Of Functions and Means: An exercise in applied logic

Jesse Hughes

Eindhoven University of Technology

February 17, 2007

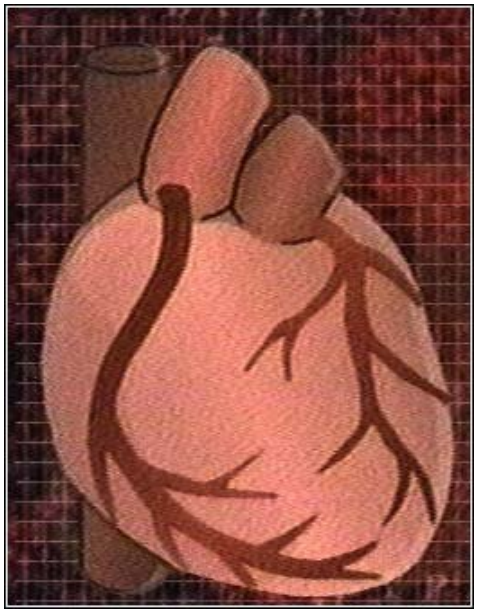
Outline

Outline

Outline

Outline

Functional ascriptions

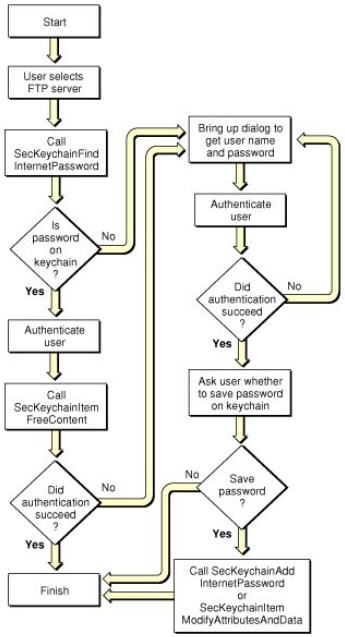


- "The function of the heart is to pump blood."

Functional ascriptions

- "The function of the heart is to pump blood."
- "That button turns on the television."

Functional ascriptions

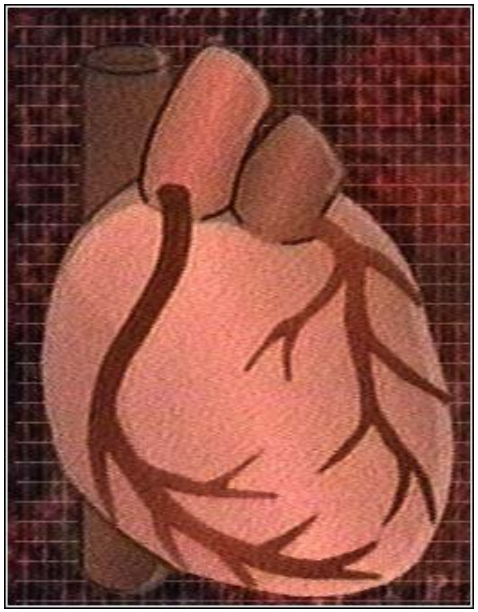


- "The function of the heart is to pump blood."
- "That button turns on the television."
- "The subroutine ensures that the user is authorized."

Functional ascriptions

- "The function of the heart is to pump blood."
- "That button turns on the television."
- "The subroutine ensures that the user is authorized."
- "The policeman is for directing traffic."

Functional ascriptions



- "The function of the heart is to pump blood."
- "That button turns on the television."
- "The subroutine ensures that the user is authorized."
- "The policeman is for directing traffic."

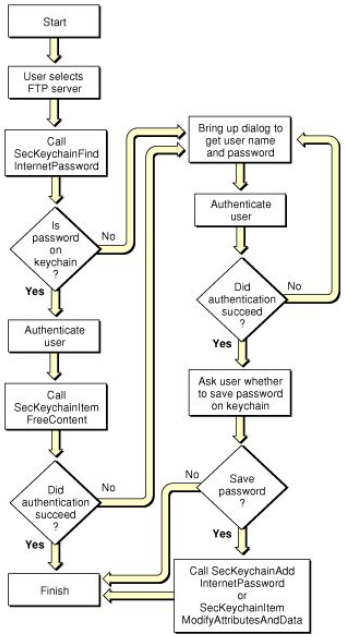
We ascribe functions to biological stuff,

Functional ascriptions

- "The function of the heart is to pump blood."
- "That button turns on the television."
- "The subroutine ensures that the user is authorized."
- "The policeman is for directing traffic."

We ascribe functions to biological stuff, artifacts,

Functional ascriptions



- "The function of the heart is to pump blood."
- "That button turns on the television."
- "The subroutine ensures that the user is authorized."
- "The policeman is for directing traffic."

We ascribe functions to biological stuff, artifacts, algorithms,

Functional ascriptions

- "The function of the heart is to pump blood."
- "That button turns on the television."
- "The subroutine ensures that the user is authorized."
- "The policeman is for directing traffic."

We ascribe functions to biological stuff, artifacts, algorithms, personal roles. . .

Functional ascriptions

- "The function of the heart is to pump blood."
- "That button turns on the television."
- "The subroutine ensures that the user is authorized."
- "The policeman is for directing traffic."

We ascribe functions to biological stuff, artifacts, algorithms, personal roles... but is one notion of function enough?

So what are functions anyway?

- A function explains a system's capacity.

So what are functions anyway?

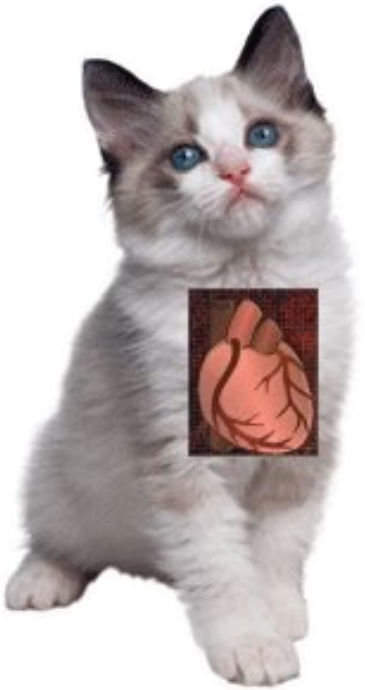
- A function explains a system's capacity.
  - The heart is a pump in the circulatory system.

So what are functions anyway?

- A function explains a system's capacity.
  - The heart is a pump in the circulatory system.
  - The remote has a function in the man-remote-TV system.

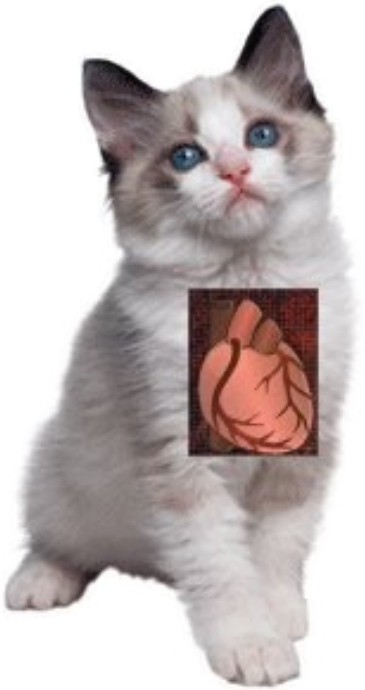
So what are functions anyway?

- A function explains a system's capacity.
  - The heart is a pump in the circulatory system.
  - The remote has a function in the man-remote-TV system.
- A function explains an item's presence.



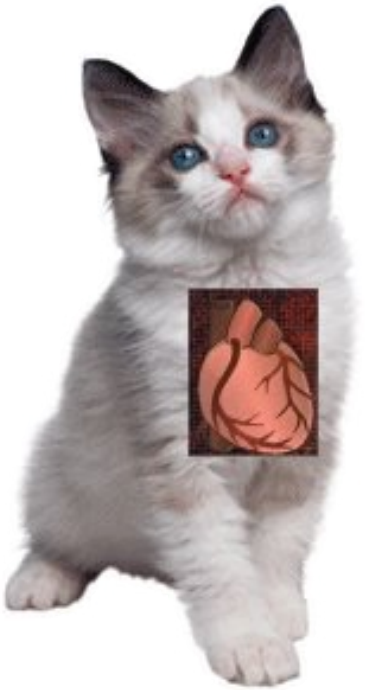
So what are functions anyway?

- A function explains a system's capacity.
  - The heart is a pump in the circulatory system.
  - The remote has a function in the man-remote-TV system.
- A function explains an item's presence.
  - The heart is there because pumping blood is advantageous to cats.



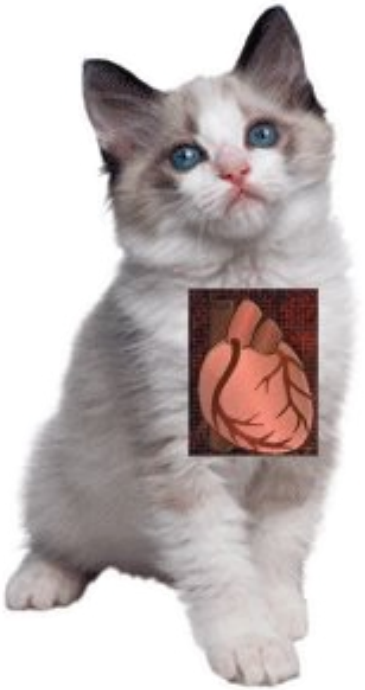
So what are functions anyway?

- A function explains a system's capacity.
  - The heart is a pump in the circulatory system.
  - The remote has a function in the man-remote-TV system.
- A function explains an item's presence.
  - The heart is there because pumping blood is advantageous to cats.
  - The remote was created to change channels.



