Dynamic Syntax in Type Theory with Records

Robin Cooper and Staffan Larsson
Centre for Linguistic Theory and Studies in Probability (CLASP)
Dept. of Philosophy, Linguistics and Theory of Science
University of Gothenburg

Supported by VR projects:
2009-1569, Semantic analysis of interaction and coordination in dialogue (SAICD);
2016-01162, Incremental Reasoning in Dialogue (IncReD);
2014-39, Centre for Linguistic Theory and Studies in Probability (CLASP)

Outline

DS in TTR

Using TTR contents in DS

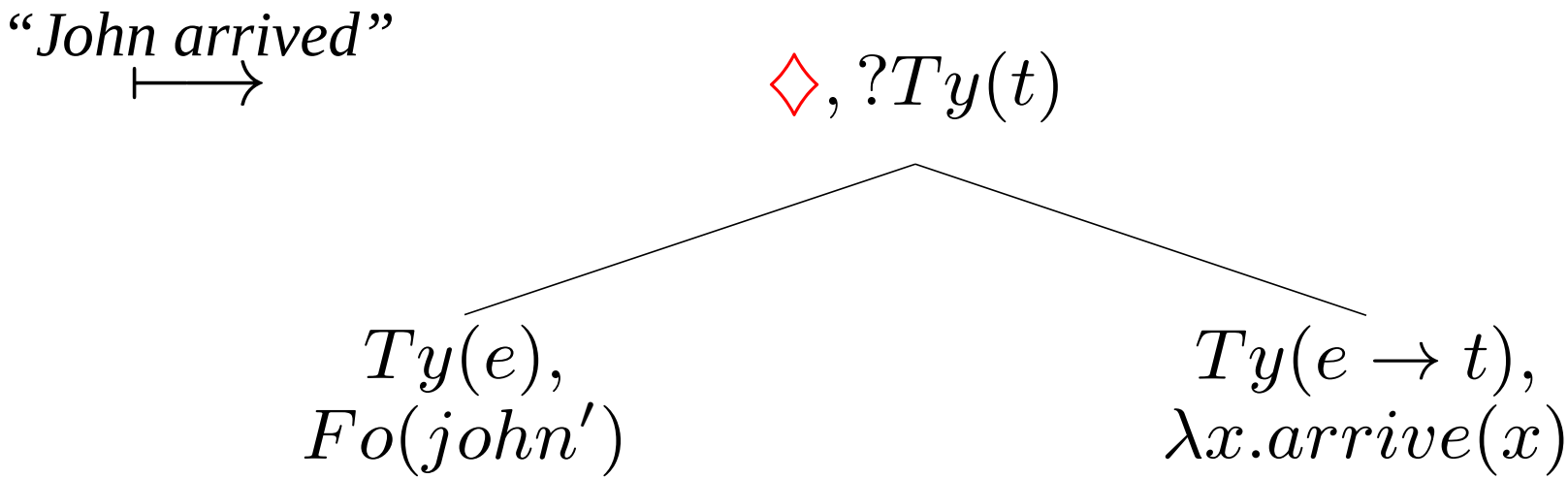
Adding speech events

Outline

Relating DS and TTR

- DS (Kempson *et al.*, 2001)
- TTR (Cooper, in prep, 2012; Cooper and Ginzburg, 2015)
- adding TTR for semantic representation (Eshghi, 2015)
- What might it look like to do everything in TTR?
- Is it even possible?

John arrived in DS



Tree nodes

Nodes seem to contain records of the type:

$$\begin{bmatrix} \text{type} & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{bmatrix}$$

Tree nodes

Nodes seem to contain records of the type:

$$\begin{bmatrix} \text{type} & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{bmatrix}$$

That is, in official notation:

$$\begin{bmatrix} \text{type} & : & \mathit{Type} \\ \text{cont} & : & \langle \lambda v{:}\mathit{Type}.v, \langle \text{type} \rangle \rangle \end{bmatrix}$$

