Towards KoS/TTR-based proof-theoretic dialogue management

Vladislav Maraev* Jonathan Ginzburg† Staffan Larsson*
Ye Tian‡,† Jean-Philippe Bernardy*

*University of Gothenburg, †Université Paris Diderot, ‡Amazon Research Cambridge

CLASP seminar · October 17, 2018

CLASP centre for linguistic theory and studies in probability

GÖTEBORGS UNIVERSITET

Conversational relevance

(a) nice

A: What would you like to cook today?
U: Miso soup
A: How many portions?
U: Four

Conversational relevance

(a) nice

A: What would you like to cook today?
U: Miso soup
A: How many portions?
U: Four

(b) not very nice

A: What would you like to cook today?
U: Four portions of miso soup
A: How many portions?
U: ???

Conversational relevance

(a) nice

A: What would you like to cook today?
U: Miso soup
A: How many portions?
U: Four

(b) not very nice

A: What would you like to cook today?
U: Four portions of miso soup
A: How many portions?
U: ???

Conversational relevance is crucial for dialogue management. In this respect one should follow dialogue theories (in our case KoS Ginzburg, 2012). There hasn't been much progress in this respect since the early 2000s (Larsson, 2002).

Why (constructive) TTR

1. We use KoS (Ginzburg, 2012) which provides one of the most detailed analysis of conversational relevance
2. KoS is based on the Type Theory with Records (TTR) (Cooper, 2005)
3. Constructive type theories are used as a framework for semantics (e.g. Ranta, 1994)
4. TTR is set-theoretic, and hereby we are presenting proof-theoretic (constructive) version of it; our implementation follows MiniTT (Coquand et al., 2009)

Outline

- TTR, subtyping and update rules
- Dialogue management
- Question-answer relevance
- Concluding remarks

Next

- **TTR, subtyping and update rules**
- **Dialogue management**
- **Question-answer relevance**
- **Concluding remarks**

TTR: types

– **judgements**
$a : Ind$
a is a witness of (the type) Ind(ividual)

TTR: types

- **judgements**
  $a : Ind$
  a is a witness of (the type) *Ind*(ividual)
- **ptypes**: $\text{greet}(\text{speaker}, \text{addressee})$

TTR: types

- **judgements**
  $a : Ind$
  a is a witness of (the type) Ind(ividual)
- **ptypes**: greet(speaker, addressee)
- **record types**

$$\begin{bmatrix} \text{speaker} & : \text{Ind} \\ \text{addressee} & : \text{Ind} \\ \text{content} & : \text{greet(speaker, addressee)} \end{bmatrix}$$

TTR: types

– **judgements**
$a : Ind$
a is a witness of (the type) Ind(ividual)
– **ptypes**: greet(speaker, addressee)
– **record types**

$$\begin{bmatrix} \text{speaker} & : \text{Ind} \\ \text{addressee} & : \text{Ind} \\ \text{content} & : \text{greet(speaker, addressee)} \end{bmatrix}$$

– **records**
(1) is a witness of the type above iff a : Ind, b : Ind, θ : greet(speaker, addressee) and the weather doesn't matter.

$$\begin{bmatrix} \text{speaker} & = \text{a} \\ \text{addressee} & = \text{b} \\ \text{content} & = \theta \\ \text{weather} & = \text{sunny} \end{bmatrix} \qquad (1)$$

TTR: type construction operators

– **Function types**
  $(T_1 \to T_2)$

TTR: type construction operators

- **Function types**
  $(T_1 \rightarrow T_2)$
- **List types**
  $[T]$, $_{ne}[T]$, operations on lists: cons, head and tail

TTR: type construction operators

- **Function types**
  $(T_1 \to T_2)$
- **List types**
  $[T]$, $_{ne}[T]$, operations on lists: cons, head and tail
- **Meet types** (merge "$\dot{\wedge}$" in Cooper's TTR)

$$\begin{bmatrix}\mathrm{f}:\mathrm{Pasta}\end{bmatrix} \wedge \begin{bmatrix}\mathrm{f}:\mathrm{Cheese}\\ \mathrm{d}:\mathrm{Wine}\end{bmatrix} \text{ reduces to } \begin{bmatrix}\mathrm{f}:\mathrm{Pasta} \wedge \mathrm{Cheese}\\ \mathrm{d}:\mathrm{Wine}\end{bmatrix}$$