So what are functions anyway?

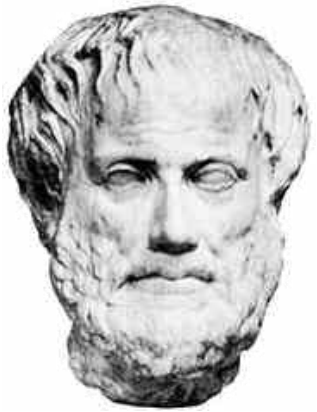
- A function explains a system's capacity.
  - The heart is a pump in the circulatory system.
  - The remote has a function in the man-remote-TV system.
- A function explains an item's presence.
  - The heart is there because pumping blood is advantageous to cats.
  - The remote was created to change channels.
  - The remote was created because previous remotes sold well.



Functions do more than explain. . .

Functions explain stuff.

- How parts contribute to the whole.
- How things came to be as they are.

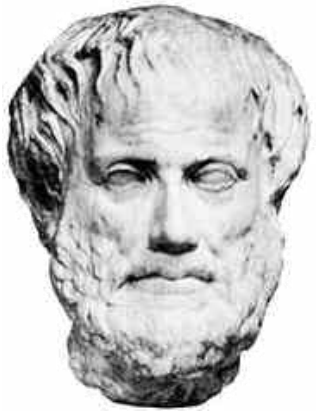


Functions do more than explain. . .

Functions explain stuff.

- How parts contribute to the whole.
- How things came to be as they are.

Philosophers like explaining stuff.



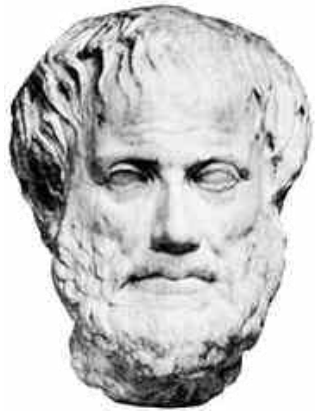
Functions do more than explain. . .

Functions explain stuff.

- How parts contribute to the whole.
- How things came to be as they are.

Philosophers like explaining stuff.

Users want to know how to use artifacts.

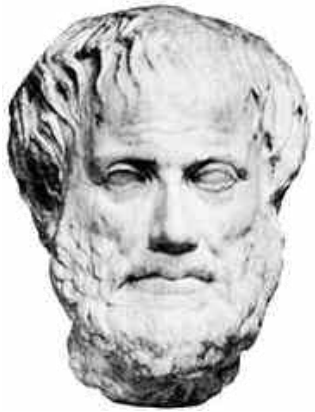


Functions do more than explain. . .

Functions explain stuff.

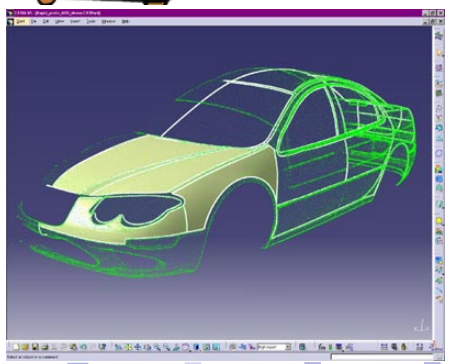
- How parts contribute to the whole.
- How things came to be as they are.

Philosophers like explaining stuff.



Users want to know how to use artifacts.

Engineers want their creations to be used.

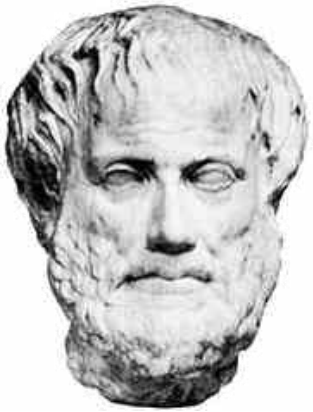


Functions do more than explain. . .

Functions explain stuff.

- How parts contribute to the whole.
- How things came to be as they are.

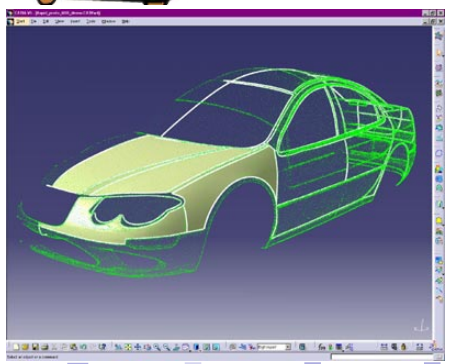
Philosophers like explaining stuff.



Users want to know how to use artifacts.

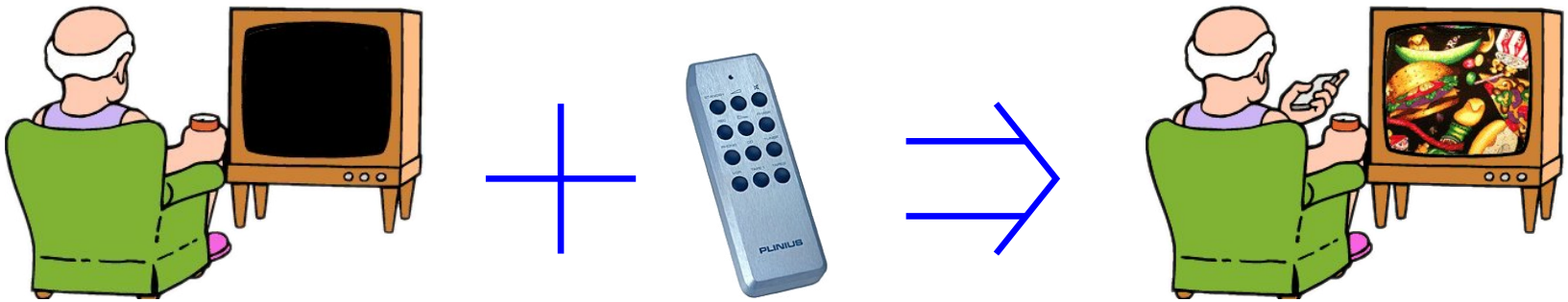
Engineers want their creations to be used.

Artifact functions have practical importance.



Functions have instrumental consequences

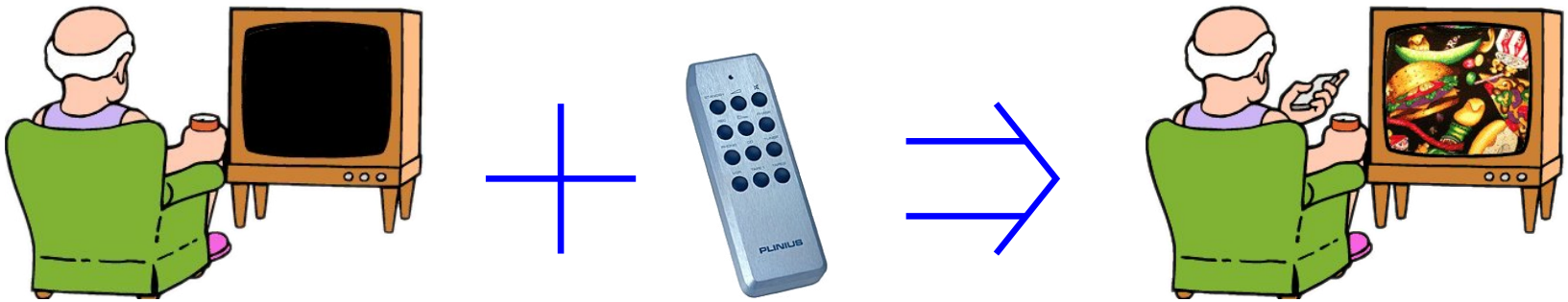
Artifactual knowledge produces practical knowledge.



desire + function ⇒ action

Functions have instrumental consequences

Artifactual knowledge produces practical knowledge.

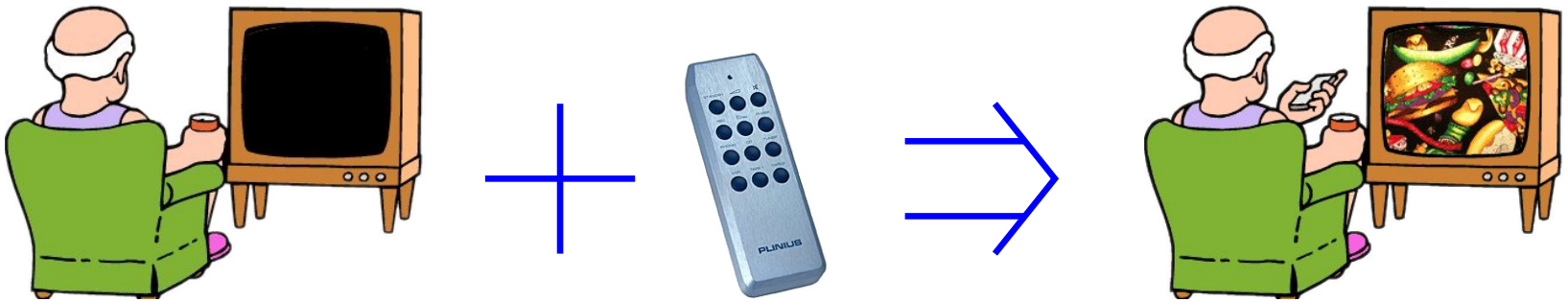


desire function action

Functions yield means-end relations!

Functions have instrumental consequences

Artifactual knowledge produces practical knowledge.



