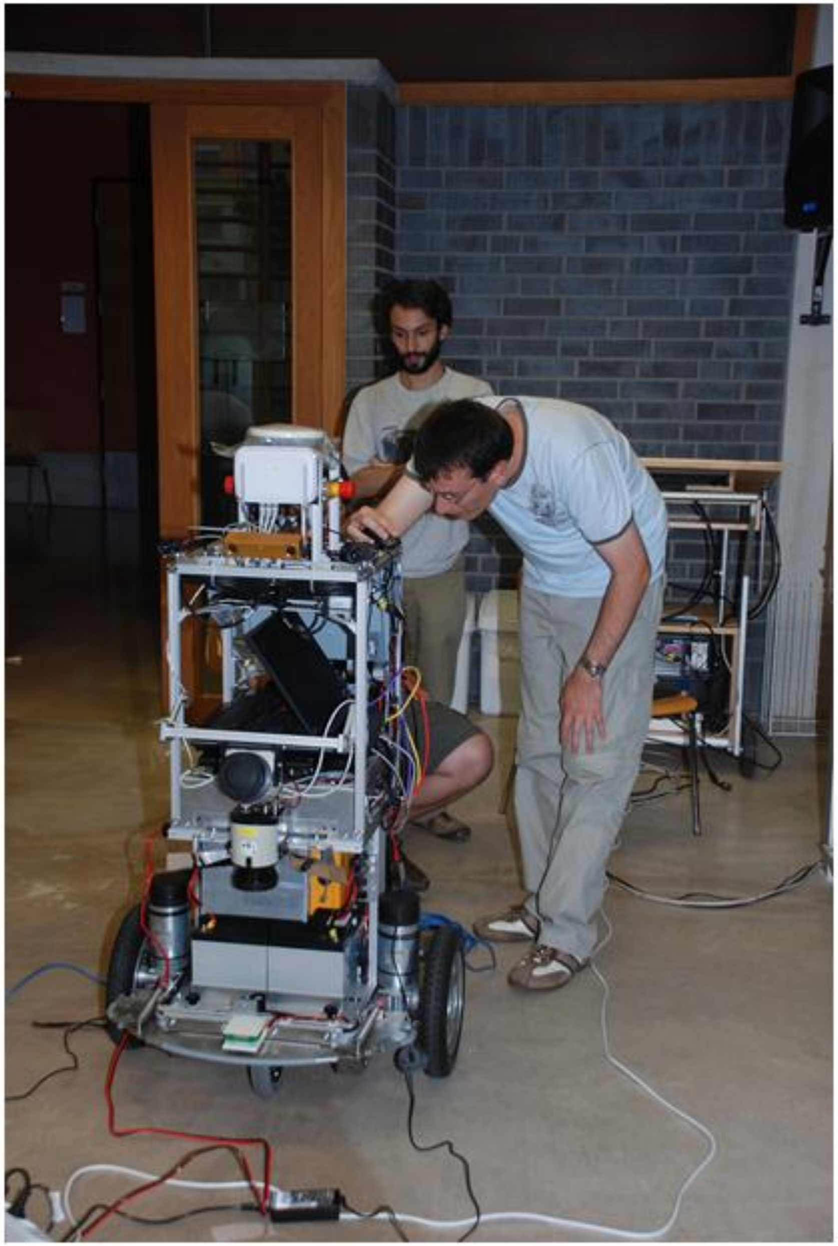
Semantic Parsing of Natural Language Input for Dialogue Systems

Jamie Frost

Oxford University Computational Linguistics Group

UNIVERSITY OF OXFORD

Video



The EUROPA Project

- Autonomous ‘pedestrian assistant’ robot designed to operate in a city/town environment.
- Provides information to pedestrians and escorts them to their requested locations.

The EUROPA Project

Laser Mapping

Global Path Planning

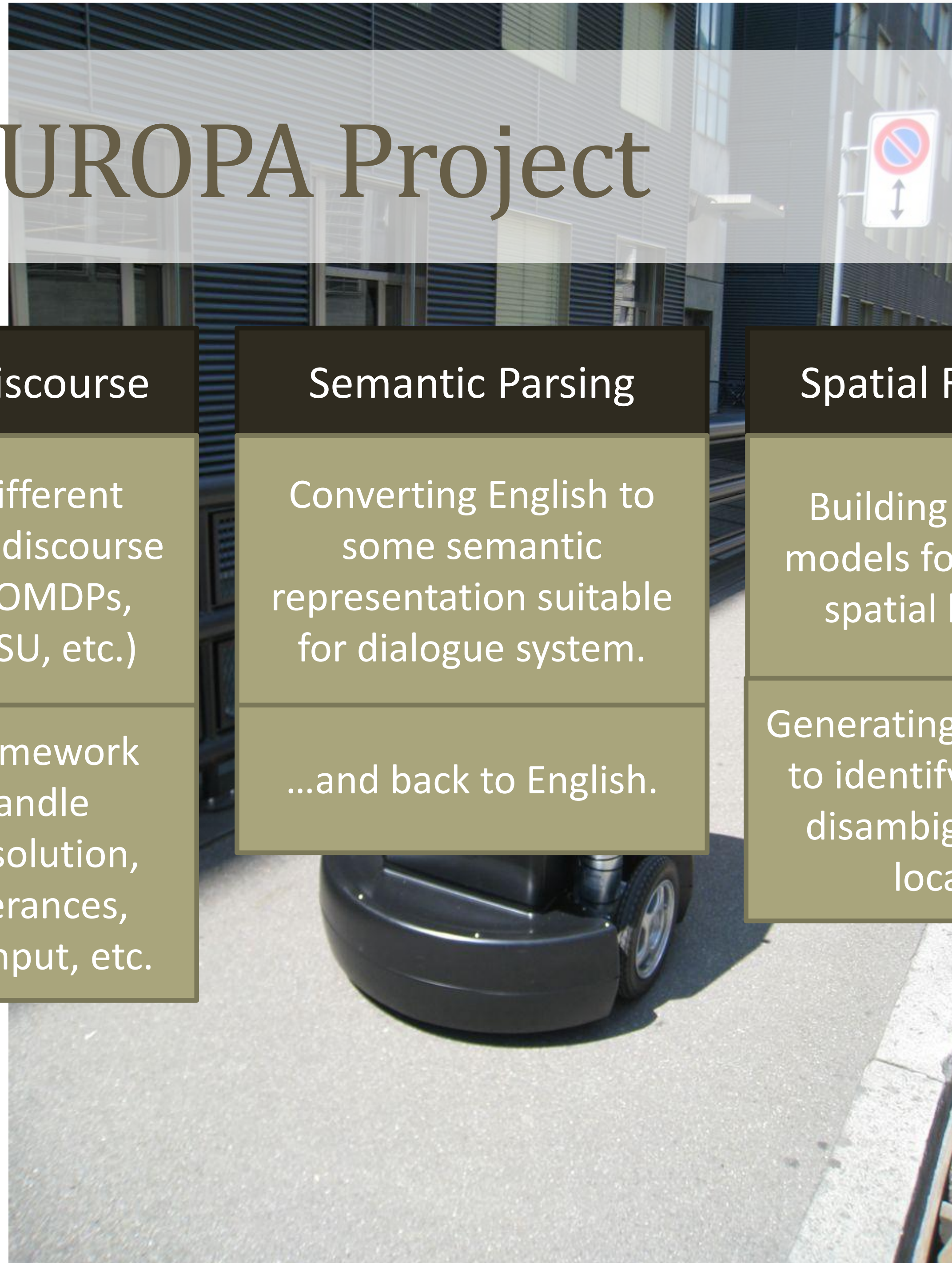
Local Path Planning

Pedestrian Tracking

Linguistic Control

Location Recognition

The EUROPA Project



Models of Discourse

Evaluating different approaches to discourse modelling (POMDPs, Plan-Based, ISU, etc.)

Building a framework that can handle anaphoric resolution, multiple utterances, multi-modal input, etc.

Semantic Parsing

Converting English to some semantic representation suitable for dialogue system.

