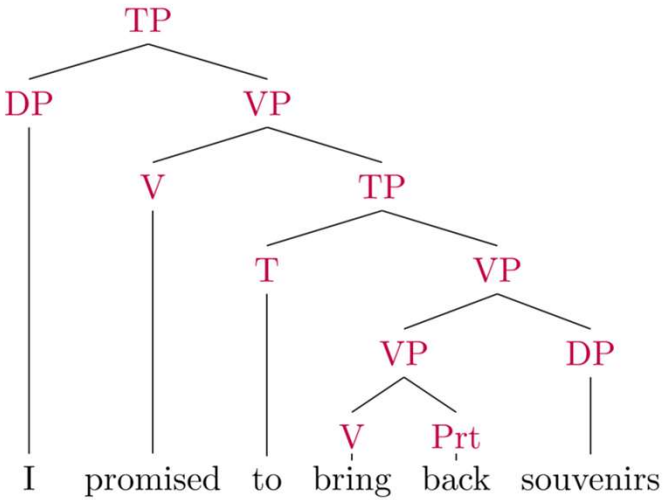
An improved approach for semantic graph composition with CCG

AUSTIN BLODGETT & NATHAN SCHNEIDER

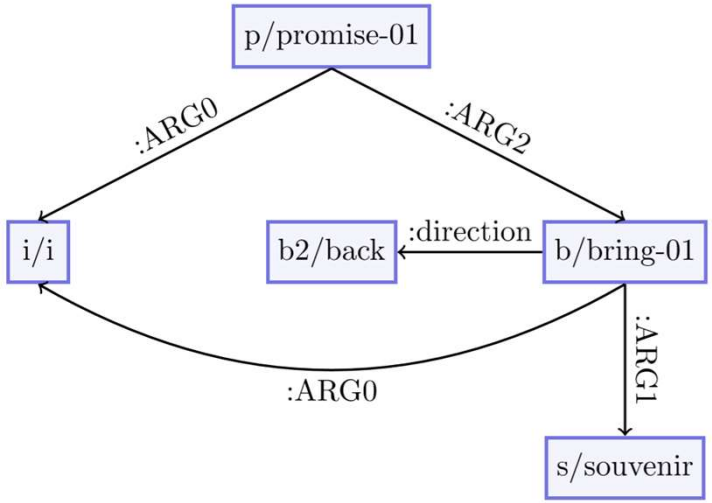
AMR has no Syntax-Semantics Interface

Syntax



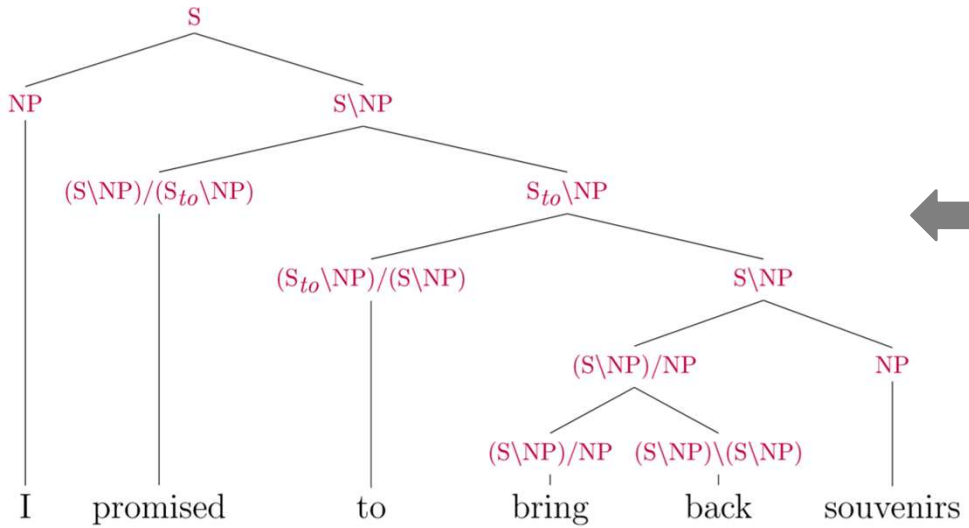
???

Semantics
(Abstract Meaning Representation)



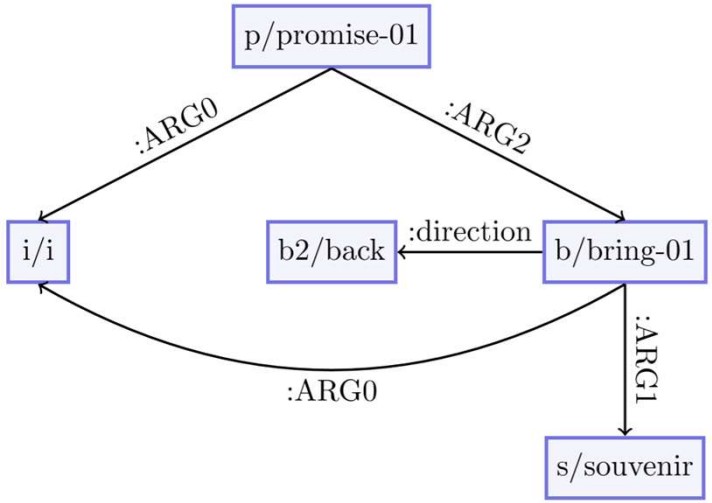
AMR + CCG

Syntax



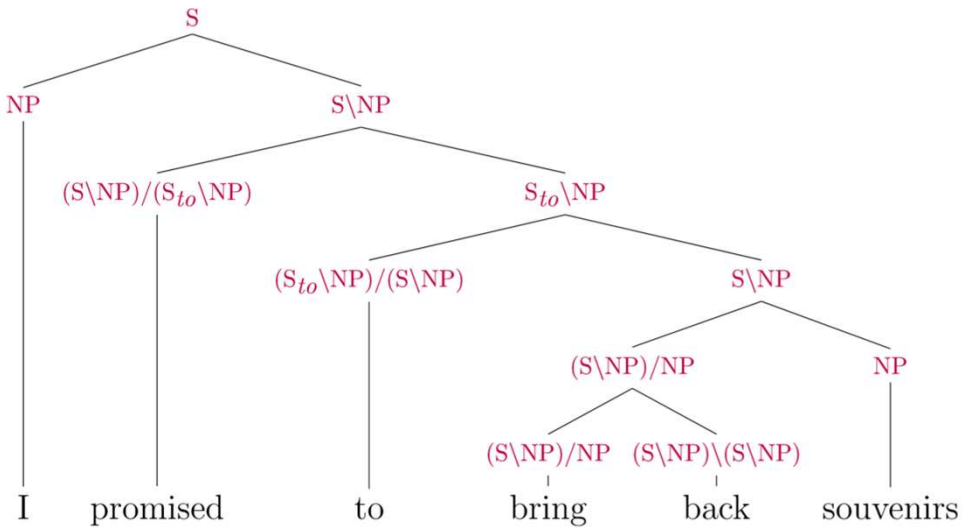
CCG

Semantics
(Abstract Meaning Representation)



Why CCG

Combinatory Categorial Grammar



- Broad-coverage parsing
- Transparent Syntax-Semantics Interface

Our Contributions

Goal: Design interpretable, linguistically meaningful AMR derivation

Contribution:

We directly represent the semantics of CCG entries as AMR subgraphs with free variables

Definitions of combinators (including new definitions of symmetric and type-raising combinators) allow for compositional derivations

Related Work

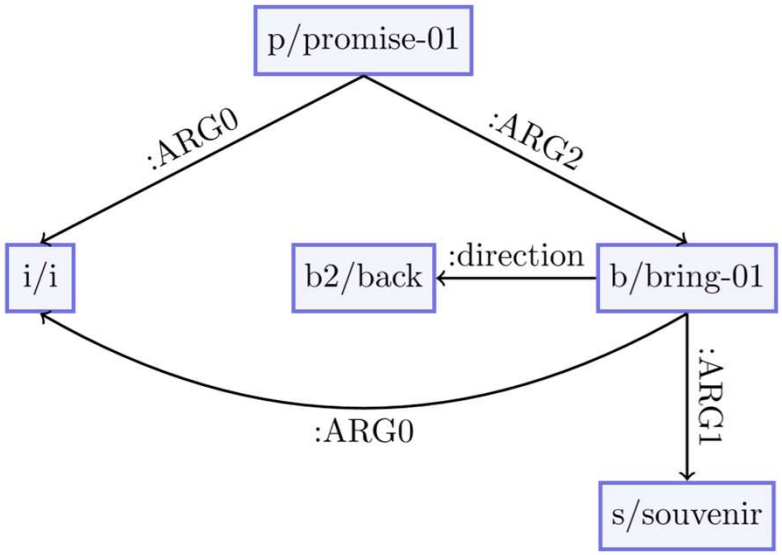
- **AMR-English Syntax alignment:** Alignment is an important precedent for compositional AMR *(Chen and Palmer, 2017; Szubert et al., 2018)*
- **AMR Parsing SOTA:** No supervision for alignment *(Lyu and Titov, 2018; Zhang et al., 2019)*
- **AMR and CCG:**
  - Induced lambda calculus semantics *(Artzi et al., 2015)*
  - Graph algebraic formalization with HR algebra *(Beschke and Menzel 2018)*
- **Other Graph Semantic Representations with CCG** *(Baldridge and Kruijff, 2002)*

Outline from Here

1. AMR: The Basics
2. CCG: The Basics
3. A Simple Example
4. Our Approach
5. Novel Combinators
6. Unsolved Problems
7. Contributions & Next Steps

AMR: The Basics

Abstract Meaning Representation



DAG (Directed Acyclic Graph) data structure to capture the meaning of a sentence.

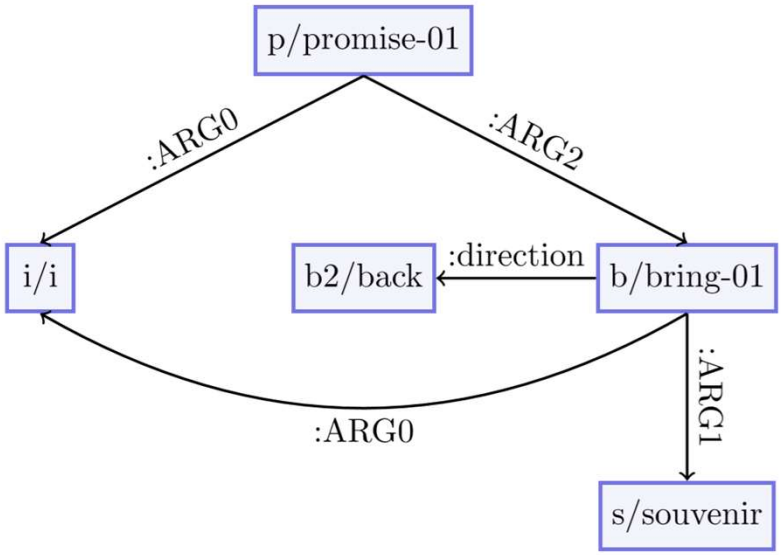
Abstracts away from syntax and morphology.

