Serial Dependency: Is It a Characteristic of Human REG?

Jette Viethen, Robert Dale and Markus Guhe

Our Message in Short

Our Message in Short

- Traditional REG algorithms choose attributes one-by-one in a serially dependent fashion.

Our Message in Short

- Traditional REG algorithms choose attributes one-by-one in a serially dependent fashion.

- This is not what people do.

Outline

1. REG and Serial Dependency

2. The iMAP Corpus

3. Modelling Human Reference Behaviour

    • Basic setup

    • Investigating serial dependency

4. Results and Conclusions

Outline

Referring Expression Generation (REG)

Referring Expression Generation (REG)

... means automatically building
referring expressions.

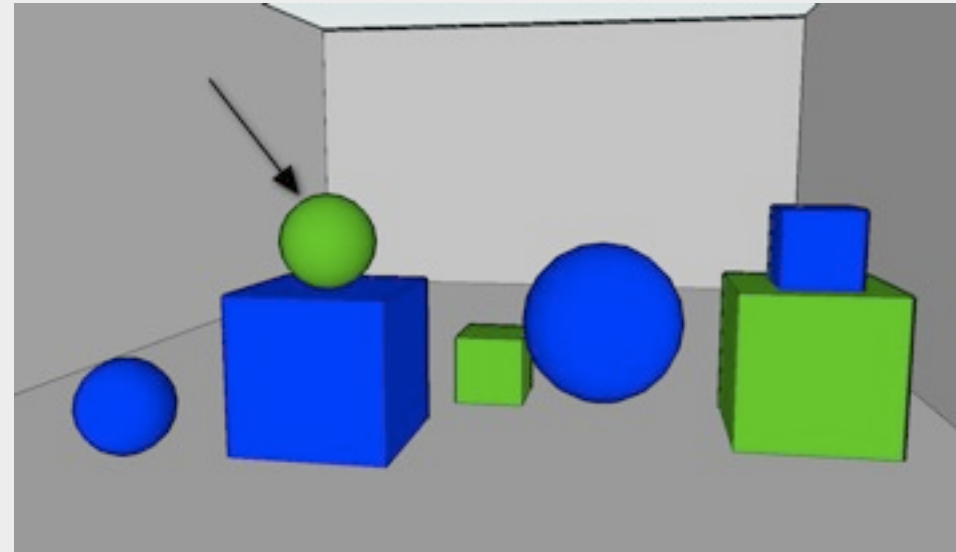# Referring Expression Generation (REG)

... means automatically building
referring expressions.

duh!

Referring Expression Generation (REG)

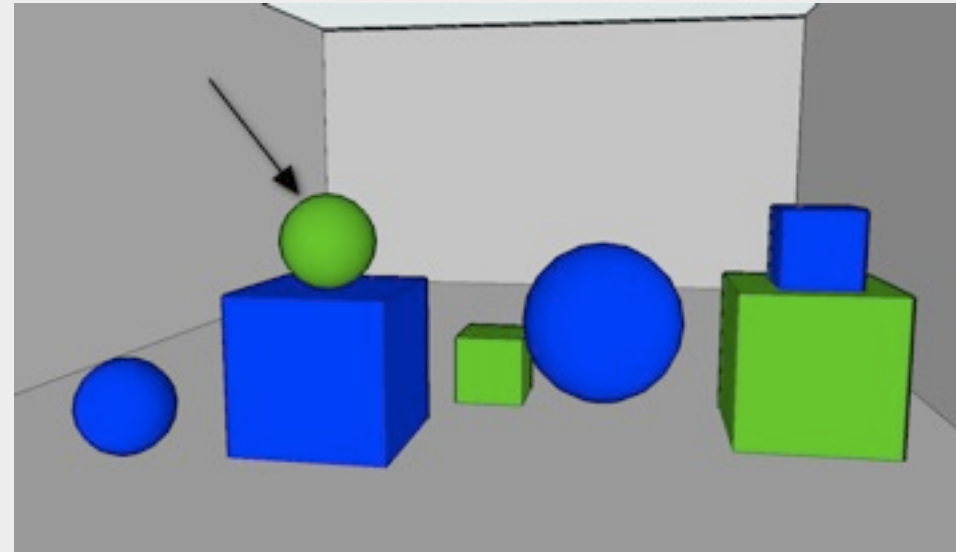... means automatically building referring expressions.

duh!

Referring Expression Generation (REG)

... means automatically building referring expressions.

duh!

the small green ball

Referring Expression Generation (REG)

... means automatically building referring expressions.

duh!

the small green ball

- **Target Referent**: object to be described
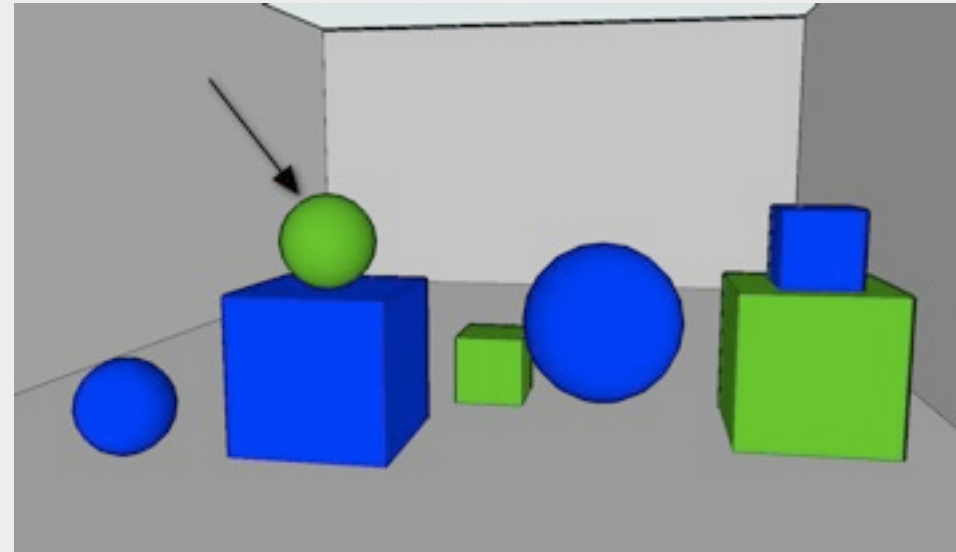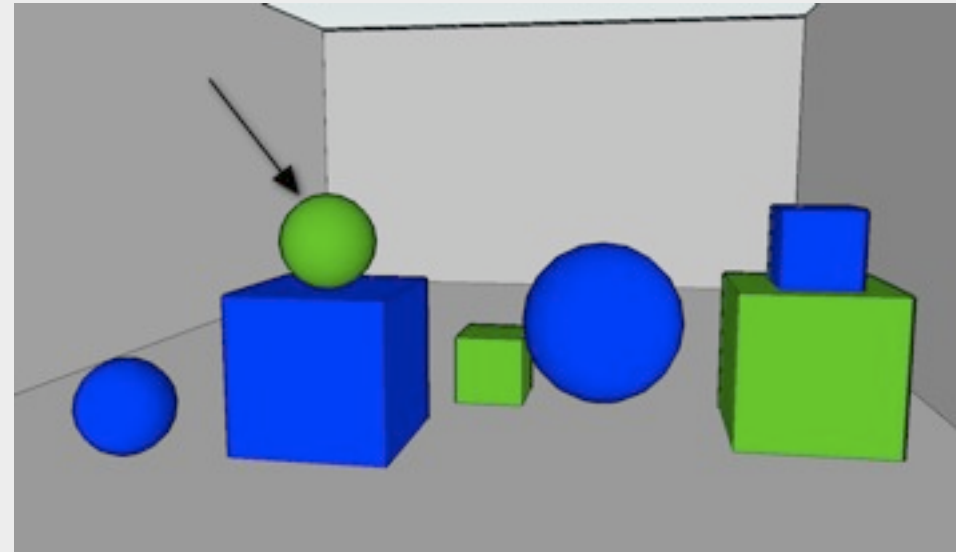
Referring Expression Generation (REG)