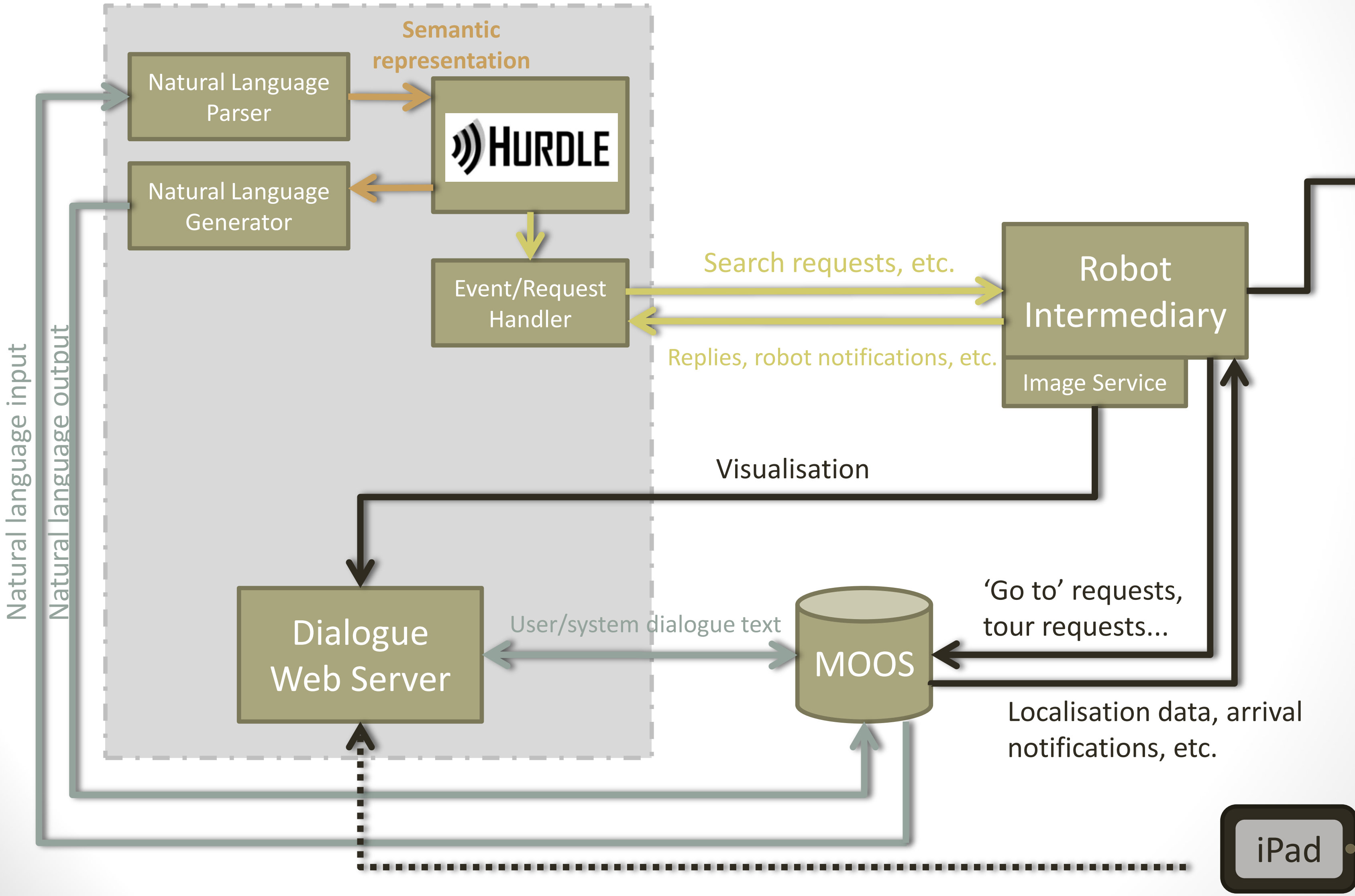
…and back to English.

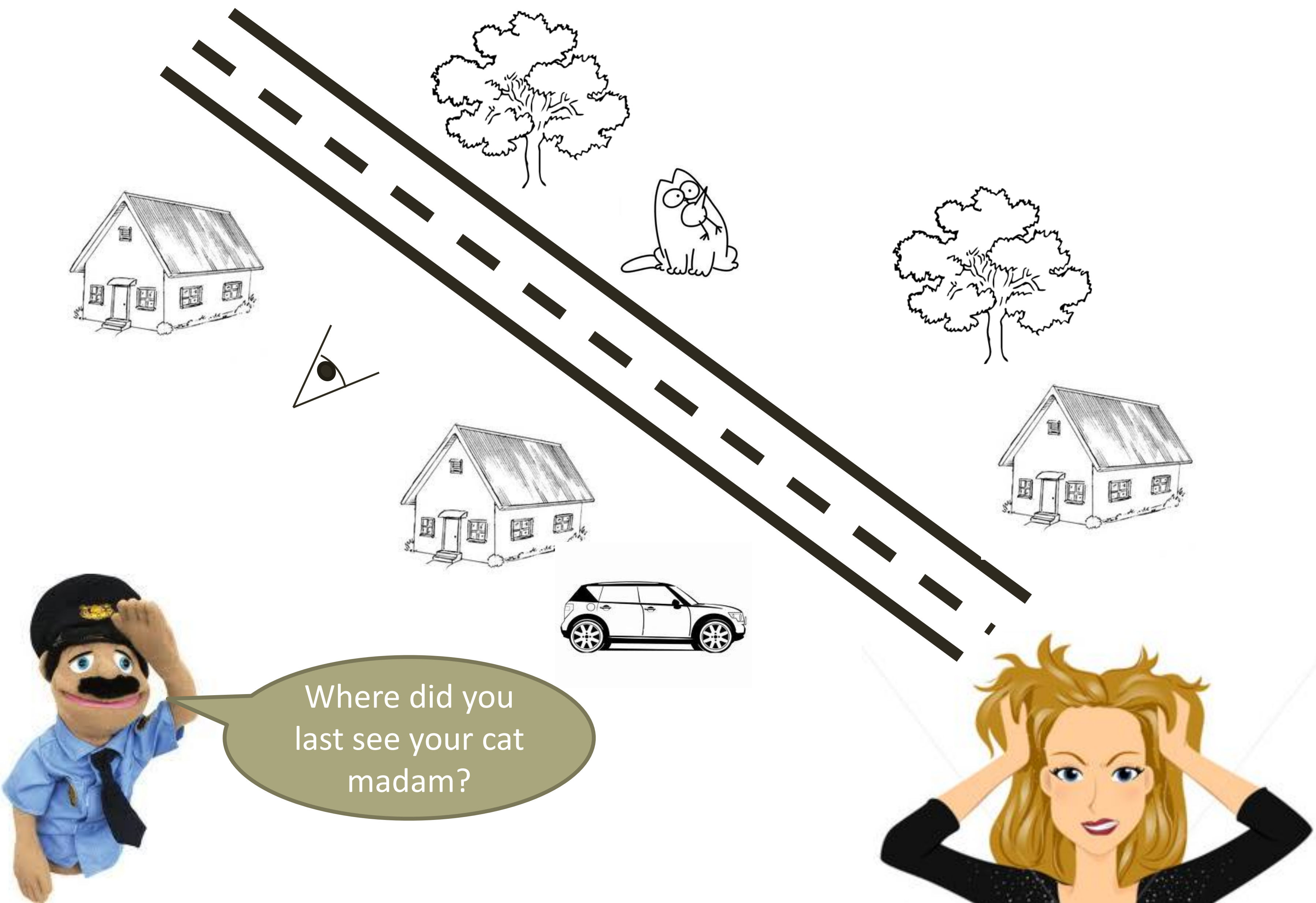
Spatial Reasoning

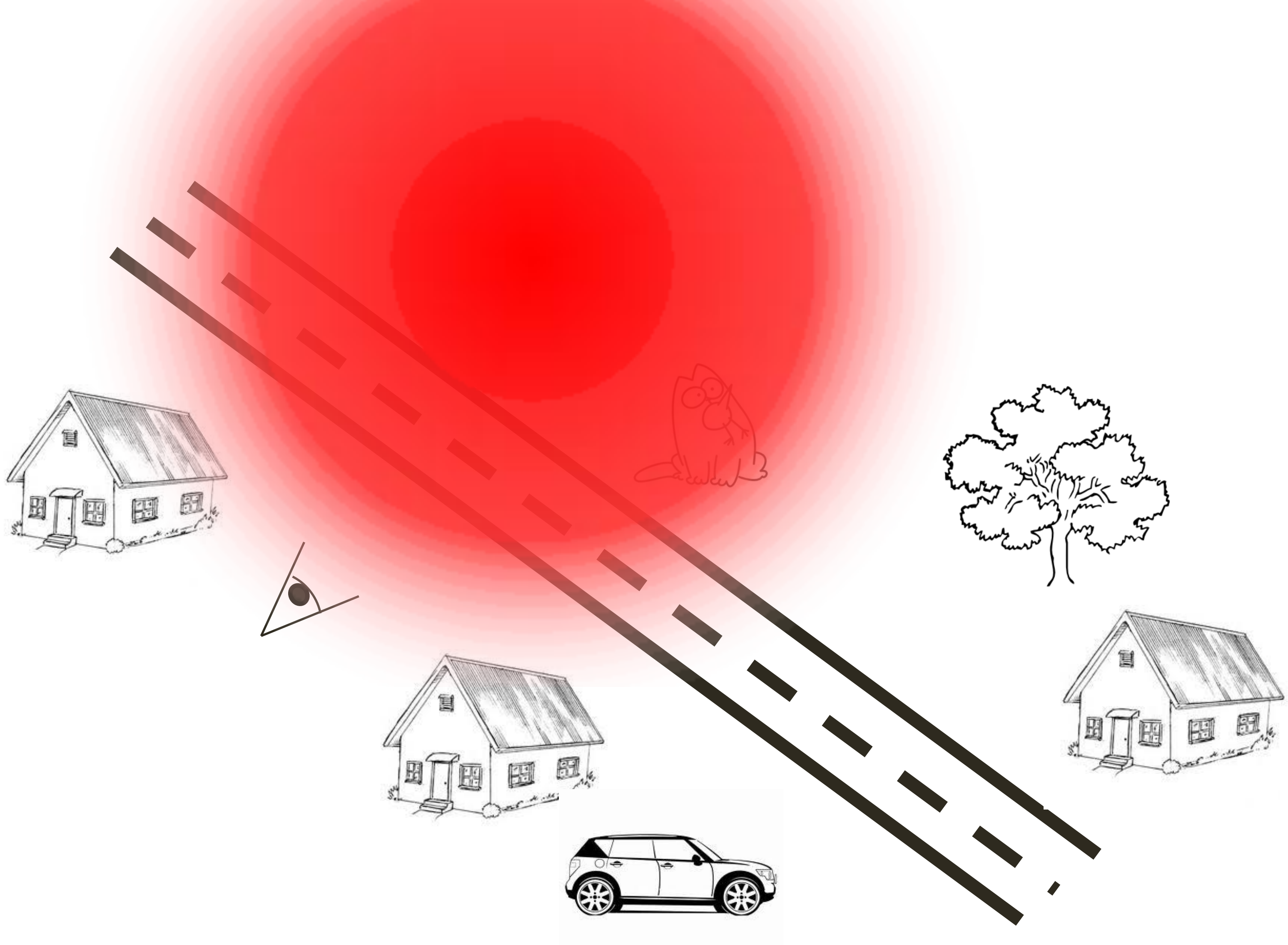
Building numerical models for aspects of spatial language.

