Accessibility for Plurals in Continuation Semantics

Sai Qian, Maxime Amblard

{sai.qian,maxime.amblard}@loria.fr

Semagramme, LORIA & INRIA Nancy Grand-Est
UFR Math-Info, Université de Lorraine

Logic and Engineering of Natural Language Semantics 9 (LENLS 9)

November 30, 2012

Outline

Overview

Key Words

Plurality (Group, Individual), Dynamic Semantics, Continuation, DRT, Anaphoric Accessibility, Functional Programming

Main goals of the presentation:

1. Investigating two plurality formations (mostly based on [Kamp and Reyle, 1993])
2. Compositionally obtaining the semantic representation for plurality under dynamic semantics

Overview

Key Words

Plurality (Group, Individual), Dynamic Semantics, Continuation, DRT, Anaphoric Accessibility, Functional Programming

Main goals of the presentation:

1. Investigating two plurality formations (mostly based on [Kamp and Reyle, 1993])
2. Compositionally obtaining the semantic representation for plurality under dynamic semantics

Cohesion & Anaphora

- Anaphora
  - Some Terminologies: *cohesion*, *anaphor*, *antecedent*
  - Anaphora ties pieces of discourse into a "unified whole"

Example (Anaphora)

Cohesion & Anaphora

- Anaphora
  - Some Terminologies: *cohesion*, *anaphor*, *antecedent*
  - Anaphora ties pieces of discourse into a "unified whole"

Example (Anaphora)

(1)
a. *John*$_1$ has *a car*$_2$. *He*$_1$ likes *it*$_2$.
b. *John*$_1$ has a car. *His*$_1$ car is red.
c. John has *a car*$_1$. *The car*$_1$ is red.
d. John has a *cool car*$_1$. Mary has a *same one*$_1$.
e. *John drives to work everyday*$_1$. *It*$_1$ takes him half an hour.

Cohesion & Anaphora

- Anaphora
  - Some Terminologies: *cohesion*, *anaphor*, *antecedent*
  - Anaphora ties pieces of discourse into a "unified whole"

Example (Anaphora)

(1) a. *John*$_1$ has *a car*$_2$. *He*$_1$ likes *it*$_2$.
b. *John*$_1$ has a car. *His*$_1$ car is red.
c. John has *a car*$_1$. *The car*$_1$ is red.
d. John has a *cool car*$_1$. Mary has a *same one*$_1$.
e. *John drives to work everyday*$_1$. *It*$_1$ takes him half an hour.

The Problem of Plurality

- The semantics of plurality is not a naïve quantitative extension of singularity

Example (Distributivity vs. Collectivity)

(2) a. John and Mary went to school.
b. John and Mary gathered in Paris.
c. John and Mary lifted a piano.

- Singular and Plural Pronouns
  - *he*, *she*, *I*: **individuals**
  - *we*, *they*, *you*: **group** of individuals

The Problem of Plurality

- The semantics of plurality is not a naïve quantitative extension of singularity

Example (Distributivity vs. Collectivity)

(2) a. John and Mary went to school.
b. John and Mary gathered in Paris.
c. John and Mary lifted a piano.

- Singular and Plural Pronouns
  - *he*, *she*, *I*: **individuals**
  - *we*, *they*, *you*: **group** of individuals

The Problem of Plurality

- The semantics of plurality is not a naïve quantitative extension of singularity

Example (Distributivity vs. Collectivity)

(2) a. John and Mary went to school.
b. John and Mary gathered in Paris.
c. John and Mary lifted a piano.

- Singular and Plural Pronouns
  - *he*, *she*, *I*: **individuals**
  - *we*, *they*, *you*: **group** of individuals

Summation Sketch

Definition (Summation) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of explicit individuals.

Example (Summation Sketch)

(3) a. John went to Bill's party with Mary. They had a nice time.

b. John loves Mary. Bill also loves Mary. They have to find a solution.

- Plural referents (groups of individuals) do not need necessarily be explicitly mentioned in the context, e.g.,

In (3-a): John $\oplus$ Bill $\oplus$ Mary;

in (3-b): John $\oplus$ Bill, John $\oplus$ Bill $\oplus$ Mary

Summation Sketch

Definition (Summation) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of explicit individuals.

Example (Summation Sketch)

(3) a. John went to Bill's party with Mary. They had a nice time.

b. John loves Mary. Bill also loves Mary. They have to find a solution.

- Plural referents (groups of individuals) do not need necessarily be explicitly mentioned in the context, e.g.,

In (3-a): John $\oplus$ Bill $\oplus$ Mary;
in (3-b): John $\oplus$ Bill, John $\oplus$ Bill $\oplus$ Mary

Summation Sketch

Definition (Summation) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of explicit individuals.

Example (Summation Sketch)

(3) a. John went to Bill's party with Mary. They had a nice time.

b. John loves Mary. Bill also loves Mary. They have to find a solution.

- Plural referents (groups of individuals) do not need necessarily be explicitly mentioned in the context, e.g.,

In (3-a): John $\oplus$ Bill $\oplus$ Mary;
in (3-b): John $\oplus$ Bill, John $\oplus$ Bill $\oplus$ Mary

Summation Sketch

Definition (Summation) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of explicit individuals.

Example (Summation Sketch)

(3) a. John went to Bill's party with Mary. They had a nice time.

b. John loves Mary. Bill also loves Mary. They have to find a solution.

- Plural referents (groups of individuals) do not need necessarily be explicitly mentioned in the context, e.g.,

In (3-a): John $\oplus$ Bill $\oplus$ Mary;
in (3-b): John $\oplus$ Bill, John $\oplus$ Bill $\oplus$ Mary

Summation Sketch Continued

Example (Summation Sketch Continued)

(4) John went to Paris. Bill and Mary gathered to Rome.

a. She enjoyed the historical monuments very much.

b. They planned the whole trip without telling her.

- Even plural referents are explicitly mentioned, the individual components can **be broken down** and **re-form** other plural referents, e.g.,

In (4-a): from Bill $\oplus$ Mary $\Rightarrow$ Mary;
in (4-b): from John, Bill $\oplus$ Mary $\Rightarrow$ John $\oplus$ Bill

Summation Sketch Continued

Example (Summation Sketch Continued)

(4) John went to Paris. Bill and Mary gathered to Rome.

a. She enjoyed the historical monuments very much.

b. They planned the whole trip without telling her.

- Even plural referents are explicitly mentioned, the individual components can **be broken down** and **re-form** other plural referents, e.g.,

In (4-a): from Bill $\oplus$ Mary $\Rightarrow$ Mary;
in (4-b): from John, Bill $\oplus$ Mary $\Rightarrow$ John $\oplus$ Bill

Summation Sketch Continued

Example (Summation Sketch Continued)

(4) John went to Paris. Bill and Mary gathered to Rome.

a. She enjoyed the historical monuments very much.

b. They planned the whole trip without telling her.

- Even plural referents are explicitly mentioned, the individual components can **be broken down** and **re-form** other plural referents, e.g.,

$$\text{In (4-a): from Bill} \oplus \text{Mary} \Rightarrow \text{Mary;}$$
$$\text{in (4-b): from John, Bill} \oplus \text{Mary} \Rightarrow \text{John} \oplus \text{Bill}$$

Abstraction Sketch

Definition (Abstraction) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of quantified noun phrases.

- Quantified NP: *quantifier* + *noun*
- Generalized quantifier: *every*, *all*, *none*, *most*, *few*, etc.

Example (Abstraction Sketch)

(5) a. Every farmer owns a donkey. *He is /They are rich.
b. Few students came on time. They were too lazy.

every $\Rightarrow$; *few* $\Rightarrow$

Abstraction Sketch

Definition (Abstraction) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of quantified noun phrases.

- Quantified NP: *quantifier* + *noun*
- Generalized quantifier: *every*, *all*, *none*, *most*, *few*, etc.

Example (Abstraction Sketch)

(5) a. Every farmer owns a donkey. *He is /They are rich.
b. Few students came on time. They were too lazy.

every $\Rightarrow$; *few* $\Rightarrow$

Abstraction Sketch

Definition (Abstraction) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of quantified noun phrases.

- Quantified NP: *quantifier* + *noun*
- Generalized quantifier: *every*, *all*, *none*, *most*, *few*, etc.

Example (Abstraction Sketch)

(5) a. Every farmer owns a donkey. *He is /They are rich.
b. Few students came on time. They were too lazy.

every $\Rightarrow$; *few* $\Rightarrow$

Abstraction Sketch

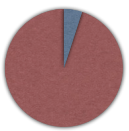
Definition (Abstraction) [Kamp and Reyle, 1993]

The process of constructing plural referents (groups of individuals) out of quantified noun phrases.

- Quantified NP: *quantifier* + *noun*
- Generalized quantifier: *every*, *all*, *none*, *most*, *few*, etc.

Example (Abstraction Sketch)

(5) a. Every farmer owns a donkey. *He is /They are rich.
b. Few students came on time. They were too lazy.

every ⇒  ; *few* ⇒ 

A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context

A New Approach to Dynamics [de Groote, 2006]

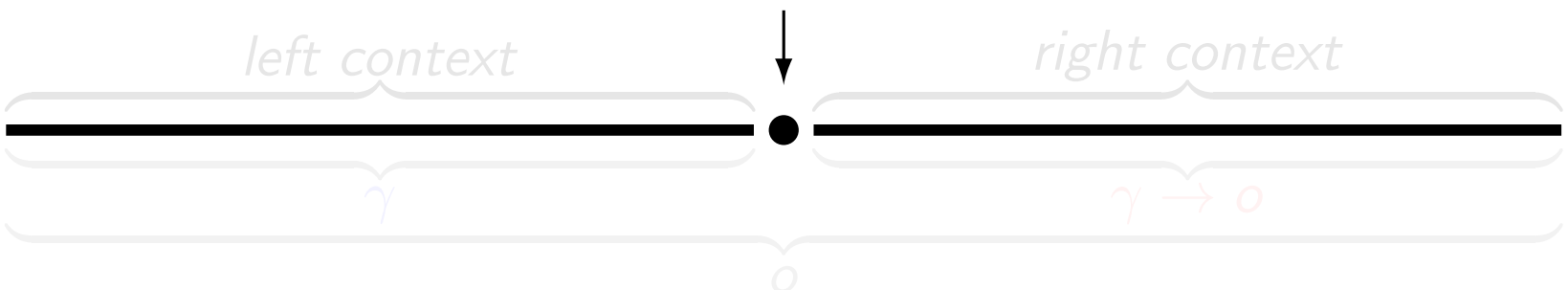
- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context

A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context

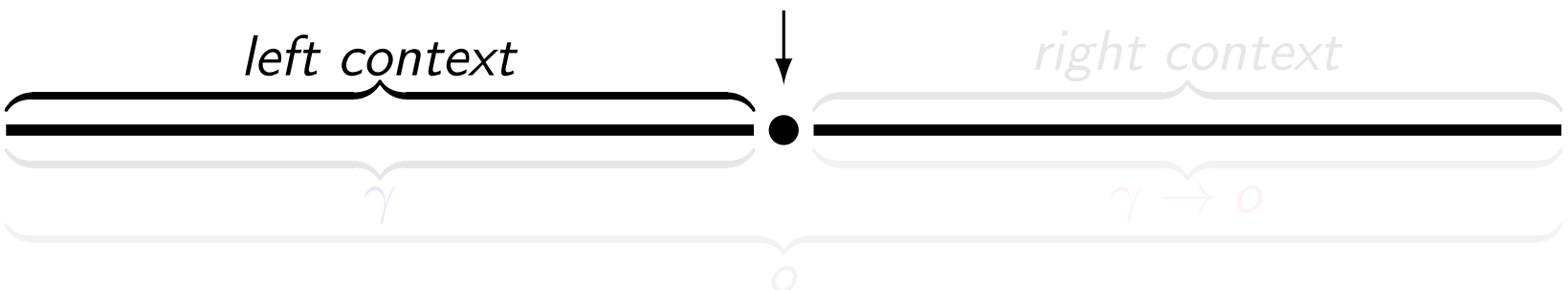
A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context



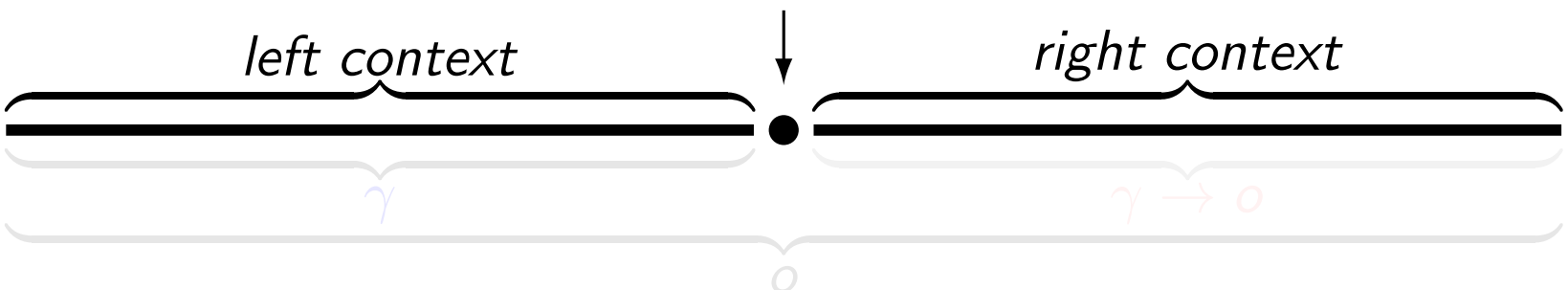
A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context



