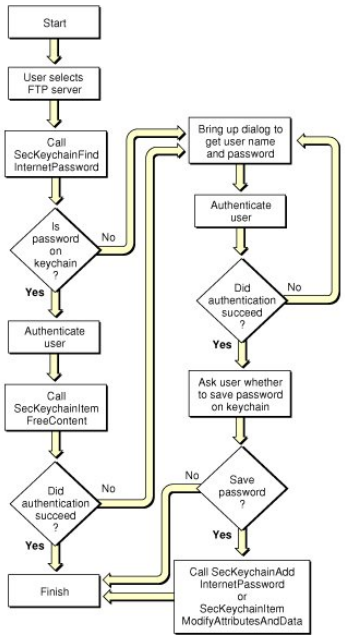
Function-bearers

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."

Function-bearers

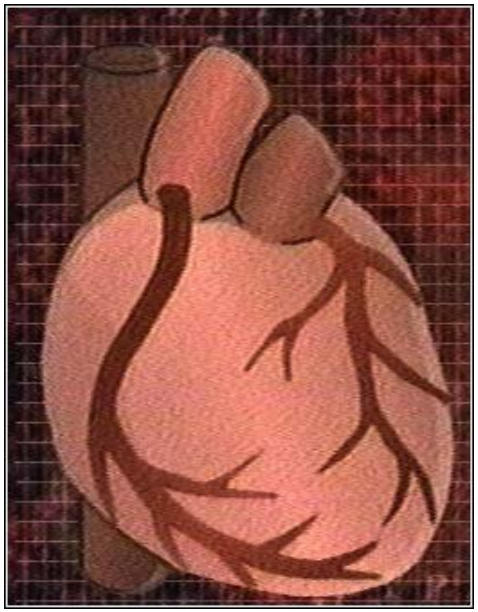
Some things have functions. We can ask, "What is it for?"



- "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."

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,

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,

Function-bearers

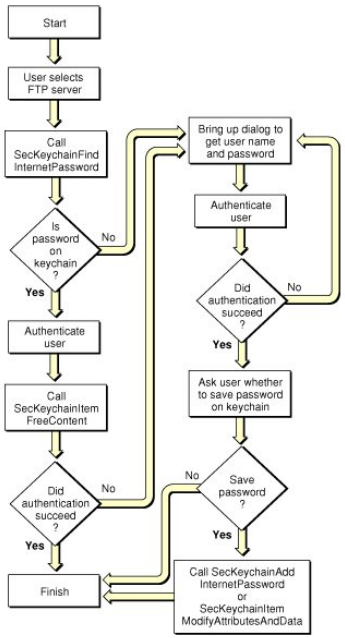
Some things have functions. We can ask, "What is it for?"

- "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,

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)

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)

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)

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)

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)

In sum: Function-bearers are capable of malfunction.

Outline

Token malfunction

"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.

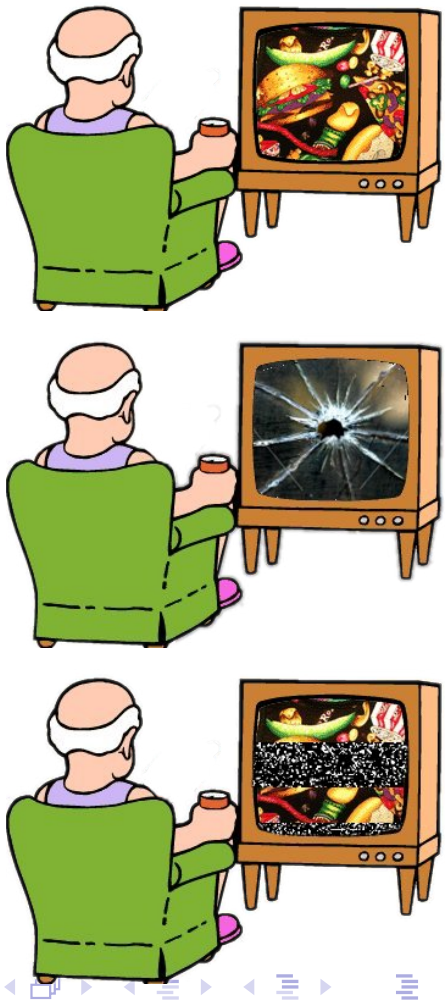
Token malfunction

"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.