... means automatically building referring expressions.
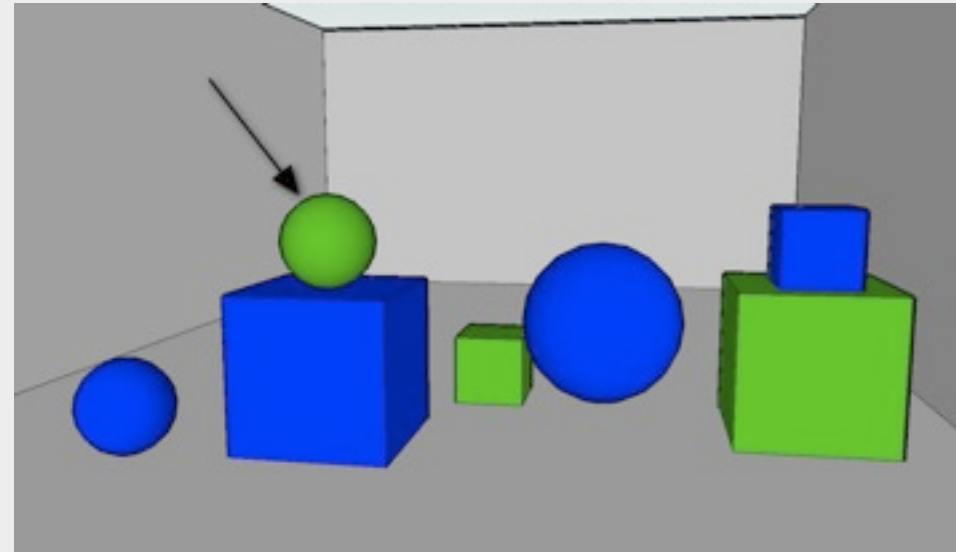
duh!

the small green ball

- **Target Referent**: object to be described

- **Distractors**: other objects in the environment that the target needs to be distinguished from

Referring Expression Generation (REG)

... means automatically building referring expressions.

duh!

the small green ball

- **Target Referent**: object to be described

- **Distractors**: other objects in the environment that the target needs to be distinguished from

- **Content Selection** from the attributes of the target and its relations to other objects (no linguistic realisation)
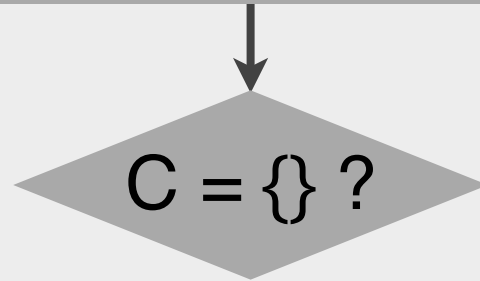
Serial Dependency in Traditional REG Algorithms
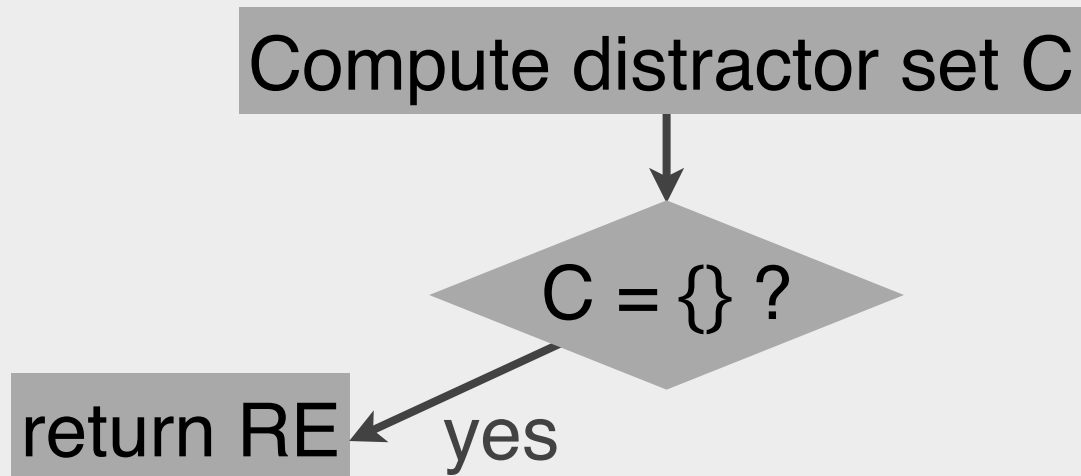
Serial Dependency in Traditional REG Algorithms

Compute distractor set C

Serial Dependency in Traditional REG Algorithms

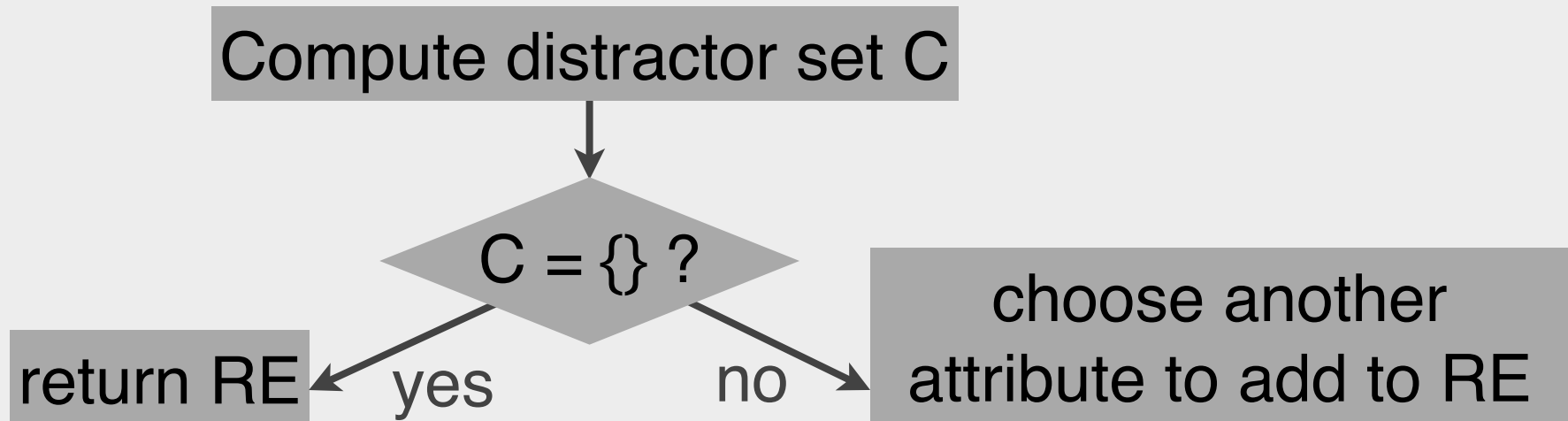Compute distractor set C

C = {} ?

