# Dynamic Representation of Conversation in a Dialogue System

Julian Hough, Matthew Purver, Arash Eshghi

Interaction, Media and Communication
School of Electronic Engineering and Computer Science
Queen Mary University of London
www.eecs.qmul.ac.uk/research/imc

The Dynamics of Conversational Dialogue (DynDial)
www.kcl.ac.uk/research/groups/ds

King's College London. March 31st, 2011

## Dialogue is Incremental

**A real dialogue system problem**

| | |
|---|---|
| A: | I want to go to . . . |
| B: | Uh-huh |
| A: | Paris. |
| B: | OK. Let’s see . . . |
| A: | By train. Tomorrow. |

## Dialogue is Incremental

**A real dialogue system problem**

| | |
|---|---|
| A: | I want to go to . . . |
| B: | Uh-huh |
| A: | Paris. |
| B: | OK. Let's see . . . |
| A: | By train. Tomorrow. |

- People don't speak in "complete" sentences - many instances of fragments and ellipsis
  - Nearly 20% of BNC "sentences" continue another "sentence" [Purver et al., 2009]
  - Over 70% continue something already apparently complete
  - Pauses, role changes, backchannels, continuations . . .

## Dialogue is Incremental

**A real dialogue system problem**

|  |  |
|---|---|
| A: | I want to go to . . . |
| B: | Uh-huh |
| A: | Paris. |
| B: | OK. Let's see . . . |
| A: | By train. Tomorrow. |

- People don't speak in "complete" sentences - many instances of fragments and ellipsis
  - Nearly 20% of BNC "sentences" continue another "sentence" [Purver et al., 2009]
  - Over 70% continue something already apparently complete
  - Pauses, role changes, backchannels, continuations . . .
- Computational linguistic processing models have some way to catch up!. . .

## What we need. . .

- An incremental grammar formalism for parsing and generation

## What we need. . .

Dynamic Syntax

- An incremental grammar formalism for parsing and generation
  - *Dynamic Syntax* [Kempson et al., 2001]

## What we need. . .

Dynamic Syntax

- An incremental grammar formalism for parsing and generation
  - *Dynamic Syntax* [Kempson et al., 2001]

- A data structure to interface linguistic processing with domain semantics

## What we need. . .

Dynamic Syntax + TTR

- An incremental grammar formalism for parsing and generation
  - *Dynamic Syntax* [Kempson et al., 2001]

- A data structure to interface linguistic processing with domain semantics
  - *Type Theory with Records (TTR)* [Cooper, 2005]

## What we need. . .

Dynamic Syntax + TTR

- An incremental grammar formalism for parsing and generation
  - *Dynamic Syntax* [Kempson et al., 2001]
- A data structure to interface linguistic processing with domain semantics
  - *Type Theory with Records (TTR)* [Cooper, 2005]
- An incremental dialogue framework

## What we need. . .

Dynamic Syntax + TTR + Jindigo

- An incremental grammar formalism for parsing and generation
  - *Dynamic Syntax* [Kempson et al., 2001]
- A data structure to interface linguistic processing with domain semantics
  - *Type Theory with Records (TTR)* [Cooper, 2005]
- An incremental dialogue framework
  - *Jindigo* [Schlangen and Skantze, 2009]

## Dynamic Syntax: an incremental formalism

- Dynamic Syntax [Kempson et al., 2001]:

## Dynamic Syntax: an incremental formalism

- Dynamic Syntax [Kempson et al., 2001]:
    - an incremental grammar framework
    - word-by-word monotonic growth of semantic representation
    - grammaticality is constraints on construction process
    - bidirectional: generation in terms of parsing
    - one of its principles is *underspecification* and *update*, which make it very good for ellipsis and anaphora resolution
    - recently been used to model split utterance/compound contributions [Purver et al., 2010]

# Dynamic Syntax: an incremental formalism

- Dynamic Syntax [Kempson et al., 2001]:
  - an incremental grammar framework
  - word-by-word monotonic growth of semantic representation
  - grammaticality is constraints on construction process
  - bidirectional: generation in terms of parsing
  - one of its principles is *underspecification* and *update*, which make it very good for ellipsis and anaphora resolution
  - recently been used to model split utterance/compound contributions [Purver et al., 2010]

**Split Turn Taking Puzzle**

| | |
|---|---|
| A: | Did you . . . |
| B: | Burn myself? |

## Dynamic Syntax: an action-based formalism

- Words are represented as *lexical actions* which are packages of tree update operations

## Dynamic Syntax: an action-based formalism

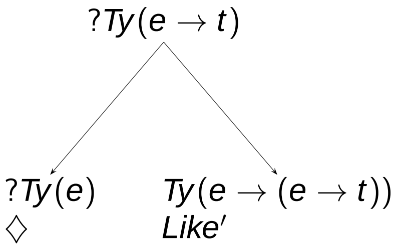
- Words are represented as *lexical actions* which are packages of tree update operations
- e.g. verbs introduce partial propositional templates:

## Dynamic Syntax: an action-based formalism

- Words are represented as *lexical actions* which are packages of tree update operations
- e.g. verbs introduce partial propositional templates:

*like*

```
IF      $?Ty(e \rightarrow t)$
THEN    make($\langle\downarrow_1\rangle$);go($\langle\downarrow\rangle$);
        put($Fo(Like')$,
        $Ty(e \rightarrow (e \rightarrow t))$)
        go($\langle\uparrow_1\rangle$); make($\langle\downarrow_0\rangle$);
        go($\langle\downarrow_0\rangle$); put($?Ty(e)$)
ELSE    ABORT
```

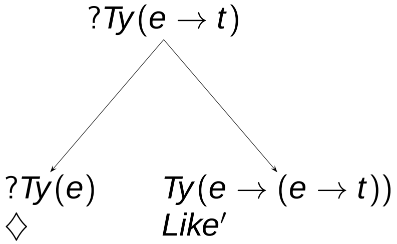


## Dynamic Syntax: an action-based formalism

- Words are represented as *lexical actions* which are packages of tree update operations
- e.g. verbs introduce partial propositional templates:

  *like*

```
IF     $?Ty(e \rightarrow t)$
THEN   make($\langle\downarrow_1\rangle$);go($\langle\downarrow\rangle$);
       put($Fo(Like')$,
       $Ty(e \rightarrow (e \rightarrow t))$)
       go($\langle\uparrow_1\rangle$); make($\langle\downarrow_0\rangle$);
       go($\langle\downarrow_0\rangle$); put($?Ty(e)$)
ELSE   ABORT
```



- *Computational actions* are general rules that can be fired independently of lexical actions. They give DS *predictivity*

## Building a tree incrementally: “John likes Mary”

Processing *John likes Mary*

$$?Ty(t), Tn(0), \diamondsuit$$

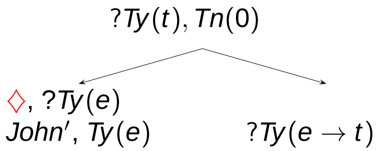
## Building a tree incrementally: “John likes Mary”

Processing *John likes Mary*

$$\begin{tikzcd}
 & ?Ty(t), Tn(0) \arrow[dl] \arrow[dr] & \\
\color{red}{\Diamond}, ?Ty(e) & & ?Ty(e \rightarrow t)
\end{tikzcd}$$

## Building a tree incrementally: “John likes Mary”

Processing *John likes Mary*
‘John



## Building a tree incrementally: "John likes Mary"

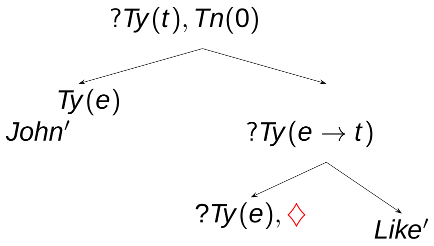
Processing *John likes Mary*
'John

$$\begin{tikzcd}[column sep=small]
 & ?Ty(t), Tn(0) \arrow[dl] \arrow[dr] & \\
\begin{array}{c} Ty(e) \\ John' \end{array} & & ?Ty(e \rightarrow t), \diamondsuit
\end{tikzcd}$$

# Building a tree incrementally: “John likes Mary”

Processing *John likes Mary*

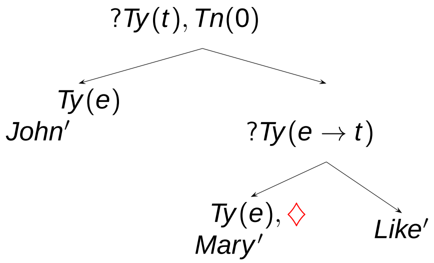
‘John likes



## Building a tree incrementally: “John likes Mary”

Processing *John likes Mary*

‘John likes  Mary’



## Building a tree incrementally: “John likes Mary”

Processing *John likes Mary*
‘John likes  Mary’

$$\begin{tikzcd}
& ?Ty(t), Tn(0) \arrow[dl] \arrow[dr] & & \\
\begin{array}{c} Ty(e) \\ John' \end{array} & & \begin{array}{c} Like'(Mary') \\ ?Ty(e \rightarrow t), \diamondsuit \end{array} \arrow[dl] \arrow[dr] & \\
& \begin{array}{c} Ty(e) \\ Mary' \end{array} & & Like'
\end{tikzcd}$$

## Building a tree incrementally: “John likes Mary”

Processing *John likes Mary*

‘John likes  Mary’

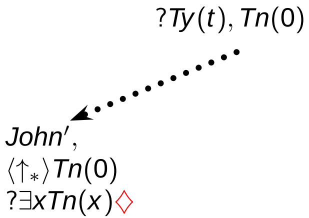
$$\begin{tikzcd}
 & \begin{array}{c} Like'(Mary')(John') \\ Ty(t), Tn(0), \diamondsuit \end{array} \arrow[dl] \arrow[dr] & & \\
\begin{array}{c} Ty(e) \\ John' \end{array} & & \begin{array}{c} Like'(Mary') \\ Ty(e \rightarrow t) \end{array} \arrow[dl] \arrow[dr] & \\
 & \begin{array}{c} Ty(e) \\ Mary' \end{array} & & Like'
\end{tikzcd}$$