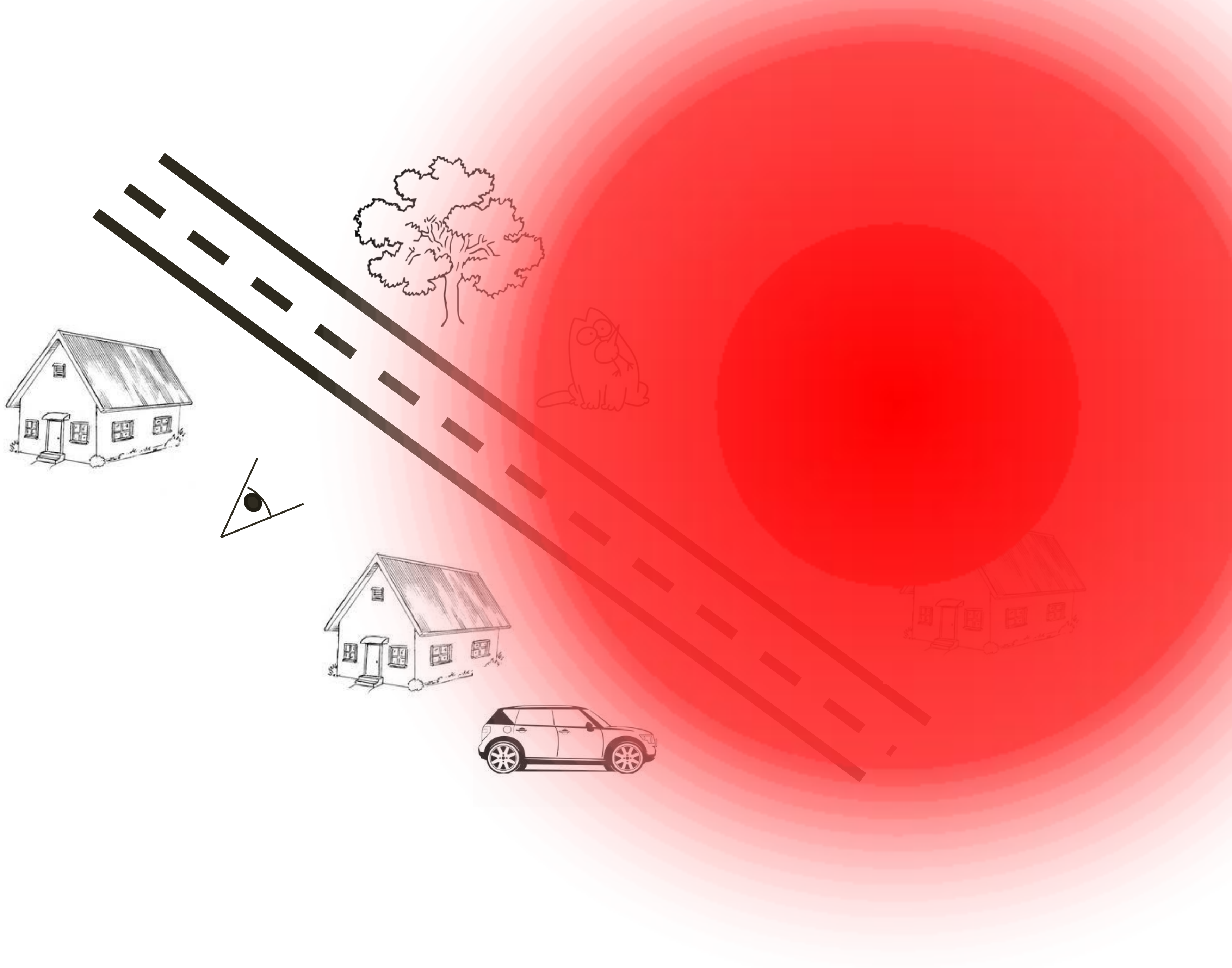
Generating expressions to identify objects or disambiguate their location.

