Incremental Semantic Interpretation of Dialogue Contributions

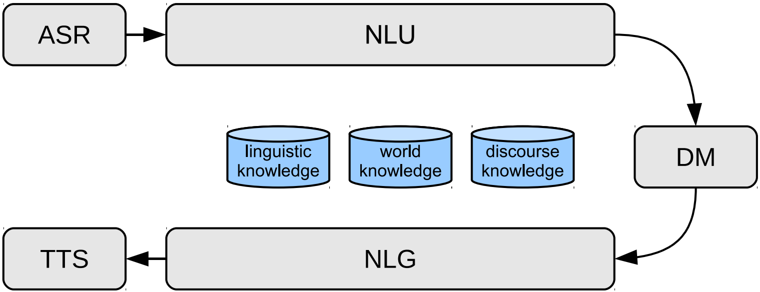
Andreas Peldszus

Institut für Linguistik, Universität Potsdam

Kolloquium Linguistik, Universität Bielefeld, 07.11.2012

Outline

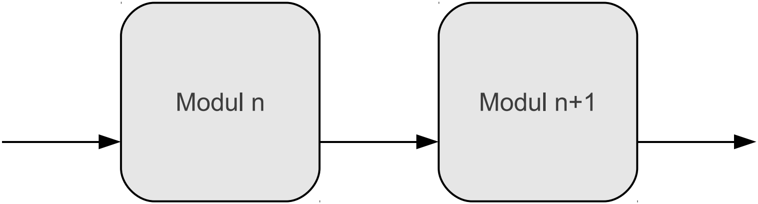
Introduction: Spoken Dialogue Systems



Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

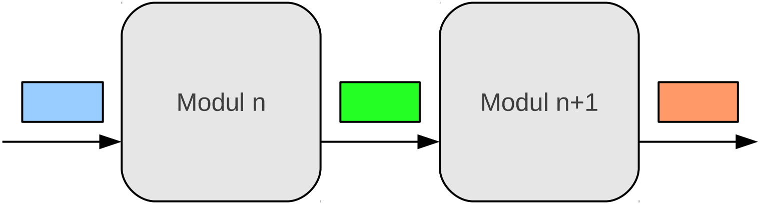


information flow between two modules

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

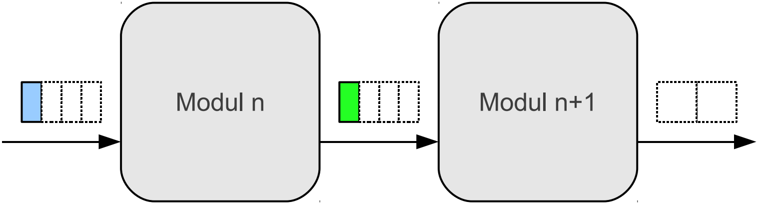


non-incremental information flow

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

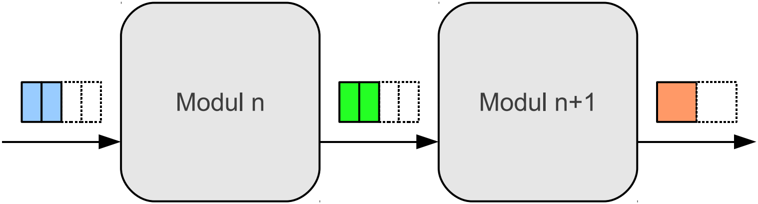


incremental information flow: *early propagation*

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

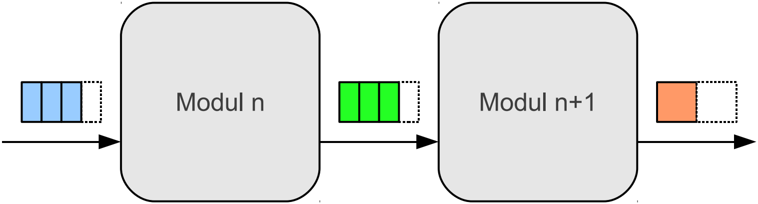


incremental information flow: *early propagation*

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

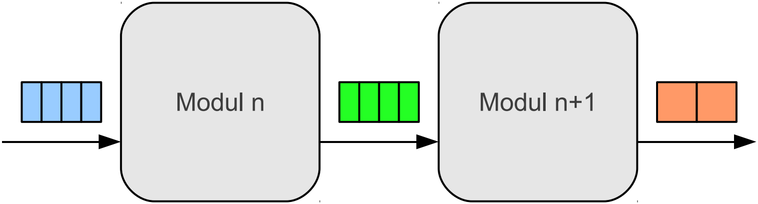


incremental information flow: *early propagation*

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

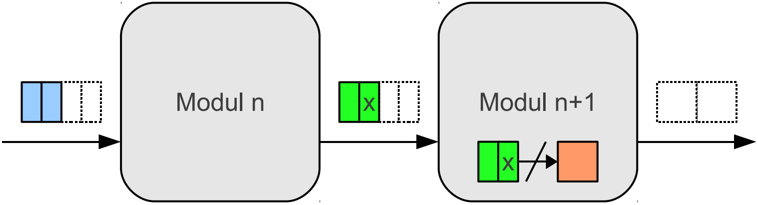


incremental information flow: *early propagation*

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

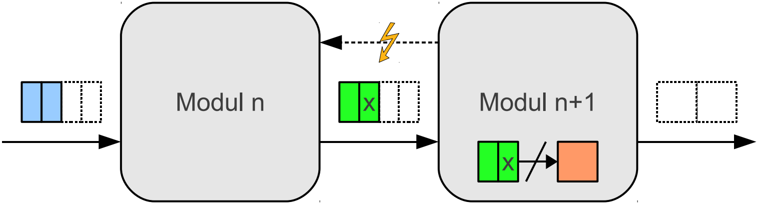


incremental information flow: *interaction / feedback*

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.

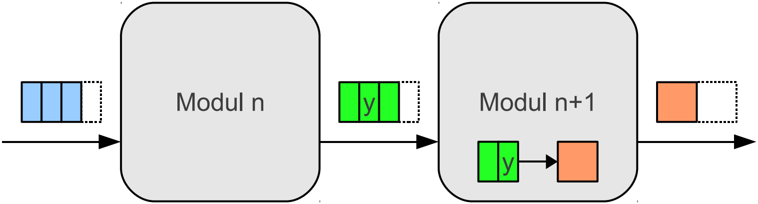


incremental information flow: *interaction / feedback*

Introduction: Incremental Processing

Incrementality, Schlangen and Skantze [2009], citing Levelt [1989]

Each processing component is triggered into activity by a minimal amount of its characteristic input.



incremental information flow: *interaction / feedback*

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
     intervening corrections, clairifications, completions
3. computational benefits

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
   - intervening corrections, clarifications, completions
3. computational benefits

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
   - intervening corrections, clairifications, completions
3. computational benefits

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
   - intervening corrections, clairifications, completions
3. computational benefits
   - speed (by early propagation): why wait with processing?
     but communicative overhead
     accuracy (by interaction): concentrate on useful things
     but risk of gardenpathing
     efficiency (by interaction): reduce workload
     but risk of increased workload with certain utterances

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
   - intervening corrections, clairifications, completions
3. computational benefits
   - speed (by early propagation): why wait with processing? but communicative overhead
   - accuracy (by interaction): concentrate on useful things but risk of gardenpathing
   - efficiency (by interaction): reduce workload but risk of increased workload with certain utterances

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
   - intervening corrections, clairifications, completions
3. computational benefits
   - speed (by early propagation): why wait with processing? but communicative overhead
   - accuracy (by interaction): concentrate on useful things but risk of gardenpathing
   - efficiency (by interaction): reduce workload but risk of increased workload with certain utterances

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
   - intervening corrections, clairifications, completions
3. computational benefits
   - speed (by early propagation): why wait with processing? but communicative overhead
   - accuracy (by interaction): concentrate on useful things but risk of gardenpathing
   - efficiency (by interaction): reduce workload but risk of increased workload with certain utterances

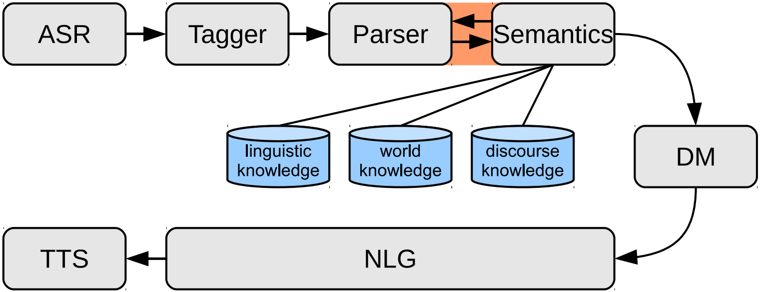
Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
   - reference resolution, early disambiguation etc
2. descriptive adequacy
   - backchannel feedback
   - intervening corrections, clairifications, completions
3. computational benefits
   - speed (by early propagation): why wait with processing?
     but communicative overhead
   - accuracy (by interaction): concentrate on useful things
     but risk of gardenpathing
   - efficiency (by interaction): reduce workload
     but risk of increased workload with certain utterances

Introduction: Motivating Incremental SDS

1. psycho-linguistic plausibility Tanenhaus and Brown-Schmidt [2008]
    - reference resolution, early disambiguation etc
2. descriptive adequacy
    - backchannel feedback
    - intervening corrections, clairifications, completions
3. computational benefits
    - speed (by early propagation): why wait with processing? but communicative overhead
    - accuracy (by interaction): concentrate on useful things but risk of gardenpathing
    - efficiency (by interaction): reduce workload but risk of increased workload with certain utterances

Introduction: Aims of the Approach



Goal: Prune semantically implausible syntactic readings, i.e. readings without referential success.

