A theory of events and situations

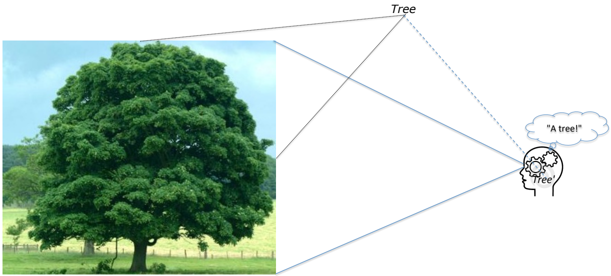
Robin Cooper
University of Gothenburg
Jonathan Ginzburg
Université Paris-Diderot, Sorbonne Paris-Cité

Type theory with records for natural language semantics,
NASSLLI 2012
Lecture 1, part 2

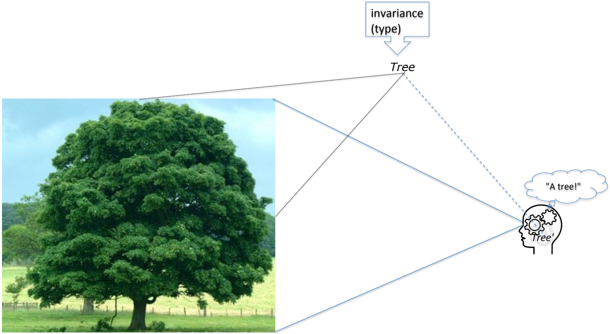
Outline

Outline

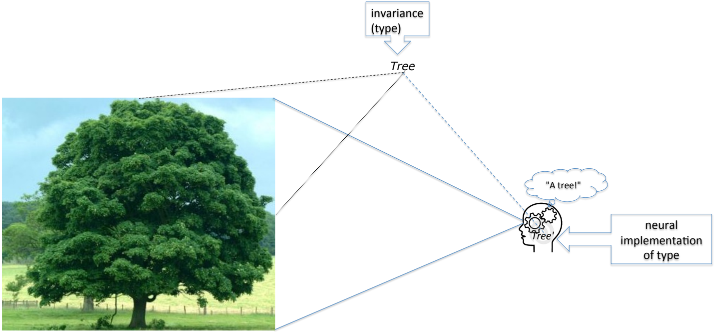
Seeing a tree (a simulation view)



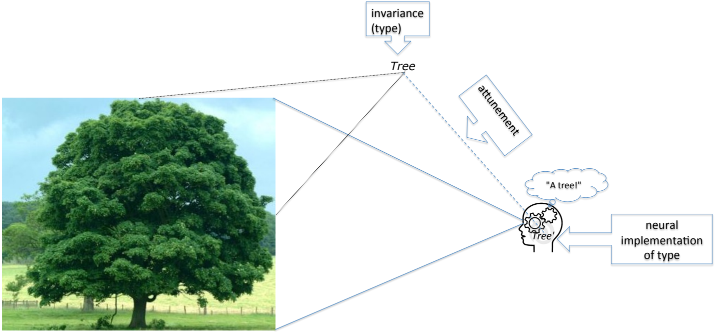
Seeing a tree (a simulation view)



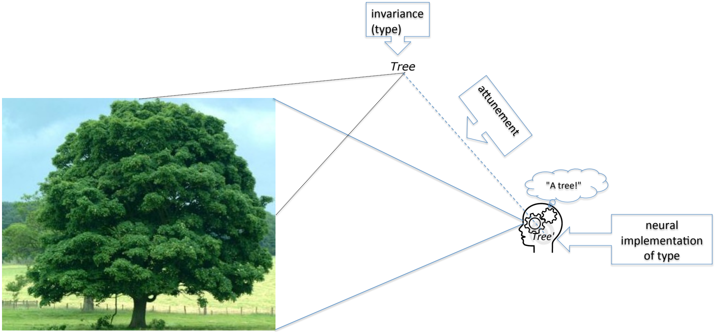
Seeing a tree (a simulation view)



