Introduction to DS-TTR: a personal view

Eleni Gregoromichelaki
Osnabrück University and
King's College London

www.kcl.ac.uk/research/groups/ds

July 17, 2017

Introductory Motivation
  What is grammar?

TTR to formalise conceptual structure
  TTR elements adopted
  Dynamic Syntax
  DS elements adopted
  Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions
  DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2

Outline

Introductory Motivation
- What is grammar?

TTR to formalise conceptual structure
- TTR elements adopted
- Dynamic Syntax
- DS elements adopted
- Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions
- DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2

developing utterances (together) in dialogue

- real **conversation** happens bit by bit, without respecting the boundaries of sentences:
- half-starts, suggested add-ons, pauses, interruptions, corrections

(1) [Context: Friends of the Earth club meeting]
A: So what is that? Is that er... booklet or something?

B: It's a book

C: Book

B: Just ... talking about al you know alternative

D: On erm... renewable yeah

B: energy really I think......

A: Yeah [BNC:D97]

developing utterances (together) in dialogue

- real **conversation** happens bit by bit, without respecting the boundaries of sentences:
- half-starts, suggested add-ons, pauses, interruptions, corrections

(1) [Context: Friends of the Earth club meeting]
A: So what is that? Is that er... booklet or something?

B: It's a book

C: Book

B: Just ... talking about al you know alternative

D: On erm... renewable yeah

B: energy really I think......

A: Yeah [BNC:D97]

grammar and dialogue: split utterances

- ordinary conversation is highly fragmentary and incremental

grammar and dialogue: split utterances

- ordinary conversation is highly fragmentary and incremental
- various actions performed with non-sentential/propositional elements, e.g. repair (clarifications, corrections, etc)

grammar and dialogue: split utterances

- ordinary conversation is highly fragmentary and incremental
- various actions performed with non-sentential/propositional elements, e.g. repair (clarifications, corrections, etc)
- however, syntactic/semantic dependencies still constrain/direct behaviour:

grammar and dialogue: split utterances

- ordinary conversation is highly fragmentary and incremental
- various actions performed with non-sentential/propositional elements, e.g. repair (clarifications, corrections, etc)
- however, syntactic/semantic dependencies still constrain/direct behaviour:
  - **split-utterances**: syntactic/semantic dependencies hold across change of speakers:

(7) A: **Have you** read ...
B: **any** of your chapters?
cf. *I have read any of your chapters

grammar and dialogue: split utterances

- ordinary conversation is highly fragmentary and incremental
- various actions performed with non-sentential/propositional elements, e.g. repair (clarifications, corrections, etc)
- however, syntactic/semantic dependencies still constrain/direct behaviour:
  - **split-utterances**: syntactic/semantic dependencies hold across change of speakers:

(9) A: **Have you** read ...
B: **any** of your chapters?
cf. *I have read any of your chapters

(10) A: Oh, I am so sorry, did **you** burn
B: **myself**? No, its OK.
cf. # Oh, I am so sorry, did you burn myself?

NL grammars

⇒ the "grammar", as a holistic model, needs to be able to express

(a) the **incremental** licensing and interpretation of NL strings

NL grammars

⇒ the "grammar", as a holistic model, needs to be able to express

(a) the **incremental** licensing and interpretation of NL strings
(b) the **context shift** (e.g. change of speaker-roles) within a single clause,

NL grammars

⇒ the "grammar", as a holistic model, needs to be able to express

(a) the **incremental** licensing and interpretation of NL strings
(b) the **context shift** (e.g. change of speaker-roles) within a single clause,
(c) while still implementing traditional **syntactic/syntactic constraints**:

(13) a. John likes himself vs. *him

b. John likes everyone [Mary does] vs.
*John likes everyone [Mary admires the man [who does]]

grammar framework: DS-TTR

- conversational data and the nature of **grammar**: the view from DS-TTR
  - no separate syntactic level of representation:
    - no syntactic categories for strings of words;
    - no phrase-structure rules;
    - no "constructions"
  - grammatical ontology of **processes** (rather than *representations*)
    - incrementality, prediction, and underspecification as properties of the grammar ("syntax")

Outline

Introductory Motivation
- What is grammar?

TTR to formalise conceptual structure
- TTR elements adopted
- Dynamic Syntax
- DS elements adopted
- Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions
- DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2

Type Theory

- Martin-Löf Type Theory
- **objects**/entities belong to types
- **propositions** are regarded as types of proofs ("propositions as types" principle)
- **proofs** are objects
  - e.g. the proofs of *there is a prime number between 212 and 222* are the prime numbers between 212 and 222
  - Ranta (1984): a proof of

    (14) John hugged Mary.

    is some event during which John hugged Mary

Type Theory With Records

- type theoretical **judgements**:
  - a : T ("object *a* is of type *T*")

Type Theory With Records

- type theoretical **judgements**:
  - a : T ("object *a* is of type *T*")
- **types** in TTR: not atomic, but complex

Type Theory With Records

- type theoretical **judgements**:
  - a : T (“object *a* is of type *T*”)
- **types** in TTR: not atomic, but complex
- **records** are sequences of label/value pairs:

$$\begin{bmatrix} l_1 & = & v_1 \\ l_2 & = & v_2 \\ l_3 & = & v_3 \end{bmatrix}$$

Type Theory With Records

- type theoretical **judgements**:
  - a : T (“object a is of type T”)
- **types** in TTR: not atomic, but complex
- **records** are sequences of label/value pairs:

$$\begin{bmatrix} l_1 & = & v_1 \\ l_2 & = & v_2 \\ l_3 & = & v_3 \end{bmatrix}$$

- **record types** are sequences of label/type pairs:

$$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2 \\ l_3 & : & T_3 \end{bmatrix}$$

TTR (Type Theory with Records) – conceptual structure

- *records* model complex entities,
  - e.g., events (including contexts)

TTR (Type Theory with Records) – conceptual structure

- *records* model complex entities,
  - e.g., events (including contexts)
- *record types* model **categorisations** of events/individuals

TTR (Type Theory with Records) – conceptual structure

- *records* model complex entities,
  - e.g., events (including contexts)
- *record types* model **categorisations** of events/individuals
  - classification of a situation to be of a certain type with potential for further elaboration

TTR: truth

- **records** are sequences of label/value pairs:

$$\begin{bmatrix} l_1 & = & v_1 \\ l_2 & = & v_2 \\ l_3 & = & v_3 \end{bmatrix}$$

TTR: truth

- **records** are sequences of label/value pairs:

$$\begin{bmatrix} l_1 & = & v_1 \\ l_2 & = & v_2 \\ l_3 & = & v_3 \end{bmatrix}$$

- **record types** are **true** iff they are *inhabited/witnessed*

TTR: expressivity

- types can be **dependent** on earlier (higher-up) types:

$$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2(l_1) \\ l_3 & : & T_3(l_1, l_2) \end{bmatrix}$$

TTR: expressivity

- types can be **dependent** on earlier (higher-up) types:

$$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2(l_1) \\ l_3 & : & T_3(l_1, l_2) \end{bmatrix}$$

- **recursivity**: we can have *nested* records and record types:

$$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & \begin{bmatrix} l'_1 & : & T'_1 \\ l'_2 & : & T'_2 \end{bmatrix} \\ l_3 & : & T_3(l_1, l_2.l'_1, l_2.l'_2) \end{bmatrix}$$

TTR: expressivity

- we have **functional** record types:

$$\lambda \; \boxed{r : \begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2 \end{bmatrix}} \left(\begin{bmatrix} l_3 & : & T_3 \\ l_4 & : & T_4(r.l_1, r.l_2) \end{bmatrix} \right)$$

TTR (Type Theory with Records) – appealing features

- synthesis of ideas of **frame semantics** and **Montague Grammar**
  - invoked frames as background knowledge
  - integrates standard formal semantic tools like the lambda calculus
- (potentially) constructivist: meanings as programs, as proofs (potentially, actions all the way down)

TTR (Type Theory with Records) – appealing features

- synthesis of ideas of **frame semantics** and **Montague Grammar**
  - invoked frames as background knowledge
  - integrates standard formal semantic tools like the lambda calculus
- (potentially) constructivist: meanings as programs, as proofs (potentially, actions all the way down)
- TTRs subtype relation allows complete semantics extraction for any partial tree, and incremental further specification as parsing proceeds