Serial Dependency in Traditional REG Algorithms

Serial Dependency in Traditional REG Algorithms
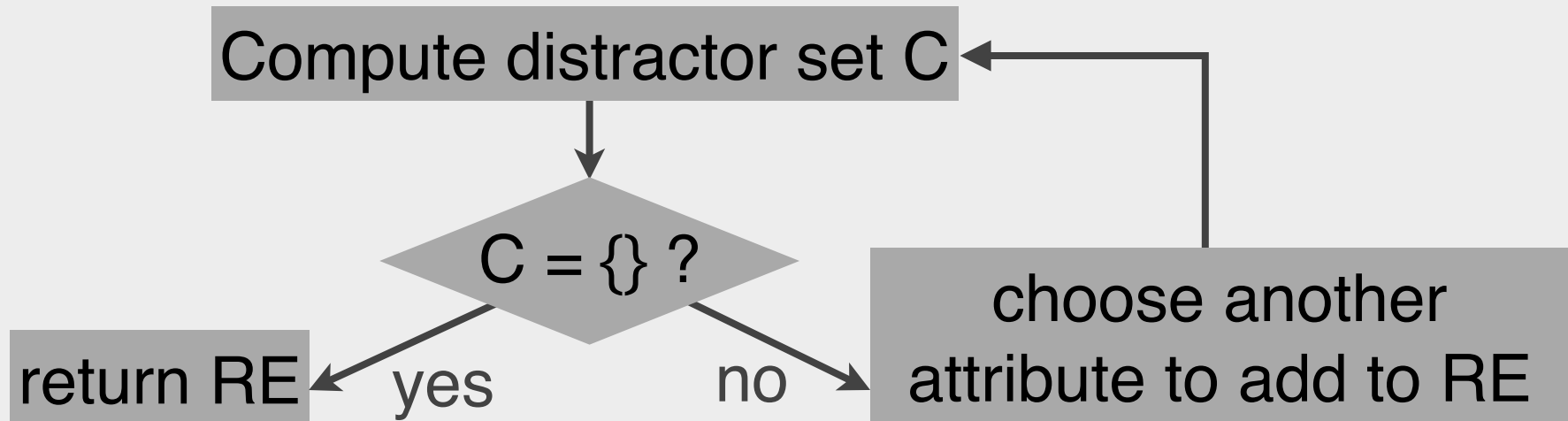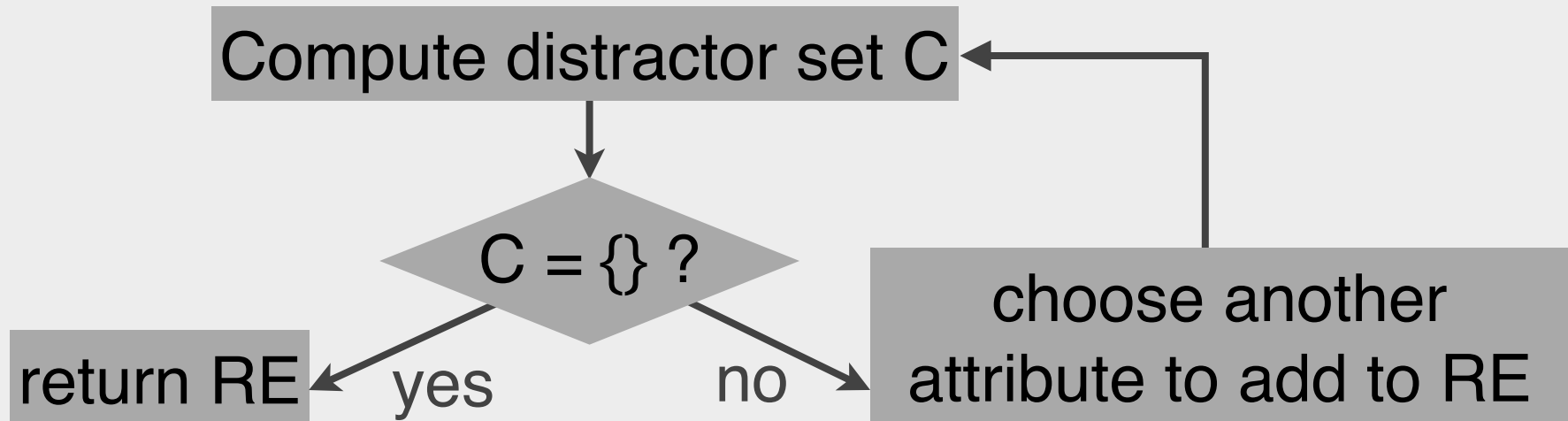
Serial Dependency in Traditional REG Algorithms

Serial Dependency in Traditional REG Algorithms

e.g.: Dale's (1989) Greedy and Dale and Reiter's (1995) Incremental Algorithm

Serial Dependency in Traditional REG Algorithms
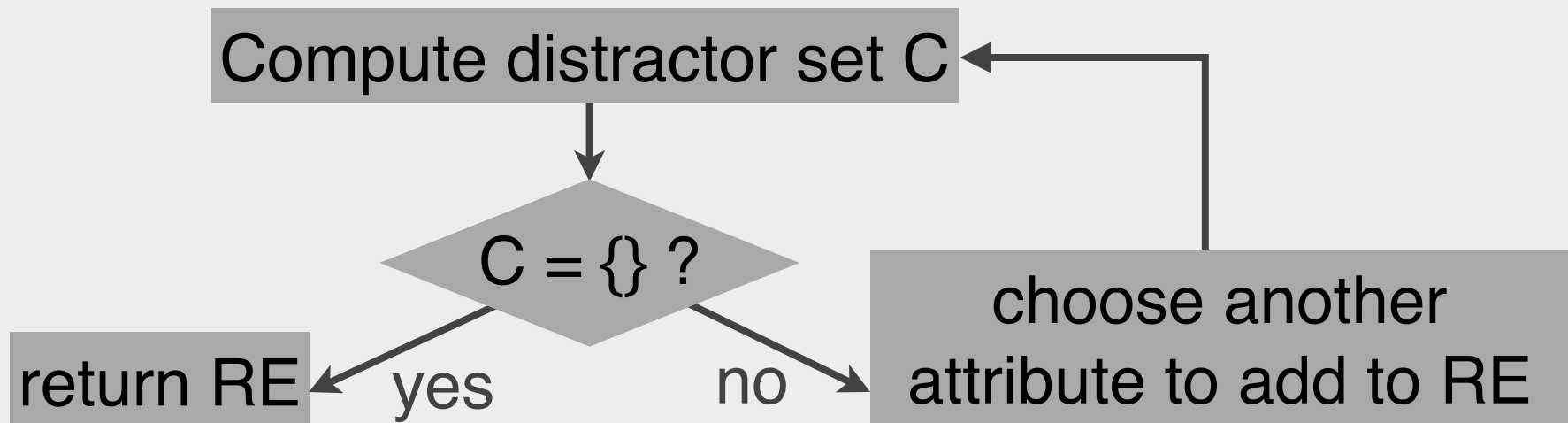
e.g.: Dale's (1989) Greedy and Dale and Reiter's (1995) Incremental Algorithm

- Distractor set C is dependent on the attributes already chosen.

