Natural Language Inference with CCG Parser and Automated Theorem Prover for DTS

Asa Tomita

Ochanomizu University
Faculty of Advanced Science
https://morning85.github.io/

ESSLLI2025, Bochum
1 Aug (Fri)

1. Introduction

Who am I ?

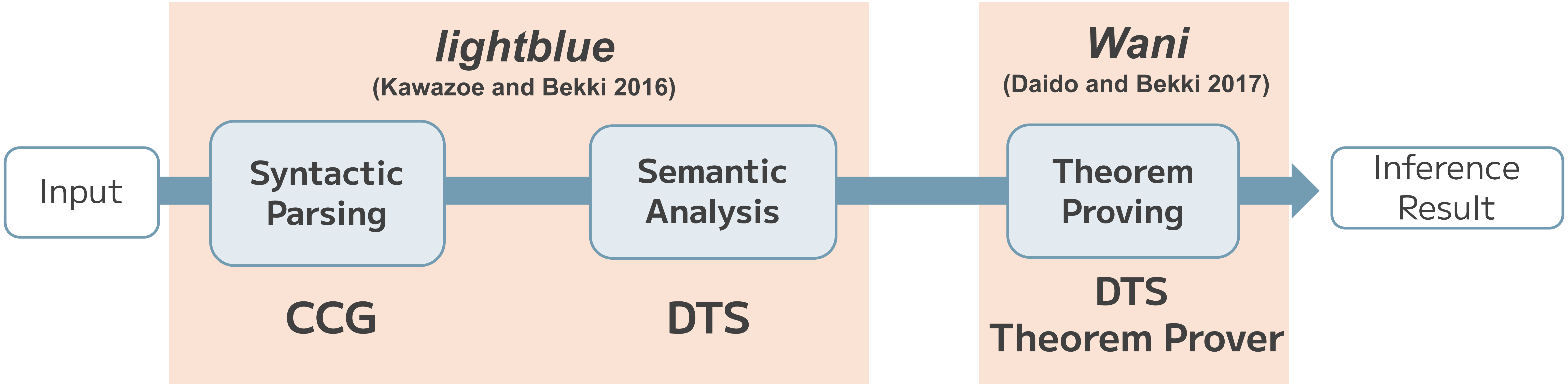
Asa Tomita **:** Ph.D. Student (1st year), Ochanomizu University
Major: Computer Science

Research Interest : Computational Linguistics

- Linguistic validity of Japanese CCG treebank
    - Guest Talk at **NALOMA** (Thu, August 7)
    - Oral Presentation at **Syntax Fest** (Fri, August 29)
- Development and improvement of Japanese CCG/DTS parser and Japanese Natural Language Inference System (← This talk)
    - Oral Presentation at **BriGap-2 Workshop, IWCS** (Wed, September 24)

Natural Language Inference Pipeline

- We propose an inference system based on CCG and DTS, in which we connect the CCG/DTS parser lightblue with the automated theorem prover Wani."



- This talk will cover the theoretical background, system design, and system evaluation.

2. Theoretical Background

Combinatory Categorial Grammar（CCG; Steedman 2000）

- a lexicalized grammar that describes syntactic structures using lexicon and combinatory rules
- Well-suited for computational implementation and empirical verification
- Parsing errors can be traced to specific lexical items and revised accordingly