Daughters

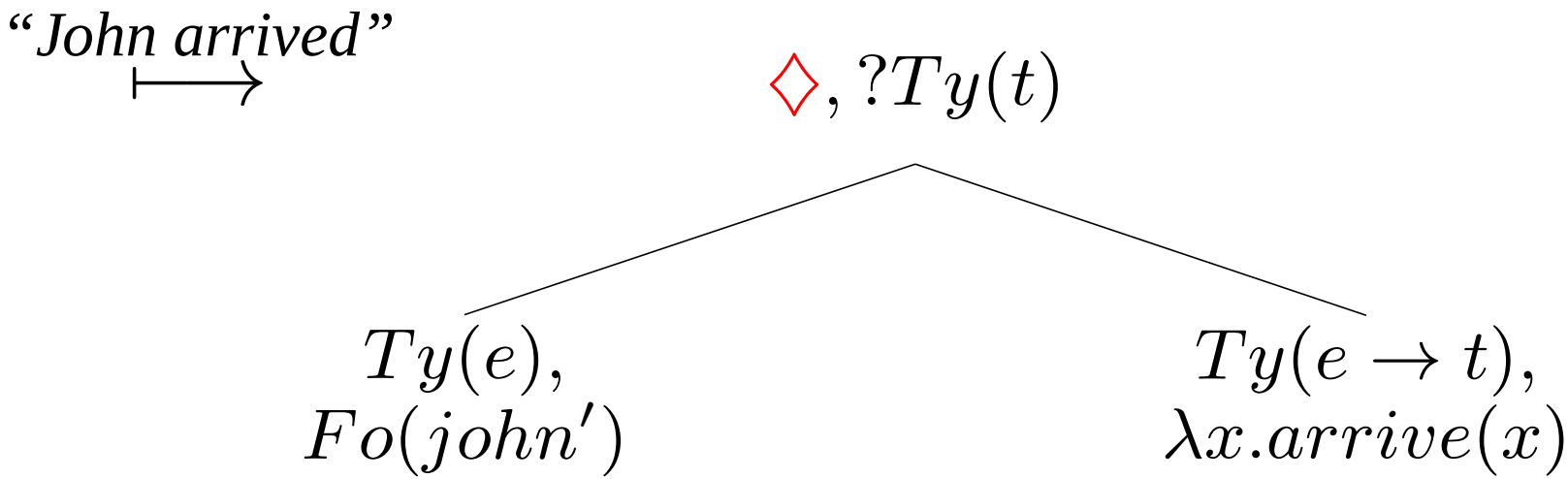
But tree nodes may have daughters.

Therefore we define a (basic, recursive) type *Tree* such that

$$a{:}\mathit{Tree} \text{ iff } a{:}\begin{bmatrix} \text{type} & : & \mathit{Type} \\ \text{cont} & : & \text{type} \\ \text{daughters} & : & \mathit{Tree}* \end{bmatrix}$$

*Tree** is the type of strings of trees (*cf.* Kleene-*) including the empty string, ϵ