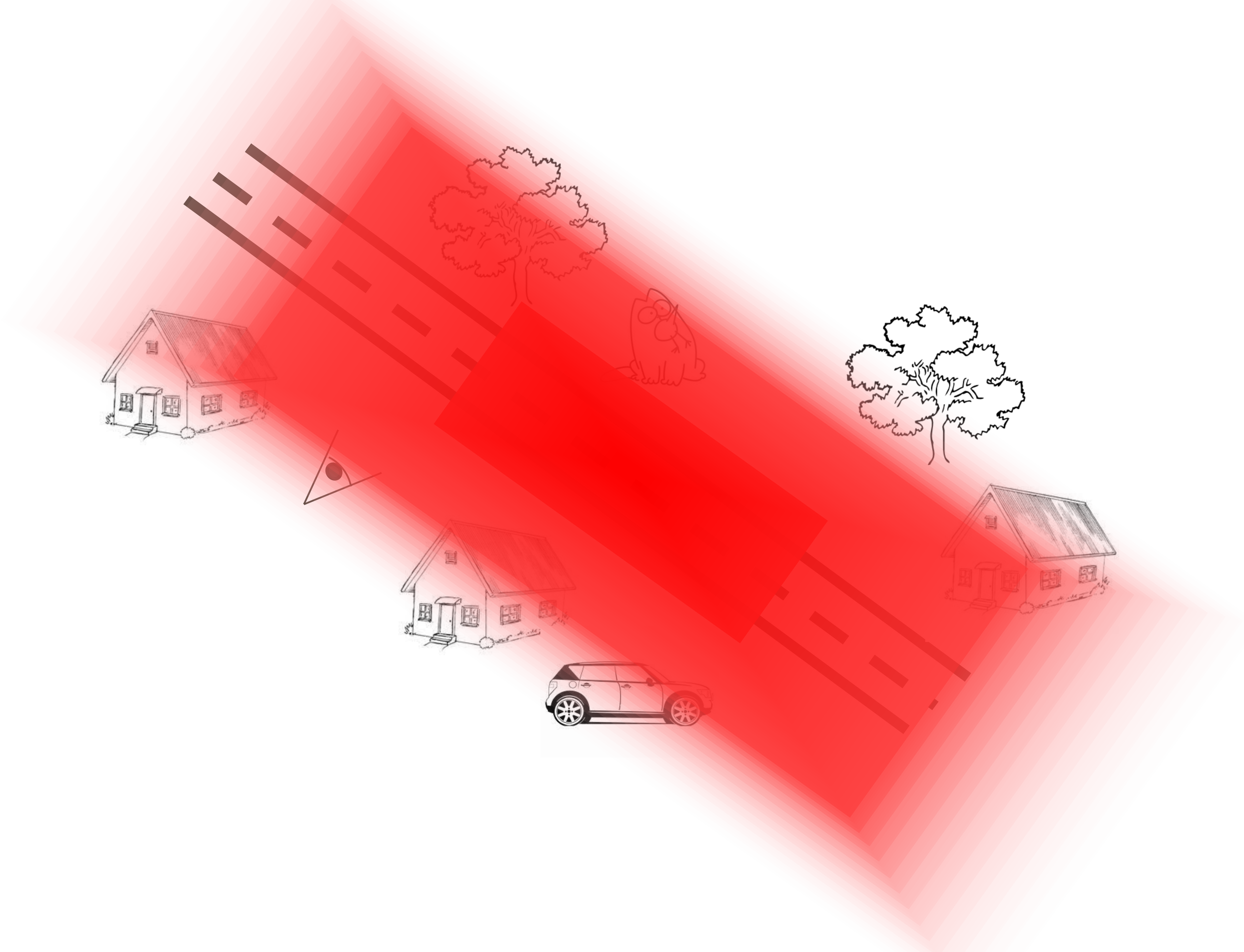
Architecture
Dialogue System

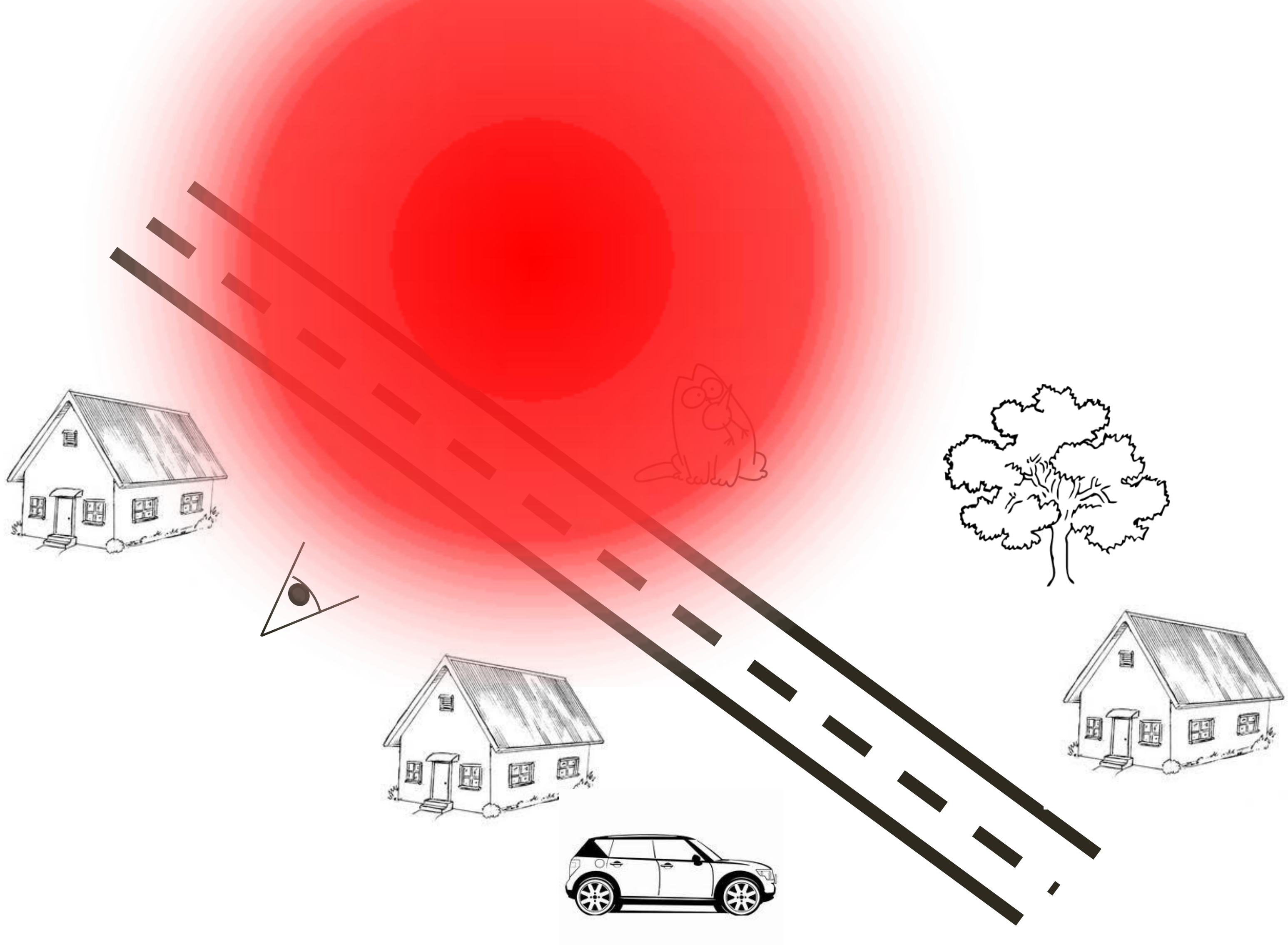


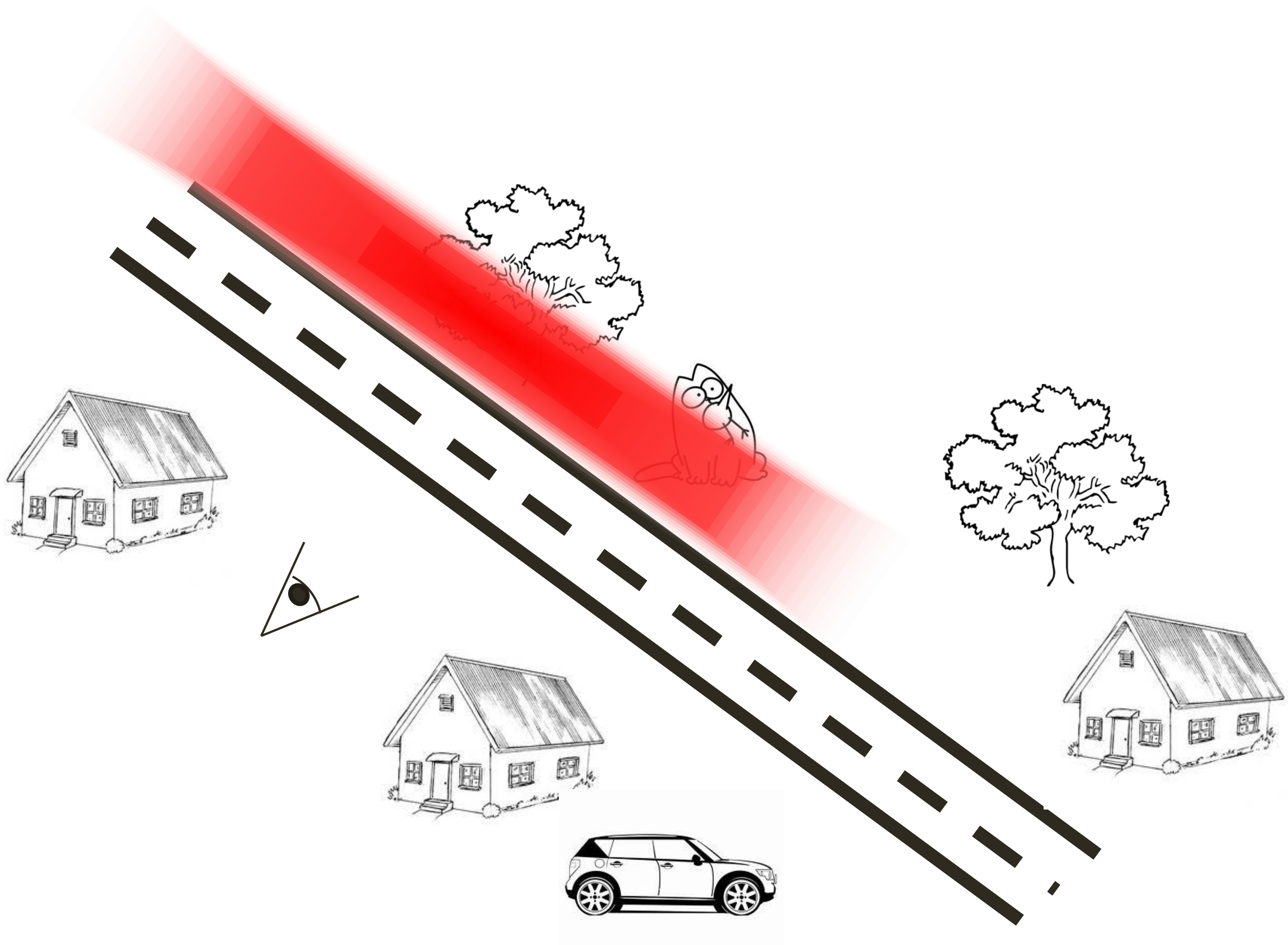

Where did you last see your cat madam?

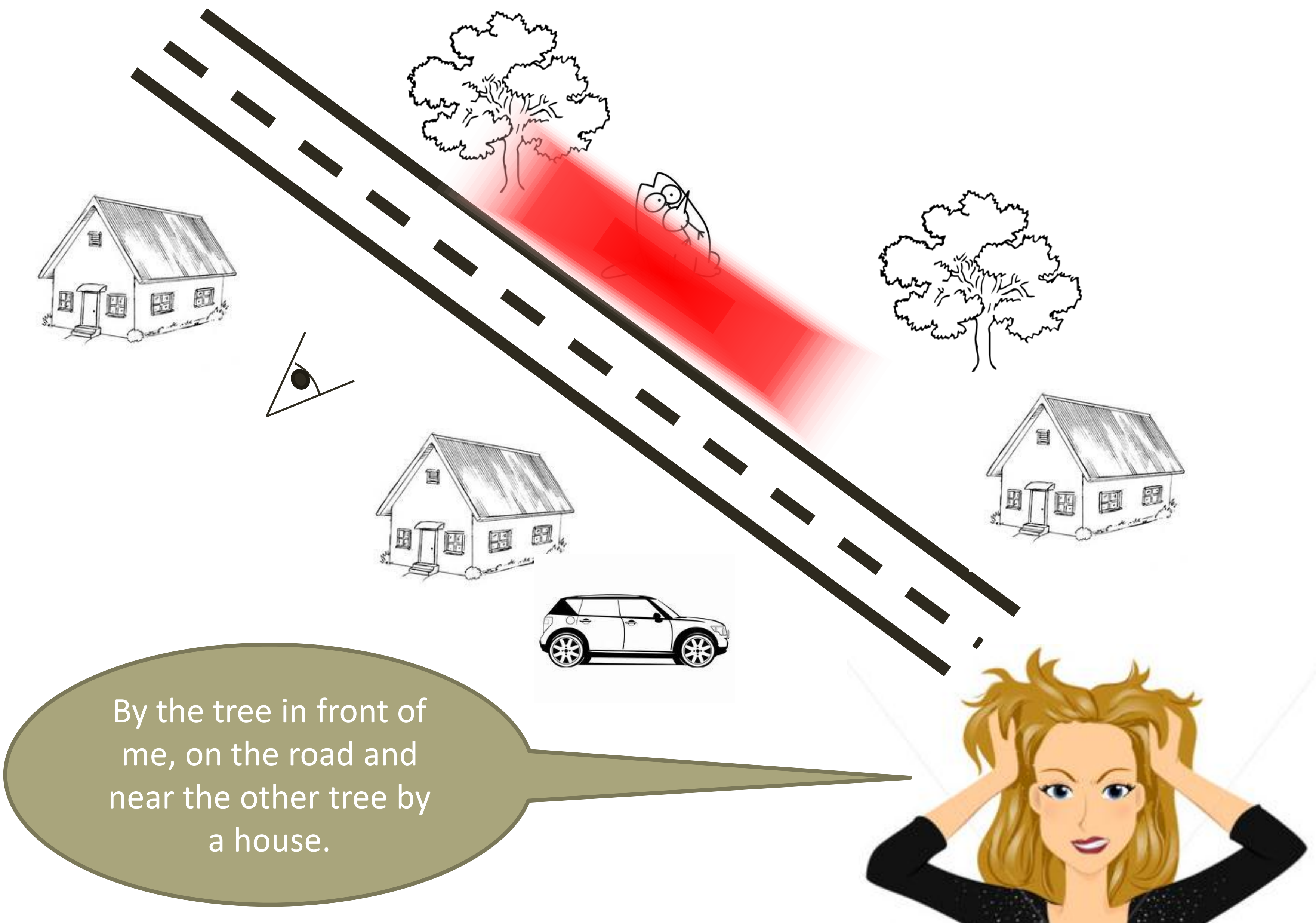











By the tree in front of me, on the road and near the other tree by a house.

