Computational Semantics
Day 4: Extensionality and intensionality

Jan van Eijck[1] & Christina Unger[2]

[1]CWI, Amsterdam, and UiL-OTS, Utrecht, The Netherlands
[2]CITEC, Bielefeld University, Germany

ESSLLI 2011, Ljubljana

The picture

string $\longrightarrow$ tree structure $\longrightarrow$ meaning representation $\longrightarrow$ evaluation in a model

The picture

string $\longrightarrow$ tree structure $\longrightarrow$ meaning representation $\longrightarrow$ evaluation in a model

Yesterday:

- How to systematically map tree structures to meaning representations

The picture

string $\longrightarrow$ tree structure $\longrightarrow$ meaning representation $\longrightarrow$ evaluation in a model

Today:

- Why the meanings we constructed up to now don't always suffice

Outline

1. Meaning as reference

2. Meaning as intension
   - Types and model structures
   - Meaning w.r.t. possible worlds
   - Meaning w.r.t. time

Recapitulation: Meaning as reference

Until now we identified meaning with reference.

- Names denote individual constants (type e), which represent the entities they refer to.
- Sentences denote truth-values (type t).
- Predicates denote functions from individuals to truth-values.

Extensionality

Extensionality

In a complex expression E, a sub-expression can be substituted by another expression that has the same meaning without changing the meaning of E.

If meaning is reference:
Expressions with the same reference should be interchangeable without changing the truth-value of the sentences they occur in.

Extensional contexts

- *Alonzo greeted the queen of the Netherlands.*
- *Alonzo greeted Beatrix.*

- *Eight is greater than seven.*
- *The number of planets in our solar system is greater than seven.*

Opaque contexts

- *Eight is necessarily greater than seven.*
- *The number of planets in our solar system is necessarily greater than seven.*

- *I believe that Alonzo greeted the queen of the Netherlands.*
- *I believe Alonzo greeted Beatrix.*

- *Alonzo is looking for the queen of the Netherlands.*
- *Alonzo is looking for Beatrix.*

Since those sentences mean different things, meaning seems to be more than reference.

Sense and reference

Frege proposed to distinguish:

- the conceptual content of an expression (*Sinn*, or *intension*)
- its actual reference (*Bedeutung*, or *extension*)

Sense and reference

Frege proposed to distinguish:

- the conceptual content of an expression (*Sinn*, or *intension*)
- its actual reference (*Bedeutung*, or *extension*)

Example:

- *the queen of the Netherlands*
  - Intension: royal head of the state of the Netherlands
  - Extension: Juliana (1949-1980), Beatrix (1980-present), ...
- *Slovenian number category*
  - Intension: grammatical category that expresses count distinction in Slovene
  - Extension: {singular,dual,plural}
- For a sentence, the intension is its truth conditions and the extension is its actual truth-value.

Intension and extension

So usually the intension is fixed, while the extension varies from context to context.

But vice versa, the extension could be the same, while the intension differs.

Example: *morning star, evening star*

- *The morning star is the evening star.*
- *The morning star is the morning star.*

Intensions

When we want to determine the reference of an expression, we have to consider the context, i.e. reference is not absolute anymore but depends on the context (time, possible worlds, anaphoric potential,. . .).

Meaning as intension

Meaning as intension

The meaning of an expression is not its extension (reference) anymore but its intension, i.e. a function that determines the reference given a certain context.

Intensions are functions from contexts to extensions.

Context: Possible worlds

Hintikka, Kripke

A *possible world* is a state of affairs that can differ from the actual state of affairs in any point.

Possible worlds can be represented as sets of propositional constants, namely all the propositions that hold in that world.

Examples

Truth and reference depend on the actual as well as possible situation.

- *The lecturer's team might have won.*
- *If the argonauts had recognized the Dolions, they wouldn't have killed them.*

Examples

Truth and reference depend on the actual as well as possible situation.