Related Work

- Purver et al. [2011] - incremental semantic construction within Dynamic Syntax, centered on grammaticality, rather monolithic
- Poesio and Traum [1997], Poesio and Rieser [2010] - incremental semantic construction with λ calculus for TAG, focus on discourse structure, no implementation yet?
- Lison [2008] - incremental semantic construction with CCG, incremental statistical parse selection, no interaction, word- versus phrase-incremental?
- Hassan et al. [2009] - very fast semi-incremental (two-pass) CCG parser, no semantics given, but could be combined with Bos [2005] CCG semantics.
- Stoness et al. [2005] - incremental semantic construction with a PCFG, bottom-up parser, reference feedback, tested on only one dialogue transcript

Related Work

- Purver et al. [2011] - incremental semantic construction within Dynamic Syntax, centered on grammaticality, rather monolithic
- Poesio and Traum [1997], Poesio and Rieser [2010] - incremental semantic construction with λ calculus for TAG, focus on discourse structure, no implementation yet?
- Lison [2008] - incremental semantic construction with CCG, incremental statistical parse selection, no interaction, word- versus phrase-incremental?
- Hassan et al. [2009] - very fast semi-incremental (two-pass) CCG parser, no semantics given, but could be combined with Bos [2005] CCG semantics.
- Stoness et al. [2005] - incremental semantic construction with a PCFG, bottom-up parser, reference feedback, tested on only one dialogue transcript

Related Work

- Purver et al. [2011] - incremental semantic construction within Dynamic Syntax, centered on grammaticality, rather monolithic
- Poesio and Traum [1997], Poesio and Rieser [2010] - incremental semantic construction with λ calculus for TAG, focus on discourse structure, no implementation yet?
- Lison [2008] - incremental semantic construction with CCG, incremental statistical parse selection, no interaction, word- versus phrase-incremental?
- Hassan et al. [2009] - very fast semi-incremental (two-pass) CCG parser, no semantics given, but could be combined with Bos [2005] CCG semantics.
- Stoness et al. [2005] - incremental semantic construction with a PCFG, bottom-up parser, reference feedback, tested on only one dialogue transcript

Related Work

- Purver et al. [2011] - incremental semantic construction within Dynamic Syntax, centered on grammaticality, rather monolithic
- Poesio and Traum [1997], Poesio and Rieser [2010] - incremental semantic construction with λ calculus for TAG, focus on discourse structure, no implementation yet?
- Lison [2008] - incremental semantic construction with CCG, incremental statistical parse selection, no interaction, word- versus phrase-incremental?
- Hassan et al. [2009] - very fast semi-incremental (two-pass) CCG parser, no semantics given, but could be combined with Bos [2005] CCG semantics.
- Stoness et al. [2005] - incremental semantic construction with a PCFG, bottom-up parser, reference feedback, tested on only one dialogue transcript

Related Work

- Purver et al. [2011] - incremental semantic construction within Dynamic Syntax, centered on grammaticality, rather monolithic
- Poesio and Traum [1997], Poesio and Rieser [2010] - incremental semantic construction with λ calculus for TAG, focus on discourse structure, no implementation yet?
- Lison [2008] - incremental semantic construction with CCG, incremental statistical parse selection, no interaction, word- versus phrase-incremental?
- Hassan et al. [2009] - very fast semi-incremental (two-pass) CCG parser, no semantics given, but could be combined with Bos [2005] CCG semantics.
- Stoness et al. [2005] - incremental semantic construction with a PCFG, bottom-up parser, reference feedback, tested on only one dialogue transcript

Theory: Incremental Semantic Construction

Incrementality in Semantic Construction

- Goal: for an input find a semantic representation that is maximally informative and open to combination with further semantic increments
- three factors that influence *a syntax driven* incremental semantic construction
  1. type of syntactic structure building (top-down-parsing, bottom-up-parsing)
  2. type of syntactic structure (left- or right-branching)
  3. type of semantic structure interpretation (bottom-up, top-down)

Incrementality in Semantic Construction

- Goal: for an input find a semantic representation that is maximally informative and open to combination with further semantic increments
- three factors that influence *a syntax driven* incremental semantic construction
  1. type of syntactic structure building (top-down-parsing, bottom-up-parsing)
  2. type of syntactic structure (left- or right-branching)
  3. type of semantic structure interpretation (bottom-up, top-down)

Incrementality in Semantic Construction

- Goal: for an input find a semantic representation that is maximally informative and open to combination with further semantic increments
- three factors that influence *a syntax driven* incremental semantic construction
  1. type of syntactic structure building (top-down-parsing, bottom-up-parsing)
  2. type of syntactic structure (left- or right-branching)
  3. type of semantic structure interpretation (bottom-up, top-down)

Incrementality in Semantic Construction

- Goal: for an input find a semantic representation that is maximally informative and open to combination with further semantic increments
- three factors that influence *a syntax driven* incremental semantic construction
  1. type of syntactic structure building
     (top-down-parsing, bottom-up-parsing)
  2. type of syntactic structure
     (left- or right-branching)
  3. type of semantic structure interpretation
     (bottom-up, top-down)

Incrementality in Semantic Construction

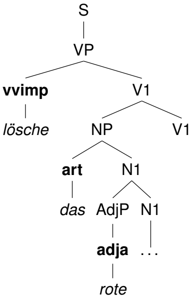
- Goal: for an input find a semantic representation that is maximally informative and open to combination with further semantic increments
- three factors that influence *a syntax driven* incremental semantic construction
  1. type of syntactic structure building (top-down-parsing, bottom-up-parsing)
  2. type of syntactic structure (left- or right-branching)
  3. type of semantic structure interpretation (bottom-up, top-down)

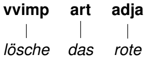
Incrementality in Semantic Construction

- Goal: for an input find a semantic representation that is maximally informative and open to combination with further semantic increments
- three factors that influence *a syntax driven* incremental semantic construction
  1. type of syntactic structure building (top-down-parsing, bottom-up-parsing)
  2. type of syntactic structure (left- or right-branching)
  3. type of semantic structure interpretation (bottom-up, top-down)

Incrementality in Semantic Construction

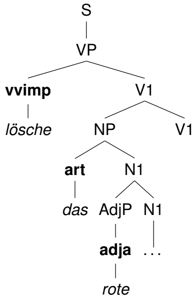
- Roark [2001]: *rooted & connected* structures are essential for incremental interpretation
- usually only provided by top-down-parsers
- drawbacks: many hypotheses, problems with left-recursion
- Roark presents solutions for both problems:

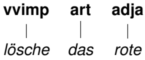




Incrementality in Semantic Construction

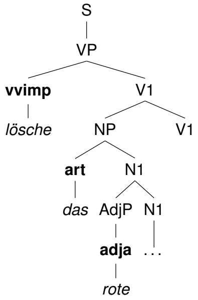
- Roark [2001]: *rooted & connected* structures are essential for incremental interpretation
- usually only provided by top-down-parsers
- drawbacks: many hypotheses, problems with left-recursion
- Roark presents solutions for both problems:

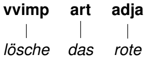




Incrementality in Semantic Construction

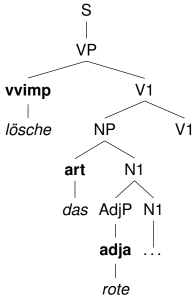
- Roark [2001]: *rooted & connected* structures are essential for incremental interpretation
- usually only provided by top-down-parsers
- drawbacks: many hypotheses, problems with left-recursion
- Roark presents solutions for both problems:
  - grammar transformations to delay hypotheses as long as possible (left-factorisation, left-corner transform)
  - probabilistic beam-search to reduce the search space

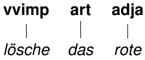




Incrementality in Semantic Construction

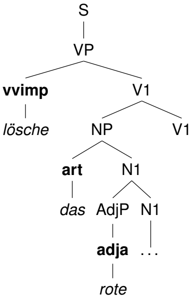
- Roark [2001]: *rooted & connected* structures are essential for incremental interpretation
- usually only provided by top-down-parsers
- drawbacks: many hypotheses, problems with left-recursion
- Roark presents solutions for both problems:
  - grammar transformations to delay hypotheses as long as possible (left-factorisation, left-corner transform)
  - probabilistic beam-search to reduce the search space

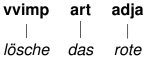




Incrementality in Semantic Construction

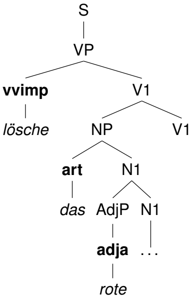
- Roark [2001]: *rooted & connected* structures are essential for incremental interpretation
- usually only provided by top-down-parsers
- drawbacks: many hypotheses, problems with left-recursion
- Roark presents solutions for both problems:
  - grammar transformations to delay hypotheses as long as possible (left-factorisation, left-corner transform)
  - probabilistic beam-search to reduce the search space

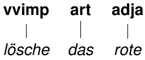




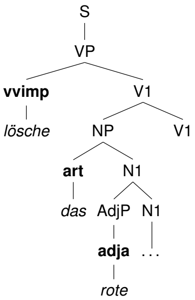
Incrementality in Semantic Construction

- Roark [2001]: *rooted & connected* structures are essential for incremental interpretation
- usually only provided by top-down-parsers
- drawbacks: many hypotheses, problems with left-recursion
- Roark presents solutions for both problems:
    - grammar transformations to delay hypotheses as long as possible (left-factorisation, left-corner transform)
    - probabilistic beam-search to reduce the search space





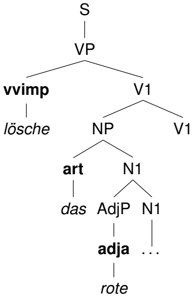
Tree-Interpretation: Bottom up?



