From Dialogue to Multilogue via Type Theory

Jonathan Ginzburg
Department of Computer Science
King's College London
Strand
London WC2R 2LS
ginzburg@dcs.kcl.ac.uk

Part 0

Overview

Main Issues

- How to scale down Dialogue to Monologue and up to Multilogue?

  —Dialogue: querier/responder

  —Monologue: self answering

  —Multilogue: multiple discussants

- Basic idea: compositionally break down interaction protocols using *conversational rules*.

- Use Type Theory with Records (TTR) (Cooper, 2005) to formulate conversational rules.
- Type Theory with Records: a framework that allows

  —Rich ontologies à la Situation Semantics

  —Dynamic semantic techniques à la DRT

  —Constraint-based Grammar à la HPSG

  —Dialogue Analysis à la KOS
- TTR notationally similar to Type Feature Structures, but substantively different

  —TTR has token type distinction: crucial for dealing with grounding/clarification potential

  —TTR contains λ-calculus: crucial for doing semantics

Structure of Talk

- Type Theory with Records: the basics
- Ontology in Constructive Type Theory:

  —Semantics

  —Grammar
- Decomposing protocols into conversational rules
- Scaling down to monologue
- Some principles for scaling up to Multilogue

Part 1

Type Theory and Semantic Ontology

Records and Record Types

- A record is is an ordered tuple of the form (1), where crucially each successive field can depend on the values of the preceding fields:

(1) a. $\begin{bmatrix} l_i = k_i \\ l_{i+1} = k_{i+1} \ \ldots \\ l_{i+j} = k_{i+j} \end{bmatrix}$

b. $\begin{bmatrix} \text{x} = \text{a} \\ \text{y} = \text{b} \\ \text{prf} = \text{p} \end{bmatrix}$

Records and Record Types

- Together with records come record *types*. A record type is simply an ordered tuple of the form (2), where again each successive type can depend on its predecessor types within the record:

(2)
$$\begin{bmatrix} l_i : T_i \\ l_{i+1} : T_{i+1} \ \ldots \\ l_{i+j} : T_{i+j} \end{bmatrix}$$

Records and Record Types

- Record types allow us to place constraints on records: the basic typing mechanism assumed is that a record r is of type RT if all the typing constraints imposed by RT are satisfied by r.
- If $a_1 : T_1, a_2 : T_2(a_1), \ldots, a_n : T_n(a_1, a_2, \ldots, a_{n-1})$,

  the record:

$$\begin{bmatrix} l_1 & = & a_1 \\ l_2 & = & a_2 \\ \ldots & & \\ l_n & = & a_n \end{bmatrix} \text{ is of type: } \begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2(l_1) \\ \ldots & & \\ l_n & : & T_n(l_1, l_2, \ldots, l_{n-1}) \end{bmatrix}$$

- Crucially, not all the fields in r need to be ‘disciplined’ by RT.

Records and Record Types

- The record

  $\begin{bmatrix} \text{runner} = \text{bo} \\ \text{time} = \text{2pm, Dec 20} \\ \text{place} = \text{batumi} \end{bmatrix}$ is of the type $\begin{bmatrix} \text{runner} : \text{Ind} \\ \text{time} : \text{Time} \\ \text{place} : \text{Loc} \end{bmatrix}$

- and of the type $\begin{bmatrix} \text{runner} : \text{Ind} \\ \text{time} : \text{Time} \end{bmatrix}$ and of the type $\begin{bmatrix} \text{runner} : \text{Ind} \end{bmatrix}$ and of the type $\begin{bmatrix} \; \end{bmatrix}$, the type that imposes no constraints.

Ontology in Type Theory

- Type Theoretic World (Cooper 2004, simplified):

  TYPE = ⟨ Type^n, BasicType, ProofType^n, RecType^n, $\langle A, F^n \rangle$, ⟩
- Type^n is the set of types of order n, built up recursively using type construction operations.
- BasicType: IND, TIME, LOC, LEX …
- ProofType^n ("interface with external reality"): tuples consisting of entities [from the model] and predicates.
- RecType^n: set of records, record types defined with respect to a set of objects used as labels.
- $\langle A, F^n \rangle$ is a model (assigning entities to BasicType, and tuples to ProofType^n).

Ontology in Type Theory

- The universe is connected to the real world via the proof types and the model.
- The model grounds the basic types.
- From these beginnings, arise structured objects via two recursive mechanisms: type construction and record cutting.

An event

- Proof types play a role akin to (atomic) situation types (SOAs) in situation semantics, serving as the smallest particles of external reality.
- Combining these into record types allows us to form 'molecules' of external reality.
- Assuming the existence of basic types TIME and LOC(ation), one could offer (3) as the most rudimentary notion of situation, namely that it is a record which carries information about spatio-temporal extent:

  (3)

$$\mathrm{SIT} =_{def} \begin{bmatrix} \text{time} : \ \text{TIME} \\ \text{loc:LOC} \end{bmatrix}$$

An event

- The type of a situation with a woman riding a bicycle would then be a record $\begin{bmatrix} \ldots \\ \text{x} = \text{a} \\ \text{c1} = \text{p1} \\ \text{y} = \text{b} \\ \text{c2} = \text{p2} \\ \text{time} = \text{t0} \\ \text{loc} = \text{l0} \\ \text{c3} = \text{p3} \\ \ldots \end{bmatrix}$ of type $\begin{bmatrix} \text{x: IND} \\ \text{c1: woman(x)} \\ \text{c2: bicycle(x)} \\ \text{y: IND} \\ \text{time : TIME} \\ \text{loc:LOC} \\ \text{c3: ride(x,y,time,loc)} \\ \\ \end{bmatrix}$

  such that: a:IND; c1: woman(a); b: IND; p2: bicycle(b); t0 : TIME; l0 : LOC;p3: ride(a,b,t0,l0);

- A theory of situations requires various topological, physical and other constraints to be imposed on the universe of records, as explained in more detail by Cooper (RL&C, 2005).

Type Constructors

- In order to do semantics, we need to have available various type construction operations, which allow for a recursive building up of the type theoretic universe.

(4) a. **function types**: if T_1 and T_2 are types, then so is $(\mathrm{T}_1 \rightarrow \mathrm{T}_2)$, the type of functions from elements of type T_1 to elements of type T_2. f:$(\mathrm{T}_1 \rightarrow \mathrm{T}_2)$ iff the domain of f is $\{a|a : T_1\}$ and the range of f is a subset of $\{a|a : T_2\}$

b. **The type of lists**: if T is a type, [T], the type of lists each of whose members is a : T, is a type.

c. **The unique type**: if T is a type and $x : T$, then T_x is a type. $a : T_x$ iff $a = x$.

Simultaneous abstraction with restrictions

- Function types allow one to model abstraction. Given the existence of record typing, this allows for a simple modelling of simultaneous, restricted abstraction— multiple (incl none) entities get 'abstracted over' simultaneously, while encoding restrictions.

