Does (Gothenburg) dialogue system technology have anything to say about cognitive systems?

Robin Cooper
Staffan Larsson
Göteborg University

Aarne Ranta
Björn Bringert
Chalmers University

The Department of Linguistics

Contents

1 Implementation and cognition **3**

2 Issue based dialogue management **7**

2.1 Introduction 8

2.2 Accommodation 18

2.3 Feedback 34

3 GF (the Grammatical Framework) **40**

3.1 Multiple languages and grammars 41

3.2 Modalities as additional languages 50

1. Implementation and cognition

- Cognitive modelling
- Engineering only
- Something in between

What does dialogue technology have to do with cognition?

- Dialogue systems interact with humans
- Generic engineering solutions may be theory-based

What kinds of relationship can there be between the technology and cognition?

- Cognitive systems work in the same way as implemented systems (unlikely)
- Implemented systems makes a prediction about behaviour in certain circumstances either in terms of
  - what events will occur (or more reliably, will *not* occur)
  - or, relative cognitive load of events – if there's more to do it might take longer or utilize more of the brain's resources

Suppose our technology makes a correct prediction about cognitive systems – so what?

- A *lack of correlation* tells us that a particular aspect of implementation is *not* cognitively relevant.
- A *correlation* tells us that there *could* be cognitive relevance.
- The more detailed correlations you can demonstrate the closer your technology is to a cognitive model.

2. Issue based dialogue management

2.1. Introduction

Information state update

- Complex structure information states
  - gameboard
  - records
- non-monotonic updates

  e.g. questions disappear from QUD (Questions under discussion)

cf. classical dynamic or update semantics

Issue based dialogue management

- Determination of the next dialogue contribution driven largely by QUD

  almost all contributions raise or address an issue (question under discussion)

- Issues can be raised by addressing them

  giving an answer to a question that hasn't be explicitly stated (question accommodation)

A simple information state type

Larsson, p. 32

$$\begin{bmatrix} \textsc{private} : \begin{bmatrix} \textsc{agenda} : \text{Stack(Action)} \\ \textsc{plan} : \text{Stack(Action)} \\ \textsc{bel} : \text{Set(Prop)} \end{bmatrix} \\ \textsc{shared} : \begin{bmatrix} \textsc{com} : \text{Set(Prop)} \\ \textsc{qud} : \text{Stack(Question)} \\ \textsc{lu} : \begin{bmatrix} \textsc{speaker} : \text{Participant} \\ \textsc{moves} : \text{Set(Move)} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

An update rule

Larsson, p.45

RULE: **integrateUsrQuit**
CLASS: integrate
PRE: { $/SHARED/LU/SPEAKER == usr
in($/SHARED/LU/MOVES, quit)
EFF: { push(/PRIVATE/AGENDA, quit)

CU Dialogue Systems Lab

Sample dialogue plan

```
<       findout(?x.transport(x))
        findout(?x.dest-city(x))
        findout(?x.depart-city(x))
        findout(?x.dept-month(x))
        findout(?x.dept-day(x))
        raise({?class(economy), ?class(business)}
        consultDB(?x.price(x)) >
```

Basic dialogue with updates

U: ”price information please”; raises price issue
- if user asks Q, push respond(Q) on AGENDA
- if respond(Q) on AGENDA and PLAN empty, find plan for Q and load to PLAN
- if findout(Q) first on PLAN, ask Q

S: ”where do you want to go?”

U: ”Paris”
- if LM=answer(A) and A **relevant to** Q, add P=Q[A] to SHARED.COM
- if P in SHARED.COM and Q topmost on QUD and P **resolves** Q, pop QUD
- if P in SHARED.COM and P **fulfils goal** of findout(Q) and findout(Q) on PLAN, pop PLAN

A problem with QUD

- If QUD=<q1,q2> and q1 is resolved, q2 is available for resolution of short answers
  - takes no account of how many turns since q2 was raised
  - but short answers a long distance away from the question are not as easily processed as an adjacent answer

ISSUES and QUD

- We extend Ginzburg's DGB by adding ISSUES of type Stack(Question)
- ISSUES contains all raised but unresolved questions
  - ISSUES determines *relevance* of user answers
- QUD used for resolving short answers
  - questions drop off QUD after N turns
  - a short answer to a question that's on ISSUES but not QUD requires adjusting QUD by copying a question on ISSUES

Typical human-human dialogue

S(alesman), C(ustomer)

S: **hi**

C: **flights to paris**

S: **when do you want to travel?**

C: **april, as cheap as possible**

...

