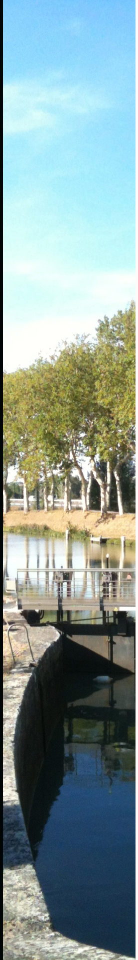


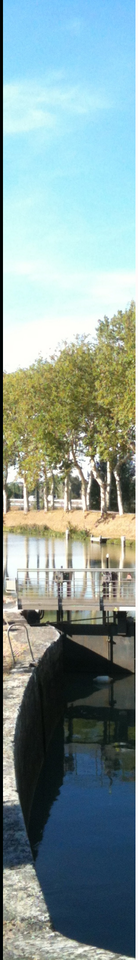
Natural Language Semantics and Computability

Richard Moot and Christian Retoré

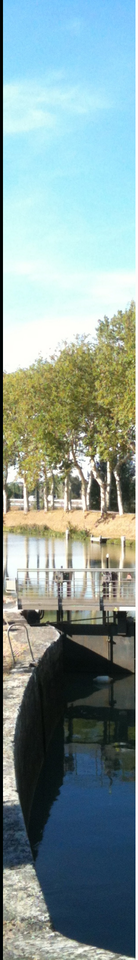
CNRS (LaBRI) and Université de Montepellier (LIRMM)

Computability in Europe

27 june 2016



A Computational Semantics



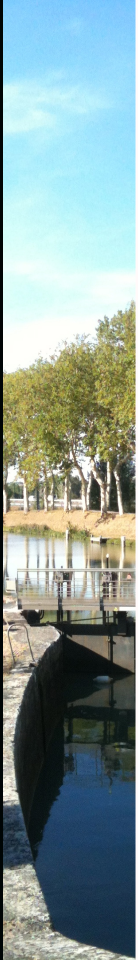
A.1. Computational semantics à la Montague

Method for transforming natural language sentences into formulas of higher-order logic.

John seeks a unicorn

$$\downarrow$$

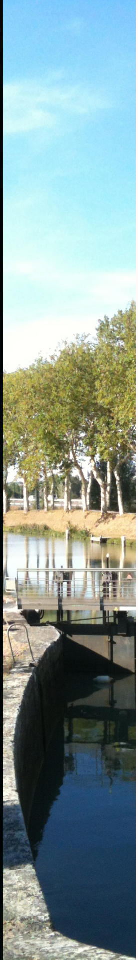
1. $\exists x.\mathtt{unicorn}(x) \wedge \mathtt{seek}(\mathtt{John}, x)$
2. $\mathtt{seek}(\mathtt{John}, \lambda P.\exists x.\mathtt{unicorn}(x) \wedge (P\, x))$



A.2. Applications: textual entailment

Scott Island is part of the Ross Dependency, claimed by New Zealand

1. Scott Island belongs to the Ross Dependency.
2. Scott Island belongs to New Zealand.



A.3. Computing meaning in categorial grammar

Lambek calculus proof

$\downarrow^{*}$

(multiplicative) intuitionistic linear logic proof

$\equiv_{\textit{Curry-Howard}}$

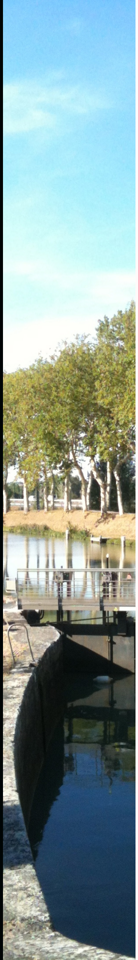
(linear) lambda term

$\downarrow_{\textit{lex}}$

Substitute the lexical (simply typed,
but not necessarily linear!) lambda terms.