Seeing a tree (a simulation view)



Seeing a tree (a simulation view)



Gibson (1986); Barwise and Perry (1983)

Judgement

- (An agent judges that) object a is of type T.
- $a : T$

Outline

TTR: Type theory with records

- The most recent published reference for the details is Cooper (2012)
- Also Cooper (2005a) for an earlier detailed treatment
- Cooper (2005b) for relation to various semantic theories
- `https://sites.google.com/site/typetheorywithrecords/drafts/ch1-draft111114.pdf` for some current work in progress we will discuss here

Basic types

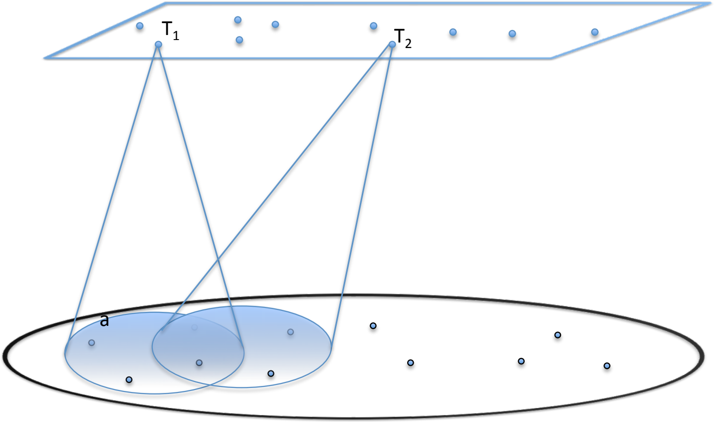
A *system of basic types* is a pair:

$$\mathbf{TYPE}_B = \langle \mathbf{Type}, A \rangle$$

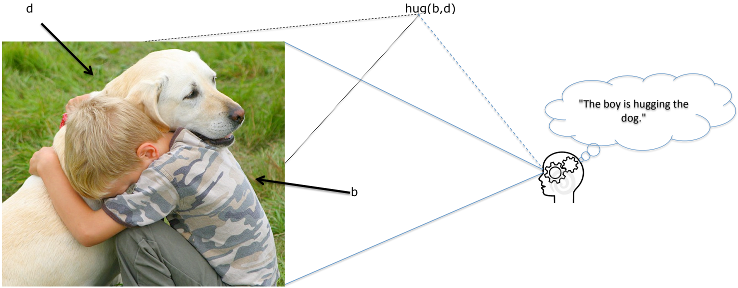
where:

1. **Type** is a non-empty set
2. A is a function whose domain is **Type**
3. for any $T \in \mathbf{Type}$, $A(T)$ is a set disjoint from **Type**
4. for any $T \in \mathbf{Type}$, $a :_{\mathbf{TYPE_B}} T$ iff $a \in A(T)$

$a : T_1$



Seeing a hugging event



Predicate signatures

A *predicate signature* is a triple

$$\langle \textbf{Pred}, \textbf{ArgIndices}, \mathit{Arity} \rangle$$

where:

1. **Pred** is a set (of predicates)
2. **ArgIndices** is a set (of indices for predicate arguments, normally types)
3. *Arity* is a function with domain **Pred** and range included in the set of finite sequences of members of **ArgIndices**.

Polymorphic predicate signatures

A *polymorphic predicate signature* is a pair

$$\langle \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle$$

where:

1. **Pred** is a set (of predicates)
2. **ArgIndices** is a set (of indices for predicate arguments, normally types)
3. *Arity* is a function with domain **Pred** and range included in the powerset of the set of finite sequences of members of **ArgIndices**.