The Idiot's Guide™ to:

FINDING YOUR CAT

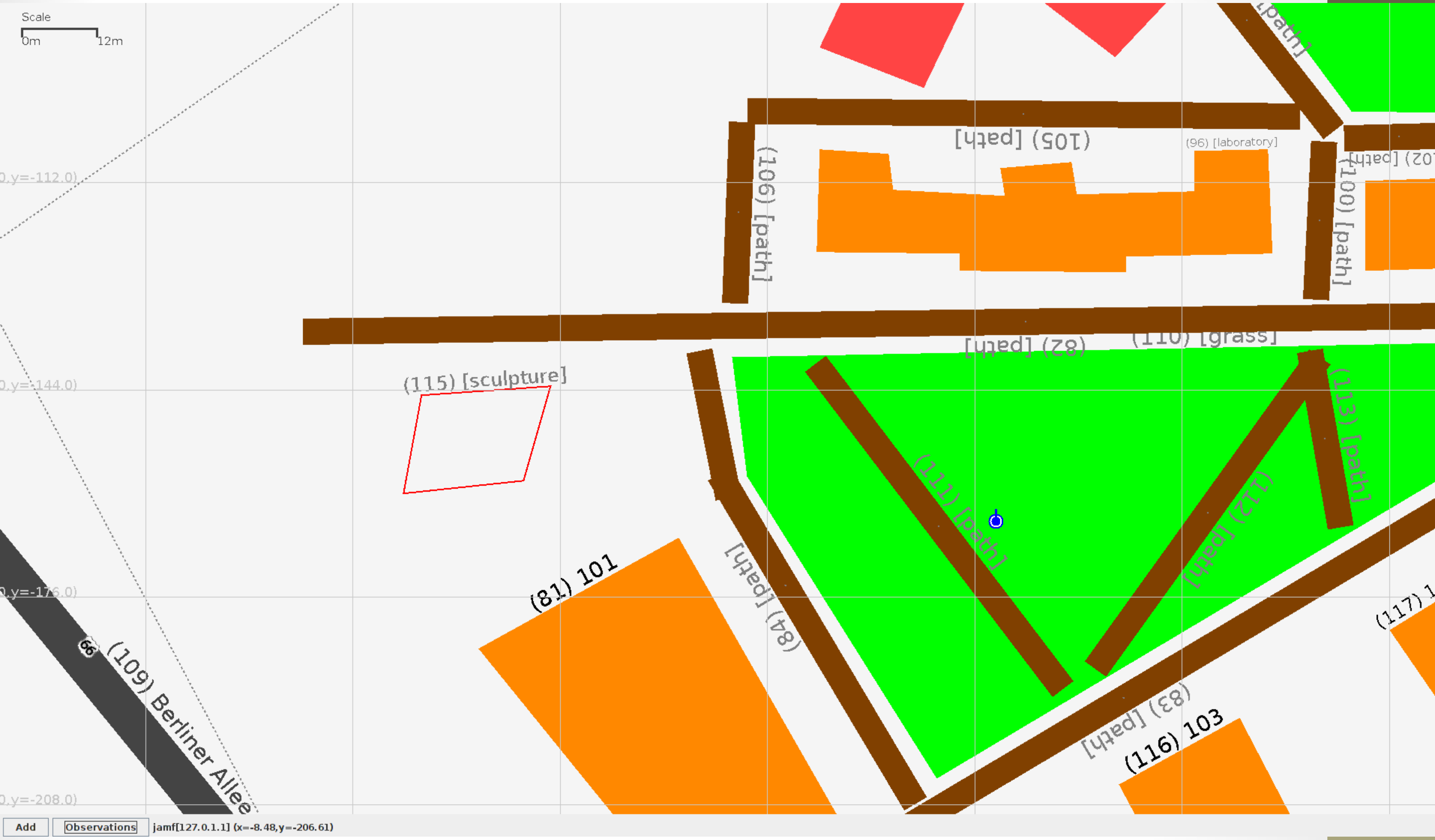
$$|F| = \int_{-\infty}^{+\infty} \prod_{i=1}^{m} \psi_m(\mathbf{x})$$

$$= \int_{-\infty}^{+\infty} \prod_{i=1}^{m} (2\pi)^{\frac{n}{2}} \sqrt{det(\Sigma_i)} \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i)$$

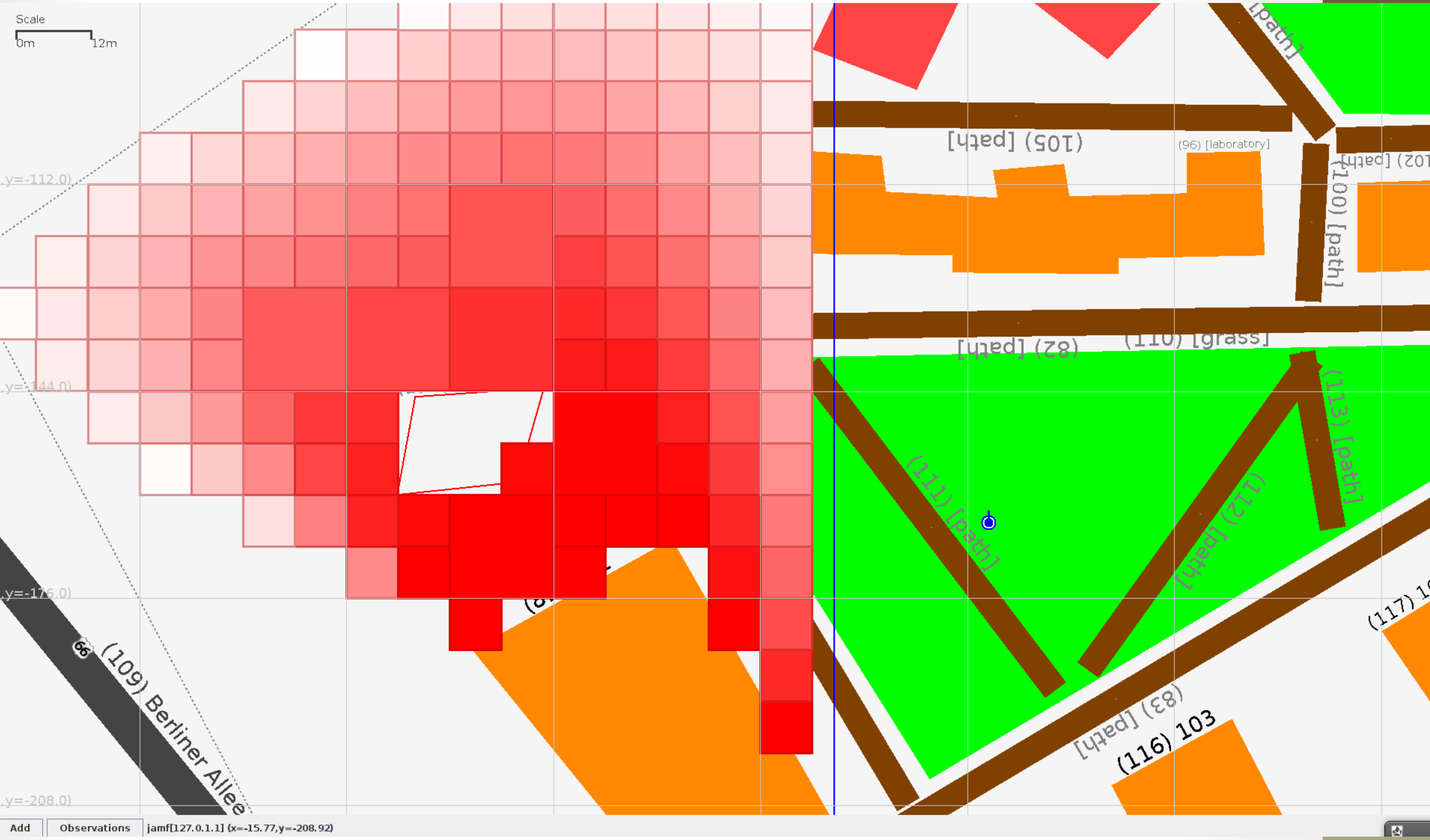
$$= (2\pi)^{\frac{mn}{2}} (\prod_{i=1}^{m} \sqrt{det(\Sigma_i)}) \int_{-\infty}^{+\infty} \mathcal{N}(\mathbf{x}|\mu, \Sigma)$$