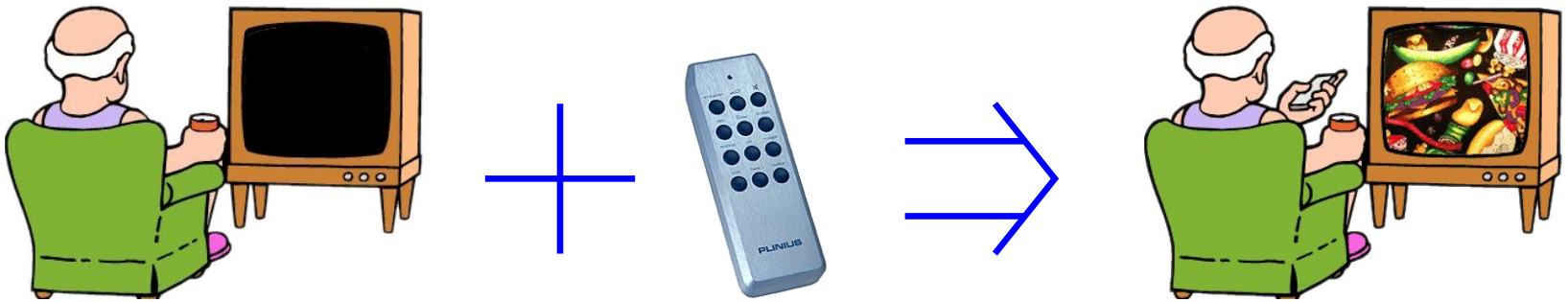
desire function action

Functions yield means-end relations!

Means-end relations can be given a formal semantics.

Functions have instrumental consequences

Artifactual knowledge produces practical knowledge.



desire function action

Functions yield means-end relations!

Means-end relations can be given a formal semantics.

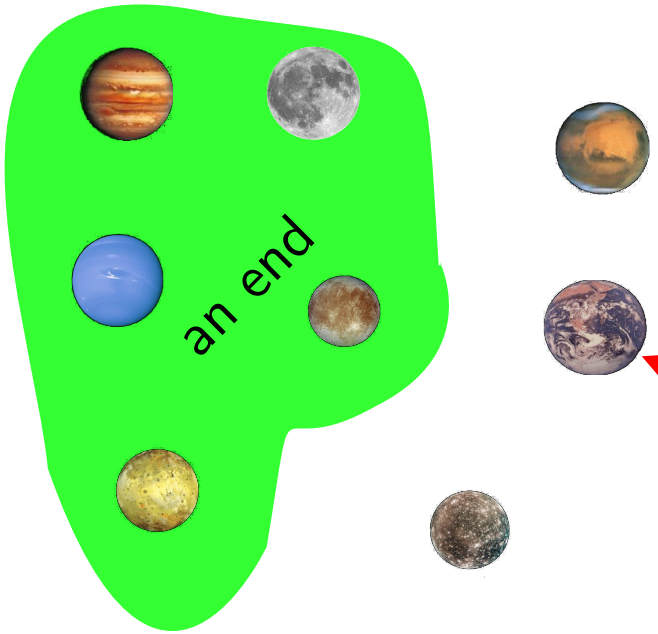
Use this to understand certain function talk: malfunction, failure, efficacy, etc.

Outline

Conceptual starting points

- An end is a condition to be realized.

Conceptual starting points

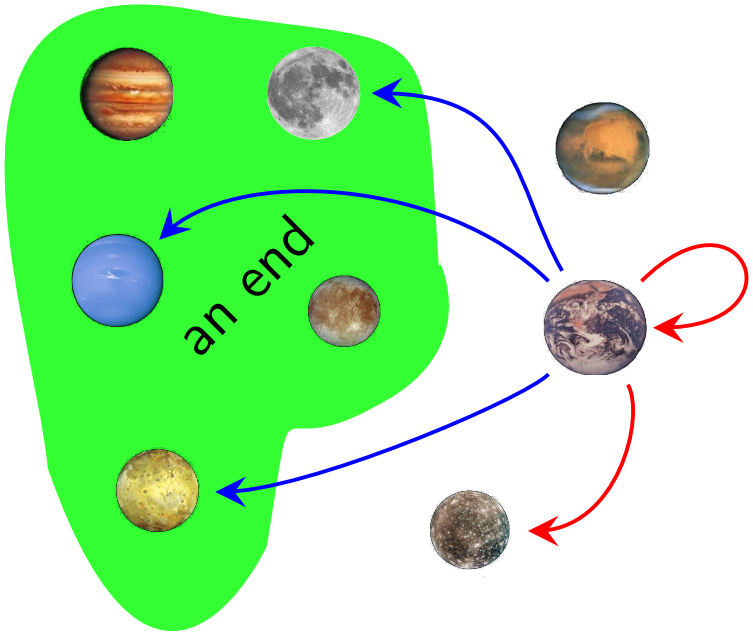


- An end is a condition to be realized.

You are here.

Think possible worlds!

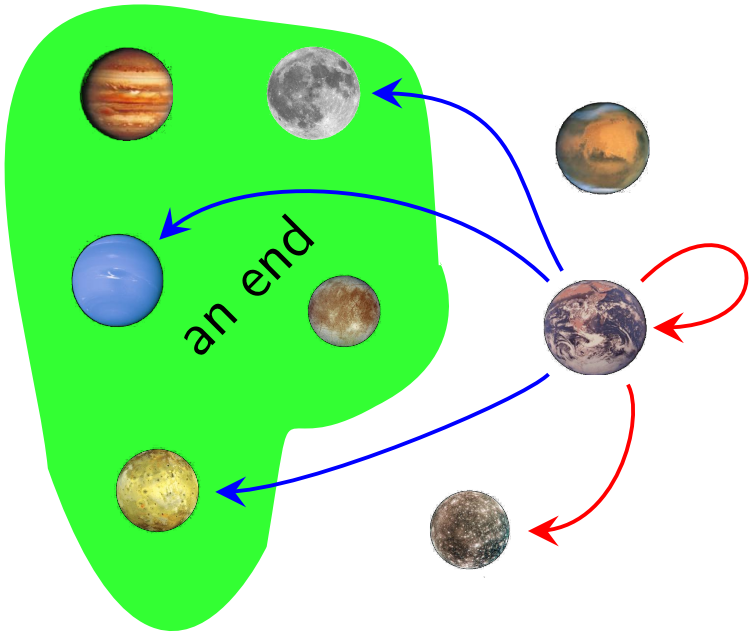
Conceptual starting points



Think possible worlds!
Think transitions!

- An end is a condition to be realized.
- A means is a way of realizing the condition.

Conceptual starting points



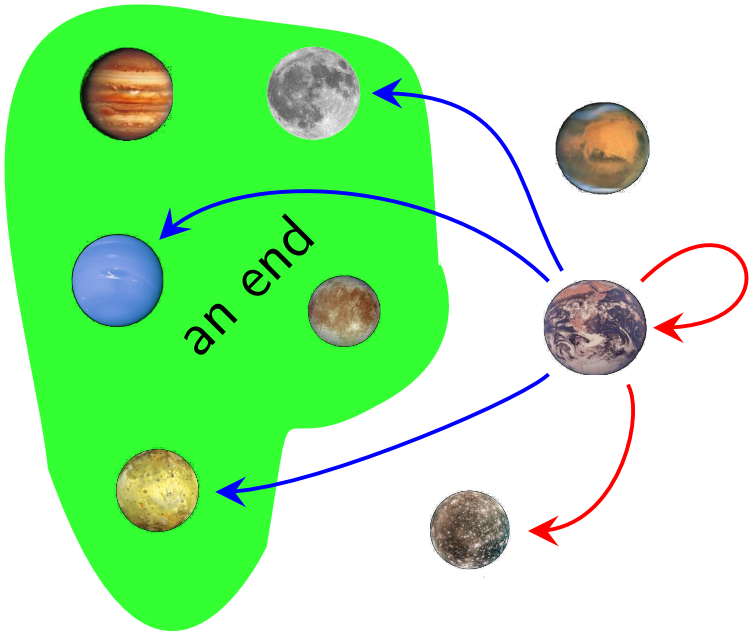
Think possible worlds!
Think transitions!

- An end is a condition to be realized.
- A means is a way of realizing the condition.

Thus:

- an end is a formula;

Conceptual starting points



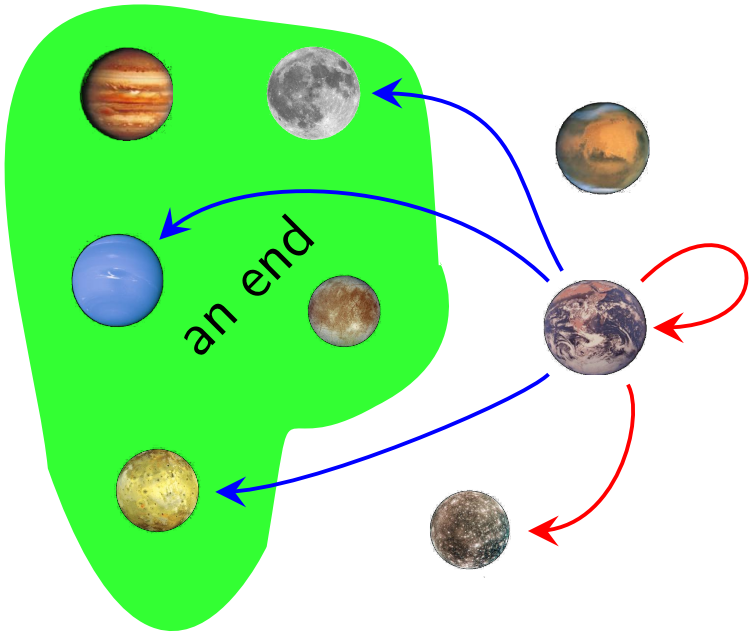
Think possible worlds!
Think transitions!

- An end is a condition to be realized.
- A means is a way of realizing the condition.