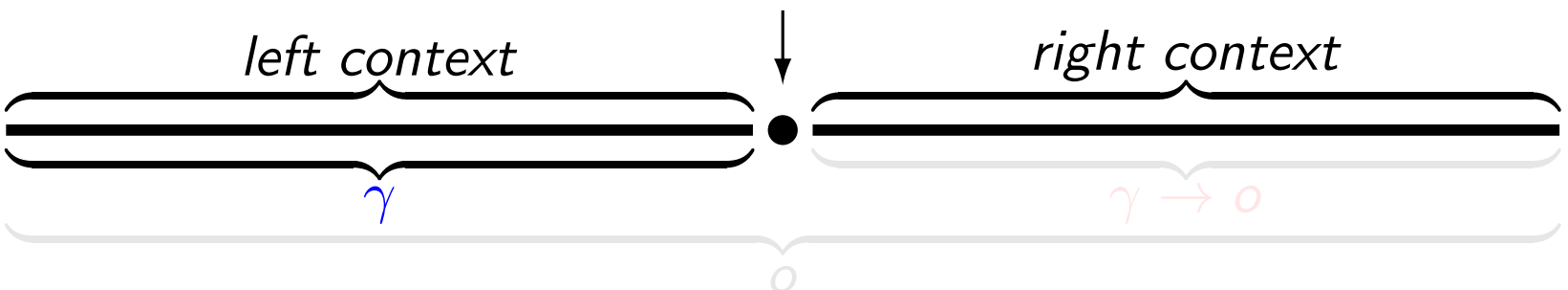
A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context



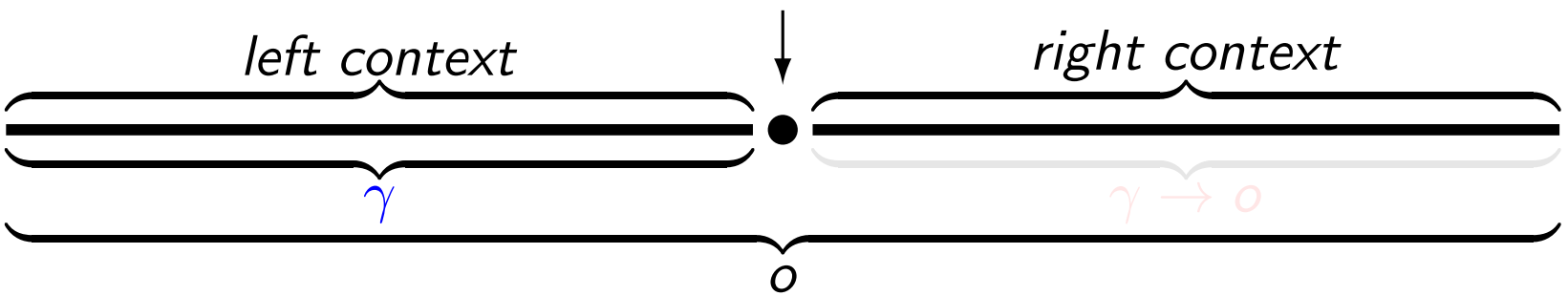
A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context



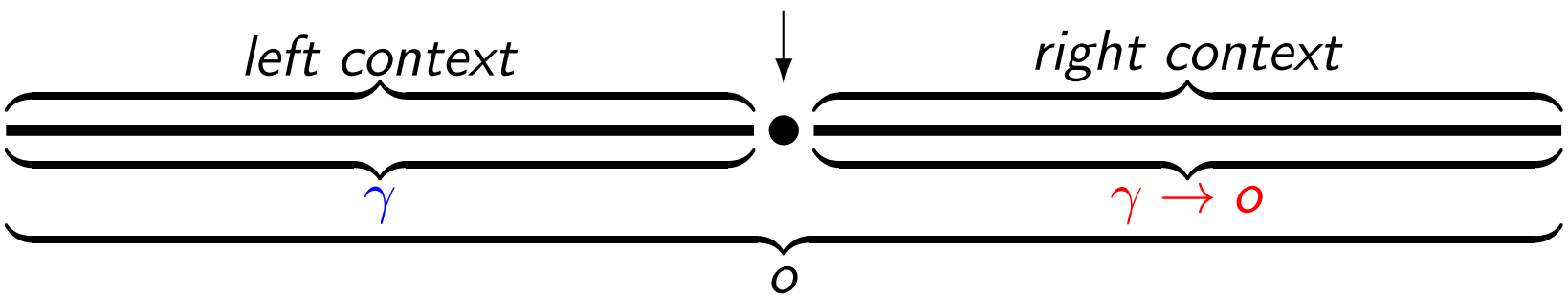
A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context



A New Approach to Dynamics [de Groote, 2006]

- A **pure** Montagovian framework for discourse dynamics
- Basic Types
  - ι (e): individuals/entities
  - o (t): propositions/truth values
  - γ: left context



Type System & Composition

Typing Rules

$[\![s]\!]$	$\gamma \to (\gamma \to o) \to o$	o
$[\![n]\!]$	$\iota \to [\![s]\!]$	$\iota \to o$
$[\![np]\!]$	$(\iota \to [\![s]\!]) \to [\![s]\!]$	$(\iota \to o) \to o$

Discourse Composition

$$[\![D.S]\!] = \lambda e\phi.[\![D]\!]e(\lambda e'.[\![S]\!]e'\phi)$$

- A general DRS corresponds to:

$$\lambda e\phi.\exists x_1 \cdots x_n.C_1 \wedge \cdots C_m \wedge \phi e'$$

- e' is a left context made of e and the variables $x_1, x_2, x_3, \ldots$

Type System & Composition

Typing Rules

$[\![s]\!]$	$\gamma \to (\gamma \to o) \to o$	o
$[\![n]\!]$	$\iota \to [\![s]\!]$	$\iota \to o$
$[\![np]\!]$	$(\iota \to [\![s]\!]) \to [\![s]\!]$	$(\iota \to o) \to o$

Discourse Composition

$$[\![D.S]\!] = \lambda e\phi.[\![D]\!]e(\lambda e'.[\![S]\!]e'\phi)$$

- A general DRS corresponds to:

$$\lambda e\phi.\exists x_1 \cdots x_n.C_1 \wedge \cdots C_m \wedge \phi e'$$

- e' is a left context made of e and the variables $x_1, x_2, x_3, \ldots$

Type System & Composition

Typing Rules

$[\![s]\!]$	$\gamma \to (\gamma \to o) \to o$	o
$[\![n]\!]$	$\iota \to [\![s]\!]$	$\iota \to o$
$[\![np]\!]$	$(\iota \to [\![s]\!]) \to [\![s]\!]$	$(\iota \to o) \to o$

Discourse Composition

$$[\![D.S]\!] = \lambda e \phi . [\![D]\!] e (\lambda e' . [\![S]\!] e' \phi)$$

- A general DRS corresponds to:

$$\lambda e \phi . \exists x_1 \cdots x_n . C_1 \wedge \cdots C_m \wedge \phi e'$$

- e' is a left context made of e and the variables $x_1, x_2, x_3, \ldots$

Type System & Composition

Typing Rules

$[\![s]\!]$	$\gamma \to (\gamma \to o) \to o$	o
$[\![n]\!]$	$\iota \to [\![s]\!]$	$\iota \to o$
$[\![np]\!]$	$(\iota \to [\![s]\!]) \to [\![s]\!]$	$(\iota \to o) \to o$

Discourse Composition

$$[\![D.S]\!] = \lambda e\phi.[\![D]\!]e(\lambda e'.[\![S]\!]e'\phi)$$

- A general DRS corresponds to:

$$\lambda e\phi.\exists x_1 \cdots x_n.C_1 \wedge \cdots C_m \wedge \phi e'$$

- e' is a left context made of e and the variables $x_1, x_2, x_3, \ldots$

Lexical Entries

Lexicon	**λ-Expression**
John/Mary	$\lambda\psi e\phi.\psi\mathbf{j}/\mathbf{m}(\mathbf{j}/\mathbf{m} :: e)\phi$
she/they	$\lambda\psi e\phi.\psi(sel_{she/they}\, e)e\phi$
smiles	$\lambda s.s(\lambda xe\phi.Smile(x) \wedge \phi e)$
kisses	$\lambda os.s(\lambda x.o(\lambda ye\phi.Kiss(x, y) \wedge \phi e))$

- Remarks
  - "::" adjoins **accessible variables** in the selection list
  - "sel_{she}" selects the **correct variable** from the list

$\iota \to \gamma \to \gamma$

$\gamma \to \iota$

Lexical Entries

Lexicon	**λ-Expression**
John/Mary	$\lambda\psi e\phi.\psi\mathbf{j}/\mathbf{m}(\mathbf{j}/\mathbf{m} :: e)\phi$
she/they	$\lambda\psi e\phi.\psi(sel_{she/they}\, e)e\phi$
smiles	$\lambda s.s(\lambda xe\phi.Smile(x) \wedge \phi e)$
kisses	$\lambda os.s(\lambda x.o(\lambda ye\phi.Kiss(x, y) \wedge \phi e))$

- Remarks
  - "::" adjoins **accessible variables** in the selection list
  - "sel_{she}" selects the **correct variable** from the list

Lexical Entries

Lexicon	**λ-Expression**
John/Mary	$\lambda\psi e\phi.\psi\mathbf{j}/\mathbf{m}(\mathbf{j}/\mathbf{m} :: e)\phi$
she/they	$\lambda\psi e\phi.\psi(sel_{she/they}\, e)e\phi$
smiles	$\lambda s.s(\lambda xe\phi.Smile(x) \wedge \phi e)$
kisses	$\lambda os.s(\lambda x.o(\lambda ye\phi.Kiss(x, y) \wedge \phi e))$

- Remarks
  - "::" adjoins **accessible variables** in the selection list

    $\iota \rightarrow \gamma \rightarrow \gamma$
  - "sel_{she}" selects the **correct variable** from the list

    $\gamma \rightarrow \iota$

Lexical Entries

Lexicon	**λ-Expression**
John/Mary	$\lambda\psi e\phi.\psi\mathbf{j}/\mathbf{m}(\mathbf{j}/\mathbf{m} :: e)\phi$
she/they	$\lambda\psi e\phi.\psi(sel_{she/they}\, e)e\phi$
smiles	$\lambda s.s(\lambda xe\phi.Smile(x) \wedge \phi e)$
kisses	$\lambda os.s(\lambda x.o(\lambda ye\phi.Kiss(x, y) \wedge \phi e))$

- Remarks
  - "::" adjoins **accessible variables** in the selection list

    $\iota \rightarrow \gamma \rightarrow \gamma$
  - "sel_{she}" selects the **correct variable** from the list

    $\gamma \rightarrow \iota$

Lexical Entries

Lexicon	**λ-Expression**
John/Mary	$\lambda\psi e\phi.\psi\mathbf{j}/\mathbf{m}(\mathbf{j}/\mathbf{m} :: e)\phi$
she/they	$\lambda\psi e\phi.\psi(sel_{she/they}\, e)e\phi$
smiles	$\lambda s.s(\lambda xe\phi.Smile(x) \wedge \phi e)$
kisses	$\lambda os.s(\lambda x.o(\lambda ye\phi.Kiss(x, y) \wedge \phi e))$

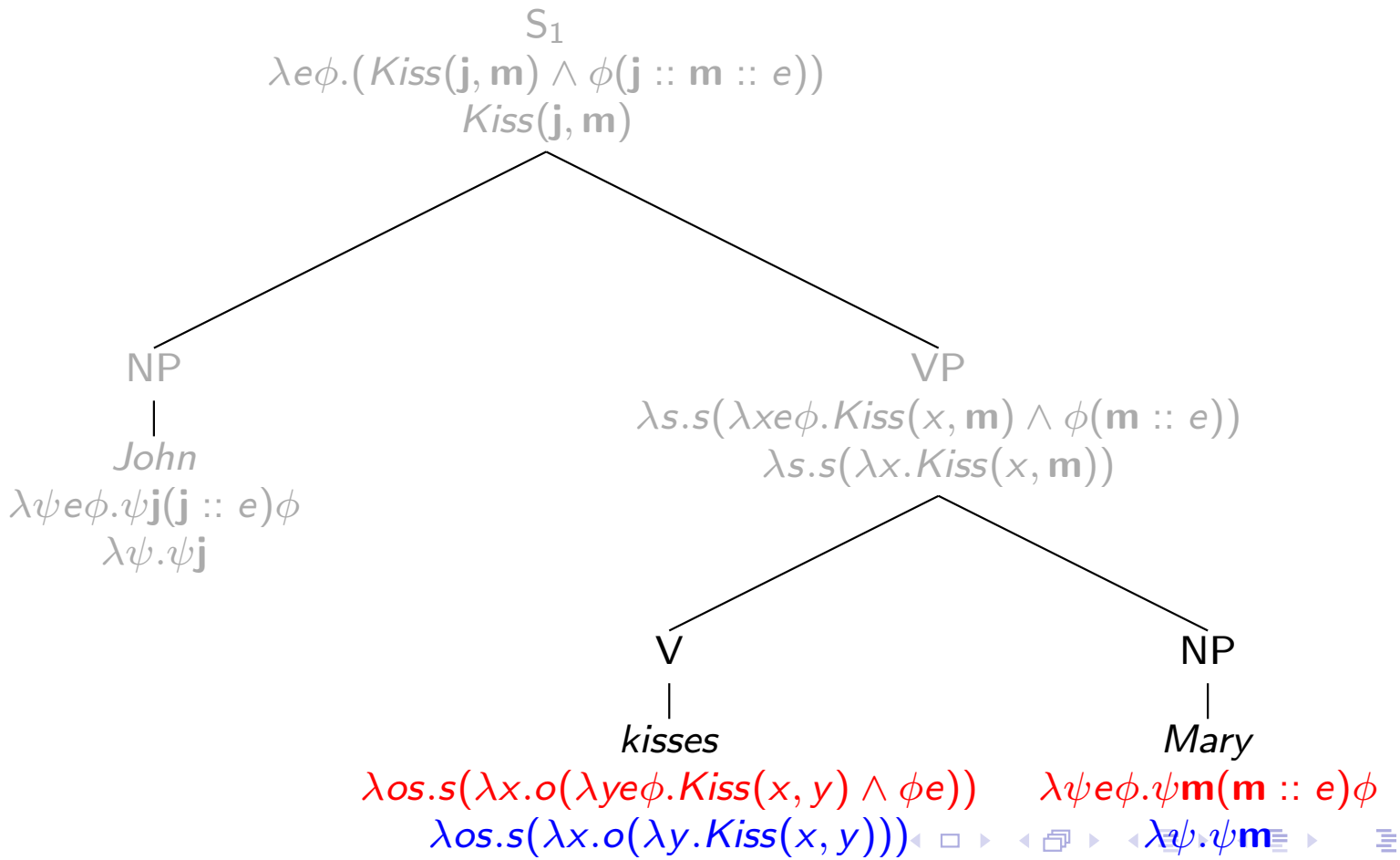
- Remarks
  - "::" adjoins **accessible variables** in the selection list