Tree-interpretation bottom-up, inside-out:

- need to underspecify open nodes
- need to re-interpret the whole tree every time it expands

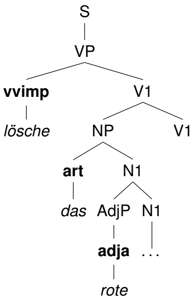
Tree-Interpretation: Bottom up?



Tree-interpretation bottom-up, inside-out:

- need to underspecify open nodes
- need to re-interpret the whole tree every time it expands

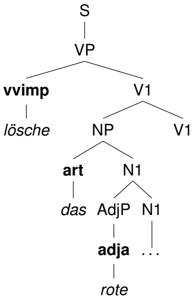
Tree-Interpretation: Bottom up?



Tree-interpretation bottom-up, inside-out:

- need to underspecify open nodes
- need to re-interpret the whole tree every time it expands

Tree-Interpretation: Top-Down!

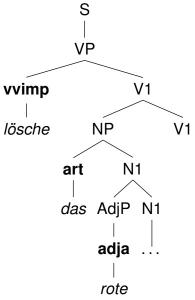


Tree-interpretation top-down, left-to-right:

- linearise semantic combination in parallel to the top-down expansion of the tree
- every expansion of a non-terminal node adds a rule-specific construction semantic increment
  (parser action: predict)
- every recognition of a terminal node adds a lexical semantic increment
  (parser action: match)

- no need to underspecify open nodes
- no need to re-interpret the tree
- *monotonic growth, synchronised with parsing*

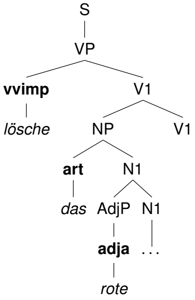
Tree-Interpretation: Top-Down!



Tree-interpretation top-down, left-to-right:

- linearise semantic combination in parallel to the top-down expansion of the tree
- every expansion of a non-terminal node adds a rule-specific construction semantic increment
  (parser action: predict)
- every recognition of a terminal node adds a lexical semantic increment
  (parser action: match)
- no need to underspecify open nodes
- no need to re-interpret the tree
- *monotonic growth, synchronised with parsing*

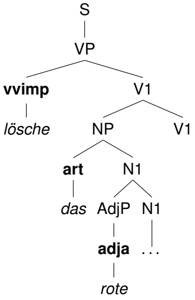
Tree-Interpretation: Top-Down!



Tree-interpretation top-down, left-to-right:

- linearise semantic combination in parallel to the top-down expansion of the tree
- every expansion of a non-terminal node adds a rule-specific construction semantic increment
  (parser action: predict)
- every recognition of a terminal node adds a lexical semantic increment
  (parser action: match)
- no need to underspecify open nodes
- no need to re-interpret the tree
- *monotonic growth, synchronised with parsing*

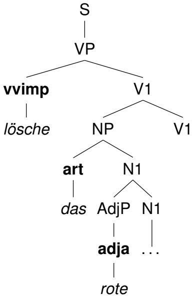
Tree-Interpretation: Top-Down!



Tree-interpretation top-down, left-to-right:

- linearise semantic combination in parallel to the top-down expansion of the tree
- every expansion of a non-terminal node adds a rule-specific construction semantic increment
  (parser action: predict)
- every recognition of a terminal node adds a lexical semantic increment
  (parser action: match)
- no need to underspecify open nodes
- no need to re-interpret the tree
- *monotonic growth, synchronised with parsing*

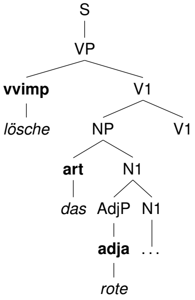
Tree-Interpretation: Top-Down!



Tree-interpretation top-down, left-to-right:

- linearise semantic combination in parallel to the top-down expansion of the tree
- every expansion of a non-terminal node adds a rule-specific construction semantic increment
  (parser action: predict)
- every recognition of a terminal node adds a lexical semantic increment
  (parser action: match)
- no need to underspecify open nodes
- no need to re-interpret the tree
- *monotonic growth, synchronised with parsing*

Tree-Interpretation: Top-Down!

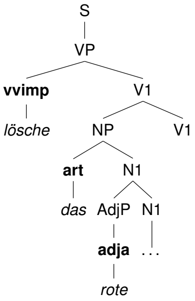


Tree-interpretation top-down, left-to-right:

- linearise semantic combination in parallel to the top-down expansion of the tree
- every expansion of a non-terminal node adds a rule-specific construction semantic increment
  (parser action: predict)
- every recognition of a terminal node adds a lexical semantic increment
  (parser action: match)

- no need to underspecify open nodes
- no need to re-interpret the tree
- *monotonic growth, synchronised with parsing*

Tree-Interpretation: Top-Down!



Tree-interpretation top-down, left-to-right:

- linearise semantic combination in parallel to the top-down expansion of the tree
- every expansion of a non-terminal node adds a rule-specific construction semantic increment
  (parser action: predict)
- every recognition of a terminal node adds a lexical semantic increment
  (parser action: match)

- no need to underspecify open nodes
- no need to re-interpret the tree
- *monotonic growth, synchronised with parsing*

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

Every dog chases a cat.
every(x, dog(x), exist(y, cat(y), chase(x,y)))
exist(y, cat(y), every(x, dog(x), chase(x,y)))

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$
\begin{array}{l}
\langle h_0, \\
\{\ell_1\text{:every}(x, h_1, h_2), \\
\ell_2\text{:dog}(x), \\
\ell_3\text{:chase}(e, x, y), \\
\ell_4\text{:some}(y, h_3, h_4), \\
\ell_5\text{:cat}(y)\ \}, \\
\{h_1 =_q \ell_2, h_3 =_q \ell_5\}\rangle
\end{array}
$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$
\begin{array}{l}
\langle h_0, \\
\{\ell_1\text{:every}(x, h_1, h_2), \\
\ell_2\text{:dog}(x), \\
\ell_3\text{:chase}(e, x, y), \\
\ell_4\text{:some}(y, h_3, h_4), \\
\ell_5\text{:cat}(y)\ \}, \\
\{h_1 =_q \ell_2, h_3 =_q \ell_5\}\rangle
\end{array}
$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$
\begin{array}{l}
\langle h_0, \\
\{\ell_1\text{:every}(x, h_1, h_2), \\
\ell_2\text{:dog}(x), \\
\ell_3\text{:chase}(e, x, y), \\
\ell_4\text{:some}(y, h_3, h_4), \\
\ell_5\text{:cat}(y)\ \}, \\
\{h_1 =_q \ell_2, h_3 =_q \ell_5\}\rangle
\end{array}
$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$
\begin{array}{l}
\langle h_0, \\
\{\ell_1\text{:every}(x, h_1, h_2), \\
\ell_2\text{:dog}(x), \\
\ell_3\text{:chase}(e, x, y), \\
\ell_4\text{:some}(y, h_3, h_4), \\
\ell_5\text{:cat}(y)\ \}, \\
\{h_1 =_q \ell_2, h_3 =_q \ell_5\}\rangle
\end{array}
$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$
\begin{array}{l}
\langle h_0, \\
\{\ell_1\text{:every}(x, h_1, h_2), \\
\ell_2\text{:dog}(x), \\
\ell_3\text{:chase}(e, x, y), \\
\ell_4\text{:some}(y, h_3, h_4), \\
\ell_5\text{:cat}(y)\ \}, \\
\{h_1 =_q \ell_2, h_3 =_q \ell_5\}\rangle
\end{array}
$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$\begin{array}{l}
\langle\ h_0, \\
\{\ \ell_1{:}a_1{:}\mathrm{every}(),\ \mathsf{BV}(a_1, x_1),\ \mathsf{RSTR}(a_1, h_1),\ \mathsf{BODY}(a_1, h_2), \\
\ell_2{:}a_2{:}\mathrm{dog}(x_1), \\
\ell_3{:}a_3{:}\mathrm{chase}(e_1),\ \mathsf{ARG}_1(a_3, x_1),\ \mathsf{ARG}_2(a_3, x_2), \\
\ell_4{:}a_4{:}\mathrm{some}(),\ \mathsf{BV}(a_4, x_2),\ \mathsf{RSTR}(a_4, h_3),\ \mathsf{BODY}(a_4, h_4), \\
\ell_5{:}a_5{:}\mathrm{cat}(x_2)\ \}, \\
\{\ h_1 =_q \ell_2,\ h_3 =_q \ell_5\ \}\ \rangle
\end{array}$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$\begin{array}{l} \langle\ h_0, \\ \{\ \ell_1{:}a_1{:}\mathrm{every}(),\ \mathsf{BV}(a_1,x_1),\ \mathsf{RSTR}(a_1,h_1),\ \mathsf{BODY}(a_1,h_2), \\ \ell_2{:}a_2{:}\mathrm{dog}(x_1), \\ \ell_3{:}a_3{:}\mathrm{chase}(e_1),\ \mathsf{ARG}_1(a_3,x_1),\ \mathsf{ARG}_2(a_3,x_2), \\ \ell_4{:}a_4{:}\mathrm{some}(),\ \mathsf{BV}(a_4,x_2),\ \mathsf{RSTR}(a_4,h_3),\ \mathsf{BODY}(a_4,h_4), \\ \ell_5{:}a_5{:}\mathrm{cat}(x_2)\ \}, \\ \{\ h_1 =_q \ell_2,\ h_3 =_q \ell_5\ \}\ \rangle \end{array}$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$
\begin{array}{l}
\langle\ h_0, \\
\{\ \ell_1{:}a_1{:}\mathrm{every}(),\ \mathsf{BV}(a_1, x_1),\ \mathsf{RSTR}(a_1, h_1),\ \mathsf{BODY}(a_1, h_2), \\
\ell_2{:}a_2{:}\mathrm{dog}(x_1), \\
\ell_3{:}a_3{:}\mathrm{chase}(e_1),\ \mathsf{ARG}_1(a_3, x_1),\ \mathsf{ARG}_2(a_3, x_2), \\
\ell_4{:}a_4{:}\mathrm{some}(),\ \mathsf{BV}(a_4, x_2),\ \mathsf{RSTR}(a_4, h_3),\ \mathsf{BODY}(a_4, h_4), \\
\ell_5{:}a_5{:}\mathrm{cat}(x_2)\ \}, \\
\{\ h_1 =_q \ell_2,\ h_3 =_q \ell_5\ \}\ \rangle
\end{array}
$$