But what about a TV with poor reception?

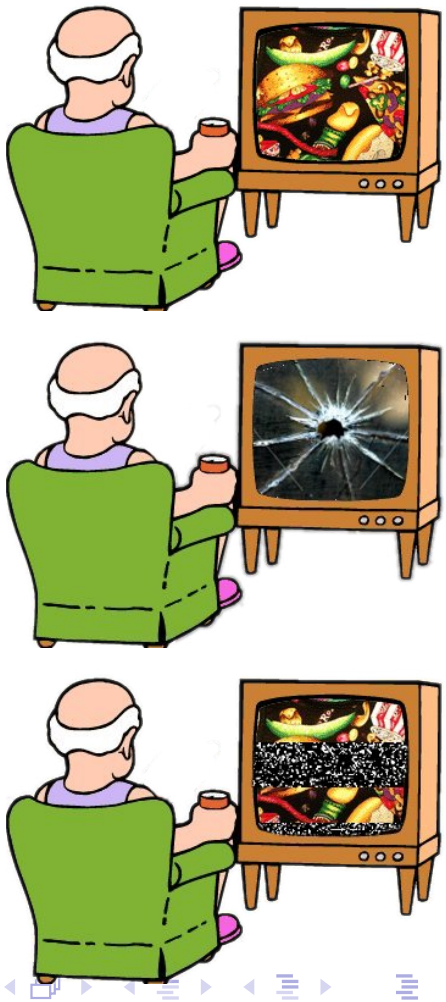


Token malfunction

"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*.



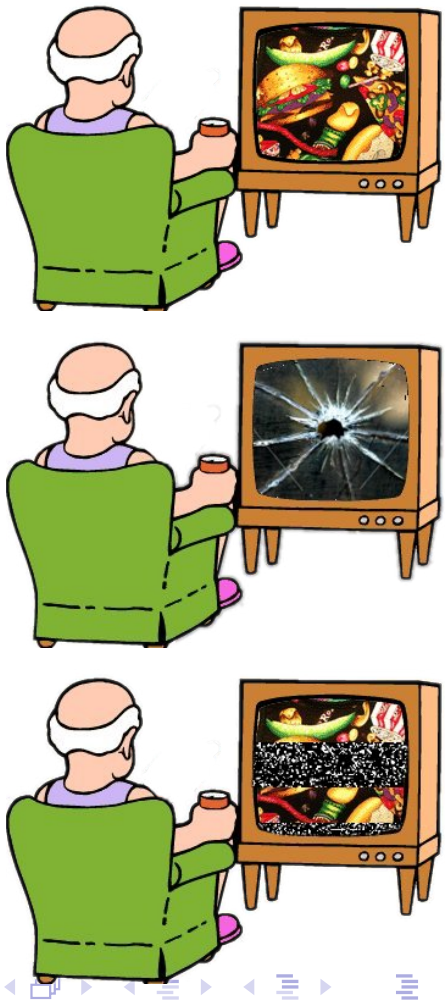
Token malfunction

"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.



