Logic-Based Parsing with Neural Networks

Compositional Models of Vector-based Semantics

Konstantinos Kogkalidis
Utrecht Institute of Linguistics OTS, Utrecht University

ESSLLI, August 2022, Galway

☮ Certified

no military applications/funding

The agenda

- exploring Æthel
- parsing with graph learning machinery

Types

ILL$_{\multimap}$ plus $\Diamond, \Box$ modalities for *dependency domain demarkation*.

$\mathbb{T}$ inductively defined as:

$$\mathbb{T} := A \mid T \multimap T \mid \Diamond^d T \mid \Box^d T \qquad A \in \mathbb{A}, T \in \mathbb{T}$$

- $\mathbb{A}$ – closed set of base types
- $\multimap$ – linear function builder
- $\Diamond$ – reserved for "necessary arguments", i.e. complements
- $\Box$ – reserved for "optional functors", i.e. adjuncts

Types

ILL$_{\multimap}$ plus $\diamondsuit$, $\Box$ modalities for *dependency domain demarkation.*

$\mathbb{T}$ inductively defined as:

$$\mathbb{T} := A \mid T \multimap T \mid \diamondsuit^d T \mid \Box^d T \qquad A \in \mathbb{A},\, T \in \mathbb{T}$$

- $\mathbb{A}$ – closed set of base types
- $\multimap$ – linear function builder
- $\diamondsuit$ – reserved for "necessary arguments", i.e. complements
- $\Box$ – reserved for "optional functors", i.e. adjuncts

Types

$\text{ILL}_{\multimap}$ plus $\diamond$, $\square$ modalities for *dependency domain demarkation.*

$\mathbb{T}$ inductively defined as:

$$\mathbb{T} := A \mid T \multimap T \mid \diamond^d T \mid \square^d T \qquad A \in \mathbb{A}, T \in \mathbb{T}$$

- $\mathbb{A}$ – closed set of base types
- $\multimap$ – linear function builder
- $\diamond$ – reserved for ”necessary arguments”, i.e. complements
- $\square$ – reserved for ”optional functors”, i.e. adjuncts

Types

$\mathrm{ILL}_{\multimap}$ plus $\diamondsuit, \square$ modalities for *dependency domain demarkation*.

$\mathbb{T}$ inductively defined as:

$$\mathbb{T} := A \mid T \multimap T \mid \diamondsuit^{d} T \mid \square^{d} T \qquad A \in \mathbb{A}, T \in \mathbb{T}$$

- $\mathbb{A}$ – closed set of base types
- $\multimap$ – linear function builder
- $\diamondsuit$ – reserved for ”necessary arguments”, i.e. complements
- $\square$ – reserved for ”optional functors”, i.e. adjuncts

Rules & Terms

- 😃 function/argument structures

$$\frac{}{\mathtt{c} : T \vdash \mathtt{c} : T}\ Lex \qquad \frac{\Gamma \vdash \mathtt{s} : T_1 \multimap T_2 \quad \Delta \vdash \mathtt{t} : T_1}{\Gamma, \Delta \vdash \mathtt{s}\ \mathtt{t} : T_2} \multimap E$$

Rules & Terms

- 😃 function/argument structures

$$\frac{}{\texttt{c} : T \vdash \texttt{c} : T}\ Lex \qquad \frac{\Gamma \vdash \texttt{s} : T_1 \multimap T_2 \quad \Delta \vdash \texttt{t} : T_1}{\Gamma, \Delta \vdash \texttt{s}\ \texttt{t} : T_2} \multimap E$$

- ☺ simple dependency demarkation

$$\frac{\Gamma \vdash \texttt{t} : T}{\langle \Gamma \rangle^d \vdash \triangle^d\, \texttt{t} : \Diamond^d T}\ \Diamond^d I \qquad \frac{\Gamma \vdash \texttt{s} : \Box^d T}{\langle \Gamma \rangle^d \vdash \blacktriangledown^d \texttt{s} : T}\ \Box^d E$$

Rules & Terms

😃 function/argument structures

$$\frac{}{\mathtt{c} : T \vdash \mathtt{c} : T}\ Lex \qquad \frac{\Gamma \vdash \mathtt{s} : T_1 \multimap T_2 \quad \Delta \vdash \mathtt{t} : T_1}{\Gamma, \Delta \vdash \mathtt{s}\ \mathtt{t} : T_2} \multimap E$$

🙂 simple dependency demarkation

$$\frac{\Gamma \vdash \mathtt{t} : T}{\langle \Gamma \rangle^d \vdash \triangle^d\ \mathtt{t} : \Diamond^d T}\ \Diamond^d I \qquad \frac{\Gamma \vdash \mathtt{s} : \Box^d T}{\langle \Gamma \rangle^d \vdash \blacktriangledown^d \mathtt{s} : T}\ \Box^d E$$

😐 hypothetical reasoning

$$\frac{}{\mathtt{x} : T \vdash \mathtt{x} : T}\ Ax \qquad \frac{\Gamma, \mathtt{x} : T_1 \vdash \mathtt{s} : T_2}{\Gamma \vdash \lambda \mathtt{x}.\mathtt{s} : T_1 \multimap T_2} \multimap I$$

Rules & Terms

😃 function/argument structures

$$\frac{}{\mathtt{c} : T \vdash \mathtt{c} : T}\ Lex \qquad \frac{\Gamma \vdash \mathtt{s} : T_1 \multimap T_2 \quad \Delta \vdash \mathtt{t} : T_1}{\Gamma, \Delta \vdash \mathtt{s}\ \mathtt{t} : T_2} \multimap E$$

🙂 simple dependency demarkation

$$\frac{\Gamma \vdash \mathtt{t} : T}{\langle \Gamma \rangle^d \vdash \triangle^d\, \mathtt{t} : \Diamond^d T}\ \Diamond^d I \qquad \frac{\Gamma \vdash \mathtt{s} : \Box^d T}{\langle \Gamma \rangle^d \vdash \blacktriangledown^d \mathtt{s} : T}\ \Box^d E$$

😐 hypothetical reasoning

$$\frac{}{\mathtt{x} : T \vdash \mathtt{x} : T}\ Ax \qquad \frac{\Gamma, \mathtt{x} : T_1 \vdash \mathtt{s} : T_2}{\Gamma \vdash \lambda \mathtt{x}.\mathtt{s} : T_1 \multimap T_2} \multimap I$$

😱 cosmic horror from the great beyond

$$\frac{\langle \Gamma \rangle^d \vdash \mathtt{s} : T}{\Gamma \vdash \blacktriangle^d \mathtt{s} : \Box^d T}\ \Box^d I \qquad \frac{\Gamma[\langle \mathtt{x} : T_1 \rangle^d] \vdash \mathtt{t} : T_2 \qquad \Delta \vdash \mathtt{s} : \Diamond^d T_1}{\Gamma[\Delta] \vdash \mathtt{t}[\mathtt{x} \mapsto \triangledown^d \mathtt{s}] : T_2}\ \Diamond^d E$$

Rules & Terms

Bonus:

ad-hoc extraction

$$\frac{\langle \Gamma, \langle \mathtt{x} : T_1 \rangle^{\mathsf{X}}, \Delta \rangle^{d} \vdash \mathtt{s} : T_2}{\langle \Gamma, \Delta \rangle^{d}, \langle \mathtt{x} : T_1 \rangle^{\mathsf{X}} \vdash \mathtt{s} : T_2} \; \mathsf{X}$$

Today's Example

alt-tab

Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

$$\diamondsuit^{relcl}(\diamondsuit^{x}\square^{x}\diamondsuit^{mod}\square^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$

Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

$$\Diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$

$$!(_) := \Diamond\Box(_)$$

Formula Decomposition: Types ≡ Trees

The type assignment of *“where”*:

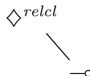
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$

$\multimap$

Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

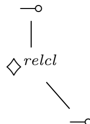
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

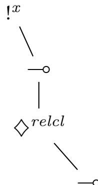
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

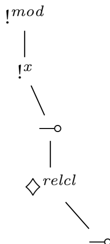
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

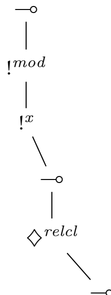
$$\diamond^{relcl}(!^{x}!^{mod}(pptart \multimap pptart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

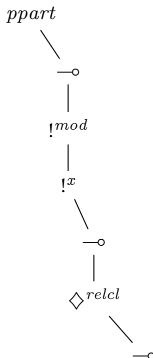
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

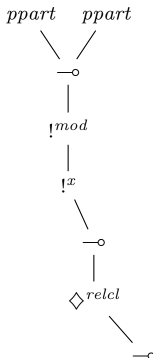
$$\diamond^{relcl}(!^{x}!^{mod}(pp art \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

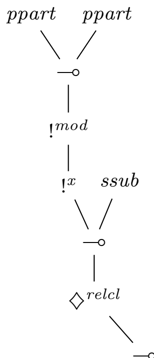
$$\diamond^{relcl}(!^{x}!^{mod}(pptart \multimap pptart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

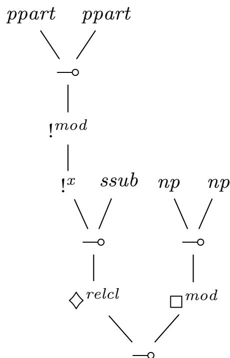
$$\diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

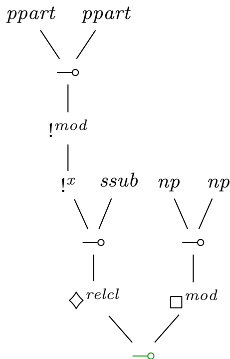
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have
- we miss

