Positive Effects of Redundant Descriptions in an Interactive Semantic Speech Interface

Lane Schwartz, Luan Nguyen, Andrew Exley, William Schuler
University of Minnesota

June 24, 2011

Motivation

NSF project: build working interactive model of speech/language processing

(parsing, word recognition depend on semantics, pragmatics in context)

Motivation

NSF project: build working interactive model of speech/language processing

(parsing, word recognition depend on semantics, pragmatics in context)

- **Cognitive appeal:** Tanenhaus et al. '95 eye-tracking evidence
  People interactively constrain search through interpretation in context

Motivation

NSF project: build working interactive model of speech/language processing

(parsing, word recognition depend on semantics, pragmatics in context)

- **Cognitive appeal:** Tanenhaus et al. '95 eye-tracking evidence
  People interactively constrain search through interpretation in context
- **Practical appeal:** context-dependent speech interfaces

  To artificial agent in 'content creation' domain:
  1. 'Add new folder ***coling***' (fix pronunciation?)
  2. 'Go to the ***coling*** folder and add new item ***semrec***' (fix pronunciation?)
  3. ...
  4. 'Select the ***semrec*** in the ***coling*** folder' (recognition should be reliable)

  Interface uses context to improve recognition, in lieu of training corpus

Motivation

NSF project: build working interactive model of speech/language processing

(parsing, word recognition depend on semantics, pragmatics in context)

- **Cognitive appeal:** Tanenhaus et al. '95 eye-tracking evidence
  People interactively constrain search through interpretation in context
- **Practical appeal:** context-dependent speech interfaces

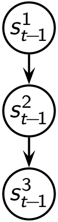
  To artificial agent in 'content creation' domain:
  1. 'Add new folder ***coling***' (fix pronunciation?)
  2. 'Go to the ***coling*** folder and add new item ***semrec***' (fix pronunciation?)
  3. ...
  4. 'Select the ***semrec*** in the ***coling*** folder' (recognition should be reliable)

  Interface uses context to improve recognition, in lieu of training corpus

  This talk: extended model allows redundancy to improve accuracy
  (only one semrec, but similar to sentry/timrec/... so add 'in coling')

Probabilistic Time-Series Model

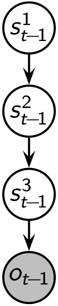
Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)



Elements hold hypoth. stacked-up **incomplete constituents**, dep. on parent (incomplete constituent: e.g. S/VP = sentence lacking verb phrase to come)

Probabilistic Time-Series Model

Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)

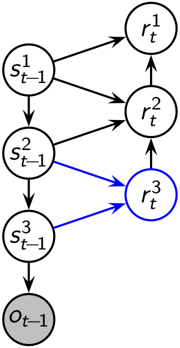


Elements hold hypoth. stacked-up **incomplete constituents**, dep. on parent (incomplete constituent: e.g. S/VP = sentence lacking verb phrase to come)

Hypothesized mem elements generate **observations**: words / acoust. features

Probabilistic Time-Series Model

Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)

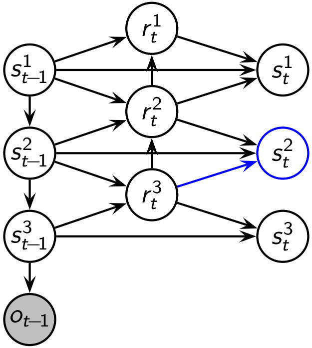


Elements in memory store may be composed (reduced) w. element above

Probability depends on antecedent vars (e.g. Det, Noun reduce to NP)

Probabilistic Time-Series Model

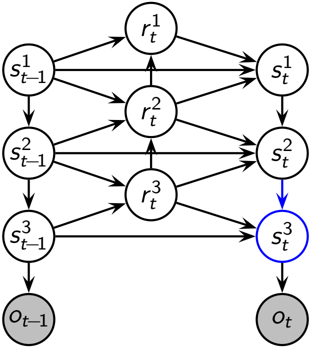
Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)



Non-reduced elements carry forward or transition (e.g. NP becomes S/VP)

Probabilistic Time-Series Model

Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)