TTR: type construction operators

- **Function types**
  $(T_1 \to T_2)$
- **List types**
  $[T]$, ${}_{ne}[T]$, operations on lists: cons, head and tail
- **Meet types** (merge "$\dot{\wedge}$" in Cooper's TTR)

$$\begin{bmatrix}\text{f} : \text{Pasta}\end{bmatrix} \wedge \begin{bmatrix}\text{f} : \text{Cheese} \\ \text{d} : \text{Wine}\end{bmatrix} \text{ reduces to } \begin{bmatrix}\text{f} : \text{Pasta} \wedge \text{Cheese} \\ \text{d} : \text{Wine}\end{bmatrix}$$

- **Singleton types**
  if T is a type and x:T, then T_x is a type. a:T_x iff $a = x$.
  In record types we use manifest field notation to a represent singleton type. Notations $[a : T_x]$ and $[a = x : T]$ represent the same object.

Subtyping

Assuming that Pasta $\sqsubseteq$ Food and spaghetti : Pasta:

$$\begin{bmatrix} \text{x} : \text{Pasta} \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{x} : \text{Food} \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \text{x} : \text{Pasta} \\ \text{c} : \text{tasty(x)} \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{x} : \text{Pasta} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \text{x} : \text{Pasta} \end{bmatrix} \not\sqsubseteq \begin{bmatrix} \text{x} : \text{Pasta} \\ \text{c} : \text{tasty(x)} \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} \text{x=spaghetti} : \text{Food} \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{x} : \text{Pasta} \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} \text{x} : \text{Food}_{\text{spaghetti}} \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{x} : \text{Pasta} \end{bmatrix} \tag{6}$$

Subtyping

Assuming that Pasta $\sqsubseteq$ Food and spaghetti : Pasta:

$$\begin{bmatrix}\text{x}:\text{Pasta}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{x}:\text{Food}\end{bmatrix} \tag{2}$$

$$\begin{bmatrix}\text{x}:\text{Pasta}\\ \text{c}:\text{tasty(x)}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{x}:\text{Pasta}\end{bmatrix} \tag{3}$$

$$\begin{bmatrix}\text{x}:\text{Pasta}\end{bmatrix} \not\sqsubseteq \begin{bmatrix}\text{x}:\text{Pasta}\\ \text{c}:\text{tasty(x)}\end{bmatrix} \tag{4}$$

$$\begin{bmatrix}\text{x=spaghetti}:\text{Food}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{x}:\text{Pasta}\end{bmatrix} \tag{5}$$

$$\begin{bmatrix}\text{x}:\text{Food}_{\text{spaghetti}}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{x}:\text{Pasta}\end{bmatrix} \tag{6}$$

Given that $s : S$, $s = \begin{bmatrix}\text{x}=\text{spaghetti}\end{bmatrix}$ and $S = \begin{bmatrix}\text{x}:\text{Food}\end{bmatrix}$

$$S_s \sqsubseteq \begin{bmatrix}\text{x=spaghetti}:\text{Food}\end{bmatrix} \tag{7}$$

Update rules

1. We maintain dialogue state as a pair of a value and a type

$$(s, S) \text{ such that } s : S$$

Update rules

1. We maintain dialogue state as a pair of a value and a type

$$(s, S) \text{ such that } s : S$$

2. Update rules are defined

$$r : A \to B$$

Update rules

1. We maintain dialogue state as a pair of a value and a type

$$(s, S) \text{ such that } s : S$$

2. Update rules are defined

$$r : A \to B$$

3. Applicability condition

$$S_s \sqsubseteq A$$

Update rules

1. We maintain dialogue state as a pair of a value and a type

$$(s, S) \text{ such that } s : S$$

2. Update rules are defined

$$r : A \to B$$

3. Applicability condition

$$S_s \sqsubseteq A$$

4. After applying of the rule r state becomes

$$(r(s), B)$$

Example (state and rules)

$$(s_i : S_i),\ \ s_1 = \begin{bmatrix}\mathrm{x} = \mathrm{spaghetti}\end{bmatrix},\ \ S_1 = \begin{bmatrix}\mathrm{x} : \mathrm{Pasta}\end{bmatrix}$$

$$r_{cook} : \begin{bmatrix}\mathrm{x} : \mathrm{Pasta}\end{bmatrix} \rightarrow \begin{bmatrix}\mathrm{x} : \mathrm{Pasta} \\ \mathrm{c} : \mathrm{cooked(x)}\end{bmatrix}$$

$$r_{cook} = \lambda s. \begin{bmatrix}\mathrm{x} = s.\mathrm{x} \\ \mathrm{c} = \theta_{\mathrm{c(x)}}\end{bmatrix},$$

where $\theta_{c(x)}$ is a proof that the spaghetti is cooked.