Serial Dependency in Traditional REG Algorithms
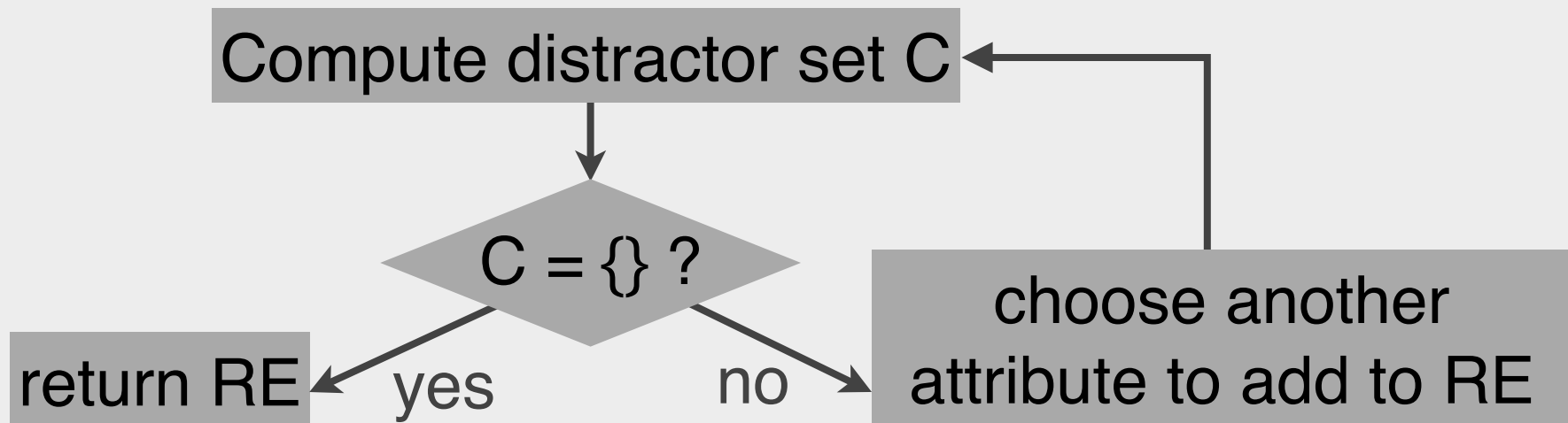
Compute distractor set C

C = {} ?

yes → return RE

no → choose another attribute to add to RE

e.g.: Dale's (1989) Greedy and Dale and Reiter's (1995) Incremental Algorithm

- Distractor set C is dependent on the attributes already chosen.
- Choice of the next attribute is dependent on how many distractors it rules out.

Outline

1. REG and Serial Dependency
2. **The iMAP Corpus**
3. Modelling Human Reference Behaviour

    • Basic setup

    • Investigating serial dependency

4. Results and Conclusions

Referring Expressions in the iMAP Corpus
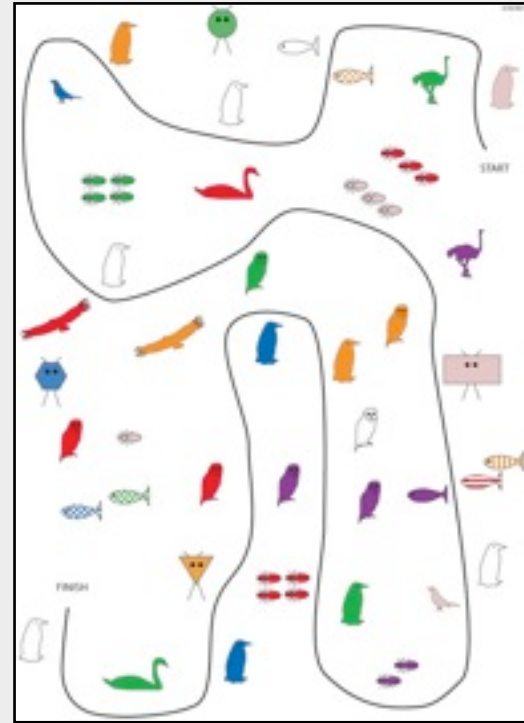
The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues

The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues
- 8 different types of landmarks

The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues
- 8 different types of landmarks
- landmarks distinguishable by
  - type
  - colour
  - one "other" attribute

The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues
- 8 different types of landmarks
- landmarks distinguishable by
  - type
  - colour
  - one "other" attribute

The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues
- 8 different types of landmarks
- landmarks distinguishable by
  - type
  - colour
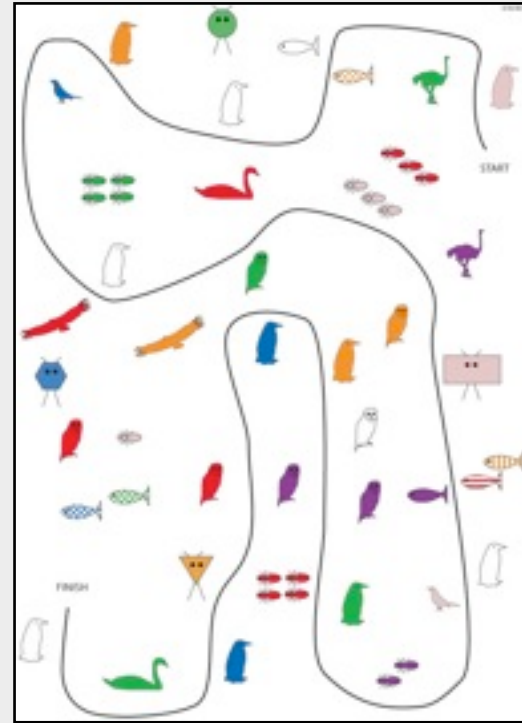 - one "other" attribute

The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues
- 8 different types of landmarks
- landmarks distinguishable by
  - type
  - colour
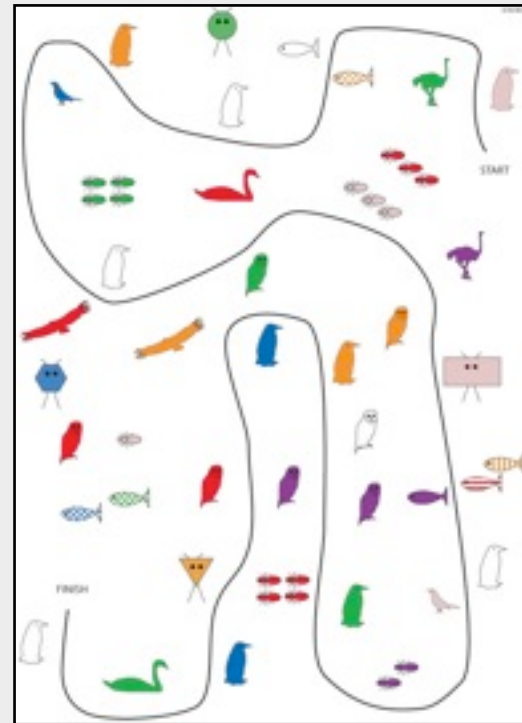 - one "other" attribute