Thus:

- an end is a formula;
- a means is an action;

Conceptual starting points



Think possible worlds!
Think transitions!

- An end is a condition to be realized.
- A means is a way of realizing the condition.

Thus:

- an end is a formula;
- a means is an action;
- Propositional Dynamic Logic is a natural setting.

PDL syntax

Propositional Dynamic Logic is a logic of actions.

PDL syntax

Propositional Dynamic Logic is a logic of actions.

Basic types:

- a set **act** of *actions*,

PDL syntax

Propositional Dynamic Logic is a logic of actions.

Basic types:

- a set **act** of *actions*,
  - Closed under:
    - *sequential composition* $\alpha;\beta$
    - *non-deterministic choice* $\alpha \cup \beta$

PDL syntax

Propositional Dynamic Logic is a logic of actions.

Basic types:

- a set **act** of *actions*,
  - Closed under:
    - *sequential composition* $\alpha;\beta$
    - *non-deterministic choice* $\alpha \cup \beta$
- a set **prop** of *propositions*.

PDL syntax

Propositional Dynamic Logic is a logic of actions.

Basic types:

- a set **act** of *actions*,
  - Closed under:
    - *sequential composition* $\alpha;\beta$
    - *non-deterministic choice* $\alpha \cup \beta$
- a set **prop** of *propositions*.
  - Closed under:
    - boolean connectives,
    - dynamic operators $[\alpha]\varphi$, $\langle\alpha\rangle\varphi$.

PDL syntax

Propositional Dynamic Logic is a logic of actions.

Basic types:

- a set **act** of *actions*,
  - Closed under:
    - *sequential composition* $\alpha;\beta$
    - *non-deterministic choice* $\alpha \cup \beta$
- a set **prop** of *propositions*.
  - Closed under:
    - boolean connectives,
    - dynamic operators $[\alpha]\varphi$, $\langle\alpha\rangle\varphi$.

Intuitions:

- $[\alpha]\varphi$: after doing α, φ *will* hold.

PDL syntax

Propositional Dynamic Logic is a logic of actions.

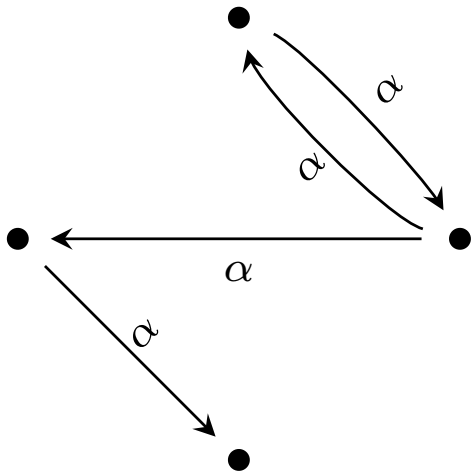
Basic types:

- a set **act** of *actions*,
  - Closed under:
    - *sequential composition* $\alpha;\beta$
    - *non-deterministic choice* $\alpha \cup \beta$
- a set **prop** of *propositions*.
  - Closed under:
    - boolean connectives,
    - dynamic operators $[\alpha]\varphi$, $\langle\alpha\rangle\varphi$.

Intuitions:

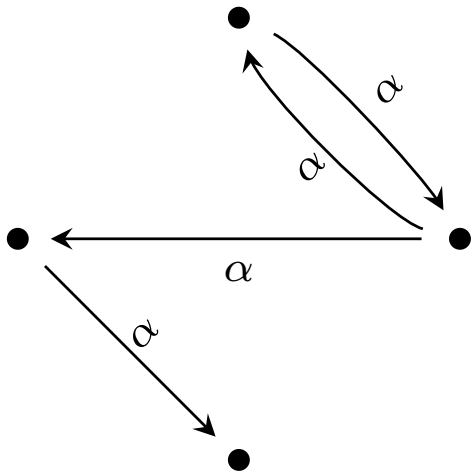
- $[\alpha]\varphi$: after doing α, φ *will* hold.
- $\langle\alpha\rangle\varphi$: after doing α, φ *might* hold.

PDL semantics



Possible world semantics with transition systems for each action α.

PDL semantics

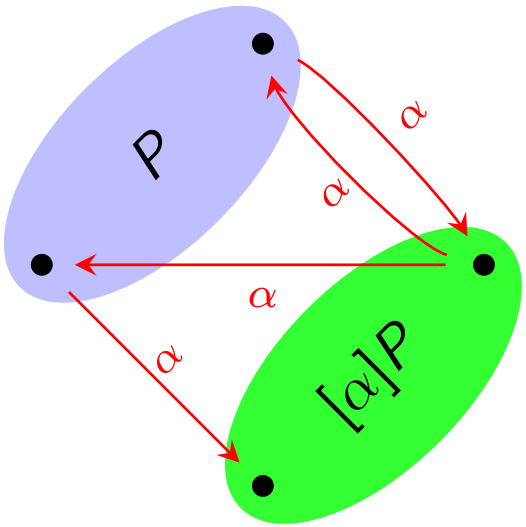


Possible world semantics with transition systems for each action α.

$w \xrightarrow{\alpha} w'$ means:

one can reach w' by doing α in w.

PDL semantics



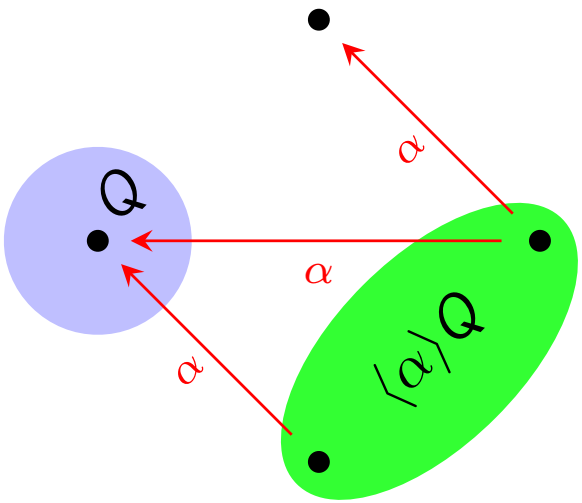
Possible world semantics with transition systems for each action α.

$w \xrightarrow{\alpha} w'$ means:

one can reach w' by doing α in w.

$$w \models [\alpha]\varphi \quad \textit{iff} \quad \forall w \xrightarrow{\alpha} w' \;.\; w' \models \varphi.$$

PDL semantics



Possible world semantics with transition systems for each action α.

$w \xrightarrow{\alpha} w'$ means:

one can reach w' by doing α in w.

$$w \models [\alpha]\varphi \quad \textit{iff} \quad \forall w \xrightarrow{\alpha} w' \;.\; w' \models \varphi.$$

$$w \models \langle\alpha\rangle\varphi \quad \textit{iff} \quad \exists w \xrightarrow{\alpha} w' \;.\; w' \models \varphi.$$

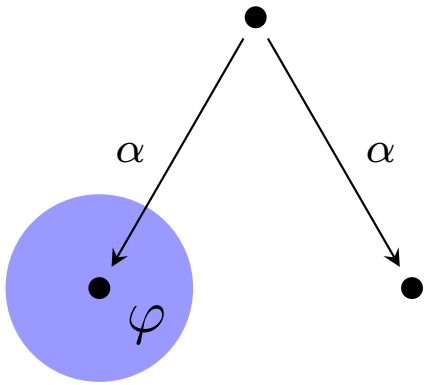
Weakly and strongly sufficient means

A sufficient means is an action α that can realize one's end φ.

Weakly and strongly sufficient means

A sufficient means is an action α that can realize one's end φ.

Two interpretations:

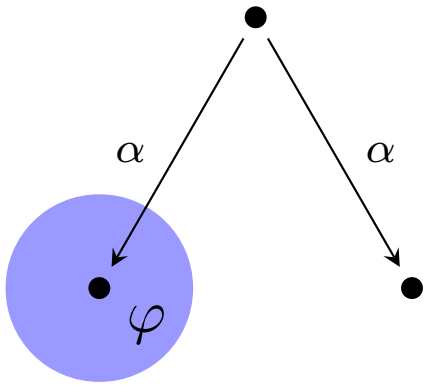


Weak: α *might* realize φ.

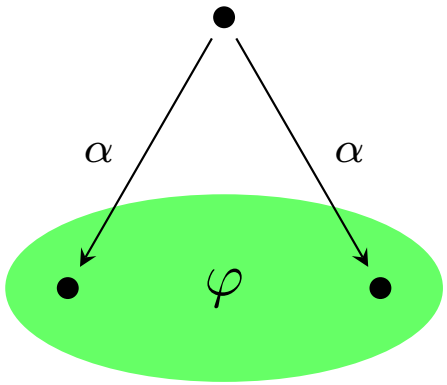
Weakly and strongly sufficient means

A sufficient means is an action α that can realize one's end φ.

Two interpretations:



Weak: α *might* realize φ.

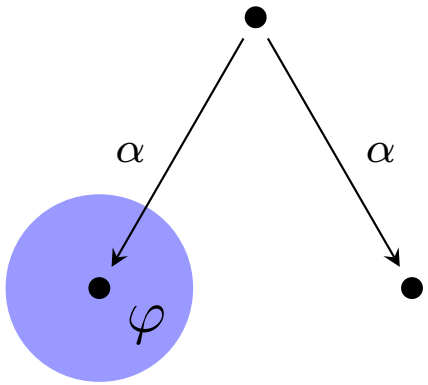


Strong: α *will* realize φ.