TTR (Type Theory with Records) – appealing features

- synthesis of ideas of **frame semantics** and **Montague Grammar**
  - invoked frames as background knowledge
  - integrates standard formal semantic tools like the lambda calculus
- (potentially) constructivist: meanings as programs, as proofs (potentially, actions all the way down)
- TTRs subtype relation allows complete semantics extraction for any partial tree, and incremental further specification as parsing proceeds
- **sublexical** conceptual **structure**
  - distributed representations
  - **atomic concepts** correspond to patterns of activation (not single neurons)

  $\Rightarrow$ complex record structures for single concepts (not atoms as in standard logics)

TTR: perceptual/action grounding

Probabilistic Type Theory with Records (probTTR)

- types are grounded in classifiers
- interface with perception: NL semantics + perception expressible in the same formalism (TTR)

TTR: perceptual/action grounding

Probabilistic Type Theory with Records (probTTR)

- types are grounded in classifiers
- interface with perception: NL semantics + perception expressible in the same formalism (TTR)

Connection with action (Cooper(2014; fthcmg))

- judgements as act(ion)s
- modelling of acts of creation of witnesses of types

TTR: perceptual/action grounding

Probabilistic Type Theory with Records (probTTR)

- types are grounded in classifiers
- interface with perception: NL semantics + perception expressible in the same formalism (TTR)

Connection with action (Cooper(2014; fthcmg))

- judgements as act(ion)s
- modelling of acts of creation of witnesses of types

However, TTR is static

Arrival: holistic logograms

Arrival: holistic logograms

Arrival: holistic logograms

Arrival: holistic logograms

TTR - introducing dynamics: desiderata for incrementalising TTR

- dynamic incremental conceptualisation implemented in DS-TTR to provide actions to:
  - modify, delete, add fields while the rest stay the same (lexical semantics)
  - compute similarity between concepts (record types((e.g. metaphor, quotation)
  - check subsumption incrementally (generation, repair, Hough)
  - extract all available semantic information incrementally (Hough)
  - encompass multimodal aspects of processing (e.g. gesture, affect, Eshghi)
  - model defeasible inference rules as functions from objects of a type to another type (e.g. associative view of reasoning) (Ellen)
  - model frequency and context effects as probabilistic type assignments

grammar as action: desiderata for DS-TTR

- DS-TTR:
  - conceptualises grammar as a set of actions
  - no syntactic level of representation for words
  - grammatical/lexical actions build/linearise (ad hoc) conceptual structure
  - procedural definitions: constraints on *how* not what
  - single level of operations integrating all aspects of context-dependency

- Incrementally building/linearising conceptual structure

- Nodes decorated with $Ty()$ type and $Fo()$ formula labels

`John likes Mary`:

$Ty(t)$, $Fo(like(John, Mary))$

- $Ty(e)$, $Fo(John)$
- $Ty(e \rightarrow t)$, $Fo(\lambda x.like(x, Mary))$
  - $Ty(e)$, $Fo(Mary)$
  - $Ty(e \rightarrow (e \rightarrow t))$, $Fo(\lambda y \lambda x.like(x, y))$

Dynamic Syntax: conceptual structure

- Incrementally building/linearising conceptual structure

- Nodes decorated with $Ty()$ type and $Fo()$ formula labels

`John likes Mary`:

- $Ty(t)$, $Fo(like(John, Mary))$
  - $Ty(e)$, $Fo(John)$
  - $Ty(e \rightarrow t)$, $Fo(\lambda x.like(x, Mary))$
    - $Ty(e)$, $Fo(Mary)$
    - $Ty(e \rightarrow (e \rightarrow t))$, $Fo(\lambda y \lambda x.like(x, y))$

- Daughter order does not reflect sentence order
- Nodes interpretable as terms in the λ-calculus

Multilevel underspecification + update (discontinuity)

- Pronouns, elliptical elements, tree relations can be introduced as **underspecified** an in need of update:

Who did Mary upset?

Starting with an **unfixed** (underspecified) node

$Tn(0), ... ?Ty(t),$

$\langle\uparrow_*\rangle Tn(0)$
$?Ty(e), ?\exists x Tn(x), \diamondsuit$

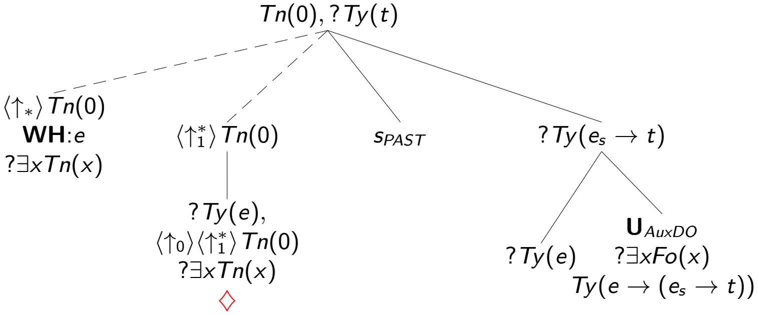
Multilevel underspecification + incremental update

- Processing `Who` *did Mary upset*

$$\begin{array}{l} Tn(0), \ldots ?Ty(t), \Diamond \\ \quad\vdots \\ \langle\uparrow_*\rangle Tn(0) \\ \mathbf{WH} : e, ?\exists x Tn(x) \end{array}$$

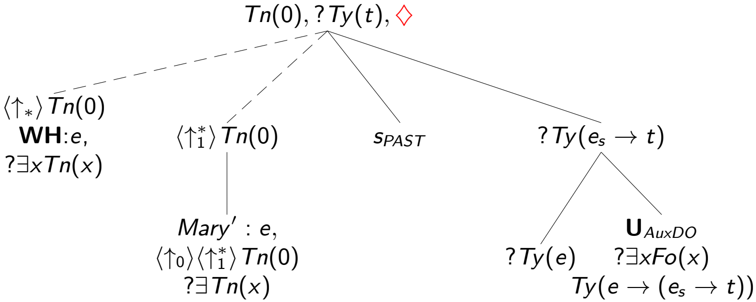
Multilevel underspecification + incremental update

- Processing `Who did` *Mary upset*
- Auxiliary projects subject-predicate template

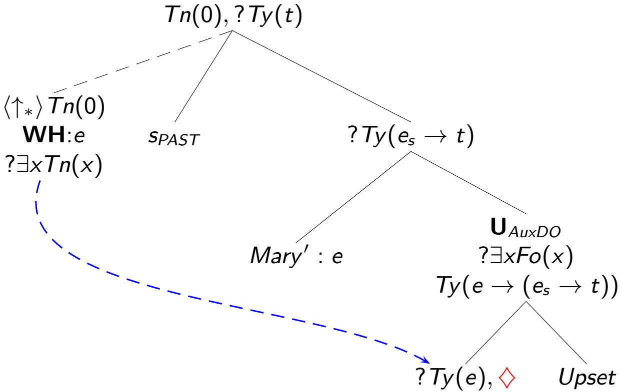


Multilevel underspecification + incremental update

Processing `Who did Mary`*upset*

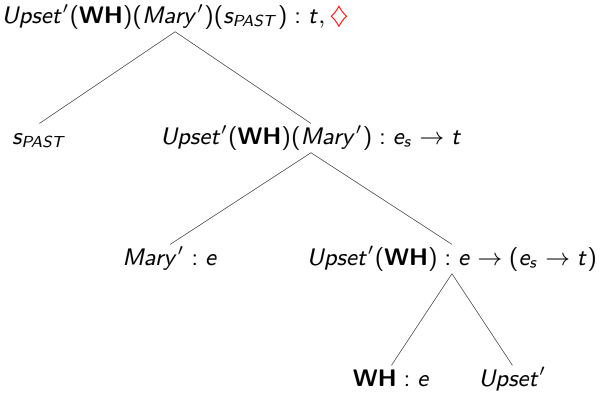


Processing `Who did Mary upset`



Structure Underspecification + update

Completing the processing of `Who did Mary upset`



Grammaticalised underspecification

Processing `John likes himself`

$?Ty(t), \diamondsuit$

Grammaticalised underspecification

Processing `John likes himself`

$$?Ty(t)$$

$$\Diamond, \; ?Ty(e) \qquad\qquad ?Ty(e \rightarrow t)$$

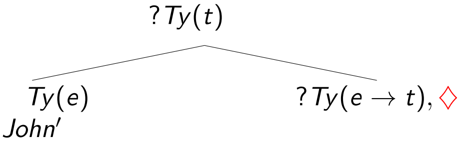
Grammaticalised underspecification

Processing John `likes himself`

$?Ty(t)$

$\Diamond$, $Ty(e)$, $John'$ $\qquad$ $?Ty(e \rightarrow t)$

Processing `John likes himself`



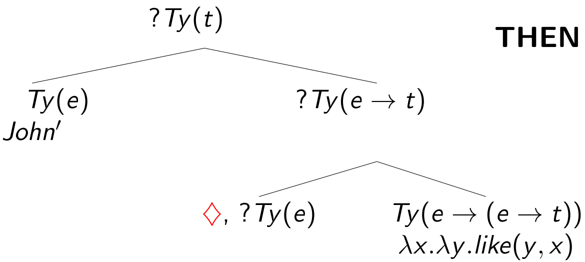
Processing `John likes himself`

$$?Ty(t)$$

$Ty(e)$ / $John'$ $\qquad$ $?Ty(e \to t)$

$\diamondsuit, ?Ty(e)$ $\qquad$ $Ty(e \to (e \to t))$ / $\lambda x.\lambda y.like(y, x)$