The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues
- 8 different types of landmarks
- landmarks distinguishable by
  - type
  - colour
  - one "other" attribute

- 34,126 REs in the corpus
- we exclude those that
  - use a spatial relation;
  - refer to more than one landmark;
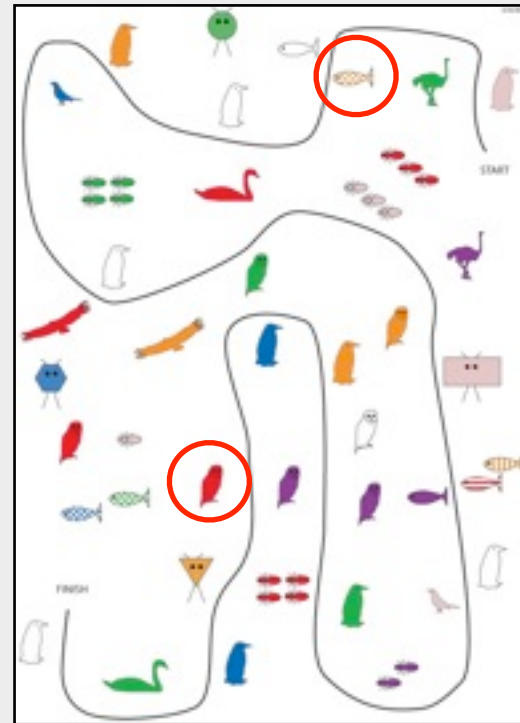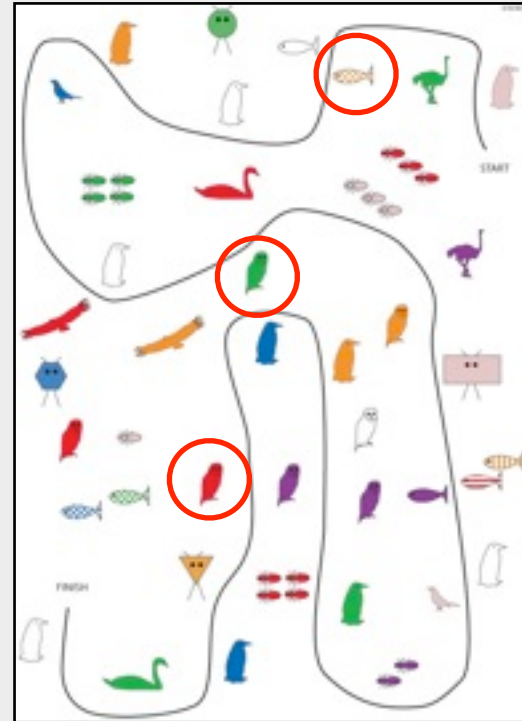 - don't use any of the three 'standard' attributes.
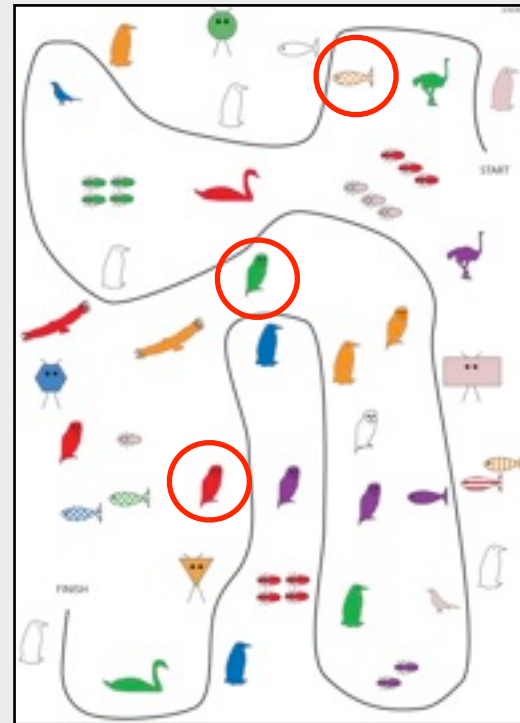

The map of the
Instruction Giver

The map of the
Instruction Follower

Referring Expressions in the iMAP Corpus

- 256 dialogues
- 8 different types of landmarks
- landmarks distinguishable by
  - type
  - colour
  - one "other" attribute

- 34,126 REs in the corpus
- we exclude those that
  - use a spatial relation;
  - refer to more than one landmark;
  - don't use any of the three 'standard' attributes.

➝ 20,141 REs (5936 initial, 14205 subsequent)
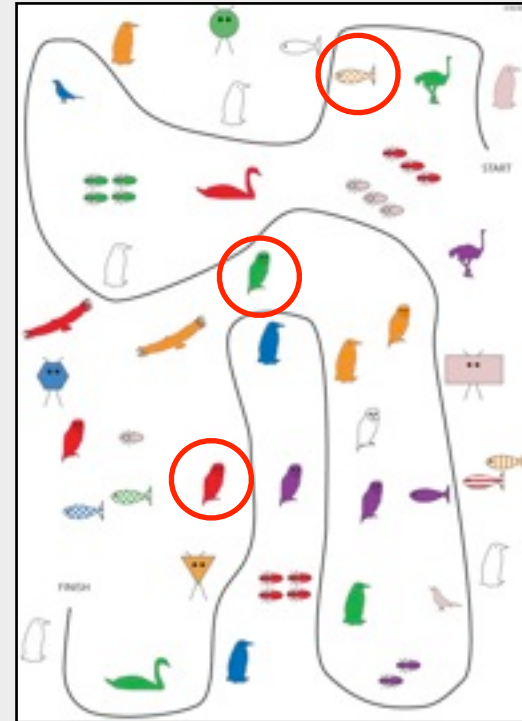

The map of the
Instruction Giver

The map of the
Instruction Follower

Content Patterns
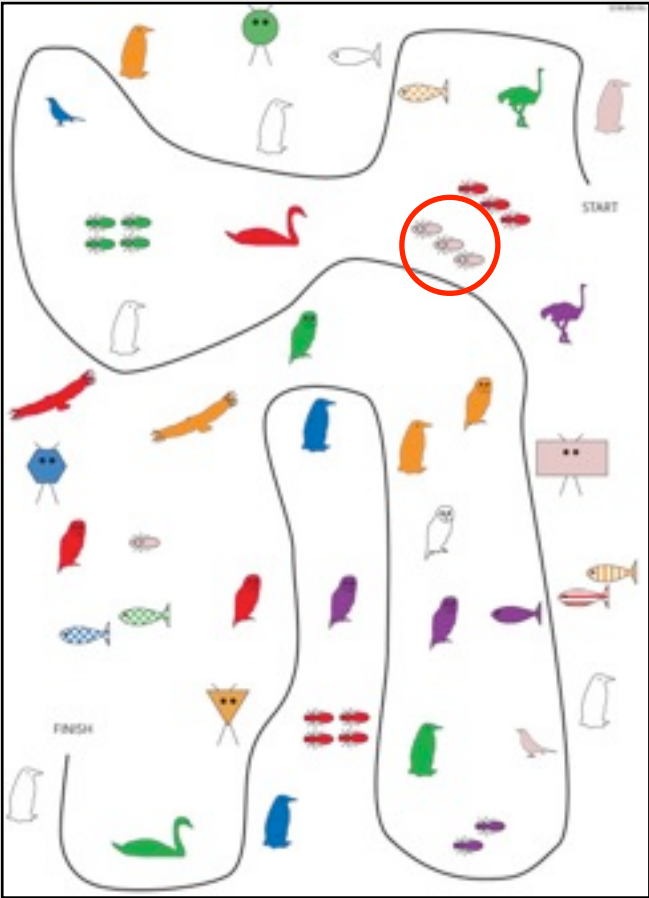
Content Patterns

Go beneath the three pink bugs.

