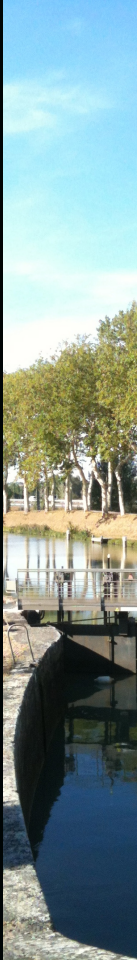


Génériques, quantificateurs généralisés et opérateurs de Hilbert

Generics, quantifiers and Hilbert's operators

Christian Retoré

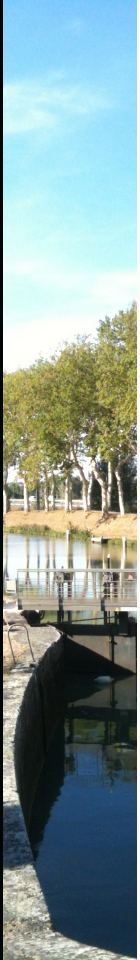
Université de Bordeaux & IRIT, Toulouse (en 2012–2013)

atelier **Déterminants et Inférences (F. Corblin)**

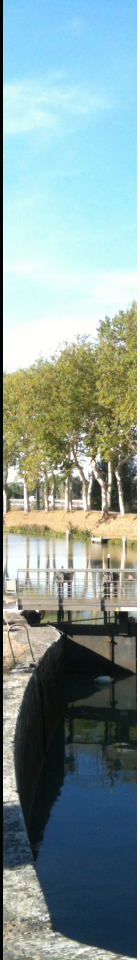
Université Paris Sorbonne,

Centre universitaire Malesherbes,

11–12 juin 2013



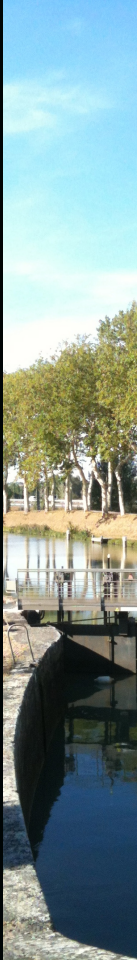
A Reminder on the usual Montagovian framework



A.1. Mind that there are TWO logics

One for expressing meanings:
formulae of first or higher order logic, single or multi sorted.

One for meaning assembly:
proofs in intuitionistic propositional logic, λ-terms expressing the well-formedness of formulae.



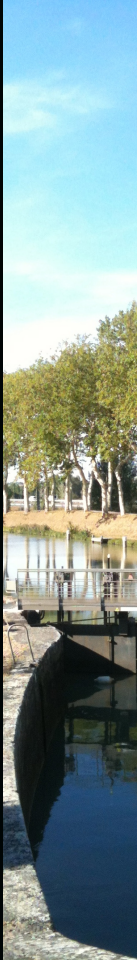
A.2. Representing formulae within lambda calculus — connectives

Assume that the base types are
- $\mathbf{e}$ (individuals, often there is just one) and
- $\mathbf{t}$ (propositions)

and that the only constants are
- the logical ones (below) and
- the relational and functional symbols of the specific logical language (on the next slide).

Logical constants:

- $\sim$ of type $\mathbf{t} \to \mathbf{t}$ (negation)
- $\supset, \&, +$ of type $\mathbf{t} \to (\mathbf{t} \to \mathbf{t})$
  (implication, conjunction, disjunction)
- two constants $\forall$ and $\exists$ of type $(\mathbf{e} \to \mathbf{t}) \to \mathbf{t}$



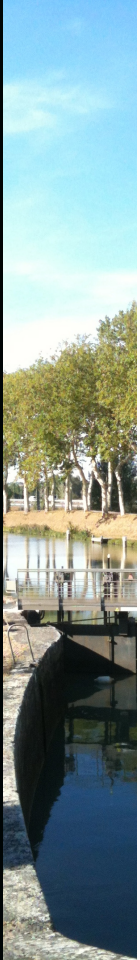
A.3. Representing formulae within lambda calculus — language constants

The language constants for multi sorted First Order Logic:

- R_q of type $\mathbf{e} \rightarrow (\mathbf{e} \rightarrow (.... \rightarrow \mathbf{e} \rightarrow \mathbf{t}))$
- f_q of type $\mathbf{e} \rightarrow (\mathbf{e} \rightarrow (.... \rightarrow \mathbf{e} \rightarrow \mathbf{e}))$

likes	$\lambda x \lambda y$ (likes y) x	$x : e,\ y : e, \text{likes} : e \rightarrow (e \rightarrow t)$
<< likes >> is a two-place predicate		

Garance	λP (P Garance)	$P : e \rightarrow t,$ Garance $: e$
<< Garance >> is viewed as the properties that << Garance >> holds		

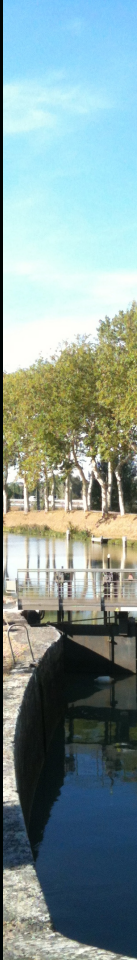


A.4. Normal terms of type t are formulae

Easy but important result

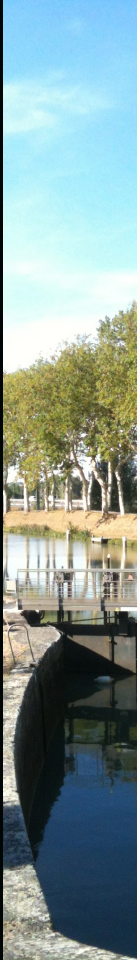
(induction on normal λ-terms preferably η-long):

1. normal λ-terms of type $\mathbf{e}$ with $x_k : \mathbf{e}$ as only free variables are logical terms with the same free variables
2. normal λ-terms (preferably η-long) of type $\mathbf{t}$ with $x_i : \mathbf{e}$ as only free variables are logical formulae with the same free variables and bound variables.



