Introduction to Formal Semantics
Lecture 5: Function Application

Volha Petukhova & Nicolaie Dominik Dascalu

Spoken Language Systems Group
Saarland Univeristy

23.05.2022

UNIVERSITÄT
DES
SAARLANDES

Overview for today

- Recap: Typed lambda calculus
- Montaque Grammar
- Functions Applications
- Generalized Quantifies

Reading:

- Coppock, E., and Champollion, L. (2021). Invitation to formal semantics. Manuscript, Boston University and New York University (Ch.6)

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b)

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$ $< e, e >$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$ $< e, e >$
6. $\lambda f.f$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$ $< e, e >$
6. $\lambda f.f$ $<< e, t >, < e, t >>$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$ $< e, e >$
6. $\lambda f.f$ $<< e, t >, < e, t >>$
7. $\lambda x \lambda y.R(x, y)$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$ $< e, e >$
6. $\lambda f.f$ $<< e, t >, < e, t >>$
7. $\lambda x \lambda y.R(x, y)$ $<< e, < e, t >>$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$ $< e, e >$
6. $\lambda f.f$ $<< e, t >, < e, t >>$
7. $\lambda x \lambda y.R(x, y)$ $<< e, < e, t >>$
8. $\lambda x \lambda y.in(x, y)$

Quizz (last week)

Task 1: identify the type of each of the following:

1. Maria e
2. x $< e, e >$
3. P $< e, t >$
4. P(a)(b) $<< e, t >, < e, t >>$
5. $\lambda x.x$ $< e, e >$
6. $\lambda f.f$ $<< e, t >, < e, t >>$
7. $\lambda x \lambda y.R(x, y)$ $<< e, < e, t >>$
8. $\lambda x \lambda y.in(x, y)$ $<< e, < e, t >>$

Types

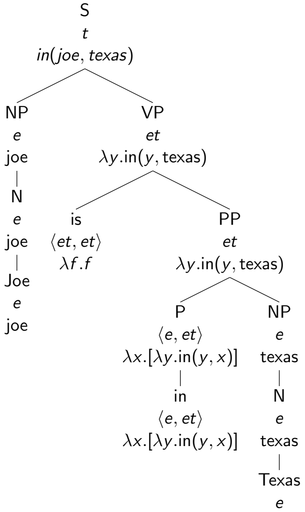
Table 3.2: Lexical denotations and their restrictions.

Denotation	Type	Restrictions	Category
tina	e	-	proper name
smile	et	-	intransitive verb
praise	$e(et)$	-	transitive verb
pianist	et	-	common noun
chinese	et	-	predicative adjective
$\mathbf{chinese}^{\mathsf{mod}}$	$(et)(et)$	intersective: $\lambda f_{et}.\lambda x_e.$ $\mathbf{chinese}(x) \wedge f(x)$	modificational adjective
$\mathbf{skillful}^{\mathsf{mod}}$	$(et)(et)$	subsective: $\lambda f_{et}.\lambda x_e.$ $(\mathbf{skillful}^{\mathsf{arb}}(f))(x) \wedge f(x)$	modificational adjective
IS	$(et)(et)$	combinator: $\lambda g_{et}.g$	copula (auxiliary verb)
A	$(et)(et)$	combinator: $\lambda g_{et}.g$	indefinite article
HERSELF	$(e(et))(et)$	combinator: $\lambda R_{e(et)}.\lambda x_e.R(x)(x)$	reflexive pronoun
NOT	$(et)(et)$	logical: $\lambda g_{et}.\lambda x_e.{\sim}(g(x))$	predicate negation
AND^{t}	$t(tt)$	logical: $\lambda x_t.\lambda y_t.y \wedge x$	sentential conjunction
AND^{et}	$(et)((et)(et))$	logical: $\lambda f_{et}.\lambda g_{et}.\lambda x_e.g(x) \wedge f(x)$	predicate conjunction

Lambda Calculator: Scratch Tool & exercise hk-chapter6

Quizz (last week)

Task 2: Replace the question mark ‘?’



Lambda Conversion

$[\lambda X.\exists x.[P(x) \wedge X(x)]](\lambda x.Man(x))$

$[\lambda X.\exists x.[P(x) \wedge X(x)]](\lambda y.R(a, y))$

$[\lambda X.\exists x.[P(x) \wedge X(x)]](\lambda x.R(a, x))$

$[\lambda X.\exists x.[P(x) \wedge X(x)]](\lambda y.R(y, x))$

$[\lambda X.\forall x[man(x) \rightarrow X(x)]](\lambda x[mortal(x)])$

$[\lambda X \lambda x.\neg X(x)](\lambda x.mortal(x))$

$[\lambda X \lambda x.X(x)](\lambda x.[\neg mortal(x)])$

Lambda Calculator: Scratch Tool & exercise hk-chapter3

Montaque Grammar

Montaque (1930 -1970) semantics or grammar

- English as a formal language
- Universal grammar
- The proper treatment of quantification in ordinary English

Montaque inspiring idea: "I reject the contention that an important theoretical difference exists between formal and natural languages. " (Montague, 1970)
"Montague grammar is a very elegant and a very simple theory of natural language semantics. Unfortunately its elegance and simplicity are obscured by a needlessly baroque formalization" (Muskens, 1995)

Translation: English Fragments to Formal Representations

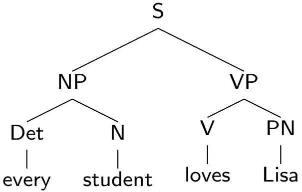
What we need:

- a specification of our formal representation language, with syntactic and semantic rules;
- a specification of the syntax of the English expressions we cover;
- a list of lexical entries;
- a list of composition rules.

Translation: English Fragments to Formal Representations (cont.)

We have:

- Typed Lambda Calculus (L_λ): the tools semanticists reach for when they want to explore language, state hypotheses, and discuss their ideas.
- Compositionality: the meaning of a phrase is a function of the meanings of its immediate syntactic constituents and the way they are combined.



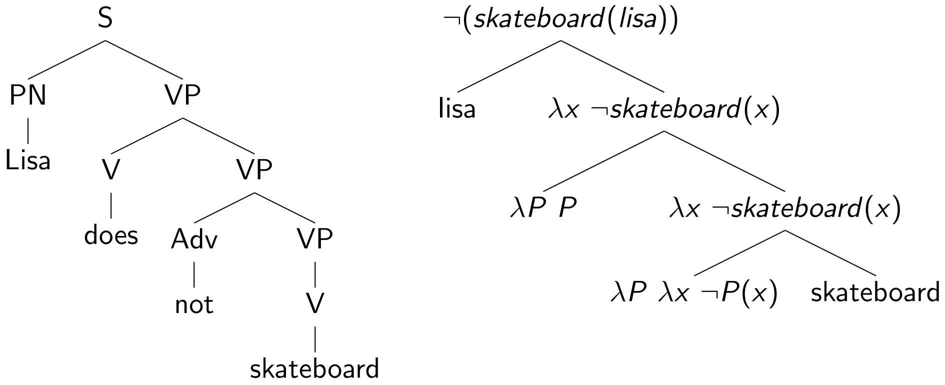
- Lexicon:

  PN: Lisa

  Neg: not

  V: skateboard
- List of composition rules: S $\rightarrow$ NP VP NP $\rightarrow$ Det N

Translation: natural language into lambda terms



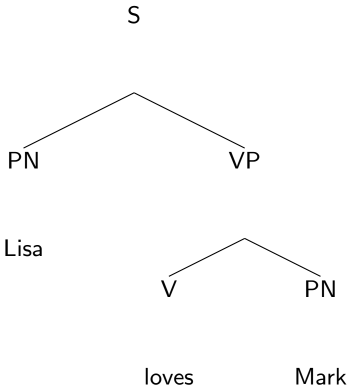
Compositionality

Composition Rule 1: Function Application

Let γ be a syntax tree whose sub-trees are α and β where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type $\langle \sigma \rangle$

then $$\gamma \rightsquigarrow \alpha'(\beta')$$



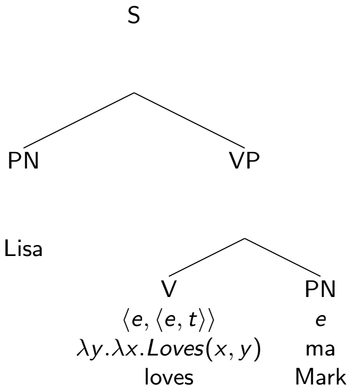
Compositionality

Composition Rule 1: Function Application

Let γ be a syntax tree whose sub-trees are α and β where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type $\langle \sigma \rangle$