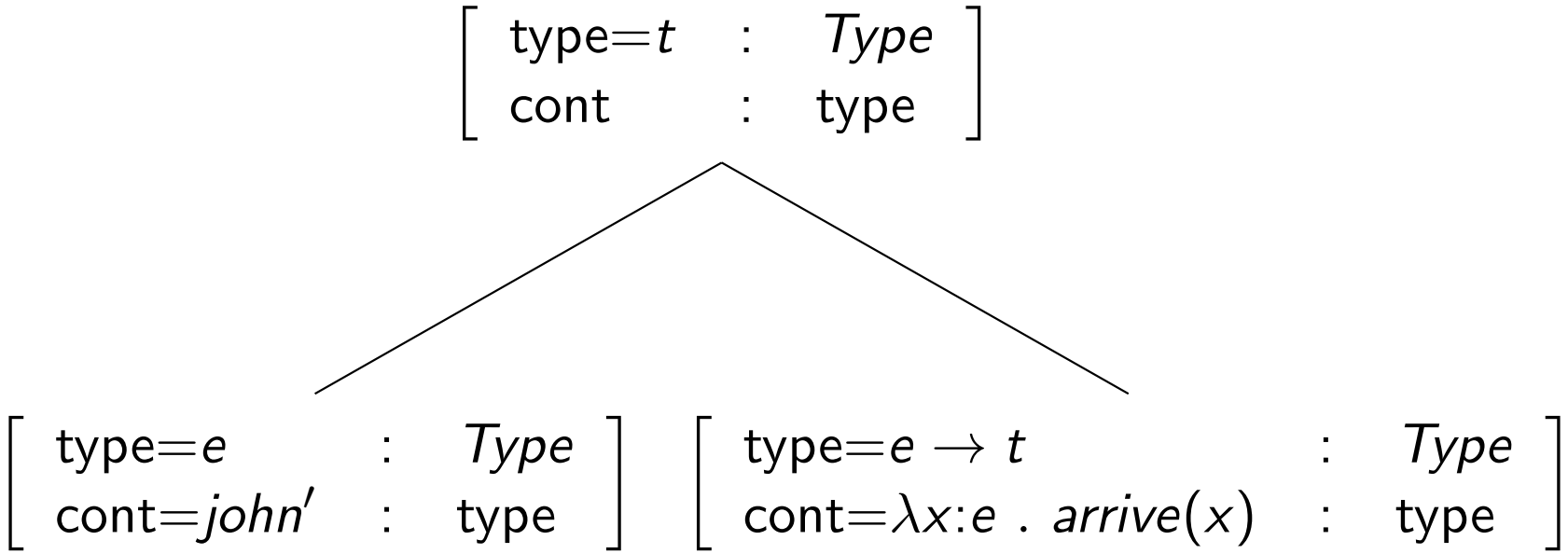
John arrived in DS (again)