    $\iota \rightarrow \gamma \rightarrow \gamma$
  - "sel_{she}" selects the **correct variable** from the list

    $\gamma \rightarrow \iota$

Compositional Example

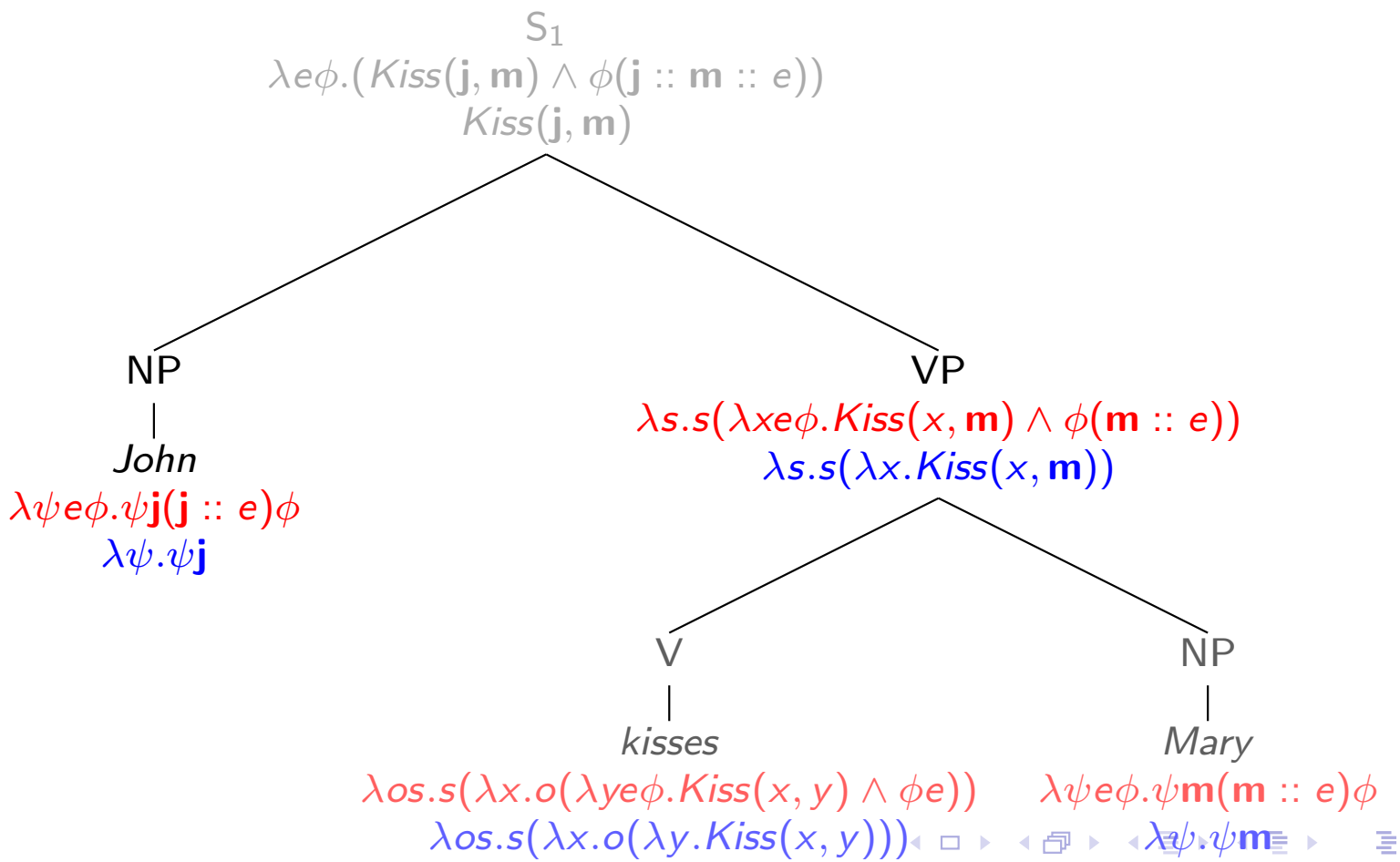
(6) John kisses Mary. She smiles.

1. 

Compositional Example

(6) John kisses Mary. She smiles.

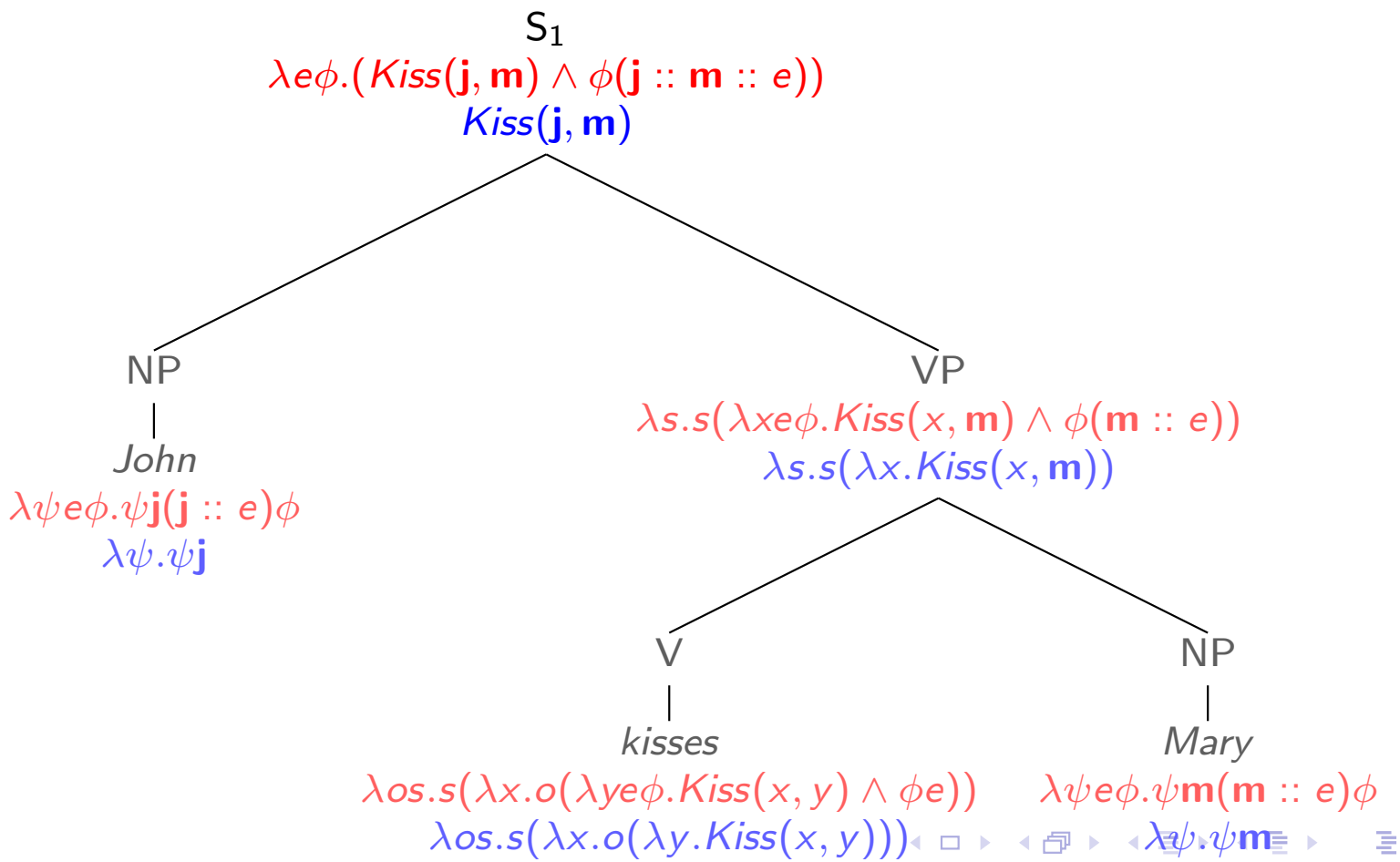
1



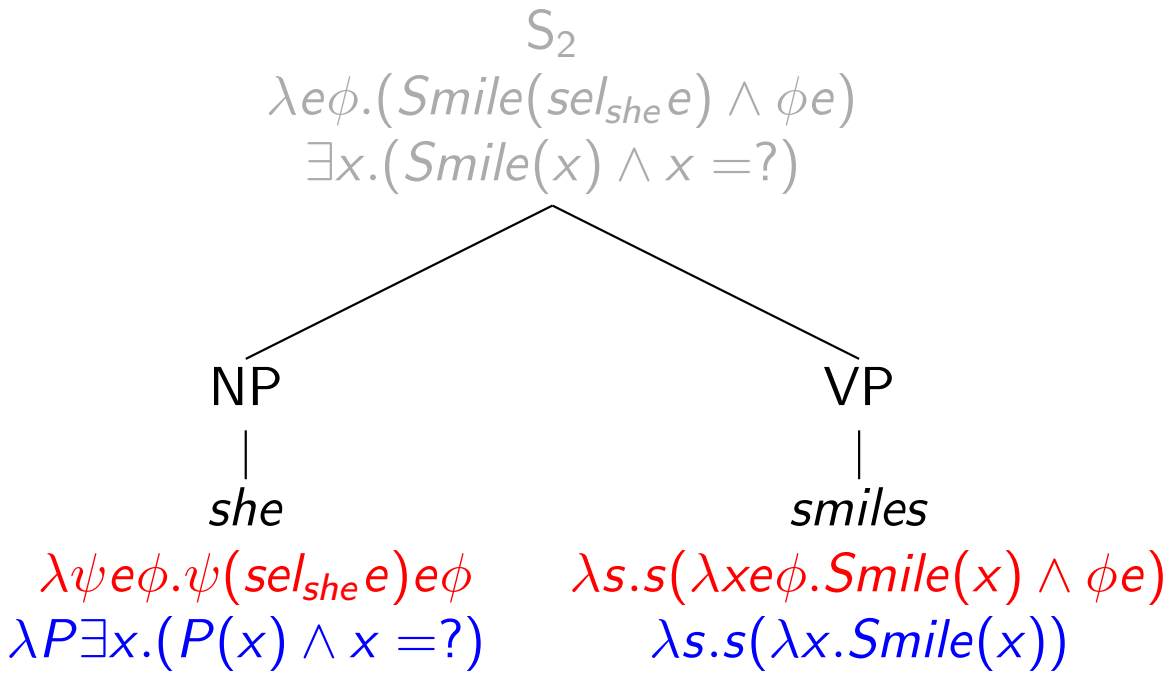
Compositional Example

(6) John kisses Mary. She smiles.

1

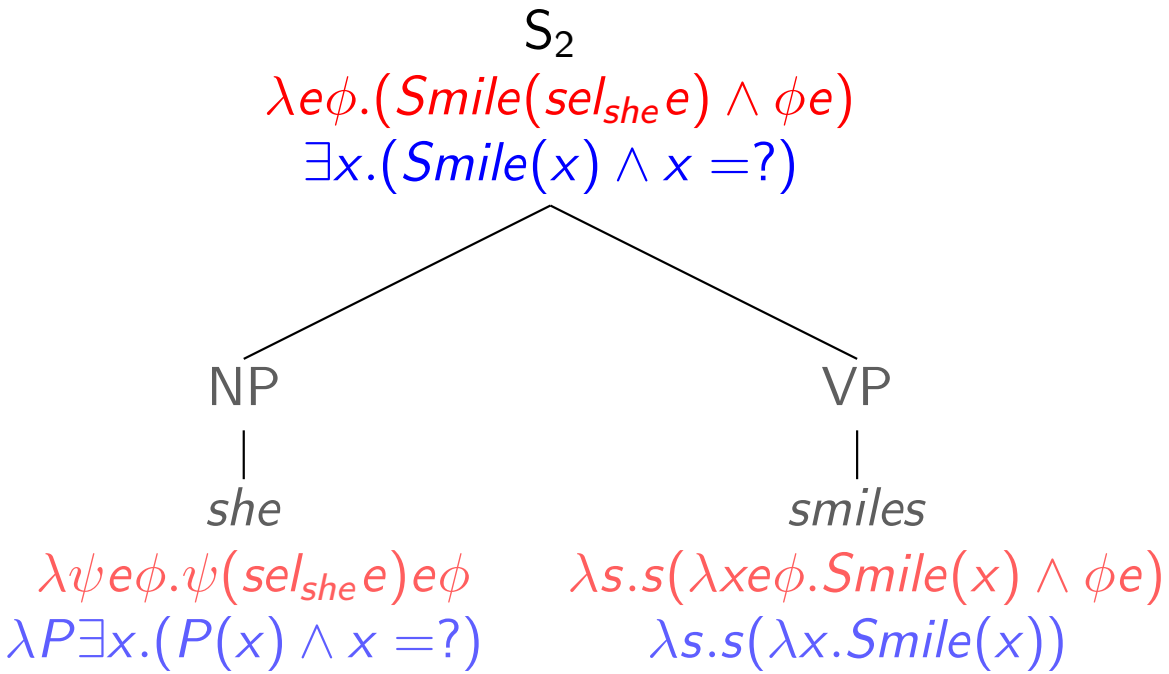


Compositional Example Continued

2. 