Weakly and strongly sufficient means

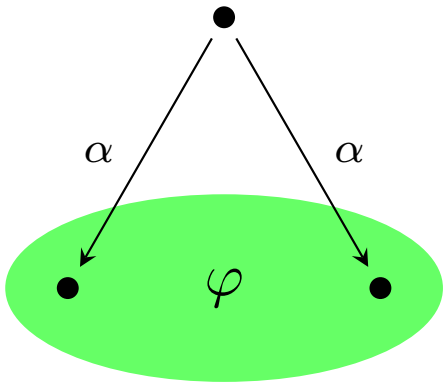
A sufficient means is an action α that can realize one's end φ.

Two interpretations:



Weak: α *might* realize φ.

$w \models \langle\alpha\rangle\varphi$



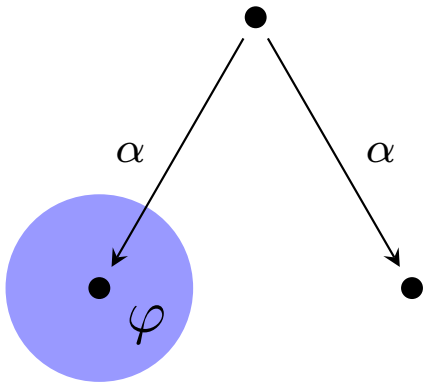
Strong: α *will* realize φ.

$w \models [\alpha]\varphi \ \wedge \ \langle\alpha\rangle\top$

Weakly and strongly sufficient means

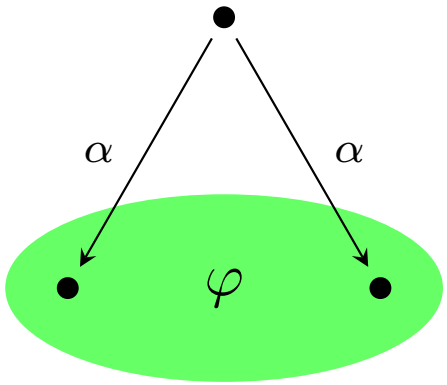
A sufficient means is an action α that can realize one's end φ.

Two interpretations:



Weak: α *might* realize φ.

$w \models \langle\alpha\rangle\varphi$



Strong: α *will* realize φ.

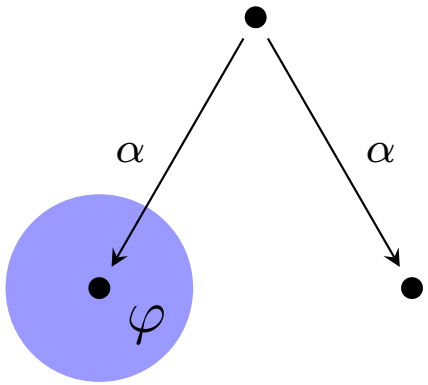
$w \models [\alpha]\varphi \wedge \underbrace{\langle\alpha\rangle\top}$

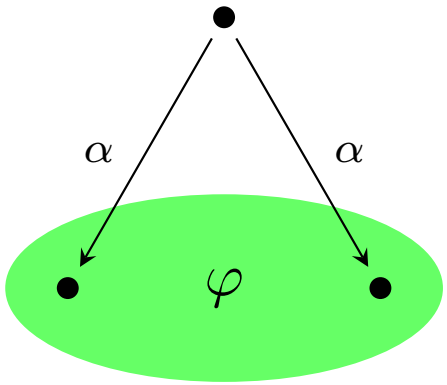
α can be done.

Weakly and strongly sufficient means

A sufficient means is an action α that can realize one's end φ.

Two interpretations:





Weak: α *might* realize φ.

$w \models \langle\alpha\rangle\varphi$

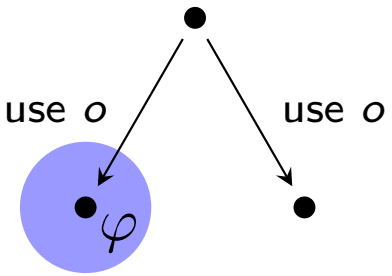
Strong: α *will* realize φ.

$w \models [\alpha]\varphi \wedge \underbrace{\langle\alpha\rangle\top}$

α can be done.

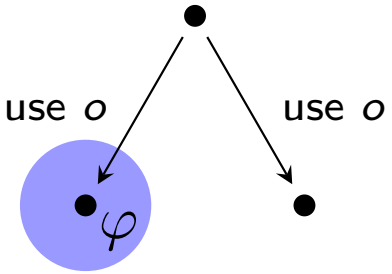
Caveat: This definition omits relevance.

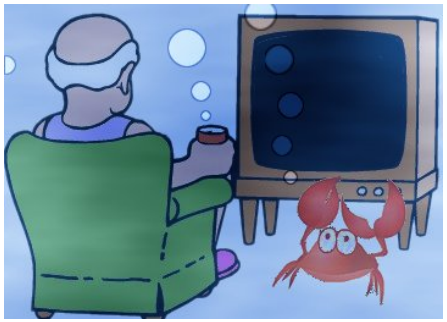
From functions to means: the proposal



If the function of o is to bring about φ, then "use o" is a weak means to φ...

From functions to means: the proposal



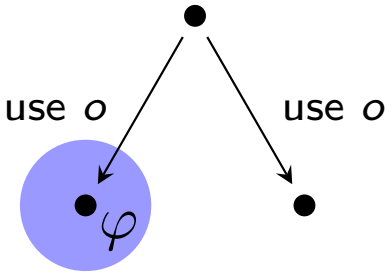


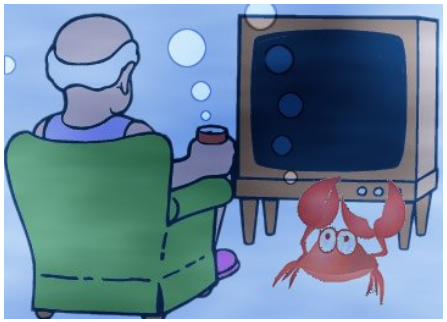
If the function of o is to bring about φ, then "use o" is a weak means to φ...

...but not always!

Sometimes a good remote does you no good...

From functions to means: the proposal





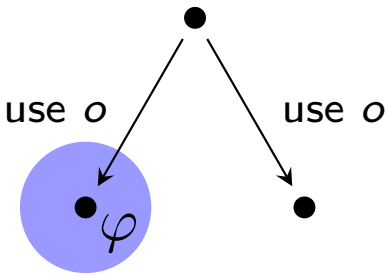
If the function of o is to bring about φ, then "use o" is a weak means to φ...

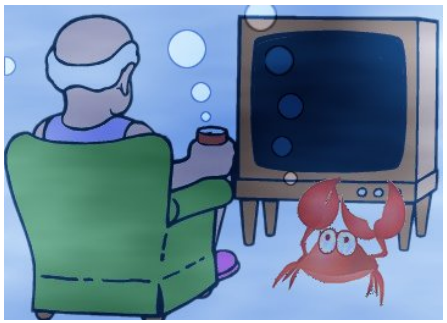
...but not always!

Sometimes a good remote does you no good...

...or maybe this remote is broken.

From functions to means: the proposal





If the function of o is to bring about φ, then "use o" is a weak means to φ...

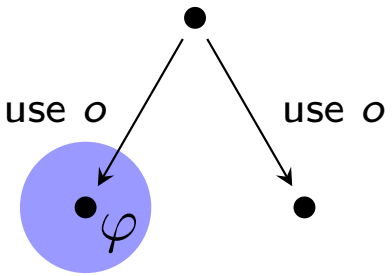
...but not always!

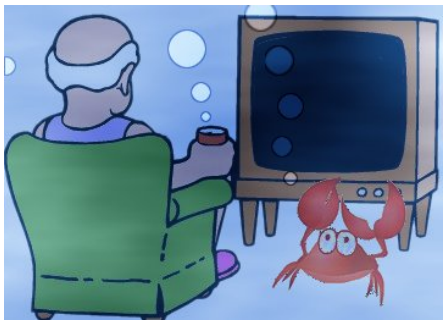
Sometimes a good remote does you no good...

...or maybe this remote is broken.

How do we go from function statements to dynamic models?

From functions to means: the proposal





If the function of o is to bring about φ, then "use o" is a weak means to φ...

...but not always!

Sometimes a good remote does you no good...

...or maybe this remote is broken.

How do we go from function statements to dynamic models?

Context matters!

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?

When/where do you use it?

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?

When/where do you use it?

How do you use it?

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?

When/where do you use it?

How do you use it?

What should it do?

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?	Artifact type
When/where do you use it?	Context specification
How do you use it?	Use plan
What should it do?	Goal

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?	Artifact type
When/where do you use it?	Context specification
How do you use it?	Use plan
What should it do?	Goal

Common features

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?	Artifact type
When/where do you use it?	Context specification
How do you use it?	Use plan ← Houkes/Vermaas
What should it do?	Goal

Lesser known features of function ascriptions

"That button turns on the television."

No context apparent there! Just a type and an end.

Useful functional knowledge includes:

What artifacts?	Artifact type
When/where do you use it?	Context specification
How do you use it?	Use plan
What should it do?	Goal

Original

How functions induce means-end relations

Given a functional ascription:

- Artifact type T
- Context specification C
- Use plan α
- Goal φ

How functions induce means-end relations

Given a functional ascription:

- Artifact type T
- Context specification C
- Use plan α
- Goal φ

We expect that: In C-contexts, using a token of type T according to plan α is a means to φ.

How functions induce means-end relations

Given a functional ascription:

- Artifact type T
- Context specification C
- Use plan α
- Goal φ

We expect that: In C-contexts, using a token of type T according to plan α is a means to φ.

This expectation is general. Success or failure comes in particular applications.

Particular uses and their models

Application: using a token $o \in T$ in a situation $c \in C$.

Particular uses and their models

Application: using a token $o \in T$ in a situation $c \in C$.