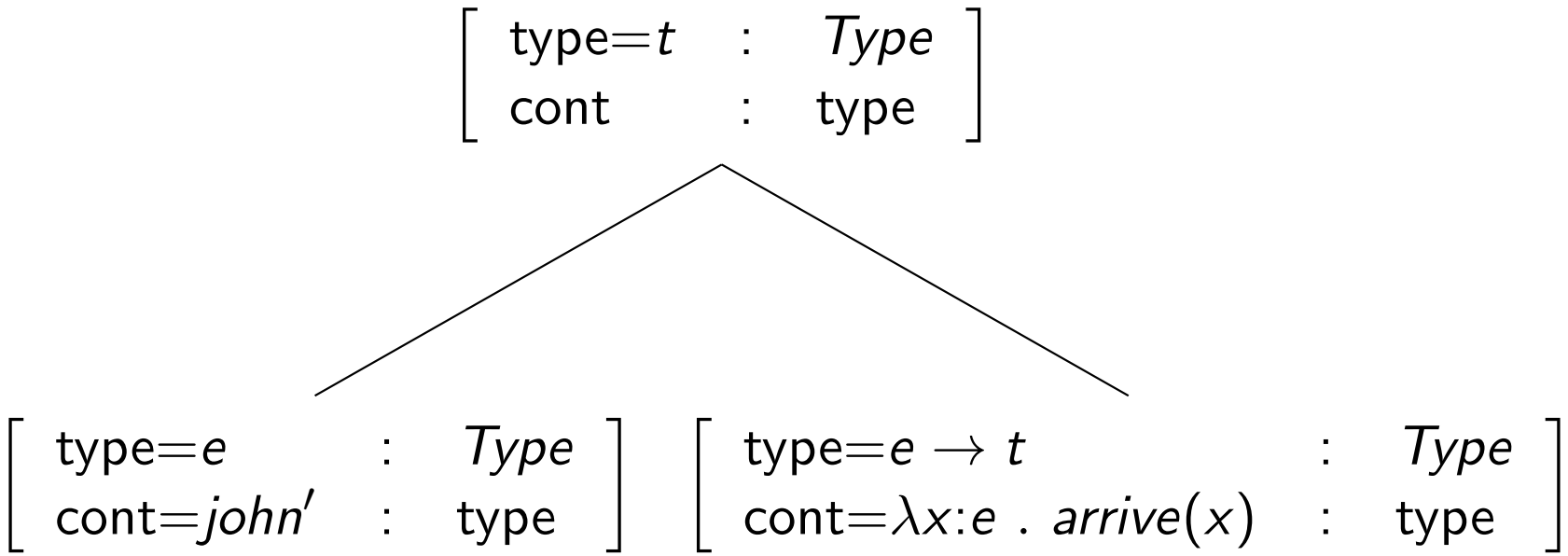
Type for the tree for *John arrived*

$$\left[\begin{array}{lcl}
\text{type}{=}t & : & \mathit{Type} \\
\text{cont} & : & \text{type} \\
\text{daughters} & : & \left[\begin{array}{lcl}
\text{type}{=}e & : & \mathit{Type} \\
\text{cont}{=}\mathit{john}' & : & \text{type} \\
\text{daughters}{=}\epsilon & : & \mathit{Tree}^*
\end{array}\right]^\frown \\
 & & \left[\begin{array}{lcl}
\text{type}{=}e \rightarrow t & : & \mathit{Type} \\
\text{cont}{=}\lambda x{:}e\ .\ \mathit{arrive}(x) & : & \text{type} \\
\text{daughters}{=}\epsilon & : & \mathit{Tree}^*
\end{array}\right]
\end{array}\right]$$

or more diagrammatically . . .

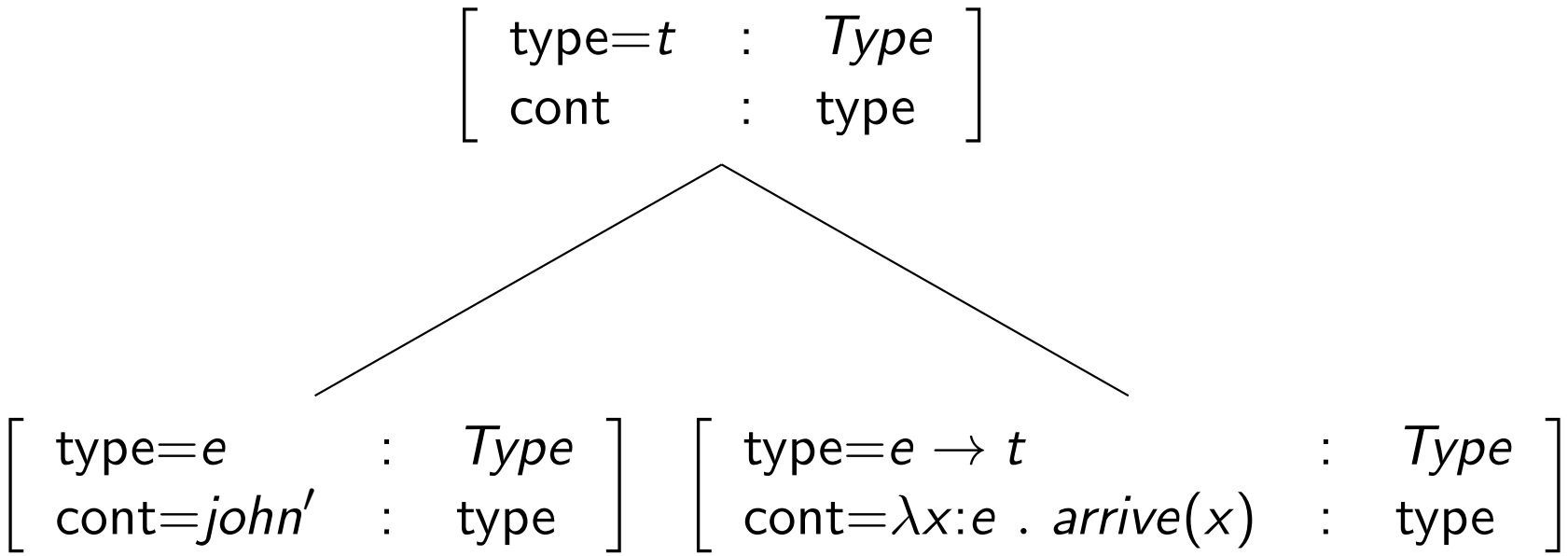


or more diagrammatically . . .



