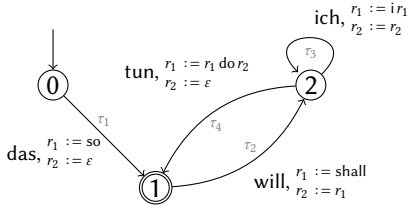
Statistical machine translation using streaming string transducers

Thomas Ruprecht
thomas.ruprecht@tu-dresden.de

Institute for Theoretical Computer Science
Faculty of Computer Science
Technische Universität Dresden, Germany

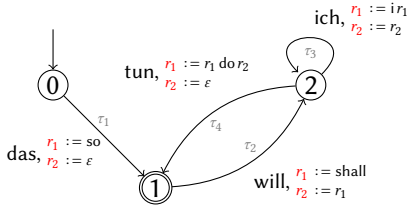
April 22, 2021

(Weighted) streaming string transducers (Alur and Deshmukh 2011)



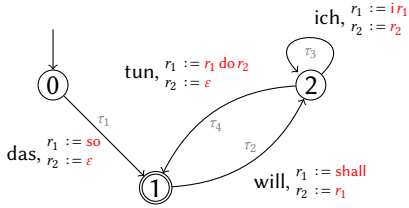
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers k*



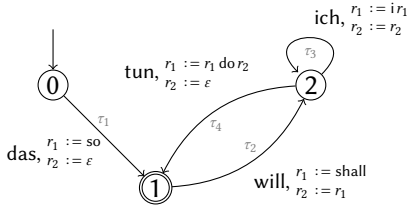
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying



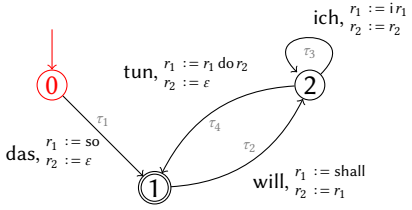
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



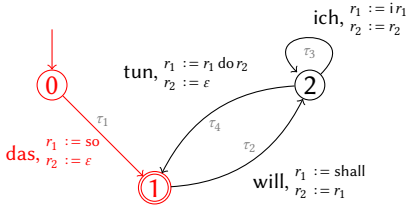
run

input

output

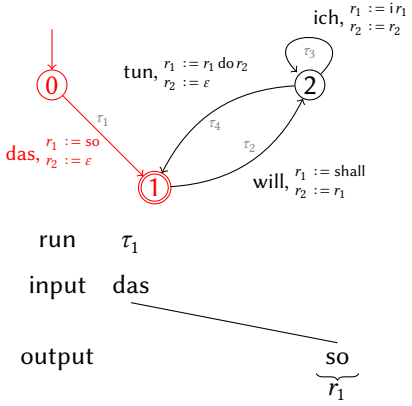
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



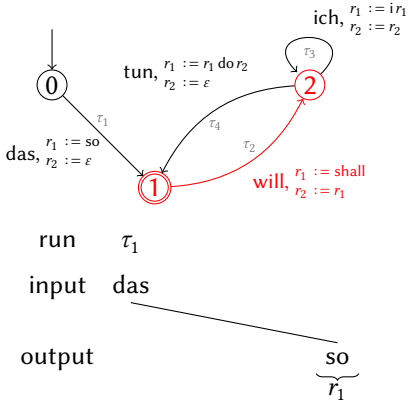
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



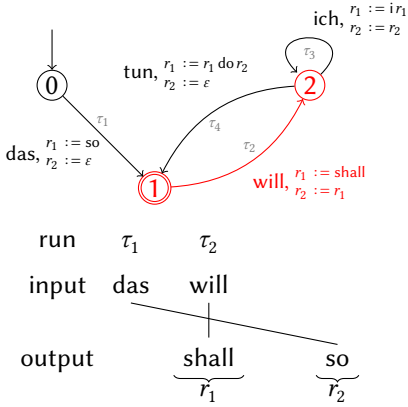
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