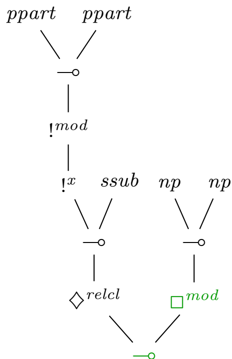


Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

- we miss

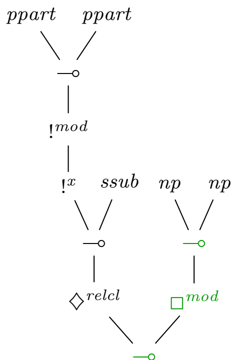


Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

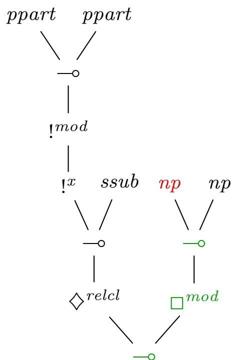
- we miss



Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

- \+ we have
- \- we miss



Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

- we miss

$ppart$ $ppart$

$\multimap$

$!^{mod}$

$!^{x}$ $ssub$ np np

$\multimap$ $\multimap$

$\Diamond^{relcl}$ $\Box^{mod}$

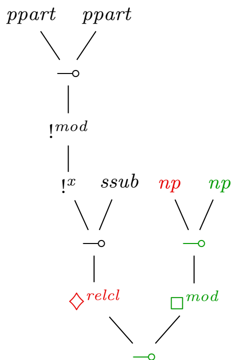
$\multimap$

Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

- we miss

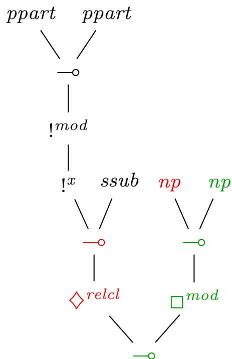


Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

- we miss

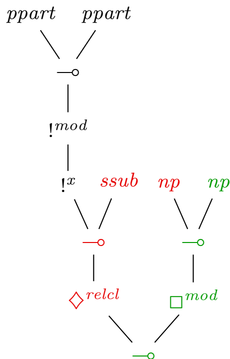


Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

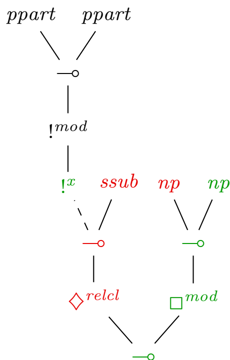
- we miss



Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

- \+ we have
- \- we miss



Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

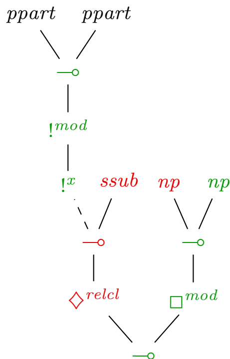
- we miss

Formula Decomposition: Polarity Induction

subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

- we miss

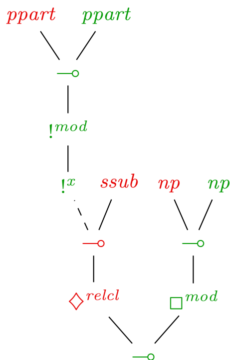


Formula Decomposition: Polarity Induction

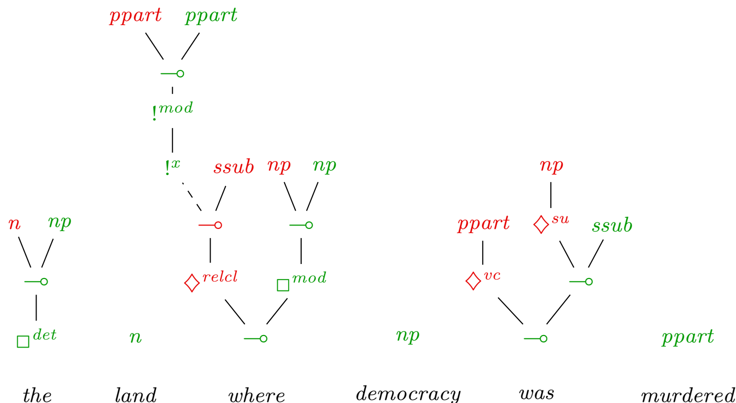
subtree polarity (*preserved* to the right, *inverted* to the left)