$$r_{serve} : \begin{bmatrix}\mathrm{x} : \mathrm{Pasta} \\ \mathrm{c} : \mathrm{cooked(x)}\end{bmatrix} \rightarrow \begin{bmatrix}\mathrm{x} : \mathrm{Pasta} \\ \mathrm{c} : \mathrm{cooked(x)} \\ \mathrm{d} : \mathrm{served(x)}\end{bmatrix}$$

$$r_{serve} = \lambda s. \begin{bmatrix}\mathrm{x} = s.\mathrm{x} \\ \mathrm{c} = s.\mathrm{c} \\ \mathrm{d} = \theta_{\mathrm{d(x)}}\end{bmatrix},$$

where $\theta_{d(x)}$ is a proof that the spaghetti is served.

Example (change of state)

$$S_1 = \left[\mathrm{x} : \mathrm{Pasta}\right]$$

$$s_1 = \left[\mathrm{x} = \mathrm{spaghetti}\right]$$

Example (change of state)

$$S_1 = \begin{bmatrix}\mathrm{x} : \mathrm{Pasta}\end{bmatrix}$$

$$s_1 = \begin{bmatrix}\mathrm{x} = \mathrm{spaghetti}\end{bmatrix}$$

$$S_2 = \begin{bmatrix}\mathrm{x} : \mathrm{Pasta} \\ \mathrm{c} : \mathrm{cooked(x)}\end{bmatrix}$$

$$s_2 = r_{cook}(s_1) = \begin{bmatrix}\mathrm{x} = \mathrm{spaghetti} \\ \mathrm{c} = \theta_{\mathrm{c(x)}}\end{bmatrix}$$

Example (change of state)

$$S_1 = \begin{bmatrix} \mathrm{x} : \mathrm{Pasta} \end{bmatrix}$$

$$s_1 = \begin{bmatrix} \mathrm{x} = \mathrm{spaghetti} \end{bmatrix}$$

$$S_2 = \begin{bmatrix} \mathrm{x} : \mathrm{Pasta} \\ \mathrm{c} : \mathrm{cooked(x)} \end{bmatrix}$$

$$s_2 = r_{cook}(s_1) = \begin{bmatrix} \mathrm{x} = \mathrm{spaghetti} \\ \mathrm{c} = \theta_{\mathrm{c(x)}} \end{bmatrix}$$

$$S_3 = \begin{bmatrix} \mathrm{x} : \mathrm{Pasta} \\ \mathrm{c} : \mathrm{cooked(x)} \\ \mathrm{d} : \mathrm{served(x)} \end{bmatrix}$$

$$s_3 = r_{serve}(s_2) = \begin{bmatrix} \mathrm{x} = \mathrm{spaghetti} \\ \mathrm{c} = \theta_{\mathrm{c(x)}} \\ \mathrm{d} = \theta_{\mathrm{d(x)}} \end{bmatrix}$$

Next

- TTR, subtyping and update rules
- **Dialogue management**
- Question-answer relevance
- Concluding remarks

Minimal example

Proof-of-concept: basic interaction

U: hello
A: Hello world!

Primary KoS types

$$InformationState =_{def} \begin{bmatrix} \text{private} : \begin{bmatrix} \text{agenda} : [\text{Move}] \end{bmatrix} \\ \text{dgb} \quad : \begin{bmatrix} \text{moves} : [\text{Move}] \end{bmatrix} \end{bmatrix}$$

$$GreetingMove =_{def} \begin{bmatrix} \text{spkr} & : \text{Ind} \\ \text{addr} & : \text{Ind} \\ \text{content} & : \begin{bmatrix} \text{c} : \text{greet}(\text{spkr}, \text{addr}) \end{bmatrix} \end{bmatrix}$$

$$(GreetingMove \sqsubseteq Move)$$

Countergreet the user

$$CGU : InformationState \wedge \left[\,\langle \text{LM is user-greeting}\rangle\,\right] \\ \quad \rightarrow InformationState \wedge \left[\,\langle \text{agenda is non-empty}\rangle\,\right]$$

$$CGU = \lambda s. \begin{bmatrix} \text{private} = \left[\text{agenda} = \text{cons}(\text{CGM}, s.\text{private.agenda})\right] \\ \text{dgb} \quad\;\; = s.\text{dgb} \end{bmatrix},$$

$$\text{where } CGM = \begin{bmatrix} \text{spkr} & = \text{agent} \\ \text{addr} & = \text{user}_0 \\ \text{content} & = \left[\text{c} = \theta\right] \end{bmatrix} \;\; (\theta \text{ is a proof of greeting})$$