RMRS introduction

- representation language: first order predicate logic with generalized quantifiers, sortal variables
- scope underspecification: break up into elementary predications with labels and holes, standard conjuction, add scope constraints
- underspecification of predicate argument structure: break up further into elementary predications with anchors, characteristic variable and argument-relations

$$
\begin{aligned}
&\langle\ h_0,\\
&\{\ \ell_1{:}a_1{:}\mathrm{every}(),\ \mathsf{BV}(a_1,x_1),\ \mathsf{RSTR}(a_1,h_1),\ \mathsf{BODY}(a_1,h_2),\\
&\ell_2{:}a_2{:}\mathrm{dog}(x_1),\\
&\ell_3{:}a_3{:}\mathrm{chase}(e_1),\ \mathsf{ARG}_1(a_3,x_1),\ \mathsf{ARG}_2(a_3,x_2),\\
&\ell_4{:}a_4{:}\mathrm{some}(),\ \mathsf{BV}(a_4,x_2),\ \mathsf{RSTR}(a_4,h_3),\ \mathsf{BODY}(a_4,h_4),\\
&\ell_5{:}a_5{:}\mathrm{cat}(x_2)\ \},\\
&\{\ h_1 =_q \ell_2,\ h_3 =_q \ell_5\ \}\ \rangle
\end{aligned}
$$

RMRS forward composition

- classic RMRS Copestake [2007] uses a *set of named slot* to represent the open positions, the syntax-semantic interface picks the right slot to fill
- only one slot of each sort open
- however, left recursive rules in incremental processing may require to have multiple slots of the same sort open
- therefore we reinterpret the slots as a stack of unnamed slots, where only the top element can be filled

RMRS forward composition

- classic RMRS Copestake [2007] uses a *set of named slot* to represent the open positions, the syntax-semantic interface picks the right slot to fill
- only one slot of each sort open
- however, left recursive rules in incremental processing may require to have multiple slots of the same sort open
- therefore we reinterpret the slots as a stack of unnamed slots, where only the top element can be filled

RMRS forward composition

- classic RMRS Copestake [2007] uses a *set of named slot* to represent the open positions, the syntax-semantic interface picks the right slot to fill
- only one slot of each sort open
- however, left recursive rules in incremental processing may require to have multiple slots of the same sort open
- therefore we reinterpret the slots as a stack of unnamed slots, where only the top element can be filled

$[\ell_3{:}a_3{:}e_1]\ \{\ [\ell_5{:}a_5{:}x_2]np-arg\ \}$
$\ell_1{:}a_1{:}\text{every}(),\ \mathsf{BV}(a_1,x_1),\ \mathsf{RSTR}(a_1,h_1),\ \mathsf{BODY}(a_1,h_2),\ h_1 =_q \ell_2,$
$\ell_2{:}a_2{:}\text{dog}(x_1),$
$\ell_3{:}a_3{:}\text{chase}(e_1),\ \mathsf{ARG}_1(a_3,x_1),\ \mathsf{ARG}_2(a_3,x_2),$
$\ell_4{:}a_4{:}\text{some}(),\ \mathsf{BV}(a_4,x_2),\ \mathsf{RSTR}(a_4,h_3),\ \mathsf{BODY}(a_4,h_4),\ h_3 =_q \ell_5$

$[\ell_6{:}a_6{:}e_6]\ \{\ \}\ \ell_6{:}a_6{:}\text{cat}(x_6)$

RMRS forward composition

- classic RMRS Copestake [2007] uses a *set of named slot* to represent the open positions, the syntax-semantic interface picks the right slot to fill
- only one slot of each sort open
- however, left recursive rules in incremental processing may require to have multiple slots of the same sort open
- therefore we reinterpret the slots as a stack of unnamed slots, where only the top element can be filled

$[\ell_3{:}a_3{:}e_1]\ \{\ \}$
$\ell_1{:}a_1{:}\text{every}()$, $\mathsf{BV}(a_1, x_1)$, $\mathsf{RSTR}(a_1, h_1)$, $\mathsf{BODY}(a_1, h_2)$, $h_1 =_q \ell_2$,
$\ell_2{:}a_2{:}\text{dog}(x_1)$,
$\ell_3{:}a_3{:}\text{chase}(e_1)$, $\mathsf{ARG}_1(a_3, x_1)$, $\mathsf{ARG}_2(a_3, x_2)$,
$\ell_4{:}a_4{:}\text{some}()$, $\mathsf{BV}(a_4, x_2)$, $\mathsf{RSTR}(a_4, h_3)$, $\mathsf{BODY}(a_4, h_4)$, $h_3 =_q \ell_5$
$\ell_6{:}a_6{:}\text{cat}(x_6)$, $\ell_6 = \ell_5$, $a_6 = a_5$, $x_6 = x_2$

RMRS forward composition

- classic RMRS Copestake [2007] uses a *set of named slot* to represent the open positions, the syntax-semantic interface picks the right slot to fill
- only one slot of each sort open
- however, left recursive rules in incremental processing may require to have multiple slots of the same sort open
- therefore we reinterpret the slots as a stack of unnamed slots, where only the top element can be filled

$[\ell_3{:}a_3{:}e_1]\ \{\ \}$
$\ell_1{:}a_1{:}\text{every}(), \mathsf{BV}(a_1, x_1), \mathsf{RSTR}(a_1, h_1), \mathsf{BODY}(a_1, h_2), h_1 =_q \ell_2,$
$\ell_2{:}a_2{:}\text{dog}(x_1),$
$\ell_3{:}a_3{:}\text{chase}(e_1), \mathsf{ARG}_1(a_3, x_1), \mathsf{ARG}_2(a_3, x_2),$
$\ell_4{:}a_4{:}\text{some}(), \mathsf{BV}(a_4, x_2), \mathsf{RSTR}(a_4, h_3), \mathsf{BODY}(a_4, h_4), h_3 =_q \ell_5$
$\ell_5{:}a_5{:}\text{cat}(x_2)$

RMRS forward composition

- classic RMRS Copestake [2007] uses a *set of named slot* to represent the open positions, the syntax-semantic interface picks the right slot to fill
- only one slot of each sort open
- however, left recursive rules in incremental processing may require to have multiple slots of the same sort open
- therefore we reinterpret the slots as a stack of unnamed slots, where only the top element can be filled

RMRS forward composition

- classic RMRS Copestake [2007] uses a *set of named slot* to represent the open positions, the syntax-semantic interface picks the right slot to fill
- only one slot of each sort open
- however, left recursive rules in incremental processing may require to have multiple slots of the same sort open
- therefore we reinterpret the slots as a stack of unnamed slots, where only the top element can be filled

RMRS forward composition

RMRS structure under construction with a stack of slots

An RMRS structure under construction is a 6-tuple $\langle GT, H, S, R, C, E \rangle$,

- with GT the global top hole h_0,
- with H the hook $[\ell{:}a{:}i]$, consisting of the local top label ℓ, the anchor a and the index i,
- with S the stack of slots of the form $[\ell_n{:}a_n{:}i_n]$,
- with R the bag of EPs and argument relations,
- with C the bag of constraints and
- with E the set of variable equalities.

RMRS forward composition

Forward slot filling combination

Given two RMRSs, one being the functor $rmrs_f = \langle GT_f, H_f, S_f, R_f, C_f, E_f \rangle$ with the top slot $\mathrm{top}(S_f) = [\ell_f{:}a_f{:}i_f]$ and one being the argument $rmrs_a = \langle GT_a, H_a, S_a, R_a, C_a, E_a \rangle$ with its hook $H_a = [\ell_a{:}a_a{:}i_a]$, the slot filling combination $rmrs_f \triangleleft rmrs_a$ yields an RMRS $rmrs = \langle GT, H, S, R, C, E \rangle$, s.t.

- $GT = GT_f = GT_a$
- $H = H_f$
- $S = \text{merge-stacks}(S_a, \mathrm{pop}(S_f))$
- $R = R_f \cup R_a$
- $C = C_f \cup C_a$
- $E = E_f \cup E_a \cup \{\ell_f = \ell_a, a_f = a_a, i_f = i_a\}$

Basic slotfilling combinators

$$
\begin{array}{lcl}
[-] & = & [\ell{:}a{:}u]\ \{\ \}\ . \\
[\circ] & = & [\ell{:}a{:}u]\ \{\ [\ell{:}a{:}u]\ \}\ . \\
[=] & = & [\ell{:}a{:}u]\ \{\ [\ell{:}a{:}u][\ell{:}a{:}u]\ \}\ . \\
[+] & = & [\ell{:}a{:}u]\ \{\ [\ell_1{:}a_1{:}u][\ell{:}a{:}u]\ \}\ . \\
[+\!\!.\,] & = & [\ell{:}a{:}u]\ \{\ [\ell{:}a{:}u][\ell_1{:}a_1{:}u]\ \}\ . \\
[+\ell] & = & [\ell{:}a{:}u]\ \{\ [\ell{:}a_1{:}u][\ell{:}a{:}u]\ \}\ .
\end{array}
$$