lexical items

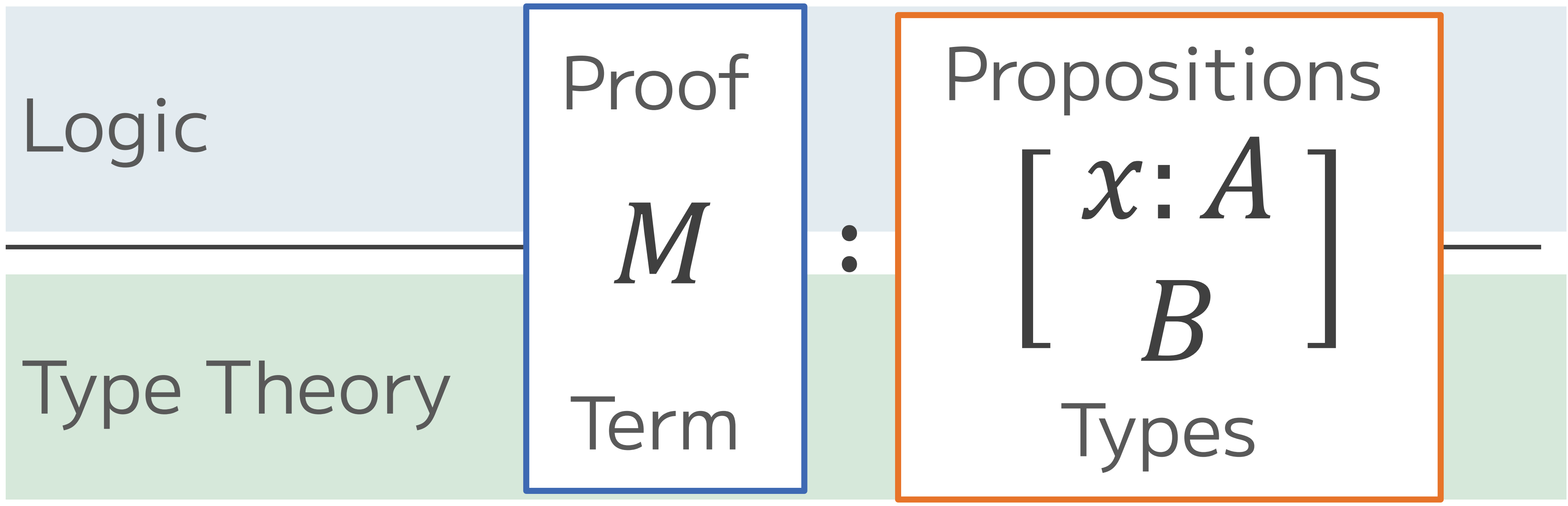
Keats $\vdash NP$
eats $\vdash (S \backslash NP)/NP$
apples $\vdash NP$

CCG Syntactic Structure

$$\dfrac{\dfrac{\text{Keats}}{NP} \quad \dfrac{\dfrac{\text{eats}}{(S \backslash NP)/NP} \quad \dfrac{\text{apples}}{NP}}{S \backslash NP}<}{S}>$$

Dependent Type Semantics (DTS; Bekki 2014, Bekki and Mineshima 2017)

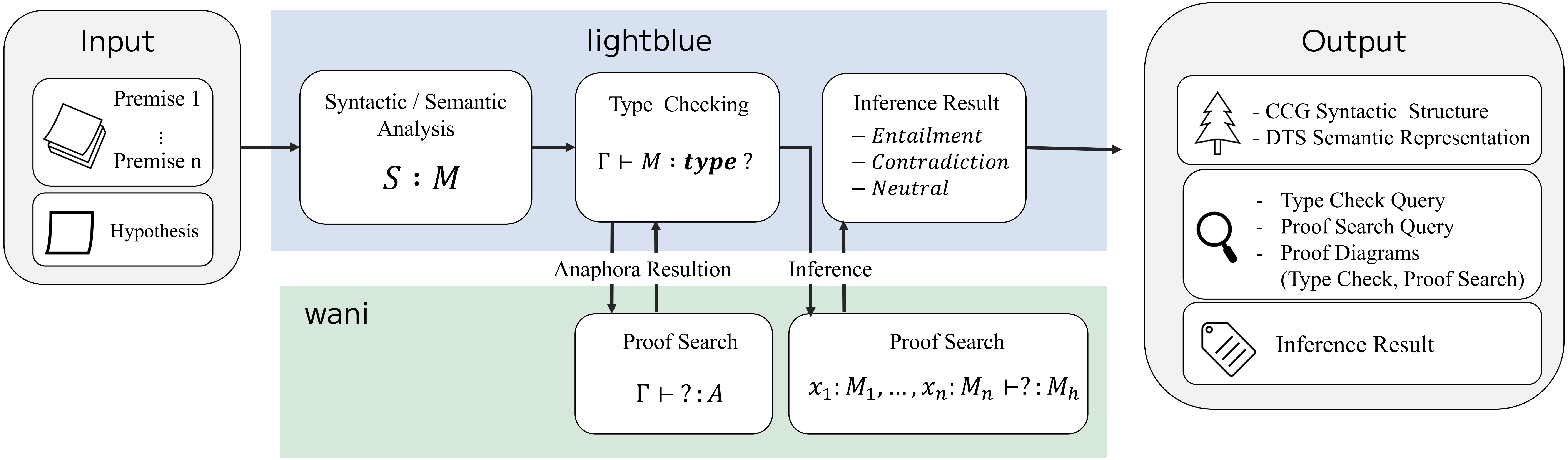
- Allows types (= propositions) to depend on terms (= proofs)
- Handles anaphora and presupposition via **proof search**
- **Type checking** ensures well-formedness of semantic representation



