Factor Graph Grammars for Probabilistic Modeling

Darcey Riley

2021-04-22

Mila Computational Calculus Reading Group

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Factor Graph Grammars (NeurIPS 2020)

David Chiang

dchiang@nd.edu

UNIVERSITY OF NOTRE DAME

Darcey Riley

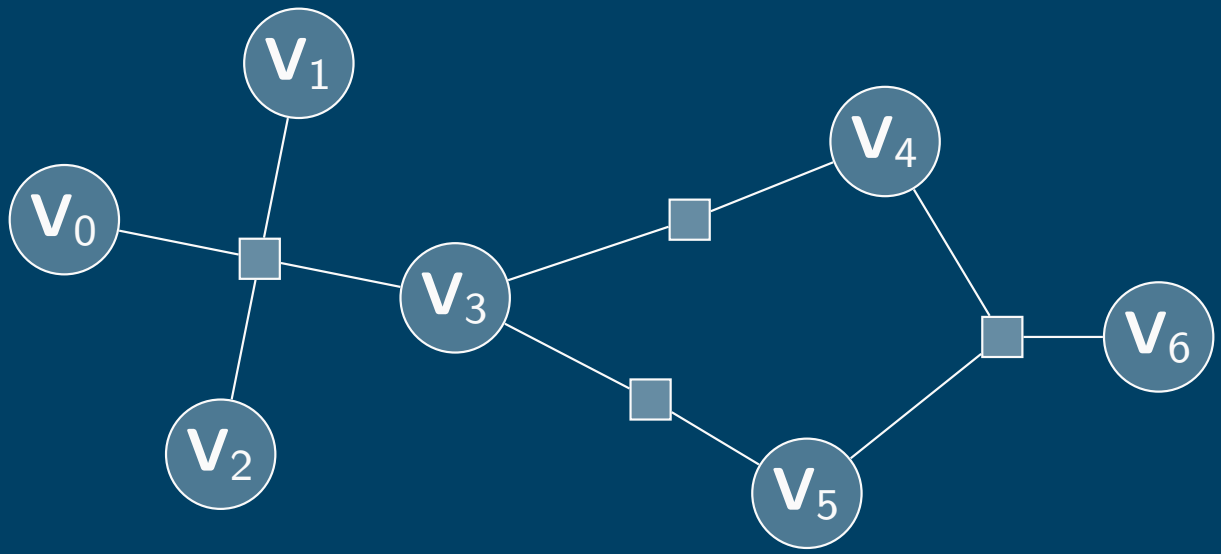
darcey.riley@nd.edu

UNIVERSITY OF NOTRE DAME

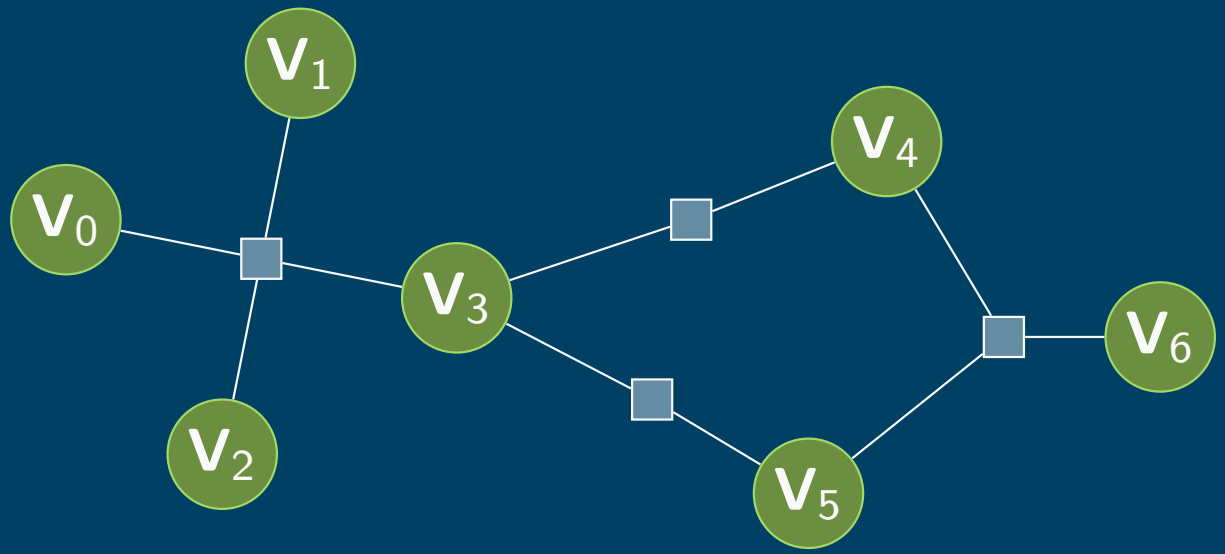
Chiang, David, and Darcey Riley. "Factor Graph Grammars." *Advances in Neural Information Processing Systems* 33 (2020).

Factor Graphs and Their Limitations

Factor Graphs

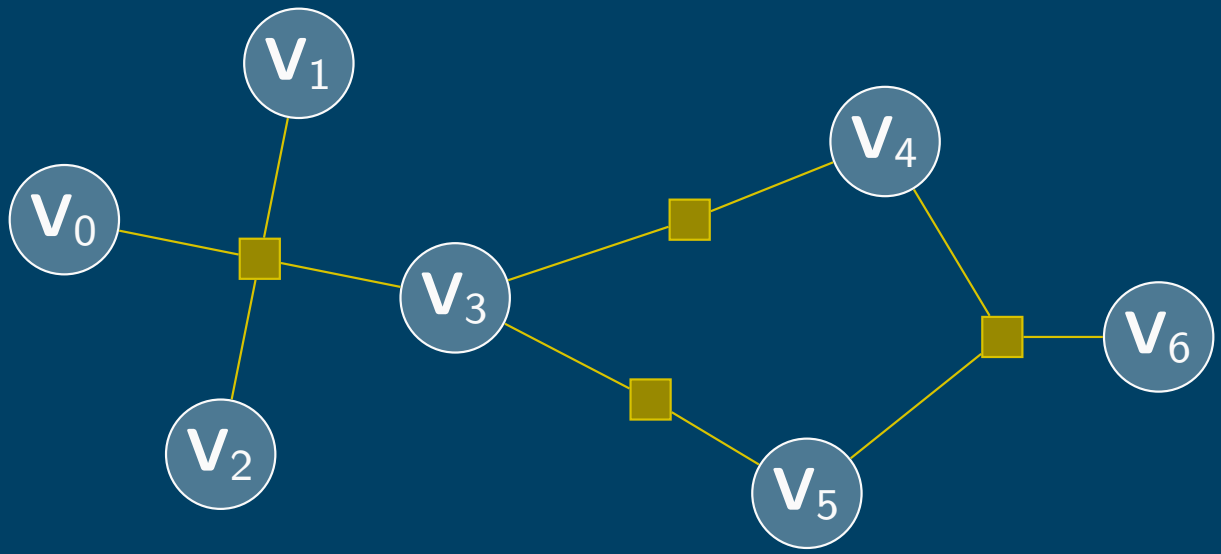


Factor Graphs



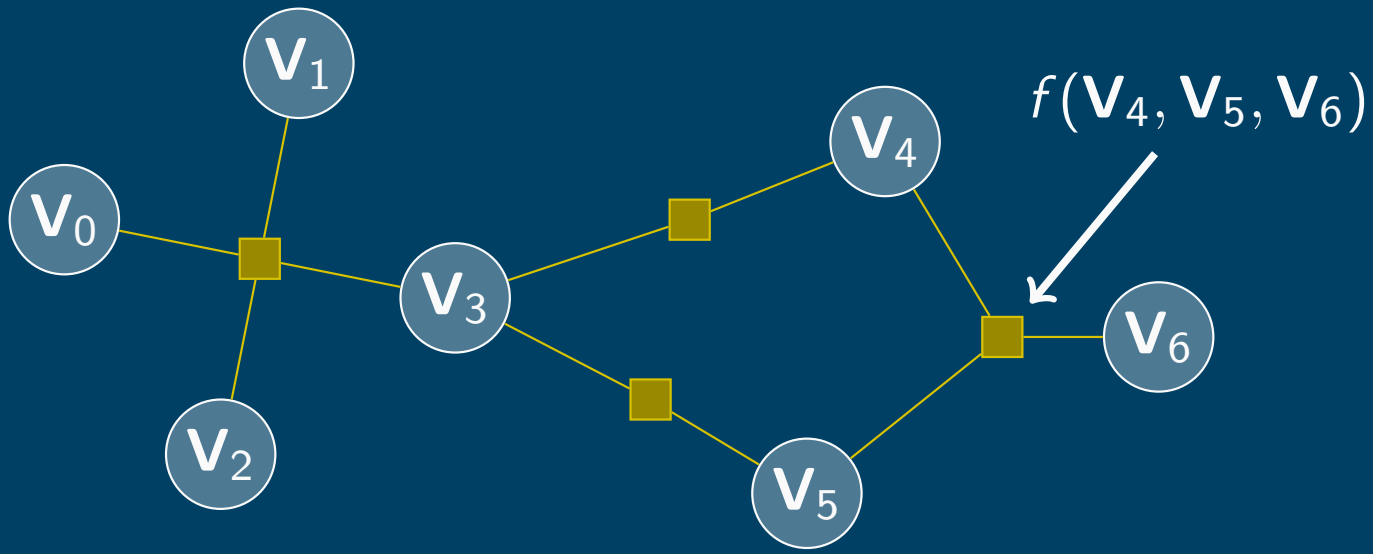
variables

Factor Graphs



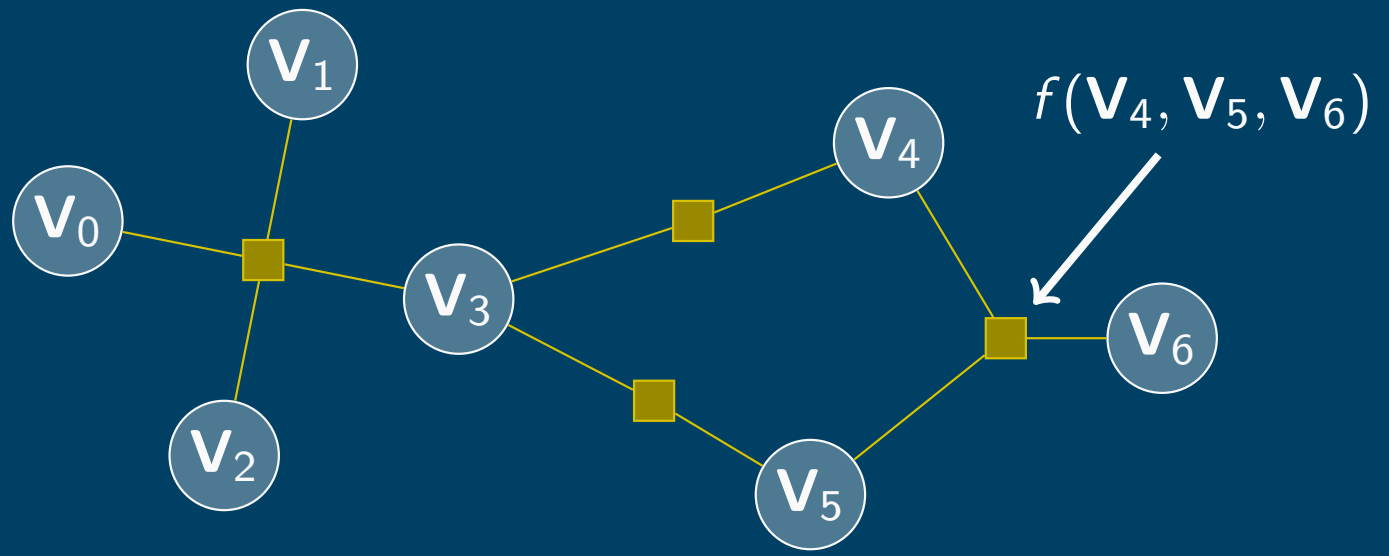
factors

Factor Graphs



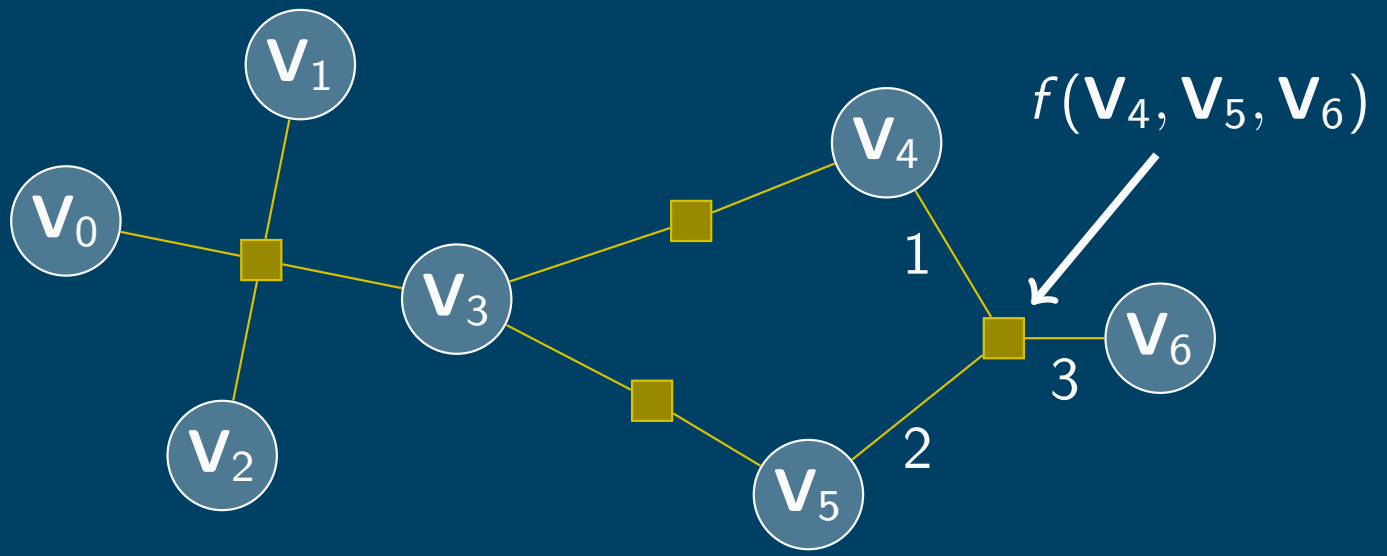
factors

Factor Graphs



factors (hyperedges)

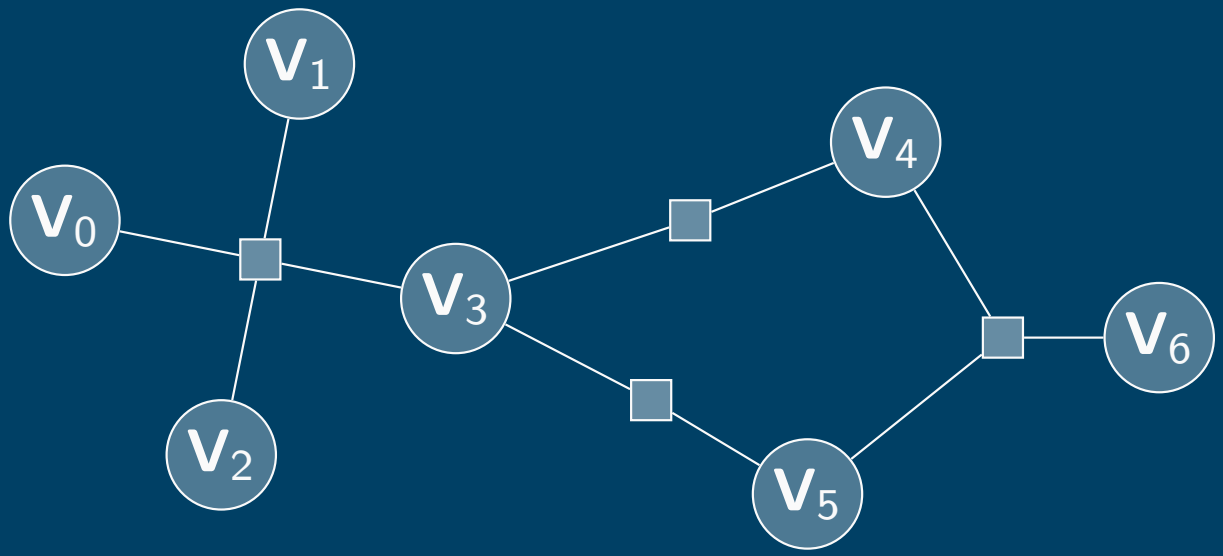
Factor Graphs



factors
(hyperedges)

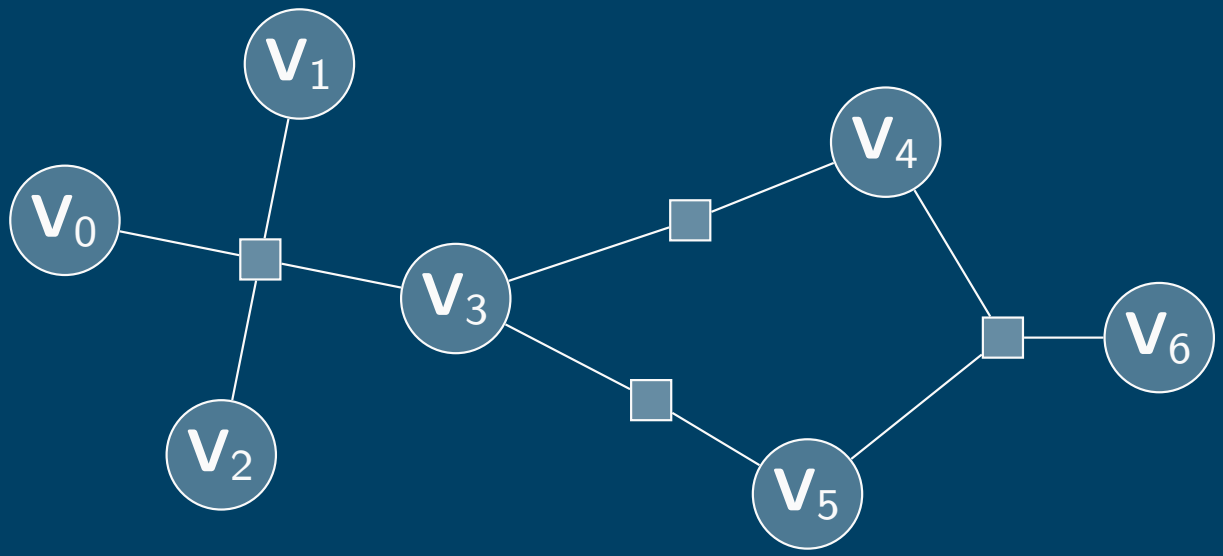
Computing Probabilities

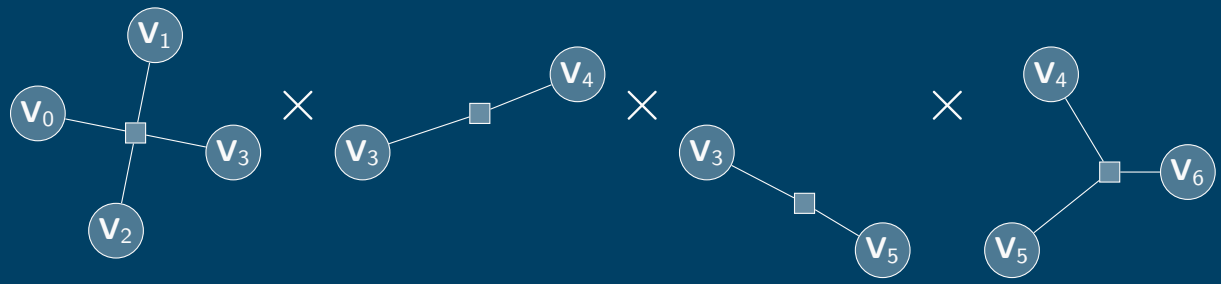
H:



Computing Probabilities

H:

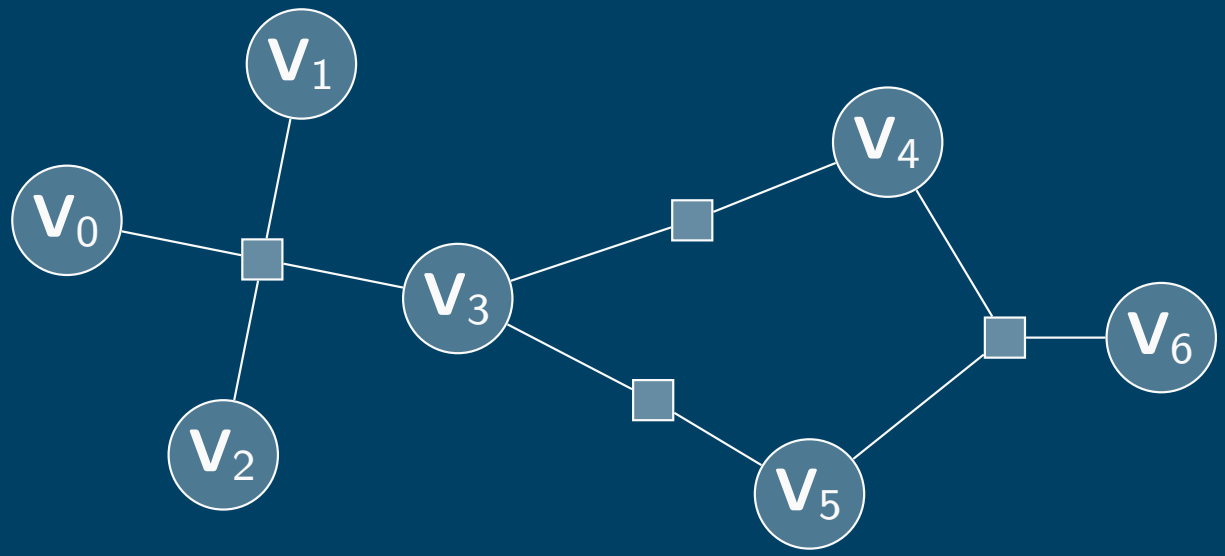




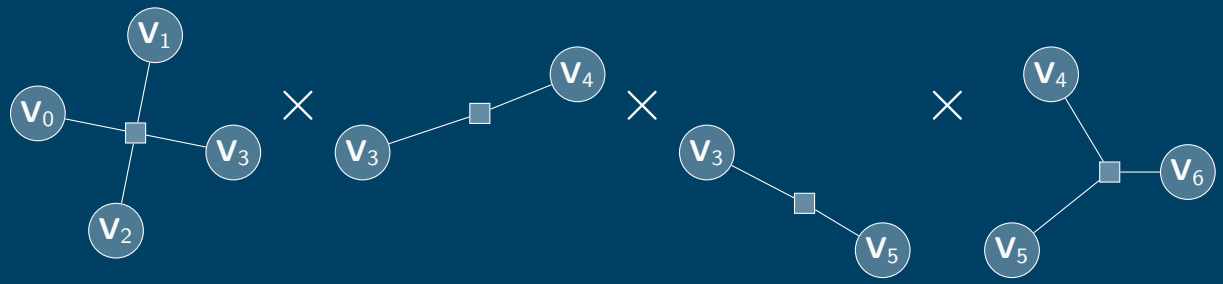
To compute weight of assignment: multiply factors together

Computing Probabilities

H:



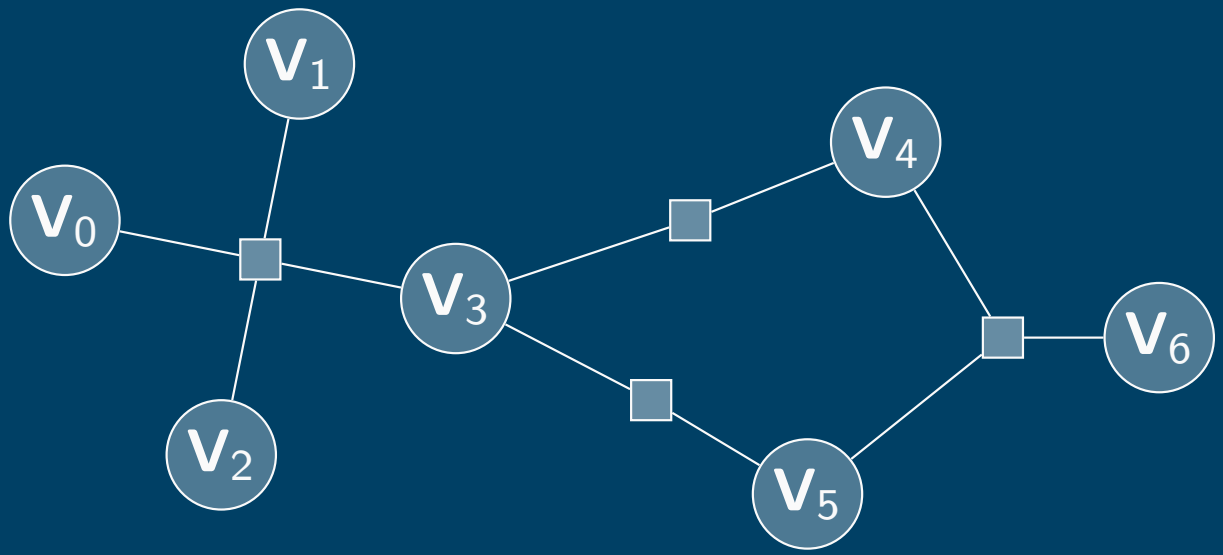
$$\frac{1}{Z}$$



To convert weight to probability: divide by normalizing constant Z

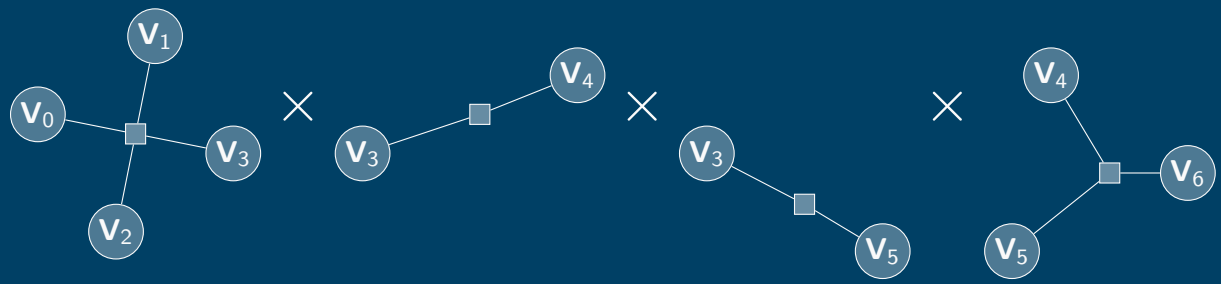
Computing Probabilities

H:



$Z(H) =$

$$\sum_{\text{assignments to } V_0, V_1, \ldots, V_6}$$

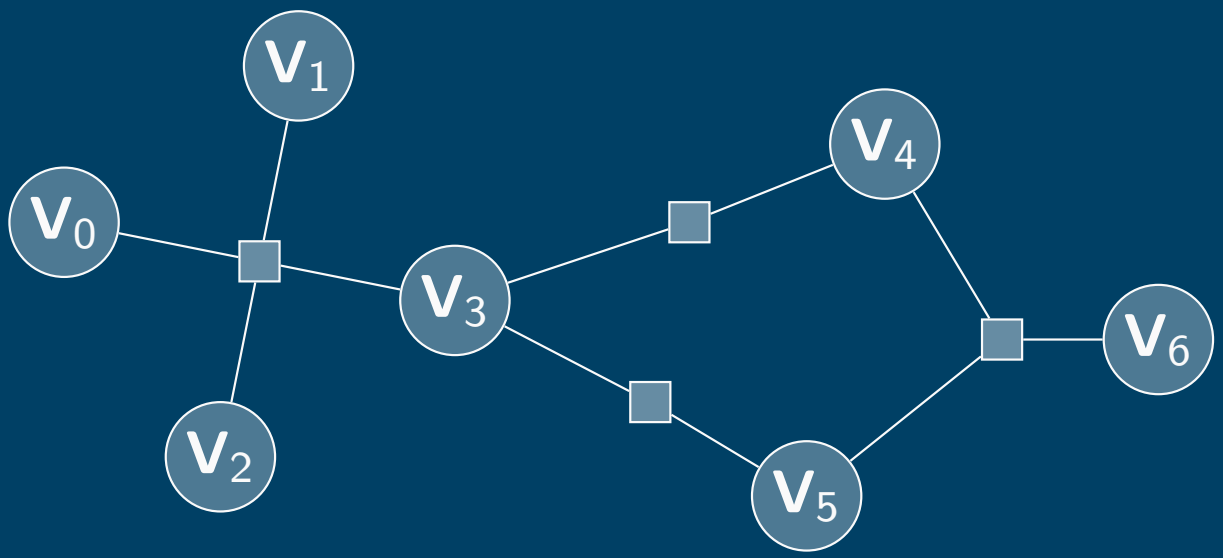


To compute normalizing constant: sum over the weights of all assignments

Limitations

Factor graphs are powerful...

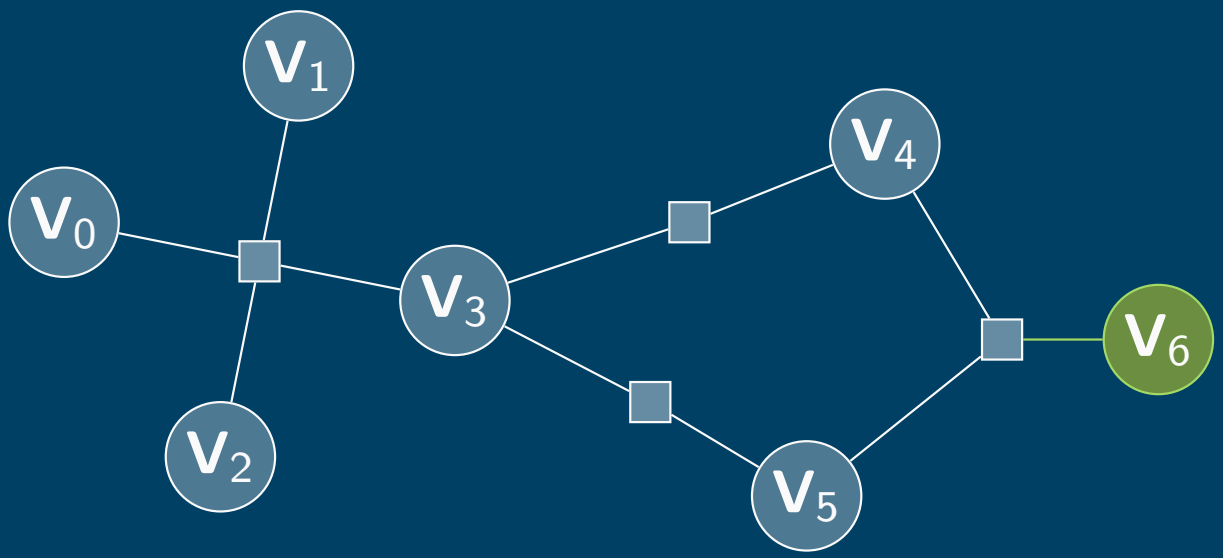
...but they are limited by their fixed structure.



Limitations

Factor graphs are powerful...

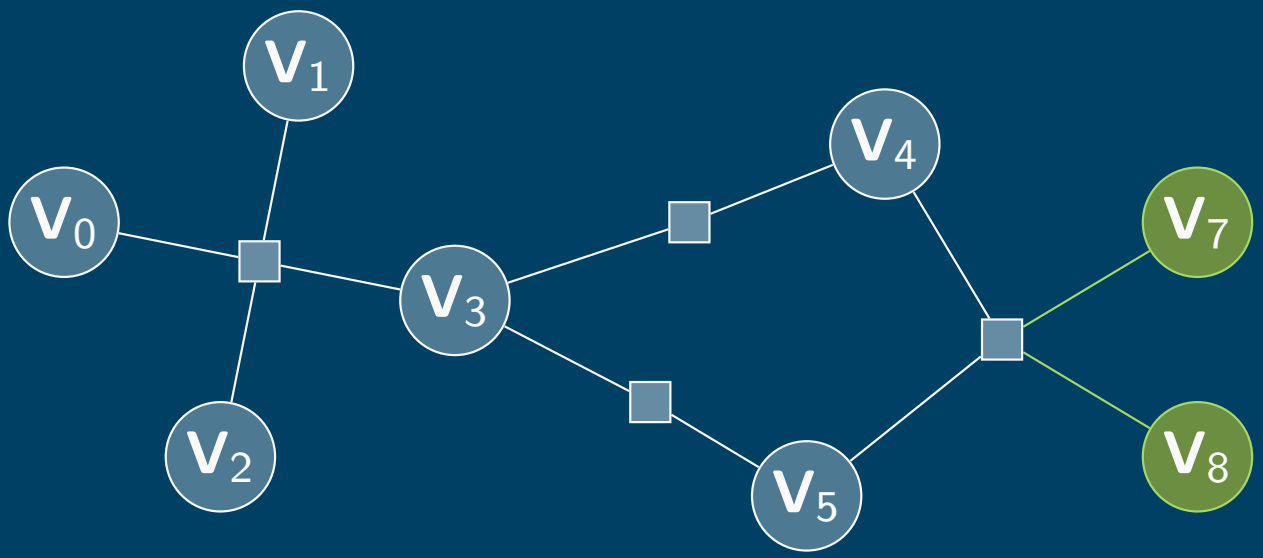
...but they are limited by their fixed structure.



Limitations

Factor graphs are powerful...

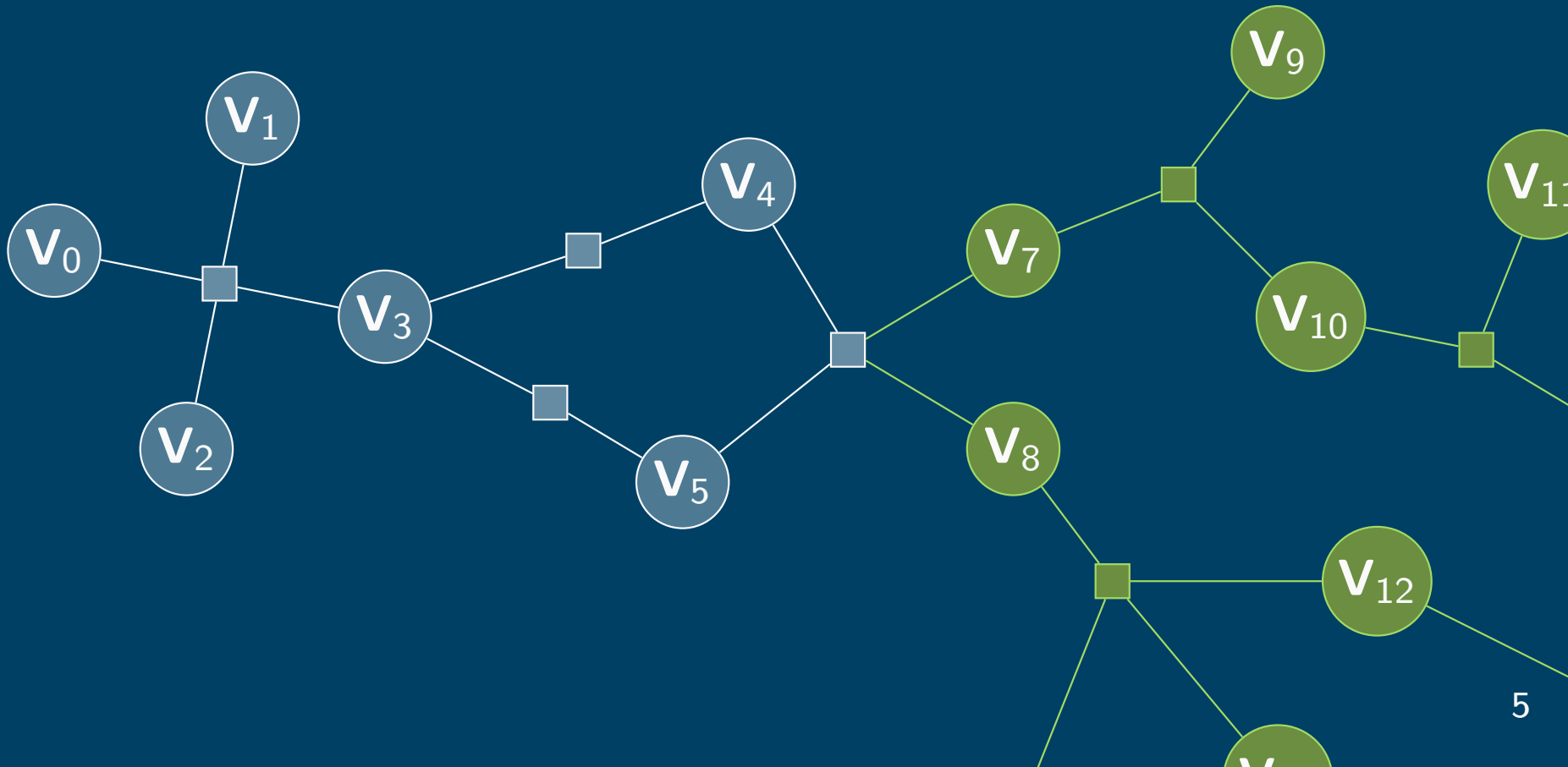
...but they are limited by their fixed structure.



Limitations

Factor graphs are powerful...

...but they are limited by their fixed structure.



Limitations

Can't represent many models from NLP.

Want a probability model over all possible sentences:

The dog was eating spaghetti.

I hate getting a flat tire.

Nobody wears orange pajamas while playing the accordion.

⋮

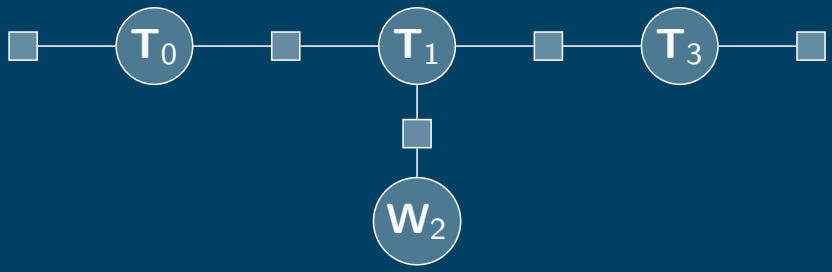
These sentences have unbounded length.