- The simultaneous abstract in (5a) can be modelled as the function in (5b), which has the type in (5c):

(5) a. $\lambda\{x_1, \ldots, x_k\}\phi(x_1, \ldots, x_k)$
$[\psi_1(x_1, \ldots, x_{n_1}), \ldots, \psi_k(x_k, \ldots, x_{n_k})]$

b. $$\begin{bmatrix} x_1 = a_1 \\ \dots \\ x_k = a_k \\ c_1 = p_1 \\ \dots \\ c_k = p_k \end{bmatrix} \mapsto \phi(a_1, \dots, a_k)$$

c. r : $$\begin{bmatrix} x_1 : T_1 \\ \dots \\ x_k : T_k \\ c_1 : \psi_1(a_1, \dots, a_{n_1}) \\ \dots \\ c_k : \psi_k(a_k, \dots, a_{n_k}) \end{bmatrix} \rightarrow \phi(r.x_1, \dots, r.x_k)$$

Simultaneous abstraction with restrictions

- A simplified analysis of the meaning of the sentence ‘I see Bo’:

(6) a. context-assgn: $\alpha = \begin{bmatrix} \text{x : Ind} \\ \text{t : Time} \\ \text{p1: speak(x,t)} \\ \text{y: Ind} \\ \text{p2: named(Bo,y)} \end{bmatrix}$

b. ρ:
$\begin{bmatrix} \text{cont : see(context-assgn.x,context-assgn.t,context-assgnr.y)} \end{bmatrix}$

c. context-assgn:$\alpha \rightarrow \rho$: (the type of) functions from records of type α into records of type ρ.

d. A function of this type maps records of the form

$$\begin{bmatrix} \ldots \\ \mathrm{x} = \mathrm{x}_0 \\ \mathrm{t} = \mathrm{t}_0 \\ \mathrm{p1} = \mathrm{c1} \\ \mathrm{y} = \mathrm{y}_0 \\ \mathrm{p2} = \mathrm{c2} \\ \ldots \end{bmatrix}$$

where x_0 : Ind,t_0 : Time, c1: speak(x_0,t_0), y_0: Ind, and c2: named(Bo,y0),

into a record of the type $\left[\mathrm{cont} : \mathrm{see}(\mathrm{x}_0,\mathrm{t}_0,\mathrm{y}_0)\right]$

Vacuous abstraction

- A vacuous abstract is a constant function of some kind—where implementation varies is the domain of the function.
- If we think of a unary abstract as involving a domain type with one field which directly influences the 'value' of the function, a binary abstract as one whose domain type contains two such fields etc, then the domain type of a 0-ary type would simply be the empty type [].
- hence: a 0-ary abstract a constant function from the universe of all records (since every record is of the type [].).
- An alternative implementation is to arbitrarily choose some fixed entity as the domain of vacuous abstracts. For instance, we could take the domain type to be the type $[]_{[]}$ whose sole member is [].

Propositions

- CTT offers a straightforward way for us to model propositions using records. A proposition is a record of the form in (7a). The type of propositions is the record type (7b):

(7) a. $\begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix}$

b. $\begin{bmatrix} \text{sit} & : & \text{Record} \\ \text{sit-type} & : & \text{RecType} \end{bmatrix}$

- Truth:

(8) A proposition $\begin{bmatrix} \text{sit} & = & r_0 \\ \text{sit-type} & = & p_0 \end{bmatrix}$ is true iff $r_0 : p_0$

Logic

- We can now import an essentially intuitionist logic to underpin the Boolean structure of our space of propositions.
- The following are the operations on types typically assumed in CTT:

  (9) a. $\neg T_0$ is the type $T_0 \rightarrow \bot$: the type $\neg T_0$ is witnessed if one can show that every situation of type T_0 is also of type $\bot$.

  b. $T_1 \wedge T_2$: to show that $T_1 \wedge T_2$ is witnessed, one needs a pair $< p1, p2 >$ where p_1 is of type T_1 and p_2 is of type T_2.

  c. $T1 \vee T2$: a witness for $T1 \vee T2$ is an entity p_0 where p_0 is of type T_1 or p_0 is of type T_2.
- Given this, simple to define the requisite Boolean operations on propositions.

Questions as Propositional Abstracts

- Most NLP and AI researchers assume questions to be λ-abstracts, i.e. they assume QPA.
- Most formal semanticists do NOT adopt QPA, since the early 1980s.

Resuscitating QPA

- QPA worth saving because it is the simplest theory that simultaneously deals with the basic desiderata (short answers, answerhood, question dependence) and is tractable.
- Simplicity: any theory of questions needs to associate an abstract-like object with interrogatives to explicate the resolution of short answers.
- Answerhood: Explicating answerhood involves a characterization that utilizes in some form an abstract-like object.
- Tractability: QPA yields a transparently implementable theory. In contrast, EAC theories that identify questions with *partitions* of propositions (e.g. Groenendijk and Stokhof 1984 et seq.) are intractable on their most transparent implementation (see e.g. Bos and Gabsdil 1999).

Questions: some simple examples

- Given the existence of sitsemian-like propositions, a proposition/situation-type-like distinction, and a theory of λ-abstraction, it is relatively straightforward to develop a theory of questions as propositional abstracts in CTT.
- A question will be a function from records into propositions:

  (10) a. Did Bo run

  b. CTT representation: maps records $\mathrm{r} : \mathrm{T}_0 = \left[\;\right]$ into propositions of the form

$$\begin{bmatrix} \mathit{sit} & = & r_1 \\ \text{sit-type} & = & \begin{bmatrix} \mathrm{c} : \mathrm{run(b)} \end{bmatrix} \end{bmatrix}$$

c. who ran

d. CTT representation: maps records r : T_{who} =
$\begin{bmatrix} \mathrm{x} & : & \mathrm{Ind} \\ \mathrm{rest} & : & \mathrm{person(x)} \end{bmatrix}$ into propositions of the form
$$\begin{bmatrix} sit & = & r_1 \\ \text{sit-type} & = & \begin{bmatrix} \mathrm{c} : \mathrm{run(r.x)} \end{bmatrix} \end{bmatrix}$$

e. who greeted what

f. CTT representation: maps records r : $\mathrm{T}_{who,what}$

$$= \left[\begin{array}{lll} \mathrm{x} & : & \mathrm{Ind} \\ \mathrm{rest} & : & \mathrm{person(x)} \\ \mathrm{y} & : & \mathrm{Ind} \\ \mathrm{rest} & : & \mathrm{thing(x)} \end{array}\right]$$

into propositions of the form

$$\left[\begin{array}{lll} sit & = & r_1 \\ \text{sit-type} & = & \left[\mathrm{c} : \mathrm{greet(r.x,r.y)}\right] \end{array}\right]$$

Fixing the type of questions

- Recall that one of the stumbling blocks facing QPA as implemented in the late 1970s was the the distinctness of types associated with interrogatives.
- This problem does not arise here due to the following elementary fact:

  Fact 1 Function type subsumption
  For any types A, A', B *if* $A \sqsubseteq A'$, *then* $(A' \rightarrow B) \sqsubseteq (A \rightarrow B)$.
- Now the types which can be associated with the functions in (10) satisfy the subtype hierarchy in (11a).
- Given this and Fact 1, we get a correspondingly inverted hierarchy for the function types in (11b):

  (11) a. $T_{who,what} \sqsubseteq T_{who} \sqsubseteq T_0$

  b. $(T_0 \rightarrow Prop) \sqsubseteq (T_{who} \rightarrow Prop) \sqsubseteq (T_{who,what} \rightarrow Prop)$

- Can be generalized to accommodate arbitrarily complex questions. (See Ginzburg, JoL&C (in press)).
- Alternative more general strategy use partial function spaces.