Compositional Example Continued

2



Compositional Example Continued

3. D

$\lambda e\phi.(Kiss(\mathbf{j},\mathbf{m}) \wedge Smile(sel_{she}(\mathbf{j} :: \mathbf{m} :: e)) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))$

???

$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$

$Kiss(\mathbf{j},\mathbf{m}) + \exists x.(Smile(x) \wedge x =?)???$

S_1

$\lambda e\phi.(Kiss(\mathbf{j},\mathbf{m}) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))$

$Kiss(\mathbf{j},\mathbf{m})$

S_2

$\lambda e\phi.(Smile(sel_{she}\, e) \wedge \phi e)$

$\exists x.(Smile(x) \wedge x =?)$

Compositional Example Continued

3. D

$\lambda e\phi.(Kiss(\mathbf{j},\mathbf{m}) \wedge Smile(sel_{she}(\mathbf{j} :: \mathbf{m} :: e)) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))$

???

$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$

$Kiss(\mathbf{j},\mathbf{m}) + \exists x.(Smile(x) \wedge x =?)???$

S_1

$\lambda e\phi.(Kiss(\mathbf{j},\mathbf{m}) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))$

$Kiss(\mathbf{j},\mathbf{m})$

S_2

$\lambda e\phi.(Smile(sel_{she}\, e) \wedge \phi e)$

$\exists x.(Smile(x) \wedge x =?)$

Compositional Example Continued

3. D

$\lambda e\phi.(Kiss(\mathbf{j},\mathbf{m}) \wedge Smile(sel_{she}(\mathbf{j} :: \mathbf{m} :: e)) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))$

???

$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$

$Kiss(\mathbf{j},\mathbf{m}) + \exists x.(Smile(x) \wedge x =?)???$

S_1

$\lambda e\phi.(Kiss(\mathbf{j},\mathbf{m}) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))$

$Kiss(\mathbf{j},\mathbf{m})$

S_2

$\lambda e\phi.(Smile(sel_{she}\, e) \wedge \phi e)$

$\exists x.(Smile(x) \wedge x =?)$

More Observations

- Recall: explicit group could be broken down to form other valid referents (singular or plural) - Example (4)
- Supposition: all sub-groups consisted of accessible referents can be potential antecedents

Example (Summation - More Observations)

(7) John was in Paris. Bill was in Rome. Mary was in Barcelona.

a. They would come back to work after the vacation.

b. They avoided the bad weather in France/Italy/Spain.

More Observations

- Recall: explicit group could be broken down to form other valid referents (singular or plural) - Example (4)
- Supposition: all sub-groups consisted of accessible referents can be potential antecedents

Example (Summation - More Observations)

(7) John was in Paris. Bill was in Rome. Mary was in Barcelona.

a. They would come back to work after the vacation.

b. They avoided the bad weather in France/Italy/Spain.

More Observations

- Recall: explicit group could be broken down to form other valid referents (singular or plural) - Example (4)
- Supposition: all sub-groups consisted of accessible referents can be potential antecedents

Example (Summation - More Observations)

(7) John was in Paris. Bill was in Rome. Mary was in Barcelona.

a. They would come back to work after the vacation.

b. They avoided the bad weather in France/Italy/Spain.

Generating All Sub-Groups

- Power Group
  - The set of all possible groups made up of any number of current accessible referents
  - A concept similar to power set in mathematics
- The summation function: $\mathfrak{Sum}$

Example (Performance of $\mathfrak{Sum}$)

- $\mathfrak{Sum}(j :: e) \Rightarrow (j :: e)$
- $\mathfrak{Sum}(m :: j :: e) \Rightarrow (m :: j :: j \oplus m :: e)$
- $\mathfrak{Sum}(b :: m :: j :: e) \Rightarrow (b :: m :: j :: b \oplus m :: b \oplus j :: m \oplus j :: b \oplus m \oplus j :: e)$
- ...

Generating All Sub-Groups

- Power Group
  - The set of all possible groups made up of any number of current accessible referents
  - A concept similar to power set in mathematics
- The summation function: $\mathfrak{Sum}$

Example (Performance of $\mathfrak{Sum}$)

- $\mathfrak{Sum}(j :: e) \Rightarrow (j :: e)$
- $\mathfrak{Sum}(m :: j :: e) \Rightarrow (m :: j :: j \oplus m :: e)$
- $\mathfrak{Sum}(b :: m :: j :: e) \Rightarrow (b :: m :: j :: b \oplus m :: b \oplus j :: m \oplus j :: b \oplus m \oplus j :: e)$
- ...

Generating All Sub-Groups

- Power Group
  - The set of all possible groups made up of any number of current accessible referents
  - A concept similar to power set in mathematics
- The summation function: $\mathfrak{Sum}$

Example (Performance of $\mathfrak{Sum}$)

- $\mathfrak{Sum}(j :: e) \Rightarrow (j :: e)$
- $\mathfrak{Sum}(m :: j :: e) \Rightarrow (m :: j :: j \oplus m :: e)$
- $\mathfrak{Sum}(b :: m :: j :: e) \Rightarrow (b :: m :: j :: b \oplus m :: b \oplus j :: m \oplus j :: b \oplus m \oplus j :: e)$
- ...

Generating All Sub-Groups

- Power Group
  - The set of all possible groups made up of any number of current accessible referents
  - A concept similar to power set in mathematics
- The summation function: $\mathfrak{Sum}$

Example (Performance of $\mathfrak{Sum}$)

- $\mathfrak{Sum}(j :: e) \Rightarrow (j :: e)$
- $\mathfrak{Sum}(m :: j :: e) \Rightarrow (m :: j :: j \oplus m :: e)$
- $\mathfrak{Sum}(b :: m :: j :: e) \Rightarrow (b :: m :: j :: b \oplus m :: b \oplus j :: m \oplus j :: b \oplus m \oplus j :: e)$
- ...

Two Supporting Functions

Definition (The Append Function $\mathfrak{App}$)

$\mathfrak{App}$ takes two lists l_1 and l_2 as arguments, $\mathfrak{App}(l_1, l_2)$ will be:

- l_2, if $l_1 = [\,]$ - the empty list;
- $head_1 :: \mathfrak{App}(tail_1, l_2)$, in which $head_1$ and $tail_1$ denote the head and the tail of l_1 respectively.

Definition (The Add Function $\mathfrak{Add}$)

$\mathfrak{Add}$ takes two arguments, an element a and a list l, $\mathfrak{Add}(a, l)$ will be:

- $[a]$ - list containing a single element a, if $l = [\,]$;
- $a \oplus head :: \mathfrak{Add}(a, tail)$, in which $head$ and $tail$ denote the head and tail of l respectively.

Two Supporting Functions

Definition (The Append Function $\mathfrak{App}$)

$\mathfrak{App}$ takes two lists l_1 and l_2 as arguments, $\mathfrak{App}(l_1, l_2)$ will be:

- l_2, if $l_1 = [\,]$ - the empty list;
- $head_1 :: \mathfrak{App}(tail_1, l_2)$, in which $head_1$ and $tail_1$ denote the head and the tail of l_1 respectively.

Definition (The Add Function $\mathfrak{Add}$)

$\mathfrak{Add}$ takes two arguments, an element a and a list l, $\mathfrak{Add}(a, l)$ will be:

- $[a]$ - list containing a single element a, if $l = [\,]$;
- $a \oplus head :: \mathfrak{Add}(a, tail)$, in which $head$ and $tail$ denote the head and tail of l respectively.

Formal Definition for $\mathfrak{Sum}$

Definition (The Summation Function $\mathfrak{Sum}$)

$\mathfrak{Sum}$ takes a list l as argument, $\mathfrak{Sum}(l)$ will be:

- [ ] - the empty list, if $l = [\,]$;
- $\mathfrak{App}(\mathfrak{Add}(head, sum\_tail), sum\_tail)$, in which *head* denotes the head of l, *sum_tail* denotes the result of $\mathfrak{Sum}(tail)$ where *tail* denotes the tail of l.

- Remarks
  - $\mathfrak{Sum}$ differs from classical power set by replacing union operation with group formation operation "$\oplus$"

Formal Definition for $\mathfrak{Sum}$

Definition (The Summation Function $\mathfrak{Sum}$)

$\mathfrak{Sum}$ takes a list l as argument, $\mathfrak{Sum}(l)$ will be:

- [ ] - the empty list, if $l = [\,]$;
- $\mathfrak{App}(\mathfrak{Add}(head, sum\_tail), sum\_tail)$, in which *head* denotes the head of l, *sum_tail* denotes the result of $\mathfrak{Sum}(tail)$ where *tail* denotes the tail of l.

- Remarks
  - $\mathfrak{Sum}$ differs from classical power set by replacing union operation with group formation operation "$\oplus$"

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a, b, c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b, c])), \mathfrak{Sum}([b, c])$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a, b, c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b, c])), \mathfrak{Sum}([b, c])$$

$$\downarrow$$

$$\mathfrak{Sum}([b, c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$

Sum Illustration Step-by-Step

$$\mathfrak{Sum}([a, b, c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b, c])), \mathfrak{Sum}([b, c])$$

$$\downarrow$$

$$\mathfrak{Sum}([b, c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$

$$\downarrow$$

$$\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a, b, c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b, c])), \mathfrak{Sum}([b, c])$$

$$\downarrow$$

$$\mathfrak{Sum}([b, c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$

$$\downarrow$$

$$\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))$$

$$\downarrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\boxed{\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])}$$

$$\downarrow$$

$$\boxed{\mathfrak{Sum}([b,c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))}$$

$$\downarrow$$

$$\boxed{\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))}$$

$$\uparrow$$

$$\boxed{\mathfrak{Sum}([\,]) = [\,]}$$

Sum Illustration Step-by-Step

$$\mathfrak{Sum}([a, b, c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b, c])), \mathfrak{Sum}([b, c])$$

$$\downarrow$$

$$\mathfrak{Sum}([b, c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$

$$\downarrow$$

$$\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))$$
$$= \mathfrak{App}(\mathfrak{Add}(c, [\,]), [\,]) = \mathfrak{App}([c], [\,])$$

$$\uparrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

Sum Illustration Step-by-Step

$$\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])$$

$$\downarrow$$

$$\mathfrak{Sum}([b,c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$

$$\downarrow$$

$$\begin{aligned}&\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))\\ &= \mathfrak{App}(\mathfrak{Add}(c, [\,]), [\,]) = \mathfrak{App}([c], [\,])\\ &= [c]\end{aligned}$$

$$\uparrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

Sum Illustration Step-by-Step

$$\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])$$

$$\downarrow$$

$$\mathfrak{Sum}([b,c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$

$$\uparrow$$

$$\begin{aligned}&\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))\\ &= \mathfrak{App}(\mathfrak{Add}(c, [\,]), [\,]) = \mathfrak{App}([c], [\,])\\ &= [c]\end{aligned}$$

$$\uparrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

Sum Illustration Step-by-Step

$$\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])$$

$$\downarrow$$

$$\mathfrak{Sum}([b,c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$
$$= \mathfrak{App}(\mathfrak{Add}(b,[c]),[c]) = \mathfrak{App}([b \oplus c, b],[c])$$

$$\uparrow$$

$$\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))$$
$$= \mathfrak{App}(\mathfrak{Add}(c,[\,]),[\,]) = \mathfrak{App}([c],[\,])$$
$$= [c]$$

$$\uparrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])$$

$$\downarrow$$

$$\begin{aligned}\mathfrak{Sum}([b,c]) &= \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c])) \\ &= \mathfrak{App}(\mathfrak{Add}(b,[c]),[c]) = \mathfrak{App}([b \oplus c, b],[c]) \\ &= [b \oplus c, b, c]\end{aligned}$$

$$\uparrow$$

$$\begin{aligned}\mathfrak{Sum}([c]) &= \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,])) \\ &= \mathfrak{App}(\mathfrak{Add}(c,[\,]),[\,]) = \mathfrak{App}([c],[\,]) \\ &= [c]\end{aligned}$$

$$\uparrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a, b, c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b, c])), \mathfrak{Sum}([b, c])$$

$$\uparrow$$

$$\begin{aligned}\mathfrak{Sum}([b, c]) &= \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c])) \\ &= \mathfrak{App}(\mathfrak{Add}(b, [c]), [c]) = \mathfrak{App}([b \oplus c, b], [c]) \\ &= [b \oplus c, b, c]\end{aligned}$$

$$\uparrow$$

$$\begin{aligned}\mathfrak{Sum}([c]) &= \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,])) \\ &= \mathfrak{App}(\mathfrak{Add}(c, [\,]), [\,]) = \mathfrak{App}([c], [\,]) \\ &= [c]\end{aligned}$$

$$\uparrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])$$
$$= \mathfrak{App}(\mathfrak{Add}(a, [b \oplus c, b, c]), [b \oplus c, b, c])$$

$$\mathfrak{Sum}([b,c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$
$$= \mathfrak{App}(\mathfrak{Add}(b, [c]), [c]) = \mathfrak{App}([b \oplus c, b], [c])$$
$$= [b \oplus c, b, c]$$

$$\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))$$
$$= \mathfrak{App}(\mathfrak{Add}(c, [\,]), [\,]) = \mathfrak{App}([c], [\,])$$
$$= [c]$$

$$\mathfrak{Sum}([\,]) = [\,]$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])$$
$$= \mathfrak{App}(\mathfrak{Add}(a, [b \oplus c, b, c]), [b \oplus c, b, c])$$
$$= \mathfrak{App}(([a \oplus b \oplus c, a \oplus b, a \oplus c, a]), [b \oplus c, b, c])$$

$$\uparrow$$

$$\mathfrak{Sum}([b,c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$
$$= \mathfrak{App}(\mathfrak{Add}(b, [c]), [c]) = \mathfrak{App}([b \oplus c, b], [c])$$
$$= [b \oplus c, b, c]$$

$$\uparrow$$

$$\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))$$
$$= \mathfrak{App}(\mathfrak{Add}(c, [\,]), [\,]) = \mathfrak{App}([c], [\,])$$
$$= [c]$$

$$\uparrow$$

$$\mathfrak{Sum}([\,]) = [\,]$$

$\mathfrak{Sum}$ Illustration Step-by-Step

$$\mathfrak{Sum}([a,b,c]) = \mathfrak{App}(\mathfrak{Add}(a, \mathfrak{Sum}([b,c])), \mathfrak{Sum}([b,c])$$
$$= \mathfrak{App}(\mathfrak{Add}(a, [b \oplus c, b, c]), [b \oplus c, b, c])$$
$$= \mathfrak{App}(([a \oplus b \oplus c, a \oplus b, a \oplus c, a]), [b \oplus c, b, c])$$
$$= [a \oplus b \oplus c, a \oplus b, a \oplus c, a, b \oplus c, b, c]$$

