Grammar in a Landscape of Affordances

Eleni Gregoromichelaki
University of Gothenburg
https://elenigregor.github.io/
http://dynamicsyntax.org/

02 December 2020

Outline

what is the nature of grammar: the view from DS-TTR

- **DS-TTR**: blend of DS and TTR [Purver et al. (2010); Eshghi et al. (2013); Hough (2015); Purver et al. (2011); Gregoromichelaki (2018); Gregoromichelaki et al. (2019b)]

- grammatical/metaphysical ontology of **processes**
  - rather than *representations*

what is the nature of grammar: the view from DS-TTR

- **DS-TTR**: blend of DS and TTR [Purver et al. (2010); Eshghi et al. (2013); Hough (2015); Purver et al. (2011); Gregoromichelaki (2018); Gregoromichelaki et al. (2019b)]
- grammatical/metaphysical ontology of **processes**
  - rather than *representations*
- **incrementality**, **underspecification**, and **predictivity** as properties of the grammar
- domain-general processes for **multimodal interaction** [e.g.,Yu et al. (2015, 2017); Yu et al., 2016)]
- learnable from minimal data via Reinforcement Learning [e.g.,Kalatzis et al. (2016); Eshghi et al. (2017b)]

DS-TTR: syntactic mechanisms

- **Dynamic Syntax** (DS) [Kempson et al. (2001); Gregoromichelaki et al. (2019b)]

DS-TTR: syntactic mechanisms

- **Dynamic Syntax** (DS) [Kempson et al. (2001); Gregoromichelaki et al. (2019b)]
  - (inter)actions are all you need to talk about "syntax"

DS-TTR: syntactic mechanisms

- **Dynamic Syntax** (DS) [Kempson et al. (2001); Gregoromichelaki et al. (2019b)]
  - (inter)actions are all you need to talk about "syntax"

- **syntactic structure over words** (tree-structures) is at best epiphenomenal
- no separate **syntactic level of representation**:
  - no syntactic categories for strings of words;
  - no phrase-structure rules;
  - sequences of words are not sequences of symbols but sequences of **affordance triggers**

DS-TTR: syntactic mechanisms

- **Dynamic Syntax** (DS) [Kempson et al. (2001); Gregoromichelaki et al. (2019b)]
  - (inter)actions are all you need to talk about "syntax"

- **syntactic structure over words** (tree-structures) is at best epiphenomenal
- no separate **syntactic level of representation**:
  - no syntactic categories for strings of words;
  - no phrase-structure rules;
  - sequences of words are not sequences of symbols but sequences of **affordance triggers**

- **grammatical affordances** are dynamic regularities extending over multiple time-steps

extending DS-TTR: conceptualisation as interaction

- **TTR** [see e.g. Cooper (2012); Cooper and Ginzburg (2015)]: linking grammar to perception and high-order conceptualisation [cf.Bengio (2019)]

extending DS-TTR: conceptualisation as interaction

- **TTR** [see e.g. Cooper (2012); Cooper and Ginzburg (2015)]: linking grammar to perception and high-order conceptualisation [cf.Bengio (2019)]
- types (concepts) as (sets of) **affordances**

extending DS-TTR: conceptualisation as interaction

- **TTR** [see e.g. Cooper (2012); Cooper and Ginzburg (2015)]: linking grammar to perception and high-order conceptualisation [cf.Bengio (2019)]
- types (concepts) as (sets of) **affordances**
  - affordances are the possibilities for interaction in the sociomaterial environment to which agents are "attuned"
  - interaction with entities
    - agents can interact with aspects of entities without necessarily recognising the entity
    - learning affordances (sensorimotor contingencies) replaces the effort of building ontologies and writing rules
- types (concepts) as **time-extended processes** (incrementality, temporality)

extending DS-TTR: conceptualisation as interaction

- **TTR** [see e.g. Cooper (2012); Cooper and Ginzburg (2015)]: linking grammar to perception and high-order conceptualisation [cf.Bengio (2019)]
- types (concepts) as (sets of) **affordances**
  - affordances are the possibilities for interaction in the sociomaterial environment to which agents are "attuned"
  - interaction with entities
    - agents can interact with aspects of entities without necessarily recognising the entity
    - learning affordances (sensorimotor contingencies) replaces the effort of building ontologies and writing rules
- types (concepts) as **time-extended processes** (incrementality, temporality)
- types induce **predictions** for
  - what is to be encountered as perceptual stimulation next or
  - predictions regarding how the agent can interact with some entity/feature of the environment

extending DS-TTR: conceptualisation as interaction

- **TTR** [see e.g. Cooper (2012); Cooper and Ginzburg (2015)]: linking grammar to perception and high-order conceptualisation [cf.Bengio (2019)]
- types (concepts) as (sets of) **affordances**
  - affordances are the possibilities for interaction in the sociomaterial environment to which agents are "attuned"
  - interaction with entities
    - agents can interact with aspects of entities without necessarily recognising the entity
    - learning affordances (sensorimotor contingencies) replaces the effort of building ontologies and writing rules
- types (concepts) as **time-extended processes** (incrementality, temporality)
- types induce **predictions** for
  - what is to be encountered as perceptual stimulation next or
  - predictions regarding how the agent can interact with some entity/feature of the environment
- at each time-step, affordances need to be **selected** from a landscape of possible affordances

common ground: joint affordances in dialogue

- unit of analysis: a group-based **distributed cognitive system**

common ground: joint affordances in dialogue

- unit of analysis: a group-based **distributed cognitive system**
- **landscape of joint affordances** defines what is available [*Skilled Intentionality* framework, Rietveld et al. (2018)]

common ground: joint affordances in dialogue

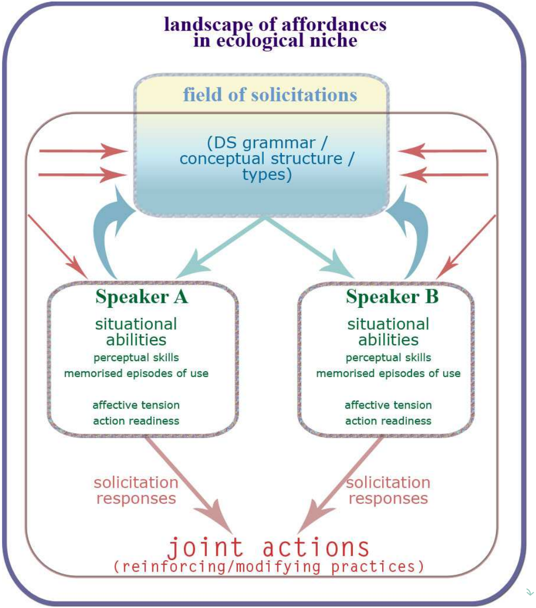
- unit of analysis: a group-based **distributed cognitive system**
- **landscape of joint affordances** defines what is available [*Skilled Intentionality* framework, Rietveld et al. (2018)]
- in **joint action**, participants' abilities, the sociomaterial environment, and the previous history of interactions codetermine a particular subset of the **field of affordances** (solicitations)

common ground: joint affordances in dialogue

- unit of analysis: a group-based **distributed cognitive system**
- **landscape of joint affordances** defines what is available [*Skilled Intentionality* framework, Rietveld et al. (2018)]
- in **joint action**, participants' abilities, the sociomaterial environment, and the previous history of interactions codetermine a particular subset of the **field of affordances** (solicitations)
- so-called *common ground* Stalnaker (1999); Clark (1996)] is a property of the relation of individual participants' affordances

common ground: joint affordances in dialogue

- unit of analysis: a group-based **distributed cognitive system**
- **landscape of joint affordances** defines what is available [*Skilled Intentionality* framework, Rietveld et al. (2018)]
- in **joint action**, participants' abilities, the sociomaterial environment, and the previous history of interactions codetermine a particular subset of the **field of affordances** (solicitations)
- so-called *common ground* Stalnaker (1999); Clark (1996)] is a property of the relation of individual participants' affordances
- incrementality and temporality means that such fields are **redefined** and **transformed** with each utterance (verbal or otherwise)



interaction first

- **actions** (procedural ‘know-how’) the basis for syntax/semantics/pragmatics

interaction first

- **actions** (procedural 'know-how') the basis for syntax/semantics/pragmatics
- **interactions**: both comprehension and production modelled together in the same space

interaction first

- **actions** (procedural ‘know-how’) the basis for syntax/semantics/pragmatics
- **interactions**: both comprehension and production modelled together in the same space
- syntactic or meaning procedures formulated as (probabilistic) **transitions** from states to states

- a specialised **Propositional Dynamic Logic** (**PDL**) with states as ⟨NL string, context⟩ and transition operators modelling basic actions and macros (packages) of such actions

  [Kempson et al. (2001)]

the dynamics of interaction

- a specialised **Propositional Dynamic Logic** (**PDL**) with states as ⟨NL string, context⟩ and transition operators modelling basic actions and macros (packages) of such actions
  [Kempson et al. (2001)]
- Dynamic Logics
  - have the means to model any type of action and event (physical, instrumental, epistemic, etc.) [e.g. Segerberg (1992)]: hence multimodal grammar definitions are seamless
  - model an internal perspective in a computation, an observer's view located at a state (node) considering the possibilities
  - potential to bring Dynamic Quantum Logic [Baltag and Smets (2012)] to interface with DS Vector Space Semantics
    [Purver et al. (forthcoming); Sadrzadeh et al. (2018); Gregoromichelaki et al. (2019a)]
- however, ideally, the π-calculus might be more suitable

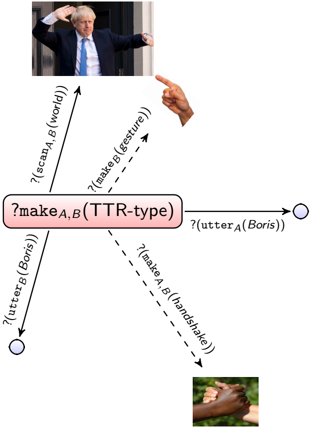
fractal structure of state transition system

(red path indicates the selected course of action)

?$\texttt{make}_{A,B}$(TTR-type)

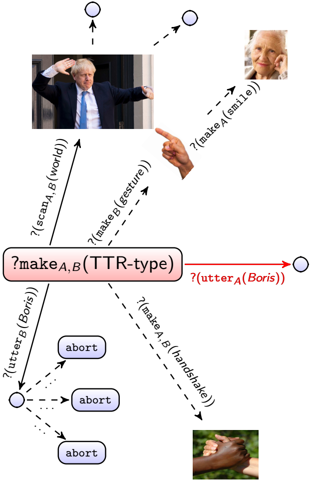
fractal structure of state transition system

(red path indicates the selected course of action)



fractal structure of state transition system

(red path indicates the selected course of action)

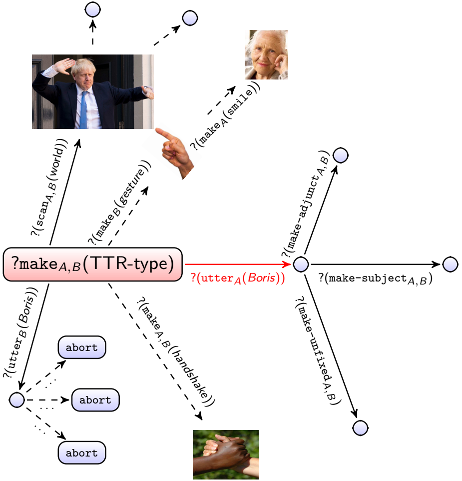


Alice to Bob: "Boris!"

fractal structure of state transition system

(red path indicates the selected course of action)

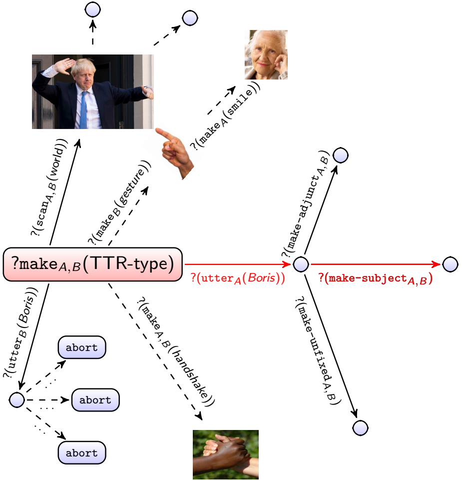
Alice to Bob: "Boris!"