Part 2

Using Type Theory to do grammar

Doing HPSG in CTT

- The basic idea is very straightforward.
- Utterance types (aka signs) modelled as record types
- Utterance tokens—speech events— modelled as records, as indeed are events in general (see Part 1)

A Simple Example

- My grammar fragment posits the sound/syntax/meaning constraint in (12a) as a rule of English. For a speech event $se0$, (12b), to be classified as being of this type, the requirements in (12c) will need to be met:

(12) a.
$$\begin{bmatrix} \textsc{phon} : \texttt{did jo leave} \\ \textsc{cat} : \text{S} \\ \textsc{c-params} : \begin{bmatrix} \text{s0: SIT} \\ \text{t0: TIME} \\ \text{j: IND} \\ \text{c3: Named(j,jo)} \end{bmatrix} \\ \text{cont} = ([\,])\begin{bmatrix} \text{sit} = \text{s0} \\ \text{sit-type} = \text{Leave(j,t0)} \end{bmatrix} : \text{Questn} \end{bmatrix}$$

b. $$\begin{bmatrix} \textsc{phon} = \text{ph1} \\ \textsc{cat} = \text{cat1} \\ \textsc{c-params} = \text{ctxt0} = \begin{bmatrix} \ldots \\ \text{s0} = \text{sit0} \\ \text{t0} = \text{time0} \\ \text{j} = \text{j0} \\ \text{c3} = \text{c30} \\ \ldots \end{bmatrix} \\ \text{cont} = \text{cont0} \end{bmatrix}$$

c. ph1 : `did jo leave`; cat1 : S;
sit0 : SIT, time0 : TIME, j0 : IND, c30 : Named(j0,jo)

$$\text{cont0} = ([])\begin{bmatrix}\text{sit} = \text{sit0} \\ \text{sit-type} = \text{Leave(j0,time0)}\end{bmatrix} : \text{Questn}$$

Not a notational variant of HPSG$_{TFS}$

- HPSG$_{CTT}$ *looks* a lot like HPSG$_{TFS}$.
- HPSG$_{CTT}$ directly provides semantic entities, whereas HPSG$_{TFS}$ simulates them.

Some Basic grammatical types: referential NP

(13)
$$\begin{bmatrix} \textsc{phon} : \texttt{jo} \\ \textsc{cat} : \text{N} \\ \textsc{cont} : \text{IND} \\ \textsc{c-params}: \left\{ \begin{bmatrix} \text{index} = \text{cont:IND} \\ \text{restr} : \text{Named(cont)(jo)} \end{bmatrix} \right\} \end{bmatrix}$$

- The content is taken to be of type IND, i.e. *cont* will denote an individual.
- A proper name generally varies with context (and is potentially ambiguous or unknown to the addressee). Hence, it projects a contextual parameter. This is represented by associating a type with the C-PARAMS field.

- By using a manifest field, the index field is equated with the value of the content. The restriction on the contextual parameter is that this individual be named the name which is the type of the phon field.

Some Basic grammatical types: instrans verb

- an auxiliary such as *did*—its content (ignoring tense) will simply be an identity function:

  (14) $$\begin{bmatrix} \textsc{phon} : \texttt{did} \\ \textsc{cat} : \mathrm{V} \\ \textsc{cont} = \lambda p.p : \mathrm{Type} \end{bmatrix}$$

- *runs* is specified as a function that maps its argument—coreferential with the subject—to a type that holds of (an event) if that individual runs:

  (15)

$$\left[\begin{array}{l}
\textsc{phon} : \texttt{runs} \\
\textsc{cat} : \left[\begin{array}{l} \textsc{head} : \mathrm{V} \\ \textsc{subj} : \left[\begin{array}{l} \text{cat: syncat} \\ \text{cont: IND} \end{array}\right] \end{array}\right] \\
\textsc{cat} : \mathrm{V} \\
\textsc{cont} = \left[\text{r:}\left[\text{runner= cat.subj.cont} : \text{IND}\right]\right]\text{.Run(r.runner)} : \text{Type}
\end{array}\right]$$

Some Basic grammatical types: Basic decl-cl rule

- The type *decl-hd-subj-cl* is the analogue of the standard ‘S $\rightarrow$ NP VP’ rule. It builds a proposition whose situation field is contextually provided and whose sit-type field arises by applying the head-dtr’s content to the subj-dtr’s content.

(16) a. *decl-hd-subj-cl* =

$$\begin{bmatrix} \text{c-params: } \begin{bmatrix}\text{s0:Rec}\end{bmatrix} \\ \text{cont} = \begin{bmatrix} \text{sit} = \text{s0} \\ \text{sit-type} = \text{hd-dtr.cont(subj-dtr.cont)} \end{bmatrix} : \text{ Prop} \end{bmatrix}$$

head : sign subj:sign

b. Jo visits Kim

$$\begin{bmatrix} \text{phon} : \texttt{Jo visits Kim} \\ \text{cont} = \begin{bmatrix} \text{sit} = \text{s0} \\ \text{sit-type} = \ \text{r:} \begin{bmatrix}\text{visitor=cat.subj-dtr.cont} : \text{IND}\end{bmatrix} \\ \text{Visit(r.visitor,k)}(\begin{bmatrix}\text{subj-dtr.cont} = \text{j}\end{bmatrix}) \\ \mapsto \text{Visit(j,k)} \end{bmatrix} : \text{Prop} \end{bmatrix}$$

$$\text{subj} : \begin{bmatrix} \textsc{phon} : \texttt{jo} \\ \textsc{cont} = \text{j: IND} \end{bmatrix} \ \text{hd-dtr:} \begin{bmatrix} \text{phon} : \texttt{visits Kim} \\ \text{cont} = \ \text{r:} \begin{bmatrix}\text{visitor=cat.subj-dtr.cont} : \text{IND}\end{bmatrix} \\ \text{Visit(r.visitor,k)} : \text{Type} \end{bmatrix}$$

Some Basic grammatical types: Basic int-cl rule

- An argument-filling *wh*-phrase like *who* gets assigned a minimally different lexical entry from a proper name:

(17)
$$\begin{bmatrix} \textsc{phon} : \texttt{who} \\ \textsc{cat} : \mathrm{N} \\ \textsc{cont} : \mathrm{IND} \\ \textsc{func-dom} : \left\langle \begin{bmatrix} \mathrm{index} = \mathrm{cont} : \mathrm{IND} \\ \mathrm{restr} : \mathrm{person(index)} \end{bmatrix} \right\rangle \end{bmatrix}$$

- It has the same content type as a proper name, but does not project a contextual parameter. The fact that the argument role it is associated with will get abstracted over, i.e. will specify the domain of a function, is captured by assigning it a non-trivial value for the field FUNCTIONAL DOMAIN (FUNC-DOM).

Some Basic grammatical types: Basic int-cl rule

- A ‘canonical’ *wh*-interrogative that involves a filler gap construction, the filler being a dislocated *wh*-phrase. Here the content arises by forming a function whose domain is the type that constitutes the filler’s func-dom value and whose range is the content of the hd-dtr:

(18) a. *wh-ns-int-cl*

$$\begin{bmatrix} \text{cont} = (\begin{bmatrix}\text{fill-dtr.func-dom.first}\end{bmatrix})\text{hd-dtr.cont} : \text{Questn} \\ \text{filler.cont} = \text{slash.cont} \end{bmatrix}$$

$$\text{filler} : \text{sign} \qquad \text{head} : \text{sign}$$

Some Basic grammatical types: declarative fragments

- The following type allows us to build declarative fragments (e.g. short answers).
- The content arises by predicating the abstract denoted by the question of the content of the fragment.
- parallelism with the antecedent is captured by dependencies between SAL-UTT and the NSU.

