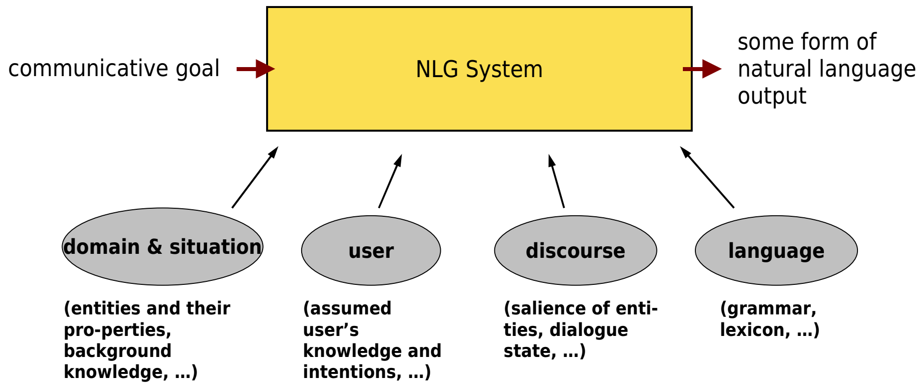
Natural Language Generation for Embodied Conversational Agents

Day 3

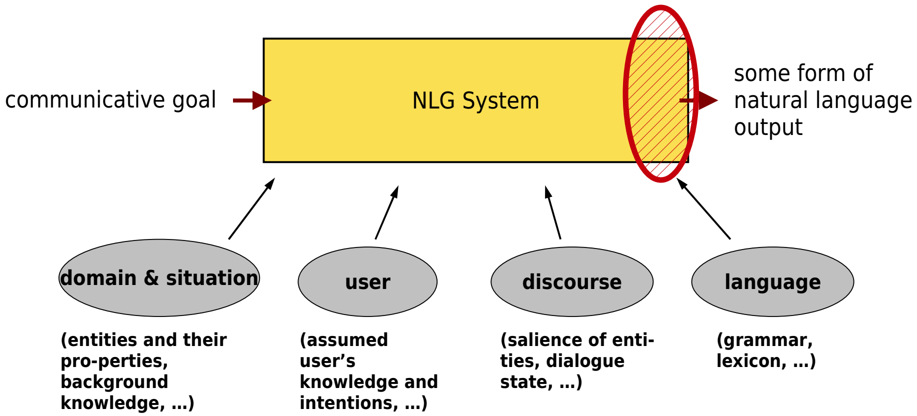
Kristina Striegnitz

ESSLLI 2008
Hamburg, Germany

Yesterday



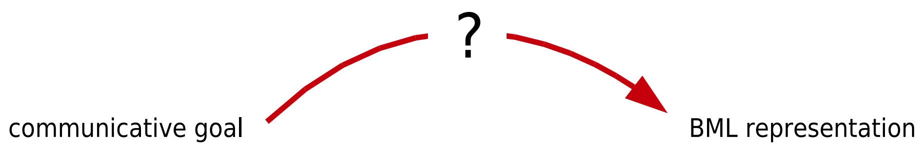
Yesterday



Yesterday – BML specifications

```
<speech id="s">
  and now take <sync id="t1"/> this bar and make it <sync id="t2"/> this
  big <sync id="t3"/>
</speech>
<gesture id="g1" type="POINT" target="obj" stroke="s:t1"/>
<gesture id="g2" type="GENERIC" stroke-start="t2" stroke-end="t3"
    hand="both"
    two handed="mirror"
    handshape=open hand"
    location="center, center, medium"
    orientation="palm inward, finger forward"
/>
```

Today – Where do those representations come from?

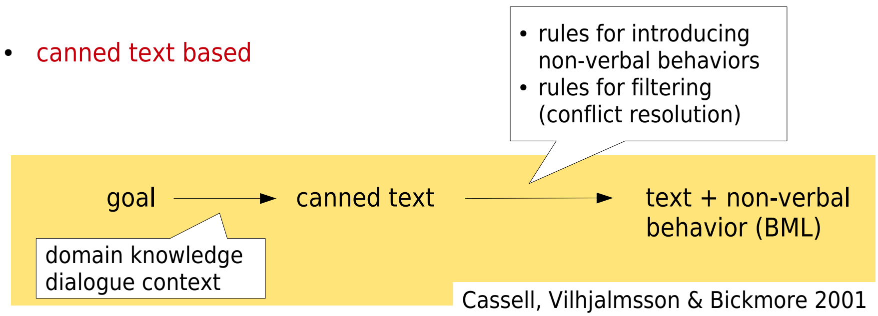


Today

- Mapping semantics to syntax
- Content determination
- Referring Expression Generation
- multimodal referring expressions
  - generating pointing gestures
  - generating iconic gestures

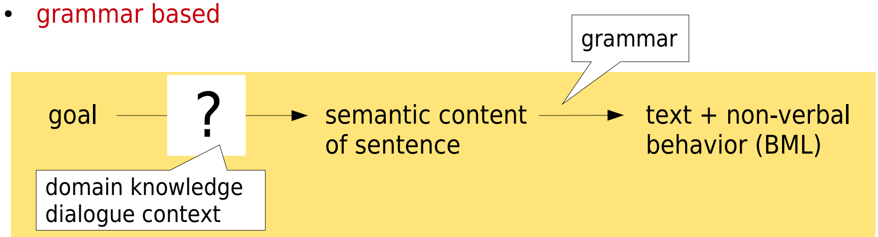
From communicative goal to BML representation

- canned text based

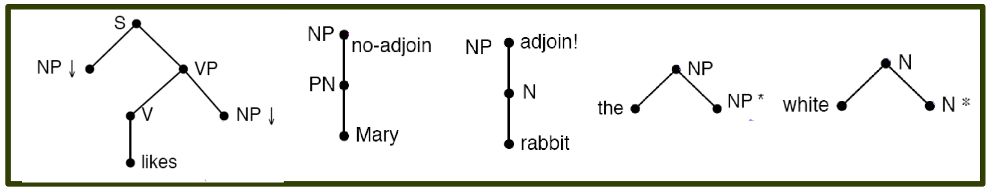


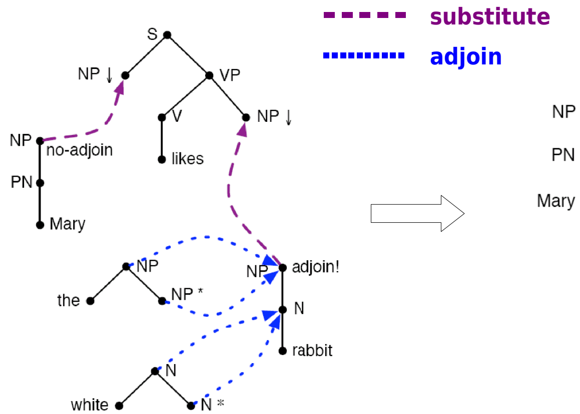
Cassell, Vilhjalmsson & Bickmore 2001
Lee & Marsella 2006