Computation friendly:
Semantically related elements are close together

Reference sentence: “I promised to bring back souvenirs”
(or: “my promise that I will bring souvenirs back”)

AMR: The Basics

Abstract Meaning Representation



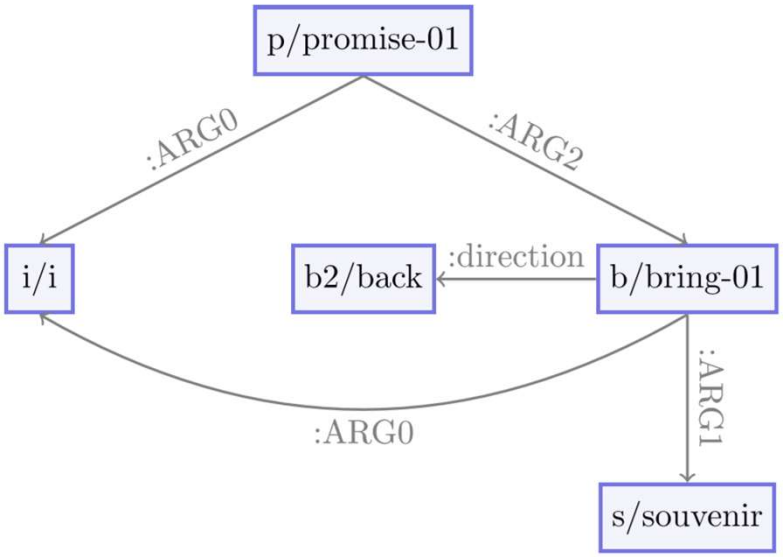
neo-Davidsonian predicate calculus

$$\exists x, y, e_1, e_2 \Big[\textsc{Promise}(e_1) \wedge \textsc{Agent}(e_1)(\textsc{speaker}) \wedge \textsc{Theme}(e_1)(e_2) \wedge \textsc{Bring}(e_2) \wedge \textsc{Agent}(e_2)(\textsc{speaker}) \wedge \textsc{Theme}(e_2)(x) \wedge \textsc{Souvenir}(x) \wedge \textsc{Direction}(e_2)(y) \wedge \textsc{Back}(y) \Big]$$

Reference sentence: “I promised to bring back souvenirs”

AMR: The Basics

Abstract Meaning Representation



neo-Davidsonian predicate calculus

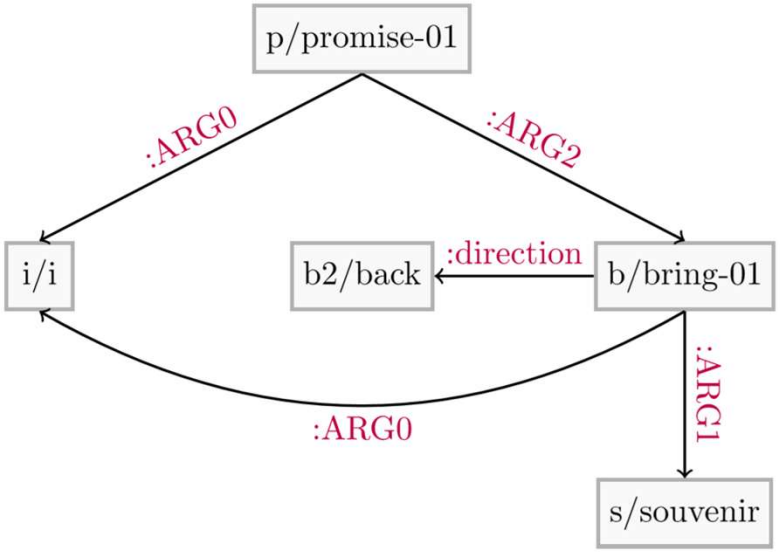
$$\textsc{Promise}(e_1) \wedge \textsc{Bring}(e_2) \wedge \textsc{Souvenir}(x) \wedge \textsc{Back}(y)$$

$$\textsc{speaker}$$

Reference sentence: “I promised to bring back souvenirs”

AMR: The Basics

Abstract Meaning Representation



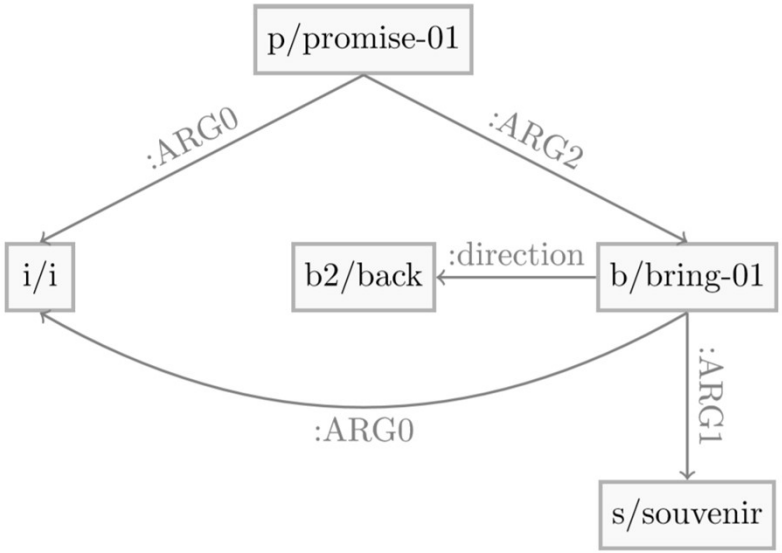
neo-Davidsonian predicate calculus

$$\textsc{Agent}(e_1)(speaker) \wedge \textsc{Theme}(e_1)(e_2) \wedge$$
$$\textsc{Agent}(e_2)(speaker) \wedge \textsc{Theme}(e_2)(x) \wedge$$
$$\textsc{Direction}(e_2)(y)$$

Reference sentence: “I promised to bring back souvenirs”

AMR: The Basics

Abstract Meaning Representation



neo-Davidsonian predicate calculus

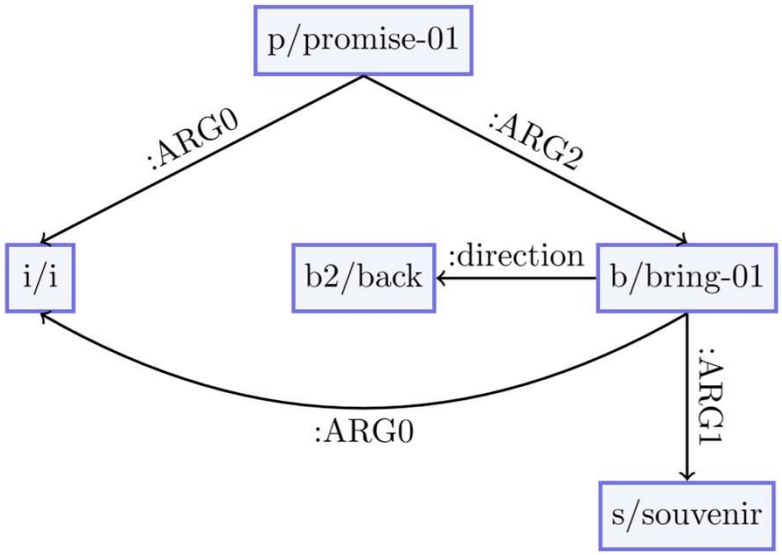
$$\exists x, y, e_1, e_2$$

AMR Ignores:
Existential quantification of variables,
Definiteness, number, tense

Reference sentence: “I promised to bring back souvenirs”

AMR: What's Missing

Abstract Meaning Representation



What's Missing?

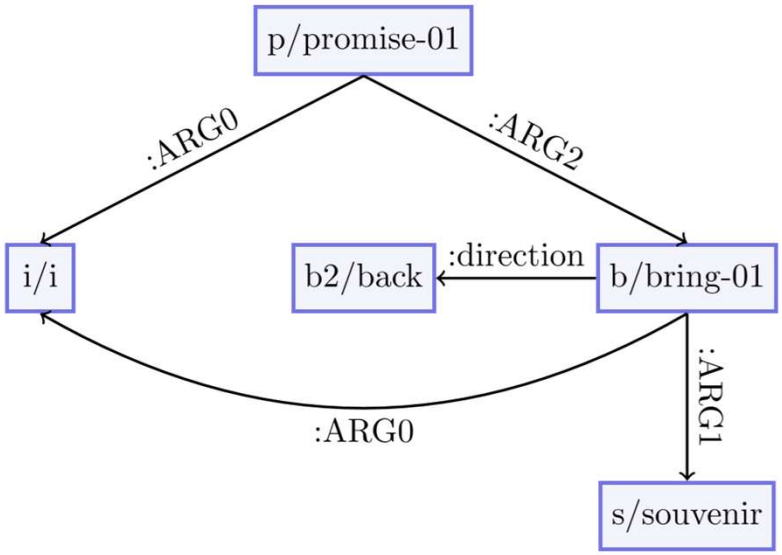
AMR is not a compositional semantics

Not clear how to derive an AMR graph given sequence of words

AMR derivation does not correspond to any linguistic process

AMR: What's Missing

Abstract Meaning Representation



What We Want?

Better structure for introducing inductive biases into AMR parsing.

AMR derivation that is similar to something linguistically meaningful, i.e. composition.