Also in the paper

1. Other update rules for integrating user moves and fulfilling the agenda
2. Full example of update chain

Next

- TTR, subtyping and update rules
- Dialogue management
- **Question-answer relevance**
- Concluding remarks

Small extension to TTR

1. Support for boolean types: ($true$: Bool) and ($false$: Bool)
2. Support for conditionals:
   - (IF $true$ THEN x ELSE y) $= x$
   - (IF $false$ THEN x ELSE y) $= y$

Questions

Question definition is a way to establish connection between possible answer and its interpretation in the context of question.

$$Question : Type$$

$$Question =_{def} \begin{bmatrix} A : \text{Type} \\ \text{Q} : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

$$PolarQuestion =_{def} \begin{bmatrix} \text{A=Bool} : \text{Type} \\ \text{Q} \qquad\quad : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

$$UnaryWhQuestion =_{def} \begin{bmatrix} \text{A=Ind} : \text{Type} \\ \text{Q} \qquad\quad : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

Example: wh-question

$$Question : Type$$

$$Question =_{def} \begin{bmatrix} A : \text{Type} \\ \text{Q} : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

$$UnaryWhQuestion =_{def} \begin{bmatrix} \text{A=Ind} : \text{Type} \\ \text{Q} \quad : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

$$[\![\text{Where do you live?}]\!] = \begin{bmatrix} \text{A} = \text{City} \\ \text{Q} = \lambda \text{a.live(a)} \end{bmatrix}$$

Example: yn-question

$$Question : Type$$

$$Question =_{def} \begin{bmatrix} A : \text{Type} \\ \text{Q} : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

$$PolarQuestion =_{def} \begin{bmatrix} \text{A=Bool} : \text{Type} \\ \text{Q} \quad : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

$$[\![\text{Do you live in Aix?}]\!] =$$

$$\begin{bmatrix} \text{A} = \text{Bool} \\ \text{Q} = \lambda\text{a. IF a THEN live(Aix) ELSE } \neg\text{live(Aix)} \end{bmatrix}$$

Answers

Recalling the definition of question:

$$Question : Type$$

$$Question =_{def} \begin{bmatrix} A : \text{Type} \\ \text{Q} : \text{A} \rightarrow \text{Prop} \end{bmatrix}$$

Answer will return record type for given question:

$$Answer : Question \rightarrow Type$$

$$Answer =_{def} \lambda q. \begin{bmatrix} \text{answer} : q.\text{A} \\ \text{sit} \quad\quad : q.\text{Q(answer)} \end{bmatrix}$$

Note: (q.A:Type) and (q.Q(answer):Prop)

Example: answer to wh-question

$$Answer : Question \rightarrow Type$$

$$Answer =_{def} \lambda q. \begin{bmatrix} \text{answer} : q.\text{A} \\ \text{sit} \quad\quad : q.\text{Q(answer)} \end{bmatrix}$$

$$[\![\text{Where do you live?}]\!] = \begin{bmatrix} \text{A} = \text{City} \\ \text{Q} = \lambda \text{a.live(a)} \end{bmatrix}$$

Example: answer to wh-question

$$Answer : Question \rightarrow Type$$

$$Answer =_{def} \lambda q. \begin{bmatrix} \text{answer} : q.\text{A} \\ \text{sit} \quad\quad : q.\text{Q(answer)} \end{bmatrix}$$

$$[\![\text{Where do you live?}]\!] = \begin{bmatrix} \text{A} = \text{City} \\ \text{Q} = \lambda\text{a.live(a)} \end{bmatrix}$$

$$Answer([\![\text{Where do you live?})]\!] = \begin{bmatrix} \text{answer} : \text{City} \\ \text{sit} \quad\quad : \text{live(answer)} \end{bmatrix}$$

$$[\![\text{in Aix}]\!]([\![\text{Where do you live?}]\!]) = \begin{bmatrix} \text{answer} = \text{Aix} \\ \text{sit} \quad\quad = \theta_{\text{la}} \end{bmatrix}$$