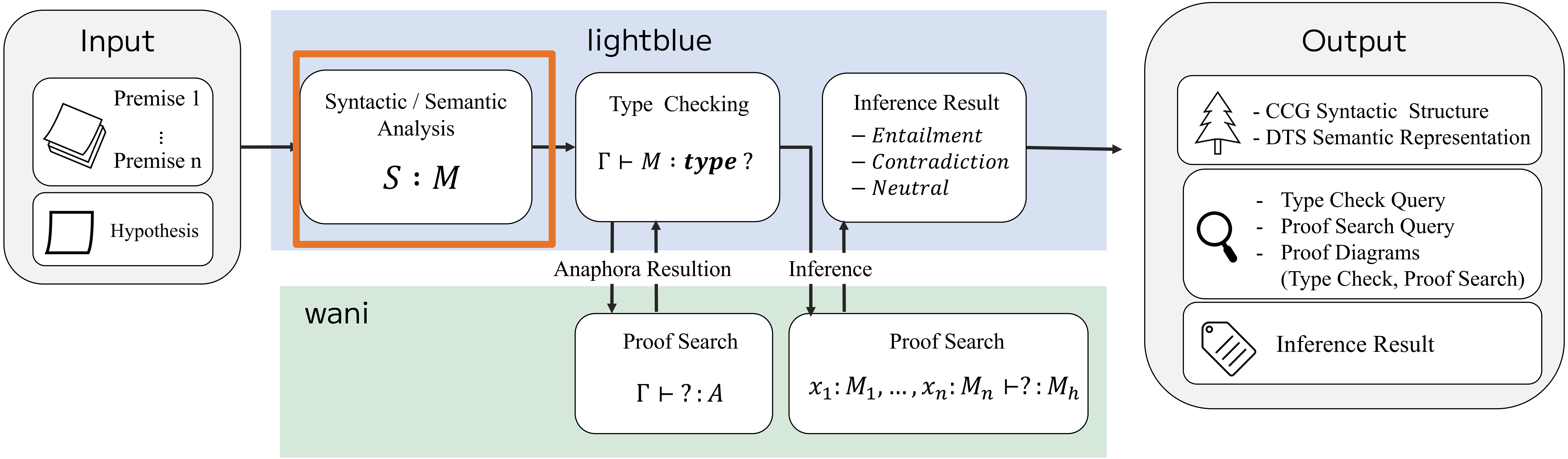
Curry-Howard Correspondence

3. Inference Pipeline

Overview of the inference pipeline



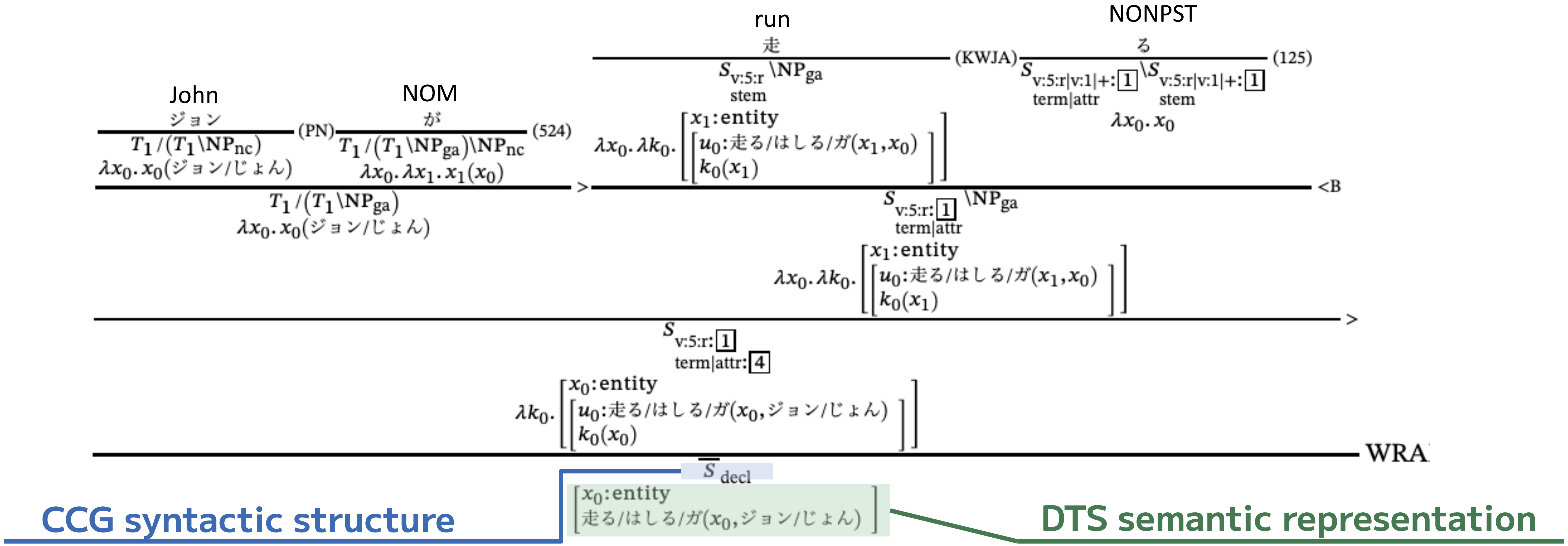
Syntactic/ Semantic Analysis



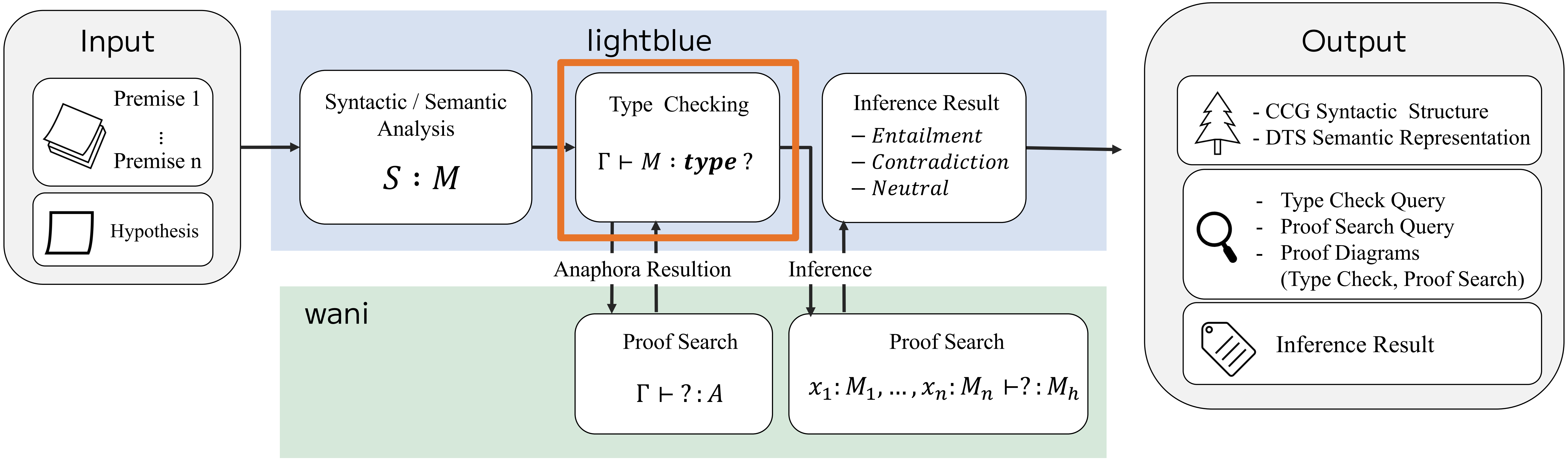
Syntactic/ Semantic Analysis

CCG/DTS parser : lightblue (Bekki and Kawazoe 2016)

When sentence is given as an input, lightblue outputs its CCG syntactic structure and DTS semantic representation.



Type Checking



Type Checking

CCG/DTS parser : lightblue (Bekki and Kawazoe 2016)

type checking is employed to verify whether a semantic representation obtained through semantic composition hold semantic felicity condition

semantic felicity condition (SFC)

A guarantee that the semantic representation of sentence has the type **type** in DTT