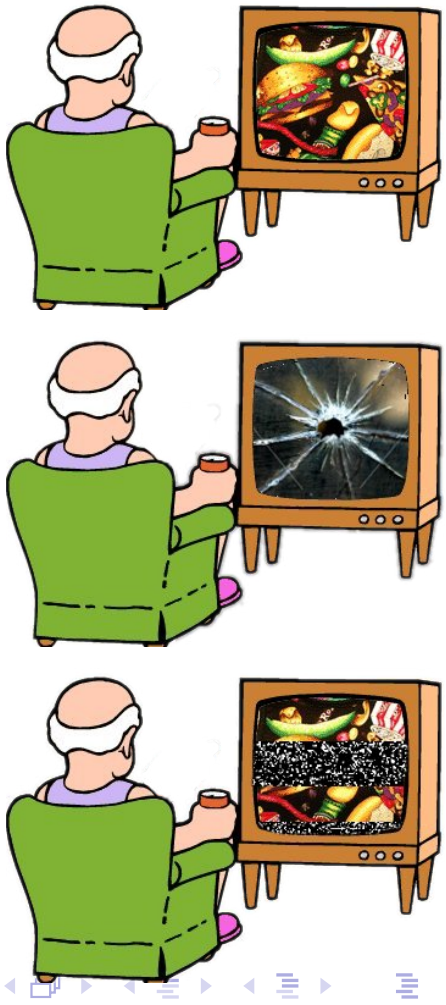
Token malfunction

"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.
- Effectively: The degree to which the goal is achieved.



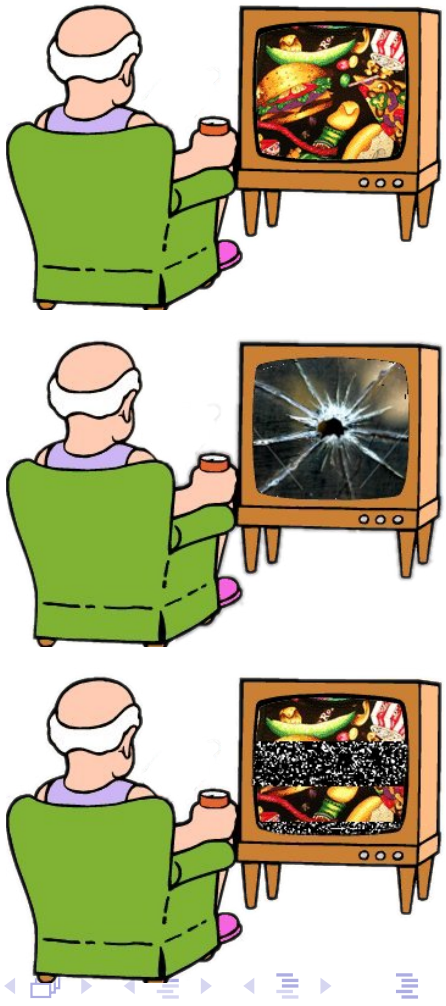
Token malfunction

"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?*



Token malfunction

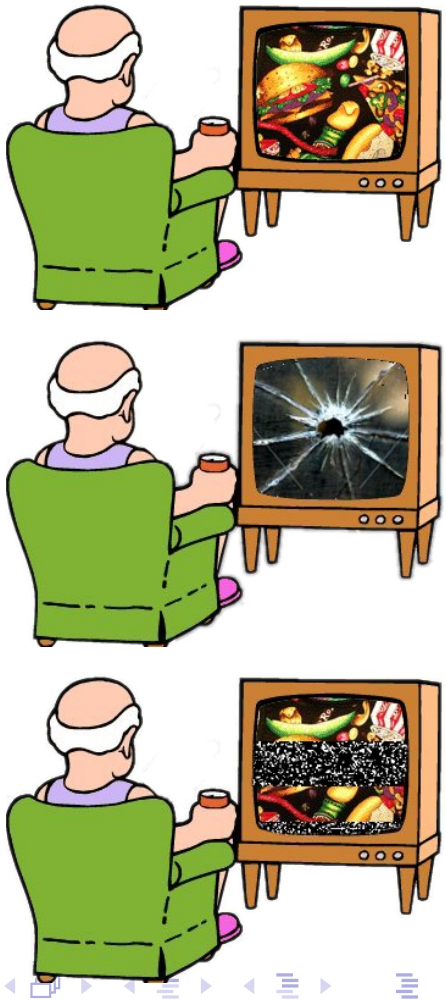
"An obvious fact about function categories is that their members can always be defective. . ." (Millikan, 1989)

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.