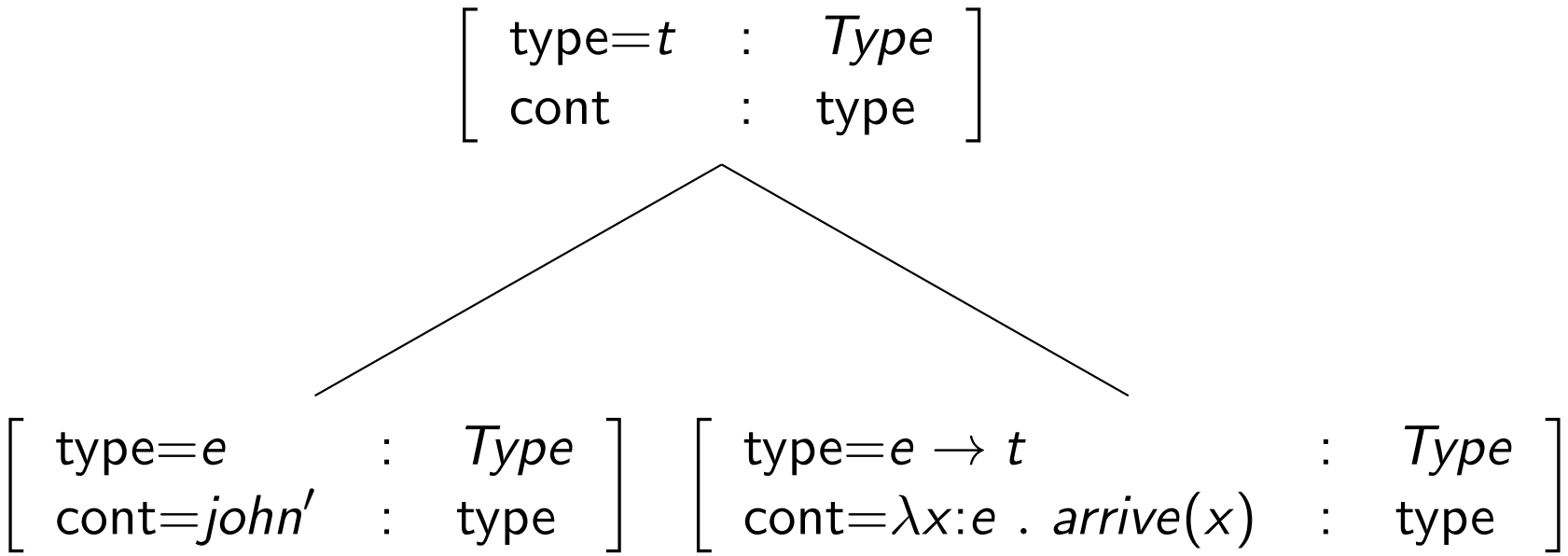
- Note that this represents a tree *type*, not a tree

or more diagrammatically . . .