+ we have

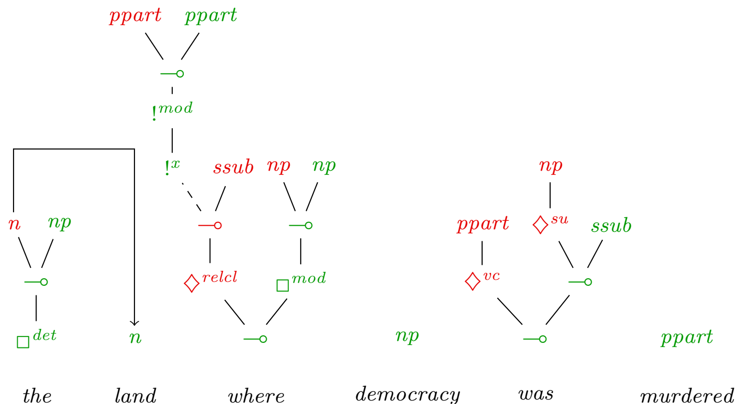
- we miss



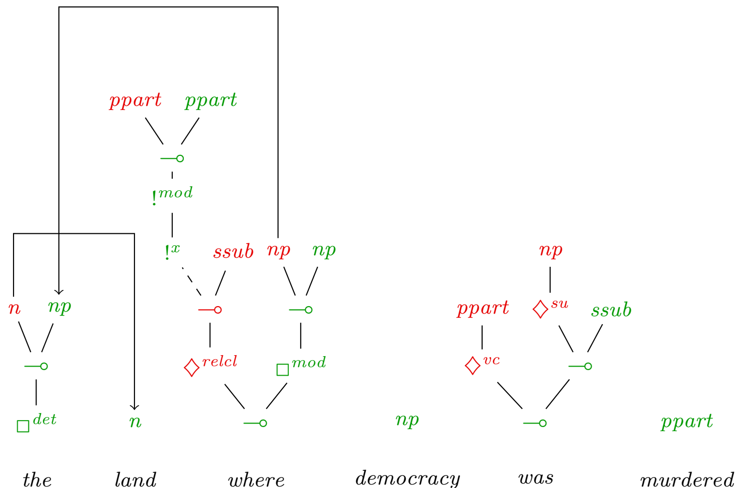
Proof Frames



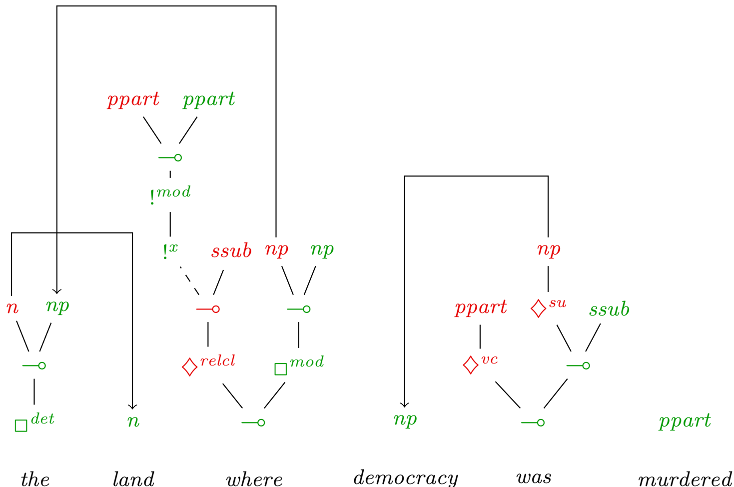
Proof Frames



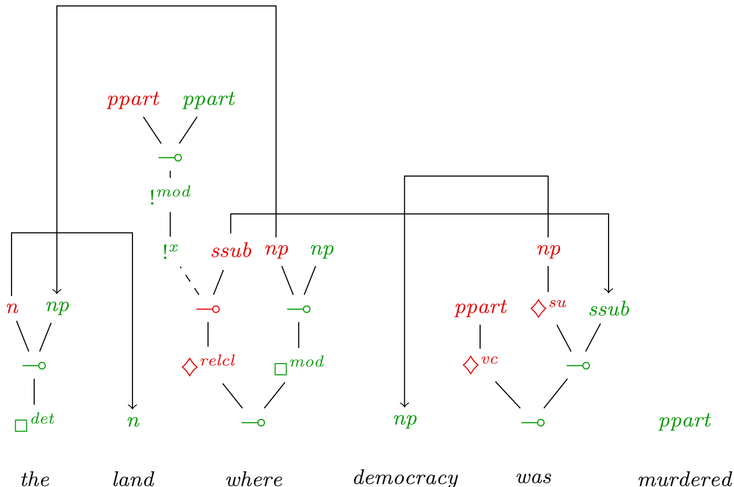
Proof Frames



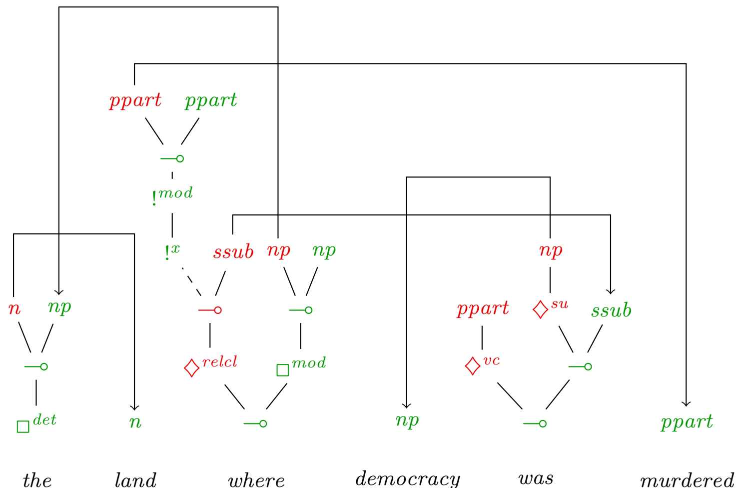
Proof Frames



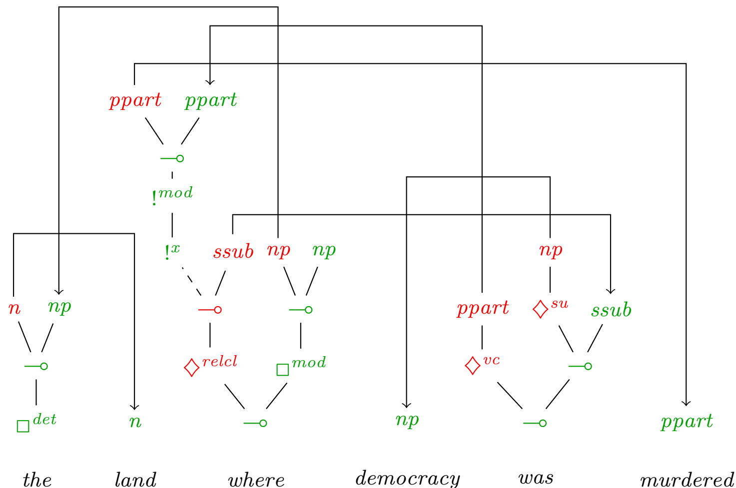
Proof Frames



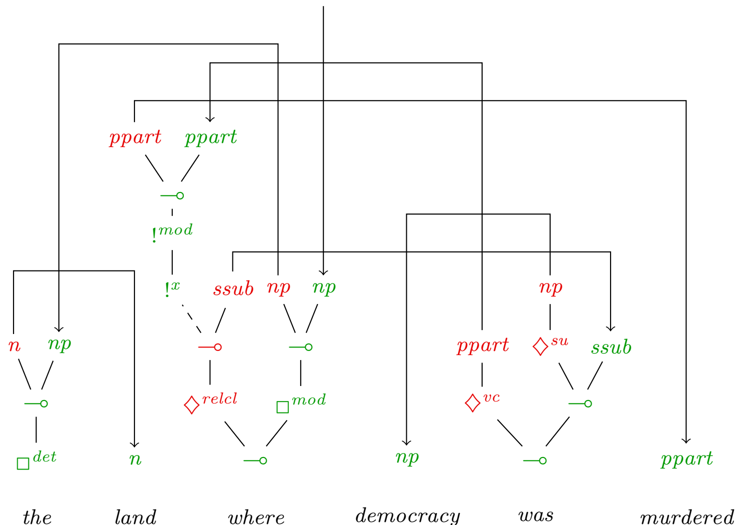
Proof Frames



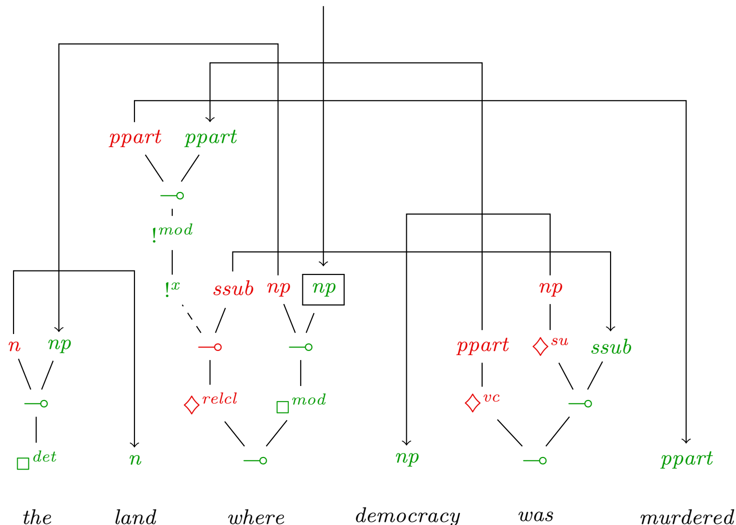
Proof Frames



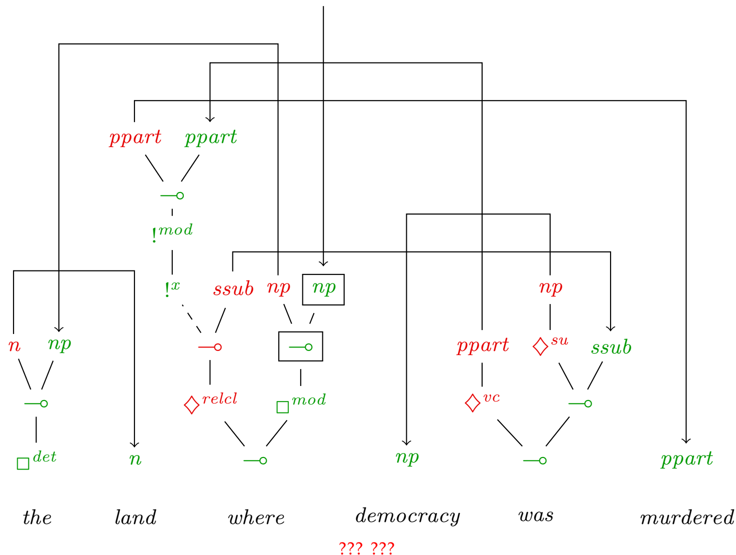
Proof Frames



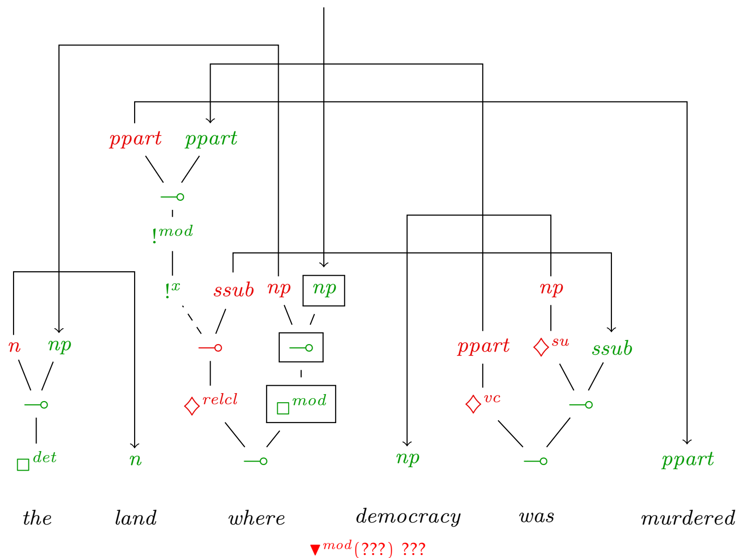
Proof ~~Frames~~ Structures



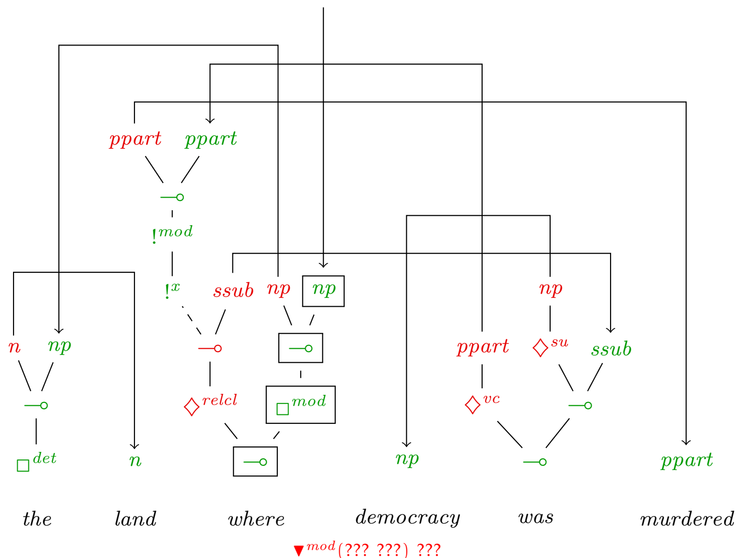
Proof ~~Frames~~ Structures



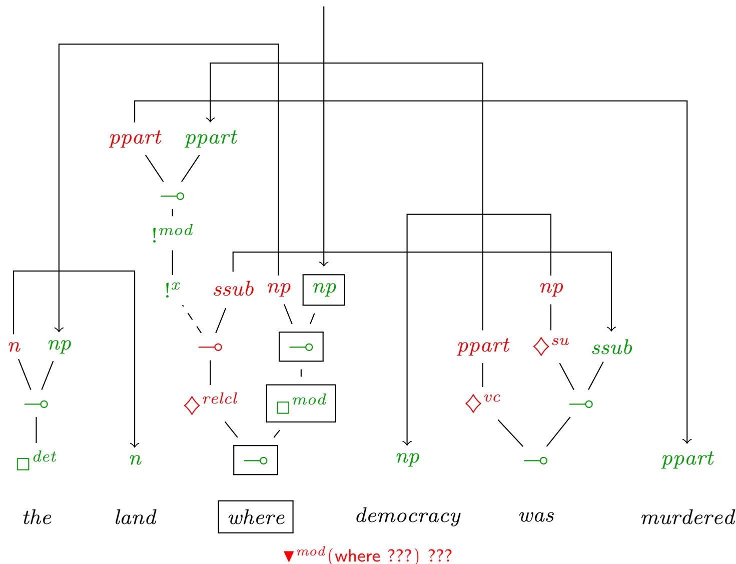
Proof ~~Frames~~ Structures



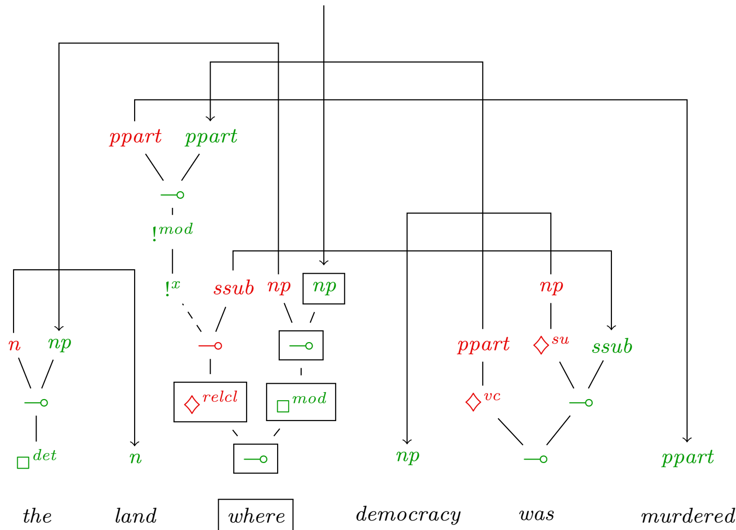
Proof ~~Frames~~ Structures



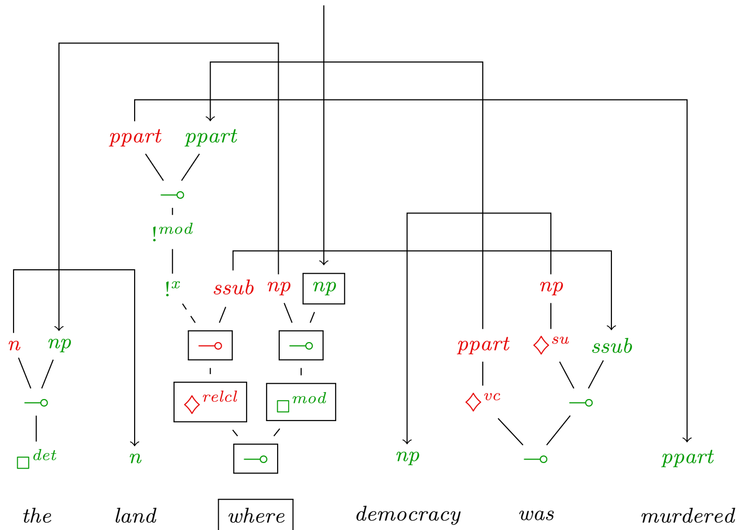