fractal structure of state transition system

(red path indicates the selected course of action)

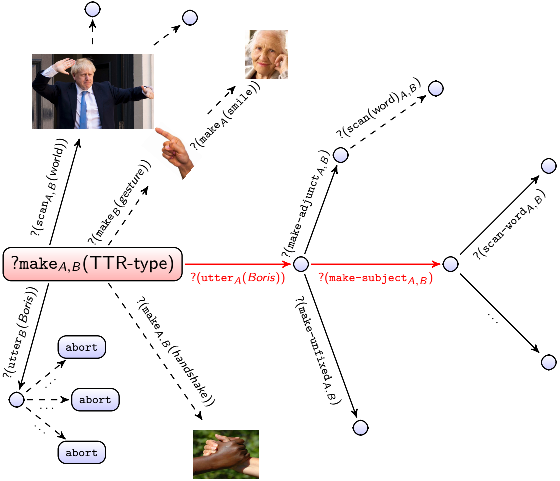
Alice to Bob: "Boris!"



fractal structure of state transition system

(red path indicates the selected course of action)

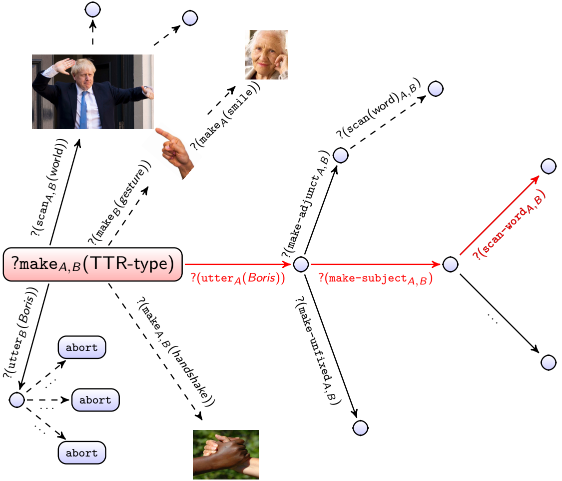
Alice to Bob: "Boris!"



fractal structure of state transition system

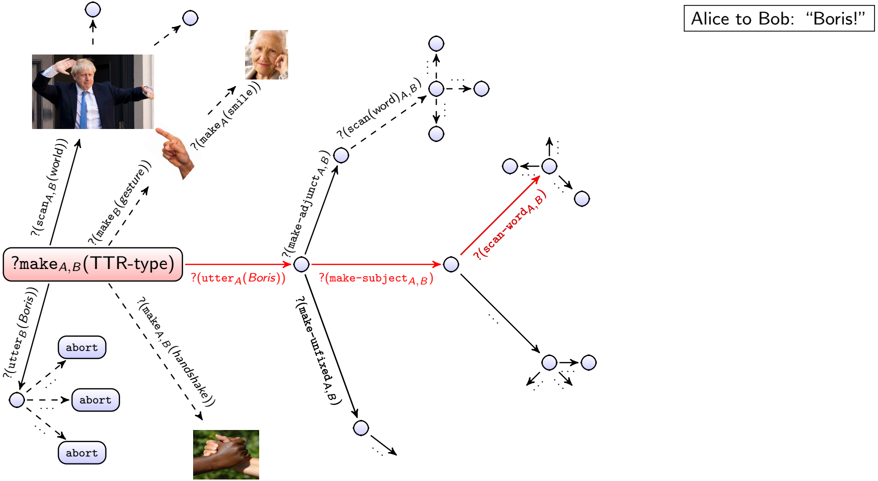
(red path indicates the selected course of action)

Alice to Bob: "Boris!"



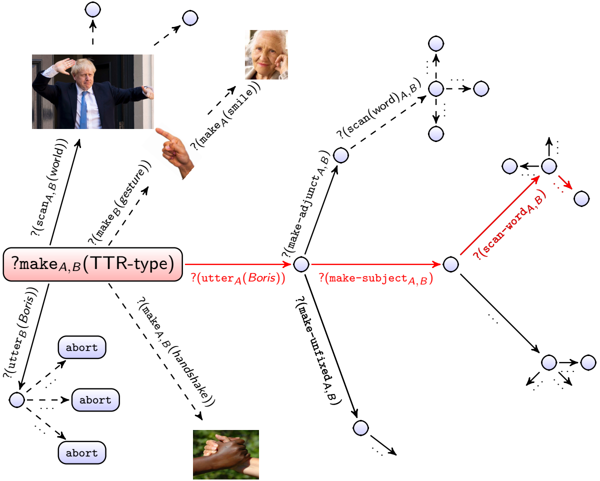
fractal structure of state transition system

(red path indicates the selected course of action)

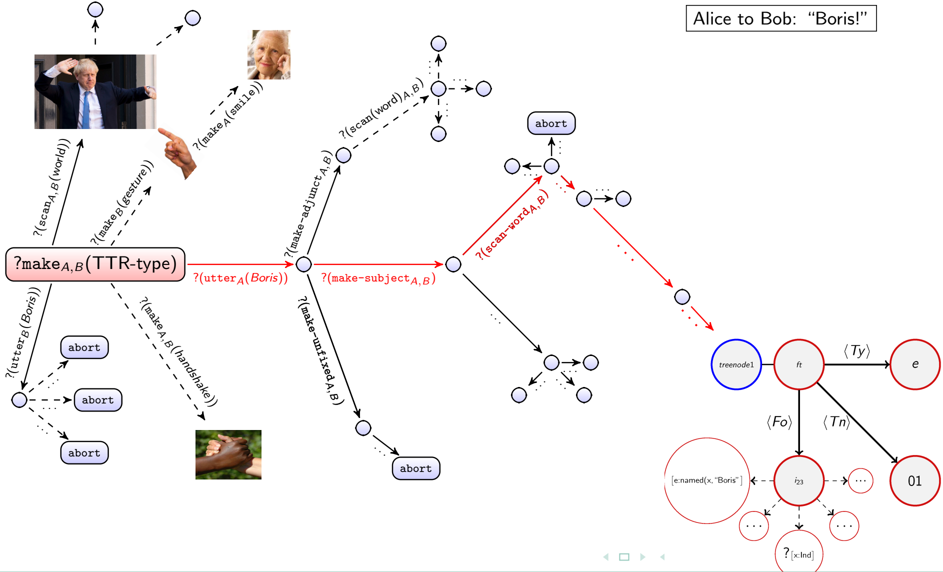


(red path indicates the selected course of action)

Alice to Bob: "Boris!"

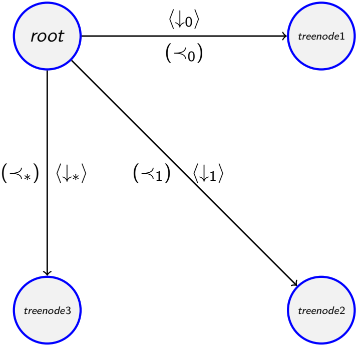


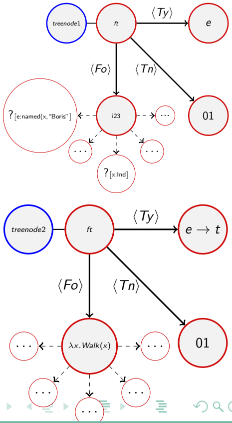
(red path indicates the selected course of action)



Outline

- DS explains all allegedly syntactic phenomena as properties of processing mechanisms or the structuration of affordances into hierarchical structures
- (partial) **tree-structures** emerge as an intermediate structural bottleneck in the DS-TTR landscape of affordances
  - trees express the bifurcation of processes





actions first

- actions as first-class citizens

actions first

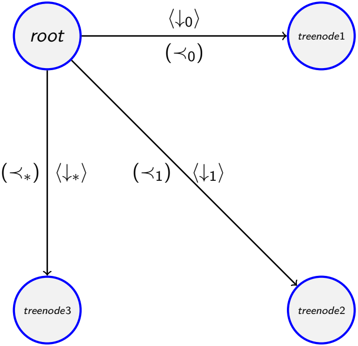
- actions as first-class citizens
  - actions can refer to actions (for, e.g. sloppy ellipsis)

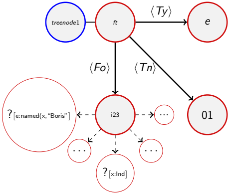
actions first

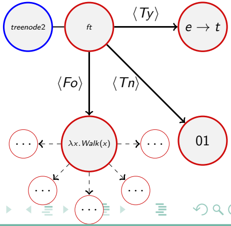
- actions as first-class citizens
  - actions can refer to actions (for, e.g. sloppy ellipsis)
    - the grammar can talk about the grammar (for, e.g., "metalinguistic" quotation)

actions first

- actions as first-class citizens
  - actions can refer to actions (for, e.g. sloppy ellipsis)
    - the grammar can talk about the grammar (for, e.g., “metalinguistic” quotation)
- (if necessary,) representation as an aspect of successful interaction [Bickhard (2009)] via predictivity







DS-TTR: parsing and generation

- from **strings** to **conceptual structure** (TTR) or vice-versa

DS-TTR: parsing and generation

- from **strings** to **conceptual structure** (TTR) or vice-versa

John arrived.

DS-TTR: parsing and generation

- from **strings** to **conceptual structure** (TTR) or vice-versa

John arrived.

John arrived
$\longmapsto$

$$\diamondsuit, Ty(t), \begin{bmatrix} x & : & john' \\ p & : & arrive'(x) \end{bmatrix}$$

$$Ty(e), \begin{bmatrix} x & : & john' \end{bmatrix}$$

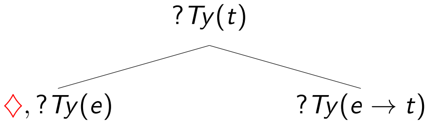
$$Ty(e \rightarrow t), \lambda r : \begin{bmatrix} x & : & e \end{bmatrix} . \begin{bmatrix} x & : & e \\ p & : & arrive'(x) \end{bmatrix}$$

incremental construction

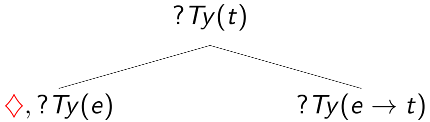
$[\text{START}] \overset{\dots \ \text{PREDICTION}}{\longmapsto}$ $\diamondsuit, ?Ty(t)$

incremental construction

$\overset{\text{PREDICTION}}{\longmapsto}$

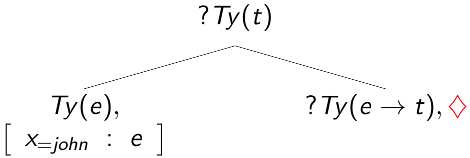


John $\longmapsto$



John	IF	$?Ty(e)$
	THEN	$\mathtt{put}(Ty(e))$
		$\mathtt{put}([\ x_{=john}\ :\ e\ ])$
	ELSE	$\mathtt{abort}$