- *The lecturer's team might have won.*
- *If the argonauts had recognized the Dolions, they wouldn't have killed them.*

To determine the truth of modal statements like

- *Possibly the Higgs boson exists.*
- *The Higgs boson necessarily exists.*

it is not important to know whether *The Higgs boson exists* is true in the actual world, but rather whether it is true in some world (so it is possible that it is true) or in all worlds (so there is no other way than for it to be true).

Capturing intensionality: Types and model structures

Plan

1. Add possible worlds to the model.
2. Relativize meaning to possible worlds.

We will have:

- world-dependent meanings
- world-independent meanings
- meanings that quantify over worlds

Adding possible worlds to the model

A model is a quadruple $M = (\mathcal{D}, W, R, \mathcal{I})$, where

- $\mathcal{D}$ is a non-empty set of entities, the *domain*
- W is a non-empty set of *possible worlds*
- R is a binary relation on W, the *accessibility relation* (where wRw' expresses that w' is accessible from w)
- $\mathcal{I}$ is an *interpretation function* that assigns an extension to all constants with respect to a possible world

Accessibility between possible worlds

Properties of the accessibility relation determine the behaviour of possibility and necessity, e.g. whether the following statements hold.

- **(reflexive)** If something is necessary, it is the case.
- **(serial)** If something is necessary, it is possible.
- **(transitive)** If something is necessary, it is necessarily necessary.
- **(symmetric)** If something is the case, it is necessarily possible.
- **(euclidean)** If something is possible, it is necessarily possible.

Implementation: Possible worlds

```
data World = W1 | W2 | W3 | W4 | W5 | W6
           deriving (Eq,Show,Bounded,Enum,Ord)

--worlds

ws :: [World]
ws =  [minBound..maxBound]

-- accessibility relation

wsAcc :: [(World,World)]
wsAcc = rtc worlds [(W1,W2),(W2,W3),(W3,W4),(W4,W5),(W5,W6)]
```

Where `rtc` is the reflexive transitive closure.

Implementation: Interpretation of constants

Now, the interpretation function for constants should be parametrized w.r.t. possible worlds.

```
int :: String -> World -> Entity -> Bool

int "wizard" w = \ [x] -> case w of
                        W1 -> x `elem` [B]
                        W2 -> x `elem` [A,B]
                        W3 -> False
                        W4 -> x `elem` [C,D,F]
                        W5 -> x `elem` [E]
                        W6 -> x `elem` [A,C,E]

int "admire" w = \ [x,y] -> case w of
                           W1 -> (x,y) `elem` [(B,B)]
                           W2 -> (x,y) `elem` [(C,A),(A,C)]
                           W4 -> (x,y) `elem` [(B,D),(D,F)]
                           W5 -> (x,y) `elem` [(A,C),(B,C),(E,F)]
                           _  -> False
```

Evaluation in a model

Expressions are evaluated w.r.t. a model M, a variable assignment g and a possible world w.

$$[\![R^n(t_1,\ldots,t_n)]\!]^{M,g,w} = 1 \text{ iff } (\mathcal{I}(t_1)(w),\ldots,\mathcal{I}(t_n)(w)) \in \mathcal{I}(R)(w)$$

$$[\![F_1 \wedge F_2]\!]^{M,g,w} = 1 \text{ iff } [\![F_1]\!]^{M,g,w} = 1 \text{ and } [\![F_2]\!]^{M,g,w} = 1$$

$$[\![\neg F]\!]^{M,g,w} = 1 \text{ iff } [\![F]\!]^{M,g,w} = 0$$

$$[\![\exists v.F]\!]^{M,g,w} = 1 \text{ iff there is some } d \in \mathcal{D} \text{ such that } [\![F]\!]^{M,g[v:=d],w} = 1$$