Processing John likes himself

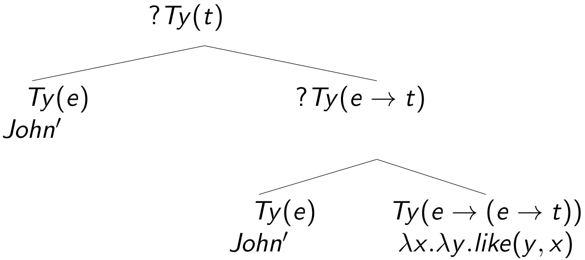


`himself`

IF	?$Ty(e)$
	$\uparrow_0\uparrow_{1*}\downarrow_0 Fo(X), Ty(e)$
THEN	put($Fo(X)$);
	put($Ty(e)$)

Grammaticalised underspecification

Processing `John likes himself`



Processing `John likes himself`

- $?Ty(t)$
  - $Ty(e)$, $John'$
  - $Ty(e \rightarrow t), \diamondsuit$
    - $Ty(e)$, $John'$
    - $Ty(e \rightarrow (e \rightarrow t))$, $\lambda x.\lambda y.like(y, x)$

Processing `John likes himself`

- $?Ty(t), \textcolor{red}{\Diamond}$
  - $Ty(e)$, $John'$
  - $Ty(e \rightarrow t)$, $\lambda y.like(y, John')$
    - $Ty(e)$, $John'$
    - $Ty(e \rightarrow (e \rightarrow t))$, $\lambda x.\lambda y.like(y, x)$

Grammaticalised underspecification

Processing `John likes himself`

$$like(John', John')$$
$$Ty(t)$$

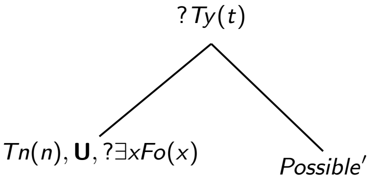
$$Ty(e)$$
$$John'$$

$$Ty(e \rightarrow t)$$
$$\lambda y.like(y, John')$$

$$Ty(e)$$
$$John'$$

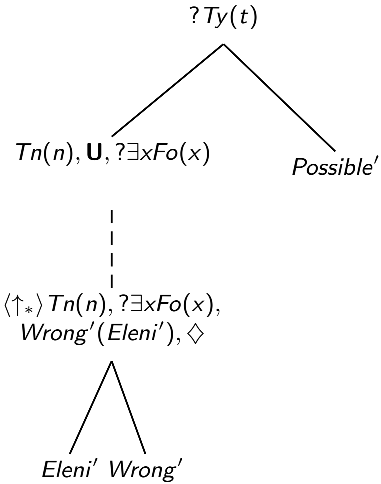
$$Ty(e \rightarrow (e \rightarrow t))$$
$$\lambda x.\lambda y.like(y, x)$$

(13c) It's possible ... I am wrong

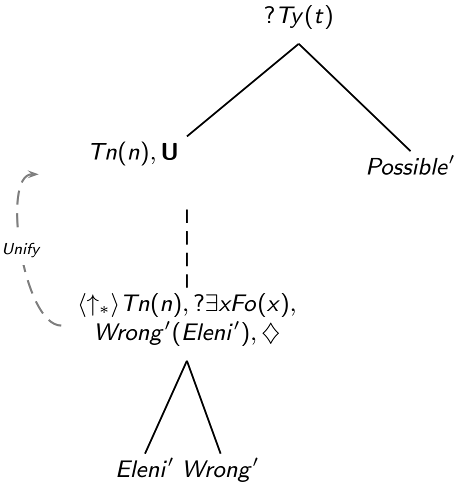


Grammaticalised cataphora - expletives

(13c) It's possible ... I am wrong



(13c) It's possible ... I am wrong



Grammaticalised cataphora - expletives

(13c) It's possible ... I am wrong

$$?Ty(t)$$

$$Tn(n), \mathbf{U}$$
$$\cancel{?\exists x Fo(x)}, Wrong'(Eleni')$$

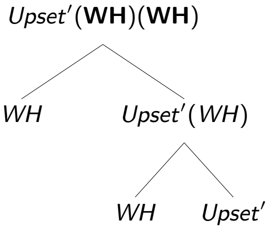
$$Possible'$$

Re-running actions – short answers

e.g. Who upset himself? John did.

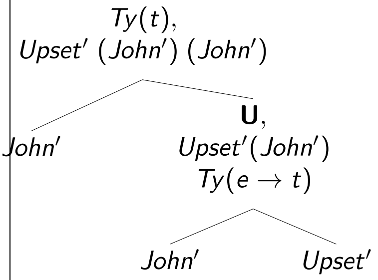
Context

TREE:



ACTIONS: $\langle \ldots$ *upset*, *himself*, COMPLETION, EVALUATION $\rangle$

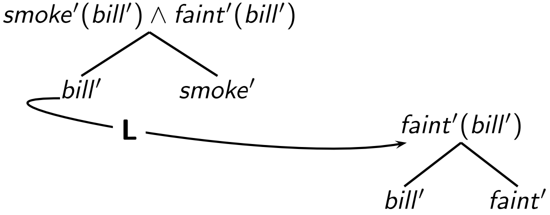
Complete parse tree



LINKed Trees - island restrictions

- **Relative clauses**: pairs of LINKed trees evaluated as conjunction

  e.g. Bill, who fainted, smokes.



- Also used for apposition, clarification and confirmation, implicatures ...

Re-running actions – ACE

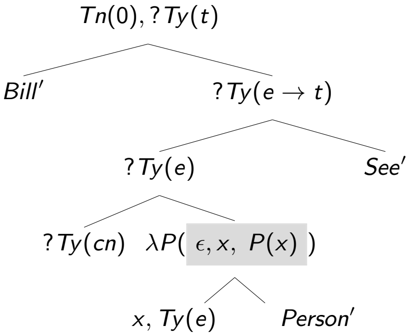
- Antecedent Contained Ellipsis

e.g. Bill saw someone [that John did]

Re-running actions – ACE

- Antecedent Contained Ellipsis

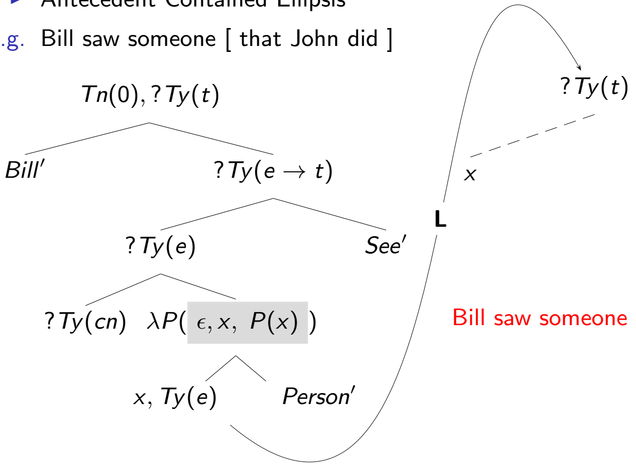
e.g. Bill saw someone [that John did]



