Building Spoken Dialogue Systems for Embodied Agents

Johan Bos

School of Informatics

The University of Edinburgh

Overview of the Course

- Why do we need/want spoken dialogue with a robot?
  - Directing,
  - Information retrieval,
  - Learning
- What is involved in enabling a (spoken) dialogue with an embodied agent (for instance a robot)?
  - understanding natural language and acting in natural language
  - Dialogue management and engagement

Outline of the course

- Part I: Natural Language Processing
  - Practical: designing a grammar for a fragment of English in a robot domain
- Part II: Inference and Interpretation
  - Practical: extending the Curt system
- Part III: Dialogue and Engagement

Contents of the Reader

- Blackburn & Bos Chapters 1,2, and 6
- Bos, Klein & Oka (EACL)
- Bugmann et al. (IBL)
- Lemon et al. (Witas system)
- Bos & Oka (Coling)
- Sidner (engagement)
- Larsson & Traum (information state)
- Bos, Klein, Lemon & Oka (DIPPER)

Today

- Some examples of dialogue with mobile robots
- Global overview of Natural Language Processing
- Speech Recognition
  - How to create a simple application using off-the-shelf software
  - More advanced methods

Example 1: Dialogue with a Mobile Robot

- Integrated Dialogue and Navigation System
- Investigate use of natural language to help with navigation problems
- System Requirements
  - Communication in spoken unrestricted English
  - Everyday usage of language
  - Combination of knowledge resources
- Ontological information, semantic representation of dialogue, inference

Interesting Language Use: Natural Descriptions

- Not:
  - *Go to grid cell 45,77!*
  - *You're in region 12.*
- But:
  - *Go to* ***Tim's office****!*
  - *You're in* ***the corridor leading to the emergency exit****.*

Interesting Language Use: Use of Pronouns

- Not:
  - *The box is in the kitchen.*
  - *Go to the kitchen and take the box.*
- But:
  - *The box is in the kitchen.*
  - *Go **there** and take **it**.*

Interesting Language Use: Quantification

- Not:
  - *Clean the kitchen.*
  - Clean the bathroom.
  - Clean the hallway.
- But:
  - *Clean every room on the first floor*

Interesting Language Use: Explaining how to do things

- *U: Go to the kitchen*
- *R: How to I go to the kitchen?*
- *U: Follow the corridor until you reach a door on your right hand side. Go through the door and you are in the kitchen*

Dialogue with Mobile Robots

- Most research on spoken dialogue based on interacting with virtual agents
- Interesting challenges and opportunities when interlocutor is a physically embodied mobile agent
  - Talk about physical environment
  - Get good indicator of dialogue success
  - Symbol Grounding
- Opens up a new vista for human-computer interaction
- Example: overview of the robot **Godot**

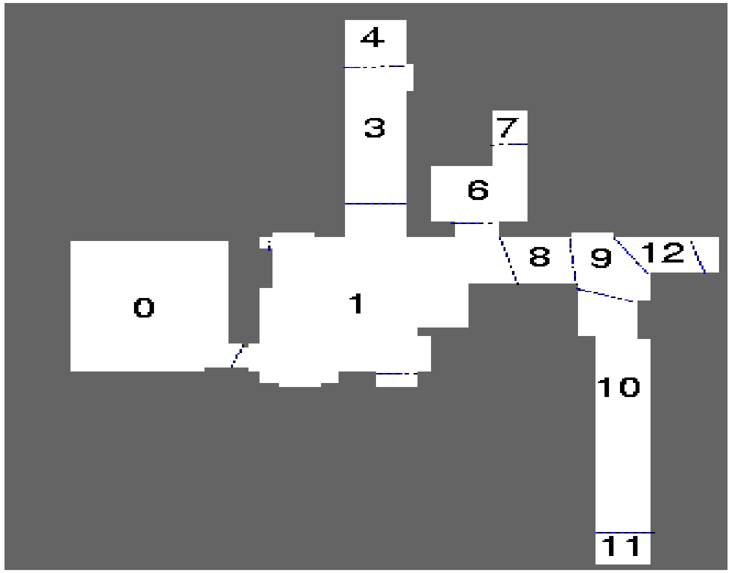
Godot – the robot

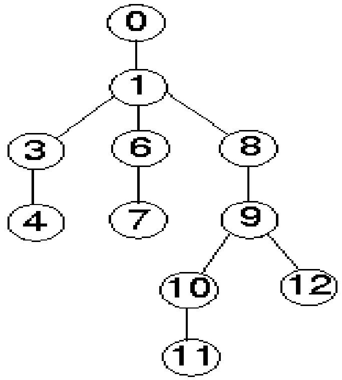
- RWI Magellan Pro mobile robot platform
- Onboard PC running Linux
- Connected via wireless LAN
- Sensors:
  - 16 sonar (occupied space)
  - 16 infrared (distance)
  - 16 collision detectors
- CCD camera on pan-tilt unit
- Shaft encoders (odometry)

The Internal Map (1/2)

- Godot moves about in the basement of our department
- Internal map with two layers
  - geometrical layer: occupancy grid to represent occupied and free space
  - topological layer: automatically constructed using Voronoi diagram decomposition
- Semantic labels attached to regions of topological layer

The Internal Map (2/2)





- Numbers in the map are identifiers of topological regions
- Use these to associate semantic representations with regions