## Ambiguity of structure: “John, Mary likes”

$$?Ty(t), Tn(0), \diamondsuit$$

## Ambiguity of structure: “John, Mary likes”

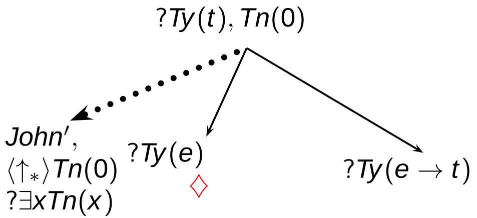
e.g. ‘John



$?Ty(t), Tn(0)$
$John',$
$\langle\uparrow_*\rangle Tn(0)$
$?\exists x Tn(x)\diamondsuit$

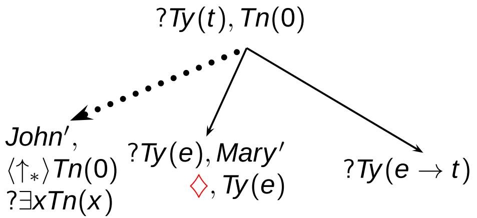
## Ambiguity of structure: “John, Mary likes”

e.g. ‘John



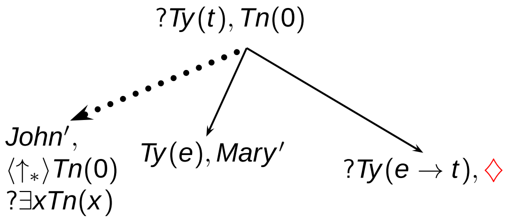
## Ambiguity of structure: “John, Mary likes”

e.g. ‘John, Mary


$?Ty(t), Tn(0)$
$John'$, $\langle\uparrow_*\rangle Tn(0)$ $?\exists x Tn(x)$
$?Ty(e), Mary'$ $\diamondsuit, Ty(e)$
$?Ty(e \rightarrow t)$

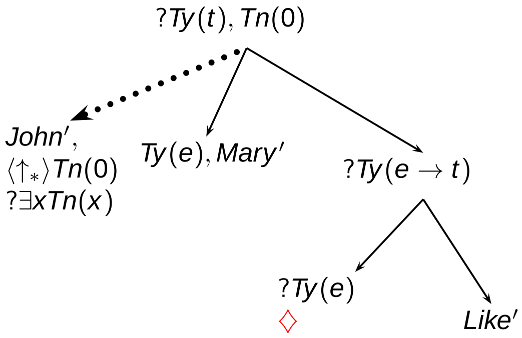
## Ambiguity of structure: “John, Mary likes”

e.g. ‘John, Mary


$?Ty(t), Tn(0)$
$John',$
$\langle\uparrow_*\rangle Tn(0)$
$?\exists x Tn(x)$
$Ty(e), Mary'$
$?Ty(e \rightarrow t), \diamondsuit$

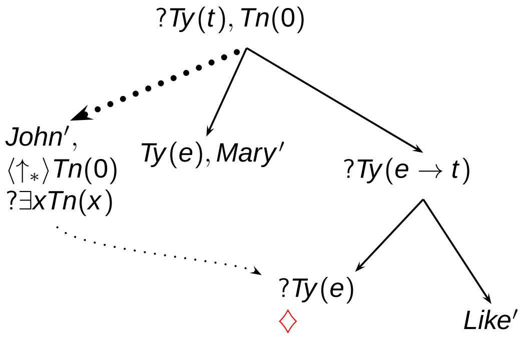
## Ambiguity of structure: “John, Mary likes”

e.g. ‘John, Mary likes’



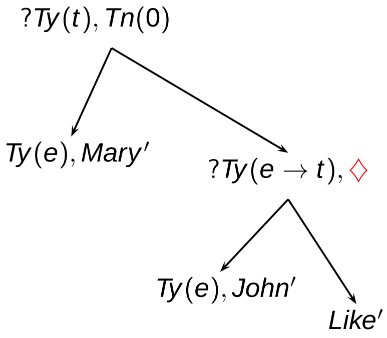
# Ambiguity of structure: “John, Mary likes”

e.g. ‘John, Mary likes’



## Ambiguity of structure: “John, Mary likes”

e.g. ‘John, Mary likes’



## Dynamic Syntax parsing process in the *DyLAN* parser

- For a word $w_i$ and the parser state at step $i$ as a set of partial trees $S_i$:

## Dynamic Syntax parsing process in the *DyLAN* parser

- For a word $w_i$ and the parser state at step $i$ as a set of partial trees $S_i$:

### The parsing process

1. Apply all lexical actions $a_i$ corresponding to $w_i$ to each partial tree in $S_{i-1}$. For each application that succeeds, add the resulting partial tree to $S_i$
2. For each tree in $S_i$, apply all possible sequences of computational actions and add the result to $S_i$

## Dynamic Syntax parsing process in the *DyLAN* parser

- For a word $w_i$ and the parser state at step $i$ as a set of partial trees $S_i$:

**The parsing process**

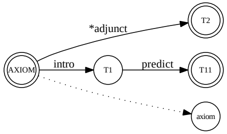
1. Apply all lexical actions $a_i$ corresponding to $w_i$ to each partial tree in $S_{i-1}$. For each application that succeeds, add the resulting partial tree to $S_i$
2. For each tree in $S_i$, apply all possible sequences of computational actions and add the result to $S_i$

- DS parsing can also be seen as a *tree* lattice [Sato, 2010]
  - Nodes = trees
  - Edges = lexical/computational actions

## DS DAG

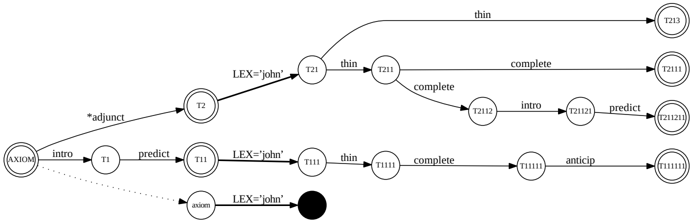
## DS DAG

## DS DAG

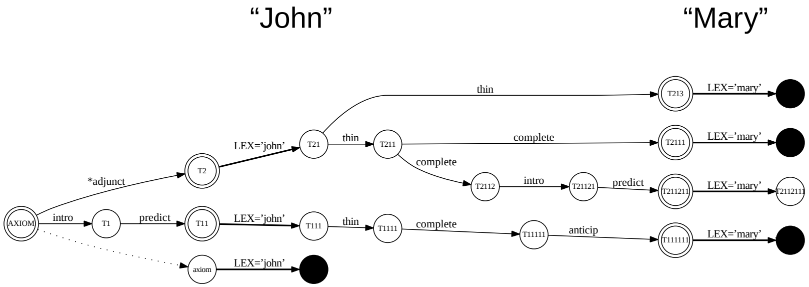


## DS DAG

“John”


thin
T213
LEX=’john’
T21
thin
T211
complete
T2111
complete
T2112
intro
T21121
predict
T211211
*adjunct
T2
AXIOM
intro
T1
predict
T11
LEX=’john’
T111
thin
T1111
complete
T11111
anticip
T111111
axiom
LEX=’john’

# DS DAG



“John”
“Mary”
AXIOM
intro
T1
predict
T11
LEX=’john’
T111
thin
T1111
complete
T11111
anticip
T111111
LEX=’mary’
*adjunct
T2
LEX=’john’
T21
thin
T211
complete
T2111
LEX=’mary’
thin
T213
LEX=’mary’
complete
T2112
intro
T21121
predict
T211211
LEX=’mary’
T2112111
axiom
LEX=’john’

## Context in terms of a DS DAG parse state

## Context in terms of a DS DAG parse state

- Syntactic context can be seen as the path back to the root node (axiom)

## Context in terms of a DS DAG parse state

- Syntactic context can be seen as the path back to the root node (axiom)
- Following this path will give you *trees*, *words* and *actions*

## Context in terms of a DS DAG parse state

- Syntactic context can be seen as the path back to the root node (axiom)
- Following this path will give you *trees*, *words* and *actions*
- Going back in context and “re-running” these actions can allow the parsing of ellipsis
    - “John likes Mary. Bill does too”

## Context in terms of a DS DAG parse state

- Syntactic context can be seen as the path back to the root node (axiom)
- Following this path will give you *trees*, *words* and *actions*
- Going back in context and “re-running” these actions can allow the parsing of ellipsis
  - “John likes Mary. Bill does too”
- Underspecified semantic placeholders can be integrated through backtracking triggers like do-auxilliaries and pronouns

## TTR and DS

## TTR and DS

- Recent work integrating DS with Type Theory with Records (TTR) [Cooper, 2005]

$$\begin{tikzcd}
 & \begin{bmatrix} x : \mathit{john} \\ p : \mathit{leave}(x) \end{bmatrix} \arrow[dl] \arrow[dr] & \\
\begin{bmatrix} x : \mathit{john} \end{bmatrix} & & \lambda \begin{bmatrix} x : e \end{bmatrix} . \begin{bmatrix} x : e \\ p : \mathit{leave}(x) \end{bmatrix}
\end{tikzcd}$$

## TTR and DS

- Recent work integrating DS with Type Theory with Records (TTR) [Cooper, 2005]

$$\begin{tikzcd}
& \left[\begin{array}{lcl} x & : & \mathit{john} \\ p & : & \mathit{leave}(x) \end{array}\right] \arrow[dl] \arrow[dr] & \\
\left[\begin{array}{lcl} x & : & \mathit{john} \end{array}\right] & & \lambda\left[\begin{array}{lcl} x & : & e \end{array}\right].\left[\begin{array}{lcl} x & : & e \\ p & : & \mathit{leave}(x) \end{array}\right]
\end{tikzcd}$$