CCG: The Basics

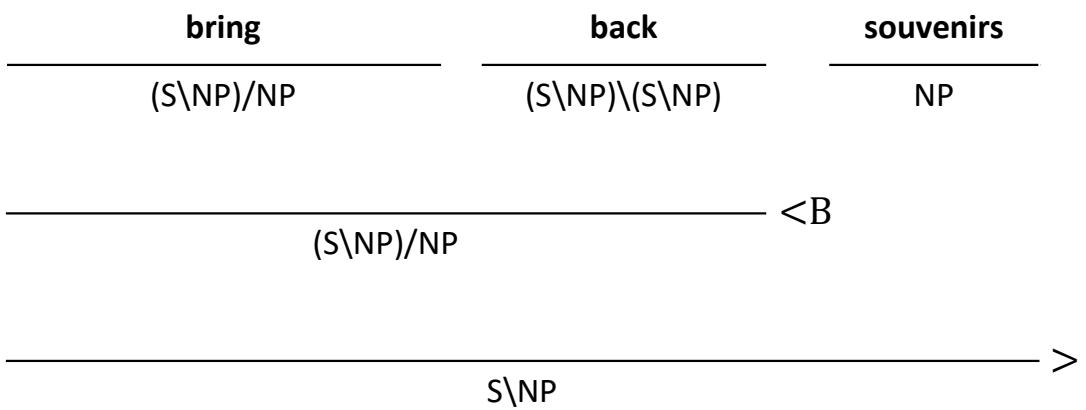
bring	**back**	**souvenirs**
(S\NP)/NP	(S\NP)\(S\NP)	NP

CCG: The Basics

bring	**back**	**souvenirs**
(S\NP)/NP	(S\NP)\(S\NP)	NP

(S\NP)/NP	<B

CCG: The Basics



CCG: The Basics

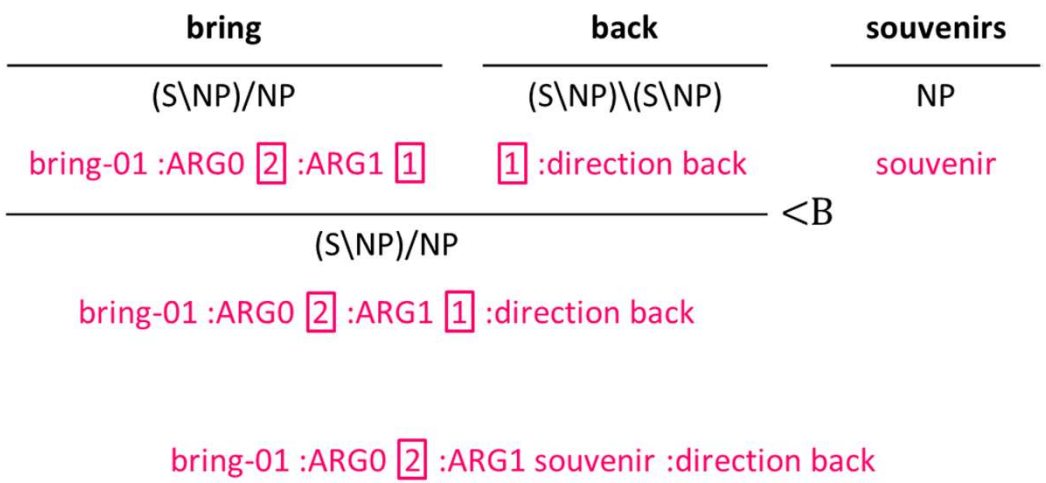
bring	back	souvenirs
(S\NP)/NP	(S\NP)\(S\NP)	NP

bring-01 :ARG0 2 :ARG1 souvenir :direction back

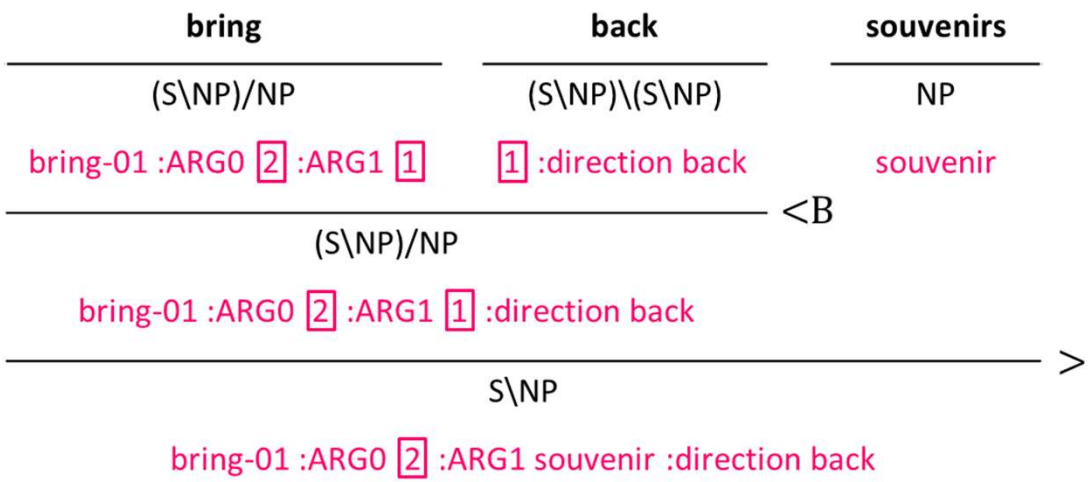
CCG: The Basics

bring-01 :ARG0 2 :ARG1 souvenir :direction back

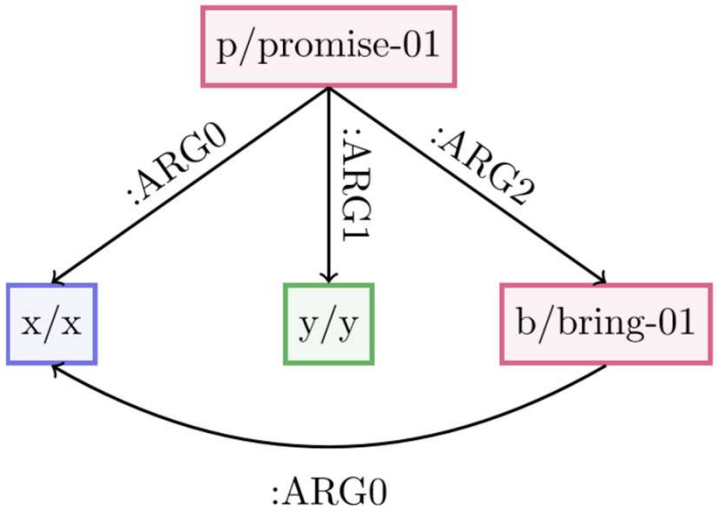
CCG: The Basics



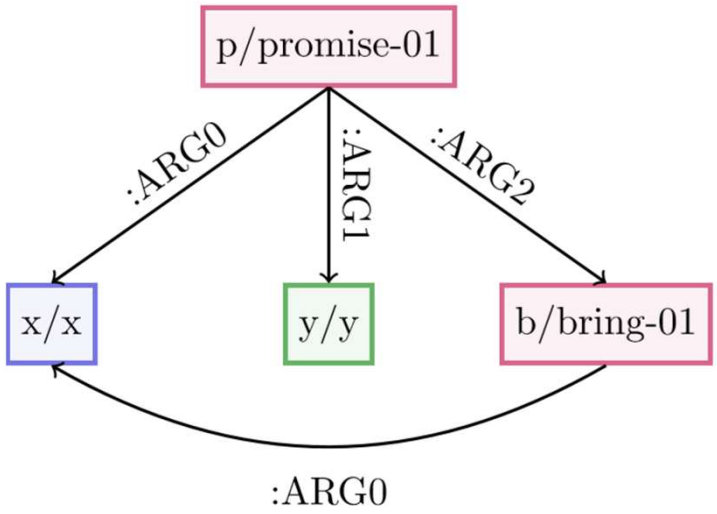
CCG: The Basics



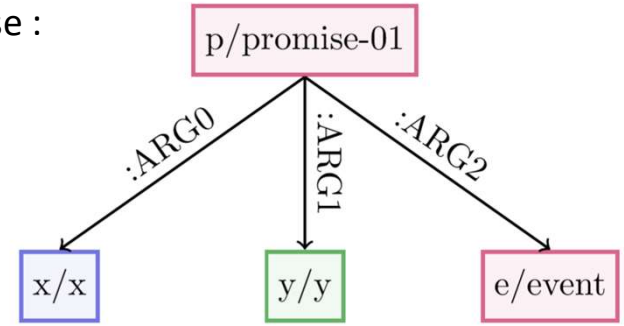
Decomposing AMR



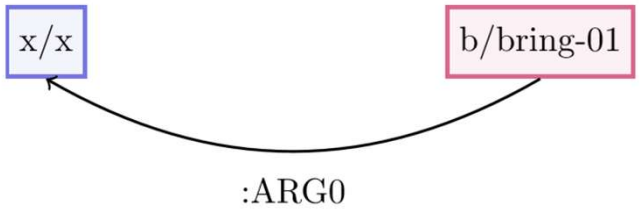
Decomposing AMR



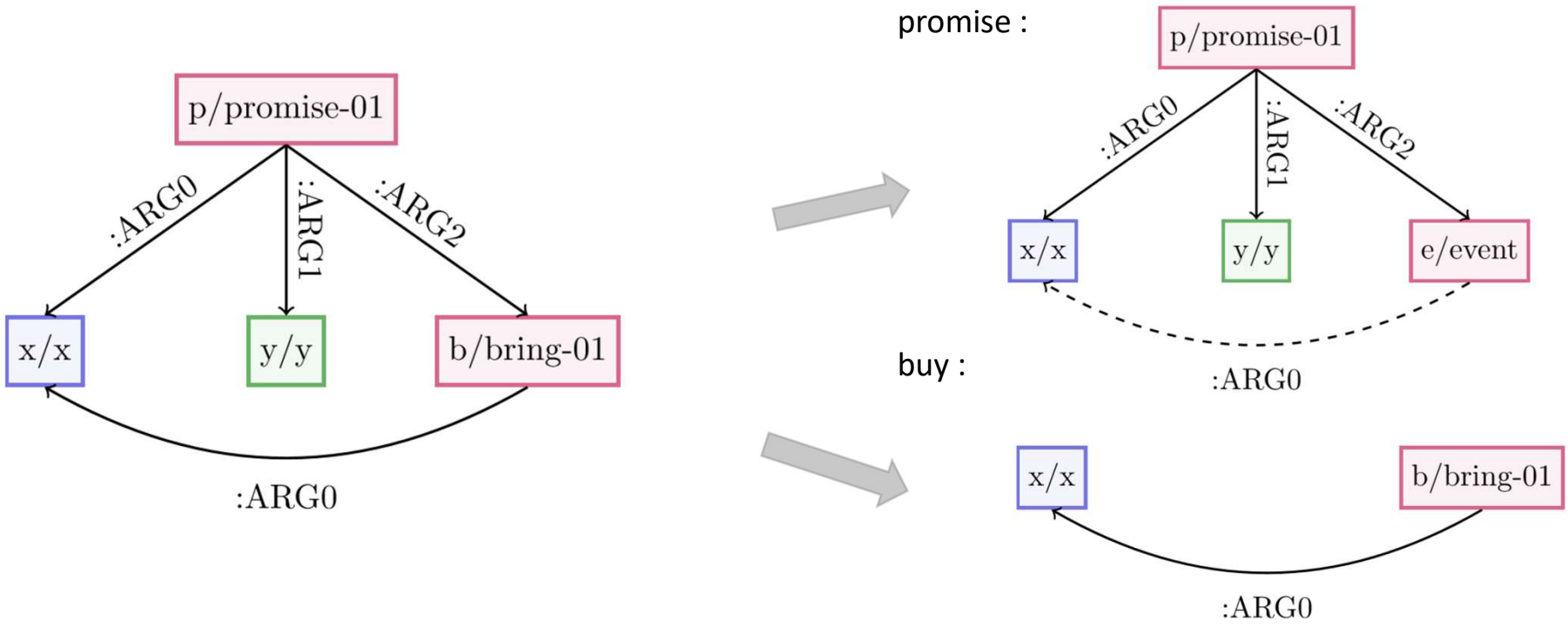
promise :



buy :



Decomposing AMR



Next Step

Convert combinators from operating on lambda term semantics to operating on AMR subgraphs.