Content Patterns

Go beneath the three pink bugs.

↓

`<other, col, type>`

Content Patterns

Go beneath the three pink bugs.

$$\downarrow$$

`<other, col, type>`

Turn towards the bottom just before the purple ostrich...

Content Patterns

Go beneath the three pink bugs.

<other, col, type>

Turn towards the bottom just before the purple ostrich...

<other, col>

Content Patterns

Go beneath the three pink bugs.

<other, col, type>

Turn towards the bottom just before the purple ostrich...

<other, col>

... between the owl and the alien.

Content Patterns

Go beneath the `three pink bugs`.

↓

`<other, col, type>`

Turn towards the bottom just before the `purple ostrich`...

↓

`<other, col>`

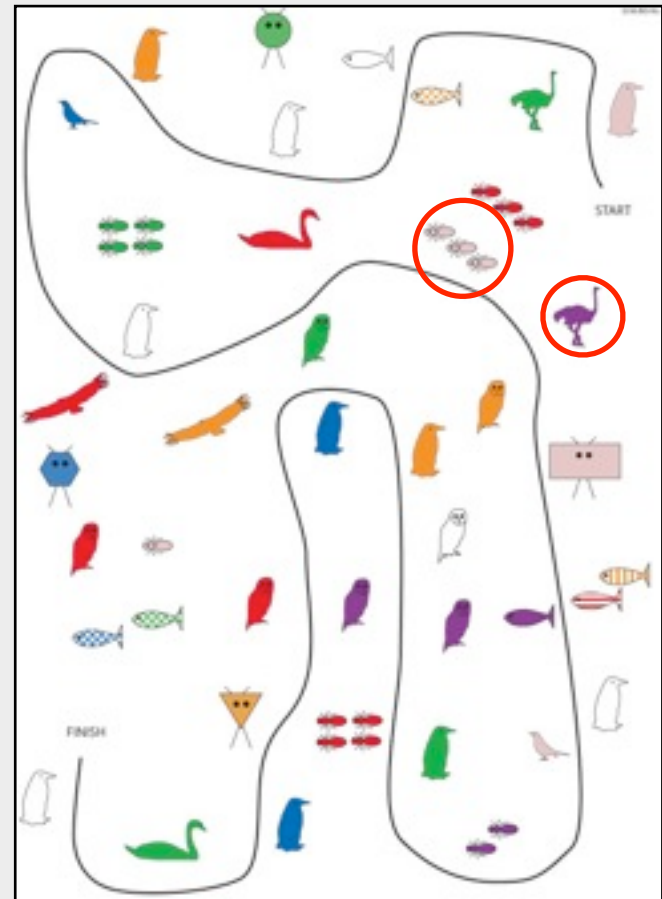... between the `owl` and the `alien`.

↓

`<other>`

Content Patterns

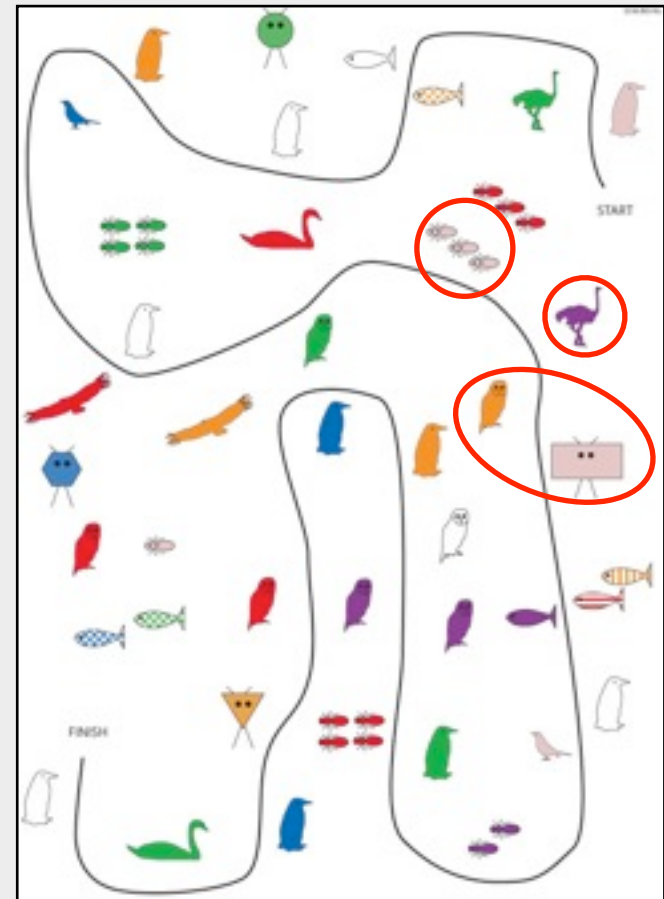Go beneath the three pink bugs.

⬇

<other, col, type>

Turn towards the bottom just before the purple ostrich...

⬇

<other, col>
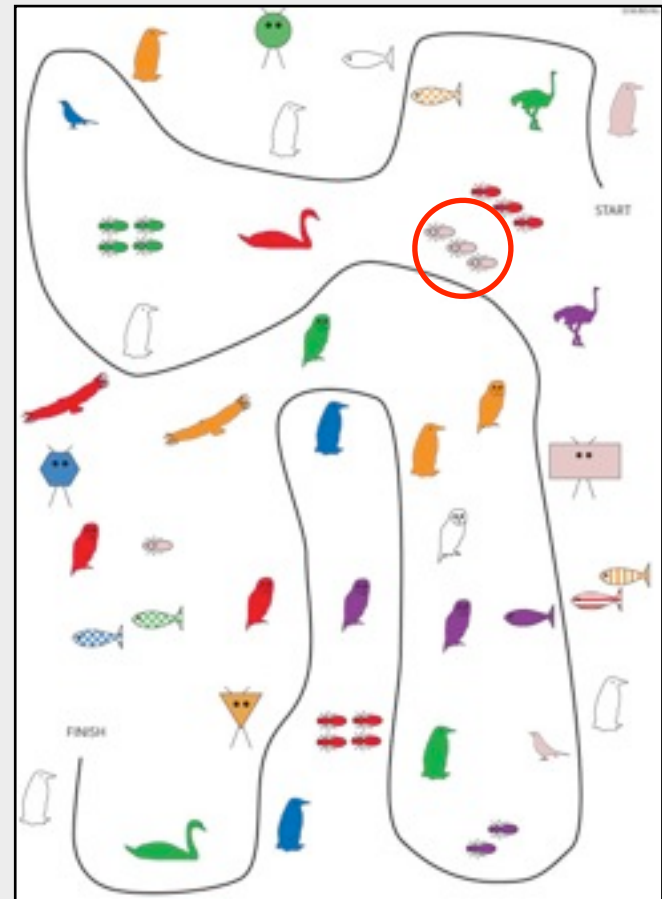
... between the owl and the alien.