Implementation

```
eval :: Model -> Assignment -> World -> Formula -> Bool
eval m g w (Atom s ts)  = (interpretation m) s w (map (intTerm m g) ts)
eval m g w (Neg f)      = not $ eval m g w f
eval m g w (Conj fs)    = and $ map (eval m g w) fs
eval m g w (Disj fs)    = or  $ map (eval m g w) fs
eval m g w (Impl f1 f2) = not $ (eval m g w f1) && not (eval m g w f2)
eval m g w (Forall n f) = all (\d -> eval m (change g n d) w f) (domain m)
eval m g w (Exists n f) = any (\d -> eval m (change g n d) w f) (domain m)
```

Relativizing meanings to possible worlds

Goal: a systematic mapping from extensional meaning representations and types to their intensional counterparts

Intensional types

Types

$$\tau ::= e \mid t \mid \tau \to \tau \mid s \to \tau$$

Where s is a new base type, the type for possible worlds. (Note that it can only occur as input to a function.)

- **Propositions** are functions of type $s \to t$ and map possible worlds to truth values.
- **Individual concepts** are functions of type $s \to e$ and map possible worlds to entities.

Intensionalization

The intensional counterpart of an extensional type τ is the type $i_1(\tau)$, where i_1 is a mapping that replaces each occurence of an atomic type by its intensional counterpart, i.e. replaces type e by type $s \to e$ and type t by $s \to t$.

	extensional type	**intensional type**
sentence	t	$s \to t$
definite description	e	$s \to e$
noun	$e \to t$	$(s \to e) \to (s \to t)$
transitive verb	$e \to (e \to t)$	$(s \to e) \to ((s \to e) \to (s \to t))$

Intensionalization (Montague)

The intensional counterpart of an extensional type τ is $s \to i_2(\tau)$, where i_2 is the following mapping:

$$i_2(e) = e$$
$$i_2(t) = t$$
$$i_2(\tau \to \tau') = (s \to \tau) \to \tau'$$

	extensional type	**intensional type**
sentence	t	$s \to t$
definite description	e	$s \to e$
noun	$e \to t$	$s \to ((s \to e) \to t)$
transitive verb	$e \to (e \to t)$	$s \to ((s \to e) \to ((s \to e) \to t))$

Intensionalization

There is a close correspondence between those two approaches to intensionalization: We can go from one type to another by repeatedly changing the order of arguments ($\tau \to \tau' \to \tau'' \rightsquigarrow \tau' \to \tau \to \tau''$).

Example:

$$\begin{aligned} i_1(e \to e \to t) &= (s \to e) \to (s \to e) \to s \to t \\ &\rightsquigarrow (s \to e) \to s \to (s \to e) \to t \\ &\rightsquigarrow s \to (s \to e) \to (s \to e) \to t \\ &= i_2(e \to e \to t) \end{aligned}$$

Intensionalization and extensionalization operator

Intensionalization:

- $^{\cap} :: (s \to \tau) \to i_1(\tau)$ takes a function that gives an extension for every world, and maps that to an intensional function that does the same job.

Extensionalization:

- $^{\cup} :: i_1(\tau) \to (s \to \tau)$ cancels the lifting effect of $^{\cap}$ again, i.e. $^{\cup\cap}x = x$.

These operators can be defined by mutual recursion over the structure of an expression (cf. Chapter 8 of the book).

World-independent meanings

Kripke proposed that names are rigid designators, i.e. that their reference is the same in all world. Thus for all possible worlds w it holds:

- $[\![Atreyu]\!]^{M,g,w} = a$
- $[\![Dorothy]\!]^{M,g,w} = d$
- $[\![Goldilocks]\!]^{M,g,w} = e$
- ...

World-dependent meanings

The reference of other expressions can vary in different worlds.