2.2. Accommodation

Accommodation

- Lewis (1979): If someone says something at *t* which requires *X* to be in the conversational scoreboard, and *X* is not in the scoreboard at *t*, then (under certain conditions) *X* will become part of the scoreboard at *t*
- Has been applied to referents and propositions, as parts of the conversational scoreboard / information state

Question accommodation

- If questions are part of the information state, they too can be accommodated
- If the latest move was an answer, and there is an action in the plan to ask a matching question, then
  - put that question on ISSUES
  - (and QUD if it is a short answer)
- Requires that the number of possible matching questions is not too large
  - (or can be narrowed down by asking clarification question)

Sample dialogue – plan accommodation

S: Welcome to the travel agency.

U: From London to Paris in April

- not relevant to any question that has been raised, or to any current task
- look in domain knowledge for a plan (for dealing with some question Q) with matching questions
- load this plan, push Q on ISSUES
- find in the plan the question(s) matching the user's answer
- integrate answer (requres matching question on ISSUES)

S: Alright, you want to know about price.

(…)

- proceed to next plan item

S: How do you want to travel?

- ISSUES=<?x.how(x), ?x.price(x)>

Hypothesis

- Baseline: integrating the answer to a question which has been raised and is uppermost on QUD
- Greater cognitive load when you accommodate questions
- Even greater when you accommodate plans

How do you test?

Some potential problems:

- Distinguishing between accommodated and non-accommodated questions is perhaps not so straightforward. Perhaps explicitly raising one question primes a collection of other questions.
- Thus accommodating some questions may require more work than others.
- There may be a great deal of variation in the cognitive load associated with integrating answers, depending both on the answer and on the question. Some issues may be inherently more complex than others.
- How do you know when a subject has to accommodate a plan rather than a question? We have no way of directly observing a subject's plan.
- Subjects may quickly become habituated to certain issues.

Possible experimental paradigm

(as naively imagined by people with no experimental qualifications...)

- Train the subject to play the role of a simple system by giving them instructions corresponding to GoDiS dialogue plans.

  “If the user wants price information, find out where they want to go, where from, how they want to travel,...”

  “If the user wants visa information find out where they are going, the purpose of their visit, whether they are a student,...”

- The “users” then engage them in dialogue requiring different conditions: no accommodation, question accommodation, plan accommodation

- How long between the user’s utterance (same answer to same question) and the subject’s indication of grounding (e.g. “OK” or simply continuing the dialogue with the next question)?

The three conditions

No accommodation
S: Welcome to the travel agency
U: Price information, please
...
S: Where do you want to go from?
U: Leaving from Heathrow.

Question accommodation
S: Welcome to the travel agency
U: Price information, please
...
S: Where do you want to go to?
U: Paris...leaving from Heathrow

⇐ contents

The three conditions, *contd.*

Plan accommodation
S: Welcome to the travel agency
U: Leaving from Heathrow

Some real data

(Dialogues collected by the Lund group in SDS project. Based on Göteborg transciption following Allwood.)

```
$B (travel agent):  japp ///
& yes?
$A (customer):  eh // jag undrar om eh // / / en
resa till phuket // den tolfte trettonde december //
& er, I wonder if..., er, a trip to Phuket, 12th or
13th December.
```

B has to load a plan and accommodate an appropriate question (issue) which the customer's utterance addresses

⇐ contents

```
$B (travel agent):  ja //
& yes
$A (customer):  [1 fyra ]1 personer /
& four people
$B (travel agent):  [1 är det ]1
& is it...
```

B accommodates: How many people will travel?

```
$B (travel agent):  yes / är det charter du har sett
där eller //
& Right...  is it a charter you saw there or...
$A (customer):  [2 n+ ja:  ]2
& well...  (expressing doubt)
$B (travel agent):  [2 eller vad har ]2 ni tänkt för
någonting /
& ...or what kind of thing have you been thinking
of?
$A (customer):  nej det det / jag tror att det blir
för dyrt med hotell utan eh /
& no, it's...it's, I think it'll be too expensive
with a hotel but, er...
```

B accommodates: Will it be too expensive for a hotel for this trip if it is booked together with the plane?