then
$$\gamma \rightsquigarrow \alpha'(\beta')$$



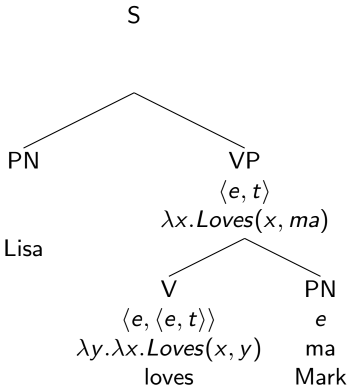
Compositionality

Composition Rule 1: Function Application

Let γ be a syntax tree whose sub-trees are α and β where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type $\langle \sigma \rangle$

then $$\gamma \rightsquigarrow \alpha'(\beta')$$



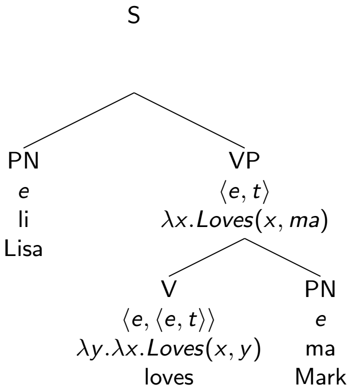
Compositionality

Composition Rule 1: Function Application

Let γ be a syntax tree whose sub-trees are α and β where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type $\langle \sigma \rangle$

then $$\gamma \rightsquigarrow \alpha'(\beta')$$



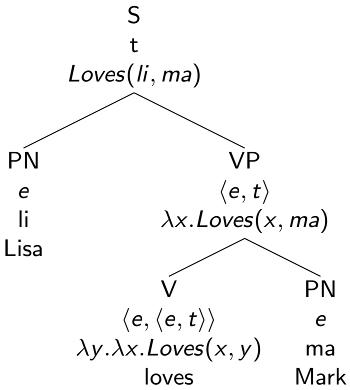
Compositionality

Composition Rule 1: Function Application

Let γ be a syntax tree whose sub-trees are α and β where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type $\langle \sigma \rangle$

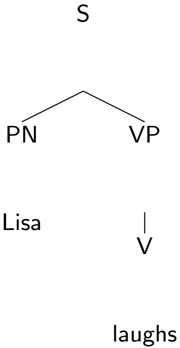
then
$$\gamma \rightsquigarrow \alpha'(\beta')$$



Compositionality (cont.)

Composition Rule 2: Non-branching Nodes

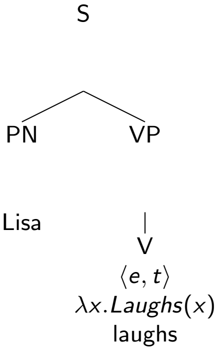
If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$ then $\beta \rightsquigarrow \alpha'$



Compositionality (cont.)

Composition Rule 2: Non-branching Nodes

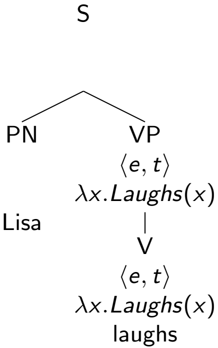
If β is a tree whose only daughter is α, where $\alpha \leadsto \alpha'$ then $\beta \leadsto \alpha'$



Compositionality (cont.)

Composition Rule 2: Non-branching Nodes

If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$ then $\beta \rightsquigarrow \alpha'$



Compositionality (cont.)

Composition Rule 2: Non-branching Nodes

If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$ then $\beta \rightsquigarrow \alpha'$

S

PN
e
li
Lisa

VP
$\langle e, t\rangle$
$\lambda x.Laughs(x)$

V
$\langle e, t\rangle$
$\lambda x.Laughs(x)$
laughs

Compositionality (cont.)

Composition Rule 2: Non-branching Nodes

If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$ then $\beta \rightsquigarrow \alpha'$

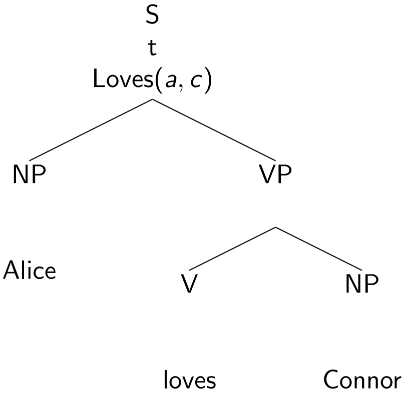
S
t
Laughs(*li*)

PN
e
li
Lisa

VP
$\langle e, t\rangle$
$\lambda x.Laughs(x)$

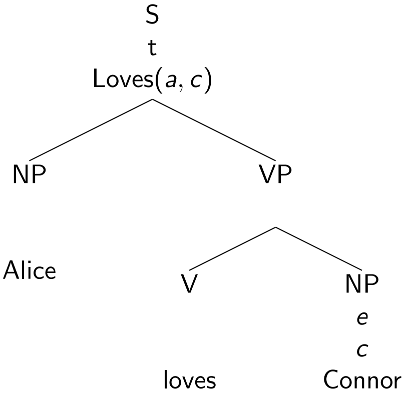
V
$\langle e, t\rangle$
$\lambda x.Laughs(x)$
laughs

Transitive verbs:
$\langle e, \langle e, t \rangle \rangle$
$\lambda y.\lambda x.P(x, y)$
Alice loves Connor



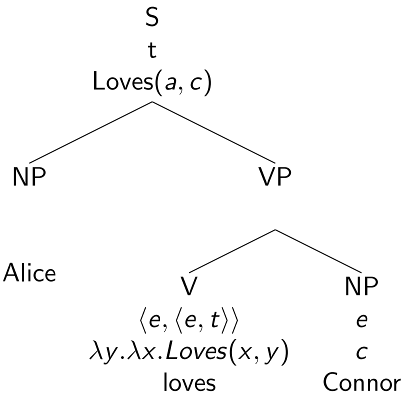
Transitive verbs

Transitive verbs:
$\langle e, \langle e, t \rangle \rangle$
$\lambda y. \lambda x. P(x, y)$
Alice loves Connor



Transitive verbs

Transitive verbs:
$\langle e, \langle e, t \rangle \rangle$
$\lambda y.\lambda x.P(x, y)$
Alice loves Connor

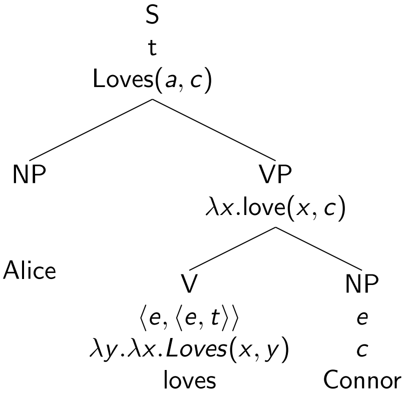


Transitive verbs:

$\langle e, \langle e, t \rangle \rangle$

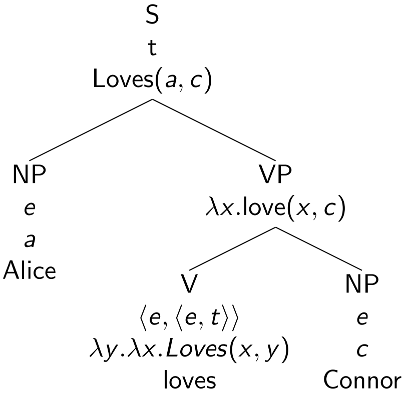
$\lambda y.\lambda x.P(x, y)$

Alice loves Connor



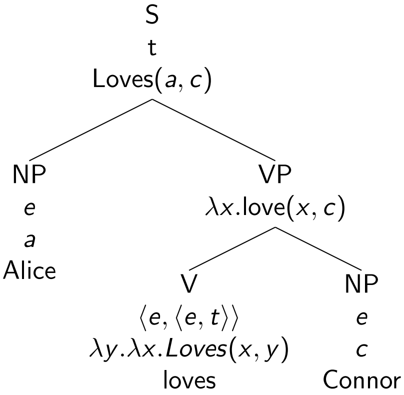
Transitive verbs

Transitive verbs:
$\langle e, \langle e, t \rangle\rangle$
$\lambda y.\lambda x.P(x, y)$
Alice loves Connor



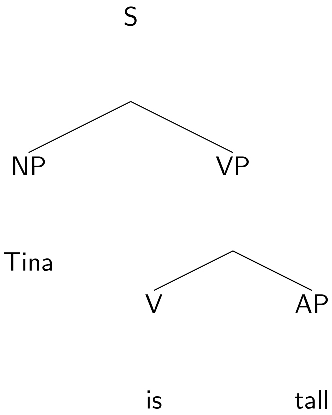
Transitive verbs

Transitive verbs:
$\langle e, \langle e, t \rangle \rangle$
$\lambda y.\lambda x.P(x, y)$
Alice loves Connor