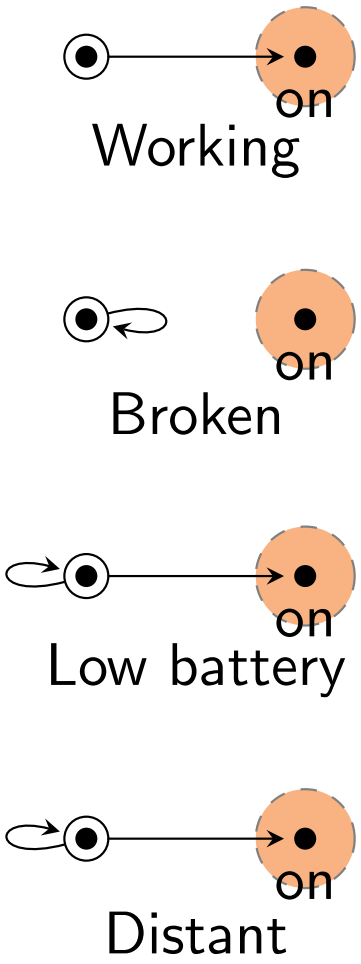
o and c provide parameters for α and φ.

Particular uses and their models

Application: using a token $o \in T$ in a situation $c \in C$.

o and c provide parameters for α and φ.

From this data, we build dynamic models.



Particular uses and their models

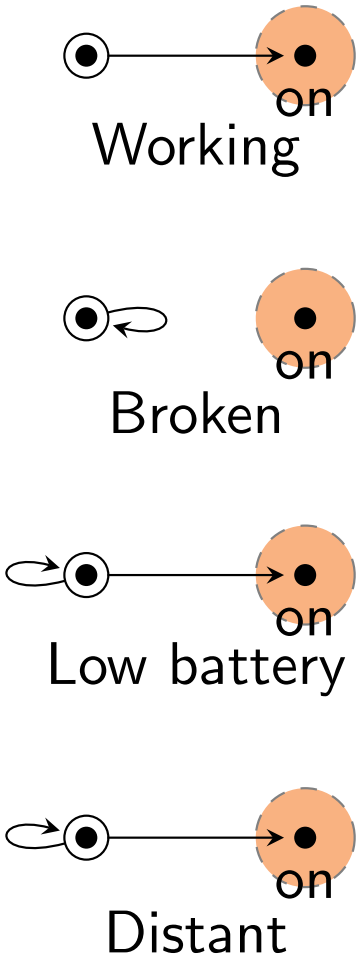
Application: using a token $o \in T$ in a situation $c \in C$.

o and c provide parameters for α and φ.

From this data, we build dynamic models.

function knowledge $\Rightarrow$ practical expectations

particular uses $\Rightarrow$ success/failure



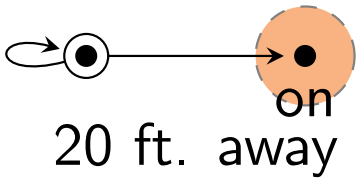
Outline

Limitations of non-determinism

But non-deterministic models only go so far.

Suppose our remote works up to 40 ft.

Limitations of non-determinism

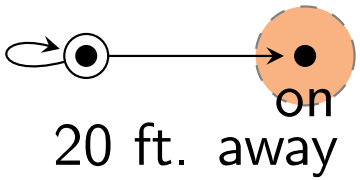


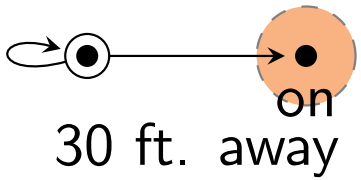
But non-deterministic models only go so far.

Suppose our remote works up to 40 ft.

At 20 ft., performance begins to degrade somewhat.

Limitations of non-determinism





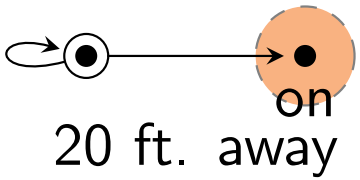
But non-deterministic models only go so far.

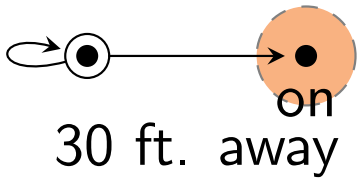
Suppose our remote works up to 40 ft.

At 20 ft., performance begins to degrade somewhat.

At 30 ft., the signal is weaker still and easier to miss.

Limitations of non-determinism





But non-deterministic models only go so far.

Suppose our remote works up to 40 ft.

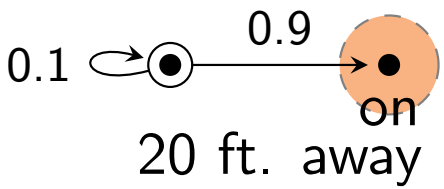
At 20 ft., performance begins to degrade somewhat.

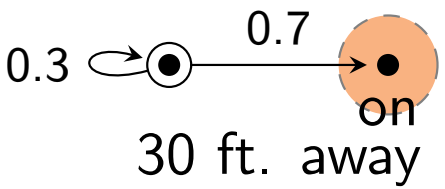
At 30 ft., the signal is weaker still and easier to miss.

At 20 ft., the remote is more reliable than at 30 ft.

Our non-deterministic models can't distinguish these situations.

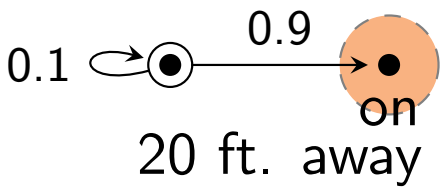
Adding probabilities to dynamic logic

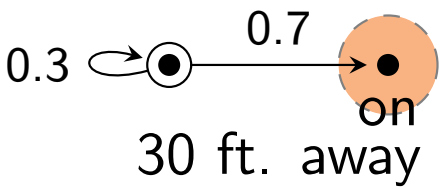




We need a probabilistic transition system.

Adding probabilities to dynamic logic

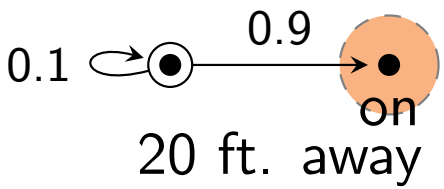


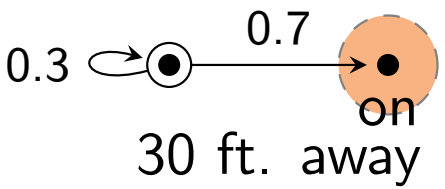


We need a probabilistic transition system.

Requires: new interpretation for formulas
corresponding logic

Adding probabilities to dynamic logic



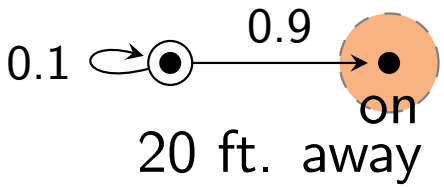


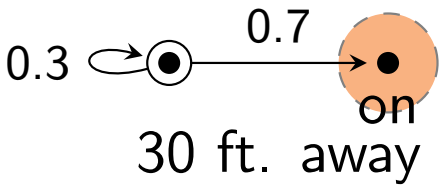
We need a probabilistic transition system.

Requires: new interpretation for formulas
corresponding logic

Intuitively: $\langle\alpha\rangle\varphi$ means "α is a reliable means to φ."

Adding probabilities to dynamic logic





We need a probabilistic transition system.

Requires: new interpretation for formulas
corresponding logic

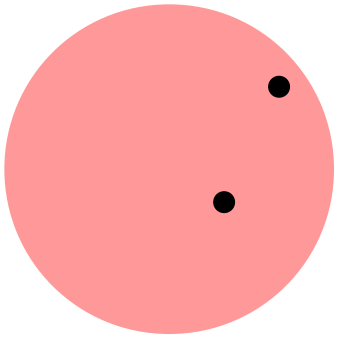
Intuitively: $\langle\alpha\rangle\varphi$ means "α is a reliable means to φ."

Fuzzy!

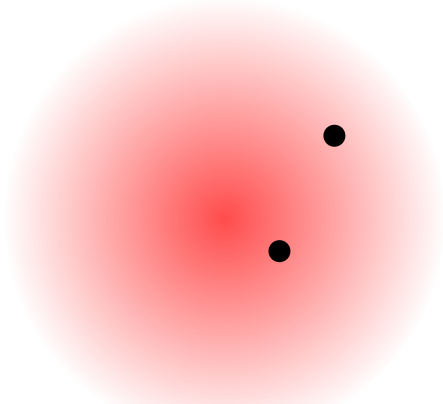
Reliability is vague.

Vague predicates can be modeled by fuzzy sets.

Fuzzy PDL



A crisp proposition

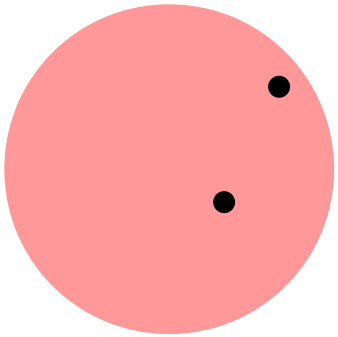


A fuzzy proposition

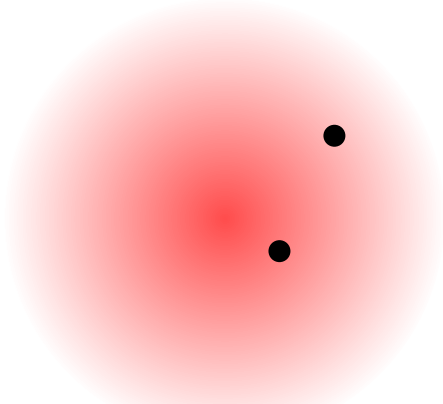
In fuzzy logic, a proposition may be more or less true.