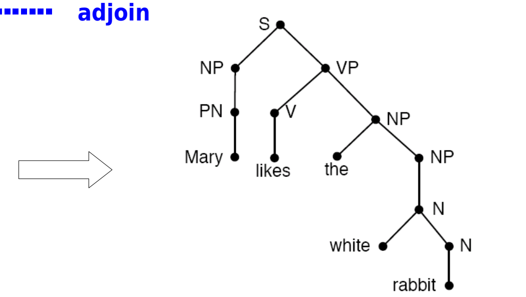
- grammar based



Excursion: LTAG – Lexicalized Tree Adjoining Grammar

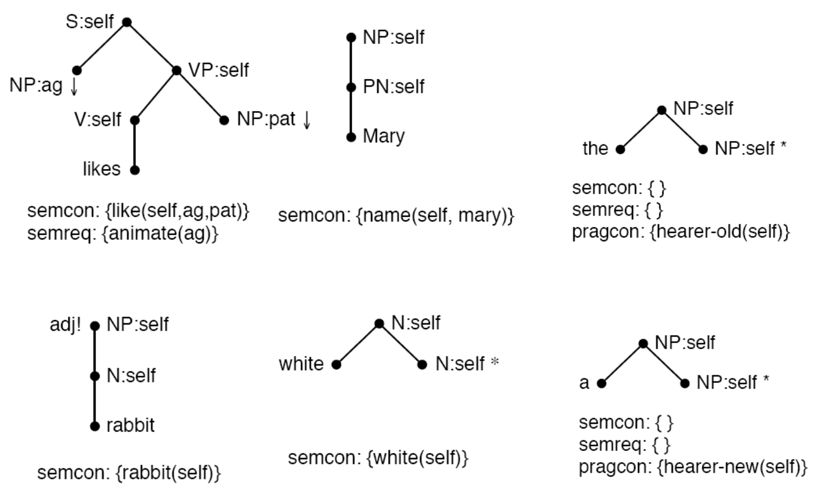






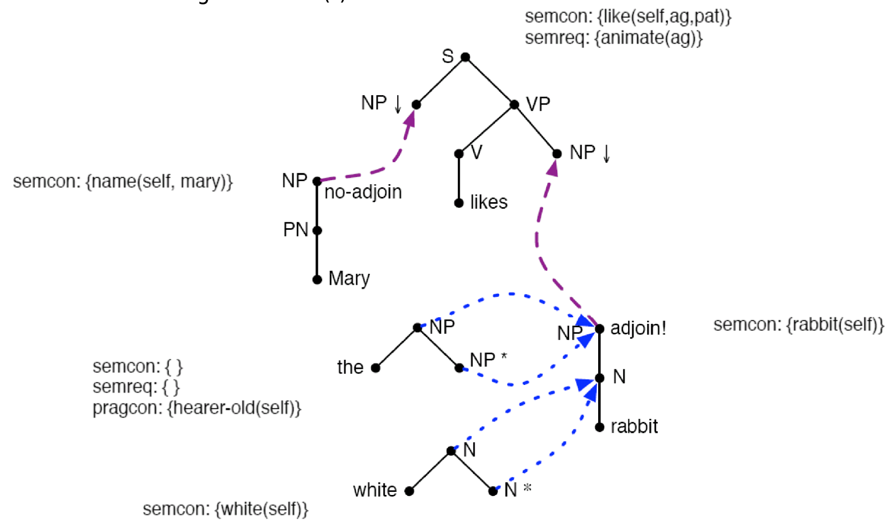
LTAG with semantics and pragmatics

Stone et al. 2003
Koller & Stone 2007



Mapping semantics to syntax

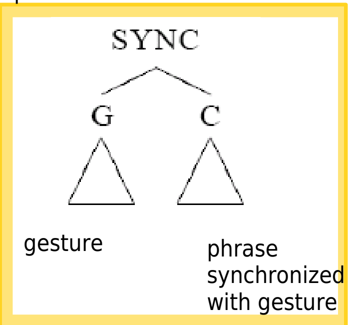
To communicate: like(e,m,r) name(m,mary) rabbit(r) white(r)
Discourse context: hearer-old(r)
Domain knowledge: animate(r)



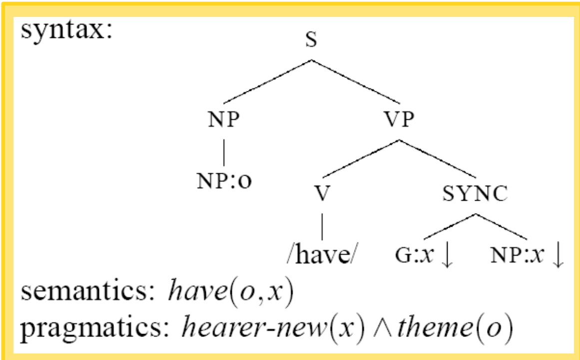
Integrating gestures

[Cassell, Stone & Yan 2000]

structure for synchronizing gestures with syntactic phrases:

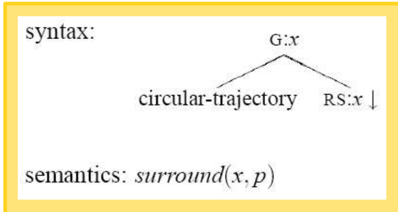


example lexical entry requiring a gesture:

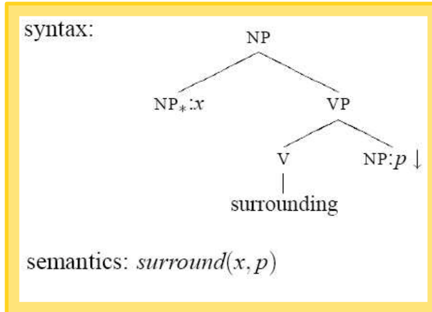


SPUD – lexical entries for gestures

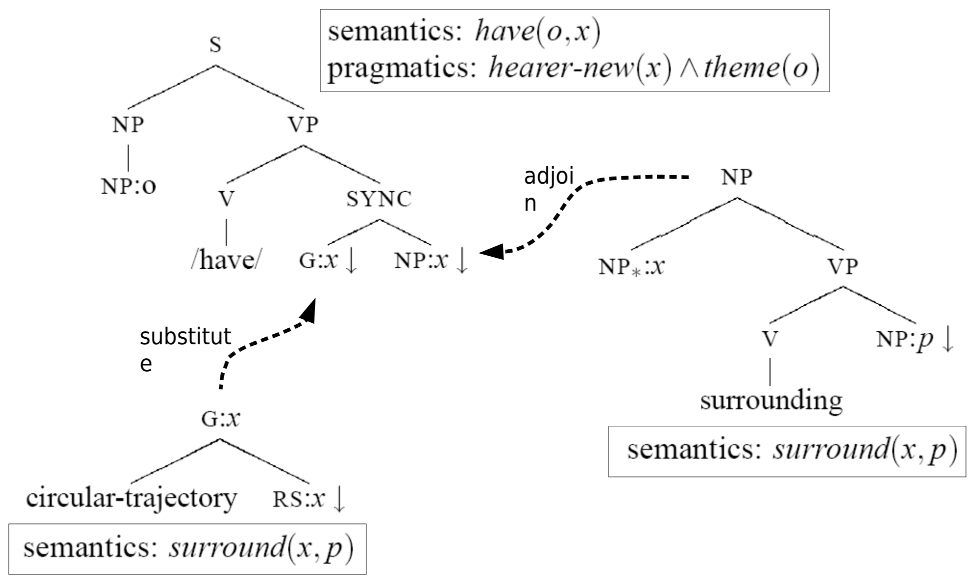
a gesture entry:



A “word” entry with the same semantics.
Gestures can be semantically redundant or complementary:

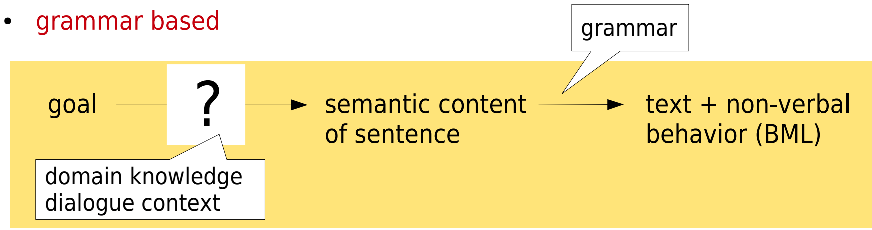


SPUD – building a multi-modal utterance specification



Where does the semantics come from?

- grammar based



For example:

goal: describe how to get from point A to point B

↓ ?

... [turn(right, b1), building(b1), tall(b1)] ...

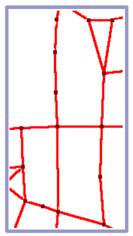
Today

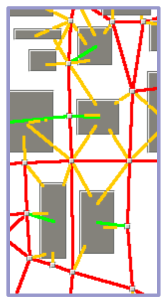
- Mapping semantics to syntax
- Content determination
- Referring Expression Generation
- multimodal referring expressions
  - generating pointing gestures
  - generating iconic gestures

Content determination example: walking directions

User: how do I get from building A to building B?

Communicative goal: describe how to get from point p_A to point p_B.

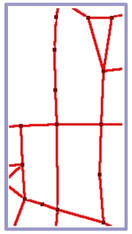


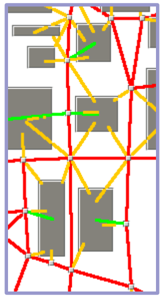


Content determination example: walking directions

User: how do I get from building A to building B?

Communicative goal: describe how to get from point p_A to point p_B.





p_A, p49, p50, p58, p63, p80, p81, p_B

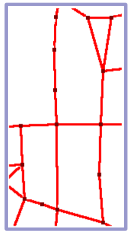
start lm_A (at p_A); end lm_B (at p_B)

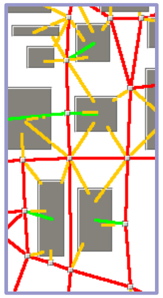
- A* search

Content determination example: walking directions

User: how do I get from building A to building B?