John,...,POINTER-MOVEMENT $\longmapsto$



$$
\begin{array}{lll}
\textit{John} & \text{IF} & ?Ty(e) \\
 & \text{THEN} & \mathtt{put}(Ty(e)) \\
 & & \mathtt{put}(\left[\ x_{=john}\ :\ e\ \right]) \\
 & \text{ELSE} & \mathtt{abort}
\end{array}
$$

incremental construction

$\ldots, \ldots, \underset{\longrightarrow}{}$ *arrives*

$$?Ty(t)$$

$$Ty(e), \left[\begin{array}{lcl} x_{=john} & : & e \end{array}\right]$$

$$Ty(e \rightarrow t), \diamondsuit \quad \lambda \; r : \left[\begin{array}{lcl} x & : & e \end{array}\right] . \left[\begin{array}{lcl} x & : & e \\ s_{=arrive} & : & e_s \\ p_{=agent(s,x)} & : & t \\ \ldots & : & \ldots \end{array}\right]$$

incremental construction

$\ldots[\textsc{tense},\ \ldots] \xrightarrow{,}\ \textsc{completion}$

$$\diamondsuit,\ Ty(t),\ \begin{bmatrix} x_{=john} & : & e \\ s_{=arrive} & : & e_s \\ p_{=agent(s,x)} & : & t \\ \ldots & : & \ldots \end{bmatrix}$$

$$Ty(e),\ \begin{bmatrix} x_{=john} & : & e \end{bmatrix}$$

$$Ty(e \rightarrow t),\ \lambda\ r : \begin{bmatrix} x & : & e \end{bmatrix} .\ \begin{bmatrix} x & : & e \\ s_{=arrive} & : & e_s \\ p_{=agent(s,x)} & : & t \\ \ldots & : & \ldots \end{bmatrix}$$

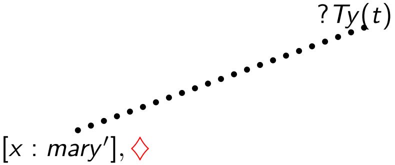
underspecification: structural

- processing **non-contiguous dependencies**
  - e.g. 'Mary, John upset'

$$?Ty(t), \Diamond$$

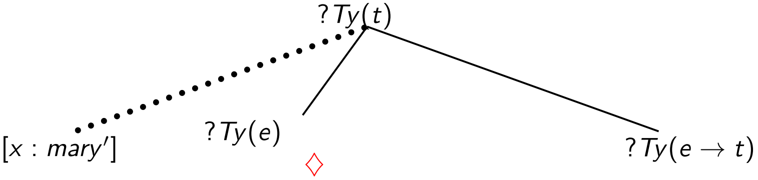
- processing **non-contiguous dependencies**
  - e.g. 'Mary, John upset'

'Mary



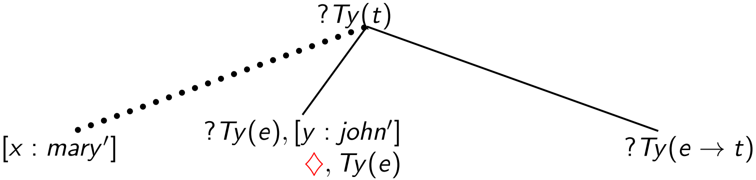
- processing **non-contiguous dependencies**
  - e.g. ‘Mary, John upset’

‘Mary



- processing **non-contiguous dependencies**
  - e.g. 'Mary, John upset'

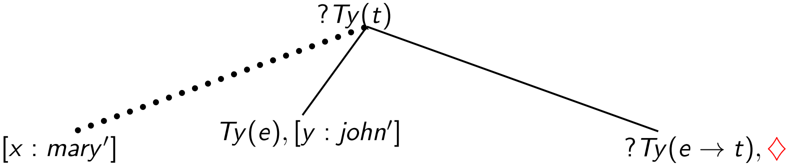
'Mary, John



underspecification: structural

- processing **non-contiguous dependencies**
  - e.g. ‘Mary, John upset’

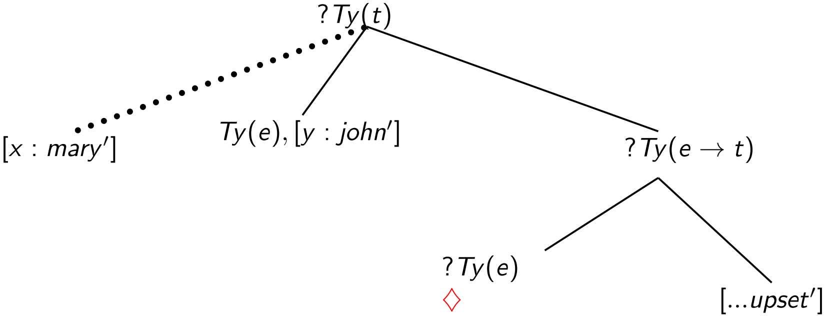
‘Mary, John



underspecification: structural

- processing **non-contiguous dependencies**
  - e.g. 'Mary, John upset'

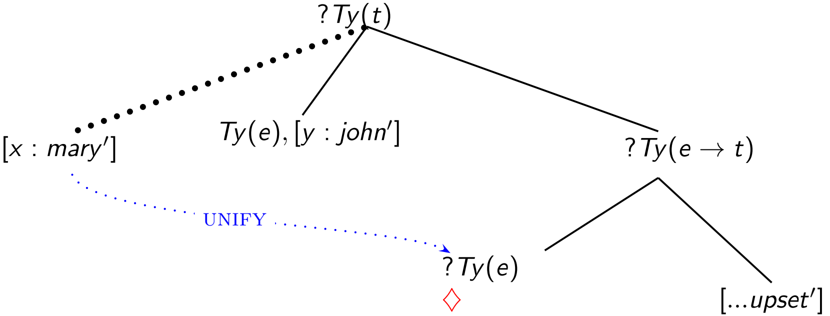
'Mary, John upset'



underspecification: structural

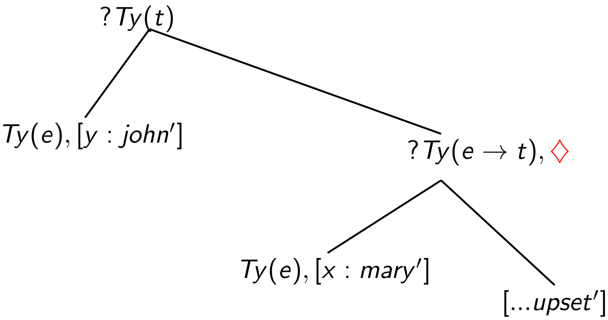
- processing **non-contiguous dependencies**
  - e.g. 'Mary, John upset'

'Mary, John upset'



- processing **non-contiguous dependencies**
  - e.g. ‘Mary, John upset’

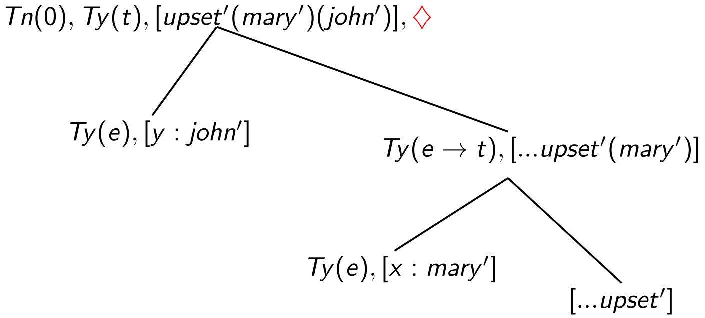
‘Mary, John upset’



underspecification: structural

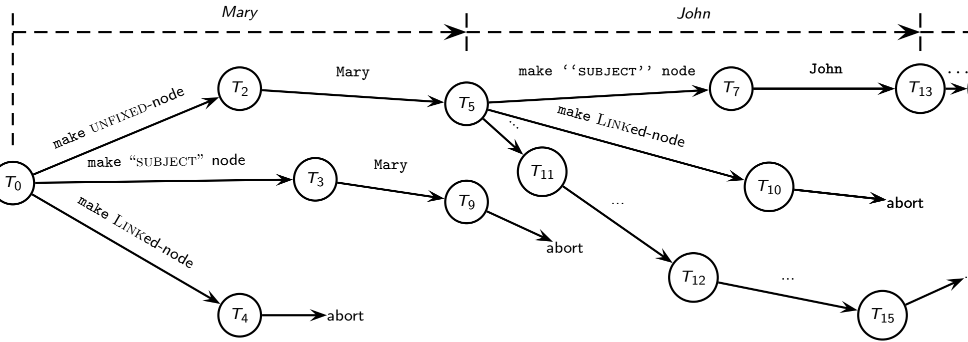
- processing **non-contiguous dependencies**
  - e.g. 'Mary, John upset'

'Mary, John upset'



a field of affordances

(simplified) options for starting to parse/produce *Mary, John upset*



utterance micro-events

$$\diamondsuit, Ty(t), \begin{bmatrix} \text{CONTEXT}: & u_1 \oplus u_2 \\ \text{CONTENT}: & \begin{bmatrix} x & : & e \\ p & : & f(x) \end{bmatrix} \end{bmatrix}$$

$$Ty(e), \begin{bmatrix} \text{CONTEXT}: & u_2 \\ \text{CONTENT}: & \begin{bmatrix} x & : & e \end{bmatrix} \end{bmatrix}$$

$$Ty(e \rightarrow t), \begin{bmatrix} \text{CONTEXT}: & u_1 \\ \text{CONTENT}: & \lambda r : \begin{bmatrix} x & : & e \end{bmatrix} . \begin{bmatrix} x & : & e \\ p & : & f(x) \end{bmatrix} \end{bmatrix}$$

including contextual parameters

John arrived $\longmapsto$

$$\diamondsuit, Ty(t), \begin{bmatrix} \textsc{context}: & \begin{bmatrix} a & : & participantA \\ b & : & participantB \\ \ldots & : & \ldots \\ u_1 & : & utt-event \\ s_{s1} & : & spkr(u_1, a) \\ s_{a1} & : & addr(u_1, b) \\ u_2 & : & utt-event \\ s_{s2} & : & spkr(u_2, a) \\ s_{a2} & : & addr(u_2, b) \\ \ldots & : & \ldots \end{bmatrix} \\ \textsc{content}: & \begin{bmatrix} x & : & john \\ p & : & arrive(x) \end{bmatrix} \end{bmatrix}$$

$$Ty(e), \begin{bmatrix} \textsc{context}: & \begin{bmatrix} u_1 & : & utt-event \\ \ldots & : & \ldots \\ s_{s1} & : & spkr(u_1, a) \\ \ldots & : & \ldots \end{bmatrix} \\ \textsc{content}: & \begin{bmatrix} x & : & john \end{bmatrix} \end{bmatrix}$$

$$Ty(e \rightarrow t), \begin{bmatrix} \textsc{context}: & \begin{bmatrix} u_2 & : & utt-event \\ \ldots & : & \ldots \\ s_{s2} & : & spkr(u_2, a) \\ \ldots & : & \ldots \end{bmatrix} \\ \textsc{content}: & \lambda r : [x : e] . \begin{bmatrix} p & : & arrive(x) \end{bmatrix} \end{bmatrix}$$

DS-TTR: some suggestions

- make types dynamic and (mainly) subpersonal

DS-TTR: some suggestions

- make types dynamic and (mainly) subpersonal
- types are not within one's head (optional)

DS-TTR: some suggestions

- make types dynamic and (mainly) subpersonal
- types are not within one's head (optional)
- types are not implemented by neural processes within one agent (optional)
- types too are (triggers for) affordances

DS-TTR: some suggestions

- make types dynamic and (mainly) subpersonal
- types are not within one's head (optional)
- types are not implemented by neural processes within one agent (optional)
- types too are (triggers for) affordances
  - they bring about properties of the environment to be picked up by agents

DS-TTR: some suggestions

- make types dynamic and (mainly) subpersonal
- types are not within one's head (optional)
- types are not implemented by neural processes within one agent (optional)
- types too are (triggers for) affordances
  - they bring about properties of the environment to be picked up by agents
  - they are agent-relative (perspectival) properties of the environment [(Chemero, 2010)]

DS-TTR: some suggestions

- make types dynamic and (mainly) subpersonal
- types are not within one's head (optional)
- types are not implemented by neural processes within one agent (optional)
- types too are (triggers for) affordances
  - they bring about properties of the environment to be picked up by agents
  - they are agent-relative (perspectival) properties of the environment [(Chemero, 2010)]
- types are not paired instantaneously with labels: types induce fields of potential paths of interaction with patterns in the environment
  - "the whole system of input and output resonates to the external information" [Gibson, 1966: 5].