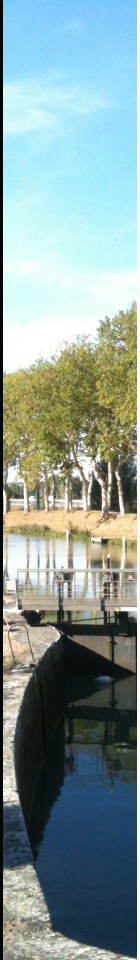
A.5. Montague semantics. Syntax/semantics.

(Syntactic type)*	=	Semantic type	
S^*	=	t	a sentence is a proposition
np^*	=	e	a noun phrase is an entity
n^*	=	$e \to t$	a noun is a subset of the set of entities
$(A\backslash B)^* = (B/A)^*$	=	$A \to B$	extends easily to all syntactic categories of a Categorial Grammar e.g. a Lambek CG



A.6. Montague semantics. Algorithm

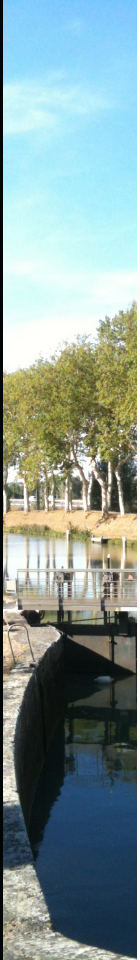
1. Replace in the lambda-term issued from the syntax the words by the corresponding term of the lexicon.
2. Reduce the resulting λ-term of type t its normal form corresponds to a formula, the ”meaning”.



A.7. Ingredients: a parse structure & a lexicon

Syntactical structure (some (club)) (defeated Leeds)

word	***semantic type*** u^* ***semantics :*** λ***-term of type*** u^* x^v *the variable or constant* x *is of type* v
some	$(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge^{t\to(t\to t)}(P\ x)(Q\ x))))$
club	$e \to t$ $\lambda x^e(\texttt{club}^{e\to t}\ x)$
defeated	$e \to (e \to t)$ $\lambda y^e\ \lambda x^e\ ((\texttt{defeated}^{e\to(e\to t)}\ x)y)$
Leeds	e Leeds



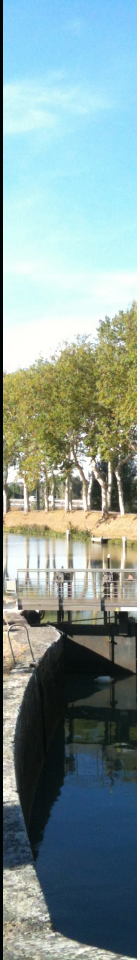
A.8. Computing the semantic representation

Put semantics terms into the parse structure & β reduce:

$$\Big(\big(\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge(P\ x)(Q\ x))))\big)\big(\lambda x^{e}(\texttt{club}^{e\to t}\ x)\big)\Big)$$
$$\Big(\big(\lambda y^{e}\ \lambda x^{e}\ ((\texttt{defeated}^{e\to(e\to t)}\ x)y)\big)\ \mathit{Leeds}^{e}\Big)$$
$$\downarrow \beta$$
$$\big(\lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge^{t\to(t\to t)}(\texttt{club}^{e\to t}\ x)(Q\ x)))))$$
$$\big(\lambda x^{e}\ ((\texttt{defeated}^{e\to(e\to t)}\ x)\mathit{Leeds}^{e})\big)$$
$$\downarrow \beta$$
$$\big(\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge(\texttt{club}^{e\to t}\ x)((\texttt{defeated}^{e\to(e\to t)}\ x)\mathit{Leeds}^{e})))\big)$$

Usually human beings prefer to write it like this:

$$\exists x : e\ (\texttt{club}(x)\ \wedge\ \texttt{defeated}(x, \mathit{Leeds}))$$



A.9. Montague: good architecture / limits

Good trick (Church):

a propositional logic for meaning assembly (proofs/λ-terms) to compute

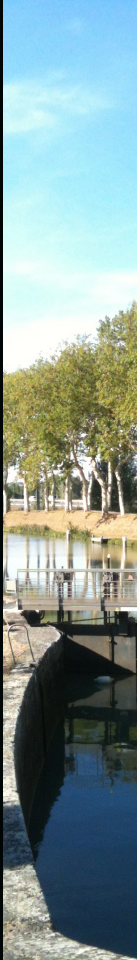
formulae another logic with first order (formulae/meaning no proofs)

The dictionary says ”barks” requires a subject of type ”animal”. How could we block:

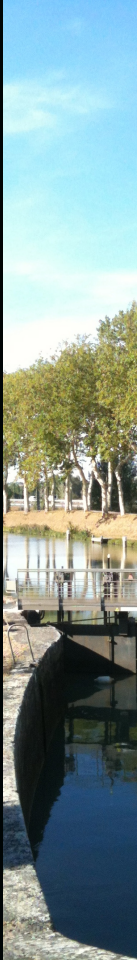
* The chair barked.

By type mismatch, $(f^{A\to X}(u^B))$ hence **many types** are needed.

If we do not want too many operations, we need to **factorise** similar operations acting on family of types and terms.