- TTR *record types* provide the semantic content of each node of the DS trees

## TTR and DS

- Recent work integrating DS with Type Theory with Records (TTR) [Cooper, 2005]

$$\begin{tikzcd}
& \left[\begin{array}{lcl} x & : & \mathit{john} \\ p & : & \mathit{leave}(x) \end{array}\right] \arrow[dl] \arrow[dr] & \\
\left[\begin{array}{lcl} x & : & \mathit{john} \end{array}\right] & & \lambda\left[\begin{array}{lcl} x & : & e \end{array}\right].\left[\begin{array}{lcl} x & : & e \\ p & : & \mathit{leave}(x) \end{array}\right]
\end{tikzcd}$$

- TTR *record types* provide the semantic content of each node of the DS trees
- LINKed trees for adjunction are easily incorporated by extending *record types*

# TTR and DS

- Recent work integrating DS with Type Theory with Records (TTR) [Cooper, 2005]

$$\begin{tikzcd}
 & \left[\begin{array}{lcl} x & : & john \\ p & : & leave(x) \end{array}\right] \arrow[dl] \arrow[dr] & \\
\left[\begin{array}{lcl} x & : & john \end{array}\right] & & \lambda\left[\begin{array}{lcl} x & : & e \end{array}\right].\left[\begin{array}{lcl} x & : & e \\ p & : & leave(x) \end{array}\right]
\end{tikzcd}$$

- TTR *record types* provide the semantic content of each node of the DS trees
- LINKed trees for adjunction are easily incorporated by extending *record types*
- Recently, a Davidsonian [Davidson, 1980] event-based semantics for tense has been incorporated [Cann, 2010]

## Incremental Semantic Construction with TTR

- Using TTR we can get incrementally constructed *record types* from our trees:

## Incremental Semantic Construction with TTR

- Using TTR we can get incrementally constructed *record types* from our trees:

I want to go . . .

$$\left[\begin{array}{lcl} e = \mathit{now} & : & e_s \\ e1 = \mathit{future} & : & e_s \\ x = \mathit{speaker} & : & e \\ p1 = \mathit{go}(e1, x) & : & t \\ p = \mathit{want}(e, x, p1) & : & t \end{array}\right] \quad \mathit{Trip} :$$

## Incremental Semantic Construction with TTR

- Using TTR we can get incrementally constructed *record types* from our trees:

I want to go to Paris

...

$$\begin{bmatrix} e = \textit{now} & : & e_s \\ e1 = \textit{future} & : & e_s \\ x1 = \textit{Paris} & : & e \\ p2 = \textit{to}(e1, x1) & : & t \\ x = \textit{speaker} & : & e \\ p1 = \textit{go}(e1, x) & : & t \\ p = \textit{want}(e, x, p1) & : & t \end{bmatrix}$$

$\textit{Trip}$ :
$\textit{to} = \textit{paris}$

## Incremental Semantic Construction with TTR

- Using TTR we can get incrementally constructed *record types* from our trees:

I want to go to Paris from London . . .

$$\begin{bmatrix} e = \mathit{now} & : & e_s \\ e1 = \mathit{future} & : & e_s \\ x1 = \mathit{Paris} & : & e \\ p2 = \mathit{to}(e1, x1) & : & t \\ x2 = \mathit{London} & : & e \\ p3 = \mathit{from}(e1, x2) & : & t \\ x = \mathit{speaker} & : & e \\ p1 = \mathit{go}(e1, x) & : & t \\ p = \mathit{want}(e, x, p1) & : & t \end{bmatrix}$$

*Trip* :
$to = paris$
$from = london$

# Incremental Semantic Construction with TTR

- Using TTR we can get incrementally constructed *record types* from our trees:

I want to go to Paris from London . . .

$$\left[\begin{array}{lcl} e = \mathit{now} & : & e_s \\ e1 = \mathit{future} & : & e_s \\ x1 = \mathit{Paris} & : & e \\ p2 = \mathit{to}(e1, x1) & : & t \\ x2 = \mathit{London} & : & e \\ p3 = \mathit{from}(e1, x2) & : & t \\ x = \mathit{speaker} & : & e \\ p1 = \mathit{go}(e1, x) & : & t \\ p = \mathit{want}(e, x, p1) & : & t \end{array}\right]$$

$\mathit{Trip}$ :
$\mathit{to} = \mathit{paris}$
$\mathit{from} = \mathit{london}$

- Provides a nice interface between Dynamic Syntax $\leftrightarrow$ domain semantic frames

## Jindigo: An incremental dialogue system

- [Schlangen and Skantze, 2009] have introduced *Jindigo*, a flexible incremental dialogue system framework

## Jindigo: An incremental dialogue system

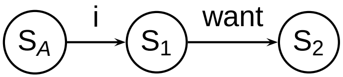
- [Schlangen and Skantze, 2009] have introduced *Jindigo*, a flexible incremental dialogue system framework
- Modular, with *incremental units* being passed from one to another, notion of *commitment*

## Jindigo: An incremental dialogue system

- [Schlangen and Skantze, 2009] have introduced *Jindigo*, a flexible incremental dialogue system framework
- Modular, with *incremental units* being passed from one to another, notion of *commitment*
- Parsing, generation and dialogue management being currently worked on [Buß et al., 2010, Schlangen et al., 2010, Skantze and Hjalmarsson, 2010]