Next Step

Convert combinators from operating on lambda term semantics to operating on AMR subgraphs.

Combinators: Function Application, Composition, Type Raising, Relation-wise Application, Relation-wise Composition.

Next Step

Convert combinators from operating on lambda term semantics to operating on AMR subgraphs.

Combinators: Function Application, Composition, Type Raising, Relation-wise Application, Relation-wise Composition.

Philosophy in CCG: Isomorphism between syntax and semantics

- Every syntactic argument should correspond to a semantic argument

Next Step

Convert combinators from operating on lambda term semantics to operating on AMR subgraphs.

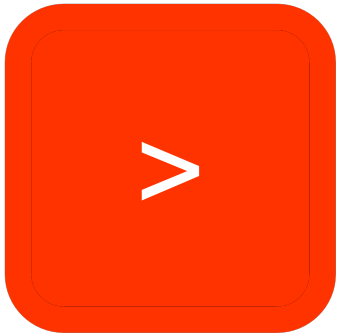
Combinators: Function Application, Composition, Type Raising, Relation-wise Application, Relation-wise Composition.

Philosophy in CCG: Isomorphism between syntax and semantics

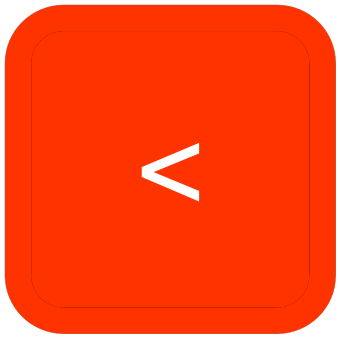
- Every syntactic argument should correspond to a semantic argument
- Our approach: # free variables ≤ # syntactic arguments
- **Why not =?**

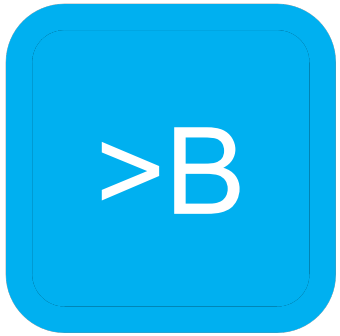
bring	**back**	**souvenirs**
(S\NP)/NP	(S\NP)\(S\NP)	NP
bring-01 :ARG0 [2] :ARG1 [1]	[1] :direction back	souvenir

Combinators: Standard CCG

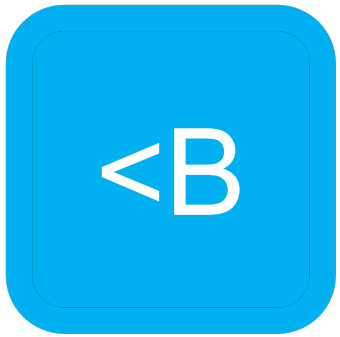


Application





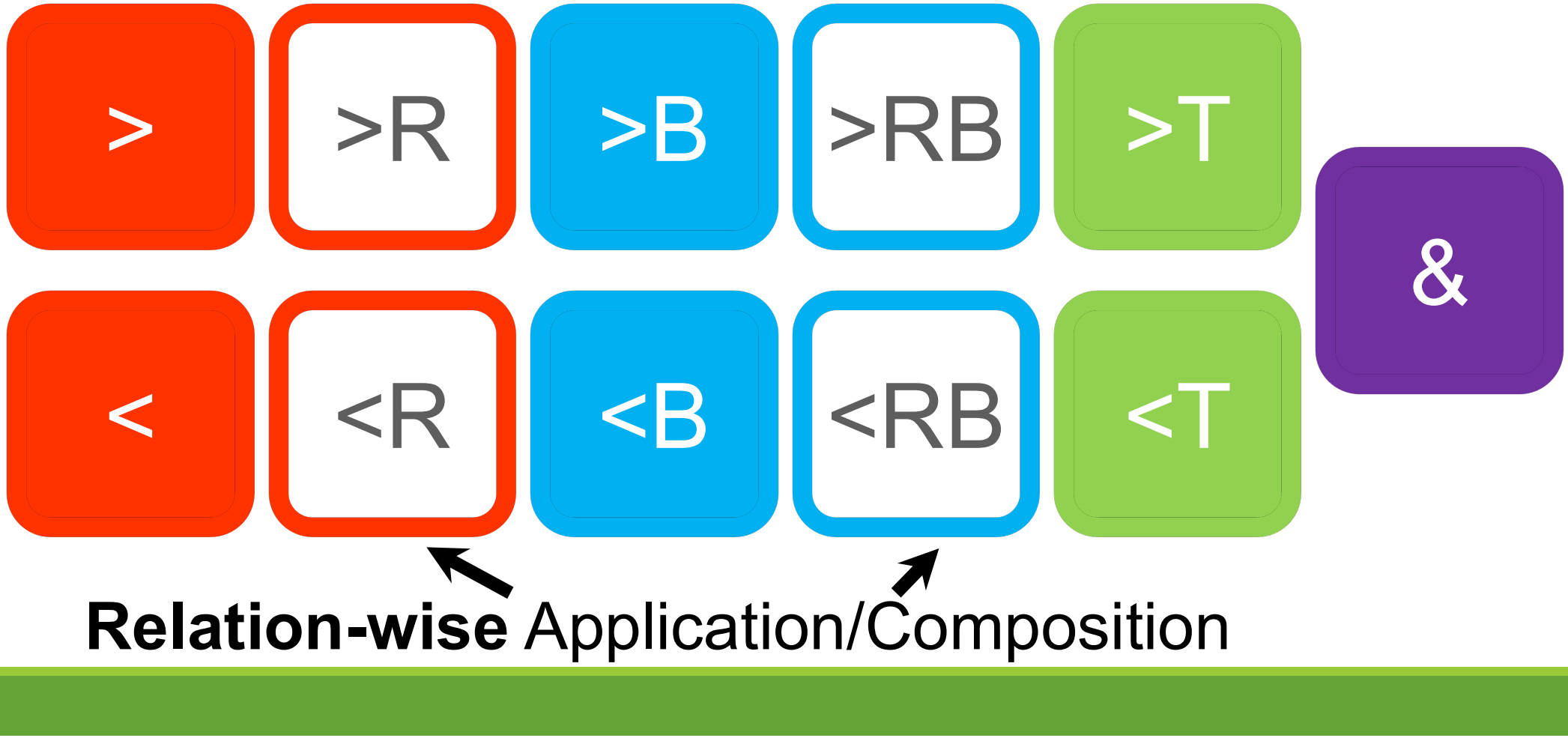
Composition



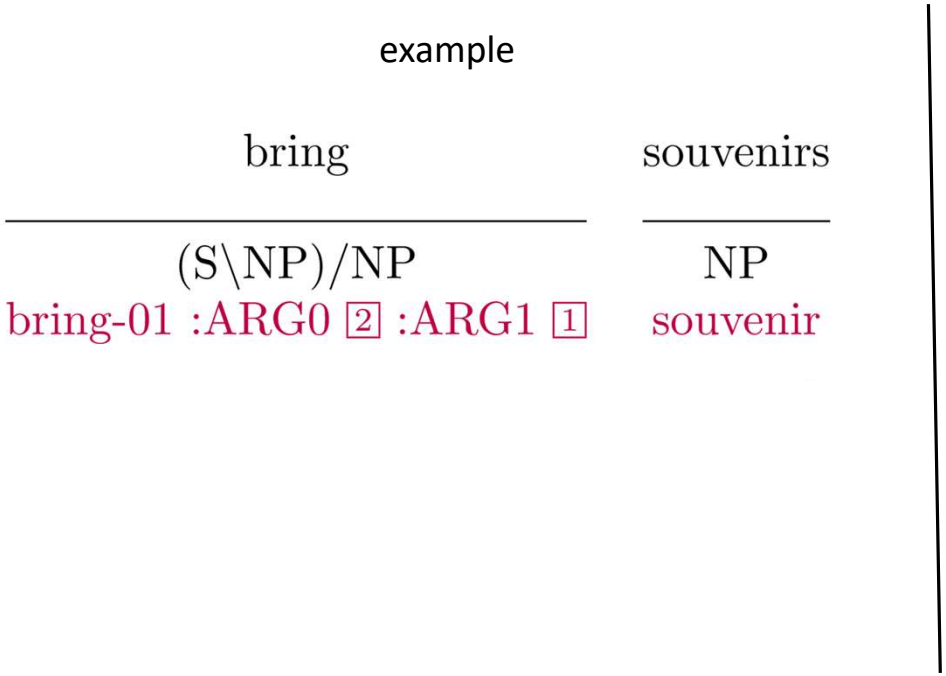
Type-Raising

Conjunction

Combinators: This Work

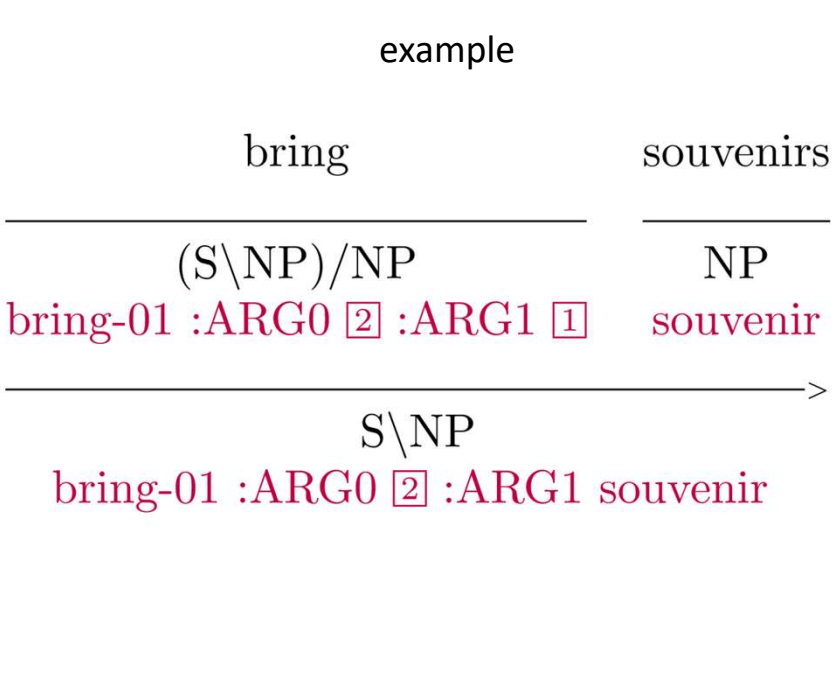


Function Application (< or >)