The Navigation Module

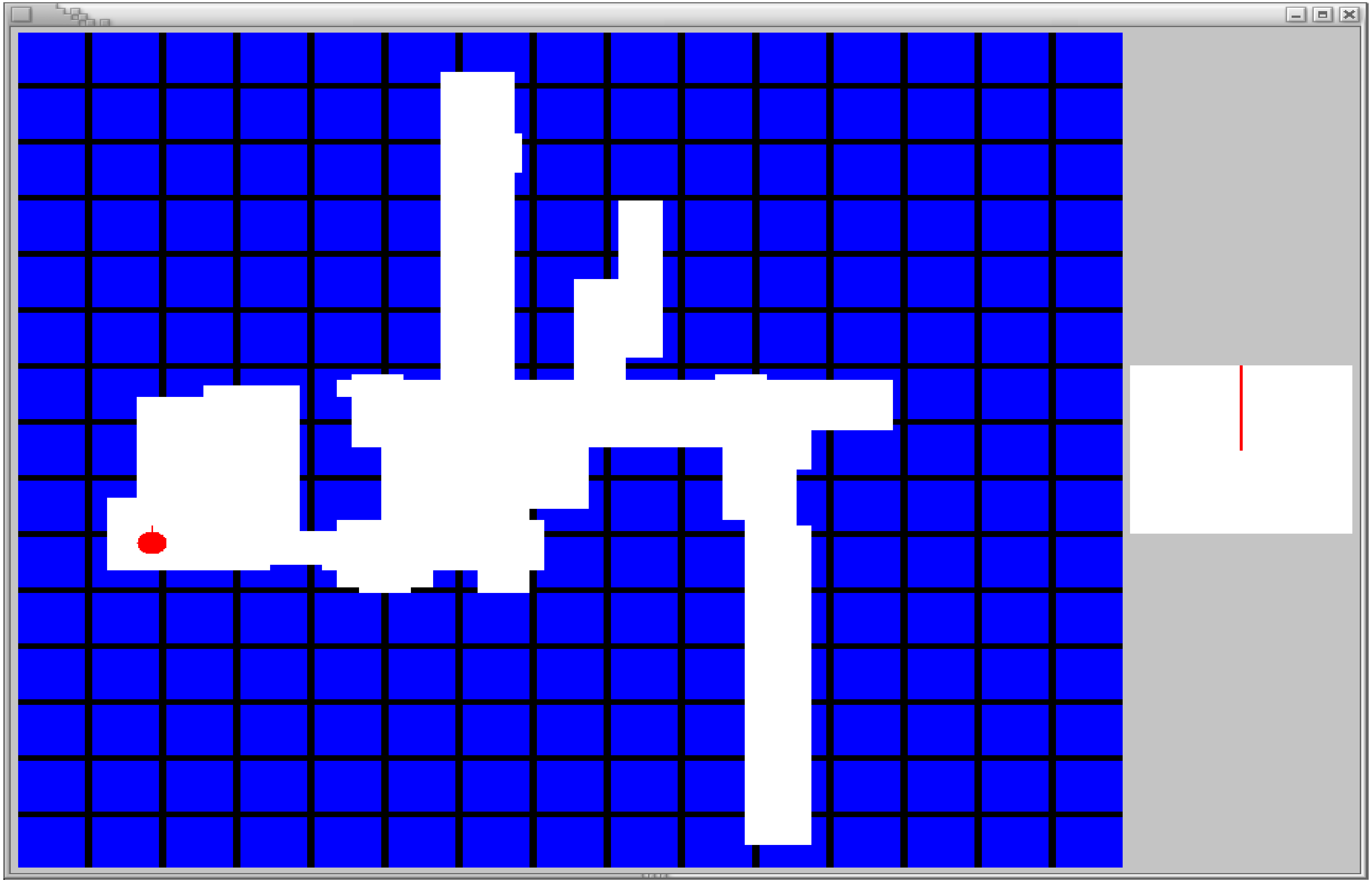
- Loops by reading sensory input and executing motor commands at regular intervals
- Sensory input:
  - Sonars, infrared, odometry
- Motor commands triggered by sensor readings or dialogue
- Topological map used to compute shortest path

Robot primitives

- Behaviour triggered by last command from dialogue system
- Commands are mapped into primitives
- Examples of primitives:
  - move_to_region(Region-Id)
  - look(Pan,Tilt)
  - turn(Angle,Speed)
  - set_region(Region-Id)
- Commands in execution can be interrupted
- Memory: Stack of commands