## Jindigo: An incremental dialogue system

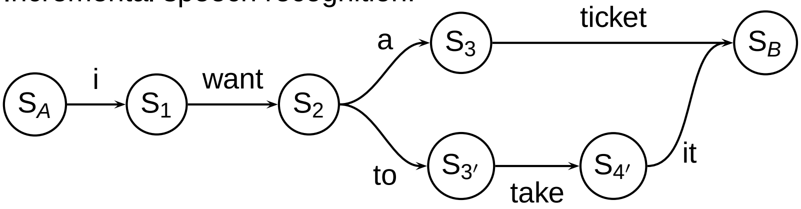
- [Schlangen and Skantze, 2009] have introduced *Jindigo*, a flexible incremental dialogue system framework
- Modular, with *incremental units* being passed from one to another, notion of *commitment*
- Parsing, generation and dialogue management being currently worked on [Buß et al., 2010, Schlangen et al., 2010, Skantze and Hjalmarsson, 2010]
- Incremental speech recognition:



$$\begin{tikzcd} S_A \arrow[r, "i"] & S_1 \arrow[r, "want"] & S_2 \end{tikzcd}$$

## Jindigo: An incremental dialogue system

- [Schlangen and Skantze, 2009] have introduced *Jindigo*, a flexible incremental dialogue system framework
- Modular, with *incremental units* being passed from one to another, notion of *commitment*
- Parsing, generation and dialogue management being currently worked on [Buß et al., 2010, Schlangen et al., 2010, Skantze and Hjalmarsson, 2010]
- Incremental speech recognition:



## Jindigo: An incremental dialogue system

- [Schlangen and Skantze, 2009] have introduced *Jindigo*, a flexible incremental dialogue system framework
- Modular, with *incremental units* being passed from one to another, notion of *commitment*
- Parsing, generation and dialogue management being currently worked on [Buß et al., 2010, Schlangen et al., 2010, Skantze and Hjalmarsson, 2010]
- Incremental speech recognition:

$$\begin{tikzcd}
 & & & S_3 \arrow[rr, "\text{ticket}"] & & S_B \\
S_A \arrow[r, "\text{i}"] & S_1 \arrow[r, "\text{want}"] & S_2 \arrow[ur, bend left, "\text{a}"] \arrow[dr, bend right, "\text{to}"'] & & & \\
 & & & S_{3'} \arrow[r, "\text{take}"'] & S_{4'} \arrow[uur, bend right, "\text{it}"'] &
\end{tikzcd}$$

- A DS DAG could interface with this?...

## Jindigo: incoporating incremental semantics

- With purely phonological accounts of incremental input processing, mid-utterance backchannels, unfinished utterances become possible in micro domains

## Jindigo: incoporating incremental semantics

- With purely phonological accounts of incremental input processing, mid-utterance backchannels, unfinished utterances become possible in micro domains
- But utterance meaning treated non-incrementally:
  - A standard dialogue systems approach of one move per utterance, fragment resolution mechanisms
  - Not much in the way of *semantics*

## Jindigo: incoporating incremental semantics

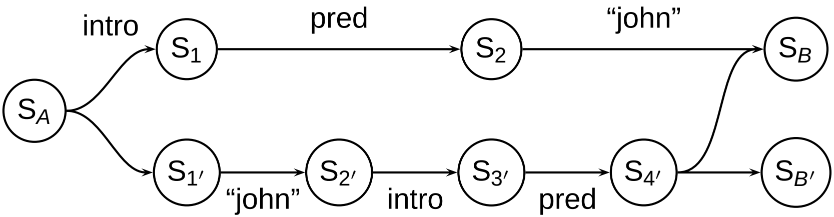
- With purely phonological accounts of incremental input processing, mid-utterance backchannels, unfinished utterances become possible in micro domains
- But utterance meaning treated non-incrementally:
    - A standard dialogue systems approach of one move per utterance, fragment resolution mechanisms
    - Not much in the way of *semantics*
- A *domain-general* incremental semantics is needed for various dialogue phenomena

## Putting voice recognition and DS parsing together

- DS $\leftrightarrow$ ASR in Jindigo
  - Incremental *word* lattice subsumes finer grained incremental *parse* lattice
  - “Big” word hypothesis edges from the ASR subsume the “thin” lexical/computational action edges from parsing

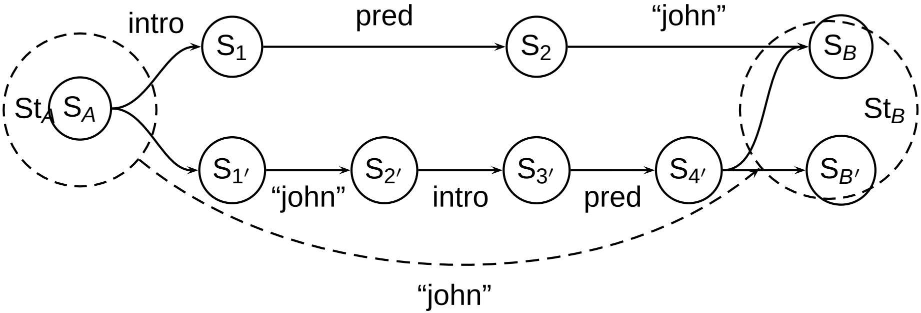
## Putting voice recognition and DS parsing together