Procedure: Substitute a *free variable* with the argument's root

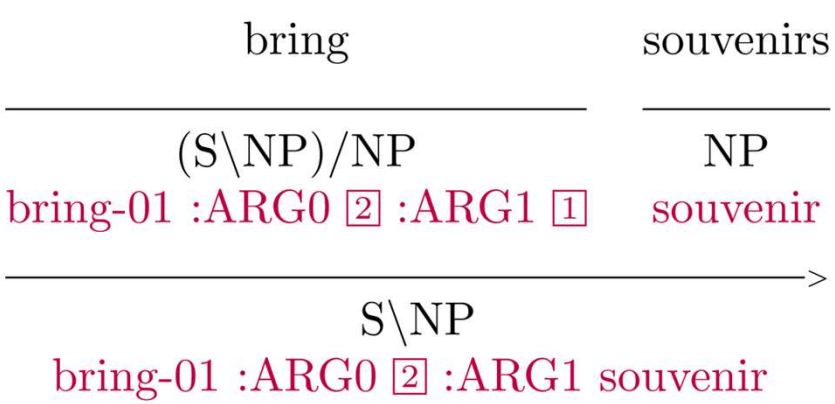
Function Application (< or >)



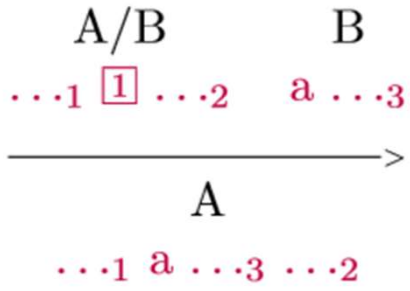
Procedure: Substitute a *free variable* with the argument's root

Function Application (< or >)

example



general form



Procedure: Substitute a *free variable* with the argument's root

Composition (<B or >B)

example

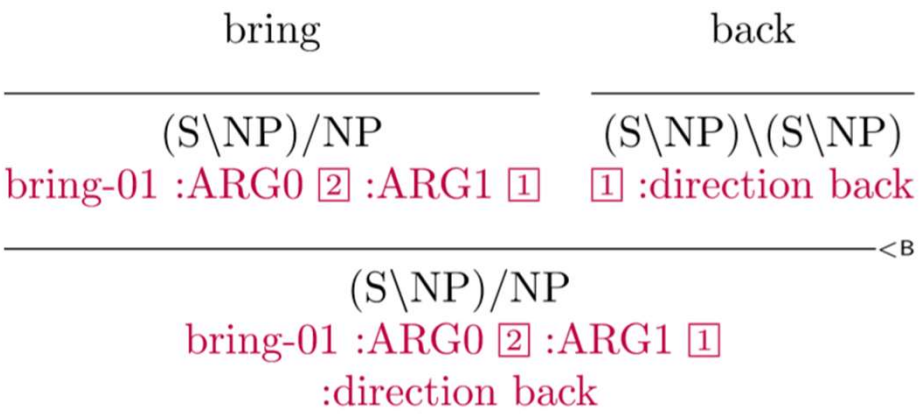
$$\frac{\text{bring} \qquad \text{back}}{\dfrac{(S\backslash NP)/NP}{\text{bring-01 :ARG0 } \boxed{2} \text{ :ARG1 } \boxed{1}} \qquad \dfrac{(S\backslash NP)\backslash(S\backslash NP)}{\boxed{1} \text{ :direction back}}}$$

general form

Procedure: Same as function application, but reorder FV list

Composition (<B or >B)

example

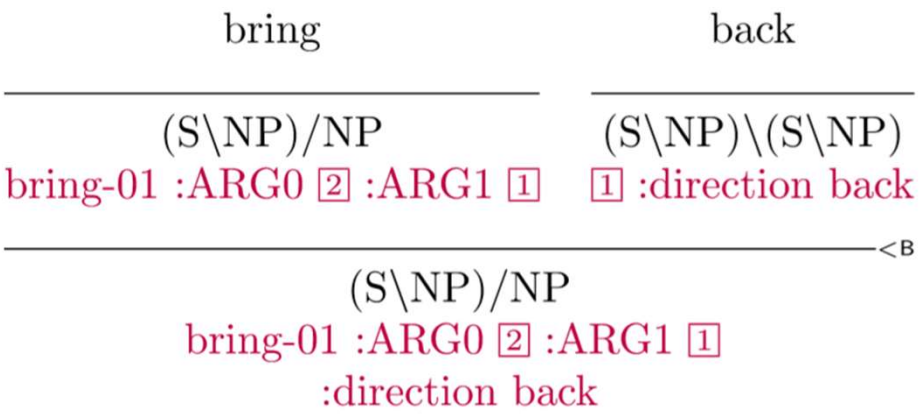


general form

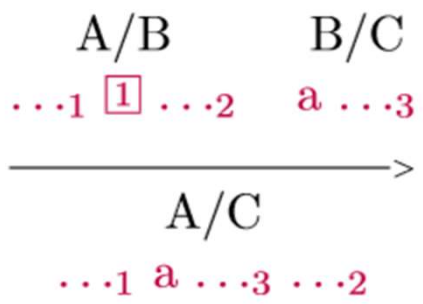
Procedure: Same as function application, but reorder FV list

Composition (<B or >B)

example



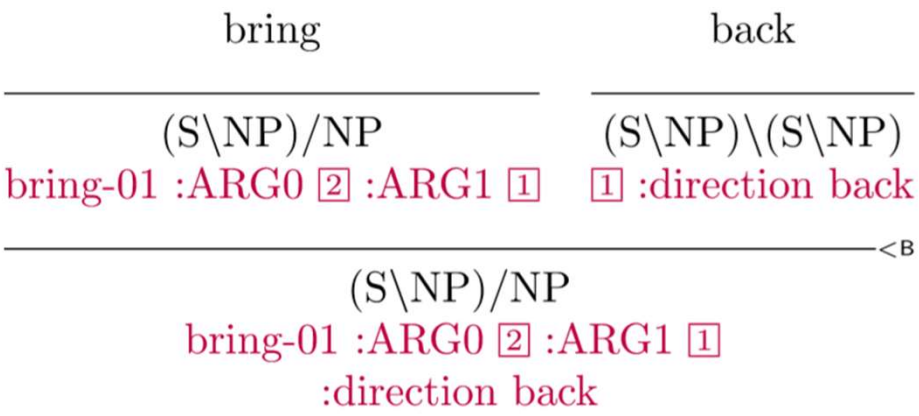
general form



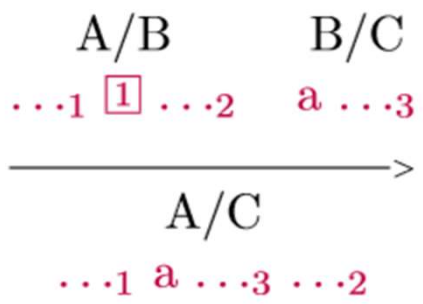
Procedure: Same as function application, but reorder FV list

Composition (<B or >B)

example



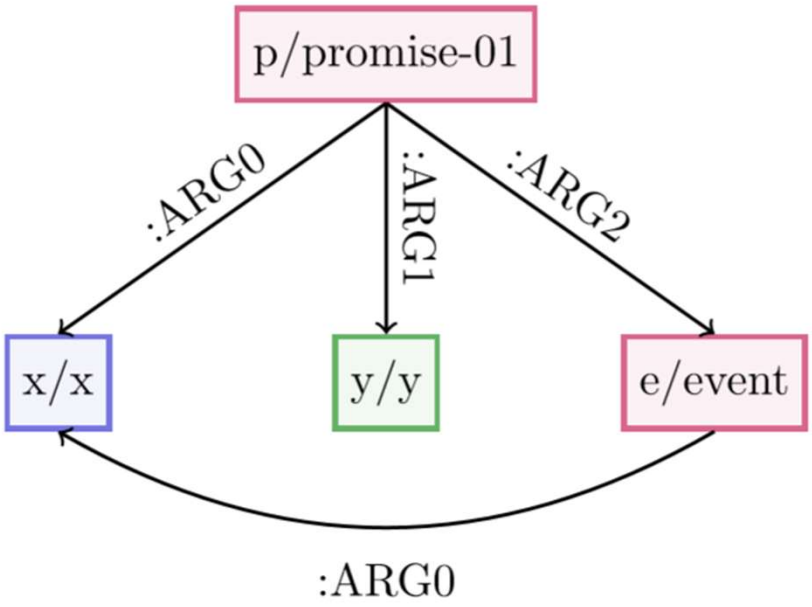
general form



* Order of free variables matters

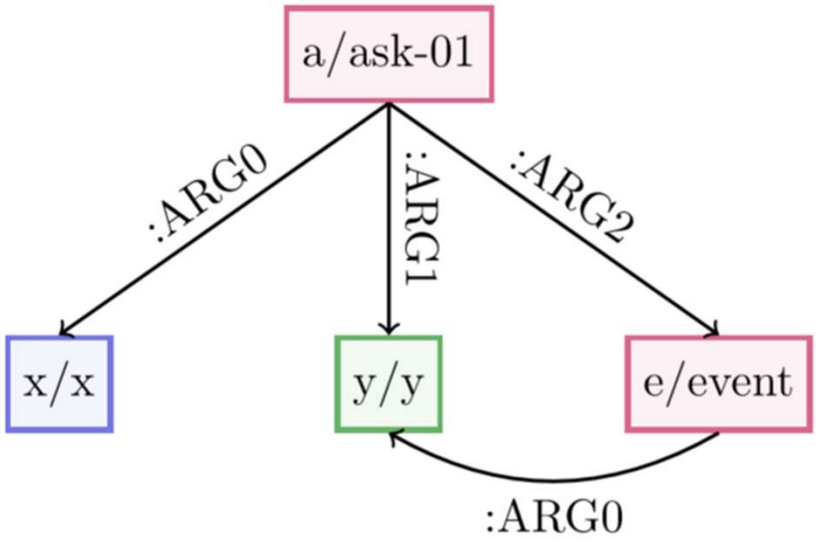
Procedure: Same as function application, but reorder FV list