Proof Diagram of Type checking

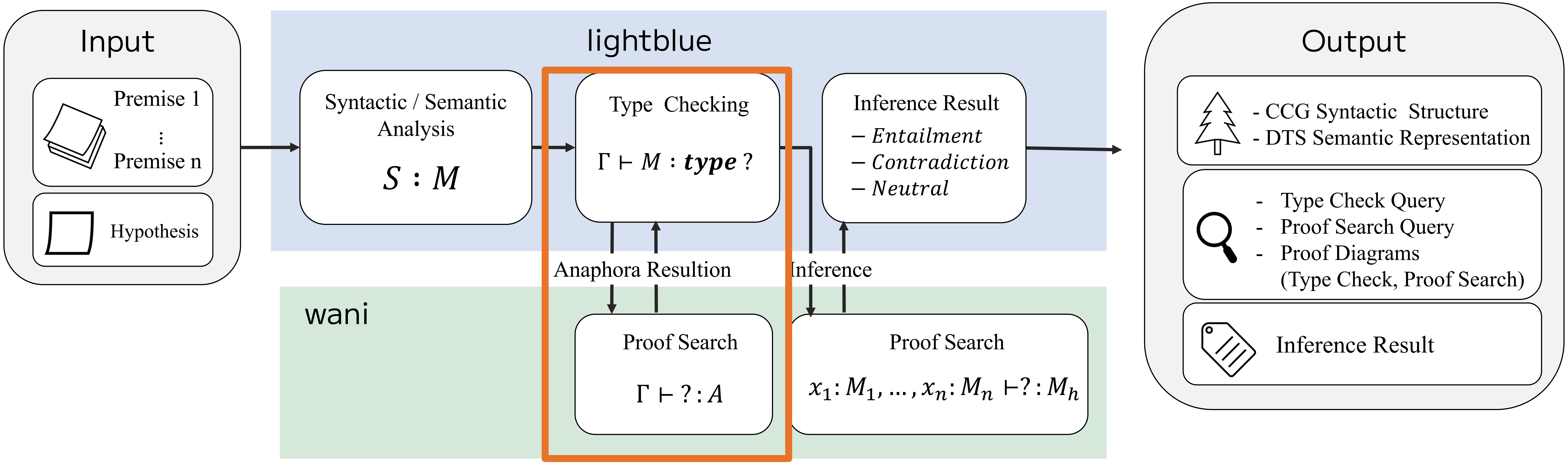
Type Check Query

$$\vdash \begin{bmatrix} x_0 : entity \\ run\ (x_0, taro) \end{bmatrix} : type\ ?$$

Proof Diagram

$$
\dfrac{
\dfrac{}{\vdash \text{entity}:\text{type}}\,(\{\}\text{F})
\qquad
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{s_0:\text{entity} \vdash \text{走る/はしる/ガ}:(x_0:\text{entity}) \to (x_1:\text{entity}) \to \text{type}}\,(\text{Con})
\quad
\dfrac{}{s_0:\text{entity} \vdash \text{太郎/たろう;太郎/たろう}:\text{entity}}\,(\text{Con})
}{s_0:\text{entity} \vdash \text{走る/はしる/ガ}(\text{太郎/たろう;太郎/たろう}):(x_0:\text{entity}) \to \text{type}}\,(\Pi\text{E})
\quad
\dfrac{}{s_0:\text{entity} \vdash s_0:\text{entity}}\,(\text{Var})
}{s_0:\text{entity} \vdash \text{走る/はしる/ガ}(s_0,\text{太郎/たろう;太郎/たろう}):\text{type}}\,(\Pi\text{E})
\quad
\dfrac{}{s_0:\text{entity}, s_1:\text{走る/はしる/ガ}(s_0,\text{太郎/たろう;太郎/たろう}) \vdash \top:\text{type}}\,(\text{TF})
}{s_0:\text{entity} \vdash \text{走る/はしる/ガ}(s_0,\text{太郎/たろう;太郎/たろう}):\text{type}}\,(\Sigma\text{F})
}{\vdash \begin{bmatrix} x_0:\text{entity} \\ \text{走る/はしる/ガ}(x_0,\text{太郎/たろう;太郎/たろう}) \end{bmatrix}:\text{type}}\,(\Sigma\text{F})
$$

Anaphora Resolution with lightblue



Anaphora Resolution with lightblue

Premise1: A man entered.

Premise2: He whistled.

$$v : \begin{bmatrix} u : \begin{bmatrix} x : entity \\ man\ (x) \end{bmatrix} \\ enter(\pi_1 u) \end{bmatrix} \vdash \begin{bmatrix} w@\begin{bmatrix} x : entity \\ male(x) \end{bmatrix} \\ whistle\ (\pi_1 w) \end{bmatrix} : type\ ?$$

Type Check Query

Anaphora Resolution with lightblue

Premise1: A man entered.