Outline

1. DS-TTR
2. Zoom into trees and treenodes
3. DS-TTR+
4. Appendix

logical structure of DS

- an NL is a set of actions describing the licensed transitions from an initial state to a result state

logical structure of DS

- an NL is a set of actions describing the licensed transitions from an initial state to a result state
- states are partially ordered

logical structure of DS

- an NL is a set of actions describing the licensed transitions from an initial state to a result state
- states are partially ordered
- states are structured objects
  - e.g., in DS, states are decorated partial trees which map to more elaborate trees as you go along

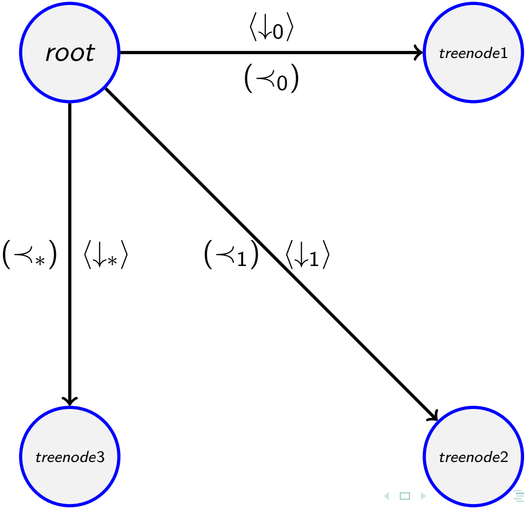
logical structure of DS: trees

NL: actions from an initial state to a result state

logical structure of DS: trees

NL: actions from an initial state to a result state

- in DS, states are partial trees (in the description language DU, they are accessed by means of modal operators)



logical structure of DS: treenodes

- DS **trees** are sets of states related by dominance (modal) operators
- in turn, each **tree node** is inhabited by a feature structure
  - an instance of "fibred semantics" [Finger, Marcelo and Gabbay, Dov (1992)]
- **feature structures** are labelled directed graphs
- features (partial functions) like *Fo*, *Tn*, *Ty*, ...
  - nesting of features is possible
- values ("decorations") like *e*, *eleni′*, 010, ...
- AVM notation:

$$tn_{01}: \begin{bmatrix} Type & e \rightarrow t \\ Formula & \lambda x Walk(x) \\ TreeNode & 01 \end{bmatrix}$$

logical structure of DS: decorations

- a treenode is a state

logical structure of DS: decorations

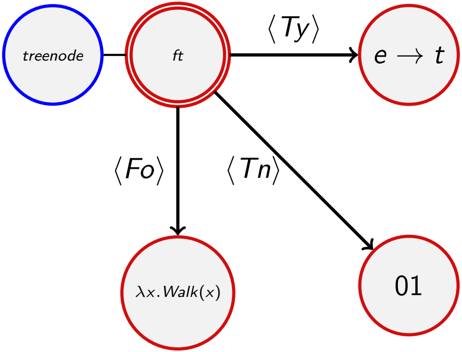
- a treenode is a state
- each treenode is assigned a feature structure

logical structure of DS: decorations

- a treenode is a state
- each treenode is assigned a feature structure
- labels are modal operators:

logical structure of DS: decorations

- a treenode is a state
- each treenode is assigned a feature structure
- labels are modal operators:



logical structure of DS: requirements

- each treenode is decorated by a feature structure (labelled directed graph) with features like *Fo*, *Tn*, *Ty*

logical structure of DS: requirements

- each treenode is decorated by a feature structure (labelled directed graph) with features like *Fo*, *Tn*, *Ty*
- a **pointer** (a proposition holding at only one node of a tree model) defines an equivalence class of tree models

logical structure of DS: requirements

- each treenode is decorated by a feature structure (labelled directed graph) with features like *Fo*, *Tn*, *Ty*
- a **pointer** (a proposition holding at only one node of a tree model) defines an equivalence class of tree models
- besides resolved feature structures, each treenode is also assigned a set of **requirements**

logical structure of DS: requirements

- each treenode is decorated by a feature structure (labelled directed graph) with features like *Fo*, *Tn*, *Ty*
- a **pointer** (a proposition holding at only one node of a tree model) defines an equivalence class of tree models
- besides resolved feature structures, each treenode is also assigned a set of **requirements**
- all (atomic) decorations can appear as requirements

logical structure of DS: requirements

- each treenode is decorated by a feature structure (labelled directed graph) with features like *Fo*, *Tn*, *Ty*
- a **pointer** (a proposition holding at only one node of a tree model) defines an equivalence class of tree models
- besides resolved feature structures, each treenode is also assigned a set of **requirements**
- all (atomic) decorations can appear as requirements
- AVM notation:

$$tn_{01}: \left[\begin{array}{ll} Tn & 01 \\ Type & e \rightarrow t \\ \hline Fo & \mathbf{x} \end{array} \right]$$

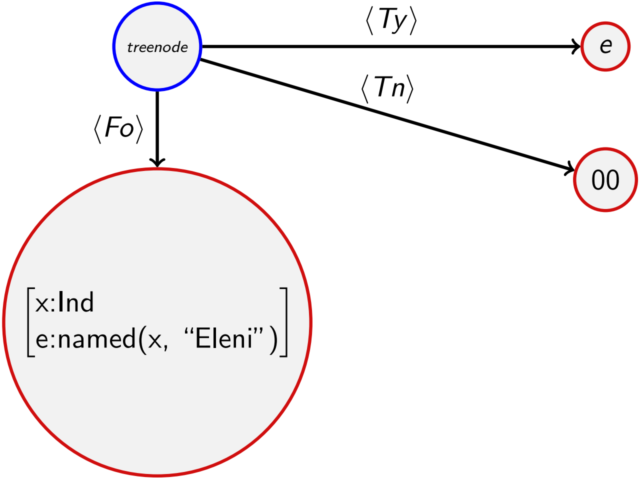
- each treenode is decorated by a feature structure (labelled directed graph) with features like Fo, Tn, Ty
- a **pointer** (a proposition holding at only one node of a tree model) defines an equivalence class of tree models
- besides resolved feature structures, each treenode is also assigned a set of **requirements**
- all (atomic) decorations can appear as requirements
- AVM notation:

$$tn_{01}: \left[\begin{array}{ll} Tn & 01 \\ Type & e \rightarrow t \\ \hline Fo & \mathbf{x} \end{array}\right]$$

- DU_F(?) language notation

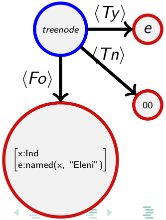
$\langle Tn\rangle(01) \bullet ?\langle Ty\rangle(e), ?\exists(\mathbf{x})\langle Fo\rangle(\mathbf{x})$

DS-TTR: TTR types as Fo values



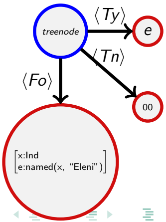
logical structure of DS

- current versions of DS-TTR: each treenode is inhabited by a TTR type within the label equivalent to *Fo*



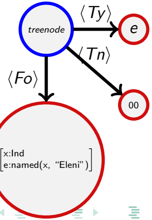
logical structure of DS

- current versions of DS-TTR: each treenode is inhabited by a TTR type within the label equivalent to *Fo*
- however, we want to be able to zoom into such Fo values



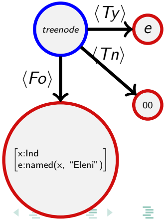
logical structure of DS

- current versions of DS-TTR: each treenode is inhabited by a TTR type within the label equivalent to *Fo*
- however, we want to be able to zoom into such Fo values
  - decompose TTR record types
- in order to create such types incrementally



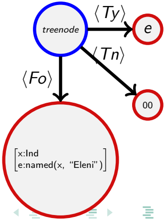
logical structure of DS

- current versions of DS-TTR: each treenode is inhabited by a TTR type within the label equivalent to *Fo*
- however, we want to be able to zoom into such Fo values
  - decompose TTR record types
- in order to create such types incrementally
- we can exploit predictions to start from some empty or underspecified type to more elaborate types



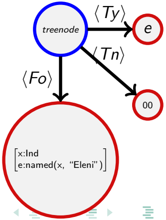
logical structure of DS

- current versions of DS-TTR: each treenode is inhabited by a TTR type within the label equivalent to *Fo*
- however, we want to be able to zoom into such Fo values
  - decompose TTR record types
- in order to create such types incrementally
- we can exploit predictions to start from some empty or underspecified type to more elaborate types
- moving along the subtype relation



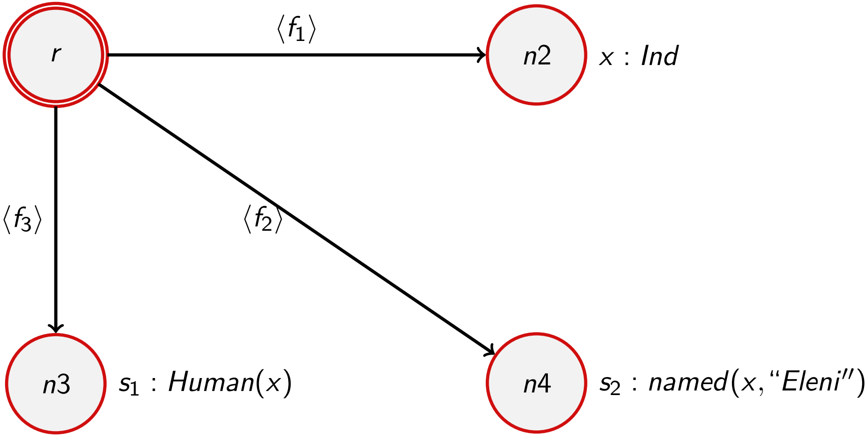
logical structure of DS

- current versions of DS-TTR: each treenode is inhabited by a TTR type within the label equivalent to *Fo*
- however, we want to be able to zoom into such Fo values
  - decompose TTR record types
- in order to create such types incrementally
- we can exploit predictions to start from some empty or underspecified type to more elaborate types
- moving along the subtype relation
  - introduce requirements to impose further specification
- additionally, we can "internalise" trees in TTR types so that it is whole TTR types that are extended by DS actions



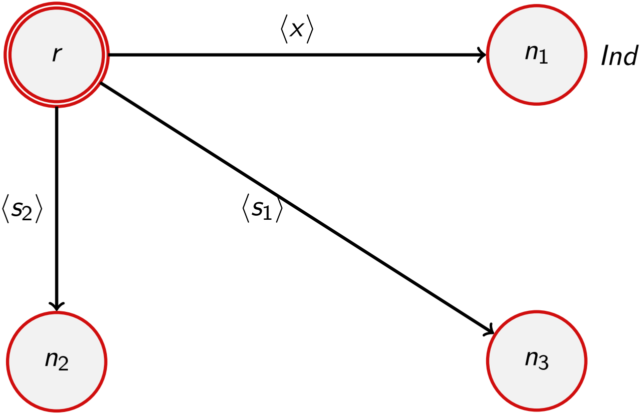
logical structure of TTR: version 1 (conservative)

- assign syntactic structure to **record types**
- **typing judgements** are propositions that hold at nodes
- fields are indexed



logical structure of TTR: version 2a

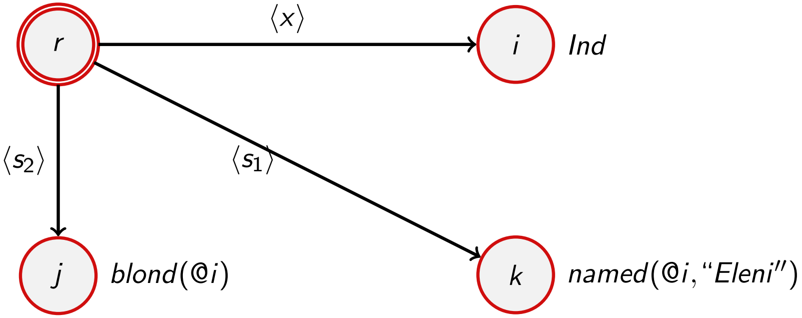
- a **record type** is a state (an initial, base, state)
- labels (discourse referents) are represented as modal operators (features)
- entities are nodes