Token malfunction

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

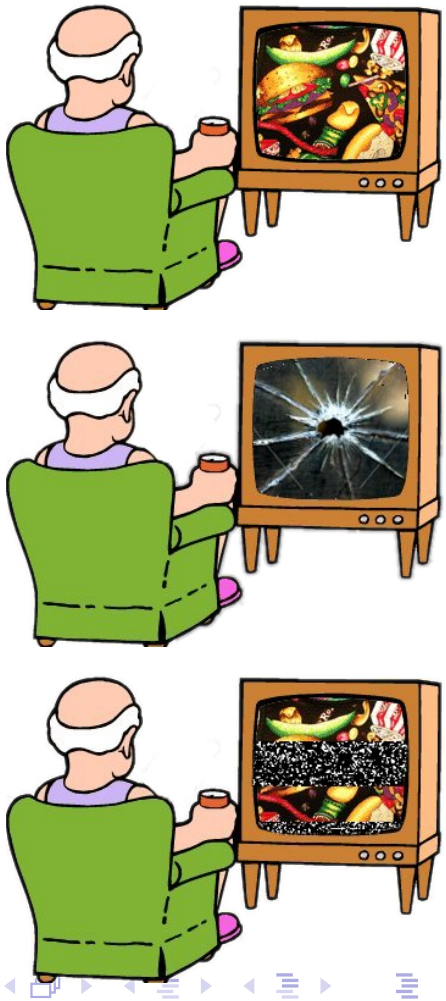
Token dysfunction (weak)

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.

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

Software type

Software tokens

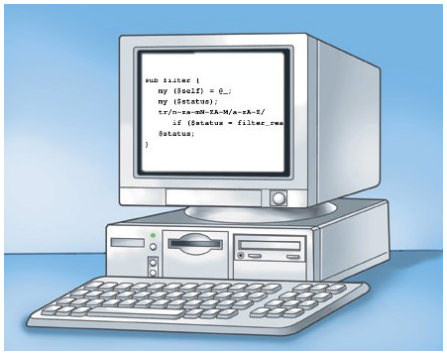
What is a software token?

Type: Defined by code.

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

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

Software type



Software token

Software tokens

What is a software token?

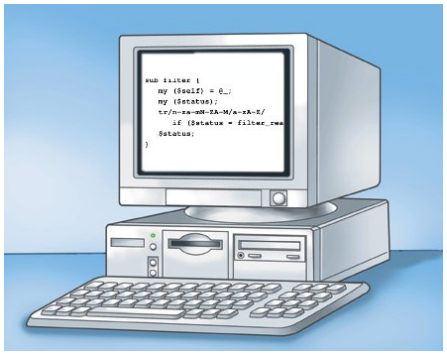
Type: Defined by code.

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

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

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

Software type



Software token

Software tokens

What is a software token?

Type: Defined by code.

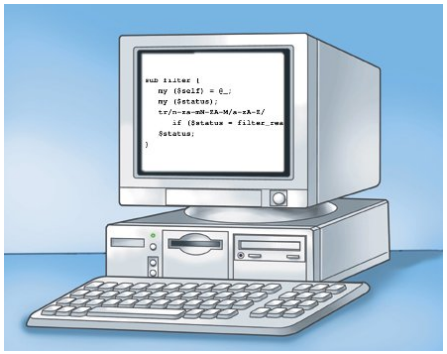
Token (roughly): a copy of code, ready for execution.

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

Software does not malfunction?

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

Software type



Software token

Type dysfunction

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

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.

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.

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.

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!

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

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

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

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

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

Strong version?