Proof ~~Frames~~ Structures



Proof ~~Frames~~ Structures

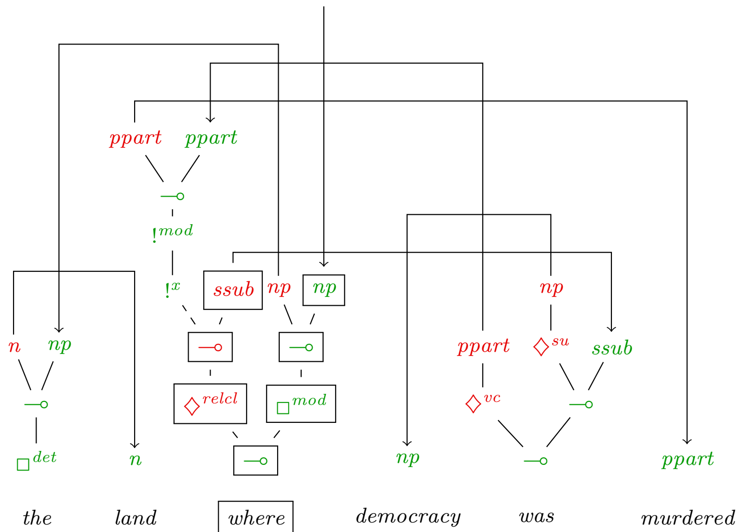


Proof ~~Frames~~ Structures

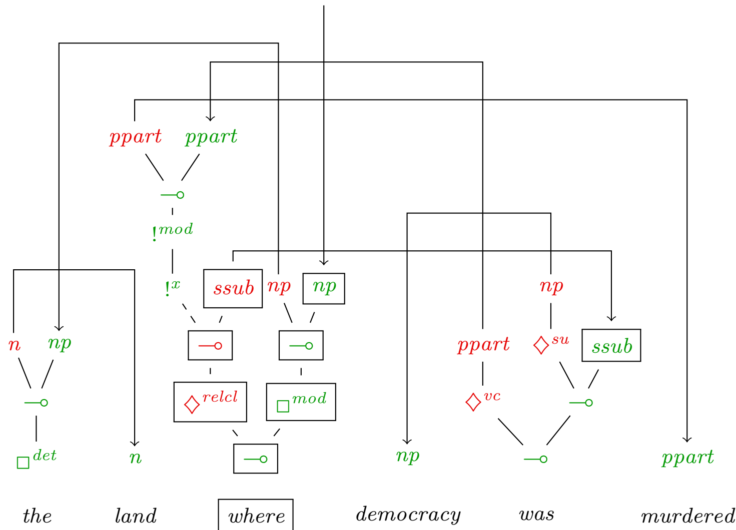


$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ $(\lambda x_0.(???)))$???

Proof ~~Frames~~ Structures



Proof ~~Frames~~ Structures



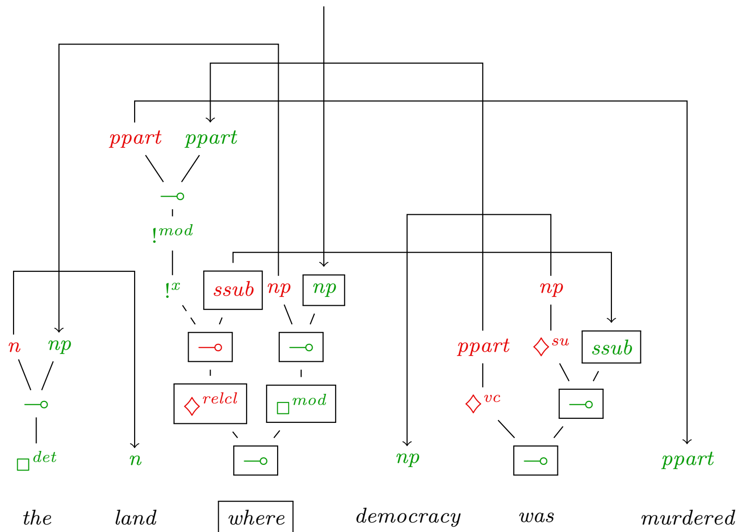
Proof ~~Frames~~ Structures

ppart ppart
$!^{mod}$
$!^{x}$ ssub np np
n np
ppart $\Diamond^{su}$ ssub
np
$\Diamond^{relcl}$ $\Box^{mod}$
$\Diamond^{vc}$
$\Box^{det}$ n np ppart

the *land* *where* *democracy* *was* *murdered*

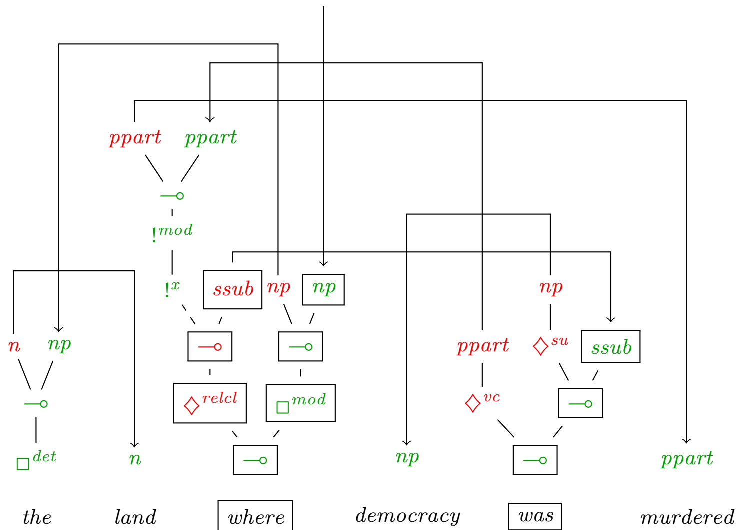
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(??? ???))) ???

Proof ~~Frames~~ Structures

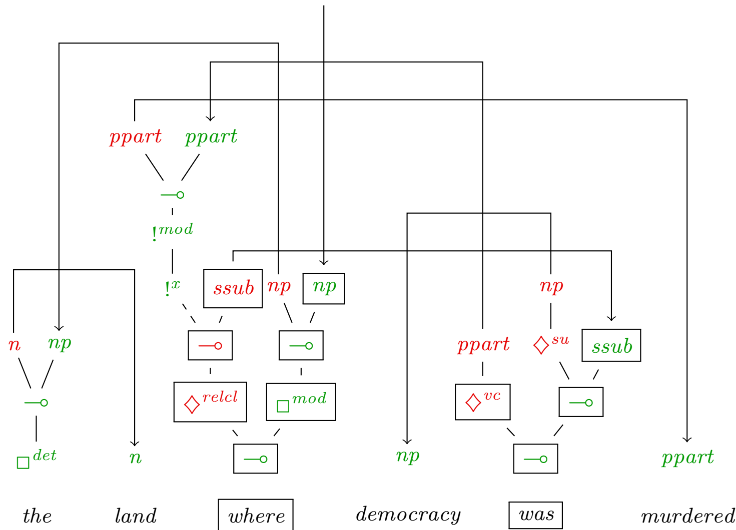


$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ $(\lambda x_0.(??? \; ??? \; ???)))$???