Classic Problems for Graph Semantics



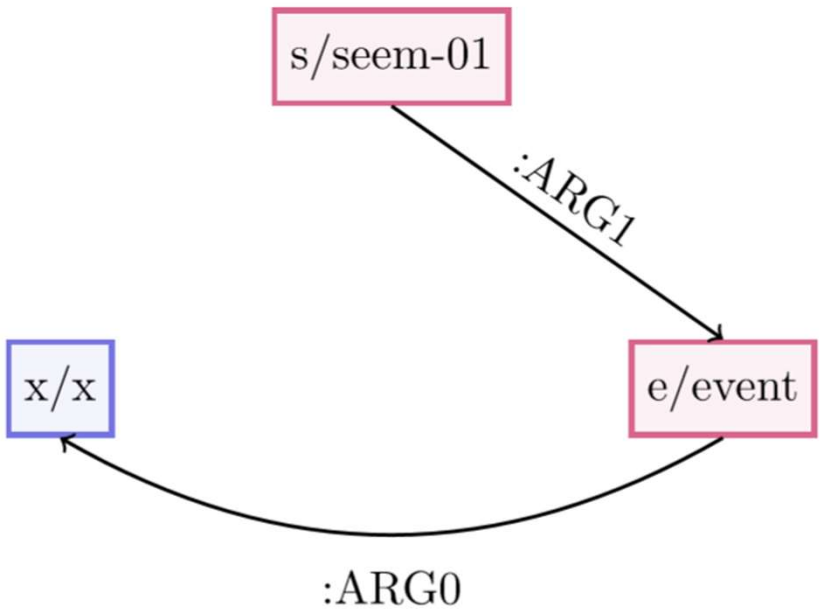
Subject Control, e.g.
“I promised John to buy a ticket”

Classic Problems for Graph Semantics



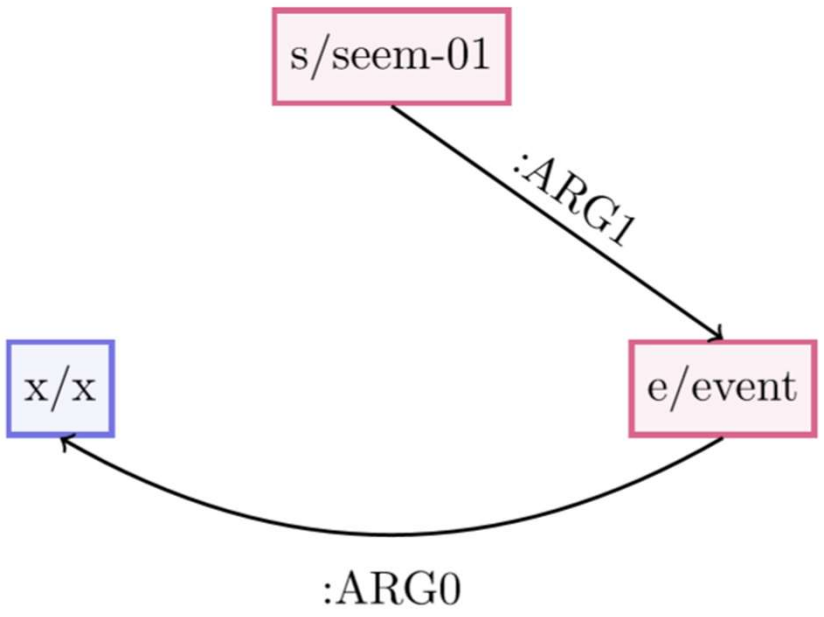
Object Control, e.g.
“I asked John to buy a ticket”

Classic Problems for Graph Semantics



Raising, e.g.
“John seems to be slacking”

Classic Problems for Graph Semantics



Raising, e.g.
"John seems to be slacking"

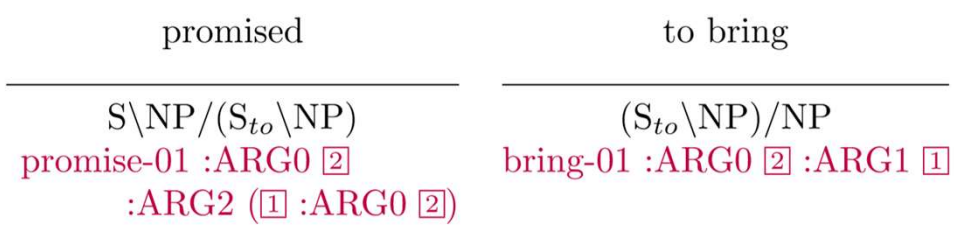
$$seem := \lambda \mathrm{P}, x\left[\textsc{Seem}(\mathrm{P}) \wedge \mathrm{P}(x)\right]$$

$$seem := \lambda e, x\left[\textsc{Seem}(e) \wedge \textsc{Agent}(x)\right]$$

Relation-wise Application (<R or >R), Relation-wise Composition (<RB or >RB)

example

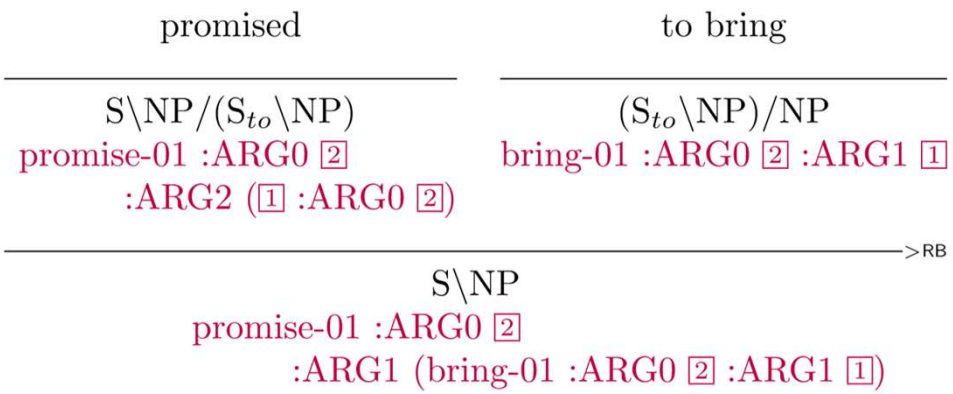
general form



Procedure: Substitute a *relation* instead of a node

Relation-wise Application (<R or >R), Relation-wise Composition (<RB or >RB)

example

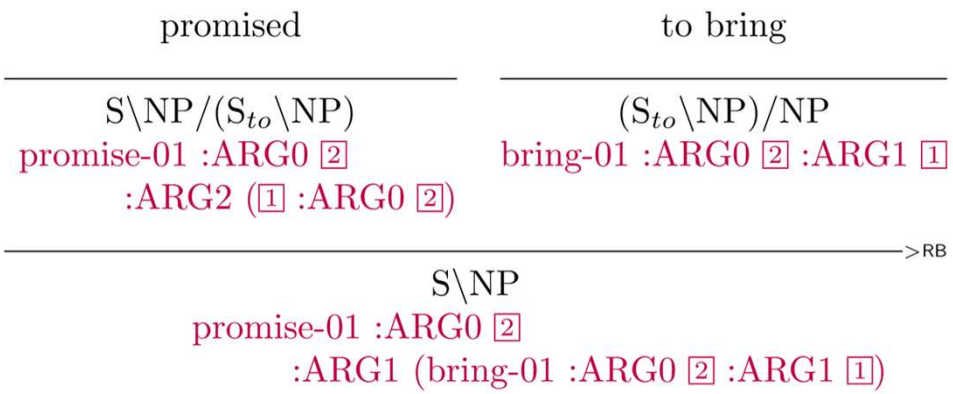


general form

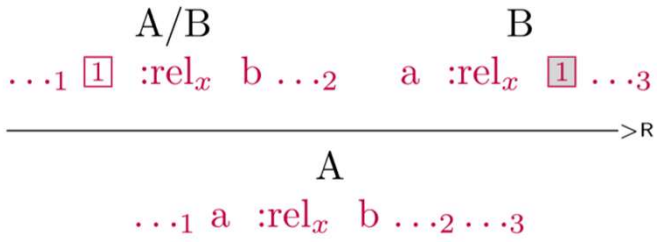
Procedure: Substitute a *relation* instead of a node

Relation-wise Application (<R or >R), Relation-wise Composition (<RB or >RB)

example



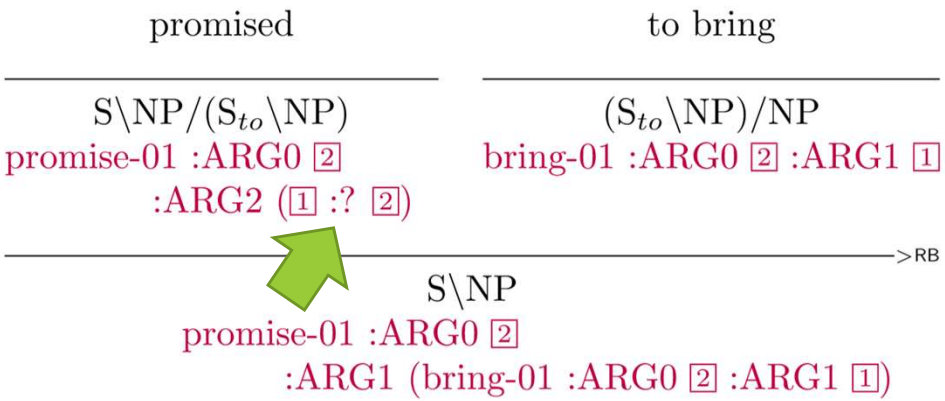
general form



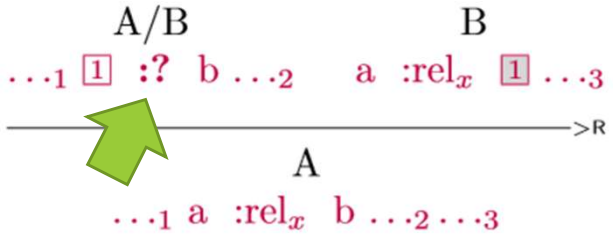
Procedure: Substitute a *relation* instead of a node