Limitations

Can't represent HMMs:

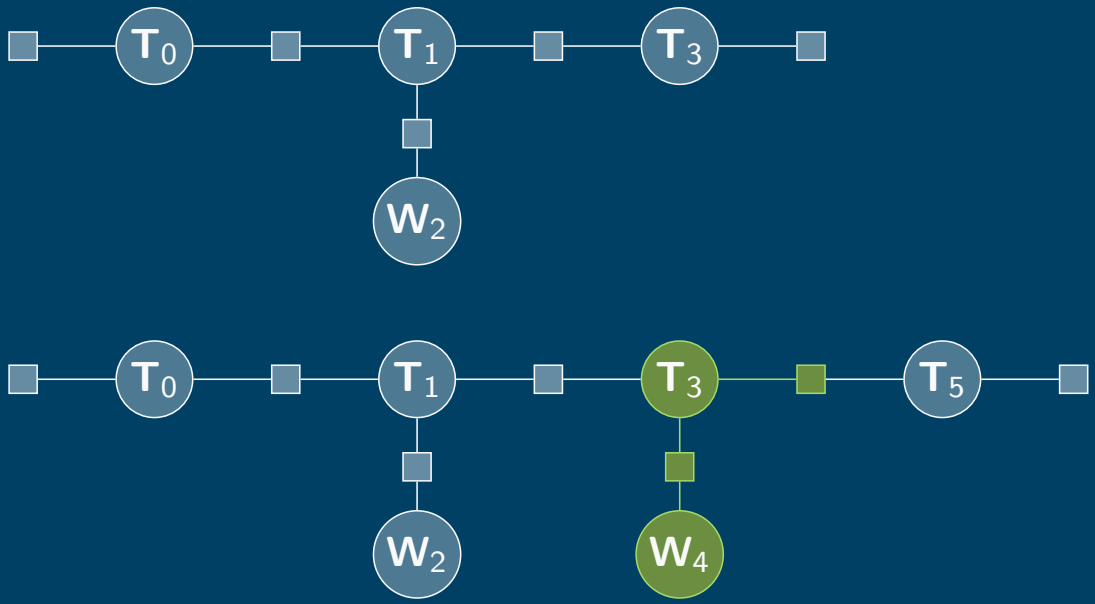
Limitations

Can't represent HMMs:



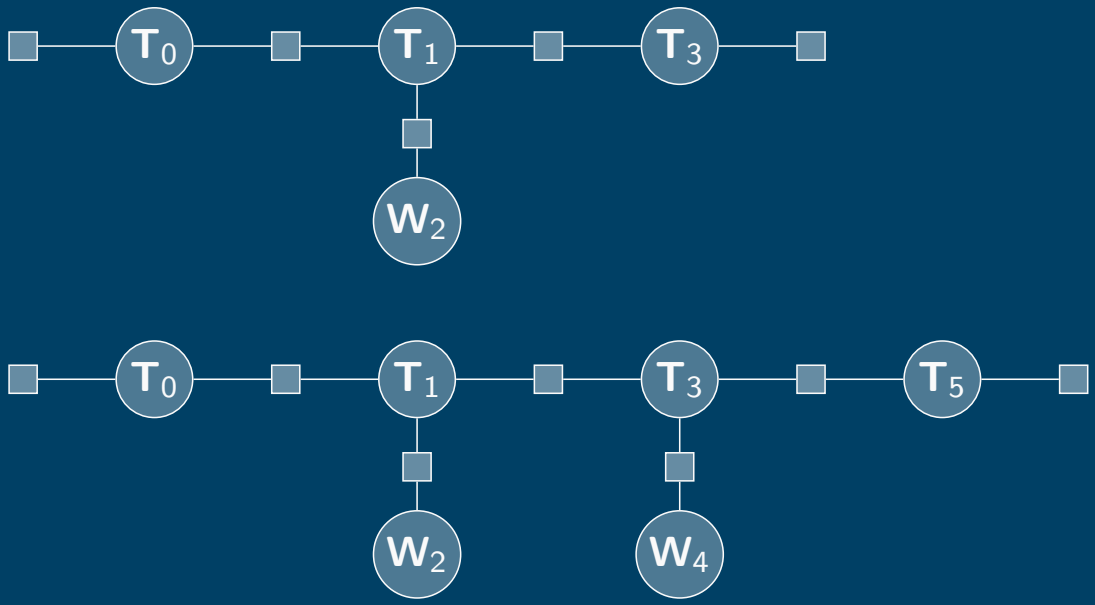
Limitations

Can't represent HMMs:



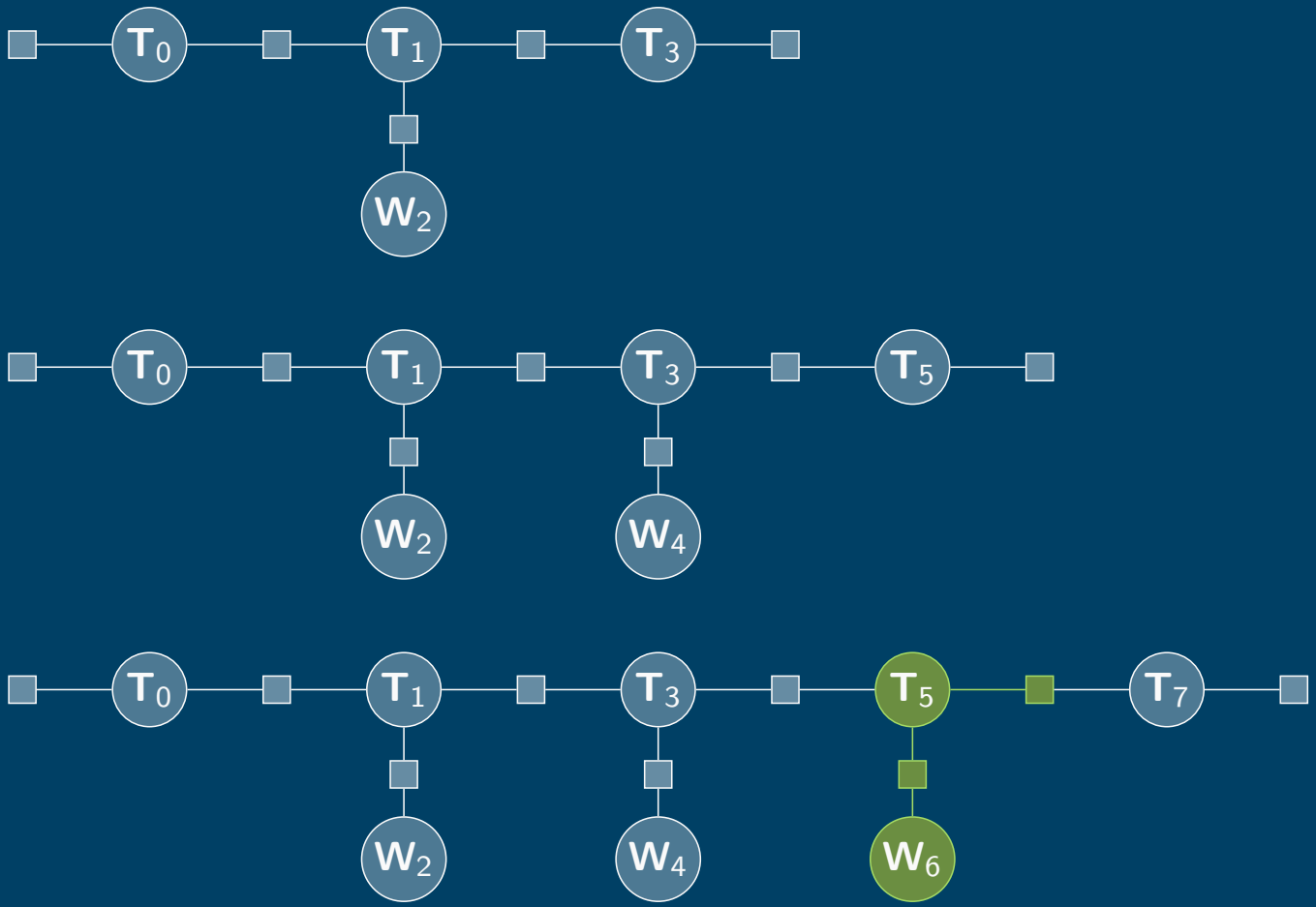
Limitations

Can't represent HMMs:



Limitations

Can't represent HMMs:

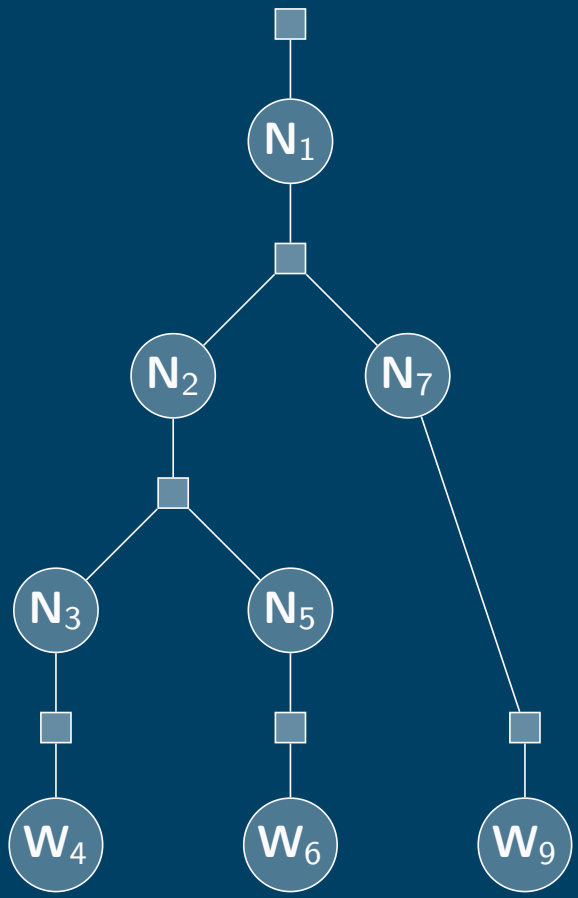


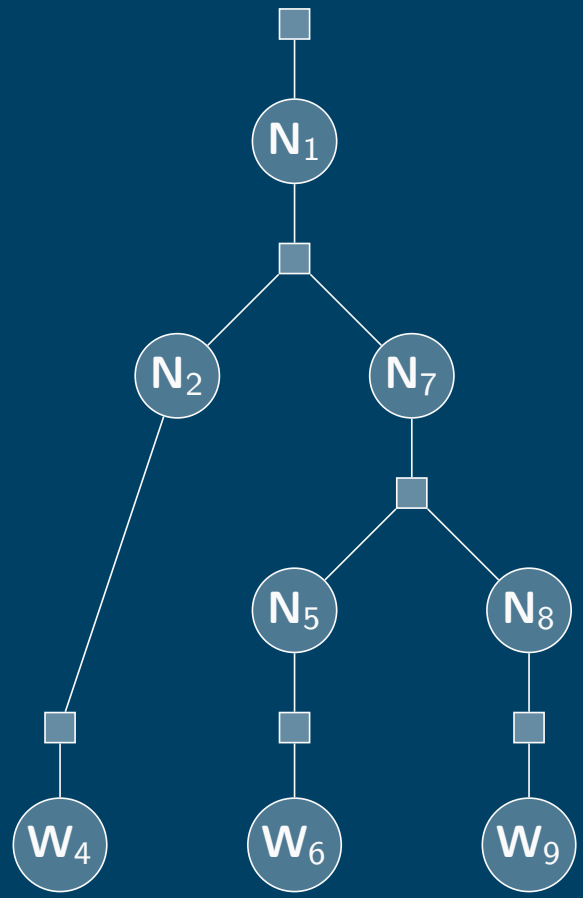
Limitations

Can't represent PCFGs:

Limitations

Can't represent PCFGs:





Solution

Use hyperedge replacement grammars

(Bauderon and Courcelle, 1987; Habel and Kreowski, 1987)

