Filling gaps with Glue

Adam Przepiórkowski[1,2] and Agnieszka Patejuk[1,3]

[1]Institute of Computer Science, **Polish Academy of Sciences**
[2]Institute of Philosophy, **University of Warsaw**
[3]Centre for Linguistics and Philology, **University of Oxford**

28th Annual Lexical-Functional Grammar Conference
22 July 2023

Problem

Consider a simple **gapping** example:

- Marge saw Lisa and Homer – Bart.

Assume a simple **desired meaning representation**:

- $see(m, l) \wedge see(h, b)$, or (better):
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Problem:

- **how to derive such representations** compositionally…
- …**without empty constituents**?

In particular:

- **one verb** introducing the representation "*see*",
- **two occurrences of "*see*"** in the complete desired representation.

Problem

Consider a simple **gapping** example:

- Marge saw Lisa and Homer – Bart.

Assume a simple **desired meaning representation**:

- $see(m, l) \wedge see(h, b)$, or (better):
- $[\exists e. see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$ $[\exists e. see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Problem:

- **how to derive such representations** compositionally…
- …**without empty constituents**?

In particular:

- **one verb** introducing the representation "*see*",
- **two occurrences of "*see*"** in the complete desired representation.

Problem

Consider a simple **gapping** example:

- Marge saw Lisa and Homer – Bart.

Assume a simple **desired meaning representation**:

- $see(m, l) \wedge see(h, b)$, or (better):
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Problem:

- **how to derive such representations** compositionally…
- …**without empty constituents**?

In particular:

- **one verb** introducing the representation "*see*",
- **two occurrences of "*see*"** in the complete desired representation.

Problem

Consider a simple **gapping** example:

- Marge saw Lisa and Homer – Bart.

Assume a simple **desired meaning representation**:

- $see(m, l) \wedge see(h, b)$, or (better):
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Problem:

- **how to derive such representations** compositionally…
- …**without empty constituents**?

In particular:

- **one verb** introducing the representation "*see*",
- **two occurrences of "*see*"** in the complete desired representation.

Problem

Consider a simple **gapping** example:

- Marge saw Lisa and Homer – Bart.

Assume a simple **desired meaning representation**:

- $see(m, l) \wedge see(h, b)$, or (better):
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Problem:

- **how to derive such representations** compositionally…
- …**without empty constituents**?

In particular:

- **one verb** introducing the representation "*see*",
- **two occurrences of "*see*"** in the complete desired representation.

Problem

Consider a simple **gapping** example:

- Marge saw Lisa and Homer – Bart.

Assume a simple **desired meaning representation**:

- $see(m, l) \wedge see(h, b)$, or (better):
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$ $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Problem:

- **how to derive such representations** compositionally…
- …**without empty constituents**?

In particular:

- **one verb** introducing the representation "*see*",
- **two occurrences of** "*see*" in the complete desired representation.

Problem

Consider a simple **gapping** example:

- Marge saw Lisa and Homer – Bart.

Assume a simple **desired meaning representation**:

- $see(m, l) \wedge see(h, b)$, or (better):
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Problem:

- **how to derive such representations** compositionally…
- …**without empty constituents**?

In particular:

- **one verb** introducing the representation "*see*",
- **two occurrences of "*see*"** in the complete desired representation.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,
- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,
- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not* implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,

- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,

- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not* implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,

- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,

- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not* implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,

- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,

- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not* implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,

- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,

- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not* implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,

- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,

- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not*** **implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,

- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,

- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not* implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Solutions

We propose **two solutions**:

(1) **standard Glue** approach,

- but assumes Champollion's (2015) approach to **event semantics**;

(2) **XLE+Glue** implementation of Glue,

- with meaning constructors collected in values of **GLUE attributes**,
- compatible with **various meaning representations**,
- but assumes that GLUE can be made **"deeply distributive"** (cf. PRED);
- however, this assumption is currently ***not* implemented in XLE**.

In either case, we assume the **syntactic analysis of gapping** proposed in Patejuk and Przepiórkowski 2017.

Idea 1

Consider the following sentence and its intended representation:

- Bart walked and whistled.
- $[\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e,b)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e,b)]$

A resource "**problem**" analogous to that in gapping:

- one occurrence of "*Bart*",
- two occurrences of "*b*".

Standard **solution**:

- represent coordination *sans* Bart: $\lambda x.\, [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e,x)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e,x)]$,
- supply and distribute Bart: $[\lambda x.\, [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e,x)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e,x)]](b)$ $\overset{\beta\text{-reduction}}{\rightsquigarrow} [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e,b)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e,b)]$

Idea 1

Consider the following sentence and its intended representation:

- Bart walked and whistled.
- $[\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, b)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, b)]$

A resource "**problem**" analogous to that in gapping:

- one occurrence of "*Bart*",
- two occurrences of "*b*".

Standard **solution**:

- represent coordination *sans* Bart:
  $\lambda x.\, [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, x)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, x)]$,
- supply and distribute Bart:
  $[\lambda x.\, [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, x)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, x)]](b)$
  $\overset{\beta\text{-reduction}}{\rightsquigarrow} [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, b)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, b)]$

Idea 1

Consider the following sentence and its intended representation:

- Bart walked and whistled.
- $[\exists e.\, \textit{walk}(e) \wedge \textit{agent}(e, b)] \wedge [\exists e.\, \textit{whistle}(e) \wedge \textit{agent}(e, b)]$

A resource "**problem**" analogous to that in gapping:

- one occurrence of "*Bart*",
- two occurrences of "*b*".

Standard **solution**:

- represent coordination *sans* Bart:
  $\lambda x.\, [\exists e.\, \textit{walk}(e) \wedge \textit{agent}(e, x)] \wedge [\exists e.\, \textit{whistle}(e) \wedge \textit{agent}(e, x)]$,
- supply and distribute Bart:
  $[\lambda x.\, [\exists e.\, \textit{walk}(e) \wedge \textit{agent}(e, x)] \wedge [\exists e.\, \textit{whistle}(e) \wedge \textit{agent}(e, x)]](b)$
  $\overset{\beta\text{-reduction}}{\rightsquigarrow} [\exists e.\, \textit{walk}(e) \wedge \textit{agent}(e, b)] \wedge [\exists e.\, \textit{whistle}(e) \wedge \textit{agent}(e, b)]$

Idea 1

Consider the following sentence and its intended representation:

- Bart walked and whistled.
- $[\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, b)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, b)]$

A resource "**problem**" analogous to that in gapping:

- one occurrence of "*Bart*",
- two occurrences of "b".

Standard **solution**:

- represent coordination *sans* Bart:
  $\lambda x.\, [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, x)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, x)]$,
- supply and distribute Bart:
  $[\lambda x.\, [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, x)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, x)]](b)$
  $\overset{\beta\text{-reduction}}{\rightsquigarrow} [\exists e.\, \mathit{walk}(e) \wedge \mathit{agent}(e, b)] \wedge [\exists e.\, \mathit{whistle}(e) \wedge \mathit{agent}(e, b)]$

Idea 2

Similarly in the running example of **gapping**:

- Marge saw Lisa and Homer – Bart.
- $[\exists e.\, \mathit{see}(e) \wedge \mathit{agent}(e, m) \wedge \mathit{theme}(e, l)] \wedge$
  $[\exists e.\, \mathit{see}(e) \wedge \mathit{agent}(e, h) \wedge \mathit{theme}(e, b)]$

The above representation may be **obtained thus**:

- $[\lambda f.[\exists e.\, f(e) \wedge \mathit{agent}(e, m) \wedge \mathit{theme}(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge \mathit{agent}(e, h) \wedge \mathit{theme}(e, b)]](\lambda e.\, \mathit{see}(e))$

The **actual solution** is based on Champollion's (2015) approach to **event semantics**. Technically, it is a little more complex:

- $\mathit{SEE}(\lambda f.[\exists e.\, f(e) \wedge \mathit{agent}(e, m) \wedge \mathit{theme}(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge \mathit{agent}(e, h) \wedge \mathit{theme}(e, b)])$, where
- $\mathit{SEE} \equiv \lambda V.\, \lambda f.\, V(\lambda e.\, \mathit{see}(e) \wedge f(e))$

Idea 2

Similarly in the running example of **gapping**:

- Marge saw Lisa and Homer – Bart.
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge [\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

The above representation may be **obtained thus**:

- $[\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge [\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)]](\lambda e.\, see(e))$

The **actual solution** is based on Champollion's (2015) approach to **event semantics**. Technically, it is a little more complex:

- $SEE(\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge [\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)])$, where
- $SEE \equiv \lambda V.\, \lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$

Idea 2

Similarly in the running example of **gapping**:

- Marge saw Lisa and Homer – Bart.
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