Examples:

- $[\![\textit{the nobel prize winner}]\!]^{M,g,w_1} = b$
- $[\![\textit{the nobel prize winner}]\!]^{M,g,w_2} = f$

- $[\![\textit{Germany wins the world championship}]\!]^{M,g,w_1} = 1$
- $[\![\textit{Germany wins the world championship}]\!]^{M,g,w_2} = 0$

Relativizing meaning to possible worlds

	Translation	Type
Atreyu	$\lambda w.a$	ie
wizard	$\lambda x \lambda w.((wizard\ w)\ (x\ w))$	$ie \to it$
laughed	$\lambda x \lambda w.((laugh\ w)\ (x\ w))$	$ie \to it$
happy	$\lambda x \lambda w.((happy\ w)\ (x\ w))$	$ie \to it$
defeat	$\lambda x \lambda y \lambda w.(((defeat\ w)\ (x\ w))\ (y\ w))$	$ie \to ie \to it$
every	$\lambda P \lambda Q \lambda w\, \forall x.(((P\ w)\ (x\ w)) \to ((Q\ w)\ (x\ w)))$	$(ie \to it) \to (ie \to it) \to it$
some	$\lambda P \lambda Q \lambda w\, \exists x.(((P\ w)\ (x\ w)) \wedge ((Q\ w)\ (x\ w)))$	$(ie \to it) \to (ie \to it) \to it$

Where ie is shorthand for $(s \to e)$ and it is shorthand for $(s \to t)$.

Example

Example: *Atreyu laughed.*

$$(\lambda x \lambda w.((\mathit{laugh}\ w)\ (x\ w))\ \ \lambda w'.a)$$
$$\rhd\ \lambda w.((\mathit{laugh}\ w)\ (\lambda w'.a\ \ w))$$
$$\rhd\ \lambda w.((\mathit{laugh}\ w)\ a)$$

Expressions quantifying over possible worlds

We add modal operators $\Diamond$ (possibly) and $\Box$ (necessarily) to our language. They quantify over possible worlds:

$$[\![\Diamond F]\!]^{M,g,w} = 1 \text{ iff there is some } w' \in W \text{ such that } [\![F]\!]^{M,g,w'} = 1$$

$$[\![\Box F]\!]^{M,g,w} = 1 \text{ iff for all } w' \in W : [\![F]\!]^{M,g,w'} = 1$$

Good enough?

Expressions quantifying over possible worlds

We add modal operators $\Diamond$ (possibly) and $\Box$ (necessarily) to our language. They quantify over possible worlds:

$$[\![\Diamond F]\!]^{M,g,w} = 1 \text{ iff there is some } w' \in W \text{ such that } wRw' \text{ and } [\![F]\!]^{M,g,w'} = 1$$

$$[\![\Box F]\!]^{M,g,w} = 1 \text{ iff for all } w' \in W \text{ with } wRw' : [\![F]\!]^{M,g,w'} = 1$$

Example

	$[\![\text{S}]\!] :: (s \to t)$
$\mathbf{S} \to \mathbf{SMOD\ S}$	$[\![\text{S}]\!] = ([\![\text{SMOD}]\!]\ [\![\text{S}]\!])$
	$[\![\text{SMOD}]\!] :: (s \to t) \to (s \to t)$
$\mathbf{SMOD} \to \textit{possibly}$	$[\![\text{SMOD}]\!] = \lambda p \lambda w.\Diamond(p\ w)$
$\mathbf{SMOD} \to \textit{necessarily}$	$[\![\text{SMOD}]\!] = \lambda p \lambda w.\Box(p\ w)$

Example

	$[\![\text{S}]\!] :: (s \to t)$
$\mathbf{S} \to \mathbf{SMOD\ S}$	$[\![\text{S}]\!] = ([\![\text{SMOD}]\!]\ [\![\text{S}]\!])$
	$[\![\text{SMOD}]\!] :: (s \to t) \to (s \to t)$
$\mathbf{SMOD} \to \textit{possibly}$	$[\![\text{SMOD}]\!] = \lambda p \lambda w.\Diamond(p\ w)$
$\mathbf{SMOD} \to \textit{necessarily}$	$[\![\text{SMOD}]\!] = \lambda p \lambda w.\Box(p\ w)$