Complex types

A *system of complex types* is a quadruple:

$$\textbf{TYPE}_C = \langle \textbf{Type}, \textbf{BType}, \langle \textbf{PType}, \textbf{Pred}, \textbf{ArgIndices}, Arity \rangle, \langle A, F \rangle \rangle$$

where:

1. $\langle \textbf{BType}, A \rangle$ is a system of basic types
2. $\textbf{BType} \subseteq \textbf{Type}$
3. for any $T \in \textbf{Type}$, if $a :_{\langle \textbf{BType}, A \rangle} T$ then $a :_{\textbf{TYPE}_\textbf{C}} T$
4. $\langle \textbf{Pred}, \textbf{ArgIndices}, Arity \rangle$ is a (polymorphic) predicate signature
5. If $P \in \textbf{Pred}$, $T_1 \in \textbf{Type}, \ldots, T_n \in \textbf{Type}$, $Arity(P) = \langle T_1, \ldots, T_n \rangle$ ($\langle T_1, \ldots, T_n \rangle \in Arity(P)$) and $a_1 :_{\textbf{TYPE}_\textbf{C}} T_1, \ldots, a_n :_{\textbf{TYPE}_\textbf{C}} T_n$ then $P(a_1, \ldots a_n) \in \textbf{PType}$
6. $\textbf{PType} \subseteq \textbf{Type}$
7. for any $T \in \textbf{PType}$, $F(T)$ is a set disjoint from $\textbf{Type}$
8. for any $T \in \textbf{PType}$, $a :_{\textbf{TYPE}_\textbf{C}} T$ iff $a \in F(T)$

A boy hugs a dog

$$\left[\begin{array}{lll} \text{e-time} & : & \mathit{Time} \\ \text{x} & : & \mathit{Ind} \\ \text{c}_{\text{boy}} & : & \text{boy(x,e-time)} \\ \text{y} & : & \mathit{Ind} \\ \text{c}_{\text{dog}} & : & \text{dog(y,e-time)} \\ \text{c}_{\text{hug}} & : & \text{hug(x,y,e-time)} \end{array}\right]$$

The official notation

$$\left[\begin{array}{lll} \text{e-time} & : & \mathit{Time} \\ \text{x} & : & \mathit{Ind} \\ \text{c}_{\text{boy}} & : & \langle \lambda v : \mathit{Ind}(\lambda t : \mathit{Time}(\text{boy}(v,t))), \langle \text{x},\text{e-time}\rangle\rangle, \\ \text{y} & : & \mathit{Ind} \\ \text{c}_{\text{dog}} & : & \langle \lambda v : \mathit{Ind}(\lambda t : \mathit{Time}(\text{dog}(v,t))), \langle \text{y},\text{e-time}\rangle\rangle \\ \text{c}_{\text{hug}} & : & \langle \lambda v_1 : \mathit{Ind}(\lambda v_2 : \mathit{Ind}(\lambda t : \mathit{Time}(\text{hug}(v_1,v_2,t)))), \\ & & \quad \langle \text{x},\text{y},\text{e-time}\rangle\rangle \end{array}\right]$$

Function types

$\mathbf{TYPE}_C$ *has function types* if

1. for any $T_1, T_2 \in \mathbf{Type}$, $(T_1 \rightarrow T_2) \in \mathbf{Type}$
2. for any $T_1, T_2 \in \mathbf{Type}$, $f :_{\mathbf{TYPE_C}} (T_1 \rightarrow T_2)$ iff f is a function whose domain is $\{a \mid a :_{\mathbf{TYPE_C}} T_1\}$ and whose range is included in $\{a \mid a :_{\mathbf{TYPE_C}} T_2\}$

Records

- A *record* is a finite set of ordered pairs (called *fields*) which is the graph of a function. If r is a record and $\langle \ell, v \rangle$ is a field in r we call ℓ a *label* and v a *value* in r and we use $r.\ell$ to denote v. $r.\ell$ is called a *path* in r. If π is a path in r whose denotation is itself a record with a label ℓ, then $\pi.\ell$ is also a path in r.

