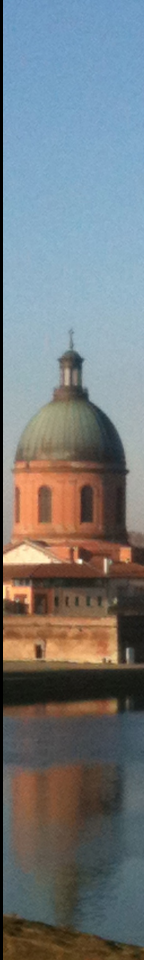


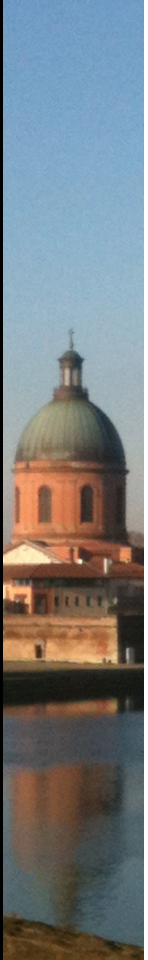
Type-theoretical natural-language semantics:

on the system F for meaning assembly

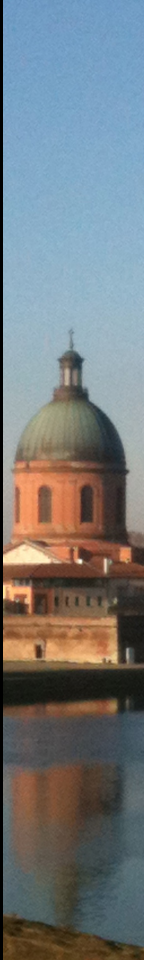
Christian Retoré

Université de Bordeaux & IRIT-CNRS Toulouse

Types 2013



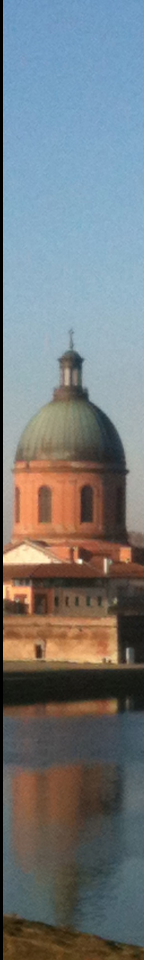
A Reminder on Montague semantics



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.



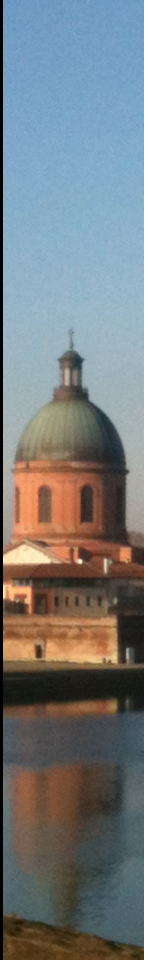
A.2. Representing formulae within lambda calculus — connectives

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 $(\mathbf{e} \to \mathbf{t}) \to \mathbf{t}$



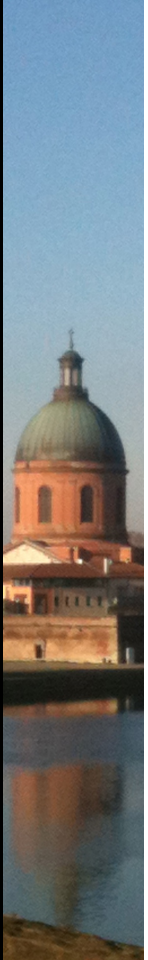
A.3. Representing formulae within lambda calculus — language constants

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

Garance	$\lambda P\ (P\ \text{Garance})$	$P : e \rightarrow t,\ \text{Garance} : e$
`<< Garance >> is viewed as the properties that << Garance >> holds`		