Example: *Possibly, Atreyu laughed.*

$$
\begin{aligned}
&(\lambda p \lambda w.\Diamond(p\ w)\ \ \lambda w'.((\mathit{laugh}\ w')\ a)) \\
\rhd\ &\lambda w.\Diamond(\lambda w'.((\mathit{laugh}\ w')\ a)\ \ w) \\
\rhd\ &\lambda w.\Diamond((\mathit{laugh}\ w)\ a)
\end{aligned}
$$

Example

Now we can also capture the meaning difference between

- *Necessarily, eight is greater than seven.*
- *Necessarily, the number of planets in our solar system is greater than seven.*

by means of a model that is such that in every possible world *eight is greater than seven* is true, but the number of planets in our solar system differs between worlds. Then the first sentence holds while the latter does not.

Implementation

```
data Formula = Atom String [Term]
             | Neg Formula
             | Conj [Formula]
             | Disj [Formula]
             | Impl Formula Formula
             | Forall Int Formula
             | Exists Int Formula
             | Poss Formula
             | Ness Formula
             deriving Eq

instance Show Formula where
  ...
  show (Poss f) = "<>" ++ show f
  show (Ness f) = "[]" ++ show f
```

Implementation

```
eval :: Model -> Assignment -> World -> Formula -> Bool
eval m g w (Atom s ts)  = (interpretation m) s w (map (intTerm m g) ts)
eval m g w (Neg f)      = not $ eval m g w f
eval m g w (Conj fs)    = and $ map (eval m g w) fs
eval m g w (Disj fs)    = or  $ map (eval m g w) fs
eval m g w (Impl f1 f2) = not $ (eval m g w f1) && not (eval m g w f2)
eval m g w (Forall n f) = all (\d -> eval m (change g n d) w f) (domain m)
eval m g w (Exists n f) = any (\d -> eval m (change g n d) w f) (domain m)
eval m g w (Poss f)     = any (\w' -> (w,w') `elem` (accessibility m)
                                      && eval m g w' f)
                              (worlds m)
eval m g w (Ness f)     = all (\w' -> not ((w,w') `elem` (accessibility m))
                                      || eval m g w' f)
                              (worlds m)
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
True
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
True

Day4_FOL> eval model ass W3 (Ness (Atom "laugh" [Const "a"]))
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
True

Day4_FOL> eval model ass W3 (Ness (Atom "laugh" [Const "a"]))
False
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
True

Day4_FOL> eval model ass W3 (Ness (Atom "laugh" [Const "a"]))
False

Day4_FOL> eval model ass W1 (Forall 1 (Impl (Atom "evil" [Var 1])
                                (Exists 2 (Atom "defeat" [Var 2,Var 1]))))
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
True

Day4_FOL> eval model ass W3 (Ness (Atom "laugh" [Const "a"]))
False

Day4_FOL> eval model ass W1 (Forall 1 (Impl (Atom "evil" [Var 1])
                                (Exists 2 (Atom "defeat" [Var 2,Var 1]))))
True
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
True

Day4_FOL> eval model ass W3 (Ness (Atom "laugh" [Const "a"]))
False

Day4_FOL> eval model ass W1 (Forall 1 (Impl (Atom "evil" [Var 1])
                                (Exists 2 (Atom "defeat" [Var 2,Var 1]))))
True

Day4_FOL> eval model ass W1 (Ness (Forall 1 (Impl (Atom "evil" [Var 1])
                                (Exists 2 (Atom "defeat" [Var 2,Var 1])))))
```

Example

```
Day4_FOL> eval model ass W1 (Poss (Atom "laugh" [Const "a"]))
True

Day4_FOL> eval model ass W3 (Ness (Atom "laugh" [Const "a"]))
False

Day4_FOL> eval model ass W1 (Forall 1 (Impl (Atom "evil" [Var 1])
                                (Exists 2 (Atom "defeat" [Var 2,Var 1]))))
True

Day4_FOL> eval model ass W1 (Ness (Forall 1 (Impl (Atom "evil" [Var 1])
                                (Exists 2 (Atom "defeat" [Var 2,Var 1])))))
True
```