Records

- A *record* is a finite set of ordered pairs (called *fields*) which is the graph of a function. If r is a record and $\langle \ell, v \rangle$ is a field in r we call ℓ a *label* and v a *value* in r and we use $r.\ell$ to denote v. $r.\ell$ is called a *path* in r. If π is a path in r whose denotation is itself a record with a label ℓ, then $\pi.\ell$ is also a path in r.
- A record r is *well-typed* with respect to a system of types **TYPE** with set of types **Type** and a set of labels L iff for each field $\langle \ell, a \rangle \in r$, $\ell \in L$ and $a :_{\mathbf{TYPE}} T$ for some $T \in \mathbf{Type}$.

Record types I

A system of complex types $\textbf{TYPE}_C = \langle\textbf{Type}, \textbf{BType}, \langle\textbf{PType}, \textbf{Pred}, \textbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F\rangle\rangle$ *has record types based on* $\langle L, \textbf{RType}\rangle$, where L is a countably infinite set (of labels) and $\textbf{RType} \subseteq \textbf{Type}$, defined by:

1. $\mathit{Rec} \in \textbf{RType}$
2. $r :_{\textbf{Type}_C} \mathit{Rec}$ iff r is a well-typed record with respect to $\textbf{TYPE}_C$ and L.
3. if $\ell \in L$ and $T \in \textbf{Type}$, then $\{\langle\ell, T\rangle\} \in \textbf{RType}$.
4. $r :_{\textbf{Type}_C} \{\langle\ell, T\rangle\}$ iff $r :_{\textbf{Type}_C} \mathit{Rec}$, $\langle\ell, a\rangle \in r$ and $a :_{\textbf{Type}_C} T$.
5. if $R \in \textbf{RType}$, $\ell \in L$, ℓ does not occur as a label in R (i.e. there is no field $\langle\ell', T'\rangle$ in R such that $\ell' = \ell$), then $R \cup \{\langle\ell, T\rangle\} \in \textbf{RType}$.

Record types II

6. $r :_{\mathbf{Type}_C} R \cup \{\langle \ell, T\rangle\}$ iff $r :_{\mathbf{Type}_C} R$, $\langle \ell, a\rangle \in r$ and $a :_{\mathbf{Type}_C} T$.
7. if R is a member of **RType**, $\ell \in L$ not occurring as a label in R, $T_1, \ldots, T_m \in \mathbf{Type}$, $R.\pi_1, \ldots, R.\pi_m$ are paths in R and $\mathcal{F}$ is a function of type $((a_1 : T_1) \to \ldots \to ((a_m : T_m) \to Type)\ldots)$, then $R \cup \{\langle \ell, \langle \mathcal{F}, \langle \pi_1, \ldots, \pi_m\rangle\rangle\rangle\} \in \mathbf{RType}$.
8. $r :_{\mathbf{TYPE}_C} R \cup \{\langle \ell, \langle \mathcal{F}, \langle \pi_1, \ldots, \pi_m\rangle\rangle\rangle\}$ iff $r :_{\mathbf{TYPE}_C} R$, $\langle \ell, a\rangle$ is a field in r, $r.\pi_1 :_{\mathbf{TYPE}_C} T_1, \ldots, r.\pi_m :_{\mathbf{TYPE}_C} T_m$ and $a :_{\mathbf{TYPE}_C} \mathcal{F}(r.\pi_1, \ldots, r.\pi_m)$.