Communicative goal: describe how to get from point p_A to point p_B.





p_A, p49, p50, p58, p63, p80, p81, p_B

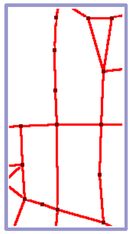
start lm_A (at p_A); right lm_C (at p49); left lm_D (at p80); end lm_B (at p_B)

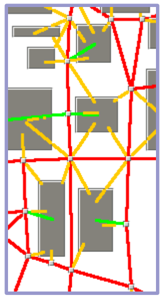
- A* search
- determine reorientation points
- pick landmarks for reorientation points

Content determination example: walking directions

User: how do I get from building A to building B?

Communicative goal: describe how to get from point p_A to point p_B.





p_A, p49, p50, p58, p63, p80, p81, p_B

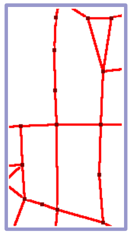
start lm_A (at p_A); right lm_C (at p49); lm_E left (at p63); left lm_D (at p80); end lm_B (at p_B)

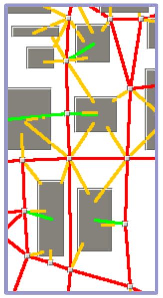
- A* search
- determine reorientation points
- pick landmarks for reorientation points
- pick landmarks for long straight segments

Content determination example: walking directions

User: how do I get from building A to building B?

Communicative goal: describe how to get from point p_A to point p_B.





p_A, p49, p50, p58, p63, p80, p81, p_B

start lm_A (at p_A); right lm_C (at p49); lm_E left (at p63); left lm_D (at p80); end lm_B (at p_B)

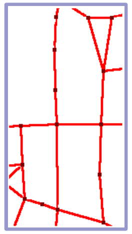
leave(lm_A), go_straight, turn(right, lm_C),
pass(lm_E,left), turn(left,lm_D), observe(lm_B,right)

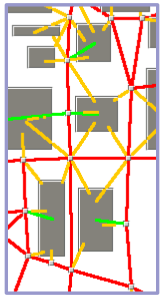
- A* search
- determine reorientation points
- pick landmarks for reorientation points
- pick landmarks for long straight segments
- map to a sequence of messages

Content determination example: walking directions

User: how do I get from building A to building B?

Communicative goal: describe how to get from point p_A to point p_B.





p_A, p49, p50, p58, p63, p80, p81, p_B

start lm_A — right lm_C — lm_E left — left lm_D — end lm_B

leave(lm_A), go_straight, turn(right, lm_C), pass(lm_E,left), turn(left,lm_D), observe(lm_B,right)

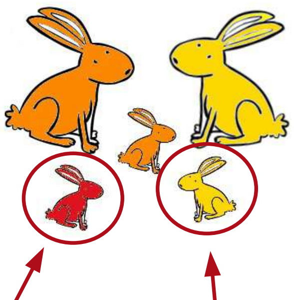
- A* search
- determine reorientation points
- pick landmarks for reorientation points
- pick landmarks for long straight segments
- map to a sequence of messages
- determine how to refer to landmarks

Today

- Mapping semantics to syntax
- Content determination
- Referring Expression Generation
- Multimodal referring expressions
  - generating pointing gestures
  - generating iconic gestures

Referring Expressions

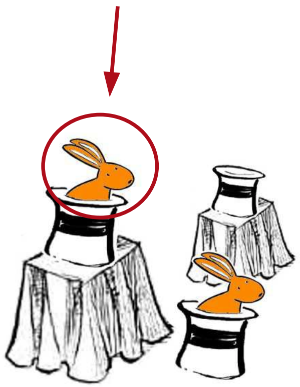
- linguistic expressions referring to objects or sets of objects
- NLG has focused on *definite descriptions*: expressions of the form '*the N*' that uniquely identifies an object in a given context



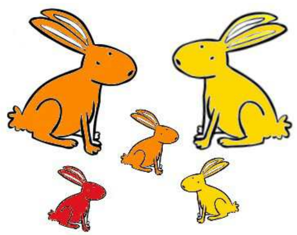
the red rabbit

the small yellow rabbit

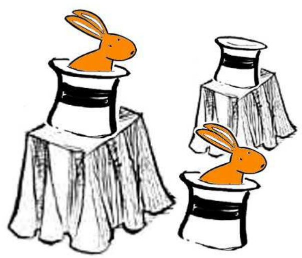
the rabbit in the hat on the table



Generating Definite Descriptions



Task: Find a description that uniquely identifies the target entity.



Lots of different algorithms

Dale 1992, Dale & Reiter 1995, Dale & Haddock 1991, Stone 2000, van Deemter 2002, Gardent 2002, Krahmer & Theune 2002, ...

Differences:

- expressivity; e.g. in terms of Description Logics:

Dale & Reiter (1995)	CL
van Deemter (2002a)	PL
Dale & Haddock (1991)	EL
Gardent (2002)	$ELU_{(\neg)}$
Krahmer et al. (2003)	EL + nominals (hybrid logic)

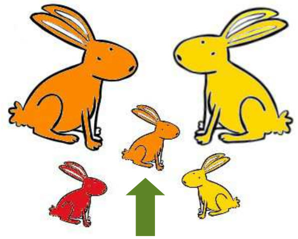
- representation of the description, strategy for constructing it, and way of determining success

Dale & Reiter: Incremental Algorithm

Input: a set of individuals with properties
a target entity
Output: a set of properties

target: r4

r1: rabbit, orange, big
r2: rabbit, yellow, big
r3: rabbit, red, small
r4: rabbit, orange, small
r5: rabbit, yellow, small



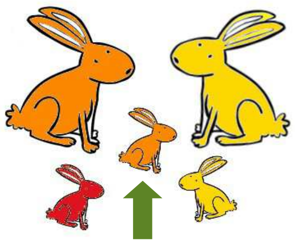
Dale & Reiter: Incremental Algorithm

Input: a set of individuals with properties
a target entity
Output: a set of properties
Algorithm: start with an empty set

target: r4

properties: ∅

r1: rabbit, orange, big
r2: rabbit, yellow, big
r3: rabbit, red, small
r4: rabbit, orange, small
r5: rabbit, yellow, small



Dale & Reiter: Incremental Algorithm

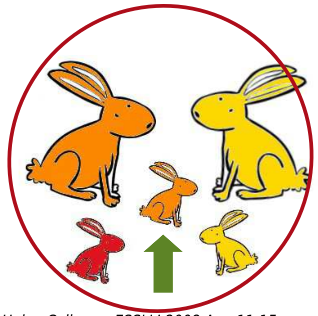
Input: a set of individuals with properties
a target entity
Output: a set of properties
Algorithm: start with an empty set
add properties until the target has no distractors (other individuals that fit the description)

target: r4

distractors: {r1, r2, r3, r5}

properties: ∅

r1: rabbit, orange, big
r2: rabbit, yellow, big
r3: rabbit, red, small
r4: rabbit, orange, small
r5: rabbit, yellow, small



Dale & Reiter: Incremental Algorithm

Input: a set of individuals with properties

a target entity

Output: a set of properties

Algorithm: start with an empty set

add properties until the target has no distractors (other individuals that fit the description)

consider properties in this order: type > color > size

target: r4

distractors: {r1, r2, r3, r5}

properties: ∅

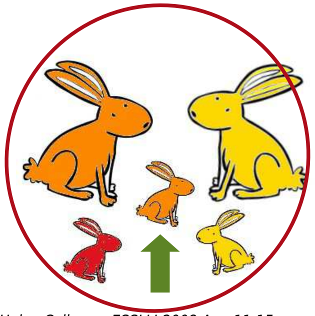
r1: rabbit, orange, big

r2: rabbit, yellow, big

r3: rabbit, red, small

r4: rabbit, orange, small

r5: rabbit, yellow, small



Dale & Reiter: Incremental Algorithm

Input: a set of individuals with properties

a target entity

Output: a set of properties

Algorithm: start with an empty set

add properties until the target has no distractors (other individuals that fit the description)

consider properties in this order: type > color > size

target: r4

distractors: {r1}

properties:
{orange}

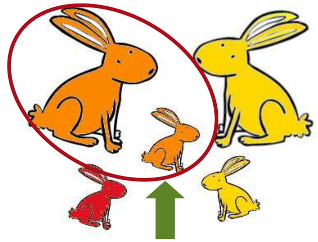
r1: rabbit, orange, big

r2: rabbit, yellow, big

r3: rabbit, red, small

r4: rabbit, orange, small

r5: rabbit, yellow, small



Dale & Reiter: Incremental Algorithm

Input: a set of individuals with properties

a target entity

Output: a set of properties

Algorithm: start with an empty set

add properties until the target has no distractors (other individuals that fit the description)

consider properties in this order: type > color > size

target: r4

distractors: ∅

properties:
{orange, small}

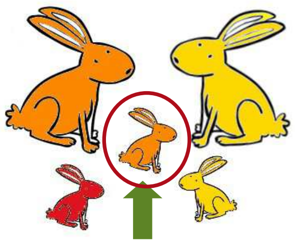
r1: rabbit, orange, big

r2: rabbit, yellow, big

r3: rabbit, red, small

r4: rabbit, orange, small

r5: rabbit, yellow, small



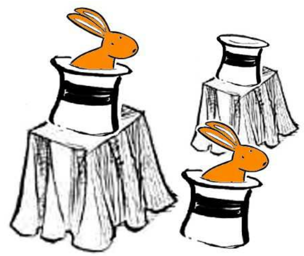
Dale & Haddock (1991): Extensions to relations