(19) *decl-frag-cl* (quant-free version)

$$\begin{bmatrix} \textsc{sal-utt} : \text{Sign} \\ \textsc{max-qud} : \text{WhQuestn} \\ \textsc{cat} : v \\ \textsc{dtrs} : \left\langle \text{hd-dtr: } \begin{bmatrix} \text{cat} = \text{sal-utt.cat} : \text{Syncat} \end{bmatrix} \right\rangle \\ \textsc{cont} = \text{max-qud(hd-dtr.cont)} : \text{Prop} \end{bmatrix}$$

Part 3

Decomposing Protocols

Dialogue Analysis

- Dialogue analyst's task: describe conventionally acceptable patterns of interaction (*protocols*), in terms of sequences of information states.
- Methodological constraint: compositionality (but as with the sentential level not obsessively [cf. the need for constructionism].

Basics of Interaction

- Larsson (2002), Cooper (2004) pioneered use of type theory to underpin issue based dialogue management.
- Public/private interface emphasized: update rules describe modification of total information state (TIS).
- Here: emphasis on description at public level (in so far as possible); some mention of public/private interface below.

- each dialogue participant's view of the common ground, the dialogue gameboard (DGB), are records of the following type:

(20) $$\begin{bmatrix} \text{facts} : \text{Prop} \\ \text{Moves} : \text{list(IllocProp)} \\ \text{QUD} : \text{list(Question)} \\ \text{c} : \neg\text{Resolve(facts, QUD)} \end{bmatrix}$$

- With respect to Moves—often focus on first element of list—LatestMove.
- The basic units of change are mappings between DGBs that specify how one DGB configuration can be modified into another. ∴ *conversational rule.*
- The types specifying its domain and its range respectively the *preconditions* and the *effects.*

Basics of Interaction

- Notationwise a conversational rule will be specified as in (21a). We will often notate such a mapping as in (21b):

(21) a. $\mathrm{r} : \begin{bmatrix} \ldots \\ \mathrm{dgb1} : \mathrm{DGB} \\ \ldots \end{bmatrix} \mapsto \begin{bmatrix} \ldots \\ \mathrm{dgb2} : \mathrm{DGB} \\ \ldots \end{bmatrix}$

b. $\begin{bmatrix} \mathrm{pre} : \mathrm{RType} \\ \mathrm{post} : \mathrm{RType} \end{bmatrix}$

Basics of Interaction: the compositional principle

- We can recognize one fundamental compositional principle:

  (22) Composition of conversational rules: given 2 conversational rules *part1*, *part2* that satisfy preconds(part2) $\sqsubseteq$ effects(part1) they can be composed yielding a new conversational rule whose preconds = preconds(part1) and whose effects = effects(part2)

- (22) will be the basic tool we use in decomposing protocols.

Greeting and Parting

- An initiating greeting typically occurs dialogue initially.
- The primary contextual effect of such a greeting is simply providing the addressee with the possibility of reciprocating with a counter-greeting.
- A countergreeting simply grounds the original greeting, requires no response, nor has other contextual effects.
- We should be careful not to build into greetings any *obligation* to countergreet, given examples like the following:

  (23) A: Hi Mo. How are you?
  B: OK. Where are you heading?

Greeting and Parting

- The conversational rule associated with greeting:

  (24)

$$\begin{bmatrix} \text{pre} = & \begin{bmatrix} \text{init-spkr: Ind} \\ \text{init-addr: Ind} \\ \text{moves} = \text{elist} : \text{list(IllocProp)} \\ \text{qud} = \text{elist} : \text{list(Question)} \\ \text{facts} = \text{commonground1} : \text{Prop} \end{bmatrix} \\ \text{post} = & \begin{bmatrix} \text{LatestMove} = \text{Greet(pre.init-spkr,pre-initaddr):IllocProp} \\ \text{qud} = \text{pre.qud} : \text{list(Question)} \\ \text{facts} = \text{pre.facts} : \text{Prop} \end{bmatrix} \end{bmatrix}$$

- Note also the need to initialize facts—a contextual parameter (cf. Clark), forced upon us by thinking in terms of TTR.

Greeting and Parting

- Countergreeting has as its precondition that LatestMove is greet(A,B).
- Assuming a distinction between greeting and *counter*greeting, motivated by existence in some languages of forms usable only as *responses* to greetings (e.g. Arabic ‘marhabteyn’, ‘sabax elnur’ etc.), intonational differences (e.g. in English initiating greeting involve fall, responsive greetings involve rise.) Boils down to the initiating/reactive distinction (see below.).

(25)

$$\left[\begin{array}{ll}
\text{pre} = & \left[\begin{array}{l}
\text{init-spkr: Ind} \\
\text{init-addr: Ind} \\
\text{LatestMove} = \\
\text{Greet(pre.init-spkr,pre-initaddr):IllocProp} \\
\text{qud} = \text{elist} : \text{list(Question)} \\
\text{facts} = \text{commonground1} : \text{Prop}
\end{array}\right] \\
\text{post} = & \left[\begin{array}{l}
\text{LatestMove} = \\
\text{CtrGreet(pre.initaddr,pre.initspkr):IllocProp} \\
\text{qud} = \text{pre.qud} : \text{list(Question)} \\
\text{facts} = \text{pre.facts} : \text{Prop}
\end{array}\right]
\end{array}\right]$$

Greeting: an example

(26) A: Hi B: Hi.

- Specify words like 'hi', 'good morning' in the lexicon as

$$\begin{bmatrix} \text{phon} : \text{HI} \\ \text{c-params} = \begin{bmatrix} \text{s} : \text{Ind} \\ \text{a: Ind} \\ \ldots \end{bmatrix} : \text{RType} \\ \text{cont} = \text{Greet(s,a)} : \text{IllocProp} \end{bmatrix}$$

Greeting and Parting

- Parting is in some sense the mirror image of greeting: the basic prep condition for parting is that the conversation is at a stage that allows it to be terminated.
- This means that QUD is empty, either because all issues previously raised have indeed been discussed sufficiently or because the parter decides to downdate those that have not:

(27)

$$\begin{bmatrix} \text{pre} = \begin{bmatrix} \text{init-spkr: Ind} \\ \text{init-addr: Ind} \\ \text{qud} = \text{elist} : \text{list(Question)} \\ \text{facts} = \text{commonground1} : \text{Prop} \end{bmatrix} \\ \text{post} = \begin{bmatrix} \text{LatestMove} = \text{Part(pre.init-spkr,pre-initaddr): IllocProp} \\ \text{qud} = \text{pre.qud} : \text{list(Question)} \\ \text{facts} = \text{pre.facts} : \text{Prop} \end{bmatrix} \end{bmatrix}$$

- Counterparting:

(28) a.