Image Viewer

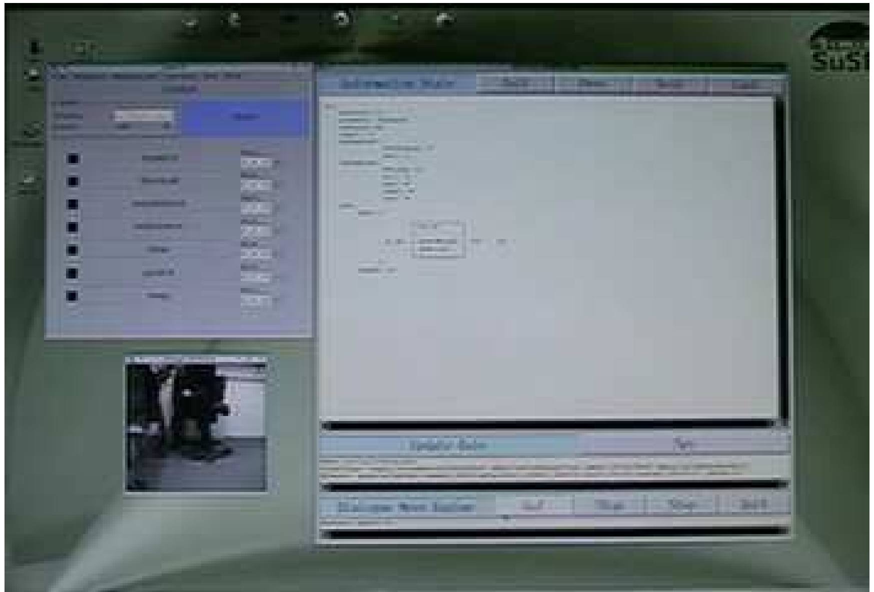
The Map Viewer



Running the System

Selected Agents

Dialogue Move Engine



Camera View

Interaction between Dialogue and Navigation Component

- Updating Occupancy Grid (use of negation)
  - U: *You're not in the kitchen.*
- Assigning and refining labels to regions in the cognitive map (informativeness)
  - U: *You're in an office.*
  - U: *This is Tim's office.*
- Position Clarification (disjunction)
  - R: *Is this the kitchen or the living room?*
- Arguments (inconsistency)
  - U: *You're in the kitchen*.
  - R: *No, I am not in the kitchen!*

Example 2:
Greta, the talking head

- Face-to-face spoken dialogue
- Combining verbal and non-verbal signals
- Express emotions, synchronise lip and facial movements (eyebrows, gaze) with speech
- Festival synthesiser

Natural Language Processing

- 1. Speech Recognition
- 2. Parsing (Syntactic Analysis)
- 3. Semantic Analysis
- 4. Dialogue Modelling
- 5. Generation
- 6. Synthesis

Ambiguities in Natural Language

- Ambiguities in NL expressions allow different interpretations (or meanings)
- Various knowledge sources help to disambiguate phrases (context, grammar, intonation, common-sense knowledge)
- There are many phenomena that can give rise to ambiguities

1. Speech Recognition

- Task: Mapping acoustic signals into symbolic representations
- Use commercial SR software (Nuance)
- Language modelling/domain modelling
- Microphone placement
- Speaker recognition/verification

Speech Ambiguities

- Mapping from acoustic signals to words not always unambiguous
- Listen for instance to:
  - I saw 26 swans

Speech Ambiguities

- Mapping from acoustic signals to words not always unambiguous!
- Listen for instance to:
  - I saw 26 swans
- Or was it:
  - I saw 20 sick swans
  - I saw 26 once
  - I saw 20 sick ones
  - …. And so on…!

2. Parsing

- Task: Assigning syntactic structure to a string of words.
- This will help to build a logical form.
- Structures are mostly represented as trees or graphs, were nodes denote syntactic categories or lexical items
- Grammar and Lexicon required

Background: lexical categories

- Det: determiner (*a, the, every, most*)
- N: noun (*man, car, hammer, cup*)
- PN: proper name (*Vincent, Mia, Butch*)
- TV: transitive verb (*saw, clean*)
- IV: intransitive verb (*smoke, go*)
- Prep: preposition (*at, in, about*)

Background: grammatical categories

- NP: noun phrase (*the man*)
- VP: verb phrase (*saw the car*)
- PP: prepositional phrase (*at the corner*)
- S: sentence (*Vincent cleans a car*)

Background: grammar rules

$S \rightarrow NP\ VP$

$NP \rightarrow Det\ N$

$NP \rightarrow PN$

$VP \rightarrow IV$

$VP \rightarrow TV\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow Prep\ NP$

$PN \rightarrow$ jules

$PN \rightarrow$ vincent

$Det \rightarrow$ every

$Det \rightarrow$ a

$N \rightarrow$ man

$N \rightarrow$ woman

$N \rightarrow$ milkshake

$N \rightarrow$ car

$IV \rightarrow$ walks

$IV \rightarrow$ talks

$TV \rightarrow$ loves

$TV \rightarrow$ likes

$TV \rightarrow$ drinks

$Prep \rightarrow$ in

$Prep \rightarrow$ with

Lexical Ambiguities

- Time flies like an arrow
  - [NP:time,VP:flies like an arrow]
  - [VP:[TV:time,NP:flies,PP:like an arrow]]
- Fruit flies like a banana
  - [NP:fruit flies,VP:like a banana]
  - [NP:fruit,[VP:flies,PP:like a banana]]

Attachment Ambiguities

- Attachment of the prepositional phrase of "*with a telescope*":
  - *I saw the boy with a telescope*.

- What did you see and how?
  - **[vp:[vp:[tv:saw,np:the boy],pp:with a telescope]]**
  - **[vp:[tv:saw,np:[np:the boy,pp:with a telescope]]]**

3. Semantic Analysis

- Task: Building a logical form – this will help us to interpret the utterance
- Human language contains a lot of ambiguities when taking out of context
- Need to deal with ambiguity resolution!
  - Scope ambiguities
  - Anaphoric/reference ambiguities

Scope Ambiguities

- Relative scope assignments of "*every week*" and "*a cyclist*":
  - *Every week a cyclist is hit by a bus in Edinburgh*.
  - *He doesn't appreciate it very much*.

- Structurally different semantic representations:
  - $\forall x(\mathbf{week}(x) \rightarrow \exists y(\mathbf{cyclist}(y) \& \ldots\ldots))$
  - $\exists y(\mathbf{cyclist}(y) \& \forall x(\mathbf{week}(x) \rightarrow \ldots\ldots))$

Anaphoric Ambiguities

- Relational noun "*part*" (implicitly anaphoric)
  - Tim: *Where were you born?*
  - Kim: *America.*
  - Tim: *Which part?*
  - Kim: *All of me, of course.*

- Different Semantic Representations:
  - **…(part(x,y)&y=america)…**
  - **…(part(x,y)&y=kim)...**

4. Dialogue Modelling

- Analysing user’s move, deciding system’s move (planning)
- Speech acts (assert, query, request)
- Initiating clarification dialogues
- Back-channelling, giving feedback
- Showing awareness
- Engagement

5. Text Generation

- Task: mapping structured information to a string of words
- How to say things
  - use of referring expressions
  - choice of words
  - prosody
- Templates vs. “deep” processing

Information Structure and Prosody

- Example 1:
  - Q: Who went to the party?
  - A: **Vincent** went to the party.
  - A: * Vincent went to the **party**.
- Example 2:
  - Q: What did Vincent do?
  - A: * **Vincent** went to the party.
  - A: Vincent went to the **party**.