The above representation may be **obtained thus**:

- $[\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)]](\lambda e.\, see(e))$

The **actual solution** is based on Champollion's (2015) approach to **event semantics**. Technically, it is a little more complex:

- $SEE(\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)])$, where
- $SEE \equiv \lambda V.\, \lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$

Idea 2

Similarly in the running example of **gapping**:

- Marge saw Lisa and Homer – Bart.
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

The above representation may be **obtained thus**:

- $[\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)]](\lambda e.\, see(e))$

The **actual solution** is based on Champollion's (2015) approach to **event semantics**. Technically, it is a little more complex:

- $SEE(\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)])$, where
- $SEE \equiv \lambda V.\, \lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$

Idea 2

Similarly in the running example of **gapping**:

- Marge saw Lisa and Homer – Bart.
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

The above representation may be **obtained thus**:

- $[\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)]](\lambda e.\, see(e))$

The **actual solution** is based on Champollion's (2015) approach to **event semantics**. Technically, it is a little more complex:

- $SEE(\lambda f.[\exists e.\, f(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge$
  $[\exists e.\, f(e) \wedge agent(e, h) \wedge theme(e, b)])$, where
- $SEE \equiv \lambda V.\, \lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the "sentence" ***Saw.***:

- $saw([\text{closure}]) \overset{\beta\text{-reduction}}{\rightsquigarrow} \exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, theme(e,l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- $Lisa(saw) \overset{\beta\text{-reduction}}{\rightsquigarrow} \lambda f.\exists e.\, see(e) \wedge theme(e,l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, agent(e,m) \wedge f(e))$
- $Marge(Lisa(saw)) \overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the "sentence" ***Saw.***:

- $saw([\text{closure}])$ $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, theme(e, l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- $Lisa(saw)$ $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$
- $Marge(Lisa(saw))$ $\overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the “sentence” ***Saw.***:

- *saw*([closure]) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, theme(e, l) \wedge f(e))$

Hence, for the “sentence” ***Saw Lisa.*** **(before closure)**:

- *Lisa*(*saw*) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$
- *Marge*(*Lisa*(*saw*)) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\,see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\,true(e)$

Hence, for the "sentence" ***Saw.***:

- *saw*([closure]) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\exists e.\,see(e) \wedge true(e)$ $\equiv$ $\exists e.\,see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\,V(\lambda e.\,theme(e,l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- *Lisa*(*saw*) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\lambda f.\exists e.\,see(e) \wedge theme(e,l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\,V(\lambda e.\,agent(e,m) \wedge f(e))$
- *Marge*(*Lisa*(*saw*)) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\,see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\,see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\,see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\,true(e)$

Hence, for the "sentence" ***Saw.***:

- *saw*([closure]) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\exists e.\,see(e) \wedge true(e)$ $\equiv$ $\exists e.\,see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\,V(\lambda e.\,theme(e,l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- *Lisa*(*saw*) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\lambda f.\exists e.\,see(e) \wedge theme(e,l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\,V(\lambda e.\,agent(e,m) \wedge f(e))$
- *Marge*(*Lisa*(*saw*)) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\,see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\,see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the "sentence" ***Saw.***:

- $saw([\text{closure}]) \overset{\beta\text{-reduction}}{\rightsquigarrow} \exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, theme(e, l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- $Lisa(saw) \overset{\beta\text{-reduction}}{\rightsquigarrow} \lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$
- $Marge(Lisa(saw)) \overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the "sentence" ***Saw.***:

- $saw([\text{closure}]) \overset{\beta\text{-reduction}}{\rightsquigarrow} \exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, theme(e, l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- $Lisa(saw) \overset{\beta\text{-reduction}}{\rightsquigarrow} \lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$
- $Marge(Lisa(saw)) \overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the "sentence" ***Saw.***:

- $saw([\text{closure}]) \overset{\beta\text{-reduction}}{\rightsquigarrow} \exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, theme(e, l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.* (before closure)**:

- $Lisa(saw) \overset{\beta\text{-reduction}}{\rightsquigarrow} \lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.* (before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$
- $Marge(Lisa(saw)) \overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the "sentence" ***Saw.***:

- $saw([\text{closure}]) \overset{\beta\text{-reduction}}{\rightsquigarrow} \exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, theme(e, l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- $Lisa(saw) \overset{\beta\text{-reduction}}{\rightsquigarrow} \lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$
- $Marge(Lisa(saw)) \overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\,see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\,true(e)$

Hence, for the "sentence" ***Saw.***:

- $saw([\text{closure}]) \overset{\beta\text{-reduction}}{\rightsquigarrow} \exists e.\,see(e) \wedge true(e) \equiv \exists e.\,see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\,V(\lambda e.\,theme(e,l) \wedge f(e))$

Hence, for the "sentence" ***Saw Lisa.*** **(before closure)**:

- $Lisa(saw) \overset{\beta\text{-reduction}}{\rightsquigarrow} \lambda f.\exists e.\,see(e) \wedge theme(e,l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\,V(\lambda e.\,agent(e,m) \wedge f(e))$
- $Marge(Lisa(saw)) \overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\,see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\,see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the “sentence” ***Saw.***:

- *saw*([closure]) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, theme(e,l) \wedge f(e))$

Hence, for the “sentence” ***Saw Lisa.* (before closure)**:

- *Lisa*(*saw*) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$ $\lambda f.\exists e.\, see(e) \wedge theme(e,l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.* (before closure)**:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e,m) \wedge f(e))$
- *Marge*(*Lisa*(*saw*)) $\overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e,l) \wedge agent(e,m) \wedge true(e)$

Champollion 2015

An **illustration of Champollion 2015** with *Marge saw Lisa*:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$
- [closure] $\rightsquigarrow$ $\lambda e.\, true(e)$

Hence, for the “sentence” ***Saw.***:

- $saw([\text{closure}]) \overset{\beta\text{-reduction}}{\rightsquigarrow} \exists e.\, see(e) \wedge true(e) \equiv \exists e.\, see(e)$

Dependents are semantic modifiers of verbs, e.g.:

- $Lisa_{theme} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, theme(e, l) \wedge f(e))$

Hence, for the “sentence” ***Saw Lisa.*** **(before closure)**:

- $Lisa(saw) \overset{\beta\text{-reduction}}{\rightsquigarrow} \lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge f(e)$

And for the sentence ***Marge saw Lisa.*** **(before closure)**:

- $Marge_{agent} \rightsquigarrow \lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$
- $Marge(Lisa(saw)) \overset{\beta\text{-reduction}}{\rightsquigarrow}$
  $\lambda f.\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge f(e)$

After closure:

- *Marge saw Lisa.* $\rightsquigarrow$ $\exists e.\, see(e) \wedge theme(e, l) \wedge agent(e, m) \wedge true(e)$

Champollion 2015 and Gapping 1

Champollion 2015:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$

Here:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.\, f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.\, f(e)$

Recall $Marge_{agent}$, $Lisa_{theme}$, etc., e.g.:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$

Then:

- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e, l) \wedge agent(e, m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e, b) \wedge agent(e, h) \wedge f(e)$

Champollion 2015 and Gapping 1

Champollion 2015:

- *saw* $\rightsquigarrow$ $\lambda f. \exists e. see(e) \wedge f(e)$

Here:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ (1) $\lambda V. \lambda f. V(\lambda e. see(e) \wedge f(e))$
- (2) $\lambda f. \exists e. f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f. \exists e. f(e)$

Recall $Marge_{agent}$, $Lisa_{theme}$, etc., e.g.:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V. \lambda f. V(\lambda e. agent(e, m) \wedge f(e))$

Then:

- (2) + $Lisa$ + $Marge$ $\rightsquigarrow$ $\lambda f. \exists e. theme(e, l) \wedge agent(e, m) \wedge f(e)$
- (2) + $Bart$ + $Homer$ $\rightsquigarrow$ $\lambda f. \exists e. theme(e, b) \wedge agent(e, h) \wedge f(e)$

Champollion 2015 and Gapping 1

Champollion 2015:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$

Here:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.\, f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.\, f(e)$

Recall $Marge_{agent}$, $Lisa_{theme}$, etc., e.g.:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$

Then:

- (2) + $Lisa$ + $Marge$ $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e, l) \wedge agent(e, m) \wedge f(e)$
- (2) + $Bart$ + $Homer$ $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e, b) \wedge agent(e, h) \wedge f(e)$

Champollion 2015 and Gapping 1

Champollion 2015:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$

Here:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.\, f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.\, f(e)$

Recall $Marge_{agent}$, $Lisa_{theme}$, etc., e.g.:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e,m) \wedge f(e))$

Then:

- (2) + $Lisa$ + $Marge$ $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e,l) \wedge agent(e,m) \wedge f(e)$
- (2) + $Bart$ + $Homer$ $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e,b) \wedge agent(e,h) \wedge f(e)$