B Quantifiers: existential quantifier, indefinite and definite determiners



B.1. Usual Montagovian treatment

(1) There's a tramp sittin' on my doorstep (song)

(2) Some girls give me money (song)

(3) Something happened to me yesterday (song)

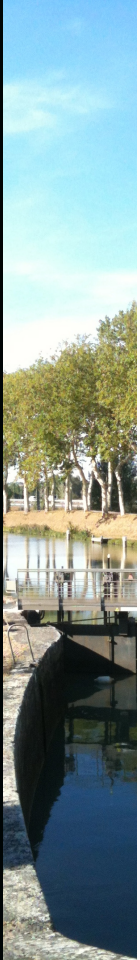
Usual view (e.g Montague)

Quantifier applies to the predicate,

$[\textit{something}] = \exists : (\mathbf{e} \rightarrow \mathbf{t}) \rightarrow \mathbf{t}$

and when there is a restriction to a class: $[\textit{some}]$

$\lambda P^{\mathbf{e}\rightarrow\mathbf{t}} \lambda Q^{\mathbf{e}\rightarrow\mathbf{t}} (\exists \lambda x^{\mathbf{t}} .\&(P\ x)(Q\ x)) : (\mathbf{e} \rightarrow \mathbf{t}) \rightarrow (\mathbf{e} \rightarrow \mathbf{t}) \rightarrow \mathbf{t}$



B.2. Some problems of this usual treatment

Syntactical structure of the sentence $\neq$ logical form.

(4) Keith played some Beatles songs.

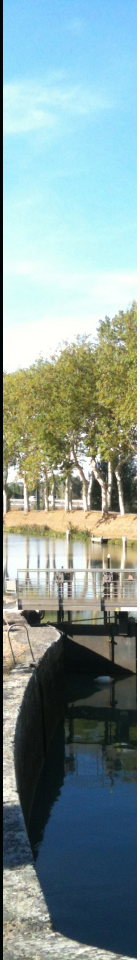
(5) semantics: (some (Beatles songs)) (Keith played _)

(6) syntax (Keith (played (some (Beatles songs))))

Example below have the same logical form.... how could they be interpreted differently.

(7) Some politicians are crooks (web)

(8) ? Some crooks are politicians (us)



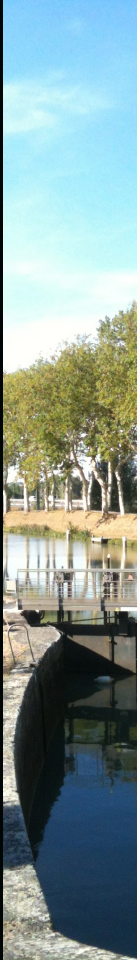
B.3. Idea (proof theory , mediaeval logic, Russell, Hilbert,... Geach,)

"*A man*" by it self seems to be already denoting something.... (Russell, Geach,...)

Idea use a generic element for the quantified NP.

This treatment of quantified NPs goes with dynamic binding and E-type pronouns.

(9) A man came in. He sat down.



B.4. Already a typed view?

At least, opposed to Frege's single sort view:

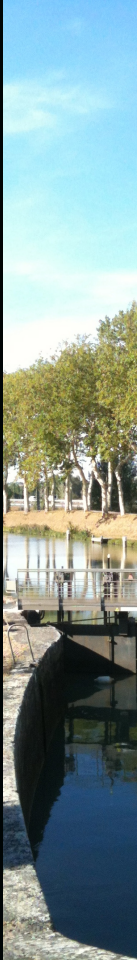
$\exists x{:}A\ P(x) \quad \equiv \quad \exists x.\ A(x)\&P(x)$

$\forall x{:}A\ P(x) \quad \equiv \quad \forall x.\ A(x) \rightarrow P(x)$

(impossible for "most of")

in ancient and especially medieval philosophy (in particular Abu Barakat, Avicenna):

we assert properties of things as being member of some class (= type?)



B.5. A solution: Hilbert's epsilon

Given a formula F there is an individual term

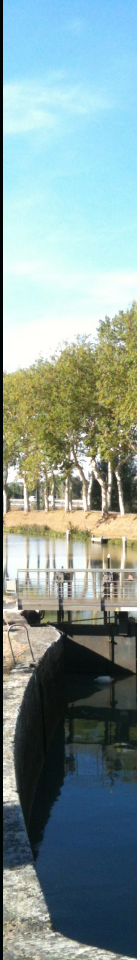
$$\varepsilon_x F(x)$$

(ε_x binds the occurrences of x in $F(x)$) with the intended meaning that $F(\varepsilon_x F) \equiv \exists x.\ F(x)$.

There is a dual term

$$\tau_x F(x)$$

with the dual meaning: $F(\tau_x F) \equiv \forall x.\ F(x)$.



B.6. Rules

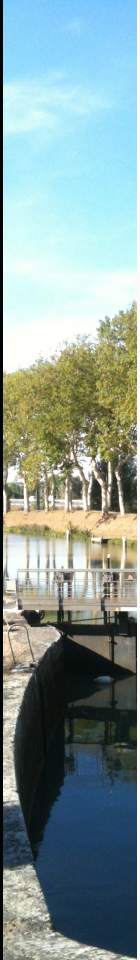
If $\forall x(A(x) \equiv B(x))$ then $\varepsilon_x A(x) = \varepsilon_x B(x)$.
The evident deduction rules for τ and ε are as follows:

- From $A(x)$ with x generic in the proof (no free occurrence of x in any hypothesis), infer $A(\tau_x.\ A(x))$
- From $B(c)$ infer $B(\varepsilon_x.\ B(x))$.

The additional rules can be found by duality:

- From $A(x)$ with x generic in the proof (no free occurrence of x in any hypothesis), infer $A(\varepsilon_x.\ \neg A(x))$
- From $B(c)$ infer $B(\tau_x.\ \neg B(x))$.

Hence we have $F(\tau_x.\ F(x)) \equiv \forall x.F(x)$ and $F(\varepsilon_x.\ F(x)) \equiv \exists x.\ F(x)$ and because of negation, one only of these operator is needed, usually the ε operator and the logic is known as the epsilon calculus. .

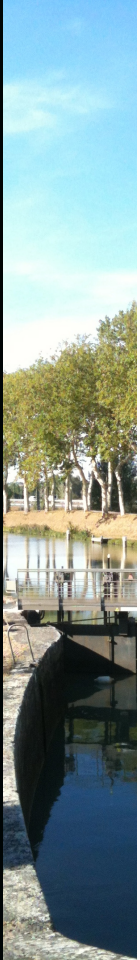


B.7. First and second epsilon theorems (Hilbert, 1934)

The first and second epsilon theorem basically say that this is an alternative formulation of first order logic.