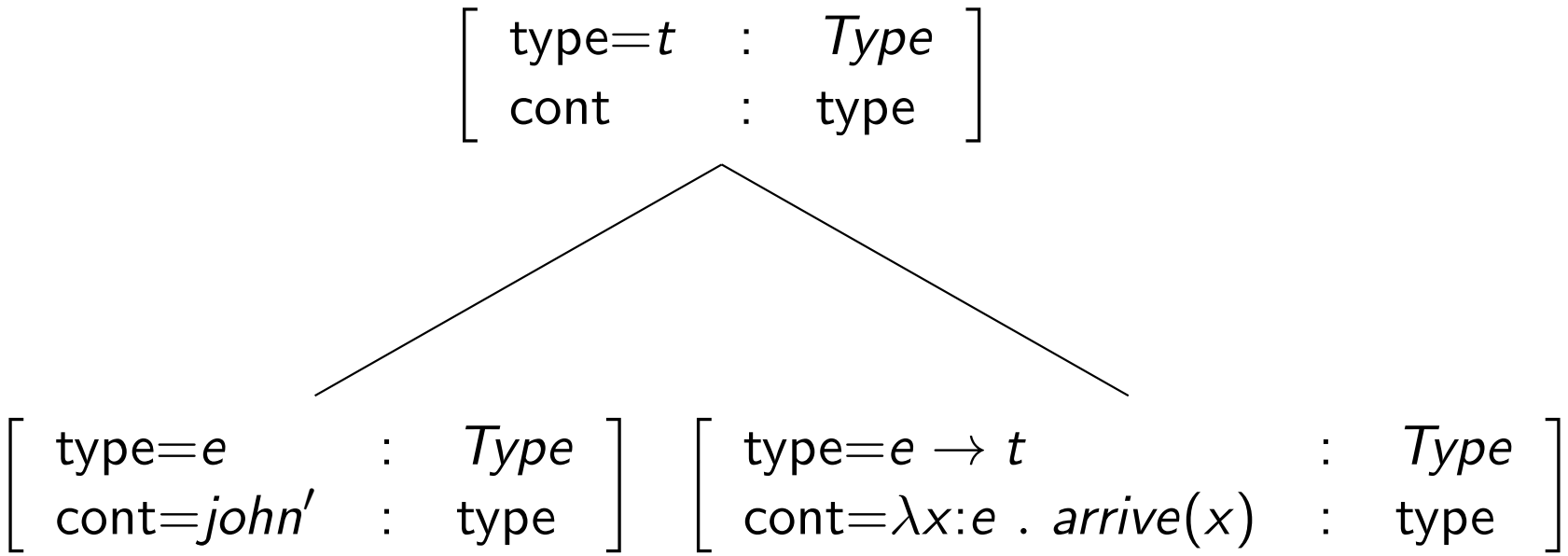
- Note that this represents a tree *type*, not a tree
- cf underspecified trees

or more diagrammatically . . .



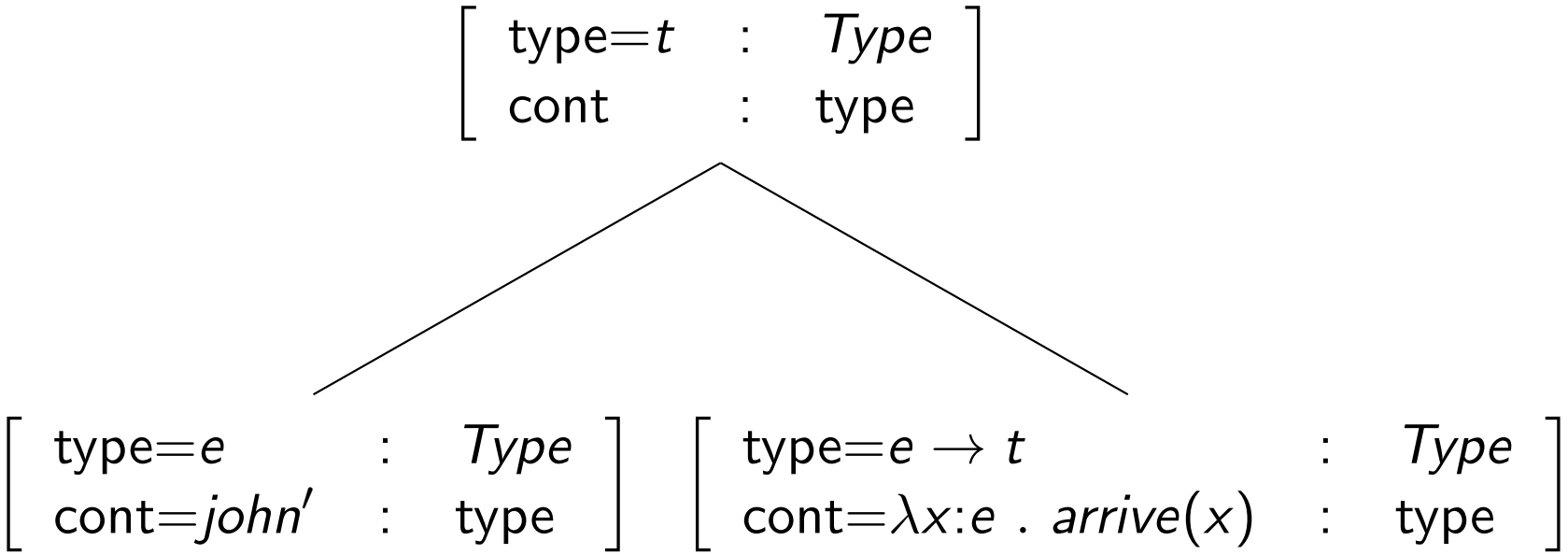
- Note that this represents a tree *type*, not a tree
- cf underspecified trees
- a record type is *fully specified* iff all its fields are manifest

or more diagrammatically . . .