Identity Function

IS:
$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
is $\rightsquigarrow \lambda P.P$
Tina is tall



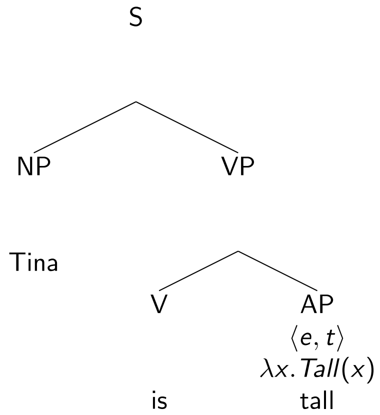
Identity Function

IS:

$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$

is $\rightsquigarrow \lambda P.P$

Tina is tall



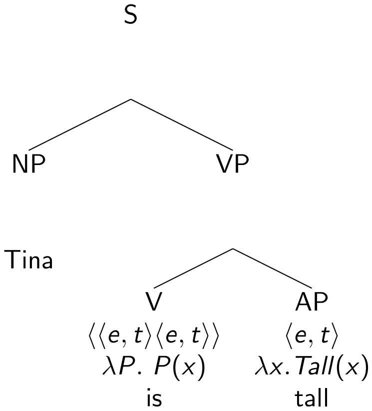
Identity Function

IS:

$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$

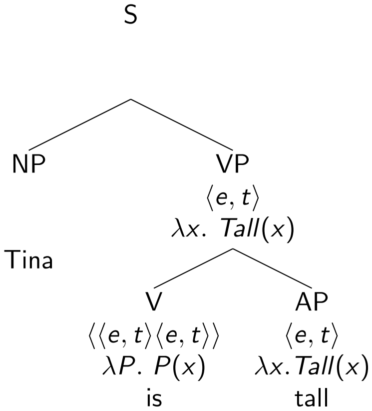
is $\rightsquigarrow \lambda P.P$

Tina is tall



Identity Function

IS:
$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$
is $\rightsquigarrow \lambda P.P$
Tina is tall



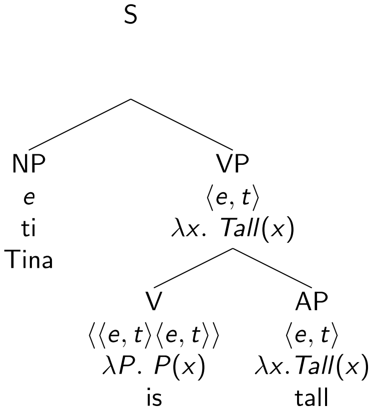
Identity Function

IS:

$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$

is $\rightsquigarrow \lambda P.P$

Tina is tall



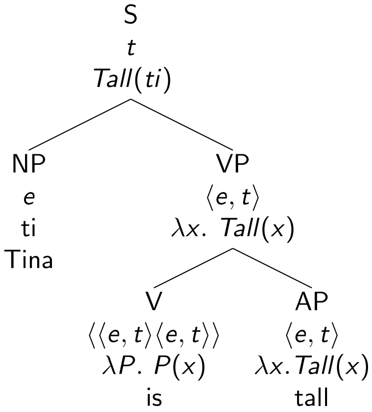
Identity Function

IS:

$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$

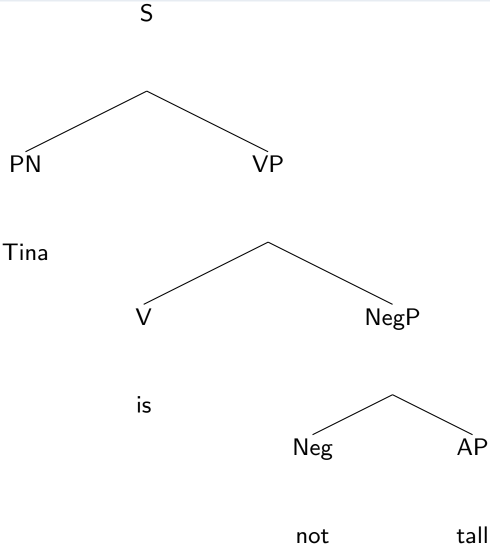
is $\rightsquigarrow \lambda P.P$

Tina is tall

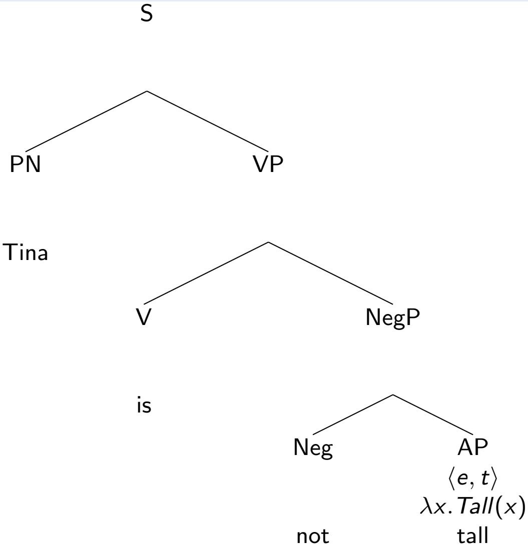


Negation

Tina is not tall

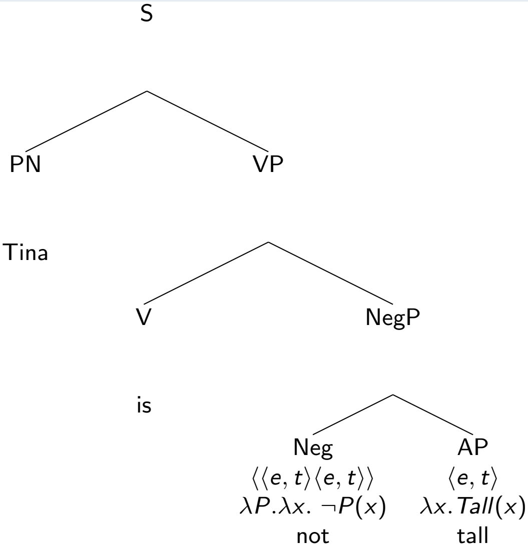


Tina is not tall



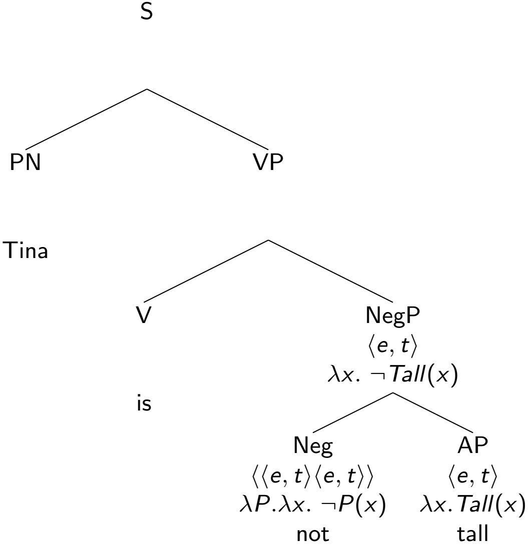
Negation

Tina is not tall



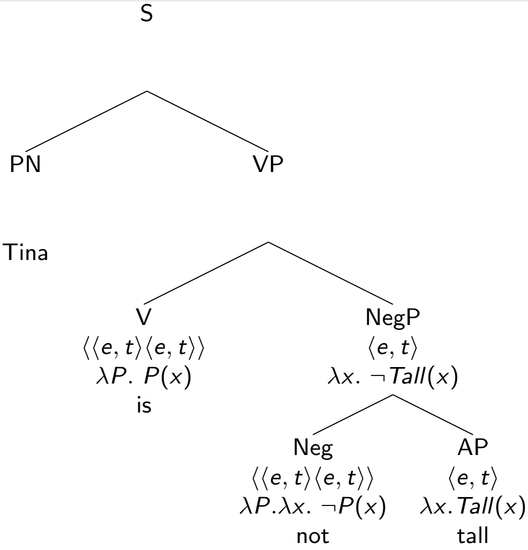
Negation

Tina is not tall



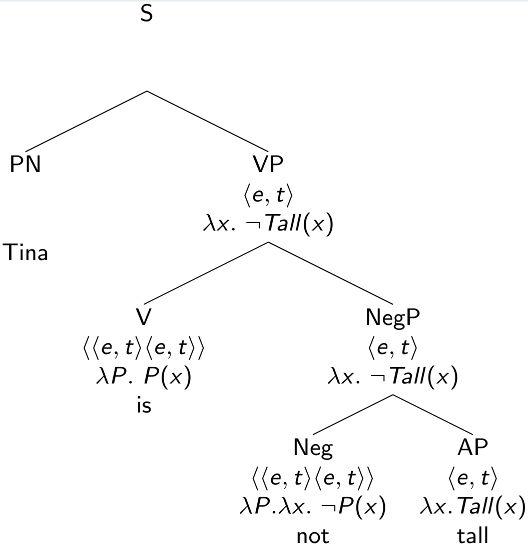
Negation

Tina is not tall



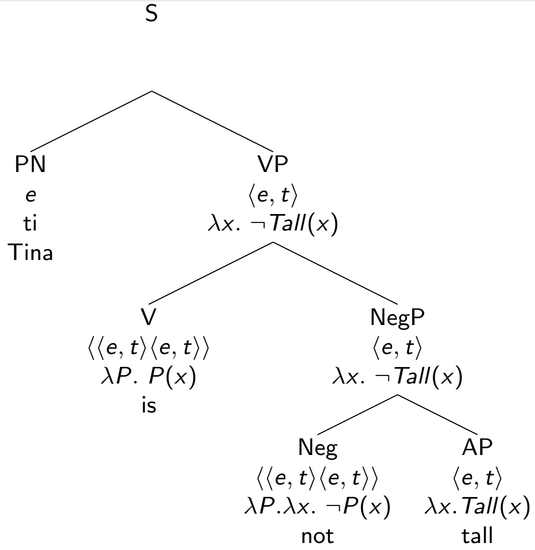
Negation

Tina is not tall



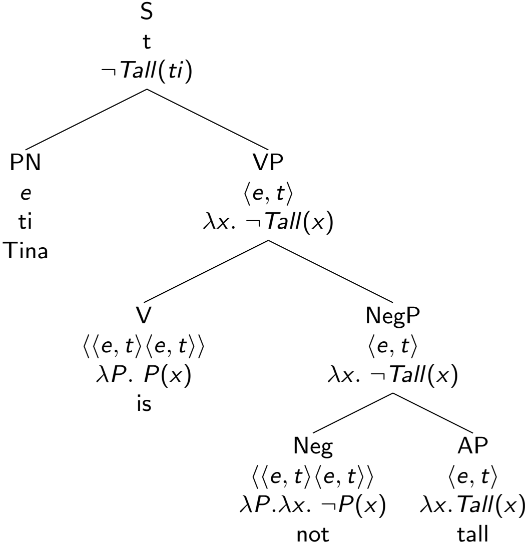
Negation

Tina is not tall



Negation

Tina is not tall



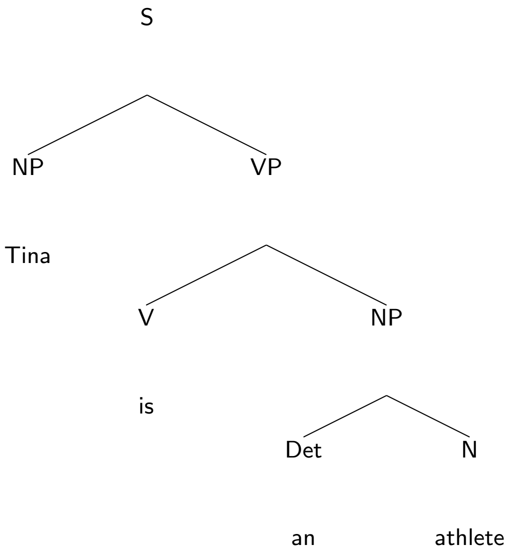
Indefinite Article

A:

$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda P.P$

Tina is an athlete.



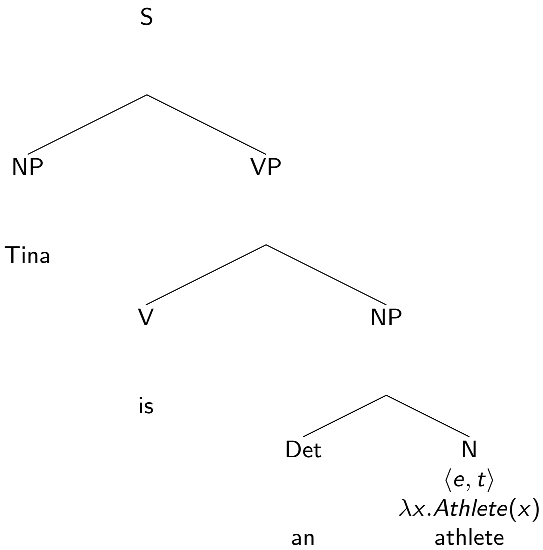
Indefinite Article

A:

$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$

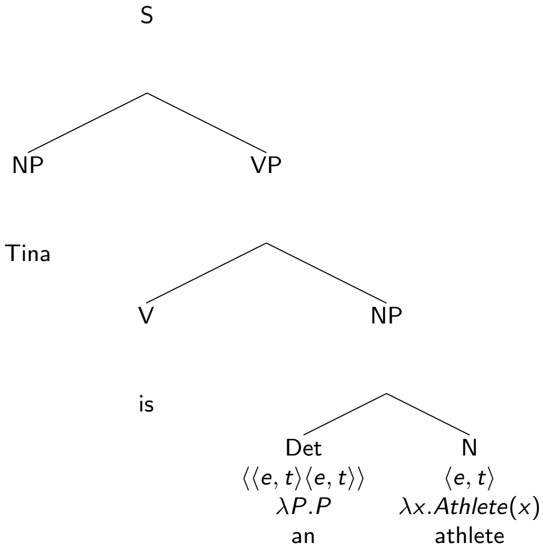
$a \rightsquigarrow \lambda P.P$

Tina is an athlete.



Indefinite Article

A:
$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
$a \rightsquigarrow \lambda P.P$
Tina is an athlete.



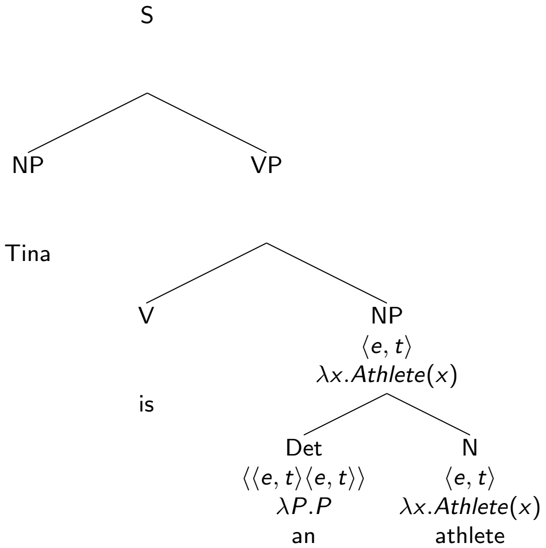
Indefinite Article

A:

$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda P.P$

Tina is an athlete.



Indefinite Article

A:

$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda P.P$

Tina is an athlete.

S

NP — Tina

VP

V: $\langle\langle e, t\rangle\langle e, t\rangle\rangle$ $\lambda P.\ P$ is

NP: $\langle e, t\rangle$ $\lambda x.Athlete(x)$

Det: $\langle\langle e, t\rangle\langle e, t\rangle\rangle$ $\lambda P.P$ an

N: $\langle e, t\rangle$ $\lambda x.Athlete(x)$ athlete

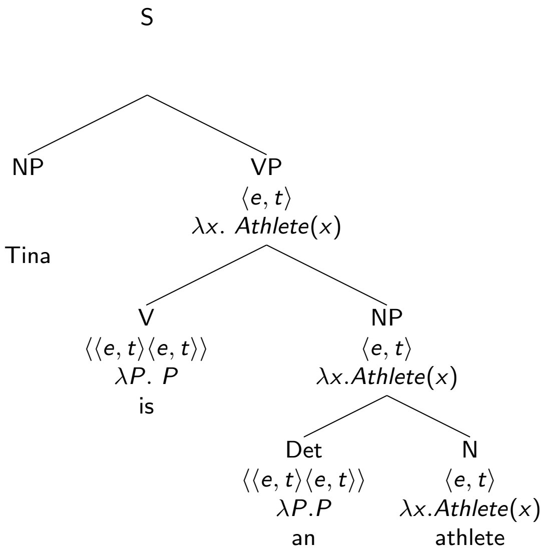
Indefinite Article

A:

$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda P.P$

Tina is an athlete.