First epsilon theorem When inferring a formula C without ε symbol nor quantifier from formulae Γ not involving the ε symbol nor quantifiers the derivation can be done within quantifier free predicate calculus.

Second epsilon theorem When inferring a formula C without ε symbol from formulae Γ not involving the ε symbol the derivation can be done within predicate calculus.



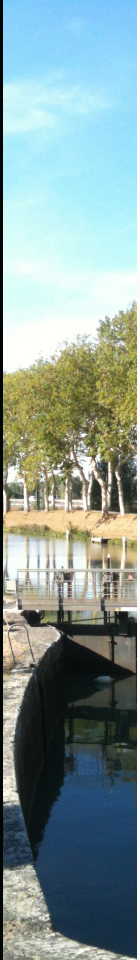
B.8. Interpretation – Asser 59, Leisenring 69

Domain M

Function, $M^p \mapsto M$, predicates $P \subset M^r$, as usual.

As part of an interpretation there is a choice function Φ which maps any part N of M to $\Phi(N) \in N$ — what about $\Phi(\emptyset)$? An arbitrary element in M.

Of course there are assignments for interpreting free variables.



B.9. Completeness

Semantic consequence.

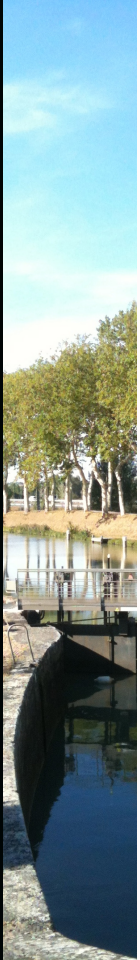
Maximally consistent sets of formulae.

Equivalence of terms $t \sim u$ if $t = u$ consequence of X

Representable parts $N = A(x)$ ($\Phi(N) = |\varepsilon_x A(x)|$, otherwise arbitrary in N).

No need of Henkin witnesses (the epsilon are there!).

Any maximal (w.r.t. semantic consequence) consistent set of formulae is true in some model.



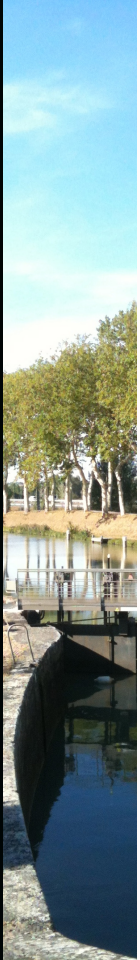
B.10. Properties

Formula of the predicate calculus -¿ epsilon formulae (even quantifier free epsilon formulae).

Converse? complex dependencies, like dynamic binding:

$come\_in(\varepsilon_x.Man(x))$ and $sit\_down(\varepsilon_x.Man(x))$
entails someone a man did both.

Heavy notation for an n-ary predicate e.g. :
$\forall x \exists y P(x, y)$ is
$\exists y P(\tau_x P(x, y), y)$ is
$P(\tau_x P(x, \varepsilon_y P(\tau_x P(x, y), y)), \varepsilon_y P(\tau_x P(x, y), y))$

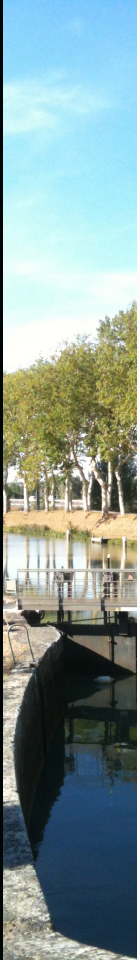


B.11. Advantages of ε

Follows syntactical structure.

Has a denotation by itself.

General presupposition $F(\varepsilon_x F)$: in natural language if we say "A man came in." we generally assert that such an individual exists and is a man.



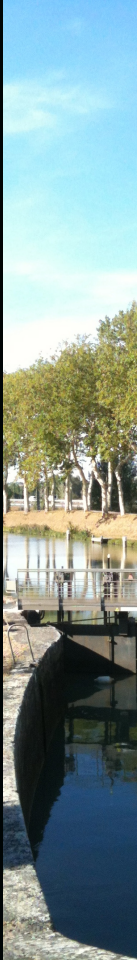
B.12. Definite and indefinite descriptions

$\iota_x F(x)$ The unique x such that (assumed to be due to Russell?)

According to von Heusinger (1995,1997, 2004) we can say the even if not unique, and it will pick up the most salient one, so *iota* can be left out.

- ε for definite descriptions
- η for indefinite description

Only a difference of interpretation: η a new one ε the most salient one.



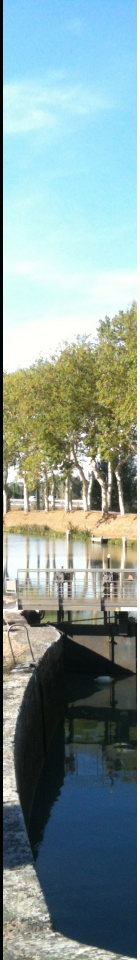
B.13. Other outcomes of Hilbert's operator

Universal quantifications: use τ ... with presupposition $P(\tau_x P(x))$

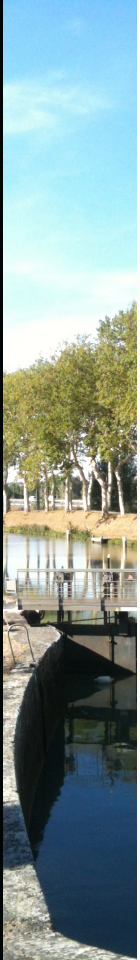
E-type pronouns:

(10) A man came in. He sat dow.

(11) $''He'' = ''A man'' = (\varepsilon_x Man(x))$.