Champollion 2015 and Gapping 1

Champollion 2015:

- *saw* $\rightsquigarrow$ $\lambda f. \exists e. see(e) \wedge f(e)$

Here:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ (1) $\lambda V. \lambda f. V(\lambda e. see(e) \wedge f(e))$
- (2) $\lambda f. \exists e. f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f. \exists e. f(e)$

Recall $Marge_{agent}$, $Lisa_{theme}$, etc., e.g.:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V. \lambda f. V(\lambda e. agent(e, m) \wedge f(e))$

Then:

- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f. \exists e. theme(e, l) \wedge agent(e, m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f. \exists e. theme(e, b) \wedge agent(e, h) \wedge f(e)$

Champollion 2015 and Gapping 1

Champollion 2015:

- *saw* $\rightsquigarrow$ $\lambda f.\, \exists e.\, see(e) \wedge f(e)$

Here:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ (1) $\lambda V.\, \lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$
- (2) $\lambda f.\, \exists e.\, f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\, \exists e.\, f(e)$

Recall $Marge_{agent}$, $Lisa_{theme}$, etc., e.g.:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e,m) \wedge f(e))$

Then:

- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\, \exists e.\, theme(e,l) \wedge agent(e,m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\, \exists e.\, theme(e,b) \wedge agent(e,h) \wedge f(e)$

Champollion 2015 and Gapping 1

Champollion 2015:

- *saw* $\rightsquigarrow$ $\lambda f.\exists e.\, see(e) \wedge f(e)$

Here:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.\, V(\lambda e.\, see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.\, f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.\, f(e)$

Recall $Marge_{agent}$, $Lisa_{theme}$, etc., e.g.:

- $Marge_{agent}$ $\rightsquigarrow$ $\lambda V.\lambda f.\, V(\lambda e.\, agent(e, m) \wedge f(e))$

Then:

- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e, l) \wedge agent(e, m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\exists e.\, theme(e, b) \wedge agent(e, h) \wedge f(e)$

Champollion 2015 and Gapping 2

From the previous slide:

- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.V(\lambda e.see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.f(e)$
- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\exists e.theme(e,l) \wedge agent(e,m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\exists e.theme(e,b) \wedge agent(e,h) \wedge f(e)$

Coordinate the two representations above (Partee and Rooth 1983):

- $\lambda f.[\exists e.theme(e,l) \wedge agent(e,m) \wedge f(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge f(e)]$

Add the idiosyncratic contribution of the verb (1):

- $\lambda f.[\exists e.theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge f(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge f(e)]$

Closure:

- $[\exists e.theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge true(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge true(e)]$

Champollion 2015 and Gapping 2

From the previous slide:

- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.\,V(\lambda e.\,see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.\,f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.\,f(e)$
- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\exists e.\,theme(e,l) \wedge agent(e,m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\exists e.\,theme(e,b) \wedge agent(e,h) \wedge f(e)$

Coordinate the two representations above (Partee and Rooth 1983):

- $\lambda f.\,[\exists e.\,theme(e,l) \wedge agent(e,m) \wedge f(e)] \wedge$
  $[\exists e.\,theme(e,b) \wedge agent(e,h) \wedge f(e)]$

Add the idiosyncratic contribution of the verb (1):

- $\lambda f.\,[\exists e.\,theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge f(e)] \wedge$
  $[\exists e.\,theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge f(e)]$

Closure:

- $[\exists e.\,theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge true(e)] \wedge$
  $[\exists e.\,theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge true(e)]$

Champollion 2015 and Gapping 2

From the previous slide:

- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.V(\lambda e.see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.f(e)$
- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\exists e.theme(e,l) \wedge agent(e,m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\exists e.theme(e,b) \wedge agent(e,h) \wedge f(e)$

Coordinate the two representations above (Partee and Rooth 1983):

- $\lambda f.[\exists e.theme(e,l) \wedge agent(e,m) \wedge f(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge f(e)]$

Add the idiosyncratic contribution of the verb (1):

- $\lambda f.[\exists e.theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge f(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge f(e)]$

Closure:

- $[\exists e.theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge true(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge true(e)]$

Champollion 2015 and Gapping 2

From the previous slide:

- *saw* $\rightsquigarrow$ (1) $\lambda V.\lambda f.V(\lambda e.see(e) \wedge f(e))$
- (2) $\lambda f.\exists e.f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\exists e.f(e)$
- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\exists e.theme(e,l) \wedge agent(e,m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\exists e.theme(e,b) \wedge agent(e,h) \wedge f(e)$

Coordinate the two representations above (Partee and Rooth 1983):

- $\lambda f.[\exists e.theme(e,l) \wedge agent(e,m) \wedge f(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge f(e)]$

Add the idiosyncratic contribution of the verb (1):

- $\lambda f.[\exists e.theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge f(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge f(e)]$

Closure:

- $[\exists e.theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge true(e)] \wedge$
  $[\exists e.theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge true(e)]$

Champollion 2015 and Gapping 2

From the previous slide:

- *saw* $\rightsquigarrow$ (1) $\lambda V.\,\lambda f.\,V(\lambda e.\,see(e) \wedge f(e))$
- (2) $\lambda f.\,\exists e.\,f(e)$
- gapped clause $\rightsquigarrow$ (2) $\lambda f.\,\exists e.\,f(e)$
- (2) + *Lisa* + *Marge* $\rightsquigarrow$ $\lambda f.\,\exists e.\,theme(e,l) \wedge agent(e,m) \wedge f(e)$
- (2) + *Bart* + *Homer* $\rightsquigarrow$ $\lambda f.\,\exists e.\,theme(e,b) \wedge agent(e,h) \wedge f(e)$

Coordinate the two representations above (Partee and Rooth 1983):

- $\lambda f.\,[\exists e.\,theme(e,l) \wedge agent(e,m) \wedge f(e)] \wedge$
  $[\exists e.\,theme(e,b) \wedge agent(e,h) \wedge f(e)]$

Add the idiosyncratic contribution of the verb (1):

- $\lambda f.\,[\exists e.\,theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge f(e)] \wedge$
  $[\exists e.\,theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge f(e)]$

Closure:

- $[\exists e.\,theme(e,l) \wedge agent(e,m) \wedge see(e) \wedge true(e)] \wedge$
  $[\exists e.\,theme(e,b) \wedge agent(e,h) \wedge see(e) \wedge true(e)]$

Limitations 1

Crucial assumption: verbs do not directly refer to their arguments.

Would not work:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ $\lambda x.\lambda y.\, see(x, y)$

Let us **try**:

- *saw* $\rightsquigarrow$ (1) $\lambda x.\lambda y.\, see(x, y)$
- (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$
- gapped clause $\rightsquigarrow$ (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$

As before: ((2) + arguments: m, l, etc.) $\times$ 2 + coordination:

- $\lambda f.\, f(m, l) \wedge f(h, b)$, apply this to (1):
- $see(m, l) \wedge see(h, b)$

Problem: relevant meaning constructors need to **assume the number of dependents and their grammatical functions**, e.g.:

- (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$:
  $(\uparrow \textsc{subj}) \multimap (\uparrow \textsc{obj}) \multimap ((\uparrow \textsc{subj}) \multimap (\uparrow \textsc{obj}) \multimap \uparrow) \multimap \uparrow$

Limitations 1

Crucial assumption: verbs do not directly refer to their arguments.

Would not work:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ $\lambda x.\lambda y.\, see(x, y)$

Let us **try**:

- *saw* $\rightsquigarrow$ (1) $\lambda x.\lambda y.\, see(x, y)$
- (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$
- gapped clause $\rightsquigarrow$ (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$

As before: ((2) + arguments: m, l, etc.) $\times$ 2 + coordination:

- $\lambda f.\, f(m, l) \wedge f(h, b)$, apply this to (1):
- $see(m, l) \wedge see(h, b)$

Problem: relevant meaning constructors need to **assume the number of dependents and their grammatical functions**, e.g.:

- (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$:
  $(\uparrow \textsc{subj}) \multimap (\uparrow \textsc{obj}) \multimap ((\uparrow \textsc{subj}) \multimap (\uparrow \textsc{obj}) \multimap \uparrow) \multimap \uparrow$

Limitations 1

Crucial assumption: verbs do not directly refer to their arguments.