to generate sets of factor graphs

Hyperedge Replacement Grammars

Review: Context-Free (String) Grammars

$$S \to TC$$
$$T \to aTb$$
$$T \to \epsilon$$
$$C \to cC$$
$$C \to \epsilon$$

generates the language
$\{a^n b^n c^m | n, m \geq 0\}$

S

TC

aTbC

aaTbbC

aaaTbbbC

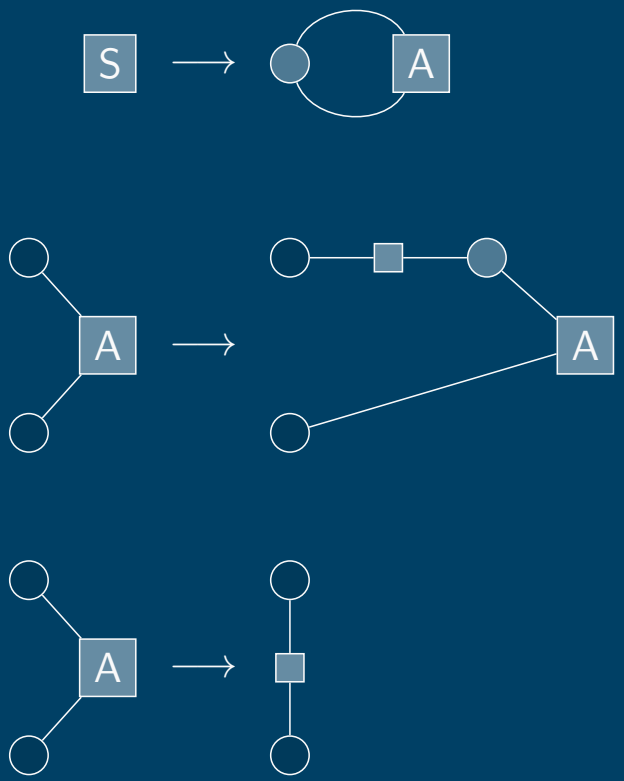
aaabbbC

aaabbbcC

aaabbbccC

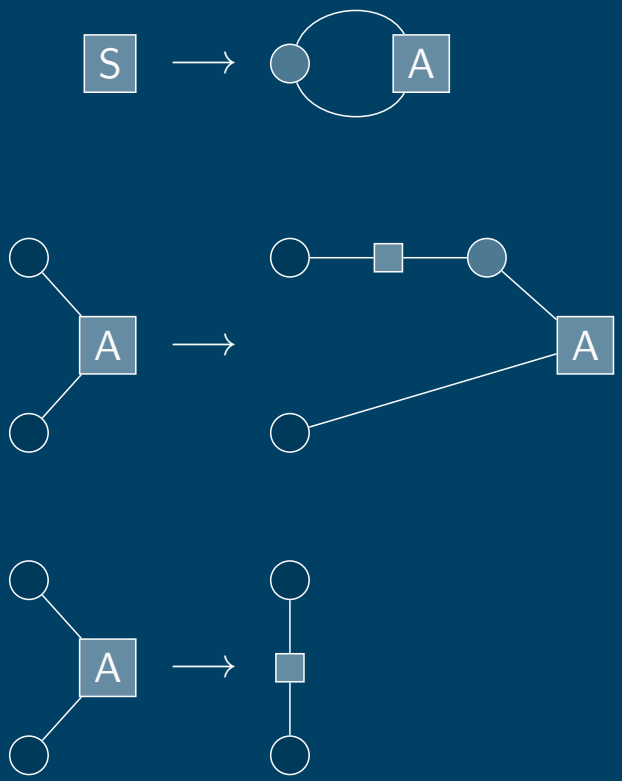
aaabbbcc

Hyperedge Replacement Grammars: Example



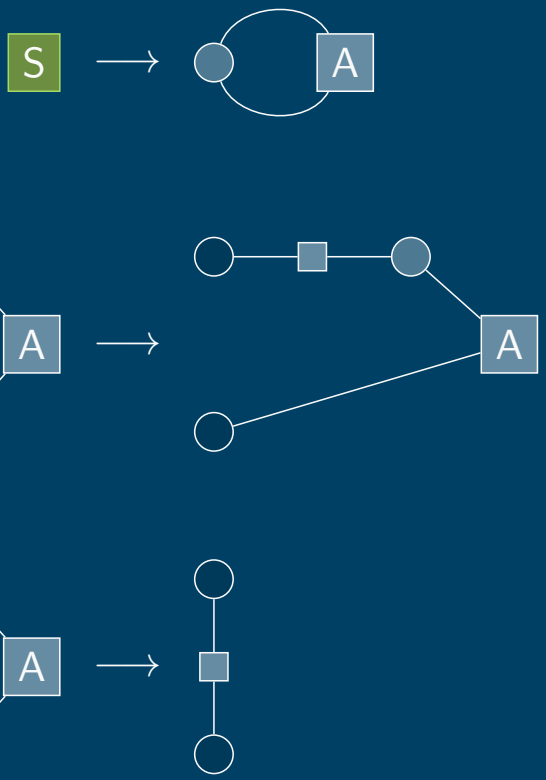
generates the language
of ring graphs

Hyperedge Replacement Grammars: Example



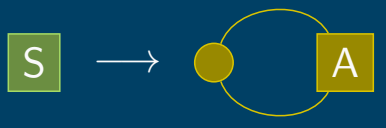
generates the language
of ring graphs

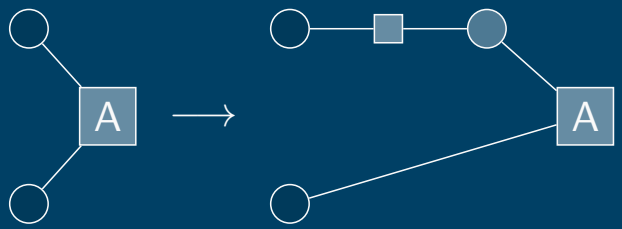
Hyperedge Replacement Grammars: Example

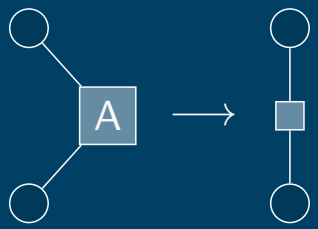


generates the language
of ring graphs

Hyperedge Replacement Grammars: Example

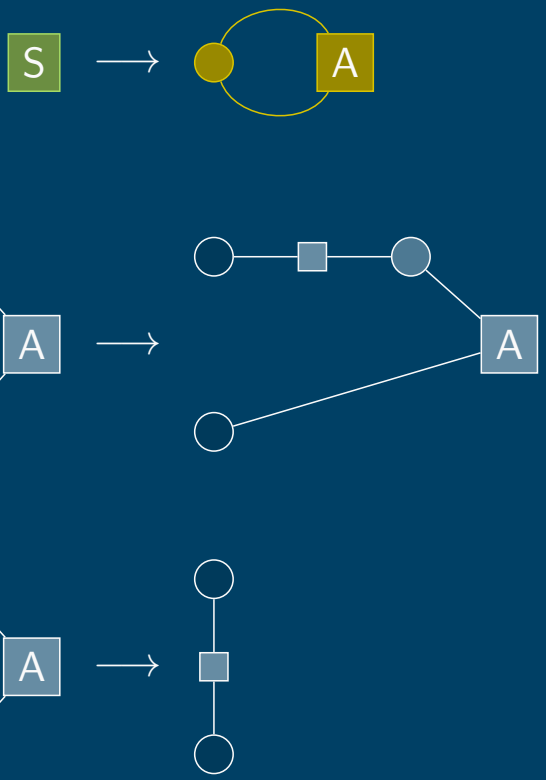






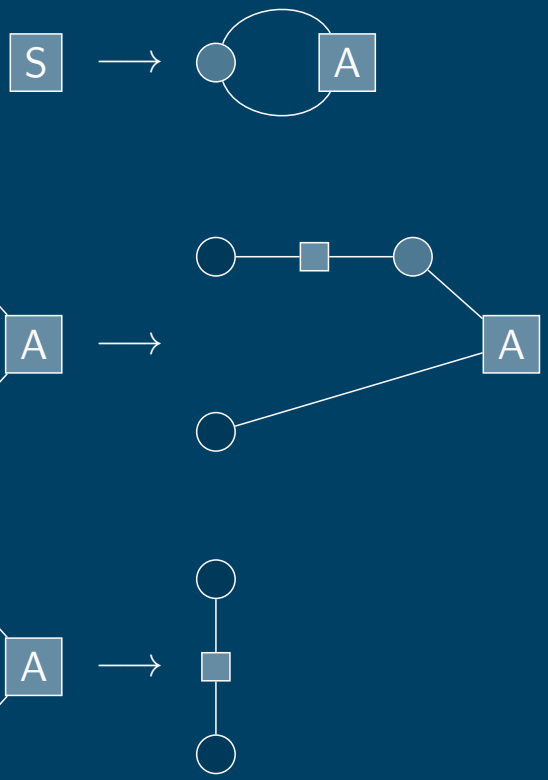
generates the language
of ring graphs

Hyperedge Replacement Grammars: Example



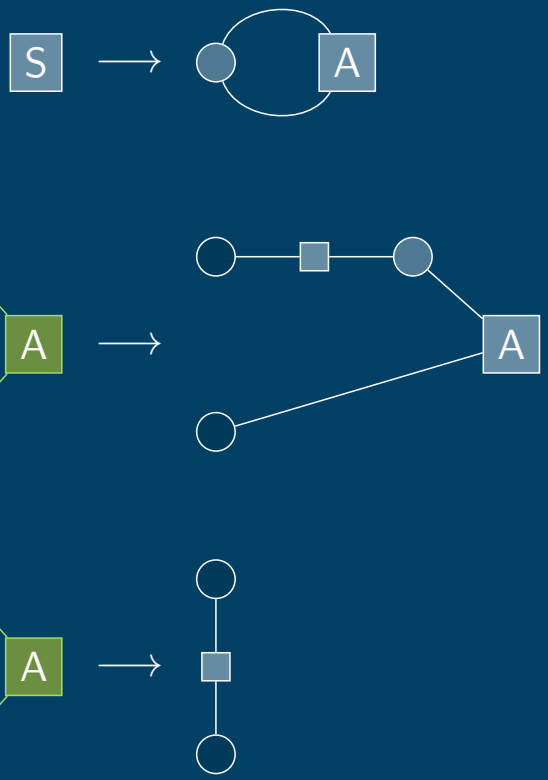
generates the language
of ring graphs

Hyperedge Replacement Grammars: Example



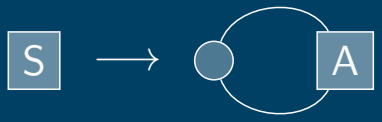
generates the language
of ring graphs

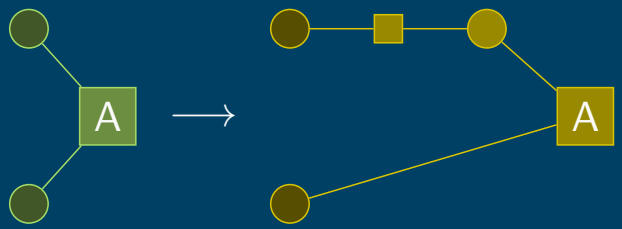
Hyperedge Replacement Grammars: Example

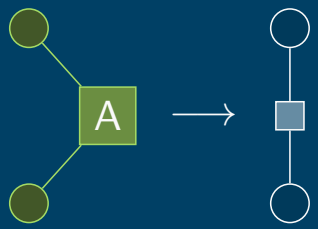


generates the language
of ring graphs

Hyperedge Replacement Grammars: Example

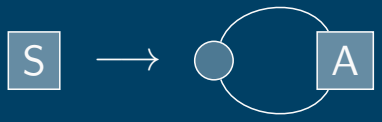


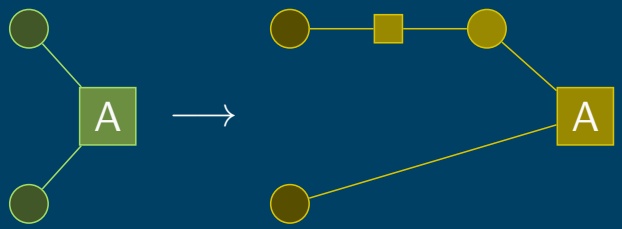


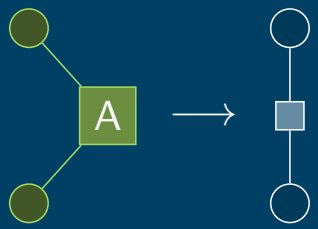


generates the language
of ring graphs

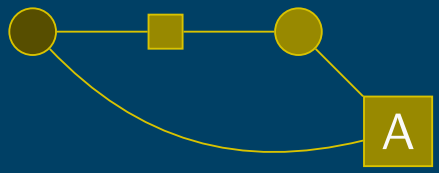
Hyperedge Replacement Grammars: Example



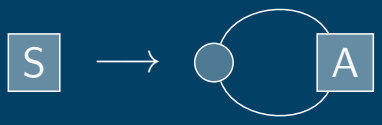


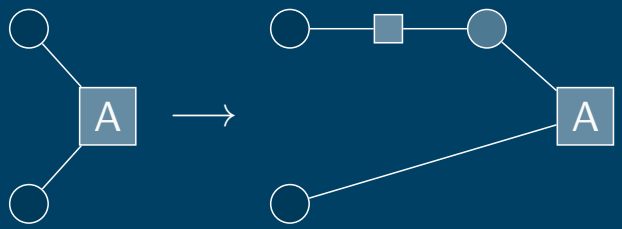


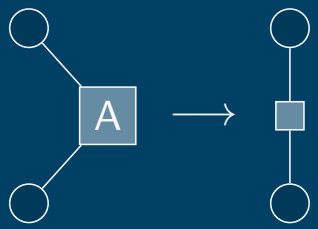
generates the language
of ring graphs



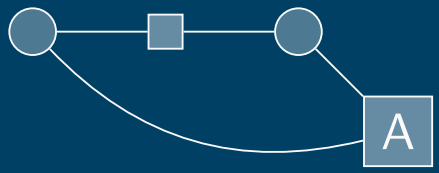
Hyperedge Replacement Grammars: Example



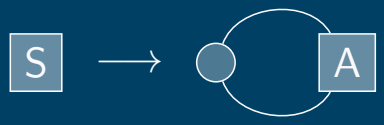


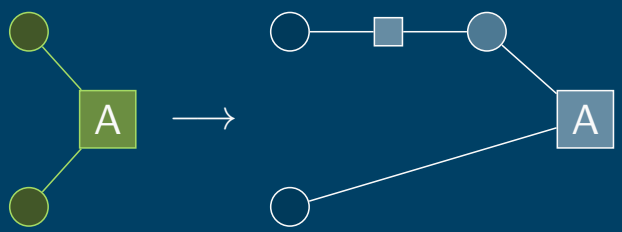


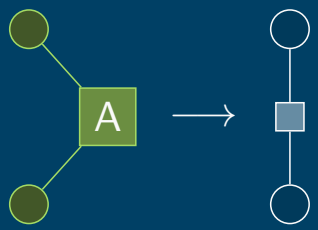
generates the language
of ring graphs



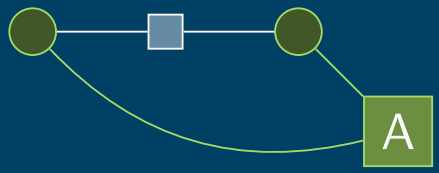
Hyperedge Replacement Grammars: Example



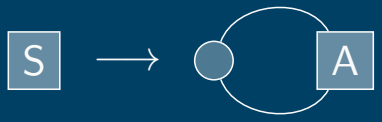


