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

Types

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

Types inductively defined by:

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

- $\multimap$ – linear function builder
- $\diamond$ – 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

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

☺ simple dependency demarkation

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

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

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

😐 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:

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

ad-hoc extraction

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"*:

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

$$!(\_) := \diamondsuit\square(\_)$$

Formula Decomposition: Types ≡ Trees

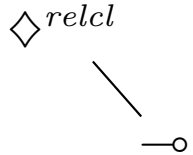
The type assignment of *"where"*:

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

Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

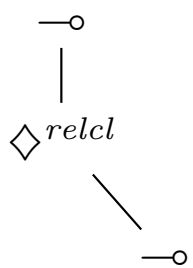
$$\Diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

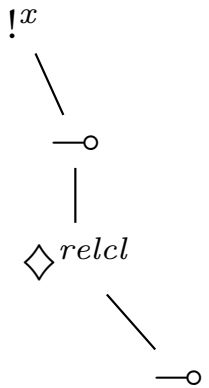
$$\Diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

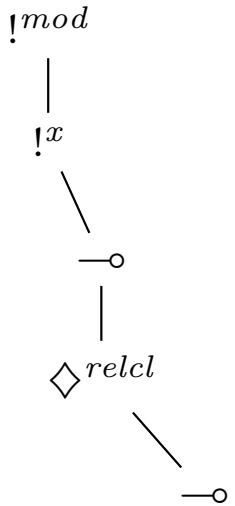
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

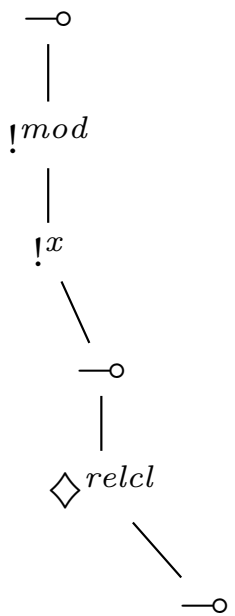
$$\diamondsuit^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \square^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

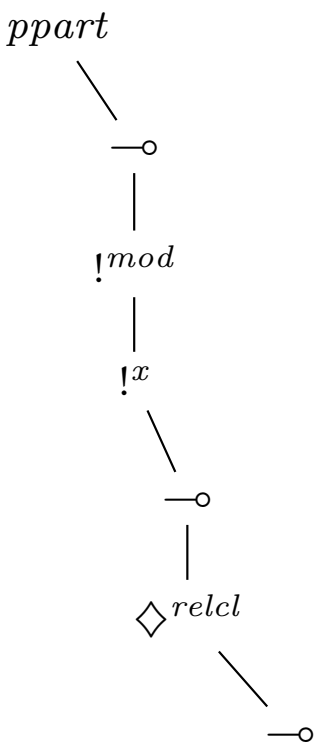
$$\Diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

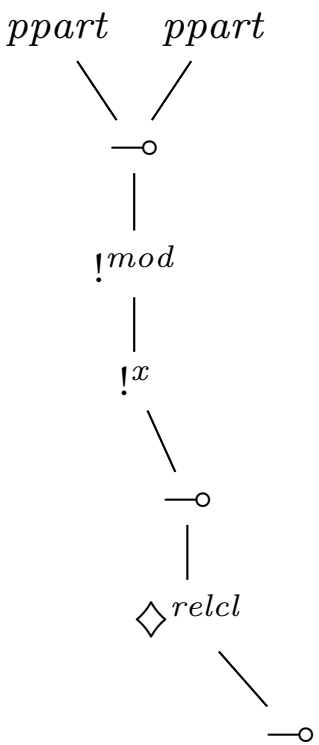
$$\Diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

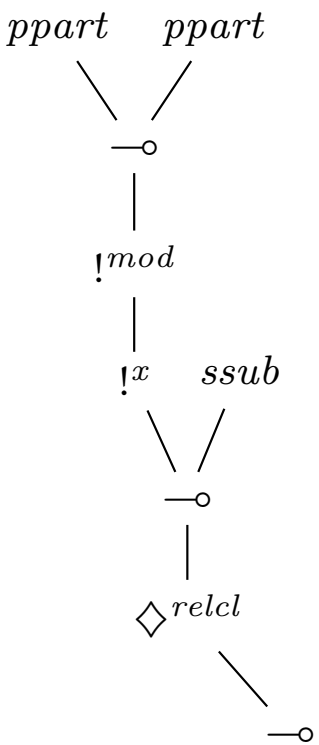
$$\Diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

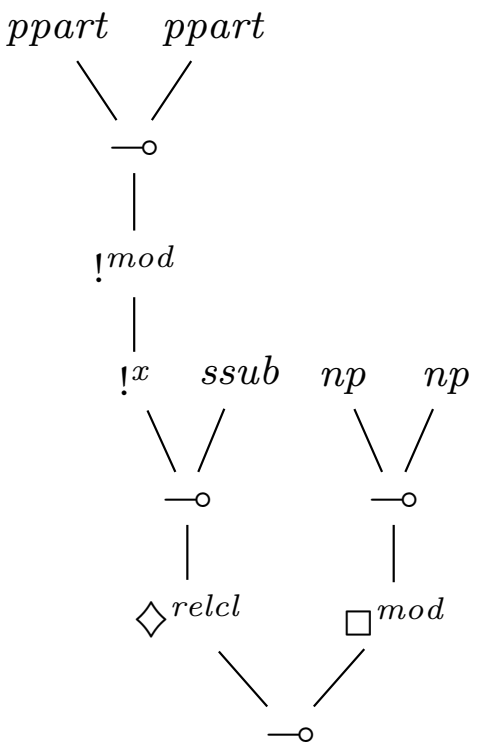
$$\Diamond^{relcl}(!^{x}!^{mod}(ppart \multimap ppart) \multimap ssub) \multimap \Box^{mod}(np \multimap np)$$



Formula Decomposition: Types ≡ Trees

The type assignment of *"where"*:

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

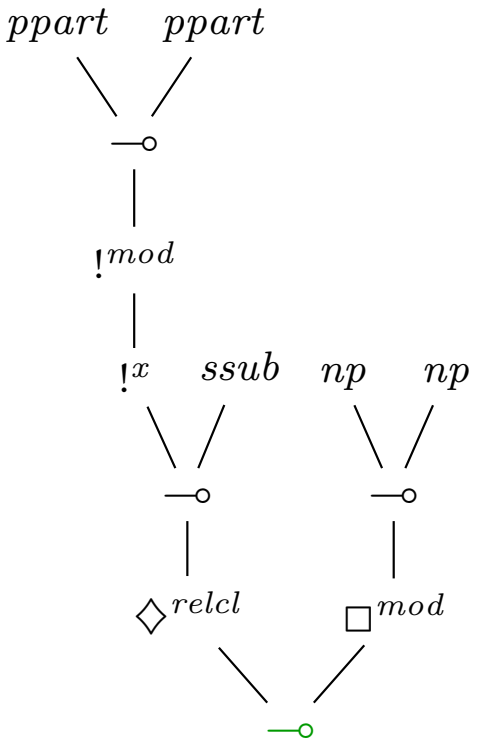


Formula Decomposition: Polarity Induction

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

\+ we have

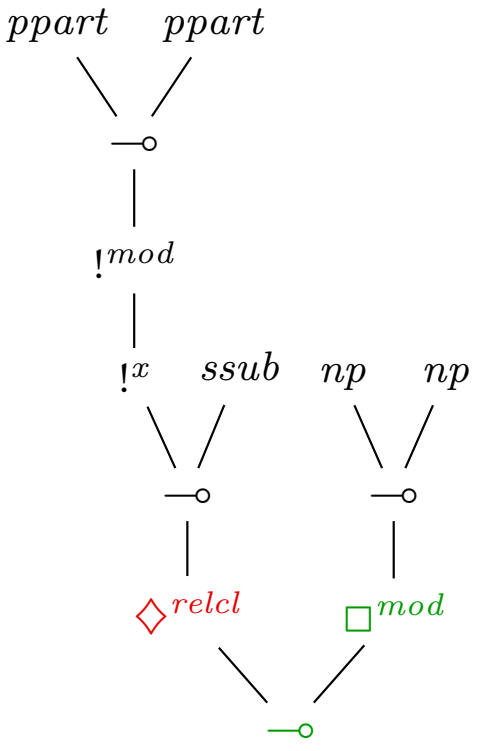
\- we miss



Formula Decomposition: Polarity Induction

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

+ we have
- we miss

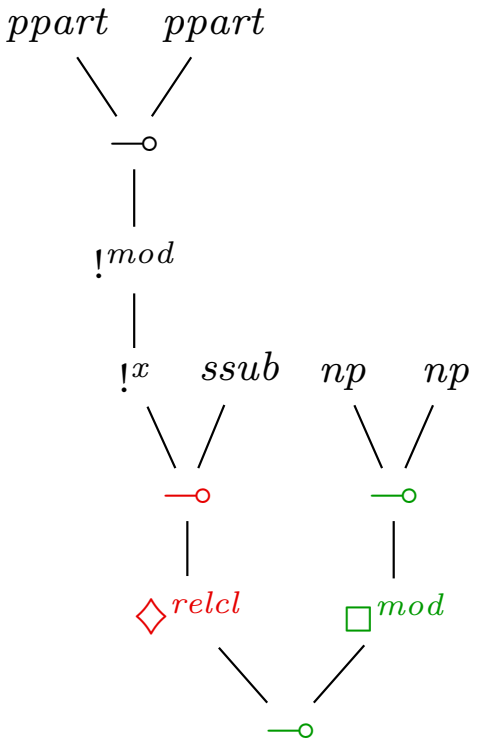


Formula Decomposition: Polarity Induction

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

\+ we have

\- we miss

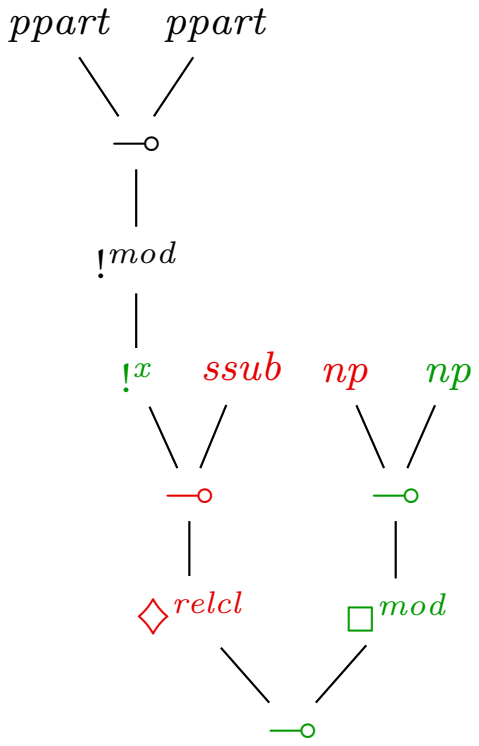


Formula Decomposition: Polarity Induction

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

\+ we have

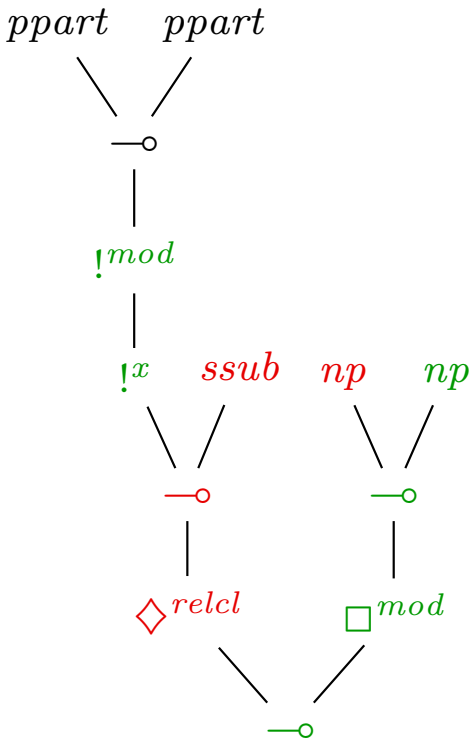
\- we miss



Formula Decomposition: Polarity Induction

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

+ we have
- we miss

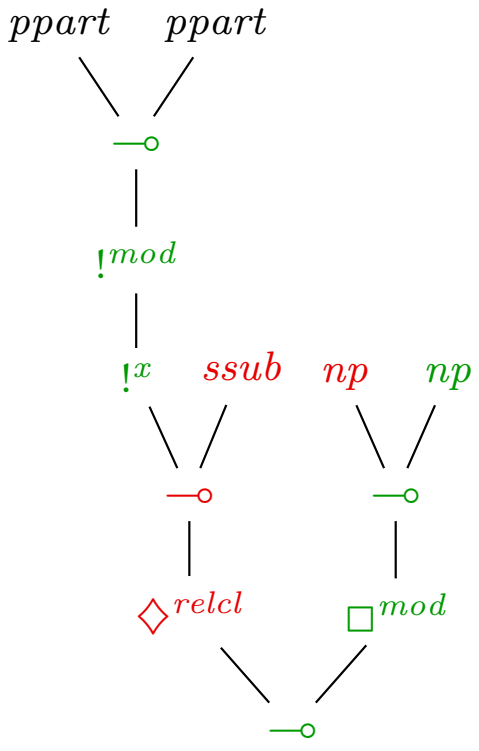


Formula Decomposition: Polarity Induction

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

+ we have

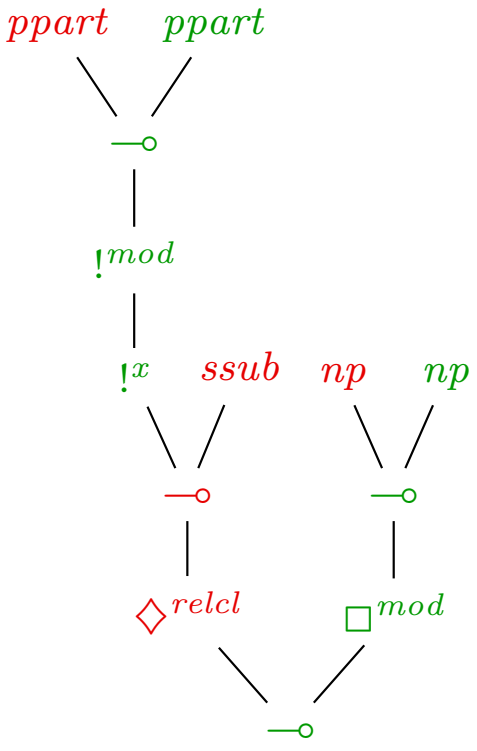
- we miss



Formula Decomposition: Polarity Induction

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

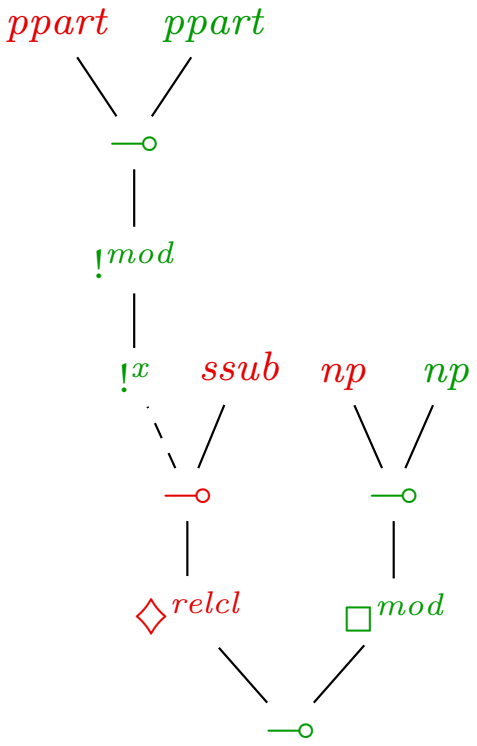
+ we have
- we miss



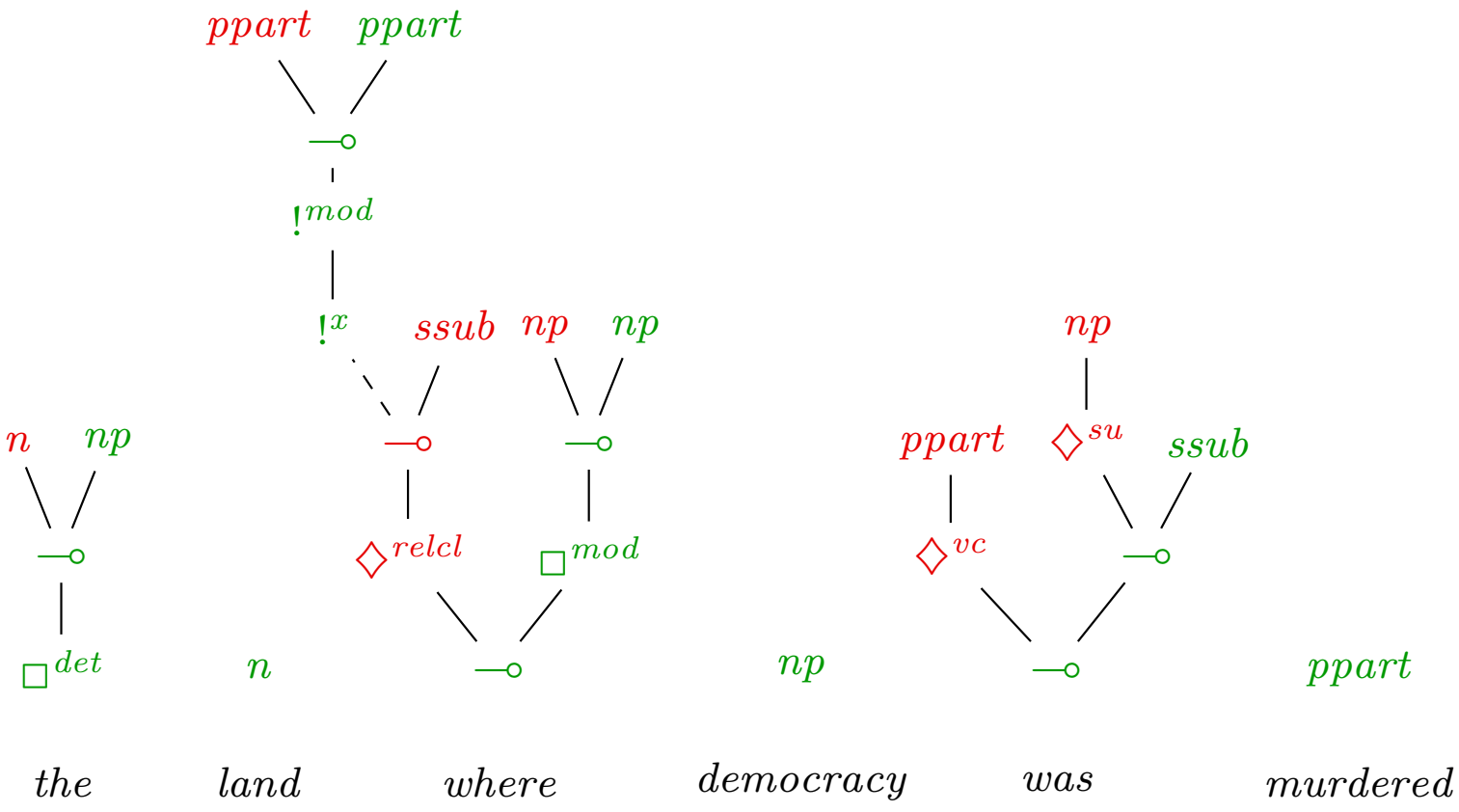
Formula Decomposition: Polarity Induction