C Typed Hilbert operators



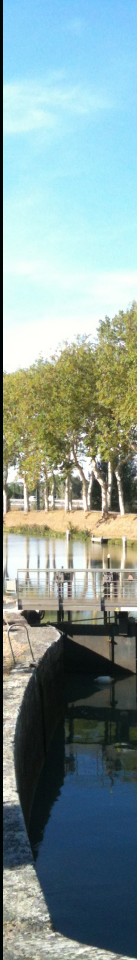
C.1. Polymorphic type system

Many sorts for including some lexical semantics.

Restriction of selection: type mismatch.

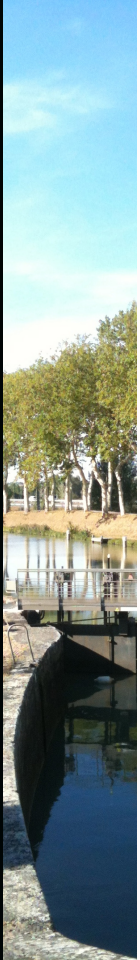
Some flexibility: organisation of the lexicon.

For operation that are uniform on types: quantification over types.



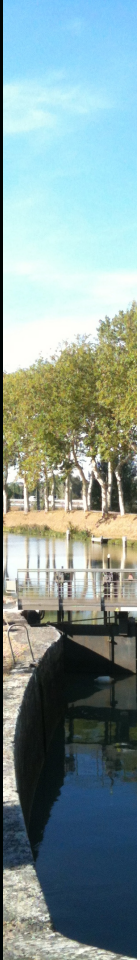
C.2. Types

- Constants types $\mathbf{e}_i$ and $\mathbf{t}$, as well as any type variable $\alpha, \beta, \ldots$ in P, are types.
- Whenever T is a type and α a type variable which may but need not occur in T, $\Lambda\alpha.\ T$ is a type.
- Whenever T_1 and T_2 are types, $T_1 \rightarrow T_2$ is also a type.



C.3. Terms

- A variable of type T i.e. $x : T$ or x^T is a term.
  Countably many variables of each type.
- $(f\ \tau)$ is a term of type U whenever $\tau : T$ and $f : T \to U$.
- $\lambda x^T.\ \tau$ is a term of type $T \to U$ whenever $x : T$, and $\tau : U$.
- $\tau\{U\}$ is a term of type $T[U/\alpha]$ whenever $\tau : \Lambda\alpha.\ T$, and U is a type.
- $\Lambda\alpha.\tau$ is a term of type $\Lambda\alpha.T$ whenever α is a type variable, and $\tau : T$ without any free occurrence of the type variable α. (Type of x in $\forall\alpha.x^\alpha$???)



C.4. Examples of second order usefulness

Arbitrary modifiers: $\Lambda\alpha\lambda x^{A} y^{\alpha} f^{\alpha\to R}.((\mathrm{read}^{A\to R\to t}\ x)\ (f\ y))$

Polymorphic conjunction:

Given predicates $P^{\alpha\to\mathbf{t}}$, $Q^{\beta\to\mathbf{t}}$ over entities of respective types α, β,

given any type ξ with two morphisms from ξ to α, to β

we can coordinate the properties P, Q of (the two images of) an entity of type ξ:

The polymorphic conjunction $\&^{\sqcap}$ is defined as the term

$$\&^{\sqcap} = \Lambda\alpha\Lambda\beta\lambda P^{\alpha\to\mathbf{t}}\lambda Q^{\beta\to\mathbf{t}} \Lambda\xi\lambda x^{\xi}\lambda f^{\xi\to\alpha}\lambda g^{\xi\to\beta}. (\mathrm{and}^{\mathbf{t}\to\mathbf{t}\to\mathbf{t}}\ (P\ (f\ x))(Q\ (g\ x)))$$

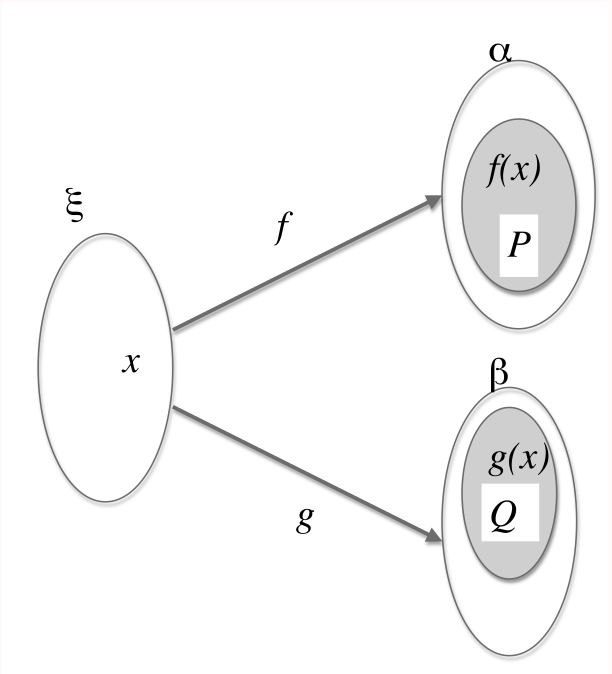
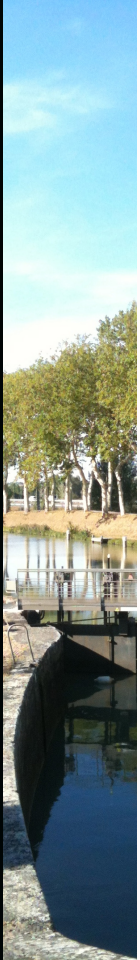
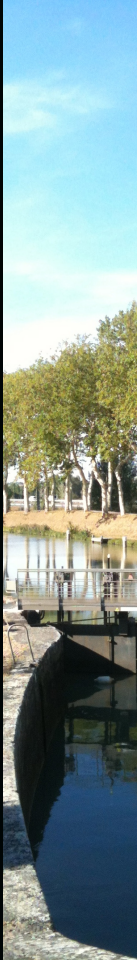


Figure 1: Polymorphic conjunction: $P(f(x))\&Q(g(x))$ with $x : \xi$, $f : \xi \to \alpha$, $g : \xi \to \beta$.