$\lambda z.blond(z) \wedge \left\langle z : \langle s_2 \rangle \langle x \rangle :\triangleq \langle 0 \rangle n_1 \right\rangle$ $\lambda y.named(y, \text{“}Eleni''') \wedge \left\langle y : \langle s_1 \rangle \langle x \rangle \triangleq \langle 0 \rangle n_1 \right\rangle$

logical structure of TTR: version 2b

- a **record type** is a state (an initial state)
- labels are represented as modal operators
- node names are *nominals* in Hybrid Logic [(Blackburn, 2000)]
  - this will bring us close to the Duesseldorf Frames implementation [Kallmeyer and Osswald (2013); Kallmeyer et al. (2015)]
- types are (predicates of) states:



- but then do we still need labels?

logical structure of TTR: set-theoretic structures

- ontology

logical structure of TTR: set-theoretic structures

- ontology
  - *urelements* proper (*up*, entities)

logical structure of TTR: set-theoretic structures

- ontology
  - *urelements* proper (*up*, entities)
  - **labels** (*l*, i.e., discourse referents)
    - labels are entities in the world, at the same level as ordinary individuals

logical structure of TTR: set-theoretic structures

- ontology
  - *urelements* proper (*up*, entities)
  - **labels** (*l*, i.e., discourse referents)
    - labels are entities in the world, at the same level as ordinary individuals
  - **labelled sets** (*ls*): sets of ordered pairs $\langle l, up \rangle$ or $\langle l, ls \rangle$

logical structure of TTR: set-theoretic structures

- ontology
  - *urelements* proper (*up*, entities)
  - **labels** (l, i.e., discourse referents)
    - labels are entities in the world, at the same level as ordinary individuals
  - **labelled sets** (*ls*): sets of ordered pairs $\langle l, up \rangle$ or $\langle l, ls \rangle$
- basic types are interpreted as sets

logical structure of TTR: set-theoretic structures

- ontology
  - *urelements* proper (*up*, entities)
  - **labels** (*l*, i.e., discourse referents)
    - labels are entities in the world, at the same level as ordinary individuals
  - **labelled sets** (*ls*): sets of ordered pairs $\langle l, up \rangle$ or $\langle l, ls \rangle$
- basic types are interpreted as sets
- an entity (picked up by a label) is of some type if it's a member of the set assigned to the type

logical structure of TTR: set-theoretic structures

- ontology
  - *urelements* proper (*up*, entities)
  - **labels** (*l*, i.e., discourse referents)
    - labels are entities in the world, at the same level as ordinary individuals
  - **labelled sets** (*ls*): sets of ordered pairs $\langle l, up \rangle$ or $\langle l, ls \rangle$
- basic types are interpreted as sets
- an entity (picked up by a label) is of some type if it's a member of the set assigned to the type
- labelled sets are the basic structure for building complex types

logical structure of TTR: set-theoretic structures

- ontology
  - *urelements* proper (*up*, entities)
  - **labels** (*l*, i.e., discourse referents)
    - labels are entities in the world, at the same level as ordinary individuals
  - **labelled sets** (*ls*): sets of ordered pairs $\langle l, up \rangle$ or $\langle l, ls \rangle$
- basic types are interpreted as sets
- an entity (picked up by a label) is of some type if it's a member of the set assigned to the type
- labelled sets are the basic structure for building complex types

logical structure of TTR: ptypes

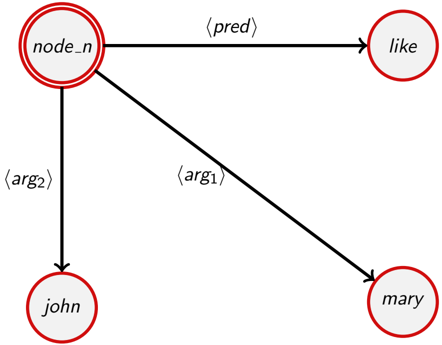
- **ptypes** (predicative types) $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types

logical structure of TTR: ptypes

- **ptypes** (predicative types) $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types
- ptypes are assigned sets of entities, labelled sets: $\{\langle \text{ pred }, P\rangle, \langle \text{arg}_1, a_1\rangle, \ldots, \langle \text{arg}_n, a_n\rangle\}$

logical structure of TTR: ptypes

- **ptypes** (predicative types) $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types
- ptypes are assigned sets of entities, labelled sets: $\{\langle \text{ pred }, P\rangle, \langle \text{arg}_1, a_1\rangle, \ldots, \langle \text{arg}_n, a_n\rangle\}$



logical structure of TTR: ptypes as in DS-TTR - Neodavidsonian (a)

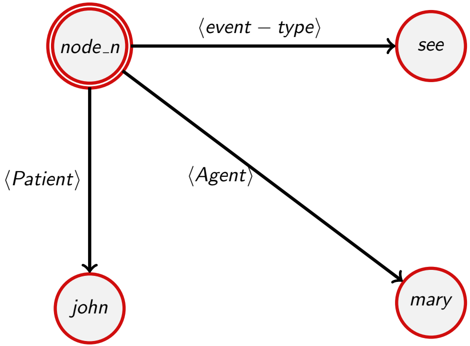
- **ptypes** (predicative types) $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types

logical structure of TTR: ptypes as in DS-TTR - Neodavidsonian (a)

- **ptypes** (predicative types) $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types
- ptypes are assigned sets of entities, labelled sets:
  $\{\langle \text{ pred }, P\rangle, \langle \text{arg}_1, a_1\rangle, \ldots, \langle \text{arg}_n, a_n\rangle\}$

logical structure of TTR: ptypes as in DS-TTR - Neodavidsonian (a)

- **ptypes** (predicative types) $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types
- ptypes are assigned sets of entities, labelled sets: $\{\langle \text{ pred } , P\rangle, \langle \text{arg}_1, a_1\rangle, \ldots, \langle \text{arg}_n, a_n\rangle\}$



logical structure of TTR: version Neodavidsonian (b)

- ptypes $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types

logical structure of TTR: version Neodavidsonian (b)

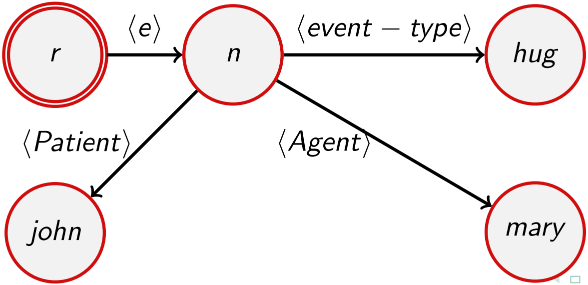
- ptypes $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types
- ptypes are assigned sets of entities, labelled sets $\{\langle \text{ pred } , P\rangle, \langle \text{arg}_1, a_1\rangle , \ldots , \langle \text{arg}_n, a_n\rangle\}$

logical structure of TTR: version Neodavidsonian (b)

- ptypes $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types
- ptypes are assigned sets of entities, labelled sets $\{\langle \text{ pred } , P\rangle, \langle \text{arg}_1, a_1\rangle , \ldots , \langle \text{arg}_n, a_n\rangle\}$
- a record type is a state, labels are modal operators, values are states: [e: hug(john, mary)]

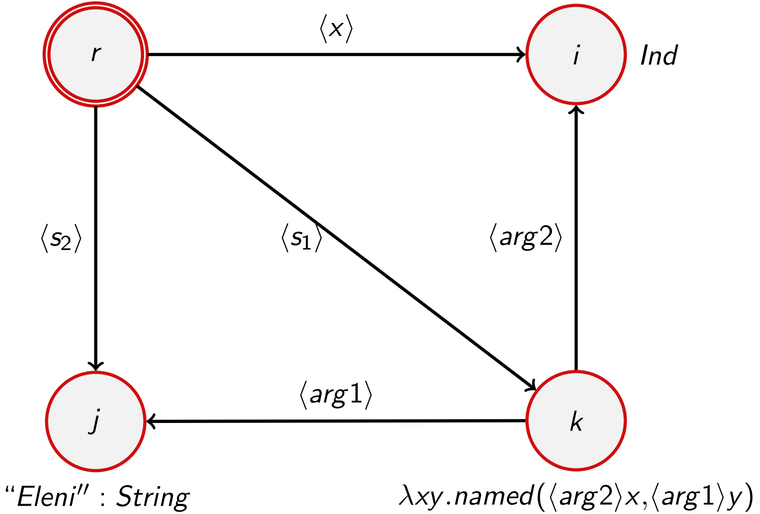
logical structure of TTR: version Neodavidsonian (b)

- ptypes $P(a_1, ..., a_n)$, are combinations of predicates and arguments of particular types
- ptypes are assigned sets of entities, labelled sets $\{\langle$ pred $, P\rangle, \langle \text{arg}_1, a_1\rangle, \ldots, \langle \text{arg}_n, a_n\rangle\}$
- a record type is a state, labels are modal operators, values are states: [e: hug(john, mary)]



logical structure of TTR: version (4a)

labels are modal operators



logical structure of TTR: situational version 4b

- labels are names of states, types are feature names (modal operators)

logical structure of TTR: situational version 4b

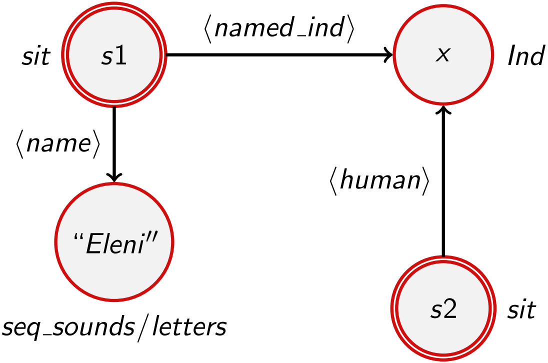
- labels are names of states, types are feature names (modal operators)
- individuals are situations

logical structure of TTR: situational version 4b

- labels are names of states, types are feature names (modal operators)
- individuals are situations
- predicates are feature names (modal operators)

logical structure of TTR: situational version 4b

- labels are names of states, types are feature names (modal operators)
- individuals are situations
- predicates are feature names (modal operators)



actions in DS: PDL

- Propositional Dynamic Logic (PDL) is a modal logic introduced to capture the behaviour of programs

actions in DS: PDL

- Propositional Dynamic Logic (PDL) is a modal logic introduced to capture the behaviour of programs
- the models for PDL formulas are transition systems
  - edges are labelled with programs (actions)
  - states are labelled with propositions

actions in DS: PDL

- Propositional Dynamic Logic (PDL) is a modal logic introduced to capture the behaviour of programs
- the models for PDL formulas are transition systems
  - edges are labelled with programs (actions)
  - states are labelled with propositions
- complex formulae (e.g. decorated trees, TTR types) and programs (DS macros) are inductively (and mutually) defined from atomic propositions and programs

actions in DS: PDL

- Propositional Dynamic Logic (PDL) is a modal logic introduced to capture the behaviour of programs
- the models for PDL formulas are transition systems
  - edges are labelled with programs (actions)
  - states are labelled with propositions
- complex formulae (e.g. decorated trees, TTR types) and programs (DS macros) are inductively (and mutually) defined from atomic propositions and programs
- formulas are closed by (at least) the standard Boolean operations

actions in DS: PDL

- Propositional Dynamic Logic (PDL) is a modal logic introduced to capture the behaviour of programs
- the models for PDL formulas are transition systems
  - edges are labelled with programs (actions)
  - states are labelled with propositions
- complex formulae (e.g. decorated trees, TTR types) and programs (DS macros) are inductively (and mutually) defined from atomic propositions and programs
- formulas are closed by (at least) the standard Boolean operations
- a program is a regular language (represented by a regular expression / finite automaton) over the set of atomic programs and tests (where tests correspond to formulas)