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

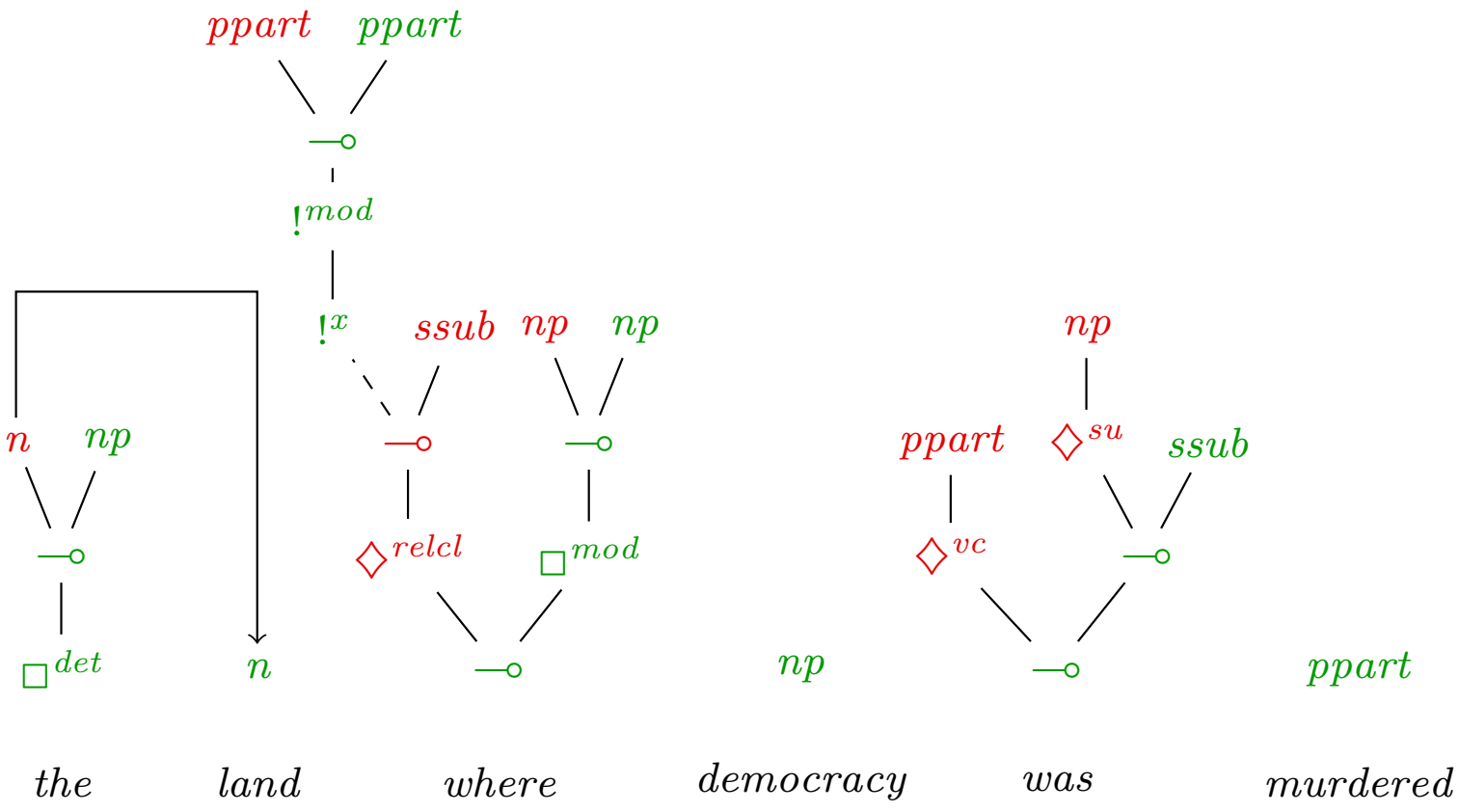
+ we have
- we miss



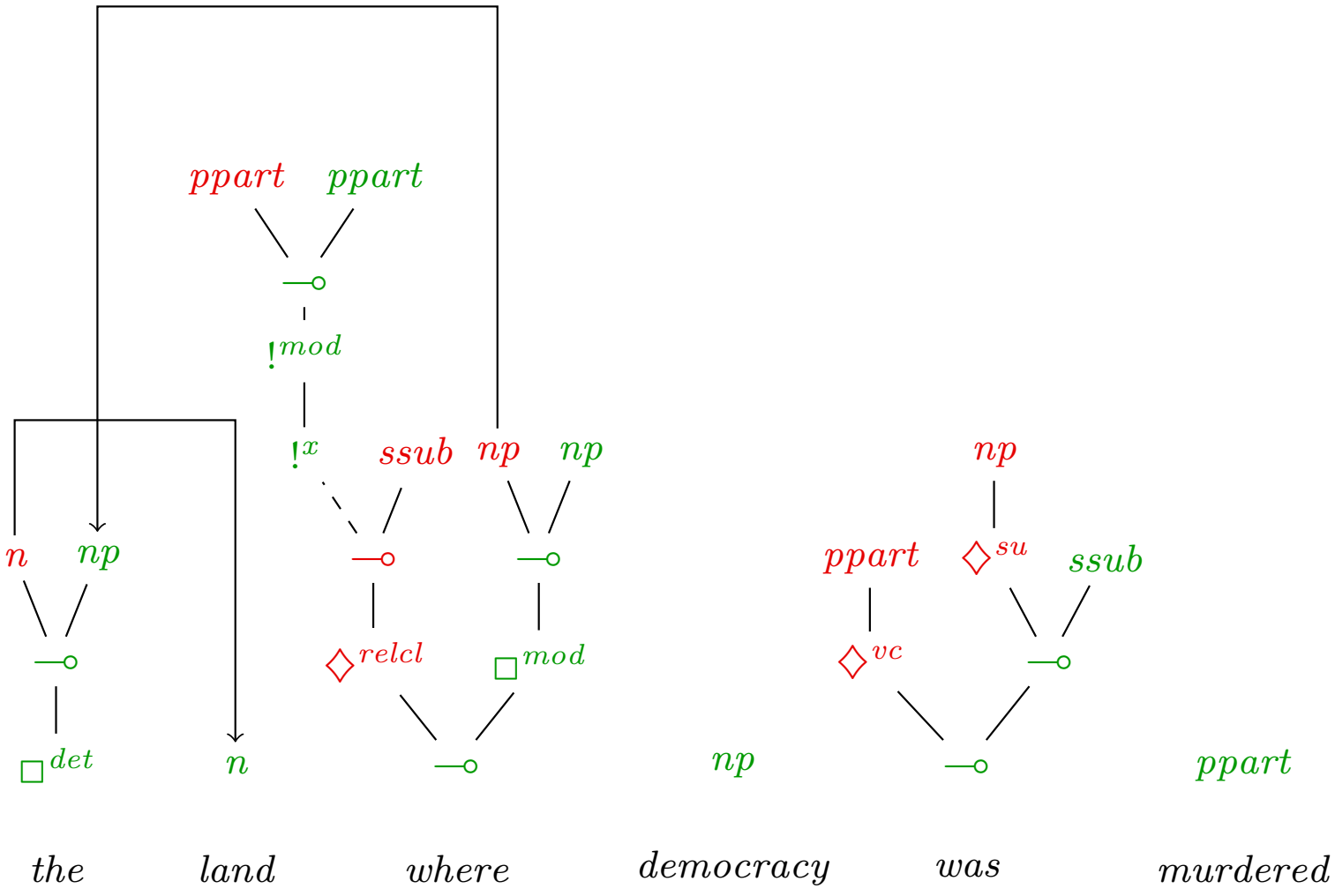
Proof Frames



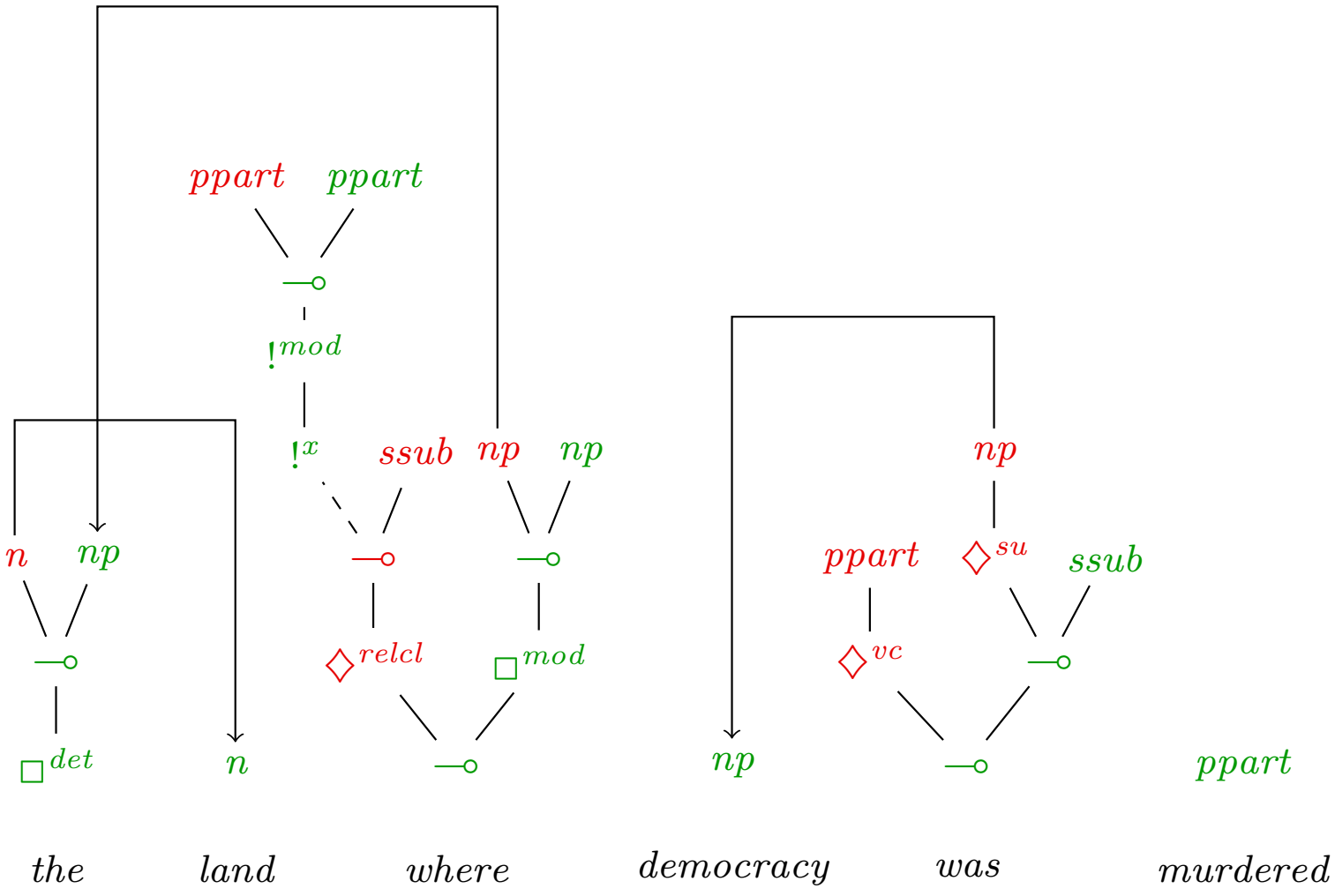
Proof Frames



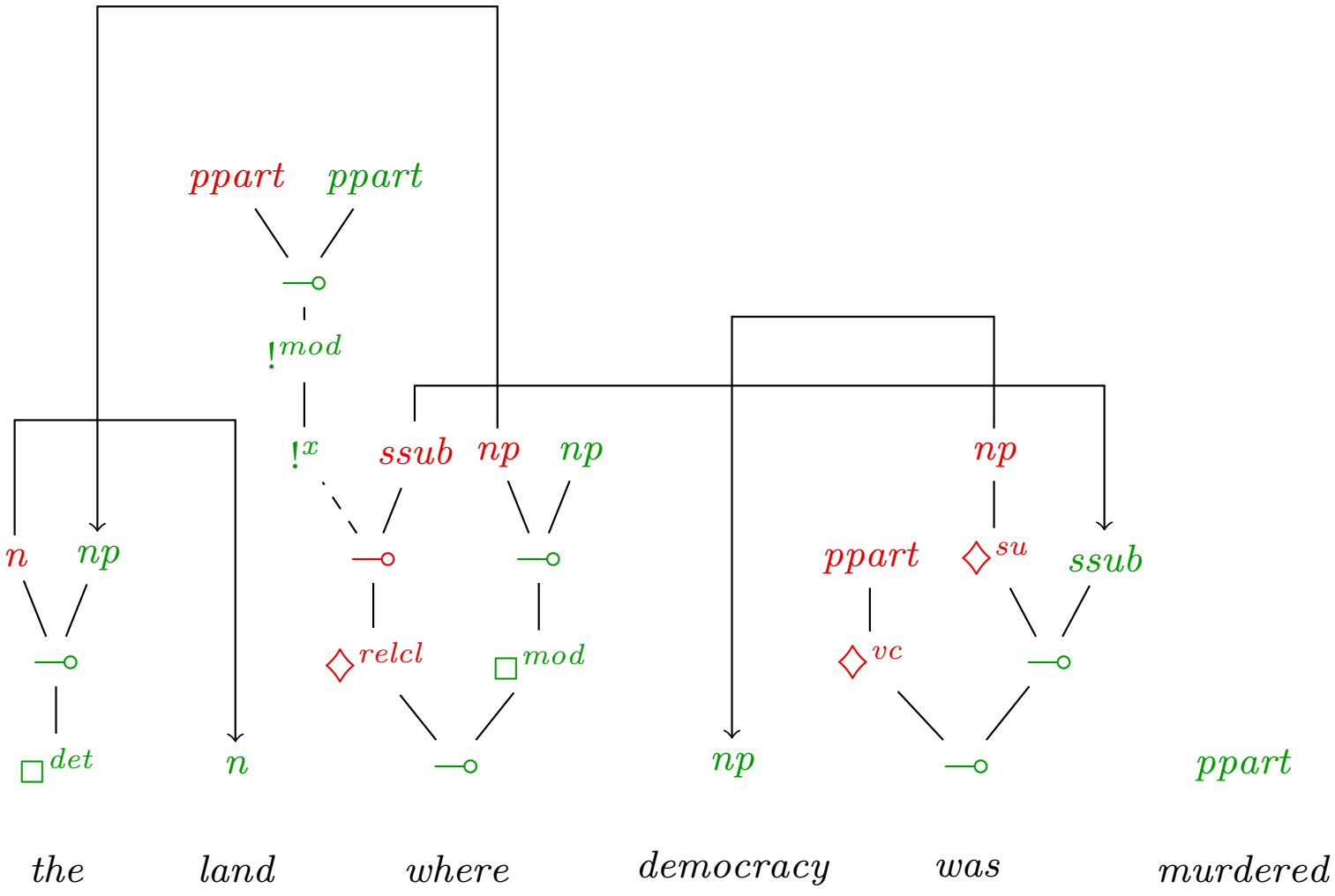
Proof Frames



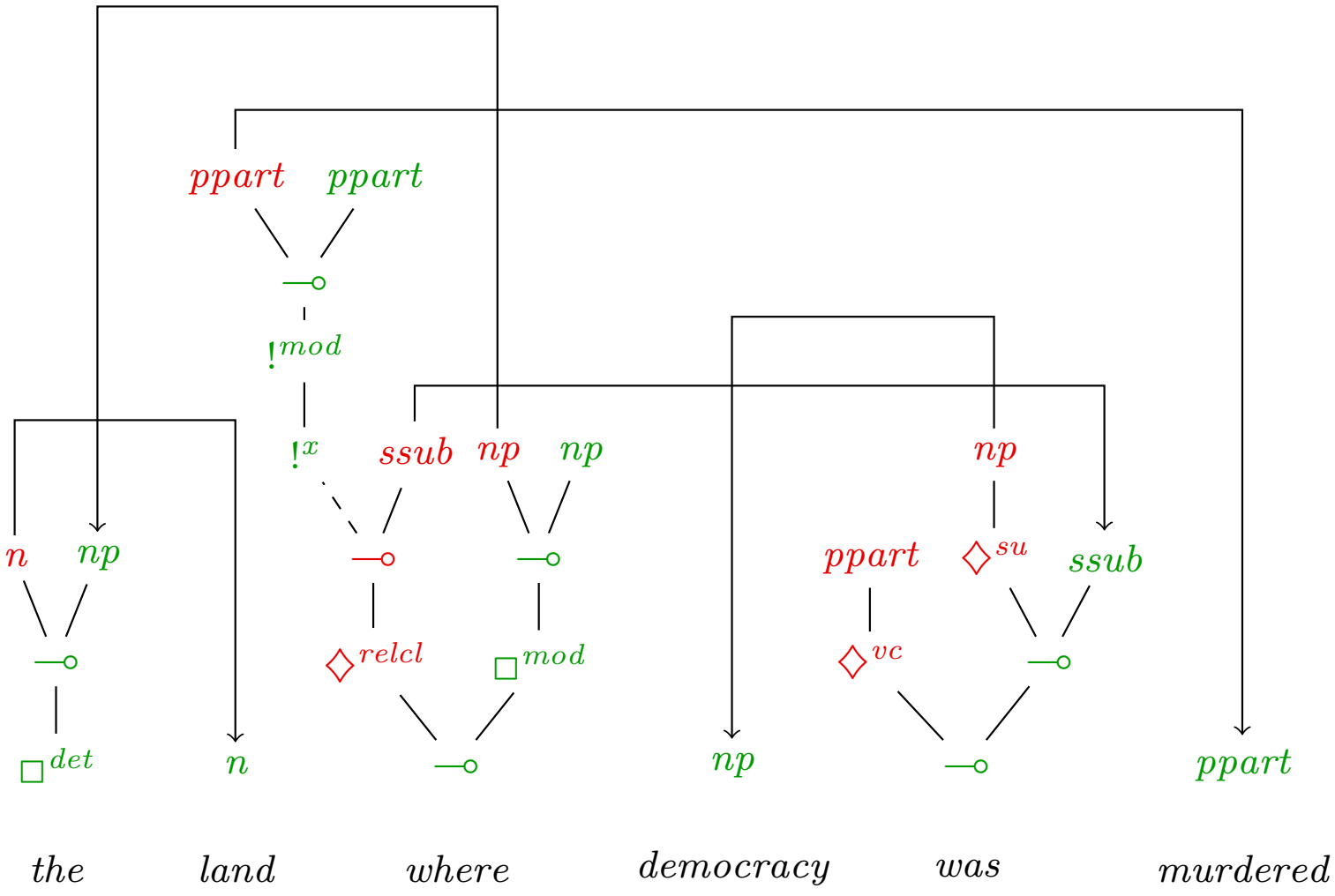
Proof Frames



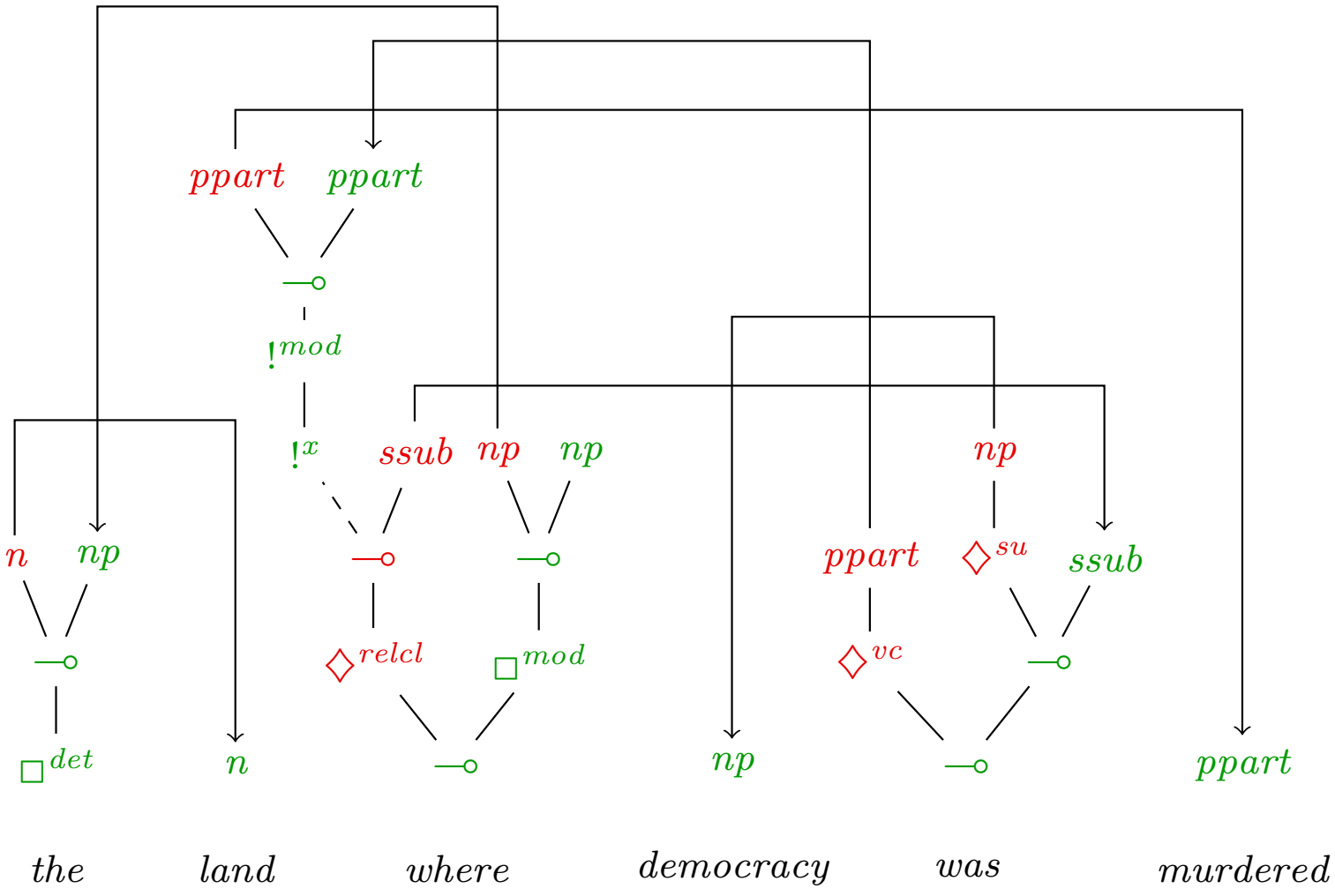
Proof Frames



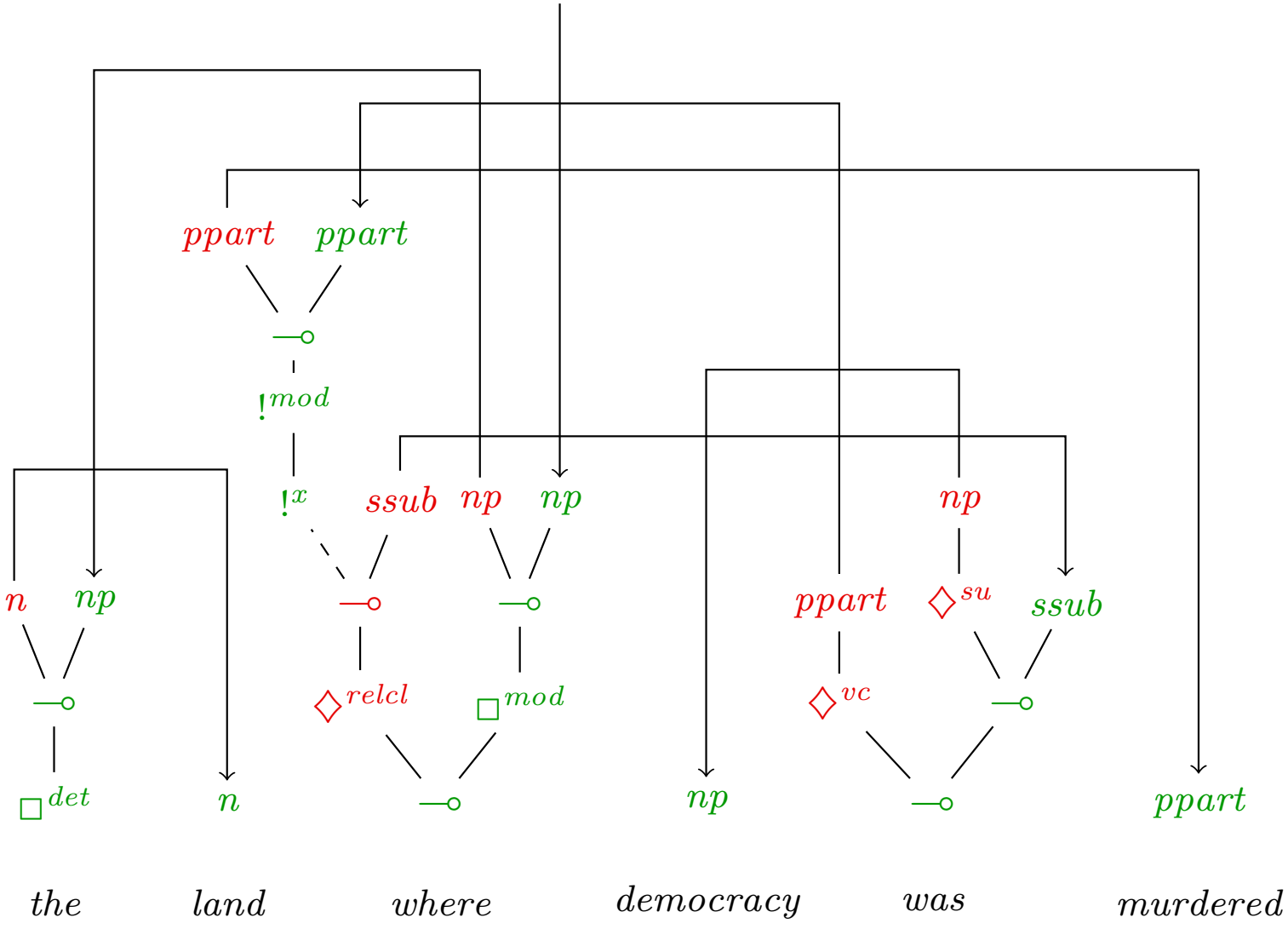
Proof Frames



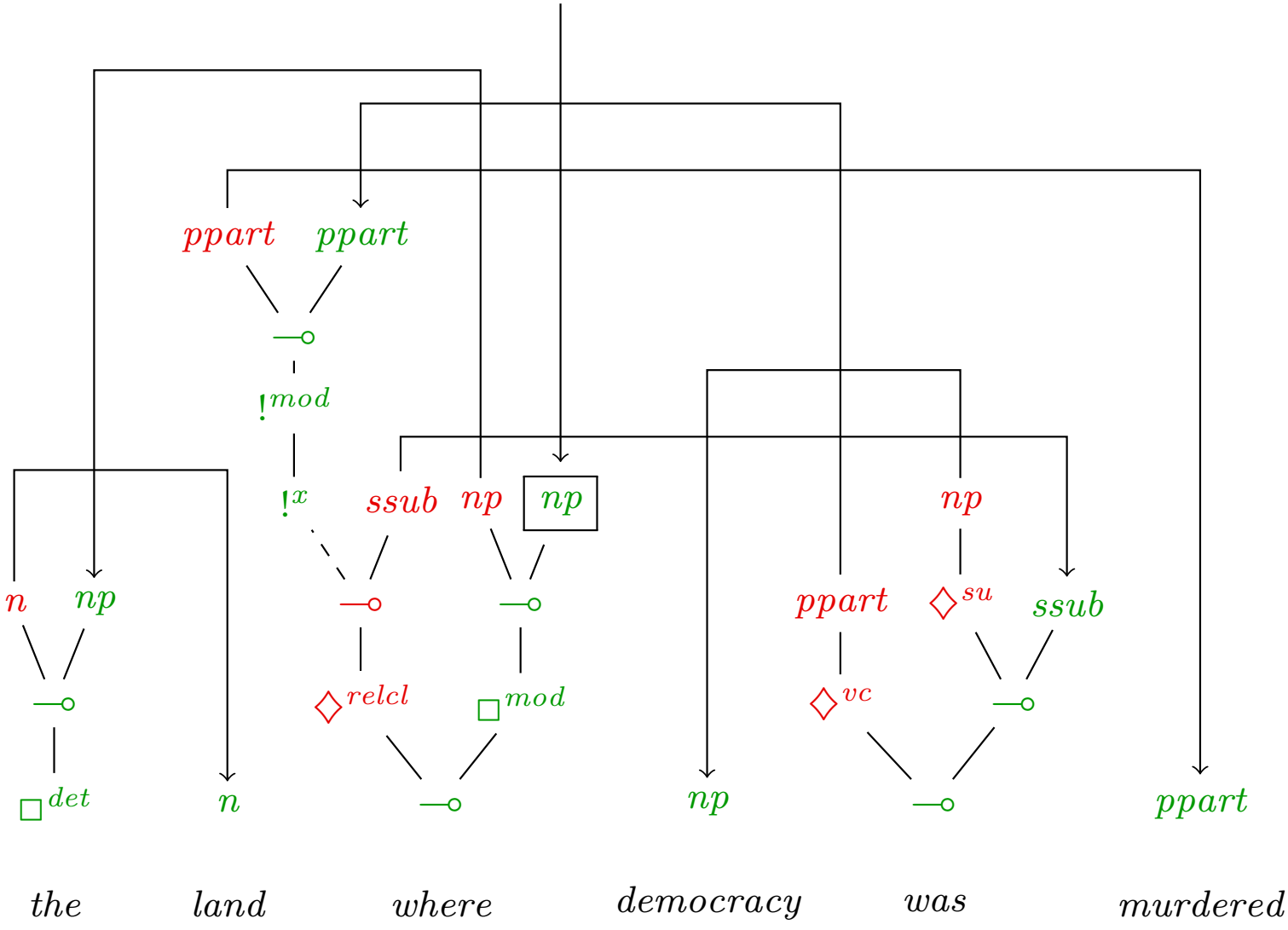
Proof Frames



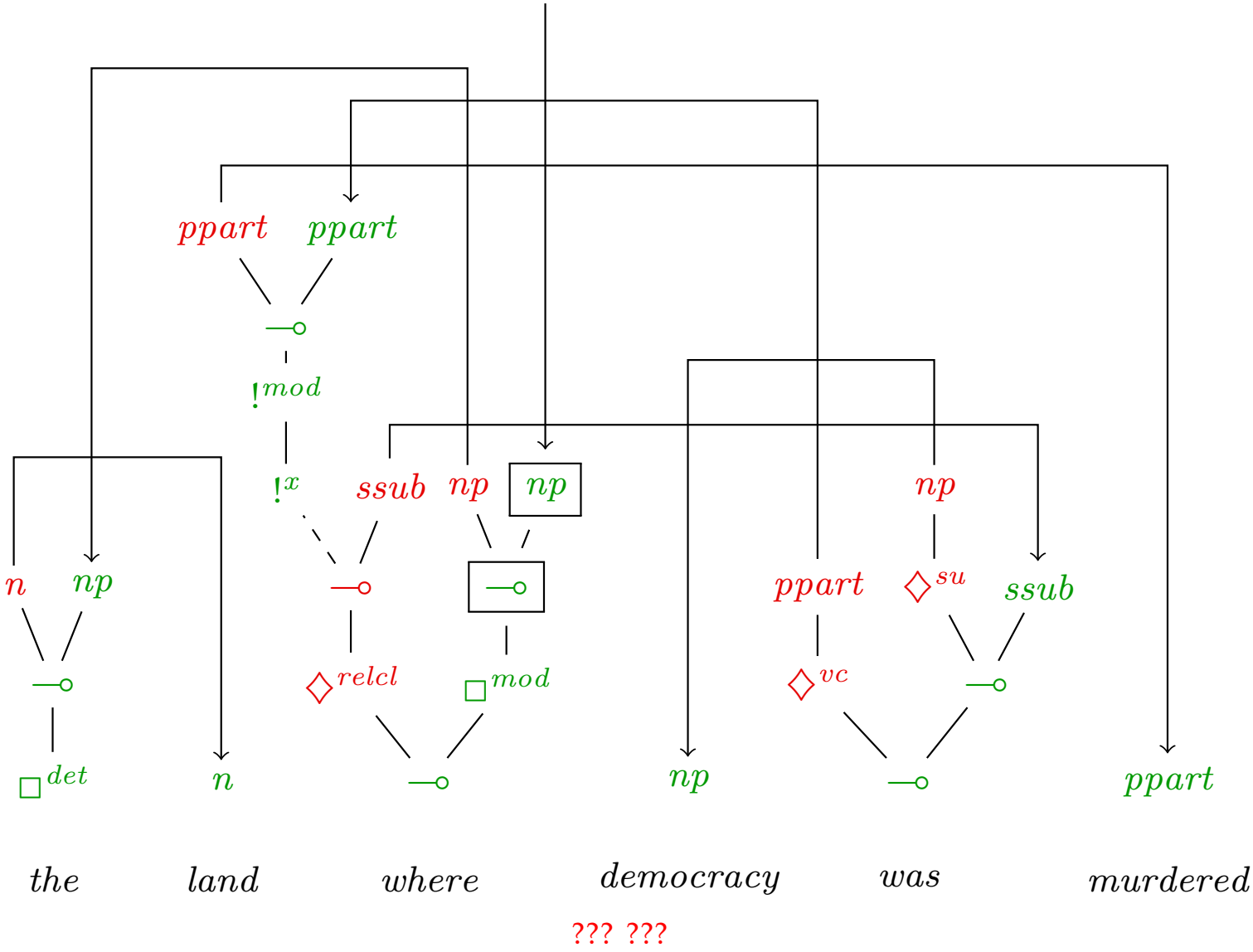
Proof Frames



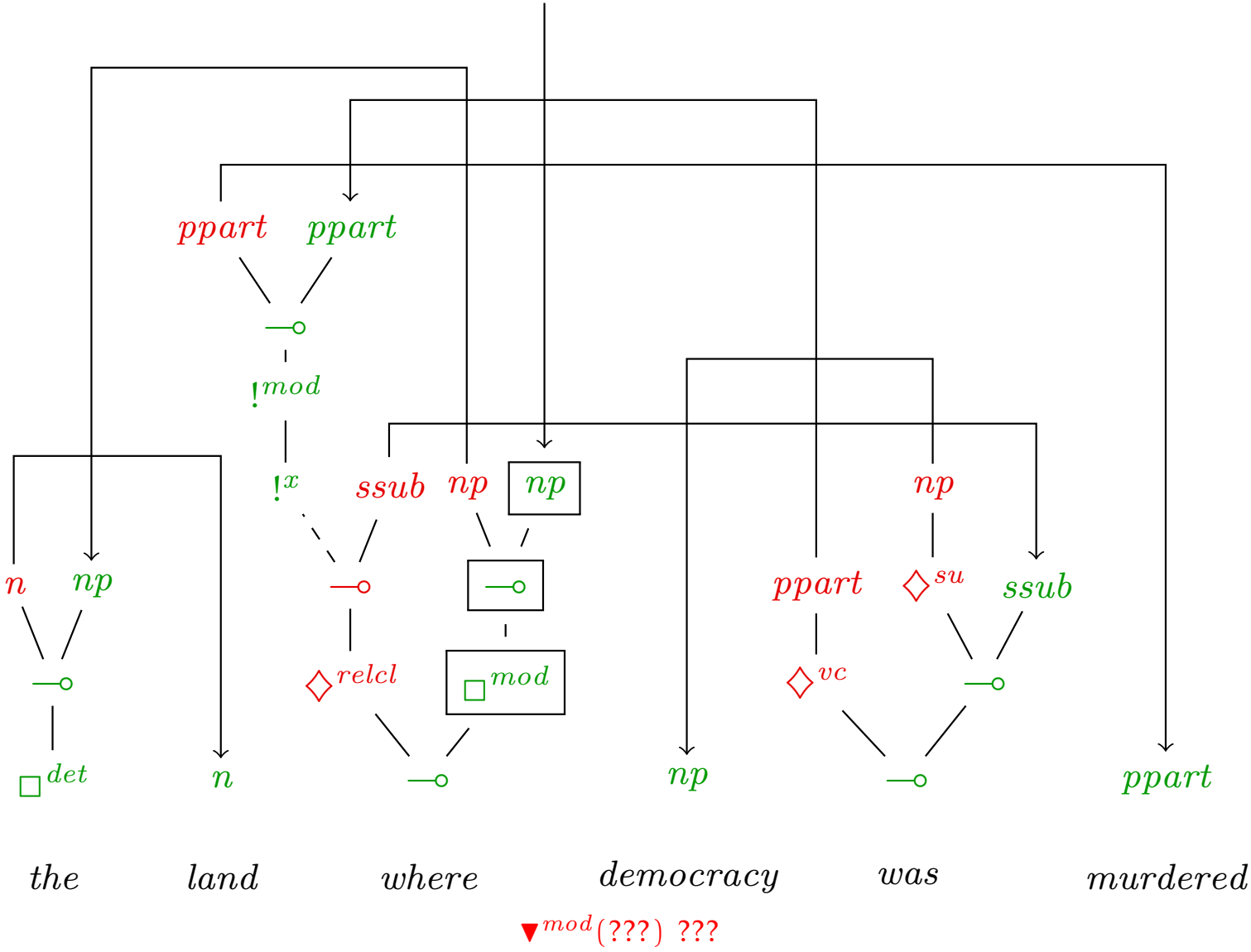
Proof ~~Frames~~ Structures