$$\left[\begin{array}{l} \text{pre} = \left[\begin{array}{l} \text{init-spkr: Ind} \\ \text{init-addr: Ind} \\ \text{LatestMove} = \\ \text{Part(pre.initspkr,pre.initaddr):IllocProp} \\ \text{qud} = \text{elist} : \text{list(Question)} \\ \text{facts} = \text{commonground1} : \text{Prop} \end{array}\right] \\ \text{post} = \left[\begin{array}{l} \text{LatestMove} = \\ \text{CounterPart(pre.init-addr,pre.initspkr):IllocProp} \\ \text{qud} = \text{pre.qud} : \text{list(Question)} \\ \text{facts} = \text{pre.facts} : \text{Prop} \end{array}\right] \end{array}\right]$$

b.
$$\left[\begin{array}{l}
\text{pre} = \left[\begin{array}{l}
\text{init-spkr: Ind} \\
\text{init-addr: Ind} \\
\text{LatestMove} = \\
\text{CtrPart(pre.initspkr,pre.initaddr):IllocProp} \\
\text{qud} = \text{elist} : \text{list(Question)} \\
\text{facts} = \text{commonground1} : \text{Prop}
\end{array}\right] \\
\text{post} = \left[\begin{array}{l}
\text{LatestMove} = \\
\text{Disengaged(pre.initaddr,pre.initspkr)} : \text{IllocProp} \\
\text{qud} = \text{pre.qud} : \text{list(Question)} \\
\text{facts} = \text{pre.facts} : \text{Prop}
\end{array}\right]
\end{array}\right]$$

Asking, Asserting, Answering, and Accepting

- Broadly speaking queries and assertions are either *issue initiating*—they introduce an issue unrelated to those currently under discussion— or they are *reactive*—they involve a reaction to a previously raised issue.
- Accounting for the the reactive ones using DGB–based conversational rules is simple. These can also be used to explicate the effects *issue initiating* moves have. Will not discuss today the background of such moves, which is intrinsically tied in with the unpublicized aspects of information states (cf. Larsson 2002, Asher and Lascarides 2003)

Asking, Asserting, Answering, and Accepting

- The most prototypical query exchange involves q getting posed by A, B adopting q and providing a response:

(29) cooperative query exchange

```
1. LatestMove.Cont = Ask(A,q):  IllocProp
2. A: push q onto QUD; release turn
3. B: push q onto QUD; take turn; make q-specific
   utterance (partial answer to q; subquestion of
   q)
```

- Two aspects of this protocol are not query specific:
  1. The protocol is like the ones we have seen for greeting and parting a 2-person turn exchange protocol (2-PTEP).
  2. The specification `make q-specific utterance` is an instance of a general constraint that characterizes the contextual background of reactive queries and assertions:

(30) QSpec:
$$\begin{bmatrix} \text{pre} = \begin{bmatrix} \text{qud} = [\text{q},\ldots] : \text{list(Question)} \\ \text{facts} = \text{commonground1} : \text{Prop} \end{bmatrix} \\ \text{post} = \begin{bmatrix} \text{LatestMove} = \text{ip0} : \text{IllocProp} \\ \text{c1} : \text{Qspecific(ip0,pre.dgb.qud.first)} \\ \text{qud} = \text{pre.qud} : \text{list(Question)} \\ \text{facts} = \text{pre.facts} : \text{Prop} \end{bmatrix} \end{bmatrix}$$

- The only query specific aspect of the protocol is:

(31) Ask QUD–incrementation:

$$\begin{bmatrix} \text{pre} = \begin{bmatrix} \text{init-spkr: Ind} \\ \text{init-addr: Ind} \\ \text{q : Question} \\ \text{LatestMove} = \text{Ask(initspkr,initaddr,q):IllocProp} \\ \text{qud : list(Question)} \\ \text{facts} = \text{commonground1 : Prop} \end{bmatrix} \\ \text{post} = \begin{bmatrix} \text{LatestMove} = \text{pre.LatestMove : IllocProp} \\ \text{qud} = \text{[q,pre.qud] : list(Question)} \\ \text{facts} = \text{pre.facts : Prop} \end{bmatrix} \end{bmatrix}$$

Asking, Asserting, Answering, and Accepting

- Basic protocol for assertion can be summarized as follows:

(32) cooperative assertion exchange

```
1. LatestMove.Cont = Assert(A,p):  IllocProp
2. A: push p?  onto QUD, release turn
3. B: push p?  onto QUD, take turn; Option 1:
   Discuss p?, Option 2:  Accept p
```

(33)

```
1. LatestMove.Cont = Accept(B,p) :  IllocProp
2. B: increment FACTS with p; pop p?  from QUD;
3. A: increment FACTS with p; pop p?  from QUD;
```

- What are the components of this protocol? Not specific to assertion is the fact that it is a 2-PTEP; similarly, the discussion option is simply an instance of QSPEC.

- This leaves two novel components: QUD incrementation with *p*? and acceptance.

(34) Assert QUD–incrementation:

$$\begin{bmatrix} \text{pre} = \begin{bmatrix} \text{init-spkr: Ind} \\ \text{init-addr: Ind} \\ \text{p : Prop} \\ \text{LatestMove = Assert(initspkr,initaddr,p):IllocProp} \\ \text{qud : list(Question)} \\ \text{facts = commonground1 : Prop} \end{bmatrix} \\ \text{post} = \begin{bmatrix} \text{LatestMove = pre.LatestMove : IllocProp} \\ \text{qud = [p?,pre.qud] : list(Question)} \\ \text{facts = pre.facts : Prop} \end{bmatrix} \end{bmatrix}$$

- Acceptance is a somewhat more involved matter because a lot of the action is not directly perceptible.
- The labour can be divided here in two: on the one hand is the action brought about by an acceptance utterance (e.g. ‘mmh’, ‘I see’).
- The background for an acceptance by B is an assertion by A and the effect is to modify LatestMove:

(35) Accept move:

$$\begin{bmatrix} \text{pre} = \begin{bmatrix} \text{init-spkr: Ind} \\ \text{init-addr: Ind} \\ \text{p : Prop} \\ \text{LatestMove} = \text{Assert(initspkr,initaddr,p):IllocProp} \\ \text{qud} = \text{[p?,pre.qud] : list(Question)} \\ \text{facts} = \text{commonground1 : Prop} \end{bmatrix} \\ \text{post} = \begin{bmatrix} \text{LatestMove} = \text{Accept(pre.initaddr,initspkr,p) : IllocProp} \\ \text{qud} = \text{pre.qud : list(Question)} \\ \text{facts} = \text{pre.facts : Prop} \end{bmatrix} \end{bmatrix}$$

- The second component of acceptance is the incrementation of FACTS by p.
- This is not quite as straightforward as it might seem: when FACTS gets incremented, we also need to ensure that p? gets downdated from QUD, to ensure that the Nonresolvedness condition is maintained.
- In order to ensure that this is the case, we need to check for all existing elements of QUD that they are not resolved by the new value of FACTS.
- Hence, accepting p involves both an update of FACTS and a downdate of QUD—minimally just removing p?, potentially removing other questions as well.

(36)

$$\left[\begin{array}{ll} \text{pre} = & \left[\begin{array}{l} \text{init-spkr: Ind} \\ \text{init-addr: Ind} \\ \text{p : Prop} \\ \text{LatestMove} = \text{Accept(initspkr,initaddr,p):IllocProp} \\ \text{qud} = \text{[p?,pre.qud] : list(Question)} \\ \text{facts} = \text{commonground1 : Prop} \end{array}\right] \\ \text{post} = & \left[\begin{array}{l} \text{LatestMove} = \text{pre.LatestMove : IllocProp} \\ \text{facts} = \text{pre.facts} \wedge \text{p: Prop} \\ \text{qud} = \text{NonResolve(pre.qud,facts) : list(Question)} \end{array}\right] \end{array}\right]$$

$$\text{NonResolve} : [q_1, \ldots, q_n], p \mapsto [\ldots q_i \ldots] \subset [q_1, \ldots, q_n],$$
$$q \in [\ldots q_i \ldots] \leftrightarrow \neg Resolve(p, q)$$