$\uparrow$

$$\mathfrak{Sum}([b,c]) = \mathfrak{App}(\mathfrak{Add}(b, \mathfrak{Sum}([c])), \mathfrak{Sum}([c]))$$
$$= \mathfrak{App}(\mathfrak{Add}(b, [c]), [c]) = \mathfrak{App}([b \oplus c, b], [c])$$
$$= [b \oplus c, b, c]$$

$\uparrow$

$$\mathfrak{Sum}([c]) = \mathfrak{App}(\mathfrak{Add}(c, \mathfrak{Sum}([\,])), \mathfrak{Sum}([\,]))$$
$$= \mathfrak{App}(\mathfrak{Add}(c, [\,]), [\,]) = \mathfrak{App}([c], [\,])$$
$$= [c]$$

$\uparrow$

$$\mathfrak{Sum}([\,]) = [\,]$$

$\mathfrak{Sum}$ in Real Practice

Example (Natural Language Example for $\mathfrak{Sum}$)

(8) a. John and Mary went to Paris.
b. Bill and Lucy went to Rome.

- Necessary Lexical Entries
  - Proper Names
    - $[\![John]\!] = \lambda\psi e\phi.\psi \mathbf{j} \mathfrak{Sum}(\mathbf{j} :: e)\phi$
  - Conjunction "*and*"
    1. $[\![and]\!]_{dis} = \lambda AB\psi e\phi.A\psi e(\lambda e'.B\psi e'\phi)$
    2. $[\![and]\!]_{coll} = \lambda AB\psi e\phi.A(\lambda x.B(\lambda y.\psi(x \oplus y)))e\phi$

$\mathfrak{Sum}$ in Real Practice

Example (Natural Language Example for $\mathfrak{Sum}$)

(8) a. John and Mary went to Paris.
b. Bill and Lucy went to Rome.

- Necessary Lexical Entries
  - Proper Names
    - $[\![John]\!] = \lambda\psi e\phi.\psi \mathbf{j} \mathfrak{Sum}(\mathbf{j} :: e)\phi$
  - Conjunction "*and*"
    1. $[\![and]\!]_{dis} = \lambda AB\psi e\phi.A\psi e(\lambda e'.B\psi e'\phi)$
    2. $[\![and]\!]_{coll} = \lambda AB\psi e\phi.A(\lambda x.B(\lambda y.\psi(x \oplus y)))e\phi$

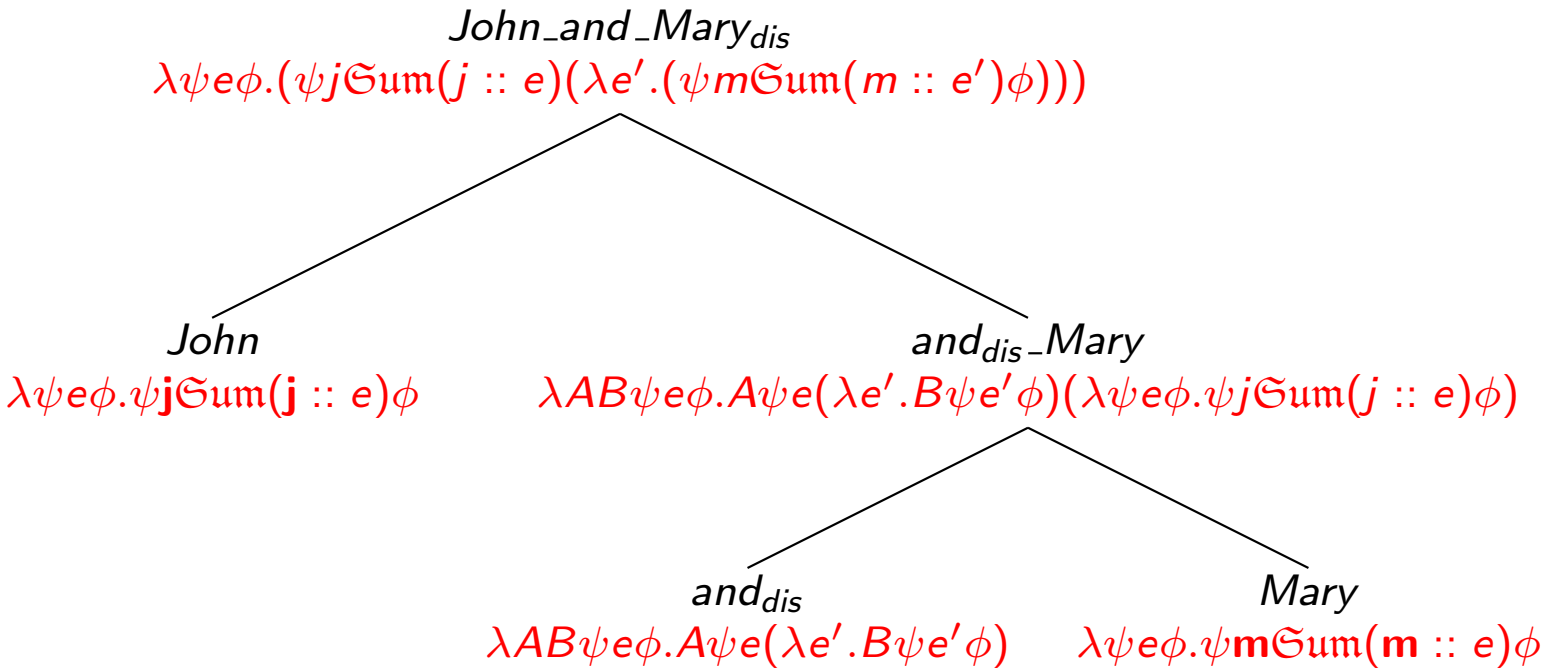
$\mathfrak{Sum}$ in Real Practice

Example (Natural Language Example for $\mathfrak{Sum}$)

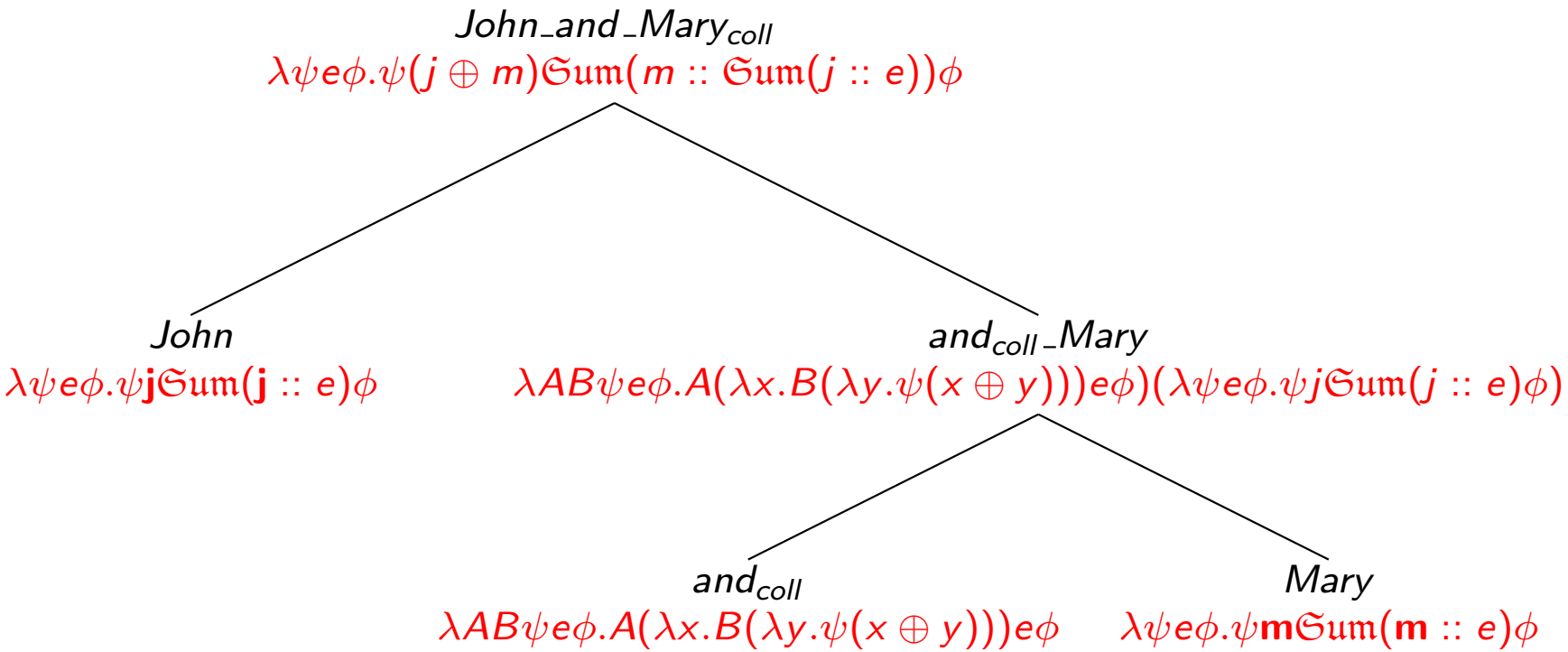
(8) a. John and Mary went to Paris.
b. Bill and Lucy went to Rome.

- Necessary Lexical Entries
  - Proper Names
    - $[\![John]\!] = \lambda\psi e\phi.\psi \mathbf{j}\mathfrak{Sum}(\mathbf{j} :: e)\phi$
  - Conjunction "*and*"
    1. $[\![and]\!]_{dis} = \lambda AB\psi e\phi.A\psi e(\lambda e'.B\psi e'\phi)$
    2. $[\![and]\!]_{coll} = \lambda AB\psi e\phi.A(\lambda x.B(\lambda y.\psi(x \oplus y)))e\phi$

Distributive "and"



Collective "and"

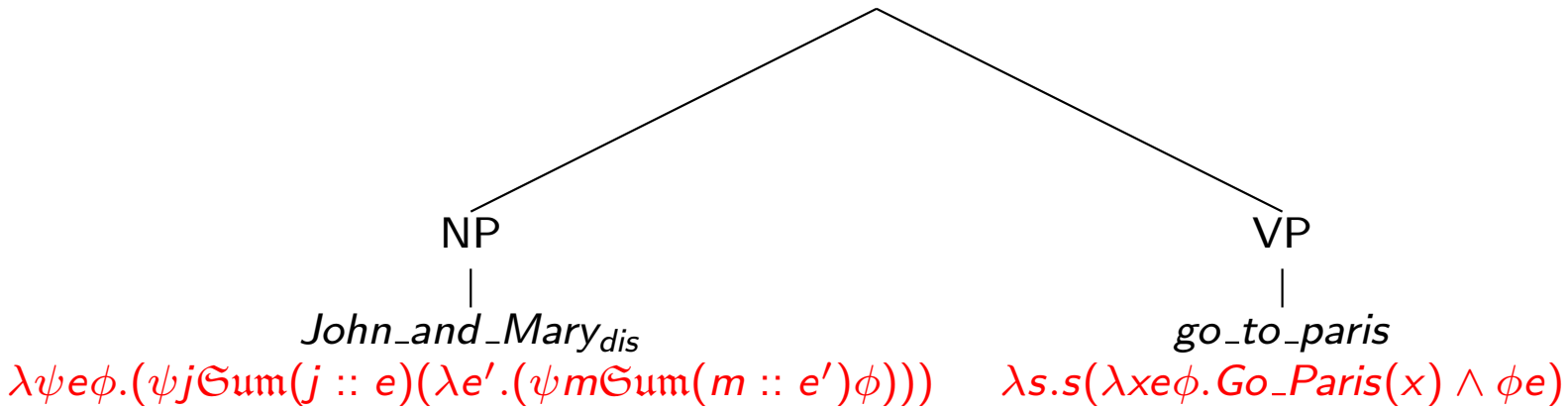


$\mathfrak{Sum}$ in Real Practice Continued

(8-a)

$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge \phi(j :: m :: j \oplus m :: e))$

$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge \phi(\mathfrak{Sum}(j :: \mathfrak{Sum}(m :: e))))$