⇐ contents

```
$B (travel agent):  < / > / ja ni ska bara ha flyget
/
& right, you just want a flight?
```

The question concerning the expense of a package trip is answered by “right” and thus popped off QUD. Confirmed by grounding:

```
$A (customer):  ja /
& yes
$B (travel agent):  okej / [3 m:  ]3
& okay, mm,..
```

No question on QUD.
But customer not entirely satisfied.

```
$A (customer):  [3 ja för ]3 / vi / tror inte du
också det att det blir väldigt eh höga summor om man
ska [4 // ]4
& yes, because...  we...  don't you also think that
it would be very, er, high costs if you're...
```

B has to reaccommodate the question: How expensive will it be for a hotel for this trip if it is booked together with the plane?

- Necessary to *reraise* the question. It cannot be already addressed as a question on QUD, e.g. by saying "yes".
- Reraising is like a second mention, more reduced than original raising

Revision by reraising and reaccommodation

S> What can I do for you?
U> add program channel five
S> Okay. Do you want to add a program?
U> yes
S> Lets see. channel five?
U> yes
S> What date?
U> channel six
S> Okay. channel six… What date?

Hypotheses

- Reraisings are effected by reduced versions of questions/issues which uniquely specify the relevant issue among recently resolved issues.
- Reaccommodation creates less cognitive load than accommodation.

  Alternative: Question accommodation and reaccommodation correspond to about the same cognitive load (since they both require looking for information in the dialogue information state) – less than plan accommodation which involves loading a plan into the information state.

2.3. Feedback

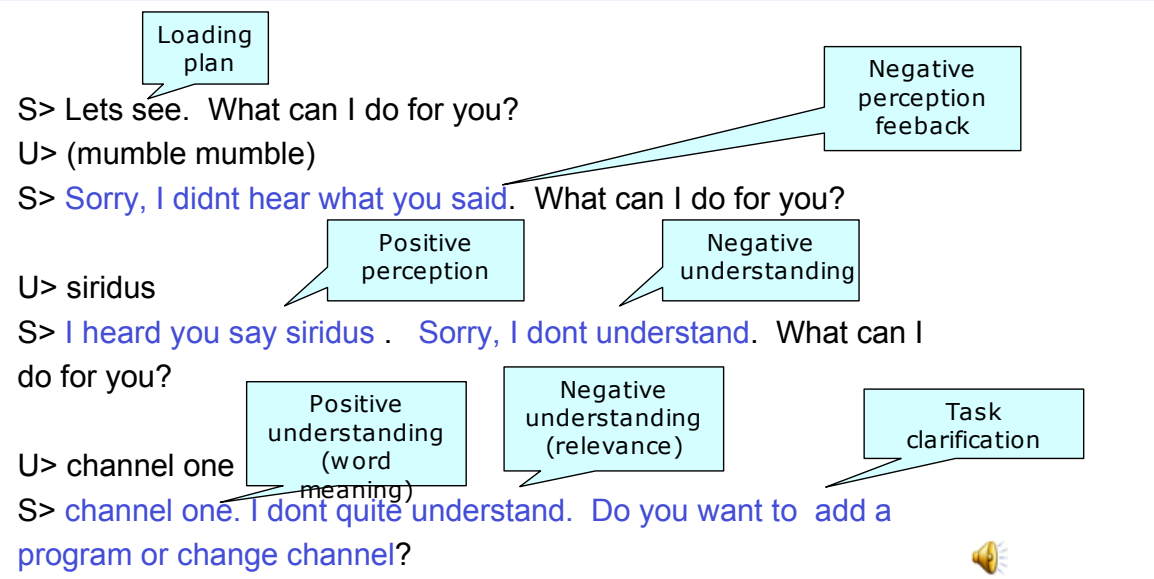
Grounding moves

- We want a typology of grounding moves
  - Both feedback and sequencing
- Some parameters:
  - Action levels in dialogue (Allwood, Clark)
    - contact
    - perception
    - Understanding
      - Word meanings
      - Contextual meaning
    - acceptance
  - Polarity
    - Positive
    - Negative
    - Interrogative (asking for confirmation)