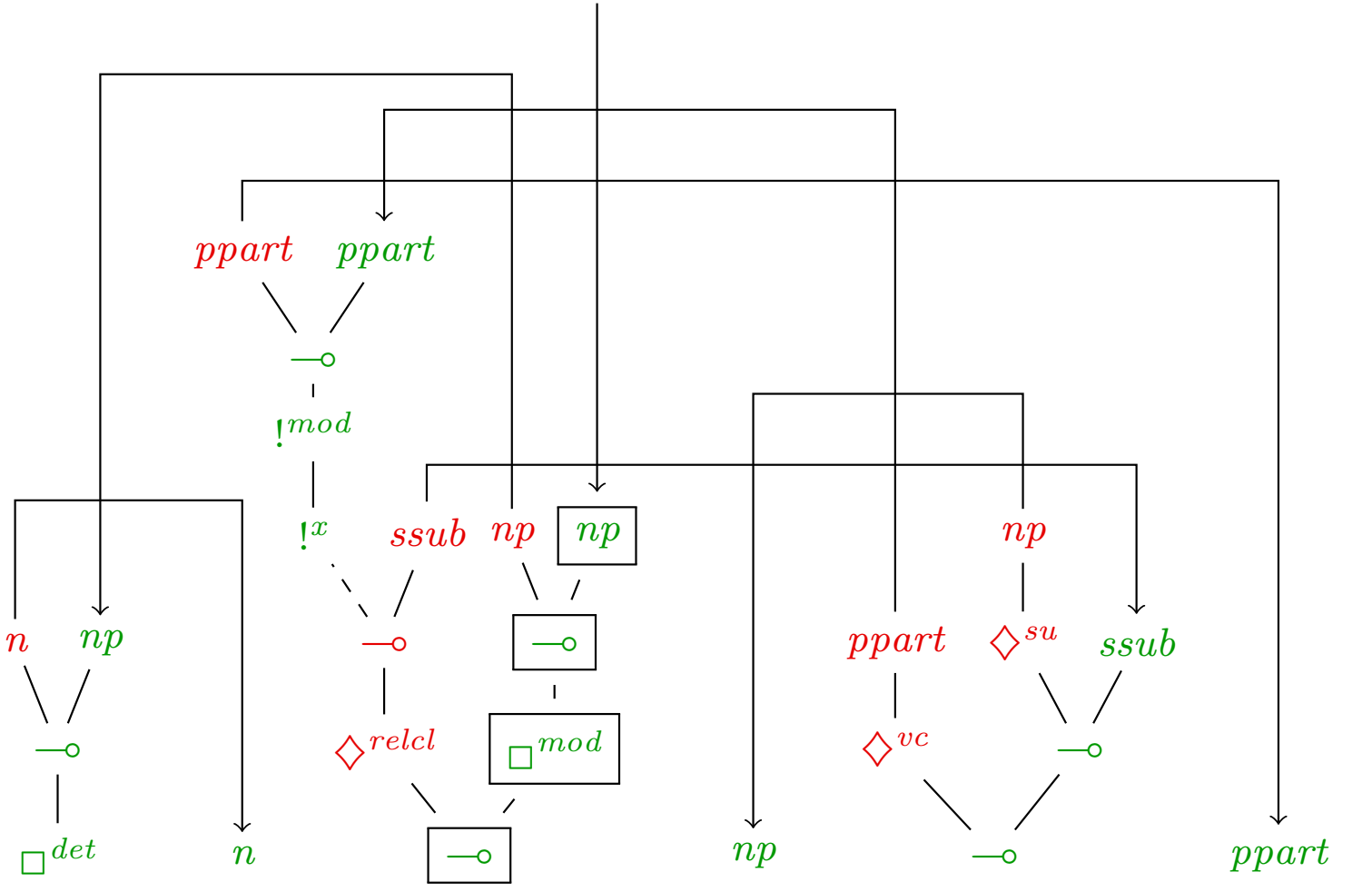
Proof ~~Frames~~ Structures



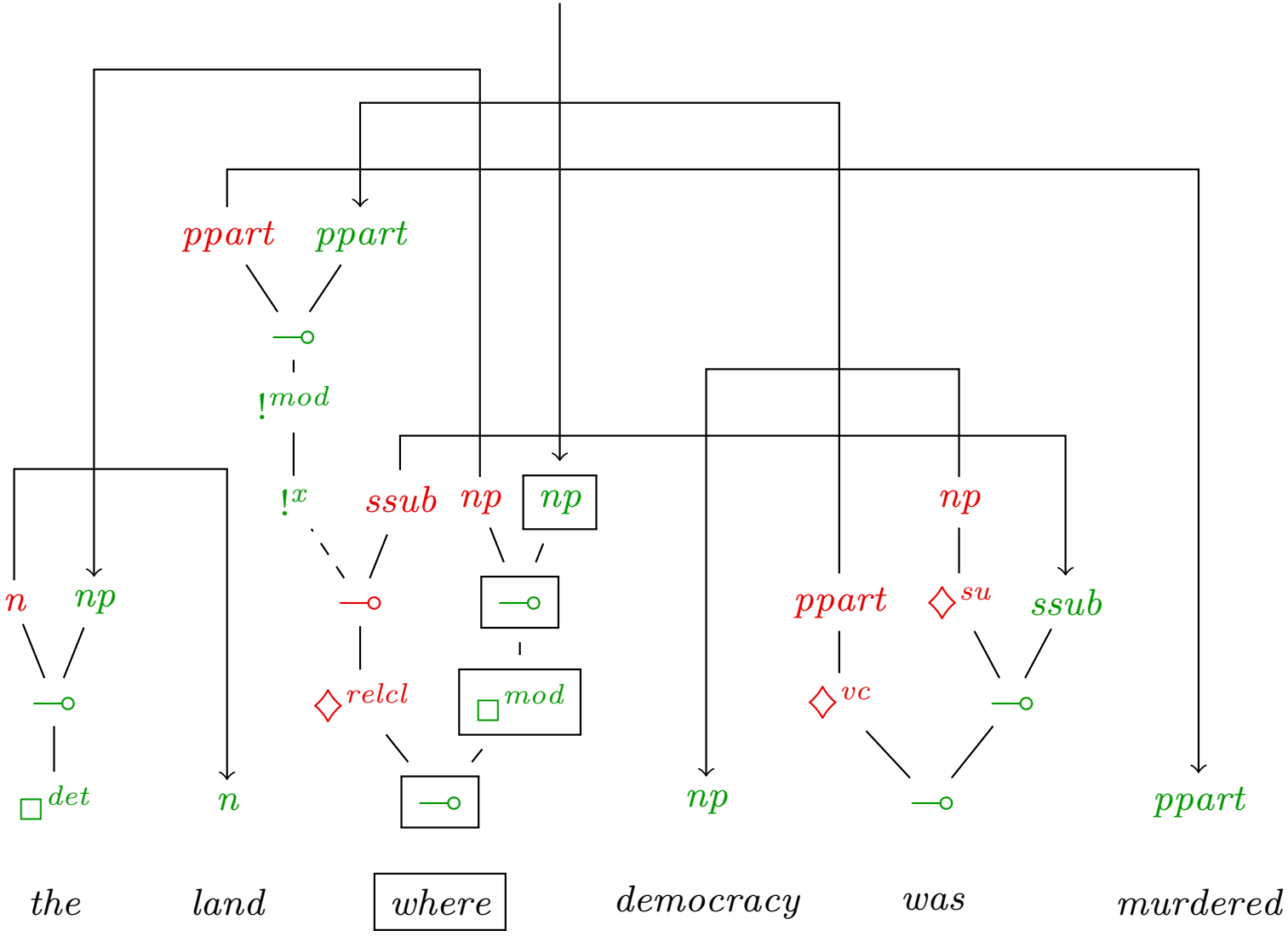
Proof ~~Frames~~ Structures



Proof ~~Frames~~ Structures

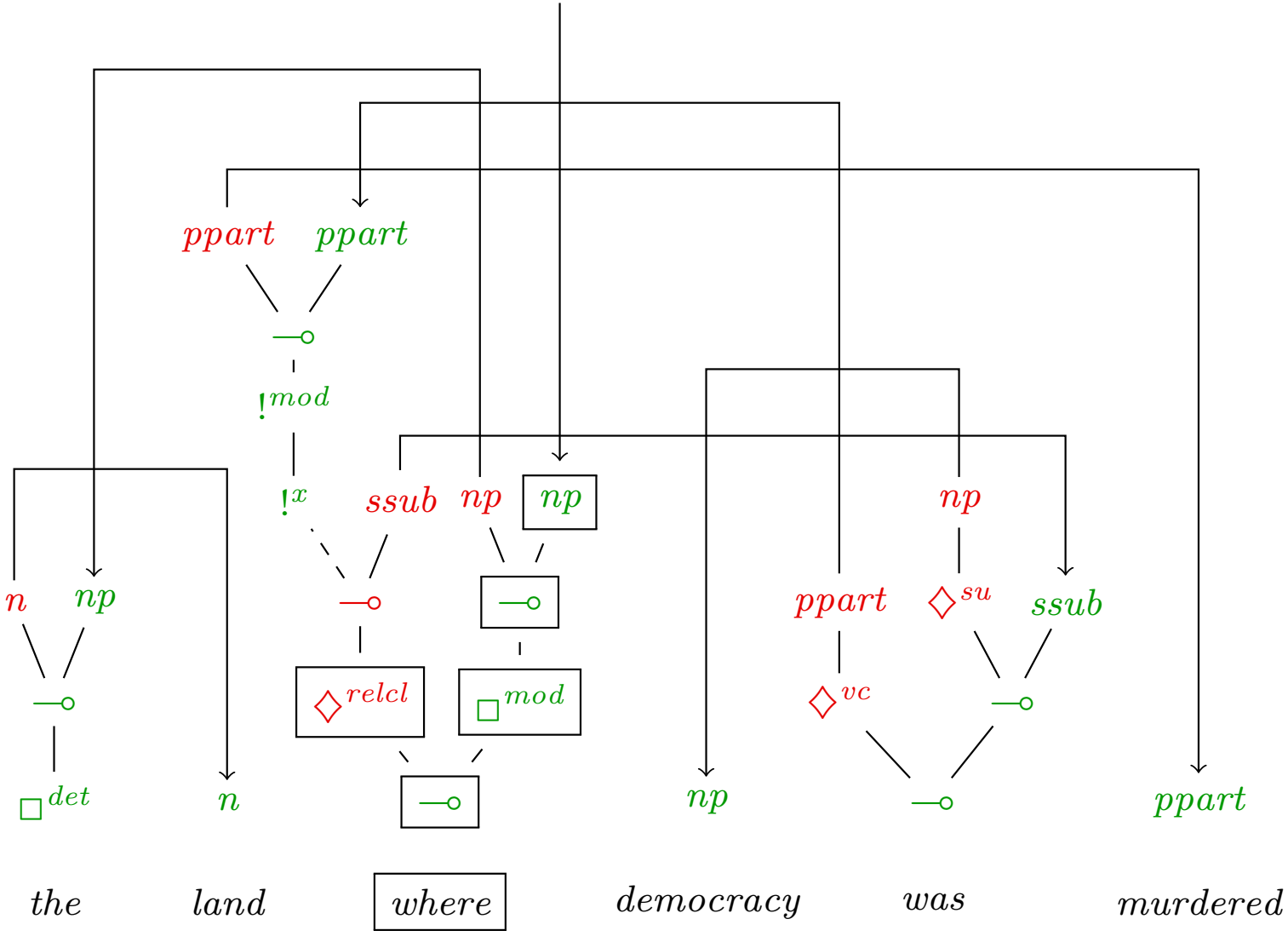


Proof ~~Frames~~ Structures



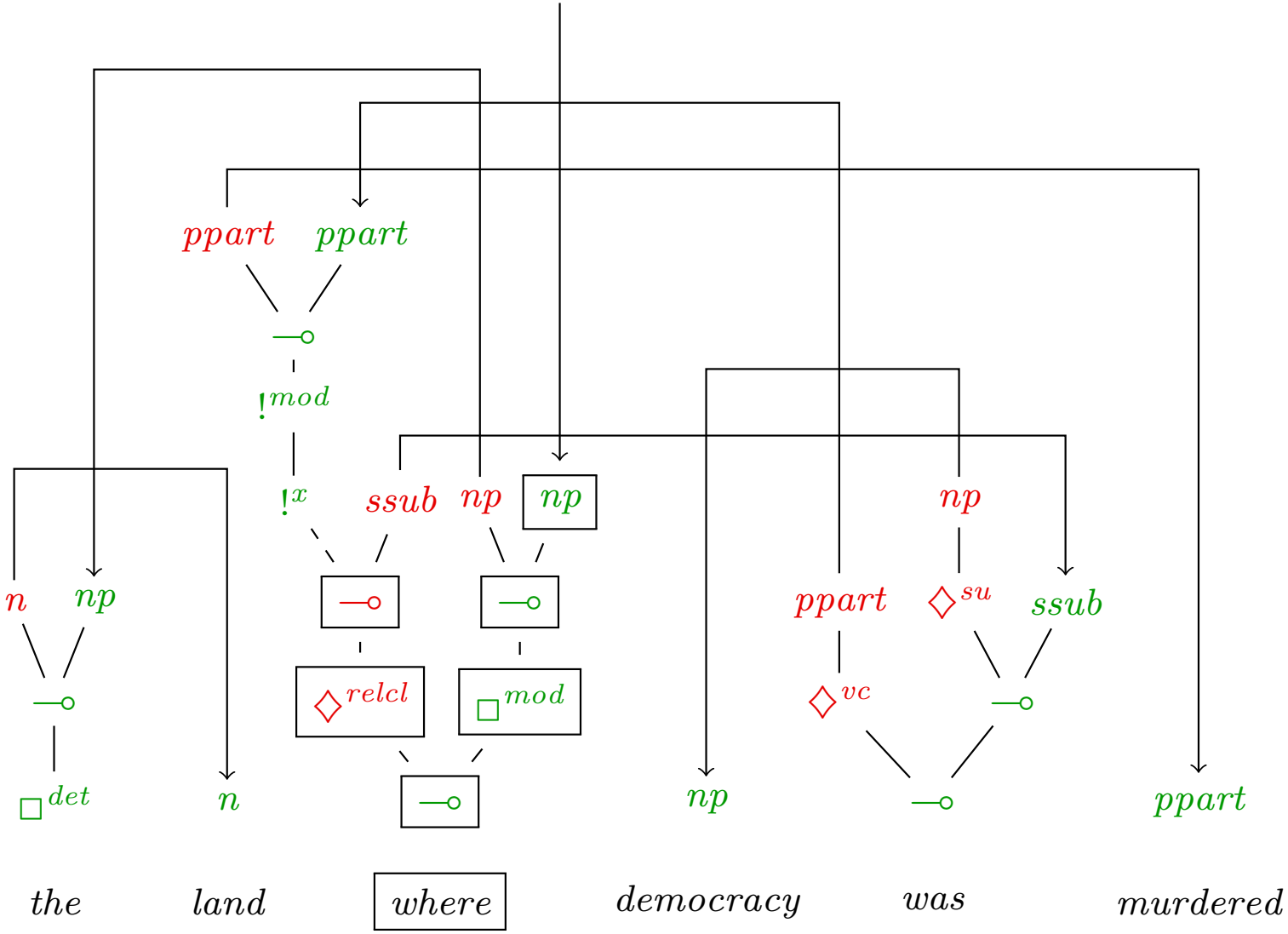
▼^{mod}(where ???) ???

Proof ~~Frames~~ Structures



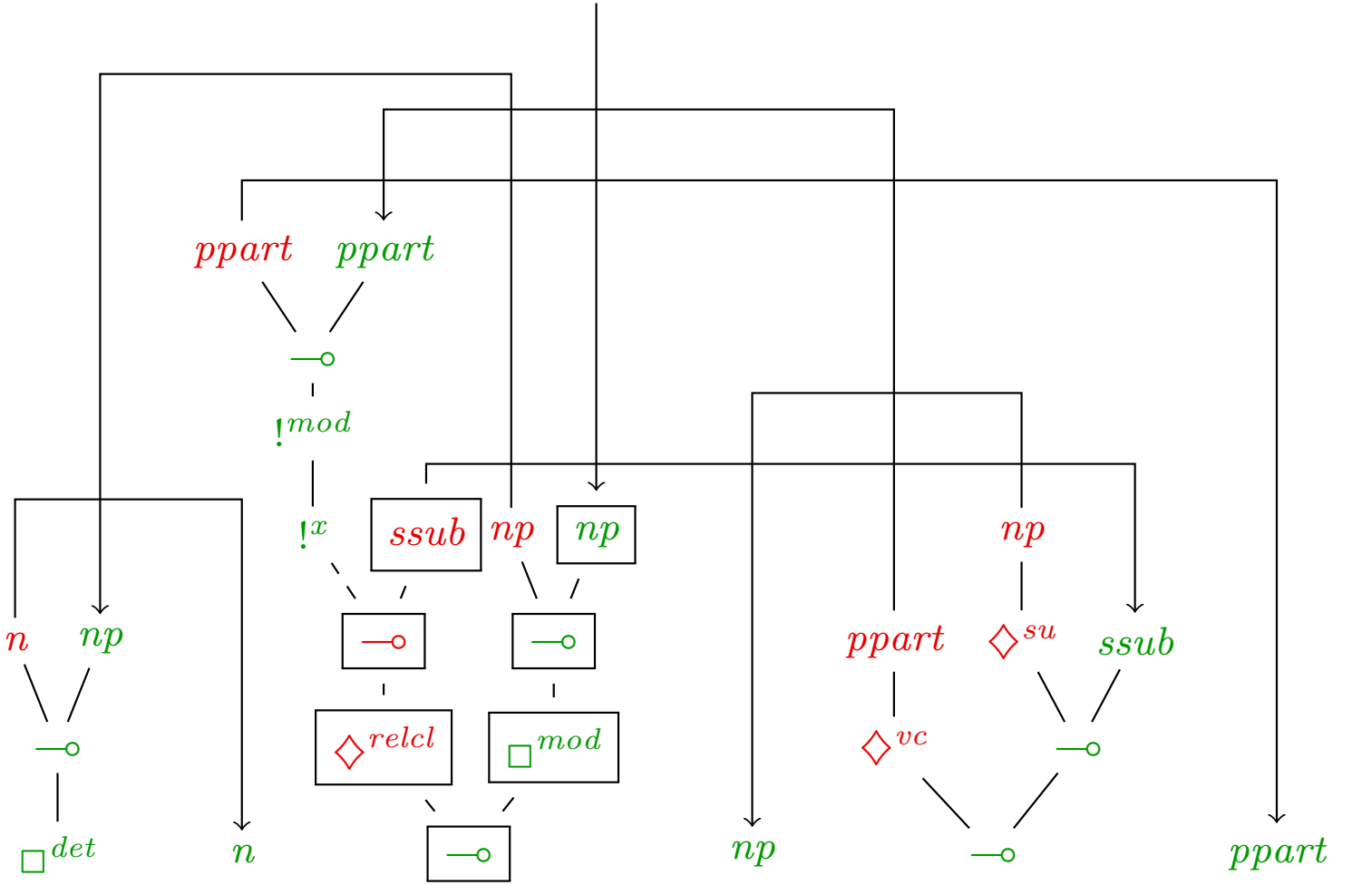
▼^{mod}(where △^{relcl}(???)) ???

Proof ~~Frames~~ Structures



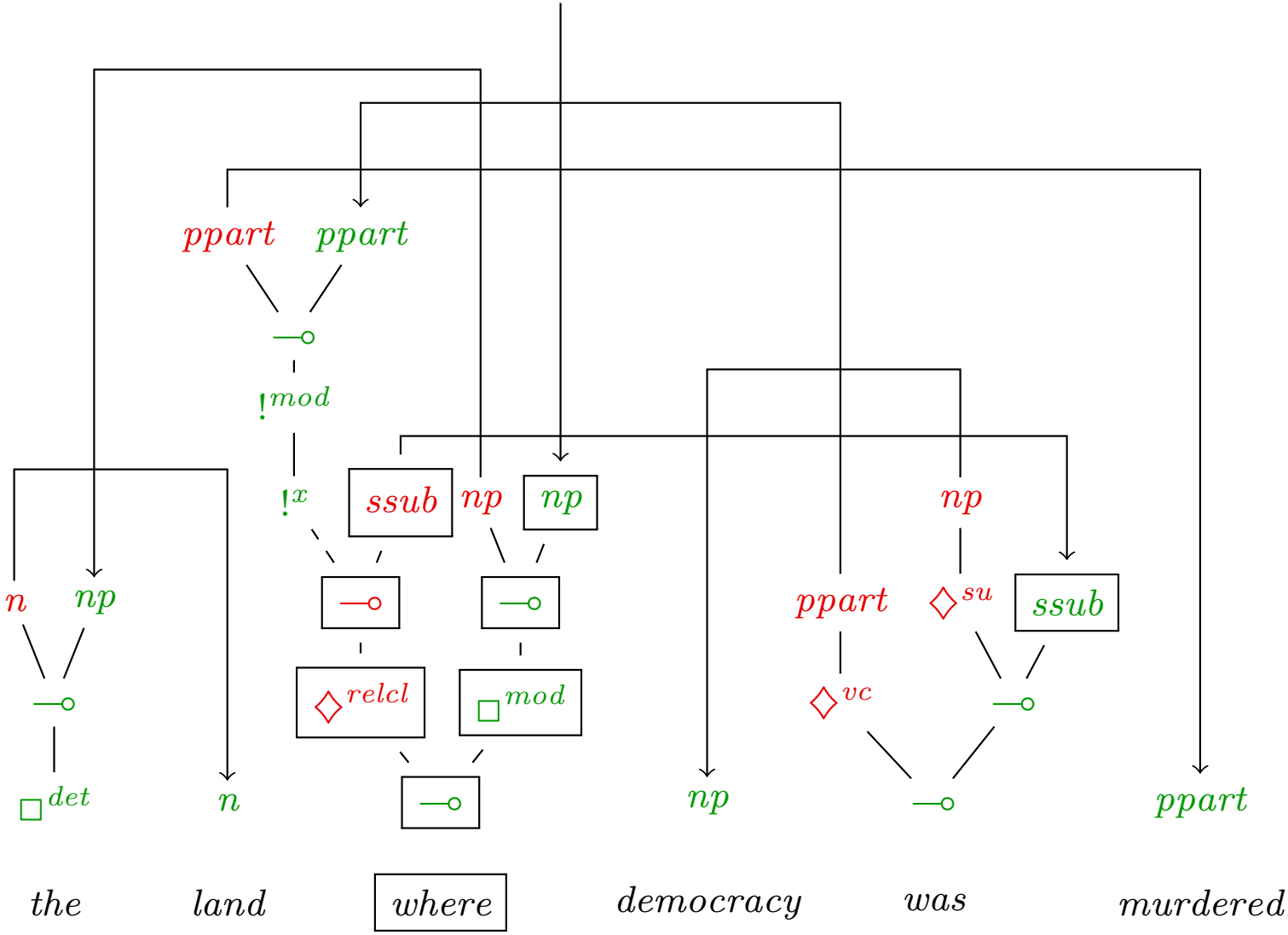
▼^{mod}(where △^{relcl} (λx_0.(???))) ???

Proof ~~Frames~~ Structures



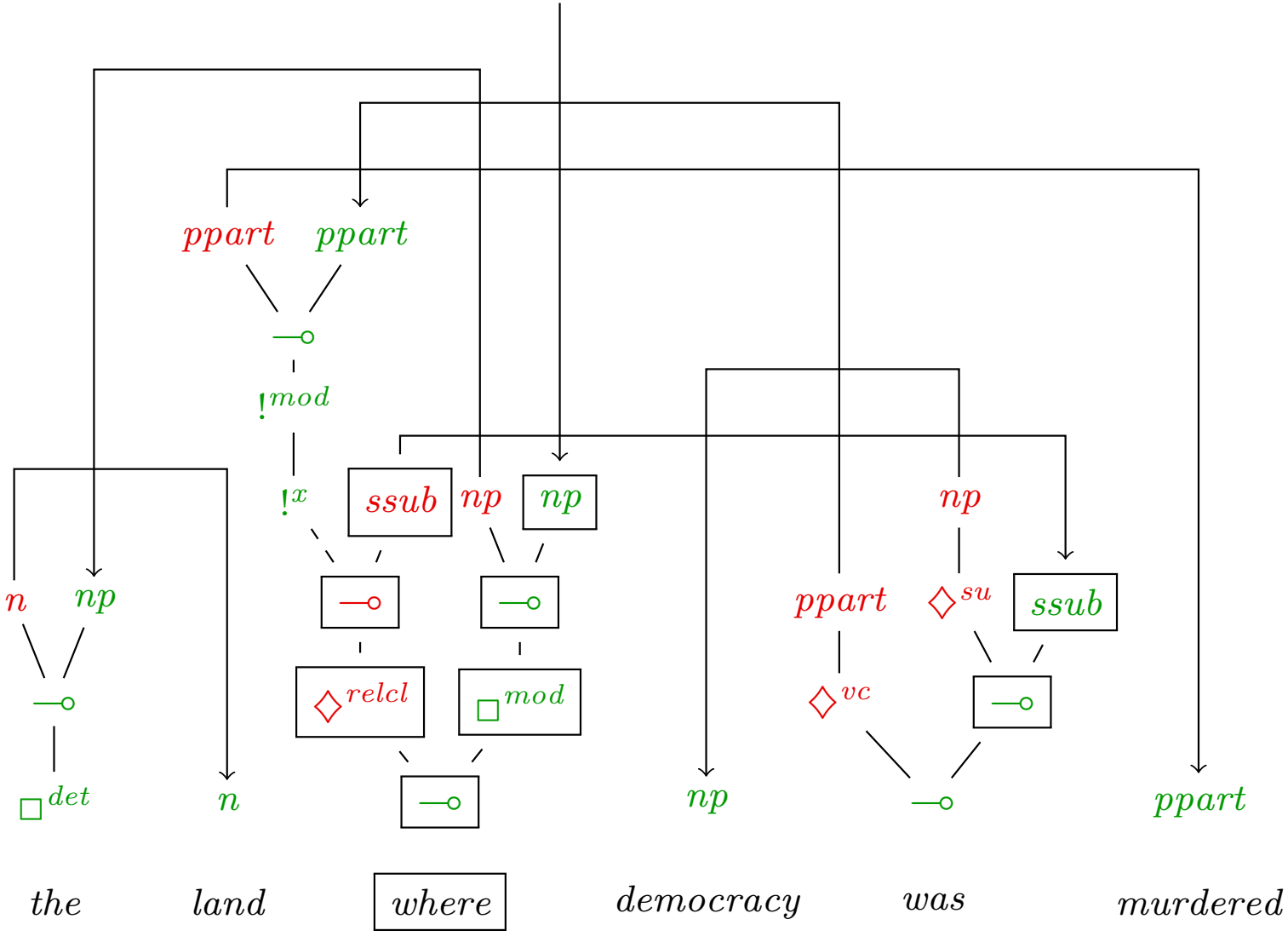
▼^{mod}(where △^{relcl} ($\lambda x_0.(???)$)) ???