Relation-wise Application (<R or >R), Relation-wise Composition (<RB or >RB)

example

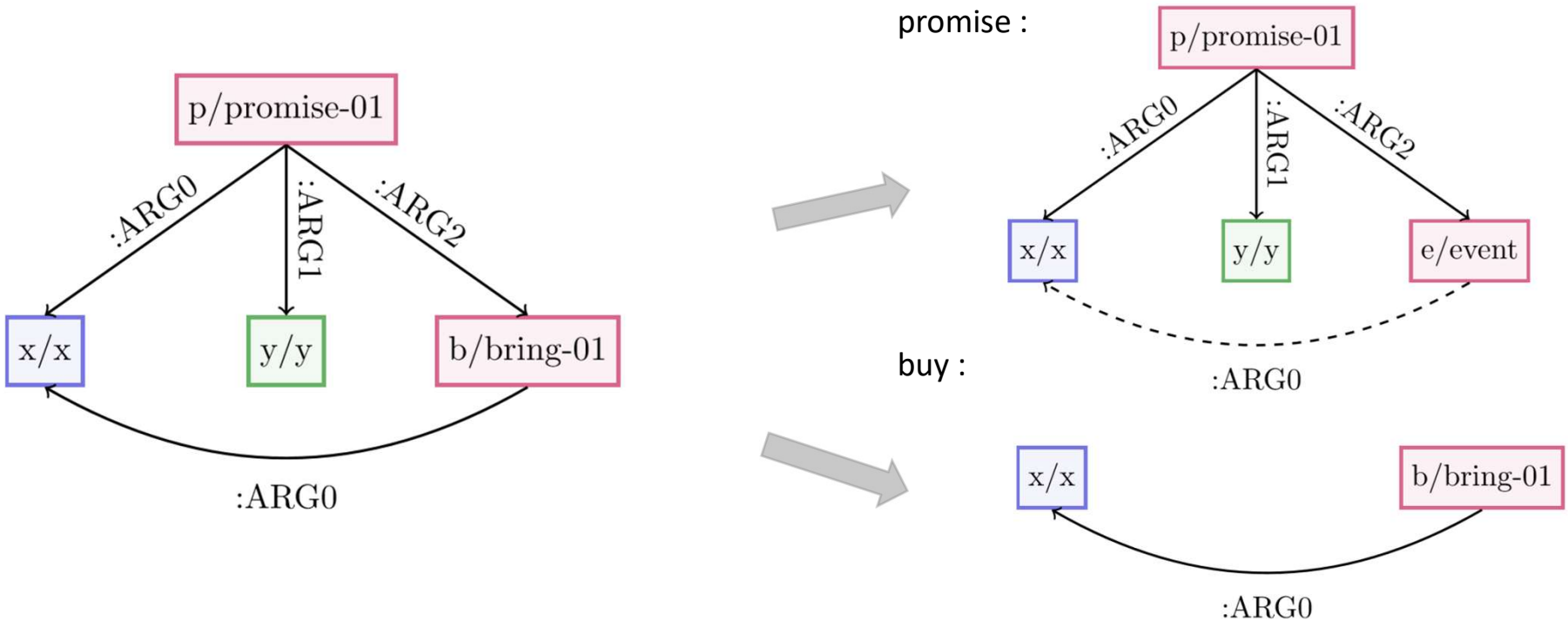


general form

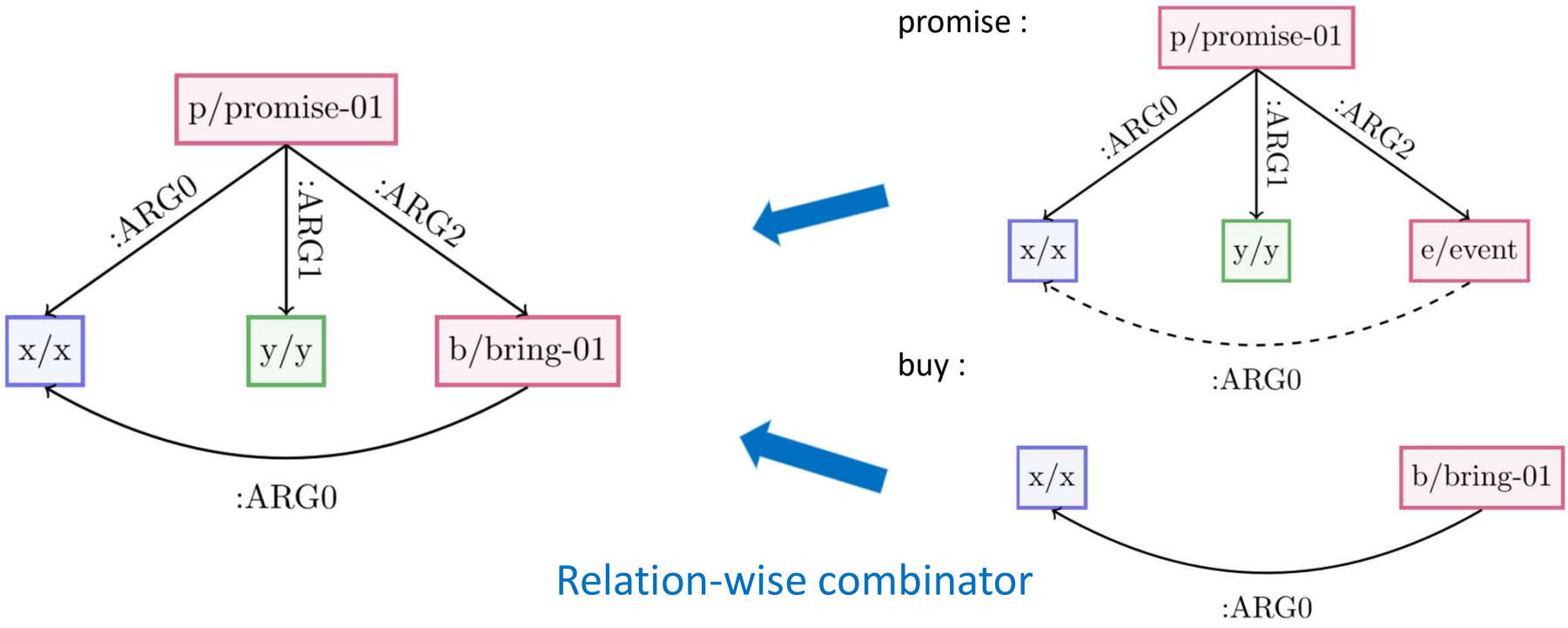


Procedure: Alternatively, use an underspecified edge **:?**

Decomposing AMR



Relation-wise derivation



Relation-wise combinator

Uses of Relation-wise Composition

We use relation-wise combinators for:

- *Control*
- *Raising*
- *Wh-questions*
- *Relative clauses*
- *Type raising*
- *Eventive Nouns*

Type Raising (<T or >T)

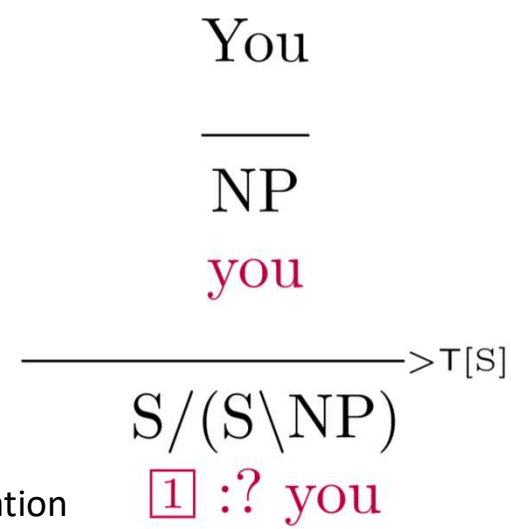
example

general form

Procedure: Construct a new root which is a free variable

Type Raising (<T or >T)

example



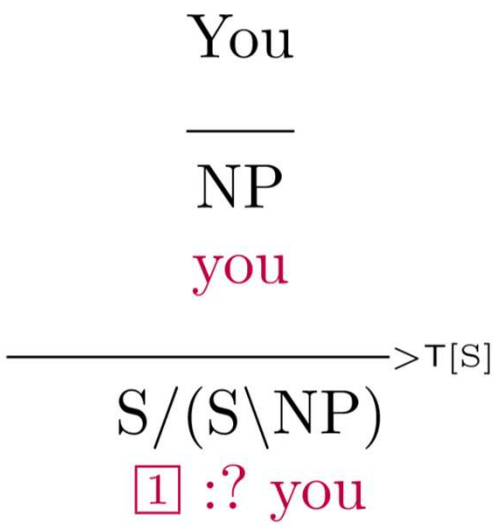
For example in complex coordination

general form

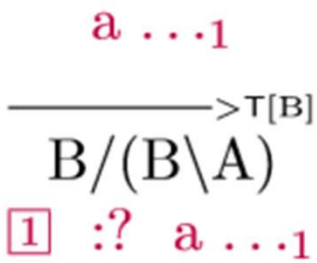
Procedure: Construct a new root which is a free variable

Type Raising (<T or >T)

example



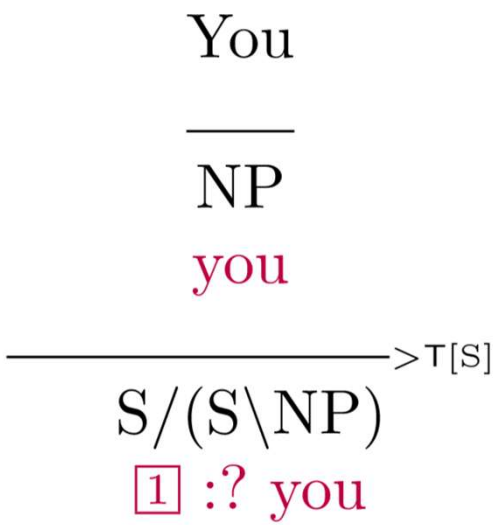
general form



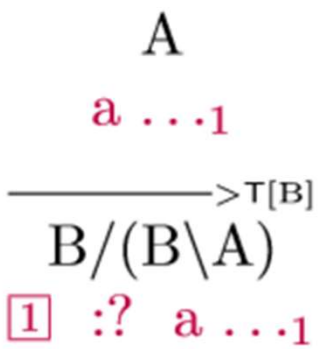
Procedure: Construct a new root which is a free variable

