



Deep Types for Categorical Grammar

A Side Effect Analysis

LAURA KORTE

Laboratory for Foundations of Computer Science

School of Informatics

University of Edinburgh

L.Korte@sms.ed.ac.uk



Outline

1. Introduction
2. Impure Functional Programming Languages
3. Deep Types for Impure Functional Programming Languages
4. Linguistic Side Effects
5. Impure Categorical Grammar
6. Deep Types for Impure Categorical Grammar
7. Discussion

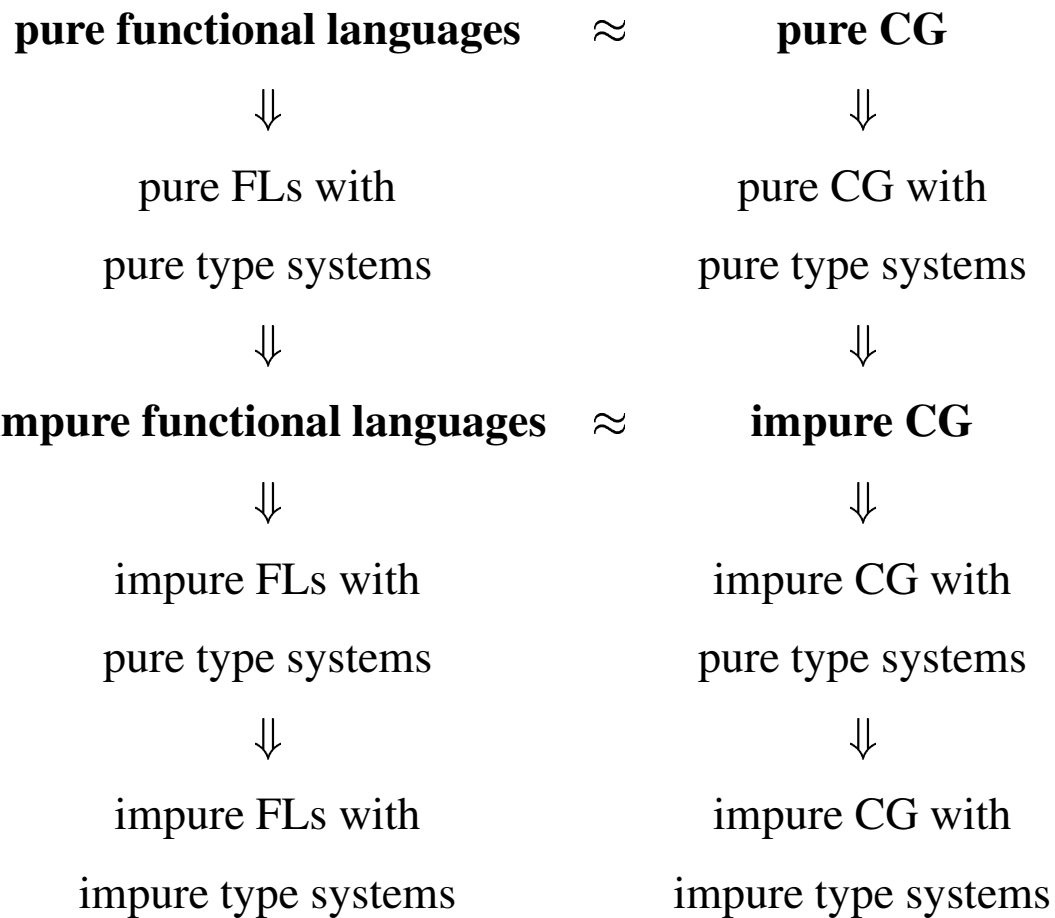
Introduction

- Interdisciplinary Research: research in one field may lead to progress in the other.
- Close relation between categorial grammar and functional programming:

semantic term	vs.	program
syntactic type	vs.	functional type

- CS can learn about tractable (weakly) context sensitive parsing from CCG.
- CCG can learn about advanced type systems and efficient implementations from CS.
- In our case: a deep type system for CCG.