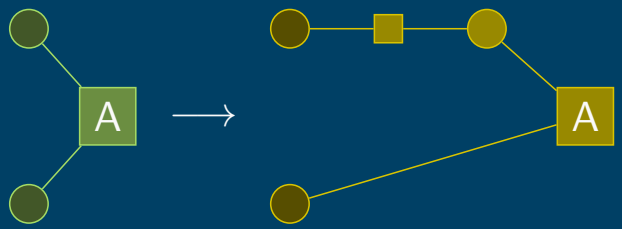


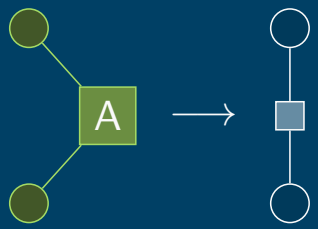
generates the language
of ring graphs



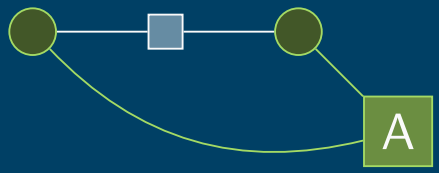
Hyperedge Replacement Grammars: Example



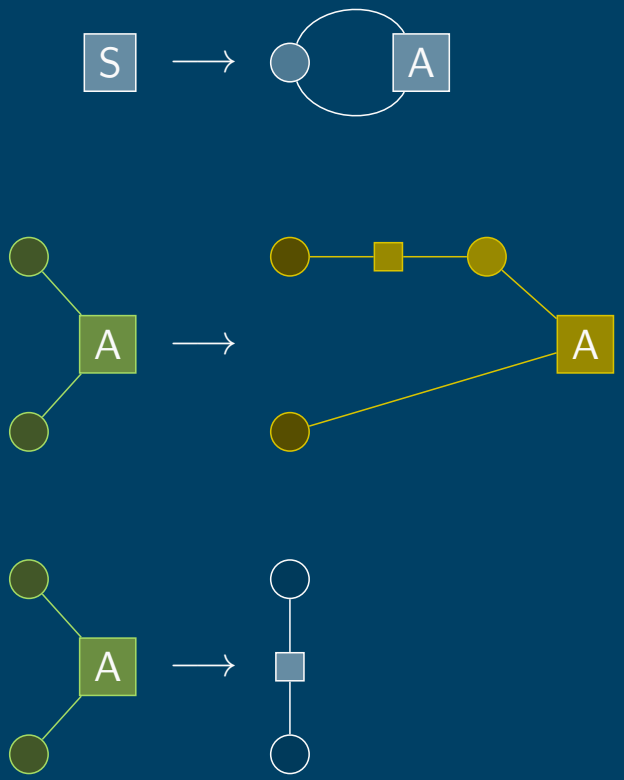




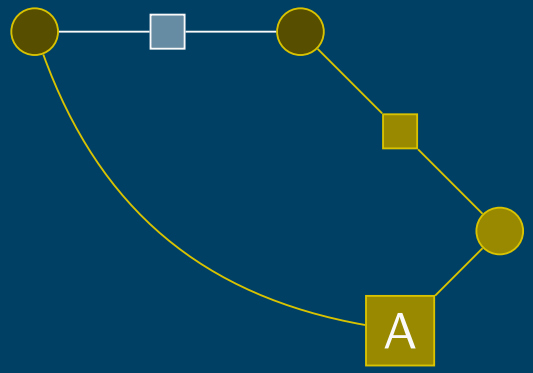
generates the language
of ring graphs



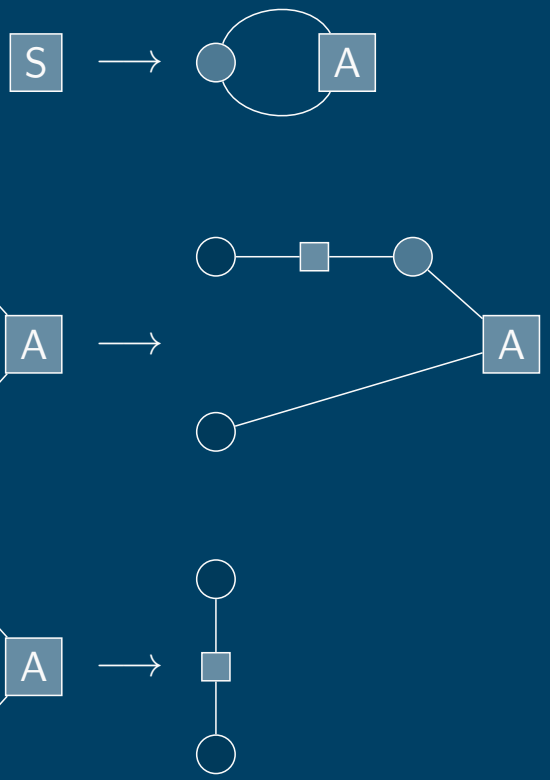
Hyperedge Replacement Grammars: Example



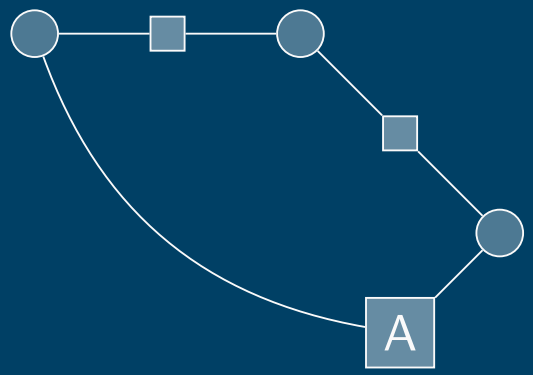
generates the language
of ring graphs



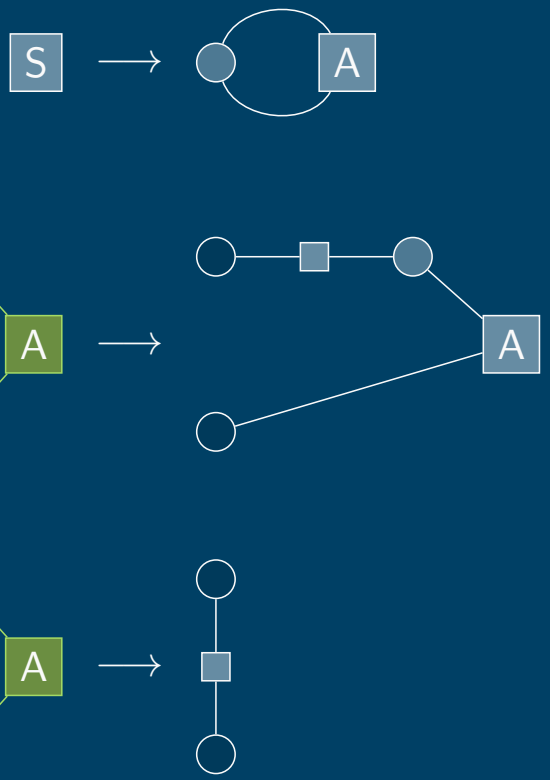
Hyperedge Replacement Grammars: Example



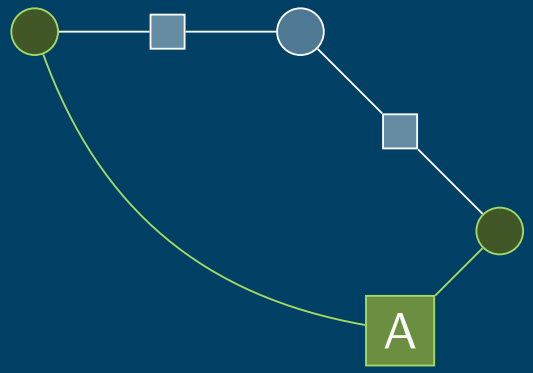
generates the language
of ring graphs



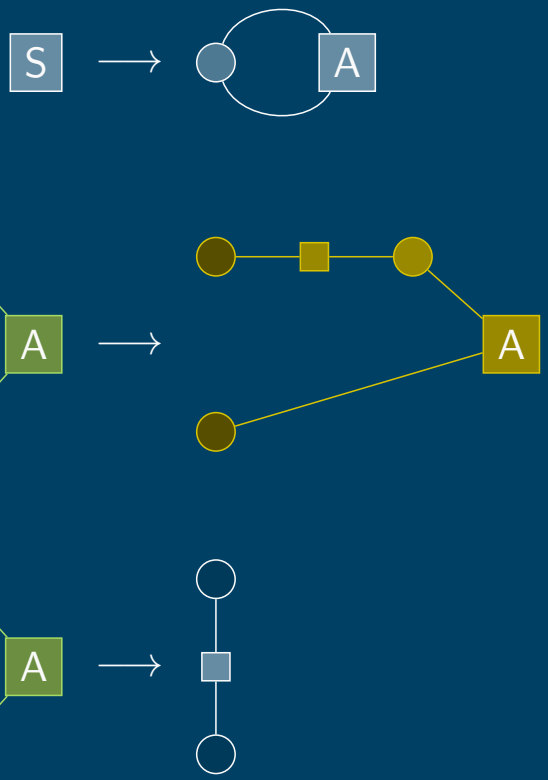
Hyperedge Replacement Grammars: Example



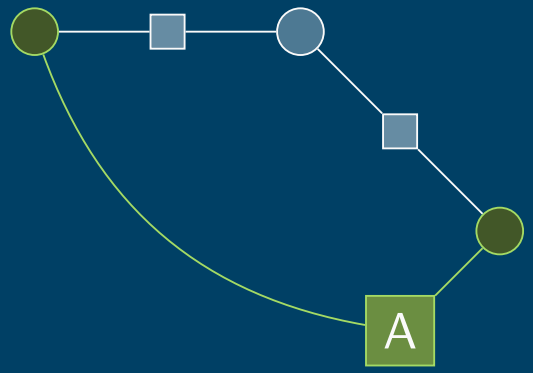
generates the language
of ring graphs



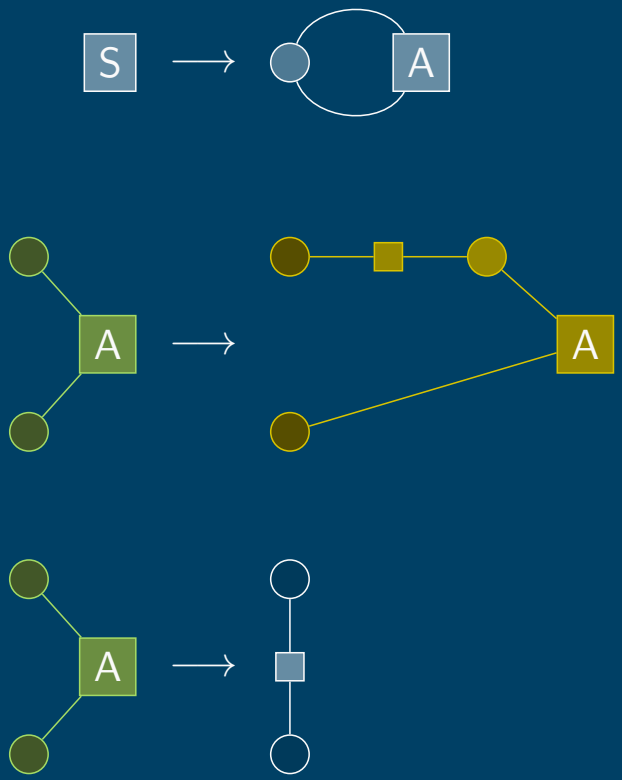
Hyperedge Replacement Grammars: Example



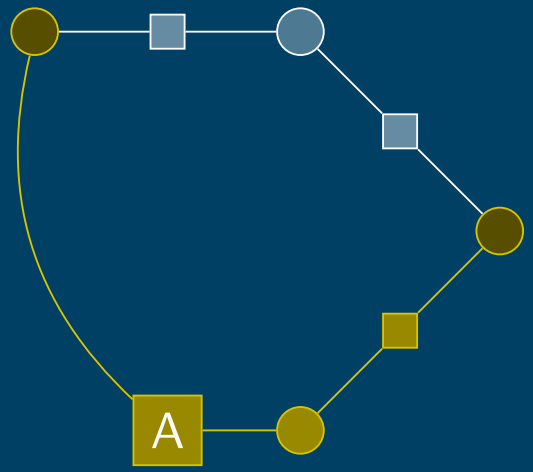
generates the language
of ring graphs



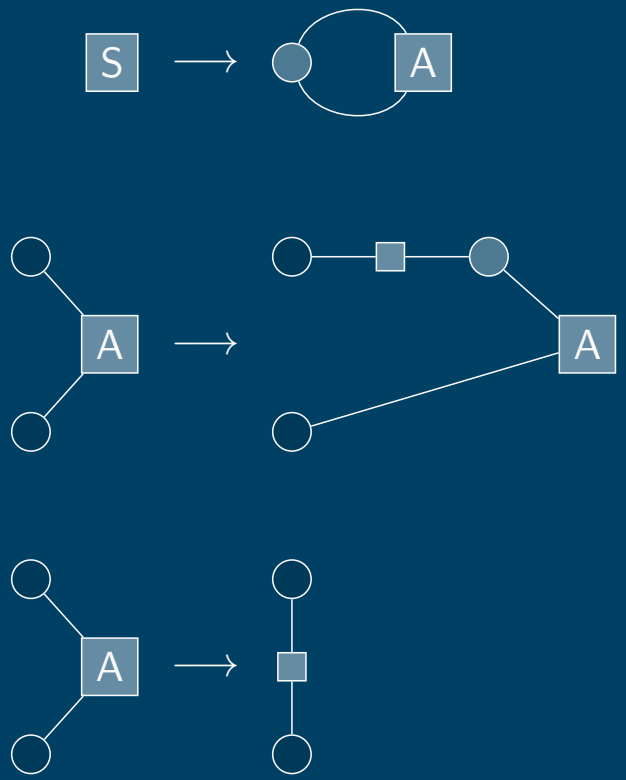
Hyperedge Replacement Grammars: Example



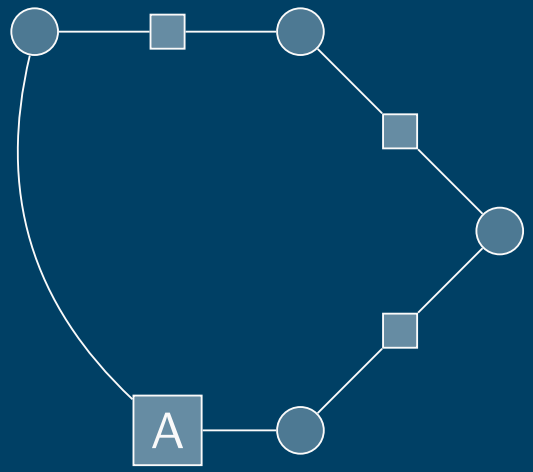
generates the language
of ring graphs



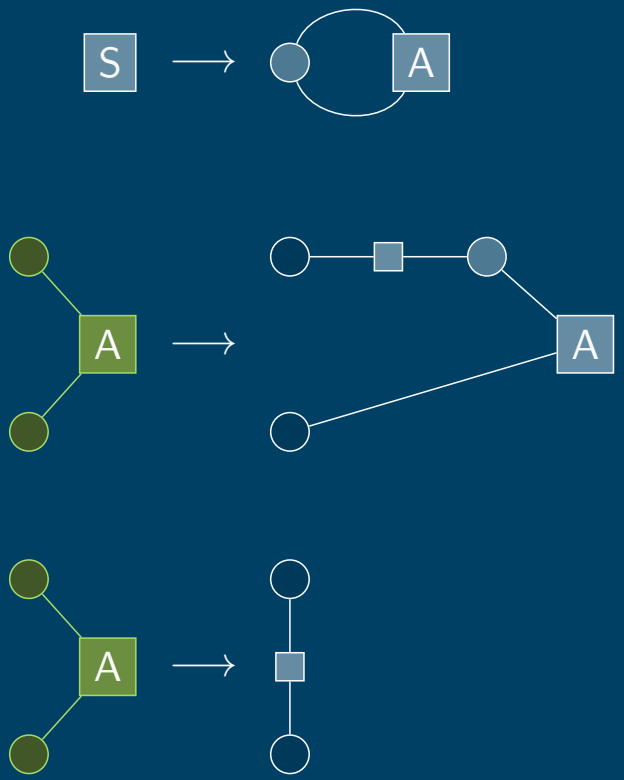
Hyperedge Replacement Grammars: Example



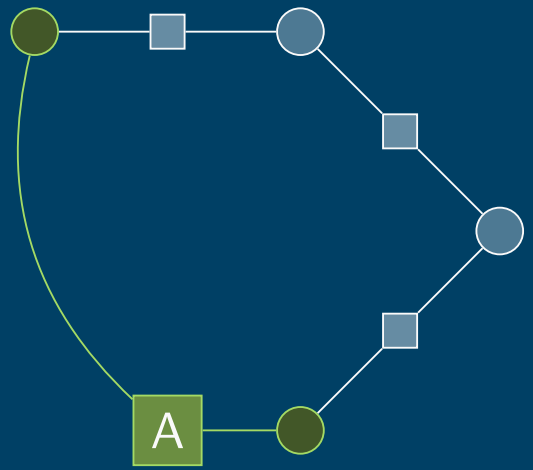
generates the language
of ring graphs



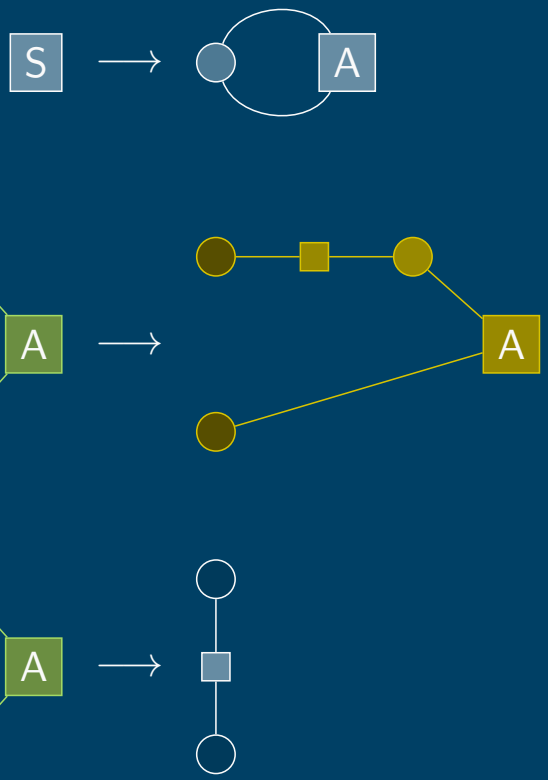
Hyperedge Replacement Grammars: Example



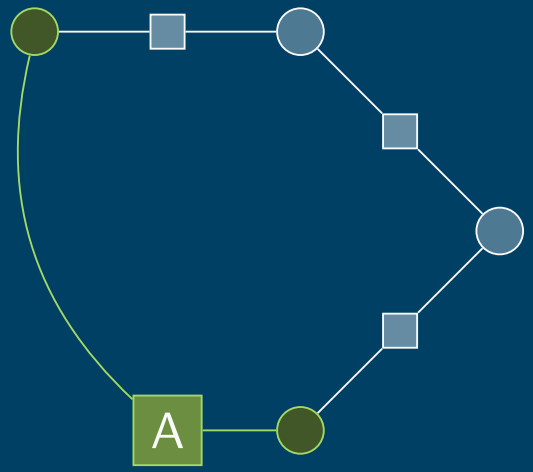
generates the language
of ring graphs



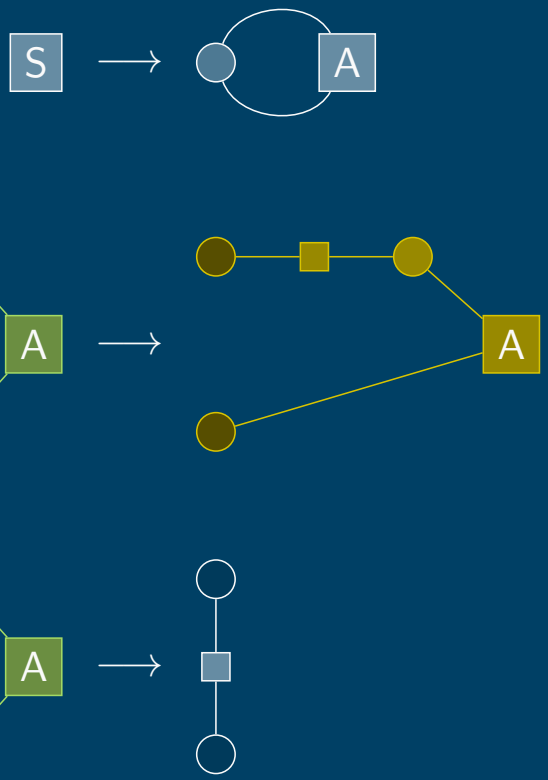
Hyperedge Replacement Grammars: Example



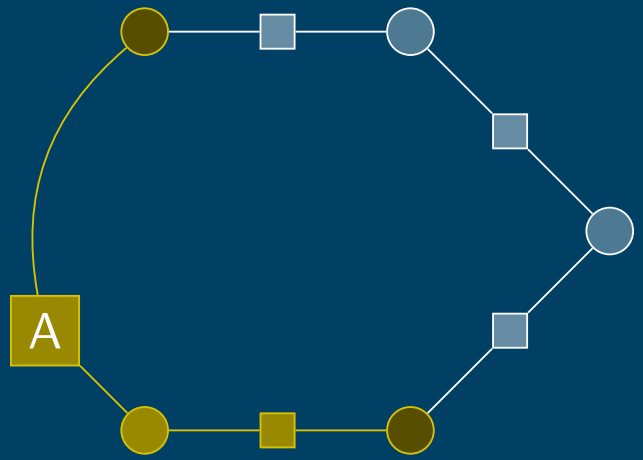
generates the language
of ring graphs



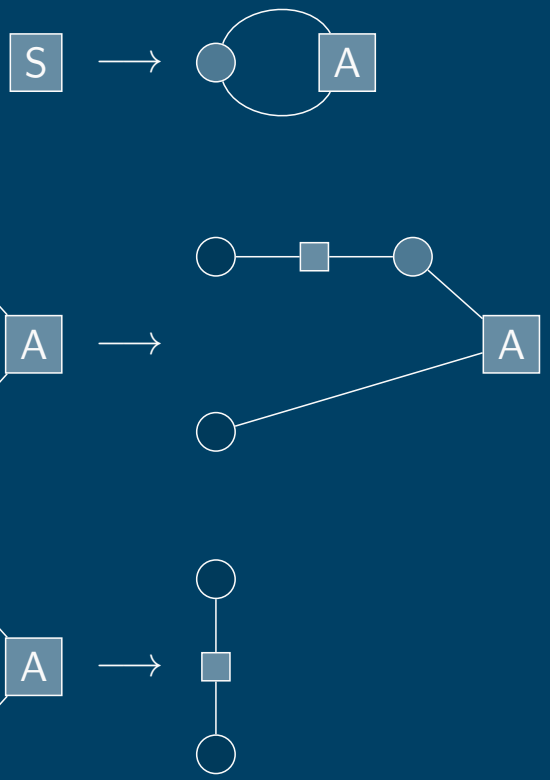
Hyperedge Replacement Grammars: Example



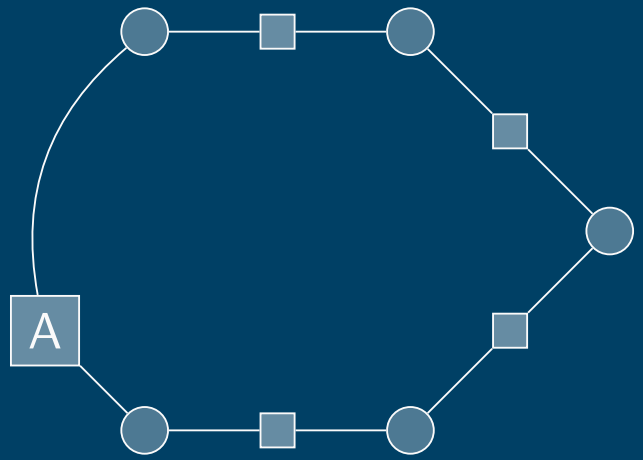
generates the language
of ring graphs



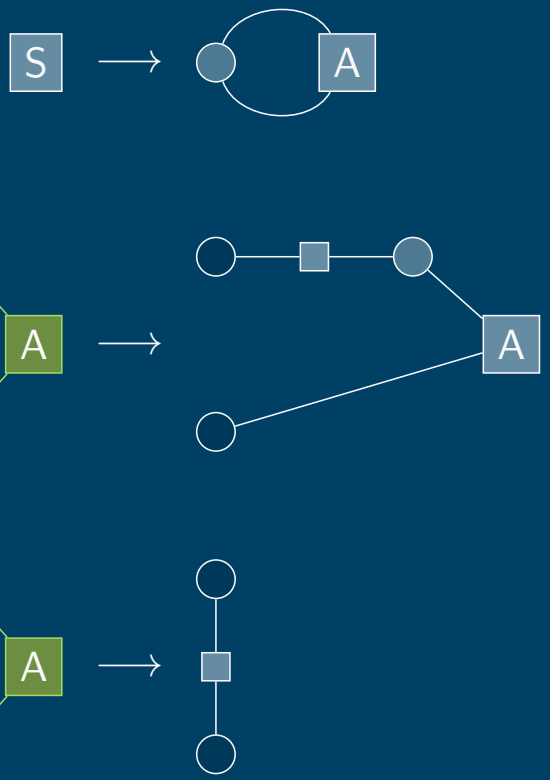
Hyperedge Replacement Grammars: Example



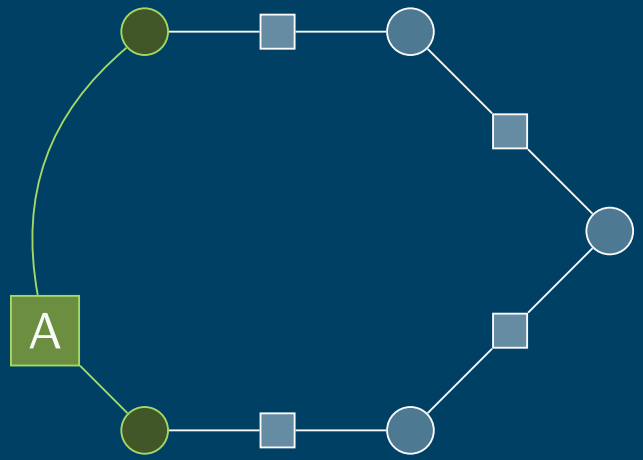
generates the language
of ring graphs



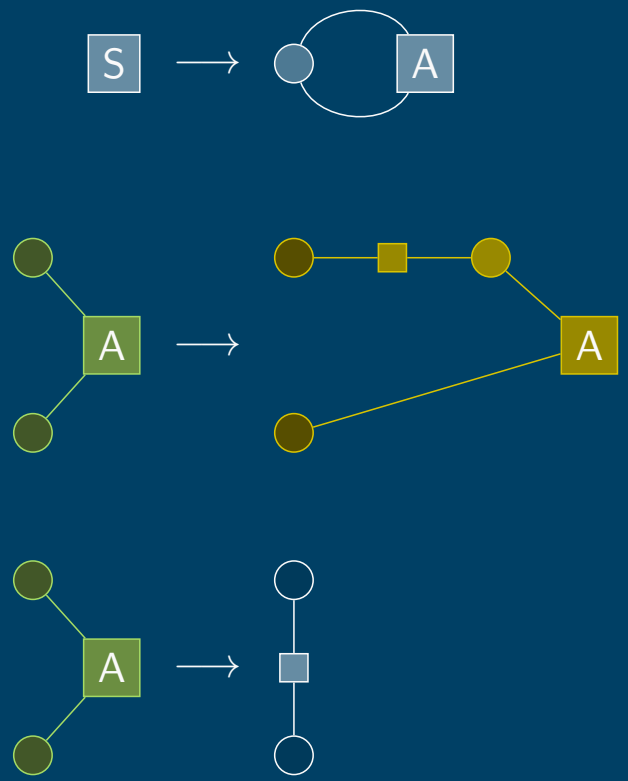
Hyperedge Replacement Grammars: Example



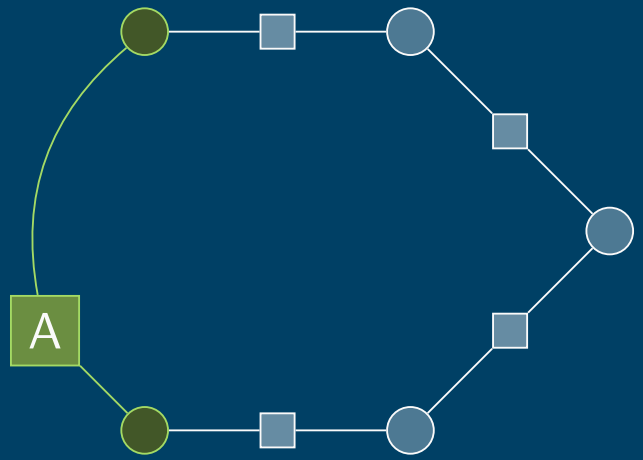
generates the language
of ring graphs



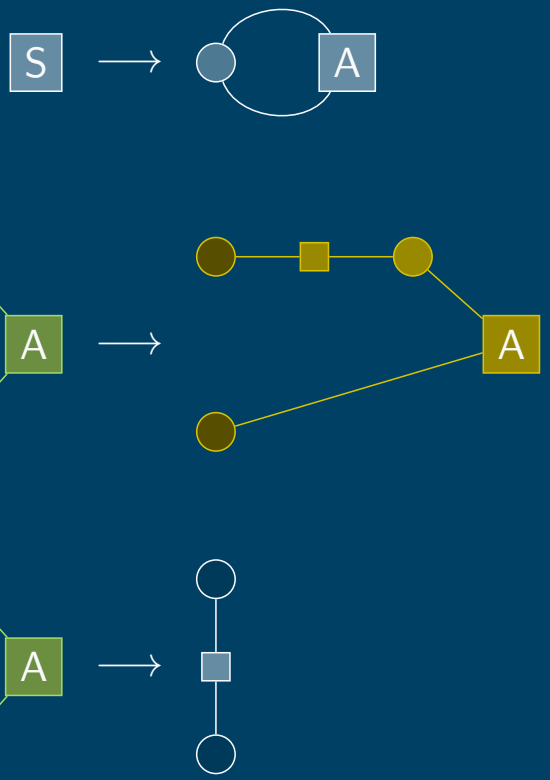
Hyperedge Replacement Grammars: Example