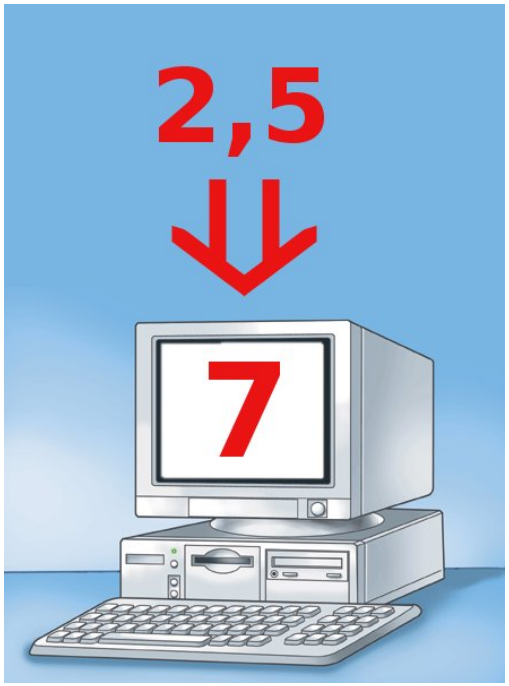
Software dysfunction and specifications

Most software comes with a specification.

Example: A program is an adder if:

Given appropriate input x, y

Output $x + y$.



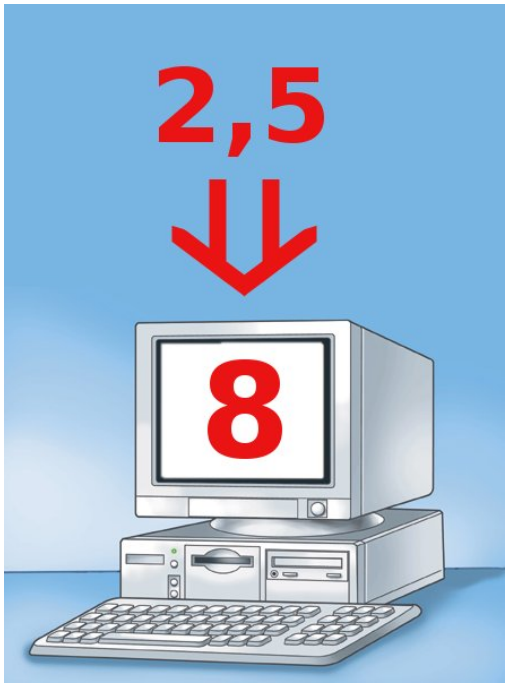
Software dysfunction and specifications

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.



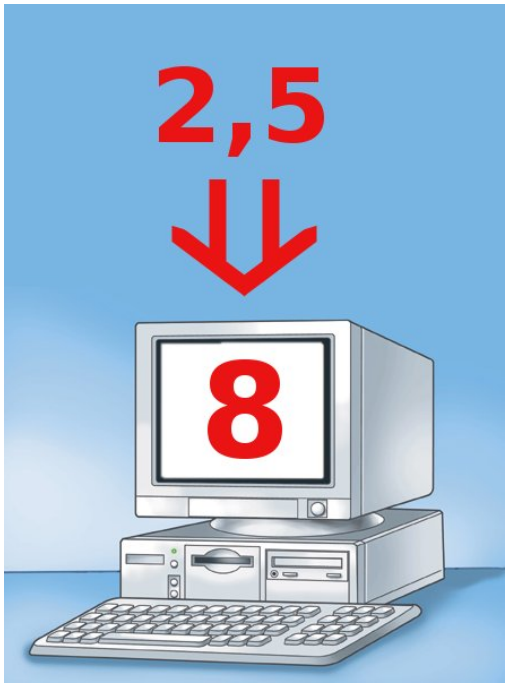
Software dysfunction and specifications

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.



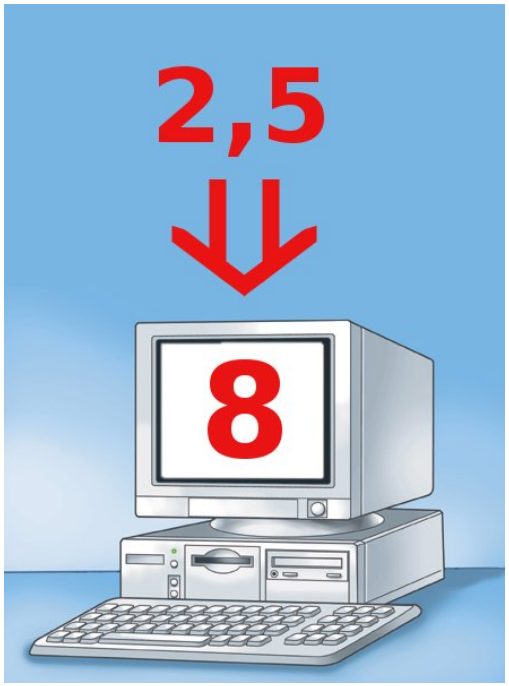
Software dysfunction and specifications

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:

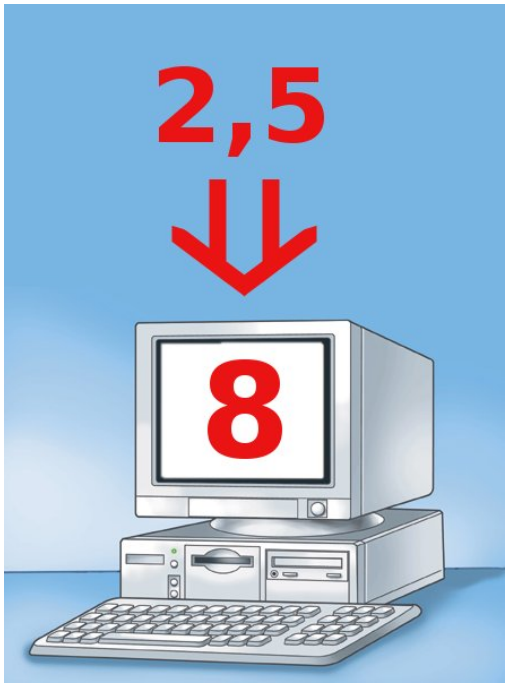
- Game rules

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

Software dysfunction

But some software is better than others.

Consider a chess program that

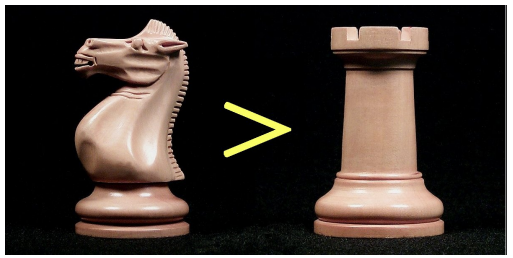
- plays chess correctly (satisfies spec.)

Software dysfunction

But some software is better than others.

Consider a chess program that

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



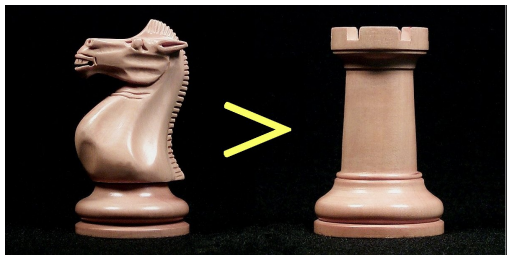
Software dysfunction

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

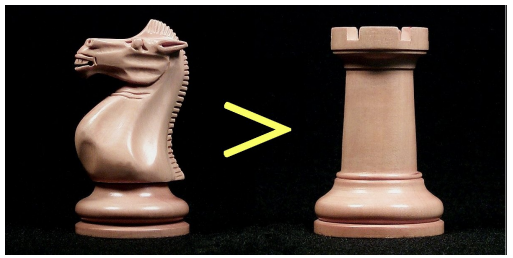
But some software is better than others.

Consider a chess program that

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

Functional goal: winning

This program does not reliably win.



Software dysfunction

But some software is better than others.

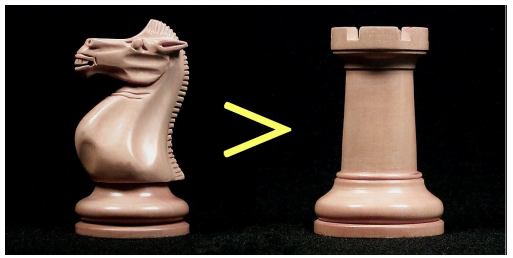
Consider a chess program that

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

Functional goal: winning

This program does not reliably win.