- DS $\leftrightarrow$ ASR in Jindigo
    - Incremental *word* lattice subsumes finer grained incremental *parse* lattice
    - “Big” word hypothesis edges from the ASR subsume the “thin” lexical/computational action edges from parsing



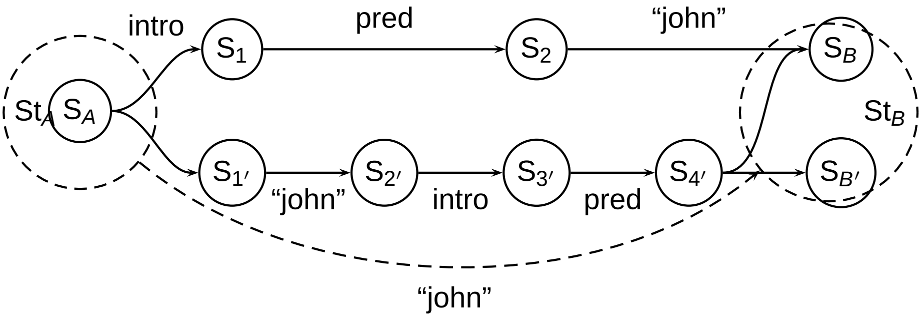
# Putting voice recognition and DS parsing together

- DS $\leftrightarrow$ ASR in Jindigo
  - Incremental *word* lattice subsumes finer grained incremental *parse* lattice
  - “Big” word hypothesis edges from the ASR subsume the “thin” lexical/computational action edges from parsing



## Putting voice recognition and DS parsing together

- DS $\leftrightarrow$ ASR in Jindigo
  - Incremental *word* lattice subsumes finer grained incremental *parse* lattice
  - “Big” word hypothesis edges from the ASR subsume the “thin” lexical/computational action edges from parsing



- The best parse hypothesis will be *committed* when it is *grounded in* a committed ASR hypothesis

## DS and TTR for domain concepts in Jindigo

## DS and TTR for domain concepts in Jindigo

- Currently *complete* DS trees have TTR representations

## DS and TTR for domain concepts in Jindigo

- Currently *complete* DS trees have TTR representations
- Work on making the TTR representation completely incremental has begun

## DS and TTR for domain concepts in Jindigo

- Currently *complete* DS trees have TTR representations
- Work on making the TTR representation completely incremental has begun
- These TTR representations are matched to domain concept frames(e.g. `Trip(to:City[Paris])`). Another level of semantic incrementality

## DS and TTR for domain concepts in Jindigo

- Currently *complete* DS trees have TTR representations
- Work on making the TTR representation completely incremental has begun
- These TTR representations are matched to domain concept frames(e.g. `Trip(to:City[Paris])`). Another level of semantic incrementality
- When concept frames are matched successfully, they are *committed* to the output buffer

## DS and TTR for domain concepts in Jindigo

- Currently *complete* DS trees have TTR representations
- Work on making the TTR representation completely incremental has begun
- These TTR representations are matched to domain concept frames(e.g. `Trip(to:City[Paris])`). Another level of semantic incrementality
- When concept frames are matched successfully, they are *committed* to the output buffer
- Extending the record types through LINK adjunction in DS is straightforward

## DS and TTR for domain concepts in Jindigo

- Currently *complete* DS trees have TTR representations
- Work on making the TTR representation completely incremental has begun
- These TTR representations are matched to domain concept frames(e.g. `Trip(to:City[Paris])`). Another level of semantic incrementality
- When concept frames are matched successfully, they are *committed* to the output buffer
- Extending the record types through LINK adjunction in DS is straightforward
- The parse state is maintained, so new trees and new record types can be introduced and replace a revoked domain frame concept

## Generation

## Generation

- Bidirectional quality of DS. Work is being done on developing the system's *generation* (NLG) module
  - recent incremental generation work is being done in terms of speech plans [Skantze and Hjalmarsson, 2010]
  - not yet in terms of online syntactic/semantic construction during generation

## Generation

- Bidirectional quality of DS. Work is being done on developing the system's *generation* (NLG) module
    - recent incremental generation work is being done in terms of speech plans [Skantze and Hjalmarsson, 2010]
    - not yet in terms of online syntactic/semantic construction during generation
- The DS Generation process [Purver and Kempson, 2004] uses the same action-based mechanism as parsing, but with a *goal tree*
    - each parse state is checked and trees kept which subsume the goal, successful lexical action = generated word
    - As the generator and parser can have access to the same parse state lattice, split utterances/compound contributions should follow straightforwardly according to the [Purver et al., 2010] account