D System F based semantics and pragmatics



D.1. Examples

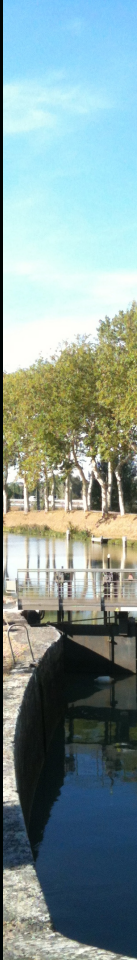
(12) Dinner was delicious but took ages.
(event / food)

(13) * The salmon we had for lunch was lightning fast.
(animal / food)

(14) I carried the books from the shelf to the attic.
Indeed, I already read them all.
(phys. / info — think of possible multiple copies of a book)

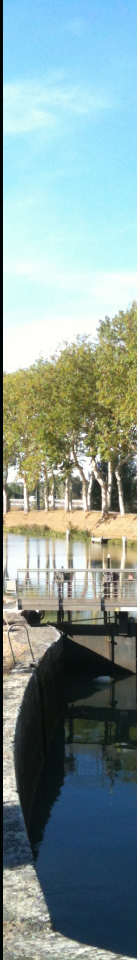
(15) Liverpool is a big place and voted last Sunday.
(geographic / people)

(16) * Liverpool is a big place and won last Sunday.
(geographic / football club)



D.2. Principles of our lexicon

- Remain within realm of Montagovian compositional semantics (for compositionality)
- Allow both predicate and argument to contribute lexical information to the compound.
- Based on optional modifiers attached to words (as opposed to derived from types).
- Integrate within existing discourse models (e.g. λ-DRT).



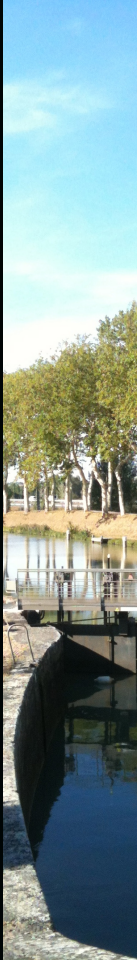
D.3. The Terms: principal or optional

A standard λ-term attached to the main sense:

- Used for compositional purposes
- Comprising detailed typing information

Some optional λ-terms (none is possible)

- Used, or not, for adaptation purposes
- Each associated with a constraint : rigid, $\varnothing$



D.4. RIGID vs FLEXIBLE use of optional terms

RIGID

Such a transformation is exclusive:

if is used, then the other associated with the same word are not used.

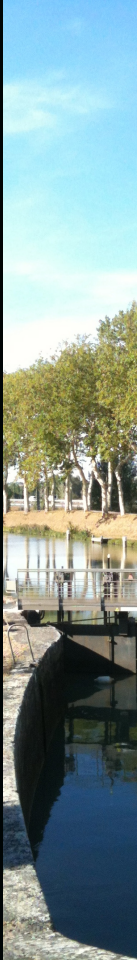
Each time we refer to the word it is with the same aspect.

FLEXIBLE

There is no constraint.

Any subset of the flexible transformation can be used:

different aspects of the words can be simultaneously used.



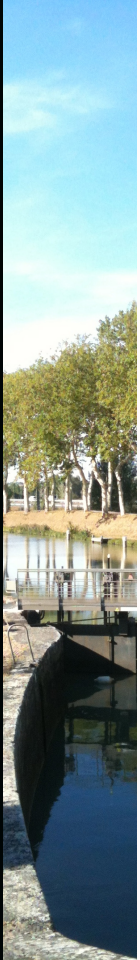
D.5. Standard behaviour

ϕ: physical objects

small stone

$$\overbrace{\overbrace{(\lambda x^{\varphi}.\,(\mathsf{small}^{\varphi\to\varphi}\,x))}^{\text{small}}\,\tau^{\varphi}}^{\text{stone}}$$

$$(\mathsf{small}\ \tau)^{\varphi}$$

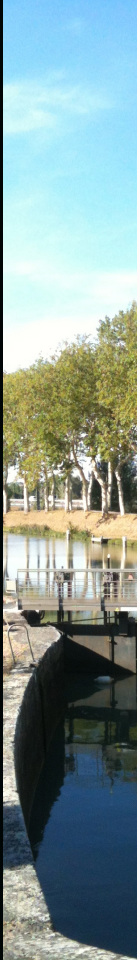


D.6. Correct copredication

word	principal λ-term	optional λ-terms	rigid/flexible
Liverpool	$liverpool^{T}$	$Id_T : T \rightarrow T$	(F)
		$t_1 : T \rightarrow F$	(R)
		$t_2 : T \rightarrow P$	(F)
		$t_3 : T \rightarrow Pl$	(F)
is_a_big_place	$big\_place : Pl \rightarrow \mathbf{t}$		
voted	$voted : P \rightarrow \mathbf{t}$		
won	$won : F \rightarrow \mathbf{t}$		

where the base types are defined as follows:

T	town
F	football club
P	people
Pl	place



D.7. Liverpool is a big place

Type mismatch:

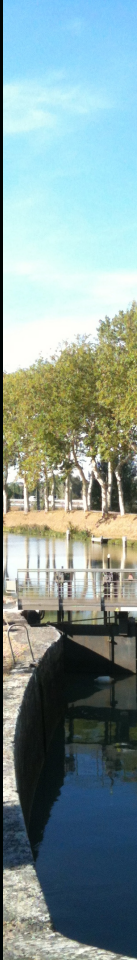
$$big\_place^{Pl \to \mathbf{t}}(Liverpool^{T}))$$

big_place applies to "*places*" (type *Pl*) and not to "*towns*" (T)

Lexicon $t_3^{T \to Pl}$ turns a town (T) into a place (Pl)

$$big\_place^{Pl \to \mathbf{t}}(t_3^{T \to Pl} Liverpool^{T}))$$

only one optional term, the (F)/ (R)difference is useless.