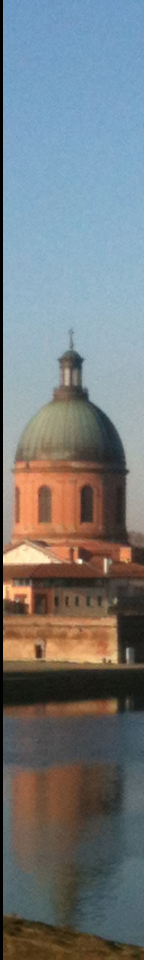


A.4. Normal terms of type $\mathbf{t}$ are formulae

Easy but important result

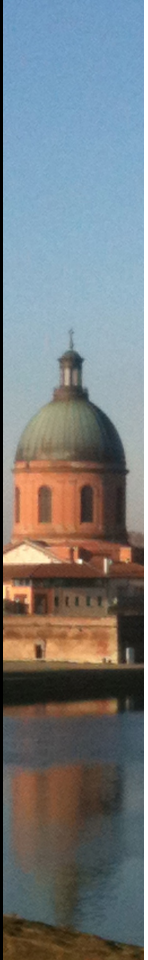
(induction on normal λ-terms preferably η-long):

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.



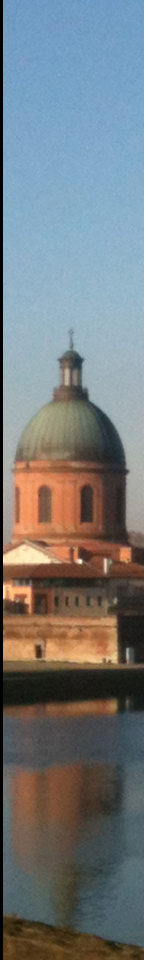
A.5. Montague semantics. Syntax/semantics.

$(\text{Syntactic type})^*$	$=$	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

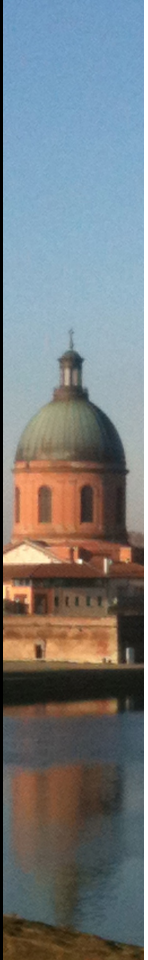
1. Replace in the lambda-term issued from the syntax the words by the corresponding term of the lexicon.
2. Reduce the resulting λ-term of type t its normal form corresponds to a formula, the ”meaning”.



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



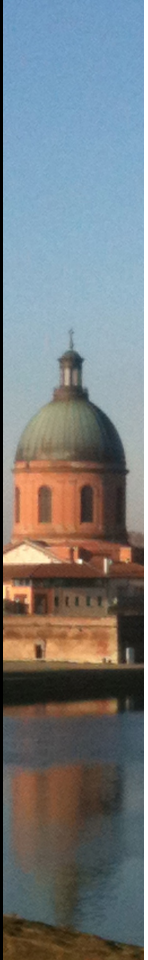
A.8. Computing the semantic representation

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

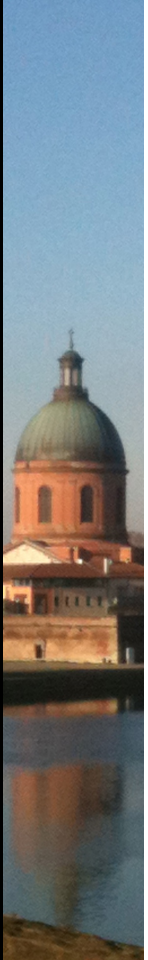
formulae another logic with first order (formulae/meaning no proofs)

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

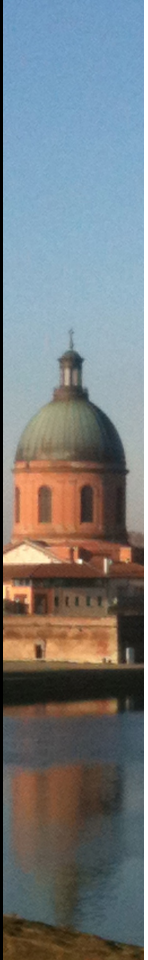
* The chair barked.