Would not work:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ $\lambda x.\lambda y.\,see(x, y)$

Let us **try**:

- *saw* $\rightsquigarrow$ (1) $\lambda x.\lambda y.\,see(x, y)$
- (2) $\lambda x.\lambda y.\lambda f.\,f(x, y)$
- gapped clause $\rightsquigarrow$ (2) $\lambda x.\lambda y.\lambda f.\,f(x, y)$

As before: ((2) + arguments: m, l, etc.) $\times$ 2 + coordination:

- $\lambda f.\,f(m, l) \wedge f(h, b)$, apply this to (1):
- $see(m, l) \wedge see(h, b)$

Problem: relevant meaning constructors need to **assume the number of dependents and their grammatical functions**, e.g.:

- (2) $\lambda x.\lambda y.\lambda f.\,f(x, y)$:

  $(\uparrow \text{SUBJ}) \multimap (\uparrow \text{OBJ}) \multimap ((\uparrow \text{SUBJ}) \multimap (\uparrow \text{OBJ}) \multimap \uparrow) \multimap \uparrow$

Limitations 1

Crucial assumption: verbs do not directly refer to their arguments.

Would not work:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ $\lambda x.\lambda y.\,see(x,y)$

Let us **try**:

- *saw* $\rightsquigarrow$ (1) $\lambda x.\lambda y.\,see(x,y)$
- (2) $\lambda x.\lambda y.\lambda f.\,f(x,y)$
- gapped clause $\rightsquigarrow$ (2) $\lambda x.\lambda y.\lambda f.\,f(x,y)$

As before: ((2) + arguments: m, l, etc.) $\times$ 2 + coordination:

- $\lambda f.\,f(m,l) \wedge f(h,b)$, apply this to (1):
- $see(m,l) \wedge see(h,b)$

Problem: relevant meaning constructors need to **assume the number of dependents and their grammatical functions**, e.g.:

- (2) $\lambda x.\lambda y.\lambda f.\,f(x,y)$:

  $(\uparrow \text{SUBJ}) \multimap (\uparrow \text{OBJ}) \multimap ((\uparrow \text{SUBJ}) \multimap (\uparrow \text{OBJ}) \multimap \uparrow) \multimap \uparrow$

Limitations 1

Crucial assumption: verbs do not directly refer to their arguments.

Would not work:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ $\lambda x.\lambda y.\, see(x, y)$

Let us **try**:

- *saw* $\rightsquigarrow$ (1) $\lambda x.\lambda y.\, see(x, y)$
- (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$
- gapped clause $\rightsquigarrow$ (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$

As before: ((2) + arguments: m, l, etc.) $\times$ 2 + coordination:

- $\lambda f.\, f(m, l) \wedge f(h, b)$, apply this to (1):
- $see(m, l) \wedge see(h, b)$

Problem: relevant meaning constructors need to **assume the number of dependents and their grammatical functions**, e.g.:

- (2) $\lambda x.\lambda y.\lambda f.\, f(x, y)$:
  $(\uparrow \text{SUBJ}) \multimap (\uparrow \text{OBJ}) \multimap ((\uparrow \text{SUBJ}) \multimap (\uparrow \text{OBJ}) \multimap \uparrow) \multimap \uparrow$

Limitations 1

Crucial assumption: verbs do not directly refer to their arguments.

Would not work:

- Marge saw Lisa and Homer – Bart.
- *saw* $\rightsquigarrow$ $\lambda x.\lambda y.\,see(x,y)$

Let us **try**:

- *saw* $\rightsquigarrow$ (1) $\lambda x.\lambda y.\,see(x,y)$
- (2) $\lambda x.\lambda y.\lambda f.\,f(x,y)$
- gapped clause $\rightsquigarrow$ (2) $\lambda x.\lambda y.\lambda f.\,f(x,y)$

As before: ((2) + arguments: m, l, etc.) $\times$ 2 + coordination:

- $\lambda f.\,f(m,l) \wedge f(h,b)$, apply this to (1):
- $see(m,l) \wedge see(h,b)$

Problem: relevant meaning constructors need to **assume the number of dependents and their grammatical functions**, e.g.:

- (2) $\lambda x.\lambda y.\lambda f.\,f(x,y)$:
  $(\uparrow \textsc{subj}) \multimap (\uparrow \textsc{obj}) \multimap ((\uparrow \textsc{subj}) \multimap (\uparrow \textsc{obj}) \multimap \uparrow) \multimap \uparrow$

Limitations 2

Because of this assumption, **this solution relies on Champollion's (2015) approach** to event semantics.

See the draft paper for the full syntax–semantics interface (and all relevant meaning constructors).

This solution has been **computationally verified** as an XLE+Glue (Dalrymple *et al.* 2020) implementation.

Limitations 2

Because of this assumption, **this solution relies on Champollion's (2015) approach** to event semantics.

See the draft paper for the full syntax–semantics interface (and all relevant meaning constructors).

This solution has been **computationally verified** as an XLE+Glue (Dalrymple *et al.* 2020) implementation.

Limitations 2

Because of this assumption, **this solution relies on Champollion's (2015) approach** to event semantics.

See the draft paper for the full syntax–semantics interface (and all relevant meaning constructors).

This solution has been **computationally verified** as an XLE+Glue (Dalrymple *et al.* 2020) implementation.

Limitations 2

Because of this assumption, **this solution relies on Champollion's (2015) approach** to event semantics.

See the draft paper for the full syntax–semantics interface (and all relevant meaning constructors).

This solution has been **computationally verified** as an XLE+Glue (Dalrymple *et al.* 2020) implementation.

- ∃e1[agent(e1,Homer) ∧ theme(e1,Bart) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[agent(e2,Marge) ∧ theme(e2,Lisa) ∧ see(e2) ∧ true(e2)]
- ∃e1[agent(e1,Homer) ∧ theme(e1,Bart) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[theme(e2,Lisa) ∧ agent(e2,Marge) ∧ see(e2) ∧ true(e2)]
- ∃e1[theme(e1,Bart) ∧ agent(e1,Homer) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[agent(e2,Marge) ∧ theme(e2,Lisa) ∧ see(e2) ∧ true(e2)]
- ∃e1[theme(e1,Bart) ∧ agent(e1,Homer) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[theme(e2,Lisa) ∧ agent(e2,Marge) ∧ see(e2) ∧ true(e2)]
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge [\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Limitations 2

Because of this assumption, **this solution relies on Champollion's (2015) approach** to event semantics.

See the draft paper for the full syntax–semantics interface (and all relevant meaning constructors).

This solution has been **computationally verified** as an XLE+Glue (Dalrymple *et al.* 2020) implementation.

- `∃e1[agent(e1,Homer) ∧ theme(e1,Bart) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[agent(e2,Marge) ∧ theme(e2,Lisa) ∧ see(e2) ∧ true(e2)]`
- `∃e1[agent(e1,Homer) ∧ theme(e1,Bart) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[theme(e2,Lisa) ∧ agent(e2,Marge) ∧ see(e2) ∧ true(e2)]`
- `∃e1[theme(e1,Bart) ∧ agent(e1,Homer) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[agent(e2,Marge) ∧ theme(e2,Lisa) ∧ see(e2) ∧ true(e2)]`
- `∃e1[theme(e1,Bart) ∧ agent(e1,Homer) ∧ see(e1) ∧ true(e1)] ∧ ∃e2[theme(e2,Lisa) ∧ agent(e2,Marge) ∧ see(e2) ∧ true(e2)]`
- $[\exists e.\, see(e) \wedge agent(e, m) \wedge theme(e, l)] \wedge [\exists e.\, see(e) \wedge agent(e, h) \wedge theme(e, b)]$

Limitations 2

Because of this assumption, **this solution relies on Champollion's (2015) approach** to event semantics.

See the draft paper for the full syntax–semantics interface (and all relevant meaning constructors).

This solution has been **computationally verified** as an XLE+Glue (Dalrymple *et al.* 2020) implementation.

- Tracy gave Lisa to Marge and Bart to Homer.
- $[\exists e.\, give(e) \wedge agent(e,t) \wedge theme(e,l) \wedge beneficiary(e,m)] \wedge$
  $[\exists e.\, give(e) \wedge \underline{agent(e,t)} \wedge \underline{theme(e,b)} \wedge beneficiary(e,h)]$
  'Tracy gave Lisa to Marge and Tracy gave Bart to Homer.'
- $[\exists e.\, give(e) \wedge agent(e,t) \wedge theme(e,l) \wedge beneficiary(e,m)] \wedge$
  $[\exists e.\, give(e) \wedge \underline{agent(e,b)} \wedge \underline{theme(e,l)} \wedge beneficiary(e,h)]$
  'Tracy gave Lisa to Marge and Bart gave Lisa to Homer.'

Idea 1

This solution is based on the **XLE+Glue** (Dalrymple *et al.* 2020) **approach to Glue Semantics**:

- typical f-structures have the **set-valued attribute** GLUE,
- **containing** (f-structure encoding of) **meaning constructors**.