Truth comes in degrees.

Fuzzy PDL



A crisp proposition

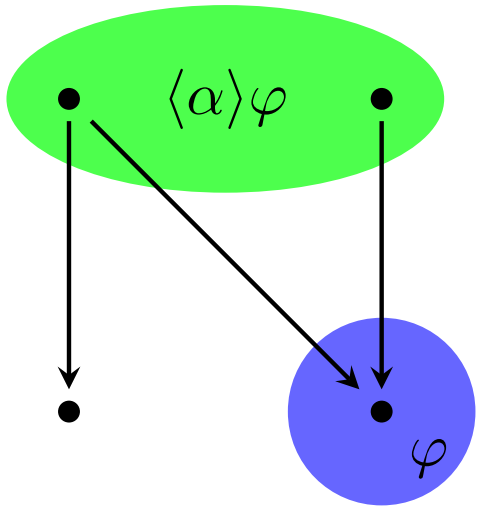


A fuzzy proposition

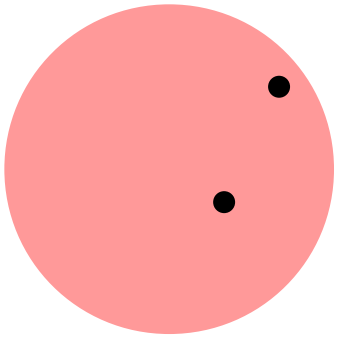
In fuzzy logic, a proposition may be more or less true.

Truth comes in degrees.

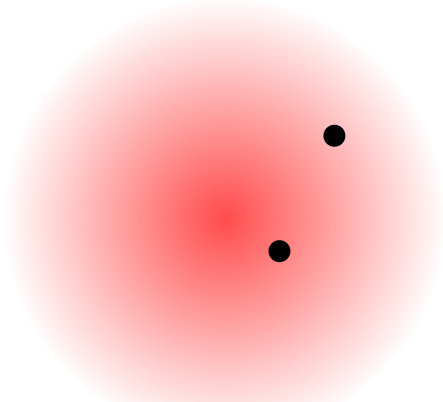
Crisp: In w, α is a means to φ



Fuzzy PDL



A crisp proposition



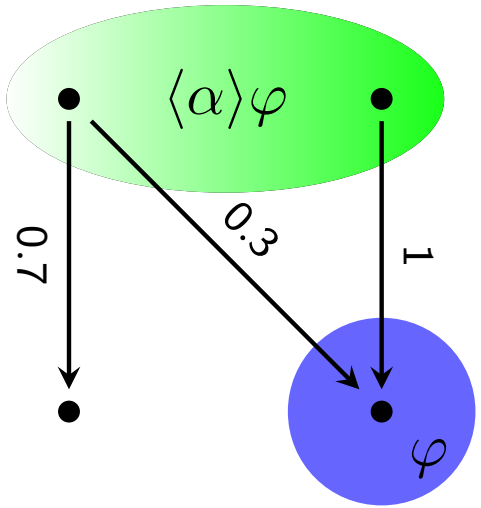
A fuzzy proposition

In fuzzy logic, a proposition may be more or less true.

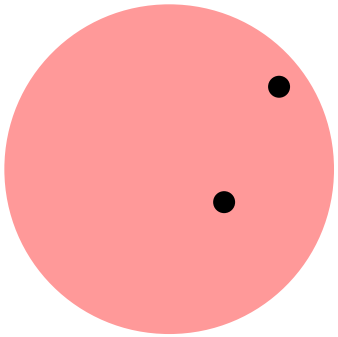
Truth comes in degrees.

Crisp: In w, α is a means to φ

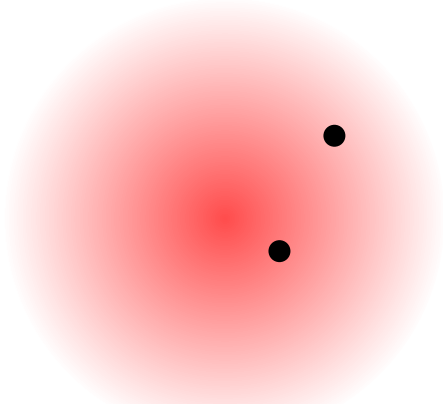
Fuzzy: In w, α is a reliable means to φ



Fuzzy PDL



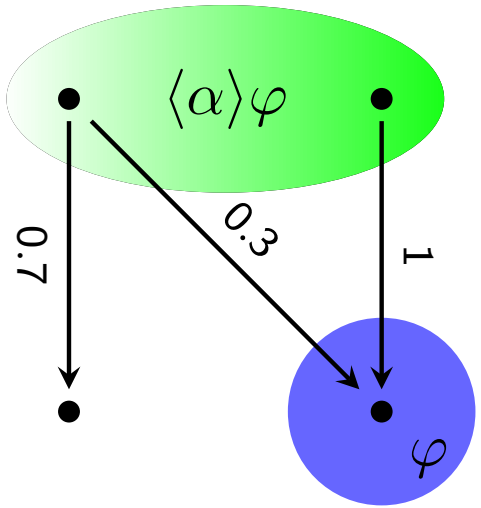
A crisp proposition



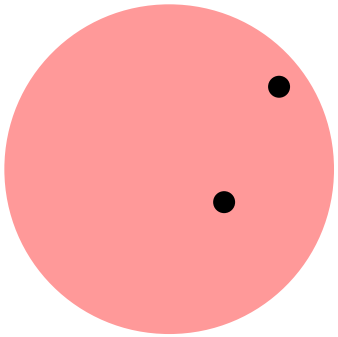
A fuzzy proposition

Fuzzy PDL: A fuzzy dynamic logic.

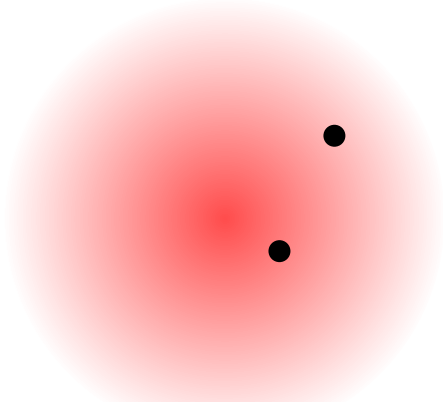
- Allows probabilistic transitions



Fuzzy PDL



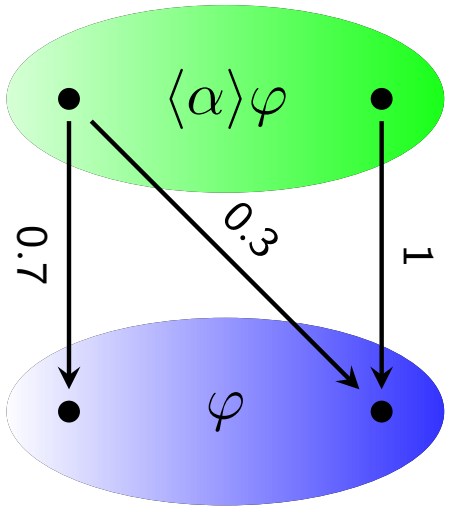
A crisp proposition



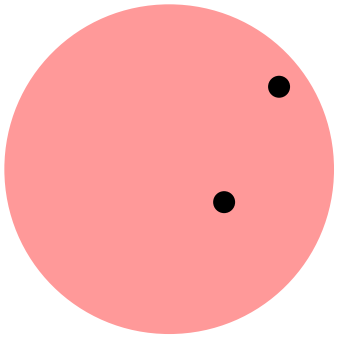
A fuzzy proposition

Fuzzy PDL: A fuzzy dynamic logic.

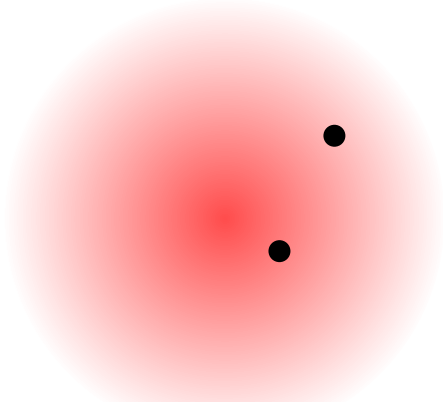
- Allows probabilistic transitions
- Allows fuzzy ends (bonus!)



Fuzzy PDL



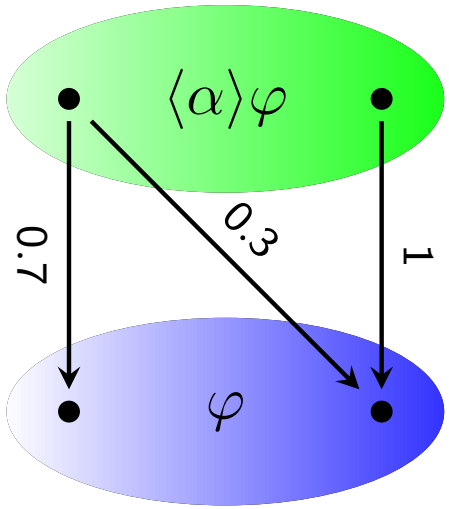
A crisp proposition



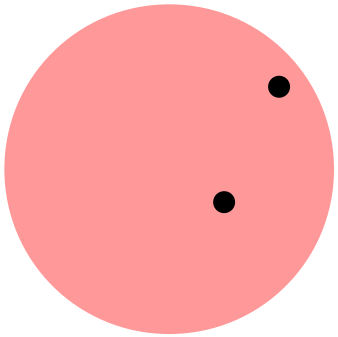
A fuzzy proposition

Fuzzy PDL: A fuzzy dynamic logic.