Implementation

```
transS :: Tree String String -> Formula
transS (Branch "S" [np@(Branch "NP" _),vp]) = (transNP np) (transVP vp)
transS (Branch "S" [smod,s])                = (transSMOD smod) (transS s)

transSMOD :: Tree String String -> Formula -> Formula
transSMOD (Branch "SMOD" [Leaf "possibly"])    = \ p -> Poss p
transSMOD (Branch "SMOD" [Leaf "necessarily"]) = \ p -> Ness p
```

Meaning w.r.t. time

We can capture time analogously to possible worlds.

1. Add time parameter to the model.
2. Relativize meaning to time.

We will have:

- time-dependent meanings (*the queen of the Netherlands*)
- time-independent meanings (*Beatrix*)
- meanings that quantify over time points (*former*)

Adding time points to the model

A *model* is a quadruple $M = (\mathcal{D}, T, R, \mathcal{I})$, where

- $\mathcal{D}$ is a non-empty set of entities, the *domain*
- T is a set of *time points*
- R is a linear ordering on T (where $i < i'$ expresses that time point i 'lies before' time point i')
- $\mathcal{I}$ is an *interpretation function* that assigns an extension to all constants with respect to a point in time

Relativizing meaning to time

Types

$$\tau ::= e \mid t \mid \tau \to \tau \mid i \to \tau$$

Where i is a new base type, the type for time points.

	extensional type	**intensional type**
sentence	t	$i \to t$
definite description	e	$i \to e$
noun	$e \to t$	$(i \to e) \to (i \to t)$
transitive verb	$e \to (e \to t)$	$(i \to e) \to ((i \to e) \to t)$

Relativizing meaning to time

	Translation	Type
Atreyu	$\lambda t.a$	ie
wizard	$\lambda x \lambda t.((wizard\ t)\ (x\ t))$	$ie \rightarrow it$
laughed	$\lambda x \lambda t.((laugh\ t)\ (x\ t))$	$ie \rightarrow it$
happy	$\lambda x \lambda t.((happy\ t)\ (x\ t))$	$ie \rightarrow it$
defeat	$\lambda x \lambda y \lambda t.(((defeat\ t)\ (x\ t))\ (y\ t))$	$ie \rightarrow ie \rightarrow it$
every	$\lambda P \lambda Q \lambda t\, \forall x.(((P\ t)\ (x\ t)) \rightarrow ((Q\ t)\ (x\ t)))$	$(ie \rightarrow it) \rightarrow (ie \rightarrow it) \rightarrow it$
some	$\lambda P \lambda Q \lambda t\, \exists x.(((P\ t)\ (x\ t)) \wedge ((Q\ t)\ (x\ t)))$	$(ie \rightarrow it) \rightarrow (ie \rightarrow it) \rightarrow it$

Where ie is shorthand for $(i \rightarrow e)$ and it is shorthand for $(i \rightarrow t)$.

Example

Example: *Atrey laughed.*

$$
\begin{aligned}
& (\lambda x \lambda t.((\mathit{laugh}\ t)\ (x\ t))\ \ \lambda t.a) \\
\rhd\ & \lambda t.((\mathit{laugh}\ t)\ (\lambda w.a\ \ t)) \\
\rhd\ & \lambda t.((\mathit{laugh}\ t)\ a)
\end{aligned}
$$

Truth w.r.t. time

In order to check for truth of an expression E in a model, we do not compute $[\![E]\!]^{M,g}$ but $[\![E]\!]^{M,g,t}$, where M is a model, g is an assignment function of variables to entities, and t is some time point.

Time-dependent expressions

The reference of some expressions changes over time.