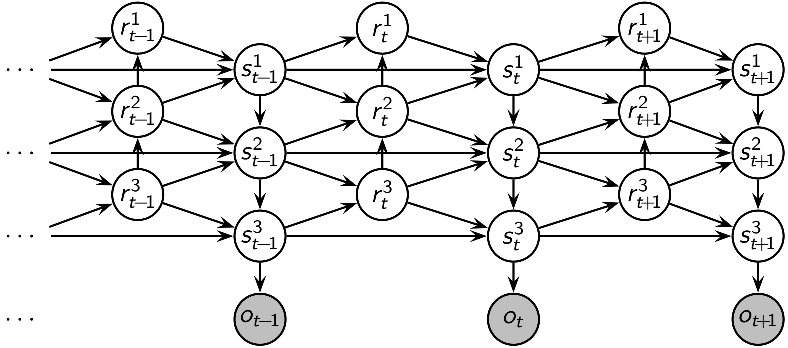


Non-reduced elements carry forward or transition (e.g. NP becomes S/VP)

Reduced elements may be expanded again (e.g. S/VP expands to Verb)

Probabilistic Time-Series Model

Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)



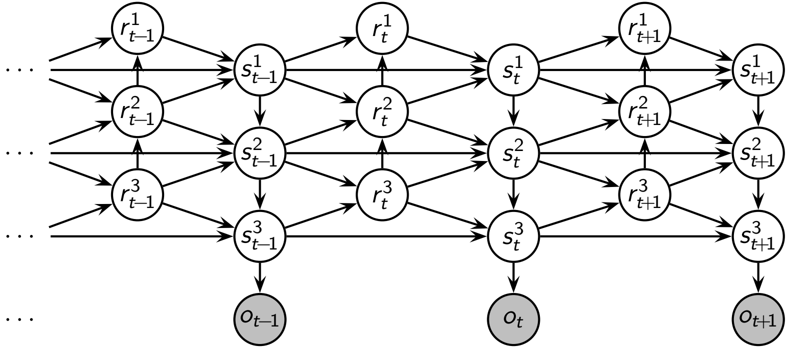
Non-reduced elements carry forward or transition (e.g. NP becomes S/VP)

Reduced elements may be expanded again (e.g. S/VP expands to Verb)

Process continues through time

Probabilistic Time-Series Model

Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)

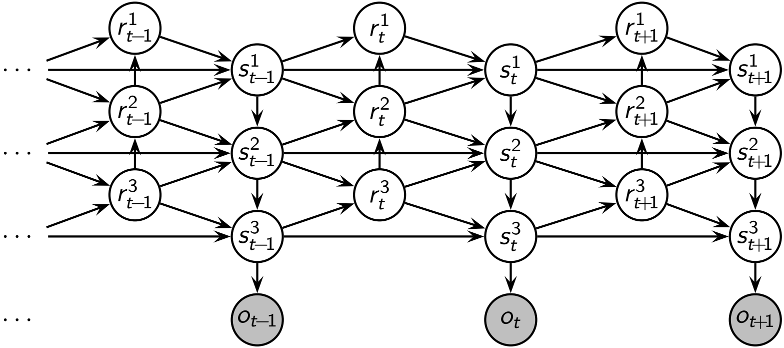


Alternate hypotheses (memory store configurations) compete w. each other:

$$\hat{s}^{1..D}_{1..T} \stackrel{\text{def}}{=} \underset{s^{1..D}_{1..T}}{\text{argmax}} \prod_{t=1}^{T} \mathsf{P}_{\Theta_{\text{LM}}}(s^{1..D}_t \mid s^{1..D}_{t-1}) \cdot \mathsf{P}_{\Theta_{\text{OM}}}(o_t \mid s^{1..D}_t)$$

Probabilistic Time-Series Model

Interactive semantics: Hierarchic Hidden Markov Model (Murphy,Paskin'01)