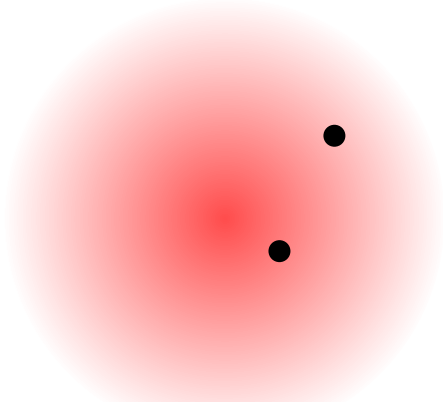
- Allows probabilistic transitions
- Allows fuzzy ends (bonus!)
- Defines efficacy for means to an end



Fuzzy PDL



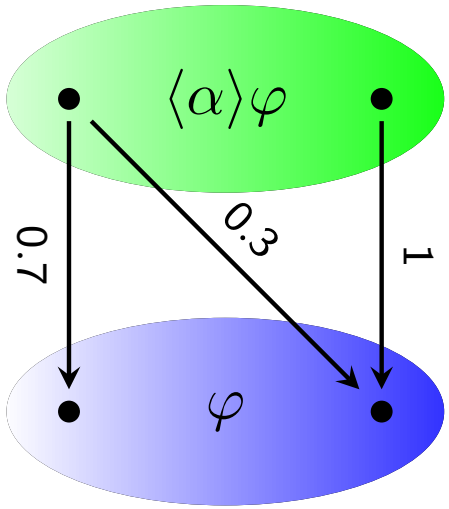
A crisp proposition



A fuzzy proposition

Fuzzy PDL: A fuzzy dynamic logic.

- Allows probabilistic transitions
- Allows fuzzy ends (bonus!)
- Defines efficacy for means to an end
- Allows performance comparisons for functions



Malfunction and failure

Definition: An application fails when the goal φ is not realized.

Malfunction and failure

Definition: An application fails when the goal φ is not realized.

Common definition: A token malfunctions when it cannot do what it is supposed to do.

Malfunction and failure

Definition: An application fails when the goal φ is not realized.

Common definition: A token malfunctions when it cannot do what it is supposed to do.

Unanalyzed!

Cannot do: possibly, probably, regularly?

Malfunction and failure

Definition: An application fails when the goal φ is not realized.

Common definition: A token malfunctions when it cannot do what it is supposed to do.

Unanalyzed!

Cannot do: possibly, probably, regularly?

Our definition: A token malfunctions when it is less reliable or effective in some contexts of use than a normal token of the same type.

Malfunction and failure

Definition: An application fails when the goal φ is not realized.

Common definition: A token malfunctions when it cannot do what it is supposed to do.

Unanalyzed!

Cannot do: possibly, probably, regularly?

Our definition: A token malfunctions when it is less reliable or effective in some contexts of use than a normal token of the same type.

Just as bad?

Malfunction and failure

Definition: An application fails when the goal φ is not realized.

Common definition: A token malfunctions when it cannot do what it is supposed to do.

Unanalyzed!

Cannot do: possibly, probably, regularly?

Our definition: A token malfunctions when it is less reliable or effective in some contexts of use than a normal token of the same type.

Just as bad?

Of course not!

A brief introduction to normality

Normal tokens are abstractions representing expectations.

- Users gain experience from use.

A brief introduction to normality

Normal tokens are abstractions representing expectations.

- Users gain experience from use.
- Users learn from technical manuals.

A brief introduction to normality

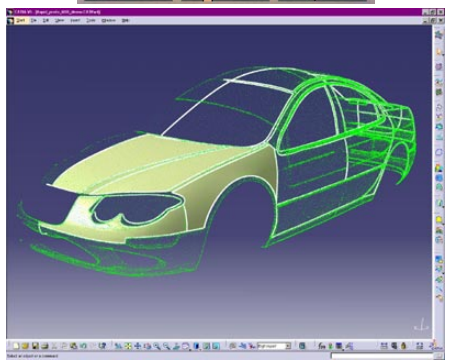
Normal tokens are abstractions representing expectations.

- Users gain experience from use.
- Users learn from technical manuals.
- Users infer the designers' intentions.

A brief introduction to normality

Normal tokens are abstractions representing expectations.

- Users gain experience from use.
- Users learn from technical manuals.
- Users infer the designers' intentions.
- Engineers have technical understanding.

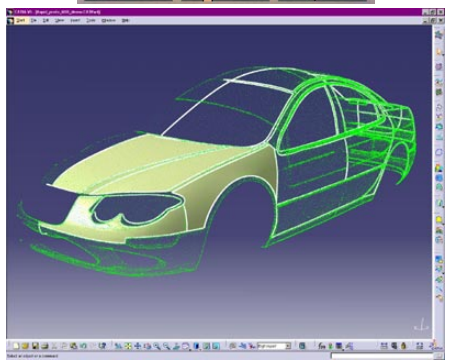


A brief introduction to normality

Normal tokens are abstractions representing expectations.

- Users gain experience from use.
- Users learn from technical manuals.
- Users infer the designers' intentions.
- Engineers have technical understanding.

This data creates expectations about how tokens ought to perform.



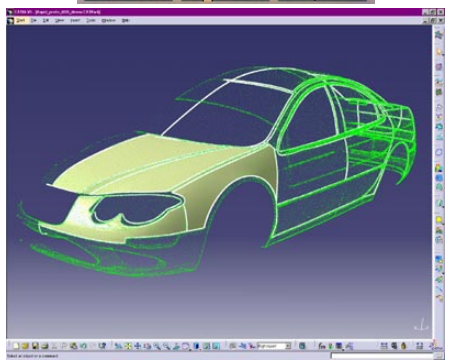
A brief introduction to normality

Normal tokens are abstractions representing expectations.

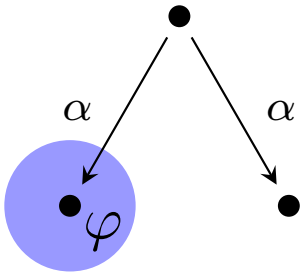
- Users gain experience from use.
- Users learn from technical manuals.
- Users infer the designers' intentions.
- Engineers have technical understanding.

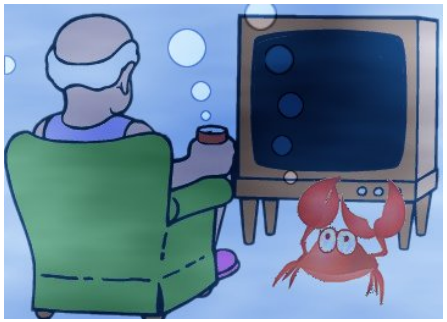
This data creates expectations about how tokens ought to perform.

Normal tokens are a useful device for presenting these expectations.



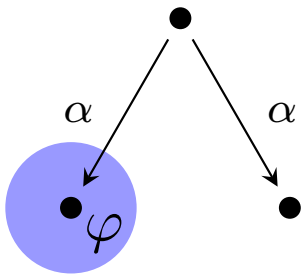
So what did we get?

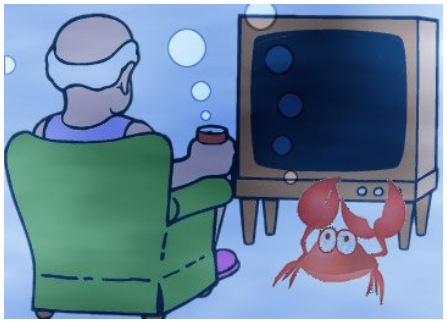




- The relation between functional knowledge and practical reasoning

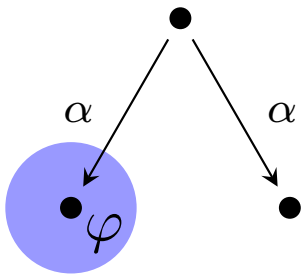
So what did we get?

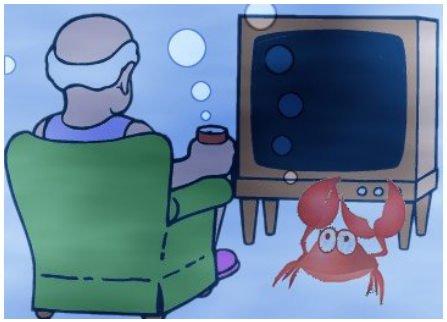




- The relation between functional knowledge and practical reasoning
- A semantics for means-end relations

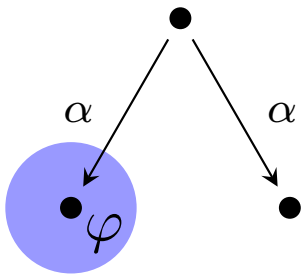
So what did we get?

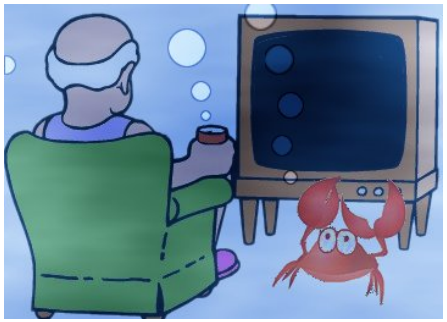




- The relation between functional knowledge and practical reasoning
- A semantics for means-end relations
- Fuzzy PDL, giving
  - Notion of efficacy of means
  - Representation of vague ends
  - Token-token, token-type and type-type comparisons for artifacts

So what did we get?





- The relation between functional knowledge and practical reasoning
- A semantics for means-end relations
- Fuzzy PDL, giving
  - Notion of efficacy of means
  - Representation of vague ends
  - Token-token, token-type and type-type comparisons for artifacts
- Distinction between malfunction and failure

Thank you!