General strategy:

- maintain a stack of targets
- focus on the one at the top
- when adding a relation, push the new individual onto the stack

Problem:

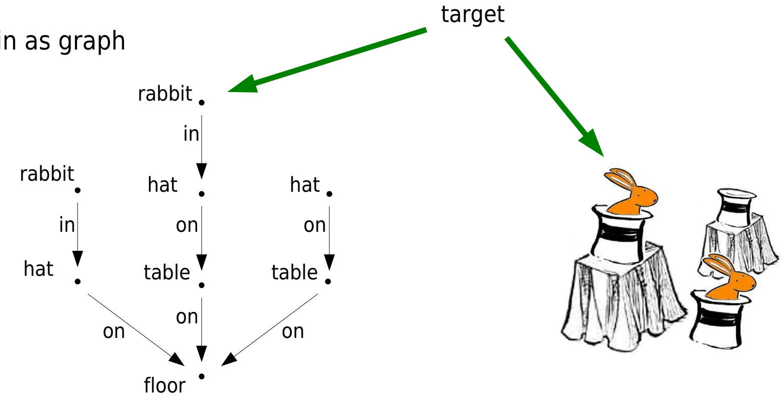
- infinite recursion



the rabbit in the hat containing the rabbit ...

Krahmer, Erk & Verleg: A graph based algorithm

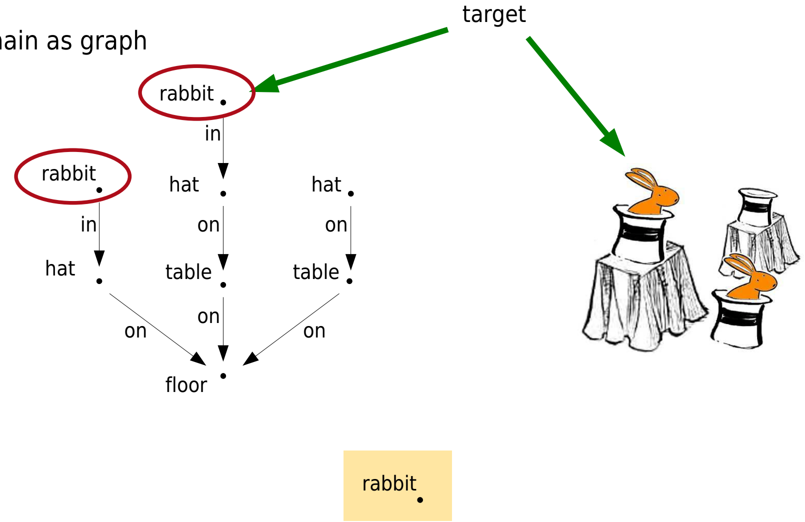
- domain as graph



- general idea: find a subgraph (covering the target) that can only be placed in the domain graph in one way

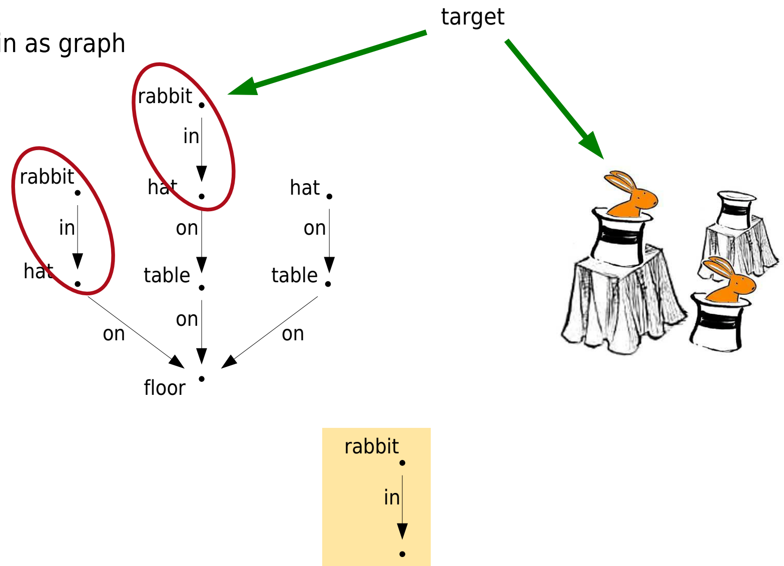
Krahmer, Erk & Verleg: A graph based algorithm

- domain as graph



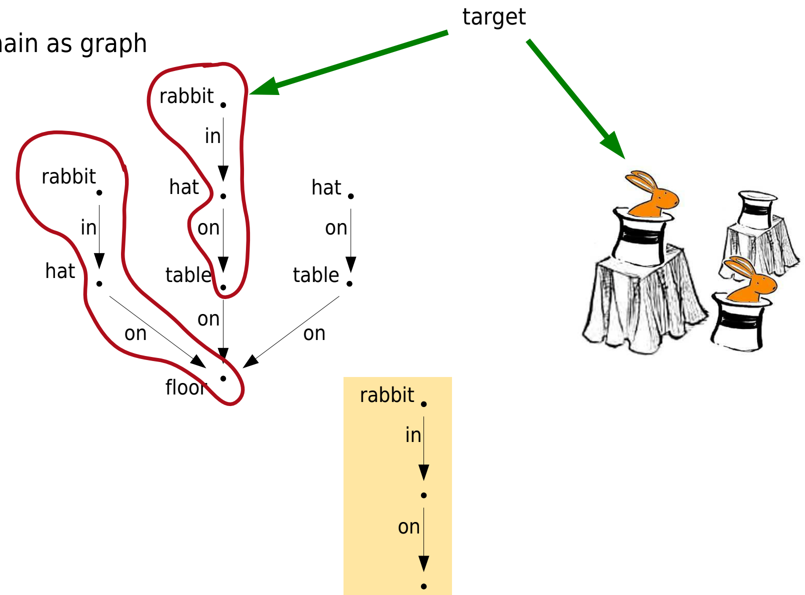
Krahmer, Erk & Verleg: A graph based algorithm

- domain as graph



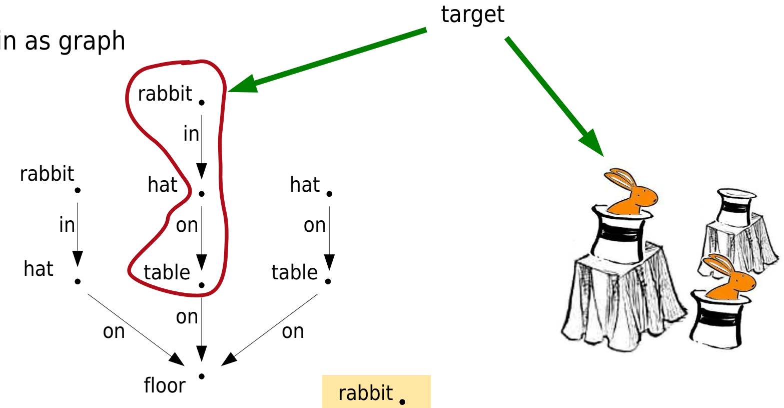
Krahmer, Erk & Verleg: A graph based algorithm

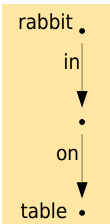
- domain as graph



Krahmer, Erk & Verleg: A graph based algorithm

- domain as graph

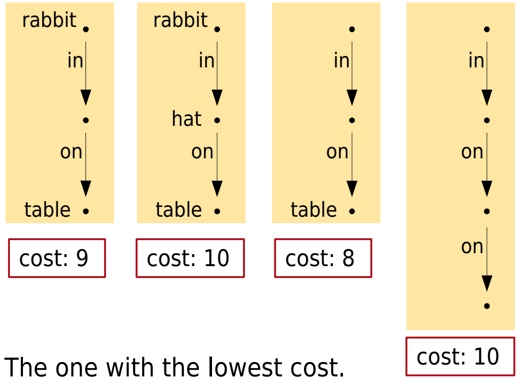


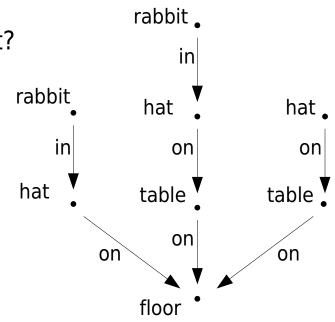


NOTE: no problem with relations.