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



Software dysfunction

But some software is better than others.

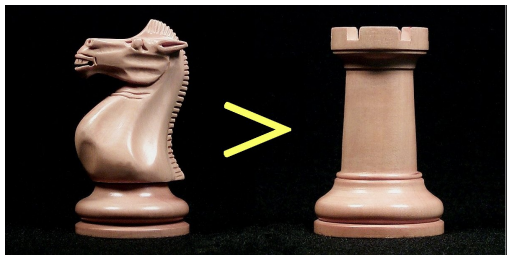
Consider a chess program that

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

Functional goal: winning

This program does not reliably win.

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



Software is capable of *type dysfunction*.

Outline

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

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.

Misfunction

Type dysfunction

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

Type misfunction

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

Misfunction

Type dysfunction

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

Type misfunction

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

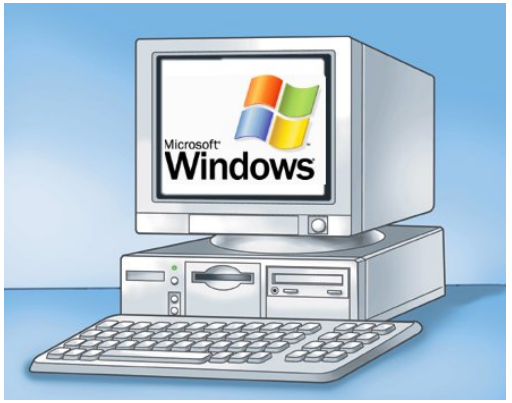
Token misfunction

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

Misfunctioning software

Type misfunction

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



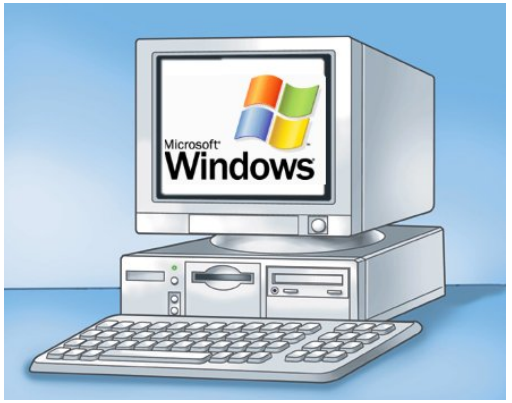
Examples:

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

Misfunctioning software

Type misfunction

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



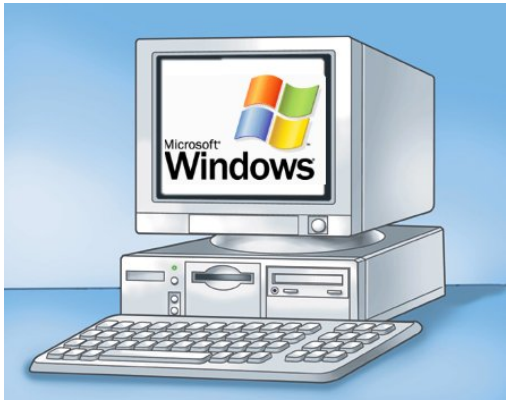
Examples:

- Misleading interfaces
  `Anna.Kournikova.jpg.vbs` appears as `Anna.Kournikova.jpg`
- Security flaws
  The ping of death

Misfunctioning software

Type misfunction

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



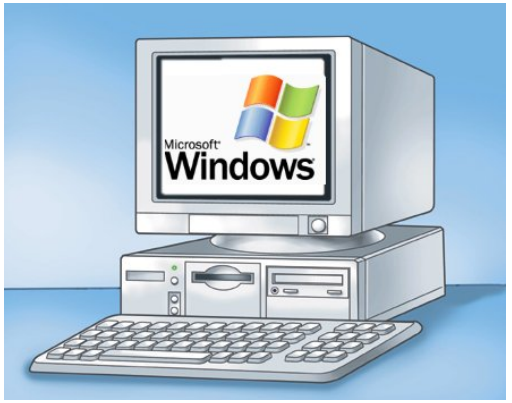
Examples:

- Misleading interfaces
  `Anna.Kournikova.jpg.vbs` appears as `Anna.Kournikova.jpg`
- Security flaws
  The ping of death. . . now dysfunction!