$$= (2\pi)^{\frac{mn}{2}} \sqrt{\prod_{i=1}^{m} det(\Sigma_i)}$$

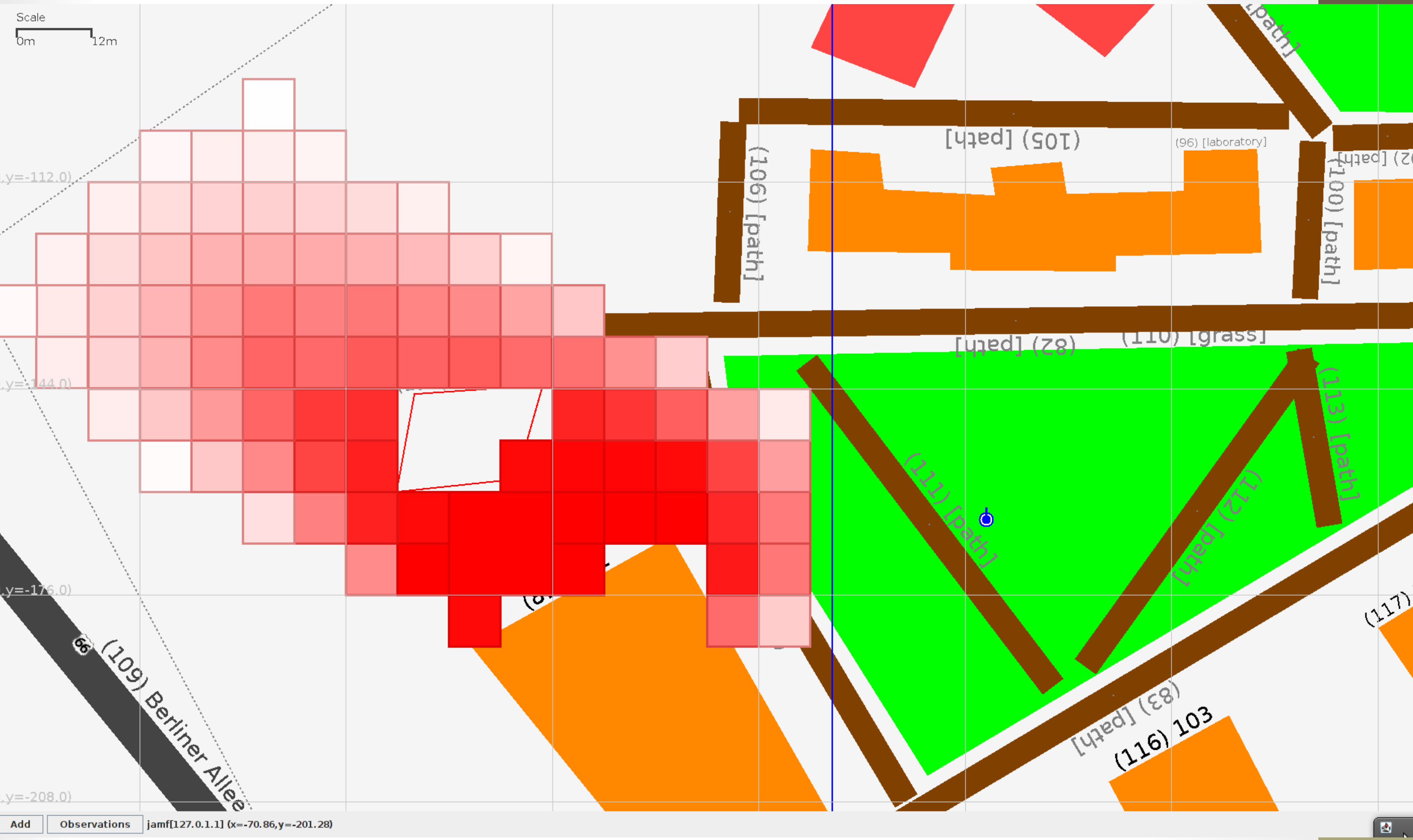
where $\Sigma = (\sum_{i=1}^{m} \Sigma_i^{-1})^{-1} \quad \mu = \Sigma \sum_{i=1}^{m} \Sigma_i^{-1} \mu_i$



Scale
0m
12m
(105) [path]
(96) [laboratory]
(106) [path]
(100) [path]
(82) [path]
(110) [grass]
(115) [sculpture]
(113) [path]
(111) [path]
(112) [path]
(84) [path]
(81) 101
(109) Berliner Allee
66
(83) [path]
(116) 103
Add
Observations
jamf[127.0.1.1] (x=-8.48,y=-206.61)



Scale
0m
12m
y=-112.0)
y=-144.0)
y=-176.0)
y=-208.0)
(109) Berliner Allee
(105) [path]
(96) [laboratory]
(100) [path]
(82) [path]
(110) [grass]
(113) [path]
(111) [path]
(112) [path]
(83) [path]
(116) 103
Add
Observations
jamf[127.0.1.1] (x=-15.77,y=-208.92)



Scale
0m
12m
y=-112.0
y=-144.0
y=-176.0
y=-208.0
(105) [path]
(96) [laboratory]
(106) [path]
(100) [path]
(82) [path]
(110) [grass]
(111) [path]
(112) [path]
(113) [path]
(83) [path]
(116) 103
(117)
(109) Berliner Allee
66
Add
Observations
jamf[127.0.1.1] (x=-70.86,y=-201.28)

(Sentence from from "TownInfo" training set.)

"hi i'm looking for a bar but i don't have much money on me and the other thing is i'd like it to be in the south of town because i've a train to catch at the station is there anywhere suitable"

Dialogue | Parsing | Visualiser

hi i'm looking for a bar but i don't have much money on me an | Parse >>

1
```
greet
```

2
```
inform(
   wantsAction(
      find(
         LOCEXPR {
            dA:    false ,
            class: 'bar'
         }
      )
   )
)
```

3
```
inform(
   wantsEnforced(
      objDesc(
         @LOCEXPR,
         LOCPROPS {
            relations: [ prep(
                            In,
                            LOCEXPR {
                               class: 'town' ,
                               part:  PART {
                                         type:
'south'
                                      }
                            }
                         )
                       ]
         }
      )
   )
)
```

4
```
requestInfoYN(
   thereExists(
      LOCEXPR {
         id: -7
      }
   )
)
```

How have discourse systems parsed language in the past?

Approach 1: Keyword Spotting

No encoding of input.
Dialogue Manager responds directly to particular keywords.
Example: automated telephone system.

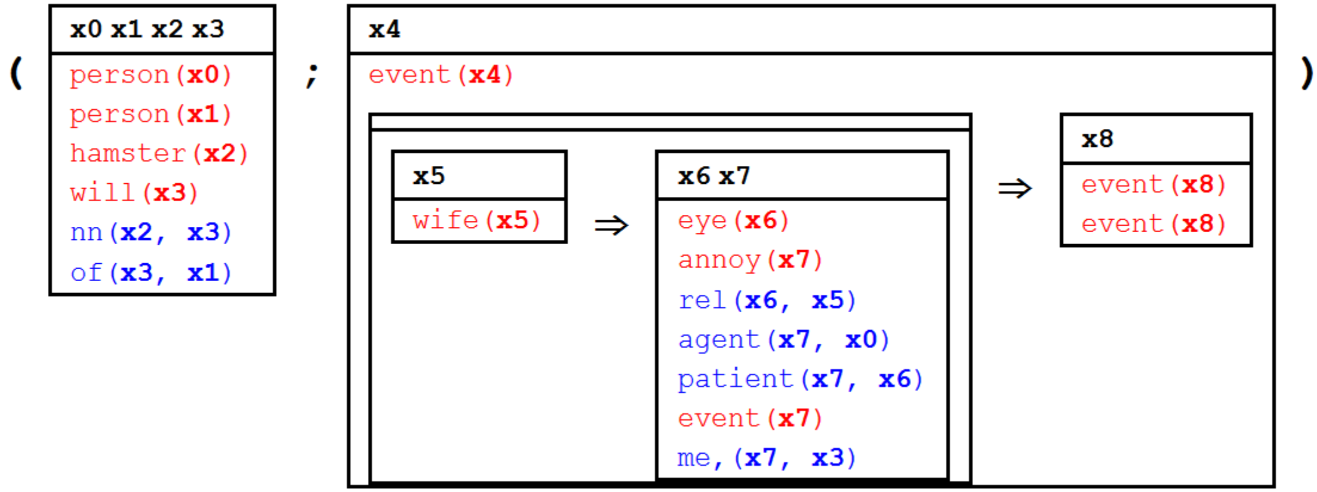
Advantages

- Predictable rigid behaviour.
- Simple to implement.

Disadvantages

- Very limited representation of semantic content.
- Dialogue Manager coupled too tightly with raw source input.

Approach 2: Full Logic Based Representation




```
answer(A,count(B,(city(B),loc(B,C),
       const(C,countryid(usa))),A))
```

"How many cities are there in the US?"

what kind of plane is used on these flights

$$\lambda y.\exists x.flight(x) \wedge from(x, bos) \wedge to(x, phi) \\ \wedge during(x, morning) \wedge aircraft(x) = y$$

Approach 3: DA Taxonomy with Key-Value Pairs

An example dialogue and its representation at the dialogue act level.

	Utterance	Dialogue act
U:	Hi, I am looking for somewhere to eat.	hello(task = find,type=restaurant)
S:	You are looking for a restaurant. What type of food do you like?	confreq(type = restaurant,food)
U:	I'd like an Italian somewhere near the museum.	inform(food = Italian,near=museum)
S:	Roma is a nice Italian restaurant near the museum.	inform(name = "Roma", type = restaurant, food = Italian,
U:	Is it reasonably priced?	confirm(pricerange = moderate)
S:	Yes, Roma is in the moderate price range.	affirm(name = "Roma", pricerange = moderate)
U:	What is the phone number?	request(phone)
S:	The number of Roma is 385456.	inform(name = "Roma", phone = "385456")
U:	Ok, thank you goodbye.	bye( )

Approach 3: DA Taxonomy with Key-Value Pairs

Advantages

- Taxonomy captures natural couplings of speech acts in dialogue (e.g. request often followed by acknowledge, question by answer, etc.)
- Easy for a Dialogue Manager to see particular information of interest.
- Simple representation lends well to Machine Learning approaches for learning dialogue policy.