A worked example: a toy grammar

S	$\rightarrow$	S **kon** S	$[\mathrm{Conj}]$
S	$\rightarrow$	VP	$[\mathrm{Arg1}] \triangleleft [\mathrm{adr}]$
VP	$\rightarrow$	**vvimp** V1	$[=]$
V1	$\rightarrow$	NP0 V1	$[\mathrm{Arg2}]$
V1	$\rightarrow$	V1 AdvP	$[\dotplus]$
V1	$\rightarrow$	V1 PP	$[\dotplus]$
V1	$\rightarrow$	ϵ	$[-]$
PP	$\rightarrow$	**appr** NP	$[\mathrm{PP}]$
AdvP	$\rightarrow$	**adv**	$[\mathrm{Adv}]$
AdjP	$\rightarrow$	**adja**	$[\mathrm{Adj}]$

NP0	$\rightarrow$	PP NP0	$[+]$
NP0	$\rightarrow$	NP	$[\circ]$
NP	$\rightarrow$	**pper**	$[\circ]$
NP	$\rightarrow$	**art** N2	$[\mathrm{Q}]$
N2	$\rightarrow$	N1	$[\circ]$
N2	$\rightarrow$	N1 NP	$[\dotplus]$
N1	$\rightarrow$	AdjP N1	$[+\ell]$
N1	$\rightarrow$	N1 PP	$[\dotplus]$
N1	$\rightarrow$	N1 AdvP	$[\dotplus]$
N1	$\rightarrow$	**nn**	$[\circ]$

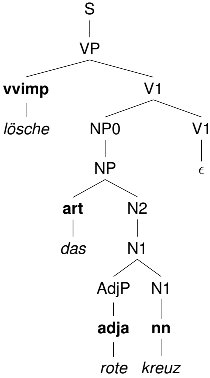
A worked example: semantic macros

$$
\begin{array}{lcll}
[\text{Arg1}] & = & [\ell{:}a{:}u] & \{\,[\ell_1{:}a_1{:}x_1][\ell{:}a{:}u]\,\}\ \mathsf{ARG}_1(a, x_1) \\
[\text{Arg2}] & = & [\ell{:}a{:}u] & \{\,[\ell_1{:}a_1{:}x_1][\ell{:}a{:}u]\,\}\ \mathsf{ARG}_2(a, x_1) \\
[\text{Arg3}] & = & [\ell{:}a{:}u] & \{\,[\ell_1{:}a_1{:}x_1][\ell{:}a{:}u]\,\}\ \mathsf{ARG}_3(a, x_1) \\
[\text{adr}] & = & [\ell{:}a{:}x] & \{\ \}\ \ell{:}a{:}\text{addressee}(x) \\
[\text{Q}] & = & [\ell{:}a{:}x] & \{\,[\ell{:}a{:}e_1][\ell_2{:}a_2{:}x]\,\}\ \mathsf{BV}(a, x), \\
 & & & \mathsf{RSTR}(a, h_1),\ \mathsf{BODY}(a, h_2),\ h_1 =_q \ell_2 \\
[\text{PP}] & = & [\ell{:}a{:}u] & \{\,[\ell_1{:}a_1{:}e_1][\ell_2{:}a_2{:}x_2]\,\}\ \mathsf{ARG}_1(a_1, u), \\
 & & & \mathsf{ARG}_2(a_1, x_2) \\
[\text{Adj}] & = & [\ell{:}a{:}x] & \{\,[\ell{:}a_1{:}e_1]\,\}\ \mathsf{ARG}_1(a_1, x) \\
[\text{Adv}] & = & [\ell{:}a{:}u] & \{\,[\ell_1{:}a_1{:}e_1]\,\}\ \mathsf{ARG}_1(a_1, u) \\
[\text{Conj}] & = & [\ell{:}a{:}u] & \{\,[\ell_1{:}a_1{:}u_1][\ell{:}a{:}u][\ell_3{:}a_3{:}u_3]\,\} \\
 & & & \mathsf{LEFT}_i(a, u_1),\ \mathsf{LEFT}_\ell(a, h_1), \\
 & & & \mathsf{RIGHT}_i(a, u_3),\ \mathsf{RIGHT}_\ell(a, h_3), \\
 & & & h_1 =_q \ell_1,\ h_3 =_q \ell_3
\end{array}
$$

A worked example: generic lexical entries

adja:	$[\ell{:}a{:}e]$	$\{\ \}$	$\ell{:}a{:}\_lemma(e)$
adv:	$[\ell{:}a{:}e]$	$\{\ \}$	$\ell{:}a{:}\_lemma(e)$
appr:	$[\ell{:}a{:}e]$	$\{\ \}$	$\ell{:}a{:}\_lemma(e)$
art:	$[\ell{:}a{:}u]$	$\{\ \}$	$\ell{:}a{:}\_lemma()$
kon:	$[\ell{:}a{:}u]$	$\{\ \}$	$\ell{:}a{:}\_lemma(u)$
nn:	$[\ell{:}a{:}x]$	$\{\ \}$	$\ell{:}a{:}\_lemma(x)$
pper	$[\ell{:}a{:}x]$	$\{\ \}$	$\ell{:}a{:}\mathrm{pper}(x)$
vvimp:	$[\ell{:}a{:}e]$	$\{\ \}$	$\ell{:}a{:}\_lemma(e)$

A worked example: derivation



$[\circ]$
$\lhd$ [Arg1] $\lhd$ [adr] $\lhd$ [=] $\lhd$ $[\![$*lösche*$]\!]$
$\lhd$ [Arg2] $\lhd$ $[\circ]$ $\lhd$ [Q] $\lhd$ $[\![$*das*$]\!]$
$\lhd$ $[\circ]$ $\lhd$ $[+\ell]$ $\lhd$ [Adj] $\lhd$ $[\![$*rote*$]\!]$
$\lhd$ $[\circ]$ $\lhd$ $[\![$*kreuz*$]\!]$
$\lhd$ $[-]$

$[\ell_0{:}a_0{:}e_0]\ \{\ \}$
$\ell_0{:}a_0{:}\_\text{löschen}(e_0)$, $\mathsf{ARG}_1(a_0, x_2)$, $\mathsf{ARG}_2(a_0, x_4)$,
$\ell_2{:}a_2{:}\text{addressee}(x_2)$,
$\ell_4{:}a_4{:}\_\text{def\_q}()$, $\mathsf{BV}(a_4, x_4)$, $\mathsf{RSTR}(a_4, h_1)$, $\mathsf{BODY}(a_4, h_2)$, $h_1 =_q \ell_7$,
$\ell_7{:}a_{10}{:}\_\text{rot}(e_{10})$, $\mathsf{ARG}_1(a_{10}, x_4)$,
$\ell_7{:}a_7{:}\_\text{kreuz}(x_4)$

A worked example: derivation

$$[\circ]$$

$$\begin{array}{c} \text{S} \\ | \\ \ldots \end{array}$$

$$[\ell_0{:}a_0{:}u_0]\ \{\ [\ell_0{:}a_0{:}u_0]\ \}\ .$$

A worked example: derivation

S
|
VP
|
...

$[\circ]$
$\triangleleft$ [Arg1]

$[\ell_0{:}a_0{:}u_0]\ \{\ [\ell_2{:}a_2{:}x_2][\ell_0{:}a_0{:}u_0]\ \}$
$\mathsf{ARG}_1(a_0, x_2)$

A worked example: derivation

S

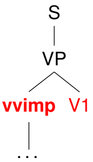
VP

...

$[\circ]$
$\lhd\ [\mathrm{Arg1}] \lhd\ [\mathrm{adr}]$

$[\ell_0{:}a_0{:}u_0]\ \{\ [\ell_0{:}a_0{:}u_0]\ \}$
$\mathsf{ARG}_1(a_0, x_2),\ \ell_2{:}a_2{:}\mathrm{addressee}(x_2)$

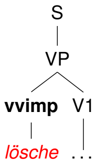
A worked example: derivation



$[\circ]$
$\lhd\ [\mathrm{Arg1}] \lhd\ [\mathrm{adr}] \lhd\ [=]$

$[\ell_0{:}a_0{:}u_0]\ \{\ [\ell_0{:}a_0{:}u_0][\ell_0{:}a_0{:}u_0]\ \}$
$\mathsf{ARG}_1(a_0, x_2),\ \ell_2{:}a_2{:}\mathrm{addressee}(x_2)$

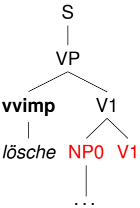
A worked example: derivation



$$[\circ]$$
$$\lhd\ [\mathrm{Arg1}] \lhd\ [\mathrm{adr}] \lhd\ [=] \lhd\ [\![\textit{lösche}]\!]$$

$$[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_0{:}a_0{:}e_0]\ \}$$
$$\ell_0{:}a_0{:}\text{\_löschen}(e_0),\ \mathrm{ARG}_1(a_0, x_2),$$
$$\ell_2{:}a_2{:}\mathrm{addressee}(x_2)$$

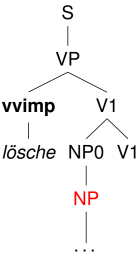
A worked example: derivation



$[\circ]$
$\lhd$ [Arg1] $\lhd$ [adr] $\lhd$ [=] $\lhd$ $[\![$*lösche*$]\!]$
$\lhd$ [Arg2]

$[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_4{:}a_4{:}x_4][\ell_0{:}a_0{:}e_0]\ \}$
$\ell_0{:}a_0{:}\text{\_löschen}(e_0)$, $\text{ARG}_1(a_0, x_2)$, $\text{ARG}_2(a_0, x_4)$,
$\ell_2{:}a_2{:}\text{addressee}(x_2)$

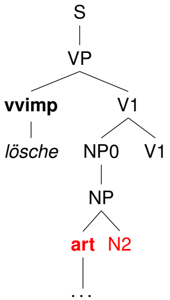
A worked example: derivation



$[\circ]$
$\lhd\ [\text{Arg1}] \lhd\ [\text{adr}] \lhd\ [=] \lhd\ [\![\textit{lösche}]\!]$
$\lhd\ [\text{Arg2}] \lhd\ [\circ]$