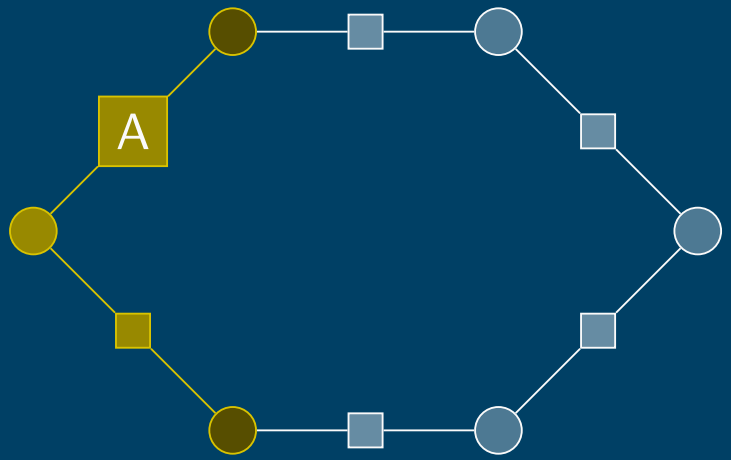
generates the language
of ring graphs



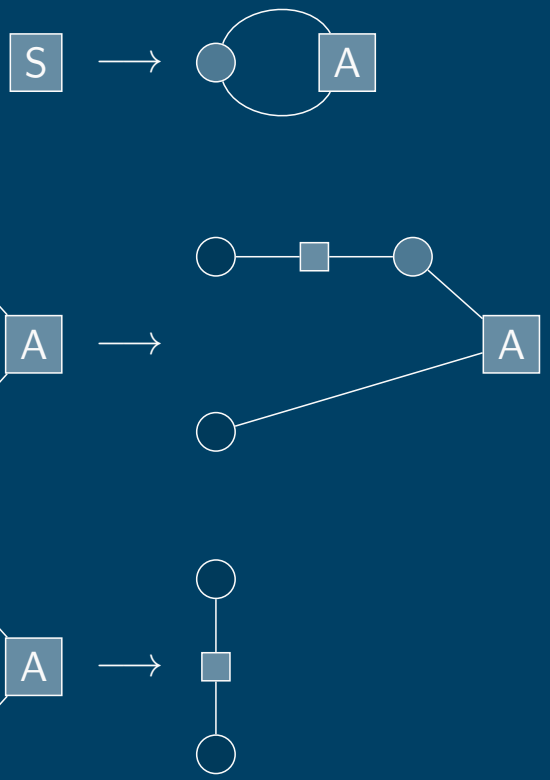
Hyperedge Replacement Grammars: Example



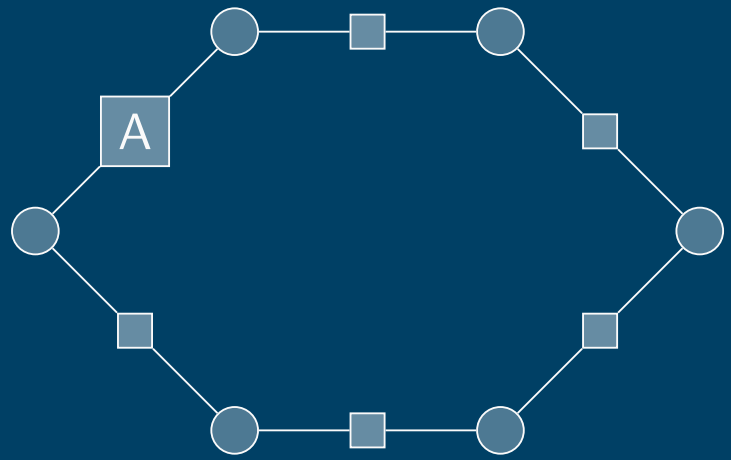
generates the language
of ring graphs



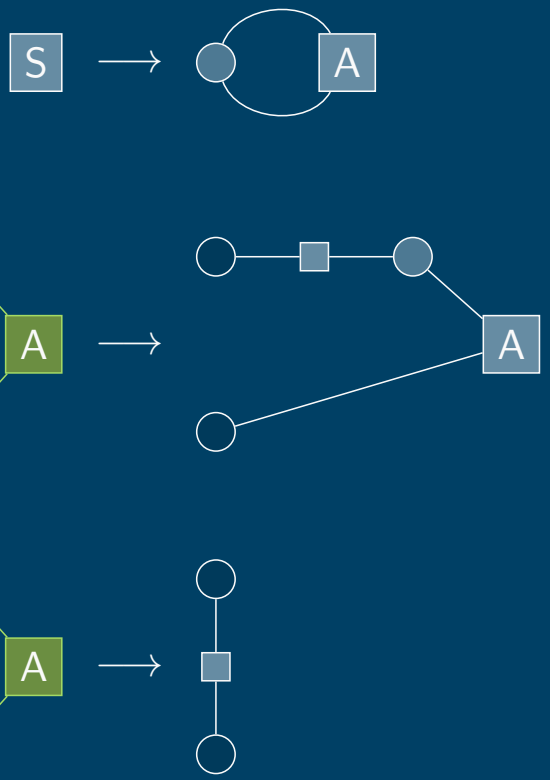
Hyperedge Replacement Grammars: Example



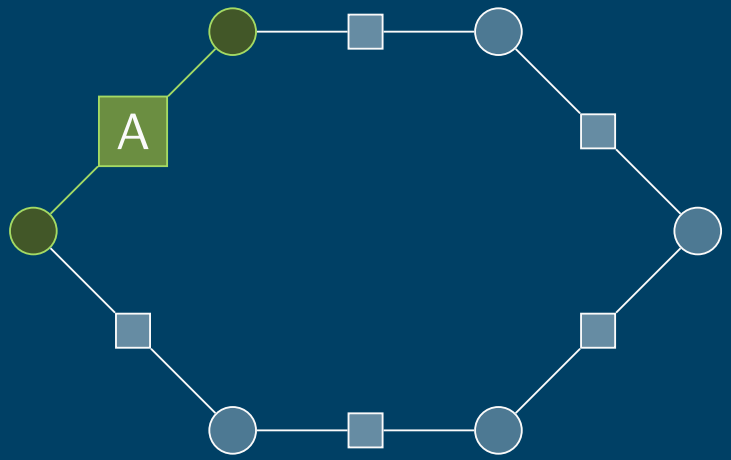
generates the language
of ring graphs



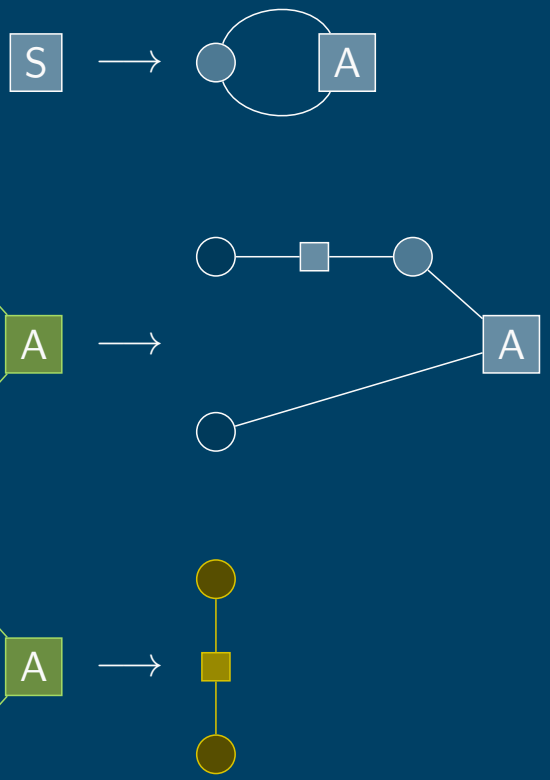
Hyperedge Replacement Grammars: Example



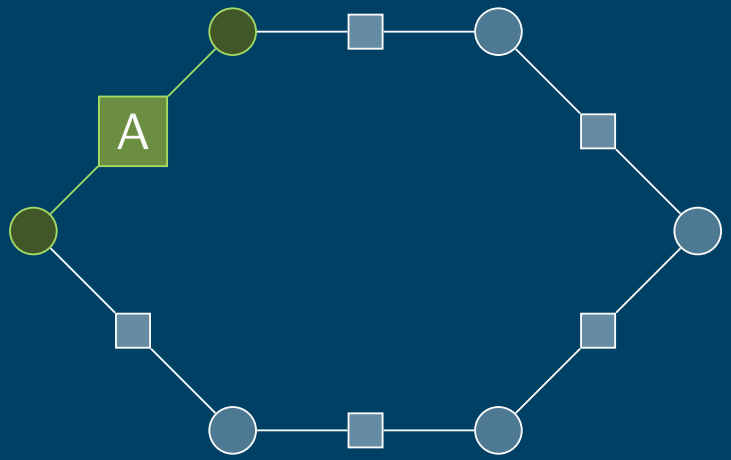
generates the language
of ring graphs



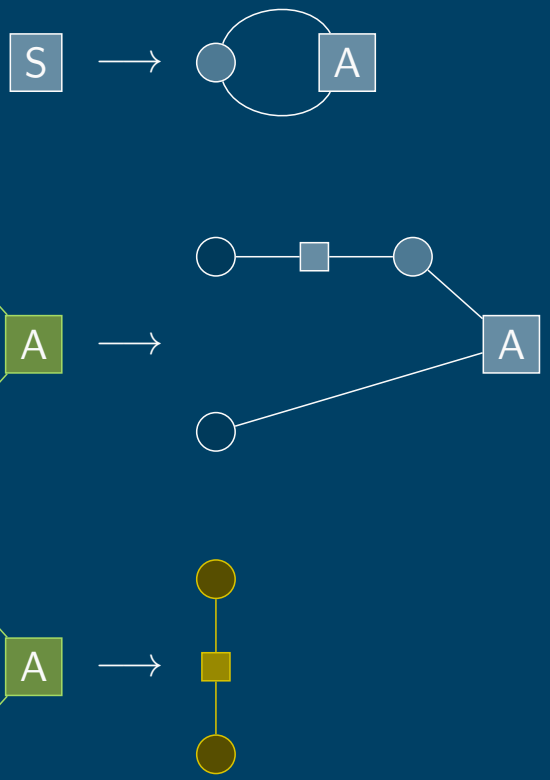
Hyperedge Replacement Grammars: Example



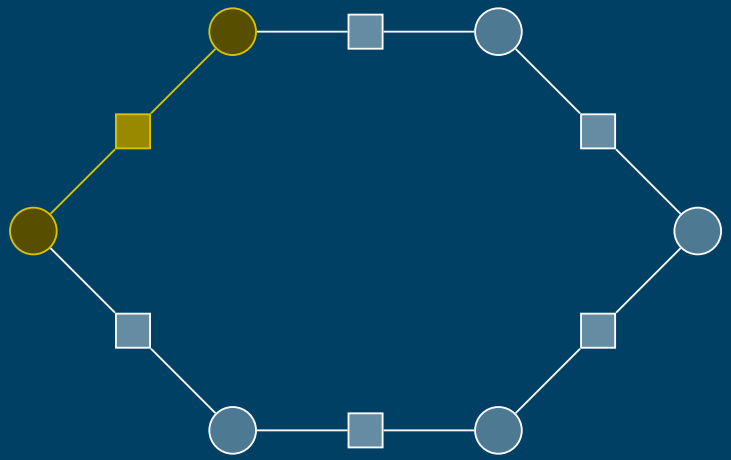
generates the language
of ring graphs



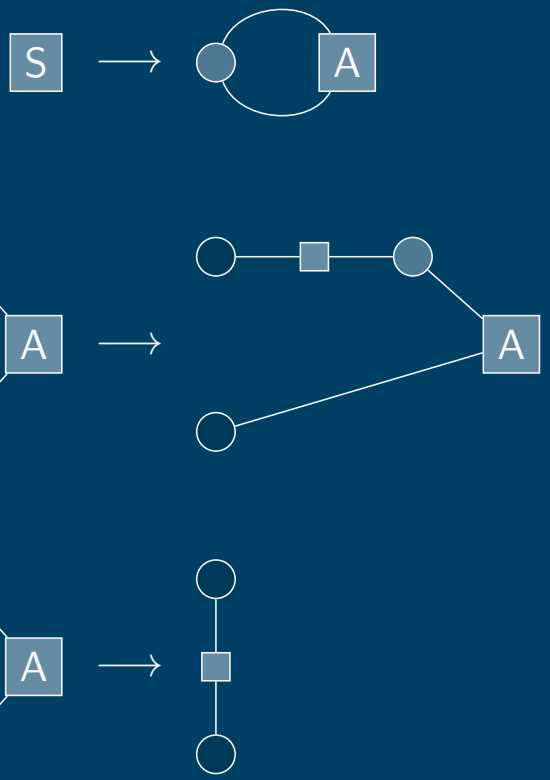
Hyperedge Replacement Grammars: Example



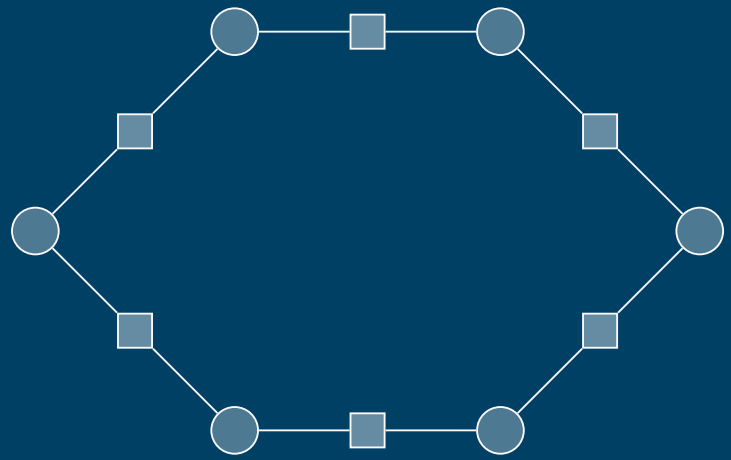
generates the language
of ring graphs



Hyperedge Replacement Grammars: Example



generates the language
of ring graphs



Factor Graph Grammars

Factor Graph Grammars

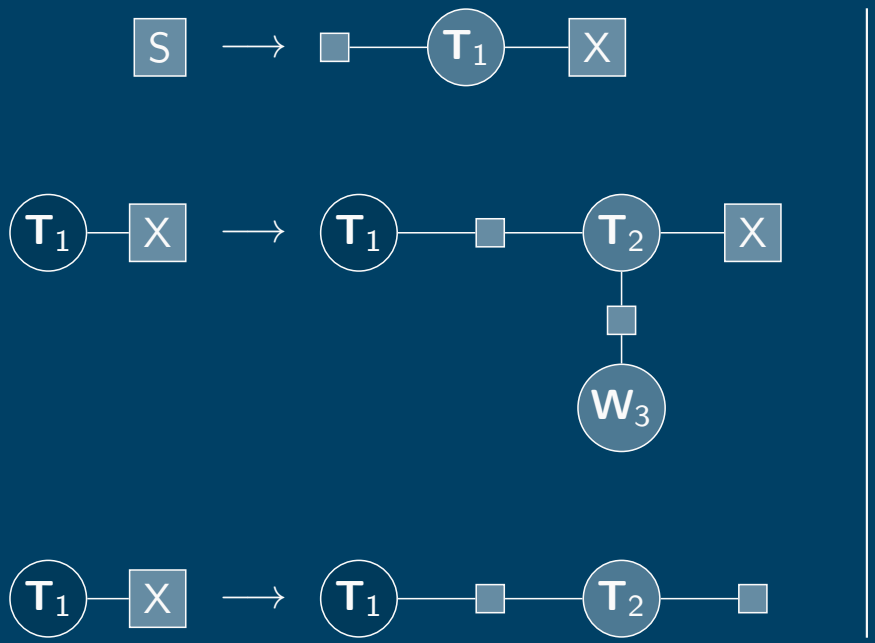
A factor graph grammar

is a hyperedge replacement grammar

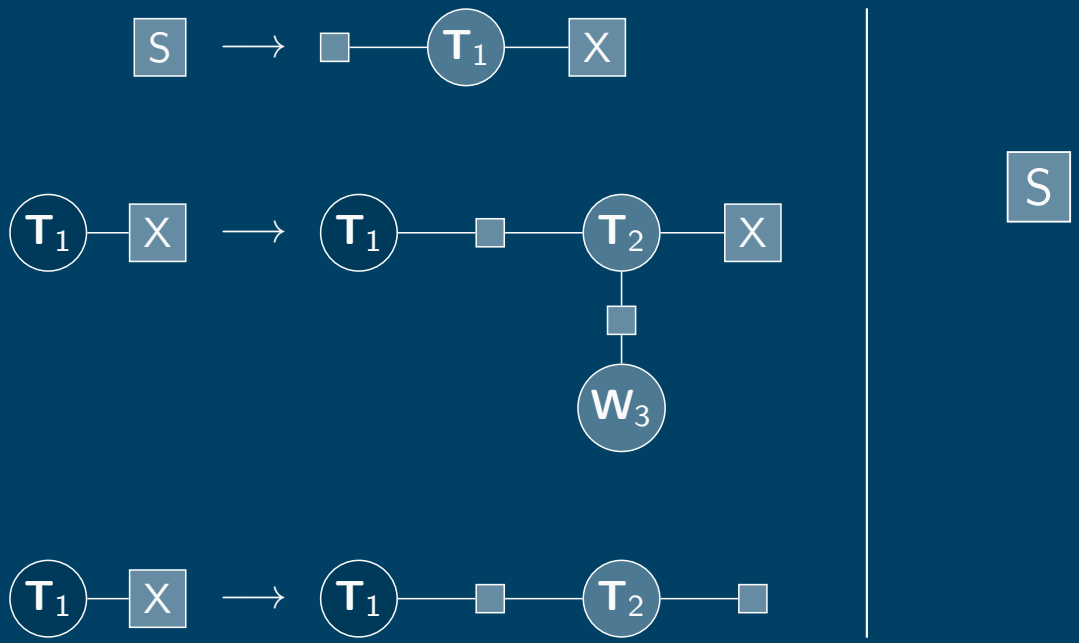
that generates factor graphs

A Factor Graph Grammar for HMMs

A Factor Graph Grammar for HMMs

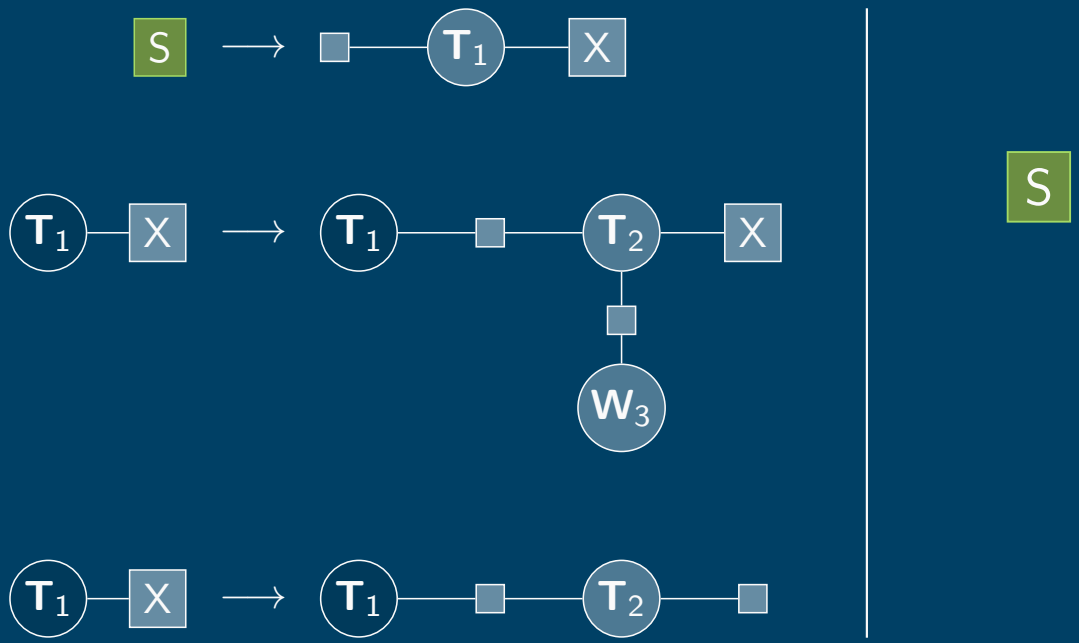


A Factor Graph Grammar for HMMs



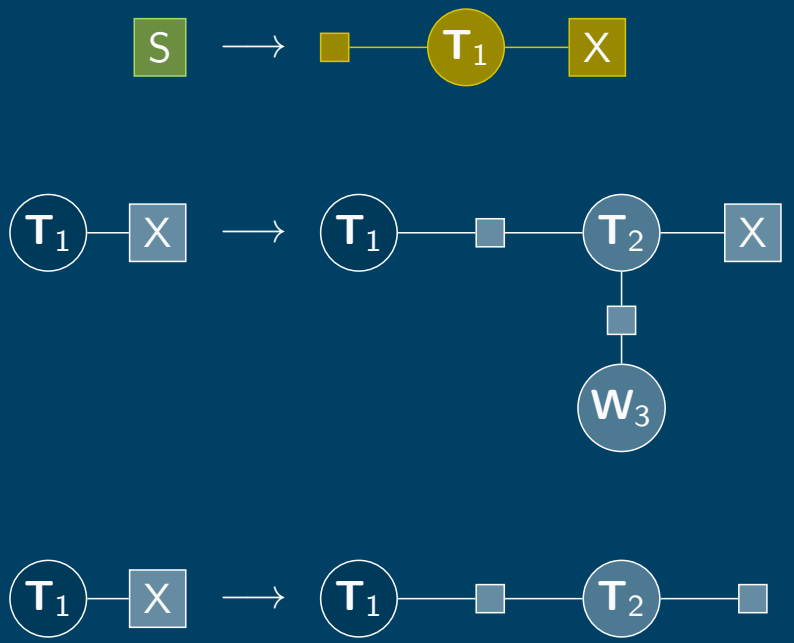
S

A Factor Graph Grammar for HMMs

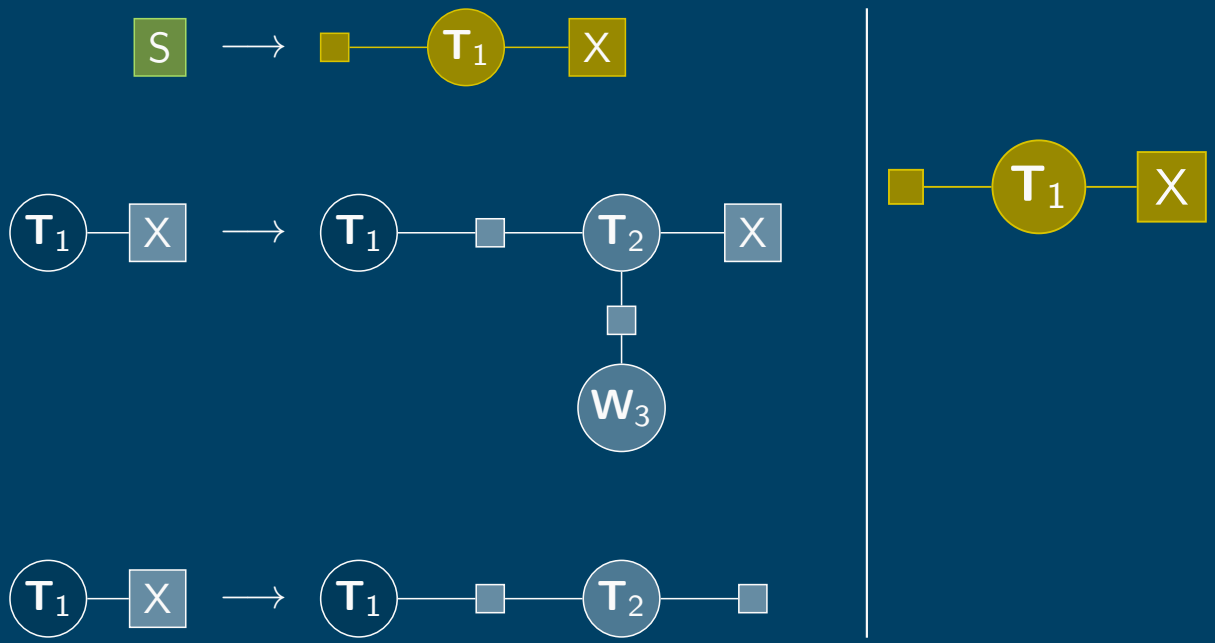


S

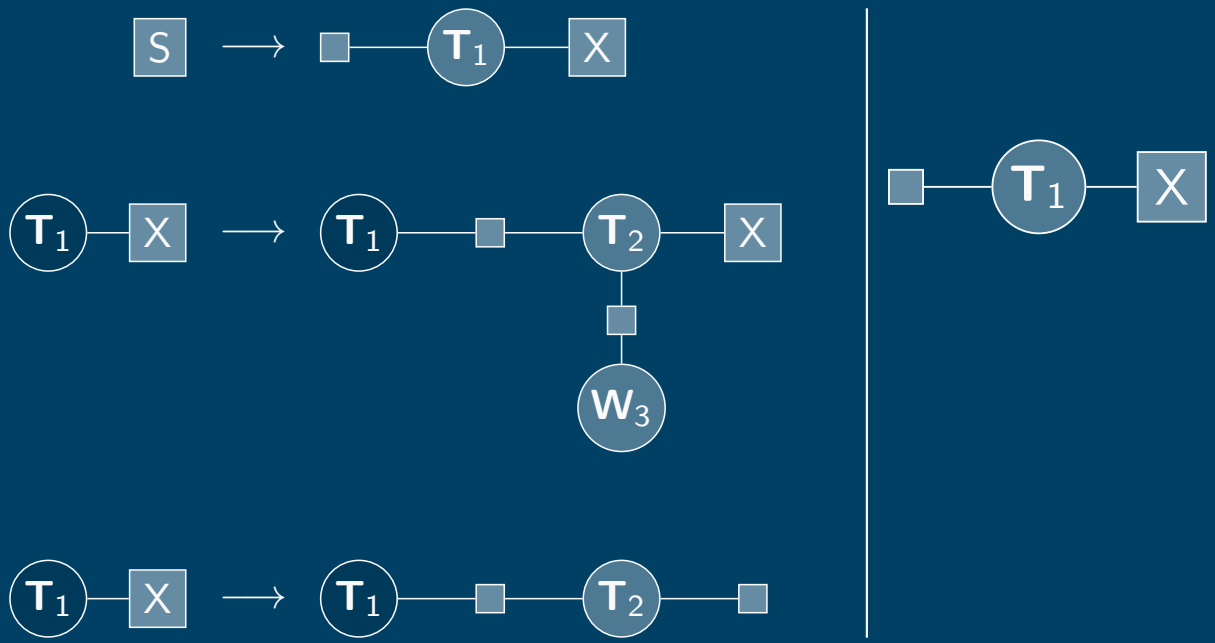
A Factor Graph Grammar for HMMs



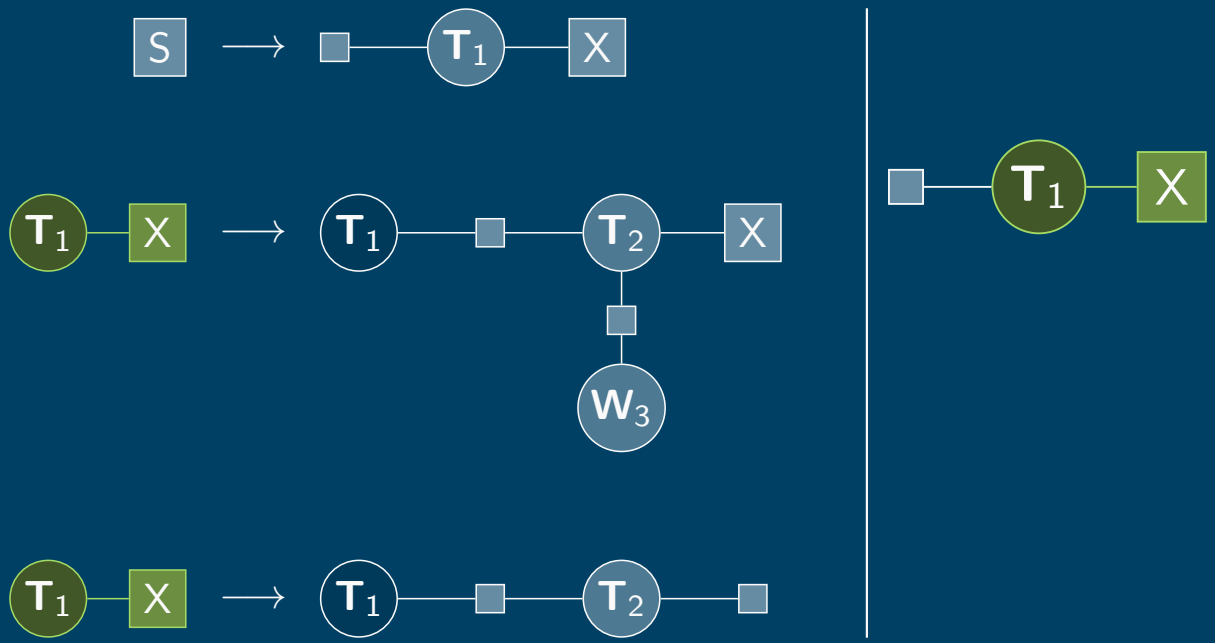
A Factor Graph Grammar for HMMs



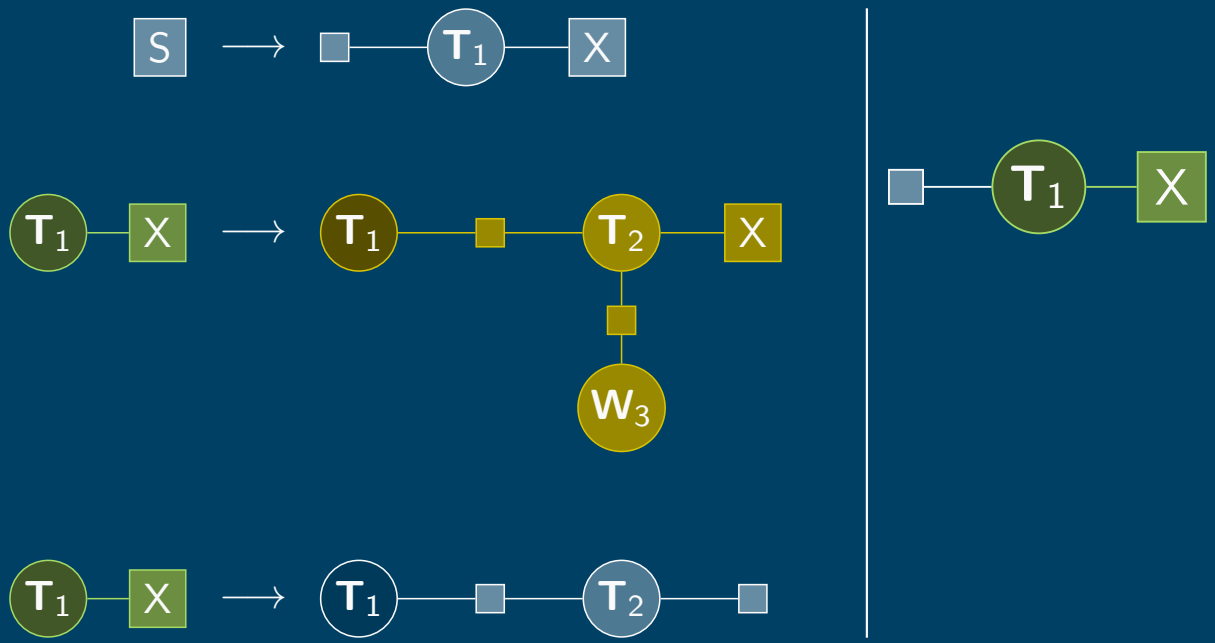
A Factor Graph Grammar for HMMs



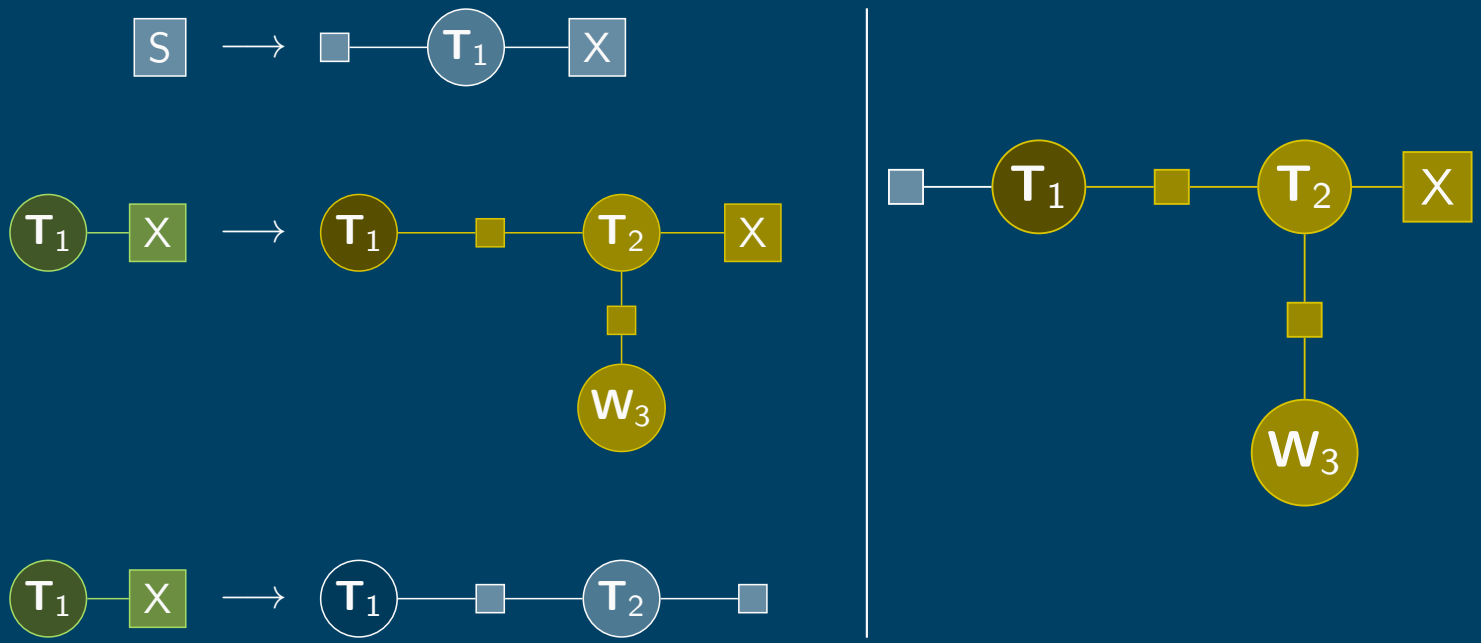
A Factor Graph Grammar for HMMs



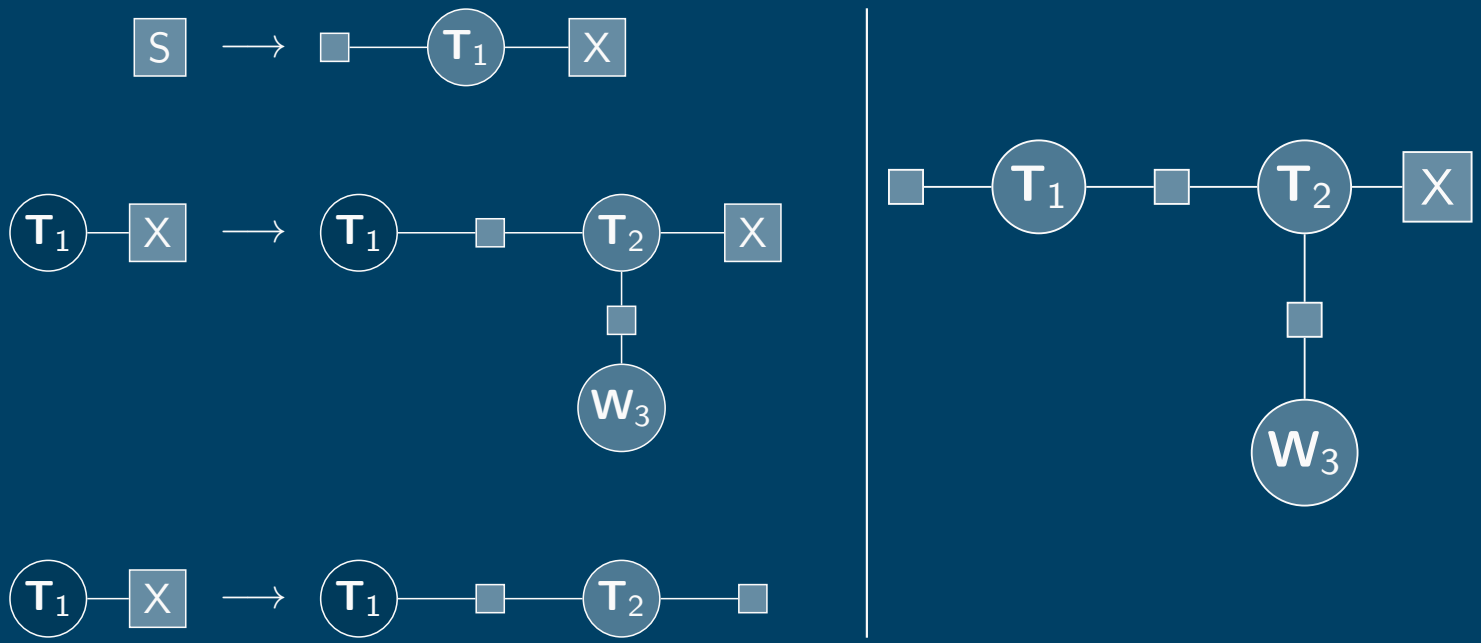
A Factor Graph Grammar for HMMs



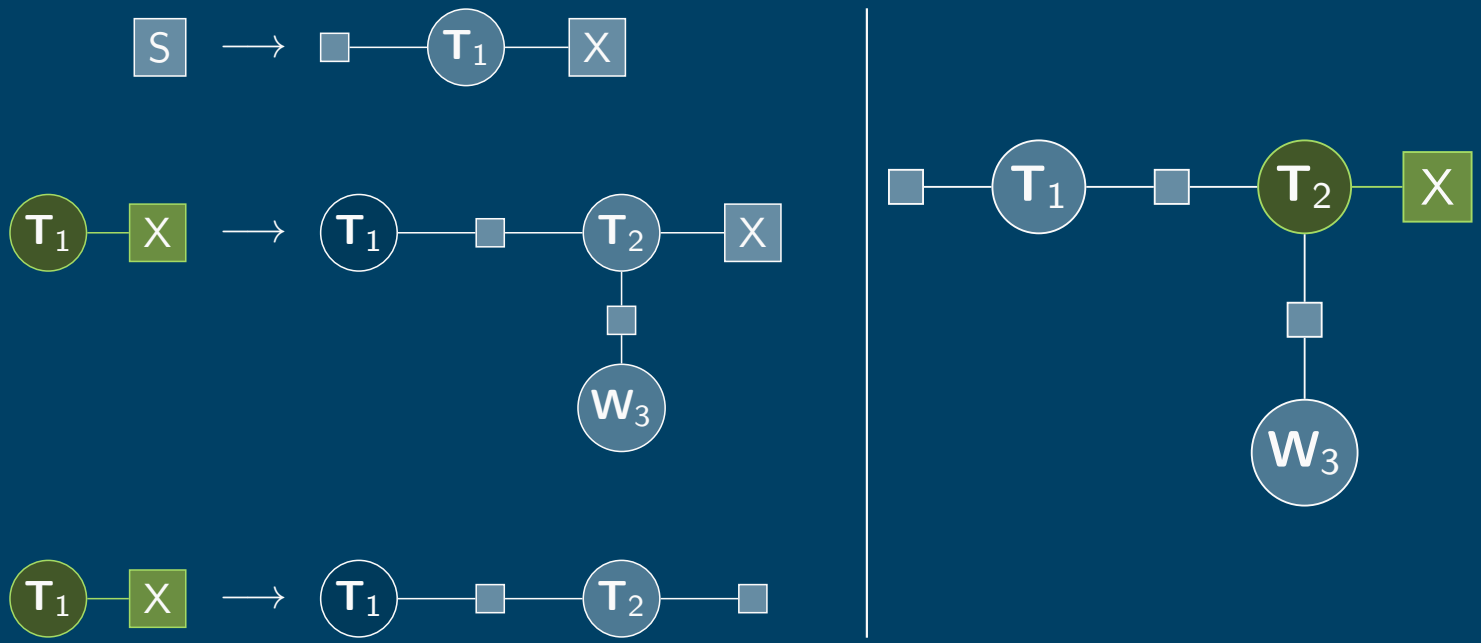
A Factor Graph Grammar for HMMs



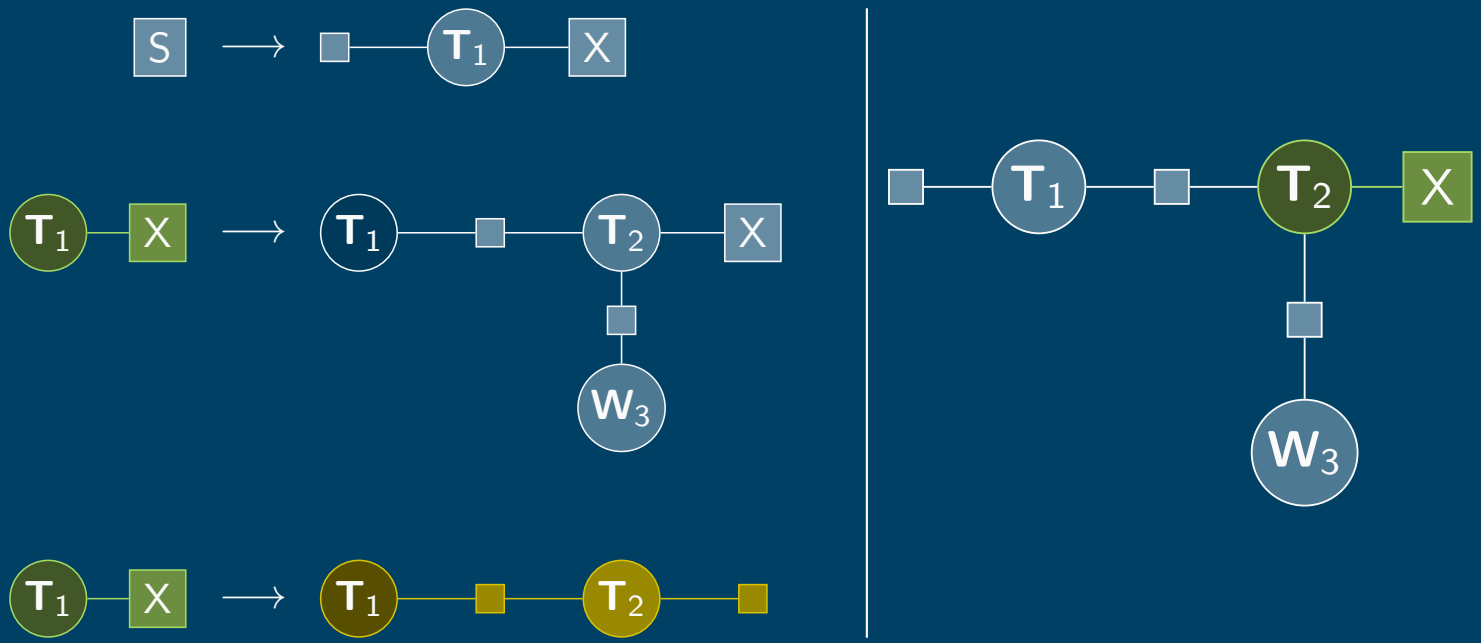
A Factor Graph Grammar for HMMs



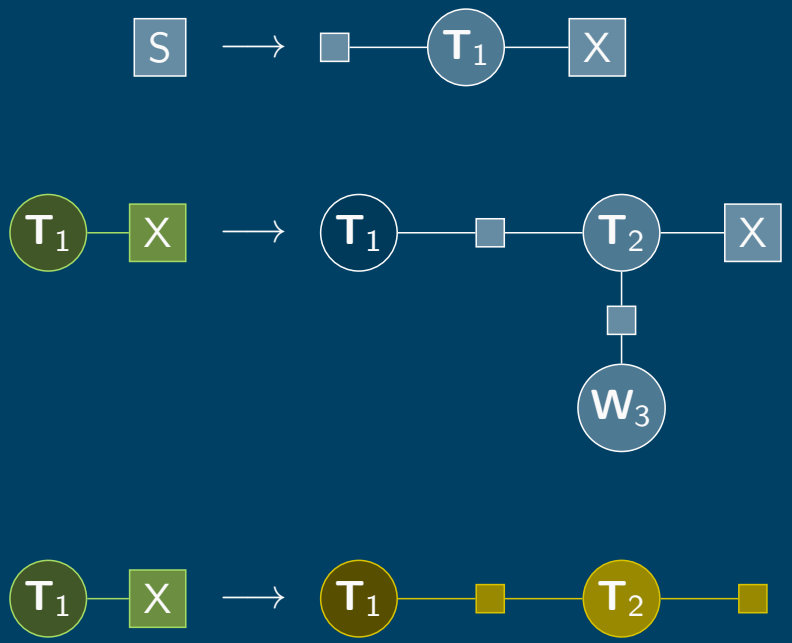
A Factor Graph Grammar for HMMs

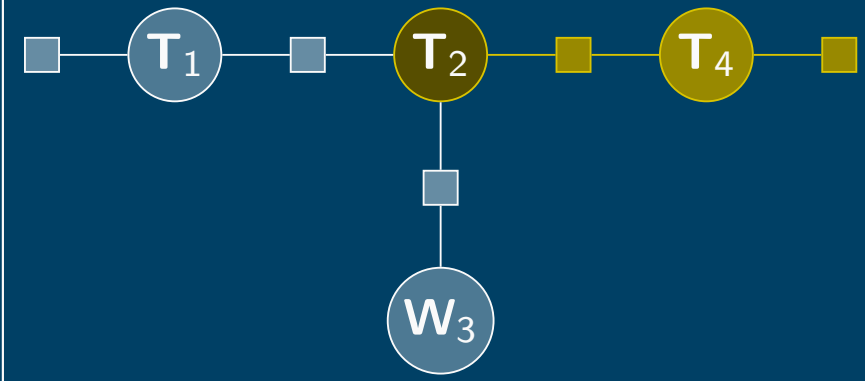


A Factor Graph Grammar for HMMs

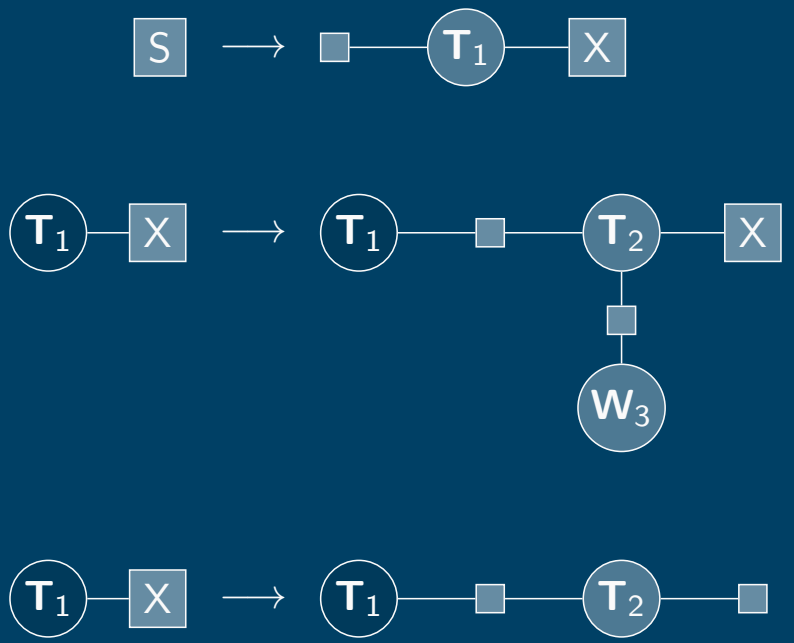