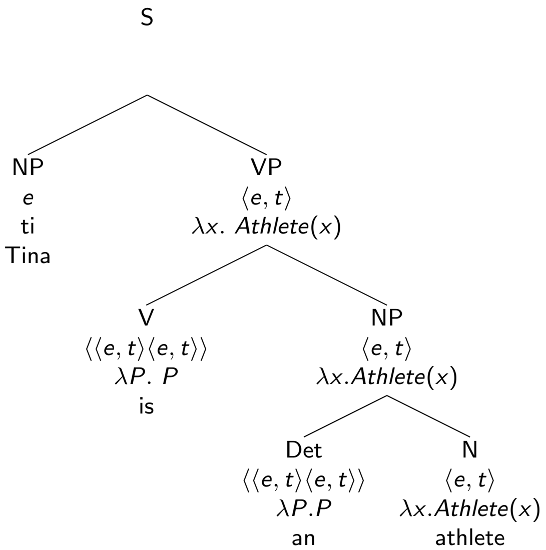
Indefinite Article

A:

$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda P.P$

Tina is an athlete.



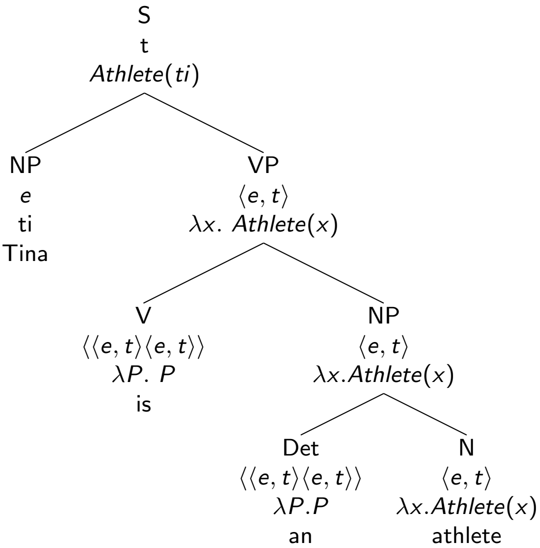
Indefinite Article

A:

$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$

$a \rightsquigarrow \lambda P.P$

Tina is an athlete.

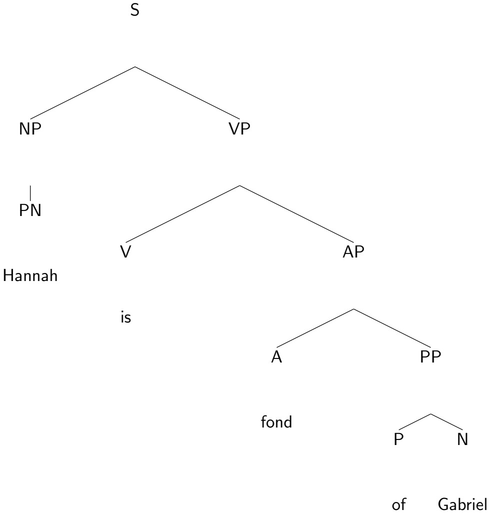


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t \rangle\rangle$

$a \rightsquigarrow \lambda x.x$

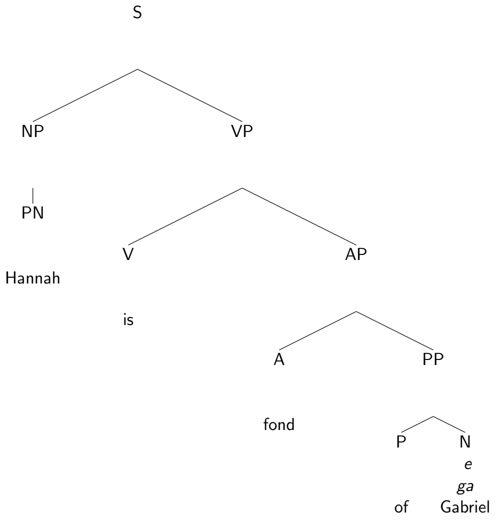
Hannah is fond of Gabriel.



Adjectives of type $\langle e, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.

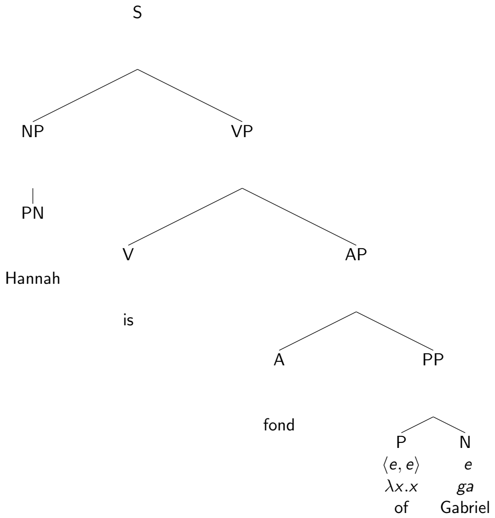


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.

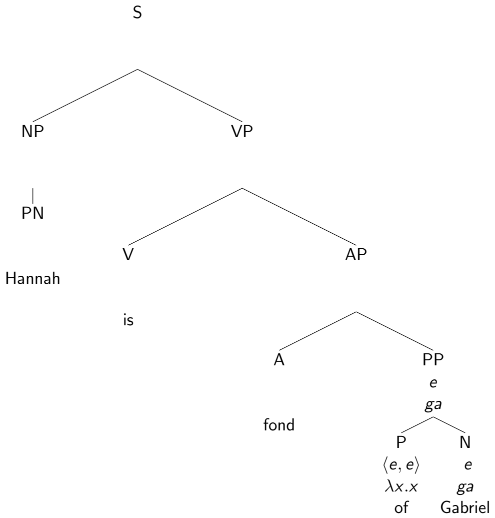


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.

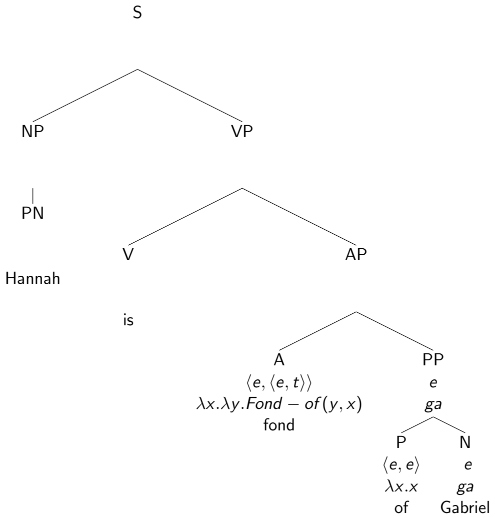


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.

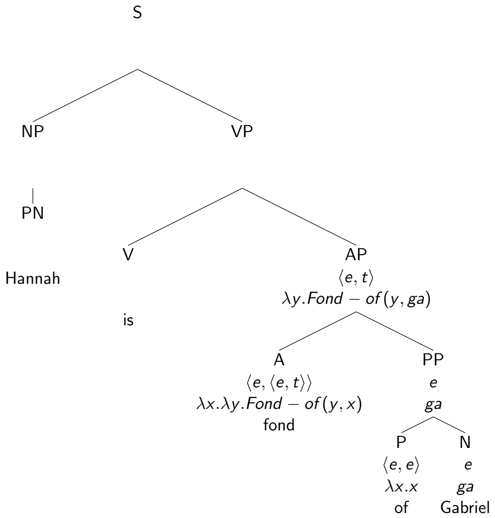


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.

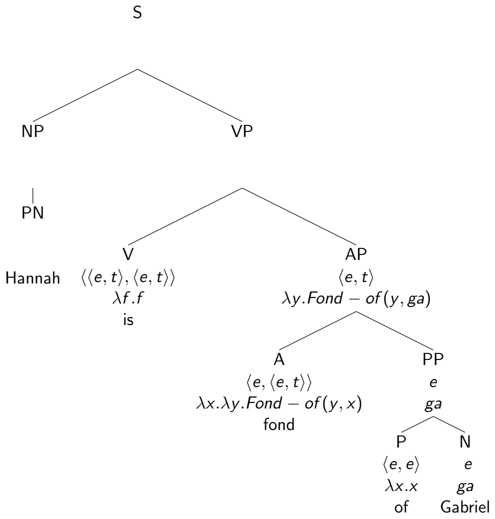


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.



Prepositional Phrases

Adjectives of type $\langle e, \langle e, t\rangle\rangle$
$a \rightsquigarrow \lambda x.x$
Hannah is fond of Gabriel.