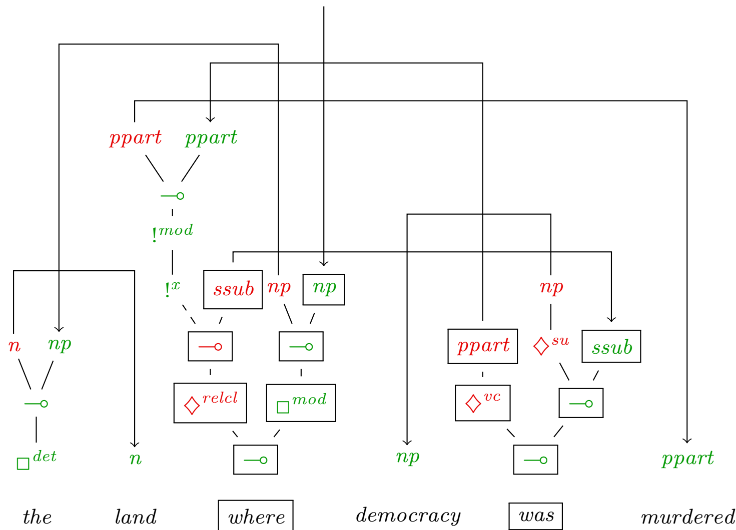
Proof ~~Frames~~ Structures



Proof ~~Frames~~ Structures

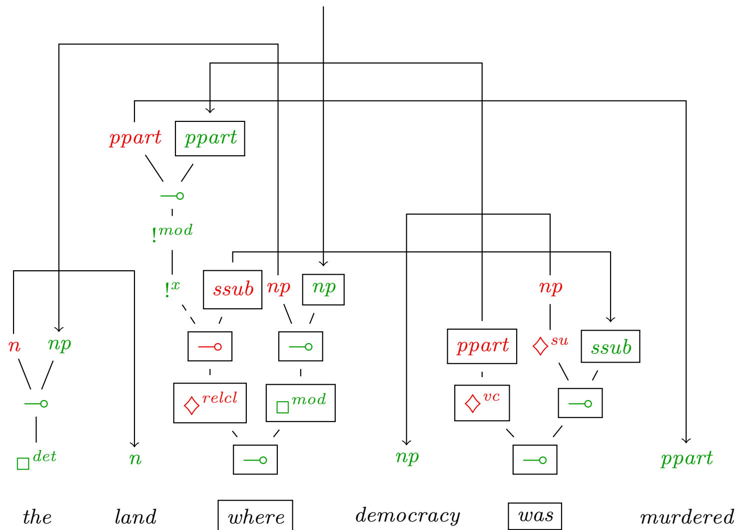


Proof ~~Frames~~ Structures



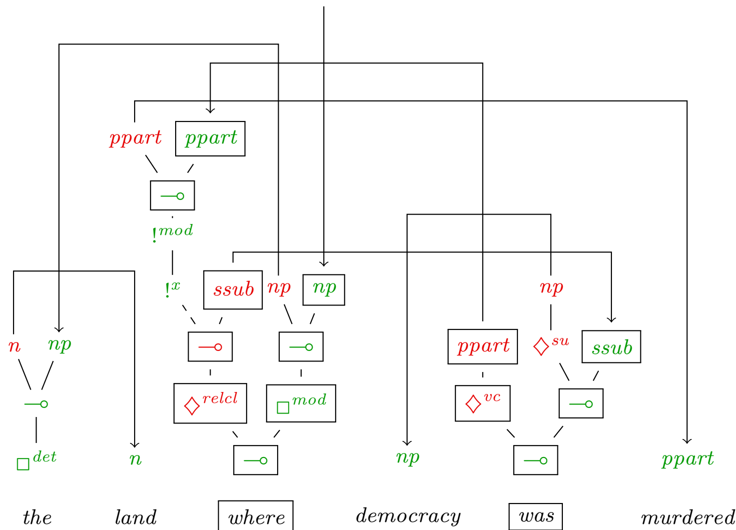
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ (???) ???))) ???

Proof ~~Frames~~ Structures

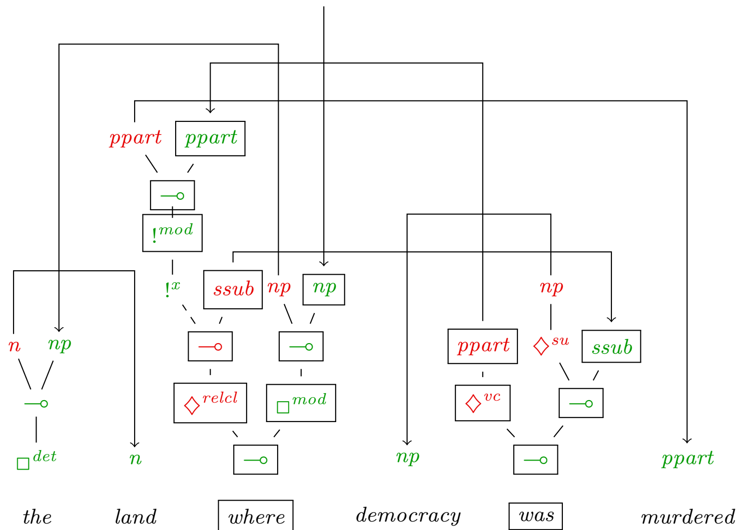


$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ (???) ???))) ???

Proof ~~Frames~~ Structures

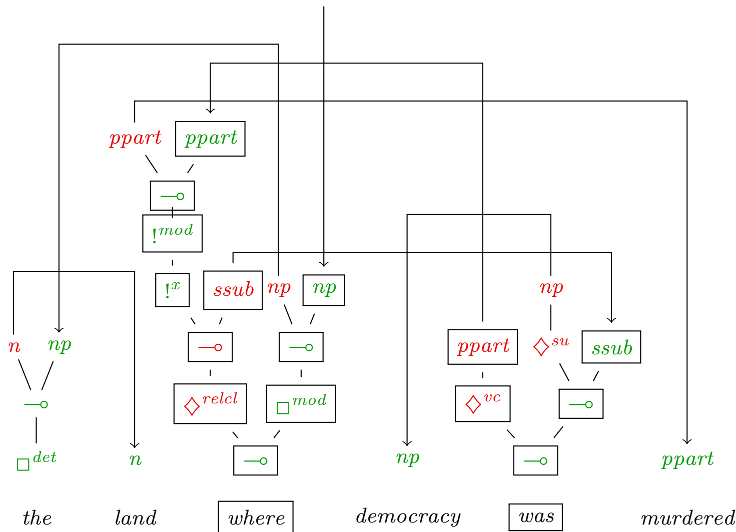


Proof ~~Frames~~ Structures

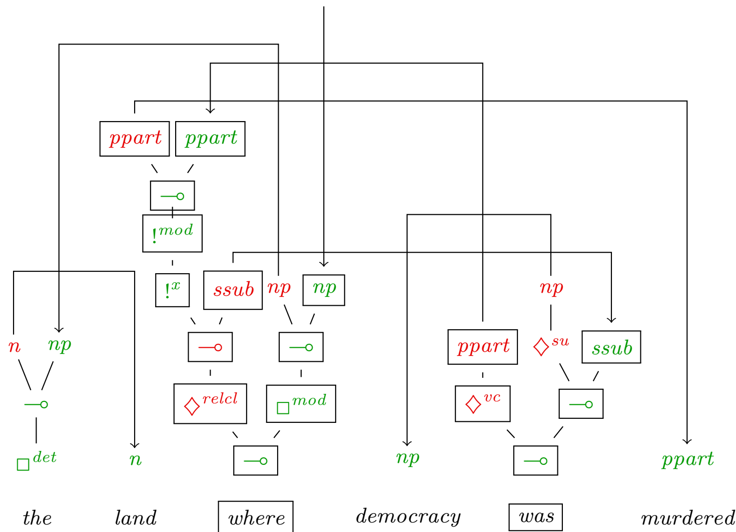


$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}$(???) ???) ???))) ???

Proof ~~Frames~~ Structures

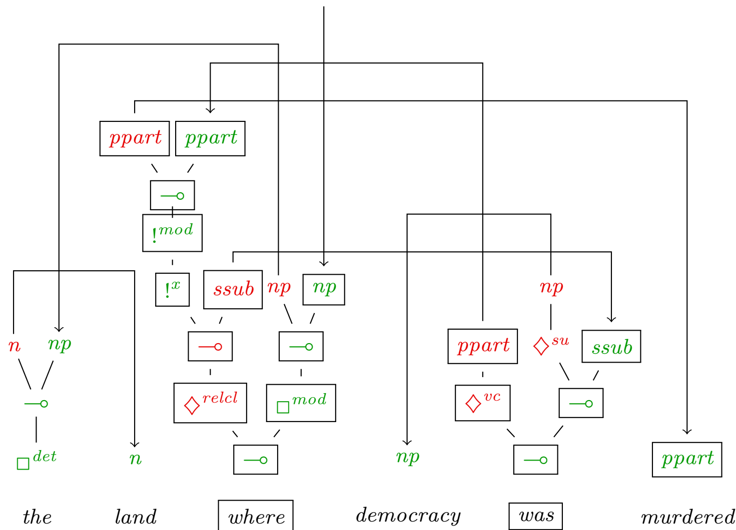


Proof ~~Frames~~ Structures



$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x} x_0$???) ???))) ???

Proof ~~Frames~~ Structures

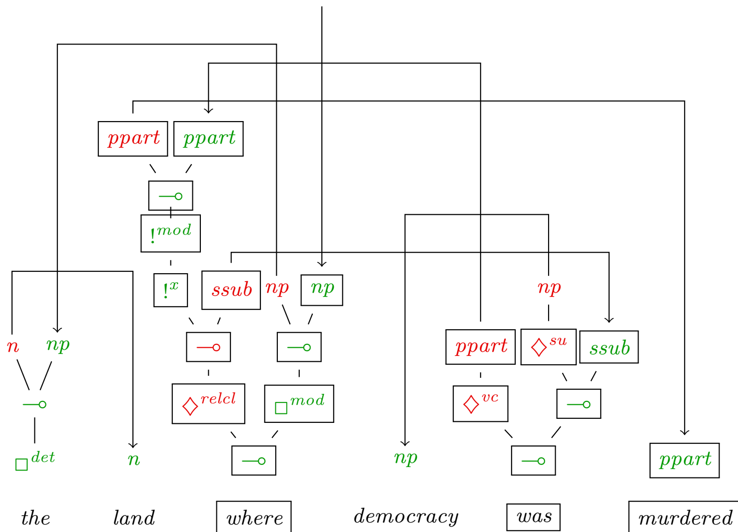


Proof ~~Frames~~ Structures

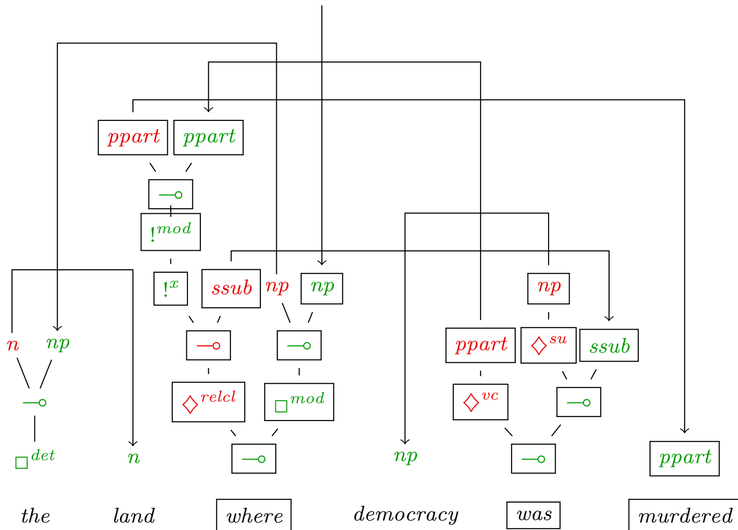
the *land* *where* *democracy* *was* *murdered*

$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) ???))) ???