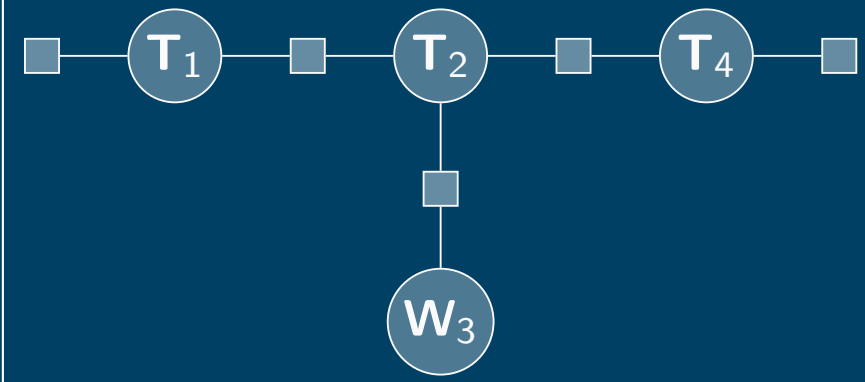
A Factor Graph Grammar for HMMs





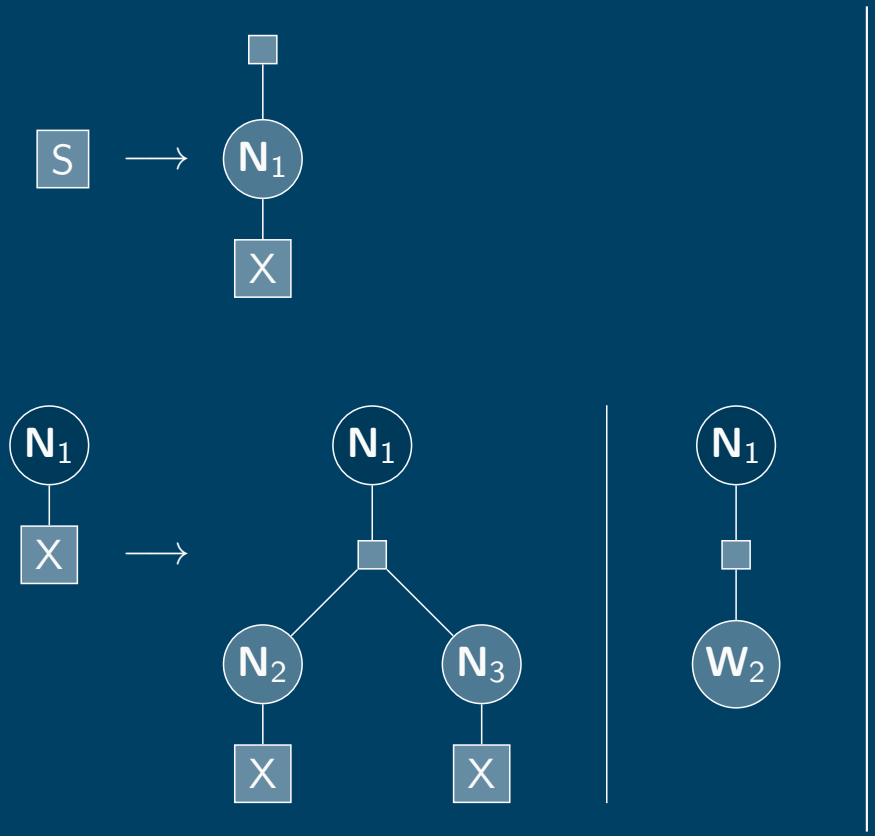
A Factor Graph Grammar for HMMs



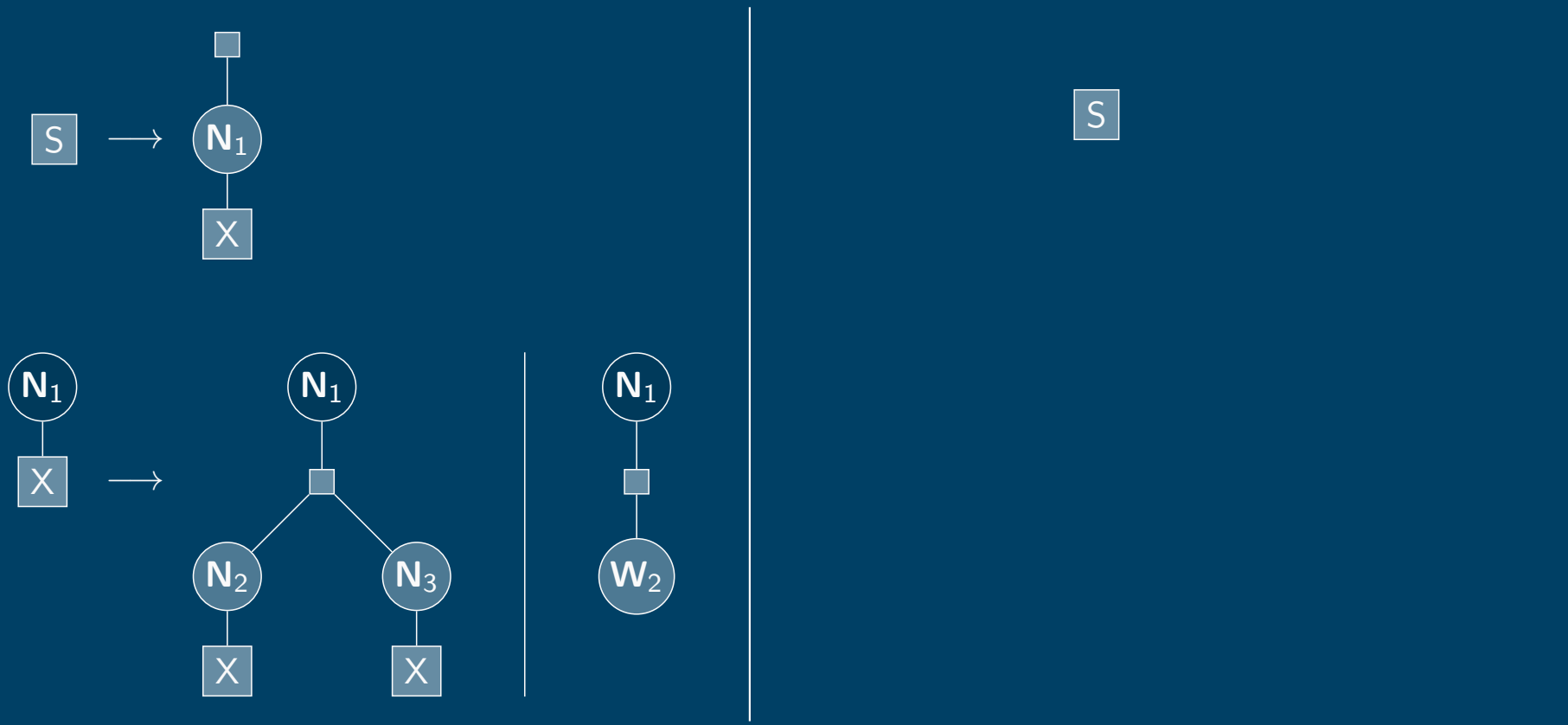


A Factor Graph Grammar for PCFGs

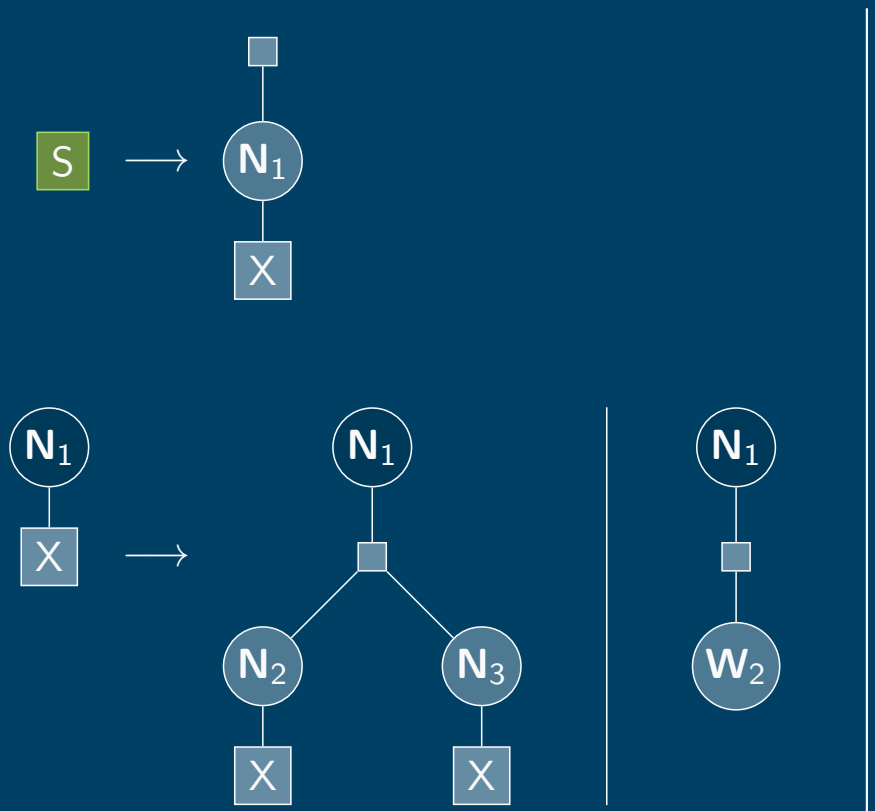
A Factor Graph Grammar for PCFGs



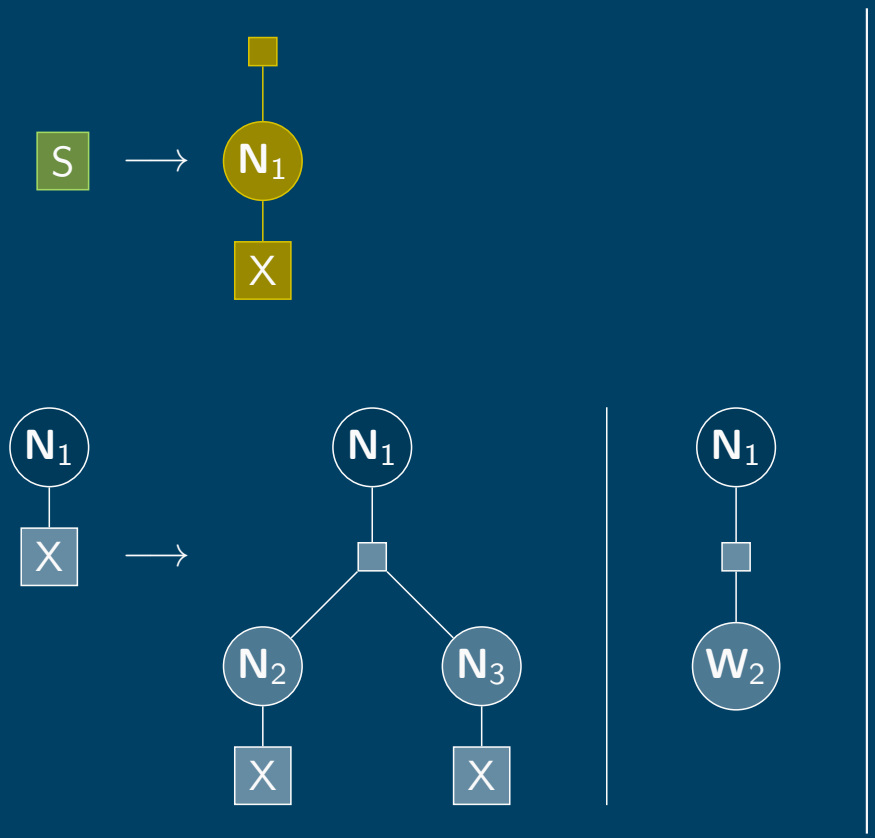
A Factor Graph Grammar for PCFGs



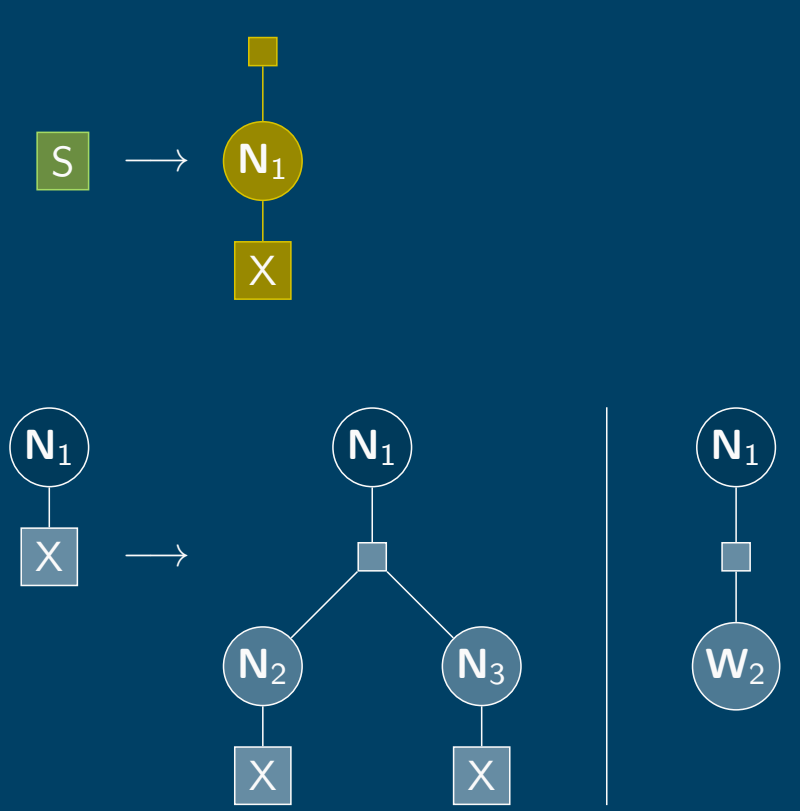
A Factor Graph Grammar for PCFGs



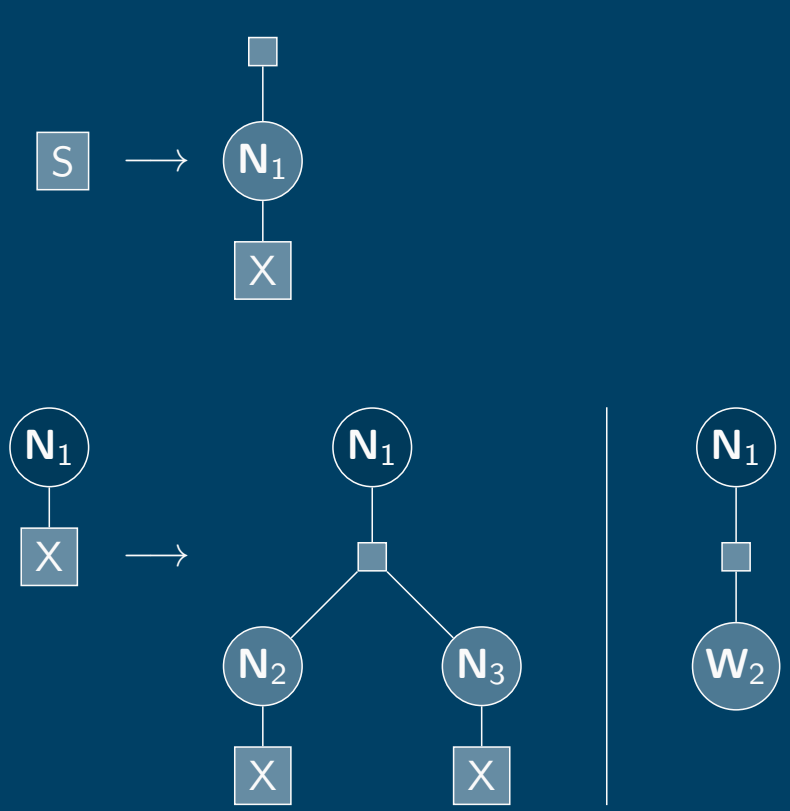
A Factor Graph Grammar for PCFGs



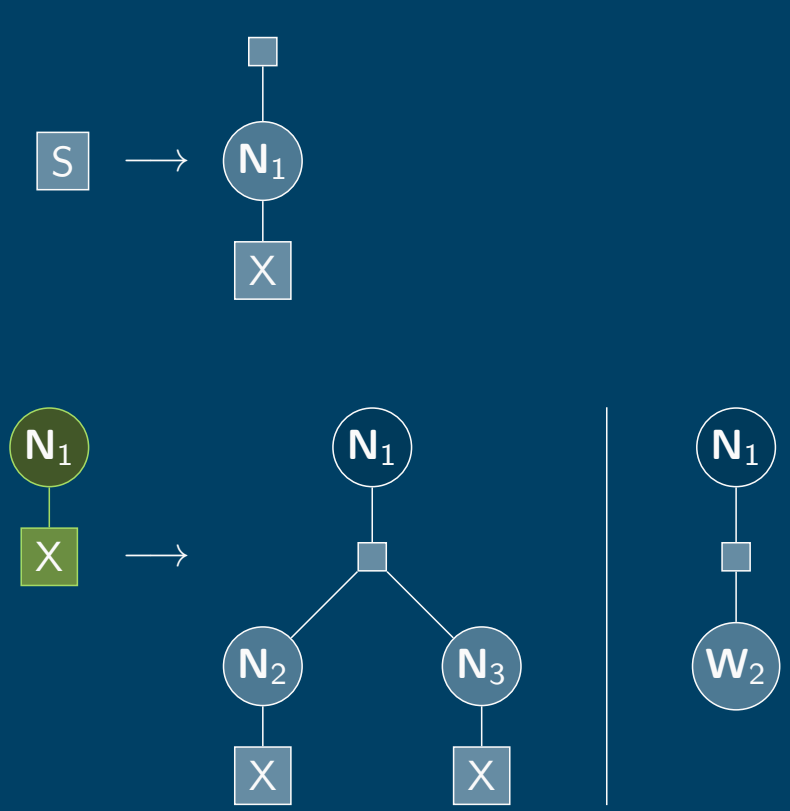
A Factor Graph Grammar for PCFGs



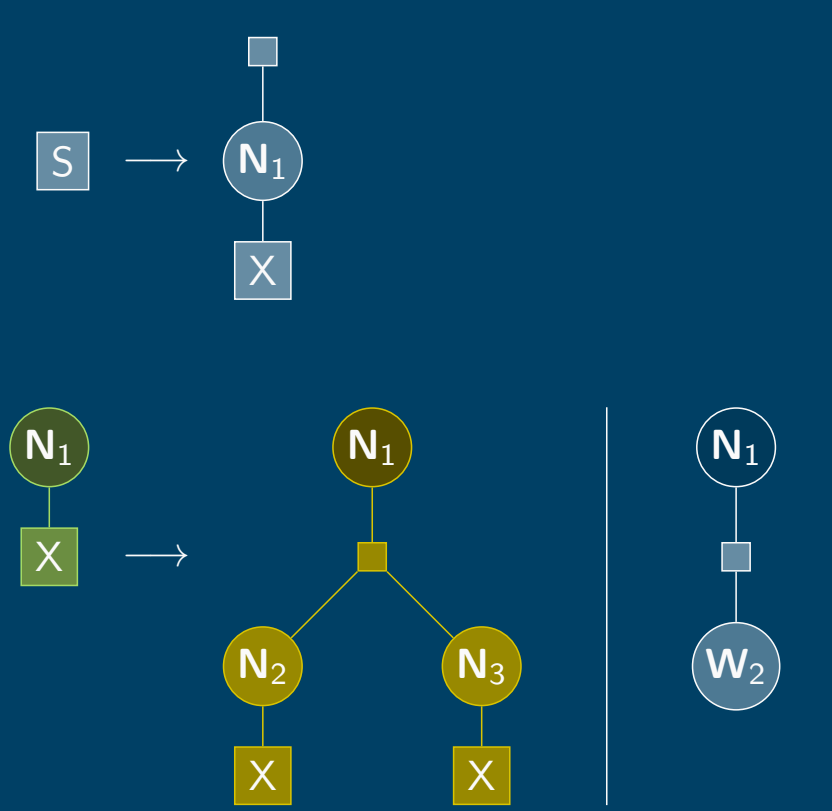
A Factor Graph Grammar for PCFGs



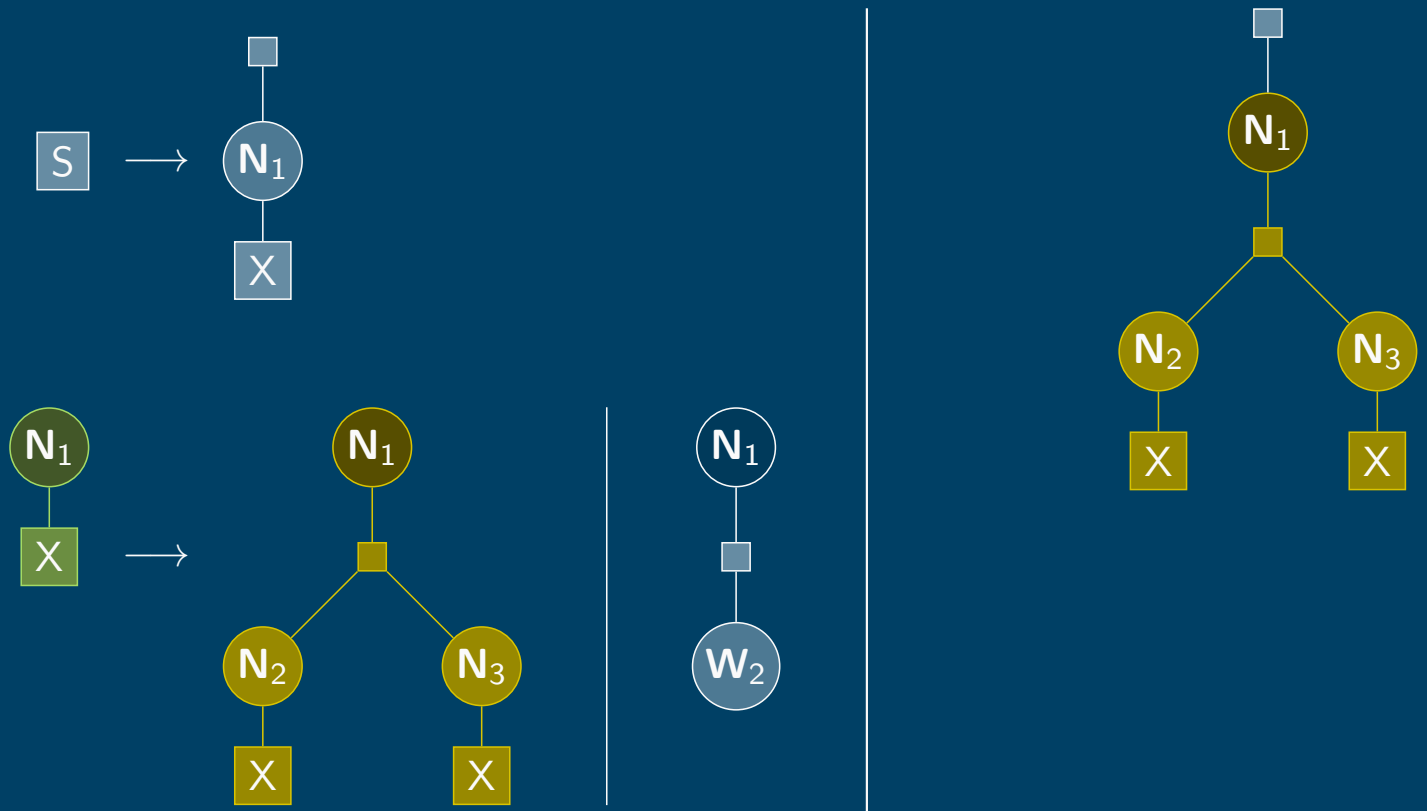
A Factor Graph Grammar for PCFGs



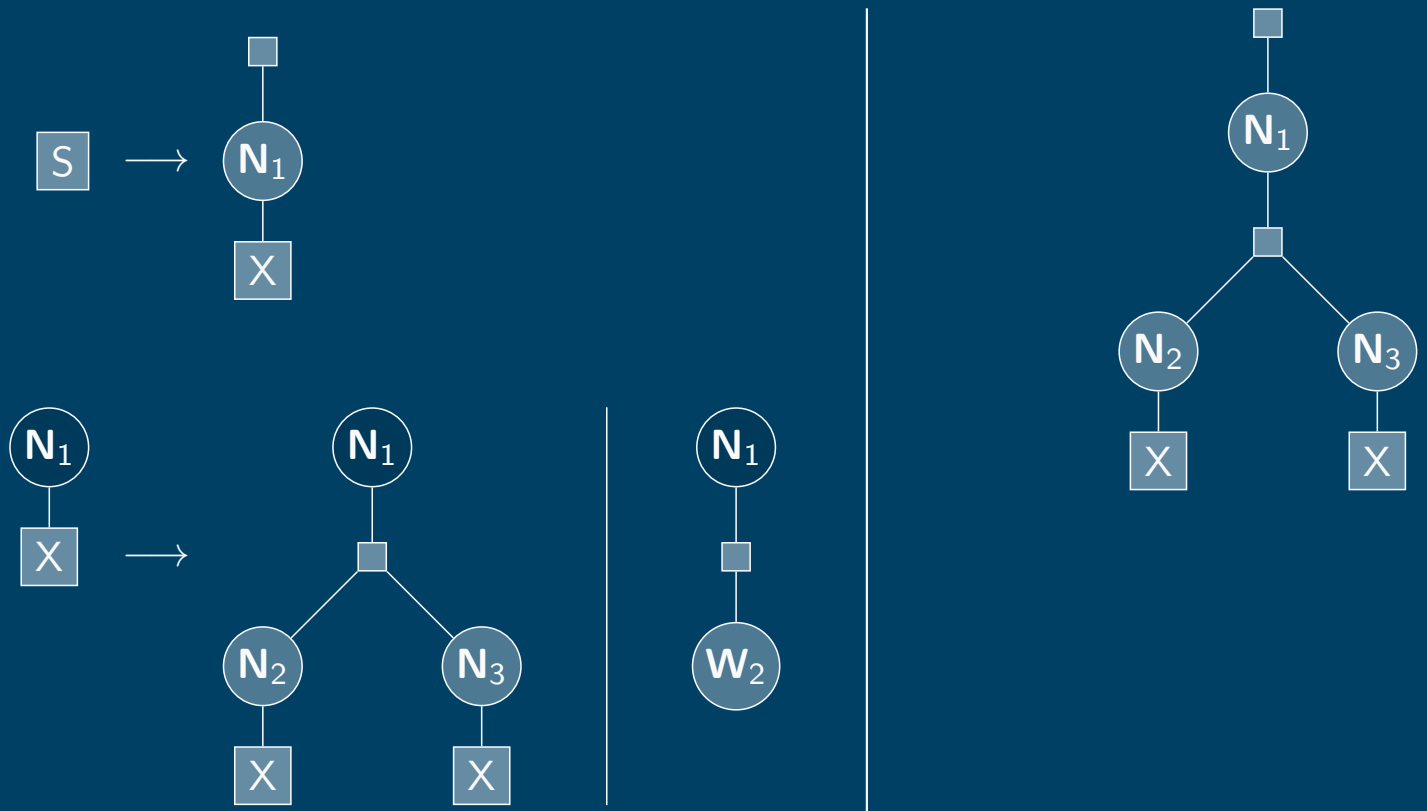
A Factor Graph Grammar for PCFGs



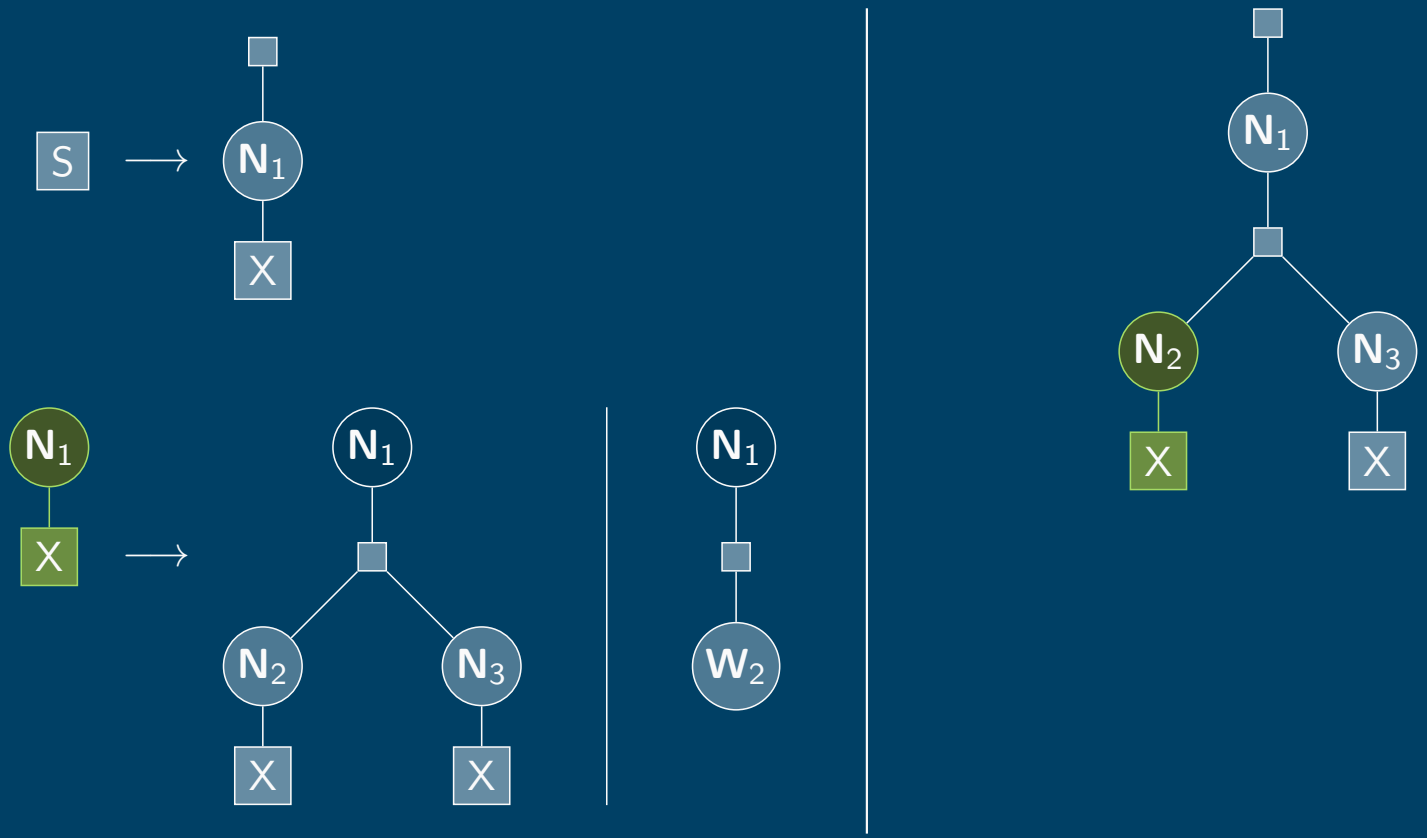
A Factor Graph Grammar for PCFGs



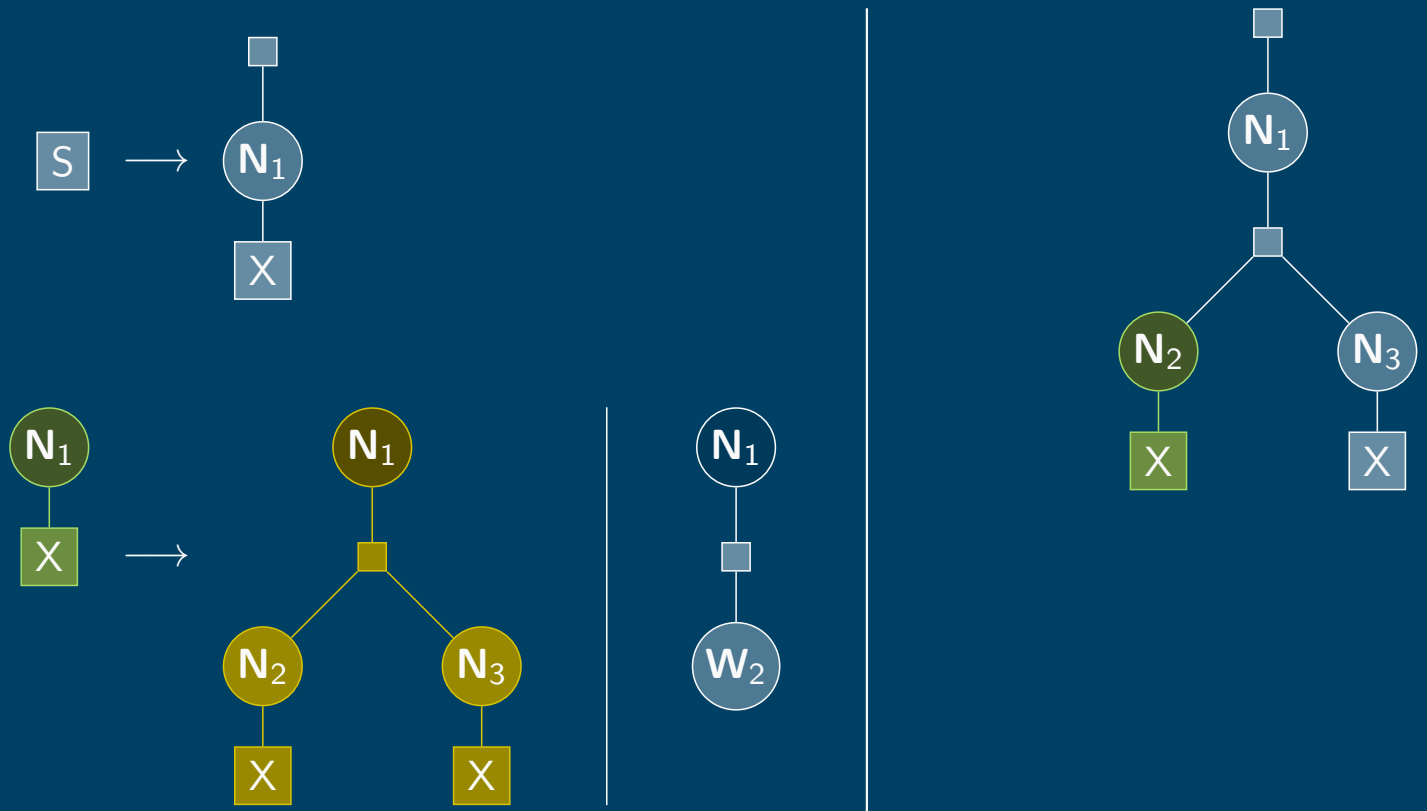
A Factor Graph Grammar for PCFGs



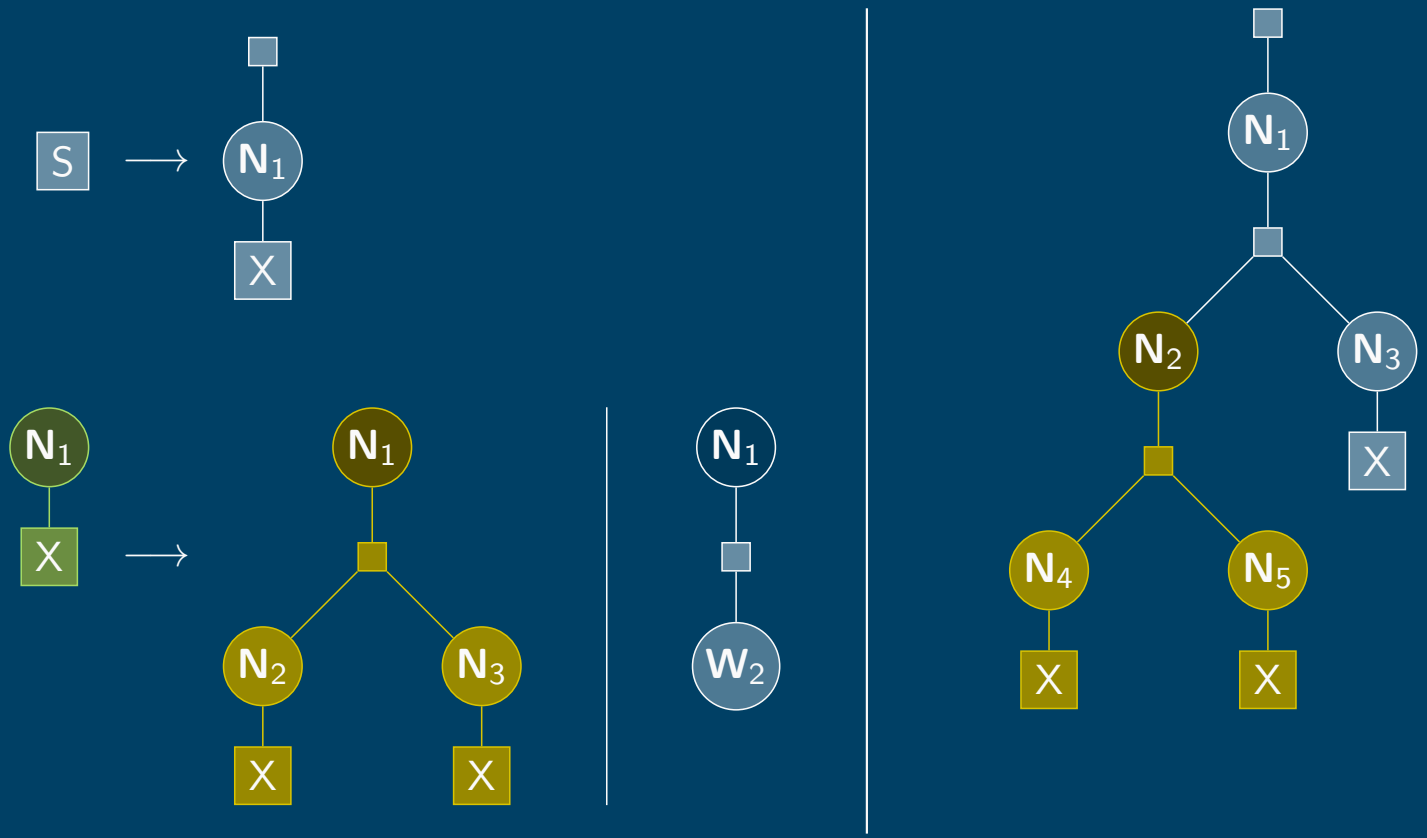
A Factor Graph Grammar for PCFGs



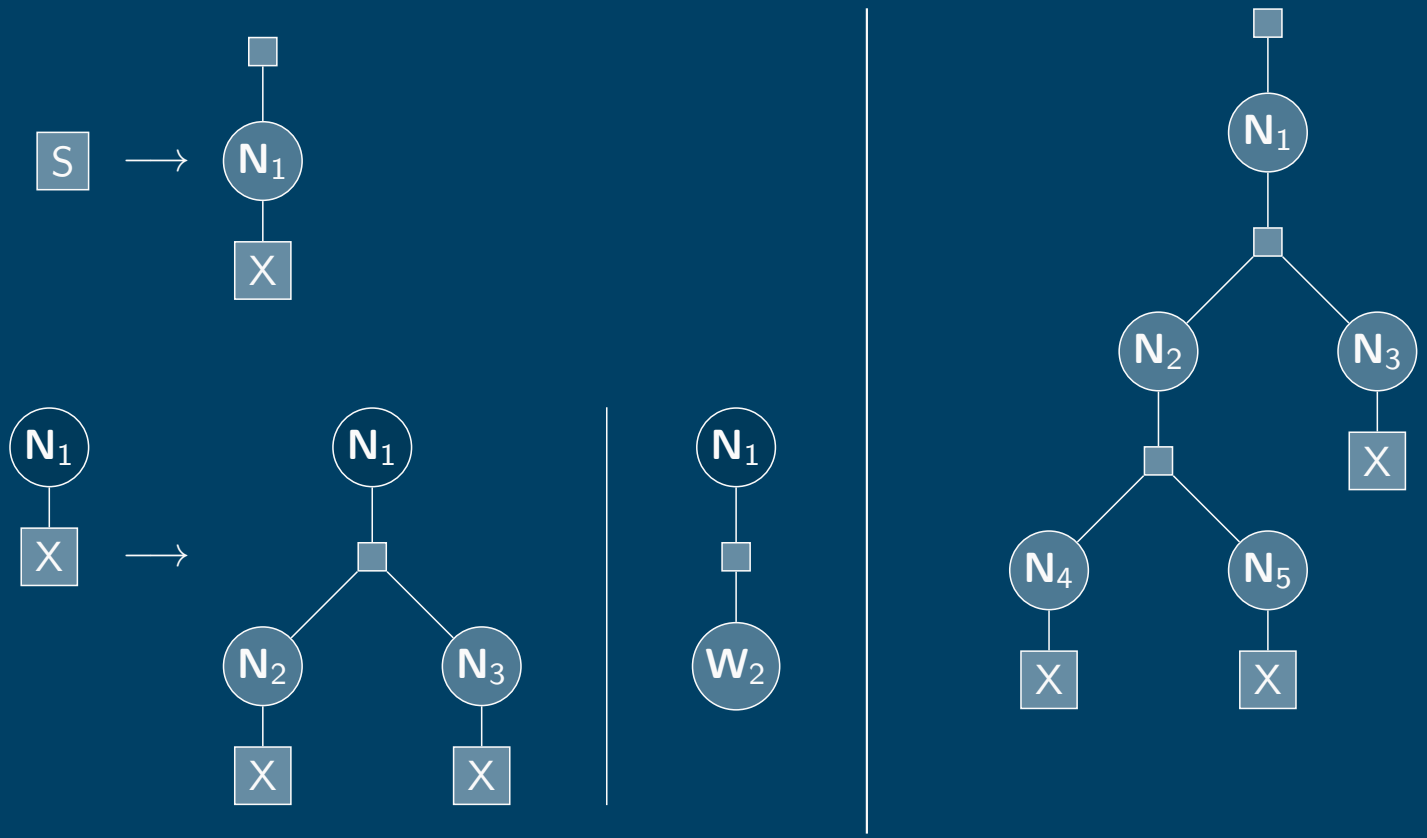
A Factor Graph Grammar for PCFGs



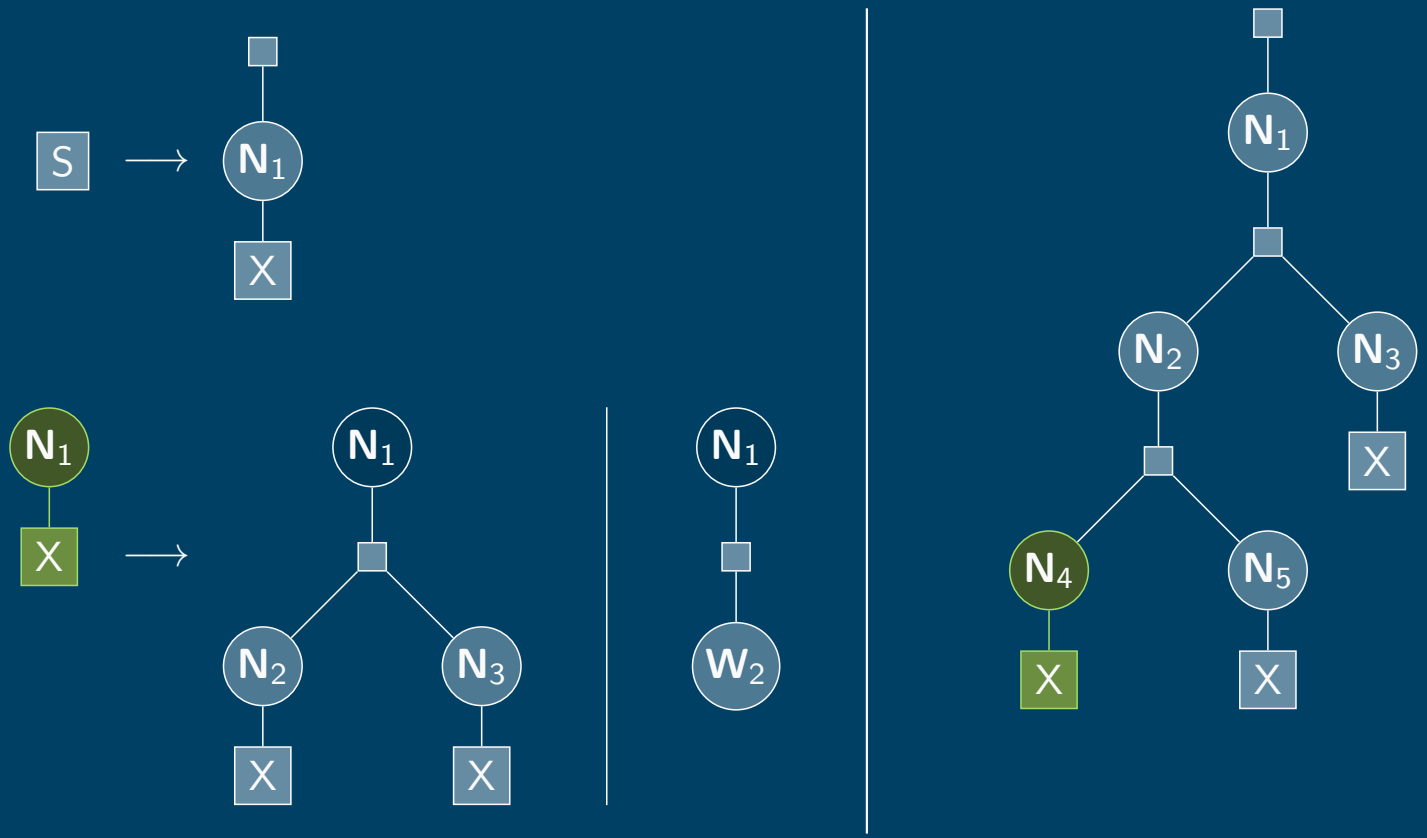
A Factor Graph Grammar for PCFGs



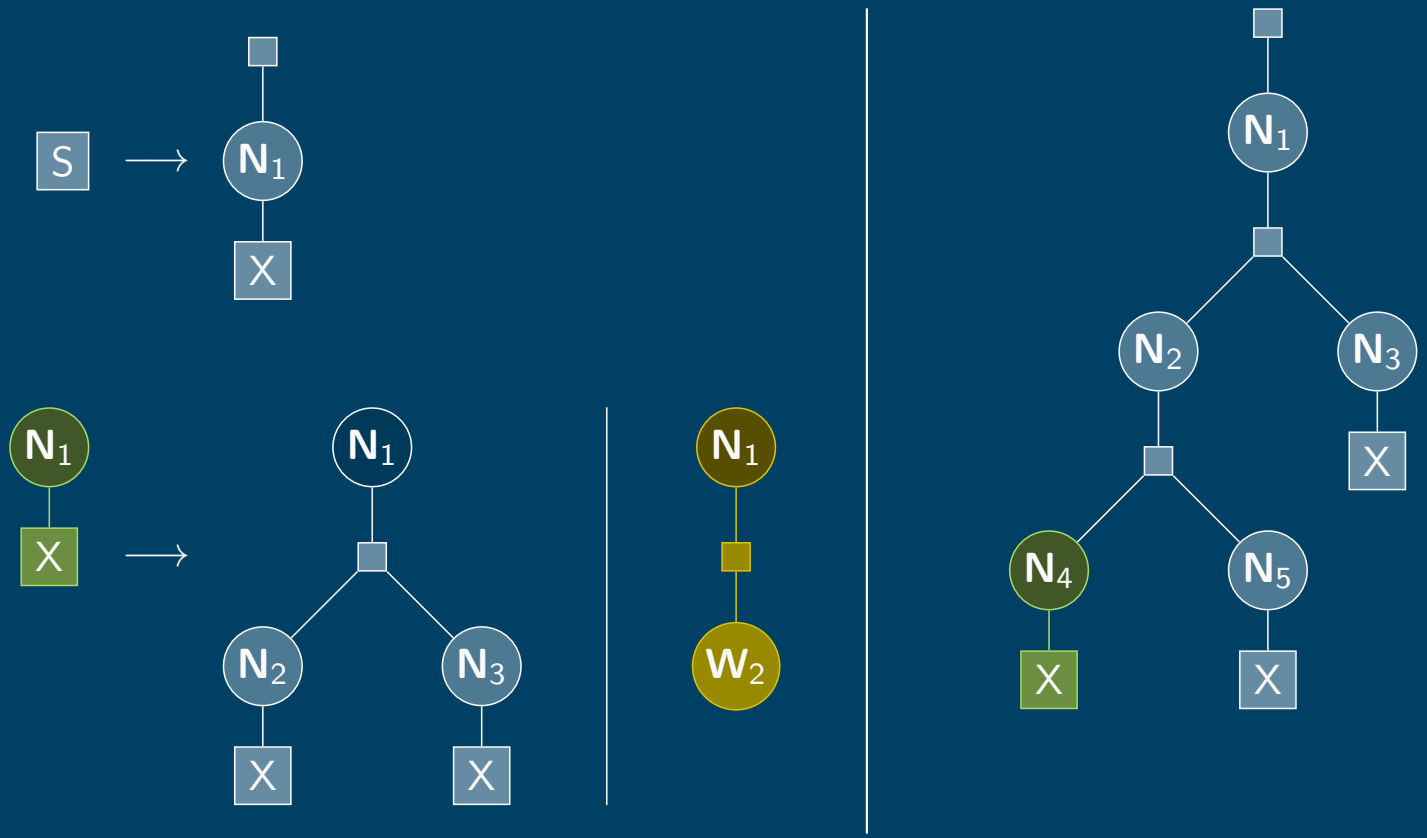
A Factor Graph Grammar for PCFGs



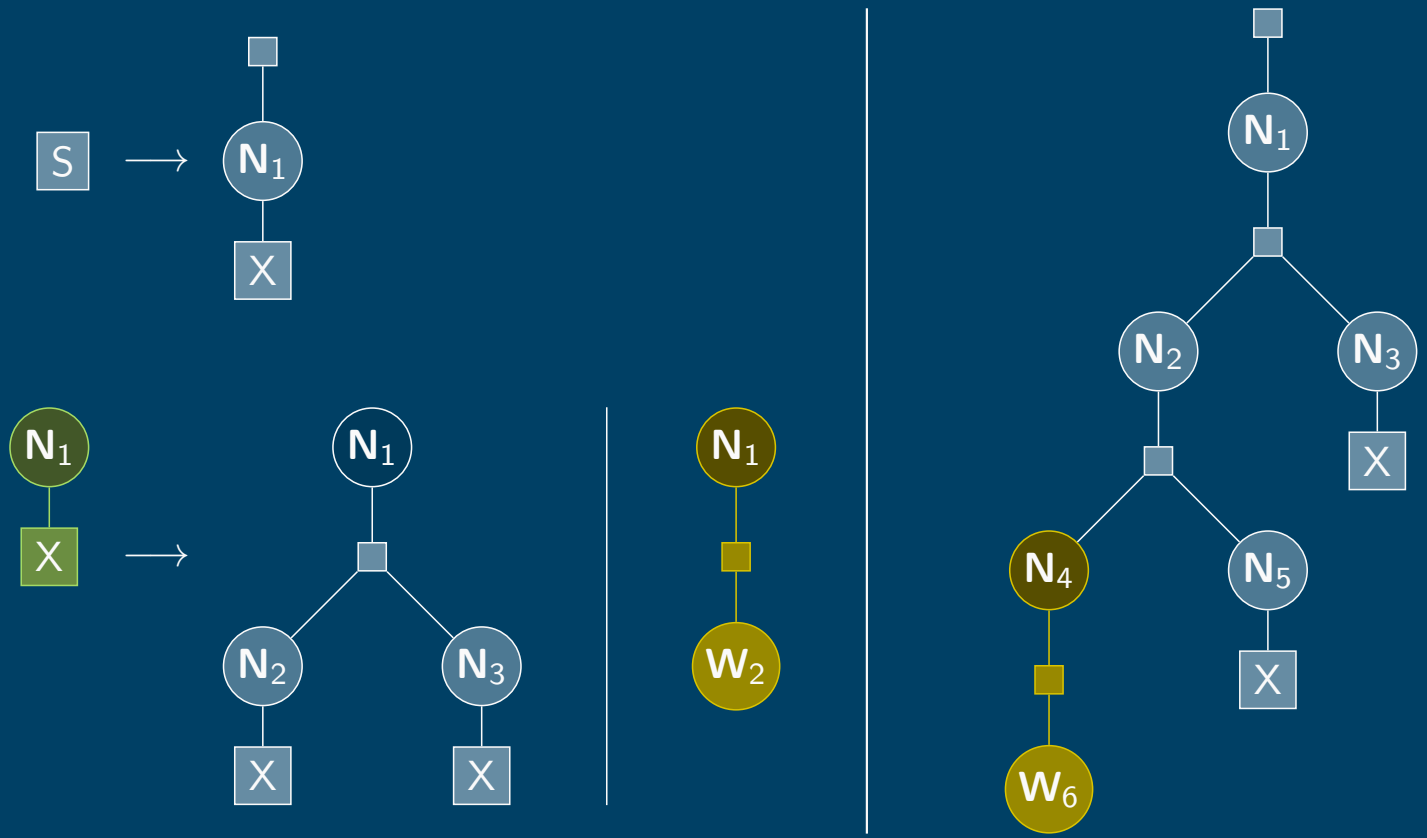
A Factor Graph Grammar for PCFGs



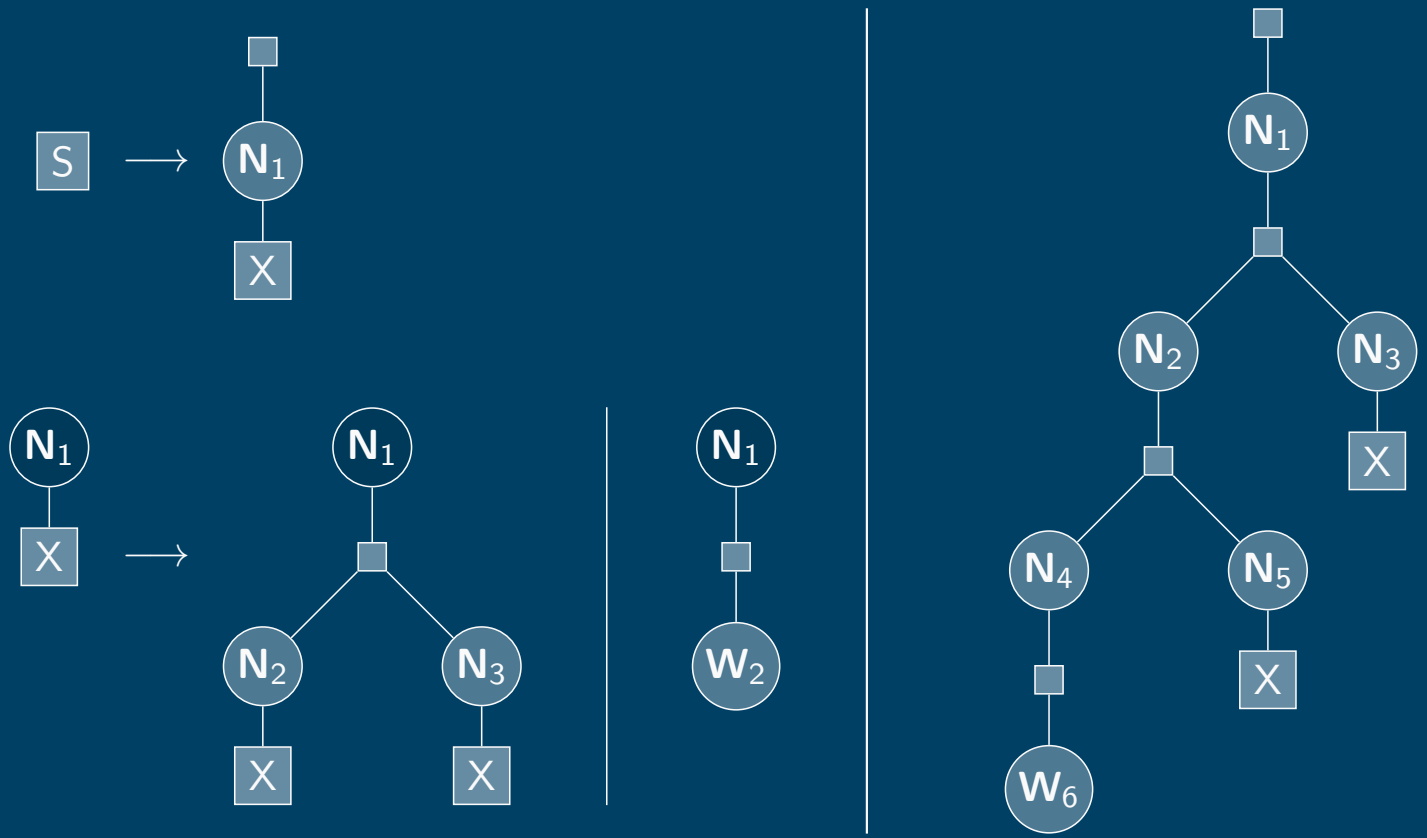
A Factor Graph Grammar for PCFGs



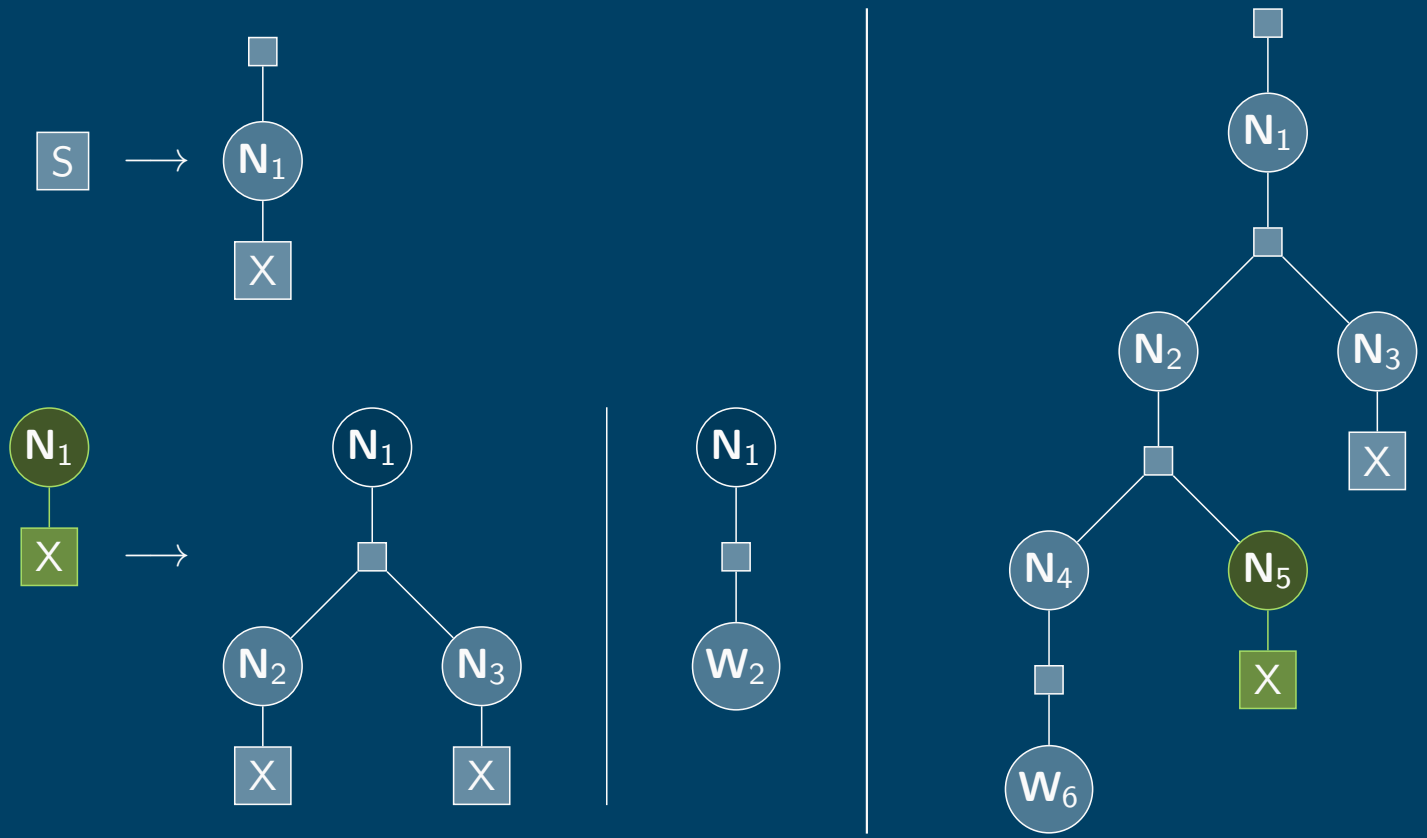
A Factor Graph Grammar for PCFGs



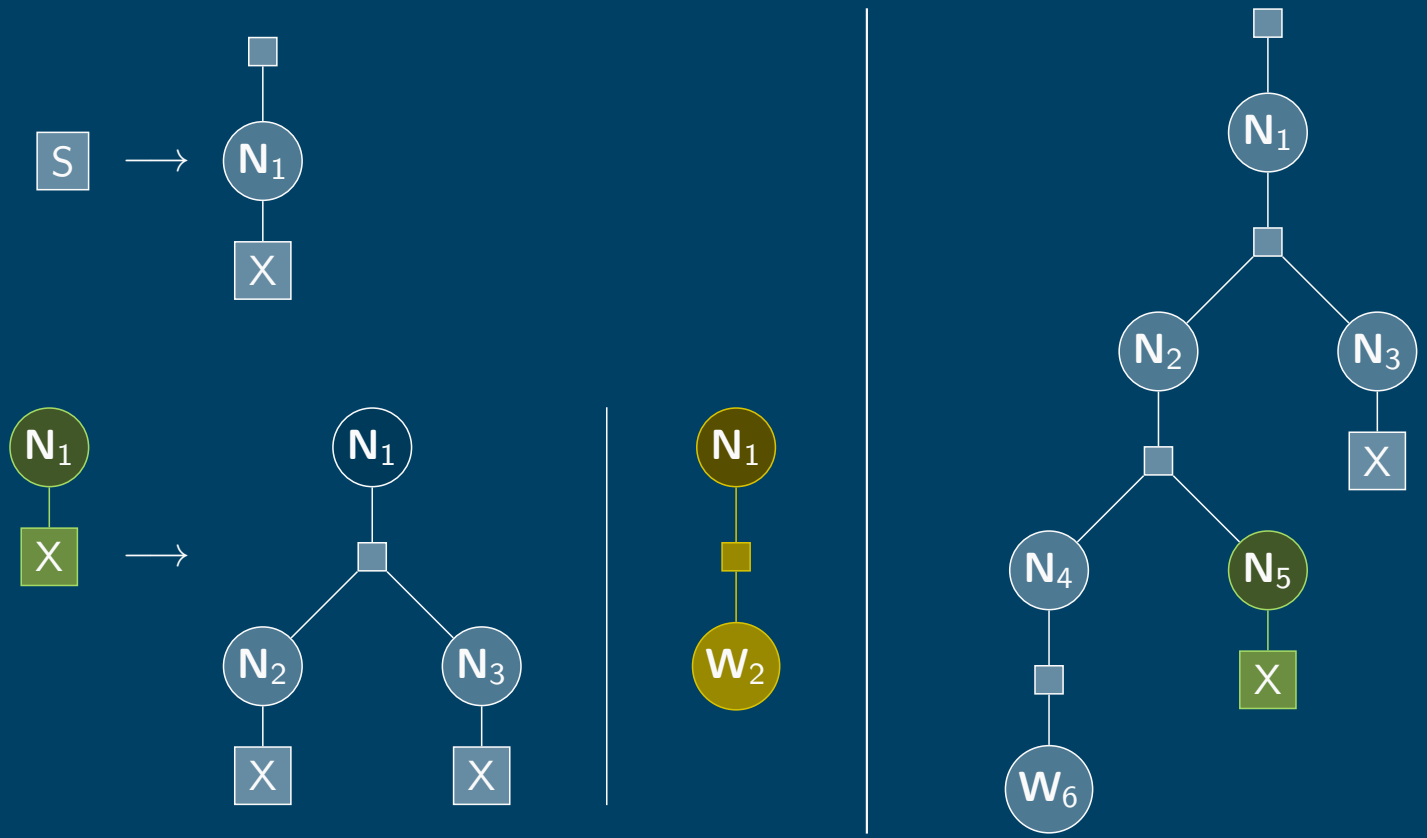
A Factor Graph Grammar for PCFGs



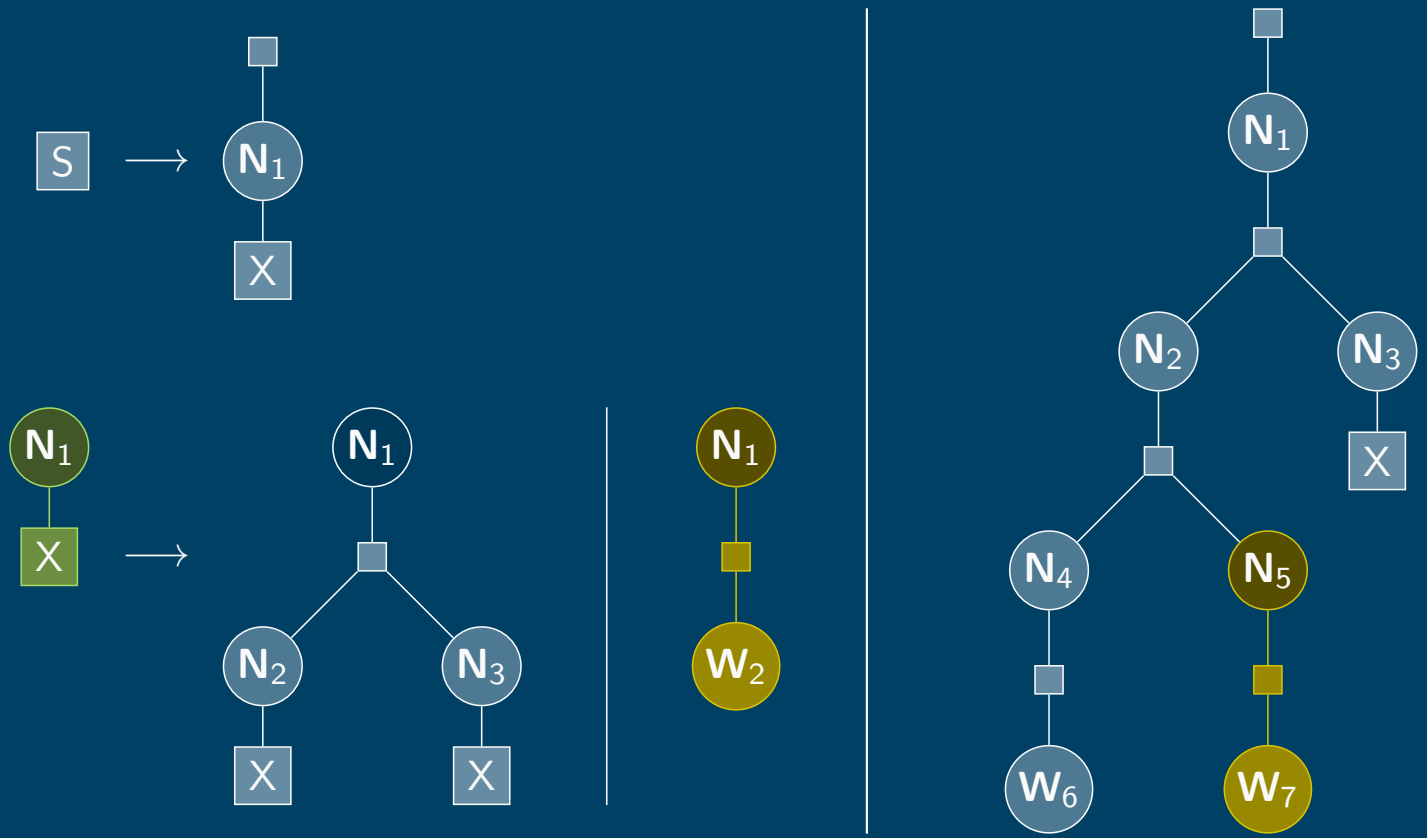
A Factor Graph Grammar for PCFGs



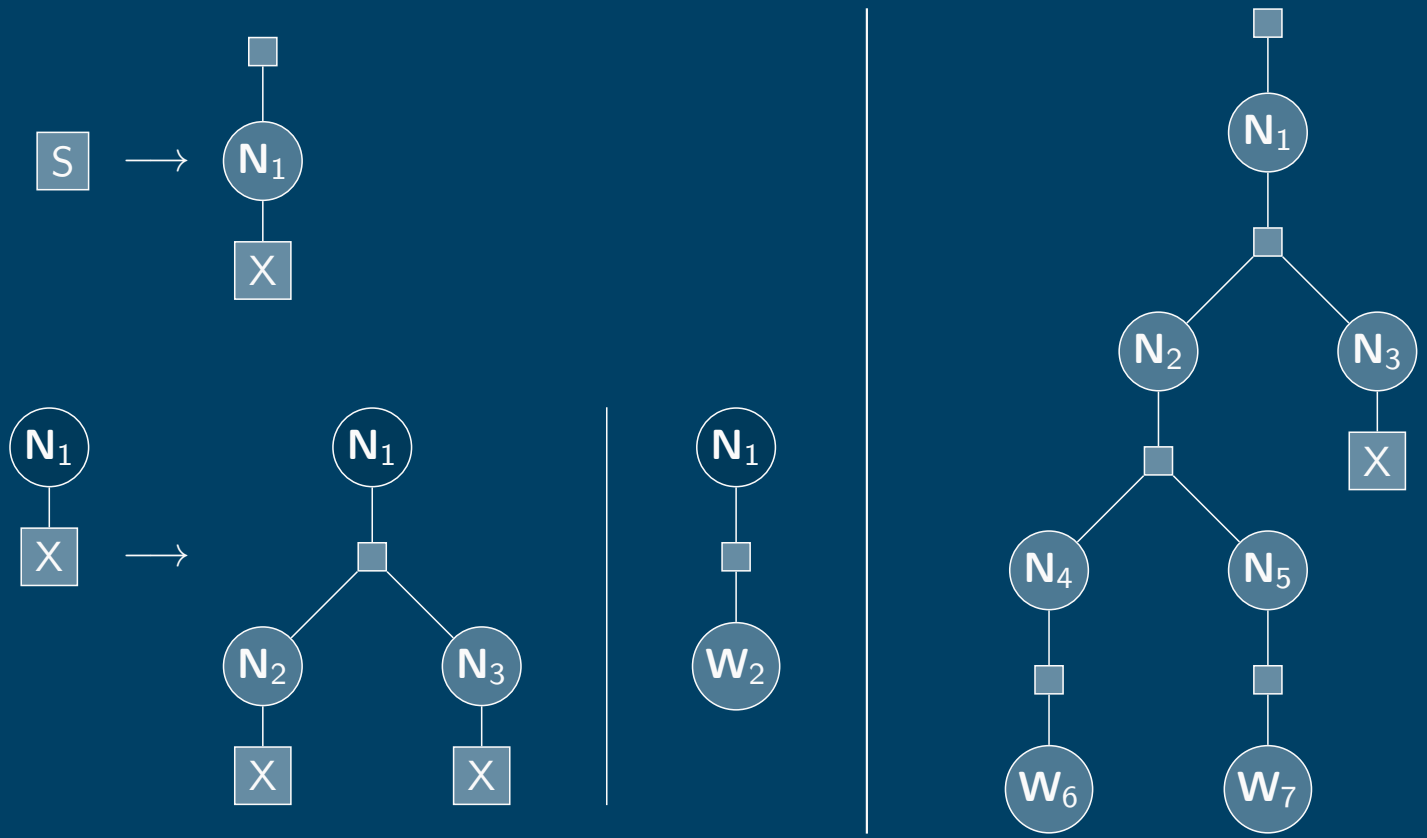
A Factor Graph Grammar for PCFGs



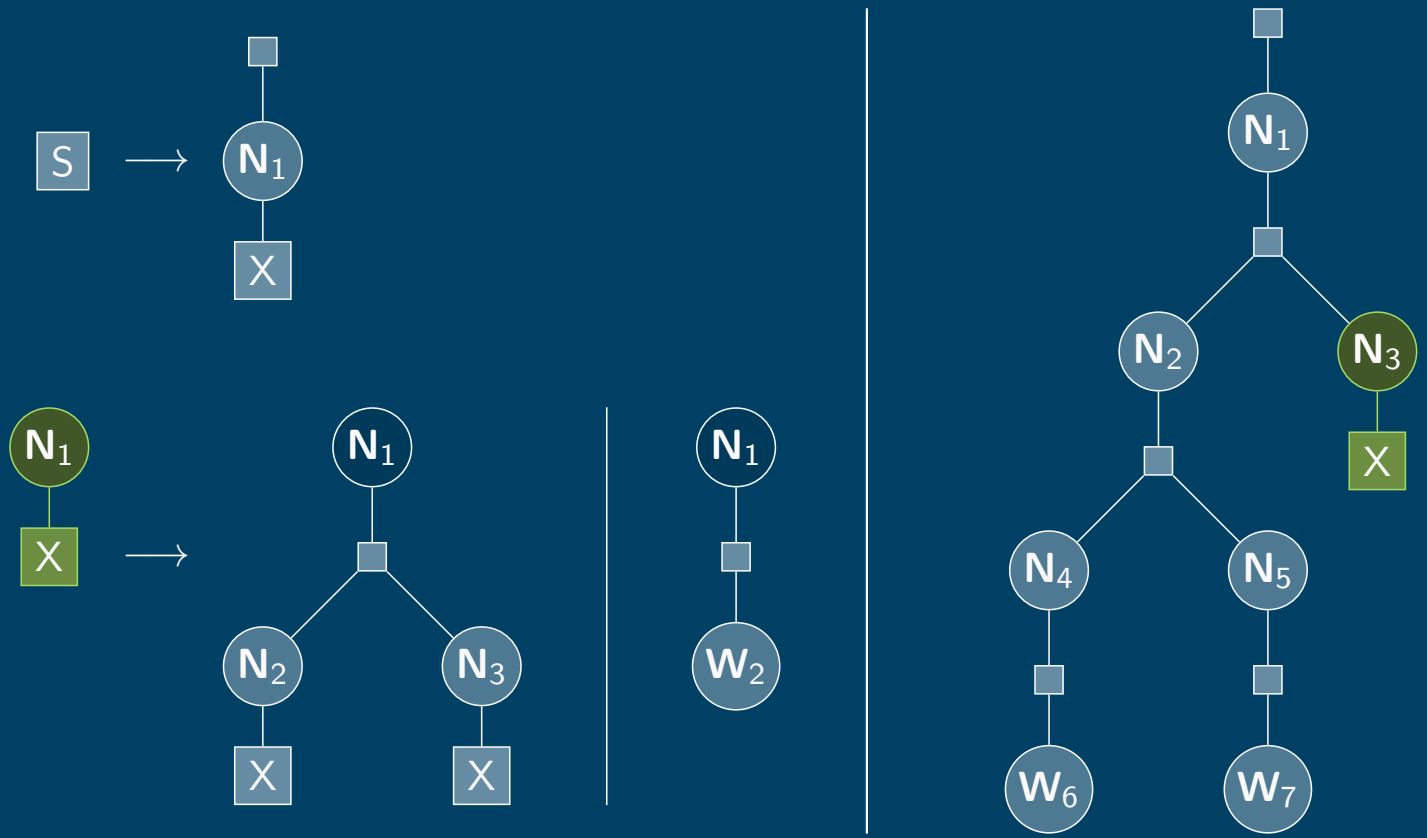
A Factor Graph Grammar for PCFGs



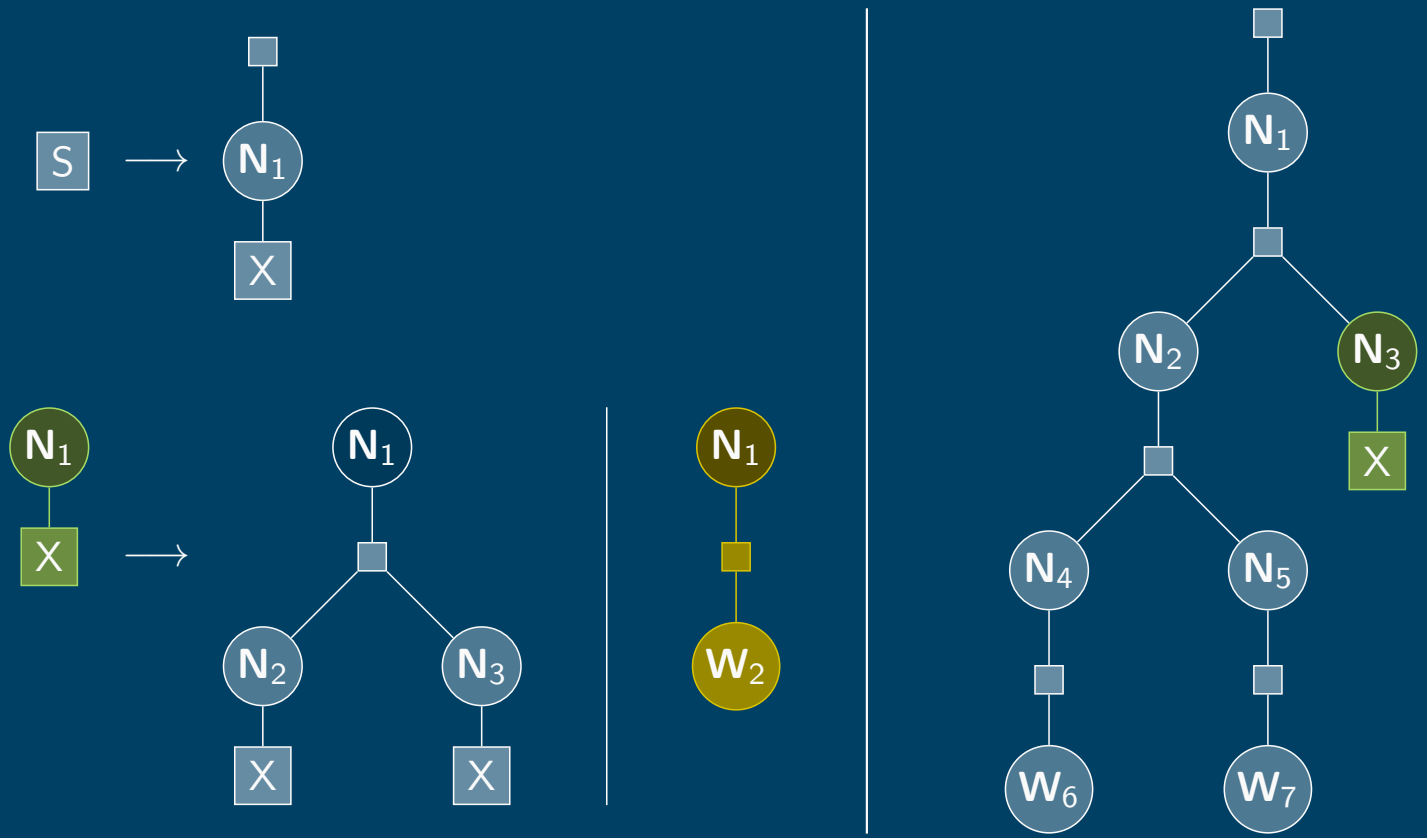
A Factor Graph Grammar for PCFGs



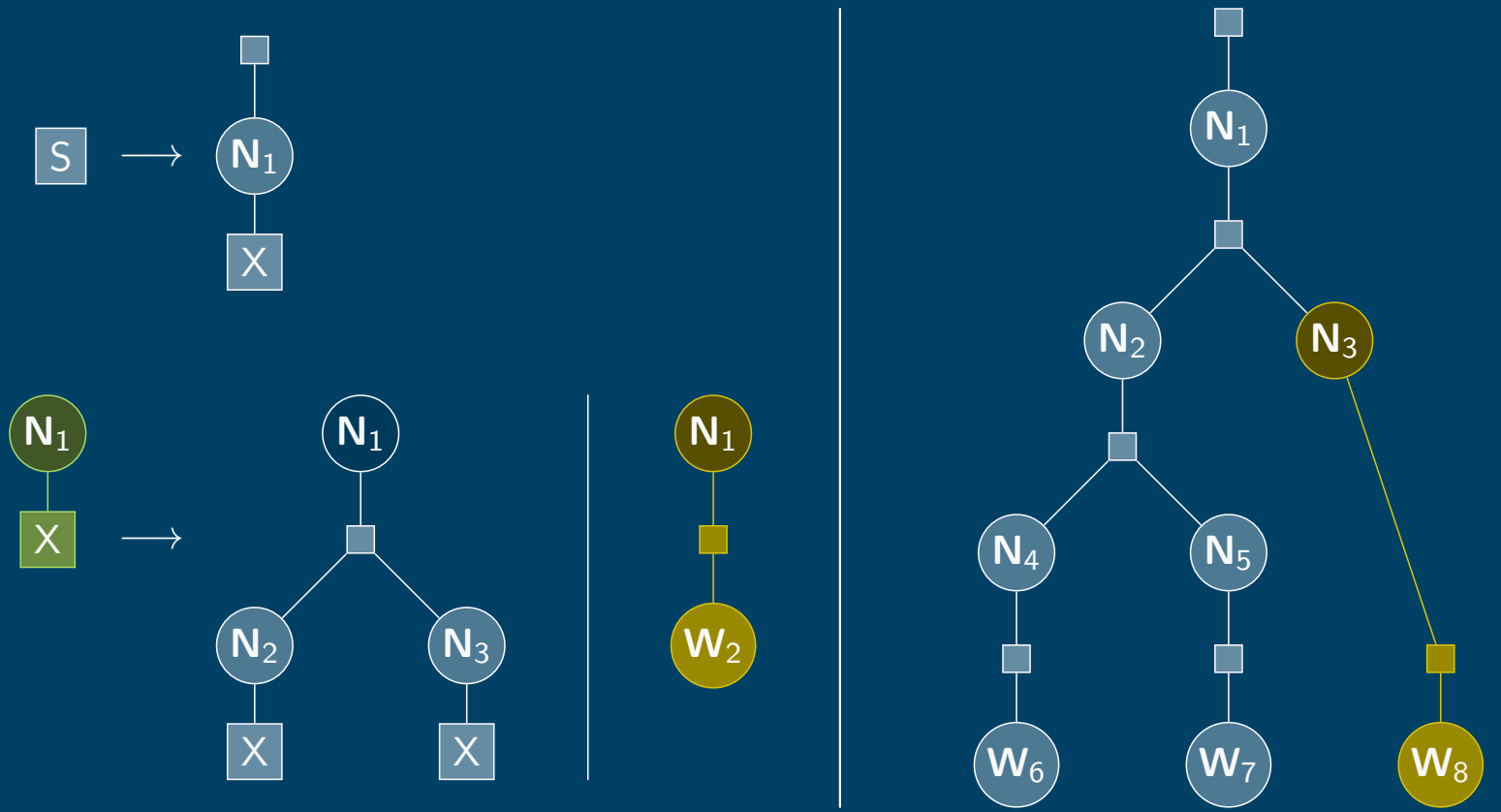
A Factor Graph Grammar for PCFGs



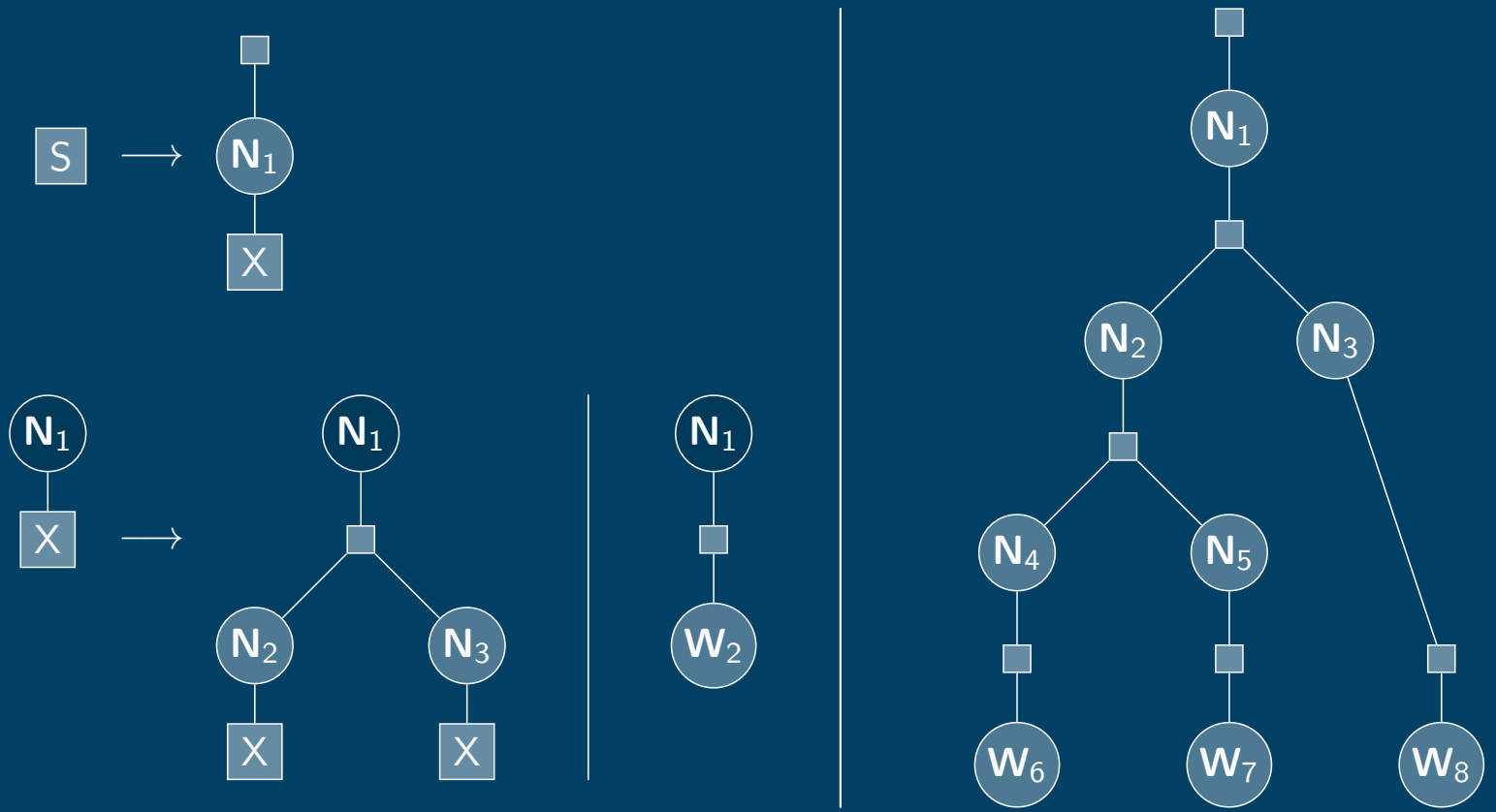
A Factor Graph Grammar for PCFGs



A Factor Graph Grammar for PCFGs



A Factor Graph Grammar for PCFGs



More FGG Ideas

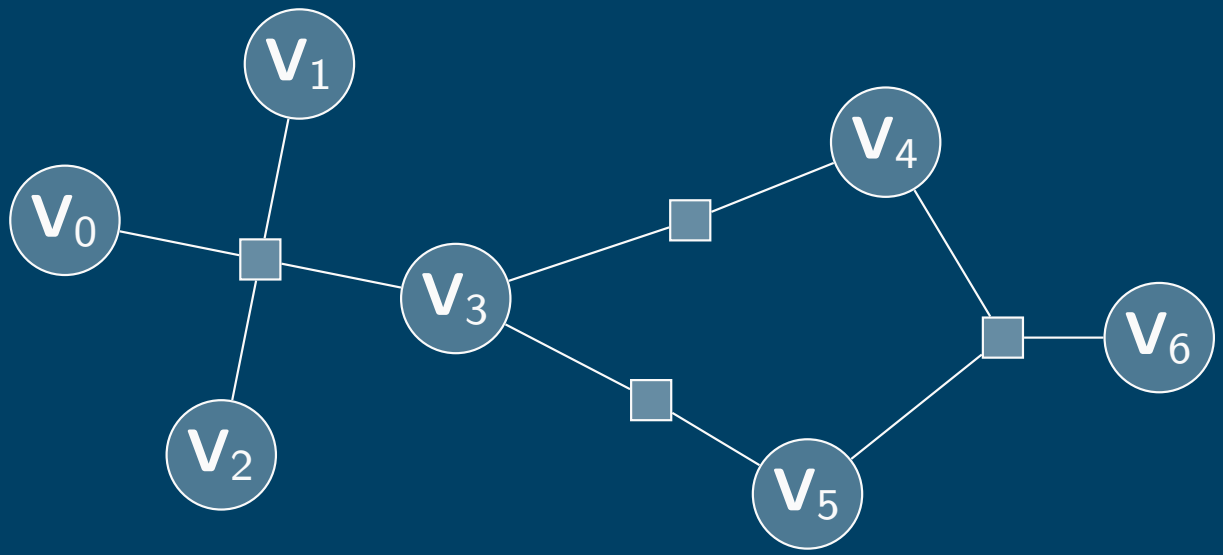
What else can you use FGGs for?