Part 4

Scaling down to Monologue

Some simple monologue cases

- How to deal with cases like self answering, successive querying/assertion?
- Note that in (37c) the self answer isn't even discussed:

(37) a. Vicki: When is, when is Easter? March, April? (BNC, KC2, 2938-2939)

b. Unknown: When was that? That was last week wasn't it? (3010-11)

c. Frederick: When's it taking place? (pause)
Joan: Erm it's the last Saturday of half term so I should think it's about erm (pause) when's half term? Eighteenth on the Monday I think (pause) so it'll be twenty (pause) erm it's the fifth is it when I, no it won't (pause) it's not as late as that is it (pause) eighteenth, nineteenth, twentieth, twenty one (pause) about the twenty second, something like that. (3067-3069)

Some simple monologue cases: querying

- Self answering is accommodated by QSpec: no turn exchange mechanism built in.
- Second query becomes QUD maximal:

(38) a. Ann: What are your shifts next week? Can you remember offhand?
James: Yes. I'm early Monday and Tuesday (pause) and Wednesday (pause) a day off Thursday (pause) Friday (pause) late (4968-4971)

b. Ann: Anyway, talking of over the road, where is she? Is she home?
Betty: No. She's in the Cottage. (5121-5124)

- Evidence from NSUs for precedence of second query:

(39) a. A: Who is the dog waiting for? Why is he making such a racket? B: ?No one. He's just happy.

b. A: Why is the dog making such a racket? Who is he waiting for? B: No one. He's just happy..

- Return to cases such as following later:

(40) a. Arthur: How old is she? Forty?
Evelyn Forty one! (456-458)

b. Evelyn: what time is it?, is it eight thirty five yet?
Arthur: twenty past (6527-8)

Some simple monologue cases: assertion

- QSpec also allows for successive assertions p_1, p_2, where p_2 is About p_1?.
- When later assertion p_2 accepted, the issue associated with the earlier assertion p_1 will be downdated iff FACTS (including p_2) resolves p_1?:

  (41) a. A: Several people showed up. Bill did.
  B: Aha.
  A: Max did.
  B: I see.

- This is an implicit mechanism for accepting p_1.
- Can accommodate further rhetorical relations, if necessary, by postulating additional conversational rules.

Part 5

Scaling up to Multilogue (joint work with Raquel Fernàndez)

The Fundamental Issue of Multilogue

- Dialogue—two person conversation—is by now a topic with an ever increasing theoretical, corpus-based, and implementational literature.
- The study of *multilogue*—conversation with 3 or more participants—is still in its early stages.
- Fundamental question: **Is multilogue an aggregate of dialogues? Or is multilogue an irreducibly different form of interaction?**
- Is dialogue a multilogue with 2 participants?

Some issues in scaling up from dialogue to multilogue

- Dialogue: current addressee is next speaker; consensus involves acceptance by addressee
- Multilogue: audience is larger, role change more complex.
- Querying: who has right/obligation to respond?
- Assertion/Uttering: does a proposition (utterance) become common ground after a single/multiple/universal acceptance?

Multiagent Systems

- Multiagent Systems: communication between autonomous software agents.
- Most interaction protocols are designed only for two participants at a time.
- The FIPA (Foundation for Intelligent Physical Agents) interaction protocols (IP) for agent communication language (FIPA, 2003) are most typically designed for two participants, an initiator and a responder.
- Some IPs permit the broadcasting of a message to a group of addressees, and the reception of multiple responses by the original initiator.
- (Dignum and Vreeswijk, 2003): such conversations can not be considered multilogue, but rather a number of parallel dialogues.

Mission Rehearsal Exercise Project

- One of few existing multilogial systems is work done by David Traum and colleagues, related to the Mission Rehearsal Exercise (MRE) Project.
- The setting of the MRE project, used for training of Army personnel, is a virtual reality environment where multiple partners (including humans and other autonomous agents) engage in multi-conversation situations.
- The model of grounding implemented in the MRE project (inspired by Traum and Poesio, 1997) can only be used in cases where there is a single initiator and responder. How to scale up to multiple addresses unclear: should the contents be considered grounded when any of the addressees has acknowledged them? Should evidence of understanding be required from every addressee?

Long Distance NSUs in Dialogue and Multilogue: Data

- The corpus we use in this investigation includes and extends the sub-corpus mentioned earlier.
- 14,315 sentences. created by excerpting a 200-speaker-turn section from 54 BNC files. Of these files, 29 are transcripts of conversations between two dialogue participants, and 25 files are multilogue transcripts.
- A total of 1285 NSUs were found in our sub-corpus. Table 1 shows the raw counts of NSUs found in the dialogue and multilogue transcripts, respectively.

	# NSUs	# BNC files
Dialogue	709	29
Multilogue	576	25
Total	1285	54

Table 1: Total of NSUs in Dialogue and Multilogue

Long Distance NSUs in Dialogue and Multilogue: Data

- All NSUs encountered within the corpus were manually classified according to the NSU typology presented in (Fernandez and Ginzburg, 2002)
- In order to be able to measure the distance between NSUs and their antecedents, all instances were additionally tagged with the sentence number of their antecedent utterance.

			Distance						
NSU Class	Example	Total	1	2	3	4	5	6	>6
Acknowledgment	*Mm mm.*	595	578	15	2				
Short Answer	*Ballet shoes.*	188	104	21	17	5	5	8	28
Affirmative Answer	*Yes.*	109	104	4			1		
Clarification Ellisis	*John?*	92	76	13	2	1			
Repeated Ack.	*His boss, right.*	86	81	2	3				
Rejection	*No.*	50	49	1					
Factual Modifier	*Brilliant!*	27	23	2	1	1			
Repeated Aff. Ans.	*Very far, yes.*	26	25	1					
Help Rejection	*No, my aunt.*	24	18	5		1			
Check Question	*Okay?*	22	15	7					
Filler	*... a cough.*	18	16	1		1			
Bare Mod. Phrase	*On the desk.*	16	11	4			1		
Sluice	*When?*	11	10	1					
Prop. Modifier	*Probably.*	11	10	1					
Conjunction Phrase	*Or a mirror.*	10	5	4	1				
		1285	1125	82	26	9	7	8	28
		%	87.6	6.3	2	0.6	0.5	0.6	2.1

Table 2: Total of NSUs sorted by Class and Distance

Long Distance NSUs in Dialogue and Multilogue: Results

- 87% of NSUs have a distance of 1 sentence (i.e. the antecedent was the immediately preceding sentence), and that the vast majority (about 96%) have a distance of 3 sentences or less.
- The proportion of long distance NSUs in multilogue is far higher than in dialogue. NSUs that have a distance of 7 sentences or more appear exclusively in multilogue transcripts.

Distance	1	2	3	4	5	6	7	8	9	10	>10
Total	467	45	15	8	6	7	3	1	2	1	21
%	41	55	55	88	86	87	100	100	100	100	100

Table 3: Total and % of NSUs in Multilogue sorted by Distance

Adjacency of grounding and affirmation utterances

- A fundamental characteristic of the remaining majoritarian classes of NSUs, Ack(nowledgements), Affirmative Answer, CE (clarification ellipsis), Repeated Ack(nowledgements), and Rejection.
- These are used either in grounding interaction, or to affirm/reject propositions.
- The overwhelming adjacency to their antecedent underlines the locality of these interactions.