$\downarrow_{\beta}$

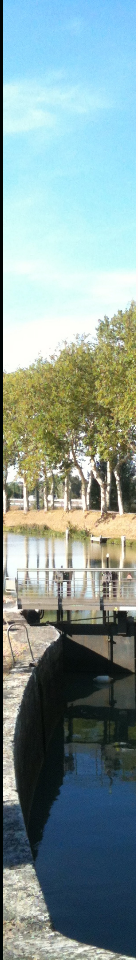
Target language:
Higher-Order Logic (HOL, as Montague)



A.4. Syntax: Lambek calculus

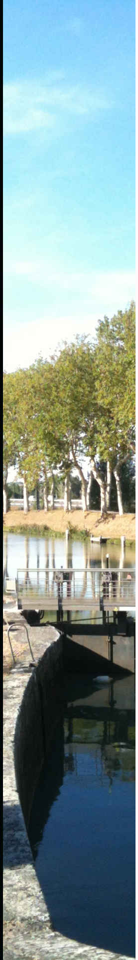
$$\frac{A/B \quad B}{A}\,/E \qquad \frac{B \quad B\backslash A}{A}\,\backslash E$$

$$\dfrac{\begin{array}{c} \ldots \quad [B]^i \\ \vdots \\ A \end{array}}{A/B}\,/I_i \qquad \dfrac{\begin{array}{c} [B]^i \quad \ldots \\ \vdots \\ A \end{array}}{B\backslash A}\,\backslash I_i$$



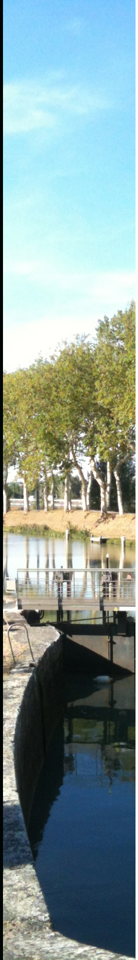
A.5. Example

everyone	contests	a	penalty
$s/(np\backslash s)$	$(np\backslash s)/np$	$((s/np)\backslash s)/n$	n



A.6. Example

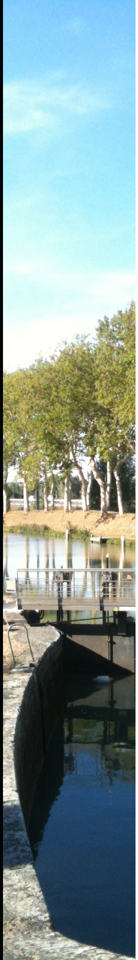
$$\begin{array}{ccc} \text{everyone} & \text{contests} & \text{a penalty} \\ s/(np\backslash s) & (np\backslash s)/np & \dfrac{((s/np)\backslash s)/n \qquad n}{(s/np)\backslash s}\,/E \end{array}$$



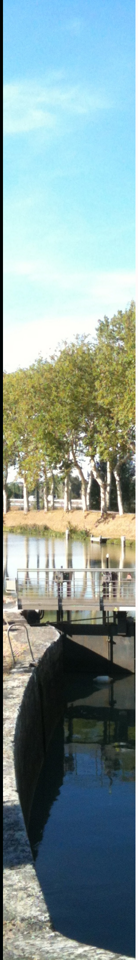
A.7. Example

$$
\begin{array}{cccc}
\text{everyone} & \text{contests} & & \text{a penalty} \\
s/(np\backslash s) & (np\backslash s)/np & np & \dfrac{((s/np)\backslash s)/n \quad n}{(s/np)\backslash s}\,/E
\end{array}
$$

A.8. Example

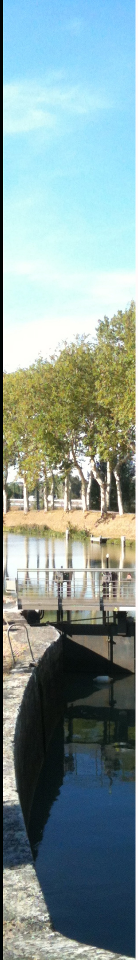


$$\begin{array}{ccc}
\text{everyone} & \text{contests} & \text{a penalty} \\
s/(np\backslash s) & \dfrac{(np\backslash s)/np \quad np}{np\backslash s}\,/E & \dfrac{((s/np)\backslash s)/n \quad n}{(s/np)\backslash s}\,/E
\end{array}$$



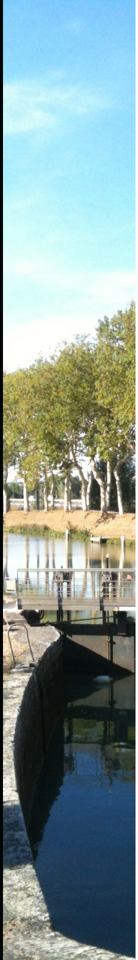
A.9. Example

$$\dfrac{\dfrac{\dfrac{s/(np\backslash s) \qquad \dfrac{(np\backslash s)/np \quad [np]^1}{np\backslash s}\,/E}{s}\,/E}{s/np}\,/I_1 \qquad \dfrac{((s/np)\backslash s)/n \quad n}{(s/np)\backslash s}\,/E}{s}\,/E$$