The type *Type*

- The previous slide introduced the type *Type*
- T : *Type* iff $T \in$ **Type**

The type *Type*

- The previous slide introduced the type *Type*
- T :*Type* iff T $\in$**Type**
- *Type*$\in$**Type**

The type *Type*

- The previous slide introduced the type *Type*
- T : *Type* iff $T \in$ **Type**
- *Type* $\in$ **Type**
- *Type* : *Type*

The type *Type*

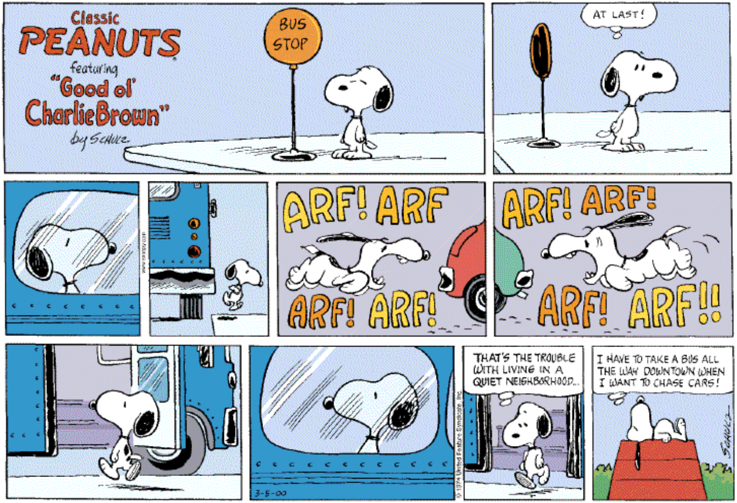
- The previous slide introduced the type *Type*
- $T : Type$ iff $T \in \mathbf{Type}$
- $Type \in \mathbf{Type}$
- $Type : Type$
- But suppose we allow some type to have the extension $\{T \in \mathbf{Type} \mid T \not: T\}$ ...

The type *Type*

- The previous slide introduced the type *Type*
- T : *Type* iff $T \in \mathbf{Type}$
- *Type* $\in \mathbf{Type}$
- *Type*: *Type*
- But suppose we allow some type to have the extension $\{T \in \mathbf{Type} \mid T \not: T\} \ldots$
- This leads us to stratification.

Outline

Fernando's string theory



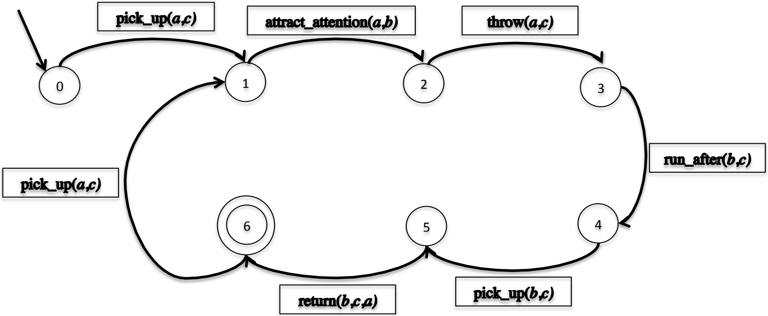
Some references to Fernando's work

Fernando (2004, 2006, 2008, 2009)

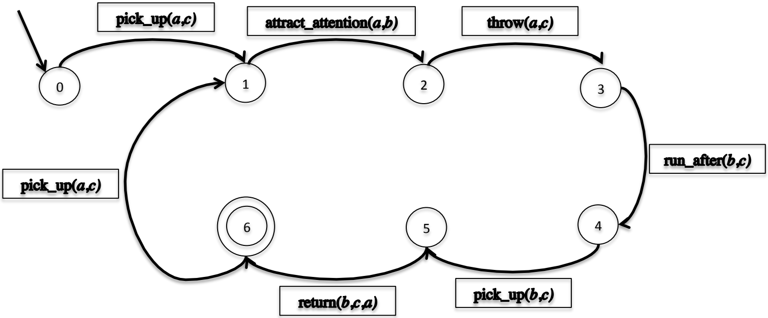
Regular types

1. if T_1, $T_2 \in$ **Type**, then $T_1 ⁀ T_2 \in$ **Type**
   $a : T_1 ⁀ T_2$ iff $a = x ⁀ y$, $x : T_1$ and $y : T_2$
2. if $T \in$ **Type** then $T^+ \in$ **Type**.
   $a : T^+$ iff $a = x_1 ⁀ \ldots ⁀ x_n$, $n > 0$ and for i, $1 \leq i \leq n$, $x_i : T$
   ...