$[\ell_0{:}a_0{:}e_0]\ \{\,[\ell_4{:}a_4{:}x_4][\ell_0{:}a_0{:}e_0]\,\}$
$\ell_0{:}a_0{:}\_\text{löschen}(e_0)$, $\mathsf{ARG}_1(a_0, x_2)$, $\mathsf{ARG}_2(a_0, x_4)$,
$\ell_2{:}a_2{:}\text{addressee}(x_2)$

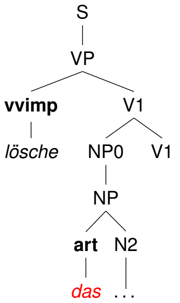
A worked example: derivation



$$
\begin{array}{l}
[\circ] \\
\lhd\ [\mathrm{Arg1}] \lhd\ [\mathrm{adr}] \lhd\ [=] \lhd\ [\![\textit{lösche}]\!] \\
\lhd\ [\mathrm{Arg2}] \lhd\ [\circ] \lhd\ [\mathrm{Q}]
\end{array}
$$

$$
\begin{array}{l}
[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_4{:}a_4{:}e_6][\ell_7{:}a_7{:}x_4][\ell_0{:}a_0{:}e_0]\ \} \\
\ell_0{:}a_0{:}\_\text{löschen}(e_0),\ \mathsf{ARG}_1(a_0, x_2),\ \mathsf{ARG}_2(a_0, x_4), \\
\ell_2{:}a_2{:}\text{addressee}(x_2), \\
\mathsf{BV}(a_4, x_4),\ \mathsf{RSTR}(a_4, h_1),\ \mathsf{BODY}(a_4, h_2),\ h_1 =_q \ell_7
\end{array}
$$

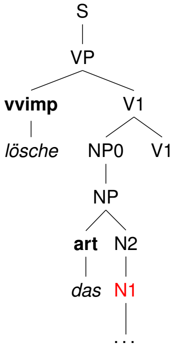
A worked example: derivation



$[\circ]$
$\lhd\ [\mathrm{Arg1}] \lhd\ [\mathrm{adr}] \lhd\ [=] \lhd\ [\![\textit{lösche}]\!]$
$\lhd\ [\mathrm{Arg2}] \lhd\ [\circ] \lhd\ [\mathrm{Q}] \lhd\ [\![\textit{das}]\!]$

$[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_7{:}a_7{:}x_4][\ell_0{:}a_0{:}e_0]\ \}$
$\ell_0{:}a_0{:}\mathrm{\_löschen}(e_0)$, $\mathsf{ARG}_1(a_0, x_2)$, $\mathsf{ARG}_2(a_0, x_4)$,
$\ell_2{:}a_2{:}\mathrm{addressee}(x_2)$,
$\ell_4{:}a_4{:}\mathrm{\_def\_q}()$, $\mathsf{BV}(a_4, x_4)$, $\mathsf{RSTR}(a_4, h_1)$, $\mathsf{BODY}(a_4, h_2)$, $h_1 =_q \ell_7$

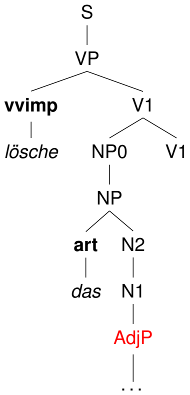
A worked example: derivation



$$[\circ]$$
$$\lhd\ [\text{Arg1}] \lhd\ [\text{adr}] \lhd\ [=] \lhd\ [\![\textit{lösche}]\!]$$
$$\lhd\ [\text{Arg2}] \lhd\ [\circ] \lhd\ [\text{Q}] \lhd\ [\![\textit{das}]\!]$$
$$\lhd\ [\circ]$$

$$[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_7{:}a_7{:}x_4][\ell_0{:}a_0{:}e_0]\ \}$$
$$\ell_0{:}a_0{:}\_\text{löschen}(e_0),\ \mathsf{ARG}_1(a_0, x_2),\ \mathsf{ARG}_2(a_0, x_4),$$
$$\ell_2{:}a_2{:}\text{addressee}(x_2),$$
$$\ell_4{:}a_4{:}\_\text{def\_q}(),\ \mathsf{BV}(a_4, x_4),\ \mathsf{RSTR}(a_4, h_1),\ \mathsf{BODY}(a_4, h_2),\ h_1 =_q \ell_7$$

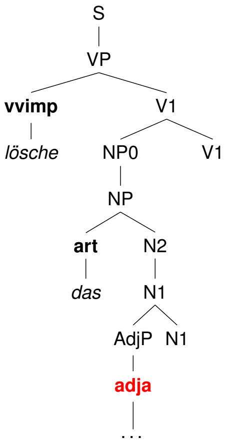
A worked example: derivation



$$[\circ] \\ \lhd\ [\mathrm{Arg1}] \lhd\ [\mathrm{adr}] \lhd\ [=] \lhd\ [\![\textit{lösche}]\!] \\ \lhd\ [\mathrm{Arg2}] \lhd\ [\circ] \lhd\ [\mathrm{Q}] \lhd\ [\![\textit{das}]\!] \\ \lhd\ [\circ] \lhd\ [+\ell]$$

$$[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_7{:}a_8{:}x_4][\ell_7{:}a_7{:}x_4][\ell_0{:}a_0{:}e_0]\ \}$$
$$\ell_0{:}a_0{:}\text{\_löschen}(e_0),\ \mathsf{ARG}_1(a_0, x_2),\ \mathsf{ARG}_2(a_0, x_4),$$
$$\ell_2{:}a_2{:}\text{addressee}(x_2),$$
$$\ell_4{:}a_4{:}\text{\_def\_q}(),\ \mathsf{BV}(a_4, x_4),\ \mathsf{RSTR}(a_4, h_1),\ \mathsf{BODY}(a_4, h_2),\ h_1 =_q \ell_7$$

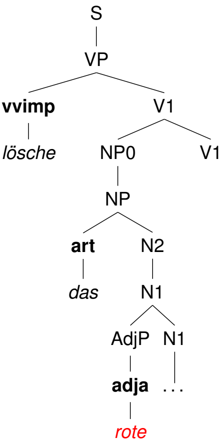
A worked example: derivation



$[\circ]$
$\lhd$ $[\text{Arg1}] \lhd$ $[\text{adr}] \lhd$ $[=] \lhd$ $[\![\textit{lösche}]\!]$
$\lhd$ $[\text{Arg2}] \lhd$ $[\circ] \lhd$ $[\text{Q}] \lhd$ $[\![\textit{das}]\!]$
$\lhd$ $[\circ] \lhd$ $[+\ell] \lhd$ $[\text{Adj}]$

$[\ell_0{:}a_0{:}e_0]$ $\{$ $[\ell_7{:}a_{10}{:}e_{10}][\ell_7{:}a_7{:}x_4][\ell_0{:}a_0{:}e_0]$ $\}$
$\ell_0{:}a_0{:}\text{\_löschen}(e_0)$, $\text{ARG}_1(a_0, x_2)$, $\text{ARG}_2(a_0, x_4)$,
$\ell_2{:}a_2{:}\text{addressee}(x_2)$,
$\ell_4{:}a_4{:}\text{\_def\_q}()$, $\text{BV}(a_4, x_4)$, $\text{RSTR}(a_4, h_1)$, $\text{BODY}(a_4, h_2)$, $h_1 =_q \ell_7$,
$\text{ARG}_1(a_{10}, x_4)$

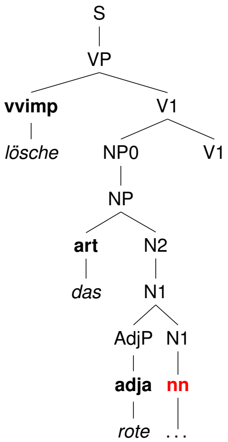
A worked example: derivation



$$
\begin{array}{l}
[\circ] \\
\lhd\ [\mathrm{Arg1}] \lhd\ [\mathrm{adr}] \lhd\ [=] \lhd\ [\![\textit{lösche}]\!] \\
\lhd\ [\mathrm{Arg2}] \lhd\ [\circ] \lhd\ [\mathrm{Q}] \lhd\ [\![\textit{das}]\!] \\
\lhd\ [\circ] \lhd\ [+\ell] \lhd\ [\mathrm{Adj}] \lhd\ [\![\textit{rote}]\!]
\end{array}
$$

$$
\begin{array}{l}
[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_7{:}a_7{:}x_4][\ell_0{:}a_0{:}e_0]\ \} \\
\ell_0{:}a_0{:}\_\mathrm{löschen}(e_0),\ \mathsf{ARG}_1(a_0, x_2),\ \mathsf{ARG}_2(a_0, x_4), \\
\ell_2{:}a_2{:}\mathrm{addressee}(x_2), \\
\ell_4{:}a_4{:}\_\mathrm{def\_q}(),\ \mathsf{BV}(a_4, x_4),\ \mathsf{RSTR}(a_4, h_1),\ \mathsf{BODY}(a_4, h_2),\ h_1 =_q \ell_7, \\
\ell_7{:}a_{10}{:}\_\mathrm{rot}(e_{10}),\ \mathsf{ARG}_1(a_{10}, x_4)
\end{array}
$$

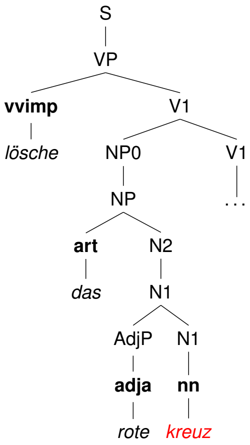
A worked example: derivation



$[\circ]$
$\triangleleft$ $[\text{Arg1}] \triangleleft$ $[\text{adr}] \triangleleft$ $[=] \triangleleft$ $[\![\textit{lösche}]\!]$
$\triangleleft$ $[\text{Arg2}] \triangleleft$ $[\circ] \triangleleft$ $[\text{Q}] \triangleleft$ $[\![\textit{das}]\!]$
$\triangleleft$ $[\circ] \triangleleft$ $[+\ell] \triangleleft$ $[\text{Adj}] \triangleleft$ $[\![\textit{rote}]\!]$
$\triangleleft$ $[\circ]$