For **example**:

- *Marge* N $(\uparrow \text{PRED}) = \text{`MARGE'}$
  $\text{`}m : \uparrow_e\text{'} \in (\uparrow \text{GLUE})$
- *saw* V $(\uparrow \text{PRED}) = \text{`SEE}\langle(\uparrow \text{SUBJ}), (\uparrow \text{OBJ})\rangle\text{'}$
  $\text{`}\lambda x.\lambda y.\, see(x, y) : (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t\text{'} \in (\uparrow \text{GLUE})$
- $$f\begin{bmatrix} \text{PRED} & \text{`SEE<}(f\ \text{SUBJ}), (f\ \text{OBJ})\text{>'} \\ \text{SUBJ} & s\begin{bmatrix} \text{PRED} & \text{`MARGE'} \\ \text{GLUE} & \left\{\text{`}m : s_e\text{'}\right\} \end{bmatrix} \\ \text{OBJ} & o\begin{bmatrix} \text{PRED} & \text{`LISA'} \\ \text{GLUE} & \left\{\text{`}l : o_e\text{'}\right\} \end{bmatrix} \\ \text{GLUE} & \left\{\text{`}\lambda x.\lambda y.\, see(x, y) : (f\ \text{SUBJ})_e \multimap (f\ \text{OBJ})_e \multimap f_t\text{'}\right\} \end{bmatrix}$$

Idea 1

This solution is based on the **XLE+Glue** (Dalrymple *et al.* 2020) **approach to Glue Semantics**:

- typical f-structures have the **set-valued attribute GLUE**,
- **containing** (f-structure encoding of) **meaning constructors**.

For **example**:

- *Marge* N $(\uparrow \text{PRED}) = \text{`MARGE'}$
  $\text{`}m : \uparrow_e\text{'} \in (\uparrow \text{GLUE})$
- *saw* V $(\uparrow \text{PRED}) = \text{`SEE}\langle(\uparrow \text{SUBJ}), (\uparrow \text{OBJ})\rangle\text{'}$
  $\text{`}\lambda x.\lambda y.\, see(x, y) : (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t\text{'} \in (\uparrow \text{GLUE})$
-
$$f\begin{bmatrix} \text{PRED} & \text{`SEE<}(f\ \text{SUBJ}), (f\ \text{OBJ})\text{>'} \\ \text{SUBJ} & s\begin{bmatrix} \text{PRED} & \text{`MARGE'} \\ \text{GLUE} & \left\{\text{`}m : s_e\text{'}\right\} \end{bmatrix} \\ \text{OBJ} & o\begin{bmatrix} \text{PRED} & \text{`LISA'} \\ \text{GLUE} & \left\{\text{`}l : o_e\text{'}\right\} \end{bmatrix} \\ \text{GLUE} & \left\{\text{`}\lambda x.\lambda y.\, see(x, y) : (f\ \text{SUBJ})_e \multimap (f\ \text{OBJ})_e \multimap f_t\text{'}\right\} \end{bmatrix}$$

Idea 1

This solution is based on the **XLE+Glue** (Dalrymple *et al.* 2020) **approach to Glue Semantics**:

- typical f-structures have the **set-valued attribute GLUE**,
- **containing** (f-structure encoding of) **meaning constructors**.

For **example**:

- *Marge* N $(\uparrow \text{PRED}) = \text{`MARGE'}$
  $\text{`}m : \uparrow_e\text{'} \in (\uparrow \text{GLUE})$
- *saw* V $(\uparrow \text{PRED}) = \text{`SEE}\langle(\uparrow \text{SUBJ}), (\uparrow \text{OBJ})\rangle\text{'}$
  $\text{`}\lambda x.\lambda y.\, see(x, y) : (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t\text{'} \in (\uparrow \text{GLUE})$
- $$f\begin{bmatrix} \text{PRED} & \text{`SEE<}(f\ \text{SUBJ}), (f\ \text{OBJ})\text{>'} \\ \text{SUBJ} & s\begin{bmatrix} \text{PRED} & \text{`MARGE'} \\ \text{GLUE} & \{\text{`}m : s_e\text{'}\} \end{bmatrix} \\ \text{OBJ} & o\begin{bmatrix} \text{PRED} & \text{`LISA'} \\ \text{GLUE} & \{\text{`}l : o_e\text{'}\} \end{bmatrix} \\ \text{GLUE} & \{\text{`}\lambda x.\lambda y.\, see(x, y) : (f\ \text{SUBJ})_e \multimap (f\ \text{OBJ})_e \multimap f_t\text{'}\} \end{bmatrix}$$

Idea 1

This solution is based on the **XLE+Glue** (Dalrymple *et al.* 2020) **approach to Glue Semantics**:

- typical f-structures have the **set-valued attribute GLUE**,
- **containing** (f-structure encoding of) **meaning constructors**.

For **example**:

- *Marge* N $(\uparrow \text{PRED}) = \text{`MARGE'}$
  $\text{`}m : \uparrow_e\text{'} \in (\uparrow \text{GLUE})$
- *saw* V $(\uparrow \text{PRED}) = \text{`SEE}\langle(\uparrow \text{SUBJ}), (\uparrow \text{OBJ})\rangle\text{'}$
  $\text{`}\lambda x.\lambda y.\, see(x, y) : (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t\text{'} \in (\uparrow \text{GLUE})$
- $$f\begin{bmatrix} \text{PRED} & \text{`SEE<}(f\ \text{SUBJ}), (f\ \text{OBJ})\text{>'} \\ \text{SUBJ} & s\begin{bmatrix} \text{PRED} & \text{`MARGE'} \\ \text{GLUE} & \left\{\text{`}m : s_e\text{'}\right\} \end{bmatrix} \\ \text{OBJ} & o\begin{bmatrix} \text{PRED} & \text{`LISA'} \\ \text{GLUE} & \left\{\text{`}l : o_e\text{'}\right\} \end{bmatrix} \\ \text{GLUE} & \left\{\text{`}\lambda x.\lambda y.\, see(x, y) : (f\ \text{SUBJ})_e \multimap (f\ \text{OBJ})_e \multimap f_t\text{'}\right\} \end{bmatrix}$$

Idea 1

This solution is based on the **XLE+Glue** (Dalrymple *et al.* 2020) **approach to Glue Semantics**:

- typical f-structures have the **set-valued attribute GLUE**,
- **containing** (f-structure encoding of) **meaning constructors**.

For **example**:

- *Marge* N $(\uparrow \text{PRED}) = \text{‘MARGE’}$
  $\text{‘}m : \uparrow_e\text{’} \in (\uparrow \text{GLUE})$
- *saw* V $(\uparrow \text{PRED}) = \text{‘SEE}\langle(\uparrow \text{SUBJ}), (\uparrow \text{OBJ})\rangle\text{’}$
  $\text{‘}\lambda x.\lambda y.\, see(x, y) : (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t\text{’} \in (\uparrow \text{GLUE})$
-
$$f\left[\begin{array}{ll} \text{PRED} & \text{‘SEE<}(f\ \text{SUBJ}), (f\ \text{OBJ})\text{>’} \\ \text{SUBJ} & s\left[\begin{array}{ll} \text{PRED} & \text{‘MARGE’} \\ \text{GLUE} & \left\{\text{‘}m : s_e\text{’}\right\} \end{array}\right] \\ \text{OBJ} & o\left[\begin{array}{ll} \text{PRED} & \text{‘LISA’} \\ \text{GLUE} & \left\{\text{‘}l : o_e\text{’}\right\} \end{array}\right] \\ \text{GLUE} & \left\{\text{‘}\lambda x.\lambda y.\, see(x, y) : (f\ \text{SUBJ})_e \multimap (f\ \text{OBJ})_e \multimap f_t\text{’}\right\} \end{array}\right]$$

Idea 1

This solution is based on the **XLE+Glue** (Dalrymple *et al.* 2020) **approach to Glue Semantics**:

- typical f-structures have the **set-valued attribute GLUE**,
- **containing** (f-structure encoding of) **meaning constructors**.

For **example**:

- *Marge* N $(\uparrow \text{PRED}) = \text{`MARGE'}$
  $\text{`}m : \uparrow_e\text{'} \in (\uparrow \text{GLUE})$
- *saw* V $(\uparrow \text{PRED}) = \text{`SEE}\langle(\uparrow \text{SUBJ}), (\uparrow \text{OBJ})\rangle\text{'}$
  $\text{`}\lambda x.\lambda y.\, see(x, y) : (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t\text{'} \in (\uparrow \text{GLUE})$
- $$f\begin{bmatrix} \text{PRED} & \text{`SEE<}(f\ \text{SUBJ}), (f\ \text{OBJ})\text{>'} \\ \text{SUBJ} & s\begin{bmatrix} \text{PRED} & \text{`MARGE'} \\ \text{GLUE} & \{\text{`}m : s_e\text{'}\} \end{bmatrix} \\ \text{OBJ} & o\begin{bmatrix} \text{PRED} & \text{`LISA'} \\ \text{GLUE} & \{\text{`}l : o_e\text{'}\} \end{bmatrix} \\ \text{GLUE} & \{\text{`}\lambda x.\lambda y.\, see(x, y) : (f\ \text{SUBJ})_e \multimap (f\ \text{OBJ})_e \multimap f_t\text{'}\} \end{bmatrix}$$

Idea 2

Key observation of the **syntactic analysis of gapping** of Patejuk and Przepiórkowski 2017:

- PRED is "deeply distributive".