D.8. Liverpool is a big place and voted

Polymorphic AND yields: $(\&^{\Pi}(big\_place)^{Pl\to\mathbf{t}}(voted)^{P\to\mathbf{t}})$

Forces $\alpha := Pl$ and $\beta := P$, the properly typed term is

$$\&^{\Pi}\{Pl\}\{P\}(big\_place)^{Pl\to\mathbf{t}}(voted)^{P\to\mathbf{t}}$$

It reduces to:

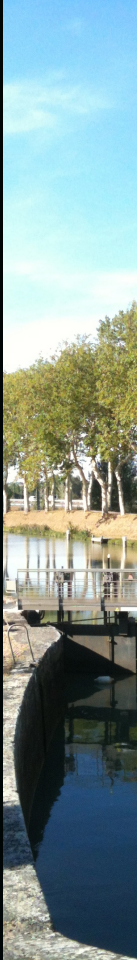
$$\Lambda\xi\lambda x^{\xi}\ \lambda f^{\xi\to\alpha}\lambda g^{\xi\to\beta}(\mathrm{and}^{\mathbf{t}\to\mathbf{t})\to\mathbf{t}}\ (big\_place\ (f\ x))(voted\ (g\ x)))$$

Syntax applies it to "*Liverpool*" so $\xi := T$ yielding

$\lambda f^{T\to Pl}\lambda g^{T\to P}(\mathrm{and}\ (is\_wide\ (f\ Liverpool^{T}))(voted\ (g\ Liverpool^{T}))))$.

The two flexible optional λ-terms $t_2 : T \to P$ and $t_3 : T \to Pl$ yield

$(\mathrm{and}\ (big\_place^{Pl\to\mathbf{t}}\ (t_3^{T\to Pl}\ Liverpool^{T}))(voted^{Pl\to\mathbf{t}}\ (t_2^{T\to P}\ Liverpool^{T})))$



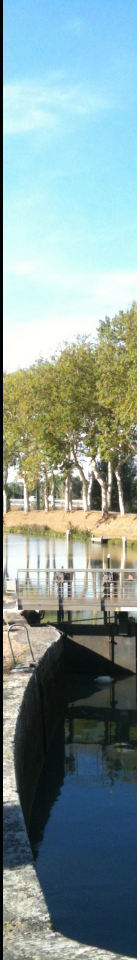
D.9. Liverpool voted and won

As previously but with *won* instead of *big_place*.

The term is:
$\lambda f^{T\to Pl}\lambda g^{T\to P}(\text{and}\ (won\ (f\ \ Liverpool^{T}))(voted\ (g\ \ Liverpool^{T})))$

for "*won*", we need to use the transformation $t_1 : T \to F$

but T_1 is rigid, hence we cannot access to the other needed transformation into a "*place*".



D.10. Typed Hilbert's operators? 1) base types.

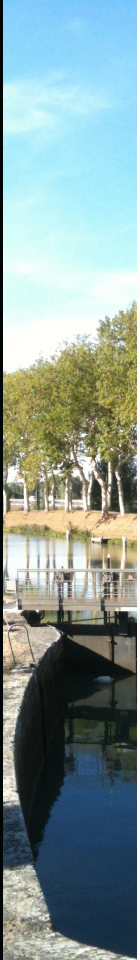
Personal intuition: there are less types than logical formulae with a single free variable, they are more constrained, they should be natural comparison classes.

What are the base types? CN (as Luo: rather sensible solution)

What about verbs and propositions?

(17) He did everything he could to stop them.

(18) And he believes whatever is politically correct and sounds good.

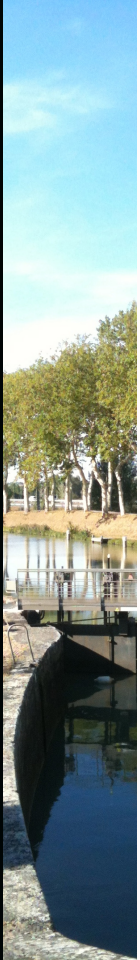


D.11. Typed Hilbert's operators? 2) predicates.

What is a predicate? what type is its domain?

For instance, is cat a property of animal or of all individuals ?

What relation between the type cat and the property of being a cat?



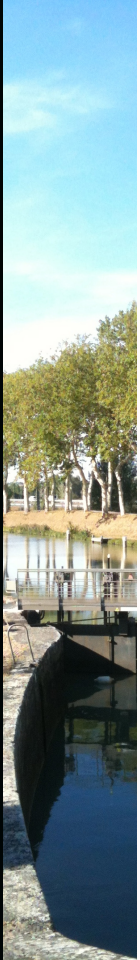
D.12. Some proposals

Predicates in a typed world:

- Simplest: predicates are of type $\mathbf{e} \rightarrow \mathbf{t}$ and they can be restricted.
- Predicates are not necessarily of type $\mathbf{e} \rightarrow \mathbf{t}$ (for being more natural) but a predicate $P^{\alpha \rightarrow \mathbf{t}}$ can be restricted and extended (they are false elsewhere). Hence, syntactically we do not need to know whether $\alpha \subset \beta$, $\alpha \subset \beta$

Types in a world with predefined predicates:

- Given a type α there is a corresponding predicate $\widehat{\alpha} : \mathbf{e} \rightarrow \mathbf{t}$. Too complicated to have a rule picking up the natural type "larger" than α. If types are some kind of an ontological tree, the mother node is good candidate.
- Alternative with, in my opinion, too many types: each formula with a single variable introduces a new type.