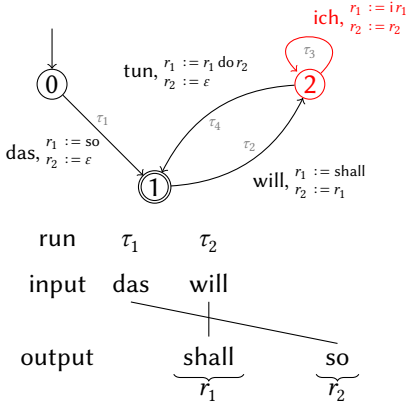
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



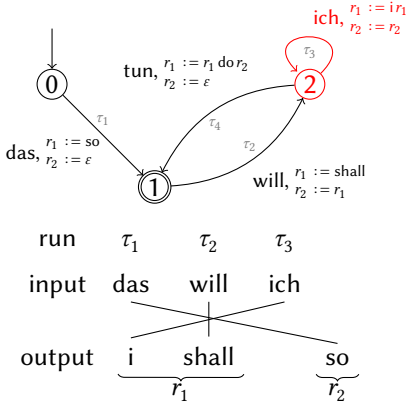
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



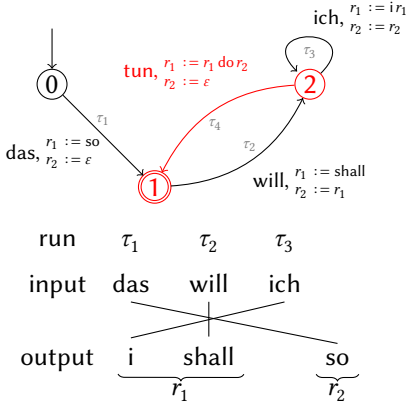
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



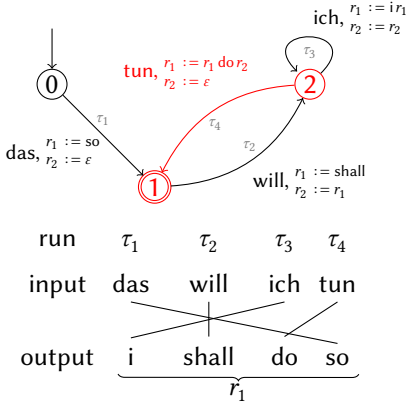
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



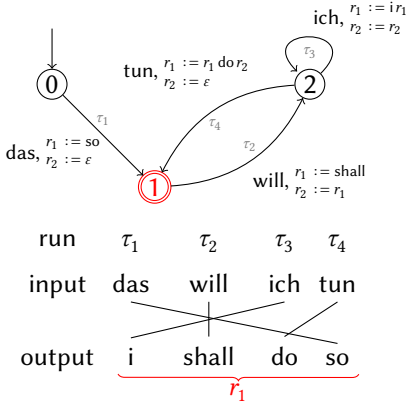
(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)



(Weighted) streaming string transducers (Alur and Deshmukh 2011)

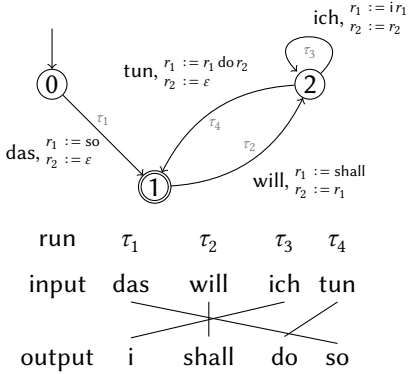
- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)
- output: first register



(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)
- output: first register
- weights: probabilities conditioned on source state

$$p(\overset{\text{das will ich tun}}{\overset{\text{i shall do so}}{\times}}) = p(\tau_1) \cdot p(\tau_2) \cdot p(\tau_3) \cdot p(\tau_4)$$

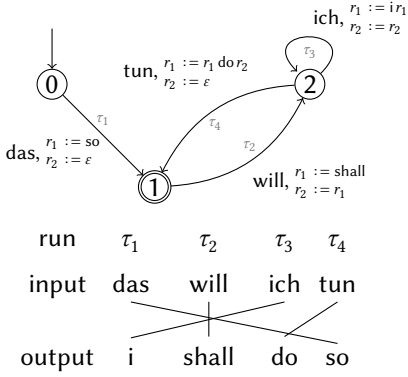


(Weighted) streaming string transducers (Alur and Deshmukh 2011)