Premise2: He whistled.

$$v : \begin{bmatrix} u : \begin{bmatrix} x : entity \\ man\ (x) \end{bmatrix} \\ enter(\pi_1 u) \end{bmatrix} \vdash \begin{bmatrix} w@\begin{bmatrix} x : entity \\ male(x) \end{bmatrix} \\ whistle\ (\pi_1 w) \end{bmatrix} : type$$

$$v : \begin{bmatrix} u : \begin{bmatrix} x : entity \\ man\ (x) \end{bmatrix} \\ enter(\pi_1 u) \end{bmatrix} \vdash whistle\ (\pi_1 \pi_1 v) : type$$

Proof search with
automated theorem prover Wani

Underspecified types are resolved by rewriting them using the proof terms obtained through proof search.

Automated theorem prover: Wani

- Wani is an automated theorem prover for DTS
  Input：set of premises and a hypothesis
  Output : DTT proof diagram

- Wani conducts proof search by combining forward and backward reasoning

  Forward reasoning ：expanding the consequences by applying elimination rule to proposition contained in the premises
  Backward reasoning ：apply the rules to the conclusion first and calculate what is necessary to prove it

Forward and Backward Reasoning

Forward Reasoning

process of constructing a natural deduction-style proof tree from top to bottom.

$$\dfrac{M \;:\; \begin{bmatrix} x : A \\ B \end{bmatrix}}{\boxed{?} : A}$$

By applying the (ΣE) rule, $\boldsymbol{\pi_1(M)}$ is obtained as a proof for " ? "

Backward Reasoning

process of constructing a proof tree from bottom to top.

$$\dfrac{A : type \qquad \begin{array}{c} \overline{x : A}^{\,i} \\ \vdots \\ M : B \end{array}}{\lambda x.M \;:\; (x : A) \to B}$$

By applying (ΠI) rule, it is determined that $\boxed{}$ is required to derive $\lambda x.M : (x : A) \to B$

Restriction of Wani

Proof search in Dependent Type Theory is undecidable

→ Introduce restrictions on proof search

1. **Time and depth limits**

   Parameters are set for computation time and the number of backward reasoning steps

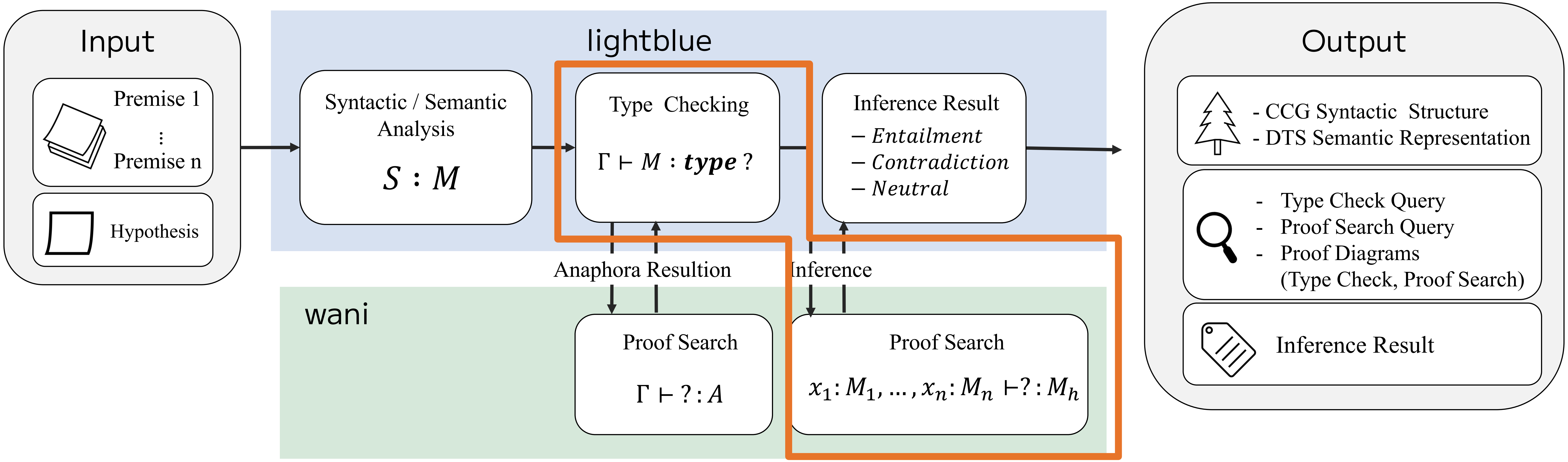
2. **Forward and backward reasoning**

   Forward reasoning is used for Σ-elimination , and backward reasoning is used for all other rules