A game of fetch



A game of fetch



$$(\text{pick\_up}(a,c)^\frown\text{attract\_attention}(a,b)^\frown\text{throw}(a,c)^\frown\text{run\_after}(b,c)^\frown \text{pick\_up}(b,c)^\frown\text{return}(b,c,a))^+$$

Getting serious about time: intervals

$$\left[\begin{array}{lcl} \text{start} & : & \mathit{Time} \\ \text{end} & : & \mathit{Time} \\ \text{c}_{<} & : & \text{start}<\text{end} \end{array}\right]$$

Getting serious about time: the fetch-game type

$$
(\begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{c}_{\text{pick\_up}}:\text{pick\_up}(a,c,\text{e-time})\end{bmatrix}{}^\frown{}_< \begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{c}_{\text{att\_att}}:\text{attract\_attent}(a,b,\text{e-time})\end{bmatrix}{}^\frown{}_<
$$

$$
\begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{c}_{\text{throw}}:\text{throw}(a,c,\text{e-time})\end{bmatrix}{}^\frown{}_< \begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{c}_{\text{run\_after}}:\text{run\_after}(b,c,\text{e-time})\end{bmatrix}{}^\frown{}_<
$$

$$
\begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{c}_{\text{pick\_up}}:\text{pick\_up}(b,c,\text{e-time})\end{bmatrix}{}^\frown{}_< \begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{c}_{\text{return}}:\text{return}(b,c,a,\text{e-time})\end{bmatrix})^{+_<}
$$

Temporal concatenation

1. If T_1 and T_2 are subtypes of $\left[\text{e-time}{:}\,\mathit{TimeInt}\right]^+$ then $T_1{}^{\frown<}T_2$ is a type.
2. $a : T_1{}^{\frown<}T_2$ iff $a = a_1{}^{\frown}a_2$, $a_1 : T_1$, $a_2 : T_2$ and last(a_1).e-time.end<first(a_2).e-time.start

Temporal Kleene-$+$

1. If T is a subtype of $\left[\text{e-time}: \mathit{TimeInt}\right]$ then $T^{+<}$ is a type
2. $a : T^{+<}$ iff $a = x_1{}^\frown \ldots {}^\frown x_n$, $n > 0$ and for i, j, $1 \leq i < j \leq n$, $x_i : T$, $x_j : T$ and $x_i{}^\frown x_j : T^{\frown <} T$

Outline

Partiality of event perception

- Do not need to observe all the frames in an event
- Suffices to observe enough to uniquely identify event types agent has in its resources

Inferring an event type from a partial observation

$$\lambda r{:}\left[\begin{array}{l}\mathrm{x}{:}\,\mathit{Ind}\\ \mathrm{c}_{\mathrm{human}}{:}\mathrm{human}(\mathrm{x})\\ \mathrm{y}{:}\,\mathit{Ind}\\ \mathrm{c}_{\mathrm{dog}}{:}\mathrm{dog}(\mathrm{y})\\ \mathrm{z}{:}\,\mathit{Ind}\\ \mathrm{c}_{\mathrm{stick}}{:}\mathrm{stick}(\mathrm{z})\\ \text{e-time}{:}\,\mathit{TimeInt}\\ \mathrm{e}{:}\left[\begin{array}{l}\text{e-time}{:}\,\mathit{TimeInt}\\ \mathrm{c}_{\leq}{:}\Uparrow\text{e-time.start}\leq\text{e-time.start}\\ \mathrm{c}_{\mathrm{pick\_up}}{:}\mathrm{pick\_up}(\Uparrow\mathrm{x},\Uparrow\mathrm{z},\text{e-time})\end{array}\right]^\frown{}^{<}\left[\begin{array}{l}\text{e-time}{:}\,\mathit{TimeInt}\\ \mathrm{c}_{\leq}{:}\text{e-time.start}\leq\Uparrow\text{e-time.start}\\ \mathrm{c}_{\mathrm{att\_att}}{:}\mathrm{att\_att}(\Uparrow\mathrm{x},\Uparrow\mathrm{y},\text{e-time})\end{array}\right]\end{array}\right]$$