⬇   ⬇

<other>   <type>

Outline

1. REG and Serial Dependency
2. The iMAP Corpus
3. Modelling Human Reference Behaviour

    • Basic setup

    • Investigating serial dependency

4. Results and Conclusions

Learning which Content Pattern to Use

- all attributes in parallel
- first independently of each other
- then in serial dependency
- C4.5 the decision trees
- 70–30 training–test set split

Learning which Content Pattern to Use

- all attributes in parallel
- first independently of each other
- then in serial dependency
- C4.5 the decision trees
- 70–30 training–test set split

```
<other>
YES  OR  NO
```

```
<col>
YES  OR  NO
```

```
<type>
YES  OR  NO
```

Learning which Content Pattern to Use

- all attributes in parallel
- first independently of each other
- then in serial dependency
- C4.5 the decision trees
- 70–30 training–test set split

Learning which Content Pattern to Use

- all attributes in parallel
- first independently of each other
- then in serial dependency
- C4.5 the decision trees
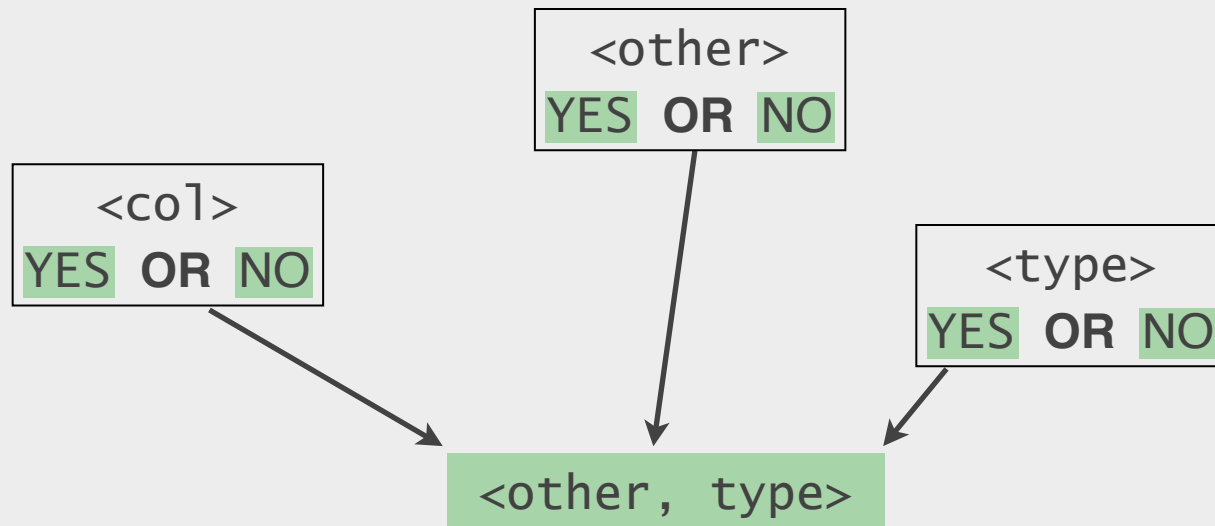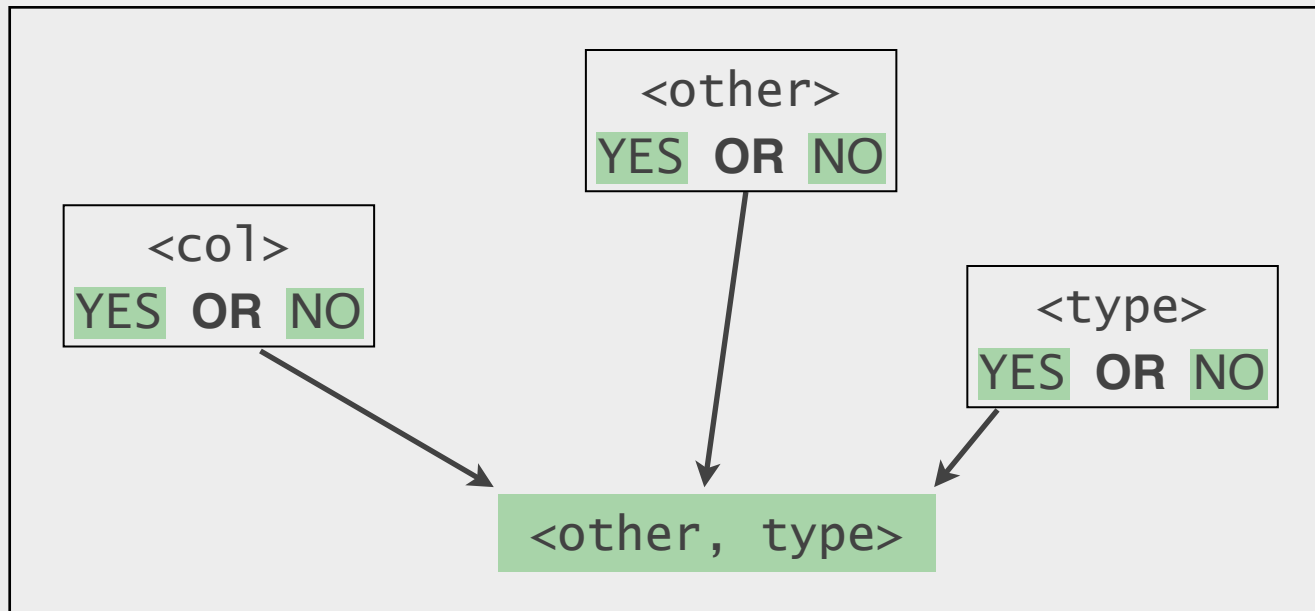- 70–30 training–test set split

Non-Chaining Features (NC)

- Traditional REG features:
  - Count_distractors, Distance_Closest, Closest_Same_*Att* ...
  - Prop_Same_*Att* (1-NC)

- Alignment features:
  - Last_Mention_*Att*, Distance_Last_Mention, Distance_Last_*Att*,
  - Count_*Att*_Used, Quartile, Mention_No, Dialogue_No ...

- Independent features:
  - Map_type, Ink_Orderliness, Mixedness
  - other_Att, *Att*_Value, *Att*_Difference, Missing, Inked_Out
  - Dyad_ID, Speaker_ID, Speaker_Role

Serial Dependency Features

- Link the three decision trees into a chain.

Serial Dependency Features

- Link the three decision trees into a chain.

Serial Dependency Features

- Link the three decision trees into a chain.

Serial Dependency Features

- Link the three decision trees into a chain.