[Star * marks ungrammatical answers]

6. Synthesis

- Task: converting a string of words to an sound file
- Use off-the-shelf software (Festival)
- Pre-recorded vs. Synthesised
- Use of talking heads (Greta)
- Prosody, emotion

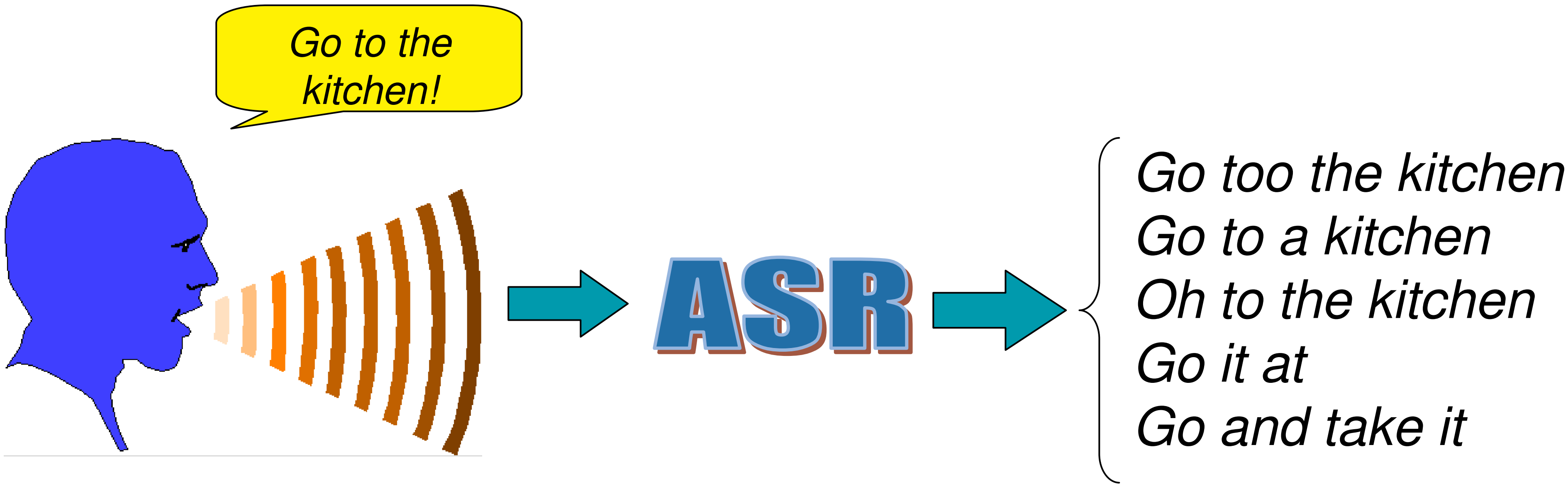
Outline of the rest of this lecture

- We will take a closer look at:
  - Speech recognition
  - Grammar engineering
- Tomorrow:
  - semantics, inference, dialogue, engagement

Automatic speech recognition for Robots

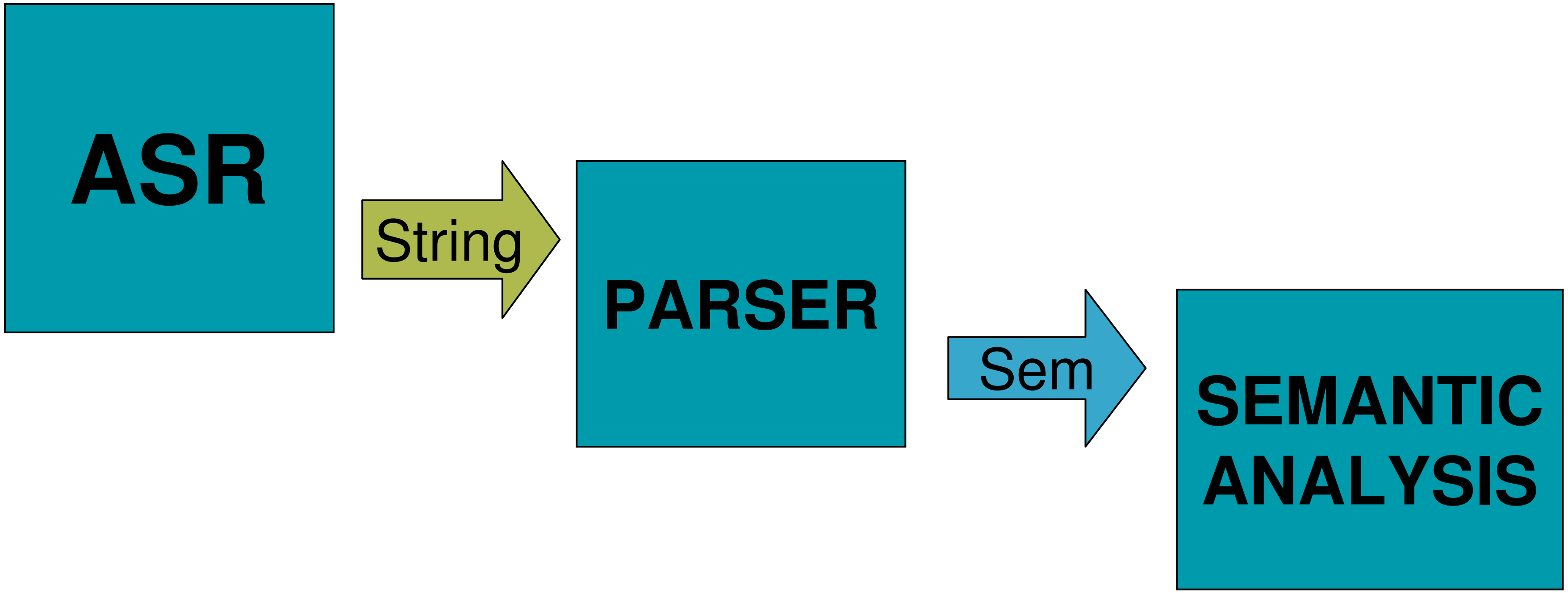
- Automatic Speech Recognition (ASR)
- How to build a simple recognition package (incl. demo)
- How to add features for natural language understanding (incl. demo)
- Why this is not a good approach
- How we can do better
  - Linguistically-motivated grammars
  - Demo of UNIANCE