Misfunctioning software

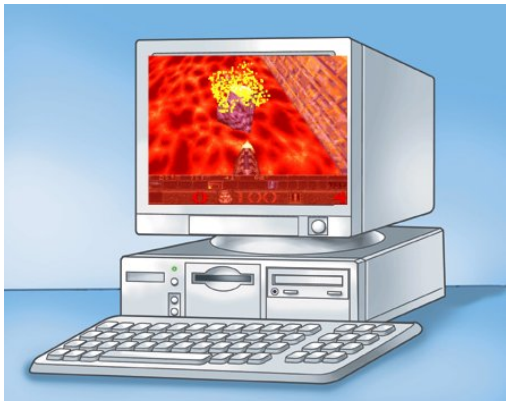
Type misfunction

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



Examples:

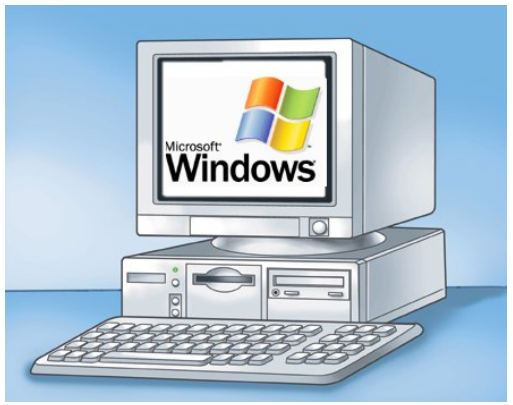
- Misleading interfaces
  `Anna.Kournikova.jpg.vbs` appears as `Anna.Kournikova.jpg`
- Security flaws
  The ping of death. . . now dysfunction!
- Unrealistic game physics
  Strafe jumping



Misfunctioning software

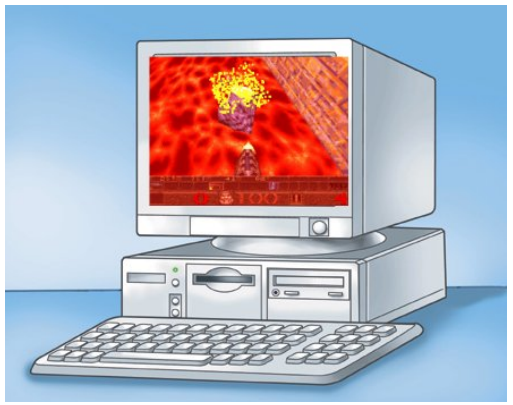
Type misfunction

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!



Malfunction: the old picture

"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)

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

- 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)

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

- 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)

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

- 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)

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

- Applies to tokens only
  . . . hence not to software.
- Strong dysfunction only
  . . . software dysfunction is weak.
- No misfunction

Malfunction: the old picture

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

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

- Applies to tokens only
  . . . hence not to software.
- Strong dysfunction only
  . . . software dysfunction is weak.
- No misfunction
  . . . missing most software bugs.

Malfunction: the new picture

	Dysfunction		Misfunction	
	token	**type**	**token**	**type**
Artifact				
Biological				
Software				

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

Malfunction: the new picture

	Dysfunction		Misfunction	
	token	**type**	**token**	**type**
Artifact	yes	yes	yes	yes
Biological				
Software				

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

Malfunction: the new picture

	Dysfunction		Misfunction	
	token	**type**	**token**	**type**
Artifact	yes	yes	yes	yes
Biological	yes	no?	yes?	yes?
Software				

Note: Outstanding issues with biological type malfunction!

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

Malfunction: the new picture

	Dysfunction		Misfunction	
	token	**type**	**token**	**type**
Artifact	yes	yes	yes	yes
Biological	yes	no?	yes?	yes?
Software	no	yes	no	yes

Note: Outstanding issues with biological type malfunction!

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

Thank you!