Proof ~~Frames~~ Structures



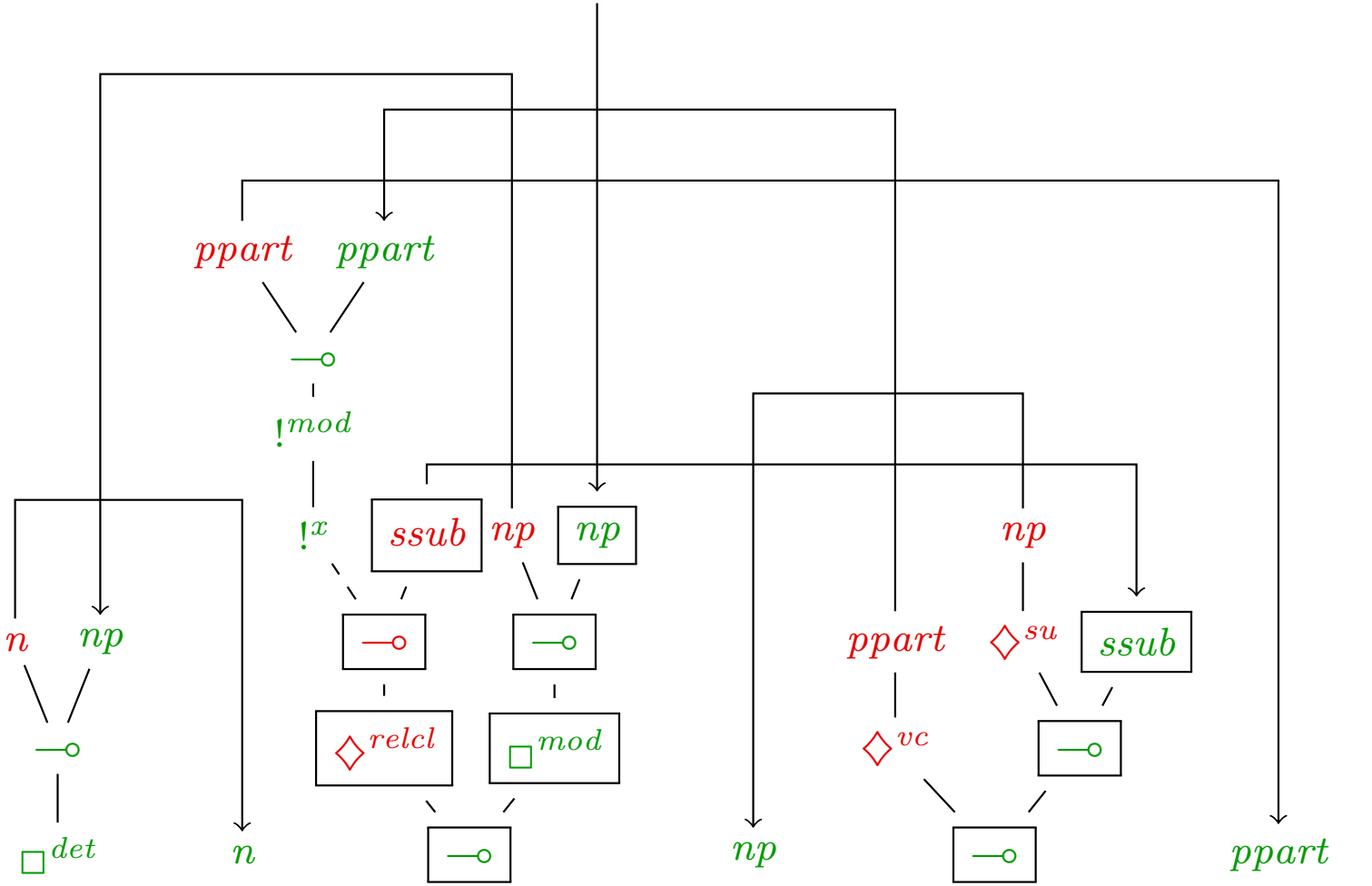
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(???))) ???

Proof ~~Frames~~ Structures



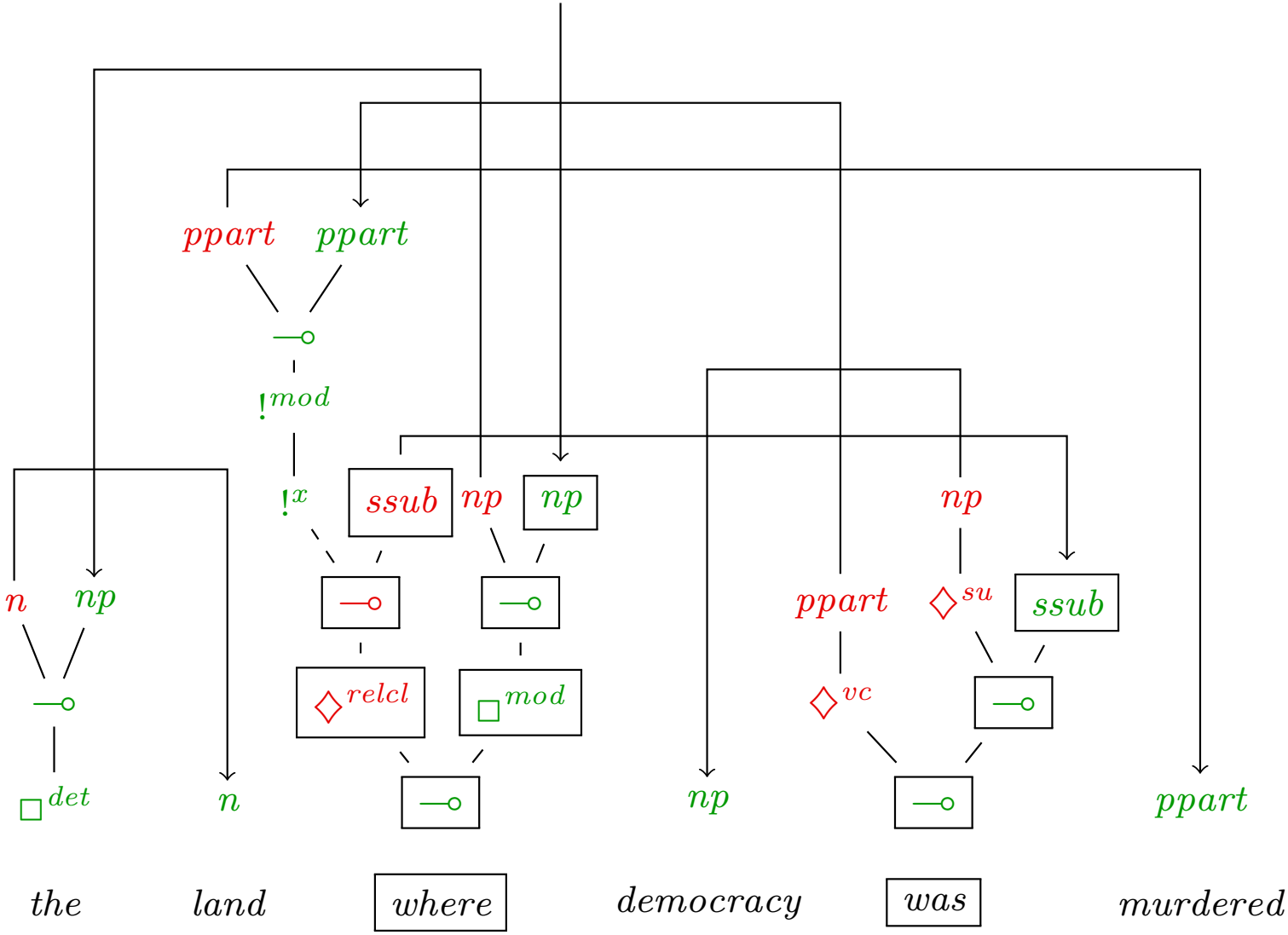
▼^{mod}(where △^{relcl} (λx_0.(??? ???))) ???

Proof ~~Frames~~ Structures



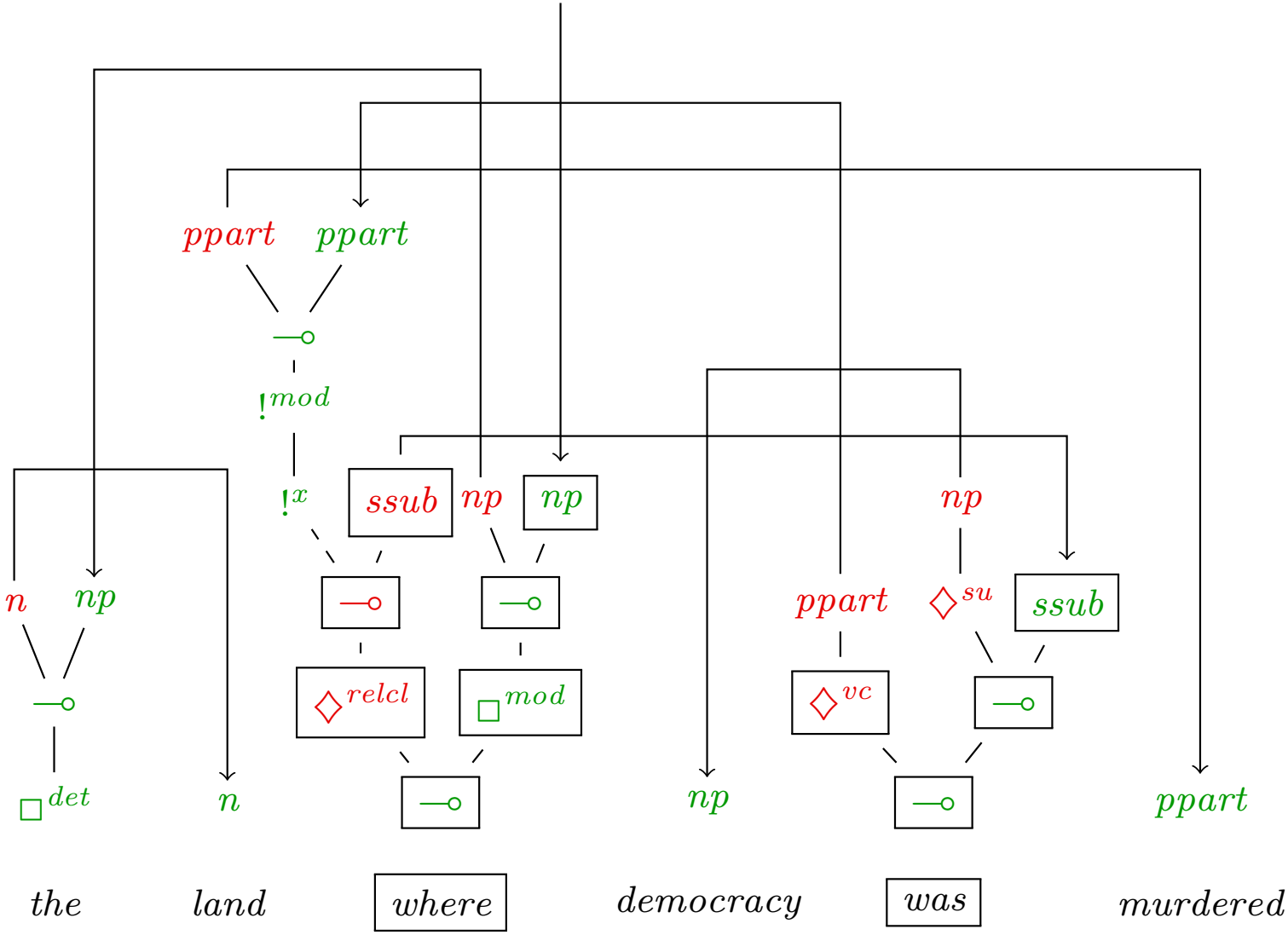
$\blacktriangledown^{mod}(\text{where}\ \ \triangle^{relcl}\ (\lambda x_0.(???\ ???\ ???)))\ ???$

Proof ~~Frames~~ Structures



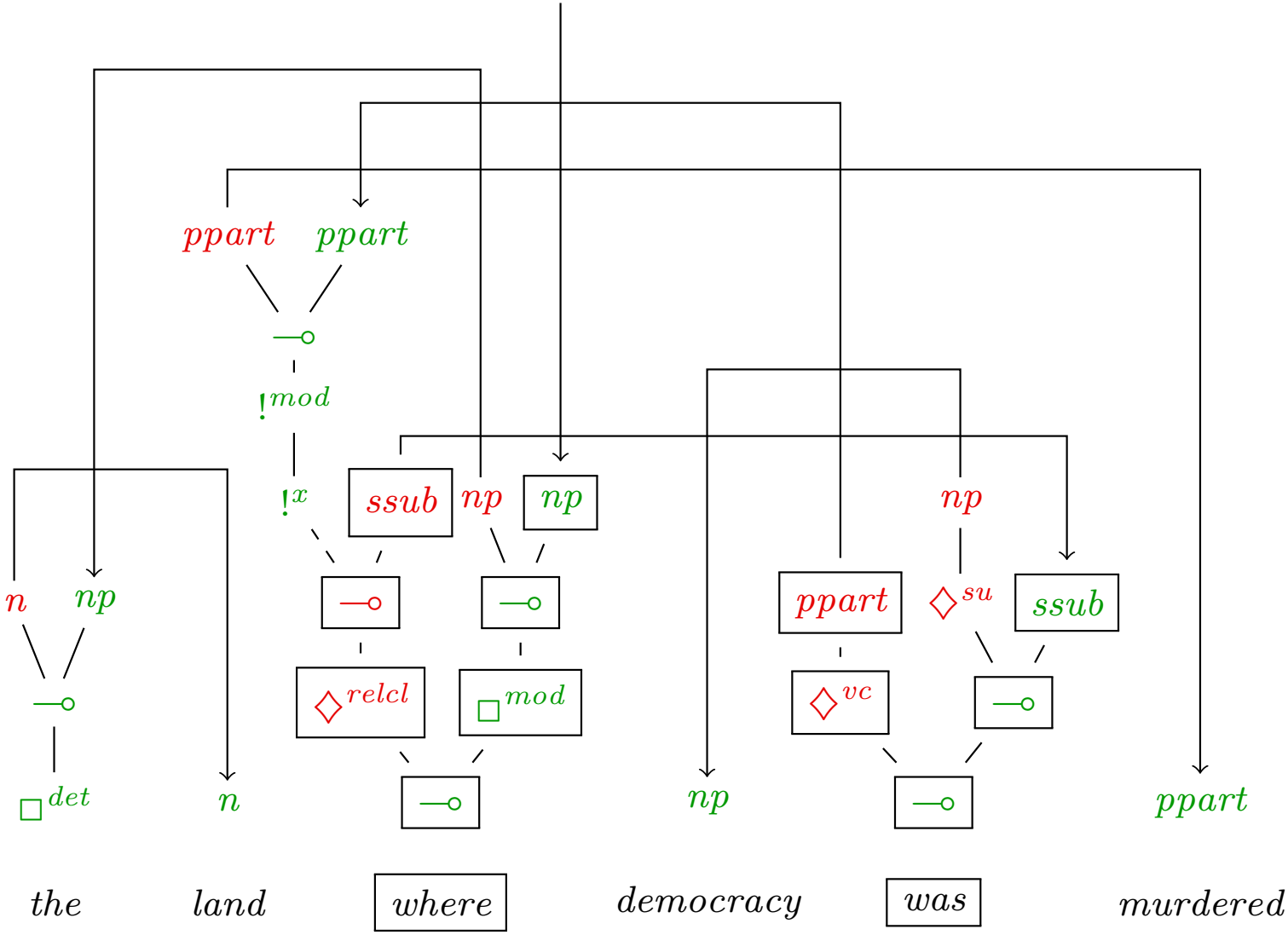
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was ??? ???))) ???

Proof ~~Frames~~ Structures



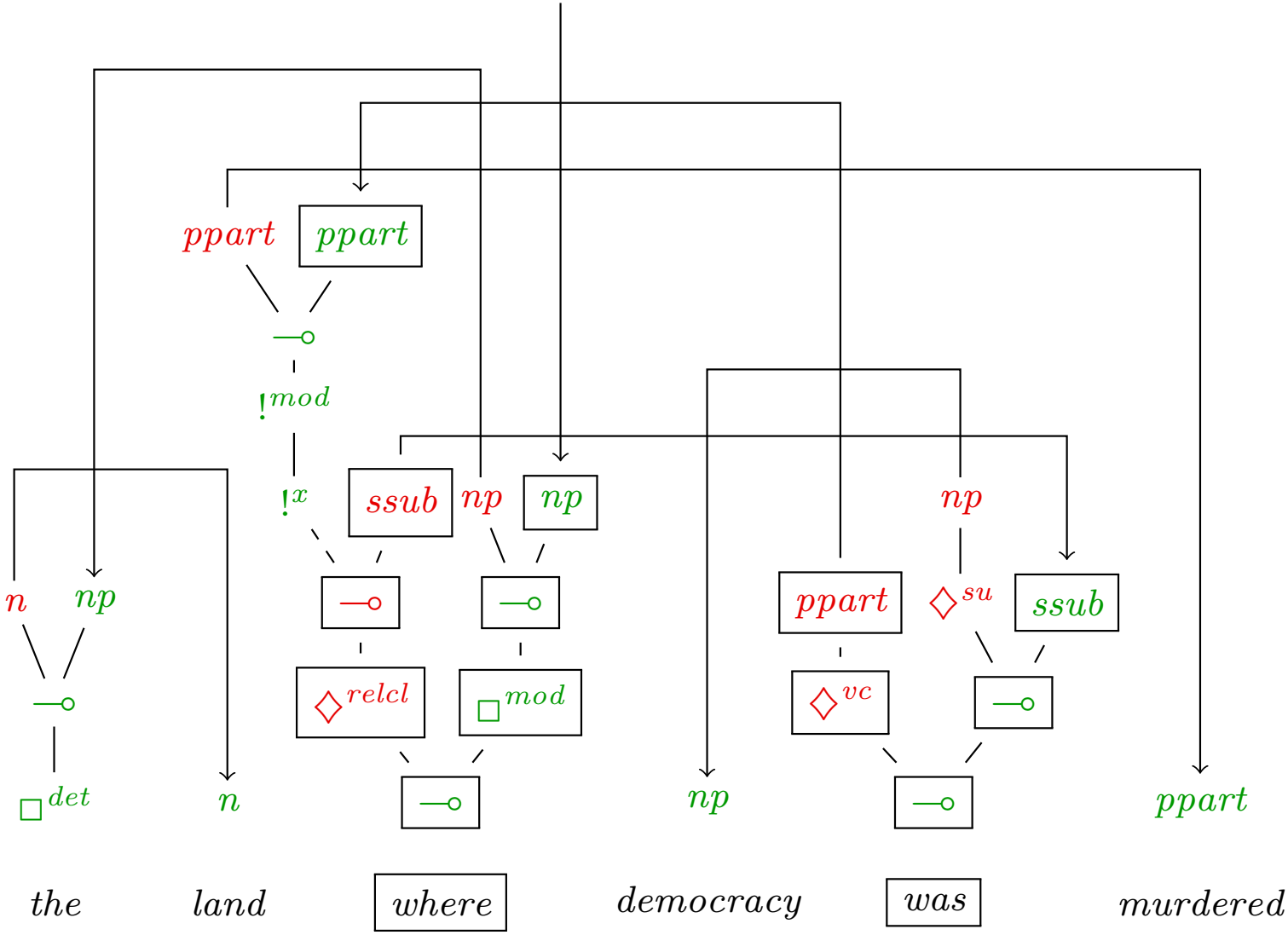
▼^{mod}(where △^{relcl} (λx$_0$.(was △^{vc} (???) ???))) ???

Proof ~~Frames~~ Structures



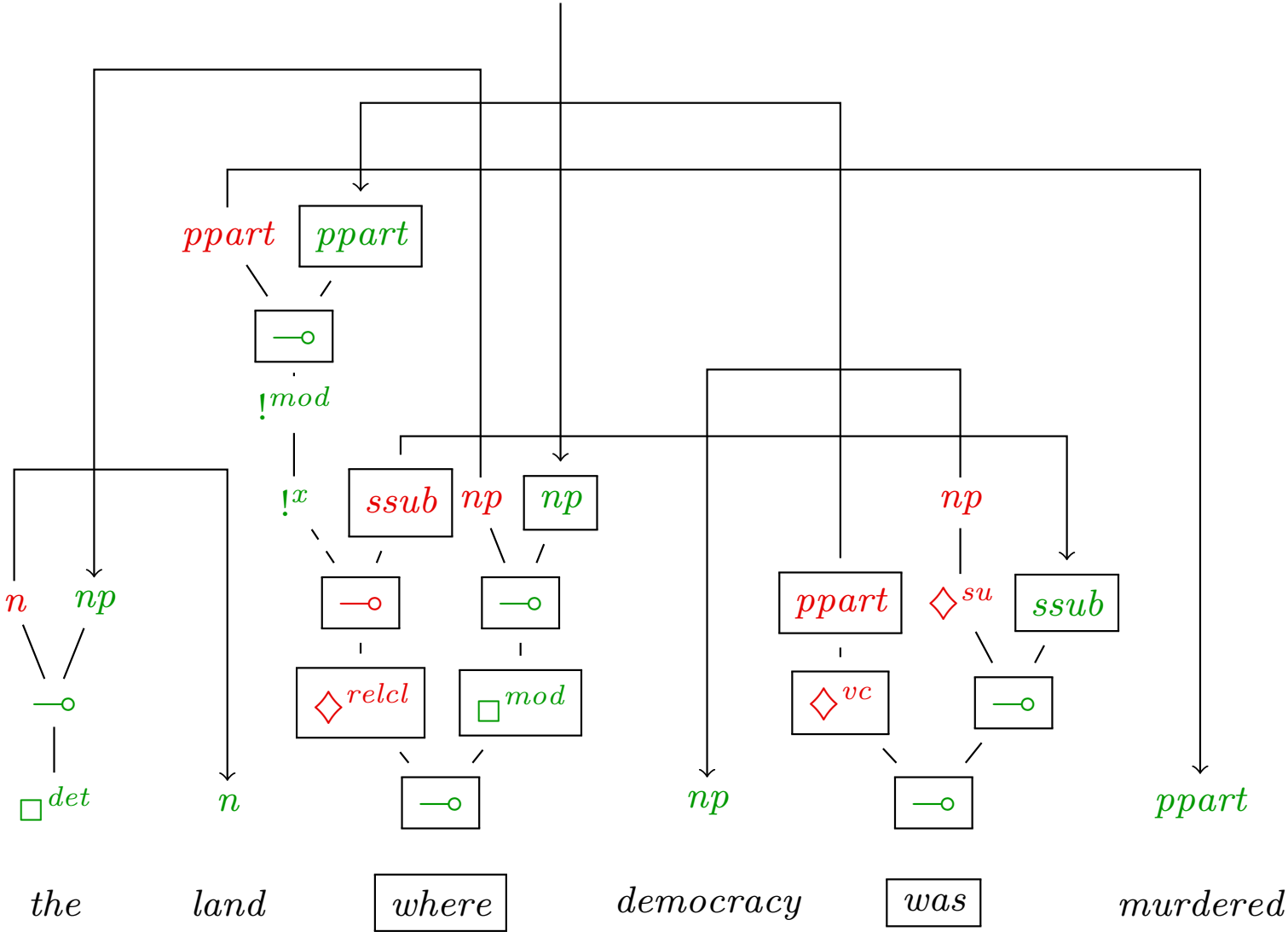
▼^{mod}(where △^{relcl} (λx$_0$.(was △^{vc} (???) ???))) ???

Proof ~~Frames~~ Structures



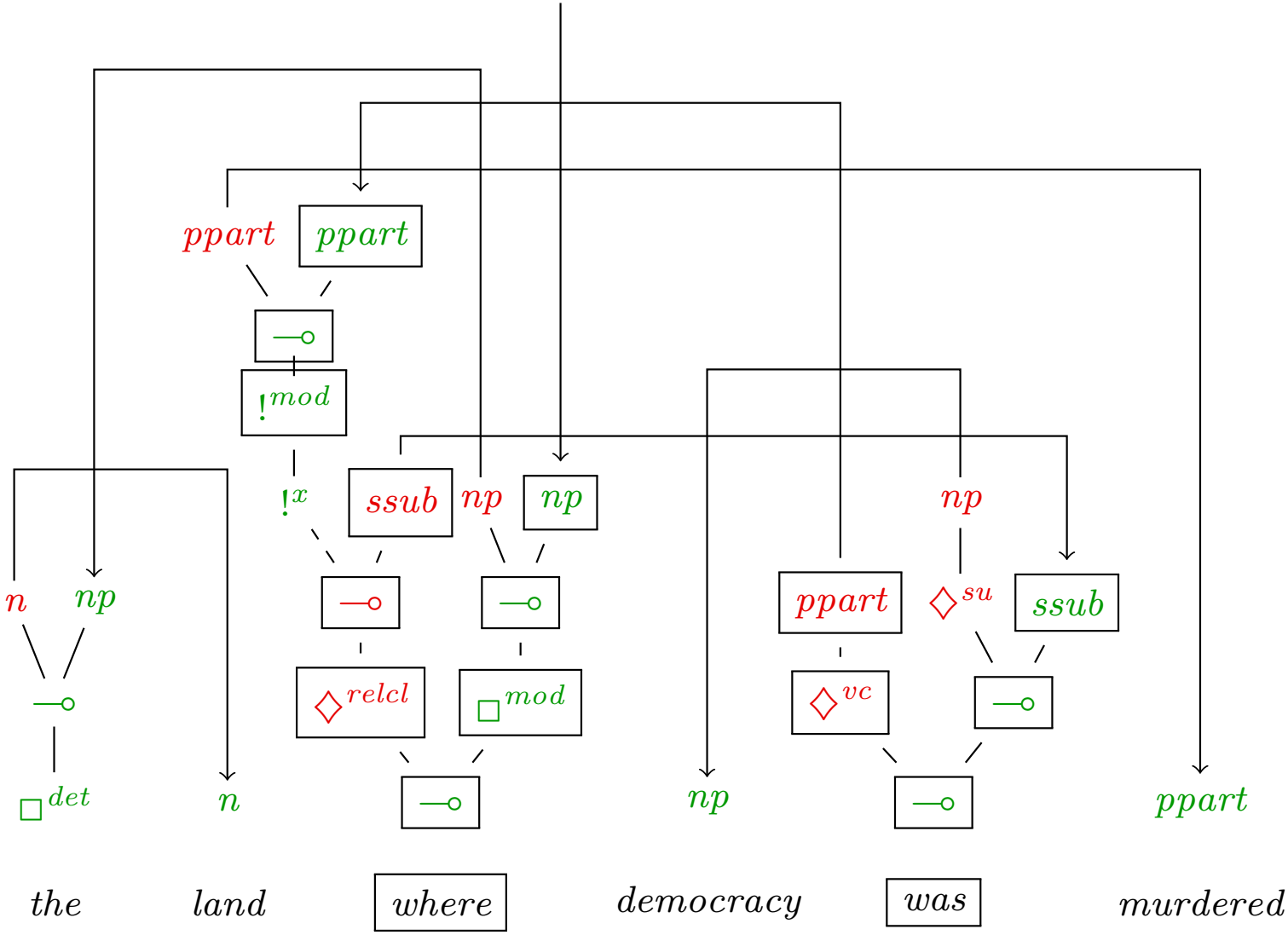
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ (???) ???))) ???