Automatic Speech Recognition



- ASR output is a lattice or a set of strings
- Many non-grammatical productions
- Use parser to select string and produce logical form for interpretation

The basic pipeline for natural language understanding in speech applications



Automatic Speech Recognition

- The words an ASR can recognize are limited and mostly tuned to a particular application
- Build a speech recognition package:
  - pronunciations of the words
  - acoustic model
  - language model
    - Grammar-based
    - Statistical model

Language Models

- Statistical Language Models (bigrams)
  - Bad: need a large corpus
  - Bad: non-grammatical output possible
  - Good: relatively high accuracy (low WER)
- Grammar-based Language Models
  - Good: no large corpus required
  - Good: output always grammatical
  - Bad: lack of robustness
- In this talk we will explore grammar-based approaches

An Example: NUANCE

- The NUANCE speech recognizer supports the Grammar Specification Language (GSL)
  - lowercase symbols: terminals
  - uppercase symbols: non-terminals
  - [X…Y] : disjunction
  - (X…Y) : conjunction
- Suppose we want to cover the following kind of expressions
  - Go to the kitchen/hallway/bedroom
  - Turn left/right
  - Enter the first/second door on your left/right

Example GSL Grammar

```
Command
[ (go to the Location)
  (turn Direction)
  (enter the Ordinal door on your Direction)]

Location
[ kitchen hallway (dining room) ]

Direction
[ left right ]
```

Natural Language Understanding

- We don’t just want a string of words from the recogniser!
- It would be nice if we could associate a semantic interpretation to a string
- Preferably a logical form of some kind
- Nuance GSL offers slot-filling
- Other methods (post-processing) are of course also possible

Interpretation: adding slots

```
Command
[ (go to the Location:a) {<destination $a>}
  (turn Direction:b) {<rotate $b>}
  (enter the Ordinal:c door on your
  Direction:d) {<door $c> <position $d>} ]

Location
[ kitchen {return(kitchen)}
  hallway {return(hallway)}
  (dining room) {return(diningroom)} ]
```

Demo of Nuance

GSL & NUANCE

- Good:
  - allows tuning to a particular application in a convenient way
- Bad:
  - Tedious to build for serious applications and difficult to maintain
  - Limited expressive power
  - Slot-filling not a serious semantics (compositional semantics preferred)

How to improve on this…

- Use a **linguistic grammar** as starting point (what's the idea behind this?)
- We will use a **unification grammar** (UG) which works with phrase structure rules
- Use a **generic semantics** in the UG
- **Compile** UG into GSL,
- and Bob is your uncle!

Example of a Linguistically-motivated Grammar

$S \rightarrow NP\ VP$

$NP \rightarrow Det\ N$

$NP \rightarrow PN$

$VP \rightarrow IV$

$VP \rightarrow TV\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow Prep\ NP$

PN → jules

PN → vincent

Det → every

Det → a

N → man

N → woman

N → milkshake

N → car

IV → walks

IV → talks

TV → loves

TV → likes

TV → drinks

Prep → in

Prep → with

What I mean by ‘Compositional Semantics’

- Semantic operations based on lambda calculus, e.g.:
  - S → NP VP (without semantics)
  - S:α(β) → NP:α VP:β (with semantics)
- Functional application and beta-conversion (no unification)
- Independent of syntactic formalism

Grammar with Compositional Semantics

$S{:}\alpha(\beta) \rightarrow NP{:}\alpha\ VP{:}\beta$

$NP{:}\alpha(\beta) \rightarrow Det{:}\alpha\ N{:}\beta$

$NP{:}\alpha \rightarrow PN{:}\alpha$

$VP{:}\alpha \rightarrow IV{:}\alpha$

$VP{:}\alpha(\beta) \rightarrow TV{:}\alpha\ NP{:}\beta$

PN: $\lambda p.p(vincent) \rightarrow$ vincent

N: $\lambda x.milkshake(x) \rightarrow$ milkshake

Det: $\lambda p.\lambda q.\forall x(p(x) \Rightarrow q(x)) \rightarrow$ every

Det: $\lambda p.\lambda q.\exists x(p(x) \wedge q(x)) \rightarrow$ a

IV: $\lambda x.walk(x) \rightarrow$ walks

TV: $\lambda u.\lambda x.u(\lambda y.love(x,y)) \rightarrow$ loves

Background: The Lambda Calculus

- Lexical semantics:

  “Vincent”: $\lambda p.p(vincent)$

  “walks”: $\lambda x.walk(x)$

- Functional Application:

  “Vincent walks”: $\lambda p.p(vincent)(\lambda x.walk(x))$

- Beta-Conversion:

  $\lambda p.p(vincent)(\lambda x.walk(x))$ =

  $\lambda x.walk(x)(vincent)$ =

  walk(vincent)

Example of a Unification Grammar we work with

```
[cat:np,case:_,num:N,per:3,refl:no,sem:apply(X,Y)] --> [cat:det,count:C,num:N,sem:X],
                                                        [cat:noun,count:C,num:N,sem:Y].

[cat:noun,count:yes,num:sg,sem:lambda(X,radio(X))] --> [lex:radio].

[cat:noun,count:yes,num:sg,sem:lambda(X,car(X))] --> [lex:car].
```

- Mostly atomic feature values, untyped
- Range of values extensionally determined
- Complex features for traces
- Feature `sem` to hold semantic representation
- Semantic representations are expressed as Prolog terms

Idea: compile Unification Grammar into NUANCE GSL

- Create a context-free backbone of the UG
- Use syntactic features in the translation to non-terminal symbols in GSL
- Previous Work:
  - Rayner et al. 2000, 2001
  - Dowding et al. 2001 (typed unification grammar)
  - Kiefer & Krieger 2000 (HPSG)
  - Moore (2000)
- Previous work does not concern semantics
- UNIANCE compiler (Sicstus Prolog)

Compilation Steps (UNIANCE)

- *Input*: UG rules and lexicon
- Feature Instantiation
- Redundancy Elimination
- Packing and Compression
- Left Recursion Elimination
- Incorporating Compositional Semantics
- *Output*: rules in GSL format

Feature Instantiation

- Create a context-free backbone of the unification grammar
- Collect range of feature values by traversing grammar and lexical rules (for features with a finite number of possible values)
- Disregard Feature **SEM**
- Result is set of rules of the form $\mathbf{C_0} \rightarrow \mathbf{C_1 \ldots C_n}$

where $\mathbf{C_i}$ has structure **cat(A,F,X)** with

A a category symbol,

F a set of instantiated feature value pairs,

X the semantic representation

Eliminating Redundant Rules

- Rules might be redundant with respect to application domain
  - (or grammar might be ill-formed)
- Two reasons for a production to be redundant:
  - A non-terminal member of a RHS does not appear in a production as LHS
  - A LHS category (not the beginner) does not appear as RHS member
- Remove such rules until fixed point is reached

Packing and Compression

- Pack together rules that share LHSs
- Compress productions by replacing a set of rules with the same RHS by a single production:
  - Replace pair $C_i \rightarrow \mathit{C}$ and $C_j \rightarrow \mathit{C}$ ($i \neq j$) by $C_k \rightarrow \mathit{C}$ (C_k a new category)
  - Substitute C_k for all occurrences of C_i and C_j in the grammar

Eliminating Left Recursion

- Left-recursive rules are common in linguistically motivated grammars
- GSL does not allow LR
- Standard way of eliminating LR
  - Aho et al. 1996, Greibach Normal Form
  - Here we only consider immediate left-recursion
- Replace pairs of A→AB, A→C
  by A→CA’, A’→BA’ and A’→ε
- Put differently: …
  by A→CA’, A’→BA’, A→C and A’→B

Incorporating Compositional Semantics

- At this stage we have a set of rules of the form LHS $\rightarrow$ ***C***, where ***C*** is a set of ordered pairs of RHS categories and corresponding semantic values
- Convert LHS and RHS to GSL categories (straightforward)
- Bookkeeping required to associate semantic variables with GSL slots
- Semantic operations are composed using the built-in `strcat/2` function

Example (Input UG)

$$\texttt{S}\langle\texttt{sem:X(Y(}\lambda e.present(e)))\rangle \rightarrow \texttt{NP}\langle\texttt{sem:X}\rangle\ \texttt{VP}\langle\texttt{sem:Y}\rangle$$

$$\texttt{NP}\langle\texttt{sem:}\lambda P.\exists x[person(x)\&P(x)]\rangle \rightarrow \texttt{somebody}$$

$$\texttt{NP}\langle\texttt{sem:}\lambda P.P(hearer)\rangle \rightarrow \texttt{you}$$

$$\texttt{VP}\langle\texttt{sem:X(Y)}\rangle \rightarrow \texttt{VP}\langle\texttt{sem:Y}\rangle\ \texttt{AV}\langle\texttt{sem:X}\rangle$$

$$\texttt{VP}\langle\texttt{sem:X}\rangle \rightarrow \texttt{IV}\langle\texttt{sem:X}\rangle$$

$$\texttt{IV}\langle\texttt{sem:}\lambda P.\lambda x.\exists e[walk(e)\&[agent(e,x)\&P(e)]]\rangle \rightarrow \texttt{walk}$$

$$\texttt{IV}\langle\texttt{sem:}\lambda P.\lambda x.\exists e[walk(e)\&[agent(e,x)\&P(e)]]\rangle \rightarrow \texttt{walks}$$

$$\texttt{AV}\langle\texttt{sem:}\lambda E.\lambda P.E(\lambda e.[home(e)\&P(e)])\rangle \rightarrow \texttt{home}$$

$$\texttt{AV}\langle\texttt{sem:}\lambda E.\lambda P.E(\lambda e.[quick(e)\&P(e)])\rangle \rightarrow \texttt{quickly}$$