Introduction



Impure Functional Programming Languages

adding side effects

Class	Program	λ -term	Rest
PURE	<code>fun id x = x</code>	$\lambda x.x$	
IMPURE	<code>fun badid x = (inc r ; x)</code>	$\lambda x.x$	<i>inc r</i>

Impure Functional Programming Languages

adding side effects

Expression	Side Effect
$x := y$	<i>update</i> reference x to value y
$!x$	<i>retrieve</i> the current value of reference x

Impure Functional Programming Languages

adding side effects

```
(**** Program ****)  
val r = ref 0  
fun inc x = (x := !x + 1)  
fun id x = x  
fun badid x = (inc r ; x)
```

expression	value of <i>r</i> after execution
inc r	1
id 4	0
badid 4	1
inc r ; badid 7	2
badid (badid 4)	2

Deep Types

for Impure Functional Programming Languages

When pure functional languages are extended to include imperative features with side effects, a type information gap arises. Deep types aim to narrow this gap by extending functional (or *shallow*) types with type-level information about the side effect behaviour of a program.

Class	Program	Type	Deep Type
PURE	<code>fun id x = x</code>	$\alpha \rightarrow \alpha$	$\alpha \longrightarrow \alpha$
IMPURE	<code>fun badid x = (inc y ; x)</code>	$\alpha \rightarrow \alpha$	$\alpha \xrightarrow{u(y)} \alpha$

Linguistic Side Effects

John shaves himself \Rightarrow (SHAVE J) J

Lexicon

john : *np* \Rightarrow $r := J$

shaves : *tv* \Rightarrow SHAVE

himself : *np* \Rightarrow ! r



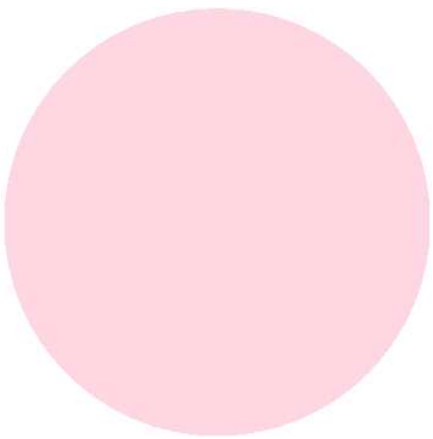
Impure Categorical Grammar

adding side effects

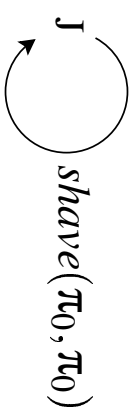
John shaves himself.

Lexicon

john	:	$[np]$	\Rightarrow	$NP_{3sm} := J; NP_{3sm}$
shaves	:	$([np] \setminus s) / [np]$	\Rightarrow	SHAVE
himself	:	$[np]$	\Rightarrow	NP_{3sm}



John shaves himself.



Impure Categorical Grammar

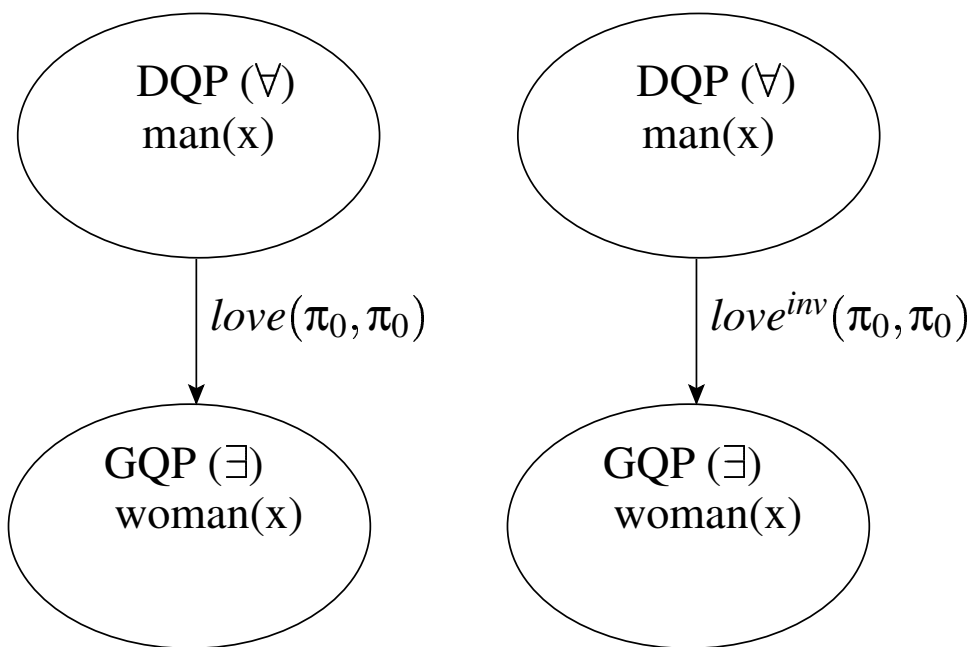
adding side effects

Every man loves a woman.