The Multilogue Short Answer (MSA) effect

- With a few exceptions, NSUs that have a distance of 3 sentences or more are exclusively short answers.
- The frequency of long distance short answers stands in strong contrast to the other NSUs classes; indeed, over 44% of short answers have more than distance 1, and over 24% have distance 4 or more, like the last answer in the following example:

(42)	Allan:	How much do you think?
	Cynthia:	Three hundred pounds.
	Sue:	More.
	Cynthia:	A thousand pounds.
	Allan:	More.
	Unknown:	<unclear>
	Allan:	Eleven hundred quid apparently.
		[BNC, G4X]

The Multilogue Short Answer (MSA) effect

- Table 4 shows the total number of short answers found in dialogue and multilogue respectively, and the proportions sorted by distance over those totals:

Short Answers	Total #	1	2	3	> 3
Dialogue	54	82	9	9	0
Multilogue	134	44	11	8	37

Table 4: % over the totals found in dialogue and multilogue

Dialogue v. Multilogue Short Answer distribution

- short answers are more common in multilogue than in dialogue—71% v. 29%.
- The distance pattern exhibited by these two groups is strikingly different: only 18% of short answers found in dialogue have a distance of more than 1 sentence, with all of them having a distance of at most 3, like the short answer in (43):

(43)	Malcolm:	[...] cos what's three hundred and sixty divided by seven?
	Anon 1:	I don't know.
	Malcolm:	Yes I don't know either!
	Anon 1:	Fifty four point fifty one point four. [BNC, KND]

- This dialogue/multilogue asymmetry argues against reductive views of multilogue as sequential dialogue.

Explaining away the dialogue short answer distribution

- In dialogue certain commonly observed conditions will enforce adjacency between short answers and their interrogative antecedents.

  (44) a. Questions have a simple, one phrase answer.

  b. Questions can be answered immediately, without preparatory or subsequent discussion.

- For multilogue (or at least certain genres thereof), (44) are less likely to be maintained: different CPs can supply different answers, even assuming that relative to each CP there is a simple, one phrase answer.

- The more CPs there are in a conversation, the smaller their common ground and the more likely the need for clarificatory interaction.

Explaining away the dialogue short answer distribution

- A pragmatic account of this type of the frequency of adjacency in dialogue short answers seems clearly preferable to any actual mechanism that would *rule out* long distance short answers. These can be perfectly felicitous—see e.g. example (45)above which would work fine if the turn uttered by Sue had been uttered by Allan instead:

(45)	Allan:	How much do you think?
	Cynthia:	Three hundred pounds.
	Allan:	More.
	Cynthia:	A thousand pounds.
	Allan:	More.
	Cynthia:	<unclear>
	Allan:	Eleven hundred quid apparently.
		[BNC, G4X]

Long Distance short answer unaffected by group size

- Not all groups have the same number of participants—but the long distance possibilities seem unaffected.
- Following Fay et al 2003, we distinguish between small groups (those with 3 to 5 participants) and large groups (those with more than 5 participants).
- The size of the group is determined by the amount of participants that are active when a particular short answer is uttered. We consider active participants those that have made a contribution within a window of 30 turns back from the turn where the short answer was uttered.

Long Distance short answer unaffected by group size

- Table 5 shows the amount of short answers found in small and large groups respectively, after examination of those short answers at more than distance 3 (a total of 46):

Group Size	Total
≤ 5	20
> 5	26

Table 5: Short Answers in small and large groups

- Large groups multilogues in the corpus are all transcripts of tutorials, training sessions or seminars, which exhibit a rather particular structure.

- The general pattern involves a question being asked by the tutor or session leader, the other participants then taking turns to answer that question. The tutor or leader acts as turn manager. She assigns the turn explicitly usually by addressing the participants by their name without need to repeat the question under discussion:

(46)	Anon1:	How important is those three components and what value would you put on them [...]
	Anon3:	Tone forty five. Body language thirty .
	Anon1:	Thank you.
	Anon4:	Oh.
	Anon1:	Melanie.
	Anon5:	twenty five.
	Anon1:	Yes.
	Anon5:	Tone of voice twenty five. [BNC, JYM]

- Small group multilogues on the other hand have a more unconstrained structure: after a question is asked, the participants tend to answer freely. Answers by different participants can follow one after the other without explicit acknowledgements nor turn management, like in (47):.

(47)	Anon 1:	How about finance then? <pause>
	Unknown 1:	Corruption
	Unknown 2:	Risk <pause dur=30>
	Unknown 3:	Wage claims <pause dur=18>

Two Benchmarks of multilogue

- **Multilogue Long Distance short answers (MLDSA)**: querying protocols for multilogue must license short answers an unbounded number of turns from the original query.
- **Multilogue adjacency of grounding/acceptance (MAG)**: assertion and grounding protocols for multilogue should license grounding/clarification/acceptance moves only adjacently to their antecedent utterance.

Dialogue Protocols for querying and assertion

(48) cooperative query exchange

```
1. LatestMove.Cont =
   Ask(A,q):  IllocProp

2. A: q becomes QUD maximal; release turn

3. B: q becomes QUD maximal; take turn; make q-specific
   utterance; release turn.
```

(49) cooperative assertion exchange

```
1. LatestMove.Cont =
   Assert(A,p):  IllocProp

2. A: p?  becomes QUD maximal, release turn

3. B: p?  becomes QUD maximal, take turn; ⟨ Option 1:
   Discuss p?, Option 2:  Accept p ⟩
```

(50)
```
1. LatestMove.Cont = Accept(B,p) :  IllocProp
2. B: increment FACTS with p; pop p?  from QUD;
3. A: increment FACTS with p; pop p?  from QUD;
```

Principles of protocol extension

- All three are intuitive, framework independent, and result in protocols that approximate interaction in certain settings.
- The final principle we consider, Add Silent Active Participants (ASAP), seems to yield the best results, relative to the benchmarks we introduced before.

Add Silent Passive Participants

- The simplest principle is Add Silent Passive Participants (ASPP). This involves adding participants who merely observe the interaction. They keep track of facts concerning a particular interaction, but their context is not facilitated for them to participate:

  (51) Given a dialogue protocol π, add roles $C_1, \ldots, C_n$ where each C_i is a silent participant: given an utterance u_0 classified as being of type T_0, C_i updates C_i.DGB.FACTS with the proposition $u_0 : T_0$.

- Applying ASPP yields essentially multilogues which are sequences of dialogues.

- A special case of this are moderated multilogues, where all dialogues involve a designated individual (who is also responsible for turn assignment.).
- ASPP is not adequate for multilogue scaling up since *inter alia* it will not fulfill the MLDSA benchmark.

Duplicate Responders

- A far stronger principle is Duplicate Responders (DR):

  (52) Given a dialogue protocol π, add roles C_1,...,C_n which duplicate the responder role

- Applying DR to the querying protocol yields the following protocol:

  (53) Querying with multiple responders

```
1. LatestMove.Cont =
   Ask(A,q): IllocProp
2. A: q becomes QUD maximal; release turn
3. Resp$_1$: q becomes QUD maximal; take turn; make
   q-specific utterance; release turn
4. Resp$_2$: q becomes QUD maximal; take turn; make
   q-specific utterance; release turn
5. ...
```

6. `Resp`$_n$`:  q becomes QUD maximal; take turn; make q-specific utterance; release turn`

- This yields interactions such as (47):

(54)

Anon 1:	How about finance then? <pause>
Unknown 1:	Corruption
Unknown 2:	Risk <pause dur=30>
Unknown 3:	Wage claims <pause dur=18>

- Pro: the querying protocol in (53) licenses long distance short answers, so satisfies the MLDSA benchmark.