Bill saw someone

- Antecedent Contained Ellipsis

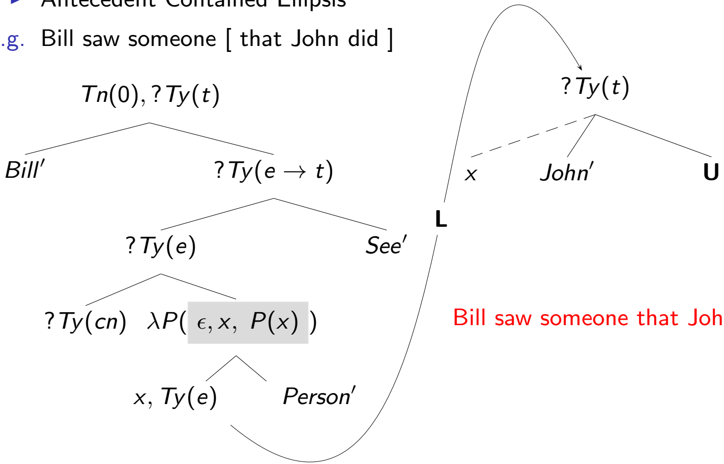
e.g. Bill saw someone [that John did]



Bill saw someone that

- Antecedent Contained Ellipsis

e.g. Bill saw someone [that John did]

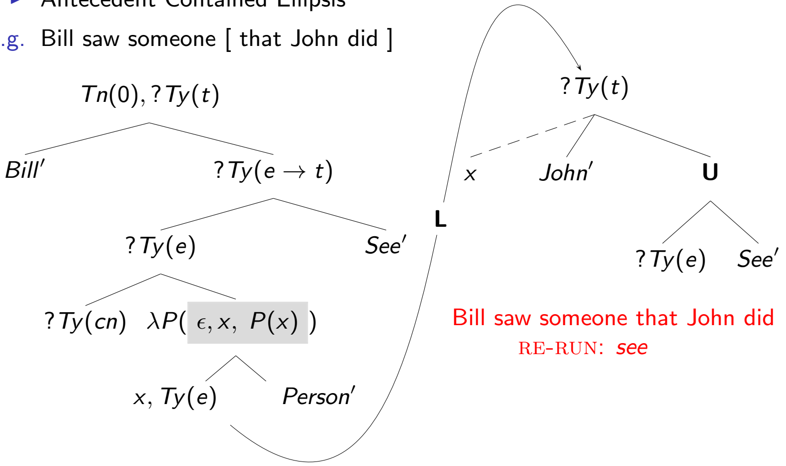


Bill saw someone that John did

Re-running actions – ACE

- Antecedent Contained Ellipsis

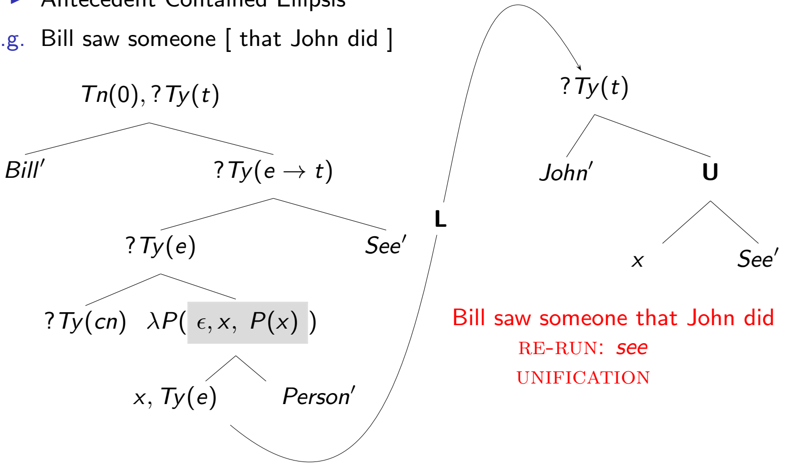
e.g. Bill saw someone [that John did]



Re-running actions – ACE

- Antecedent Contained Ellipsis

e.g. Bill saw someone [that John did]



Bill saw someone that John did

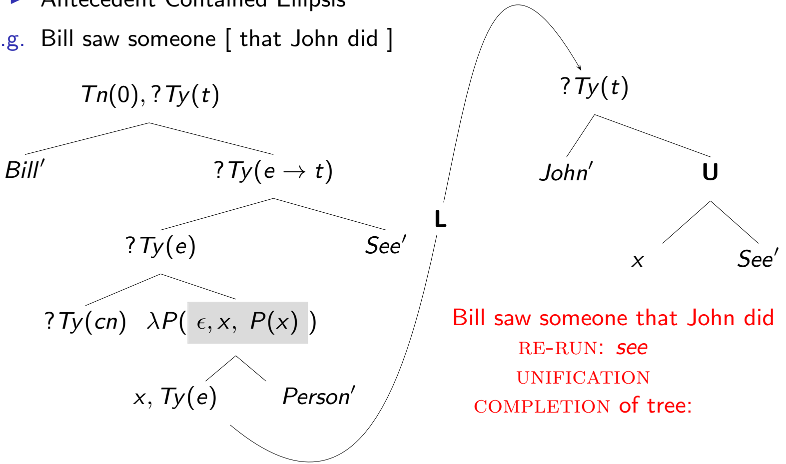
RE-RUN: *see*

UNIFICATION

Re-running actions – ACE

- Antecedent Contained Ellipsis

e.g. Bill saw someone [that John did]



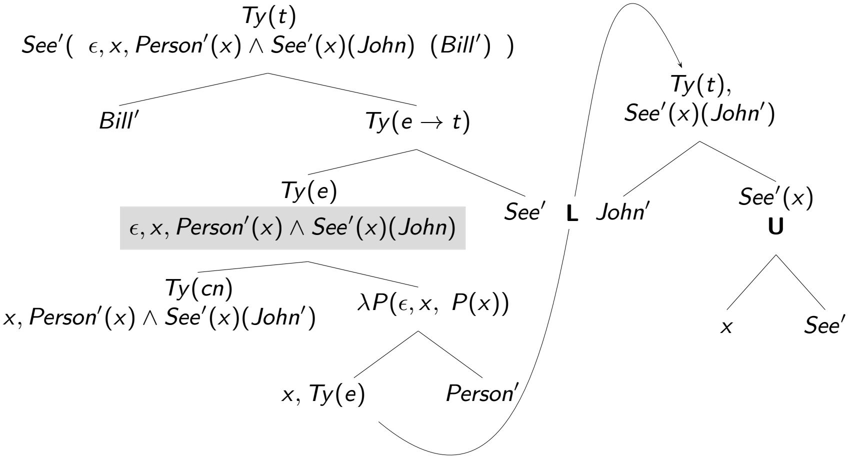
Bill saw someone that John did

RE-RUN: *see*

UNIFICATION

COMPLETION of tree:

e.g. Bill saw someone that John did



Generation

- Speakers go through the same actions, except they also have a somewhat richer goal tree.
- Each word licensed must update partial tree towards the goal tree via *subsumption* constraint

* Generating `Someone fainted`

GOAL TREE

$$Ty(t), \diamondsuit$$
$$faint(\epsilon, person(x))$$

$$Ty(e), \quad Ty(e \rightarrow t)$$
$$\epsilon, x, person(x) \quad \lambda y.faint(y)$$

TEST TREE

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

Generation

- Speakers go through the same actions, except they also have a somewhat richer goal tree.
- Each word licensed must update partial tree towards the goal tree via *subsumption* constraint

* Generating `Someone fainted`

GOAL TREE

$$Ty(t), \diamondsuit$$
$$faint(\epsilon, person(x))$$

$$Ty(e), \quad \epsilon, x, person(x) \qquad\qquad Ty(e \rightarrow t) \quad \lambda y.faint(y)$$

TEST TREE

$$?Ty(t),$$

$$\diamondsuit, ?Ty(e) \qquad\qquad ?Ty(e \rightarrow t)$$

- Speakers go through the same actions, except they also have a somewhat richer goal tree.
- Each word licensed must update partial tree towards the goal tree via *subsumption* constraint

* Generating `Someone fainted`

GOAL TREE

$$Ty(t), \diamondsuit$$
$$faint(\epsilon, person(x))$$

$Ty(e)$, $\epsilon, x, person(x)$ — $Ty(e \rightarrow t)$ $\lambda y.faint(y)$

TEST TREE

$$?Ty(t),$$

$\diamondsuit$, $Ty(e)$ $\epsilon, x, person(x)$ — $?Ty(e \rightarrow t)$

Gen: "Someone

Generation

- Speakers go through the same actions, except they also have a somewhat richer goal tree.
- Each word licensed must update partial tree towards the goal tree via *subsumption* constraint