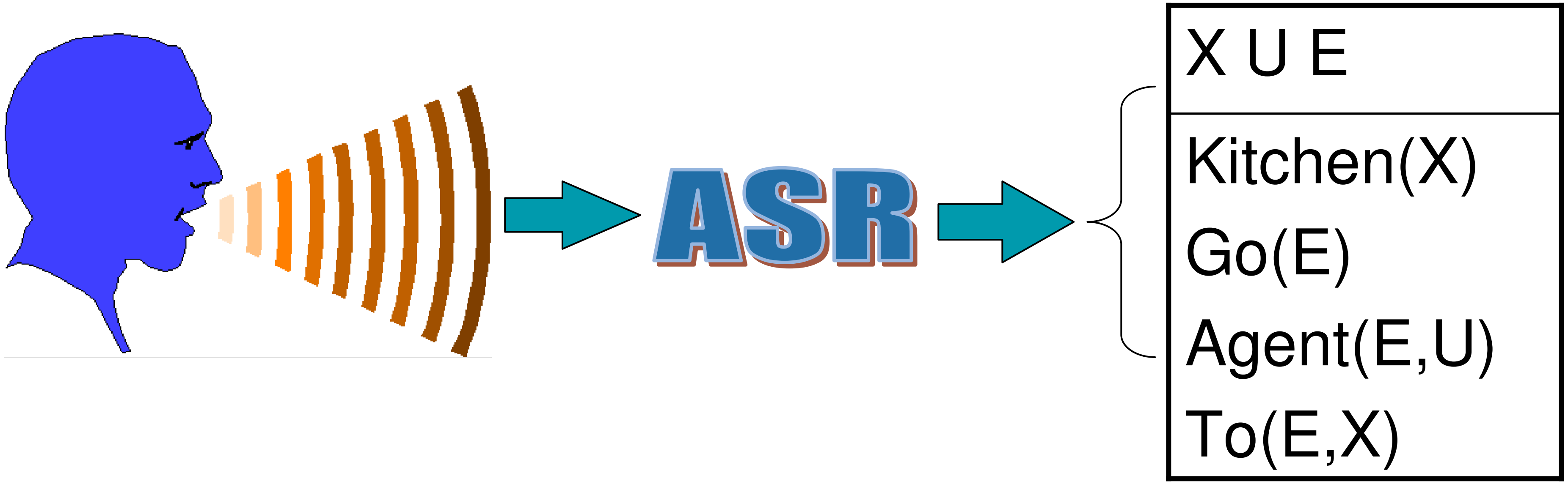
Example (GSL Output)

```
.U [S:sem {<sem strcat($sem "."))>}]

Av
[( quickly ) {return( "l(A,l(B,a(A,l(C,and(quick(C),a(B,C))))))")}
 ( home ) {return( "l(D,l(E,a(D,l(F,and(home(F),a(E,F))))))")}]

IvNumsgPer2
[( walk )
 {return( "l(A,l(B,exists(C,and(walk(C),and(agent(C,B),a(A,C))))))")}]

IvNumsgPer3
[( walks )
 {return( "l(A,l(B,exists(C,and(walk(C),and(agent(C,B),a(A,C))))))")}]

NpCasenomNumsgPer2
[( you ) {return( "l(A,a(A,hearer))")}]

NpCasenomNumsgPer3
[( somebody ) {return( "l(A,exists(B,and(person(B),a(A,B))))")}]

S
[( NpCasenomNumsgPer3:a  VpNumsgPer3:b )
 {return(strcat("a(" strcat(strcat($a strcat("," strcat("a("
          strcat(strcat($b strcat("," strcat("l(C," strcat("present(C)"
          ")")))) ")")))) ")")))}

 ( NpCasenomNumsgPer2:a  VpNumsgPer2:b )
 {return(strcat("a(" strcat(strcat($a strcat("," strcat("a("
          strcat(strcat($b strcat("," strcat("l(F," strcat("present(F)"
          ")")))) ")")))) ")")))}]

VpNumsgPer2
[( IvNumsgPer2:a ) {return($a)}
 ( IvNumsgPer2:a  RecNewcatvpNumsgPer2:b )
 {return(strcat("a(" strcat(strcat($b strcat("," $a)) ")")))}]

VpNumsgPer3
[( IvNumsgPer3:a ) {return($a)}
 ( IvNumsgPer3:a  RecNewcatvpNumsgPer3:b )
 {return(strcat("a(" strcat(strcat($b strcat("," $a)) ")")))}]
```

Example (Nuance Output)