Krahmer, Erk & Verleg: the cost of REs

- If there are several possibilities, which one is best?



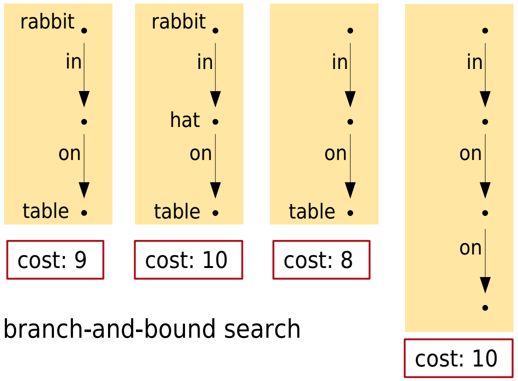


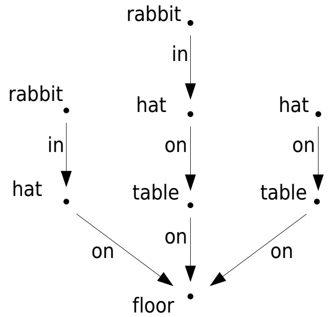
→ The one with the lowest cost.

E.g.: each vertex: 1, type properties: 1, relations: 2

- many different cost schemes are possible

Krahmer, Erk & Verleg: searching for the cheapest RE



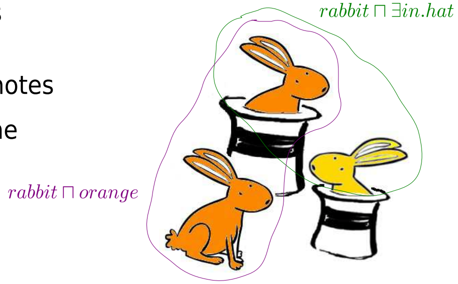


- branch-and-bound search
- Will always find cheapest RE.
- Which solution is found first depends on order in which subgraph is built.
- First solution gives a first upper bound on the cost which needs to be underbid by later solution candidates.

Areces, Koller & Striegnitz: Description logic formulas as REs

Background

- DL formulas denote sets of individuals
- REG = compute a DL formula that denotes exactly the singleton set containing the target

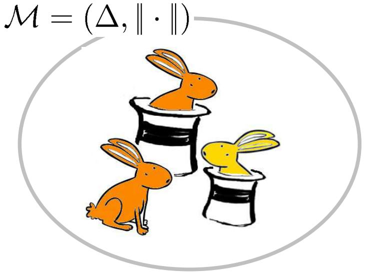


- One DL problem: given a model, find all groups of individuals that cannot be distinguished from each other through the logical language. (similarity sets)
- There are very efficient algorithms for computing similarity sets.
- Our approach: adapt such an algorithm for REG.

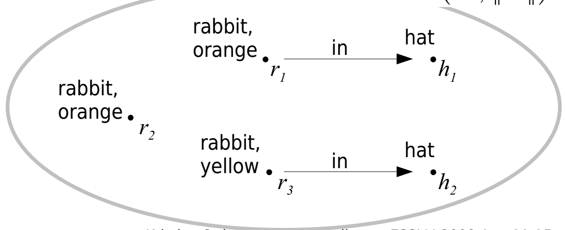
Description Logics

DL formulas	interpretation	example	
$\top$	Δ	$\top$	$\{ r_1, r_2, r_3, h_1, h_2 \}$
p	$\lVert p \rVert \subseteq \Delta$	$rabbit$	$\{ r_1, r_2, r_3 \}$
$\neg\varphi$	$\Delta - \lVert\varphi\rVert$	$\neg rabbit$	$\{ h_1, h_2 \}$
$\varphi \sqcap \varphi'$	$\lVert\varphi\rVert \cap \lVert\varphi'\rVert$	$rabbit \sqcap orange$	$\{ r_1, r_2 \}$
$\exists R.\varphi$	$\{i \mid \text{for some } i', i' \in \lVert\varphi\rVert \text{ and } (i,i') \in \lVert R \rVert\}$	$\exists in.hat$	$\{ r_1, r_3 \}$

$\mathcal{M} = (\Delta, \lVert\cdot\rVert)$



$\mathcal{M} = (\Delta, \lVert\cdot\rVert)$

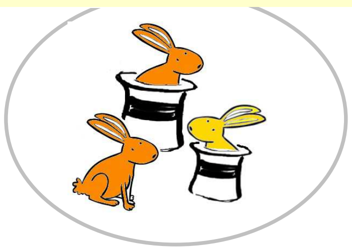


Description Logics

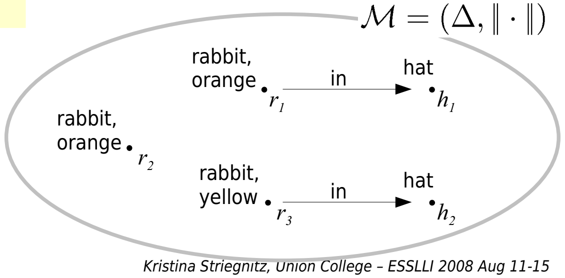
DL formulas	interpretation	example	
$\top$	Δ	$\top$	$\{ r_1, r_2, r_3, h_1, h_2 \}$
p	$\lVert p \rVert \subseteq \Delta$	$rabbit$	$\{ r_1, r_2, r_3 \}$
$\neg\varphi$	$\Delta - \lVert\varphi\rVert$	$\neg rabbit$	$\{ h_1, h_2 \}$
$\varphi \sqcap \varphi'$	$\lVert\varphi\rVert \cap \lVert\varphi'\rVert$	$rabbit \sqcap orange$	$\{ r_1, r_2 \}$
$\exists R.\varphi$	$\{i \mid \text{for some } i', i' \in \lVert\varphi\rVert \text{ and } (i,i') \in \lVert R \rVert\}$	$\exists in.hat$	$\{ r_1, r_3 \}$

EL

ACL

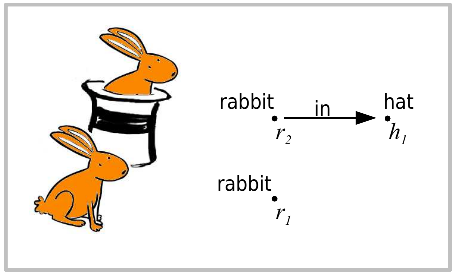


$\mathcal{M} = (\Delta, \lVert\cdot\rVert)$



$\mathcal{L}$-Similarity

Individual i is $\mathcal{L}$-similar to i' if there is **no** $\mathcal{L}$-formula that holds of i but not of i'.

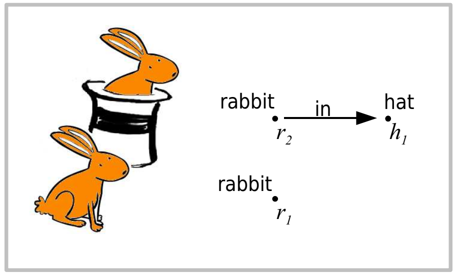


r_1 is EL-similar to r_2, but not vice versa.

r_1 is not ALC-similar to r_2.

$\mathcal{L}$-Similarity sets

The $\mathcal{L}$-similarity set of i is the set of all individuals to which i is $\mathcal{L}$-similar.



For every $\mathcal{L}$-similarity set there is an $\mathcal{L}$-formula that denotes exactly the individuals in the set.