* Generating `Someone fainted`

GOAL TREE

$Ty(t), \diamondsuit$
$faint(\epsilon, person(x))$

$Ty(e),$
$\epsilon, x, person(x)$

$Ty(e \rightarrow t)$
$\lambda y.faint(y)$

TEST TREE

$?Ty(t),$

$Ty(e)$
$\epsilon, x, person(x)$

$?Ty(e \rightarrow t), \diamondsuit$

Gen: "Someone

Generation

- Speakers go through the same actions, except they also have a somewhat richer goal tree.
- Each word licensed must update partial tree towards the goal tree via *subsumption* constraint

* Generating `Someone fainted`

GOAL TREE

$$Ty(t), \diamondsuit \quad faint(\epsilon, person(x))$$

$$Ty(e), \; \epsilon, x, person(x) \qquad Ty(e \to t) \; \lambda y.faint(y)$$

TEST TREE

$$?Ty(t),$$

$$Ty(e) \; \epsilon, x, person(x) \qquad ?Ty(e \to t), \diamondsuit \; \lambda y.faint(y)$$

Gen: "Someone fainted"

Generation

- Speakers go through the same actions, except they also have a somewhat richer goal tree.
- Each word licensed must update partial tree towards the goal tree via *subsumption* constraint

* Generating `Someone fainted`

GOAL TREE

$Ty(t), \diamondsuit$
$faint(\epsilon, person(x))$

$Ty(e),$
$\epsilon, x, person(x)$

$Ty(e \rightarrow t)$
$\lambda y.faint(y)$

TEST TREE

$Ty(t), \diamondsuit$
$faint(\epsilon, x, person(x))$

$Ty(e)$
$\epsilon, x, person(x)$

$Ty(e \rightarrow t)$
$\lambda y.faint(y)$

Gen: "Someone fainted"

Alignment: rerunning of actions induces parallellism

- Using **actions** from context – *sloppy readings*:

(1) A: John upset his mother.
B: Harry too.

(2) A: The man [who arrested John] failed to read him his rights.
B: The man who arrested Tom did too.

Alignment: rerunning of actions induces parallellism

- Using **actions** from context – *sloppy readings*:

(1) A: John upset his mother.
B: Harry too.

(2) A: The man [who arrested John] failed to read him his rights.
B: The man who arrested Tom did too.

- Also more general parallellism effects, e.g. scope:

(4) A: A consultant interviewed every patient.
B: A junior doctor too.

Outline

Introductory Motivation
- What is grammar?

TTR to formalise conceptual structure
- TTR elements adopted
- Dynamic Syntax
- DS elements adopted
- Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions
- DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2

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

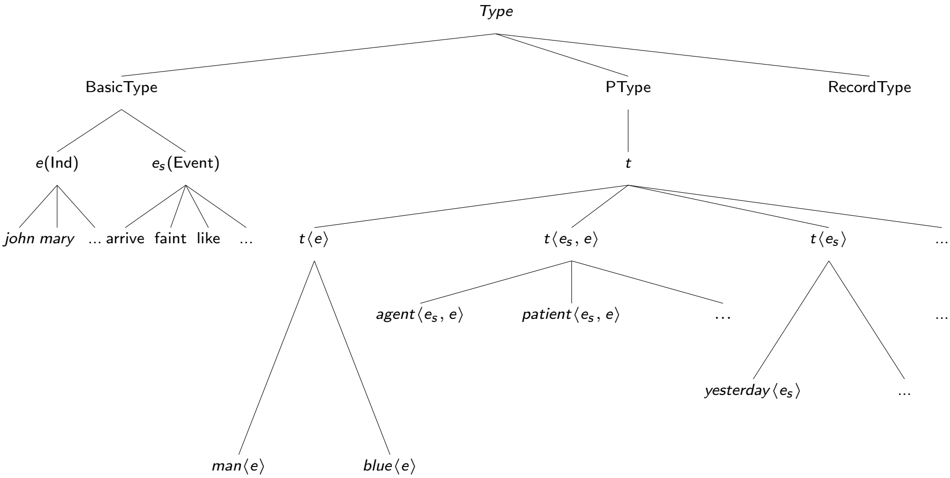
- from strings to conceptual structure (TTR) or vice-versa
- *John arrived.*

`John arrived`
$\longmapsto$

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

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

DS-TTR: Types (simplified)



DS-TTR: actions from DS +

- parsing/linearising (syntactic/lexical):
  - `go` [treenode]
  - `make`[treenode]
  - `put`[field/value/label/...]
  - `IF` [value] `THEN` [actions], `ELSE` [...]
  - `run`(list⟨actions⟩[...])

DS-TTR: actions from DS +

- parsing/linearising (syntactic/lexical):
  `go` [treenode]
  `make`[treenode]
  `put`[field/value/label/. . .]
  `IF` [value] `THEN` [actions], `ELSE` [. . .]
  `run`(list⟨actions⟩[. . .])

- manipulating complex type articulation
  `add`[fields]
  `remove`[fields]
  `test`[subtyping relation]
  . . .

DS-TTR: actions from DS +

- parsing/linearising (syntactic/lexical):
  `go` [treenode]
  `make`[treenode]
  `put`[field/value/label/…]
  `IF` [value] `THEN` [actions], `ELSE` […]
  `run`(list⟨actions⟩[…])

- manipulating complex type articulation
  `add`[fields]
  `remove`[fields]
  `test`[subtyping relation]
  …

- exploring the context:
  `freshput`[variable/metavariable]
  `find`[value/label/…],
  `substitute`[values for metavariables]
  …

incremental construction

$[\text{START}] \xmapsto{\ldots \text{ PREDICTION}}$ $\Diamond, ?Ty(t)$

incremental construction

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

$$?Ty(t)$$

$$\Diamond, ?Ty(e) \qquad\qquad ?Ty(e \rightarrow t)$$

incremental construction

$\overset{\texttt{John}}{\longmapsto}$

$$?Ty(t)$$

$$\diamondsuit, ?Ty(e) \qquad\qquad ?Ty(e \rightarrow t)$$

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

incremental construction

$\texttt{John}, \ldots, \text{POINTER-MOVEMENT} \atop \longmapsto$

$$?Ty(t)$$

$$Ty(e), \left[\ x_{=john} \ : \ e \ \right] \qquad ?Ty(e \rightarrow t), \diamondsuit$$

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

incremental construction

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

$$?Ty(t)$$

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

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

incremental construction

$\ldots[\text{TENSE}, \ \ldots], \underset{\longrightarrow}{} \text{COMPLETION}$

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

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

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

underspecification: structural

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

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

underspecification: structural

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

`Mary`

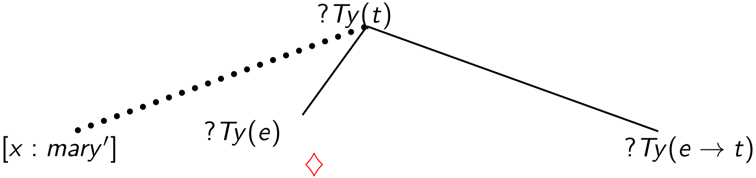
$?Ty(t)$

$[x : mary'], \diamondsuit$

underspecification: structural

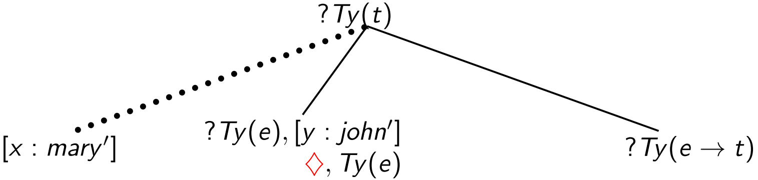
- Processing **non-contiguous dependencies**
  - e.g. *Mary, John upset*

Mary



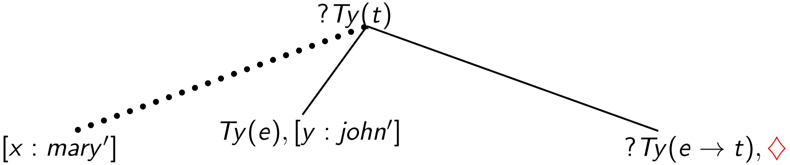
- Processing **non-contiguous dependencies**
  - e.g. *Mary, John upset*