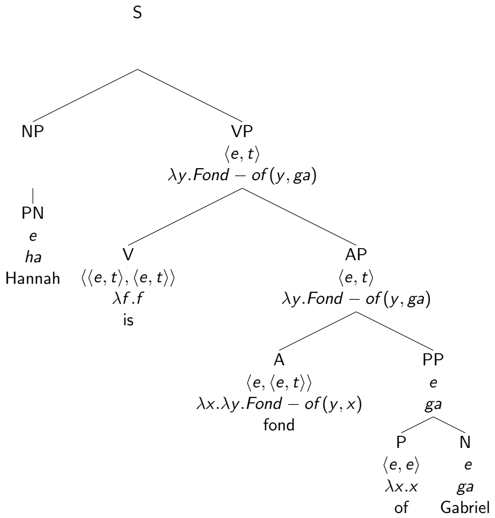
- S
  - NP
    - PN
      - Hannah
  - VP $\langle e, t\rangle$ $\lambda y.Fond-of(y, ga)$
    - V $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$ $\lambda f.f$ is
    - AP $\langle e, t\rangle$ $\lambda y.Fond-of(y, ga)$
      - A $\langle e, \langle e, t\rangle\rangle$ $\lambda x.\lambda y.Fond-of(y, x)$ fond
      - PP e ga
        - P $\langle e, e\rangle$ $\lambda x.x$ of
        - N e ga Gabriel

Prepositional Phrases

Adjectives of type $\langle e, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.

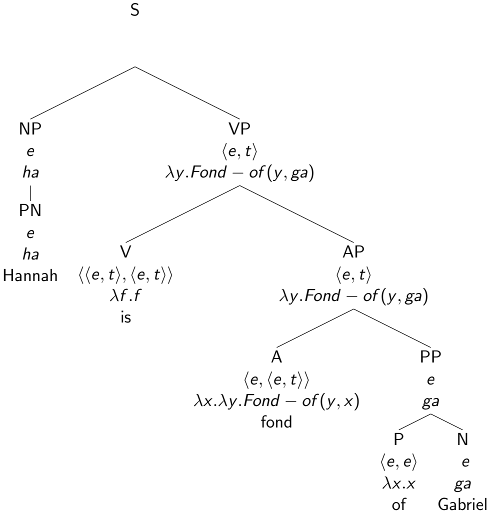


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.

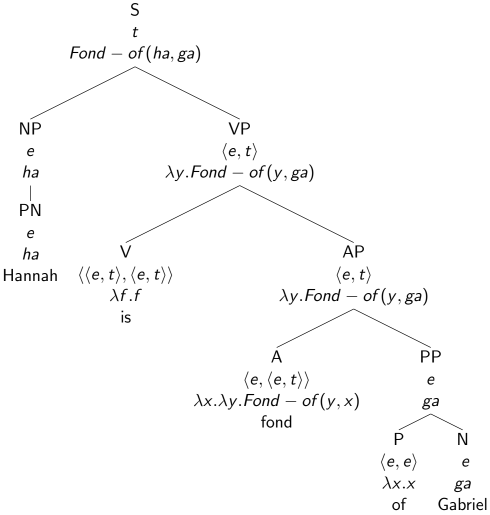


Prepositional Phrases

Adjectives of type $\langle e, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda x.x$

Hannah is fond of Gabriel.



Quantification: type $\langle\langle e, t\rangle, t\rangle$

Everybody, *everything*, *somebody*, *something*, *nobody* and *nothing*

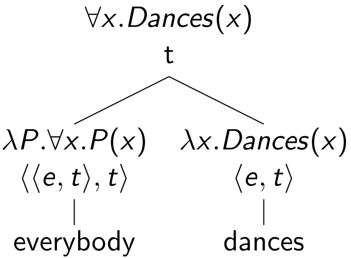
Example

Everybody dances.

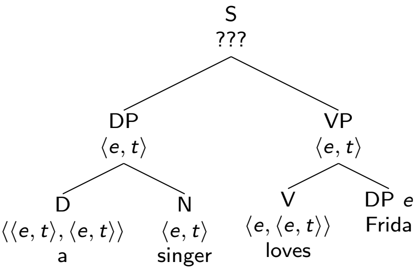
everybody $\rightsquigarrow \lambda P.\forall x.P(x)$ $\langle\langle e, t\rangle, t\rangle$

something $\rightsquigarrow \lambda P.\exists x.P(x)$ $\langle\langle e, t\rangle, t\rangle$

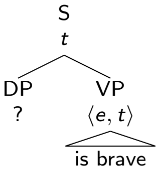
nobody $\rightsquigarrow \lambda P.\neg\exists x.P(x)$ $\langle\langle e, t\rangle, t\rangle$



Quantification: why not e



no composition rule to combine two expressions of type $\langle e, t\rangle$. Similarly: somebody/everybody/nobody is brave...



e?

Quantification: why not e (cont.)

e should validate subset-to-superset inference, e.g.

Subset-to-superset inference

Susan came yesterday morning.
$\therefore$ Susan came yesterday.

Law of non-contradiction

e do not always adhere to $[p \wedge \neg p]$ false for any p
Mont Blanc is higher than 4,000m, and Mont Blanc is not higher than 4,000m. $[p \wedge \neg p] \vdash \bot$
More than two mountains are higher than 4,000m, and more than two mountains are not higher than 4,000m. $[p \wedge \neg p] \nvdash \bot$

Quantification: predicates of predicates

Every, *some* and *no*

Example

Every cat meows.

every $\rightsquigarrow \lambda P.\lambda P'.\forall x.[P(x) \rightarrow P'(x)]$ $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$

some $\rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$ $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$

nobody $\rightsquigarrow \lambda P.\lambda P'.\neg\exists x.[P(x) \wedge P'(x)]$ $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$

Quantification: type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ (cont.)

- S — t — $\forall x.[Cat(x) \rightarrow Meows(x)]$
  - DP — $\langle\langle e,t\rangle,t\rangle$ — $\lambda P'.\forall x.[Cat(x) \rightarrow P'(x)]$
    - D — $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ — $\lambda P.\lambda P'.\forall x.[P(x) \rightarrow P'(x)]$
      - *every*
    - NP — $\langle e,t\rangle$ — $\lambda x.Cat(x)$
      - *cat*
  - VP — $\langle e,t\rangle$ — $\lambda x.Meows(x)$
    - V — $\langle e,t\rangle$ — $\lambda x.Meows(x)$
      - *meows*

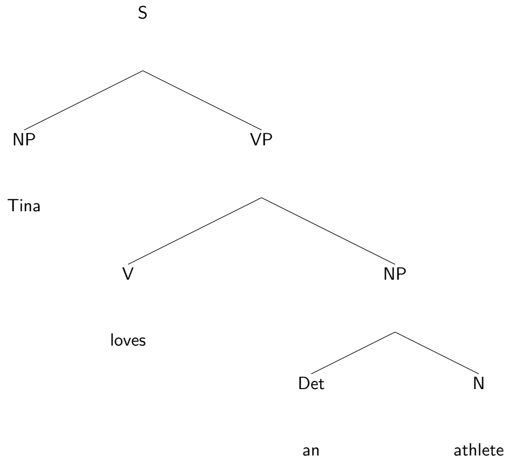
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t \rangle, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$

Tina loves an athlete.



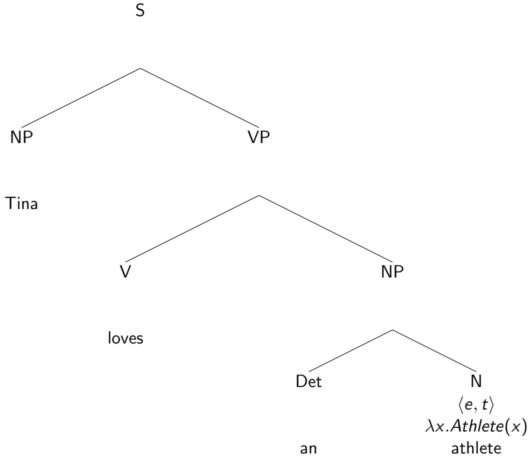
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t \rangle, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$

Tina loves an athlete.



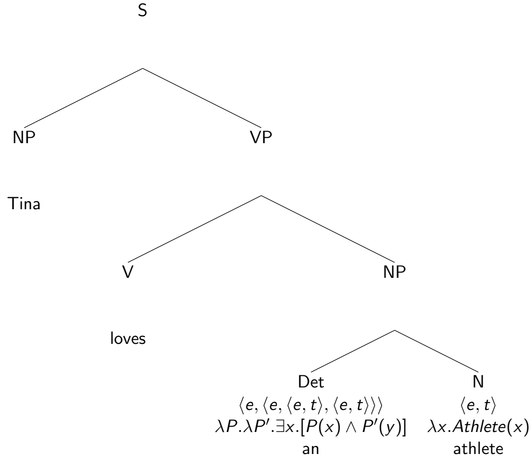
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t \rangle, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$

Tina loves an athlete.