Disadvantages

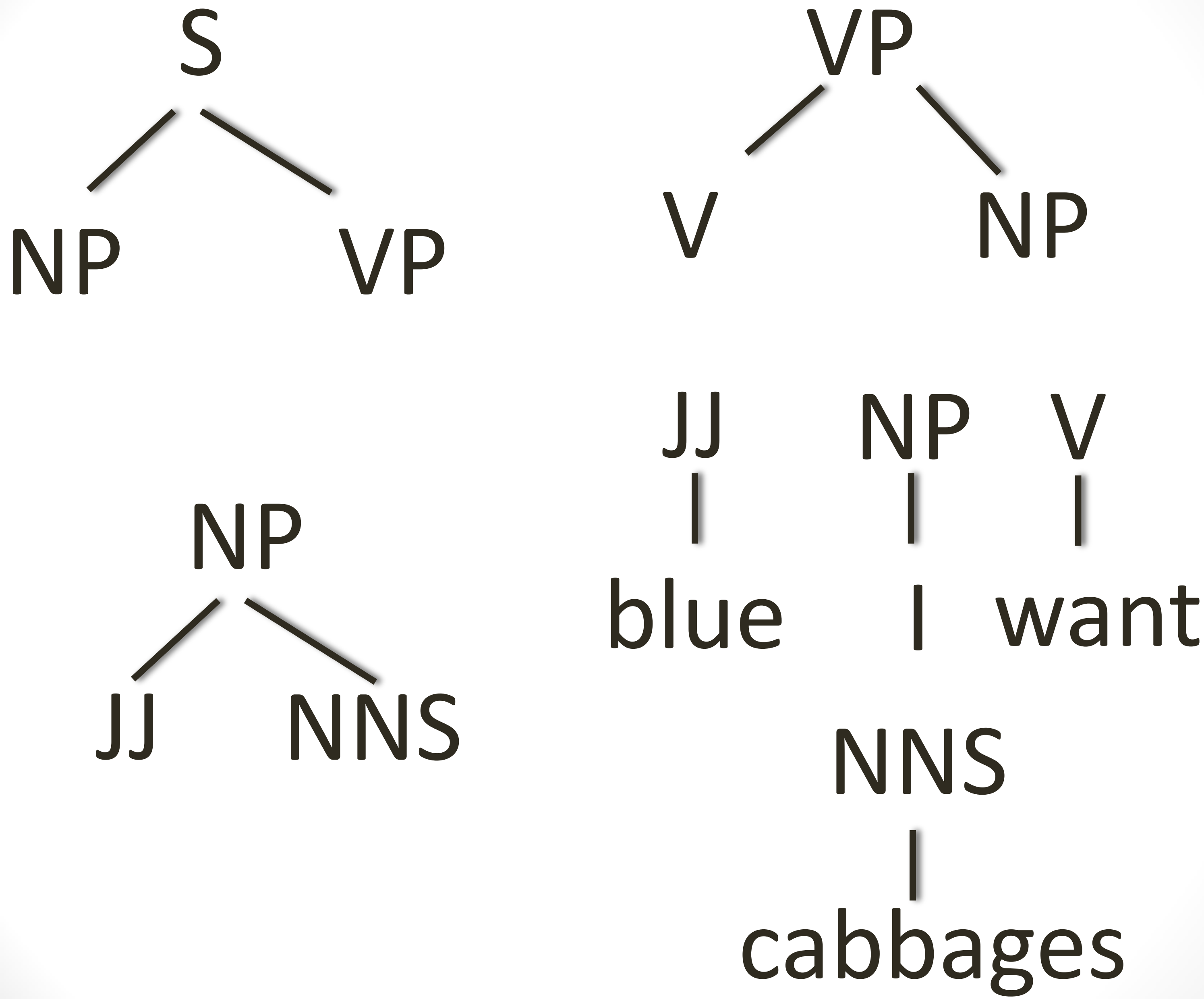
- Limited semantic encoding.

Our Approach

- Target semantic language represented as a **Context Free Grammar**.
- CFG can be automatically generated by our Dialogue Manager framework.

Advantages

- Allows very expressive representation (e.g. English language definable with CFG) yet with a rigid tree like structure.
- Easy to extract subtrees representing data we're interested in.


S
NP
VP
VP
V
NP
NP
JJ
NNS
JJ
NP
V
blue
I
want
NNS
cabbages

S S

NP[1] VP[2] NP[1] VP[2]

VP VP

V[1] NP[2] V[1] NP[2]

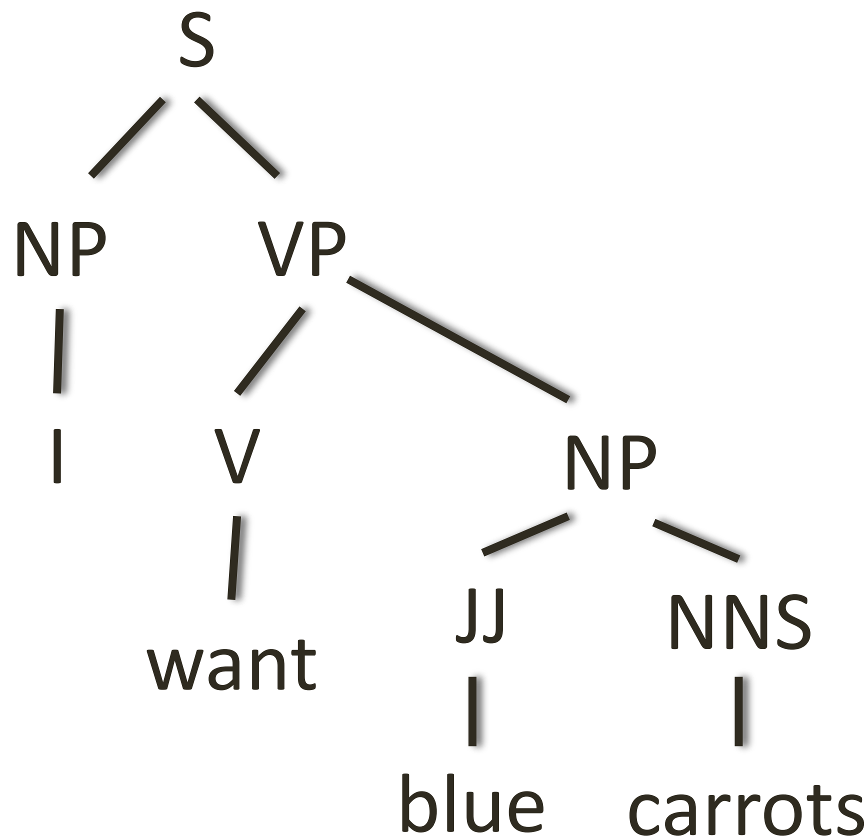
NP → JJ[1] NNS[2] ⟺ NP → des NNS[2] JJ[1]