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 ΛTy_n: an instance of F for semantics



B.1. System F

Types:

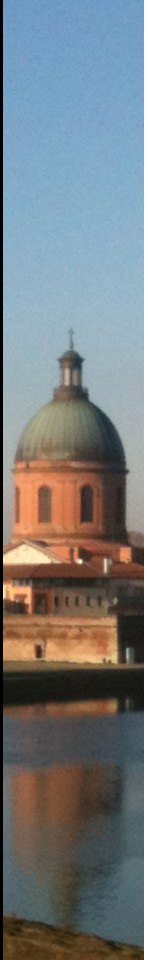
- $\mathbf{t}$ (prop)
- many entity types $\mathbf{e}_i$
- type variables $\alpha, \beta, \ldots$
- $\Pi\alpha.\ T$
- $T_1 \to T_2$

Terms

- Constants and variables for each type
- $(f^{T \to U} a^T) : U$
- $(\lambda x^T.\ u^U) : T \to U$
- $t^{(\Lambda\alpha.\ T)}\{U\} : T[U/\alpha]$
- $\Lambda\alpha.u^T : \Pi\alpha.T$ — no free α in a free variable of u.

The reduction is defined as follows:

- $(\Lambda\alpha.\tau)\{U\}$ reduces to $\tau[U/\alpha]$ (remember that α and U are types).
- $(\lambda x.\tau)u$ reduces to $\tau[u/x]$ (usual reduction).



B.2. Unnecessary type operators

The following defined types have the same elimination and introduction behaviour.

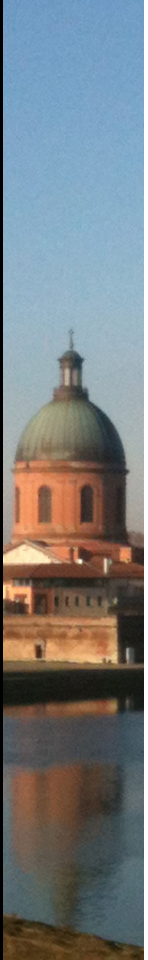
(1) Product $A \wedge B$ can be defined as
$\Pi\alpha.\ (A \to B \to \alpha) \to \alpha$

(2) Sum $A \vee B$ can be defined as
$\Pi\alpha.\ (A \to \alpha) \to (B \to \alpha) \to \alpha$

(3) Existential quantification:
$\Sigma\beta.\ ((\Pi\alpha.\ (V[X] \to \beta)) \to \beta)$

(4) Inductive types (Church numerals, lists, trees, etc.)

A problem: encodings are unnatural. On going work: include predefined types (e.g. Gödel's integers of system T).



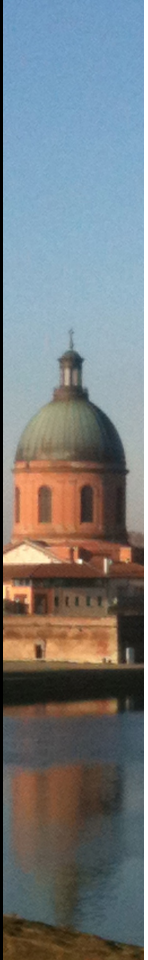
B.3. Basic facts on system F

We do not really need system F but any type system with quantification offer types. F is syntactically the simplest.

Confluence and strong normalisation — requires the comprehension axiom for all formulae of HA_2. (Girard 1971)

A concrete categorical interpretation with coherence spaces that shows that there are distinct functions from A to B.