$[\ell_0{:}a_0{:}e_0]\ \{\ [\ell_7{:}a_7{:}x_4][\ell_0{:}a_0{:}e_0]\ \}$
$\ell_0{:}a_0{:}\text{\_löschen}(e_0)$, $\text{ARG}_1(a_0, x_2)$, $\text{ARG}_2(a_0, x_4)$,
$\ell_2{:}a_2{:}\text{addressee}(x_2)$,
$\ell_4{:}a_4{:}\text{\_def\_q}()$, $\text{BV}(a_4, x_4)$, $\text{RSTR}(a_4, h_1)$, $\text{BODY}(a_4, h_2)$, $h_1 =_q \ell_7$,
$\ell_7{:}a_{10}{:}\text{\_rot}(e_{10})$, $\text{ARG}_1(a_{10}, x_4)$

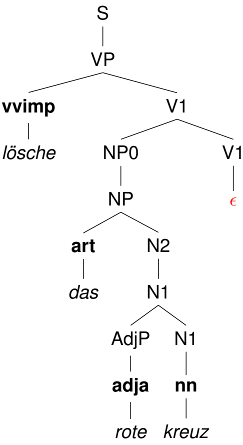
A worked example: derivation



$[\circ]$
$\lhd$ [Arg1] $\lhd$ [adr] $\lhd$ [=] $\lhd$ $[\![\textit{lösche}]\!]$
$\lhd$ [Arg2] $\lhd$ $[\circ]$ $\lhd$ [Q] $\lhd$ $[\![\textit{das}]\!]$
$\lhd$ $[\circ]$ $\lhd$ $[+\ell]$ $\lhd$ [Adj] $\lhd$ $[\![\textit{rote}]\!]$
$\lhd$ $[\circ]$ $\lhd$ $[\![\textit{kreuz}]\!]$

$[\ell_0{:}a_0{:}e_0]$ $\{ [\ell_0{:}a_0{:}e_0] \}$
$\ell_0{:}a_0{:}\_\text{löschen}(e_0)$, $\text{ARG}_1(a_0, x_2)$, $\text{ARG}_2(a_0, x_4)$,
$\ell_2{:}a_2{:}\text{addressee}(x_2)$,
$\ell_4{:}a_4{:}\_\text{def\_q}()$, $\text{BV}(a_4, x_4)$, $\text{RSTR}(a_4, h_1)$, $\text{BODY}(a_4, h_2)$, $h_1 =_q \ell_7$,
$\ell_7{:}a_{10}{:}\_\text{rot}(e_{10})$, $\text{ARG}_1(a_{10}, x_4)$,
$\ell_7{:}a_7{:}\_\text{kreuz}(x_4)$

A worked example: derivation



$$
\begin{array}{l}
[\circ] \\
\triangleleft\ [\mathrm{Arg1}] \triangleleft\ [\mathrm{adr}] \triangleleft\ [=] \triangleleft\ [\![\textit{lösche}]\!] \\
\triangleleft\ [\mathrm{Arg2}] \triangleleft\ [\circ] \triangleleft\ [\mathrm{Q}] \triangleleft\ [\![\textit{das}]\!] \\
\triangleleft\ [\circ] \triangleleft\ [+\ell] \triangleleft\ [\mathrm{Adj}] \triangleleft\ [\![\textit{rote}]\!] \\
\triangleleft\ [\circ] \triangleleft\ [\![\textit{kreuz}]\!] \\
\triangleleft\ [-]
\end{array}
$$

$$
\begin{array}{l}
[\ell_0{:}a_0{:}e_0]\ \{\ \} \\
\ell_0{:}a_0{:}\_\text{löschen}(e_0),\ \mathsf{ARG}_1(a_0, x_2),\ \mathsf{ARG}_2(a_0, x_4), \\
\ell_2{:}a_2{:}\text{addressee}(x_2), \\
\ell_4{:}a_4{:}\_\text{def\_q}(),\ \mathsf{BV}(a_4, x_4),\ \mathsf{RSTR}(a_4, h_1),\ \mathsf{BODY}(a_4, h_2),\ h_1 =_q \ell_7, \\
\ell_7{:}a_{10}{:}\_\text{rot}(e_{10}),\ \mathsf{ARG}_1(a_{10}, x_4), \\
\ell_7{:}a_7{:}\_\text{kreuz}(x_4)
\end{array}
$$

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars
   - delay attachment decisions

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars
   - delay attachment decisions
   - transform semantic rules accordingly (in theory perhaps, automatically no)
   - augment semantic construction with backward function composition in nested evaluation contexts (done)

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars
   - delay attachment decisions
   - transform semantic rules accordingly (in theory perhaps, automatically no)
   - augment semantic construction with backward function composition in nested evaluation contexts (done)

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars
   - delay attachment decisions
   - transform semantic rules accordingly (in theory perhaps, automatically no)
   - augment semantic construction with backward function composition in nested evaluation contexts (done)

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars
   - delay attachment decisions
   - transform semantic rules accordingly (in theory perhaps, automatically no)
   - augment semantic construction with backward function composition in nested evaluation contexts (done)

Further work

Grammar transformation can increase the parsing performance:

1. left-factored grammars
   - delay rule identification for rules with equal first childs
   - transform semantic rules accordingly (in theory yes, automatically no)
   - no further refinements to the semantic construction needed
2. left-corner transformed grammars
   - delay attachment decisions
   - transform semantic rules accordingly (in theory perhaps, automatically no)
   - augment semantic construction with backward function composition in nested evaluation contexts (done)

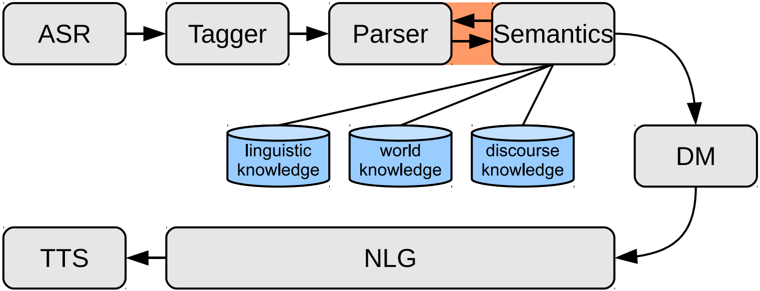
Recapitulation

- incrementality: early propagation
- incrementality: interaction / feedback
- top-down left-to-right tree interpretation
- monotonous, incremental construction

Application: Incremental Reference Feedback

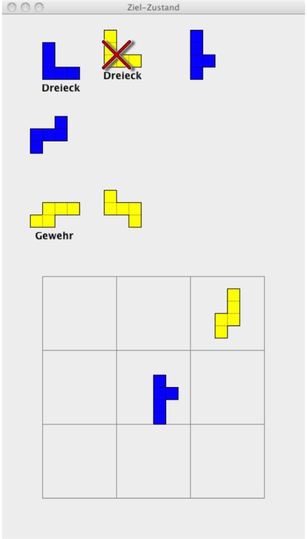
Joint work with Timo Baumann, Okko Buß, and David Schlangen

Idea



Incremental top-down parser, RMRS construction and reference resolution have been implemented in the InproTK Schlangen et al. [2010] for the Pentamino domain Fernández and Schlangen [2007].

Setting: Domain



Pentamino puzzle domain:

- Task oriented dialogue, instructor gives commands to a constructor in order to realise a desired state of the world
- Selecting, moving, deleting, rotating and mirroring of the puzzle pieces

Setting: Domain

A corpus collected in a WOz experiment:

- 20 participants, 284 games
- about 3000 utterances (each with dumped world state) of which. . .
- about 1600 have semantic annotation from the wizards next action, of which. . .
- about 1000 are free of pronouns.
- small command language, yet free and spontanous speech

Setting: Domain

A corpus collected in a WOz experiment:

- 20 participants, 284 games
- about 3000 utterances (each with dumped world state) of which. . .
- about 1600 have semantic annotation from the wizards next action, of which. . .
- about 1000 are free of pronouns.
- small command language, yet free and spontanous speech

Beispiele

in der ersten reihe das rote teil löschen
äh und jetzt nimm das kreuz .. das grüne in der linken spalte
und leg es nach ganz unten
nimm [das [blaue teil] [links] [oben] [neben der treppe]]
und im uhrzeigersinn drehen

Setting: Domain

A corpus collected in a WOz experiment:

- 20 participants, 284 games
- about 3000 utterances (each with dumped world state) of which...
- about 1600 have semantic annotation from the wizards next action, of which...
- about 1000 are free of pronouns.
- small command language, yet free and spontanous speech

Beispiele

in der ersten reihe das rote teil löschen
äh und jetzt nimm das kreuz .. das grüne in der linken spalte
und leg es nach ganz unten
nimm [das [blaue teil] [links] [oben] [neben der treppe]]
und im uhrzeigersinn drehen

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
3. Grammar
4. Parser
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
4. Parser
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
4. Parser
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, ~35 rules
4. Parser
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
     incremental top-down beam-search, without conditioning features
     extended with robust parsing operations (deletions, insertions, tag repair)
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution
   - derives variable assignments for all nominal predicate structures, calculates the set of domain objects
     definite NPs recieve a resolution value, depending on whether they resolve to a singleton set, a set with more elements or an empty set

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution
   - derives variable assignments for all nominal predicate structures
   - calculates the set of domain objects
   - definite NPs recieve a resolution value, depending on whether they resolve to a singleton set, a set with more elements or an empty set

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution
   - derives variable assignments for all nominal predicate structures
   - calculates the set of domain objects
   - definite NPs recieve a resolution value, depending on whether they resolve to a singleton set, a set with more elements or an empty set

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution
   - derives variable assignments for all nominal predicate structures
   - calculates the set of domain objects
   - definite NPs recieve a resolution value, depending on whether they resolve to a singleton set, a set with more elements or an empty set

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution
   - derives variable assignments for all nominal predicate structures
   - calculates the set of domain objects
   - definite NPs recieve a resolution value, depending on whether they resolve to a singleton set, a set with more elements or an empty set

Setting: Modules

1. ASR
   - leave-one-out language model training
   - run transcript versus offline recognition
2. Tagger
   - simple lookup tagger with handmade dictionary
3. Grammar
   - small handmade core-grammar, weights set by hand, 35 rules
4. Parser
   - simplified version of Roark [2001]
   - incremental top-down beam-search, without conditioning functions
   - extended with robust parsing operations (deletions, insertions, tag-repair)
5. Reference Resolution
   - derives variable assignments for all nominal predicate structures
   - calculates the set of domain objects
   - definite NPs recieve a resolution value, depending on whether they resolve to a singleton set, a set with more elements or an empty set

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to. . .