JJ → blue ⟺ JJ → bleues

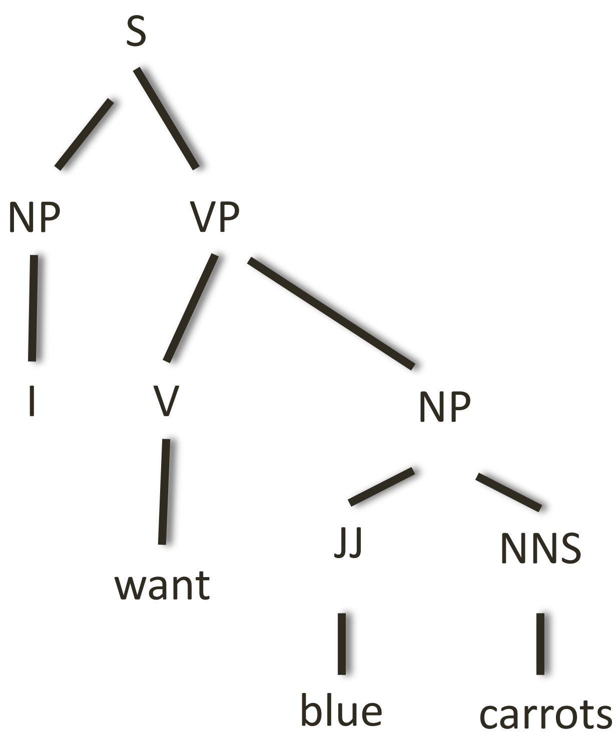
NNS → carrots ⟺ NNS → carottes

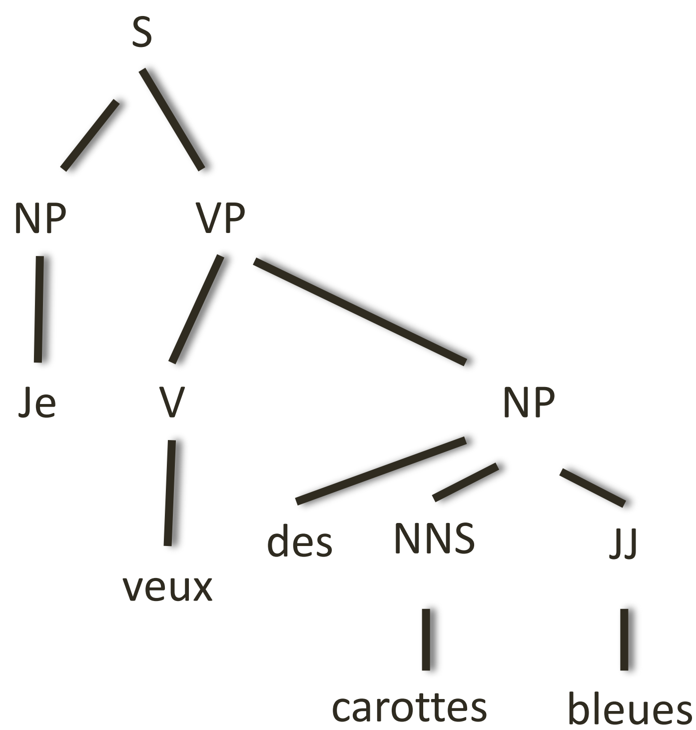
Synchronous Context Free Grammar

I want blue carrots



I want blue carrots





Examples

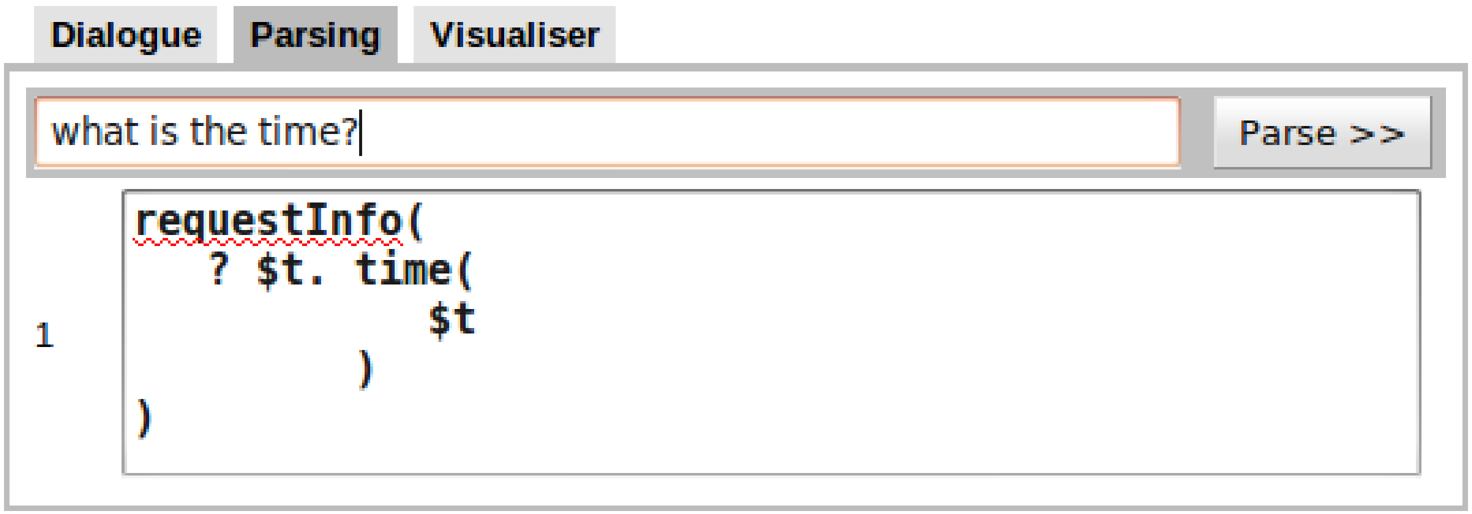
Dialogue | Parsing | Visualiser

I'm looking for the cat in the alley

Parse >>

```
inform(
   wantsAction(
      find(
         LOCEXPR {
            dA:          true ,
            relations: [ prep(
                            In,
                            LOCEXPR {
                               dA:    true ,
                               class: 'alley'
                            }
                         )
                       ] ,
            class:       'cat'
         }
      )
   )
)
```

Examples


Dialogue
Parsing
Visualiser
what is the time?
Parse >>
1
requestInfo(
? $t. time(
$t
)
)

Example Rule

```
(C the closest [1](CWITHADJ) to [2](L))
       =>      (C LOCEXPR { [1](CWITHADJ) ,
               dA : true ,
               relations : \[ nearOrdinal \( [2](L) , 1 \)\]
               } )
```

```
(CWITHADJP [1](ADJS) [2]*plural )
       =>      (CWITHADJ class : [2]*singularise ,
               multiple : true ,
               [1](ADJS) )
```

Dialogue Act Segmentation

```
(ROOT [1](SCOMP))
                =>      (ROOT \[ [1](SCOMP) \] )

(SCOMP [1](S) [2](SCOMP))
                =>      (S [1](S) , [2](SCOMP))

(SCOMP [1](S))
                =>      (SCOMP [1](S))
```

Yes, I want the muffin. Go get it.

Dealing with superfluous info

```
(ROOT [1](WILDCARDS) [2](SCOMP) [3](WILDCARDS))
     =>  (ROOT \[ [1](WILDCARDS) , [2](SCOMP) , [3](WILDCARDS) \] )
(ROOT [1](SCOMP) [2](WILDCARDS))
     =>  (ROOT \[ [1](SCOMP) , [2](WILDCARDS) \] )
(ROOT [1](WILDCARDS) [2](SCOMP))
     =>  (ROOT \[ [1](WILDCARDS) , [2](SCOMP) \] )

(SCOMP [1](S) [2](SCOMP))
     =>  (S [1](S) , [2](SCOMP))
(SCOMP [1](S) [2](WILDCARDS) [3](SCOMP))
     =>  (SCOMP [1](S) , [2](WILDCARDS) , [3](SCOMP))

(WILDCARDS *)
     =>  (WILDCARDS ignore)
(WILDCARDS * [1](WILDCARDS))
     =>  (WILDCARDS ignore , [1](WILDCARDS))
```

"hi i'm looking for a bar but i don't have much money on me and the other thing is i'd like it to be in the south of town because i've a train to catch at the station is there anywhere suitable"

Challenges

- Considering all possible segmentations and allowing data to be ‘superfluous’ leads to lots of possible translations.
- Could use Probabilistic SCFGs can give a measure of the strength of the translation. Requires training data to obtain probabilities associated with rules.
- But for simplicity, we use simple heuristics to choose the best tree – i.e. the one that maximises the amount of parsed information.

Where does target grammar come from?

source

translation method

input type

use input IPAD SCFG(‘patternfile’) U **LIST<DIALOGUEACT>**;

use input ROBOT RAW R **PROP**;

use output IPAD SCFG(‘patternfile’);

HURDLE

Where does target grammar come from?

```
enum DIALOGUEACT {
        acknowledge,
        clarify(PROP),
        greet,
        informYes,
        informNo,
        informDontKnow,
        inform(PROP),
        requestInfo(QUD),
        requestInfoYN(PROP),
        requestAction(TASK),
        …
}
```

```
structure LOCEXPR {
        INT[?@] id,
        PART[?@] part,
        INT[?] classid,
        STR[?] class,
        LIST<PREPOSITION>[?] relations,
        STR[?] name,
        LIST<ATTRVAL>[?] attrs,
        BOOL[?] isVisible,
        BOOL[?] dA,
        BOOL[?] multiple
};

const REAL WALKINGDISTANCE = 150;
```

Problem?

The phone number of Worcester College is 78300

inform(objAttr(LOCEXPR{name:'Worcester College'}, (phone, 78300)))

- Non-isomorphic translations not easily represented by SCFG.
- i.e. Transformation of grammatical structure more complicated than renamings and swapping siblings.

- **Synchronous Tree Substitution Grammars (STSGs)** solve the problem, as they allow longer range dependencies.

Problem?

The phone number of Worcester College is 78300

inform(objAttr(LOCEXPR{name:'Worcester College'}, (phone, 78300)))

STSG > SCFG

Tree languages

Problem?

The phone number of Worcester College is 78300

inform(objAttr(LOCEXPR{name:'Worcester College'}, (phone, 78300)))

STSG = SCFG

String languages

Problem?

The phone number of Worcester College is 78300

inform(objAttr(LOCEXPR{name:'Worcester College'}, (phone, 78300)))

- We don't ultimately care whether we have the correct syntax tree of the source sentence.
- If our target grammar is **unambiguous**, we care only about the string (and indeed, our Dialogue Manager accepts parsed inputs in string form.
- Therefore **SCFG is sufficient**.

- But non-isomorphism property means that we'll likely have lots more rules.

Can we learn a SCFG?

HDP-SCFG

$\pi|\alpha \sim \text{GEM}(\alpha)$ (Draw top-level constituent prior distribution)

$\phi^S|\alpha^S, \pi \sim \text{DP}(\alpha^S, \pi)$ (Draw start-symbol distribution)

$\phi^T_z|\alpha^Y \sim \text{Dirichlet}(\alpha^Y)$

$\phi^M_z|\alpha^M, \pi \sim \text{DP}(\alpha^M, \pi\pi^T)$

$\phi^R_z|\alpha^R, \pi \sim \text{DP}(\alpha^R, \pi\pi^T)$

$\phi^E_z|\alpha^E, P_0 \sim \text{DP}(\alpha^E, P_0)$

Can generate 3 different types of rules:

- Monotone: Z → < X[1] Y[2] , X[1] Y[2] >
- Reordering: Z → < X[1] Y[2] , Y[2] X[1] >
- Emission: Z → < a , b >

$z_1|\phi^S \sim \text{Multinomial}(\phi^S)$ (First draw the start symbol)

For each node i in the synchronous derivation $\mathbf{z}$ with category z_i:

$t_i|\phi^T_{z_i} \sim \text{Multinomial}(\phi^T_{z_i})$ (Draw a rule type)

if t_i = Emission then:

$\langle \mathbf{e}, \mathbf{f} \rangle|\phi^E_{z_i} \sim \text{Multinomial}(\phi^E_{z_i})$ (Draw source and target phrases)

if t_i = Monotone Production then:

$\langle z_{l\boxed{1}} z_{r\boxed{2}}, z_{l\boxed{1}} z_{r\boxed{2}} \rangle|\phi^M_{z_i} \sim \text{Multinomial}(\phi^M_{z_i})$ (Draw left and right (source) child constituents)

if t_i = Reordering Production then:

$\langle z_{l\boxed{1}} z_{r\boxed{2}}, z_{r\boxed{2}} z_{l\boxed{1}} \rangle|\phi^R_{z_i} \sim \text{Multinomial}(\phi^R_{z_i})$

Rules in this form are synchronous equivalent of Chomsky Normal Form.

Summary

- We can use a variety of different methods to parse input for the purposes of dialogue.
- Often a trade off between the level of semantic content we capture and the ease of processing it.
- Use of SCFGs has a number of advantages:
  - Ties in well with Machine Translation theory.
  - And therefore gives us a means by which we can potentially **learn** a SCFG using Machine Learning.
  - Expressive representation (although can't for example represent logical operators very effectively, e.g. $\forall$, $\exists$, $\wedge$, $\rightarrow$).
  - Can be generated automatically based on the particular task domain.
- Attempted to build framework (HURDLE) that puts large emphasis on the ease for industry to develop complex systems as easily as possible, and without the need for too much specialist knowledge.

Any questions?