3. **Pruning**

   Pruning is applied during certain backward reasoning steps

Proof search with Wani

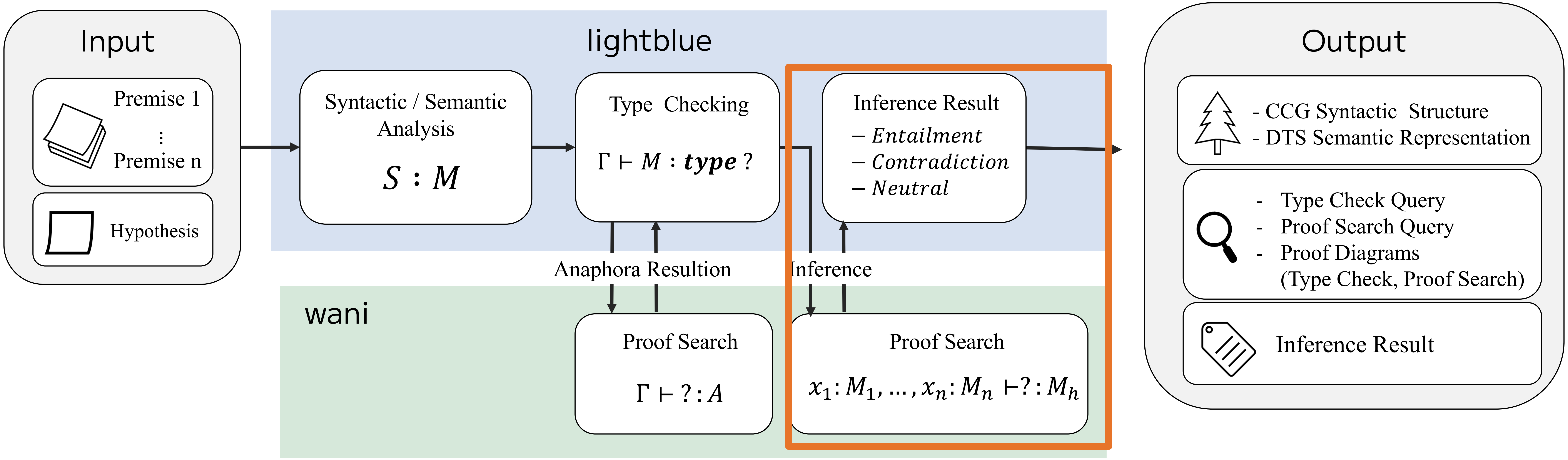


Proof search with Wani

- Wani checks whether a proof term of type M_h (the semantic representation of the hypothesis) can be constructed from the semantic representations of the premise sentences.
- If proof term is found, Wani returns a proof diagram as an output

$$t : \begin{bmatrix} u : \begin{bmatrix} v : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{girl}(x) \end{bmatrix} \\ \begin{bmatrix} y : \mathbf{entity} \\ \mathbf{thesis}(y) \end{bmatrix} \end{bmatrix} \\ \mathbf{write}(\pi_1\pi_1 u, \pi_1\pi_2 u) \end{bmatrix} \vdash \boxed{?} : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{girl(x)} \end{bmatrix}$$

Proof search with Wani



Proof search with Wani

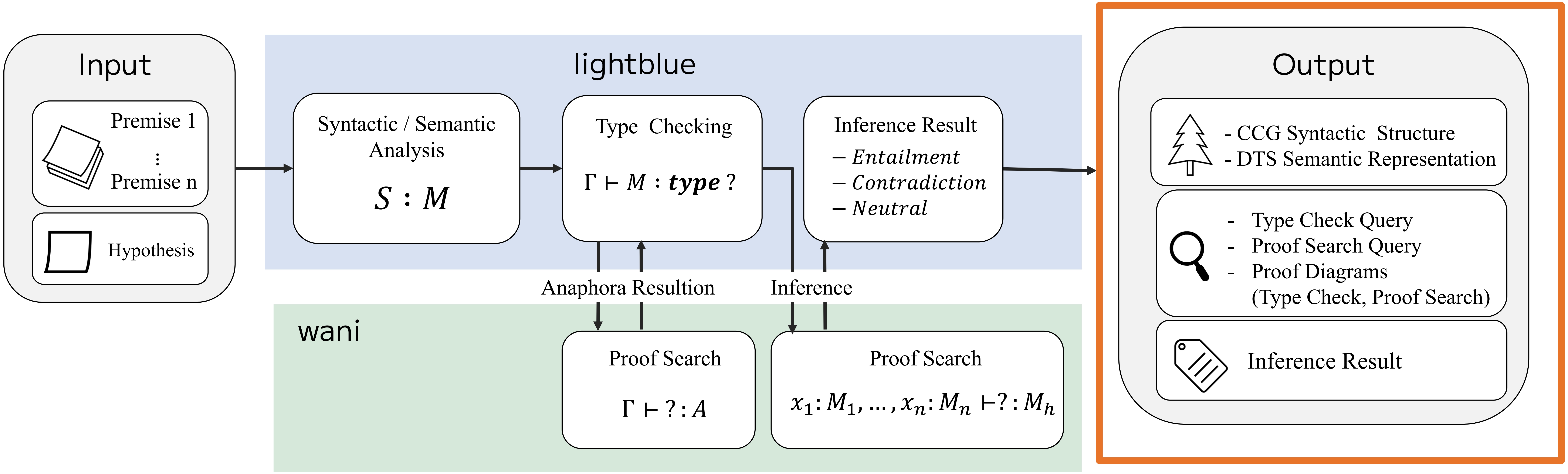
Based on the output from Wani, lightblue assigns one of three inference label