- Linear chains of nodes
- Anything tree-shaped (phylogeny trees, abstract syntax trees)
- More grammatical formalisms (TAG, LCFRS)
- Various types of DAGs (AMRs, control flow graphs, git commit histories?)
- RNA secondary structures

Sum-Product in a Factor Graph

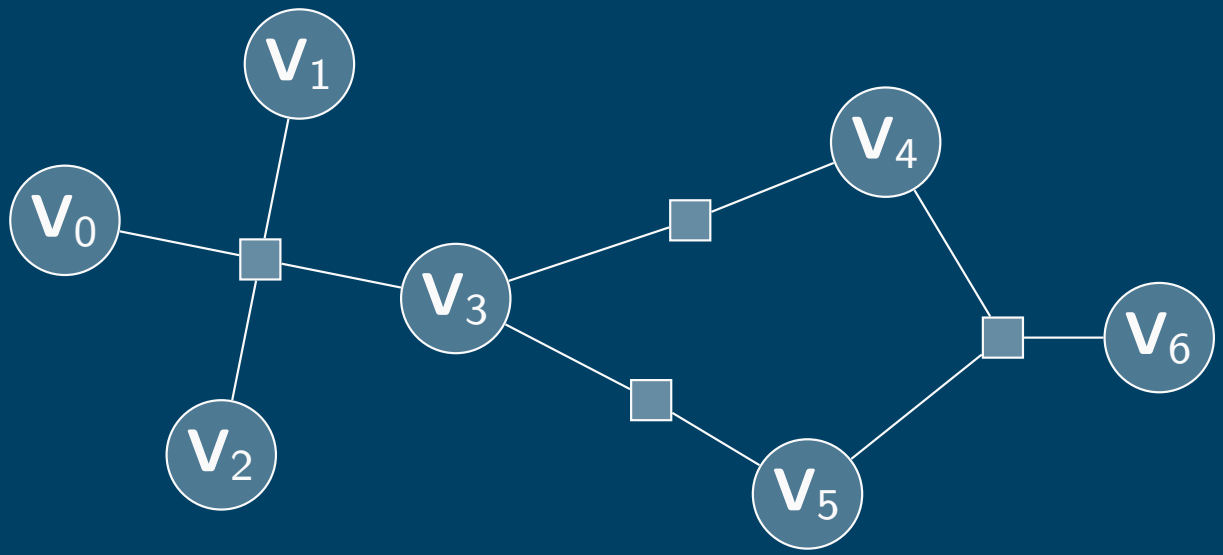
Sum-Product in a Factor Graph

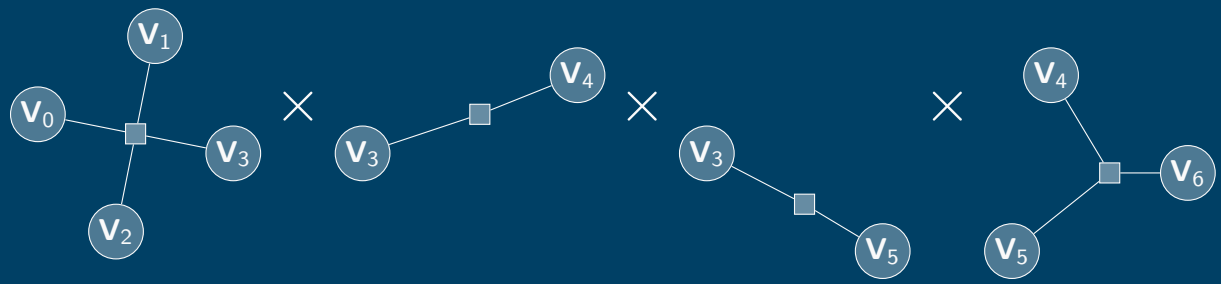
H:



Sum-Product in a Factor Graph

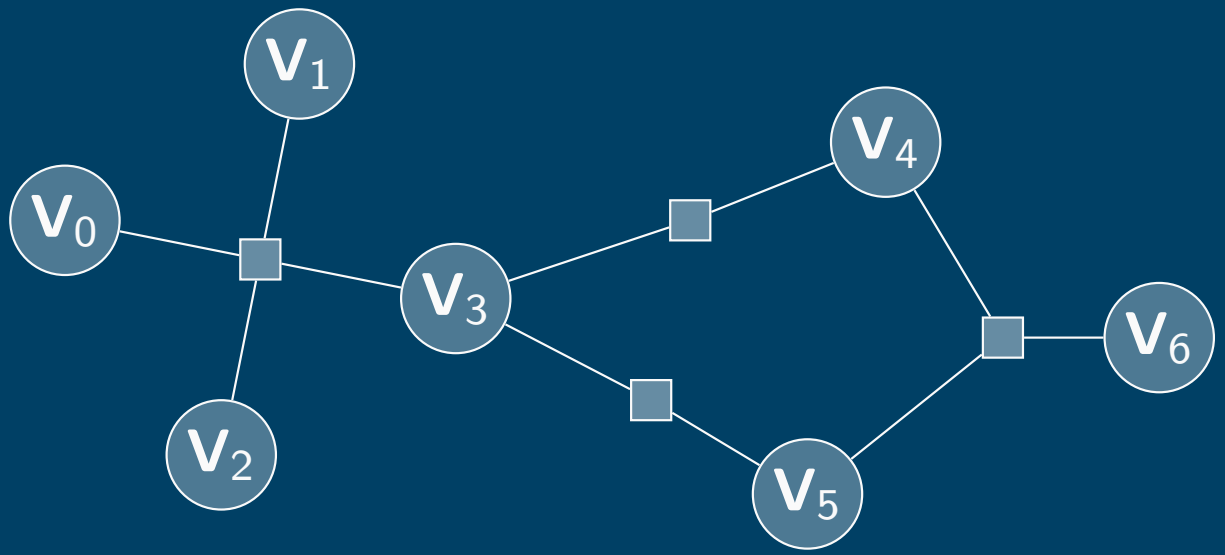
H:



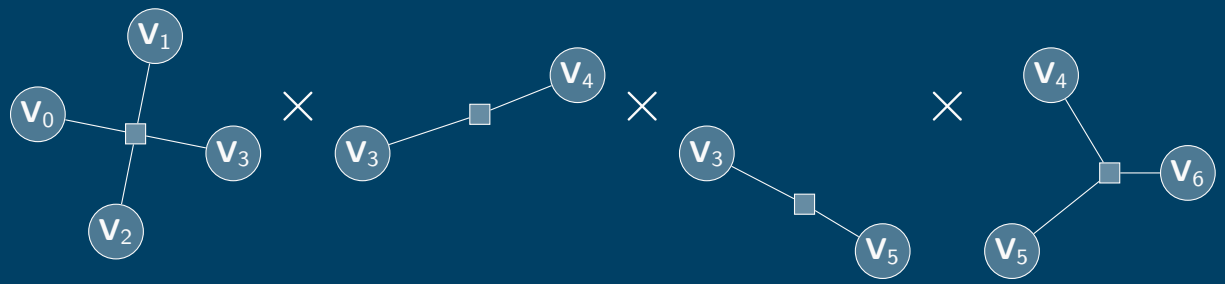


Sum-Product in a Factor Graph

H:

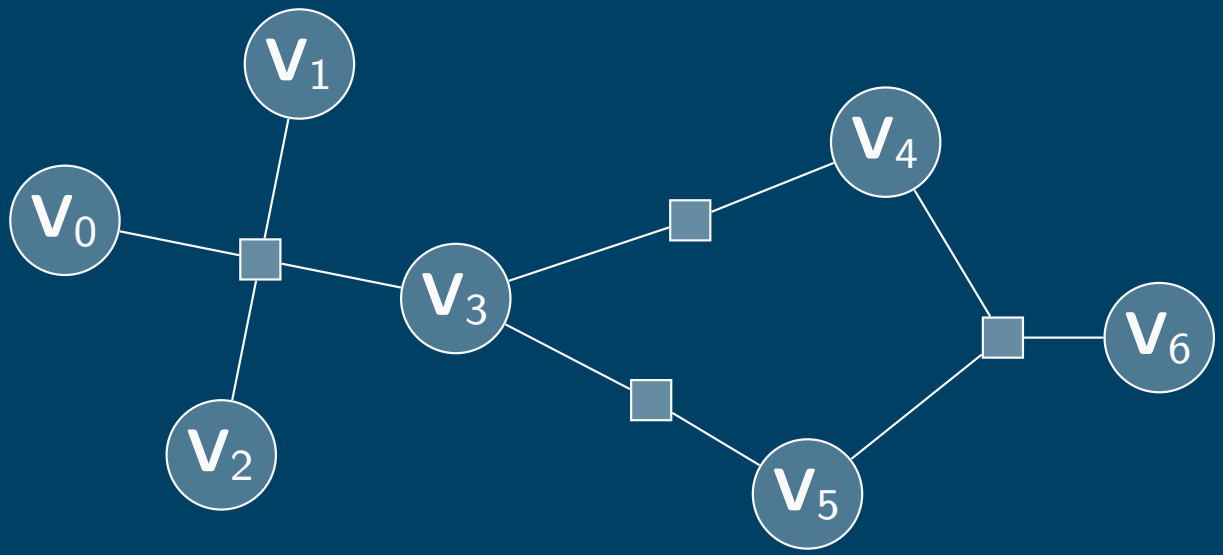


$$\frac{1}{Z}$$



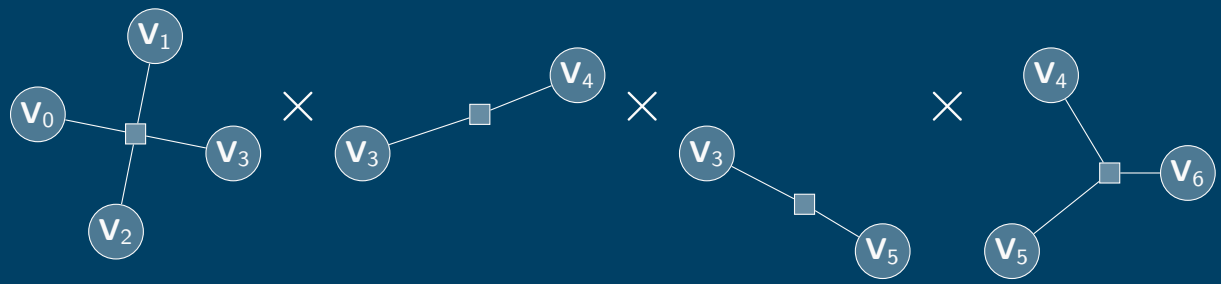
Sum-Product in a Factor Graph

H:



$Z(H) =$

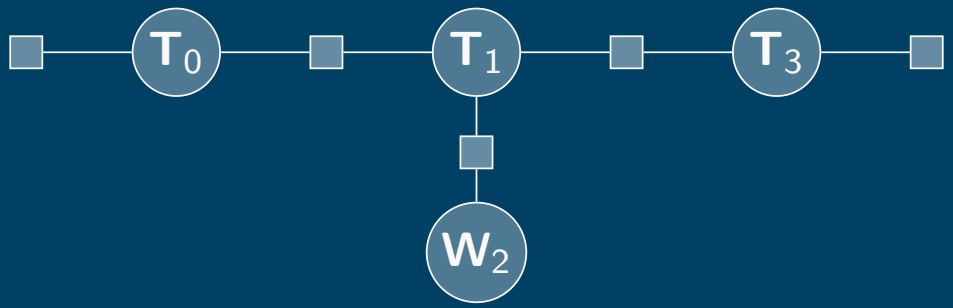
$$\sum_{\text{assignments to } V_0, V_1, \ldots, V_6}$$



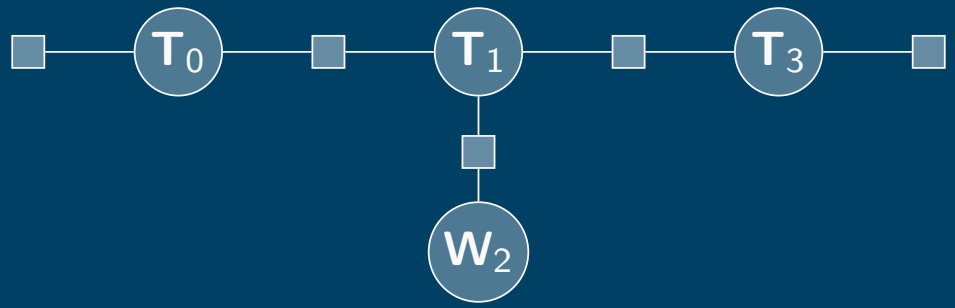
To compute sum-product of factor graph: sum over the weights of all assignments

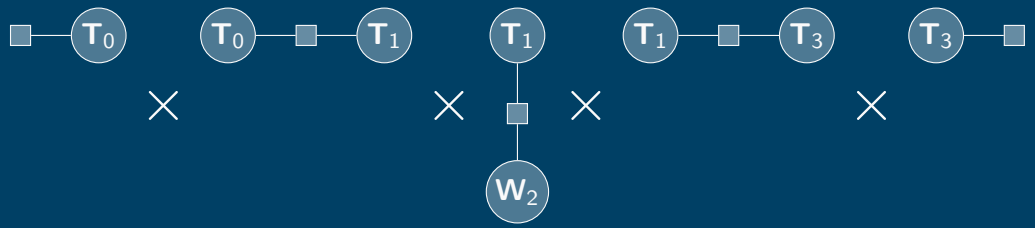
Sum-Product in a Factor Graph Grammar

Sum-Product in a Factor Graph Grammar



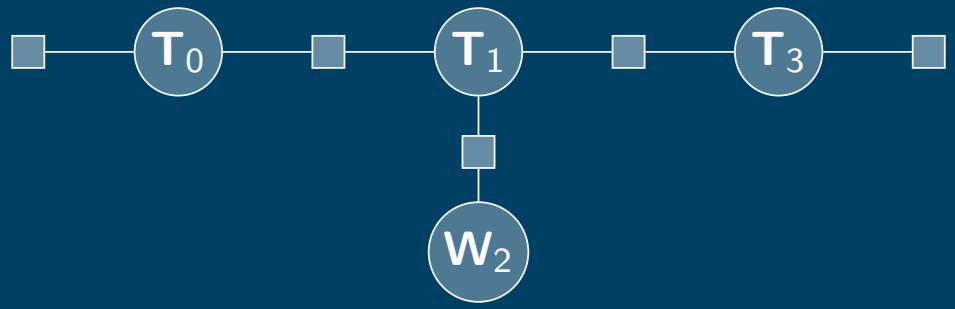
Sum-Product in a Factor Graph Grammar

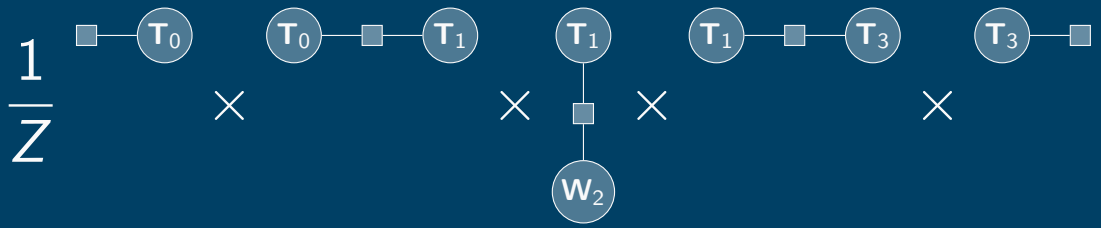




To compute weight of assignment: multiply factors together

Sum-Product in a Factor Graph Grammar

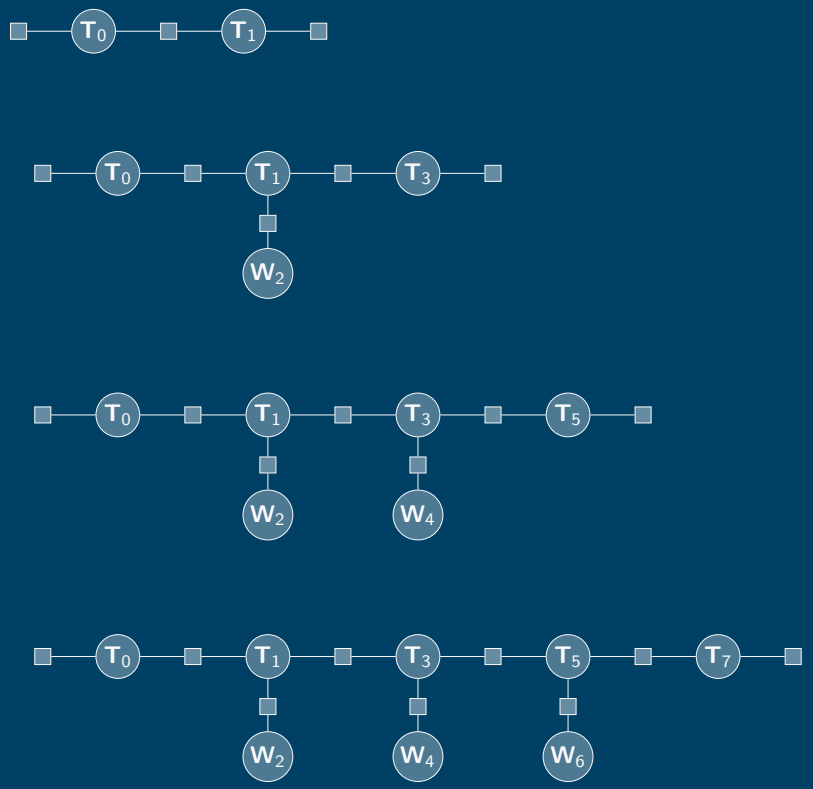




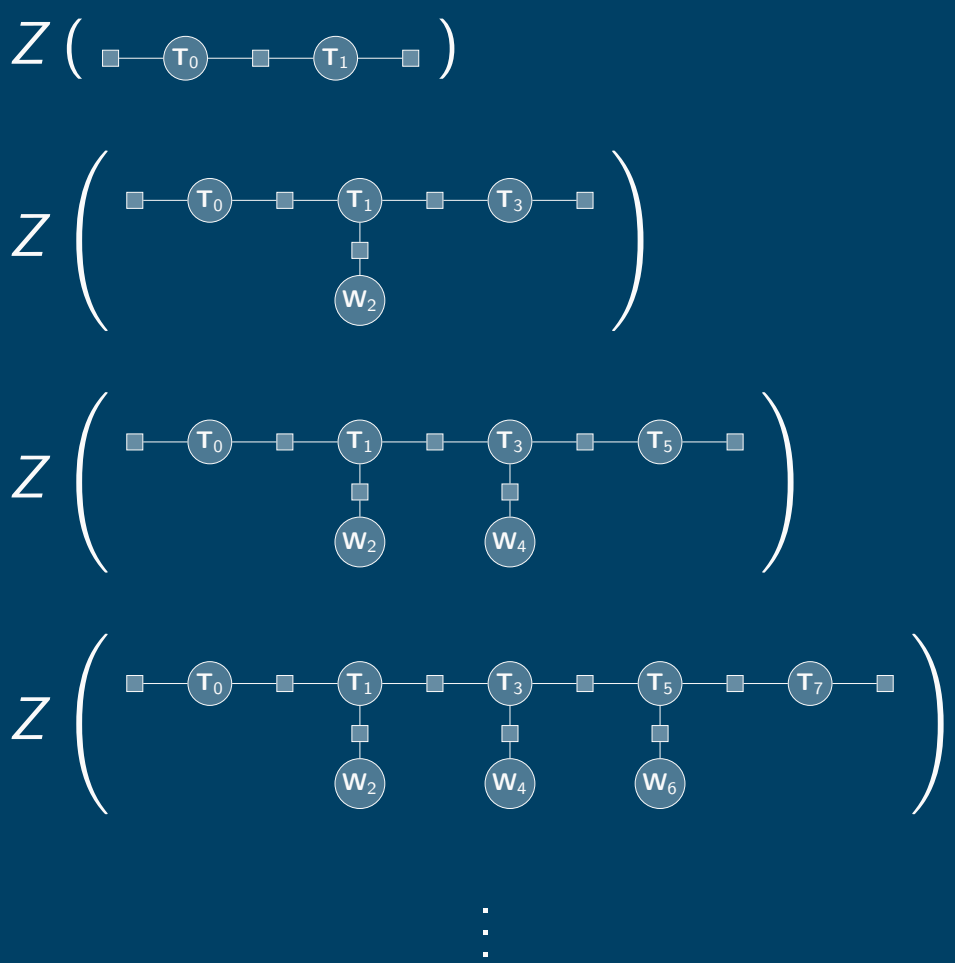
To convert weight to probability: divide by sum-product Z

Sum-Product in a Factor Graph Grammar

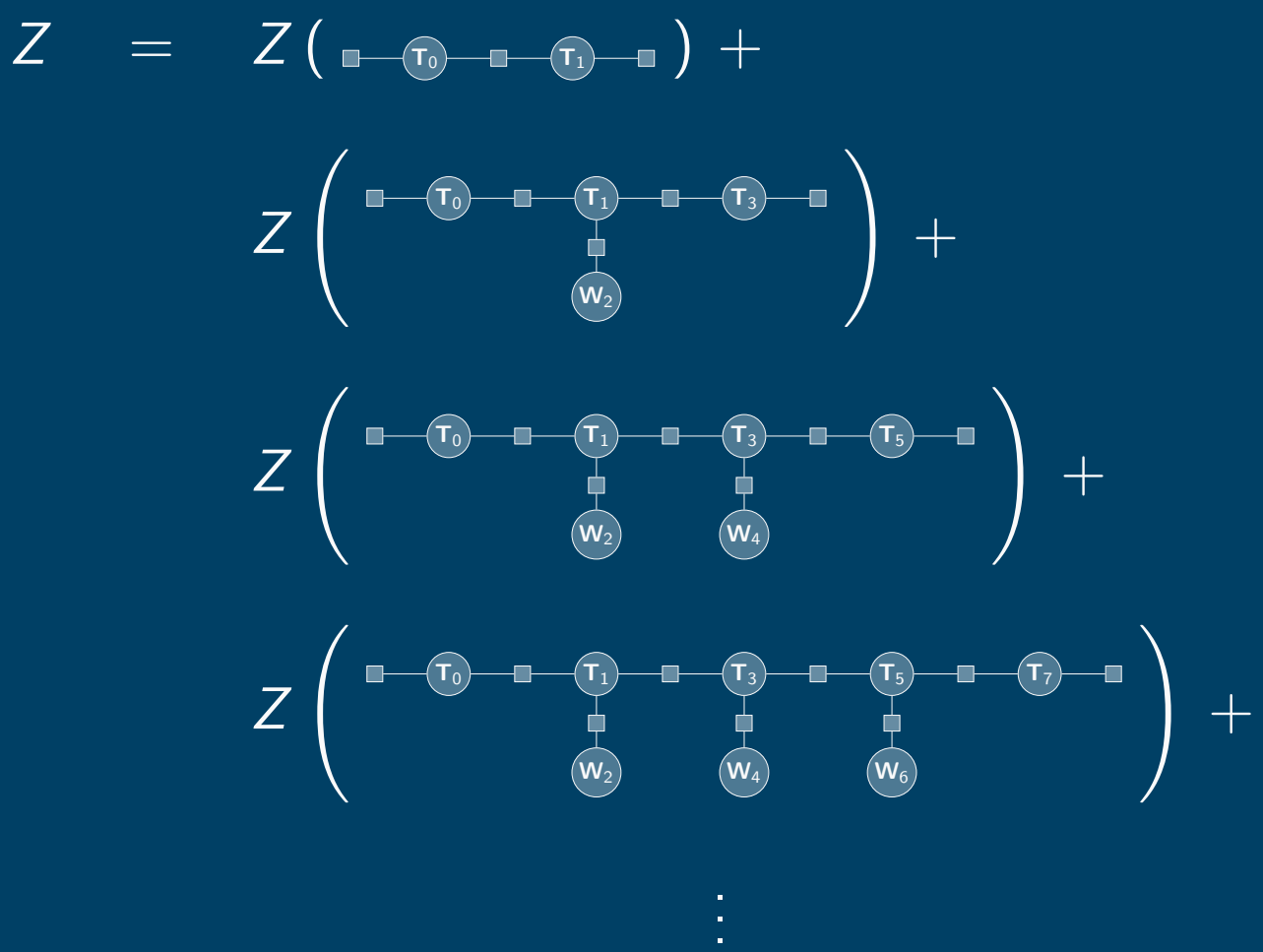
Sum-Product in a Factor Graph Grammar



Sum-Product in a Factor Graph Grammar



Sum-Product in a Factor Graph Grammar



To compute sum-product of factor graph grammar: sum over the sum-products of all derivations

Inference

Inference

3 cases:

- **Infinite graph language, finite variable domains:**
  use extension of variable elimination
- **Finite graph language, infinite variable domains:**
  convert to one big factor graph
- **Infinite graph language, infinite variable domains:**
  undecideable

Inference: Extending Variable Elimination

Can have infinite graphs, must have finite variable domains

Inference: Extending Variable Elimination

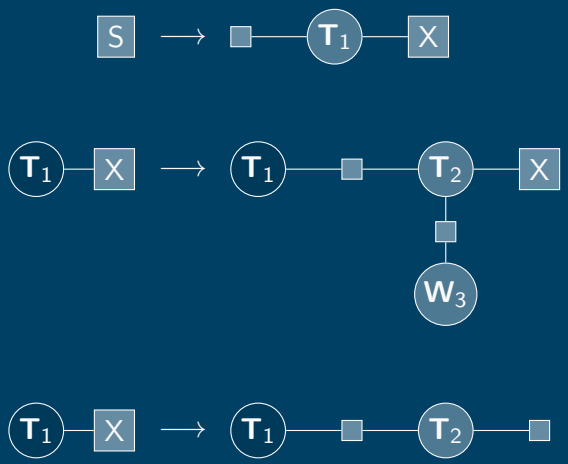
Can have infinite graphs, must have finite variable domains

Write down a system of equations that lets us solve for Z:

Inference: Extending Variable Elimination

Can have infinite graphs, must have finite variable domains

Write down a system of equations that lets us solve for Z:

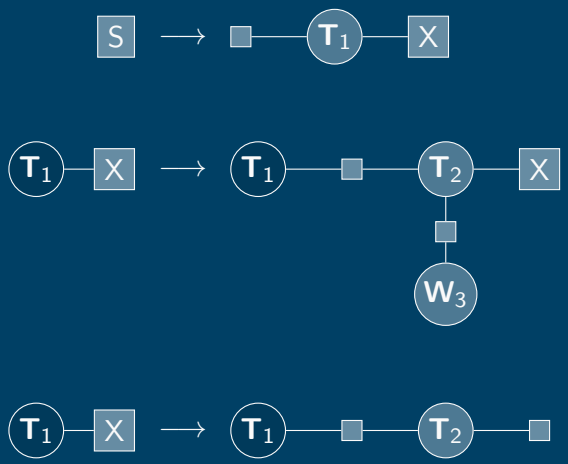


Inference: Extending Variable Elimination

Can have infinite graphs, must have finite variable domains

Write down a system of equations that lets us solve for Z:

- One eq. for each left-hand side + assignment to ext. nodes

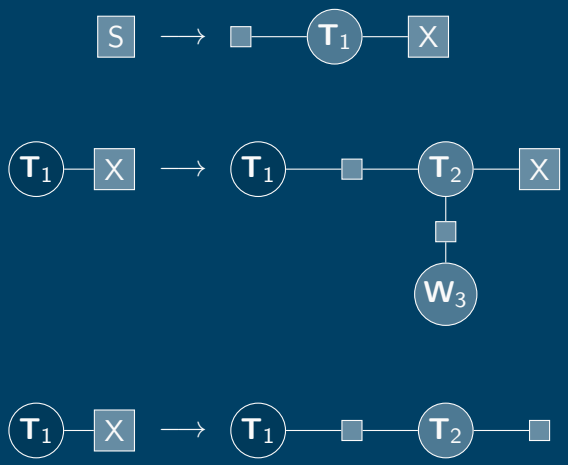


Inference: Extending Variable Elimination

Can have infinite graphs, must have finite variable domains

Write down a system of equations that lets us solve for Z:

- One eq. for each left-hand side + assignment to ext. nodes
- One eq. for each right-hand side + assignment to ext. nodes

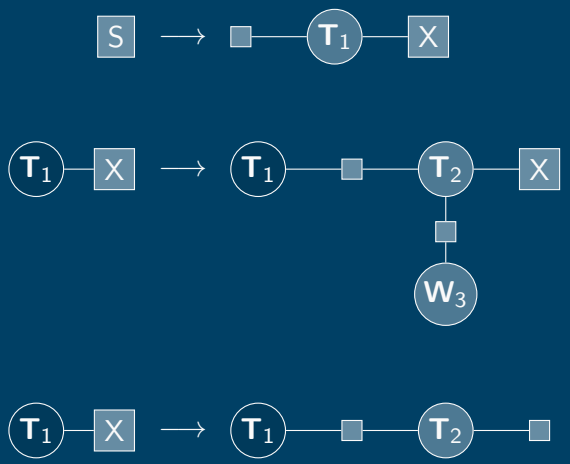


Inference: Extending Variable Elimination

Can have infinite graphs, must have finite variable domains

Write down a system of equations that lets us solve for Z:

- One eq. for each left-hand side + assignment to ext. nodes
- One eq. for each right-hand side + assignment to ext. nodes



$$Z\left(\boxed{\text{S}}\right) = Z\left(\square\text{—}\boxed{\text{T}_1}\text{—}\boxed{\text{X}}\right)$$

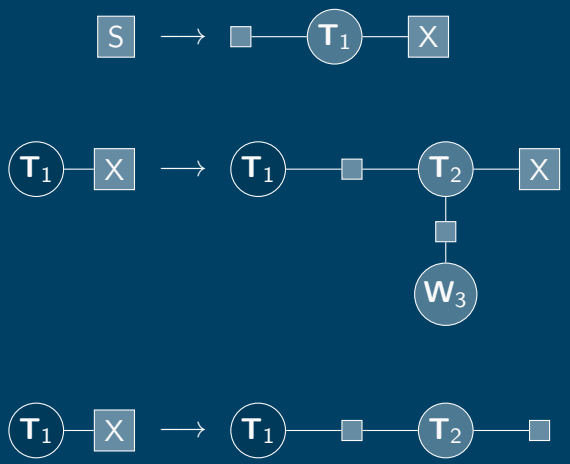
$$Z\left(\boxed{t_1}\text{—}\boxed{\text{X}}\right) = Z\left(\boxed{t_1}\text{—}\square\text{—}\boxed{\text{T}_2}\text{—}\boxed{\text{X}}\;;\;\boxed{\text{T}_2}\text{—}\square\text{—}\boxed{\text{W}_3}\right) + Z\left(\boxed{t_1}\text{—}\square\text{—}\boxed{\text{T}_2}\text{—}\square\right)$$

Inference: Extending Variable Elimination

Can have infinite graphs, must have finite variable domains

Write down a system of equations that lets us solve for Z:

- One eq. for each left-hand side + assignment to ext. nodes
- One eq. for each right-hand side + assignment to ext. nodes



$$Z\left(\square - \mathbf{T}_1 - \mathbf{X}\right) = \sum_{t_1 \in \mathbf{T}_1} \square - t_1 \times Z\left(t_1 - \mathbf{X}\right)$$

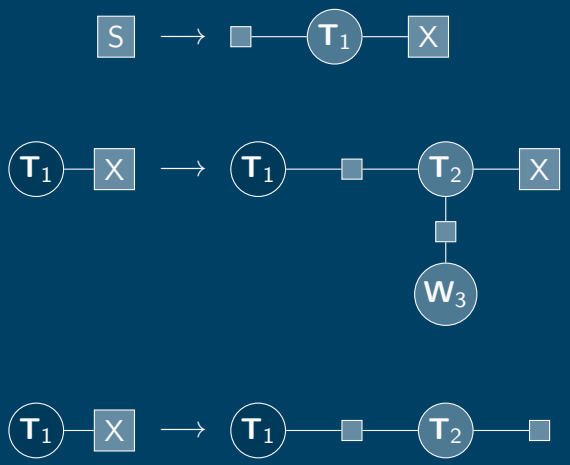
$$Z\left(t_1 - \square - \mathbf{T}_2 - \square\right) = \sum_{t_2 \in \mathbf{T}_2} t_1 - \square - t_2 \times t_2 - \square$$

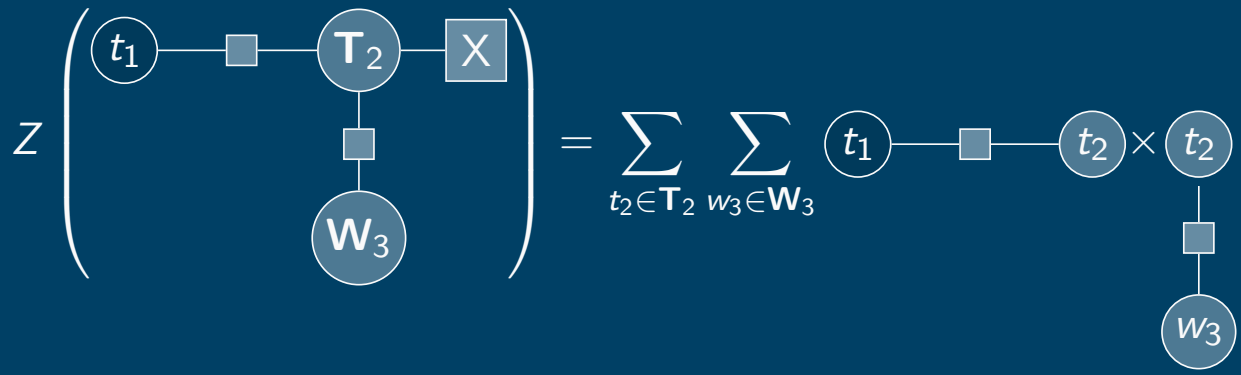
Inference: Extending Variable Elimination

Can have infinite graphs, must have finite variable domains

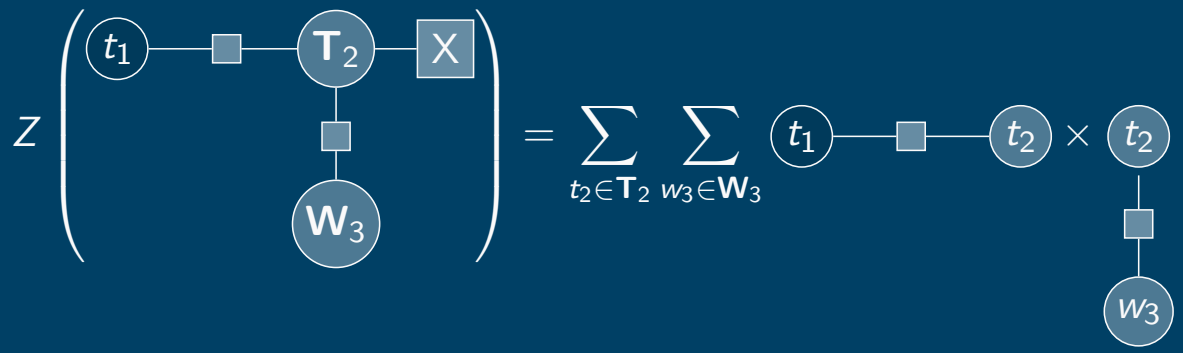
Write down a system of equations that lets us solve for Z:

- One eq. for each left-hand side + assignment to ext. nodes
- One eq. for each right-hand side + assignment to ext. nodes





Computational Complexity



Number of equations: $O(|G|m^k)$

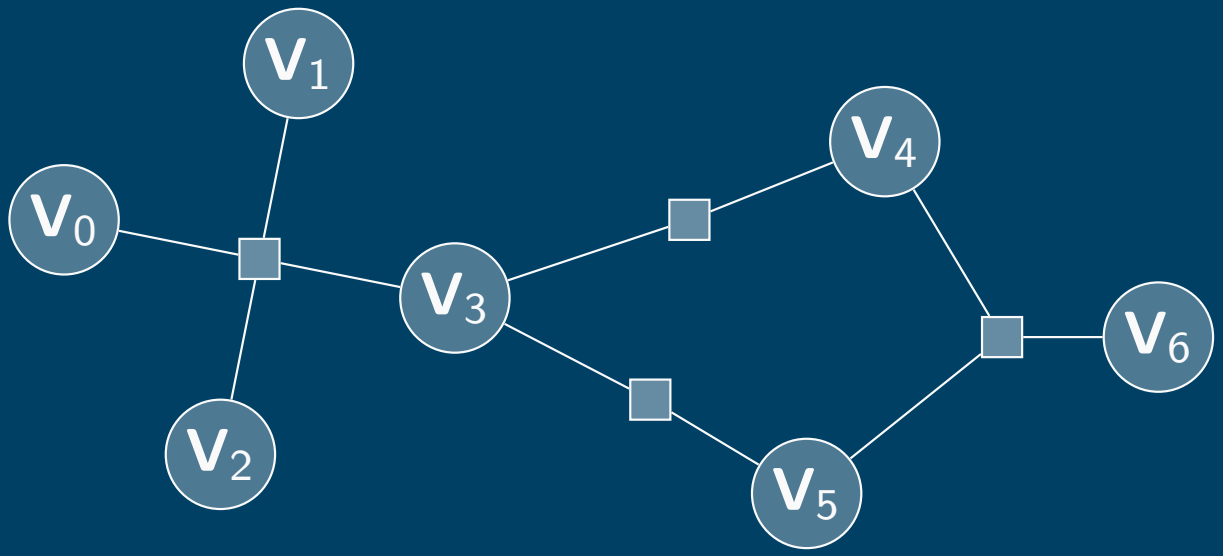
- $|G|$: number of rules
- m: max size of any variable domain
- k: max # of nodes in RHS

Cases:

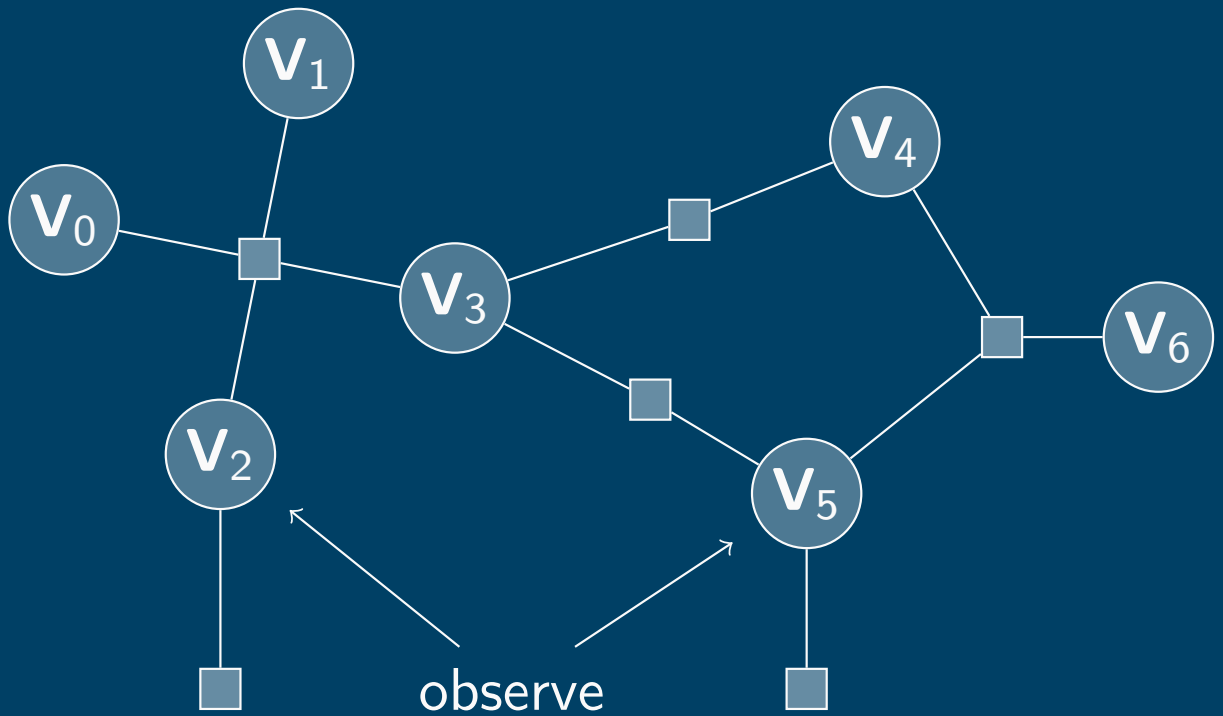
- **Non-recursive:** compute one at a time, $O(|G|m^k)$
- **Linear:** solve system of linear equations, $O(|G|^3 m^{3(k+1)})$
- **Otherwise:** solve iteratively