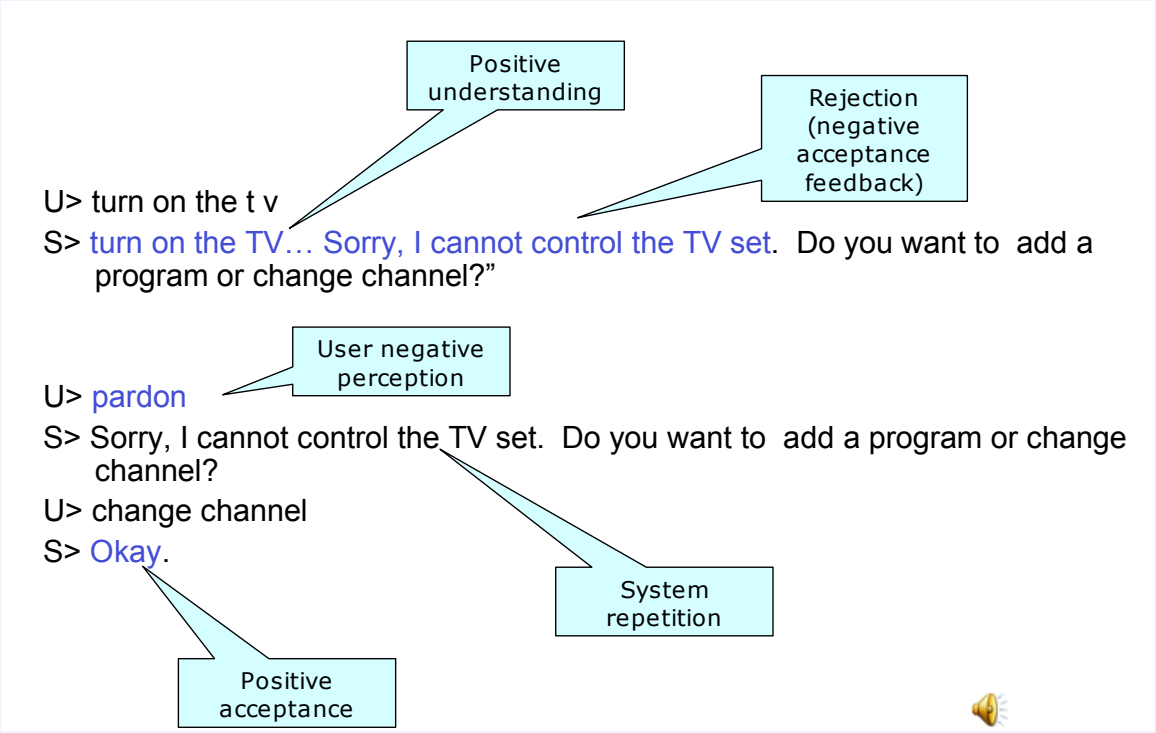
Some grounding moves in GoDiS

- Formal representation
  - icm:Level/Type{*Polarity}{:Content}
- Feedback moves
  - icm:und*neg – ”I don’t understand”
  - icm:und*pos:P – ”To Paris.”
  - icm:acc*neg:Q – ”Sorry, I can’t …”
  - icm:acc*pos – ”Okay”
- Feedback type selected depending on
  - Quality of recognised speech
  - Whether system can find a (relevant) interpretation
  - Whether system can accept what’s been said
- Sequencing moves
  - icm:reraise:Q – ”Returning to the issue of Q”
  - icm:loadplan – ”Let’s see…”

Grounding on multiple levels



Grounding on multiple levels, *contd.*



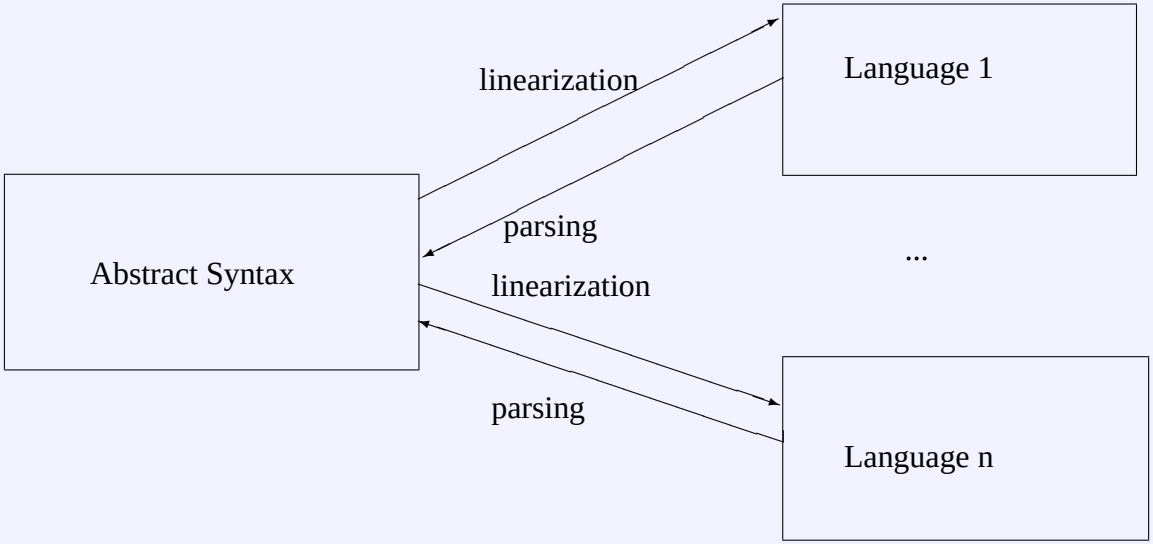
Some questions we would like to know the answer to