- fixed number of *registers* k
- register assignment:
  - map k strings to k strings
  - non-copying
- *origin semantics*: map output pos. to "responsible" input pos. (Bojańczyk 2014)
- output: first register
- weights: probabilities conditioned on source state

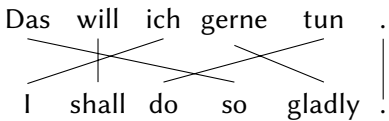
$$p(\overset{\text{das will ich tun}}{\underset{\text{i shall do so}}{\times}}) = p(\tau_1) \cdot p(\tau_2) \cdot p(\tau_3) \cdot p(\tau_4)$$

  - sum weight of multiple runs in A for input & output (& origins)



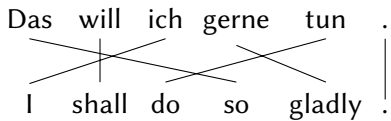
Why sst?

- origin graphs resemble translation alignments
  - functional, total wrt. output



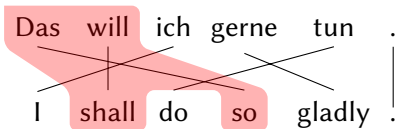
Why sst?

- origin graphs resemble translation alignments
  - functional, total wrt. output
- conceptionally simple, linear model
- capture *gaps* in alignments



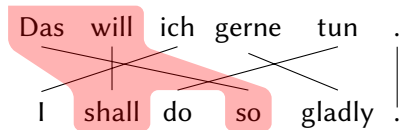
Why sst?

- origin graphs resemble translation alignments
  - functional, total wrt. output
- conceptionally simple, linear model
- capture *gaps* in alignments



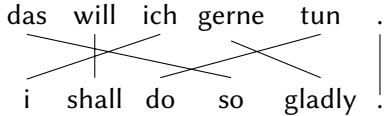
Why sst?

- origin graphs resemble translation alignments
  - functional, total wrt. output
- conceptionally simple, linear model
- capture *gaps* in alignments
- possible application in neural machine translation



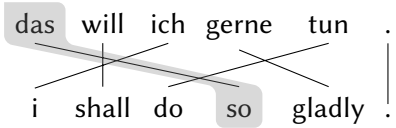
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input



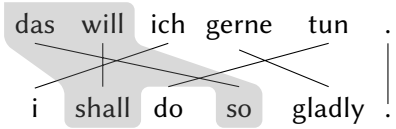
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input



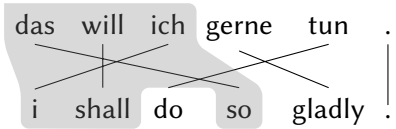
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input



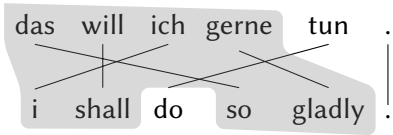
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input



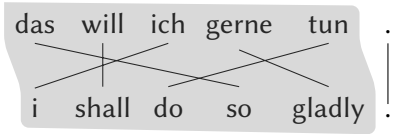
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input



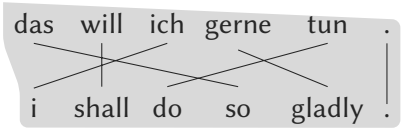
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input



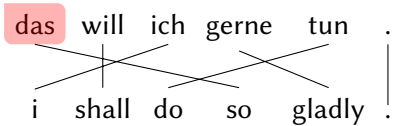
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input



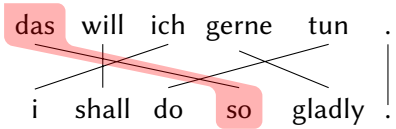
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:



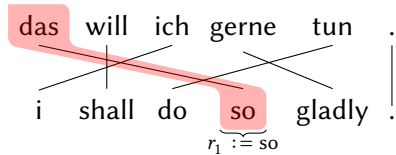
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers:
      subsequent outputs with origin $\leq i$



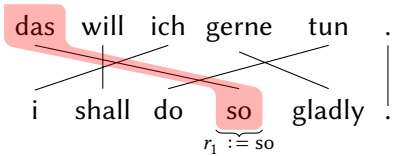
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers:
      subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal



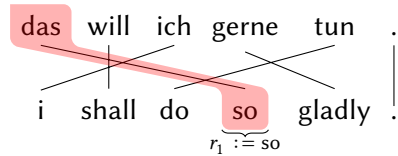
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers:
      subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable
      (combine subsequent pos.)