Terms of type $\mathbf{t}$ with constants of mutisorted FOL (resp. HOL) correspond to multisorted formulae of FOL (resp. HOL)



B.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:

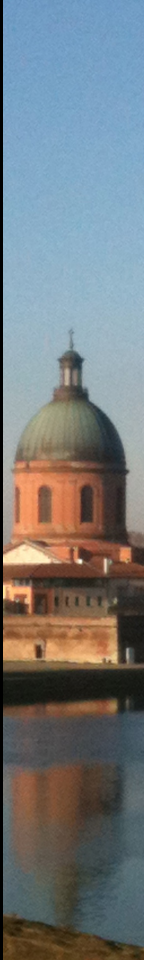
Given predicates $P^{\alpha\to\mathbf{t}}$, $Q^{\beta\to\mathbf{t}}$ over entities of respective types α, β,

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

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



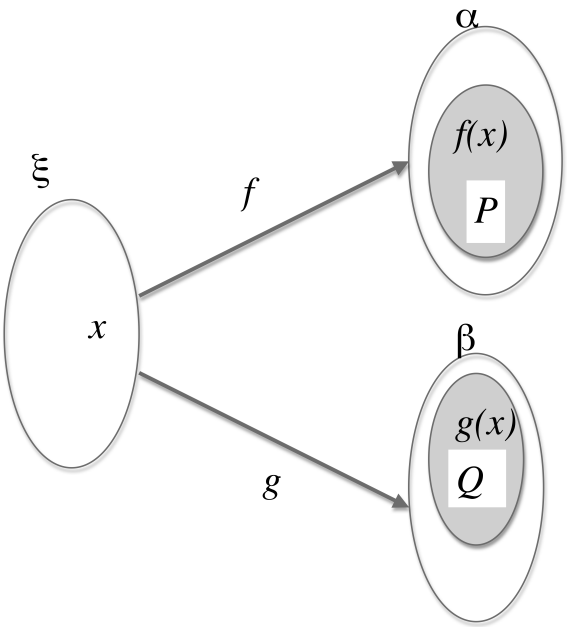
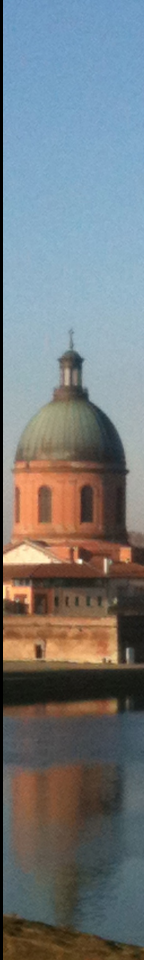
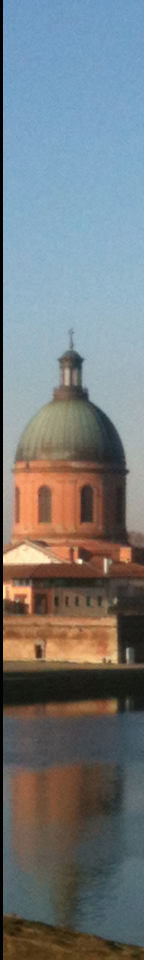


Figure 1: Polymorphic conjunction: $P(f(x))\&Q(g(x))$ with $x : \xi$, $f : \xi \to \alpha$, $g : \xi \to \beta$.



C System F based semantics and pragmatics



C.1. Examples

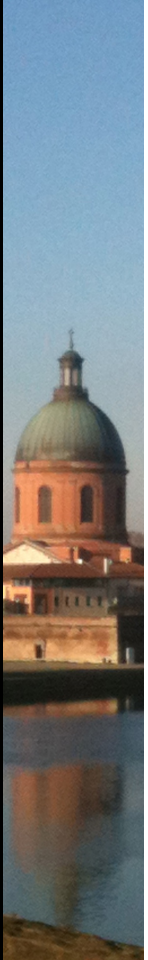
(1) Dinner was delicious but took ages.
(event / food)

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

(3) 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)

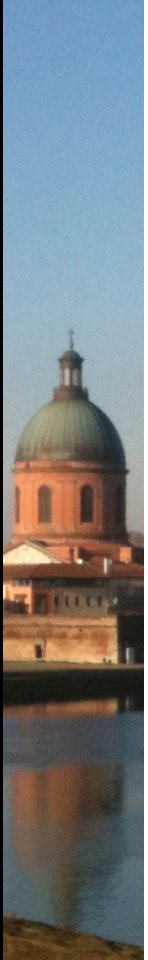
(4) Liverpool is a big place and voted last Sunday.
(geographic / people)

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



C.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).