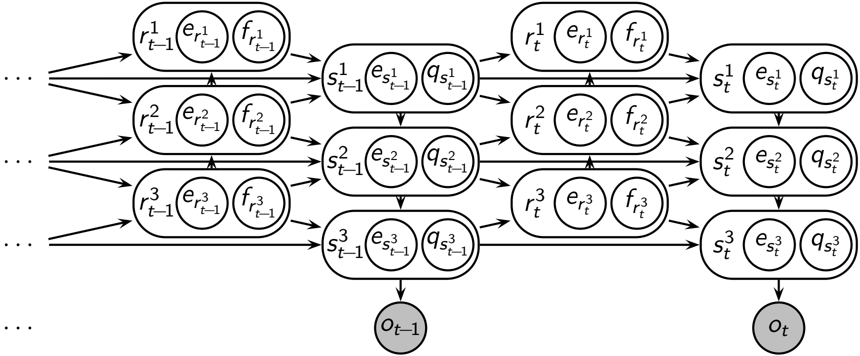


$$\mathsf{P}_{\Theta_{\mathrm{LM}}}(s_t^{1..D} \mid s_{t\text{-}1}^{1..D}) = \sum_{r_t^{1..D}} \mathsf{P}_{\Theta_{\mathrm{Reduce}}}(r_t^{1..D} \mid s_{t\text{-}1}^{1..D}) \cdot \mathsf{P}_{\Theta_{\mathrm{Shift}}}(s_t^{1..D} \mid r_t^{1..D}\ s_{t\text{-}1}^{1..D})$$

$$\stackrel{\mathrm{def}}{=} \sum_{r_t^{1..D}} \prod_{d=1}^{D} \mathsf{P}_{\Theta_\rho}(r_t^d \mid r_t^{d+1} s_{t\text{-}1}^d s_{t\text{-}1}^{d\text{-}1}) \cdot \mathsf{P}_{\Theta_\sigma}(s_t^d \mid r_t^{d+1} r_t^d\ s_{t\text{-}1}^d s_t^{d\text{-}1})$$

Probabilistic Time-Series Model

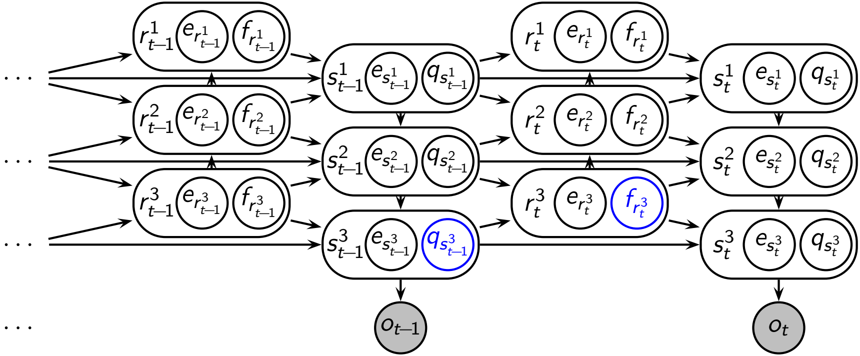
Add interactive semantics — simply factor HHMM states:



— factor r, s into interdependent syntactic (q/f) and referential (e) states:

Probabilistic Time-Series Model

Add interactive semantics — simply factor HHMM states:

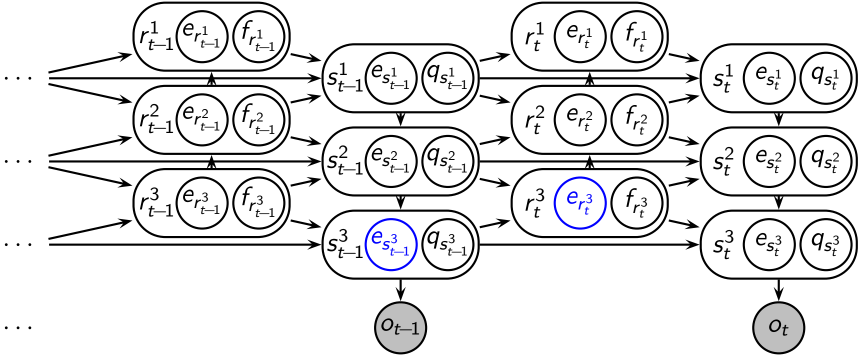


— factor r, s into interdependent syntactic (q/f) and referential (e) states:

- **incomplete syntactic states:** e.g. $q = \text{S/VP}$ (with f as a reduce flag)

Probabilistic Time-Series Model

Add interactive semantics — simply factor HHMM states:



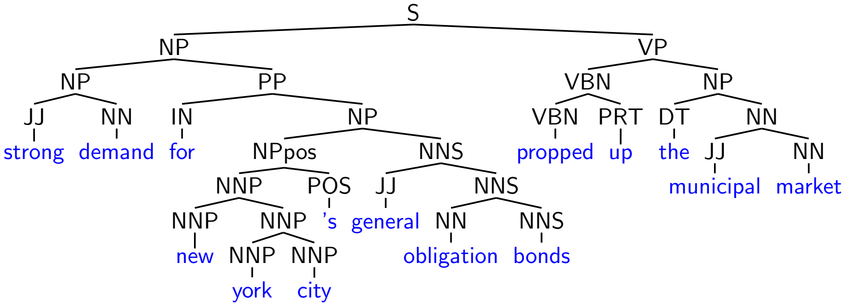
— factor r, s into interdependent syntactic (q/f) and referential (e) states:

- **incomplete syntactic states:** e.g. $q = \mathrm{S/VP}$ (with f as a reduce flag)
- **incomplete referential states:** e.g. $e = \{i_{\mathrm{coling}}, i_{\mathrm{naacl}}\}$ (entity set/class)

Connecting Generative Grammar to Time-Series Model

Sequences of memory stores correspond directly to familiar phrase structure:

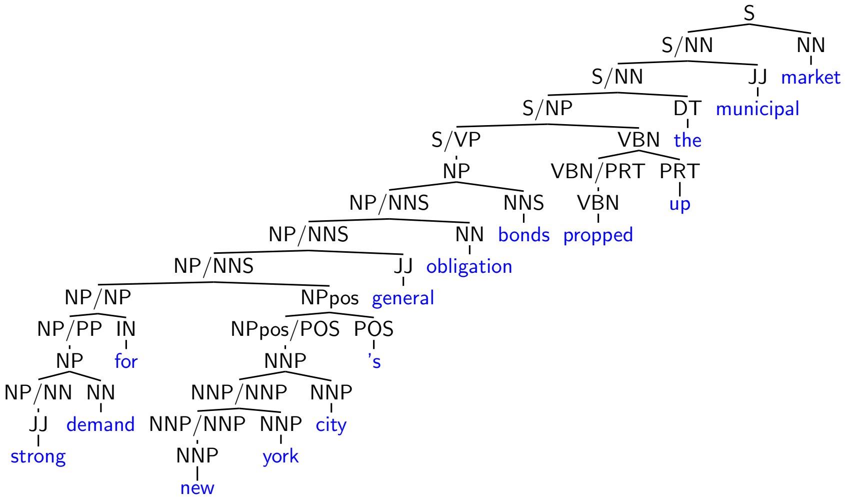
(trees from Penn Treebank, modified to featurize empty categories)



Correspondence requires flatter, more memory-efficient representation...

Connecting Generative Grammar to Time-Series Model

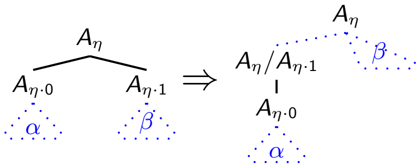
'Right-corner transform' map right-embedded sequence → left-embedded seq. (allows new constituents to be immediately composed)



Right-Corner Transform

Transform is simple — **three cases** on right-embedded sequence:
(η, μ are paths of 0:left/1:right)

Beginning case:

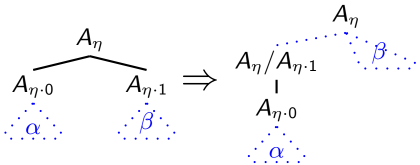


(left child → top/right)

Right-Corner Transform

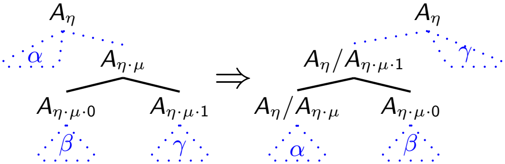
Transform is simple — **three cases** on right-embedded sequence:
(η, μ are paths of 0:left/1:right)

Beginning case:



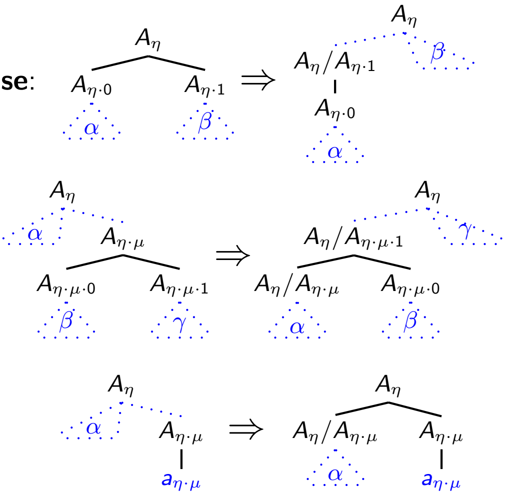
(left child → top/right)