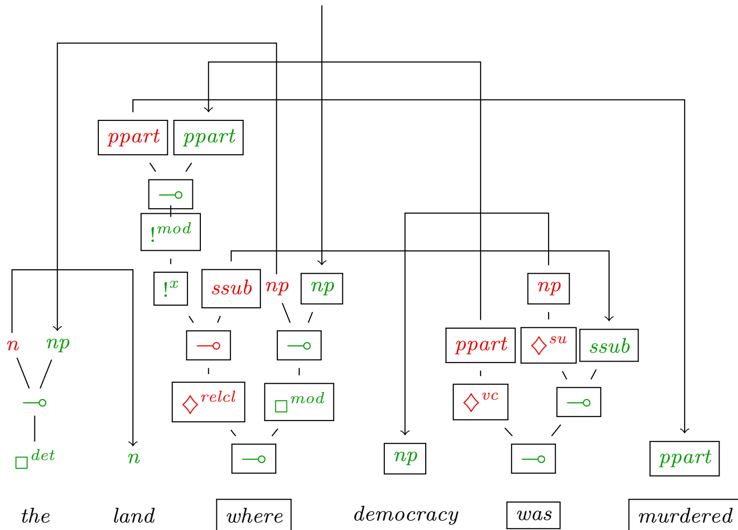
Proof ~~Frames~~ Structures



Proof ~~Frames~~ Structures

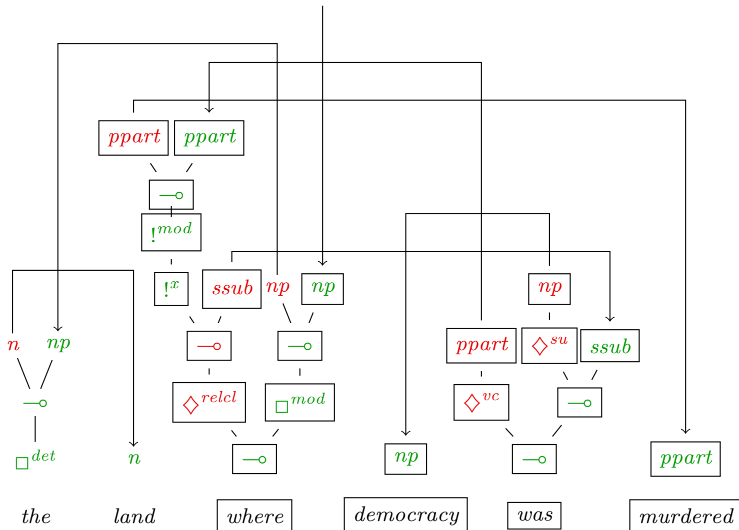


Proof ~~Frames~~ Structures

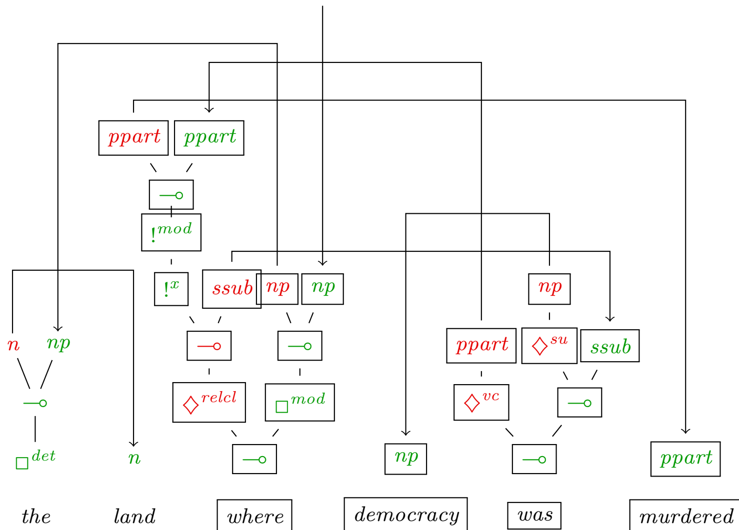


$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ $(\lambda x_0.$(was $\triangle^{vc}$ $(\blacktriangledown^{mod} \triangledown^{mod} \blacktriangledown^{x} \triangledown^{x} x_0$ murdered) $\triangle^{su}$???))) ???

Proof ~~Frames~~ Structures

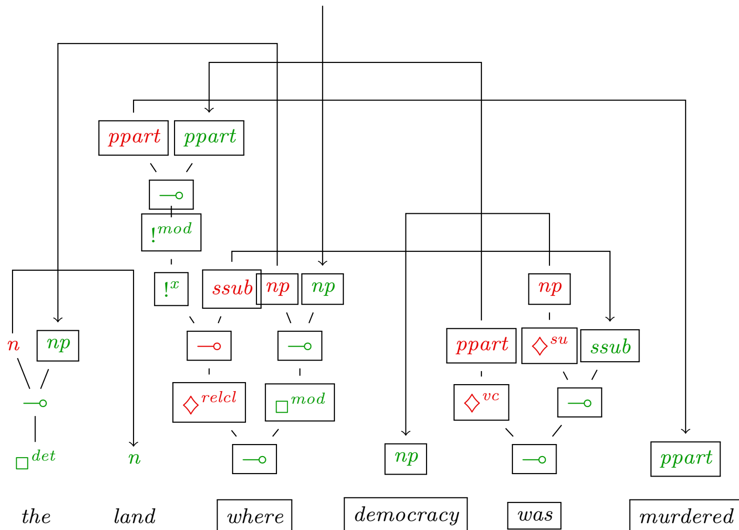


Proof ~~Frames~~ Structures



▼mod(where △relcl (λx0.(was △vc (▼mod▽mod▼x▽xx0 murdered) △su democracy))) ???

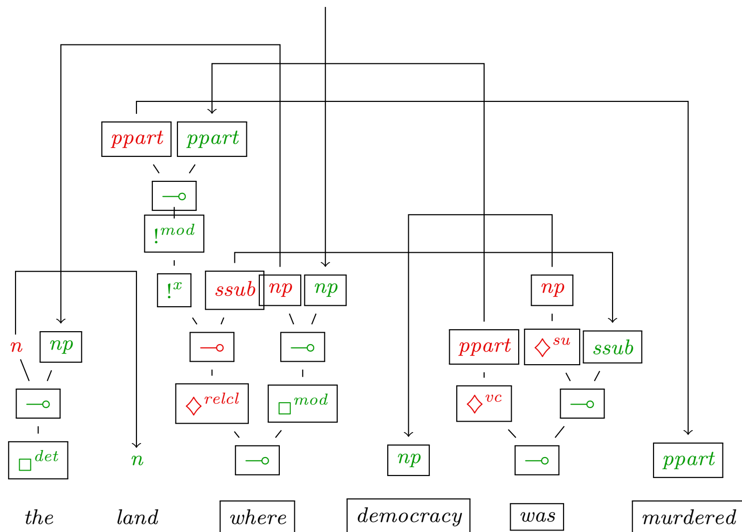
Proof ~~Frames~~ Structures



Proof ~~Frames~~ Structures

$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x} x_0$ murdered) $\triangle^{su}$ democracy))) (??? ???)

Proof ~~Frames~~ Structures

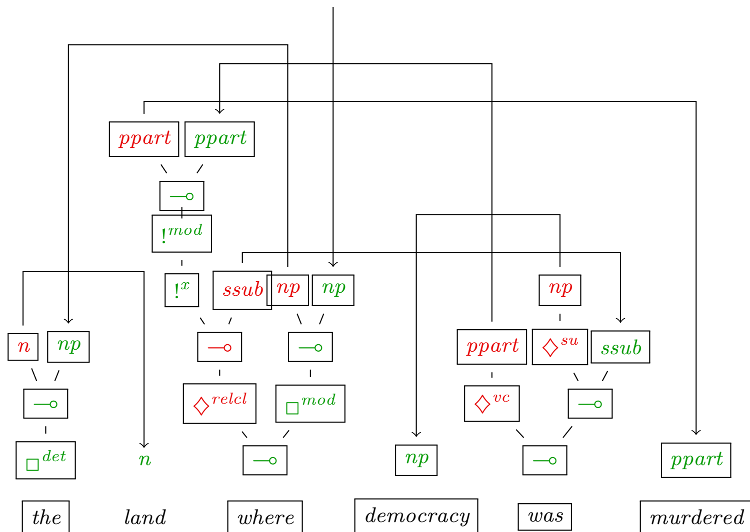


$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ $(\lambda x_0.$(was $\triangle^{vc}$ $(\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) $(\blacktriangledown^{det}$??? ???)

Proof ~~Frames~~ Structures

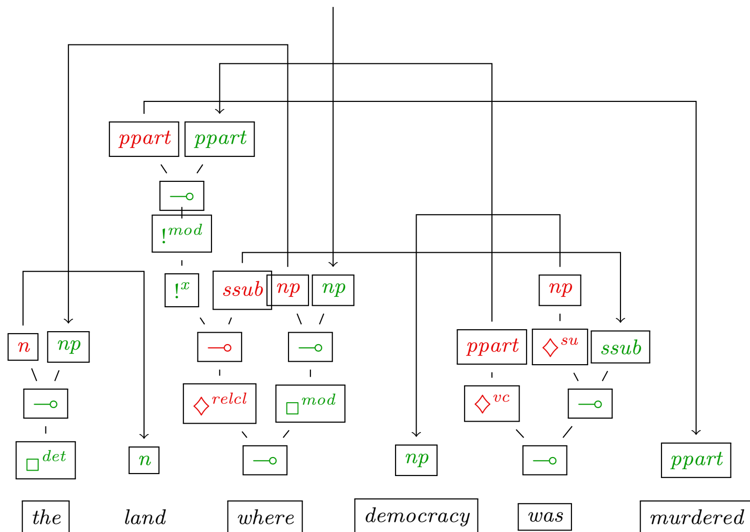
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ($\blacktriangledown^{det}$the ???)

Proof ~~Frames~~ Structures



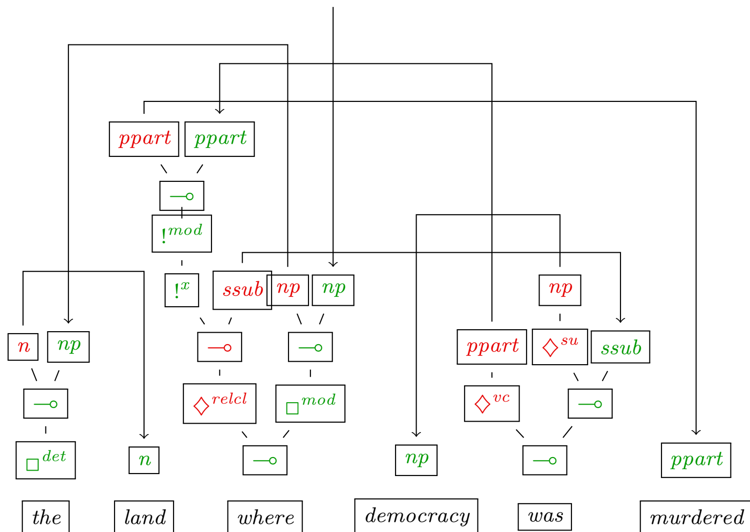
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ($\blacktriangledown^{det}$the ???)