$$\left(\left[\begin{array}{l}\text{e-time}{:}\,\mathit{TimeInt}\\ \mathrm{c}_{\leq}{:}r.\text{e-time.start}\leq\text{e-time.start}\\ \mathrm{e}{:}\mathrm{play\_fetch}(r.\mathrm{x},r.\mathrm{y},r.\mathrm{z},\text{e-time})\end{array}\right]\right)$$

Three views of this inference

Three views of this inference

- function from objects (events) to a *type*

Three views of this inference

- function from objects (events) to a *type* – $\lambda a : T_1(T_2[a])$

Three views of this inference

- function from objects (events) to a *type* – $\lambda a : T_1(T_2[a])$
- a *dependent type*

Three views of this inference

- function from objects (events) to a *type* – $\lambda a : T_1(T_2[a])$
- a *dependent type*
- perceiving something and inferring the type of something not (yet) perceived from that perception

Three views of this inference

- function from objects (events) to a *type* – $\lambda a : T_1(T_2[a])$
- a *dependent type*
- perceiving something and inferring the type of something not (yet) perceived from that perception
- we will see a number of other uses of dependent types, for example as the interpretation of verb phrases

Outline

Summary

- Type theory as a formal theory related to perception
- A basic introduction to TTR
- Events as strings
- Partial observation of events and inference
- Frames in lexical semantics

Bibliography I

Barwise, Jon and John Perry (1983) *Situations and Attitudes*, Bradford Books, MIT Press, Cambridge, Mass.

Cooper, Robin (2005a) Austinian truth, attitudes and type theory, *Research on Language and Computation*, Vol. 3, pp. 333–362.

Cooper, Robin (2005b) Records and Record Types in Semantic Theory, *Journal of Logic and Computation*, Vol. 15, No. 2, pp. 99–112.

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, Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Bibliography II

Cooper, Robin and Staffan Larsson (2009) Compositional and ontological semantics in learning from corrective feedback and explicit definition, in J. Edlund, J. Gustafson, A. Hjalmarsson and G. Skantze (eds.), *Proceedings of DiaHolmia: 2009 Workshop on the Semantics and Pragmatics of Dialogue*, pp. 59–66.

Fernando, Tim (2004) A finite-state approach to events in natural language semantics, *Journal of Logic and Computation*, Vol. 14, No. 1, pp. 79–92.

Fernando, Tim (2006) Situations as Strings, *Electronic Notes in Theoretical Computer Science*, Vol. 165, pp. 23–36.

Bibliography III

Fernando, Tim (2008) Finite-state descriptions for temporal semantics, in H. Bunt and R. Muskens (eds.), *Computing Meaning, Volume 3* (*Studies in Linguistics and Philosophy* 83), pp. 347–368, Springer.

Fernando, Tim (2009) Situations in LTL as strings, *Information and Computation*, Vol. 207, No. 10, pp. 980–999.

Gibson, James J. (1986) *The Ecological Approach to Visual Perception*, Lawrence Erlbaum Associates.

Larsson, Staffan (2007) A general framework for semantic plasticity and negotiation, in H. Bunt and E. C. G. Thijsse (eds.), *Proceedings of the 7th International Workshop on Computational Semantics (IWCS-7)*, pp. 101–117.

Bibliography IV

Larsson, Staffan and Robin Cooper (2009) Towards a formal view of corrective feedback, in A. Alishahi, T. Poibeau and A. Villavicencio (eds.), *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pp. 1–9.