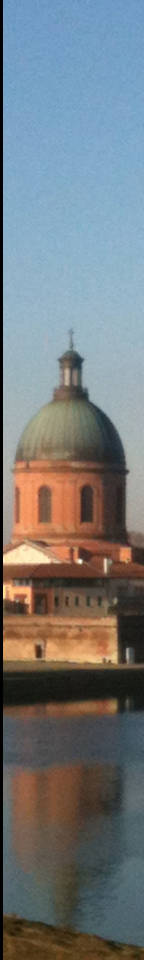
C.3. The Terms: principal or optional

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,* $\varnothing$



C.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.

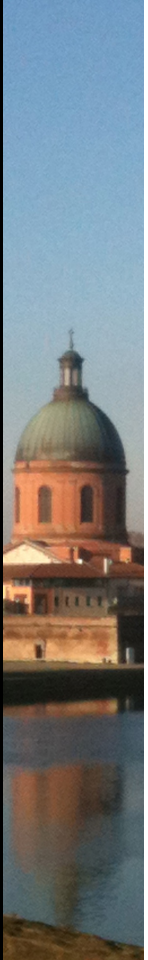
Each time we refer to the word it is with the same aspect.

FLEXIBLE

There is no constraint.

Any subset of the flexible transformation can be used:

different aspects of the words can be simultaneously used.



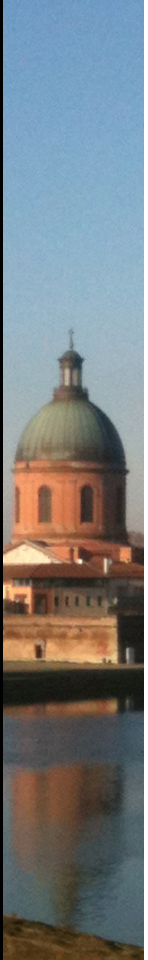
C.5. Standard behaviour

ϕ: physical objects

small stone

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

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

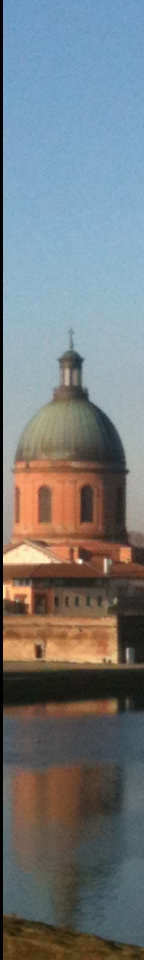


C.6. Correct copredication

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



C.7. Liverpool is a big place

Type mismatch:

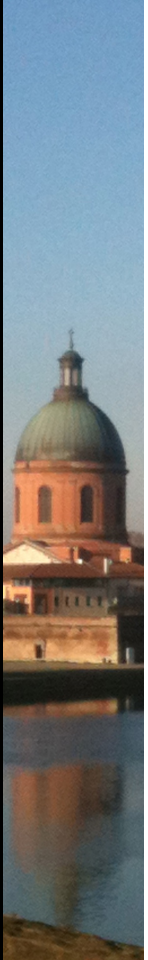
$$big\_place^{Pl \to \mathbf{t}}(Liverpool^{T}))$$

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.



C.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

$$\&^{\Pi}\{Pl\}\{P\}(is\_wide)^{Pl\to\mathbf{t}}(voted)^{P\to\mathbf{t}}$$

It reduces to:

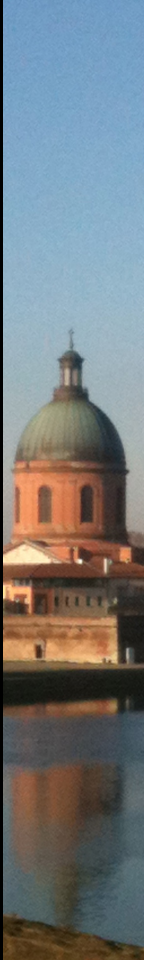
$$\Lambda\xi\lambda x^{\xi}\ \lambda f^{\xi\to\alpha}\lambda g^{\xi\to\beta}(\mathrm{and}^{\mathbf{t}\to\mathbf{t})\to\mathbf{t}}\ (is\_wide\ (f\ x))(voted\ (g\ x)))$$