- What feedback, if any, is associated with
  - plan loading/accommodation
  - question accommodation
  - reraising/reaccommodation
- How is user feedback coordinated with system utterances?

3. GF (the Grammatical Framework)

3.1. Multiple languages and grammars

Multilingual grammars



Rules

I want to go from x to y.

Abstract:

```
fun GoFromTo : Place -> Place -> Request ;
```

Concrete English:

```
lin GoFromTo x y = {s = ["I want to go from"]
                        ++ x.s ++ "to" ++ y.s} ;
```

Concrete Finnish:

```
lin GoFromTo x y = {s = ["haluan matkustaa"]
                        ++ x.s ! El
                        ++ y.s ! Ill} ;
```

Resource Grammar (English)– Abstract syntax

```
cat S ;   -- sentence
cat CN ;  -- common noun
cat NP ;  -- noun phrase
cat VP ;  -- verb phrase
cat Adj ; -- adjective

fun PredVP  : NP -> VP -> S ;
fun PredAdj : Adj -> VP ;
fun Indef   : CN -> NP ;
```

Resource Grammar (English) – Concrete syntax

```
param Number = Sg | Pl ;
param Person = P1 | P2 | P3 ;

lincat CN = {s : Number => Str} ;
lincat NP = {s : Str ; n : Number ; p : Person} ;
lincat VP = {s : Number => Person => Str} ;

lin PredVP np vp = {s = np.s ++ vp.s ! np.n ! np.p} ;
lin PredAdj adj  = {s = \\n,p => verbBe ! n ! p
                             ++ adj.s} ;
lin Indef cn     = {s = artIndef ++ cn.s ! Sg ;
                    n = Sg ;
                    p = P3} ;
```

Using resource grammar

In English arithmetic: n *is even*

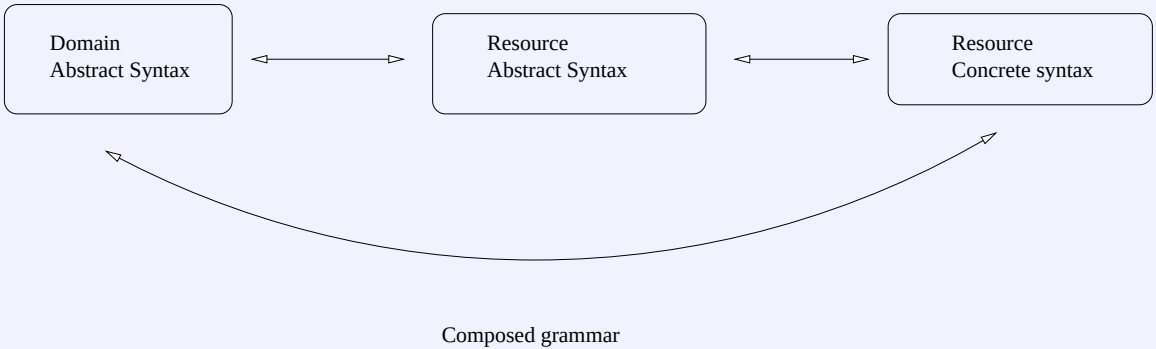
Abstract syntax for arithmetic

```
cat Nat ;
cat Prop ;

fun Zero : Nat ;
fun Even : Nat -> Prop ;
```

Concrete syntax for arithmetic is abstract syntax for resource grammar

```
lincat Prop = S ;
lincat Nat = NP ;

lin Even x = PredVP x (PredAdj adjEven) ;
```

Grammar composition



Languages as frameworks for constructing grammars

- English as a framework for constructing formal languages
- cf. maze game experiments (Garrod and Anderson, 1987)
- note that only part of the resource grammar will be used in a given domain grammar (e.g. lexical ambiguity may disappear)
- the search space for processing becomes much smaller

Multiple grammars in dialogue systems

Potentially:

- interpretation grammar
- generation grammar
- speech recognition grammar (important that not based on whole resource grammar)
- grammars for different languages

Keep these grammars in sync by relating them to the same abstract domain syntax.