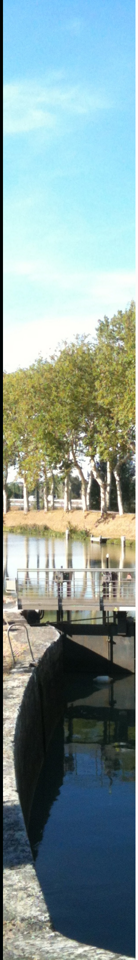
A.10. Syntax and semantics

(Syntactic type)*	$=$	Semantic type	
s^*	$=$	t	a sentence is a proposition
np^*	$=$	e	a pronoun is an entity
n^*	$=$	$e \rightarrow t$	a noun is a set of entities
$(a \backslash b)^* = (b/a)^*$	$=$	$a^* \rightarrow b^*$	extends $(\_)^*$ to all formulas



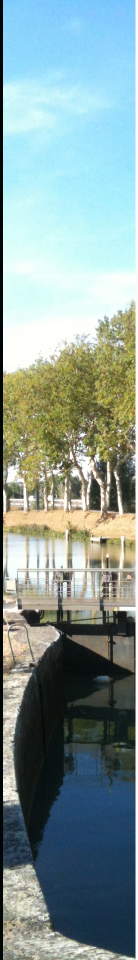
A.11. Syntactic proof

$$\dfrac{\dfrac{\dfrac{s/(np\backslash s) \qquad \dfrac{(np\backslash s)/np \qquad [np]^1}{np\backslash s}\,/E}{s}\,/E}{s/np}\,/I_1 \qquad \dfrac{((s/np)\backslash s)/n \qquad n}{(s/np)\backslash s}\,/E}{s}\,/E$$



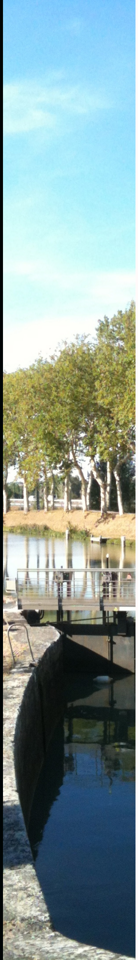
A.12. Semantic proof

$$\dfrac{\dfrac{\dfrac{(e \to t) \to t \qquad \dfrac{e \to (e \to t) \qquad [e]^1}{e \to t}\ {\to}E}{t}\ {\to}E}{e \to t}\ {\to}I_1 \qquad \dfrac{(e \to t) \to ((e \to t) \to t) \qquad e \to t}{(e \to t) \to t}\ {\to}E}{t}\ {\to}E$$



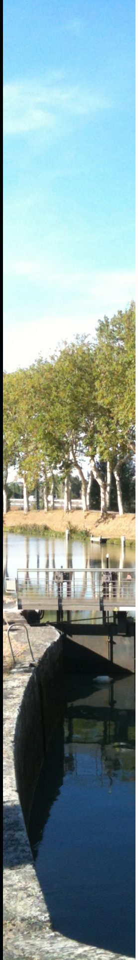
A.13. Curry-Howard isomorphism

$$
\dfrac{\dfrac{\dfrac{e^{(e\to t)\to t} \qquad \dfrac{c^{e\to(e\to t)} \qquad [x^e]^1}{(c\ x)^{e\to t}}\to E}{(e\ (c\ x))^t}\to E}{(\lambda x.(e\ (c\ x)))^{e\to t}}\to I_1 \qquad \dfrac{a^{(e\to t)\to((e\to t)\to t)} \qquad p^{e\to t}}{(a\ p)^{(e\to t)\to t}}\to E}{((a\ p)(\lambda x.(e\ (c\ x))))^t}\to E
$$