Duplicate Responders

- Con: the contextual updates it enforces will not enable it to deal with the following (constructed) variant on (47), in other words does not afford responders to comment on previous responders, as opposed to the original querier:

  (55) A: Who should we invite for the conference?

  B: Svetlanov.

  C: No (=Not Svetlanov), Zhdanov

  D: No (= Not Zhdanov, $\neq$ Not Svetlanov), Gergev

Duplicate Responders

- Applying DR to the assertion protocol will yield the following protocol:

(56) Assertion with multiple responders

```
1. LatestMove.Cont =
   Assert(A,p):  IllocProp
2. A: p?  becomes QUD maximal, release turn
3. Resp$_1$:  p?  becomes QUD maximal, take turn; ⟨
   Option 1:  Discuss p?, Option 2:  Accept p ⟩
4. Resp$_2$:  p?  becomes QUD maximal, take turn; ⟨
   Option 1:  Discuss p?, Option 2:  Accept p ⟩
5. ...
6. Resp$_n$:  p?  becomes QUD maximal, take turn; ⟨
   Option 1:  Discuss p?, Option 2:  Accept p ⟩
```

- One significant problem with this protocol—equally applicable to the corresponding DRed grounding protocol—is that it licences long distance acceptance and thus is inconsistent with the MAG benchmark.

Add Silent Active Participants

- A principle intermediate between ASPP and DR is Add Silent Active Participants (ASAP):

  (57) Given a dialogue protocol π, add roles $C_1,\ldots,C_n$, which affect the same contextual update as the interaction initiator.

Add Silent Active Participants: assertion

- Applying ASAP to the dialogue assertion protocol yields the following protocol:

(58) Assertion for a conversation involving
{A,B,C_1,. . . ,C_n}

```
1. LatestMove.Cont = Assert(A,p):  IllocProp
2. A: p? becomes QUD maximal; release turn
3. C_i:  p? becomes QUD maximal
4. B: p? becomes QUD maximal; take turn; ⟨Option 1:
   Accept p, Option 2:  Discuss q_1, where q_1 is
   QUD--maximal⟩
```

(59)

```
1. LatestMove.Cont = Accept(B,p) :  IllocProp
2. B: increment FACTS with p; pop p? from QUD;
3. C_i:increment FACTS with p; pop p? from QUD;
4. A: increment FACTS with p; pop p? from QUD;
```

Add Silent Active Participants: assertion

- This protocol satisfies the MAG benchmark in that acceptance is strictly local.
- The protocol enforces *communal acceptance*—acceptance by one CP can count as acceptance by all other addressees of an assertion.
- There is an obvious rational motivation for this, given the difficulty of a CP constantly monitoring an entire audience (when this consists of more than one addressee) for acceptance signals.

- It also enforces quick reaction to an assertion—anyone wishing to dissent from p must get their reaction in early i.e. immediately following the assertion since further discussion of $p?$ is not countenanced if acceptance takes place.

- The latter can happen of course as a consequence of a dissenter not being quick on their feet; on this protocol to accommodate such cases would require some type of backtracking.

Add Silent Active Participants: querying

- Applying ASAP to the dialogue querying protocol yields the following protocol:

(60) Querying for a conversation involving $\{$ A,B,C$_1$,...,C$_n\}$

```
1. LatestMove.Cont =
   Ask(A,q): IllocProp
2. A: q becomes QUD maximal; release turn
3. C_i: q becomes QUD maximal
4. B: q becomes QUD maximal; take turn; make
   q_1-specific utterance, where q_1 is QUD--maximal.
```

- This improves on the DR generated protocol because it does allow responders to comment on previous responders—the context is modified as in the dialogue protocol.

- As it stands, this protocol won't fully deal with examples such as (47)—the issue introduced by each successive participant takes precedence given that QUD is assumed to be a stack.
- This can be remedied by slightly modifying this latter assumption: we will assume that when a question q gets added to QUD it doesn't subsume *all* existing questions in QUD, but rather only those on which q does not *depend*
- Dependence: common assumption in work on questions, e.g. Ginzburg and Sag, 2000. Intuitively corresponding to the notion of 'is a subquestion of'.
- Formally: q_1 depends on q_2 iff any proposition p such that p resolves q_2 also satisfies p is about q_1.

Add Silent Active Participants: querying

- New definition of QUD maximality:

  (61) q is QUD$^{mod(dependence)}$ maximal iff for any q_0 in QUD such that ¬Depend(q, q_1): $q \succ q_0$.

- This is conceptually attractive because it reinforces that the order in QUD has an intuitive semantic basis.

- This ensures that any polar question p? introduced into QUD, whether by an assertion or by a query, subsequent to a wh-question q on which p? depends does not subsume q.

- Hence, q will remain accessible as an antecedent for NSUs, *as long as no new unrelated topic has been introduced.*

- Also provides a way of dealnig with cases like:

  (62) a. Arthur: How old is she? Forty?

Evelyn Forty one! (456-458)

b. Evelyn: what time is it?, is it eight thirty five yet?
Arthur: twenty past (6527-8)

- Assuming this modification to QUD is implemented in the above ASAP–generated protocols, both MLDSA and MAG benchmarks are fulfilled.

ASAP and conversational rules

- We can reuse the conversational rules discussed earlier, modulo additional roles for participants.

Conclusions and Future Work

- We have used Type Theory with Records (Cooper, 2005) to formulate conversational rules that apply to monologue, dialogue, and multilogue.
- Some rules make no reference to turn exchange (e.g QSpec).
- Others intrinsically do: e.g. acceptance (cf. also clarification requests)
- NSU data indicates locality of most interaction processes.
- QUD potentially used for unbounded resolution.

References

1. Robin Cooper (2004) ‘A type theoretic approach to information state update in issue based dialogue management.’ Talk given at Catalog’04, available from http://www.ling.gu.se/c̃ooper

2. Robin Cooper (2005) ‘Austinian Truth in Martin-Löf Type Theory’, *Research on Language and Computation.*

3. Nicholas Fay, Simon Garrod, and Jean Carletta (2000) ‘Group discussion as interactive dialogue or serial monologue.’ *Psychological Science*, pages 481–486.

4. Raquel Fernández and Jonathan Ginzburg (2002) ‘Non-sentential utterances: A corpus study.’ *Traitement automatique des languages. Dialogue*, 43(2):13–42.

5. FIPA (2003) ‘The foundation for intelligent physical agents. interaction protocol specifications.’ http://www.fipa.org.

6. Jonathan Ginzburg and Ivan A. Sag (2000) *Interrogative Investigations: the form, meaning and use of English Interrogatives.* Number 123 in CSLI Lecture Notes. CSLI Publications, Stanford: California.

7. Jonathan Ginzburg (1996) 'Interrogatives: Questions, facts, and dialogue.' In: Shalom Lappin, editor, *Handbook of Contemporary Semantic Theory.* Blackwell, Oxford.

8. Jonathan Ginzburg (2005) 'Abstraction and Ontology: questions as propositional abstracts in type theory with records'. *Journal of Logic and Computation.*

9. Staffan Larsson (2002) *Issue based Dialogue Management.* Ph.D. thesis, Gothenburg University.

10. Massimo Poesio and David Traum (1997) 'Conversational actions and discourse situations.' *Computational Intelligence 13*(3).

11. David Traum and Jeff Rickel (2002) ‘Embodied agents for multi-party dialogue in immersive virtual world.’ In: *Proceedings of the first International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2002)*, pages 766–773.