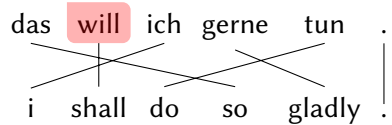
Induction from origin graphs

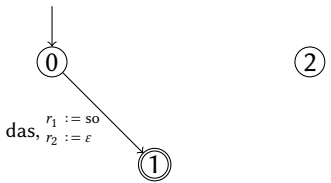
- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers



Induction from origin graphs

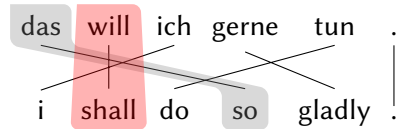
- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers

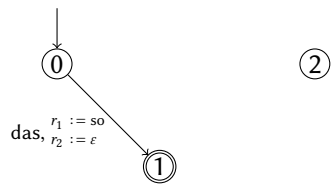




Induction from origin graphs

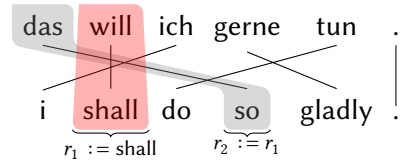
- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers

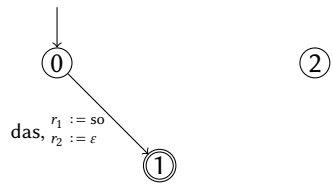




Induction from origin graphs

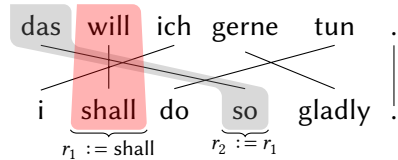
- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers

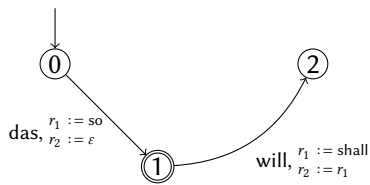




Induction from origin graphs

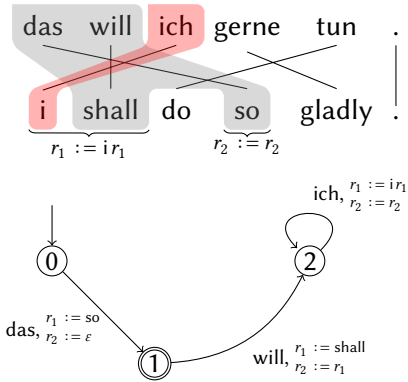
- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers





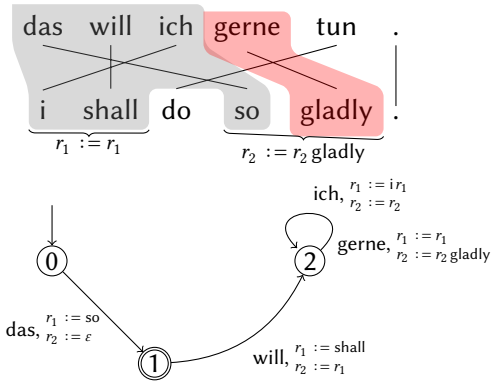
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers



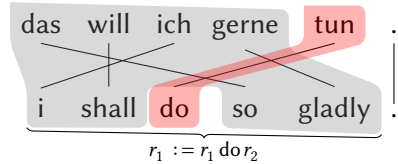
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers



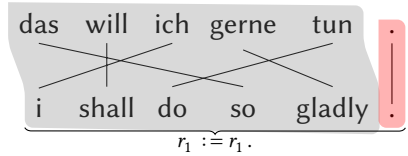
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers



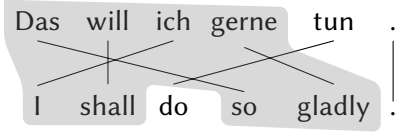
Induction from origin graphs

- k: max no. output spans with origin in a prefix of the input
- for each input position i:
  - register assignment:
    - nonempty registers: subsequent outputs with origin $\leq i$
    - origin $= i \longrightarrow$ terminal
    - origin $< i \longrightarrow$ variable (combine subsequent pos.)
  - states: no. used variables & no. used registers



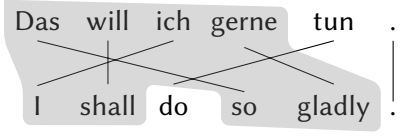
Properties of induced sst

- graphs used in induction $\subseteq$ sst's origin semantics



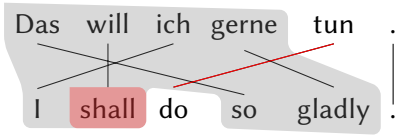
Properties of induced sst

- graphs used in induction $\subseteq$ sst's origin semantics
- minimal k:
  - origin graph recognizable by sst with k registers $\iff$ input positions *crossed by* at most k output positions (Bojańczyk et al. 2017)



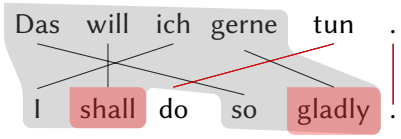
Properties of induced sst

- graphs used in induction $\subseteq$ sst's origin semantics
- minimal k:
  - origin graph recognizable by sst with k registers $\iff$ input positions *crossed by* at most k output positions (Bojańczyk et al. 2017)



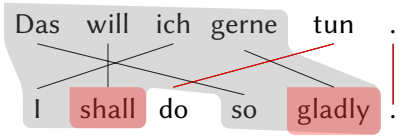
Properties of induced sst

- graphs used in induction $\subseteq$ sst's origin semantics
- minimal k:
  - origin graph recognizable by sst with k registers $\iff$ input positions *crossed by* at most k output positions (Bojańczyk et al. 2017)



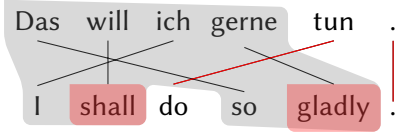
Properties of induced sst

- graphs used in induction $\subseteq$ sst's origin semantics
- minimal k:
  - origin graph recognizable by sst with k registers $\iff$ input positions *crossed by* at most k output positions (Bojańczyk et al. 2017)
  - output j *crosses* input i $\iff$ output j is the last of subsequent positions with origin $\leq i$



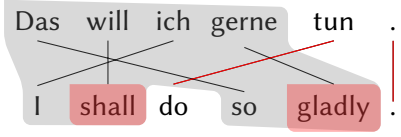
Properties of induced sst

- graphs used in induction $\subseteq$ sst's origin semantics
- minimal k:
  - origin graph recognizable by sst with k registers $\iff$ input positions *crossed by* at most k output positions (Bojańczyk et al. 2017)
  - output j *crosses* input i $\iff$ output j is the last of subsequent positions with origin $\leq i$
- unambiguous:
  - at most one run (sequence of reg. ass.) for each origin graph (output & origin ass.)



Properties of induced sst

- graphs used in induction $\subseteq$ sst's origin semantics
- minimal k:
  - origin graph recognizable by sst with k registers $\iff$ input positions *crossed by* at most k output positions (Bojańczyk et al. 2017)
  - output j *crosses* input i $\iff$ output j is the last of subsequent positions with origin $\leq i$
- unambiguous:
  - at most one run (sequence of reg. ass.) for each origin graph (output & origin ass.)
  - $\Longrightarrow$ exactly one —"— for each —"— in set used for induction



Weights for induced sst

- induced sst from set of origin graphs X

Weights for induced sst

- induced sst from set of origin graphs X

- likelihood of origin graphs:

$$\prod_{x \in X} p(x)$$

Weights for induced sst

- induced sst from set of origin graphs X
- transition $\tau(x, i)$ induced for i-th input of $x \in X$