Middle case:



(left child → top/right)

Transform is simple — **three cases** on right-embedded sequence:
(η, μ are paths of 0:left/1:right)



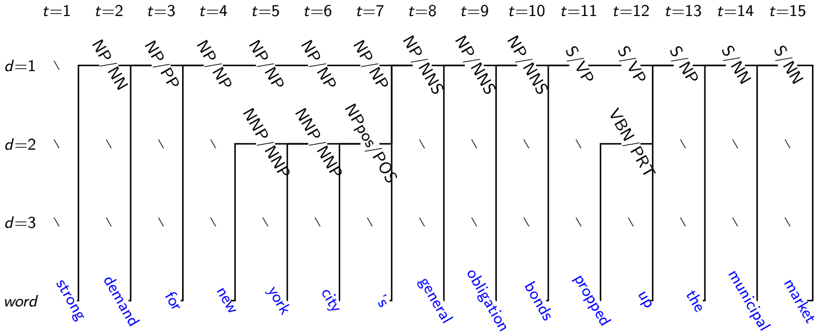
Beginning case: (left child → top/right)

Middle case: (left child → top/right)

Ending case: (left child → top/right)

Connecting Generative Grammar to Time-Series Model

Align levels to a grid, to train HHMM:



Time-order parsing based on familiar phrase structure grammar rules

Interactive Interpretation

Add interactive meaning:

- last year: first-order objects (individual files/directories)
  runs in real time w. 4000 individuals, 4000 words

Interactive Interpretation

Add interactive meaning:

- last year: first-order objects (individual files/directories)
  runs in real time w. 4000 individuals, 4000 words
- now: redundancy requires second-order objects (sets of individuals)
  runs in real time w. 100 individuals, 1000 words

Interactive Interpretation

Add interactive meaning:

- last year: first-order objects (individual files/directories)
  runs in real time w. 4000 individuals, 4000 words
- now: redundancy requires second-order objects (sets of individuals)
  runs in real time w. 100 individuals, 1000 words

  syntax, semantics defined w. familiar grammar rules, set operations
  (someday, by user – fully extensible language model)

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.file(x)$
  $\lambda y.\lambda x.contain(x, y)$

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.file(x)$
  $\lambda y.\lambda x.contain(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile} : \lambda X.\{x \mid x \in X \wedge executable(x)\}$
  $\textsc{Contain} : \lambda X.\{y \mid x \in X \wedge contain(x, y)\}$
  $\textsc{Contain}' : \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge contain(x, y)\}$

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.file(x)$
  $\lambda y.\lambda x.contain(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile} : \lambda X.\{x \mid x \in X \wedge executable(x)\}$
  $\textsc{Contain} : \lambda X.\{y \mid x \in X \wedge contain(x, y)\}$
  $\textsc{Contain}' : \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge contain(x, y)\}$

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.\mathit{file}(x)$
  $\lambda y.\lambda x.\mathit{contain}(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile} : \lambda X.\{x \mid x \in X \wedge \mathit{executable}(x)\}$
  $\textsc{Contain} : \lambda X.\{y \mid x \in X \wedge \mathit{contain}(x, y)\}$
  $\textsc{Contain}' : \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge \mathit{contain}(x, y)\}$
- operators can be associated with rules
  RC $\rightarrow$ ($\textsc{Contain}$) containing NP ($\textsc{Contain}'$)
  NP $\rightarrow$ ($\textsc{ExeFile}$) executables

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.file(x)$
  $\lambda y.\lambda x.contain(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile} : \lambda X.\{x \mid x \in X \wedge executable(x)\}$
  $\textsc{Contain} : \lambda X.\{y \mid x \in X \wedge contain(x, y)\}$
  $\textsc{Contain}' : \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge contain(x, y)\}$
- operators can be associated with rules, traversed/composed in order:
  RC $\rightarrow$ ($\textsc{Contain}$) containing NP ($\textsc{Contain}'$)
  NP $\rightarrow$ ($\textsc{ExeFile}$) executables
  $\{d_1 d_2 d_3\} \circ \textsc{Contain} = \{f_2 f_3\} \ldots$ (start w. directories, get contents)