Serial Dependency Features

- Link the three decision trees into a chain.

```
  <type>           <col>          <other>
YES OR NO  →     YES OR NO  →   YES OR NO
```
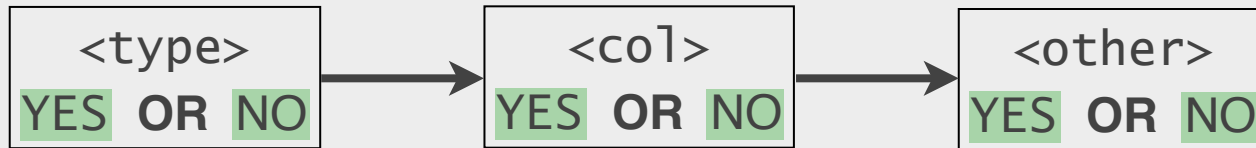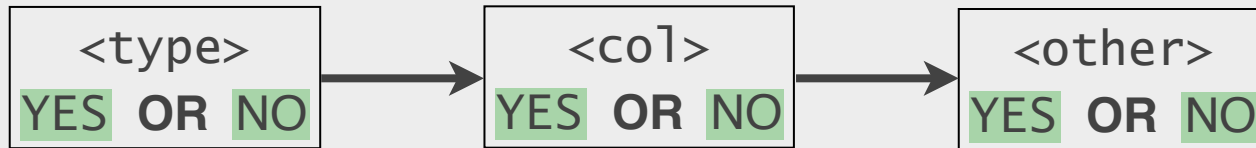
- Three chaining features:

Serial Dependency Features
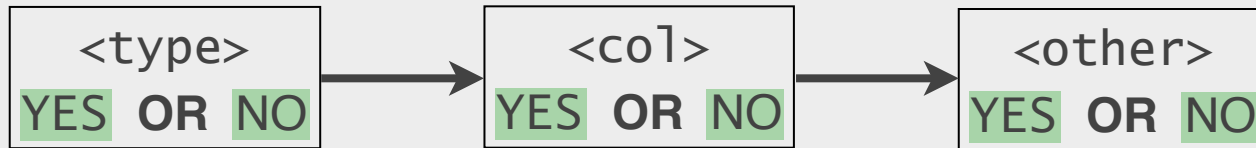
- Link the three decision trees into a chain.

```
  ┌─────────────┐      ┌─────────────┐      ┌─────────────┐
  │   <type>    │─────▶│    <col>    │─────▶│   <other>   │
  │ YES OR NO   │      │ YES OR NO   │      │ YES OR NO   │
  └─────────────┘      └─────────────┘      └─────────────┘
```

- Three chaining features:

  1. **DP_Att**: The discriminatory power of the attribute under consideration.

Serial Dependency Features
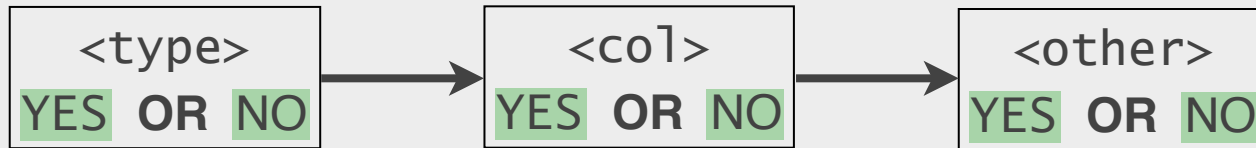
- Link the three decision trees into a chain.

```
┌─────────────┐      ┌─────────────┐      ┌─────────────┐
│   <type>    │      │    <col>    │      │   <other>   │
│ YES OR NO   │─────▶│ YES OR NO   │─────▶│ YES OR NO   │
└─────────────┘      └─────────────┘      └─────────────┘
```

- Three chaining features:

  1. **DP_Att**: The discriminatory power of the attribute under consideration.
  2. **DP_RE**: The discriminatory power of the set of attributes already chosen.

13

Serial Dependency Features

- Link the three decision trees into a chain.

`<type>` YES **OR** NO	→	`<col>` YES **OR** NO	→	`<other>` YES **OR** NO

- Three chaining features:

1. **DP_Att**: The discriminatory power of the attribute under consideration.

2. **DP_RE**: The discriminatory power of the set of attributes already chosen.

3. **Incl_[*Att*]**: The decision made by the decision tree for *Att*, if it was earlier in the chain.
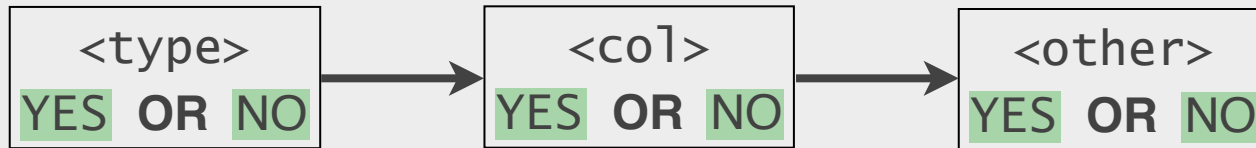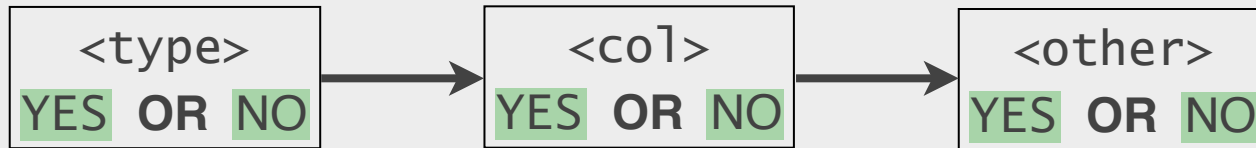
13

Serial Dependency Features

- Link the three decision trees into a chain.

- Three chaining features:

  1. **DP_Att**: The discriminatory power of the attribute under consideration.

  2. **DP_RE**: The discriminatory power of the set of attributes already chosen.

  3. **Incl_[*Att*]**: The decision made by the decision tree for *Att*, if it was earlier in the chain.

- DP_Att and DP_RE are the factors that introduce serial dependency into traditional REG algorithms.

The Models

- **1**: only DP_Att
- **2**: only DP_RE
- **3**: only Incl_[*Att*]
- **1+2**: DP_Att and DP_RE
- **2+3**: DP_RE and Incl_[*Att*]
- **1+2+3**: all chaining features
- **1+2+3+NC**: all chaining and non-chaining features
- **NC**: all non-chaining features

Outline

Accuracy of Replicating the Content Patterns

Model	Initial References	Subsequent References	All References
1	39.9%	43.7%	40.4%
2	42.0%	41.8%	38.5%
3	39.0%	41.4%	37.4%
1+2	42.3%	44.3%	41.5%
2+3	42.0%	41.8%	38.5%
1+2+3	42.9%	44.3%	41.4%
1+2+3+NC	**72.5%**	**66.4%**	**68.6%**
NC	**72.3%**	**66.0%**	**68.2%**

Discriminatory Power vs. Visual Salience

1: DP_Att – proportion of *remaining* distractors ruled out

- reflects the discriminatory power of Att

1-NC: Prop_Same_*Att* – proportional to the number of distractors ruled out *at the start*

- reflects the visual salience of the attribute.

Model	Initial References	Subsequent References	All References
1	39.9%	43.7%	40.4%
1–NC	39.9%	**49.0%**	**46.0%**

Conclusions

Conclusions

- There is no evidence for serial dependency in the reference behaviour of speakers in the iMAP Corpus.

Conclusions

- There is no evidence for serial dependency in the reference behaviour of speakers in the iMAP Corpus.

- Visual salience of an attribute is more influential than discriminatory power in determining whether it will be used in a RE.