For **example**:

- $(f\ \textsc{pred}) = \text{`}\textsc{see}\langle(f\ \textsc{subj}),(f\ \textsc{obj})\rangle\text{'}$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \textsc{subj} & \begin{bmatrix} \textsc{pred} & \text{`}\textsc{Marge}\text{'} \end{bmatrix} \\ \textsc{obj} & \begin{bmatrix} \textsc{pred} & \text{`}\textsc{Lisa}\text{'} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textsc{subj} & \begin{bmatrix} \textsc{pred} & \text{`}\textsc{Homer}\text{'} \end{bmatrix} \\ \textsc{obj} & \begin{bmatrix} \textsc{pred} & \text{`}\textsc{Bart}\text{'} \end{bmatrix} \end{bmatrix} \right\}$$

- results in:

$$f = \left\{ \begin{bmatrix} \textsc{pred} & \text{`}\textsc{see}<\boxed{1},\boxed{2}>\text{'} \\ \textsc{subj} & \boxed{1}\begin{bmatrix} \textsc{pred} & \text{`}\textsc{Marge}\text{'} \end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix} \textsc{pred} & \text{`}\textsc{Lisa}\text{'} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textsc{pred} & \text{`}\textsc{see}<\boxed{3},\boxed{4}>\text{'} \\ \textsc{subj} & \boxed{3}\begin{bmatrix} \textsc{pred} & \text{`}\textsc{Homer}\text{'} \end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix} \textsc{pred} & \text{`}\textsc{Bart}\text{'} \end{bmatrix} \end{bmatrix} \right\}$$

Idea 2

Key observation of the **syntactic analysis of gapping** of Patejuk and Przepiórkowski 2017:

- **PRED is “deeply distributive”.**

For **example**:

- $(f\ \textsc{pred}) = \text{‘}\textsc{see}\langle(f\ \textsc{subj}),(f\ \textsc{obj})\rangle\text{’}$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \textsc{subj} & \begin{bmatrix} \textsc{pred} & \text{‘Marge’} \end{bmatrix} \\ \textsc{obj} & \begin{bmatrix} \textsc{pred} & \text{‘Lisa’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textsc{subj} & \begin{bmatrix} \textsc{pred} & \text{‘Homer’} \end{bmatrix} \\ \textsc{obj} & \begin{bmatrix} \textsc{pred} & \text{‘Bart’} \end{bmatrix} \end{bmatrix} \right\}$$

- results in:

$$f = \left\{ \begin{bmatrix} \textsc{pred} & \text{‘SEE<}\boxed{1},\boxed{2}\text{>’} \\ \textsc{subj} & \boxed{1}\begin{bmatrix} \textsc{pred} & \text{‘Marge’} \end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix} \textsc{pred} & \text{‘Lisa’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textsc{pred} & \text{‘SEE<}\boxed{3},\boxed{4}\text{>’} \\ \textsc{subj} & \boxed{3}\begin{bmatrix} \textsc{pred} & \text{‘Homer’} \end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix} \textsc{pred} & \text{‘Bart’} \end{bmatrix} \end{bmatrix} \right\}$$

Idea 2

Key observation of the **syntactic analysis of gapping** of Patejuk and Przepiórkowski 2017:

- **PRED is “deeply distributive”**.

For **example**:

- $(f \text{ PRED}) = \text{‘SEE}\langle(f \text{ SUBJ}), (f \text{ OBJ})\rangle\text{’}$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘MARGE’} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{‘LISA’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘HOMER’} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{‘BART’} \end{bmatrix} \end{bmatrix} \right\}$$

- results in:

$$f = \left\{ \begin{bmatrix} \text{PRED} & \text{‘SEE<}\boxed{1},\boxed{2}\text{>’} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{‘MARGE’} \end{bmatrix} \\ \text{OBJ} & \boxed{2}\begin{bmatrix} \text{PRED} & \text{‘LISA’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{‘SEE<}\boxed{3},\boxed{4}\text{>’} \\ \text{SUBJ} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{‘HOMER’} \end{bmatrix} \\ \text{OBJ} & \boxed{4}\begin{bmatrix} \text{PRED} & \text{‘BART’} \end{bmatrix} \end{bmatrix} \right\}$$

Idea 2

Key observation of the **syntactic analysis of gapping** of Patejuk and Przepiórkowski 2017:

- **PRED is "deeply distributive"**.

For **example**:

- $(f \text{ PRED}) = \text{‘SEE}\langle(f \text{ SUBJ}), (f \text{ OBJ})\rangle\text{’}$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘MARGE’} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{‘LISA’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘HOMER’} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{‘BART’} \end{bmatrix} \end{bmatrix} \right\}$$

- results in:

$$f = \left\{ \begin{bmatrix} \text{PRED} & \text{‘SEE<}\boxed{1}\text{,}\boxed{2}\text{>’} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{‘MARGE’} \end{bmatrix} \\ \text{OBJ} & \boxed{2}\begin{bmatrix} \text{PRED} & \text{‘LISA’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{‘SEE<}\boxed{3}\text{,}\boxed{4}\text{>’} \\ \text{SUBJ} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{‘HOMER’} \end{bmatrix} \\ \text{OBJ} & \boxed{4}\begin{bmatrix} \text{PRED} & \text{‘BART’} \end{bmatrix} \end{bmatrix} \right\}$$

Idea 2

Key observation of the **syntactic analysis of gapping** of Patejuk and Przepiórkowski 2017:

- **PRED is "deeply distributive"**.

For **example**:

- $(f\ \text{PRED}) = \text{`SEE}\langle(f\ \text{SUBJ}), (f\ \text{OBJ})\rangle\text{'}$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`MARGE'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{`LISA'} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`HOMER'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{`BART'} \end{bmatrix} \end{bmatrix} \right\}$$

- results in:

$$f = \left\{ \begin{bmatrix} \text{PRED} & \text{`SEE<}\boxed{1},\boxed{2}\text{>'} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{`MARGE'} \end{bmatrix} \\ \text{OBJ} & \boxed{2}\begin{bmatrix} \text{PRED} & \text{`LISA'} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{`SEE<}\boxed{3},\boxed{4}\text{>'} \\ \text{SUBJ} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{`HOMER'} \end{bmatrix} \\ \text{OBJ} & \boxed{4}\begin{bmatrix} \text{PRED} & \text{`BART'} \end{bmatrix} \end{bmatrix} \right\}$$

Idea 2

Key observation of the **syntactic analysis of gapping** of Patejuk and Przepiórkowski 2017:

- **PRED is "deeply distributive"**.

For **example**:

- $(f \text{ PRED}) = \text{‘SEE}\langle (f \text{ SUBJ}), (f \text{ OBJ}) \rangle\text{’}$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘MARGE’} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{‘LISA’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘HOMER’} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{‘BART’} \end{bmatrix} \end{bmatrix} \right\}$$

- results in:

$$f = \left\{ \begin{bmatrix} \text{PRED} & \text{‘SEE}<\boxed{1},\boxed{2}>\text{’} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{‘MARGE’} \end{bmatrix} \\ \text{OBJ} & \boxed{2}\begin{bmatrix} \text{PRED} & \text{‘LISA’} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{‘SEE}<\boxed{3},\boxed{4}>\text{’} \\ \text{SUBJ} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{‘HOMER’} \end{bmatrix} \\ \text{OBJ} & \boxed{4}\begin{bmatrix} \text{PRED} & \text{‘BART’} \end{bmatrix} \end{bmatrix} \right\}$$

Idea 3

We would like **GLUE to behave like PRED**:

- $`\lambda x.\lambda y.\,see(x,y) : (f\ \textsc{subj})_e \multimap (f\ \textsc{obj})_e \multimap f_t' \in (f\ \textsc{glue})$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \textsc{pred} & \text{`see<}\boxed{1},\boxed{2}\text{>'} \\ \textsc{subj} & \boxed{1}\begin{bmatrix}\textsc{pred} & \text{`Marge'}\end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix}\textsc{pred} & \text{`Lisa'}\end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textsc{pred} & \text{`see<}\boxed{3},\boxed{4}\text{>'} \\ \textsc{subj} & \boxed{3}\begin{bmatrix}\textsc{pred} & \text{`Homer'}\end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix}\textsc{pred} & \text{`Bart'}\end{bmatrix} \end{bmatrix} \right\}$$