Proof ~~Frames~~ Structures



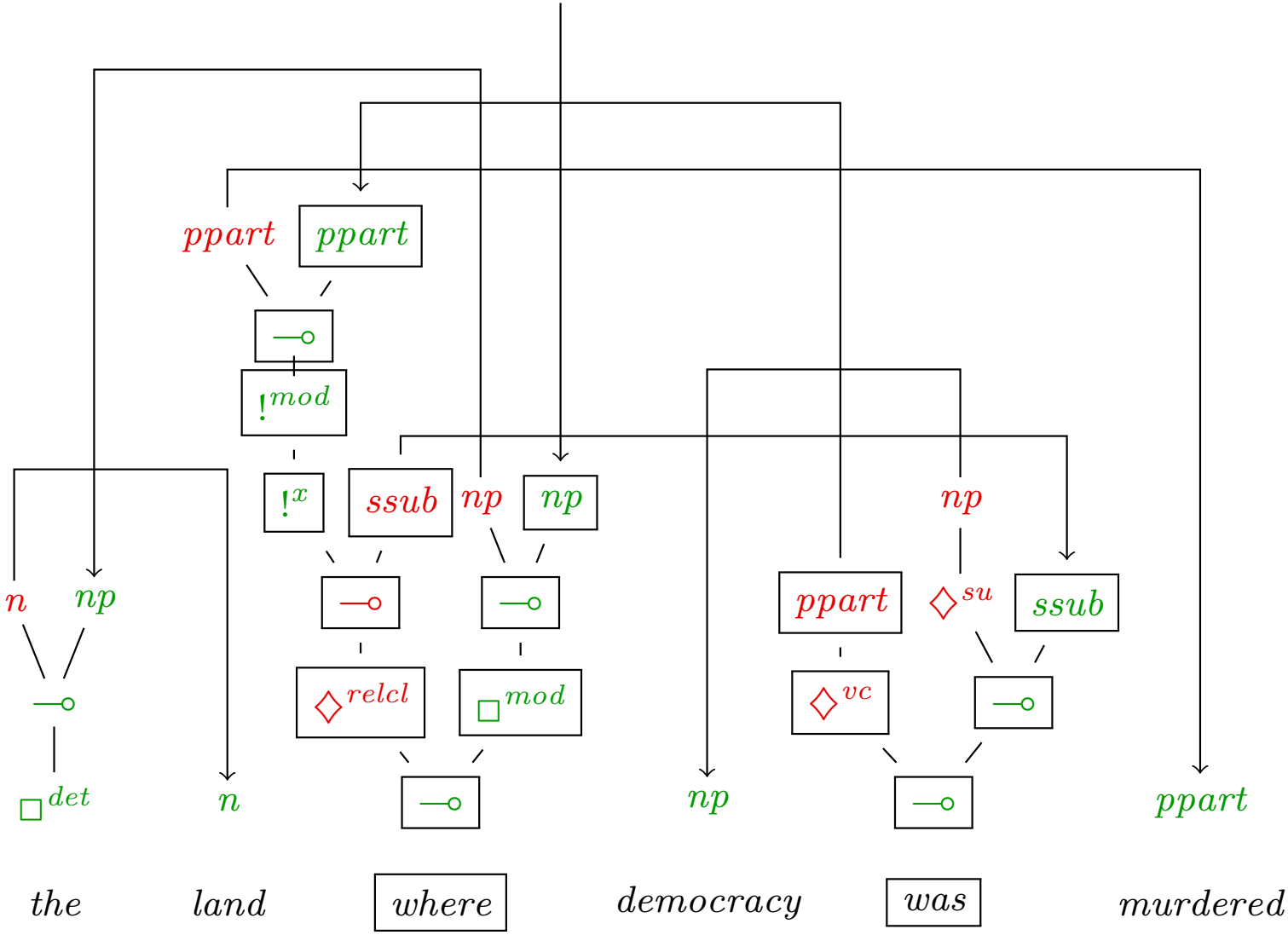
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ (??? ???) ???))) ???

Proof ~~Frames~~ Structures



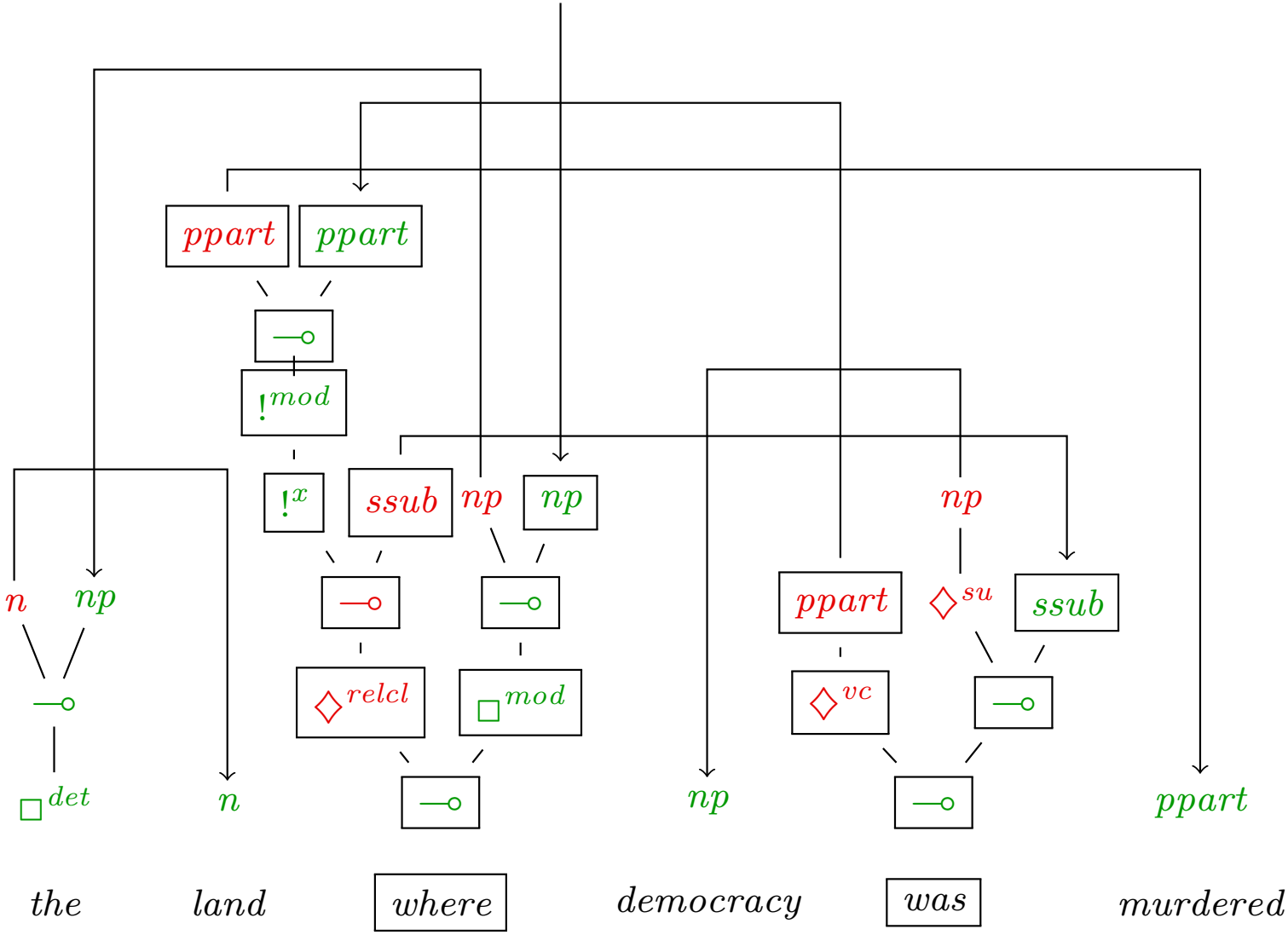
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}$(???) ???) ???))) ???

Proof ~~Frames~~ Structures



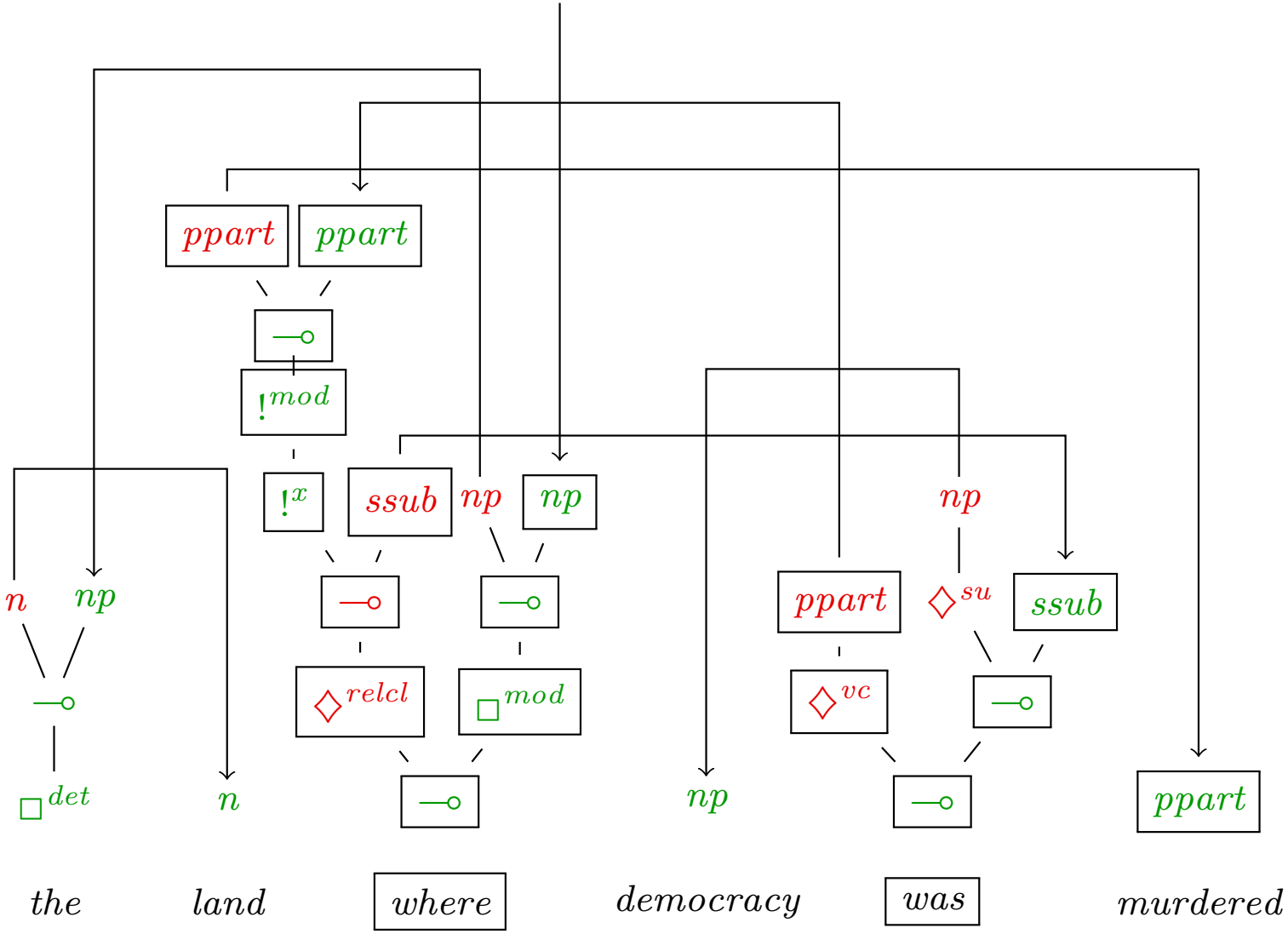
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}$??? ???) ???))) ???

Proof ~~Frames~~ Structures



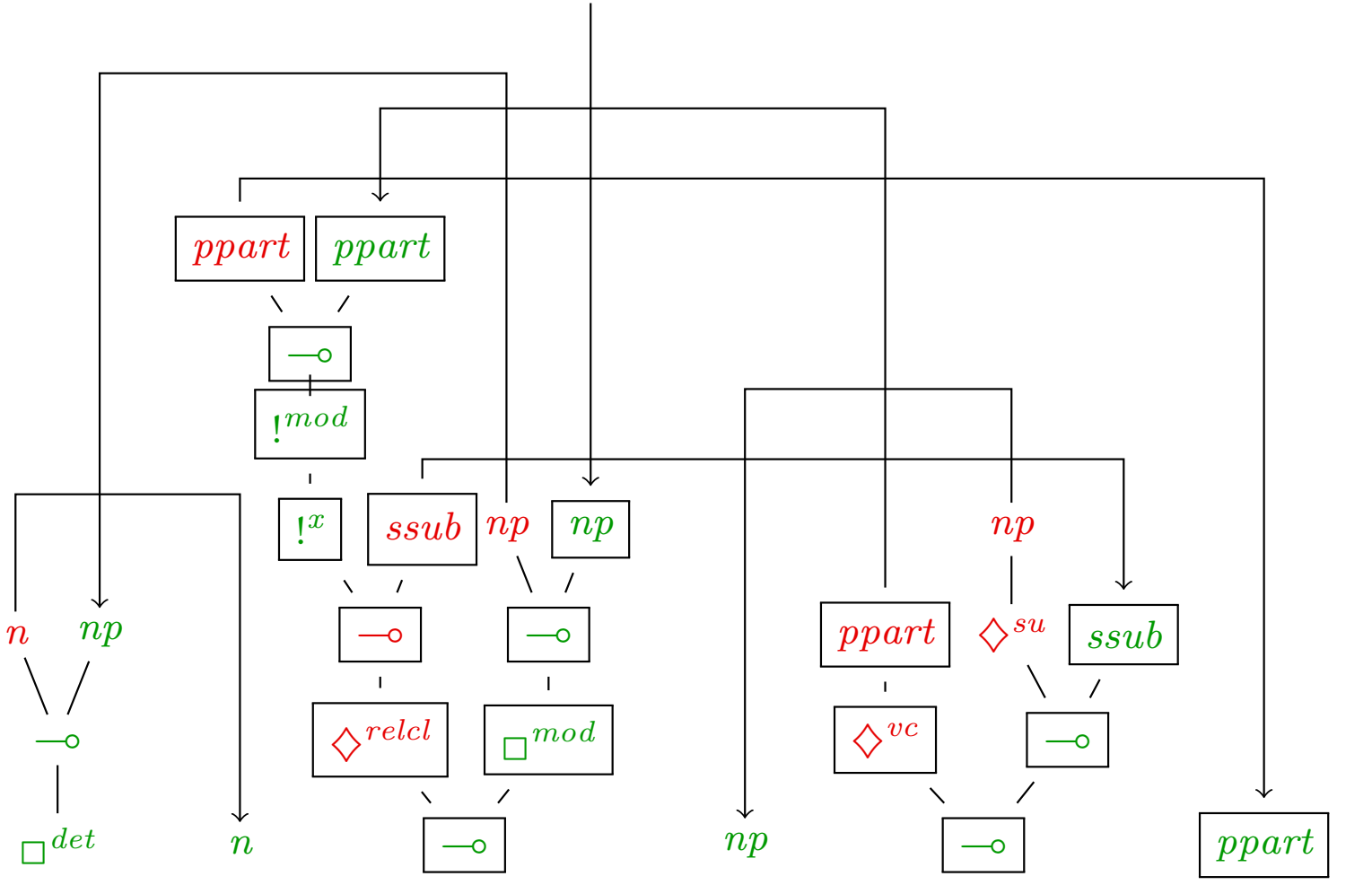
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$???) ???))) ???

Proof ~~Frames~~ Structures



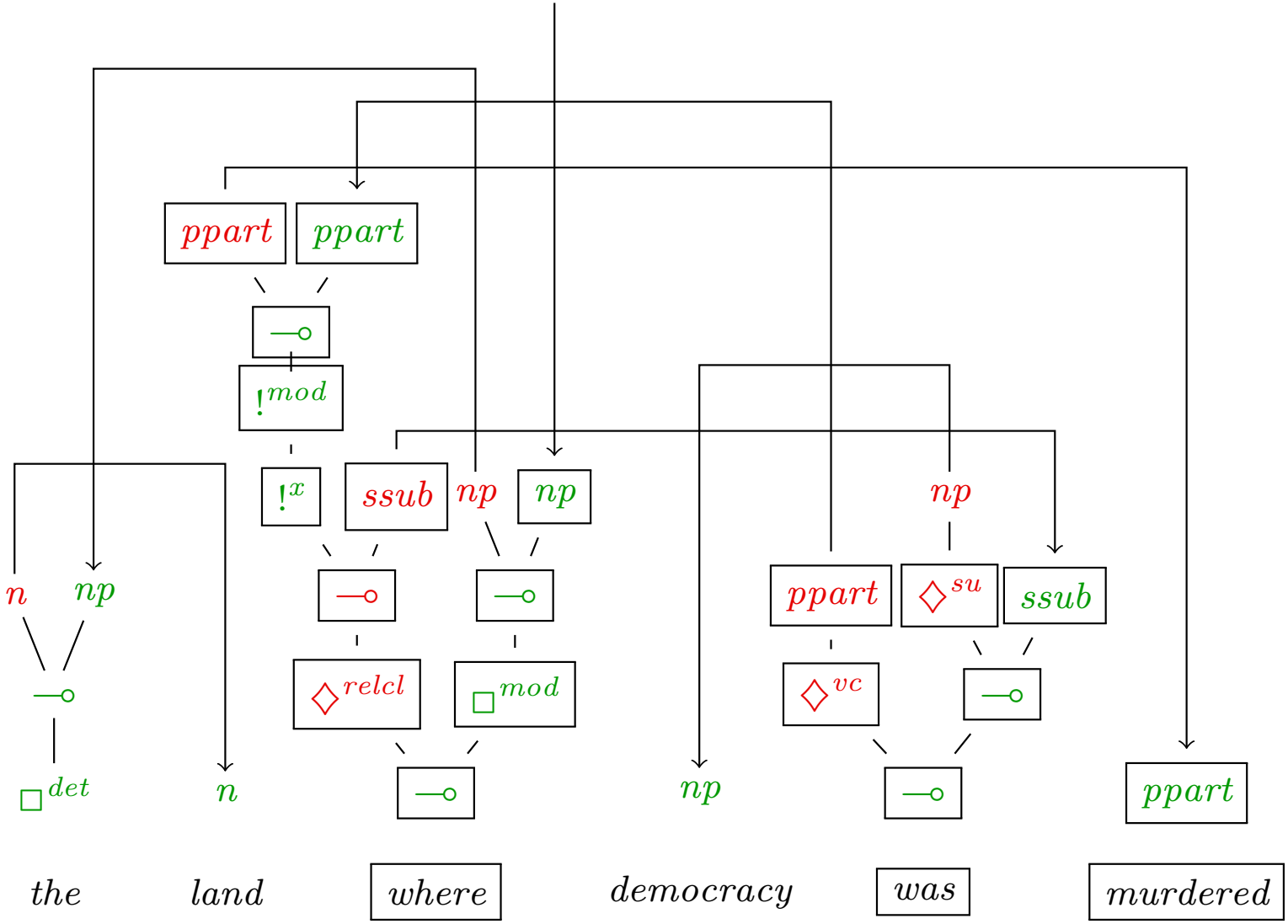
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$???) ???))) ???

Proof ~~Frames~~ Structures



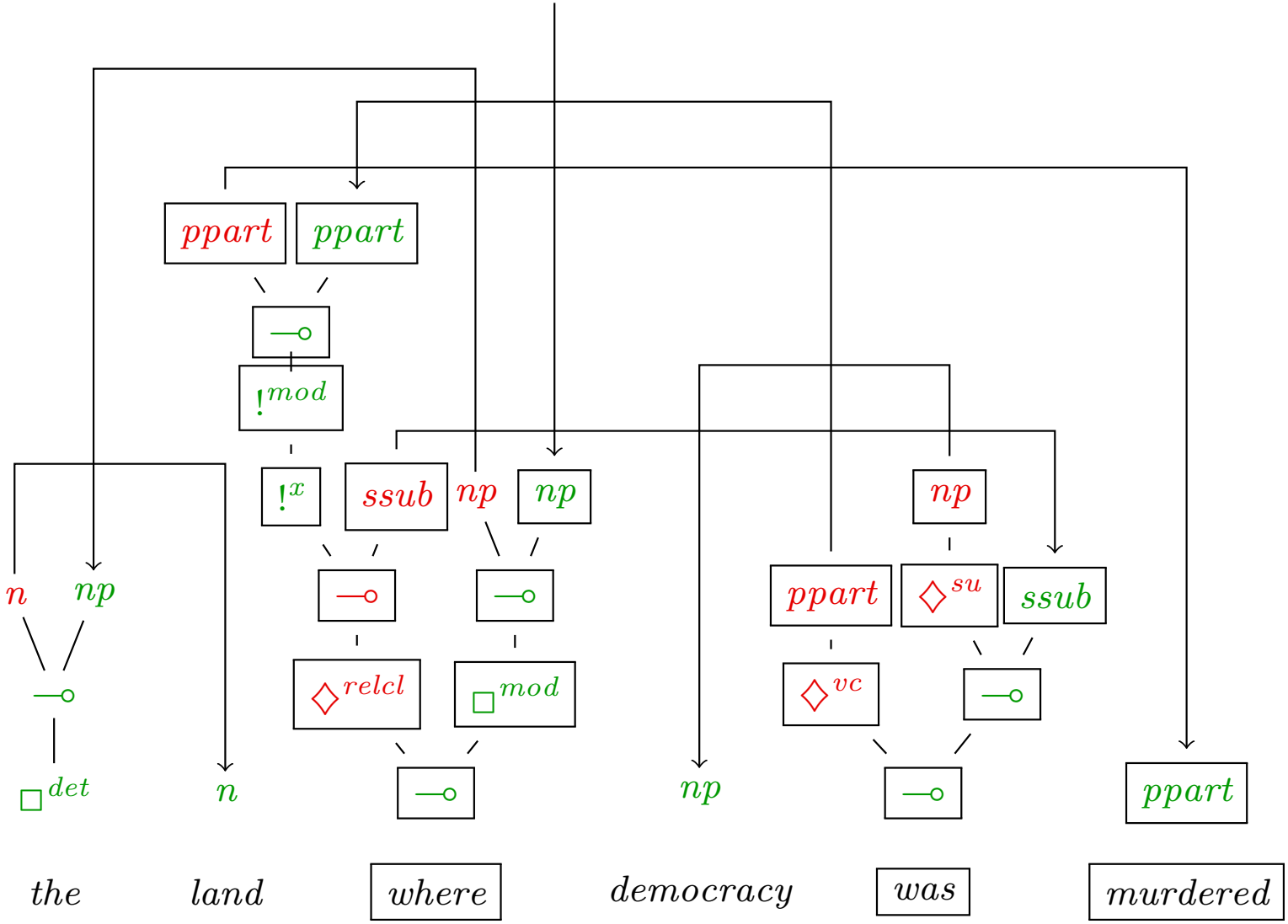
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) ???))) ???

Proof ~~Frames~~ Structures



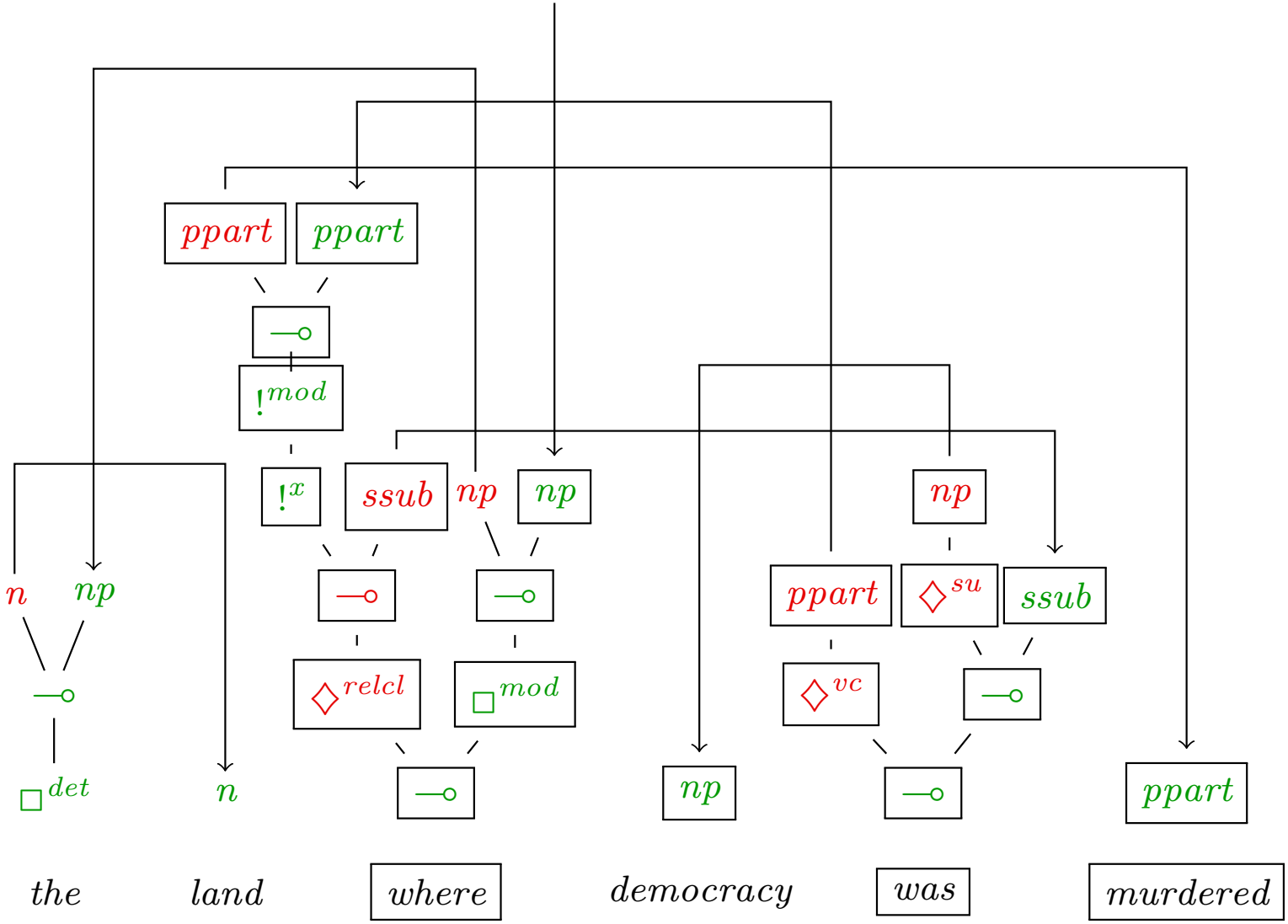
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$???))) ???

Proof ~~Frames~~ Structures



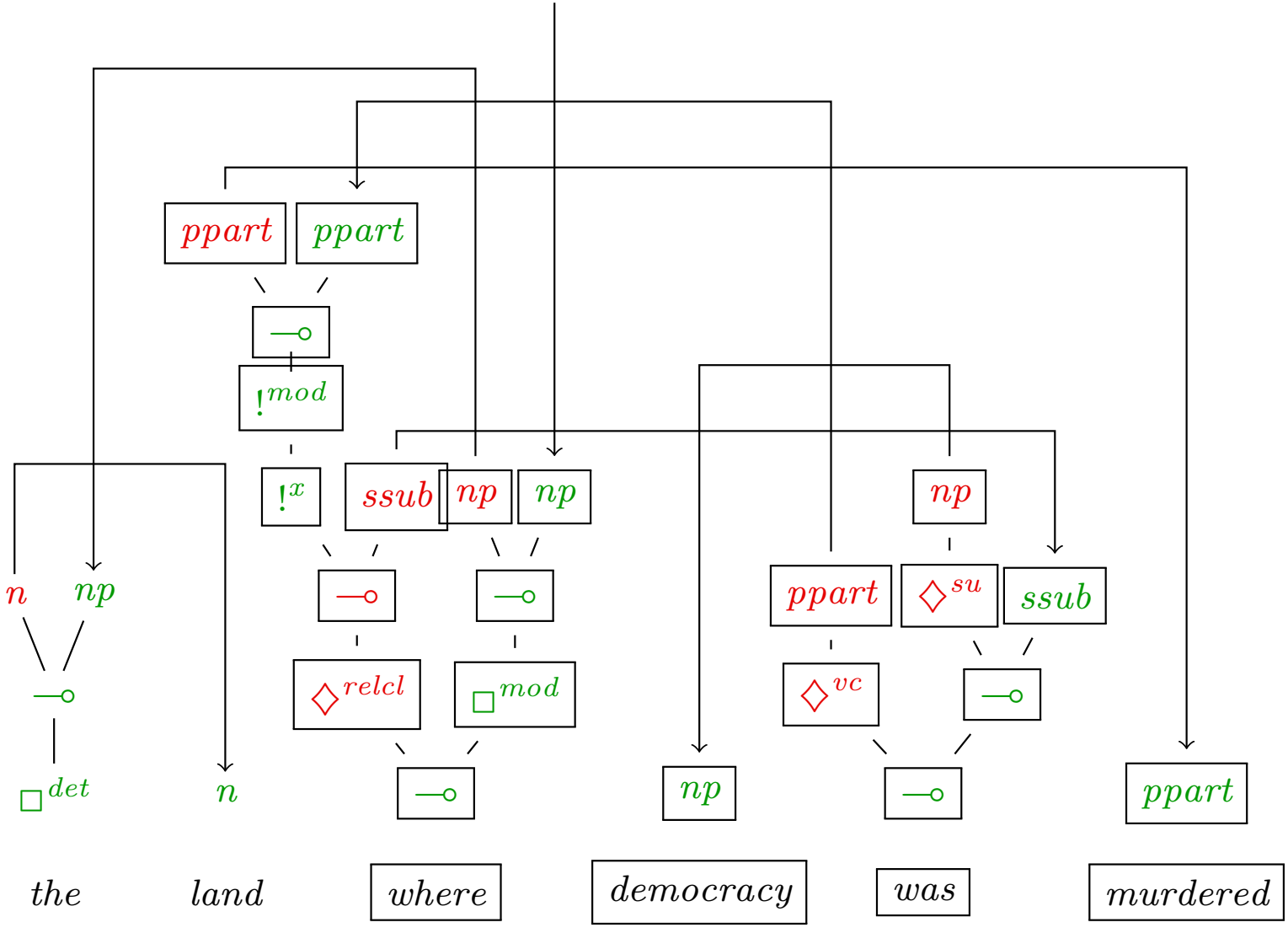
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$???))) ???