- Note that this represents a tree *type*, not a tree
- cf underspecified trees
- a record type is *fully specified* iff all its fields are manifest
- it is *underspecified* otherwise

or more diagrammatically . . .



- Note that this represents a tree *type*, not a tree
- cf underspecified trees
- a record type is *fully specified* iff all its fields are manifest
- it is *underspecified* otherwise
- this type is underspecified with respect to the path 'cont'

More underspecification

$$\begin{bmatrix} \text{type}{=}t & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{bmatrix}$$

$$\begin{bmatrix} \text{type}{=}e & : & \mathit{Type} \\ \text{cont}{=}\mathit{john}' & : & \text{type} \end{bmatrix}$$

cf. D-theory (Marcus *et al.*, 1983), functional uncertainty in LFG

Components in records

- An object, a, is a *component* of a record, r, in symbols, $a \varepsilon r$, just in case there is some path, π, in r such that $r.\pi = a$
- A string $a_0^\frown \ldots ^\frown a_n$ can be viewed as a record
$$\begin{bmatrix} t_0 & = & a_0 \\ \vdots & & \\ t_n & = & a_n \end{bmatrix}$$
- Thus there are paths into strings occuring as a component in a record

Types of objects containing a certain component

If a is an object of some type and T is a type, then $T_{a\varepsilon}$ is a type.
$b : T_{a\varepsilon}$ iff $b : T$ and $a \varepsilon b$.

Type for the tree with unattached node

$$\left[\begin{array}{lcl} \text{type}{=}t & : & \mathit{Type} \\ \text{cont} & : & \text{type} \\ \text{daughters} & : & \mathit{Tree}^{*} \left[\begin{array}{lcl} \text{type}{=}e & : & \mathit{Type} \\ \text{cont}{=}\mathit{john}' & : & \text{type} \end{array}\right]_{\varepsilon} \end{array}\right]$$

Lexical entry for *John*

john:

IF	$?Ty(e)$
THEN	put($Ty(e)$)
	put($Fo(john')$)
ELSE	abort

Lexical entries as update rules

If $T_i = \begin{bmatrix} \text{type=e} & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{bmatrix}$,

then set T_{i+1} to be $\begin{bmatrix} \text{type=e} & : & \mathit{Type} \\ \text{cont=}\mathit{john}' & : & \text{type} \end{bmatrix}$

Lexical entries as type refinements

$$\begin{bmatrix} \text{type}{=}\text{e} & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{bmatrix} \Rightarrow \begin{bmatrix} \text{type}{=}\text{e} & : & \mathit{Type} \\ \text{cont}{=}\mathit{john}' & : & \text{type} \end{bmatrix}$$

Type rewrite rule which is a type refinement because $RHS \sqsubseteq LHS$

More general version of type refinement rule

$$\text{If } T \sqsubseteq \begin{bmatrix} \text{type=e} & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{bmatrix}:$$

$$T \Rightarrow T \wedge_{\cdot} \begin{bmatrix} \text{type=e} & : & \mathit{Type} \\ \text{cont=}\mathit{john}' & : & \text{type} \end{bmatrix}$$

Outline

Alternative using TTR-style content

$$\left[\begin{array}{lcl} \text{type}=\left[\text{x}:\mathit{Ind}\right] & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{array}\right] \Rightarrow \left[\begin{array}{lcl} \text{type}=\left[\text{x}:\mathit{Ind}\right] & : & \mathit{Type} \\ \text{cont}=\left[\text{x}=\text{john}\right] & : & \text{type} \end{array}\right]$$