- **should** result in:

$$f = \left\{ \boxed{5}\begin{bmatrix} \textsc{pred} & \text{`see<}\boxed{1},\boxed{2}\text{>'} \\ \textsc{subj} & \boxed{1}\begin{bmatrix}\textsc{pred} & \text{`Marge'}\end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix}\textsc{pred} & \text{`Lisa'}\end{bmatrix} \\ \textsc{glue} & \left\{`\lambda x.\lambda y.\,see(x,y) : \boxed{1}_e \multimap \boxed{2}_e \multimap \boxed{5}_t'\right\} \end{bmatrix}, \boxed{6}\begin{bmatrix} \textsc{pred} & \text{`see<}\boxed{3},\boxed{4}\text{>'} \\ \textsc{subj} & \boxed{3}\begin{bmatrix}\textsc{pred} & \text{`Homer'}\end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix}\textsc{pred} & \text{`Bart'}\end{bmatrix} \\ \textsc{glue} & \left\{`\lambda x.\lambda y.\,see(x,y) : \boxed{3}_e \multimap \boxed{4}_e \multimap \boxed{6}_t'\right\} \end{bmatrix} \right\}$$

Idea 3

We would like **GLUE to behave like PRED**:

- $\text{`}\lambda x.\lambda y.\,see(x,y) : (f\ \textsc{subj})_e \multimap (f\ \textsc{obj})_e \multimap f_t\text{'} \in (f\ \textsc{glue})$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \textsc{pred} & \text{`}\textsc{see}\langle \boxed{1},\boxed{2} \rangle\text{'} \\ \textsc{subj} & \boxed{1}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Marge}\text{'}\end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Lisa}\text{'}\end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textsc{pred} & \text{`}\textsc{see}\langle \boxed{3},\boxed{4} \rangle\text{'} \\ \textsc{subj} & \boxed{3}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Homer}\text{'}\end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Bart}\text{'}\end{bmatrix} \end{bmatrix} \right\}$$

- should result in:

$$f = \left\{ \boxed{5}\begin{bmatrix} \textsc{pred} & \text{`}\textsc{see}\langle \boxed{1},\boxed{2} \rangle\text{'} \\ \textsc{subj} & \boxed{1}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Marge}\text{'}\end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Lisa}\text{'}\end{bmatrix} \\ \textsc{glue} & \left\{\text{`}\lambda x.\lambda y.\,see(x,y) : \boxed{1}_e \multimap \boxed{2}_e \multimap \boxed{5}_t\text{'}\right\} \end{bmatrix}, \boxed{6}\begin{bmatrix} \textsc{pred} & \text{`}\textsc{see}\langle \boxed{3},\boxed{4} \rangle\text{'} \\ \textsc{subj} & \boxed{3}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Homer}\text{'}\end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix}\textsc{pred} & \text{`}\textsc{Bart}\text{'}\end{bmatrix} \\ \textsc{glue} & \left\{\text{`}\lambda x.\lambda y.\,see(x,y) : \boxed{3}_e \multimap \boxed{4}_e \multimap \boxed{6}_t\text{'}\right\} \end{bmatrix} \right\}$$

Idea 3

We would like **GLUE to behave like PRED**:

- $`\lambda x.\lambda y.\,see(x,y) : (f\ \textsc{subj})_e \multimap (f\ \textsc{obj})_e \multimap f_t' \in (f\ \textsc{glue})$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \textsc{pred} & \text{`SEE<}\boxed{1},\boxed{2}\text{>'} \\ \textsc{subj} & \boxed{1}\begin{bmatrix}\textsc{pred} & \text{`MARGE'}\end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix}\textsc{pred} & \text{`LISA'}\end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textsc{pred} & \text{`SEE<}\boxed{3},\boxed{4}\text{>'} \\ \textsc{subj} & \boxed{3}\begin{bmatrix}\textsc{pred} & \text{`HOMER'}\end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix}\textsc{pred} & \text{`BART'}\end{bmatrix} \end{bmatrix} \right\}$$

- **should** result in:

$$f = \left\{ \boxed{5}\begin{bmatrix} \textsc{pred} & \text{`SEE<}\boxed{1},\boxed{2}\text{>'} \\ \textsc{subj} & \boxed{1}\begin{bmatrix}\textsc{pred} & \text{`MARGE'}\end{bmatrix} \\ \textsc{obj} & \boxed{2}\begin{bmatrix}\textsc{pred} & \text{`LISA'}\end{bmatrix} \\ \textsc{glue} & \left\{ `\lambda x.\lambda y.\,see(x,y) : \boxed{1}_e \multimap \boxed{2}_e \multimap \boxed{5}_t' \right\} \end{bmatrix}, \boxed{6}\begin{bmatrix} \textsc{pred} & \text{`SEE<}\boxed{3},\boxed{4}\text{>'} \\ \textsc{subj} & \boxed{3}\begin{bmatrix}\textsc{pred} & \text{`HOMER'}\end{bmatrix} \\ \textsc{obj} & \boxed{4}\begin{bmatrix}\textsc{pred} & \text{`BART'}\end{bmatrix} \\ \textsc{glue} & \left\{ `\lambda x.\lambda y.\,see(x,y) : \boxed{3}_e \multimap \boxed{4}_e \multimap \boxed{6}_t' \right\} \end{bmatrix} \right\}$$

Idea 3

We would like **GLUE to behave like PRED**:

- '$\lambda x.\lambda y.\,see(x,y) : (f\ \text{SUBJ})_e \multimap (f\ \text{OBJ})_e \multimap f_t$' $\in (f\ \text{GLUE})$
- when combined with specifications amounting to:

$$f = \left\{ \begin{bmatrix} \text{PRED} & \text{'SEE<}\boxed{1},\boxed{2}\text{>'} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{'MARGE'}\end{bmatrix} \\ \text{OBJ} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{'LISA'}\end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{'SEE<}\boxed{3},\boxed{4}\text{>'} \\ \text{SUBJ} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{'HOMER'}\end{bmatrix} \\ \text{OBJ} & \boxed{4}\begin{bmatrix}\text{PRED} & \text{'BART'}\end{bmatrix} \end{bmatrix} \right\}$$

- **should** result in:

$$f = \left\{ \boxed{5}\begin{bmatrix} \text{PRED} & \text{'SEE<}\boxed{1},\boxed{2}\text{>'} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{'MARGE'}\end{bmatrix} \\ \text{OBJ} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{'LISA'}\end{bmatrix} \\ \text{GLUE} & \left\{\text{'}\lambda x.\lambda y.\,see(x,y) : \boxed{1}_e \multimap \boxed{2}_e \multimap \boxed{5}_t\text{'}\right\} \end{bmatrix}, \boxed{6}\begin{bmatrix} \text{PRED} & \text{'SEE<}\boxed{3},\boxed{4}\text{>'} \\ \text{SUBJ} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{'HOMER'}\end{bmatrix} \\ \text{OBJ} & \boxed{4}\begin{bmatrix}\text{PRED} & \text{'BART'}\end{bmatrix} \\ \text{GLUE} & \left\{\text{'}\lambda x.\lambda y.\,see(x,y) : \boxed{3}_e \multimap \boxed{4}_e \multimap \boxed{6}_t\text{'}\right\} \end{bmatrix} \right\}$$

Idea 4

Given an appropriate treatment of **conjunctions**, this would lead to the following (fuller) structure:

- Marge saw Lisa and Homer – Bart.

$$
\boxed{0}\left[\begin{array}{ll}
 & \left\{ \boxed{5}\left[\begin{array}{ll}
\text{PRED} & \text{`SEE}\langle\boxed{1},\boxed{2}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\left[\begin{array}{ll}\text{PRED} & \text{`MARGE'} \\ \text{GLUE} & \{\text{`}m : \boxed{1}_e\text{'}\}\end{array}\right] \\
\text{OBJ} & \boxed{2}\left[\begin{array}{ll}\text{PRED} & \text{`LISA'} \\ \text{GLUE} & \{\text{`}l : \boxed{2}_e\text{'}\}\end{array}\right] \\
\text{GLUE} & \{\text{`}\lambda x.\lambda y.\, see(x,y) : \boxed{1}_e \multimap \boxed{2}_e \multimap \boxed{5}_t\text{'}\}
\end{array}\right],
\boxed{6}\left[\begin{array}{ll}
\text{PRED} & \text{`SEE}\langle\boxed{3},\boxed{4}\rangle\text{'} \\
\text{SUBJ} & \boxed{3}\left[\begin{array}{ll}\text{PRED} & \text{`HOMER'} \\ \text{GLUE} & \{\text{`}h : \boxed{3}_e\text{'}\}\end{array}\right] \\
\text{OBJ} & \boxed{4}\left[\begin{array}{ll}\text{PRED} & \text{`BART'} \\ \text{GLUE} & \{\text{`}b : \boxed{4}_e\text{'}\}\end{array}\right] \\
\text{GLUE} & \{\text{`}\lambda x.\lambda y.\, see(x,y) : \boxed{3}_e \multimap \boxed{4}_e \multimap \boxed{6}_t\text{'}\}
\end{array}\right] \right\} \\
\text{CONJ} & \left[\begin{array}{ll}
\text{FORM} & \text{AND} \\
\text{GLUE} & \{\text{`}\lambda p.\lambda q.\, p \wedge q : \boxed{5}_t \multimap \boxed{6}_t \multimap \boxed{0}_t\text{'}\}
\end{array}\right]
\end{array}\right]
$$

- leading to the **desired representation**: $see(m, l) \wedge see(h, b)$.