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.file(x)$
  $\lambda y.\lambda x.contain(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile} : \lambda X.\{x \mid x \in X \wedge executable(x)\}$
  $\textsc{Contain} : \lambda X.\{y \mid x \in X \wedge contain(x, y)\}$
  $\textsc{Contain}' : \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge contain(x, y)\}$
- operators can be associated with rules, traversed/composed in order:
  RC → ($\textsc{Contain}$) containing NP ($\textsc{Contain}'$)
  NP → ($\textsc{ExeFile}$) executables
  $\{d_1 d_2 d_3\} \circ \textsc{Contain} \circ \textsc{ExeFile} = \{f_2\} \ldots$ (get executable contents)

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.file(x)$
  $\lambda y.\lambda x.contain(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile} : \lambda X.\{x \mid x \in X \wedge executable(x)\}$
  $\textsc{Contain} : \lambda X.\{y \mid x \in X \wedge contain(x, y)\}$
  $\textsc{Contain}' : \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge contain(x, y)\}$
- operators can be associated with rules, traversed/composed in order:
  RC $\rightarrow$ ($\textsc{Contain}$) containing NP ($\textsc{Contain}'$)
  NP $\rightarrow$ ($\textsc{ExeFile}$) executables
  $\{d_1 d_2 d_3\} \circ \textsc{Contain} \circ \textsc{ExeFile} \circ \textsc{Contain}' = \{d_2 s_1\} \ldots$ (containers)

Interactive Interpretation – transitions

Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.file(x)$
  $\lambda y.\lambda x.contain(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile}: \lambda X.\{x \mid x \in X \wedge executable(x)\}$
  $\textsc{Contain}: \lambda X.\{y \mid x \in X \wedge contain(x, y)\}$
  $\textsc{Contain}': \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge contain(x, y)\}$
- operators can be associated with rules, traversed/composed in order:
  RC → ($\textsc{Contain}$) containing NP ($\textsc{Contain}'$)
  NP → ($\textsc{ExeFile}$) executables
  $\{d_1 d_2 d_3\} \circ \textsc{Contain} \circ \textsc{ExeFile} \circ \textsc{Contain}'(d_1 d_2 d_3) = \{d_2\} \quad (\lambda X{:}\cap)$

Interactive Interpretation – transitions

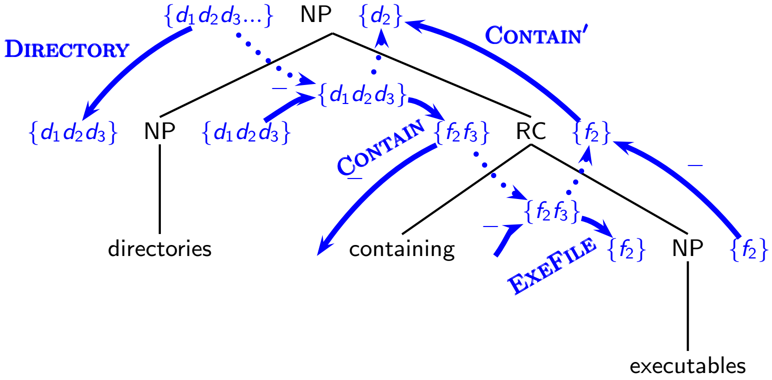
Interactive interpretation defined using $e \rightarrow e$ transitions in HHMM

- assume finite domain of individuals $\mathcal{E}$
- assume functions true or false over individuals:
  $\lambda x.\mathit{file}(x)$
  $\lambda y.\lambda x.\mathit{contain}(x, y)$
- functions define transitions/operators from set to set:
  $\textsc{ExeFile} : \lambda X.\{x \mid x \in X \wedge \mathit{executable}(x)\}$
  $\textsc{Contain} : \lambda X.\{y \mid x \in X \wedge \mathit{contain}(x, y)\}$
  $\textsc{Contain}' : \lambda Y.\lambda X.\{x \mid x \in X \wedge y \in Y \wedge \mathit{contain}(x, y)\}$
- operators can be associated with rules, traversed/composed in order:
  RC $\rightarrow$ ($\textsc{Contain}$) containing NP ($\textsc{Contain}'$)
  NP $\rightarrow$ ($\textsc{ExeFile}$) executables
  $\{d_1 d_2 d_3\} \circ \textsc{Contain} \circ \textsc{ExeFile} \circ \textsc{Contain}'(d_1 d_2 d_3) = \{d_2\} \quad (\lambda X{:}\cap)$