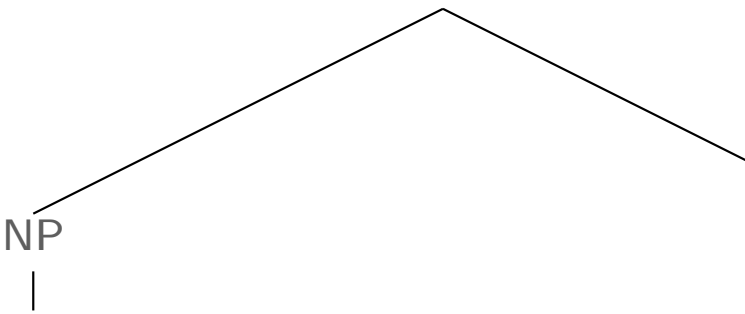


$\mathfrak{Sum}$ in Real Practice Continued

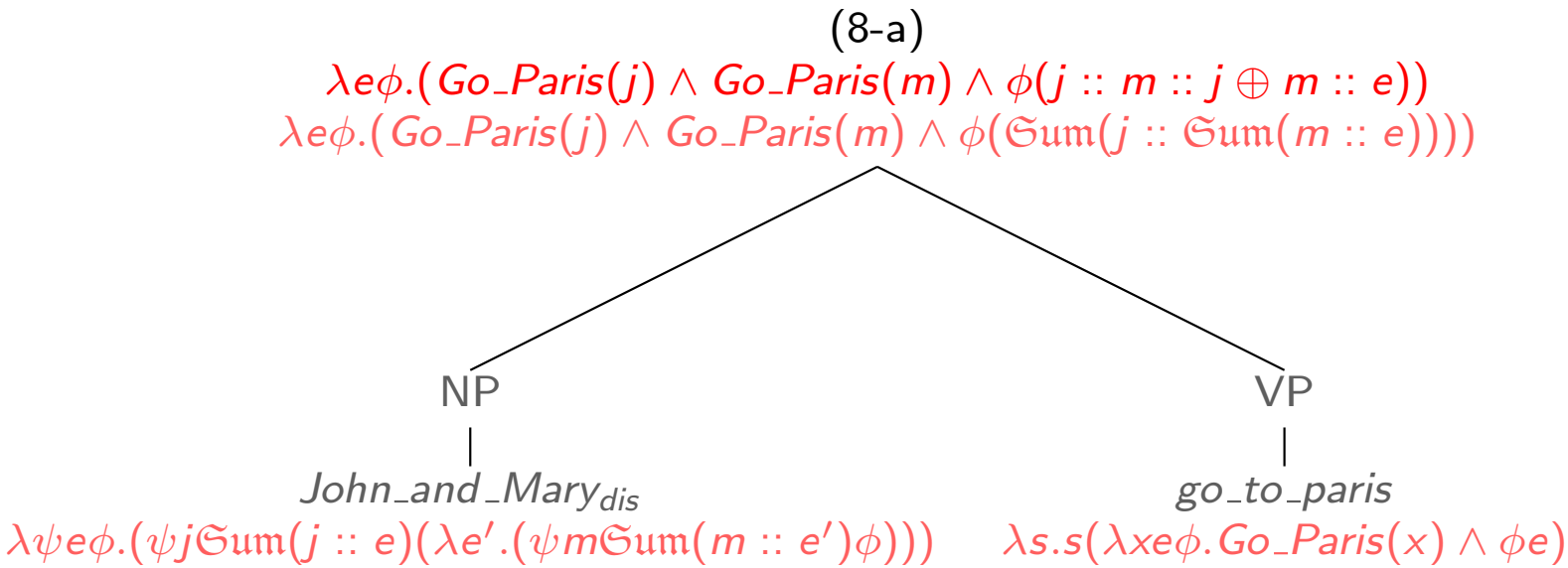
(8-a)

$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge \phi(j :: m :: j \oplus m :: e))$

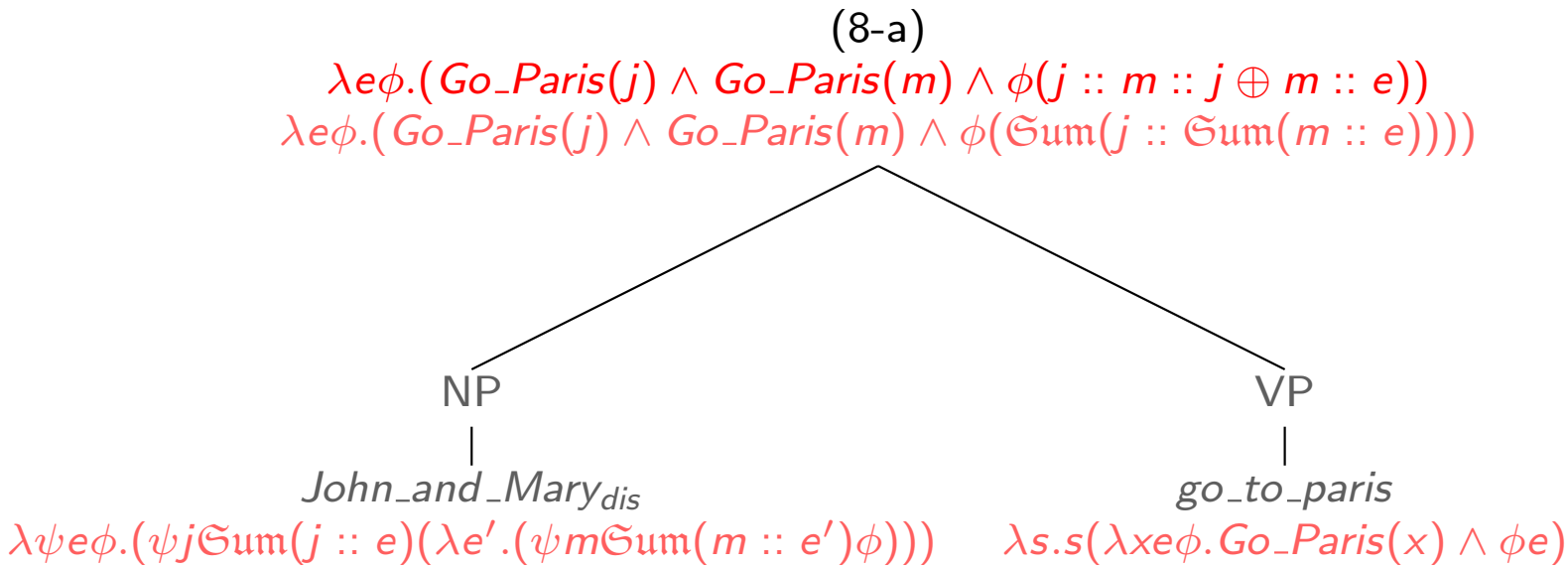
$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge \phi(\mathfrak{Sum}(j :: \mathfrak{Sum}(m :: e))))$



$\mathfrak{Sum}$ in Real Practice Continued



$\mathfrak{Sum}$ in Real Practice Continued



Similar for (8-b)

$\mathfrak{Sum}$ in Real Practice Continued

(8)

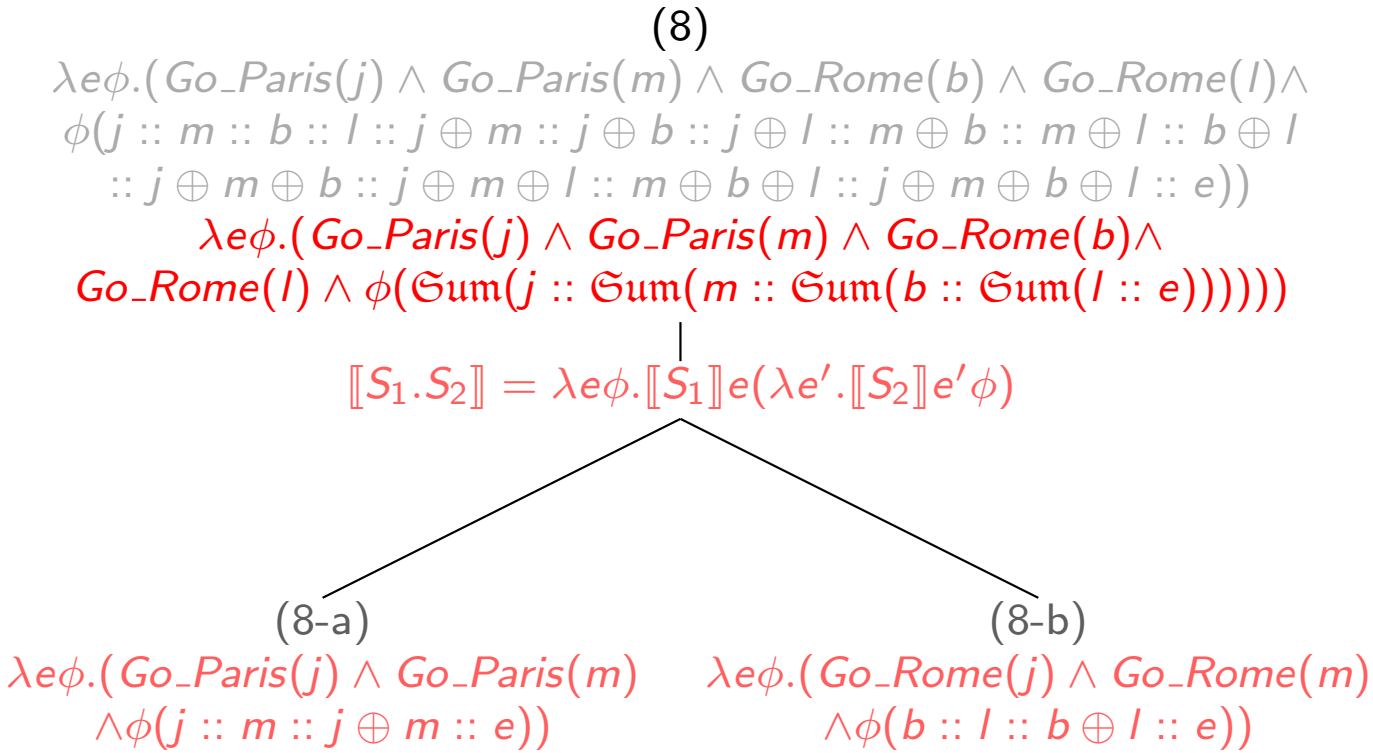
$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge Go\_Rome(b) \wedge Go\_Rome(l) \wedge$
$\phi(j :: m :: b :: l :: j \oplus m :: j \oplus b :: j \oplus l :: m \oplus b :: m \oplus l :: b \oplus l$
$:: j \oplus m \oplus b :: j \oplus m \oplus l :: m \oplus b \oplus l :: j \oplus m \oplus b \oplus l :: e))$
$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge Go\_Rome(b) \wedge$
$Go\_Rome(l) \wedge \phi(\mathfrak{Sum}(j :: \mathfrak{Sum}(m :: \mathfrak{Sum}(b :: \mathfrak{Sum}(l :: e))))))$

$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$

(8-a)
$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m)$
$\wedge\phi(j :: m :: j \oplus m :: e))$

(8-b)
$\lambda e\phi.(Go\_Rome(j) \wedge Go\_Rome(m)$
$\wedge\phi(b :: l :: b \oplus l :: e))$

Sum in Real Practice Continued



$\mathfrak{Sum}$ in Real Practice Continued

(8)

$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge Go\_Rome(b) \wedge Go\_Rome(l) \wedge$
$\phi(j :: m :: b :: l :: j \oplus m :: j \oplus b :: j \oplus l :: m \oplus b :: m \oplus l :: b \oplus l$
$:: j \oplus m \oplus b :: j \oplus m \oplus l :: m \oplus b \oplus l :: j \oplus m \oplus b \oplus l :: e))$
$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m) \wedge Go\_Rome(b) \wedge$
$Go\_Rome(l) \wedge \phi(\mathfrak{Sum}(j :: \mathfrak{Sum}(m :: \mathfrak{Sum}(b :: \mathfrak{Sum}(l :: e))))))$

$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$

(8-a)
$\lambda e\phi.(Go\_Paris(j) \wedge Go\_Paris(m)$
$\wedge\phi(j :: m :: j \oplus m :: e))$

(8-b)
$\lambda e\phi.(Go\_Rome(j) \wedge Go\_Rome(m)$
$\wedge\phi(b :: l :: b \oplus l :: e))$

More Observations

Example (Abstraction - More Observations)

(9) Two of five students went to school.

a. They worked hard.

b. They had to hand in the homework by tomorrow.

- QNP: *Generalized Quantifier* + *Noun*
- More than one potential group referents are introduced by the same NP

More Observations

Example (Abstraction - More Observations)

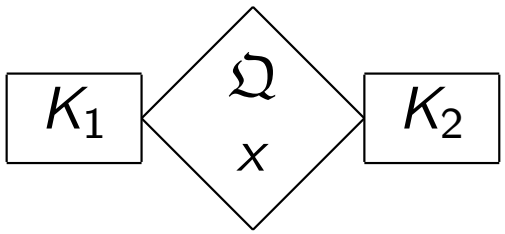
(9) Two of five students went to school.

a. They worked hard.

b. They had to hand in the homework by tomorrow.