Example: answer to yn-question

$$Answer : Question \to Type$$

$$Answer =_{def} \lambda q. \begin{bmatrix} \text{answer} : q.\text{A} \\ \text{sit} \quad\quad : q.\text{Q(answer)} \end{bmatrix}$$

$$[\![\text{Do you live in Aix?}]\!] =$$

$$\begin{bmatrix} \text{A} = \text{Bool} \\ \text{Q} = \lambda\text{a.IF a THEN live(Aix) ELSE } \neg\text{live(Aix)} \end{bmatrix}$$

Example: answer to yn-question

$$Answer : Question \to Type$$

$$Answer =_{def} \lambda q. \begin{bmatrix} \text{answer} : q.\text{A} \\ \text{sit} \quad : q.\text{Q(answer)} \end{bmatrix}$$

$$[\![\text{Do you live in Aix?}]\!] =$$

$$\begin{bmatrix} \text{A} = \text{Bool} \\ \text{Q} = \lambda\text{a.IF a THEN live(Aix) ELSE} \neg\text{live(Aix)} \end{bmatrix}$$

$$Answer([\![\text{Do you live in Aix?}]\!]) =$$

$$\begin{bmatrix} \text{answer} = \text{Bool} \\ \text{sit} \quad = \text{IF answer THEN live(Aix) ELSE} \neg\text{live(Aix)} \end{bmatrix}$$

$$[\![\text{yes}]\!]([\![\text{Do you live in Aix?}]\!]) = \begin{bmatrix} \text{answer} = \text{true} \\ \text{sit} \quad = \theta_{\text{la}} \end{bmatrix}$$

Also in the paper:

1. Dealing with answers in form of propositions ("I live in Aix" instead of "in Aix")
2. Partial (and incremental) resolution of answers, for utterances like:

   A: What do you want today?
   U: A beer, please, and chips.

Next

- TTR, subtyping and update rules
- Dialogue management
- Question-answer relevance
- **Concluding remarks**

Concluding remarks

1. Our aim is to maintain tight connection between dialogue theory and dialogue systems' design
2. Dream system:
   - maintains rich information state
   - uses domain-dependent and domain-independent dialogue rules
   - learns probabilities for the rules

Questions?

$$\begin{bmatrix} \text{speaker=Vlad} : \text{Ind} \\ \text{attendees} : [\text{Ind}] \\ \text{content} : \text{thank(speaker, attendees)} \end{bmatrix}$$

References I

Cooper, R. (2005). Records and record types in semantic theory. *Journal of Logic and Computation*, 15(2):99–112.

Coquand, T., Kinoshita, Y., Nordström, B., and Takeyama, M. (2009). A simple type-theoretic language: Mini-TT.

Ginzburg, J. (2012). *The interactive stance*. Oxford University Press.

Larsson, S. (2002). *Issue-based dialogue management*. Department of Linguistics, Göteborg University.

Ranta, A. (1994). Type-theoretical grammar.

Update rules (Robin Cooper)

Update episode:

$$\lambda r : \left[\text{agenda}{=}[\left[\text{e} : \text{pick\_up}(\text{a}, \text{c})\right]] : [\text{RecType}]\right]$$
$$\lambda e : \left[\text{e} : \text{pick\_up}(\text{a}, \text{c})\right] .$$
$$\left[\text{agenda} = \left[\text{e} : \text{attract\_attention}(\text{a}, \text{b})\right] : [\text{RecType}]\right]$$

has a type: $\left[\text{agenda} : [\text{RecType}]\right] \rightarrow (Rec \rightarrow RecType)$

> *... [it maps] an information state containing an agenda (modelled as a record containing an agenda field) and an event (modelled as a record) to a record type.*

Update rules (our approach)

$$
\begin{aligned}
IS &= \left[\text{agenda} : [\text{Event}]\right] \\
UR &: IS \rightarrow Event \rightarrow IS \\
UR_k &: IS \wedge \left[\text{agenda} = [\left[\text{e} : \text{pick\_up(a, c)}\right]] : [\text{Event}]\right] \\
&\quad \rightarrow \left[\text{e} : \text{pick\_up(a, c)}\right] \\
&\quad \rightarrow IS \wedge \left[\text{agenda} = [\left[\text{e} : \text{attract\_attention(a, b)}\right]] : [\text{Event}]\right]
\end{aligned}
$$

and we also specify how the value is assigned:

$$UR_k : \lambda s.\lambda e. \left[\text{agenda} = [\left[\text{e} = \theta_{(\text{a,b})}\right]]\right],$$

where $\theta_{(a,b)}$: attract_attention(a, b) is a proof that an event can be counted as such.