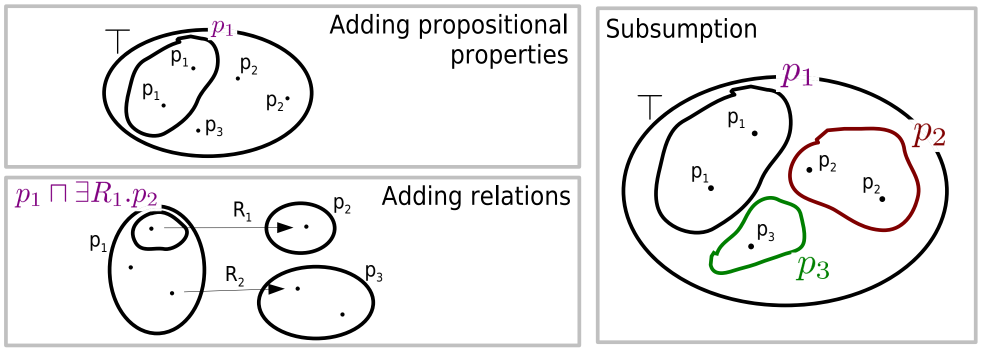
Computing EL-similarity sets

general idea: add properties that define smaller subsets

delete sets once they are subsumed by a set of smaller sets

continue until a) the result is a set of singletons or

b) no progress is made



Generating REs by computing similarity sets

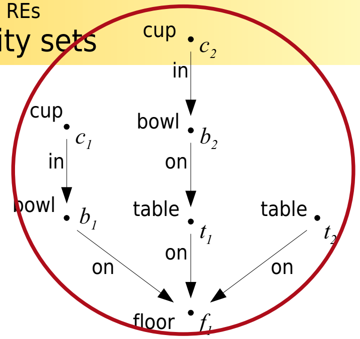
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

$RE \leftarrow \{\top\}$
for $p \in$ prop **do**
  add$_{\mathcal{L}}(p, RE)$
while *exists some* $\varphi \in RE, |\|\varphi\||^{\mathcal{M}} > 1$ **do**
  for $\varphi \in RE, R \in$ rel **do**
    add$_{\mathcal{L}}(\exists R.\varphi, RE)$
  if *made no changes to* RE **then**
    exit

Algorithm 3: add$_{\mathcal{EL}}(\varphi, RE)$

for $\psi \in RE$ *with* $|\|\psi\|| > 1$ **do**
  if $\psi \sqcap \varphi$ *is not subsumed in* RE **and**
  $\|\psi \sqcap \varphi\| \neq \emptyset$ **and** $\|\psi \sqcap \varphi\| \neq \|\psi\|$ **then**
    add $\psi \sqcap \varphi$ to RE
    remove subsumed formulas from RE



RE:

$\top$ $\{ c_1, c_2, b_1, b_2, t_1, t_2, f_1 \}$

Generating REs by computing similarity sets

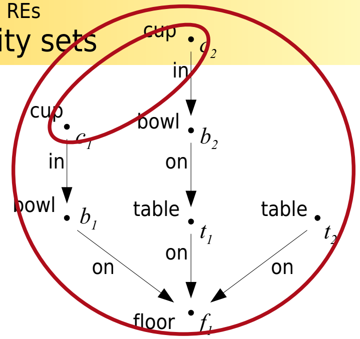
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

$RE \leftarrow \{\top\}$
for $p \in$ prop **do**
  add$_{\mathcal{L}}(p, RE)$
while *exists some* $\varphi \in RE, |\|\varphi\||^{\mathcal{M}} > 1$ **do**
  for $\varphi \in RE, R \in$ rel **do**
    add$_{\mathcal{L}}(\exists R.\varphi, RE)$
  if *made no changes to* RE **then**
    exit

Algorithm 3: add$_{\mathcal{EL}}(\varphi, RE)$

for $\psi \in RE$ *with* $|\|\psi\|| > 1$ **do**
  if $\psi \sqcap \varphi$ *is not subsumed in* RE **and**
  $\|\psi \sqcap \varphi\| \neq \emptyset$ **and** $\|\psi \sqcap \varphi\| \neq \|\psi\|$ **then**
    add $\psi \sqcap \varphi$ to RE
    remove subsumed formulas from RE



RE:

$\top$ $\{ c_1, c_2, b_1, b_2, t_1, t_2, f_1 \}$

cup $\{ c_1, c_2 \}$

Generating REs by computing similarity sets

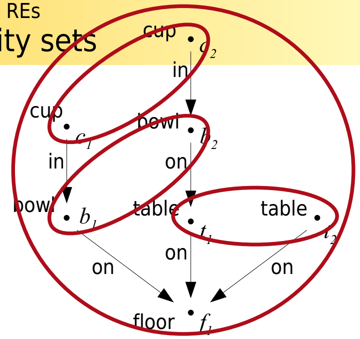
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

$RE \leftarrow \{\top\}$
for $p \in$ prop **do**
  add$_{\mathcal{L}}(p, RE)$
while *exists some* $\varphi \in RE, |\|\varphi\||^{\mathcal{M}} > 1$ **do**
  for $\varphi \in RE, R \in$ rel **do**
    add$_{\mathcal{L}}(\exists R.\varphi, RE)$
  if *made no changes to* RE **then**
    exit

Algorithm 3: add$_{\mathcal{EL}}(\varphi, RE)$

for $\psi \in RE$ *with* $\|\psi\| > 1$ **do**
  if $\psi \sqcap \varphi$ *is not subsumed in* RE **and**
  $\|\psi \sqcap \varphi\| \neq \emptyset$ **and** $\|\psi \sqcap \varphi\| \neq \|\psi\|$ **then**
    add $\psi \sqcap \varphi$ to RE
    remove subsumed formulas from RE



RE:

$\top$	$\{c_1, c_2, b_1, b_2, t_1, t_2, f_1\}$
cup	$\{c_1, c_2\}$
bowl	$\{b_1, b_2\}$
table	$\{t_1, t_2\}$

Generating REs by computing similarity sets

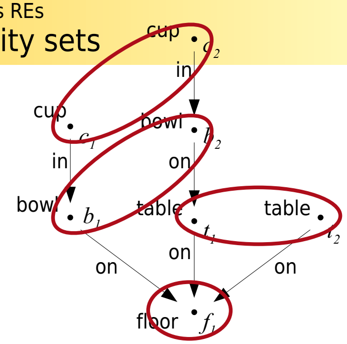
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

$RE \leftarrow \{\top\}$
for $p \in$ prop **do**
  add$_{\mathcal{L}}(p, RE)$
while *exists some* $\varphi \in RE, |\|\varphi\||^{\mathcal{M}} > 1$ **do**
  for $\varphi \in RE, R \in$ rel **do**
    add$_{\mathcal{L}}(\exists R.\varphi, RE)$
  if *made no changes to* RE **then**
    exit

Algorithm 3: add$_{\mathcal{EL}}(\varphi, RE)$

for $\psi \in RE$ *with* $\|\psi\| > 1$ **do**
  if $\psi \sqcap \varphi$ *is not subsumed in* RE **and**
  $\|\psi \sqcap \varphi\| \neq \emptyset$ **and** $\|\psi \sqcap \varphi\| \neq \|\psi\|$ **then**
    add $\psi \sqcap \varphi$ to RE
    remove subsumed formulas from RE



RE:

~~$\top$~~	~~$\{c_1, c_2, b_1, b_2, t_1, t_2, f_1\}$~~
cup	$\{c_1, c_2\}$
bowl	$\{b_1, b_2\}$
table	$\{t_1, t_2\}$
floor	$\{f_1\}$

Generating REs by computing similarity sets

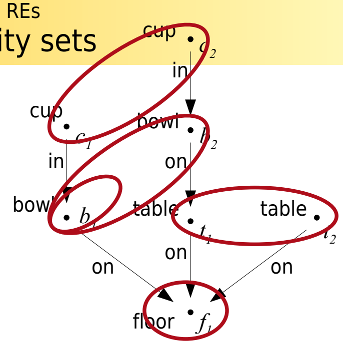
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

$RE \leftarrow \{\top\}$
for $p \in$ prop **do**
  add$_{\mathcal{L}}(p, RE)$
while *exists some* $\varphi \in RE, |\|\varphi\||^{\mathcal{M}} > 1$ **do**
  for $\varphi \in RE, R \in$ rel **do**
    add$_{\mathcal{L}}(\exists R.\varphi, RE)$
  if *made no changes to* RE **then**
    exit

Algorithm 3: add$_{\mathcal{EL}}(\varphi, RE)$

for $\psi \in RE$ *with* $\|\psi\| > 1$ **do**
  if $\psi \sqcap \varphi$ *is not subsumed in* RE **and**
  $\|\psi \sqcap \varphi\| \neq \emptyset$ **and** $\|\psi \sqcap \varphi\| \neq \|\psi\|$ **then**
    add $\psi \sqcap \varphi$ to RE
    remove subsumed formulas from RE



RE:

cup	$\{c_1, c_2\}$
bowl	$\{b_1, b_2\}$
table	$\{t_1, t_2\}$
floor	$\{f_1\}$
$bowl \sqcap \exists on.floor$	$\{b_1\}$