`Mary, John`



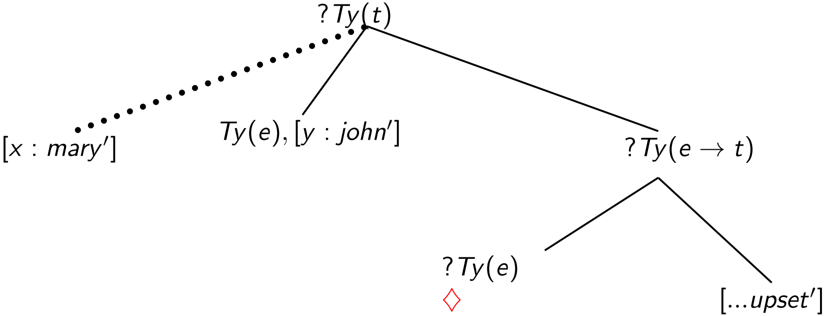
- Processing **non-contiguous dependencies**
  - e.g. *Mary, John upset*

`Mary, John`



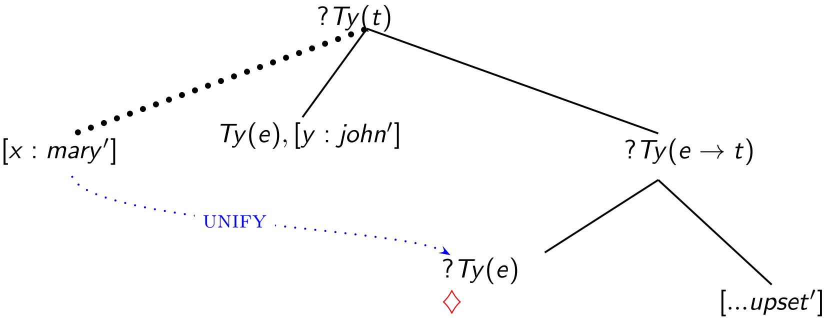
- Processing **non-contiguous dependencies**
  - e.g. *Mary, John upset*

Mary, John upset



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

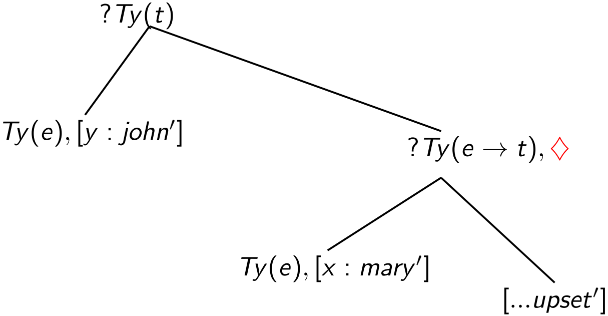
`Mary, John upset`



underspecification: structural

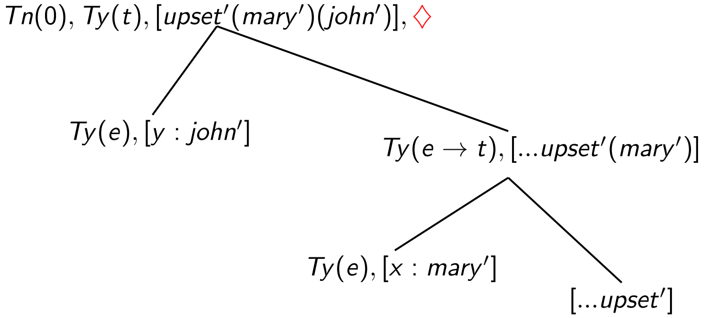
- Processing **non-contiguous dependencies**
  - e.g. *Mary, John upset*

`Mary`, `John upset`



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

`Mary, John upset`



utterance micro-events

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

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

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

including contextual parameters

`John arrived` $\longmapsto$

$$\diamondsuit,\ Ty(t),\ \left[\begin{array}{ll} \text{CONTEXT}: & \left[\begin{array}{lll} a & : & participantA \\ b & : & participantB \\ \ldots & : & \ldots \\ u_1 & : & utt-event \\ s_{s1} & : & spkr(u_1, a) \\ s_{a1} & : & addr(u_1, b) \\ u_2 & : & utt-event \\ s_{s2} & : & spkr(u_2, a) \\ s_{a2} & : & addr(u_2, b) \\ \ldots & : & \ldots \end{array}\right] \\ \text{CONTENT}: & \left[\begin{array}{lll} x & : & john \\ p & : & arrive(x) \end{array}\right] \end{array}\right]$$

$$Ty(e),\ \left[\begin{array}{ll} \text{CONTEXT}: & \left[\begin{array}{lll} u_1 & : & utt-event \\ \ldots & : & \ldots \\ s_{s1} & : & spkr(u_1, a) \\ \ldots & : & \ldots \end{array}\right] \\ \text{CONTENT}: & \left[\begin{array}{lll} x & : & john \end{array}\right] \end{array}\right]$$

$$Ty(e \rightarrow t),\ \left[\begin{array}{ll} \text{CONTEXT}: & \left[\begin{array}{lll} u_2 & : & utt-event \\ \ldots & : & \ldots \\ s_{s2} & : & spkr(u_2, a) \\ \ldots & : & \ldots \end{array}\right] \\ \text{CONTENT}: & \lambda[x].\left[\begin{array}{lll} p & : & arrive(x) \end{array}\right] \end{array}\right]$$

A: John ...

B: arrives $\mapsto$

$$\left[\begin{array}{ll} \text{CONTEXT}: & \\ u_{1\oplus 2} & : utt-event \\ s_1 & : \boxed{spkr(A, u_1)} \\ s_2 & : \boxed{spkr(B, u_2)} \\ \ldots & \end{array}\right]$$

$$\left[\begin{array}{l} \text{CONTENT}: \\ Ty(t), \\ \left[\begin{array}{ll} s_{=now} & : e_s \\ x_{=john} & : e \\ p_{=arrive(s,x)} & : t \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{CONTEXT}: & \\ u_1 & : utt-event \\ s_1 & : \boxed{spkr(A, u_1)} \\ \ldots & \\ \mathbf{G} & : l-use \end{array}\right]$$

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

$$\left[\begin{array}{ll} \text{CONTEXT}: & \\ u_2 & : utt-event \\ s_2 & : \boxed{spkr(B, u_2)} \\ \ldots & \\ \mathbf{G} & : l-use \end{array}\right]$$

$$\left[\begin{array}{l} \text{CONTENT}: \\ Ty(e \rightarrow t), \\ \lambda r : \left[\begin{array}{ll} x & : e \end{array}\right] . \left[\begin{array}{ll} s_{=now} & : e_s \\ x & : e \\ p_{=arrive(s,x)} & : t \end{array}\right] \end{array}\right]$$

I:

$$\begin{array}{ll} \text{IF} & ?Ty(e), \left[\ \text{CONTEXT} \ : \ \left[\ s_s \ : \ spkr(\mathbf{u}, \mathbf{x})\ \right]\ \right] \\ \text{THEN} & \mathtt{put}(Ty(e)) \\ & \mathtt{put}((\mathbf{x})) \\ \text{ELSE} & \mathtt{abort} \end{array}$$

myself:

$$\begin{array}{ll} \text{IF} & ?Ty(e), \left[\ \text{CONTEXT} \ : \ \left[\ s_s \ : \ spkr(\mathbf{u}, \mathbf{x})\ \right]\ \right], \\ & \uparrow_0\uparrow_{1*}\downarrow_0 Fo(\mathbf{x}) \\ \text{THEN} & \mathtt{put}(Ty(e)) \\ & \mathtt{put}(Fo(\mathbf{x})) \\ \text{ELSE} & \mathtt{abort} \end{array}$$

I:

IF	$?Ty(e), \left[\ \text{CONTEXT} : \left[\ s_s : spkr(\mathbf{u}, \mathbf{x})\ \right]\ \right]$
THEN	$\mathtt{put}(Ty(e))$
	$\mathtt{put}((\mathbf{x}))$
ELSE	$\mathtt{abort}$

myself:

IF	$?Ty(e), \left[\ \text{CONTEXT} : \left[\ s_s : spkr(\mathbf{u}, \mathbf{x})\ \right]\ \right],$
	$\uparrow_0\uparrow_{1*}\downarrow_0 Fo(\mathbf{x})$
THEN	$\mathtt{put}(Ty(e))$
	$\mathtt{put}(Fo(\mathbf{x}))$
ELSE	$\mathtt{abort}$

A: Did you burn ...