actions in DS: PDL

- Propositional Dynamic Logic (PDL) is a modal logic introduced to capture the behaviour of programs
- the models for PDL formulas are transition systems
  - edges are labelled with programs (actions)
  - states are labelled with propositions
- complex formulae (e.g. decorated trees, TTR types) and programs (DS macros) are inductively (and mutually) defined from atomic propositions and programs
- formulas are closed by (at least) the standard Boolean operations
- a program is a regular language (represented by a regular expression / finite automaton) over the set of atomic programs and tests (where tests correspond to formulas)
- for each program α and each formula (tree, TTR-type) φ, $\langle\alpha\rangle\varphi$ or $\langle do(\alpha)\rangle\varphi$ is a formula
  - this means that there is an execution of program α that ends in a state where φ holds

actions in DS: specialisation of PDL

- an NL is a set of actions (programs) describing the licensed transitions from an initial state to a result state

actions in DS: specialisation of PDL

- an NL is a set of actions (programs) describing the licensed transitions from an initial state to a result state
- states in DS: inhabited by pointed (partial) decorated trees

actions in DS: specialisation of PDL

- an NL is a set of actions (programs) describing the licensed transitions from an initial state to a result state
- states in DS: inhabited by pointed (partial) decorated trees
- licensing of transitions between pointed-partial-tree decorated states: NL rules/words and inferences

actions in DS: specialisation of PDL

- an NL is a set of actions (programs) describing the licensed transitions from an initial state to a result state
- states in DS: inhabited by pointed (partial) decorated trees
- licensing of transitions between pointed-partial-tree decorated states: NL rules/words and inferences
  - e.g. from *Axiom* ($?Ty(t)$) to a complete $Ty(t)$ tree
    - via a partial ordering $\leq$ on tree models

actions in DS: specialisation of PDL

- an NL is a set of actions (programs) describing the licensed transitions from an initial state to a result state
- states in DS: inhabited by pointed (partial) decorated trees
- licensing of transitions between pointed-partial-tree decorated states: NL rules/words and inferences
  - e.g. from *Axiom* ($?Ty(t)$) to a complete $Ty(t)$ tree
    - via a partial ordering $\leq$ on tree models
- DS actions:
  - lexical
  - computational
  - pragmatic (*Substitution*, scope resolution, inference)

actions in DS: specialisation of PDL

- in DS, an action constant (program) α determines a binary relation among decorated pointed partial trees (the models): $(\langle \mathcal{PPTM}, \mathcal{T} n \rangle, \langle \mathcal{PPTM}, \mathcal{T}' n' \rangle)$

actions in DS: specialisation of PDL

- in DS, an action constant (program) α determines a binary relation among decorated pointed partial trees (the models): $(\langle \mathcal{PPTM}, \mathcal{T}n \rangle, \langle \mathcal{PPTM}, \mathcal{T}'n' \rangle)$
  - the second member of the relation extends the first in some way

actions in DS: specialisation of PDL

- in DS, an action constant (program) α determines a binary relation among decorated pointed partial trees (the models): $(\langle \mathcal{PPTM}, \mathcal{T}n \rangle, \langle \mathcal{PPTM}, \mathcal{T}'n' \rangle)$
  - the second member of the relation extends the first in some way
- actions in DS are specialised for building structures of decorated trees
- need to be extended for TTR, where TTR operations are not adequate
  - `add`[fields]
  - `remove`[fields]
  - `test`[subtyping relation]
  - ...

actions in DS: specialisation of PDL

- in DS, an action constant (program) α determines a binary relation among decorated pointed partial trees (the models): $(\langle \mathcal{PPTM}, \mathcal{T}n \rangle, \langle \mathcal{PPTM}, \mathcal{T}'n' \rangle)$
  - the second member of the relation extends the first in some way
- actions in DS are specialised for building structures of decorated trees
- need to be extended for TTR, where TTR operations are not adequate
  - `add`[fields]
  - `remove`[fields]
  - `test`[subtyping relation]
  - ...
- *ACT*, the set of basic actions, consists of action constants: $ACT = \{\texttt{ABORT}, 1, \texttt{make}(\#), \texttt{put}(\Sigma), \texttt{go}(\#), \texttt{merg}(\#)\}$

actions in DS

- in DS, an action constant α determines a binary relation among decorated pointed partial trees (the models)

actions in DS

- in DS, an action constant α determines a binary relation among decorated pointed partial trees (the models)
- the action modalities formulae $[\alpha]$ and $\langle\alpha\rangle$ are interpreted as follows:

actions in DS

- in DS, an action constant α determines a binary relation among decorated pointed partial trees (the models)
- the action modalities formulae $[\alpha]$ and $\langle\alpha\rangle$ are interpreted as follows:
  - $\langle \mathcal{PPTM}, \mathcal{T}n \rangle \vDash [\alpha]\phi$

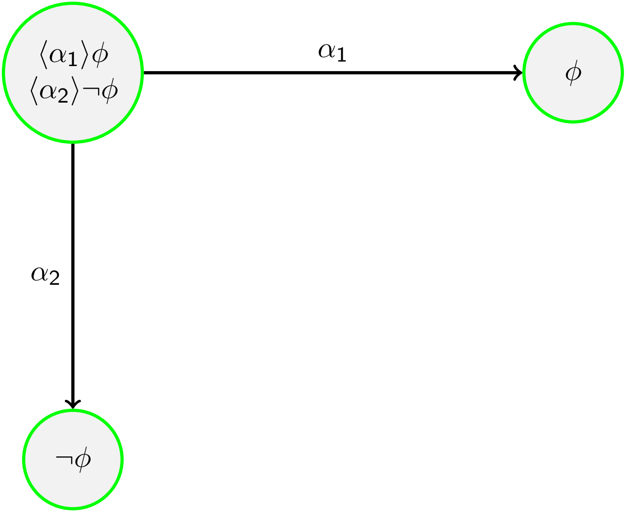
actions in DS

- in DS, an action constant α determines a binary relation among decorated pointed partial trees (the models)
- the action modalities formulae $[\alpha]$ and $\langle\alpha\rangle$ are interpreted as follows:
  - $\langle \mathcal{PPTM}, \mathcal{T}n \rangle \vDash [\alpha]\phi$
    iff for all $\mathcal{T}'n' \in PPTM$ such that $\alpha(\mathcal{T}n, \mathcal{T}'n')$ we have $\langle \mathcal{PPTM}, \mathcal{T}'n' \rangle \vDash \phi$

actions in DS

- in DS, an action constant α determines a binary relation among decorated pointed partial trees (the models)
- the action modalities formulae $[\alpha]$ and $\langle\alpha\rangle$ are interpreted as follows:
  - $\langle \mathcal{PPTM}, \mathcal{T}n \rangle \models [\alpha]\phi$
    iff for all $\mathcal{T}'n' \in PPTM$ such that $\alpha(\mathcal{T}n, \mathcal{T}'n')$ we have
    $\langle \mathcal{PPTM}, \mathcal{T}'n' \rangle \models \phi$
  - $\langle \mathcal{PPTM}, \mathcal{T}n \rangle \models \langle\alpha\rangle\phi$
    iff there is some $\mathcal{T}'n' \in PPTM$ such that $\alpha(\mathcal{T}n, \mathcal{T}'n')$ and
    $\langle \mathcal{PPTM}, \mathcal{T}'n' \rangle \models \phi$

actions as modal operators in DS



composite actions in DS

- given ACT, the set of basic actions consisting of $ACT = \{\mathtt{ABORT}, 1, \mathtt{make}(\#), \mathtt{put}(\Sigma), \mathtt{go}(\#), \mathtt{merg}(\#)\}$

composite actions in DS

- given ACT, the set of basic actions consisting of $ACT = \{\texttt{ABORT}, 1, \texttt{make}(\#), \texttt{put}(\Sigma), \texttt{go}(\#), \texttt{merg}(\#)\}$
- basic actions are combined by the regular operations
- plus an IF THEN ELSE construction

composite actions in DS

- given ACT, the set of basic actions consisting of $ACT = \{\mathtt{ABORT}, 1, \mathtt{make}(\#), \mathtt{put}(\Sigma), \mathtt{go}(\#), \mathtt{merg}(\#)\}$
- basic actions are combined by the regular operations
- plus an IF THEN ELSE construction
- to form $\mathcal{A}(ACT)$, the set of composite actions

composite actions in DS

- given ACT, the set of basic actions consisting of $ACT = \{\mathtt{ABORT}, 1, \mathtt{make}(\#), \mathtt{put}(\Sigma), \mathtt{go}(\#), \mathtt{merg}(\#)\}$
- basic actions are combined by the regular operations
- plus an IF THEN ELSE construction
- to form $\mathcal{A}(ACT)$, the set of composite actions
- the set $\mathcal{A}(ACT)$ is the smallest set in $PPTM \times PPTM$ satisfying

1. $ACT \subseteq \mathcal{A}(ACT)$, and
2. for $\alpha, \alpha' \in \mathcal{A}(ACT), \Sigma(\bar{x}) \subseteq DU_F(?)$,
   where $\bar{x}$ is a sequence of all variables occurring free in Σ, we have $\alpha;\alpha', \alpha + \alpha', \alpha^*, \langle \Sigma(\bar{x}), \alpha, \alpha' \rangle \in \mathcal{A}(ACT)$ where $;, +, *$ have their usual interpretation and
   $\langle \Sigma(\bar{x}), \alpha, \alpha' \rangle =$
   $\{\langle \mathcal{T}n, \mathcal{T}'n' \rangle \in \alpha[\bar{t}/\bar{x}] \,|\bar{t} \in (\mathcal{D} \cup MV)^*, \langle \mathcal{PPTM}, \mathcal{T}n \rangle| = \Sigma[\bar{t}/\bar{x}]\}$
   $\{\langle \mathcal{T}n, \mathcal{T}'n' \rangle \in \alpha' | \neg\exists \bar{t} \in (\mathcal{D} \cup MV)^*, \langle \mathcal{PPTM}, \mathcal{T}n \rangle = \Sigma[\bar{t}/\bar{x}]\}$

actions in DS: macros

- elements of $\mathcal{A}(ACT)$, the set of composite actions, can be combined into *macros*, actions executed in sequence:
  - lexical actions
  - Introduction-Prediction, Completion, Elimination, LINK-introduction, ...

actions in DS: macros

- elements of $\mathcal{A}(ACT)$, the set of composite actions, can be combined into *macros*, actions executed in sequence:
  - lexical actions
  - Introduction-Prediction, Completion, Elimination, LINK-introduction, ...
- DS actions develop decorated trees
  - lexical
  - computational
    - can be translated as IF-THEN-ELSE composites
  - pragmatic
    - *Substitution*,
    - scope resolution
    - (pragmatic) inference

actions in DS: macros

- elements of $\mathcal{A}(ACT)$, the set of composite actions, can be combined into *macros*, actions executed in sequence:
  - lexical actions
  - Introduction-Prediction, Completion, Elimination, LINK-introduction, ...
- DS actions develop decorated trees
  - lexical
  - computational
    - can be translated as IF-THEN-ELSE composites
  - pragmatic
    - *Substitution*,
    - scope resolution
    - (pragmatic) inference

actions in DS-TTR

- DS-TTR actions: use extended set of DS actions to build incrementally structured *Fo* values as shown earlier
  - each state is a (partial) TTR record type
  - described by a modal formula ("feature structure") concerning its structure and requirements
- with transitions licensed via the subtype relation (ordering $\leq$)
  - e.g. from an empty TTR type to a further specified TTR type

actions in DS-TTR

- DS-TTR actions: use extended set of DS actions to build incrementally structured *Fo* values as shown earlier
  - each state is a (partial) TTR record type
  - described by a modal formula ("feature structure") concerning its structure and requirements
- with transitions licensed via the subtype relation (ordering $\leq$)
  - e.g. from an empty TTR type to a further specified TTR type
- but remove maximal states from DS models
- remove restriction that elimination of requirements (predictions) is criterion of wellformedness
  - requirements are constantly generated and always present to guide further development

physical actions in DS-TTR

- exploit PDL specialisations (multimodal languages) in the literature

physical actions in DS-TTR

- exploit PDL specialisations (multimodal languages) in the literature
- introduce operators for results, opportunity, ability, "epistemic status", etc.