Generating REs by computing similarity sets

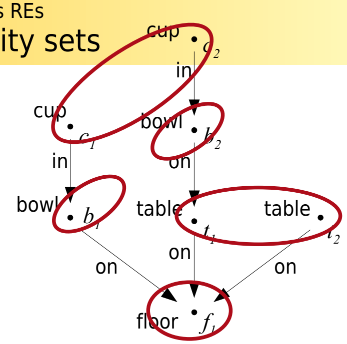
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

$RE \leftarrow \{\top\}$
for $p \in$ prop **do**
  add$_{\mathcal{L}}(p, RE)$
while *exists some* $\varphi \in RE, |\|\varphi\||^{\mathcal{M}} > 1$ **do**
  for $\varphi \in RE, R \in$ rel **do**
    add$_{\mathcal{L}}(\exists R.\varphi, RE)$
  if *made no changes to* RE **then**
    exit

Algorithm 3: add$_{\mathcal{EL}}(\varphi, RE)$

for $\psi \in RE$ *with* $\|\psi\| > 1$ **do**
  if $\psi \sqcap \varphi$ *is not subsumed in* RE **and**
  $\|\psi \sqcap \varphi\| \neq \emptyset$ **and** $\|\psi \sqcap \varphi\| \neq \|\psi\|$ **then**
    add $\psi \sqcap \varphi$ to RE
    remove subsumed formulas from RE



RE:

cup	$\{c_1, c_2\}$
~~*bowl*~~	~~$\{b_1, b_2\}$~~
table	$\{t_1, t_2\}$
floor	$\{f_1\}$
$bowl \sqcap \exists on.floor$	$\{b_1\}$
$bowl \sqcap \exists on.table$	$\{b_2\}$

Generating REs by computing similarity sets

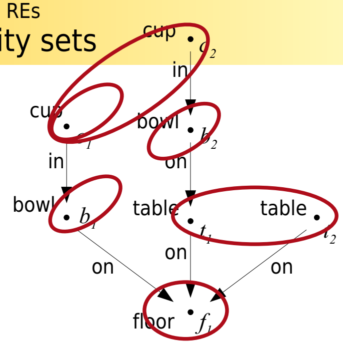
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

```
RE ← {⊤}
for p ∈ prop do
    add_L(p, RE)
while exists some φ ∈ RE, ||φ||^M > 1 do
    for φ ∈ RE, R ∈ rel do
        add_L(∃R.φ, RE)
    if made no changes to RE then
        exit
```

Algorithm 3: $\mathsf{add}_{\mathcal{EL}}(\varphi, RE)$

```
for ψ ∈ RE with ||ψ|| > 1 do
    if ψ ⊓ φ is not subsumed in RE and
    ||ψ ⊓ φ|| ≠ ∅ and ||ψ ⊓ φ|| ≠ ||ψ|| then
        add ψ ⊓ φ to RE
        remove subsumed formulas from RE
```



RE:

cup	$\{c_1, c_2\}$
$table$	$\{t_1, t_2\}$
$floor$	$\{f_1\}$
$bowl \sqcap \exists on.floor$	$\{b_1\}$
$bowl \sqcap \exists on.table$	$\{b_2\}$
$cup \sqcap \exists in.(bowl \sqcap \exists on.floor)$	$\{c_1\}$

Generating REs by computing similarity sets

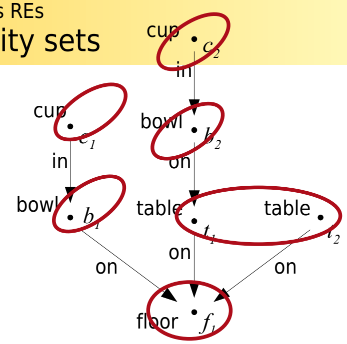
Algorithm 1: Computing the $\mathcal{L}$-similarity sets

Input: A model $\mathcal{M} = (\Delta, \|\cdot\|)$
Output: A set RE of formulas such that $\{\|\varphi\| \mid \varphi \in RE\}$ is the set of $\mathcal{L}$-similarity sets of $\mathcal{M}$.

```
RE ← {⊤}
for p ∈ prop do
    add_L(p, RE)
while exists some φ ∈ RE, ||φ||^M > 1 do
    for φ ∈ RE, R ∈ rel do
        add_L(∃R.φ, RE)
    if made no changes to RE then
        exit
```

Algorithm 3: $\mathsf{add}_{\mathcal{EL}}(\varphi, RE)$

```
for ψ ∈ RE with ||ψ|| > 1 do
    if ψ ⊓ φ is not subsumed in RE and
    ||ψ ⊓ φ|| ≠ ∅ and ||ψ ⊓ φ|| ≠ ||ψ|| then
        add ψ ⊓ φ to RE
        remove subsumed formulas from RE
```

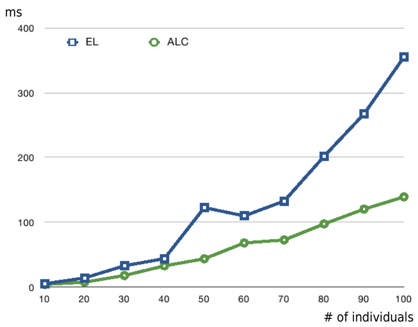


RE:

~~cup~~	~~$\{c_1, c_2\}$~~
$table$	$\{t_1, t_2\}$
$floor$	$\{f_1\}$
$bowl \sqcap \exists on.floor$	$\{b_1\}$
$bowl \sqcap \exists on.table$	$\{b_2\}$
$cup \sqcap \exists in.(bowl \sqcap \exists on.floor)$	$\{c_1\}$
$cup \sqcap \exists in.(bowl \sqcap \exists on.table)$	$\{c_2\}$

Features

- Res for all individuals are computed in parallel
- very efficient
- order of properties is the only way to control the resulting description
- lots of possible extensions using existing DL algorithms and results



Today

- Mapping semantics to syntax
- Content determination
- Referring Expression Generation
- Multimodal referring expressions
  - generating pointing gestures
  - generating iconic gestures

Multimodal referring expressions

Examples

Generating pointing gestures

- when to point
- how precisely to point (to object or to region)
- what info to put into the accompanying language

Some work on generating pointing gestures:

Claasen (1992)

Lester et al. (1999)

Kranstedt & Wachsmuth (2005)

van der Sluis & Krahmer (2007) 🡐 extends the graph-based algorithm

Lester et al. 1999

- Can the referent easily be confused with other objects?

  Are there are recently mentioned objects nearby?

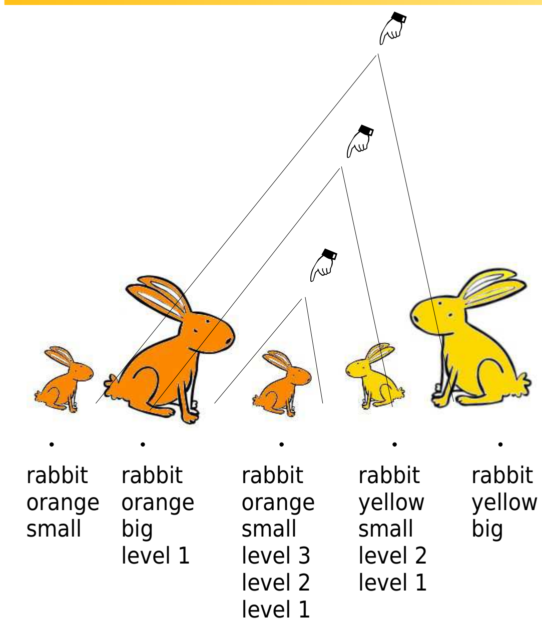
  Are there are other objects of the same type nearby?

  Is the target referent particularly small?

  If so, point.

- Pointing is always unambiguous. If necessary, the agent moves toward the object to point.

van der Sluis & Krahmer: REs + pointing gestures



- different levels of pointing are represented as labels in the domain graph
- what level of pointing and what linguistic material is chosen depends on the costs
- cost of pointing depends on size of target and on the distance the hand has to move:

  cost(pointing) = $\log_2(D/W + 1)$

Generating iconic gestures

- when to use iconic gesture
- what gesture to use

When to use iconic gestures

- with rhematic material (roughly: material that contributes new information to the discourse) (Cassell 2000)
- dependent on domain:
  - to express shape and location in object descriptions (Yan 2000)
  - to express path, manner and speed in motion descriptions (Cassell & Prevost 1996)

What gesture to use

- most commonly: use a gesticon

  a collection of pre-animated gestures associated with specific semantic meanings

- alternatively: generate gestures on the fly (based on geometric and visible properties of the referent)

NUMACK: Generating gestures on the fly

Goal: Generate gestures on the fly based on information about the referent

Domain: Giving walking directions (across Northwestern University's campus)

gestures referring to landmarks

Iconic Gestures

- Iconic gestures visually resemble what they depict.
- They encode information that may be redundant with the content of the accompanying speech or may add to it.
- No stable form-meaning pairing:
  - same gesture can be used to refer to different things
  - same thing can be referred to using different gestures
  - gesture on its own is insufficient for interpretation
- Iconic gestures are interpreted in context (speech, previous discourse, domain, dialogue situation) to depict specific entities.