Proof ~~Frames~~ Structures



$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod} \triangledown^{mod} \blacktriangledown^{x} \triangledown^{x} x_0$ murdered) $\triangle^{su}$ democracy))) ($\blacktriangledown^{det}$the ???)

Proof ~~Frames Structures~~ Nets



$\blacktriangledown^{mod}$(where $\vartriangle^{relcl}$ (λx_0.(was $\vartriangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\vartriangle^{su}$ democracy))) ($\blacktriangledown^{det}$the land)

written and directed by

KOKOS

"Lets neural sh!t up"
– Alonzo Church, 1973

"Lets neural sh!t up"
– Alonzo Church, 1973

disclaimer: not an actual quote

Main ingredients

- Type assignment (supertagging)
  parallel tree decoding with dynamic graph convolutions
- Axiom linking (neural bijections)
  optimal transport with Sinkhorn iterations
- Formal verification
  proof net traversal

Supertagging 101

goal

maximize $p(T_1, \ldots T_n)$ conditional on some input $(w_1, \ldots w_n)$

the catch

types are sparse $\implies$ fixed vocabulary classification = no good

Supertagging 101

goal

maximize $p(T_1, \ldots T_n)$ conditional on some input $(w_1, \ldots w_n)$

the catch

types are sparse $\implies$ fixed vocabulary classification = no good

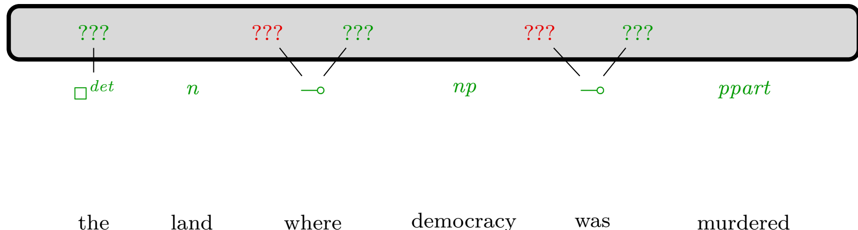
Supertagging as parallel graph completion

???	???	???	???	???	???
the	land	where	democracy	was	murdered

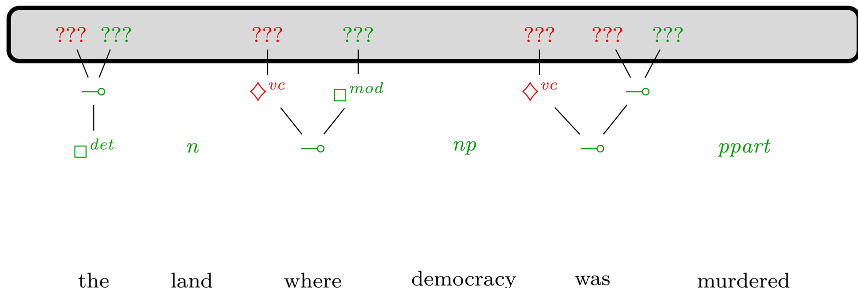
Supertagging as parallel graph completion

???	???	???	???	???	???
the	land	where	democracy	was	murdered

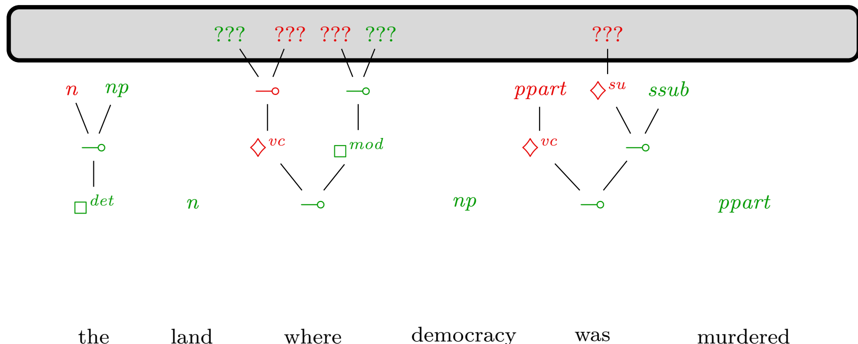
Supertagging as parallel graph completion



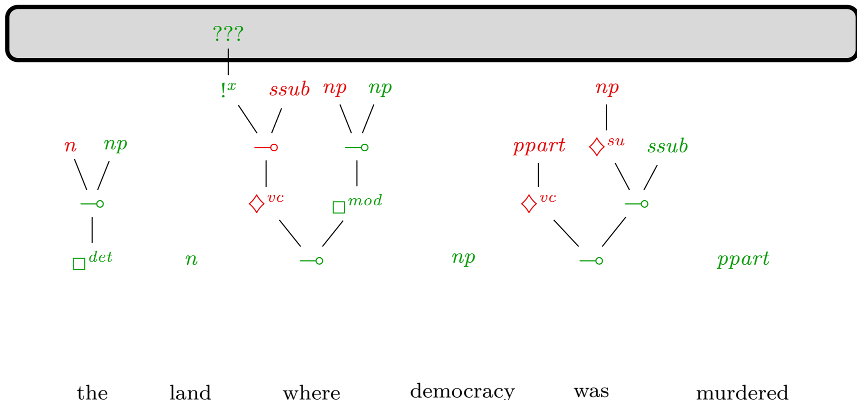
Supertagging as parallel graph completion



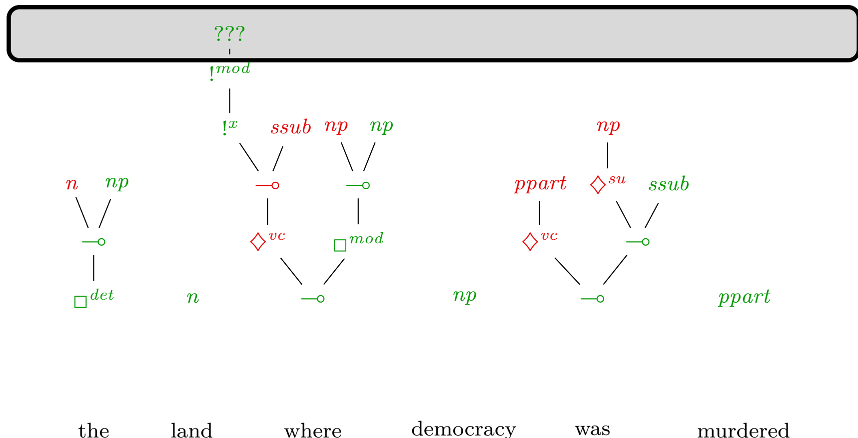
Supertagging as parallel graph completion



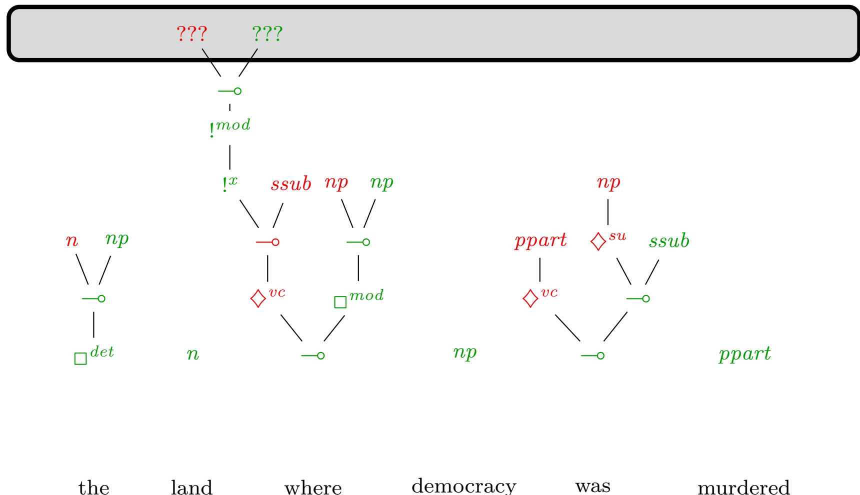
Supertagging as parallel graph completion



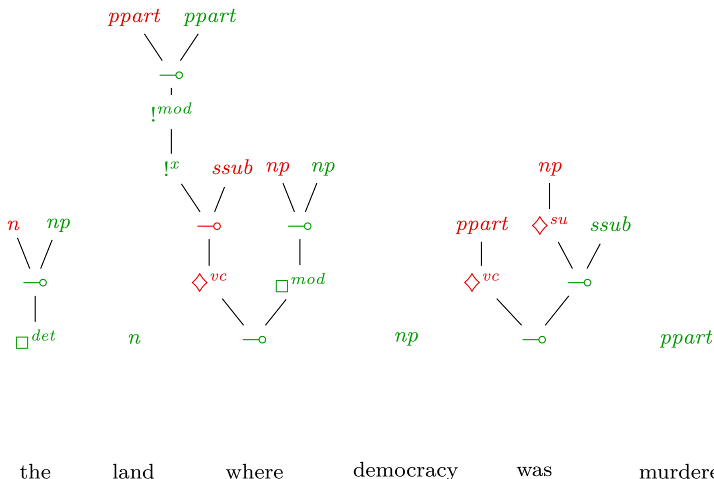
Supertagging as parallel graph completion



Supertagging as parallel graph completion



Supertagging as parallel graph completion



Not just a gray rectangle!

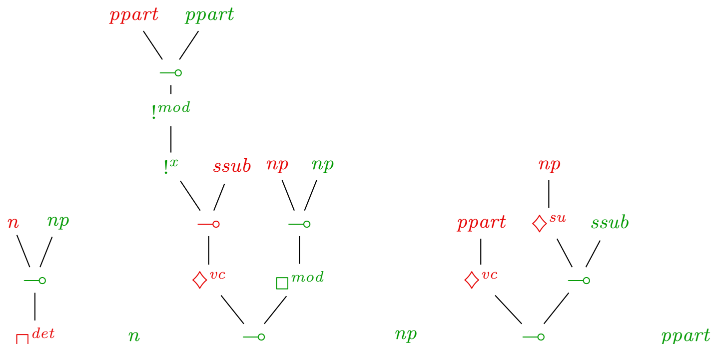
1 decoding step per tree depth; 3 message-passing rounds per step

- *contextualize: states* $\rightarrow$ *states*
  universal transformer encoder w/ relative weights
  (many-to-many, update states with neighborhood context)
- *predict: state* $\rightarrow$ *nodes*
  token classification w/ dynamic tree embeddings
  (one-to-many, predict fringe nodes from current state)
- *feedback: nodes* $\rightarrow$ *state*
  heterogeneous graph attention
  (many-to-one, update state with last predicted nodes)

[Kogkalidis & Moortgat, ???]