D.13. An example with an indefinite article

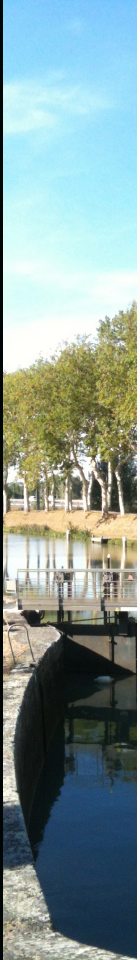
(19) a cat is sleeping under your car.

If cat is a property of type, say animal $\rightarrow$ **t**

- a: $\Lambda\alpha(\alpha \rightarrow \mathbf{t}) \rightarrow \alpha$ ("a" is a polymorphic ε). α gets instantiated as The general presupposition $P(a(P))$ yields $cat(a(cat))$. The epsilon term should be interpreted as an individual enjoying the property. If there is none, as an ideal individual, not in the denotation of P.

If cat is a type,

- It can be turned into a property and then do as above.
- a:$\Lambda\alpha.\alpha$ is strange but works (no problem to have a constant of type $\perp$) There is no need to add any presupposition, since because of the type of a one has $a(cat) : cat$.



D.14. von Heusinger intepretation

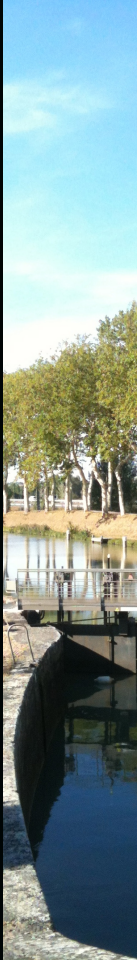
Interpretation of ε calculus is complicated Indeed, some ε-formulae have no equivalent formula in predicate logic. They can involve intricate dependencies, like Henkin branching quantifiers.

von Heusinger 's interpretation (IMHO: cosi cosi): a new one should be considered at each time. But...

(20) An old man came in he sat down. A tall man went out.

(21) $\varepsilon_x(M(x)\&O(x)) \neq \varepsilon_x(M(x)\&T(x))$

Although they are not the same ε-terms, their referents ought to be different.



D.15. Definite description, universal quantifiers

The formalisation also works for definite descriptions:

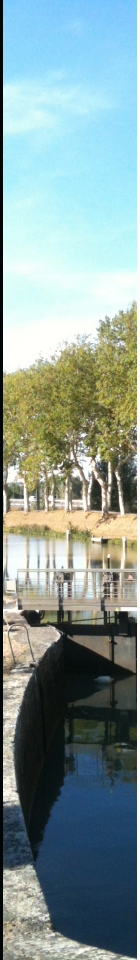
(22) The cat is sleeping under your car.

A difference is at interpretation: ι should be interpreted at the unique such that, and

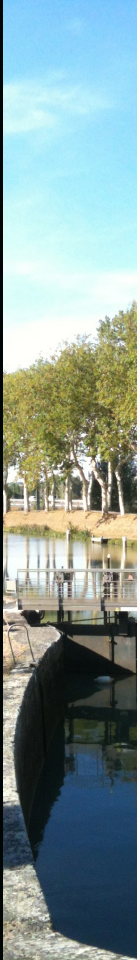
In fact the simplest is the universal quantifier:

(23) Any cat sleeps a lot.

The presupposition $cat(\tau_x cat(x))$ (obtained from the general $P(\tau_x P(x))$)
The $\tau_x cat(x)$ is interpreted as an additional virtual element.



E Conclusion



E.1. What we have seen so far

A **general framework** for

the logical syntax of **compositional semantics**

some lexical **semantics phenomena**

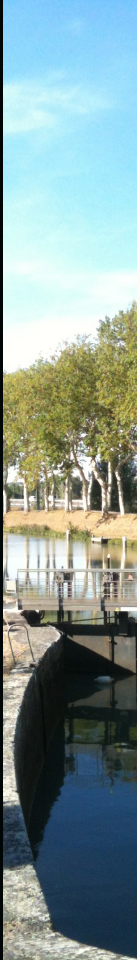
Guidelines:

Terms: semantics, sense, instructions for computing references

Types: pragmatics, defined from the context

Idiosyncratic (language specific) Lot of lexical informations. J'ai crevé. / ??? I went flatJe suis venu à pieds, mon vélo est crevé / *déraillé.

Practically: partly implemented in Grail, Moot's wide coverage categorial parser, for fragments with a hand-typed semantic lexicon — but with λ-DRT instead of HOL in lambda calculus.



E.2. Perspective 1: base types, specialisation relations, subtyping

What are the base types? (linguistic idea comparison classes, classifiers for languages with classifiers)

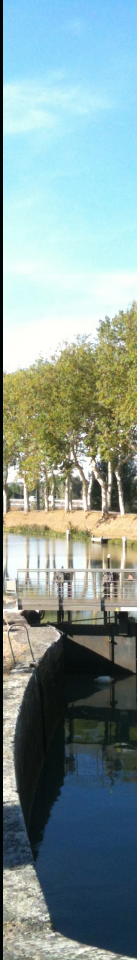
How can they be acquired?

Can the optional modifiers be acquired, at least the specialisation modifiers?

Relation to ontologies? (lexical vs. world-knwoledge ontologies)

What would be an adequate notion of subtyping? (for a systematic coding of the ontological specialisation relations that are often admissible in the language).

Extension to system F of coercive subtyping



E.3. Perspective 2: formulae vs. types

Typing ($\sim$ presupposition) is irrefutable $\mathit{sleeps}(x : \mathit{cat})$

Type to Formula: a type cat can be mirrored as a formula that can be refuted $\underline{\mathit{cat}} : \mathbf{e} \rightarrow \mathbf{t}$ $\quad \underline{\mathit{cat}}(x) : \mathbf{t}$

Formula to Type? Is any formula with a single free variable a type? $\mathit{cat}(x) \wedge \mathit{belong}(x, \mathit{john}) \wedge \mathit{sleeps}(x)$:type?

At least it is not an implicit comparison class nor a classifier.