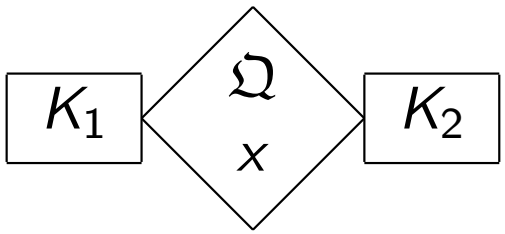
- QNP: *Generalized Quantifier* + *Noun*
- More than one potential group referents are introduced by the same NP

Abstraction in DRT [Kamp and Reyle, 1993]

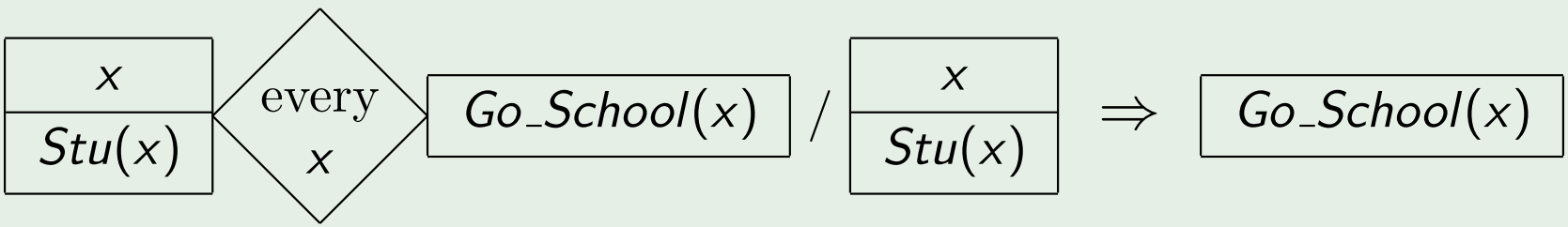
- Duplex Condition: the relation between two sets, which is constrained by the property of QNP
- 

Example (Duplex Condition)

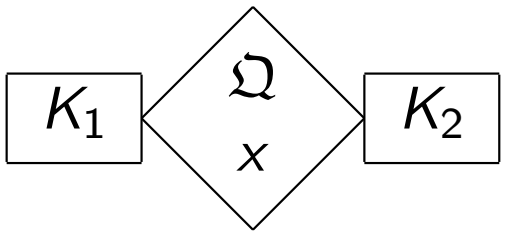
Abstraction in DRT [Kamp and Reyle, 1993]

- Duplex Condition: the relation between two sets, which is constrained by the property of QNP
- 

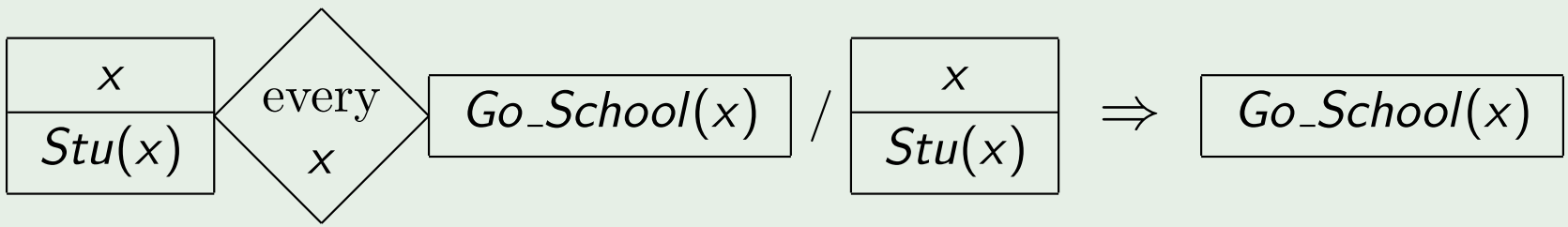
Example (Duplex Condition)

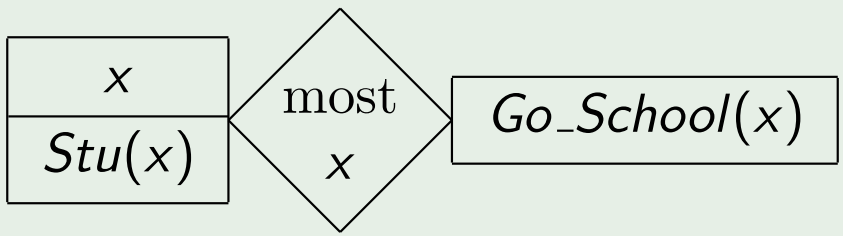


Abstraction in DRT [Kamp and Reyle, 1993]

- Duplex Condition: the relation between two sets, which is constrained by the property of QNP
- 

Example (Duplex Condition)





Three Groups

- Maximum Group
- Reference Group / Refset Anaphora
- Complement Group / Compset Anaphora

Figure: Structure Denoted by Generalized Quantifiers

Three Groups

- Maximum Group
- Reference Group / Refset Anaphora
- Complement Group / Compset Anaphora

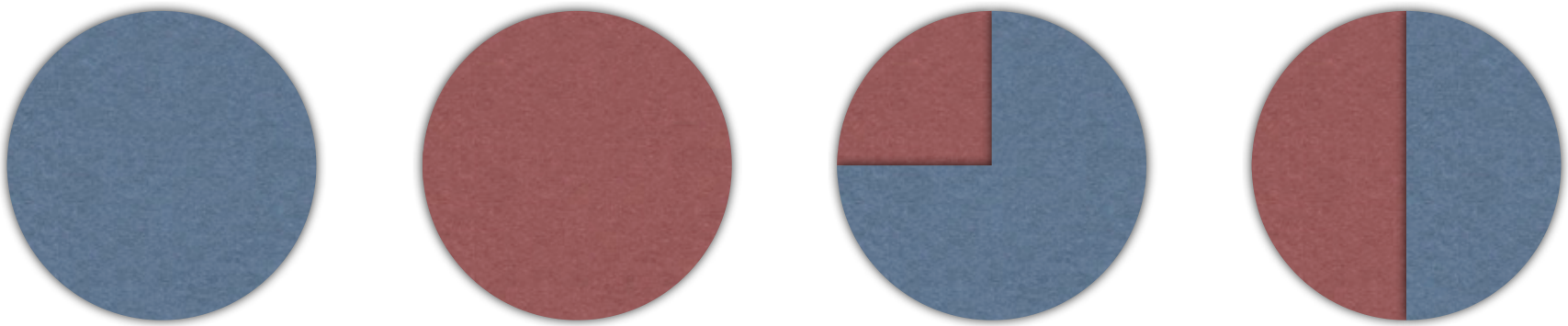


Figure: Structure Denoted by Generalized Quantifiers

Unveiling All Groups

- Proposition: to unveil all potential groups formed from abstraction
- Lexical Entry

Generalized Quantifier

$[\![GQ]\!] = \lambda\psi AB e\phi.Quan(\psi)x.((Axe\lambda e.\top)Rel(\psi)(Bxe\lambda e.\top)) \wedge \phi((\mathfrak{Abs}(\psi, x) :: e)$

- "*Quan*()" and "*Rel*()" are quantifier-sensitive
  - $Quan(every) = \forall$, $Quan(a) = \exists$
  - $Rel(every) = \rightarrow$, $Rel(a) = \wedge$

Unveiling All Groups

- Proposition: to unveil all potential groups formed from abstraction
- Lexical Entry

Generalized Quantifier

$[\![GQ]\!] = \lambda\psi AB e\phi.Quan(\psi)x.((Axe\lambda e.\top)Rel(\psi)(Bxe\lambda e.\top)) \wedge \phi((\mathfrak{Abs}(\psi, x) :: e)$

- "*Quan*()" and "*Rel*()" are quantifier-sensitive
  - $Quan(every) = \forall$, $Quan(a) = \exists$
  - $Rel(every) = \rightarrow$, $Rel(a) = \wedge$

Unveiling All Groups

- Proposition: to unveil all potential groups formed from abstraction
- Lexical Entry

Generalized Quantifier

$[\![GQ]\!] = \lambda\psi AB e\phi.Quan(\psi)x.((Axe\lambda e.\top)Rel(\psi)(Bxe\lambda e.\top)) \wedge \phi((\mathfrak{Abs}(\psi, x) :: e)$

- "*Quan*()" and "*Rel*()" are quantifier-sensitive
  - $Quan(every) = \forall$, $Quan(a) = \exists$
  - $Rel(every) = \rightarrow$, $Rel(a) = \wedge$

Formal Definition for $\mathfrak{Abs}$

Definition (The Abstraction Function $\mathfrak{Abs}$)

$\mathfrak{Abs}$ takes two arguments: a generalized quantifier q and the related individual variable x. The output, namely $\mathfrak{Abs}(q, x)$ will be a left context consisting of two group referents R_i and C_i:

- R: the reference group of individuals denoted by the quantifier;
- C: the complement group of individuals denoted by the quantifier;
- i: the index that signifies the dependency of the two groups.

Example (Entry for "every")

$$[\![every]\!] = [\![GQ]\!](every)$$
$$\Rightarrow \lambda ABe\phi.Quan(every)x.(Axe\lambda e.\top Rel(every)Bxe\lambda e.\top) \wedge \phi(\mathfrak{Abs}(every, x) :: e)$$
$$\Rightarrow \lambda ABe\phi.\forall x.(Axe\lambda e.\top \rightarrow Bxe\lambda e.\top) \wedge \phi(\mathfrak{Abs}(every, x) :: e)$$

Formal Definition for $\mathfrak{Abs}$

Definition (The Abstraction Function $\mathfrak{Abs}$)

$\mathfrak{Abs}$ takes two arguments: a generalized quantifier q and the related individual variable x. The output, namely $\mathfrak{Abs}(q, x)$ will be a left context consisting of two group referents R_i and C_i:

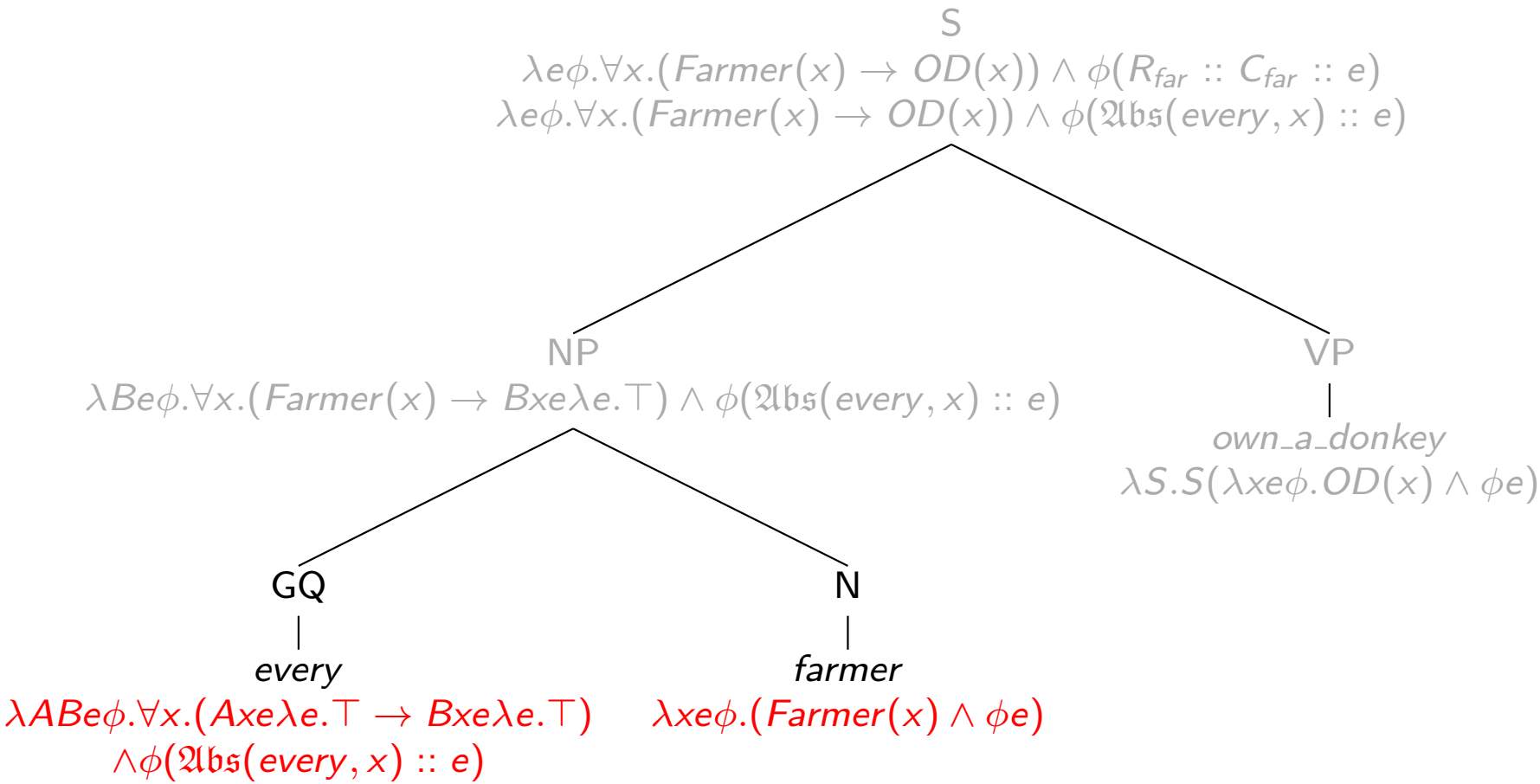
- R: the reference group of individuals denoted by the quantifier;
- C: the complement group of individuals denoted by the quantifier;
- i: the index that signifies the dependency of the two groups.