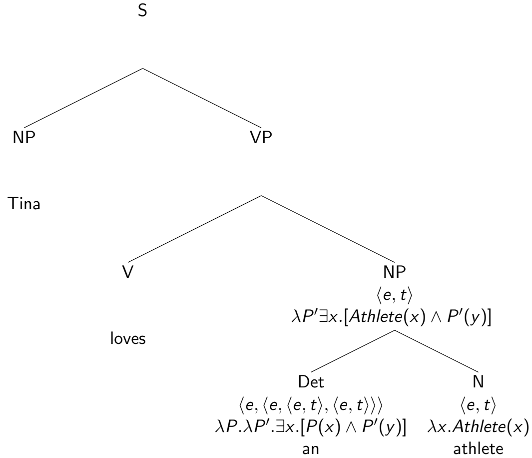
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t \rangle, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$

Tina loves an athlete.



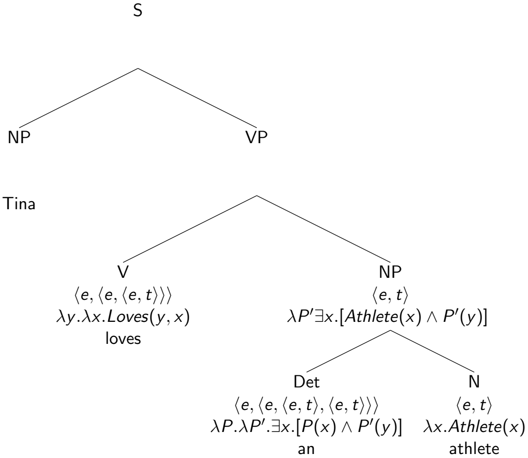
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t \rangle, \langle e, t \rangle\rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$

Tina loves an athlete.



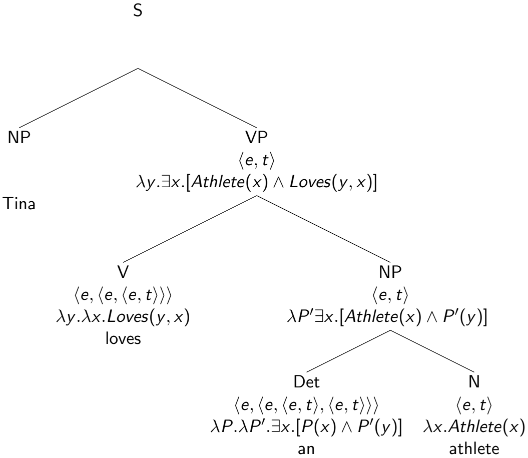
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t \rangle, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \land P'(x)]$

Tina loves an athlete.



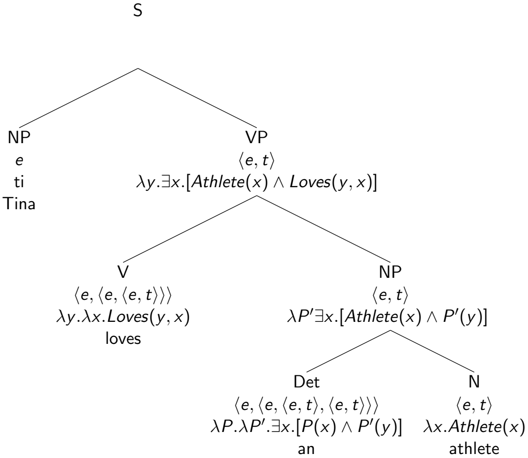
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t\rangle, \langle e, t\rangle\rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$

Tina loves an athlete.



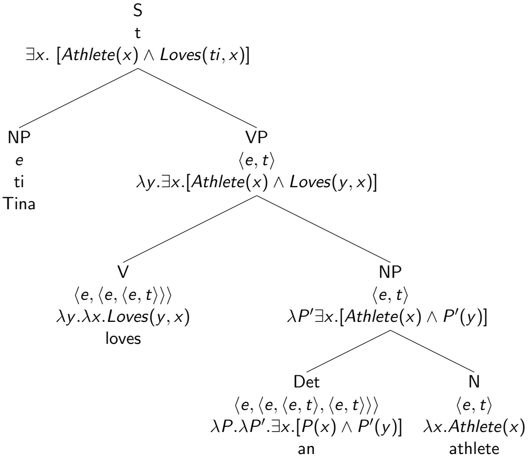
Indefinite Article: in NP in object position

A:

$\langle e, \langle e, t \rangle, \langle e, t \rangle \rangle$

$a \rightsquigarrow \lambda P.\lambda P'.\exists x.[P(x) \wedge P'(x)]$

Tina loves an athlete.



Generalised Quantifiers

There are many more quantifiers: few, exactly two, more than five sonatas, most of the

Generalised Quantifiers

There are many more quantifiers: few, exactly two, more than five sonatas, most of the quantification over time and space such as always, sometimes, never, annually, everywhere, somewhere and nowhere;

Generalised Quantifiers

There are many more quantifiers: few, exactly two, more than five sonatas, most of the quantification over time and space such as always, sometimes, never, annually, everywhere, somewhere and nowhere;
expressions such as twice (as in I called you twice and more than five times, or twice every day in I will call you twice every day and many more.

Generalised Quantifiers

There are many more quantifiers: few, exactly two, more than five sonatas, most of the quantification over time and space such as always, sometimes, never, annually, everywhere, somewhere and nowhere;
expressions such as twice (as in I called you twice and more than five times, or twice every day in I will call you twice every day and many more.

A **generalized quantifier (GQ)** is an expression that denotes a set of sets

Generalised Quantifiers

There are many more quantifiers: few, exactly two, more than five sonatas, most of the quantification over time and space such as always, sometimes, never, annually, everywhere, somewhere and nowhere;
expressions such as twice (as in I called you twice and more than five times, or twice every day in I will call you twice every day and many more.

A **generalized quantifier (GQ)** is an expression that denotes a set of sets

Example

the generalized quantifier every boy denotes the set of sets of which every boy is a member:
$\{X \mid \forall x(Boy(x) \rightarrow x \in X)\}$

Generalised Quantifiers (cont.)

In formal logic, if p is a formula that denotes a proposition then the expressions $\forall x.p$ and $\exists y.p$ are quantifications, saying that p is true of all individual objects and that p is true of at least one such object, respectively.

Generalised Quantifiers (cont.)

In formal logic, if p is a formula that denotes a proposition then the expressions $\forall x.p$ and $\exists y.p$ are quantifications, saying that p is true of all individual objects and that p is true of at least one such object, respectively.

Such quantifications, which range over all individual objects in a universe of discourse, cannot be expressed in natural languages. It just is not possible to say that something is true "for all" or "for some", where "all" and "some" would refer to any conceivable object.

Generalised Quantifiers (cont.)

In formal logic, if p is a formula that denotes a proposition then the expressions $\forall x.p$ and $\exists y.p$ are quantifications, saying that p is true of all individual objects and that p is true of at least one such object, respectively.

Such quantifications, which range over all individual objects in a universe of discourse, cannot be expressed in natural languages. It just is not possible to say that something is true "for all" or "for some", where "all" and "some" would refer to any conceivable object.

Noun phrases (NPs), expressing (generalized) quantifiers in natural language, typically consist of two parts: (1) a noun, in grammatical analysis called the 'head' of the NP, possibly with one or more adjectives, prepositional phrases or other modifiers, and (2) one or more determiners such as "a", "the", "all", "some", "most", "half of the", and "less than 200".

Generalised Quantifiers (cont.)

The head noun with its modifiers is called the **restrictor** of the quantifier and indicates a certain domain thatthe quantifier ranges over. The term **source domain** is used to indicate the set of entities (or, alternatively, the property that characterises these entities) that the restrictor refers to.

Generalised Quantifiers (cont.)

The head noun with its modifiers is called the **restrictor** of the quantifier and indicates a certain domain thatthe quantifier ranges over. The term **source domain** is used to indicate the set of entities (or, alternatively, the property that characterises these entities) that the restrictor refers to.

The presence of a restrictor component forms the fundamental difference between quantification in logic and quantification in natural language: quantification in logic is always understood as ranging over the set of all entities in a given universe of discourse, whereas quantification in natural language is restricted to a source domain that is made explicit in the quantifier's restrictor.

Generalised Quantifiers (cont.)

The head noun with its modifiers is called the **restrictor** of the quantifier and indicates a certain domain thatthe quantifier ranges over. The term **source domain** is used to indicate the set of entities (or, alternatively, the property that characterises these entities) that the restrictor refers to.