`yes` : A proof term of M_h is constructed (i.e., the hypothesis is entailed)

`no` : A proof term of type $\neg M_h$ is constructed (i.e., contradiction)

`unknown` : No proof term is constructed.

Output of an inference system



4.Evaluation Experiment

Dataset : JSeM

- JSeM (Kawazoe et al., 2015)2 is a dataset for Japanese Natural Language Inference
- Each problem consists of premises, a hypothesis, an inference label (yes, no, unknown, undef*) and is organized into sections categorized in accordance with linguistic phenomena.

* Unacceptable sentences.

Experimental Setup : ccg2lambda

ccg2lambda (Mineshima et al., 2015)

- A formal inference system grounded in CCG
- Uses semantic templates to construct higher-order logical forms
- Applies theorem proving with Coq to verify inference

Comparison between our system and ccg2lambda

	Our system (lightblue + Wani)	ccg2lambda
Syntactic Parsing	CCG (parser：lightblue)	CCG (parser：depccg)
- modification of the lexicon	✓ Easy to modify lexical items	⊖ needs retraining using valid treebank
Semantic Analysis	DTS （parser：lightblue)	Higher-order Logic
- Anaphora resolution	✓	✗
- Consistency of the semantic representation	✓ SFC can be checked with Type checking	✗ No SFC
Theorem Prover	Wani	Coq
- Output of proof diagrams	✓	✗

Experimental Setup

Evaluation Target

36 problems in the Verb section of JSeM

Metrics

- Accuracy
- Parsing success rate :
  Proportion of problems for which a complete syntactic structures are obatained
- Type checking success rate：
  Measures whether semantic representations are well-typed

Result

- Outperformed ccg2lambda in accuracy, recall, precision, and F1
- Higher precision, recall, and F1 than GPT-4o

Key difference from GPT
- returns "yes(entailment) " only when a proof term is constructed
→ Ensures formal verification rather than guesswork
→ Inference is treated as proof construction, not mere classification

System	GPT 4o	ccg2lambda	**Our System**
Parsing	-	-	0.90
Type Check	-	-	1.0
Accuracy	0.750	0.556	0.667
Precision	0.287	0.172	**0.397**
Recall	0.333	0.250	**0.342**
F1	0.308	0.204	**0.319**

Table 1: Performance Comparison with Other Systems

Error Analysis – External knowledge

P：ITELは1988年から1992年までAPCOMを所有していた。

(ITEL owned APCOM from 1988 to 1992.)

H：ITELは1990年にAPCOMを所有していた。

(ITEL owned APCOM in 1990.)

Ground Truth：Yes

Prediction　：Unknown

It requires temporal world knowledge that 1990 falls within the range from 1988 to 1992 - which is not explicitly encoded in the system.

Error Analysis – Parsing Error

太郎は	次郎から	花子を	紹介さ	れ	た
Taro-wa	Jiro-kara	Hanako-o	shokaisa	re	ta
Taro-NOM	Jiro-from	Hanako-ACC	introduce	passive	PST

(Taro was introduced to Hanako by Jiro)

the parser failed to correctly recognize *kara* (“from”) in the passive construction as semantically corresponding to the dative in the active counterpart.

Conclusion

- Proposed a linguistically-motivated NLI system, integrating syntactic parsing, semantic composition, type checking, and proof search.
- Achieved improved inference accuracy over existing systems.
- This system will contribute to refining and verifying theoretical assumptions