1. Just Syntax (JS)
   - single-best derivation of only syntax
2. External Filtering (EF)
   - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
4. Combined Interaction and Filtering (CIF)

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to. . .

1. Just Syntax (JS)
   - single-best derivation of only syntax
2. External Filtering (EF)
   - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
4. Combined Interaction and Filtering (CIF)

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to...

1. Just Syntax (JS)
   - single-best derivation of only syntax
2. External Filtering (EF)
   - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
   - single-best derivation with reference feedback
4. Combined Interaction and Filtering (CIF)

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to. . .

1. Just Syntax (JS)
   - single-best derivation of only syntax
2. External Filtering (EF)
   - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
   - single-best derivation with reference feedback
4. Combined Interaction and Filtering (CIF)

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to…

1. Just Syntax (JS)
    - single-best derivation of only syntax
2. External Filtering (EF)
    - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
    - single-best derivation with reference feedback
4. Combined Interaction and Filtering (CIF)
    - best of 5-best derivations with reference feedback

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to. . .

1. Just Syntax (JS)
   - single-best derivation of only syntax
2. External Filtering (EF)
   - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
   - single-best derivation with reference feedback
4. Combined Interaction and Filtering (CIF)
   - best of 5-best derivations with reference feedback

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to…

1. Just Syntax (JS)
   - single-best derivation of only syntax
2. External Filtering (EF)
   - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
   - single-best derivation with reference feedback
4. Combined Interaction and Filtering (CIF)
   - best of 5-best derivations with reference feedback

Baseline / Variants

Compare the gold semantic annotation with the resolving world object according to...

1. Just Syntax (JS)
   - single-best derivation of only syntax
2. External Filtering (EF)
   - the 5-best derivations are filtered externally for the best reference resolution value
3. Syntax Pragmatic Interaction (SPI)
   - single-best derivation with reference feedback
4. Combined Interaction and Filtering (CIF)
   - best of 5-best derivations with reference feedback

Results: Accuracy

		JS	EF	SPI	CIF
transcript	-1	563	518	364	363
	0	197	198	267	268
	1	264	308	392	392
	str.acc.	25.7%	30.0%	38.2%	38.2%
	rel.acc.	44.9%	49.3%	64.2%	64.3%
	incr.scr	-1567.66	-1248.26	-535.862	-503.75
	avg.incr.scr	-1.52	-1.22	-0.52	-0.49
recognition	-1	362	348	254	255
	0	122	121	173	173
	1	143	158	196	195
	str.acc.	22.6%	25.0%	31.0%	30.8%
	rel.acc.	41.2%	44.1%	58.3%	58.1%
	incr.scr	-1905.76	-1729.77	-1105.43	-1075.62
	avg.incr.scr	-1.86	-1.69	-1.01	-1.05

Results: Efficiency

	readings	expansions	degrades	pruned deriv.	survived deriv.
no interaction	2186	247052	0	0	69510
interaction	1934	210137	140626	7310	60654
	88%	85%			87%

(for the transcript only, with robust parser operations activated)

Discussion

- grammar does not cover all that is wanted
- yet, there'll always be strange constructions in spontanous speech
- semantic gold featured a discourse-history oracle: 30% of the corpus utterances had a semantic annotation without including a proper NP ("und nochmal drehen bitte"). -30% strict -10% relaxed max accuracy
- first experiments, only hand-set parameters

Discussion

- grammar does not cover all that is wanted
- yet, there'll always be strange constructions in spontanous speech
- semantic gold featured a discourse-history oracle: 30% of the corpus utterances had a semantic annotation without including a proper NP ("und nochmal drehen bitte"). -30% strict -10% relaxed max accuracy
- first experiments, only hand-set parameters

Discussion

- grammar does not cover all that is wanted
- yet, there'll always be strange constructions in spontanous speech
- semantic gold featured a discourse-history oracle: 30% of the corpus utterances had a semantic annotation without including a proper NP ("und nochmal drehen bitte"). -30% strict -10% relaxed max accuracy
- first experiments, only hand-set parameters

Discussion

- grammar does not cover all that is wanted
- yet, there'll always be strange constructions in spontanous speech
- semantic gold featured a discourse-history oracle: 30% of the corpus utterances had a semantic annotation without including a proper NP ("und nochmal drehen bitte"). -30% strict -10% relaxed max accuracy
- first experiments, only hand-set parameters

Recapitulation

- incrementality: early propagation
- incrementality: interaction / feedback
- top-down left-to-right tree interpretation
- monotonous, incremental construction
- feedback improves NLUs accuracy and efficiency

Future Work

- write better grammar, train grammar
- probabilistic reference resolution
- optimise parameters
- automatic semantic rule aquisition
- VP advisoring
- experiment in a larger domain, more complex language

Future Work

- write better grammar, train grammar
- probabilistic reference resolution
- optimise parameters
- automatic semantic rule aquisition
- VP advisoring
- experiment in a larger domain, more complex language

Future Work

- write better grammar, train grammar
- probabilistic reference resolution
- optimise parameters
- automatic semantic rule aquisition
- VP advisoring
- experiment in a larger domain, more complex language

Future Work

- write better grammar, train grammar
- probabilistic reference resolution
- optimise parameters
- automatic semantic rule aquisition
- VP advisoring
- experiment in a larger domain, more complex language

Future Work

- write better grammar, train grammar
- probabilistic reference resolution
- optimise parameters
- automatic semantic rule aquisition
- VP advisoring
- experiment in a larger domain, more complex language

Future Work

- write better grammar, train grammar
- probabilistic reference resolution
- optimise parameters
- automatic semantic rule aquisition
- VP advisoring
- experiment in a larger domain, more complex language

Thank You!

Literatur I

Johan Bos. Towards wide-coverage semantic interpretation. In *In Proceedings of Sixth International Workshop on Computational Semantics IWCS-6*, pages 42–53, 2005.

Ann Copestake. Semantic composition with (robust) minimal recursion semantics. In *Proceedings of the Workshop on Deep Linguistic Processing*, DeepLP '07, pages 73–80, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://portal.acm.org/citation.cfm?id=1608912.1608925.

Raquel Fernández and David Schlangen. Referring under restricted interactivity conditions. In Simon Keizer, Harry Bunt, and Tim Paek, editors, *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, pages 136–139, Antwerp, Belgium, September 2007.

Hany Hassan, Khalil Sima'an, and Andy Way. Lexicalized semi-incremental dependency parsing. In *Proceedings of the International Conference RANLP-2009*, pages 128–134, Borovets, Bulgaria, September 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/R09-1025.

Willem J. M. Levelt. *Speaking*. MIT Press, Cambridge, USA, 1989.

Pierre Lison. Robust processing of situated spoken dialogue. Master's thesis, Universität des Saarlandes, Saarbrücken, December 2008. URL http://www.dfki.de/~plison/pubs/thesis/main.thesis.plison2008.pdf.

Massimo Poesio and Hannes Rieser. Completions, coordination, and alignment in dialogue. *Dialogue & Discourse*, 1(1):1–89, 2010.

Massimo Poesio and David R. Traum. Conversational actions and discourse situations. *Computational Intelligence*, 13(3):309–347, 1997. doi: http://dx.doi.org/10.1111/0824-7935.00042.

Matthew Purver, Arash Eshghi, and Julian Hough. Incremental semantic construction in a dialogue system. In J. Bos and S. Pulman, editors, *Proceedings of the 9th International Conference on Computational Semantics (IWCS)*, pages 365–369. Oxford, UK, January 2011.

Brian Edward Roark. *Robust probabilistic predictive syntactic processing: motivations, models, and applications*. PhD thesis, Department of Cognitive and Linguistic Sciences, Brown University, Providence, RI, USA, 2001. AAI3006783.

D. Schlangen, T. Baumann, H. Buschmeier, O. Buss, S. Kopp, G. Skantze, and R. Yaghoubzadeh. Middleware for incremental processing in conversational agents. In *Proceedings of SigDial*, Tokyo, Japan, sep 2010. URL http://www.speech.kth.se/prod/publications/files/3448.pdf.

David Schlangen and Gabriel Skantze. A general, abstract model of incremental dialogue processing. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718. Association for Computational Linguistics, mar 2009.

Scott C. Stoness, James Allen, Greg Aist, and Mary Swift. Using real-world reference to improve spoken language understanding. In *AAAI Workshop on Spoken Language Understanding*, pages 38–45, 2005.

Michael K. Tanenhaus and Sarah Brown-Schmidt. Language processing in the natural world. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1493):1105–1122, March 2008. doi: 10.1098/rstb.2007.2162.