Semantics now 'ride along' through transform as operator/transition chains

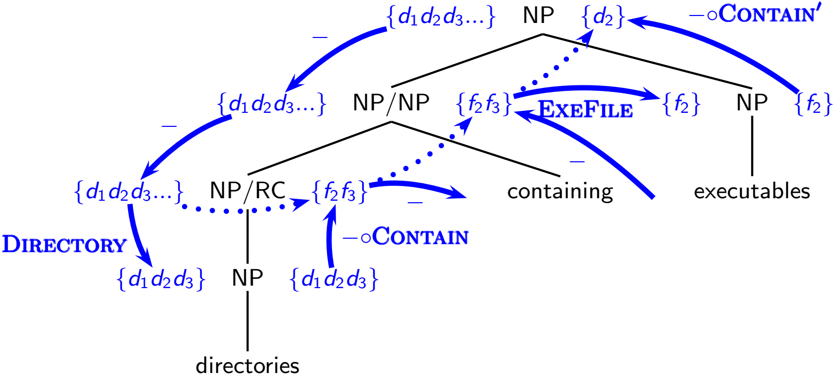
Right-Corner Transform on Operator Chains

Operator chains prior to transform (dot arcs show λX dependencies):



Right-Corner Transform on Operator Chains

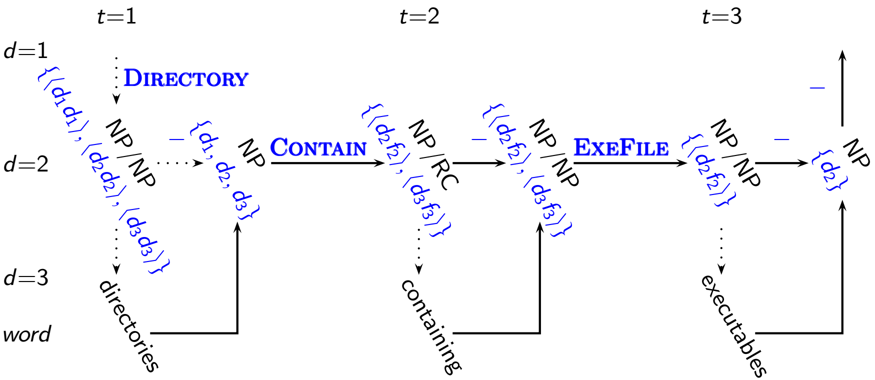
Operator chains following transform:



Left legs turn into right legs, but operators keep order, structure

Right-Corner Transform on Operator Chains

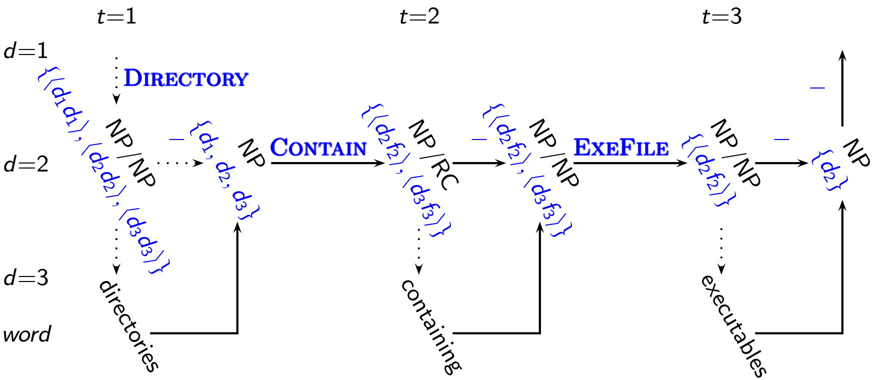
Transformed operations aligned to time-series model:



Time-order interpretation from familiar grammar rules, set operations

Right-Corner Transform on Operator Chains

Transformed operations aligned to time-series model:



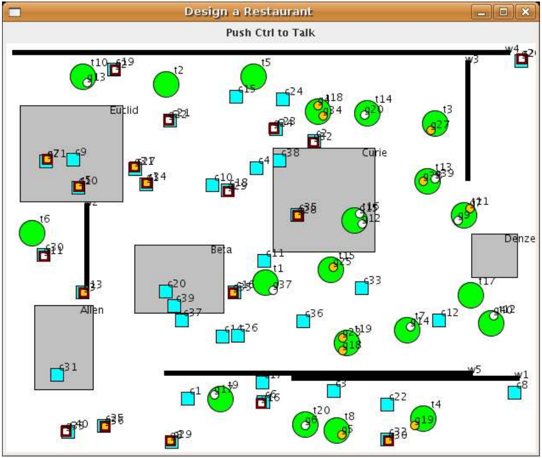
Time-order interpretation from familiar grammar rules, set operations

Interpretations dynamically calculated, then used to rate hypotheses (prob. based on denotation cardinality before and after each operation)

Evaluation: scalable to second-order denotations

Bounded model allows 2ndorder denotations in real time speech (13% SER)

Restaurant domain: 'select the glasses on chairs'



Evaluation: scalable to second-order denotations

Beam=100, Individuals=110, Lexicon=50 — 100 directives test:

Subject	Sentence error rate	Corrected on 1st retry	Corrected on 2nd retry
1	2 / 20	2	-
2	2 / 20	1	1
3	3 / 20	2	1
4	4 / 20	4	-
5	2 / 20	1	1
Total	13%	10	3

Evaluation: scalable to second-order denotations

Beam=100, Individuals=110, Lexicon=50 — 100 directives test:

Subject	Sentence error rate	Corrected on 1st retry	Corrected on 2nd retry
1	2 / 20	2	-
2	2 / 20	1	1
3	3 / 20	2	1
4	4 / 20	4	-
5	2 / 20	1	1
Total	13%	10	3

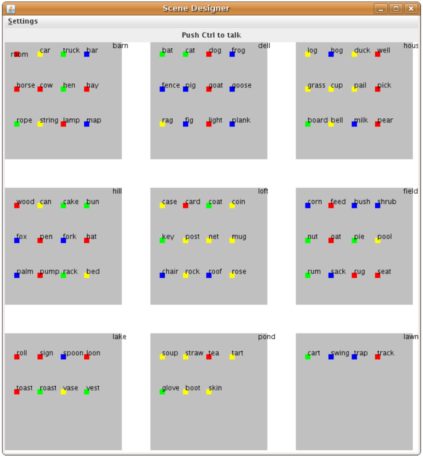
Beam=100, Individuals=110, Lexicon=1000 — 60 directives test:

Subject	Sentence error rate
1	2 / 20
2	1 / 20
3	5 / 20
Total	13%

Evaluation: redundancy improves accuracy

Does interactive model let speaker be redundanct to improve communication?

Monosyllabic domain: 'select the seat [to the right of the rug]'



Evaluation: redundancy improves accuracy

Beam=100, Individuals=100, Lexicon=100 — 1000 directives test:

Subject	Sentence error rate without redundancy	Sentence error rate with redundancy
1	54 / 100	37 / 100
2	32 / 100	21 / 100
3	25 / 100	18 / 100
4	28 / 100	12 / 100
5	24 / 100	15 / 100
All	32.6%	20.6%

Natural model of using redundancy to ensure correct interpretation.

Conclusion

In summary: useful model of interactive comprehension!

Conclusion

In summary: useful model of interactive comprehension!

- Interactive interpretation with second-order referents (sets of indivs)

Conclusion

In summary: useful model of interactive comprehension!

- Interactive interpretation with second-order referents (sets of indivs)
- Runs in real time

Conclusion

In summary: useful model of interactive comprehension!

- Interactive interpretation with second-order referents (sets of indivs)
- Runs in real time
- Based on familiar notions of phrase structure, semantic composition

Conclusion

In summary: useful model of interactive comprehension!

- Interactive interpretation with second-order referents (sets of indivs)
- Runs in real time
- Based on familiar notions of phrase structure, semantic composition
- Interactive interpretation lets user be redundant to improve accuracy

Conclusion

Thank you!