- likelihood of origin graphs:

$$\prod_{x \in X} p(x) = \prod_{x \in X} \prod_{1 \leq i \leq |x|} p(\tau(x, i))$$

Weights for induced sst

- induced sst from set of origin graphs X
- transition $\tau(x, i)$ induced for i-th input of $x \in X$
- $c(\tau')$: no. occurrences of τ' during induction
  $\mathrm{supp}(c)$: set of induced transitions
- likelihood of origin graphs:

$$\prod_{x \in X} p(x) = \prod_{x \in X} \prod_{1 \leq i \leq |x|} p(\tau(x, i)) = \prod_{\tau' \in \mathrm{supp}(c)} p(\tau')^{c(\tau')}$$

Weights for induced sst

- induced sst from set of origin graphs X
- transition $\tau(x, i)$ induced for i-th input of $x \in X$
- $c(\tau')$: no. occurrences of τ' during induction
  $\mathrm{supp}(c)$: set of induced transitions
- likelihood of origin graphs:

$$\prod_{x \in X} p(x) = \prod_{x \in X} \prod_{1 \leq i \leq |x|} p(\tau(x, i)) = \prod_{\tau' \in \mathrm{supp}(c)} p(\tau')^{c(\tau')}$$

- max. likelihood estimate: $p(\tau') \propto c(\tau')$

Weights for induced sst

- induced sst from set of origin graphs X
- transition $\tau(x, i)$ induced for i-th input of $x \in X$
- $c(\tau')$: no. occurrences of τ' during induction
  $\mathrm{supp}(c)$: set of induced transitions
- likelihood of origin graphs:

$$\prod_{x \in X} p(x) = \prod_{x \in X} \prod_{1 \leq i \leq |x|} p(\tau(x, i)) = \prod_{\tau' \in \mathrm{supp}(c)} p(\tau')^{c(\tau')}$$

- max. likelihood estimate: $p(\tau') \propto c(\tau')$
- for input: best run $\hat{=}$ best output & alignment

Conclusion

- induction of wsst from origin graphs

Conclusion

- induction of wsst from origin graphs
- simple computation of best output & alignment for input

Conclusion

- induction of wsst from origin graphs
- simple computation of best output & alignment for input
- replacement for word-to-word translation

Conclusion

- induction of wsst from origin graphs
- simple computation of best output & alignment for input
- replacement for word-to-word translation
  - context barely considered

Conclusion

- induction of wsst from origin graphs
- simple computation of best output & alignment for input
- replacement for word-to-word translation
  - context barely considered
  - $\Longrightarrow$ combine with output language model

Conclusion

- induction of wsst from origin graphs
- simple computation of best output & alignment for input
- replacement for word-to-word translation
  - context barely considered
  - $\Longrightarrow$ combine with output language model
- cannot recognize nested structures in input

Conclusion

- induction of wsst from origin graphs
- simple computation of best output & alignment for input
- replacement for word-to-word translation
  - context barely considered
  - $\Longrightarrow$ combine with output language model
- cannot recognize nested structures in input
- model quality depends on alignments

References

Alur, R. and J. V. Deshmukh (2011). “Nondeterministic streaming string transducers”. In: *ICALP*. Springer.

Bojańczyk, M. (2014). “Transducers with origin information”. In: *ICALP*. Springer.

Bojańczyk, M. et al. (2017). “Which classes of origin graphs are generated by transducers?” In: *ICALP*.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). “Maximum Likelihood from Incomplete Data via the EM Algorithm”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1.

Kaeshammer, M. (2013). “Synchronous Linear Context-Free Rewriting Systems for Machine Translation”. In: *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*. ACL.

Lopez, A. (2008). “Statistical Machine Translation”. In: *ACM Comput. Surv.* 40.3. DOI: 10.1145/1380584.1380586.

Maier, W. and A. Søgaard (2008). “Treebanks and mild context-sensitivity”. In: *Proceedings of Formal Grammar*.

Nederhof, M.-J. and H. Vogler (2019). “Regular transductions with MCFG input syntax”. In: *Proceedings of FSMNLP*. ACL. DOI: 10.18653/v1/W19-3109.