B: myself?

split utterances with indexicals

Eleni: I burnt . . .
Bill: yourself! (as usual)

split utterances with indexicals

Eleni: I

$$\left[\ \text{CXT} : \left[\ s_1 \ : \ \mathit{spk}(\mathit{Eleni}, u_1)\ \right]\ \right]$$

split utterances with indexicals

Eleni: I

I:

IF	$?Ty(e), \left[\text{CONTEXT} : \left[s_s : spkr(\mathbf{u}, \mathbf{x}) \right] \right]$
THEN	$\mathtt{put}(Ty(e))$
	$\mathtt{put}((\mathbf{x}))$
ELSE	$\mathtt{abort}$

$$\left[\begin{array}{l} \text{CXT} : \left[s_1 : spk(Eleni, u_1) \right] \\ \text{CNT} : \left[x_{=Eleni} : e \right] \end{array} \right]$$

split utterances with indexicals

Eleni: I burnt ...

$$\left[\begin{array}{ll} \text{CXT}: & \left[\begin{array}{lll} s_1 & : & spk(Eleni, u_1) \end{array}\right] \\ \text{CNT}: & \left[\begin{array}{lll} x_{=Eleni} & : & e \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{CXT}: & \left[\begin{array}{lll} s_2 & : & spk(Eleni, u_2) \end{array}\right] \\ & \\ & \end{array}\right]$$

split utterances with indexicals

Eleni: I burnt . . .

$$\left[\begin{array}{l} \text{CXT} : \left[\begin{array}{lll} s_1 & : & spk(Eleni, u_1) \end{array}\right] \\ \text{CNT} : \left[\begin{array}{lll} x_{=Eleni} & : & e \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{l} \text{CXT} : \left[\begin{array}{lll} s_2 & : & spk(Eleni, u_2) \end{array}\right] \\ \text{CNT} : \lambda \left[\begin{array}{lll} x & : & e \\ y & : & e \end{array}\right] . \left[\begin{array}{lll} x & : & e \\ y & : & e \\ p & : & burn'(y, x) \end{array}\right] \end{array}\right]$$

split utterances with indexicals

Eleni: I burnt . . .
Bill: yourself! (as usual)

yourself:

IF	$?Ty(e), \left[\ \textsc{context} : \left[\ s_s : addr(\mathbf{u}, \mathbf{x})\ \right]\ \right],$ $\uparrow_0\uparrow_{1*}\downarrow_0 Fo(\mathbf{x})$
THEN	$\mathtt{put}(Ty(e))$ $\mathtt{put}(Fo(\mathbf{x}))$
ELSE	$\mathtt{abort}$

$$\left[\begin{array}{l} \mathrm{CXT} : \left[\ s_1 : spk(Eleni, u_1)\ \right] \\ \mathrm{CNT} : \left[\ x_{=Eleni} : e\ \right] \end{array}\right]$$

$$\left[\ \mathrm{CXT} : \left[\begin{array}{lcl} s_3 & : & spk(Bill, u_3) \\ s_4 & : & addr(Eleni, u_3) \end{array}\right]\ \right]$$

$$\left[\begin{array}{l} \mathrm{CXT} : \left[\ s_2 : spk(Eleni, u_2)\ \right] \\ \mathrm{CNT} : \lambda \left[\begin{array}{lcl} x & : & e \\ y & : & e \end{array}\right] . \left[\begin{array}{lcl} x & : & e \\ y & : & e \\ p & : & burn'(y, x) \end{array}\right] \end{array}\right]$$

split utterances with indexicals

Eleni: I burnt . . .
Bill: yourself! (as usual)

	IF	$?Ty(e)$, $\left[\ \textsc{context} : \left[\ s_s : addr(\mathbf{u}, \mathbf{x})\ \right]\ \right]$,
		$\uparrow_0\uparrow_{1*}\downarrow_0\ Fo(\mathbf{x})$
yourself:	THEN	$\mathtt{put}(Ty(e))$
		$\mathtt{put}(Fo(\mathbf{x}))$
	ELSE	$\mathtt{abort}$

$$\left[\begin{array}{l} \text{CXT} : \left[\ s_1 : spk(Eleni, u_1)\ \right] \\ \text{CNT} : \left[\ x_{=Eleni} : e\ \right] \end{array}\right]$$

$$\left[\begin{array}{l} \text{CXT} : \left[\begin{array}{lcl} s_3 & : & spk(Bill, u_3) \\ s_4 & : & addr(Eleni, u_3) \end{array}\right] \\ \text{CNT} : \left[\ y_{=Eleni} : e\ \right] \end{array}\right] \qquad \left[\begin{array}{l} \text{CXT} : \left[\ s_2 : spk(Eleni, u_2)\ \right] \\ \text{CNT} : \lambda \left[\begin{array}{lcl} x & : & e \\ y & : & e \end{array}\right] . \left[\begin{array}{lcl} x & : & e \\ y & : & e \\ p & : & burn'(y, x) \end{array}\right] \end{array}\right]$$

split utterances with indexicals

Eleni: I burnt . . .
Bill: yourself! (as usual)

$$\begin{bmatrix} \text{CXT}: & \begin{bmatrix} s_1 & : & spk(Eleni, u_1) \end{bmatrix} \\ \text{CNT}: & \begin{bmatrix} x_{=Eleni} & : & e \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{CXT}: & \begin{bmatrix} U_1 & : & u_2 \oplus u_3 \end{bmatrix} \\ \text{CNT}: & \lambda \begin{bmatrix} x & : & e \end{bmatrix} . \begin{bmatrix} x & : & e \\ y & : & A \\ p & : & burn(y, x) \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{CXT}: & \begin{bmatrix} s_3 & : & spk(Bill, u_3) \\ s_4 & : & addr(Eleni, u_3) \end{bmatrix} \\ \text{CNT}: & \begin{bmatrix} y_{=Eleni} & : & e \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{CXT}: & \begin{bmatrix} s_2 & : & spk(Eleni, u_2) \end{bmatrix} \\ \text{CNT}: & \lambda \begin{bmatrix} x & : & e \\ y & : & e \end{bmatrix} . \begin{bmatrix} x & : & e \\ y & : & e \\ p & : & burn'(y, x) \end{bmatrix} \end{bmatrix}$$

split utterances with indexicals

Eleni: I burnt . . .
Bill: yourself! (as usual)

$$Ty(t), \left[\begin{array}{ll} \text{CONTEXT}: & \left[\begin{array}{lll} ..., U_0 & : & u_1 \oplus u_2 \oplus u_3 ... \end{array}\right] \\ \text{CONTENT}: & \left[\begin{array}{lll} x_{=Eleni} & : & e \\ y_{=Eleni} & : & e \\ p & : & burn(x, y) \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{CXT}: & \left[\begin{array}{lll} s_1 & : & spk(Eleni, u_1) \end{array}\right] \\ \text{CNT}: & \left[\begin{array}{lll} x_{=Eleni} & : & e \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{CXT}: & \left[\begin{array}{lll} U_1 & : & u_2 \oplus u_3 \end{array}\right] \\ \text{CNT}: & \lambda \left[\begin{array}{lll} x & : & e \end{array}\right] . \left[\begin{array}{lll} x & : & e \\ y & : & A \\ p & : & burn(y, x) \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{CXT}: & \left[\begin{array}{lll} s_3 & : & spk(Bill, u_3) \\ s_4 & : & addr(Eleni, u_3) \end{array}\right] \\ \text{CNT}: & \left[\begin{array}{lll} y_{=Eleni} & : & e \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{CXT}: & \left[\begin{array}{lll} s_2 & : & spk(Eleni, u_2) \end{array}\right] \\ \text{CNT}: & \lambda \left[\begin{array}{lll} x & : & e \\ y & : & e \end{array}\right] . \left[\begin{array}{lll} x & : & e \\ y & : & e \\ p & : & burn'(y, x) \end{array}\right] \end{array}\right]$$

further dialogue phenomena

- self-**repair**

  A: Peter went swimming with Susan, um, or rather, surfing, yesterday. [‘Peter went surfing with Susan yesterday’]

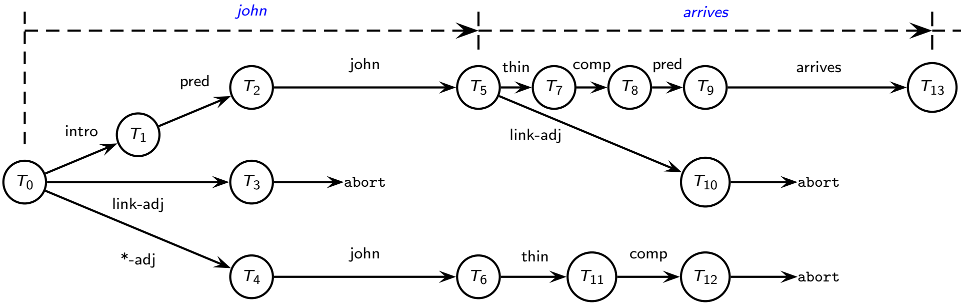
- other-repair, **clarification** (echoing)

  A: Peter went swimming with Susan
  B: Susan?