3.2. Modalities as additional languages

Modalities as concrete syntaxes

Concrete mouse clicks:

```
lin GoFromTo x y = {s = Click x ++ Click y ++ ClickGo} ;
```

Two senses of multimodality

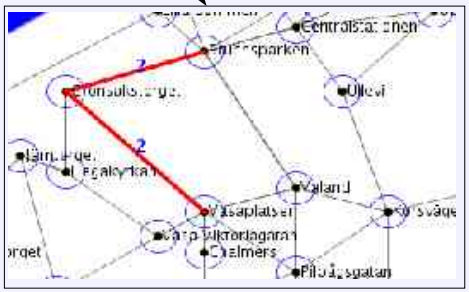
Parallel: grammar for many languages (with or a shared abstract syntax)
As above: parallel concrete syntaxes generating tables, diagrams, animations, etc
Integrated: grammar for a "mixed language" (with code switching)
The concrete syntax combines modalities, e.g. utterances with pointing gestures.
I want to go from here (CLICK) to here (CLICK)

Parallel multimodality

- Complete information in each modality:

Route 2 [Brunnsparken, Grönsakstorget, Vasaplatsen]

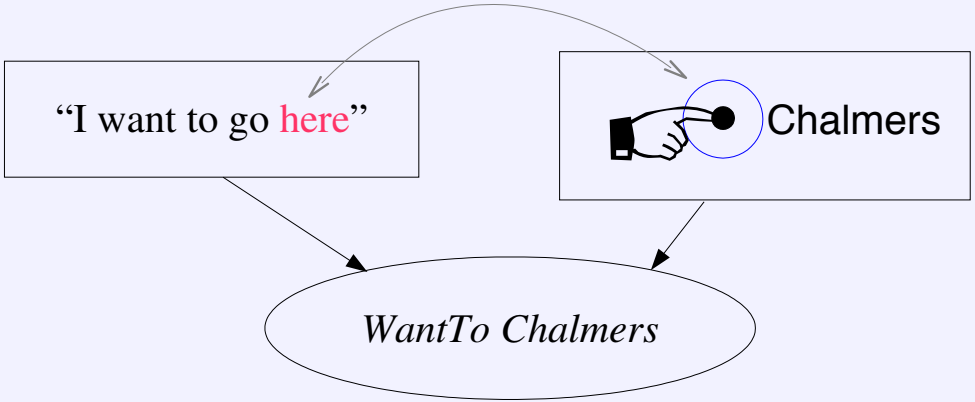
"Take line two from Brunnsparken to Vasaplatsen"



⇐ contents

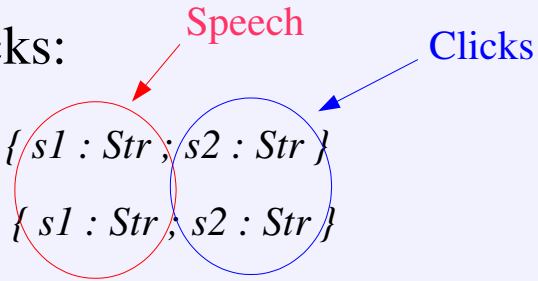
Integrated multimodality

- Information presented by a combination of modalities:



Integrated multimodality in GF

- English + clicks:




```
lincat Input = { s1 : Str ; s2 : Str }

lincat Place = { s1 : Str ; s2 : Str }

lin GoFromTo x y = {
  s1 = ["I want to go from"] ++ x.s1 ++ "to" ++ y.s1;
  s2 = x.s2 ++ y.s2
}

lin NamedPlace p = { s1 = p.s ; s2 = "" }

lin ClickPlace c = { s1 = "here" ; s2 = c.s }
```

Coordination of speech and other modalities

The GF approach – loose coordination (cf. Michael Johnston)

Only the order of clicks matters, not how they are aligned with the utterance.

Hypothesis

- Loose coordination for interaction with artifacts (aynchronous from grammar perspective)
- Tight coordination for gestures (synchronous from grammar perspective, cf prosody)

Conclusions

Possible areas where the concerns of dialogue technology and cognitive systems may intersect:

- accommodation – different cognitive loads for different kinds of accommodation?
- what feedback associated with accommodation?
- how is feedback coordinated with another speaker's utterance?
- natural languages as frameworks for constructing domain/activity specific languages
- loosely and tightly coordinated multimodality