Example (Entry for "every")

$[\![every]\!] = [\![GQ]\!](every)$
$\Rightarrow \lambda ABe\phi.Quan(every)x.(Axe\lambda e.\top Rel(every)Bxe\lambda e.\top) \wedge$
$\phi(\mathfrak{Abs}(every, x) :: e)$
$\Rightarrow \lambda ABe\phi.\forall x.(Axe\lambda e.\top \rightarrow Bxe\lambda e.\top) \wedge \phi(\mathfrak{Abs}(every, x) :: e)$

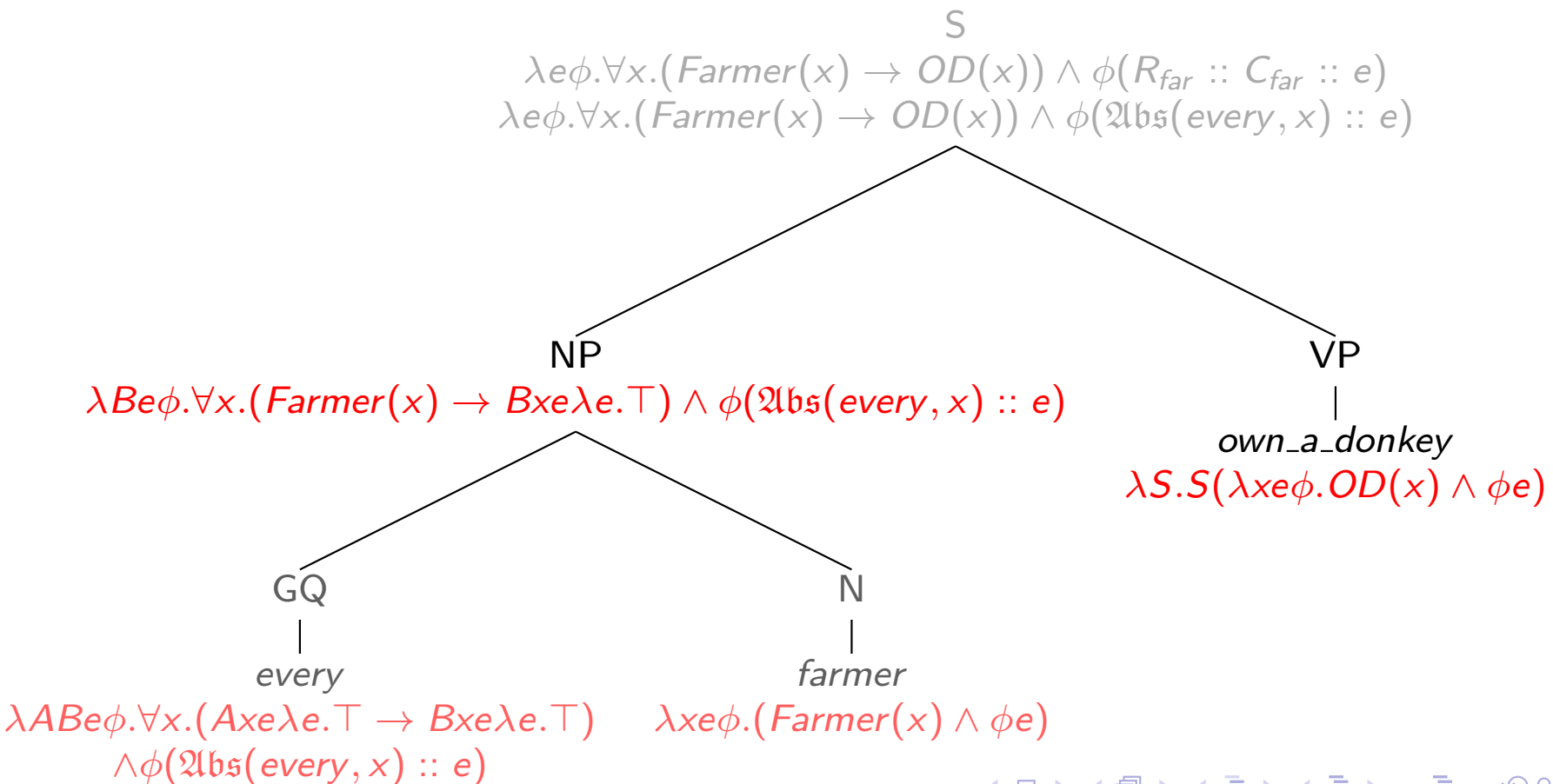
$\mathfrak{Abs}$ in Real Practice

(10) Every farmer owns a donkey.



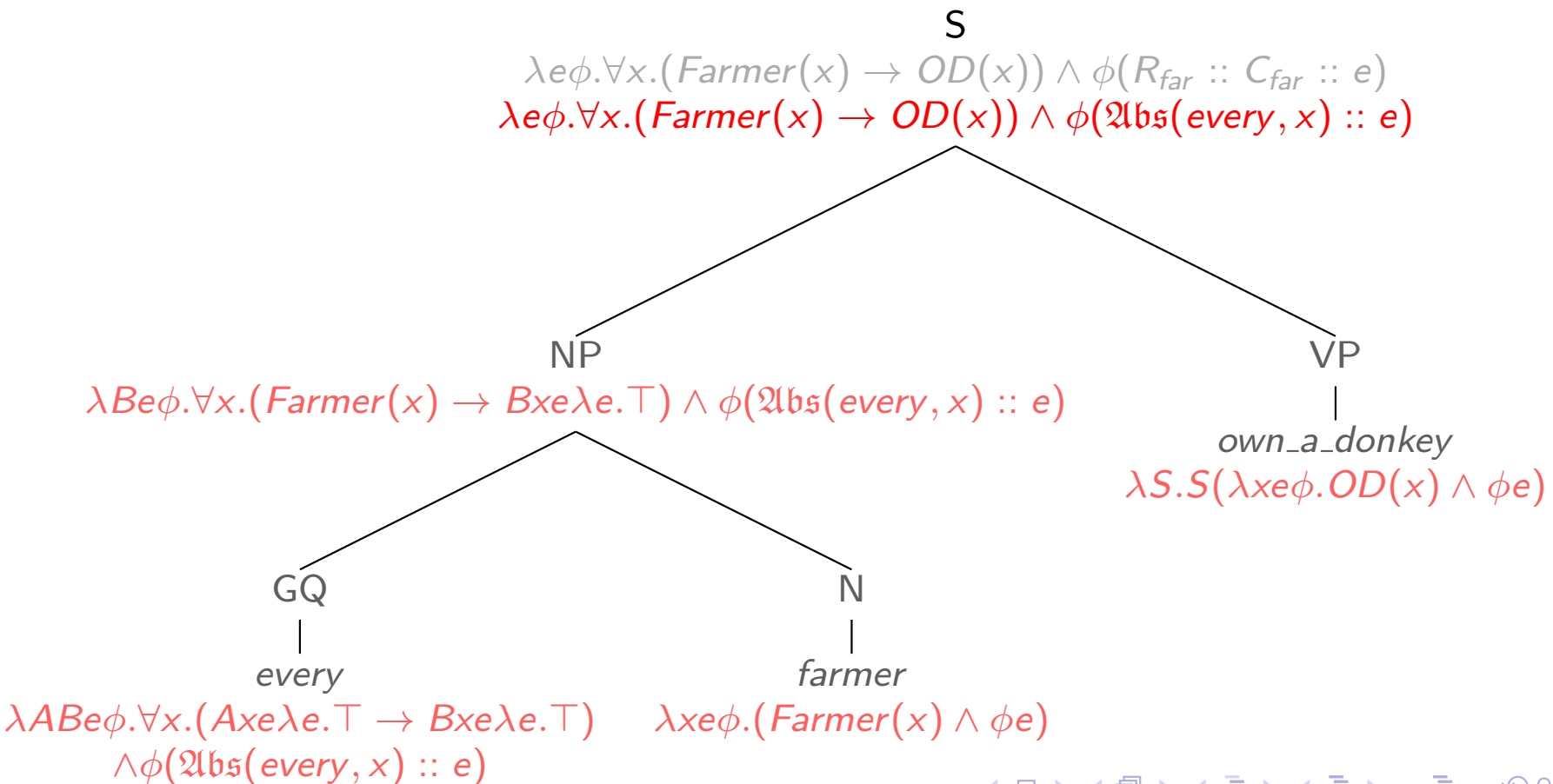
$\mathfrak{Abs}$ in Real Practice

(10) Every farmer owns a donkey.



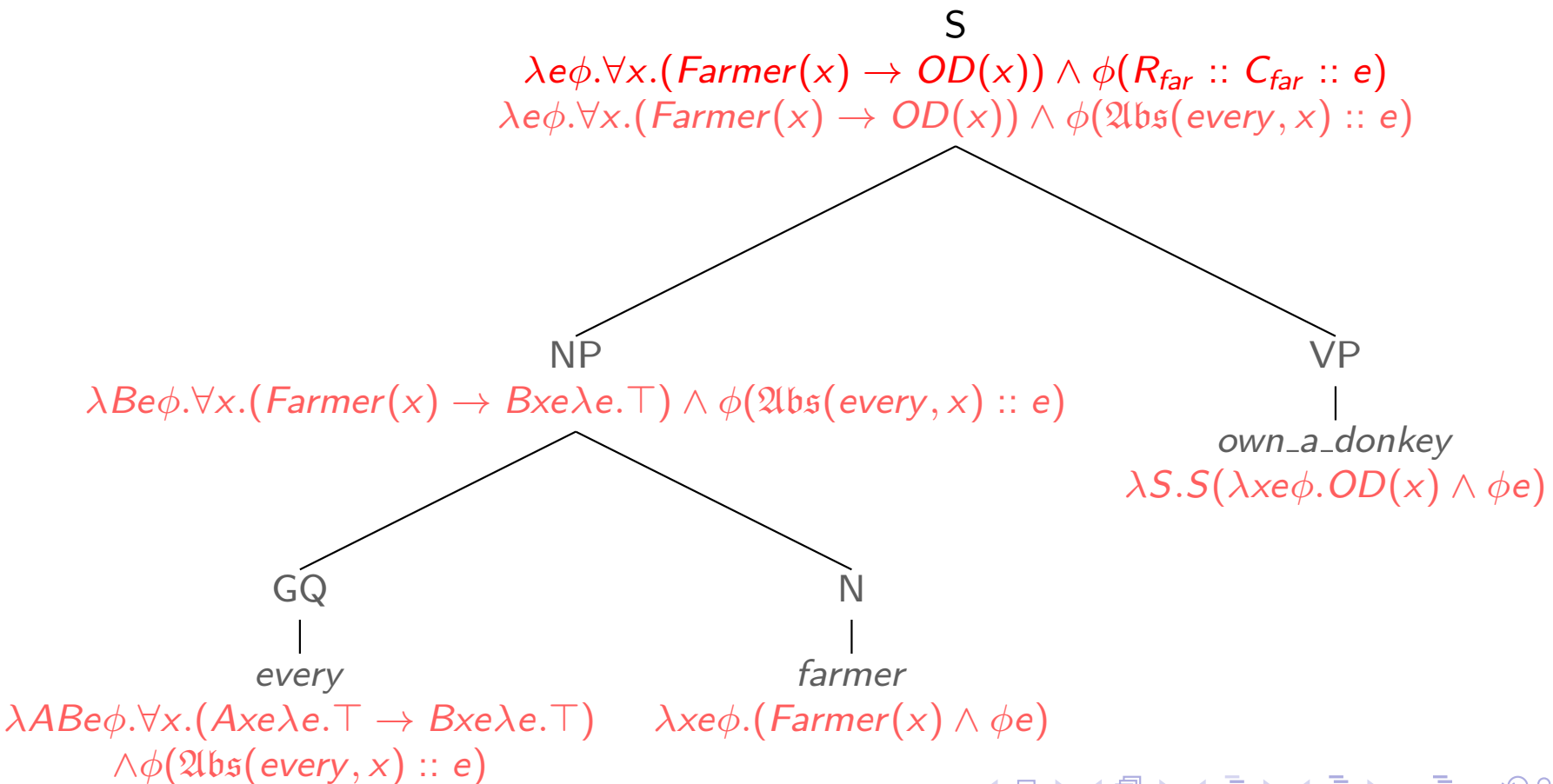
$\mathfrak{Abs}$ in Real Practice

(10) Every farmer owns a donkey.



$\mathfrak{Abs}$ in Real Practice

(10) Every farmer owns a donkey.



Summary

- Conclusion
  - Investigating plural anaphora within a new dynamic semantic framework
  - A potential list containing accessible plural referents is provided for summation and abstraction respectively
  - The framework is sound on the aspect of compositionality
  - The proposal is not responsible for the complete task of anaphora resolution
- Future Work
  - More elaborate definition on $\mathfrak{Sum}$ and $\mathfrak{Abs}$
  - Concern of over generation
  - Taking rhetorical structure into consideration
  - Combining with event semantics

Summary

- Conclusion
  - Investigating plural anaphora within a new dynamic semantic framework
  - A potential list containing accessible plural referents is provided for summation and abstraction respectively
  - The framework is sound on the aspect of compositionality
  - The proposal is not responsible for the complete task of anaphora resolution
- Future Work
  - More elaborate definition on $\mathfrak{Sum}$ and $\mathfrak{Abs}$
  - Concern of over generation
  - Taking rhetorical structure into consideration
  - Combining with event semantics

References


Asher, N. and Pogodalla, S. (2011).
Sdrt and continuation semantics.
New Frontiers in Artificial Intelligence, pages 3–15.


de Groote, P. (2006).
Towards a montagovian account of dynamics.
Proceedings of Semantics and Linguistic Theory XVI.


Gillon, B. (1996).
Collectivity and distributivity internal to english noun phrases.
Language Sciences, 18(1):443–468.


Kamp, H. and Reyle, U. (1993).
From discourse to logic: Introduction to model theoretic semantics of natural language, formal logic and discourse representation theory, volume 42.
Kluwer Academic Dordrecht, The Netherlands.


Schwertel, U., Hess, M., and Fuchs, N. (2003).
Plural Semantics for Natural Language Understanding.
PhD thesis, PhD thesis, Faculty of Arts–University of Zurich, 2005. Available at http://www. ifi. unizh. ch/attempto/publications.