Type for *John arrived* with TTR content

$$\begin{bmatrix} \text{type}=\mathit{RecType}:\mathit{Type} \\ \text{cont}=\begin{bmatrix} \text{x}=\text{john}:\mathit{Ind} \\ \text{p:arrive(x)} \end{bmatrix}:\text{type} \end{bmatrix}$$

$$\begin{bmatrix} \text{type}=\left[\text{x}:\mathit{Ind}\right]:\mathit{Type} \\ \text{cont}=\left[\text{x}=\text{john}\right]:\text{type} \end{bmatrix} \qquad \begin{bmatrix} \text{type}=\left(\left[\text{x}:\mathit{Ind}\right]\rightarrow\mathit{RecType}\right):\mathit{Type} \\ \text{cont}=\lambda r:\left[\text{x}:\mathit{Ind}\right] \ . \ \begin{bmatrix} \text{x}=r.\text{x}:\mathit{Ind} \\ \text{p:arrive(x)} \end{bmatrix}:\text{type} \end{bmatrix}$$

Outline

Split utterances

A. You burned . . .
B. . . . myself

Speech events

$$SEvent = \begin{bmatrix} \mathsf{sp} & : & \mathit{Ind} \\ \mathsf{au} & : & \mathit{Ind} \\ \mathsf{e} & : & \mathit{Phon} \\ \mathsf{c}_{\mathsf{sp}} & : & \mathsf{speaker(e,sp)} \\ \mathsf{c}_{\mathsf{au}} & : & \mathsf{audience(e,au)} \end{bmatrix}$$

Indexical pronouns

I, me $\left[\begin{array}{lll} \text{s-event} & : & \mathit{SEvent} \\ \text{type}=\left[\text{x}:\mathit{Ind}\right] & : & \mathit{Type} \\ \text{cont} & : & \text{type} \end{array}\right] \Rightarrow$

$\left[\begin{array}{lll} \text{s-event} & : & \mathit{SEvent} \\ \text{type}=\left[\text{x}:\mathit{Ind}\right] & : & \mathit{Type} \\ \text{cont}=\left[\text{x}=\text{s-event.sp}\right] & : & \text{type} \end{array}\right]$

- different s-event for each incremental item, therefore potentially different speakers for different incremental items
- sign-based approach

Still to be done

- a neat solution for the focus, $\Diamond$ – possibly just an additional field
- linked nodes
- ...
- working out (and implementing) a detailed grammar based on these ideas

Why?

- because we can (or perhaps, ultimately, *cannot*)
- TTR would like to bill itself as a foundation for various linguistic theories
- . . . allowing integration of analyses in different theories or pointing up relations between them
- Perhaps there are elements in TTR which DS would find useful in addition to the use of TTR for semantic representation
- In general, the more theoretical light from different perspectives we can shed on incremental processing of language, the better

Support from VR projects:

- 2009-1569, Semantic analysis of interaction and coordination in dialogue (SAICD)
- 2016-01162, Incremental Reasoning in Dialogue (IncReD)
- 2014-39, Centre for Linguistic Theory and Studies in Probability (CLASP)

Bibliography I

Cooper, Robin (2012) Type Theory and Semantics in Flux, in R. Kempson, N. Asher and T. Fernando (eds.), *Handbook of the Philosophy of Science*, Vol. 14: Philosophy of Linguistics, pp. 271–323, Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Cooper, Robin (in prep) Type theory and language: from perception to linguistic communication. Draft of book chapters available from `https://sites.google.com/site/typetheorywithrecords/drafts`.

Cooper, Robin and Jonathan Ginzburg (2015) Type Theory with Records for Natural Language Semantics, in S. Lappin and C. Fox (eds.), *The Handbook of Contemporary Semantic Theory*, second edition, pp. 375–407, Wiley-Blackwell.

Bibliography II

Eshghi, Arash (2015) DS-TTR: An incremental, semantic, contextual parser for dialogue, pp. 172–173.

Kempson, Ruth, Wilfried Meyer-Viol and Dov Gabbay (2001) *Dynamic syntax: the flow of language understanding*, Blackwell.

Marcus, Mitchell P., Donald Hindle and Margaret M. Fleck (1983) D-theory: Talking About Talking About Trees, in *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, pp. 129–136, Association for Computational Linguistics, Stroudsburg, PA, USA.