The presence of a restrictor component forms the fundamental difference between quantification in logic and quantification in natural language: quantification in logic is always understood as ranging over the set of all entities in a given universe of discourse, whereas quantification in natural language is restricted to a source domain that is made explicit in the quantifier's restrictor.

Example

Everybody must hand in his essay before Thursday next week.
The proposal was accepted by all the twenty-seven member countries.

Generalised Quantifiers (cont.)

The head noun with its modifiers is called the **restrictor** of the quantifier and indicates a certain domain thatthe quantifier ranges over. The term **source domain** is used to indicate the set of entities (or, alternatively, the property that characterises these entities) that the restrictor refers to.

The presence of a restrictor component forms the fundamental difference between quantification in logic and quantification in natural language: quantification in logic is always understood as ranging over the set of all entities in a given universe of discourse, whereas quantification in natural language is restricted to a source domain that is made explicit in the quantifier's restrictor.

Example

Everybody must hand in his essay before Thursday next week.
The proposal was accepted by all the twenty-seven member countries.

Westerstahl (1985) introduced the term **'context set'** to designate the contextually determined subset of a source domain that is relevant in a quantified predication.

Generalised Quantifiers (cont.)

The determiner part may be a sequence of determiners of different types, distinguished by sequencing and co-occurrence restrictions. For example, in English grammar it is customary the make a distinction between:

- **predeterminers** express the (absolute or proportional) quantitative involvement of the source domain, and may, in addition, say something about the distribution of a quantifying predicate over the source domain;
- **central determiners** express the definiteness of the NP;
- **postdeterminers** express a proposition about the cardinality of the reference domain

Example

All of her nine grandchildren are boys.

Generalised Quantifiers (cont.)

Generalised Quantifiers (cont.)

Generalised Quantifiers (cont.)

Some quantifiers are FOL definable.

Example

exactly two things $\rightsquigarrow \lambda P.\exists x.\exists y.\neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg\exists z.P(z) \wedge \neg(z = x) \wedge \neg(z = y)$

Generalised Quantifiers (cont.)

Some quantifiers are FOL definable.

Example

exactly two things $\rightsquigarrow \lambda P.\exists x.\exists y.\neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg\exists z.P(z) \wedge \neg(z = x) \wedge \neg(z = y)$

Others are not FOL definable

Example

most swans $=_{def} \{P \subseteq D_e : |SWAN \cap P| > |SWAN - P|\}$
most swans $=_{def} \{P \subseteq D_e : |P| > |D_e - P|\}$
one of the three cats $=_{def} \{P \subseteq D_e : |P \cup CAT|/|CAT| \geq 1/3\}$

Generalised Quantifiers (cont.)

Some quantifiers are FOL definable.

Example

exactly two things $\rightsquigarrow \lambda P.\exists x.\exists y.\neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg\exists z.P(z) \wedge \neg(z = x) \wedge \neg(z = y)$

Others are not FOL definable

Example

most swans $=_{def} \{P \subseteq D_e : |SWAN \cap P| > |SWAN - P|\}$
most swans $=_{def} \{P \subseteq D_e : |P| > |D_e - P|\}$
one of the three cats $=_{def} \{P \subseteq D_e : |P \cup CAT|/|CAT| \geq 1/3\}$

solution: relate and compare sets; represent in second-order logics

Example

Generalised Quantifiers (cont.)

Some quantifiers are FOL definable.

Example

exactly two things $\rightsquigarrow \lambda P.\exists x.\exists y.\neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg\exists z.P(z) \wedge \neg(z = x) \wedge \neg(z = y)$

Others are not FOL definable

Example

most swans $=_{def} \{P \subseteq D_e : |SWAN \cap P| > |SWAN - P|\}$
most swans $=_{def} \{P \subseteq D_e : |P| > |D_e - P|\}$
one of the three cats $=_{def} \{P \subseteq D_e : |P \cup CAT|/|CAT| \geq 1/3\}$

solution: relate and compare sets; represent in second-order logics

Example

$\exists X[|X| = 2 \wedge \forall x[x \in X \rightarrow Man(x)] \wedge \exists y[Piano(y) \wedge Carry(x, y)]]$

Generalised Quantifiers (cont.)

Some quantifiers are FOL definable.

Example

exactly two things $\rightsquigarrow \lambda P.\exists x.\exists y.\neg(x = y) \land P(x) \land P(y) \land \neg\exists z.P(z) \land \neg(z = x) \land \neg(z = y)$

Others are not FOL definable

Example

most swans $=_{def} \{P \subseteq D_e : |SWAN \cap P| > |SWAN - P|\}$
most swans $=_{def} \{P \subseteq D_e : |P| > |D_e - P|\}$
one of the three cats $=_{def} \{P \subseteq D_e : |P \cup CAT|/|CAT| \geq 1/3\}$

solution: relate and compare sets; represent in second-order logics

Example

$\exists X[|X| = 2 \land \forall x[x \in X \rightarrow Man(x)] \land \exists y[Piano(y) \land Carry(x, y)]]$
Two men carried a piano upstairs.

Generalised Quantifiers

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def}\ C \subseteq E$

Generalised Quantifiers

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def}\ C \subseteq E$
some of the crew escaped the blast $=_{def}\ |C \cap E| \neq \{\}$

Generalised Quantifiers

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def}$ $C \subseteq E$
some of the crew escaped the blast $=_{def}$ $|C \cap E| \neq \{\}$
none of the crew escaped the blast $=_{def}$ $|C \cap E| = \{\}$

Generalised Quantifiers

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def}\ C \subseteq E$
some of the crew escaped the blast $=_{def} |C \cap E| \neq \{\}$
none of the crew escaped the blast $=_{def} |C \cap E| = \{\}$
most of the crew escaped the blast $=_{def} |C \cap E| > 1/2|c|$

Generalised Quantifiers

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def}$ $C \subseteq E$
some of the crew escaped the blast $=_{def}$ $|C \cap E| \neq \{\}$
none of the crew escaped the blast $=_{def}$ $|C \cap E| = \{\}$
most of the crew escaped the blast $=_{def}$ $|C \cap E| > 1/2|c|$

seems universal across all human languages! Is SEMANTIC UNIVERSAL property

Generalised Quantifiers

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def}$ $C \subseteq E$
some of the crew escaped the blast $=_{def}$ $|C \cap E| \neq \{\}$
none of the crew escaped the blast $=_{def}$ $|C \cap E| = \{\}$
most of the crew escaped the blast $=_{def}$ $|C \cap E| > 1/2|c|$

seems universal across all human languages! Is SEMANTIC UNIVERSAL property

All quantifiers seem to have the property of CONSERVATIVITY: quantifiers only care about the elements in the first set they combine with, and so they ignore anything in the second set that's not already in the first.
$[Q](A)(B) \leftrightarrow [Q](A)(A \cap B)$

Generalised Quantifiers

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def} C \subseteq E$
some of the crew escaped the blast $=_{def} |C \cap E| \neq \{\}$
none of the crew escaped the blast $=_{def} |C \cap E| = \{\}$
most of the crew escaped the blast $=_{def} |C \cap E| > 1/2|c|$

seems universal across all human languages! Is SEMANTIC UNIVERSAL property

All quantifiers seem to have the property of CONSERVATIVITY: quantifiers only care about the elements in the first set they combine with, and so they ignore anything in the second set that's not already in the first.
$[Q](A)(B) \leftrightarrow [Q](A)(A \cap B)$

Example

all of the crew escaped the blast does not care who else escaped the blast $|E - C|$.

Example

all of the crew escaped the blast $\rightsquigarrow \forall x[C(x) \implies E(x)]$ or $=_{def}\ C \subseteq E$
some of the crew escaped the blast $=_{def}\ |C \cap E| \neq \{\}$
none of the crew escaped the blast $=_{def}\ |C \cap E| = \{\}$
most of the crew escaped the blast $=_{def}\ |C \cap E| > 1/2|c|$

seems universal across all human languages! Is SEMANTIC UNIVERSAL property

All quantifiers seem to have the property of CONSERVATIVITY: quantifiers only care about the elements in the first set they combine with, and so they ignore anything in the second set that's not already in the first.
$[Q](A)(B) \leftrightarrow [Q](A)(A \cap B)$

Example

all of the crew escaped the blast does not care who else escaped the blast $|E - C|$.

Quantifiers that do not depend on $|E \cup C|$ satisfy EXTENSION, e.g. only, there constructions

Quantifiers in object position

Exercises: hk-chapter6 & example 3

Quizz for Today

TBA