Parse/Generation States

- $$Parse/GenIU = \begin{bmatrix} words & : & list(Words) \\ actions & : & list(Actions) \\ tree & : & PointedTree \\ totalctxt & : & list(Tree) \\ cnt & : & RT \\ localctxt & : & RT \end{bmatrix}$$

parsing-paths context DAG



- actions (edges) are transitions between partial trees (nodes)
- processing paths probabilistically ranked

DS and TTR issues

- DS features to be maintained:
  - action-based syntax
  - no syntactic representation - grammaticality as constraints on update of semantic structures
  - incremental semantics
  - unified view of anaphora, ellipsis (quotation)
  - treat continuations as continuations
  - speech acts as system updates

DS and TTR issues to be resolved

DS-TTR: problems of integration

- Purver et al (2010)/Eshghi et al (2015): both LINK and TTR-extension: LINKed trees are extensions of RT (concatenation modulo relabelling)
  - can dispense with LINK but island restrictions?
- Purver et al (2010)/Eshghi et al (2015): both TTR and epsilon calculus?
- modality and propositional attitudes: possible worlds vs propositions as types
- monotonicity: multiple parsing paths – predictivity???
- dialogue moves: inferred, represented, encoded, default

Outline

Introductory Motivation
- What is grammar?

TTR to formalise conceptual structure
- TTR elements adopted
- Dynamic Syntax
- DS elements adopted
- Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions
- DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2

DS-TTR integration - Hannes

- What was the original **motivation** for combining DS and TTR and what was the gain to be expected?
- More specifically: What are the concrete **interface points** for DS and TTR integration?
- **Update**, **action** and **context** figure prominently in DS as well as in TTR. Are the notions implied similar and, if so, in which respects?
- How are DS **tree construction** and the **build-up of record types** related in DS-TTR? It seems that if we use DSs lexical actions as the main integration point of DS and TTR and, consequently, put record types into them, we get in principle two "up-ward working" compositional processes, one for the conceptual structure of DS and the other one for the record type construction. Is this impression wrong?
  - In more detail: Assume for the sake of discussion that both representations get their own semantics, however expressed, then we would have two different semantic values encoded in one DS-TTR-representation. Again, is this impression wrong?

DS-TTR integration issues-Hannes

- DS, as I see it, is **incremental** due to the *unfixed-node* conventions and the representation of the *main verb waiting for input*. Are there comparable mechanisms in TTR? Does TTR have different ones from those?
- Reconciliation of DS **quantifier theory** using the epsilon calculus and the Generalized Quantifier approach taken in TTR will require major changes in either the one or the other paradigm, right?

DS-TTR integration issues-Hannes

- If we look at TTR we see that it is **pragmatics and dialogue based** tout court (see the modelling of turn-exchange using dialogue game bords), plan-based (see the notion of agenda) and relies heavily on **mental sates** (see the labels "private" and "shared" in the information states).
- In contrast, DS relies on **interaction via grammar** defined on LOFT and avoids use of mental states. Does this fact impose a limit on the integratability of DS-TTR?

DS-TTR integration issues-Hannes

- I see a sort of **division of labour** between DS and TTR in the following way:
  - DS can, due to its generation-and-parsing facility cope with, e.g. types of ellipses, split utterances, self- or other-repairs, and across-sentence-clitics.
  - TTR can reconstruct dialogue interaction in a very fine-grained way using different types of modal notions.

DS-TTR integration issues-Hannes

- Both are motivated in different ways and favour different domains of application.
- A good deal of action and interaction in dialogue seems to be **automatic**, take e.g. alignment, hesitation phenomena, mid-turn acknowledgements, repair indicators and similar things. They are not **intentional** in the sense of "to be reconstructed with an intention operator defined on propositional content" Hence, this seems to be the "natural 'mechanistic' domain" of DS.
- On the contrary, modal notion based concepts seem to have their natural site in TTR. It may of course be controversial which phenomena are to be reconstructed using which technology. Is this an acceptable way to fix the divide between DS and TTR?

DS-TTR integration issues-Hannes

- One gets the idea that **TTR is more directed towards philosophy** (theory of perception and action, allusions to Aristotle, Kant and Russell, semantic puzzles, theory of proper names, reflecting the Montague-Partee-tradition) **DS more towards linguistics** (considering a wealth of natural languages, treating fine-grained data, e.g. morphology). So there is a division of labour in this sense as well. Right or wrong?

Outline

Introductory Motivation
- What is grammar?

TTR to formalise conceptual structure
- TTR elements adopted
- Dynamic Syntax
- DS elements adopted
- Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions
- DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2

action-based grammar

- **NL conceptual representations** not domain-specific, common to action/perception
- **syntax, lexicon** = set of actions (affordances) that predict, induce, develop structured contexts
- **coordinated action** (e.g. conversation) relies on:
  - action-oriented predictive simulative processing
  - non-conceptual procedural mechanisms (not high-order inference)

  $\Rightarrow$ interaction/coordination is an effect achievable directly from grammar-defined procedures, i.e. from low-level non-conceptual mechanisms
  (cf. Bickhard, 1992; Hurley, 2008; Pezzulo, 2011, 2014; Butterfill & Apperly 2013)

Thanks!

and thanks to:

Ellen Breitholtz, Ronnie Cann, Stelios Chatzikyriakidis, Robin Cooper, Arash Eshghi, Jonathan Ginzburg, Andrew Gargett, Pat Healey, Christine Howes, Ruth Kempson, Wilfried Meyer-Viol, Greg Mills, Matt Purver, Yo Sato, Graham White.

Outline

Introductory Motivation
- What is grammar?

TTR to formalise conceptual structure
- TTR elements adopted
- Dynamic Syntax
- DS elements adopted
- Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions
- DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2

quotation and linguistic processing

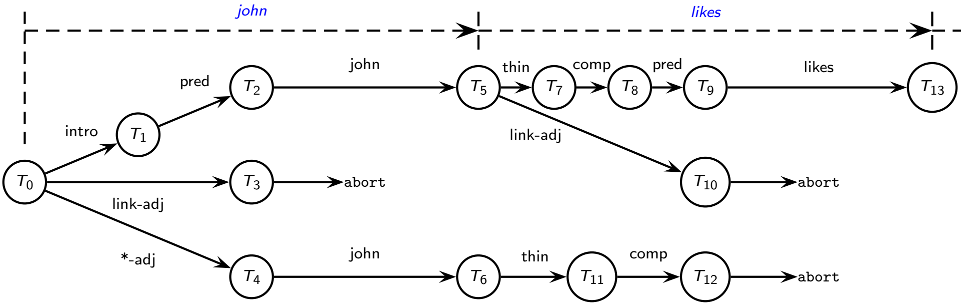
- quotational puns:

(15) The menu says that this restaurant serves "breakfast at any time" so I ordered French toast during the Renaissance. [Steven Wright joke]

(16) 'Marriage' is not a word, it's a sentence.

⇒ the **grammar** needs to be able to keep track of abandonned **parsing paths** as well as current viable ones.

parsing-paths context DAG



- actions (edges) are transitions between partial trees (nodes)
- processing paths probabilistically ranked

Outline

Introductory Motivation

What is grammar?

TTR to formalise conceptual structure

TTR elements adopted

Dynamic Syntax

DS elements adopted

Dynamic Syntax (DS)

DS-TTR

Hannes Rieser's Questions

General conclusions

DS-TTR and cognition - abandoning competence vs performance

Appendix 1

Appendix 2