Lexicon

every man	: $[np]$	$\Rightarrow NP_{3sm} := (\{x man(x)\} : DQP); NP_{3sm}$
loves	: $([np] \setminus s) / [np]$	\Rightarrow LOVES
a woman	: $[np]$	$\Rightarrow NP_{3sf} := (\{y woman(y)\} : GQP); NP_{3sf}$

Every man loves a woman.



Quantifiers

(Beghelli and Stowell, 1997)

Interrogative (WhQP)	<i>what, which</i>
Negative (NQP)	<i>nobody, no</i>
Distributive-Universal (DQP)	<i>every, each</i>
Counting (CQP)	<i>few, at most five, more than two</i>
Group Denoting (GQP)	<i>a, some, the, several, three</i>

Quantifiers

Relation	Inverse?
$DQP \rightarrow GQP$	yes
$GQP \rightarrow DQP$	yes
$GQP \rightarrow CQP$	no
\vdots	\vdots

Impure Categorical Grammar

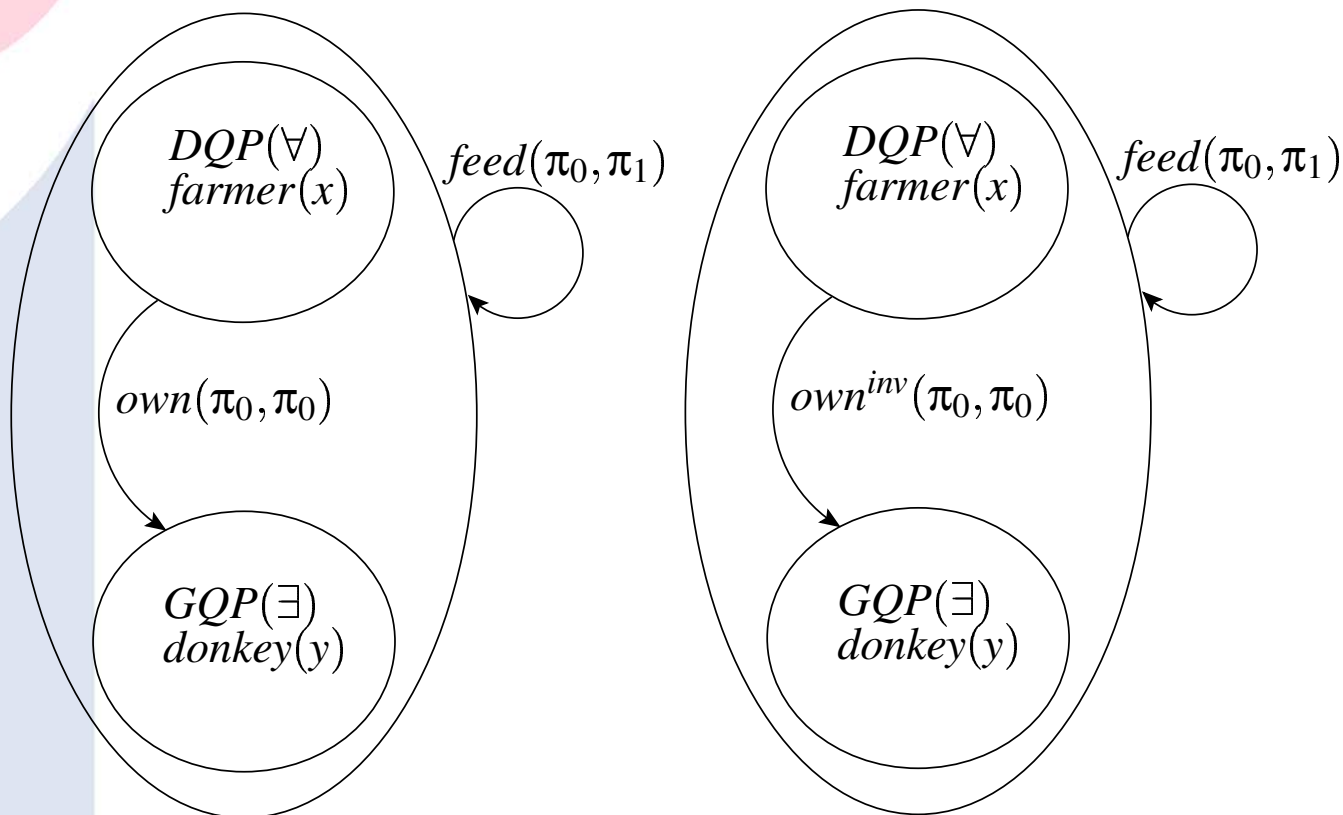
adding side effects

Every farmer who owns a donkey, feeds it.

Lexicon

every	:	$[np]/n$	\Rightarrow	$\lambda P.NP_{3s} := (\{x P(x)\} : DQP); NP_{3s}$
farmer	:	n	\Rightarrow	FARMER
who	:	$([np] \setminus [np]) / ([np] \setminus s)$	\Rightarrow	$open; (\lambda v.close; (\lambda r.(r := !r \cap \{x v(x)\}); r))$
owns	:	$([np] \setminus s) / [np]$	\Rightarrow	OWN
a	:	$[np]/n$	\Rightarrow	$\lambda P.NP_{3s} := (\{y P(y)\} : GQP); NP_{3s}$
donkey	:	n	\Rightarrow	DONKEY
feeds	:	$([np] \setminus s) / [np]$	\Rightarrow	FEED
it	:	$[np]$	\Rightarrow	NP_{3s}

Every farmer who owns a donkey, feeds it.



$$\frac{}{\langle w, \sigma \rangle \Downarrow \langle x : \tau, \sigma \rangle} \text{ (Axiom)}$$

- where $w \mapsto x : \tau$

$$\frac{\langle w_1, \sigma \rangle \Downarrow \langle x_1 : \tau_2 / \tau_1, \sigma' \rangle \quad \langle w_2, \sigma' \rangle \Downarrow \langle x_2 : \tau_1, \sigma'' \rangle}{\langle w_1 \bullet w_2, \sigma \rangle \Downarrow \langle x_1 x_2 : \tau_2, \sigma'' \rangle} (>)$$

- where x_1 and x_2 are in normal form