Example: *the queen of the Netherlands*

- $[\![\textit{the queen of the Netherlands}]\!]^{M,g,t_0} = b$
- $[\![\textit{the queen of the Netherlands}]\!]^{M,g,t_1} = j$

Quantifying over time: 'former'

Some expressions switch the relevant time point for evaluation to the past or future.

Example: *former queen of the Netherlands*

- $\lambda t \lambda x. \neg((\mathit{queenOfNL}\ t)\ (x\ t)) \wedge \exists t'.(t' < t) \wedge ((\mathit{queenOfNL}\ t')\ (x\ t'))$

(Note that we have to add the relation symbol $<$ to our language.)

Quantifying over time: 'former'

Some expressions switch the relevant time point for evaluation to the past or future.

Example: *former queen of the Netherlands*

- $\lambda t \lambda x. \neg((\mathit{queenOfNL}\ t)\ (x\ t)) \wedge \exists t'.(t' < t) \wedge ((\mathit{queenOfNL}\ t')\ (x\ t'))$

(Note that we have to add the relation symbol $<$ to our language.)

$\mathbf{N} \rightarrow \mathbf{ADJ\ N}$	$[\![\text{ADJ}]\!] = \lambda x.(([\![\text{ADJ}]\!]\ x) \wedge ([\![\text{N}]\!]\ x))$
$\mathbf{N} \rightarrow \mathbf{ADJ\ N}$	$[\![\text{ADJ}]\!] = ([\![\text{ADJ}]\!]\ [\![\text{N}]\!])$
$\mathbf{ADJ} \rightarrow \mathit{former}$	$[\![\text{ADJ}]\!] = \lambda P \lambda x \lambda t.\ \neg((P\ t)\ (x\ t)) \wedge \exists t'.(t' < t) \wedge ((P\ t')\ (x\ t'))$

Quantifying over time: Past tense

We can specify the meaning of a past tense verb like *laughed* as follows:

$$\lambda x \lambda t. \exists t'. (t' < t) \wedge ((\mathit{laugh}\ t')\ (x\ t')) :: \mathit{ie} \rightarrow \mathit{it}$$

Example: *Atreyu laughed.*

$$\begin{aligned} & (\lambda x \lambda t. \exists t'. (t' < t) \wedge ((\mathit{laugh}\ t')\ (x\ t'))\ \ \lambda t''. a) \\ \rhd\ & \lambda t.\ \exists t'. (t' < t) \wedge ((\mathit{laugh}\ t')\ a) \end{aligned}$$

Example: Past tense

Or more systematically by adding an operator **PAST** (and a corresponding operator **FUTURE**) to our language.

- $\textbf{PAST} = \lambda P \lambda x \lambda t. \exists t'. (t' < t) \wedge ((P\ t')\ (x\ t')) :: (ie \to it) \to (ie \to it)$
- $\textbf{FUTURE} = \lambda P \lambda x \lambda t. \exists t'. (t' > t) \wedge ((P\ t')\ (x\ t')) :: (ie \to it) \to (ie \to it)$

Example: *Atreyu laughed.*

$$\begin{aligned} & ((\textbf{PAST}\ \lambda x \lambda t. ((laugh\ t')\ (x\ t')))\ \ \lambda t''. a) \\ \rhd\ & (\lambda x \lambda t. \exists t'. (t' < t) \wedge ((laugh\ t')\ (x\ t'))\ \ \lambda t''. a) \\ \rhd\ & \lambda t. \exists t'. (t' < t) \wedge ((laugh\ t')\ a) \end{aligned}$$

Implementation

The implementation can be done analogously to the case of possible worlds.

- Extend the model with time points and a corresponding ordering relation.
- Extend our predicate logical language with modal operators `Past` and `Future`.
- Evaluate predicate logical formulas w.r.t. a model, an assignment, and a time point.

(Feel free to try it!)

Course overview

Day 2:
Meaning representations and (predicate) logic

Day 3:
Lambda calculus and the composition of meanings

Day 4:
Extensionality and intensionality

- **Day 5:**
  From strings to truth conditions and beyond