Querying a Factor Graph Grammar

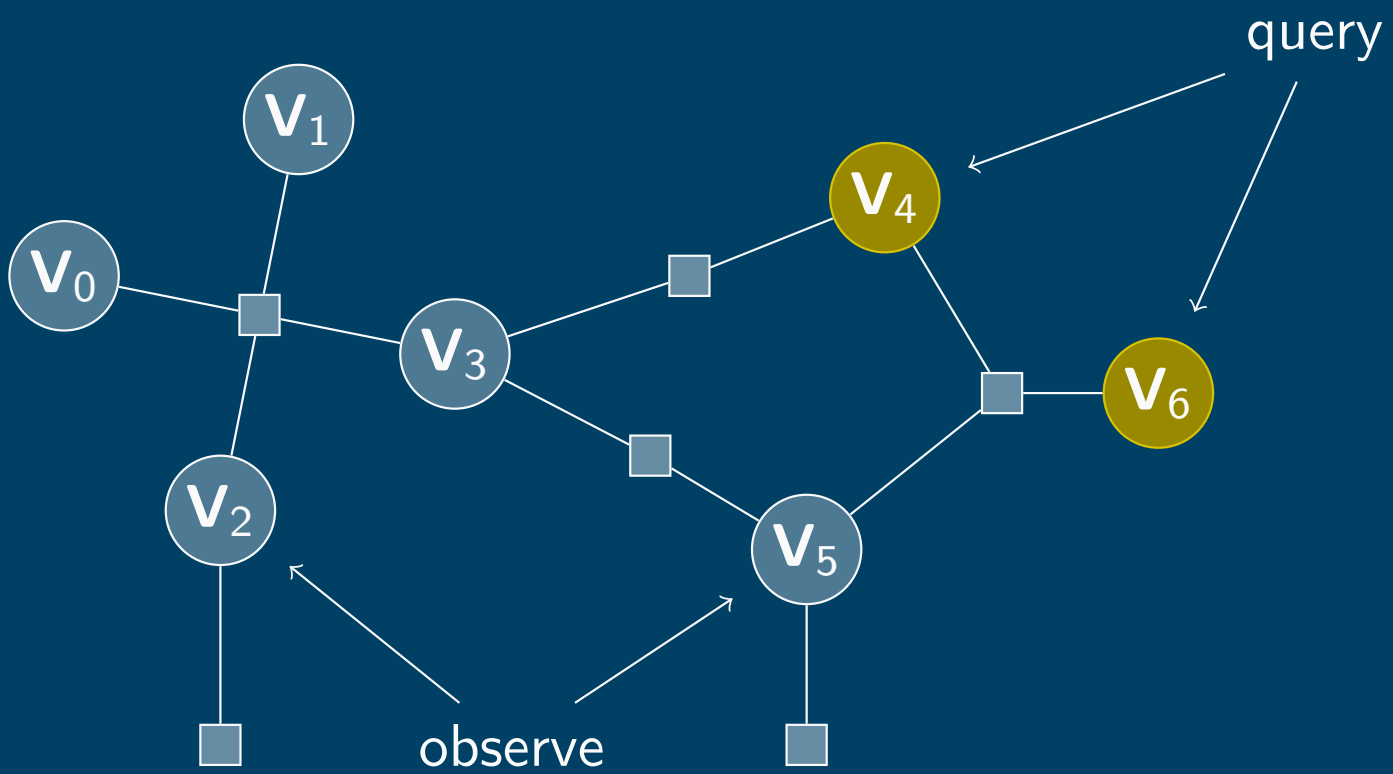
Queries



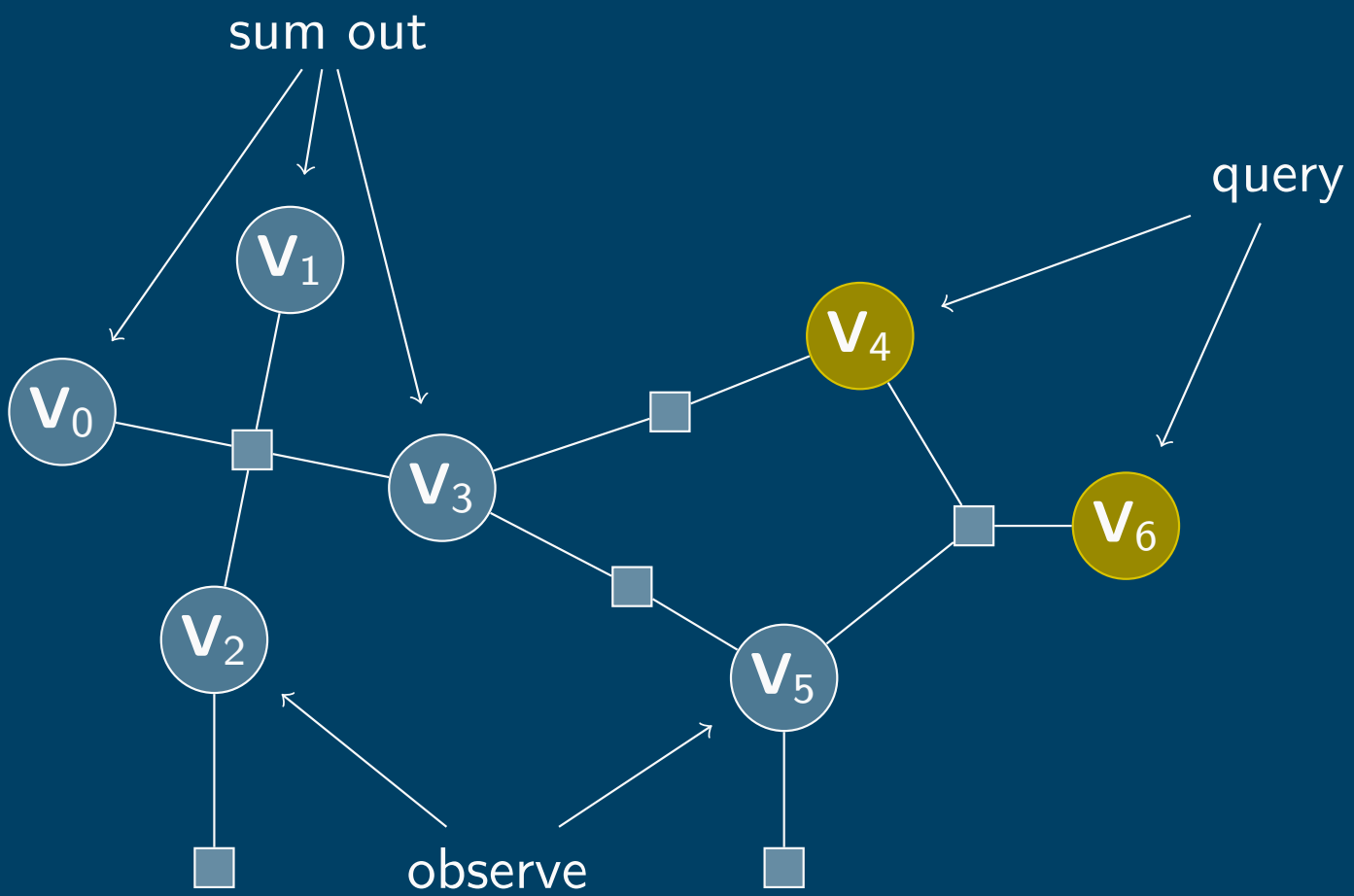
Queries



Queries

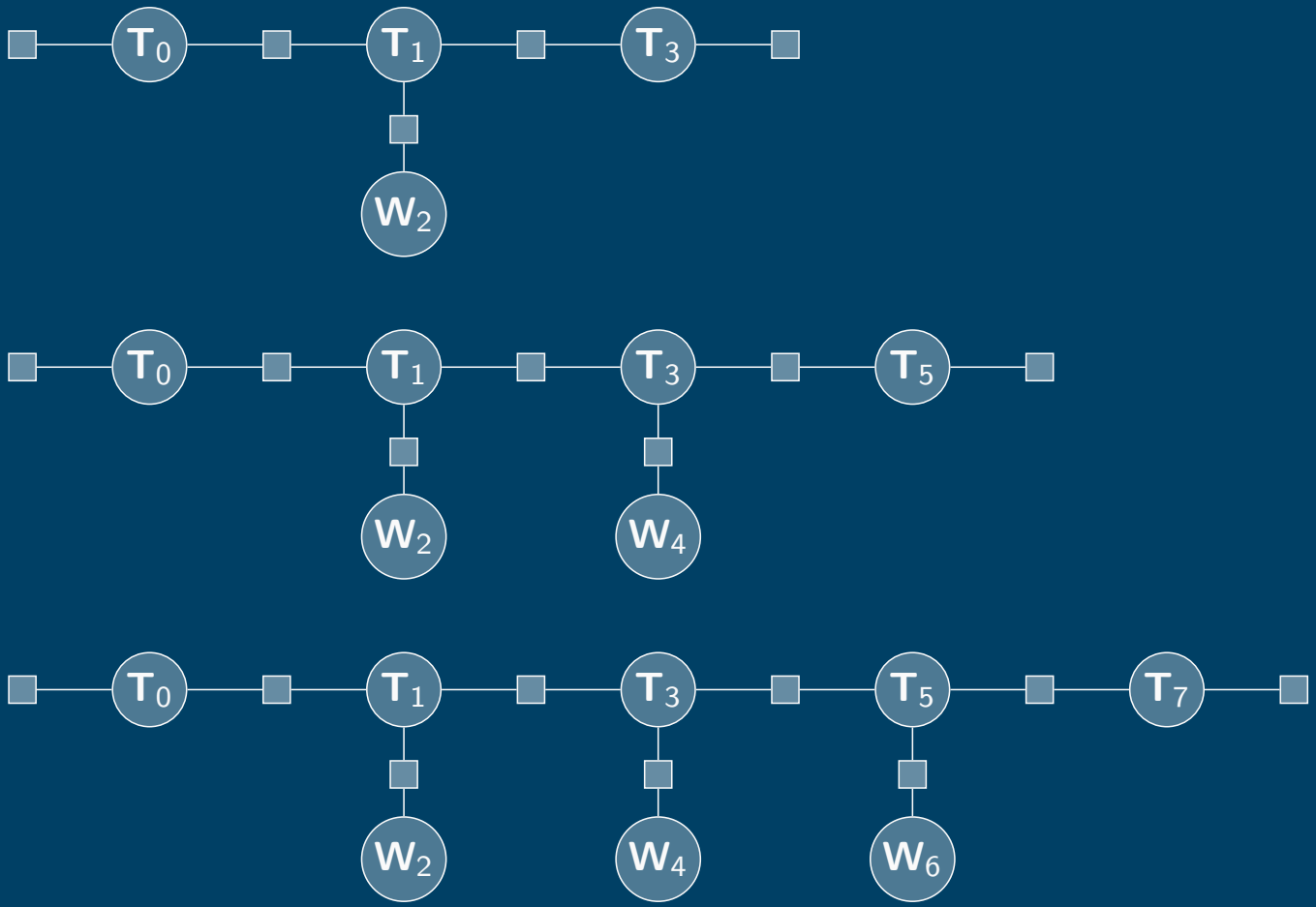


Queries



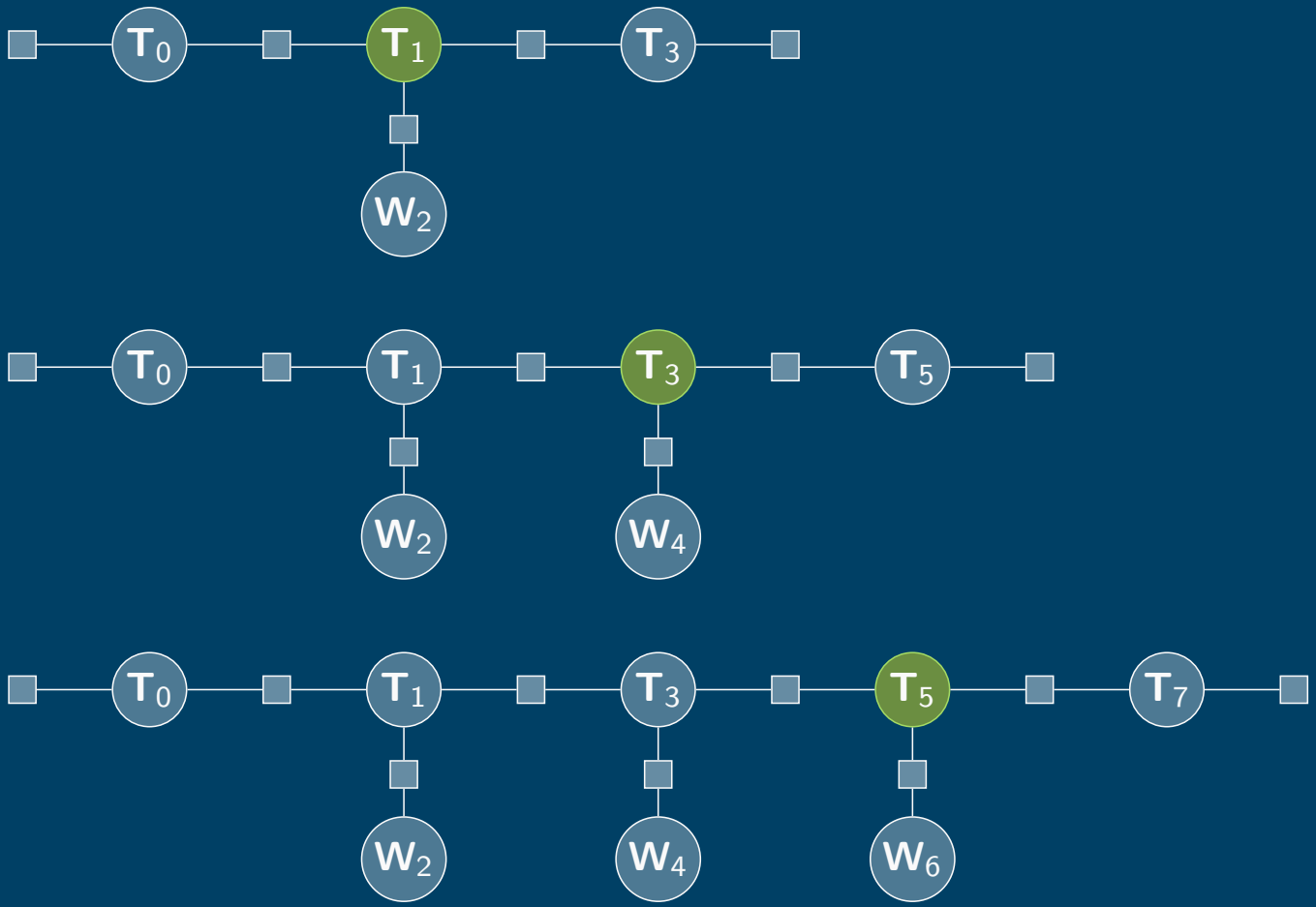
Queries

How to observe a value for the second-to-last tag?



Queries

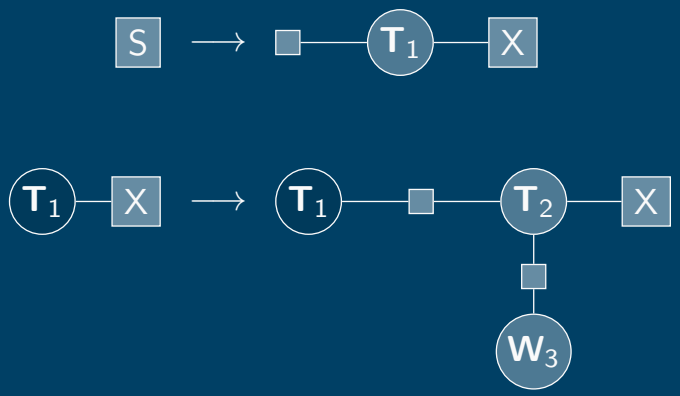
How to observe a value for the second-to-last tag?



Queries

Instead of identifying variables in the derived graph,

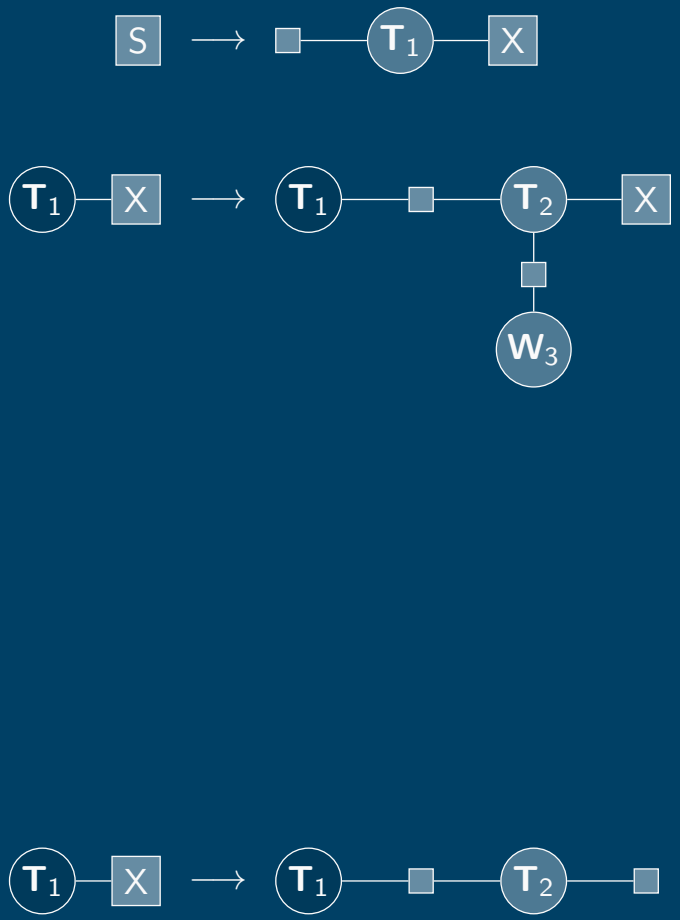
identify variables in the grammar

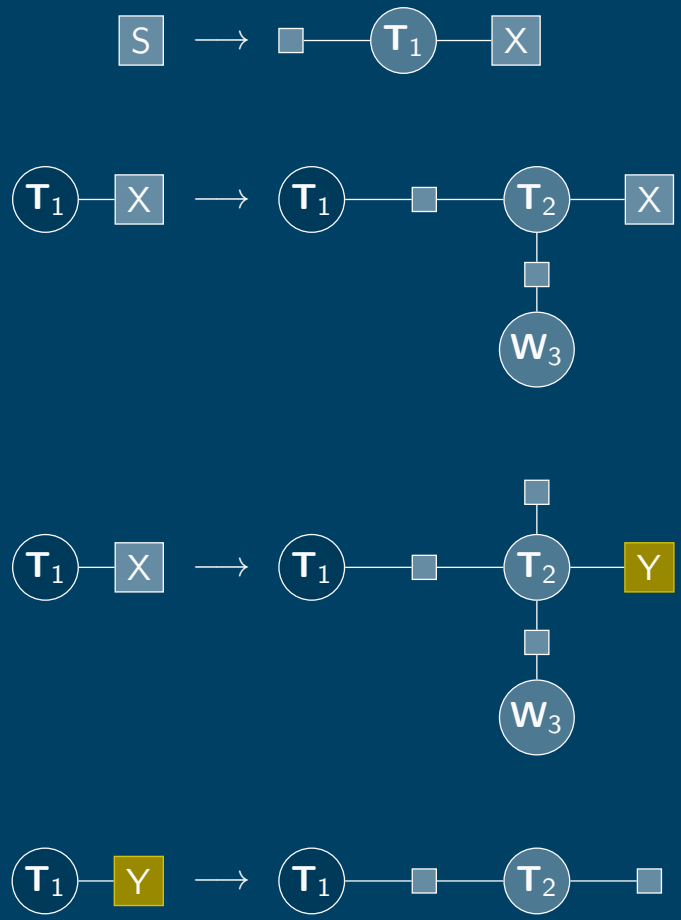


Queries

Instead of identifying variables in the derived graph,

identify variables in the grammar

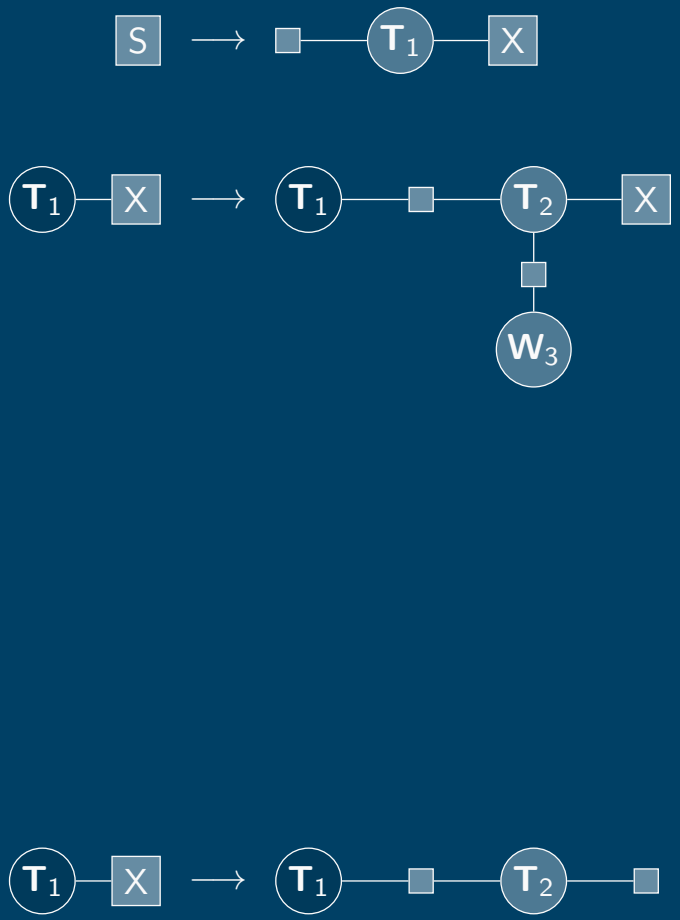




Queries

Instead of identifying variables in the derived graph,

identify variables in the grammar



Conjunction

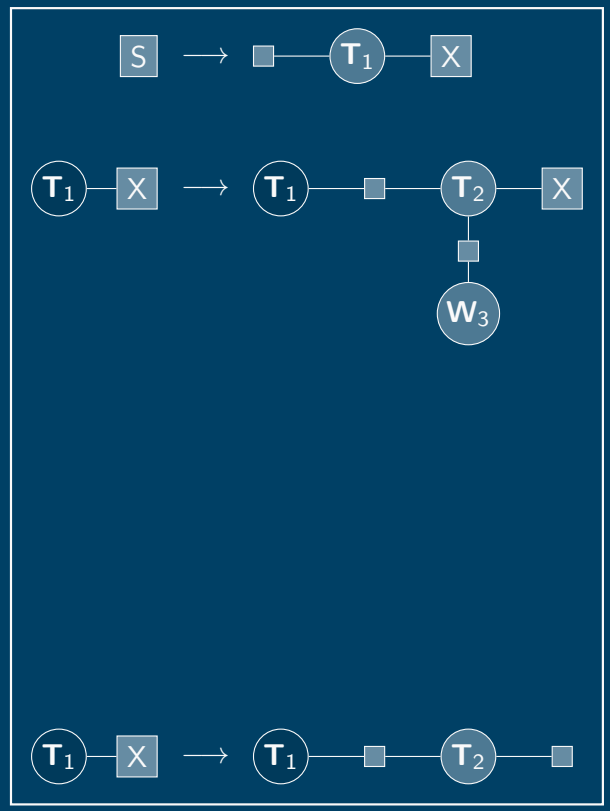
Conjunction lets you modularize grammar into two halves:

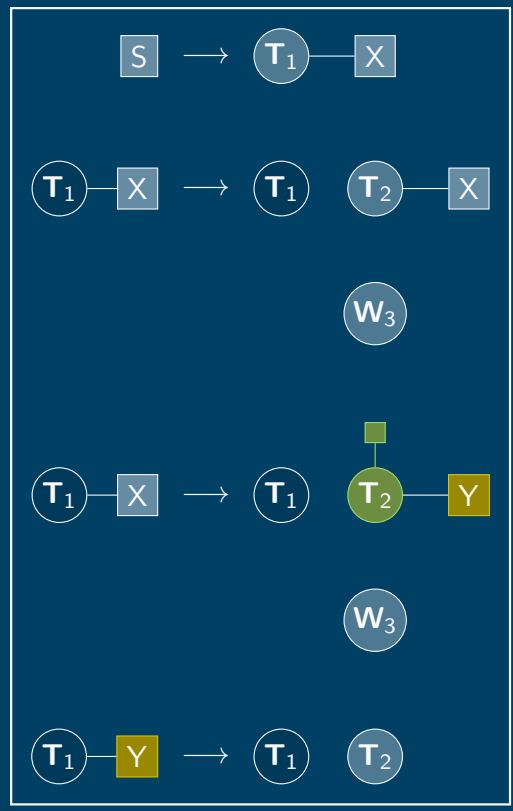
the original grammar and the query grammar

Conjunction

Conjunction lets you modularize grammar into two halves:

the original grammar and the query grammar

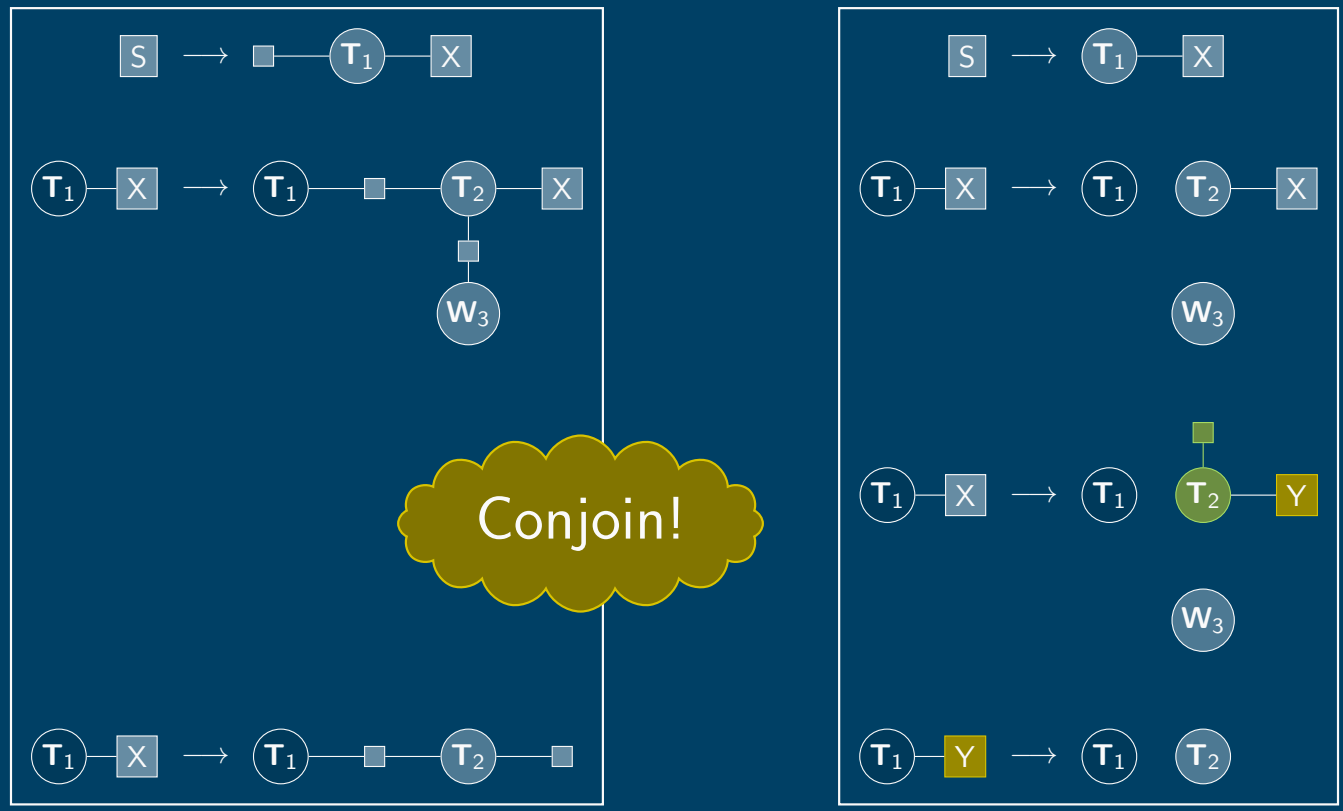




Conjunction

Conjunction lets you modularize grammar into two halves:

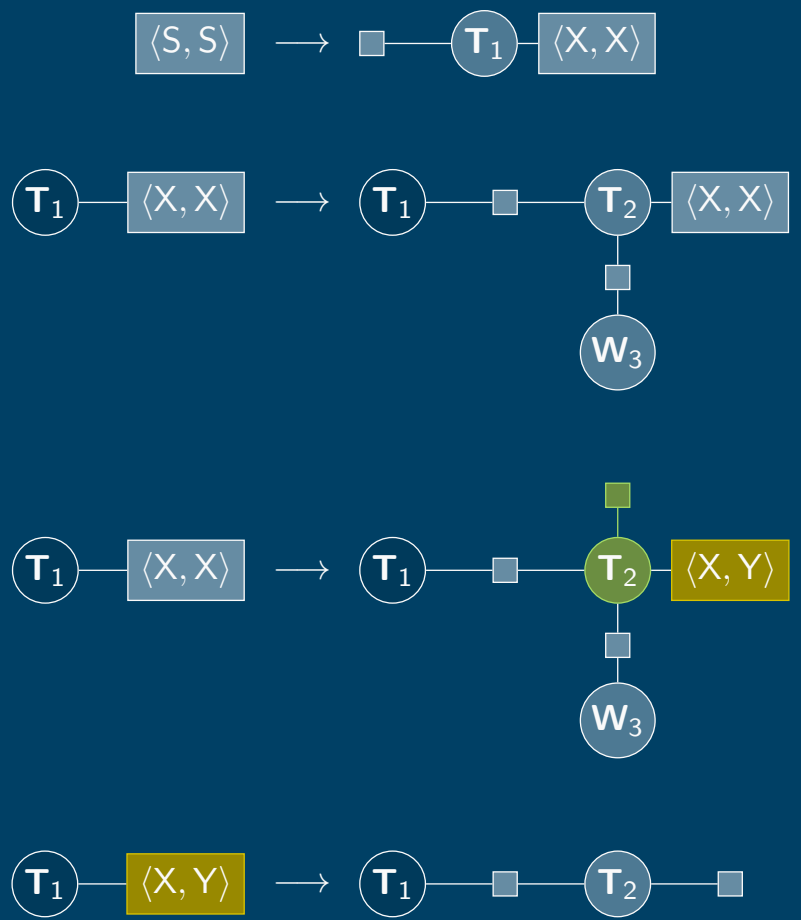
the original grammar and the query grammar



Conjunction

Conjunction lets you modularize grammar into two halves:

the original grammar and the query grammar



Related Work

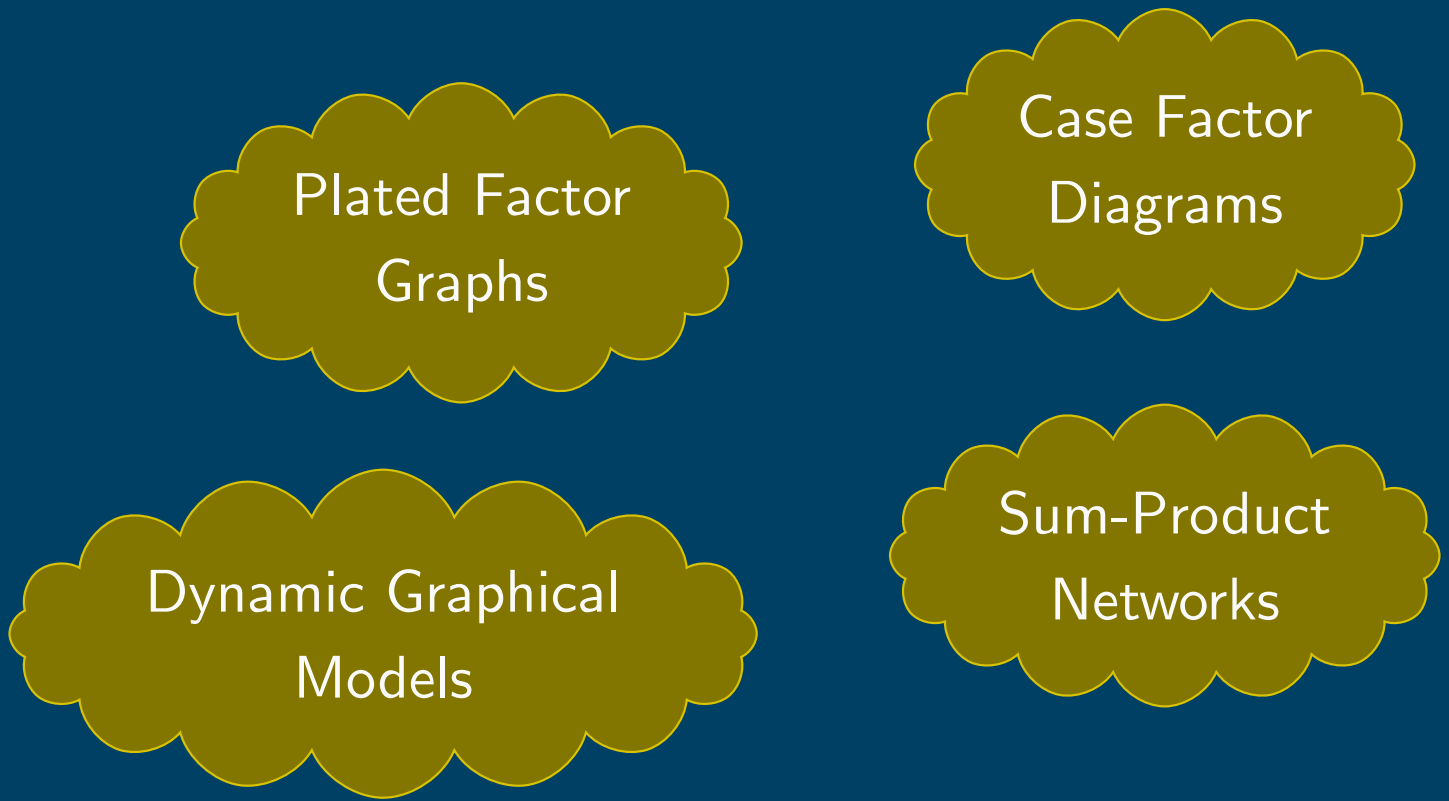
Related Work

Lots of previous work generalizing PGMs.

Related Work

Lots of previous work generalizing PGMs.

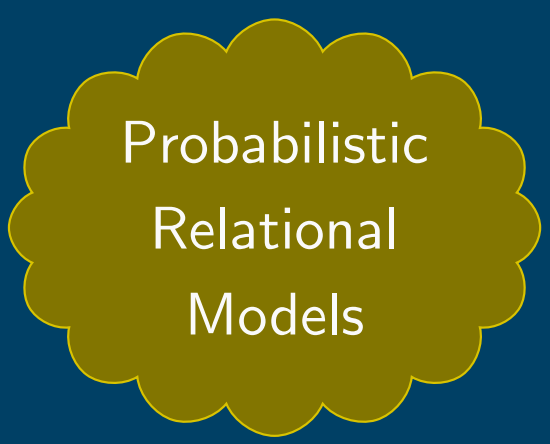
Tractable formalisms:

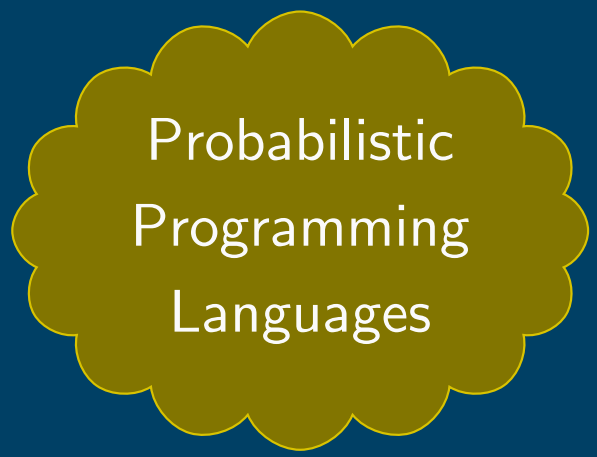


Related Work

Lots of previous work generalizing PGMs.

Expressive formalisms:





Related Work

Lots of previous work generalizing PGMs.

Factor graph grammars:

- General enough to subsume the tractable formalisms
- Simple enough to have tractable inference in many important cases

Ongoing and Future Work

Translating Recursive Probabilistic Programs to Factor Graph Grammars (PROBPROG 2020)

David Chiang
dchiang@nd.edu

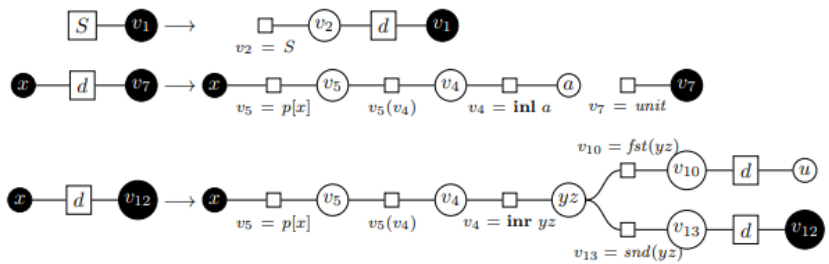
Chung-chieh Shan
ccshan@indiana.edu

Chiang, David, and Chung-chieh Shan. "Translating Recursive Probabilistic Programs to Factor Graph Grammars." *arXiv preprint arXiv:2010.12071* (2020).

Translating Recursive Probabilistic Programs to Factor Graph Grammars (PROBPROG 2020)

```
# sample a tree from a PCFG
fun d(x) =
  case sample p[x] of
    inl a =>
      unit
  | inr yz =>
      let u = d(fst(yz)) in
      d(snd(yz));
d(S)
```

translates to



Future Work

- Implementation
- Approximate inference
- Better ways to query FGGs
- Automatic structure learning for FGG rules

Thank you!

References

Michel Bauderon and Bruno Courcelle. 1987. Graph expressions and graph rewriting. *Mathematical Systems Theory*, 20:83–127.

Annegret Habel and Hans-Jörg Kreowski. 1987. May we introduce to you: Hyperedge replacement. In *Proc. Third International Workshop on Graph Grammars and Their Application to Computer Science*, pages 15–26.