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

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



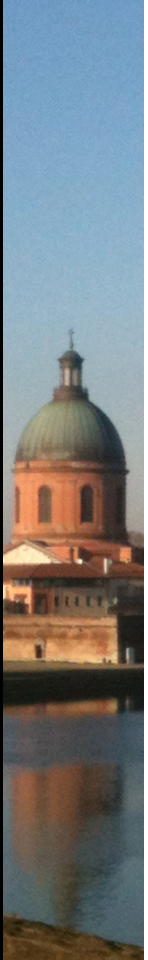
C.9. Liverpool voted and won

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

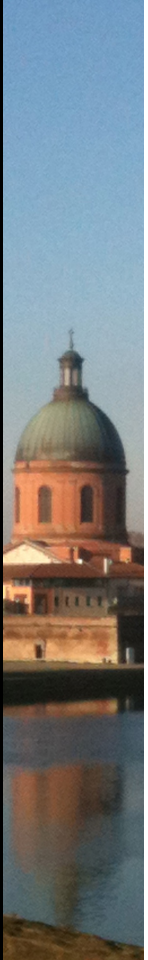
The term is:
$\lambda f^{T \to Pl} \lambda g^{T \to P}(\text{and}\ (won\ (f\ Liverpool^{T}))(voted\ (g\ Liverpool^{T}))))$

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 Integrating other aspects



D.1. Quantifier: critics of the standard solution

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

(6) Keith played some Beatles songs.

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

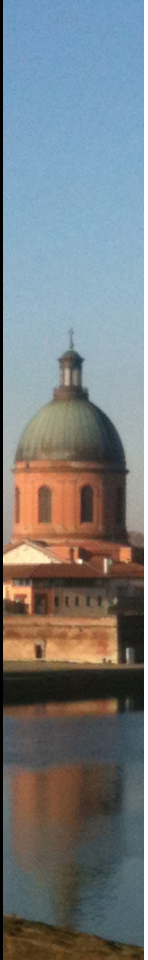
(8) semantics: (some (Beatles songs)) ($\lambda x.$ Keith played x)

Asymmetry class / predicate

(9) Some politicians are crooks

(10) ? Some crooks are politicians

(11) $\exists x.\ \mathit{crook}(x) \& \mathit{politician}(x)$



D.2. A solution: Hilbert's epsilon

$\varepsilon : \Lambda\alpha(\alpha \to \mathbf{t}) \to \alpha$ —- remember $F(\varepsilon_x F) \equiv \exists x.\ F(x)$.

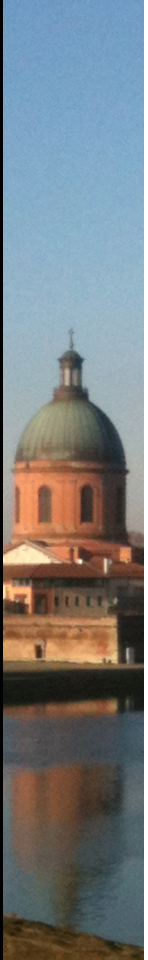
Follows syntactical structure. General presupposition $F(\varepsilon_x F)$ is added.

Also solves the so-called E-type pronouns interpretation:

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

(13) $\text{"}He\text{"} = \text{"}Aman\text{"} = (\varepsilon_x\ M(x))$.