Type Raising (<T or >T)

example



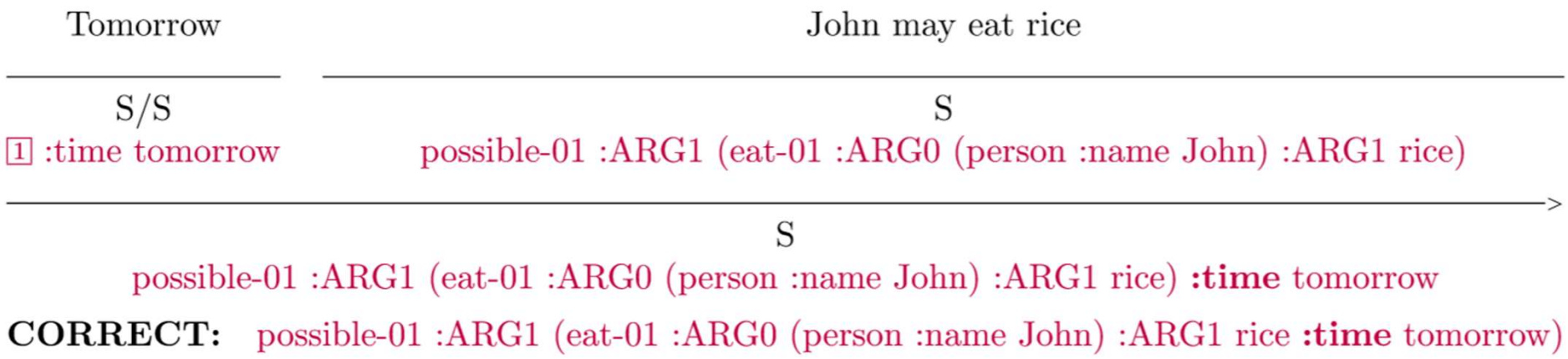
general form



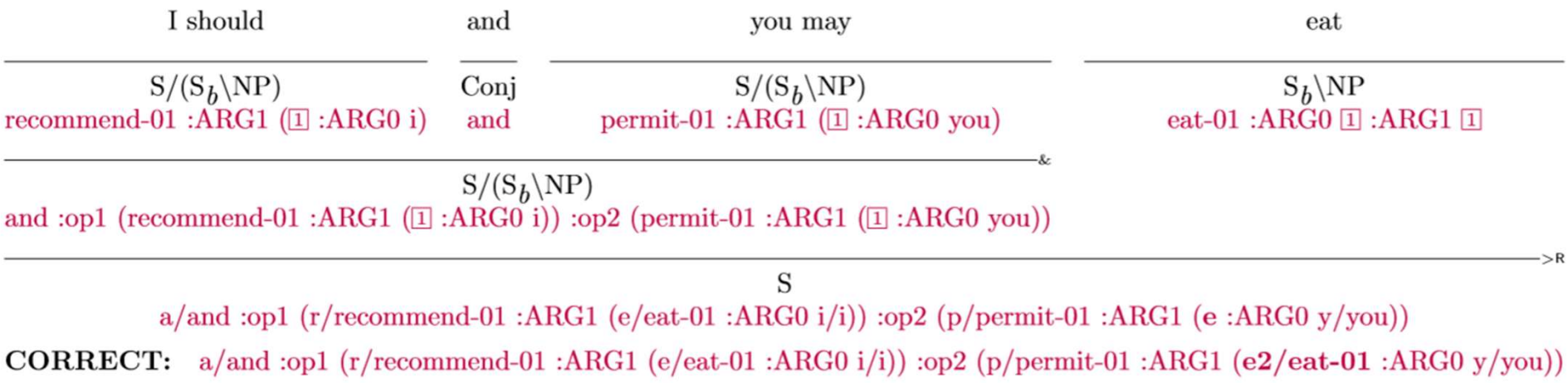
fill edge with relation-wise combinator

Procedure: Construct a new root which is a free variable

Unsolved: Modifying Modals



Unsolved: Coordination



Contributions & Next Steps

Contributions:

Contributions & Next Steps

Contributions:

Novel CCG Combinators for AMR Semantics.

Contributions & Next Steps

Contributions:

Novel CCG Combinators for AMR Semantics.

Formal and Interpretable approach to make AMR compositional, and thus more linguistically meaningful.

Contributions & Next Steps

Contributions:

Novel CCG Combinators for AMR Semantics.

Formal and Interpretable approach to make AMR compositional, and thus more linguistically meaningful.

Next Steps:

Contributions & Next Steps

Contributions:

Novel CCG Combinators for AMR Semantics.

Formal and Interpretable approach to make AMR compositional, and thus more linguistically meaningful.

Next Steps:

CCG-friendly English-AMR alignments

Contributions & Next Steps

Contributions:

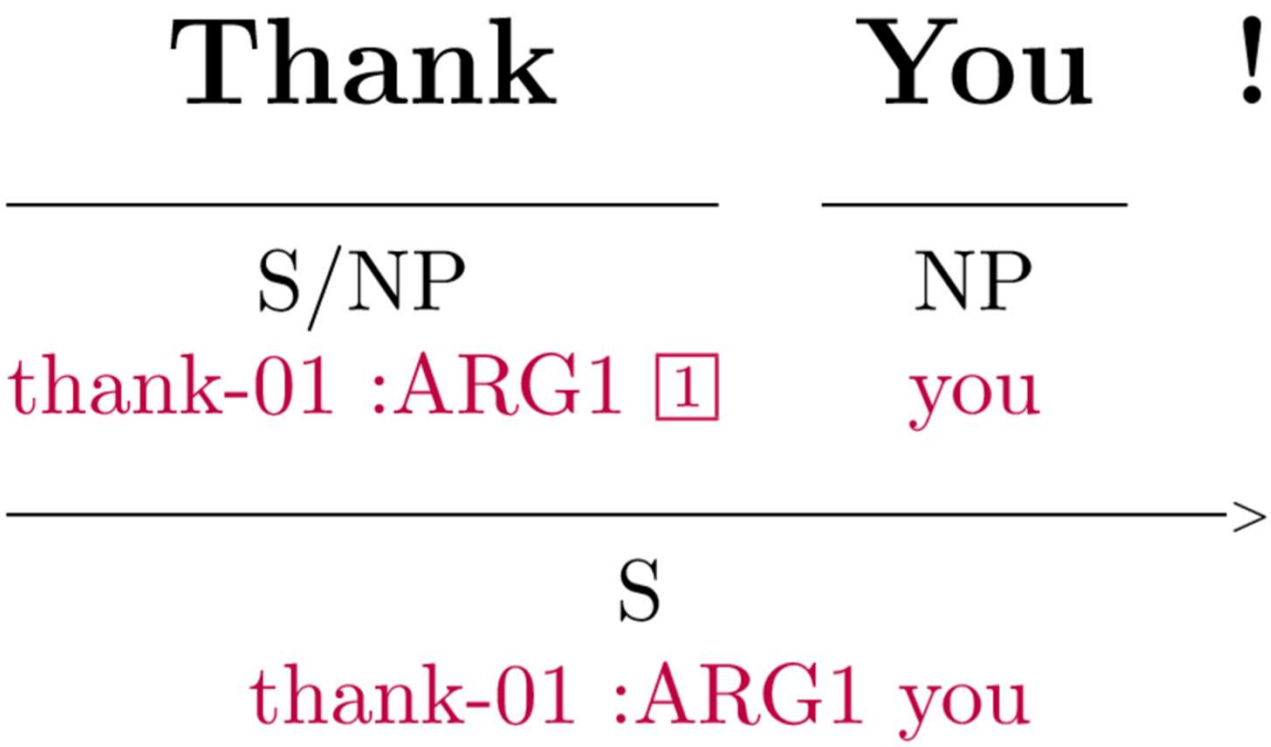
Novel CCG Combinators for AMR Semantics.

Formal and Interpretable approach to make AMR compositional, and thus more linguistically meaningful.

Next Steps:

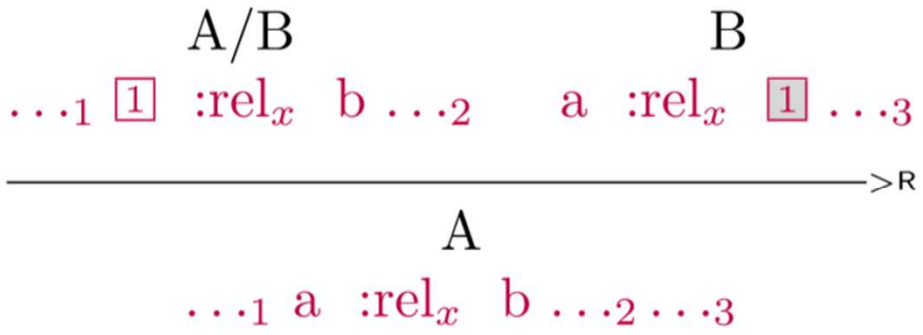
CCG-friendly English-AMR alignments

Automatic Aligner


Thank
You
!
S/NP
thank-01 :ARG1 1
NP
you
>
S
thank-01 :ARG1 you

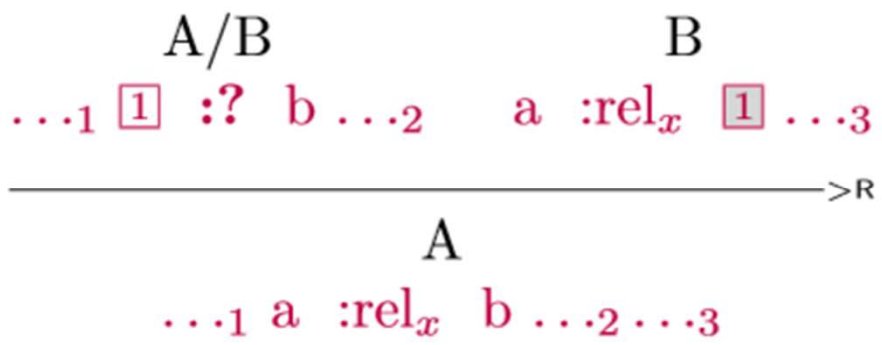
Appendix: Relation-wise case 1 vs. 2

1) Same edge on both sides



Constrains which constituents can combine (heuristically or with statistics). Possibly better at dealing with noise.

2) Underspecified (or free variable) edge



More general.
Captures the fact that syntactic information is only *correlated* with semantic roles.

Appendix: Type Raising (more detail)