## Generation

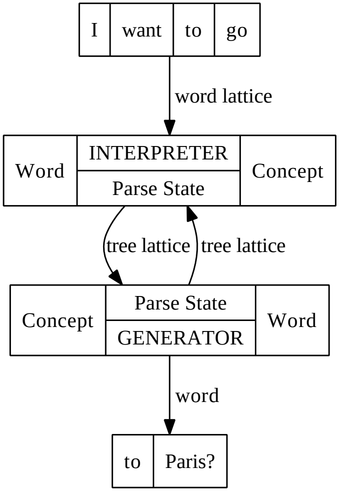
- Bidirectional quality of DS. Work is being done on developing the system's *generation* (NLG) module
  - recent incremental generation work is being done in terms of speech plans [Skantze and Hjalmarsson, 2010]
  - not yet in terms of online syntactic/semantic construction during generation
- The DS Generation process [Purver and Kempson, 2004] uses the same action-based mechanism as parsing, but with a *goal tree*
  - each parse state is checked and trees kept which subsume the goal, successful lexical action = generated word
  - As the generator and parser can have access to the same parse state lattice, split utterances/compound contributions should follow straightforwardly according to the [Purver et al., 2010] account
- This is work in progress!

## Module interaction: sharing tree lattices

- Tree lattice “parse state” part of generation process, so can be shared between modules. . .

## Module interaction: sharing tree lattices

- Tree lattice "parse state" part of generation process, so can be shared between modules. . .



## Jindigo demo

[Jindigo demo]

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

  And because

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

And because this is such

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

  And because                this is such
  this is for television

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

| | |
|---|---|
| And because | this is such |
| this is for television | it’s a |

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

| | |
|---|---|
| And because | this is such |
| this is for television | it's a |
| | we have a |

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

| | | |
|---|---|---|
| And because | this is such | |
| this is for television | it’s a | |
| | we have a | market range of Interna... |

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

| | | |
|---|---|---|
| And because | this is such | |
| this is for television | it’s a | |
| | we have a | market range of Interna... |
| | like it’s an | International Market Range |

## Future work: repair simulation

- Simulating error phenomena such as self-repair and hesitation should be possible

|  |  |  |
|---|---|---|
| And because | this is such |  |
| this is for television | it's a |  |
|  | we have a | market range of Interna... |
|  | like it's an | International Market Range |

  - the incremental goal tree subsumption checking of the DS generation process [Purver and Kempson, 2004]
  - repair strategy: if a new goal tree from a dialogue manager does not subsume the current one, *backtrack* through the context DAG until a tree is found where subsumption does occur and then start generating again from there
  - error causes: possible information flow deadlocks between jindigo modules

## Thanks for listening!

Thanks to:

Ruth Kempson, Pat Healey, Christine Howes,

Graham White, Eleni Gregoromichelaki, Yo Sato

Buß, O., Baumann, T., and Schlangen, D. (2010).
Collaborating on utterances with a spoken dialogue system using an ISU-based approach to incremental dialogue management.
In *Proceedings of the SIGDIAL 2010 Conference*, pages 233–236, Tokyo, Japan. Association for Computational Linguistics.

Cann, R. (2010).
Towards an account of the english auxiliary system.
In Gregoromichelaki, E., Kempson, R., and Howes, C., editors, *The Dynamics of Lexical Interfaces*. CSLI. to appear.

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

Davidson, D. (1980).
*Essays on Actions and Events*.
Clarendon Press, Oxford, UK.

Kempson, R., Meyer-Viol, W., and Gabbay, D. (2001).
*Dynamic Syntax: The Flow of Language Understanding*.
Blackwell.

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

Purver, M., Howes, C., Gregoromichelaki, E., and Healey, P. G. T. (2009).
Split utterances in dialogue: a corpus study.
In *Proceedings of the 10th Annual SIGDIAL Meeting on Discourse and Dialogue (SIGDIAL 2009 Conference)*, pages 262–271, London, UK. Association for Computational Linguistics.

Purver, M. and Kempson, R. (2004).

Incremental context-based generation for dialogue.
In Belz, A., Evans, R., and Piwek, P., editors, *Proceedings of the 3rd International Conference on Natural Language Generation (INLG04)*, number 3123 in Lecture Notes in Artifical Intelligence, pages 151–160, Brockenhurst, UK. Springer.

Sato, Y. (2010).
Local ambiguity, search strategies and parsing in Dynamic Syntax.
In Gregoromichelaki, E., Kempson, R., and Howes, C., editors, *The Dynamics of Lexical Interfaces*. CSLI. to appear.

Schlangen, D., Baumann, T., Buschmeier, H., Buß, O., Kopp, S., Skantze, G., and Yaghoubzadeh, R. (2010).
Middleware for incremental processing in conversational agents.
In *Proceedings of the SIGDIAL 2010 Conference*, pages 51–54, Tokyo, Japan. Association for Computational Linguistics.

Schlangen, D. and Skantze, G. (2009).
A general, abstract model of incremental dialogue processing.
In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 710–718, Athens, Greece. Association for Computational Linguistics.

Skantze, G. and Hjalmarsson, A. (2010).
Towards incremental speech generation in dialogue systems.
In *Proceedings of the SIGDIAL 2010 Conference*, pages 1–8, Tokyo, Japan. Association for Computational Linguistics.