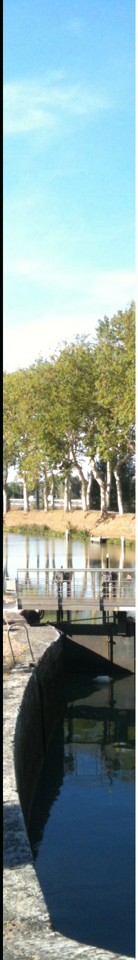
A.14. Semantic lexicon

Word	***semantic type*** u^* ***semantic term:*** λ***-term of type*** u^* x^v *the variable or the constant* x *is of type* v
everyone	$(e \to t) \to t$ $\lambda P^{e\to t} \forall^{(e\to t)\to t}(\lambda x^e (P\ x))$
contests	$e \to (e \to t)$ $\lambda y^e\ \lambda x^e\ ((\mathtt{contests}^{e\to(e\to t)}\ x)\ y)$
a	$(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e (\wedge^{t\to(t\to t)}(P\ x)(Q\ x))))$
penalty	$e \to t$ $\lambda x^e (\mathtt{penalty}^{e\to t}\ x)$



A.15. Simplified semantic lexicon

Mot	***semantic type*** u^* ***semantic term:*** λ***-terme de type*** u^* x^v *the variable or the constant* x *is of type* v
everyone	$(e \to t) \to t$ $\lambda P^{e\to t} \forall x^e (P\ x)$
contests	$e \to (e \to t)$ $\lambda y^e\ \lambda x^e\ \mathtt{contests}(x, y)$
a	$(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\ \lambda Q^{e\to t}\ \exists x^e (P\ x) \wedge (Q\ x)$
penalty	$e \to t$ $\lambda x^e \mathtt{penalty}(x)$

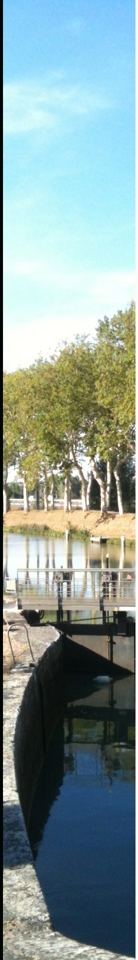


A.16. Substitution

$$(a\ p) \equiv (\lambda P^{e\to t}\ \lambda Q^{e\to t}\ \exists x^e (P\ x) \wedge (Q\ x))\ \lambda y^e \texttt{penalty}(y)$$

$$\lambda Q^{e\to t}\ \exists x^e ((\lambda y^e \texttt{penalty}(y))\ x) \wedge (Q\ x))$$

$$\lambda Q^{e\to t}\ \exists x^e \texttt{penalty}(x) \wedge (Q\ x)$$



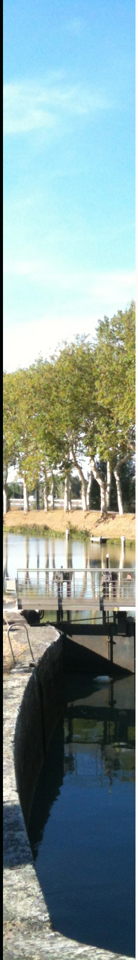
A.17. Substitution

$$(e\ (c\ x)) \equiv (\lambda P^{e\to t} \forall y^e (P\ y))((\lambda z^e\ \lambda v^e\ \texttt{contests}(v, z))\ x)$$

$$(\lambda P^{e\to t} \forall y^e (P\ y))(\lambda v^e\ \texttt{contests}(v, x))$$

$$\forall y^e ((\lambda v^e\ \texttt{contests}(v, x))\ y)$$

$$\forall y^e \texttt{contests}(y, x)$$

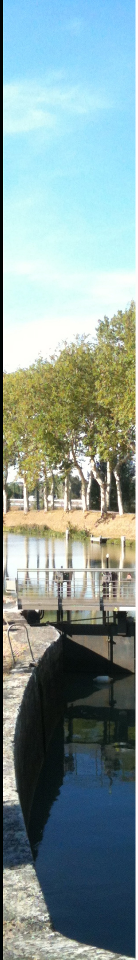


A.18. Substitution

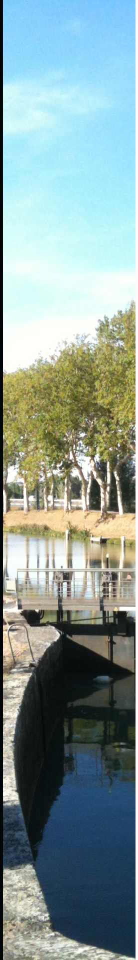
$$
\begin{aligned}
&((a\ p)(\lambda x.(e\ (c\ x))))\\
&\qquad\qquad \equiv (\lambda Q^{e\to t}\ \exists x^{e} \texttt{penalty}(x) \wedge (Q\ x))\\
&\qquad\qquad\qquad (\lambda y.\forall z^{e} \texttt{contests}(z, y))
\end{aligned}
$$

$$\exists x^{e} \texttt{penalty}(x) \wedge ((\lambda y.\forall z^{e} \texttt{contests}(z, y))\ x)$$

$$\exists x^{e} \texttt{penalty}(x) \wedge \forall z^{e} \texttt{contests}(z, x)$$