Idea 4

Given an appropriate treatment of **conjunctions**, this would lead to the following (fuller) structure:

- Marge saw Lisa and Homer – Bart.

$$\boxed{0}\left[\begin{array}{ll} & \left\{ \boxed{5}\left[\begin{array}{ll} \textsc{pred} & \text{`see<}\boxed{1},\boxed{2}\text{>'} \\ \textsc{subj} & \boxed{1}\left[\begin{array}{ll} \textsc{pred} & \text{`Marge'} \\ \textsc{glue} & \left\{\text{`}m : \boxed{1}_e\text{'}\right\} \end{array}\right] \\ \textsc{obj} & \boxed{2}\left[\begin{array}{ll} \textsc{pred} & \text{`Lisa'} \\ \textsc{glue} & \left\{\text{`}l : \boxed{2}_e\text{'}\right\} \end{array}\right] \\ \textsc{glue} & \left\{\text{`}\lambda x.\lambda y.\, see(x,y) : \boxed{1}_e \multimap \boxed{2}_e \multimap \boxed{5}_t\text{'}\right\} \end{array}\right], \boxed{6}\left[\begin{array}{ll} \textsc{pred} & \text{`see<}\boxed{3},\boxed{4}\text{>'} \\ \textsc{subj} & \boxed{3}\left[\begin{array}{ll} \textsc{pred} & \text{`Homer'} \\ \textsc{glue} & \left\{\text{`}h : \boxed{3}_e\text{'}\right\} \end{array}\right] \\ \textsc{obj} & \boxed{4}\left[\begin{array}{ll} \textsc{pred} & \text{`Bart'} \\ \textsc{glue} & \left\{\text{`}b : \boxed{4}_e\text{'}\right\} \end{array}\right] \\ \textsc{glue} & \left\{\text{`}\lambda x.\lambda y.\, see(x,y) : \boxed{3}_e \multimap \boxed{4}_e \multimap \boxed{6}_t\text{'}\right\} \end{array}\right] \right\} \\ \textsc{conj} & \left[\begin{array}{ll} \textsc{form} & \textsc{and} \\ \textsc{glue} & \left\{\text{`}\lambda p.\lambda q.\, p \wedge q : \boxed{5}_t \multimap \boxed{6}_t \multimap \boxed{0}_t\text{'}\right\} \end{array}\right] \end{array}\right]$$

- leading to the **desired representation**: $see(m, l) \wedge see(h, b)$.

Idea 4

Given an appropriate treatment of **conjunctions**, this would lead to the following (fuller) structure:

- Marge saw Lisa and Homer – Bart.

$$
\boxed{0}\left[\begin{array}{ll}
 & \left\{ \boxed{5}\left[\begin{array}{ll}
\text{PRED} & \text{`SEE<}\boxed{1},\boxed{2}\text{>'} \\
\text{SUBJ} & \boxed{1}\left[\begin{array}{ll}\text{PRED} & \text{`MARGE'} \\ \text{GLUE} & \left\{\text{`}m : \boxed{1}_e\text{'}\right\}\end{array}\right] \\
\text{OBJ} & \boxed{2}\left[\begin{array}{ll}\text{PRED} & \text{`LISA'} \\ \text{GLUE} & \left\{\text{`}l : \boxed{2}_e\text{'}\right\}\end{array}\right] \\
\text{GLUE} & \left\{\text{`}\lambda x.\lambda y.\, see(x, y) : \boxed{1}_e \multimap \boxed{2}_e \multimap \boxed{5}_t\text{'}\right\}
\end{array}\right],
\boxed{6}\left[\begin{array}{ll}
\text{PRED} & \text{`SEE<}\boxed{3},\boxed{4}\text{>'} \\
\text{SUBJ} & \boxed{3}\left[\begin{array}{ll}\text{PRED} & \text{`HOMER'} \\ \text{GLUE} & \left\{\text{`}h : \boxed{3}_e\text{'}\right\}\end{array}\right] \\
\text{OBJ} & \boxed{4}\left[\begin{array}{ll}\text{PRED} & \text{`BART'} \\ \text{GLUE} & \left\{\text{`}b : \boxed{4}_e\text{'}\right\}\end{array}\right] \\
\text{GLUE} & \left\{\text{`}\lambda x.\lambda y.\, see(x, y) : \boxed{3}_e \multimap \boxed{4}_e \multimap \boxed{6}_t\text{'}\right\}
\end{array}\right] \right\} \\
\text{CONJ} & \left[\begin{array}{ll}
\text{FORM} & \text{AND} \\
\text{GLUE} & \left\{\text{`}\lambda p.\lambda q.\, p \wedge q : \boxed{5}_t \multimap \boxed{6}_t \multimap \boxed{0}_t\text{'}\right\}
\end{array}\right]
\end{array}\right]
$$

- leading to the **desired representation**: $see(m, l) \wedge see(h, b)$.

Problem

Problem:

- GLUE does not behave like PRED in XLE,
- **not even when it is declared as distributive.**

Apparently,

- the **"deep distributivity"** of PRED is **hardcoded in XLE**,
- without the possibility of **declaring other attributes as "deeply distributive"**.

Problem

Problem:

- GLUE does not behave like PRED in XLE,
- **not even when it is declared as distributive**.

Apparently,

- the **“deep distributivity”** of PRED is **hardcoded in XLE**,
- without the possibility of **declaring other attributes as “deeply distributive”**.

Problem

Problem:

- GLUE does not behave like PRED in XLE,
- **not even when it is declared as distributive**.

Apparently,

- the **"deep distributivity"** of PRED is **hardcoded in XLE**,
- without the possibility of **declaring other attributes as "deeply distributive"**.

Problem

Problem:

- GLUE does not behave like PRED in XLE,
- **not even when it is declared as distributive**.

Apparently,

- the **“deep distributivity”** of PRED is **hardcoded in XLE**,
- without the possibility of **declaring other attributes as “deeply distributive”**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to simple clauses.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Summary

Two approaches to gapping at the syntax–semantics interface:

- **standard Glue + Champollion's (2015) event semantics**:
  - elegant solution (in the words of a Reviewer),
  - standard Glue mechanism of multiple use of resources,
  - implemented in XLE+Glue;
- **XLE+Glue + "deep distributivity" of GLUE**:
  - does not (need to) assume Champollion's (2015) event semantics,
  - multiplication of meaning constructors via distributivity,
  - does not work because there is no way to make GLUE behave like PRED.

Currently a **proof of concept**, limited empirically:

- to **coordination** (cf. Park 2019 and Bîlbîie *et al.* 2023),
- to **simple clauses**.

Thank you for your attention!

Bîlbîie, G., de la Fuente, I., and Abeillé, A. (2023). Factivity and complementizer omission in English embedded gapping. *Journal of Linguistics*, **59**, 389–426.

Champollion, L. (2015). The interaction of compositional semantics and event semantics. *Linguistics and Philosophy*, **38**(1), 31–66.

Dalrymple, M., Patejuk, A., and Zymla, M.-M. (2020). XLE+Glue – A new tool for integrating semantic analysis in XLE. In M. Butt and T. H. King, eds., *The Proceedings of the LFG'20 Conference*, pp. 89–108. CSLI Publications.

Park, S.-H. (2019). *Gapping: A Constraint-Based Syntax-Semantics Interface*. Ph.D. dissertation, State University of New York at Buffalo.

Partee, B. H. and Rooth, M. (1983). Generalized conjunction and type ambiguity. In R. Bäuerle, C. Schwarze, and A. von Stechow, eds., *Meaning, Use and Interpretation of Language*, pp. 361–383. Walter de Gruyter.

Patejuk, A. and Przepiórkowski, A. (2017). Filling the gap. In M. Butt and T. H. King, eds., *The Proceedings of the LFG'17 Conference*, pp. 327–347. CSLI Publications.