For applying ε a type say cat, a type have predicative counterpart cat (type) $\widehat{cat} : \mathbf{e} \to \mathbf{t}$. (if needed domains can be restrained / extended)



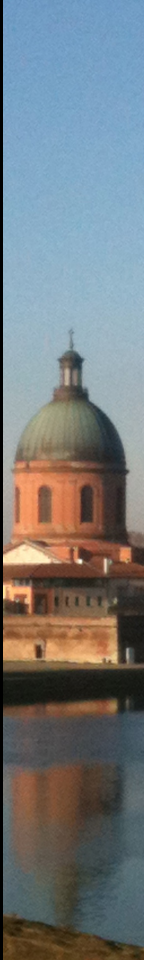
D.3. Other application in natural language semantics

Generalised quantifiers (most)

Plurals

Virtual traveller / fictive motion

Deverbals: “*The signature of the contract took ages / is unreadable.*”



D.4. On going work: adding coercive subtyping (as Luo & Soloviev)

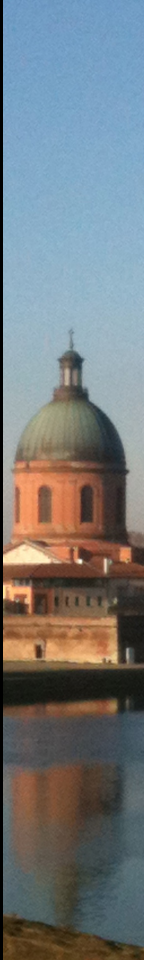
coercive application $\dfrac{f : A \to B \quad u : A_0 \quad A_0 < A}{(f\ a) : B}$

$$\frac{A < B \quad C < D}{B \to A < C \to D} \qquad \frac{A < B}{X \to A < X \to B} \qquad \frac{A < B}{B \to X < A \to X}$$

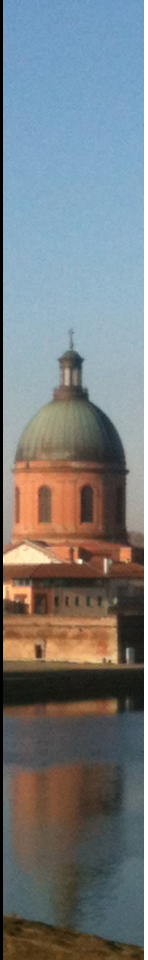
$$\frac{S[X] < T[X]}{\Pi X.S[X] < \Pi X.T[X]} \qquad \frac{U < T[X]}{U < \Pi X.T[X]}\ \textit{no free } X \textit{ in } U \qquad \frac{S[W] < U}{\Pi X.S[X] < U}$$

$$\frac{U < \Pi X.T[X]}{U < T[A]} \qquad \frac{\Pi X.S[X] < U}{S[A] < U}$$

On going work: transitivity of $<$ is unnecessary (by induction on formula degree, number of intermediate formulae with maximal degree, depth of intermediate formulae of maximum degree)



E Conclusion



E.1. What we have seen so far

A **general framework** for

the logical syntax of **compositional semantics**
some **lexical semantics/pragmatics** phenomena

Guidelines:

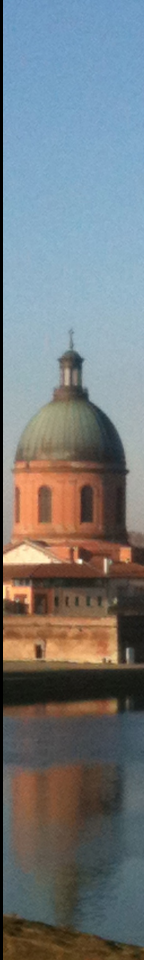
Terms: semantics, instructions for computing references
Types: pragmatics, defined from the context

Idiosyncratic (language specific) transformations compatible with the types but not forced by the types.

(14) Mon vélo est crevé. /??? My bike is flat.

(15) Classe $\rightarrow$ room promotion $\not\rightarrow$ room

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



E.2. Perspective

Pursue on coercive sub typing and predefined inductive types

What are the base types? Defined or acquired?

Linear types for more complex incompatibilities.

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

Cohabitation of types and formulae of first/higher order logic:

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

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

Formula to Type? Is any formula with a single free variable a type? $cat(x) \wedge belong(x, john) \wedge sleeps(x)$:type? At least it is not a natural comparison class.