Proof ~~Frames~~ Structures



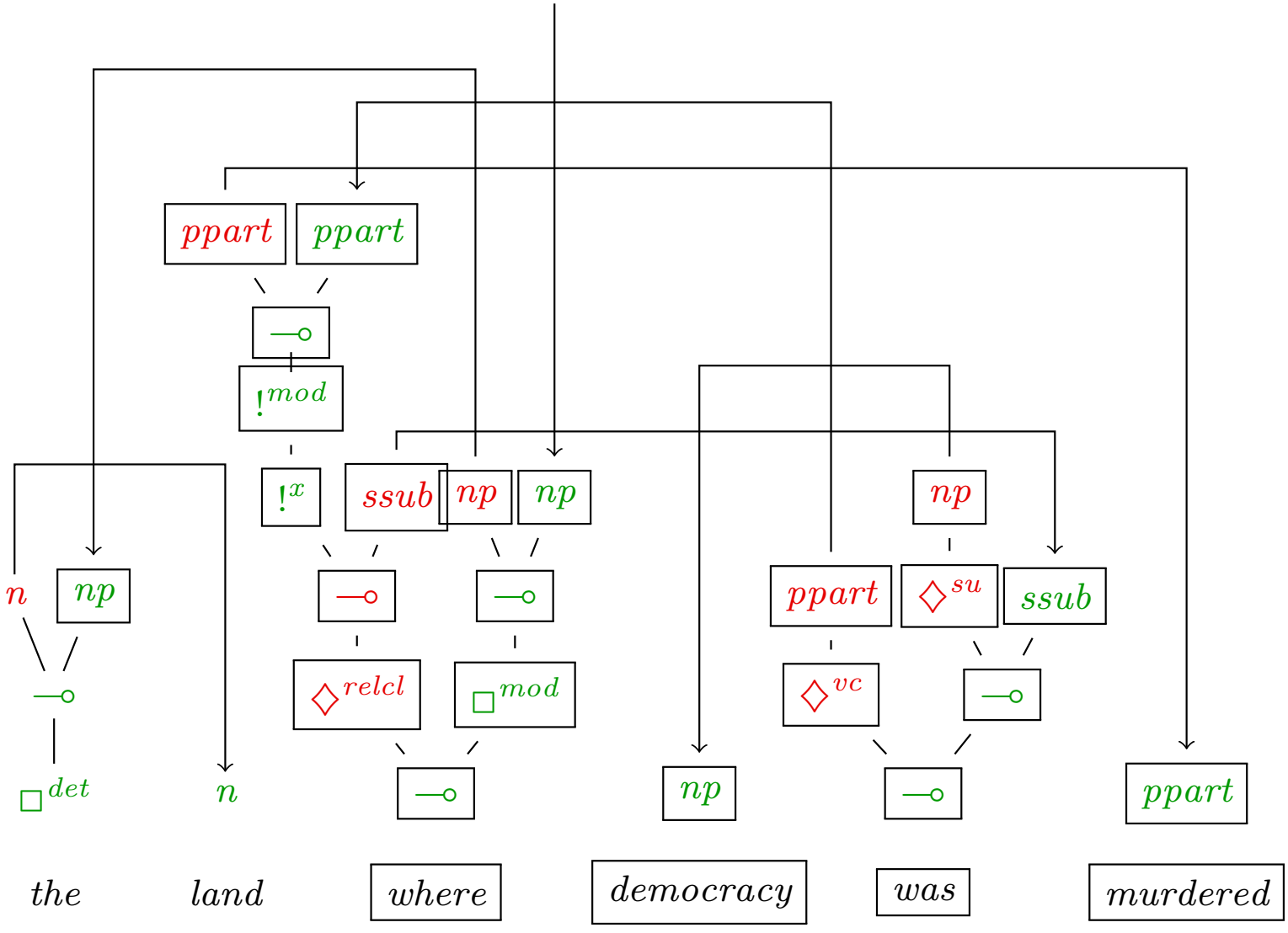
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ???

Proof ~~Frames~~ Structures



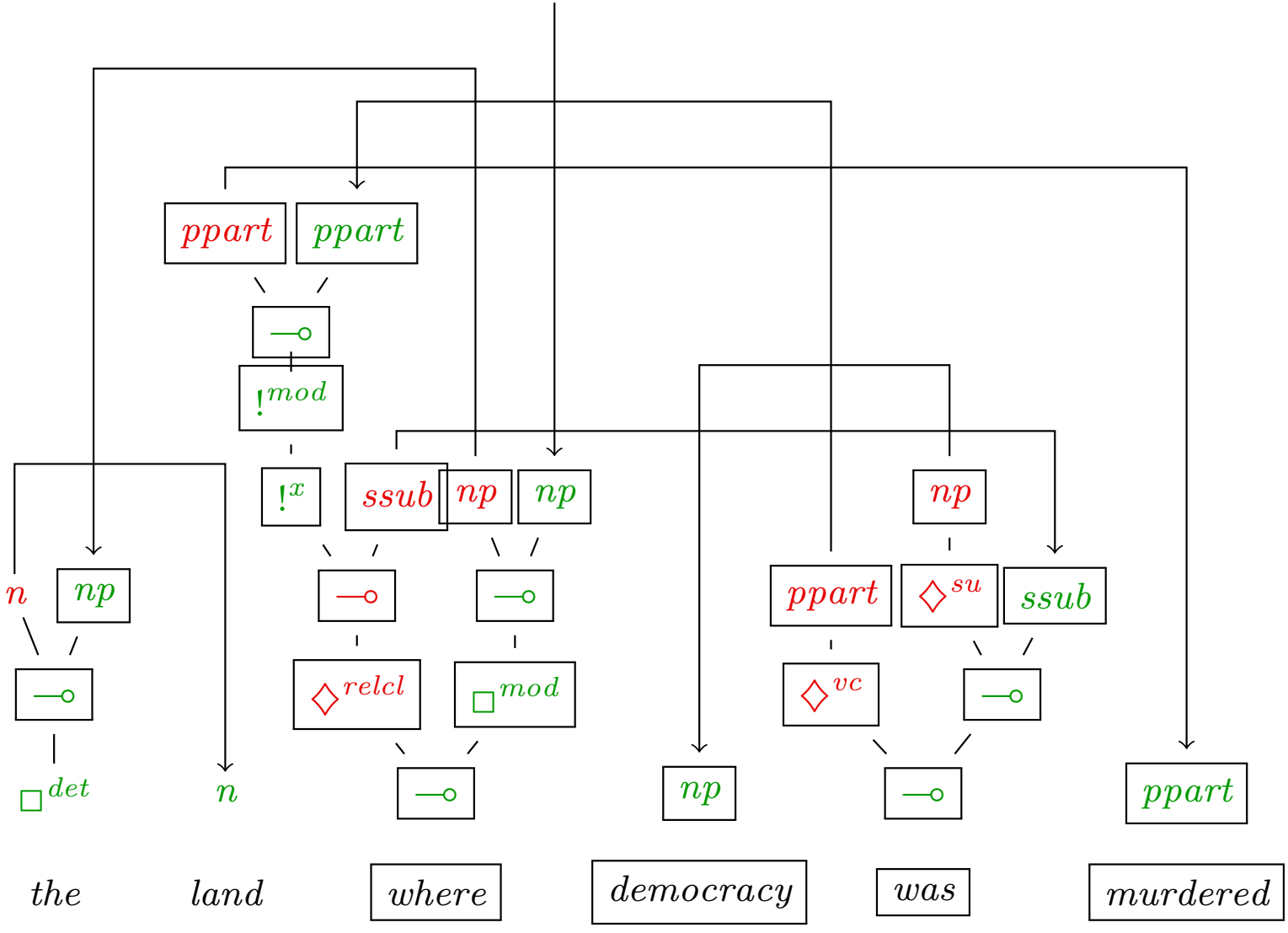
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ???

Proof ~~Frames~~ Structures



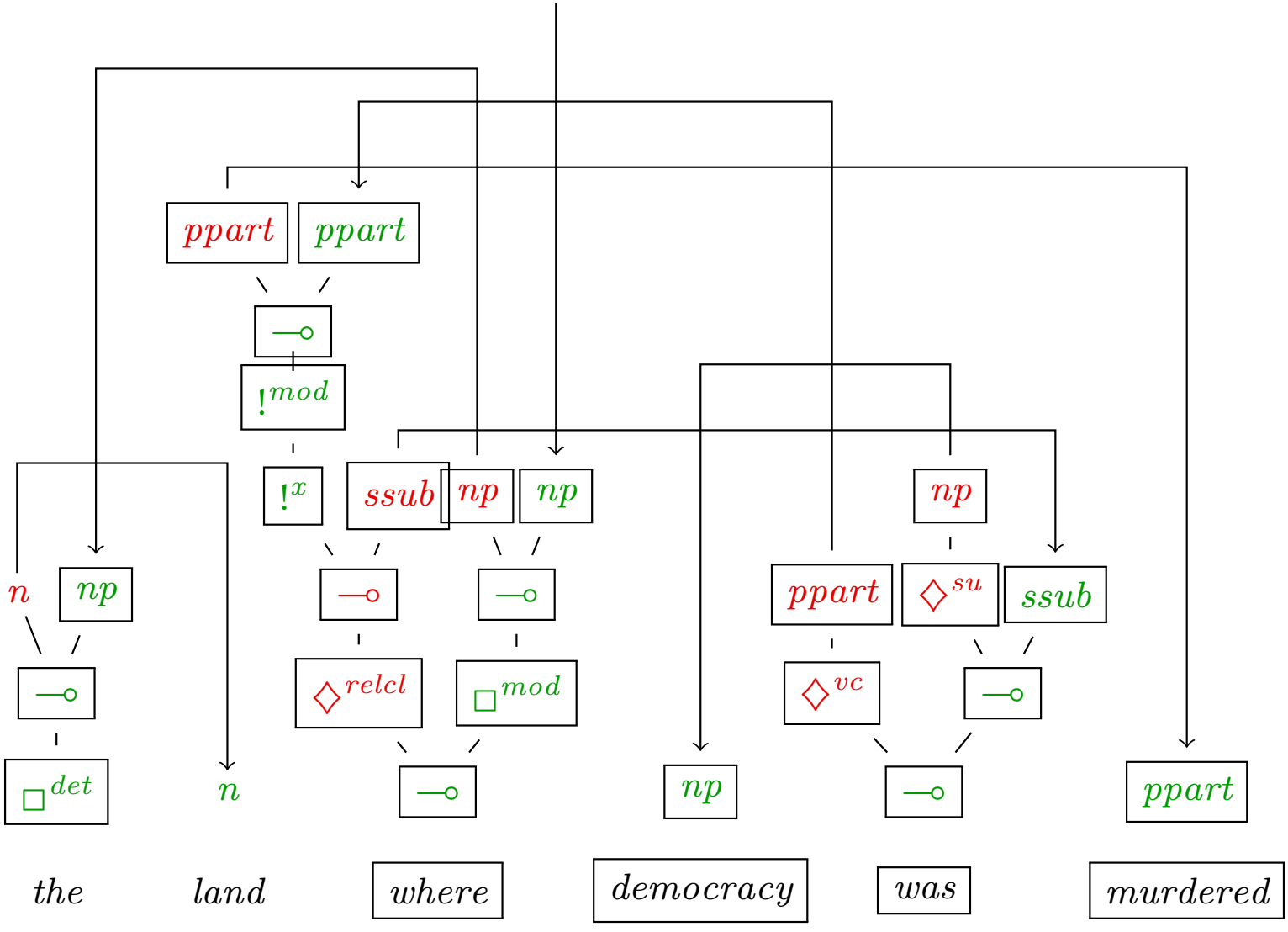
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ???

Proof ~~Frames~~ Structures



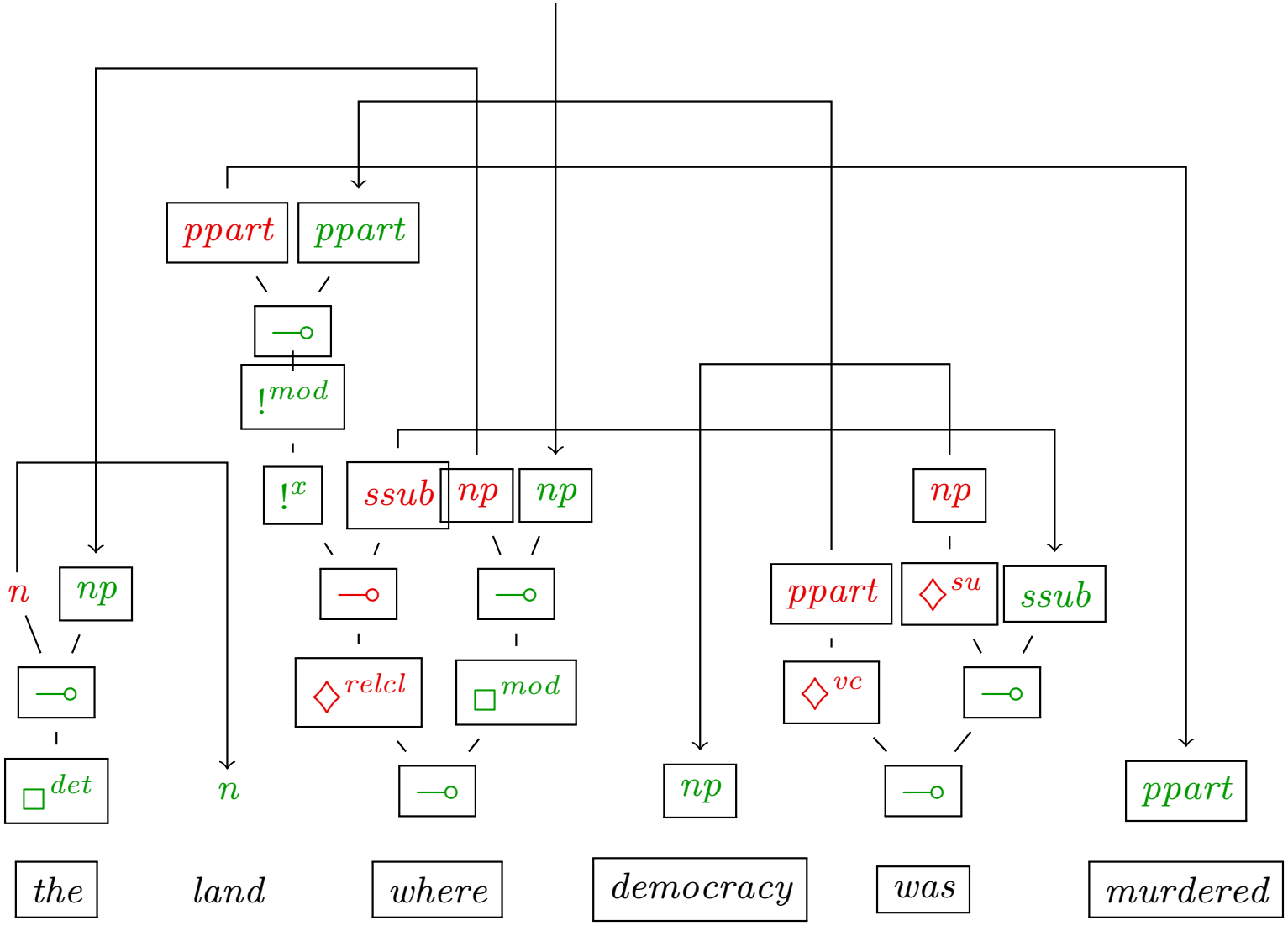
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) (??? ???)

Proof ~~Frames~~ Structures



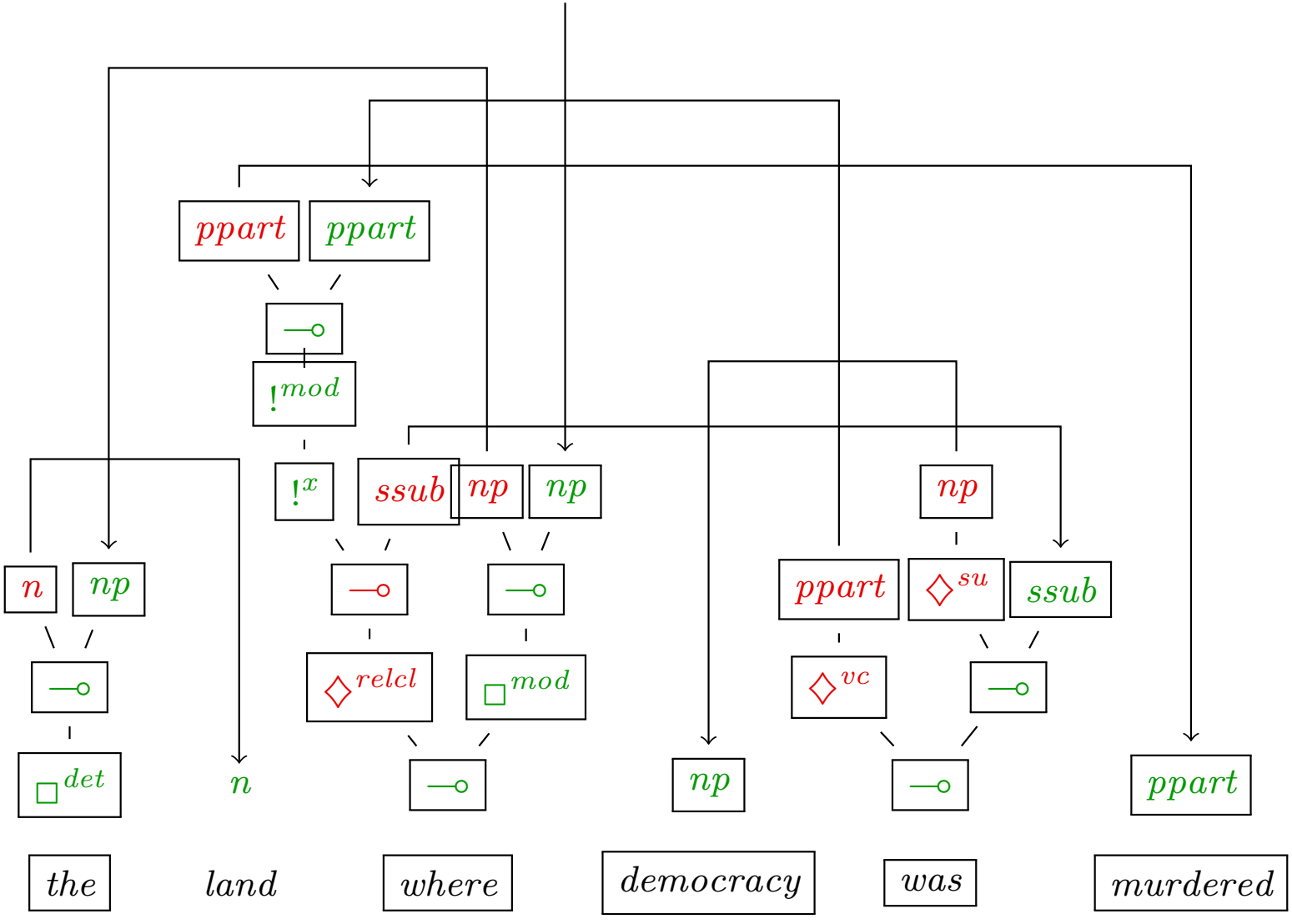
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ($\blacktriangledown^{det}$??? ???)

Proof ~~Frames~~ Structures



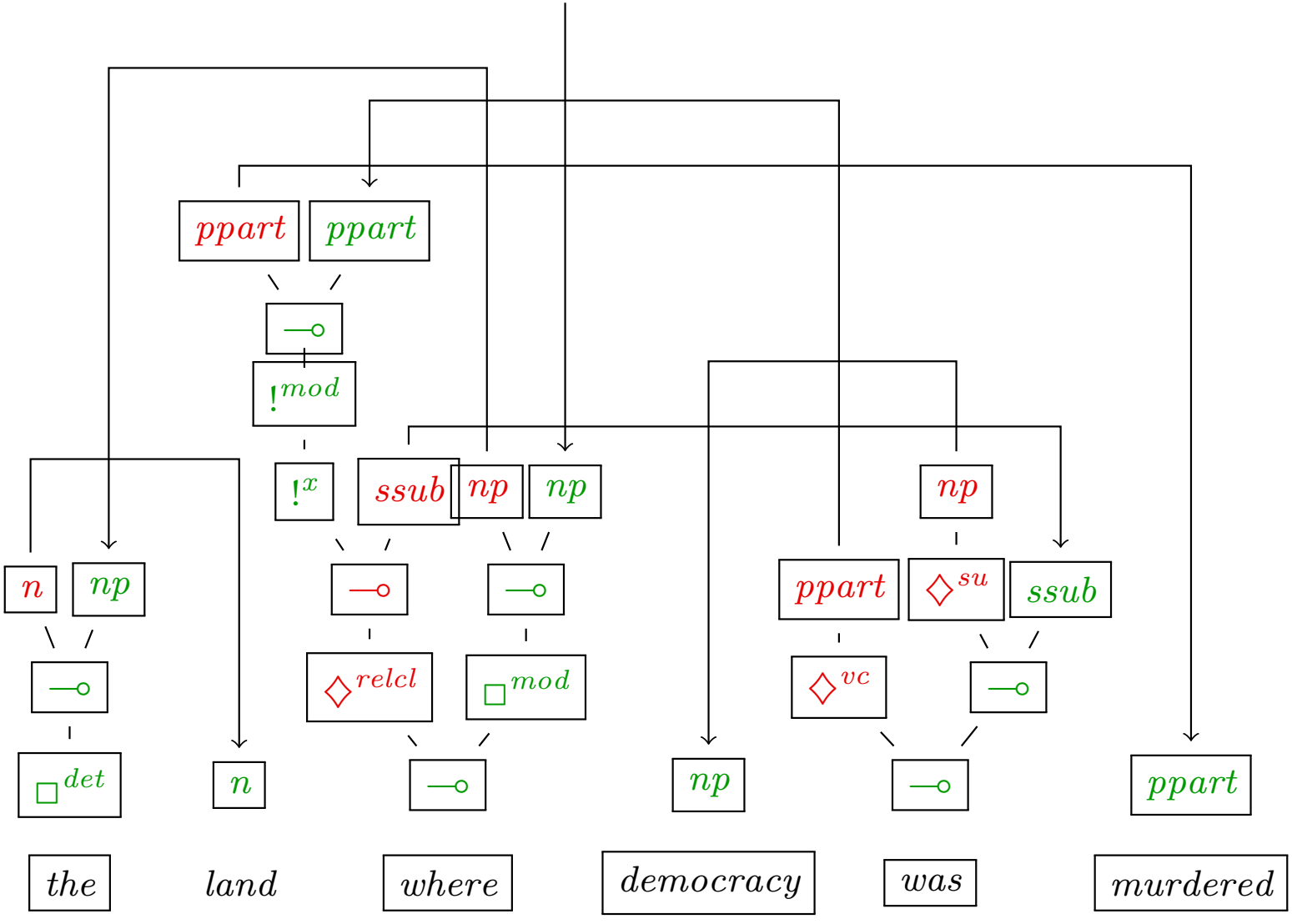
▼^{mod}(where △^{relcl} (λx$_0$.(was △^{vc} (▼^{mod}▽^{mod}▼^{x}▽^{x}x$_0$ murdered) △^{su} democracy))) (▼^{det}the ???)

Proof ~~Frames~~ Structures



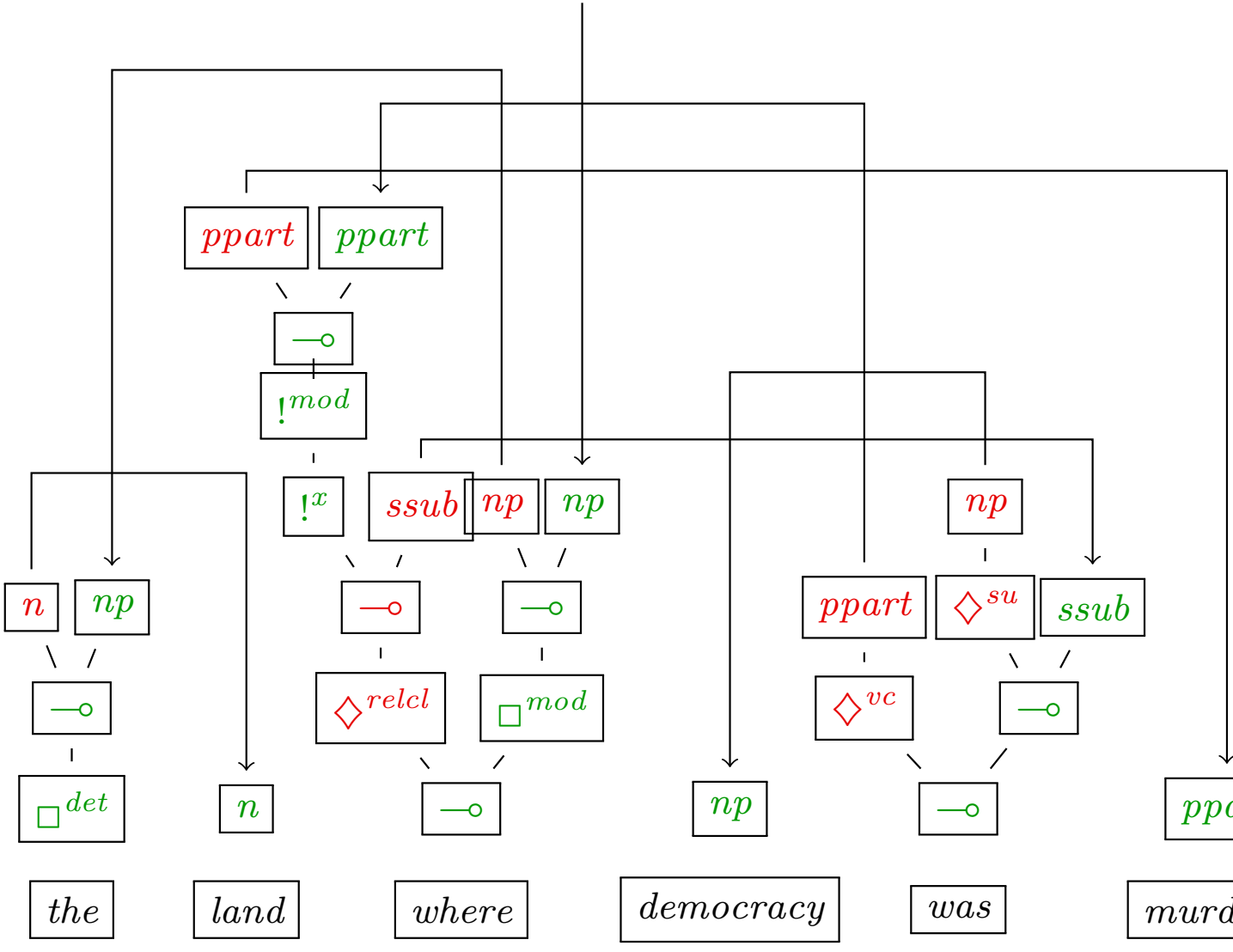
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ($\blacktriangledown^{det}$the ???)