$$\frac{\langle w_1, \sigma \rangle \Downarrow \langle x_1 : \tau_1, \sigma' \rangle \quad \langle w_2, \sigma' \rangle \Downarrow \langle x_2 : \tau_1 \setminus \tau_2, \sigma'' \rangle}{\langle w_1 \bullet w_2, \sigma \rangle \Downarrow \langle x_2 x_1 : \tau_2, \sigma'' \rangle} (<)$$

- where x_1 and x_2 are in normal form

$$\frac{\langle w, \sigma \rangle \Downarrow \langle x : \tau, \sigma' \rangle \quad \langle x, \sigma' \rangle \Downarrow_c \langle x' : \tau, \sigma'' \rangle}{\langle w, \sigma \rangle \Downarrow \langle x' : \tau, \sigma'' \rangle} (T)$$

Impure Categorical Grammar

Conclusions

- System is extremely flexible, it can be molded to fit many domain specific needs.
- Computational complexity is no greater than that of CCG.
- Shifting scoping and anaphora resolution to the side effect domain, cleans up syntactic clutter.
- The system has the potential to provide solutions for other traditionally troublesome areas like intensionality, interrogatives, focus and presupposition because they can all be specified in term of side effects (Shan 2003).

A large pink circle is positioned in the upper left quadrant. A light blue, irregular shape is located in the lower left quadrant, partially overlapping the pink circle and extending towards the bottom left corner of the slide.

Deep Types for Impure Categorical Grammar

1. Necessary to specify complete behaviour.
2. Solving complexity issues (like Johnson and Morrill for TLG).
The cheese the mouse the rat the cat ate ate ate smelled horrible.
3. Syntactical + referential correctness checks (without deriving meaning).
He_? said he_? liked spaghetti.

Discussion

- Are there any specific examples we would like ICG to deal with?
- What are the exact implications of lowering the type of quantifiers to $[np]$?
- Suggestions for applications?
- Related work?