"it's got [like steeples]"

"there's a church"

Gestures referring to landmarks

- Functions:
  - locating landmarks
  - depicting shape of landmarks
- Many gestures have both a locating and a shape depicting component.
- Speakers take on different perspectives when describing routes.

"on your left once you hit this parking lot [is the Allen Center]"

"it's got [like steeples]"

Route perspective gestures

- Direction giver takes on perspective of a person walking the route.
- Gestures locate landmarks with respect to this imaginary direction follower's position and orientation.
- Most common type of gesture for referring to landmarks: 54%.

"on your left once you hit this parking lot [is the Allen Center]"

Survey perspective gestures

- Gestures lay out a map in front of the speaker's body.
- Landmarks are located with respect to the imaginary direction follower's body and relative to other landmarks.
- 16% of all gestures referring to landmarks.

'[University Hall'll] be on your right, [on the left is Kresge], and [then straight ahead is Harris]"

Non-locating gestures

- do not locate landmarks
- depict shape
- 16% of all gestures referring to landmarks.

"on your left once you hit this parking lot [is the Allen Center]..."

"...and [it's really big]"

Questions

- When should we use which perspective?
- How is location and shape information depicted in the gesture?

Gesture perspective in the data

Gesture perspective seems to be (at least partly) determined by dialogue function.

- Non-locating gestures tend to occur in elaborations.
- Survey perspective gestures tend to occur in answers to clarification questions and in re-descriptions of route segments.
- Non-locating and survey perspective gestures tend to not occur in plain forward looking statements.
- Route perspective gestures tend to occur in plain statements.
- They tend to not occur in answers to clarification questions, re-descriptions of route segments, or elaborations.

Gesture perspective in the system

- Non-locating gestures are used in elaborations which don't mention the location of the landmark.

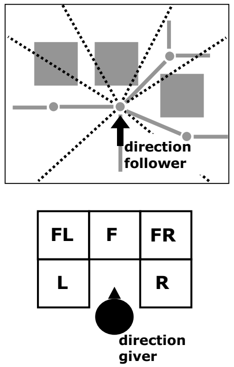
  E.g.: "*Dearborn Observatory is on your left. It is a building with a dome.*"

- Survey perspective gestures are used for re-descriptions of route segments at "difficult" reorientation points.
- Route perspective is used for all other gestures.

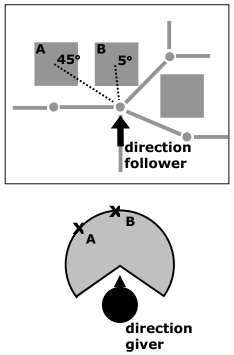
Expressing location in iconic gestures

- Given the position and orientation that a person walking the route would have at the current point of the directions, calculate the angle to the referent(s).
- Map those angles to positions in the gesture space.

route perspective

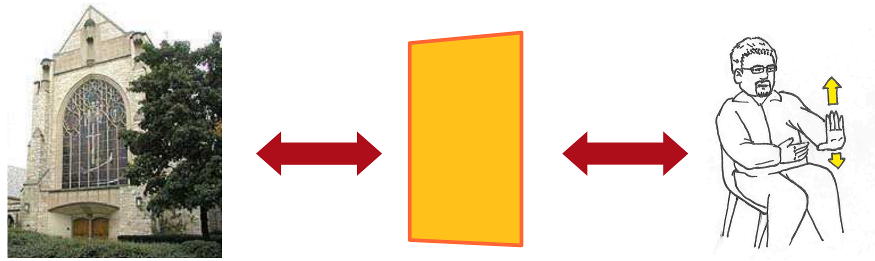


survey perspective



Expressing shape in iconic gestures

Hypothesis: gesture morphology is related to visual and spatial properties of the referent.



Kristina Striegnitz, Union College – ESSLLI 2008 Aug 11-15

Some evidence in the data

- gestures in 10 direction giving dialogues coded for gesture morphology (hand shape, hand position, palm direction, extended finger direction)
- landmarks these gestures refer to were coded for salient visual features
- looked at flat handshapes
- hypotheses:

  1) palm down ⇔ horizontal surface

  2) fingers up ⇔ vertical surface

  3) fingers forward & palm sideways ⇔ path

- confirmed hypothesis 2 and 3

Kristina Striegnitz, Union College – ESSLLI 2008 Aug 11-15

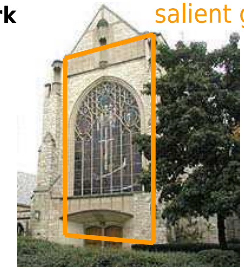
Problems with the study

- landmarks may have more than one visually salient feature
  - did not take into account discourse context
  - did not take into account direction from which landmark was approached
- did not differentiate between perspectives

Kristina Striegnitz, Union College – ESSLLI 2008 Aug 11-15

Gesture Planning

referent landmark



salient geometric form

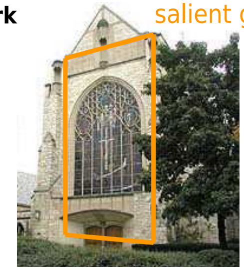
image description features

shape: *plane*
orientation: *vertical & orthogonal to DF's orient*
primary axis: *vertical*
location wrt. DF: *-10°*

perspective: *route*

Gesture Planning

referent landmark



salient geometric form

image description features

gesture form features

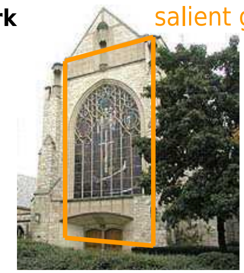
shape: *plane*
orientation: *vertical & orthogonal to DF's orient*
primary axis: *vertical*
location wrt. DF: *-10°*

perspective: *route*

handshape: *flat (ASL B or 5)*
palm direction: *away from body*
finger direction: *up*
trajectory: *linear up*
hand location: *center left & arm stretched*

Gesture Planning

referent landmark



salient geometric form

image description features

gesture form features

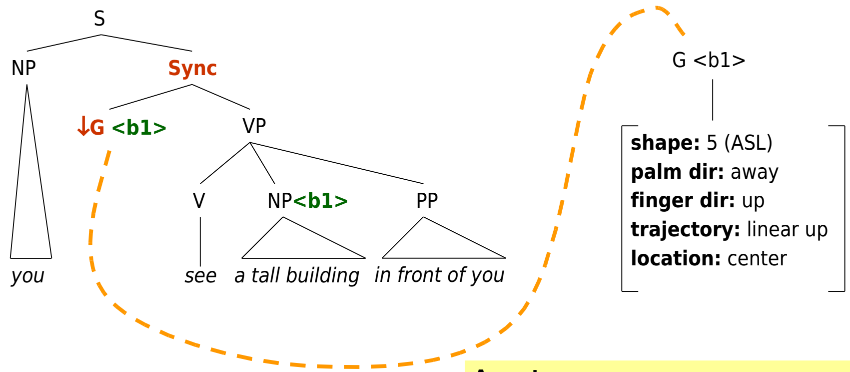
shape: *plane*
orientation: *vertical & orthogonal to DF's orient*
primary axis: *vertical*
location wrt. DF: *-10°*

perspective: *route*

handshape: *flat (ASL B or 5)*
palm direction: *away from body*
finger direction: *up*
trajectory: *linear up*
hand location: *center left & arm stretched*

Integrating gesture planning and utterance construction

Goal: *describe event e1*



Assert:
rel_loc(e1,b1,df,front)
building(b1)
tall(b1)

+

Assert:
shape(*b1, plane*)
orientation(*b1, vertical, orthogonal to DF*)
primary_axis(*b1, vertical)*
rel_location(*b1, front)*

Today

- Mapping semantics to syntax
- Content determination
- Referring Expression Generation
- Multimodal referring expressions
  - generating pointing gestures
  - generating iconic gestures

Tomorrow: discourse and dialogue phenomena