physical actions in DS-TTR

- exploit PDL specialisations (multimodal languages) in the literature
- introduce operators for results, opportunity, ability, "epistemic status", etc.
- specify actions/abilities/opportunities for various agents
  - operator indices representing actions (mental or physical) by various agents

physical actions in DS-TTR

- exploit PDL specialisations (multimodal languages) in the literature
- introduce operators for results, opportunity, ability, "epistemic status", etc.
- specify actions/abilities/opportunities for various agents
    - operator indices representing actions (mental or physical) by various agents
- the ability of each agent is formalised by a factual (non-modal) operator **AB**
    - stands for each agent's skills

physical actions in DS-TTR

- exploit PDL specialisations (multimodal languages) in the literature
- introduce operators for results, opportunity, ability, "epistemic status", etc.
- specify actions/abilities/opportunities for various agents
  - operator indices representing actions (mental or physical) by various agents
- the ability of each agent is formalised by a factual (non-modal) operator **AB**
  - stands for each agent's skills
- results and opportunities formalised as in PDL of the DS kind, only allow different agents to be assigned actions
  - $\langle \mathrm{do}_i(\alpha) \rangle \varphi$ means that agent *i* has the opportunity to perform the action α so that φ will result from this performance

physical actions in DS-TTR

- exploit PDL specialisations (multimodal languages) in the literature
- introduce operators for results, opportunity, ability, "epistemic status", etc.
- specify actions/abilities/opportunities for various agents
  - operator indices representing actions (mental or physical) by various agents
- the ability of each agent is formalised by a factual (non-modal) operator **AB**
  - stands for each agent's skills
- results and opportunities formalised as in PDL of the DS kind, only allow different agents to be assigned actions
  - $\langle \mathrm{do}_i(\alpha) \rangle \varphi$ means that agent i has the opportunity to perform the action α so that φ will result from this performance
- e.g., for coverbal gestures, exploit ASL formalisation of gesture formats and assign interpretations, e.g., social relation affordances instituted

DS-TTR+: adding physical actions

- introduce a language $\mathrm{L}_{DS}(TTR)$ to serve as propositional language based
  on a set TTR of TTR-type structures (tree-structured?)

DS-TTR+: adding physical actions

- introduce a language $\mathrm{L}_{DS}(TTR)$ to serve as propositional language based
  on a set TTR of TTR-type structures (tree-structured?)
  - ordered across the subtyping relation

DS-TTR+: adding physical actions

- introduce a language $\mathrm{L}_{DS}(TTR)$ to serve as propositional language based
  on a set TTR of TTR-type structures (tree-structured?)
  - ordered across the subtyping relation
- form the language $\mathrm{L}^{\mathrm{DS-TTR}}(\mathrm{L}_{DS}(TTR), \mathrm{AG}, ACT)$, based on the sets of propositions ($\mathrm{L}_{DS}(TTR)$), agents (AG), and atomic DS actions (ACT)
- plus a set of requirements ($?\phi$) for each $\mathrm{L}^{\mathrm{DS-TTR}}$ formula ϕ

DS-TTR+: adding physical actions

- introduce a language $\mathrm{L}_{DS}(TTR)$ to serve as propositional language based
  on a set TTR of TTR-type structures (tree-structured?)
  - ordered across the subtyping relation
- form the language $\mathrm{L}^{\mathrm{DS-TTR}}(\mathrm{L}_{DS}(TTR), \mathrm{AG}, ACT)$, based on the sets of propositions ($\mathrm{L}_{DS}(TTR)$), agents (AG), and atomic DS actions (ACT)
- plus a set of requirements ($?\phi$) for each $\mathrm{L}^{\mathrm{DS-TTR}}$ formula ϕ
- the language contains:
  - (at least) the propositional connectives (plus TTR operations),
    the execution and result operator $\langle \mathrm{do}\_(\_) \rangle\ \_$,
    the ability operator AB_ _,
    the action constructors:

    `confirm_` (confirmations/tests),
    `_;_` (sequential composition),
    `if _ then _ else _` (conditional composition) and
    `while_ do_` (repetitive composition)

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies
$\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies
  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - `confirm` $\varphi \in \mathcal{A}(ACT)$

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies
  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - `confirm` $\varphi \in \mathcal{A}(ACT)$
  - $\alpha_1; \alpha_2 \in \mathcal{A}(ACT)$: sequential composition

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies
  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - `confirm` $\varphi \in \mathcal{A}(ACT)$
  - $\alpha_1; \alpha_2 \in \mathcal{A}(ACT)$: sequential composition
  - IF ϕ THEN α_1 ELSE $\alpha_2 \in \mathcal{A}(ACT)$: conditional composition (`confirm/observe/test` ϕ and then perform action 1, otherwise perform action 2)

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies
  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - `confirm` $\varphi \in \mathcal{A}(ACT)$
  - $\alpha_1; \alpha_2 \in \mathcal{A}(ACT)$: sequential composition
  - IF ϕ THEN α_1 ELSE $\alpha_2 \in \mathcal{A}(ACT)$: conditional composition (`confirm/observe/test` ϕ and then perform action 1, otherwise perform action 2)
  - `while` ϕ `do` $\alpha \in \mathcal{A}(ACT)$: repetitive composition
- actions, outcomes, abilities, opportunities

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies
  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - `confirm` $\varphi \in \mathcal{A}(ACT)$
  - $\alpha_1; \alpha_2 \in \mathcal{A}(ACT)$: sequential composition
  - IF ϕ THEN α_1 ELSE $\alpha_2 \in \mathcal{A}(ACT)$: conditional composition (`confirm/observe/test` ϕ and then perform action 1, otherwise perform action 2)
  - `while` ϕ `do` $\alpha \in \mathcal{A}(ACT)$: repetitive composition
- actions, outcomes, abilities, opportunities
  - if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$, $i \in \mathrm{AG}$, $\alpha \in \mathcal{A}(ACT)$ then

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - `confirm` $\varphi \in \mathcal{A}(ACT)$
  - $\alpha_1; \alpha_2 \in \mathcal{A}(ACT)$: sequential composition
  - IF ϕ THEN α_1 ELSE $\alpha_2 \in \mathcal{A}(ACT)$: conditional composition (`confirm/observe/test` ϕ and then perform action 1, otherwise perform action 2)
  - `while` ϕ `do` $\alpha \in \mathcal{A}(ACT)$: repetitive composition
- actions, outcomes, abilities, opportunities
  - if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$, $i \in \mathrm{AG}$, $\alpha \in \mathcal{A}(ACT)$ then
  - $\mathbf{AB}_i\alpha \in \mathrm{L}^{\mathrm{DS-TTR}}$ (agent i has the skill to execute α)

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - $\texttt{confirm}\ \varphi \in \mathcal{A}(ACT)$
  - $\alpha_1; \alpha_2 \in \mathcal{A}(ACT)$: sequential composition
  - IF ϕ THEN α_1 ELSE $\alpha_2 \in \mathcal{A}(ACT)$: conditional composition (confirm/observe/test ϕ and then perform action 1, otherwise perform action 2)
  - $\texttt{while}\ \phi\ \texttt{do}\ \alpha \in \mathcal{A}(ACT)$: repetitive composition
- actions, outcomes, abilities, opportunities
  - if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$, $i \in \mathrm{AG}$, $\alpha \in \mathcal{A}(ACT)$ then
  - $\mathbf{AB}_i\alpha \in \mathrm{L}^{\mathrm{DS-TTR}}$ (agent i has the skill to execute α)
  - $\langle do_i(\alpha)\rangle\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ (agent i has the opportunity to execute α and ϕ will ensue)

the language $\mathrm{L}^{\mathrm{DS-TTR}}$

- introduce all the propositional connectives:
- $\varphi, \psi \in \mathrm{L}^{\mathrm{DS-TTR}}$ implies
  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, , \top, \bot \in \mathrm{L}^{\mathrm{DS-TTR}}$
- action constructors form the $\mathcal{A}(ACT)$ from ACT
  if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ then
  - $\texttt{confirm}\ \varphi \in \mathcal{A}(ACT)$
  - $\alpha_1; \alpha_2 \in \mathcal{A}(ACT)$: sequential composition
  - IF ϕ THEN α_1 ELSE $\alpha_2 \in \mathcal{A}(ACT)$: conditional composition (confirm/observe/test ϕ and then perform action 1, otherwise perform action 2)
  - $\texttt{while}\ \phi\ \texttt{do}\ \alpha \in \mathcal{A}(ACT)$: repetitive composition
- actions, outcomes, abilities, opportunities
  - if $\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$, $i \in \mathrm{AG}$, $\alpha \in \mathcal{A}(ACT)$ then
  - $\mathbf{AB}_i\alpha \in \mathrm{L}^{\mathrm{DS-TTR}}$ (agent i has the skill to execute α)
  - $\langle do_i(\alpha)\rangle\phi \in \mathrm{L}^{\mathrm{DS-TTR}}$ (agent i has the opportunity to execute α and ϕ will ensue)

application: proper names in DS-TTR

(1a) DS

IF	$?\,Ty(e)$
THEN	$\texttt{put}(Ty(e))$
	$\texttt{put}(Fo(john'))$
ELSE	$\texttt{abort}$

(1a) DS

IF	$?Ty(e)$
THEN	$\mathtt{put}(Ty(e))$
	$\mathtt{put}(Fo(john'))$
ELSE	$\mathtt{abort}$

(1b) DS-TTR

IF	$?Ty(e)$
THEN	$\mathtt{put}(Ty(e))$
	$\mathtt{put}(\mathrm{Fo}(\left[\ x \ : \ john\ \right]))$
ELSE	$\mathtt{abort}$

proper names in TTR

- Cooper (2018, in prep)

proper names in TTR

- Cooper (2018, in prep)
- $\lambda x : \textit{Ind} \ . \ \text{Lex}_{\textit{PropName}} \ (\text{“Eleni”}, \text{x})$

proper names in TTR

- Cooper (2018, in prep)
- $\lambda x : \mathit{Ind}\ .\ \mathrm{Lex}_{PropName}\ (\text{“Eleni”}, \mathrm{x})$
- $\begin{bmatrix} \text{sp-event} : \text{“Eleni”} \\ \text{cont} = x : \mathit{Ind} \\ \text{e: named(x, “Eleni”)} \end{bmatrix}$

$$\begin{bmatrix} \text{sp-event=e} & : & \textit{“Eleni”} \\ \text{cont=} \begin{bmatrix} \text{bg=} \begin{bmatrix} \text{x:Ind} \\ \text{e:named(x, “Eleni”)} \end{bmatrix} \\ \text{fg=}\lambda r\text{:} \begin{bmatrix} \text{x:Ind} \\ \text{e:named(x, “Eleni”)} \end{bmatrix} .\ \text{r.x} \end{bmatrix} & : & \begin{bmatrix} \text{bg:}\mathit{RecType} \\ \text{fg :(bg}\rightarrow \mathit{Ind}) \end{bmatrix} \end{bmatrix}$$

proper names in DS-TTR+

- conservative version with indexed fields

proper names in DS-TTR+

- conservative version with indexed fields
- fields are ordered pairs of (reserved) labels $\{f_1, f_2, ..., f_n\}$ and judgments

proper names in DS-TTR+

- conservative version with indexed fields
- fields are ordered pairs of (reserved) labels $\{f_1, f_2, ..., f_n\}$ and judgments