Proof ~~Frames~~ Structures



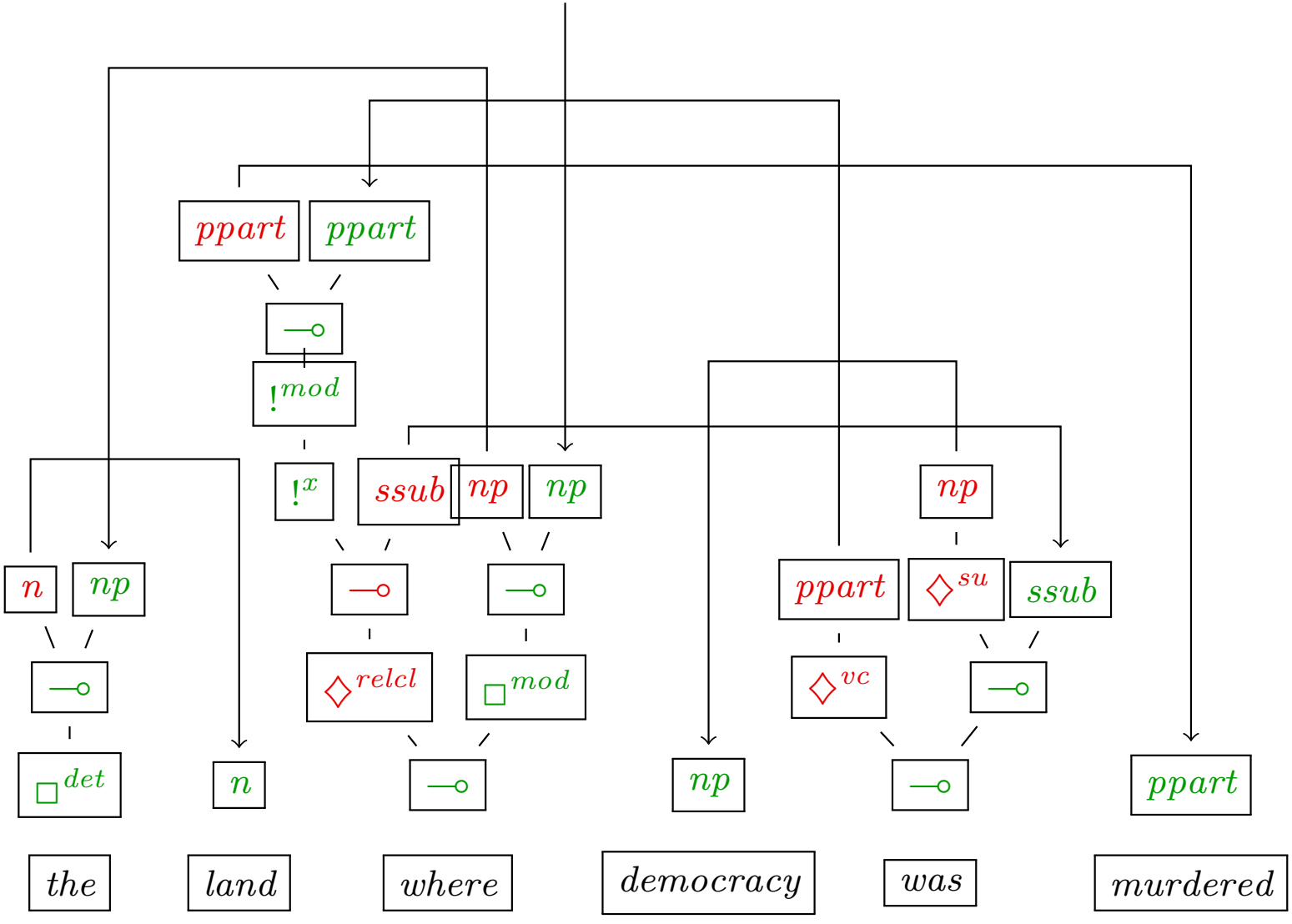
$\blacktriangledown^{mod}$(where $\triangle^{relcl}$ (λx_0.(was $\triangle^{vc}$ ($\blacktriangledown^{mod}\triangledown^{mod}\blacktriangledown^{x}\triangledown^{x}x_0$ murdered) $\triangle^{su}$ democracy))) ($\blacktriangledown^{det}$the ???)

Proof ~~Frames~~ Structures



$\blacktriangledown^{mod}$(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 land)

Proof ~~Frames Structures~~ Nets



$\blacktriangledown^{mod}$(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 land)

written and directed by

KOKOS

"Lets neural shit up"
– Alonzo Church, 1973

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 $(w_1, \ldots w_n)$

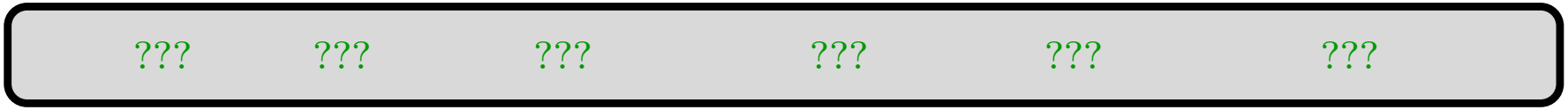
the catch

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

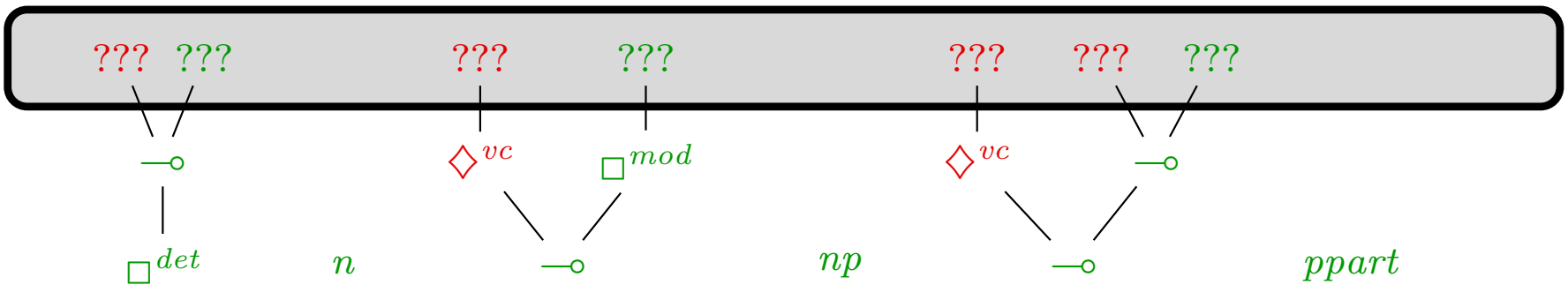


the land where democracy was murdered

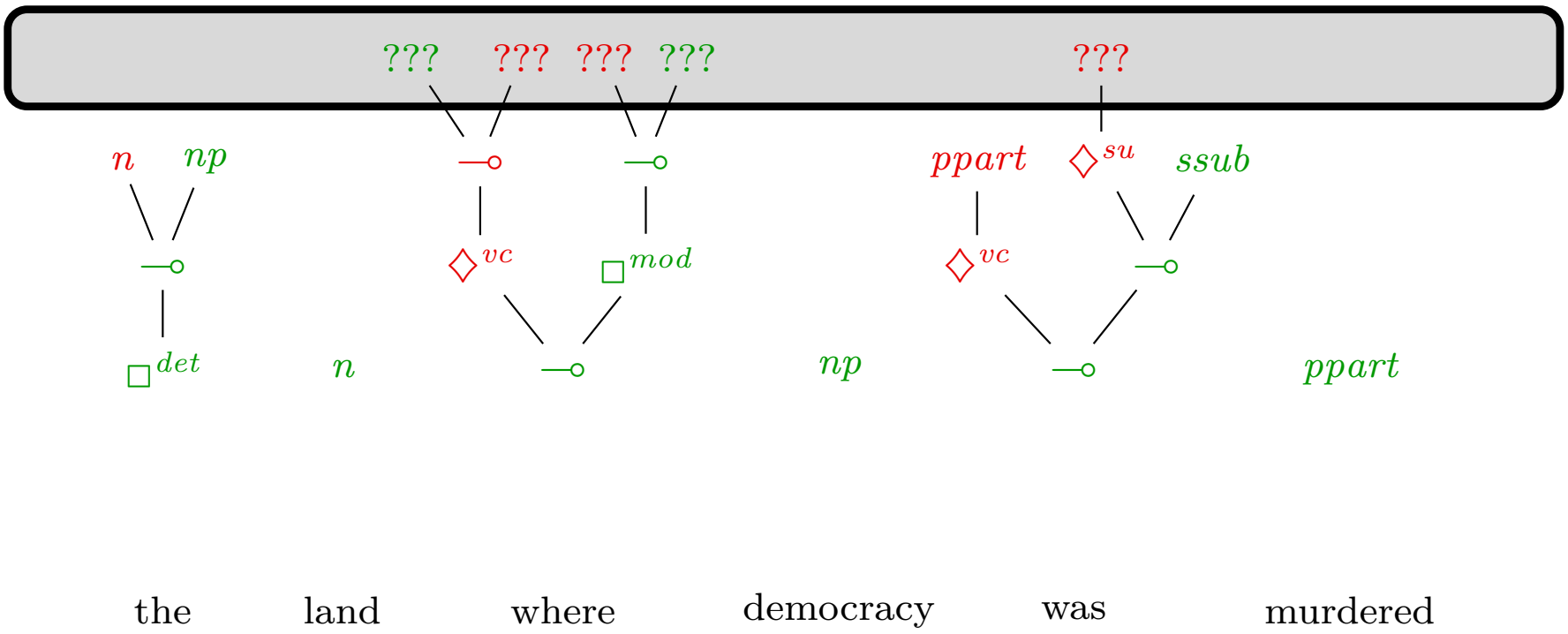
Supertagging as parallel graph completion

the land where democracy was murdered

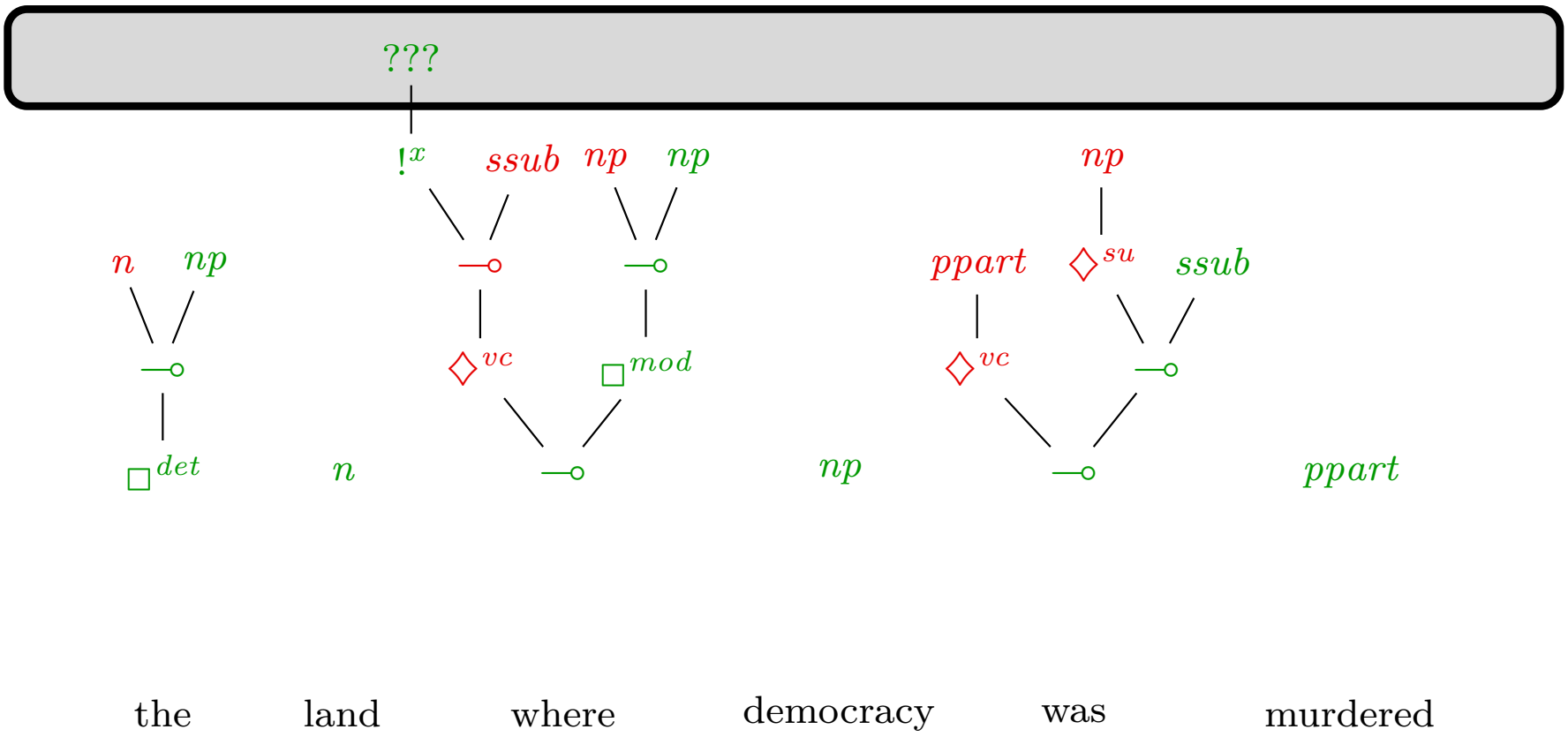
Supertagging as parallel graph completion



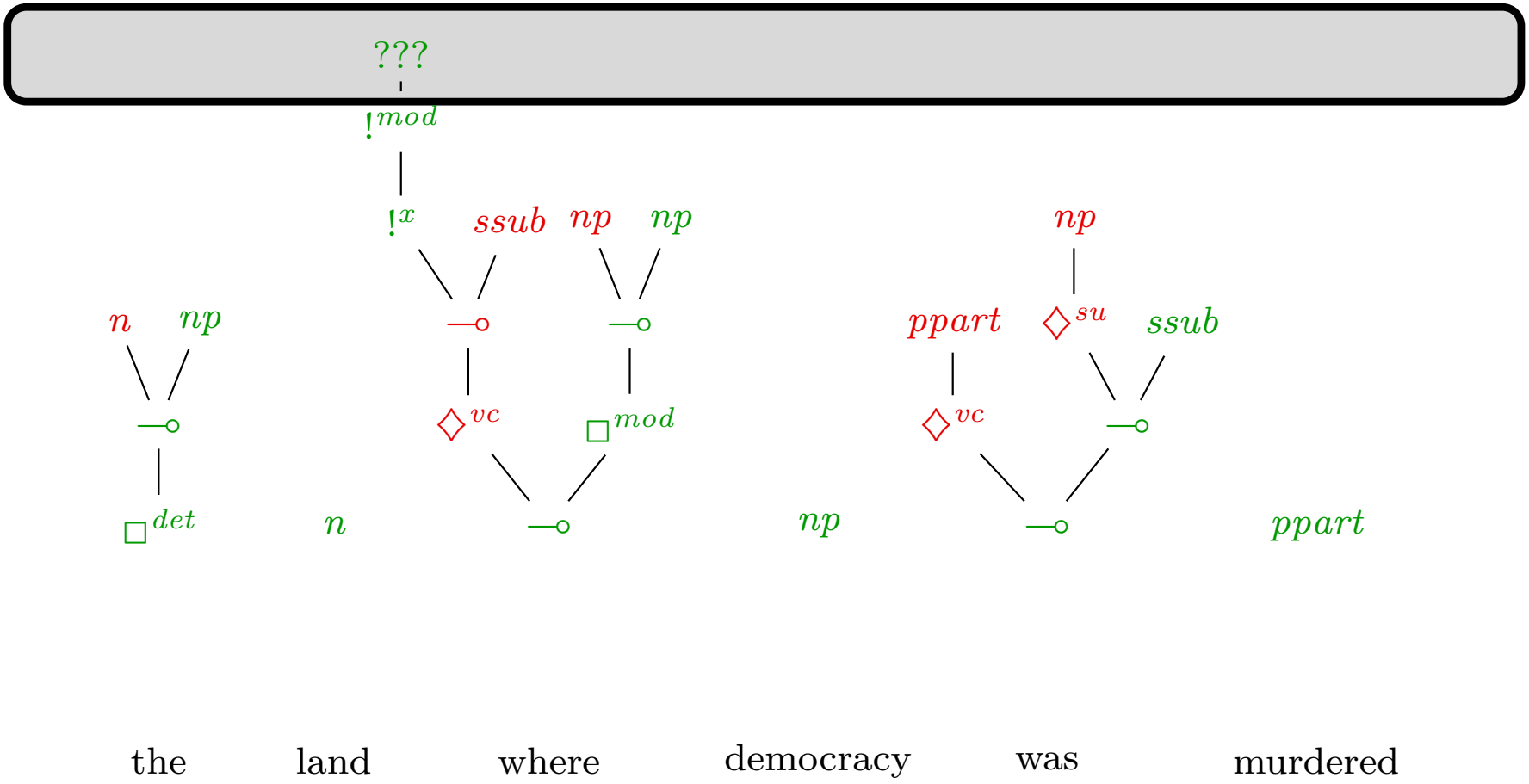
Supertagging as parallel graph completion



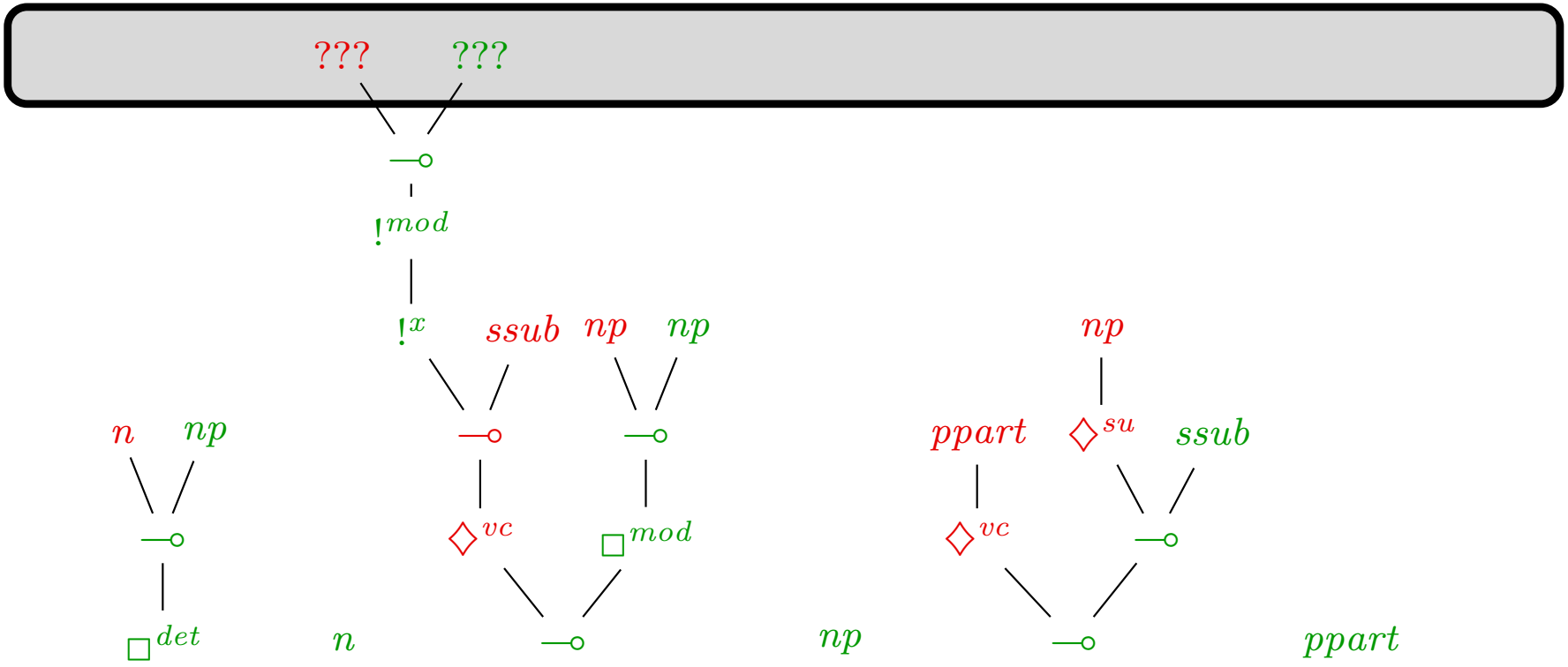
Supertagging as parallel graph completion



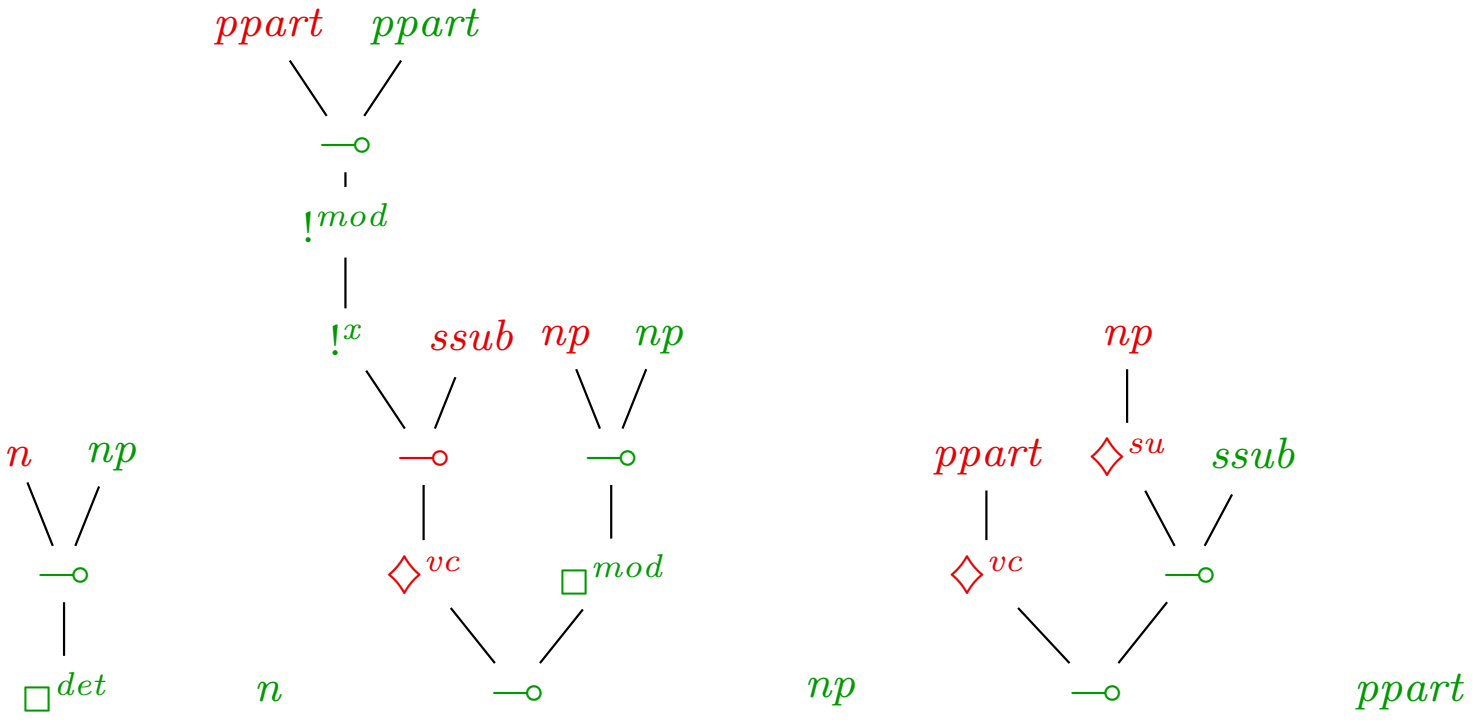
Supertagging as parallel graph completion