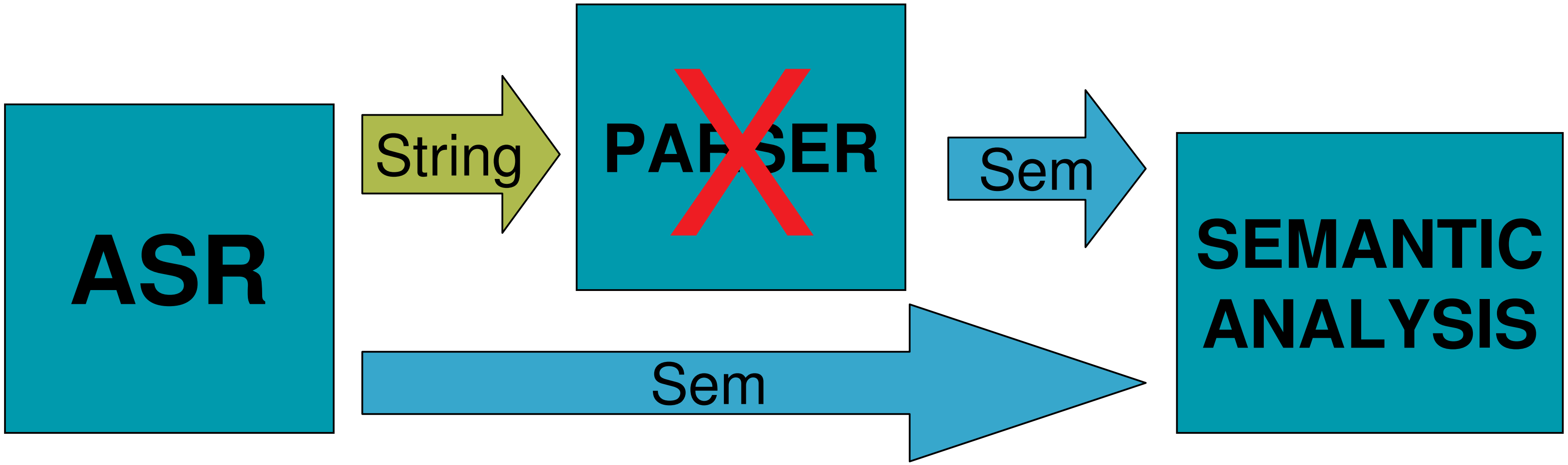
```
File 2:   u1_GA_MD_2.wav
Grammar: .Dialogue
Transcript.: you turn the second road on the left hand side
Result #0-0  you turn the second road on the left hand side (conf: 47, NL conf: )
you turn the second road on the left hand side
you turn the second road on the left hand side
Rec Errors:   0 ins,   0 del,   0 sub,   10 Words :Word Error Rate = 0.00
Rec Total:    0 ins,   0 del,   0 sub,   15 Words: Word Error Rate = 0.00
Precision:  100.00
Recall:     100.00   Reject:  0.00
Fmeasure:   100.00
NL Res.#0
sem
assert(a(l(A,l(B,l(C,m(udr([],[D],[E],[C:alfa(E,dei,drs([E],[system(E)]),D)],[leq(C,B)]),a(a(a(A
,E),B),D))))),a(a(l(A,l(B,l(C,a(A,l(D,l(E,l(F,m(udr([],[G],[H],[F:merge(drs([H],[event(H),agent(
H,C),patient(H,D),nonreflexive(H),turn(H)]),G)],[leq(F,E)]),a(a(a(B,H),E),G)))))))))),a(l(A,l(B,l
(C,l(D,m(m(udr([],[E,F],[G],[D:alfa(G,def,merge(drs([G],[]),E),F)],[leq(D,C)]),a(a(a(A,G),C),E))
,a(a(a(B,G),C),F)))))),a(l(A,l(B,l(C,l(D,m(udr([],[E,F,G],[H,I],[D:merge(drs([H],[group(H),secon
d(B,H),neg(E)]),G),E:drs([I],[neg(drs([],[equiv(drs([],[member(I,H)]),F)]))])],[leq(D,C)]),m(a(a
(a(A,I),C),F),a(a(a(A,B),C),G))))))),a(l(A,a(a(l(B,l(C,l(D,l(E,l(F,m(udr([G],[H,I],[],[F:merge(H
,G)],[leq(I,G)]),m(a(a(a(C,D),E),H),a(a(a(B,D),E),I)))))))),a(a(l(A,l(B,l(C,l(D,l(E,a(a(a(l(F,a(
B,a(A,F))),C),D),E)))))),l(A,l(B,l(C,l(D,udr([],[],[],[D:drs([],[on(A,B)])],[leq(D,C)]))))),a(l
(A,l(B,l(C,l(D,m(m(udr([],[E,F],[G],[D:alfa(G,def,merge(drs([G],[]),E),F)],[leq(D,C)]),a(a(a(A,G
),C),E)),a(a(a(B,G),C),F)))))),a(l(A,l(B,l(C,l(D,m(udr([],[E],[],[D:merge(E,drs([],[lefthand(B)]
))],[leq(D,C)]),a(a(a(A,B),C),E)))))),l(A,l(B,l(C,udr([],[],[],[C:drs([],[side(A)])],[leq(C,B)])
))))))),A)),l(A,l(B,l(C,udr([],[],[],[C:drs([],[road(A)])],[leq(C,B)])))))))),l(C,l(D,l(E,udr([]
,[],[],[E:drs([],[present(C)])],[leq(E,D)])))))))
```

Automatic speech recognition with our new approach



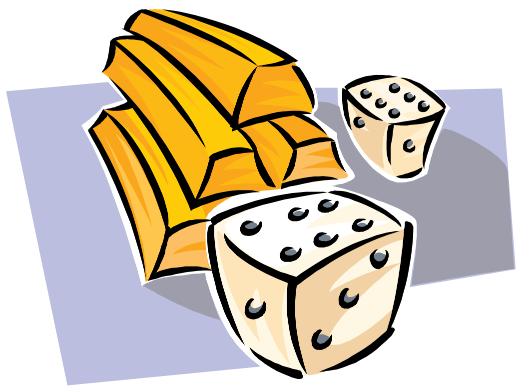
- Put compositional semantics in language models
- ASR output comprises logical forms (e.g., a DRS)
- No need for subsequent parsing

This is nice because it makes the parser redundant



Further Improvements: Adding Probabilities to GSL

- Include probabilities to increase recognition accuracy
- Done by bootstrapping GSL grammar:
  - Use first version of GSL to parse a domain specific corpus
  - Create table with syntactic constructions and frequencies
  - Choose closest attachment in case of structural ambiguities
  - Add obtained probabilities to original GSL grammar



Practical: Grammar Engineering

- Collect a (small) corpus of your choice
- Assign syntactic categories to the words appearing in the corpus and create a lexicon
- Define a grammar covering the utterances of your corpus
- Implement and test everything using the Prolog program **lambda.pl**