  $\langle f_1, \langle label : Type \rangle \rangle$
- record types are sets of fields

proper names in DS-TTR+

- conservative version with indexed fields
- fields are ordered pairs of (reserved) labels $\{f_1, f_2, ..., f_n\}$ and judgments

$$\langle f_1, \langle label : Type \rangle\rangle$$

- record types are sets of fields
- lexical entry for proper name:

$$
\begin{array}{ll}
\text{IF} & ?Ty(e) \wedge \langle do_H(\texttt{hear})\rangle \text{“Eleni”} \\
\text{THEN} & ?\langle do_{S\&H}(\texttt{make-go})\rangle(\langle f_1\rangle)); \ \text{put}(?\mathbf{AB}_{S\&H}(\left[\text{x}:Ind\right])), \\
 & ?\langle do_{S\&H}(\texttt{make-go})\rangle(\langle f_2\rangle); \ \text{put}?\mathbf{AB}_{S\&H}(\left[\text{s}_1:(\text{named}(\text{x}, \text{“Eleni”})\right])) \\
 & ?\langle do_{S\&H}(\texttt{make-go})\rangle; \ \text{put}?\mathbf{AB}_{S\&H}(\left[\text{s}_2:(\text{acquainted}(\text{speaker, hearer, x})\right] \\
 & ?\langle do_{S\&H}(\texttt{make-go})\rangle(\langle f_2\rangle); \ \text{put}?\mathbf{AB}_{S\&H}(\left[\text{s}_1:(\text{named}(\text{x}, \text{“Eleni”})\right])) \\
 & ?\langle do_{S\&H}(\texttt{make-go})\rangle; \ \text{put}?(\left[\text{s}_3:(\text{scared}(\text{Pipkin, x})\right])) \\
 & ?\langle do_{Pipkin}(\texttt{run})\rangle\top \wedge [do_{Pipkin}(\texttt{avoid-Eleni})]\bot \\
 & \ldots \\
\text{ELSE} & \texttt{abort}
\end{array}
$$

conclusion

- holistic view of **grammar** as guiding (production) or characterising (comprehension) behaviours
  - via distributed knowledge of sensorimotor contingencies [Noë (2004)]
  - without necessarily building internal models of the world

conclusion

- holistic view of **grammar** as guiding (production) or characterising (comprehension) behaviours
  - via distributed knowledge of sensorimotor contingencies [Noë (2004)]
  - without necessarily building internal models of the world
- incremental and predictive architecture and integration of **multimodal action/perception** within a single formal model [Eshghi et al. (2017a); Eshghi and Lemon (2014)]

conclusion

- holistic view of **grammar** as guiding (production) or characterising (comprehension) behaviours
  - via distributed knowledge of sensorimotor contingencies [Noë (2004)]
  - without necessarily building internal models of the world
- incremental and predictive architecture and integration of **multimodal action/perception** within a single formal model [Eshghi et al. (2017a); Eshghi and Lemon (2014)]
- grounded **symbolic** representations as distributively **emergent** during interactions from basic action/interaction substratum
- alternative semantic background: **Vector Space** models conceived as **exemplar** theories of conceptualisation [Sadrzadeh et al. (2018); Gregoromichelaki et al. (2019a)]
- **grammar** as coordination constraints on (**joint/collective**) **action**:

conclusion

- holistic view of **grammar** as guiding (production) or characterising (comprehension) behaviours
  - via distributed knowledge of sensorimotor contingencies [Noë (2004)]
  - without necessarily building internal models of the world
- incremental and predictive architecture and integration of **multimodal action/perception** within a single formal model [Eshghi et al. (2017a); Eshghi and Lemon (2014)]
- grounded **symbolic** representations as distributively **emergent** during interactions from basic action/interaction substratum
- alternative semantic background: **Vector Space** models conceived as **exemplar** theories of conceptualisation [Sadrzadeh et al. (2018); Gregoromichelaki et al. (2019a)]
- **grammar** as coordination constraints on (**joint/collective**) **action**:
- necessitates viewing natural language as a type of **value-driven** "skill" (know-how)

conclusion

- holistic view of **grammar** as guiding (production) or characterising (comprehension) behaviours
  - via distributed knowledge of sensorimotor contingencies [Noë (2004)]
  - without necessarily building internal models of the world
- incremental and predictive architecture and integration of **multimodal action/perception** within a single formal model [Eshghi et al. (2017a); Eshghi and Lemon (2014)]
- grounded **symbolic** representations as distributively **emergent** during interactions from basic action/interaction substratum
- alternative semantic background: **Vector Space** models conceived as **exemplar** theories of conceptualisation [Sadrzadeh et al. (2018); Gregoromichelaki et al. (2019a)]
- **grammar** as coordination constraints on (**joint/collective**) **action**:
- necessitates viewing natural language as a type of **value-driven** "skill" (know-how)
  - a skill to *act jointly* effecting changes in the sociomaterial world (including others' and own cognition)

conclusion

- holistic view of **grammar** as guiding (production) or characterising (comprehension) behaviours
  - via distributed knowledge of sensorimotor contingencies [Noë (2004)]
  - without necessarily building internal models of the world
- incremental and predictive architecture and integration of **multimodal action/perception** within a single formal model [Eshghi et al. (2017a); Eshghi and Lemon (2014)]
- grounded **symbolic** representations as distributively **emergent** during interactions from basic action/interaction substratum
- alternative semantic background: **Vector Space** models conceived as **exemplar** theories of conceptualisation [Sadrzadeh et al. (2018); Gregoromichelaki et al. (2019a)]
- **grammar** as coordination constraints on (**joint/collective**) **action**:
- necessitates viewing natural language as a type of **value-driven** "skill" (know-how)
  - a skill to *act jointly* effecting changes in the sociomaterial world (including others' and own cognition)
  - social rationality underpinned by performative *commitments* and *entitlements* rather than individual inference

conclusion

- holistic view of **grammar** as guiding (production) or characterising (comprehension) behaviours
  - via distributed knowledge of sensorimotor contingencies [Noë (2004)]
  - without necessarily building internal models of the world
- incremental and predictive architecture and integration of **multimodal action/perception** within a single formal model [Eshghi et al. (2017a); Eshghi and Lemon (2014)]
- grounded **symbolic** representations as distributively **emergent** during interactions from basic action/interaction substratum
- alternative semantic background: **Vector Space** models conceived as **exemplar** theories of conceptualisation [Sadrzadeh et al. (2018); Gregoromichelaki et al. (2019a)]
- **grammar** as coordination constraints on (**joint/collective**) **action**:
- necessitates viewing natural language as a type of **value-driven** "skill" (know-how)
  - a skill to *act jointly* effecting changes in the sociomaterial world (including others' and own cognition)
  - social rationality underpinned by performative *commitments* and *entitlements* rather than individual inference

Thank you for your attention!

Outline

References I

Alexandru Baltag and Sonja Smets. The dynamic turn in quantum logic. *Synthese*, 186(3):753–773, 2012.

Yoshua Bengio. The Consciousness Prior. *arXiv:1709.08568 [cs, stat]*, December 2019.

Mark H Bickhard. The interactivist model. *Synthese*, 166(3):547–591, 2009.

Patrick Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.

Herbert H. Clark. *Using Language*. Cambridge University Press, 1996.

Robin Cooper. Type theory and semantics in flux. In Ruth Kempson, Nicholas Asher, and Tim Fernando, editors, *Handbook of the Philosophy of Science*, volume 14: Philosophy of Linguistics, pages 271–323. North Holland, 2012.

Robin Cooper and Jonathan Ginzburg. Type theory with records for natural language semantics. *The Handbook of Contemporary Semantic Theory*, pages 375–407, 2015.

Arash Eshghi and Oliver Lemon. How domain-general can we be? Learning incremental dialogue systems without dialogue acts. In *Proceedings of Semdial 2014 (DialWatt)*, 2014.

References II

Arash Eshghi, Julian Hough, and Matthew Purver. Incremental grammar induction from child-directed dialogue utterances. In *Proceedings of the 4th Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 94–103, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Arash Eshghi, Igor Shalyminov, and Oliver Lemon. Bootstrapping incremental dialogue systems from minimal data: the generalisation power of dialogue grammars? In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017a.

Arash Eshghi, Igor Shalyminov, and Oliver Lemon. Interactional Dynamics and the Emergence of Language Games. In *Proceedings of ESSLLI 2017*, 2017b.

Finger, Marcelo and Gabbay, Dov. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 1(3):203–233, 1992.

Eleni Gregoromichelaki. Quotation in dialogue. In *The Semantics and Pragmatics of Quotation*, pages 195–255. Springer, 2018.

References III

Eleni Gregoromichelaki, Christine Howes, Arash Eshghi, Ruth Kempson, Mehrnoosh Sadrzadeh, Julian Hough, and Matthew Purver Gijs Wijnholds. Normativity, meaning plasticity, and the significance of vector space semantics. In *Proceedings of the 23rd Workshop on the Semantics and Pragmatics of Dialogue.*, 2019a.

Eleni Gregoromichelaki, Christine Howes, and Ruth Kempson. Actionism in syntax and semantics. In *Gothenburg Proceedings of Conference on Dialogue and Perception.* CLASP, 2019b.

Julian Hough. *Modelling Incremental Self-Repair Processing in Dialogue*. PhD thesis, Queen Mary University of London, 2015.

Dimitrios Kalatzis, Arash Eshghi, and Oliver Lemon. Bootstrapping incremental dialogue systems: Using linguistic knowledge to learn from minimal data. In *Proceedings of the NIPS 2016 Workshop on Learning Methods for Dialogue*, Barcelona, 2016.

Laura Kallmeyer and Rainer Osswald. Syntax-driven semantic frame composition in lexicalized tree adjoining grammars. *Journal of Language Modelling*, 1, 2013.

References IV

Laura Kallmeyer, Rainer Osswald, and Sylvain Pogodalla. Progression and Iteration in Event Semantics—An LTAG Analysis Using Hybrid Logic and Frame Semantics. In *The 11th Syntax and Semantics Conference in Paris (CSSP 2015)*, Paris, France, October 2015.

Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. *Dynamic Syntax: The Flow of Language Understanding*. Wiley-Blackwell, 2001.

Alva Noë. *Action in Perception*. MIT press, 2004.

Matthew Purver, Eleni Gregoromichelaki, Wilfried Meyer-Viol, and Ronnie Cann. Splitting the 'I's and crossing the 'You's: Context, speech acts and grammar. In P. Łupkowski and M. Purver, editors, *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue*, pages 43–50, Poznań, June 2010. Polish Society for Cognitive Science.

Matthew Purver, Arash Eshghi, and Julian Hough. Incremental semantic construction in a dialogue system. In J. Bos and S. Pulman, editors, *Proceedings of the 9th International Conference on Computational Semantics*, pages 365–369, Oxford, UK, January 2011.

References V

Matthew Purver, Mehrnoosh Sadrzadeh, Ruth Kempson, Gijs Wijnholds, and Julian Hough. Incremental composition in Distributional Semantics. *Journal of Logic, Language and Information*, forthcoming.

Erik Rietveld, Damiaan Denys, and Maarten Van Westen. Ecological-Enactive Cognition as Engaging with a Field of Relevant Affordances. In *The Oxford Handbook of 4E Cognition*, page 41. Oxford University Press, 2018.

Mehrnoosh Sadrzadeh, Matthew Purver, Julian Hough, and Ruth Kempson. Exploring semantic incrementality with dynamic syntax and vector space semantics. In *Proceedings of the 22nd Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2018 - AixDial)*, Aix-en-Provence, 2018.

Krister Segerberg. Getting started: Beginnings in the logic of action. *Studia logica*, 51(3-4):347–378, 1992.

Robert Stalnaker. *Context and Content*. Oxford University Press, 1999.

Yanchao Yu, Arash Eshghi, and Oliver Lemon. Learning how to learn: an adaptive dialogue agent for incrementally learning visually grounded word meanings. pages 10–19. doi: $10.18653/v1/W17\text{-}\subset 2802$. URL `http://arxiv.org/abs/1709.10423`.

References VI

Yanchao Yu, Arash Eshghi, and Oliver Lemon. Comparing attribute classifiers for interactive language grounding. In *Proceedings of the Fourth Workshop on Vision and Language*, pages 60–69, 2015.

Yanchao Yu, Arash Eshghi, and Oliver Lemon. Incremental Generation of Visually Grounded Language in Situated Dialogue. In *Proceedings of INLG 2016*, Los Angeles, 2016.

Yanchao Yu, Arash Eshghi, and Oliver Lemon. Learning how to learn: An adaptive dialogue agent for incrementally learning visually grounded word meanings. *arXiv:1709.10423 [cs]*, pages 10–19, 2017. doi: 10.18653/v1/W17-⊂2802.