Supertagging as parallel graph completion

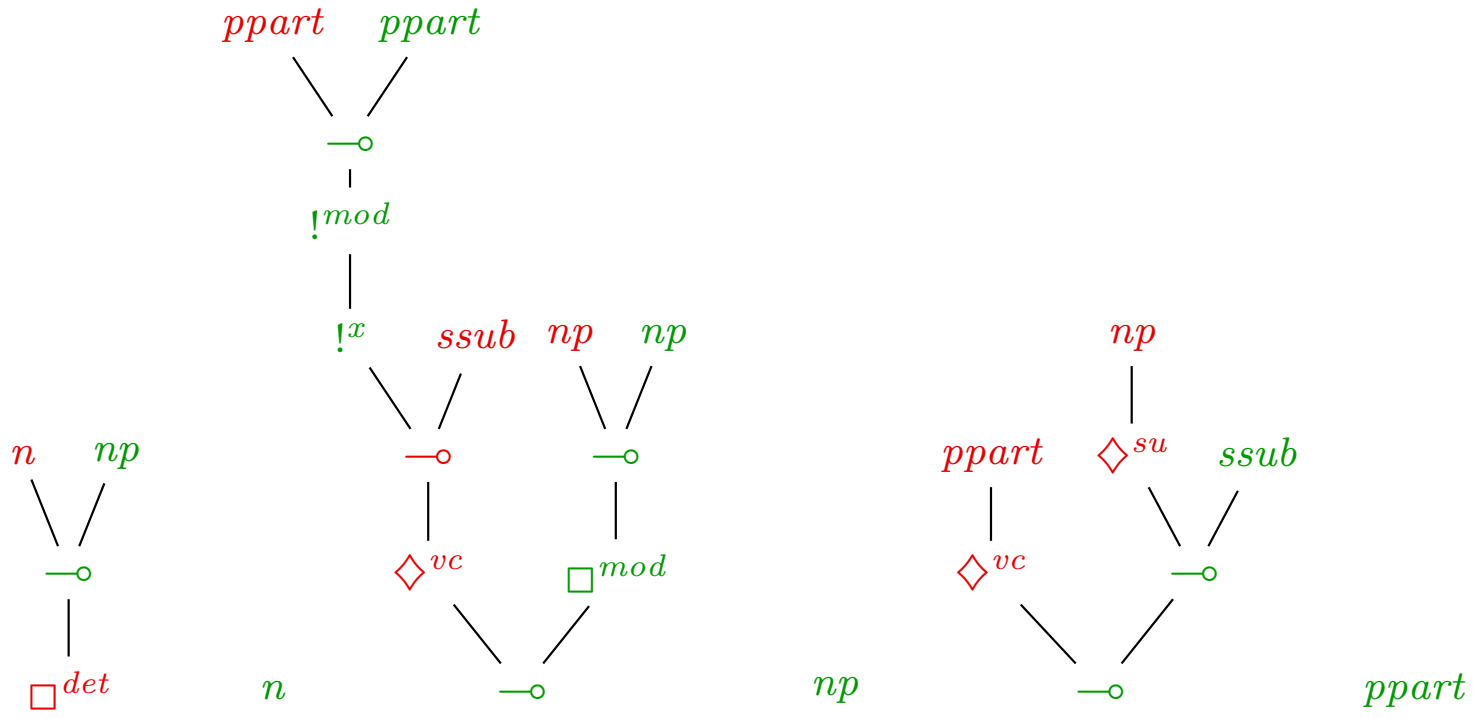


Supertagging as parallel graph completion



Axiom Linking

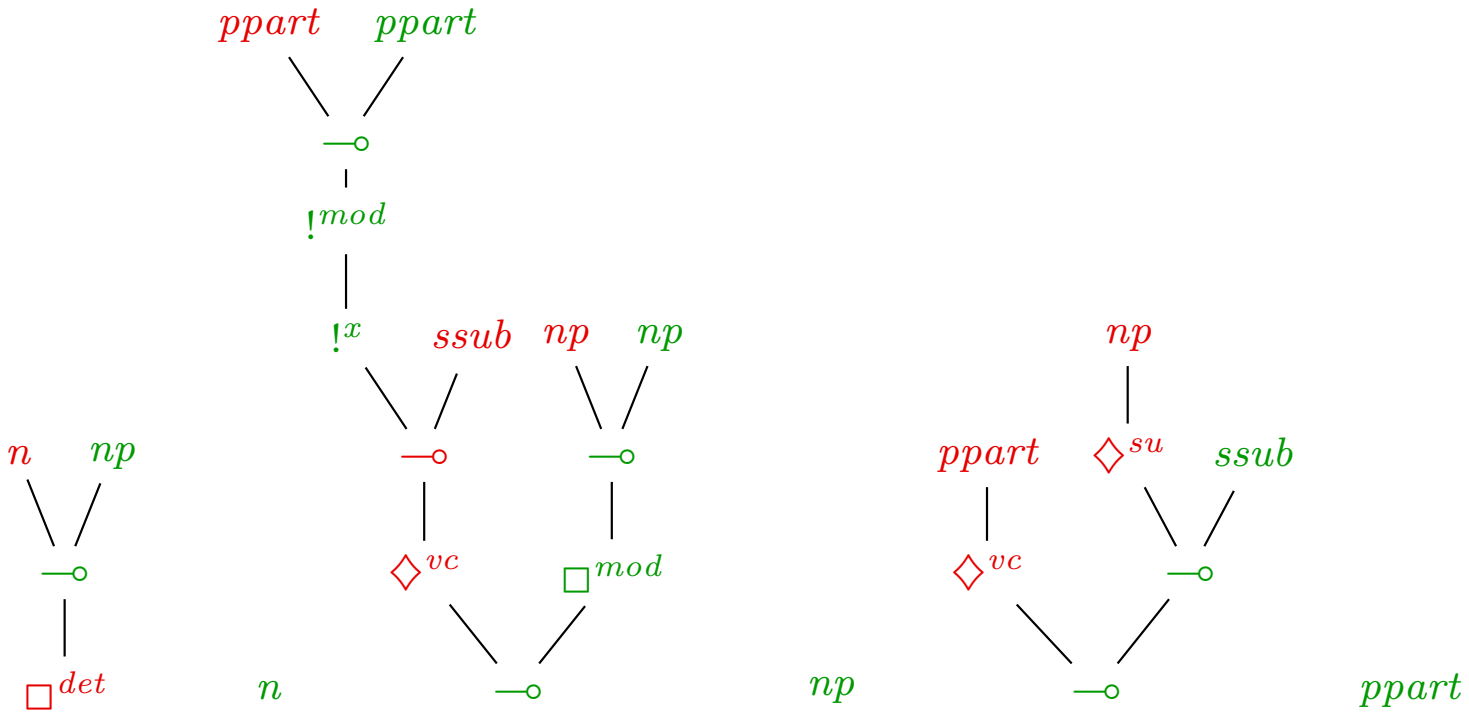
missing edges ☹



Axiom Linking

missing edges ☹

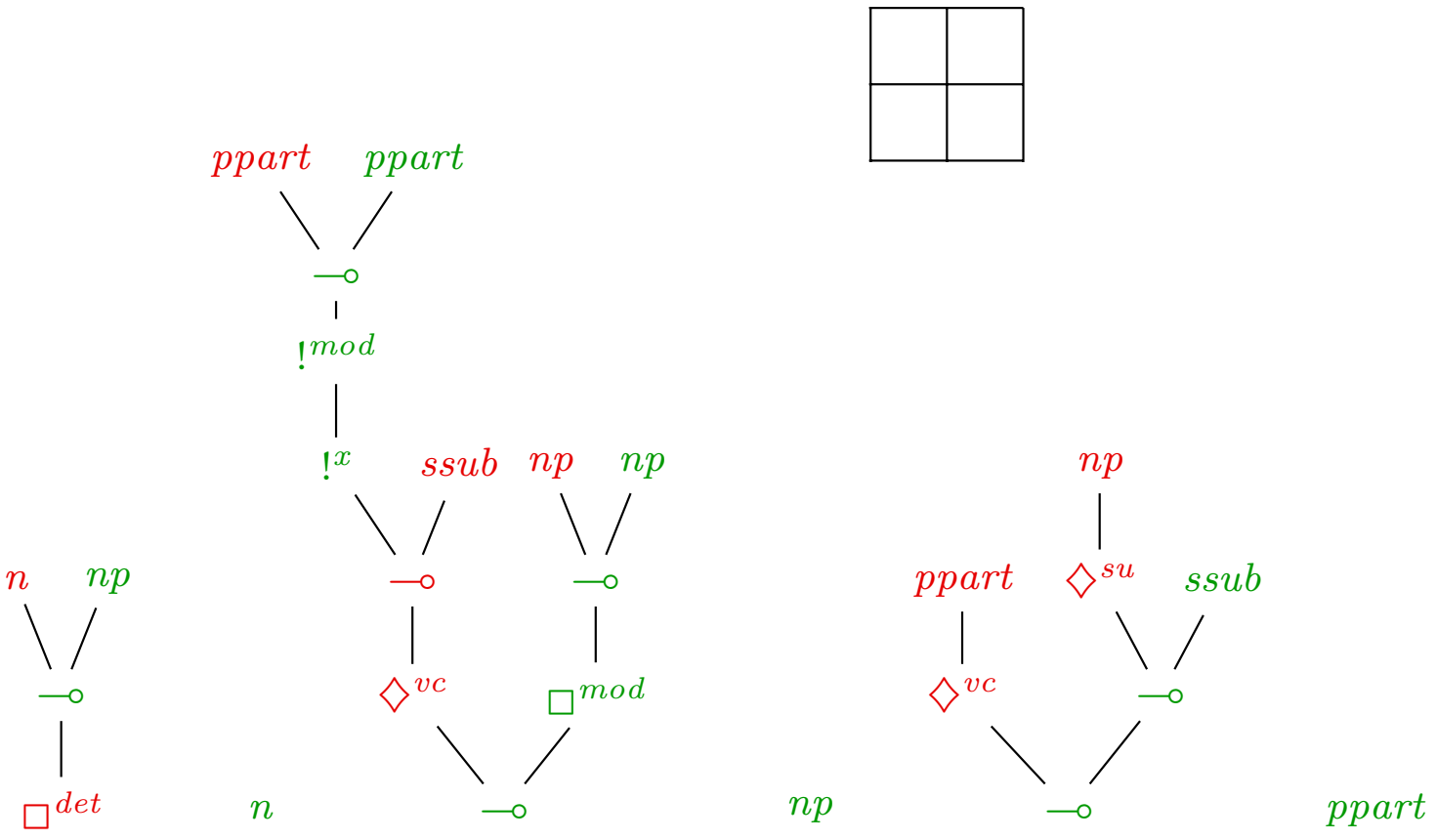
- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*



Axiom Linking

missing edges ☹

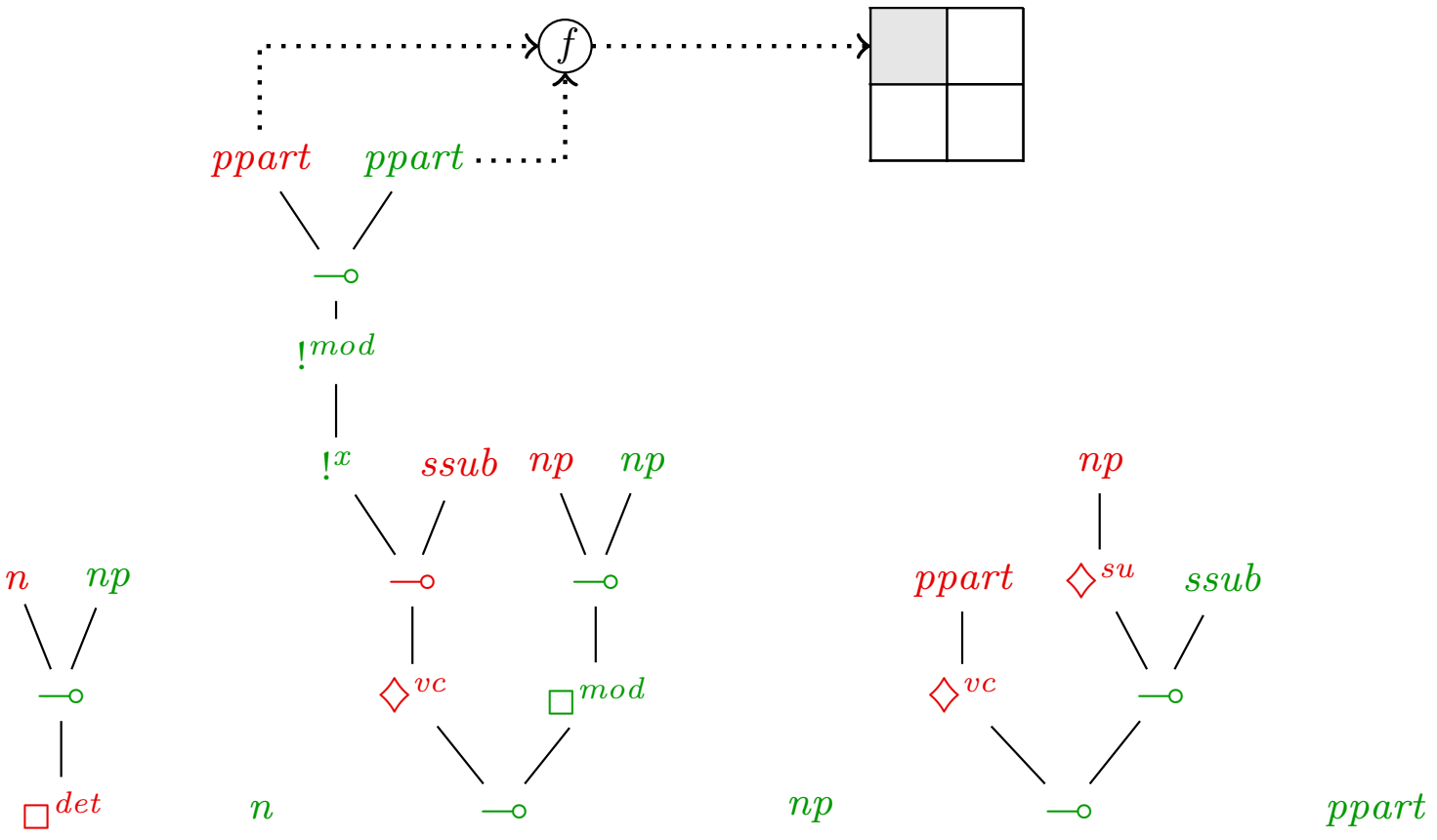
- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*



Axiom Linking

missing edges ☹

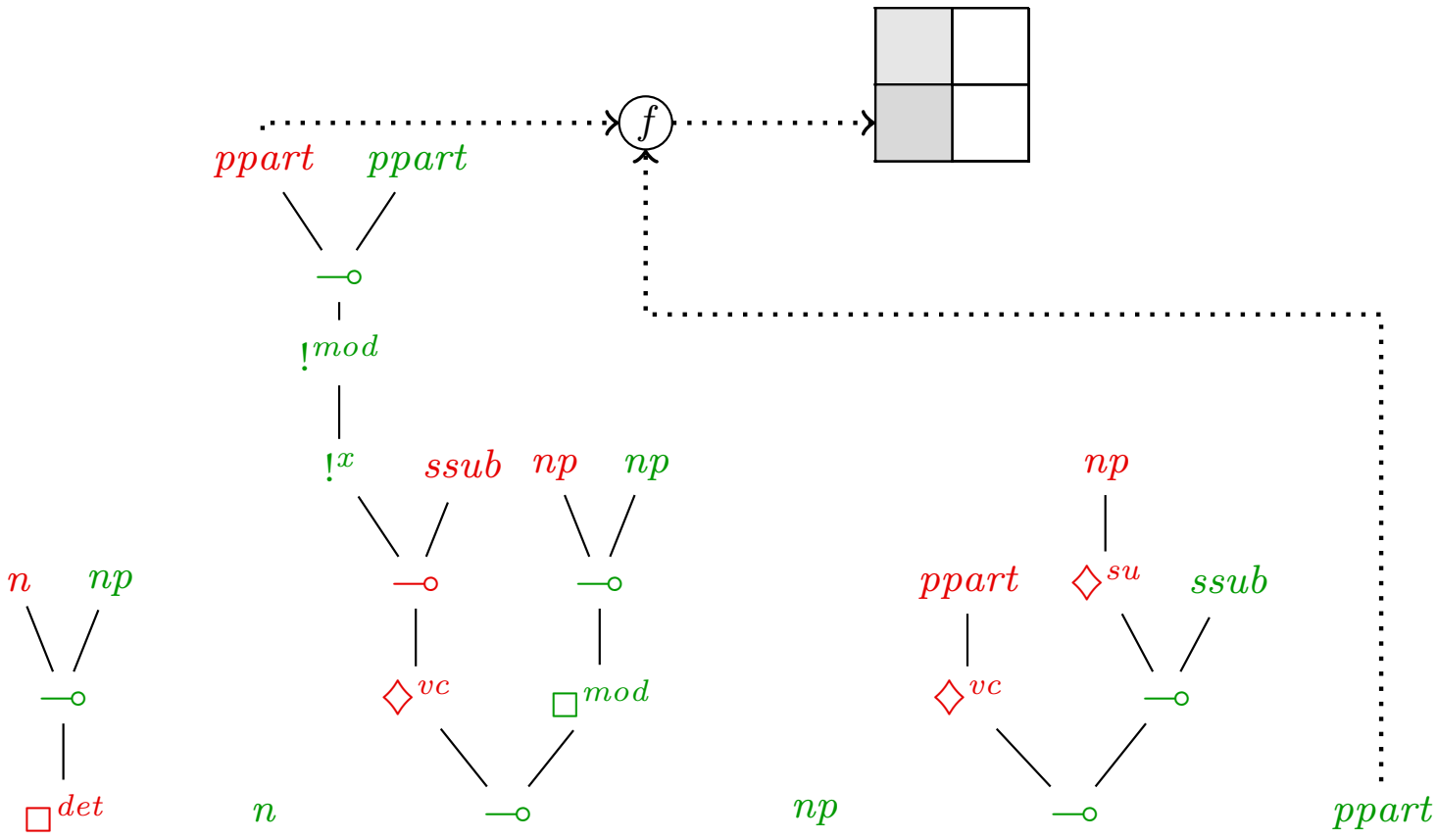
- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*



Axiom Linking

missing edges ☹

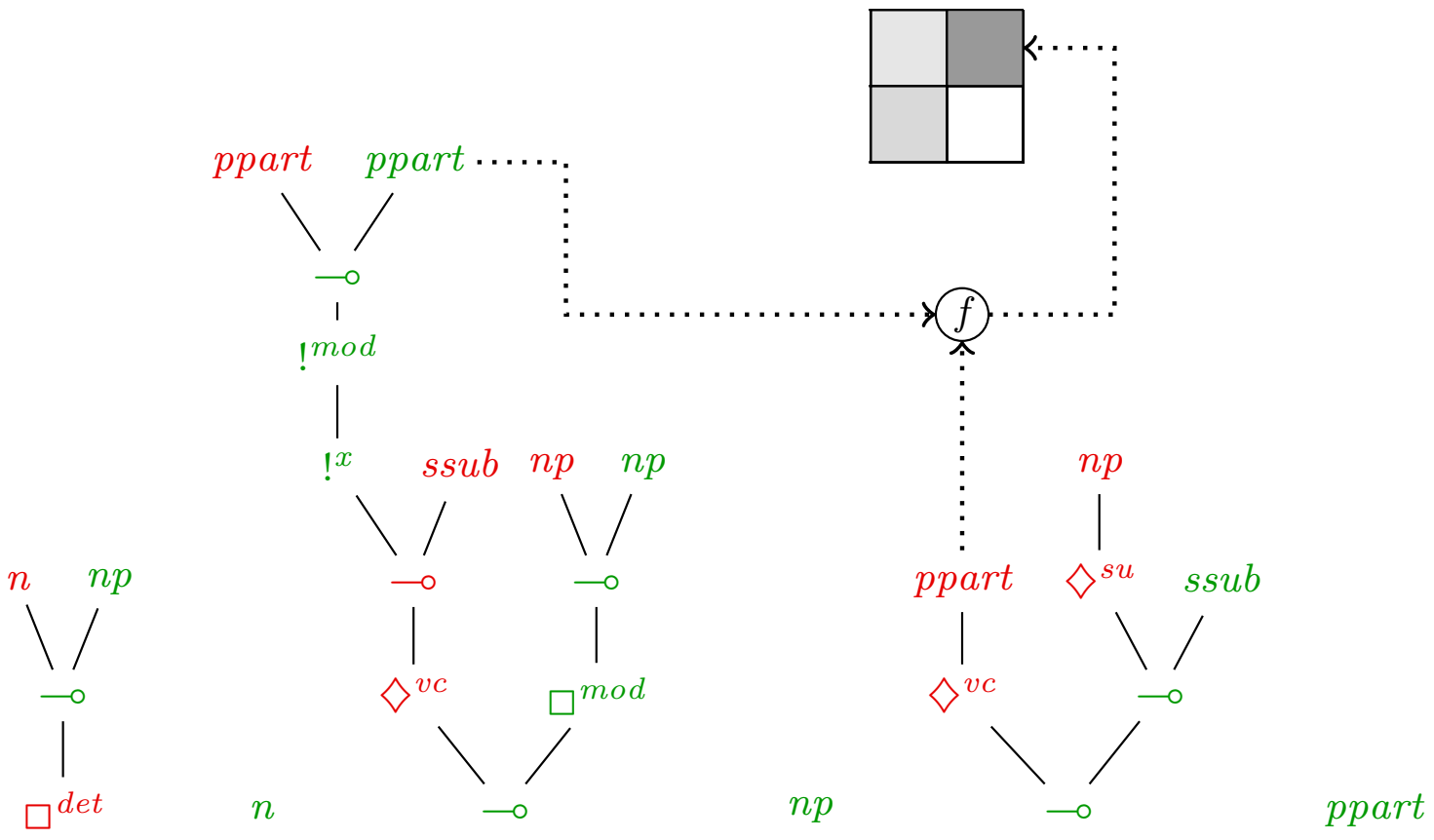
- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*



Axiom Linking

missing edges ☹

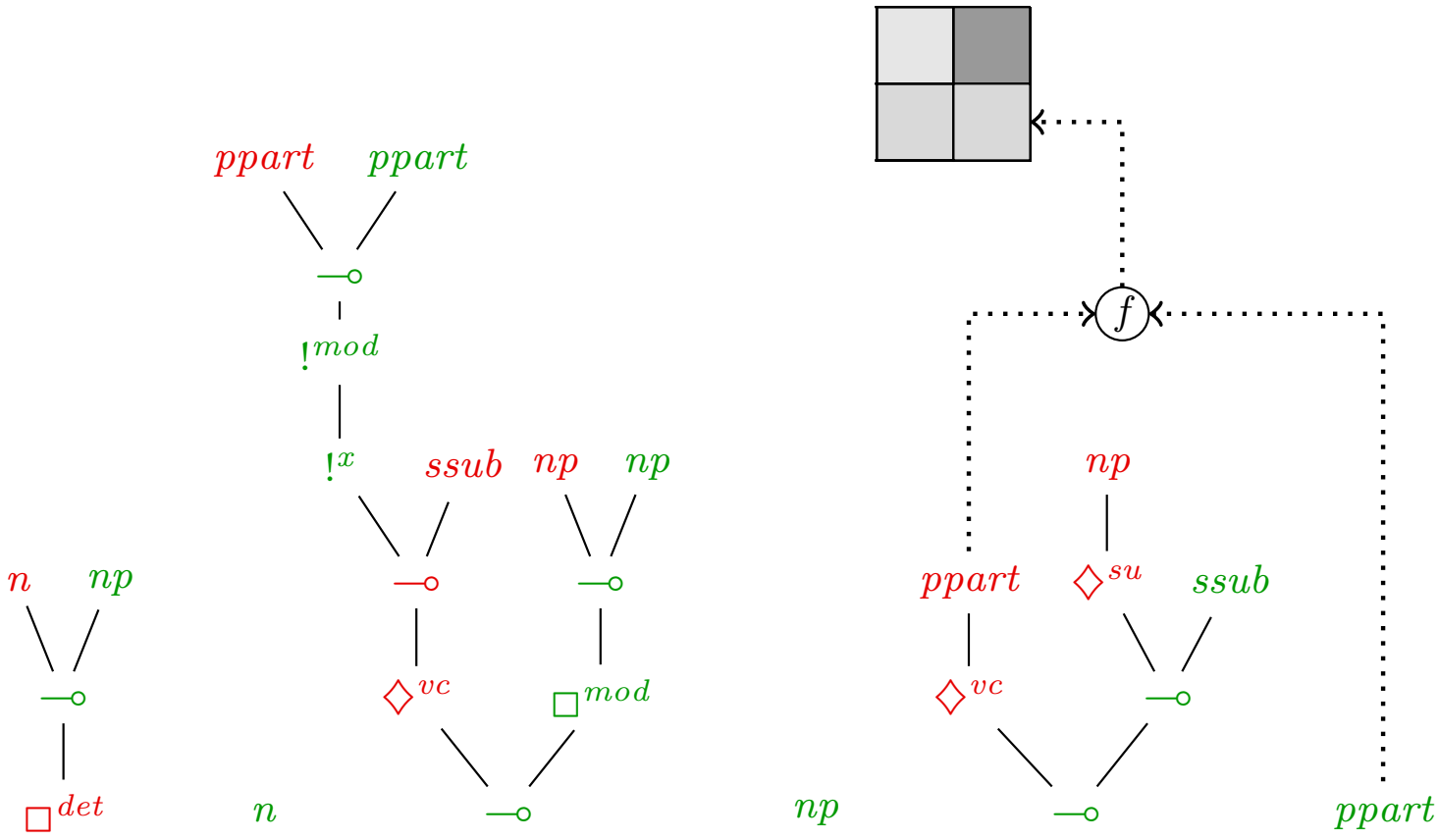
- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*



Axiom Linking

missing edges ☹

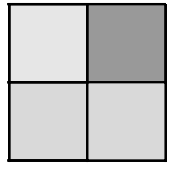
- ! only consider edges between atoms of the *same* sign and *different* polarity
- ! each atom can only be used *once*

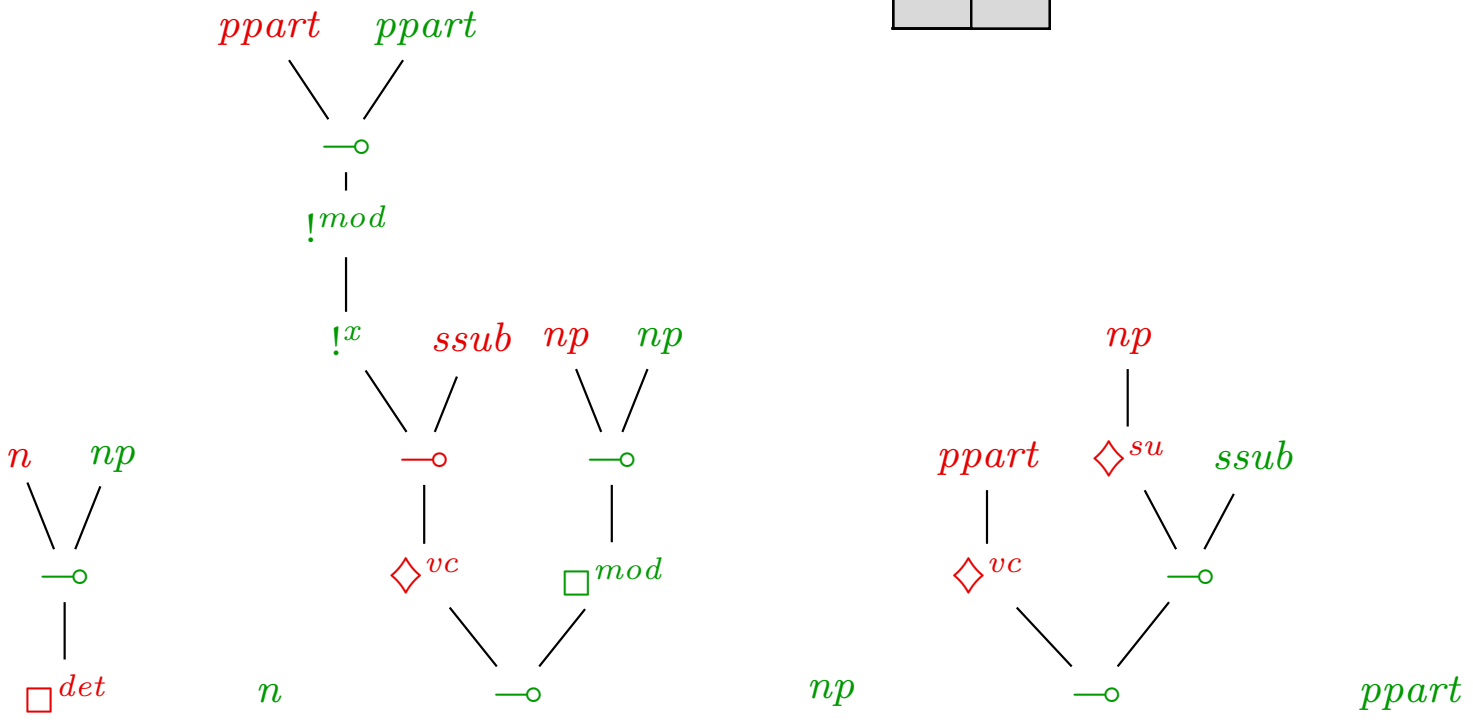


Axiom Linking

missing edges ☹

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

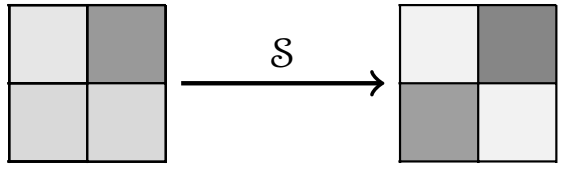


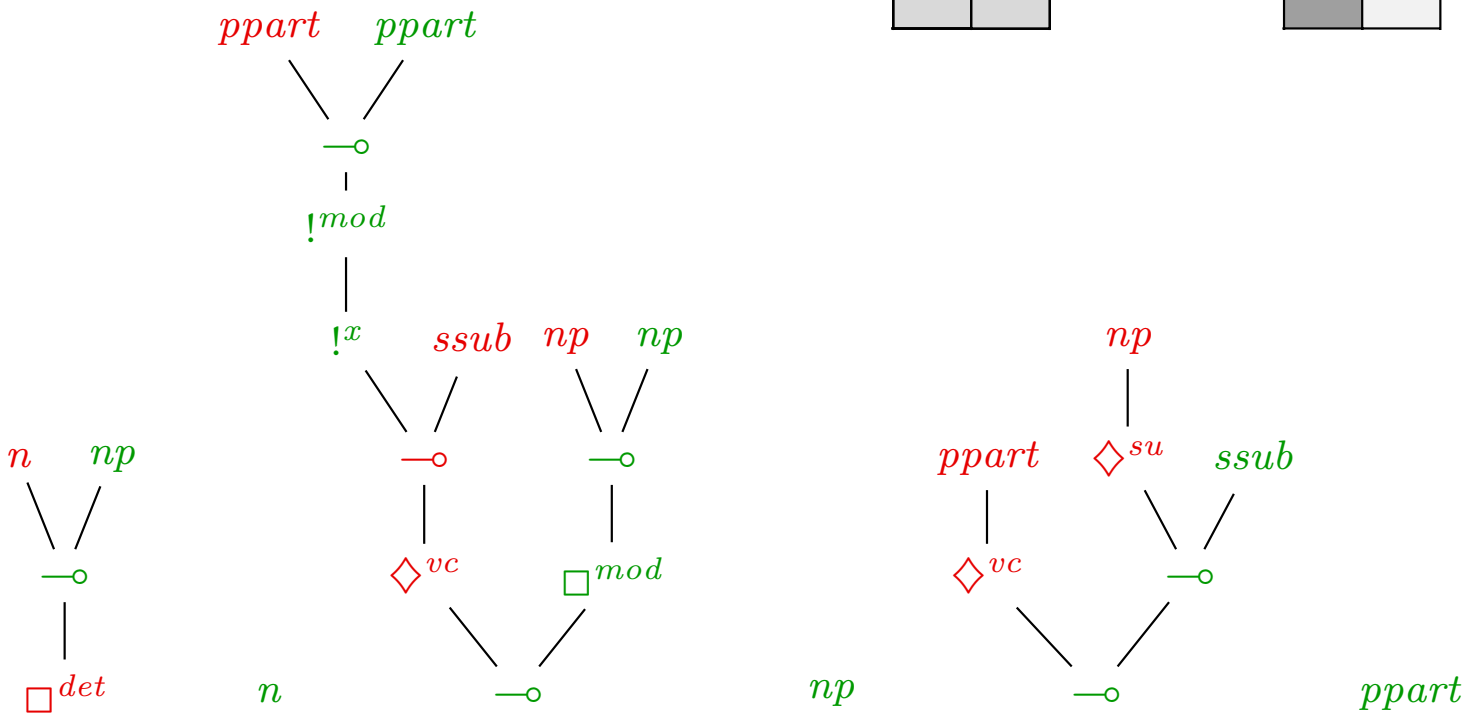


Axiom Linking

missing edges ☹

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





Axiom Linking

no missing edges! ☺