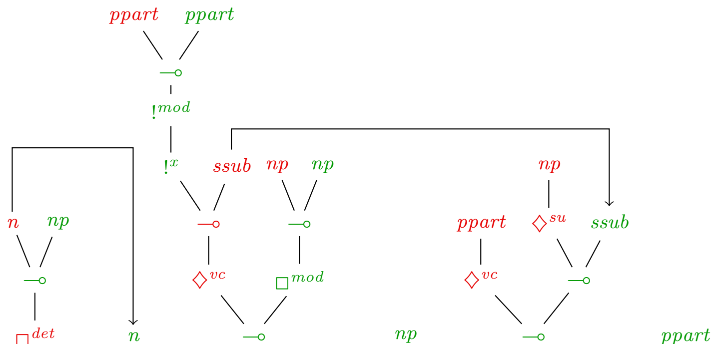
Axiom Linking

- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*



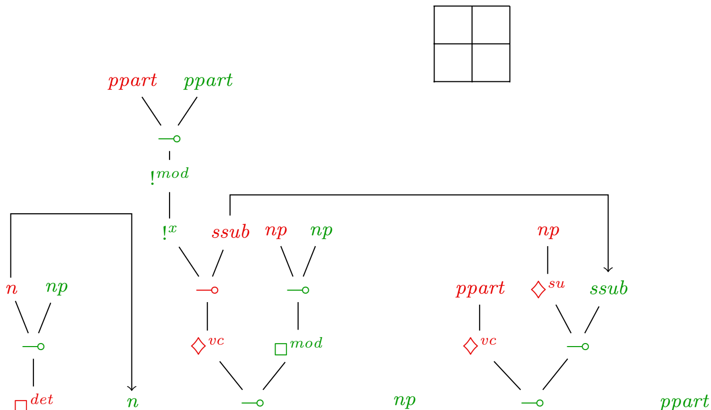
Axiom Linking

- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*



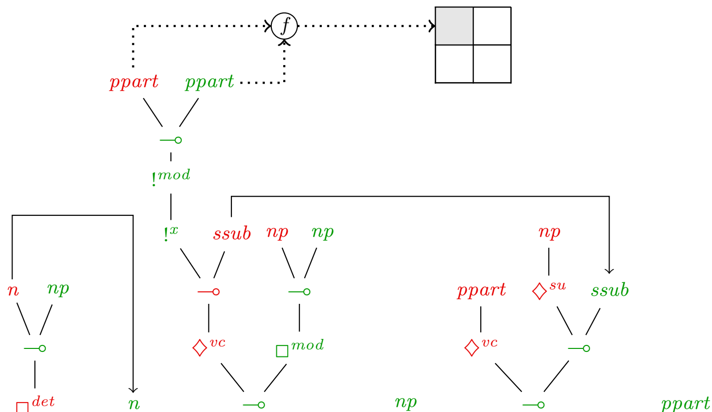
Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



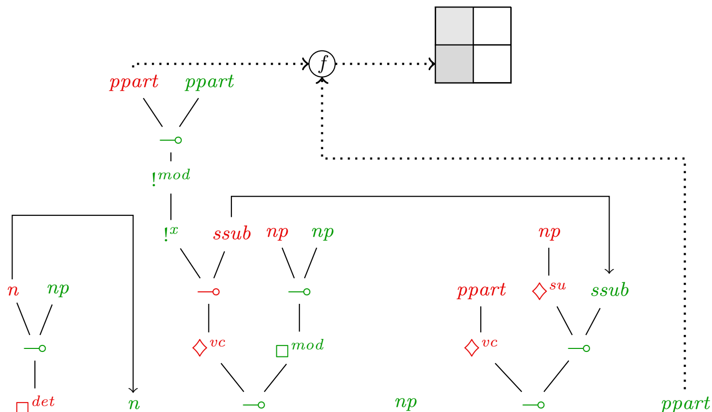
Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



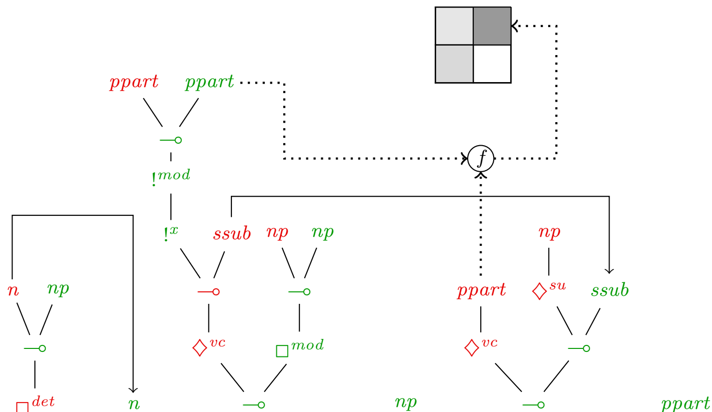
Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



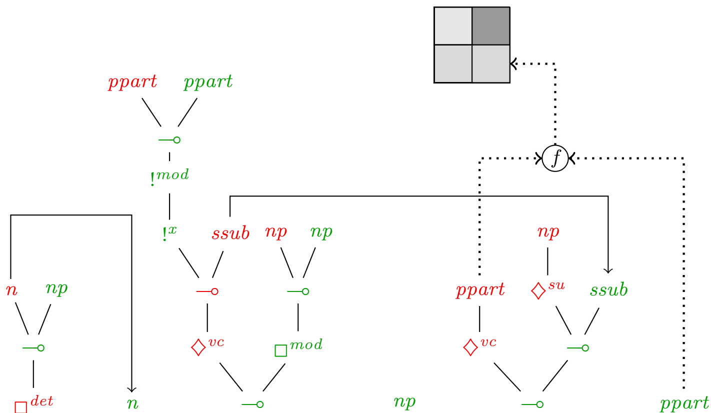
Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



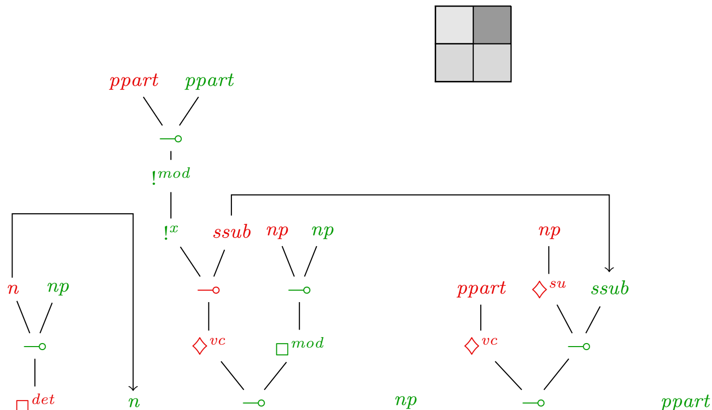
Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



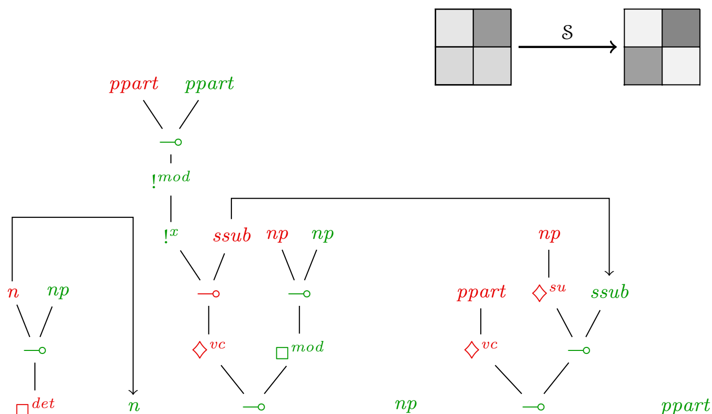
Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



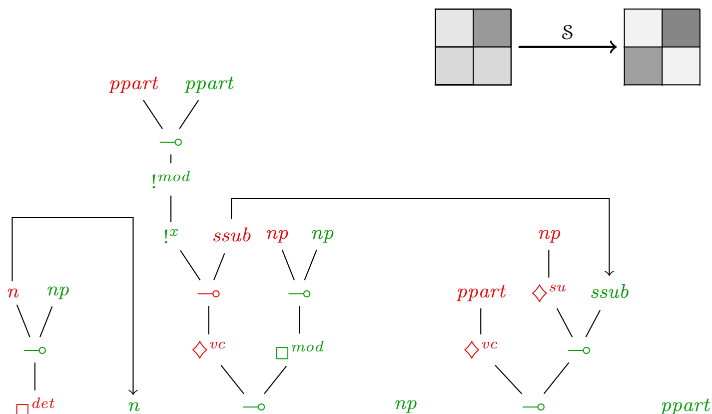
Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



Axiom Linking

- ☹ missing edges!
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*



Axiom Linking

- ☺ less missing edges
  - ! only consider edges between atoms of the *same* sign and *different* polarity
  - ! each atom can only be used *once*

your turn

break the parser!

Reality check

- tagger: applicable to any CG (applied)
- parser: adaptable to any flavor of linear logic (adapted)
- proofs: plug your own semantics / apply your own morphisms (todo)

Reality check

- tagger: applicable to any CG (applied)
- parser: adaptable to any flavor of linear logic (adapted)
- proofs: plug your own semantics / apply your own morphisms (todo)

Reality check

- tagger: applicable to any CG (applied)
- parser: adaptable to any flavor of linear logic (adapted)
- proofs: plug your own semantics / apply your own morphisms (todo)

Reality check

- tagger: applicable to any CG (applied)
- parser: adaptable to any flavor of linear logic (adapted)
- proofs: plug your own semantics / apply your own morphisms (todo)