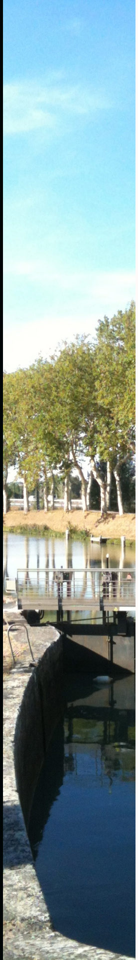


B Complexity



B.1. What is the precise complexity question?

1. is there a reading?
2. what is the best/most likely reading?
3. what are all possible readings?

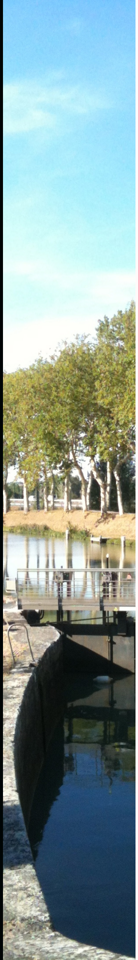


B.2. Complexity of the syntax

Finding a proof for the Lambek calculus (and many of its extensions) is NP complete.
A sentence with n quantifiers can have up to $n!$ readings. A simple counting argument shows that the Lambek calculus (though not its extensions) cannot generate all readings for an n quantifier sentence as distinct proofs.

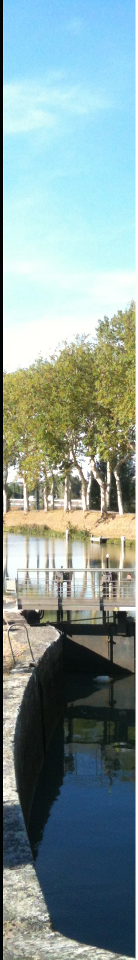
Open question is there an algorithm producing shared meaning representations with the following properties:

1. the algorithm outputs no when the sentence is ungrammatical,
2. there is a fairly simple algorithm (say of a low-degree polynomial at worst) for recovering all readings from the shared representation,
3. the shared structure is polynomial in the size of the input.



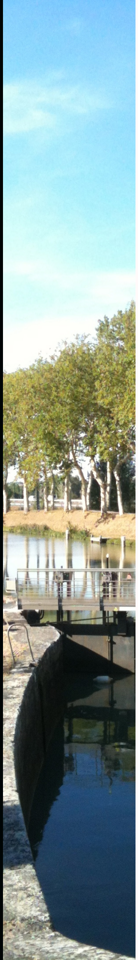
B.3. Complexity of the semantics: normalization

- Normalizing simply typed lambda terms is known to be of non-elementary complexity (Schwichtenberg 1982).
- In practice, this is not a big bottleneck (Bos et al. 2004, Moot 2010).
- One explanation for this efficiency is that lambda terms used in grammars are in soft linear logic (Lafont 2004).

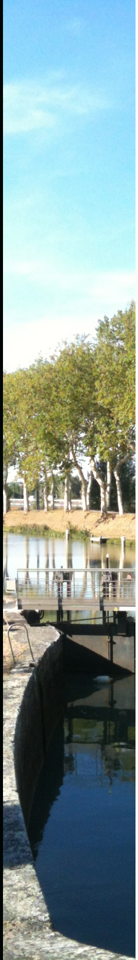


B.4. Complexity of the semantics: inference

- Formulas in higher-order logic, though of independent interest to formal semanticists, are only as useful as what we can do with them.
- Already for first-order logic, logical inference is of course undecidable.
- In spite of this, there are a number of inference systems which perform fairly well (generally, this means high precision combined with low recall).



C Conclusion



C.1. Conclusion

1. Though the complexity of parsing has been widely studied, much more remains to be done for computational semantics.

2. In many cases (extensions of the Lambek calculus combined with soft linear logic) computing the semantics is NP complete.