Yesod による日本語推論システム lightblue の開発環境の構築

佐伯小遥　富田朝　松原舞　戸次大介
お茶の水女子大学
{saeki.koharu，tomita.asa，matsubara.mai，bekki}@is.ocha.ac.jp

概要

本論文では，プログラミング言語 Haskell の Web アプリケーションフレームワーク Yesod にて実装された，日本語自動推論システム lightblue の開発環境 express を提案する．本環境は，lightblue の推論フロー（統語・意味解析，型検査，および証明探索）に対話的な操作性を導入したものである．また，事前計算およびキャッシュ機構を取り入れることで，特定条件下での推論の検証を可能にし，対話的な試行錯誤に基づく開発作業の効率化に貢献する．

1　はじめに

理論言語学に基づく自然言語推論は，統語・意味解析を明示的に行い，推論の妥当性や説明可能性を保証できる点で重要である．特に，形式理論に基づく推論システムは，照応解決や前提束縛といった複雑な言語現象を厳密に取り扱うことを可能にし，人間の言語理解に関する理論仮説を計算機上で検証するための基盤を提供してきた．

1.1　日本語自動推論システム lightblue

lightblue[1, 2] は，理論言語学に基づく日本語自動推論システムである．統語理論には組合せ範疇文法 (Combinatory Categorial Grammar; CCG) [3, 4] を、意味理論には依存型意味論（Dependent Type Semantics; DTS）[5] をそれぞれ採用している．また，DTS の部分体系における定理自動証明器 wani[6, 7] と接続し，証明探索を行うことで，照応解決や前提束縛を含む複雑な言語現象の解析・自動推論を実現している．このように lightblue は，形式的に定義された統語・意味理論に基づき，高い説明性を備えた推論を行うことを特徴とする．

1.2　lightblue による推論の処理過程

本節では，lightblue による推論の処理過程を紹介する．

1.2.1　統語・意味解析

lightblue では，入力文に対して CCG に基づく統語解析を行い，詳細な統語情報を含む構造を導出する．解析には CYK チャートパーシングを用いるため，文法的に許容される複数の統語構造が得られる場合がある．

続いて，得られた各統語構造に対して，DTS に基づく意味合成が行われる．なお，DTS は依存型理論 (Dependent Type Theory; DTT) [8] に依拠する意味論の枠組みである．

1.2.2　型検査

合成された文の意味表示に対しては，その整合性を確認するための型検査が行われる [9]．型検査では，文の意味表示が DTT における type 型を持つかどうかが検証される．

意味表示に未指定項が含まれている場合には，証明探索を通じた照応解決が同時に行われる．このとき，複数の証明項が得られる場合があり，型検査の結果は複数に分岐しうる．また，入力文が複数文からなる場合には，先行文の意味表示が後続文の型検査における文脈として与えられる．この過程においても，未指定項の解決により分岐が生じるため，文脈の分岐と照応解決の分岐が組み合わさることで，型検査の結果は文ごとに乗算的に増加する．

1.2.3　証明探索

すべての文に対する型検査が成功した後，証明探索が行われる．具体的には，前提文の意味表示から，仮説文の意味表示の型を持つ証明項が構成可能かどうかを判定する．証明項が見つかった場合には *Yes*，意味表示の否定への証明項が見つかった場合には *No*，いずれも得られない場合には *Unknown* という推論結果を出力する．

この過程において出力される証明図は，推論結果の妥当性を明示的に示すものであり，lightblue の説明性を支える重要な要素となる．

1.2.4　解析結果の構造的特徴

以上のように，lightblue による推論は，統語・意味解析，型検査，証明探索という複数の段階から構成され，各段階において理論的に許容される複数の候補を保持しながら進行する．特に，統語構造の曖昧性と，文脈依存的な型検査・証明探索が組み合わさることで，推論過程全体としては，図 1 に示すように，多数の分岐を伴う構造となり，考慮すべき推論候補の組合せは文が進むにつれて急激に増大する．探索回数は，k 個の入力文に対し，各文で i 通りの統語構造と各統語構造あたり j 通りの型検査結果を独立に選ぶため，最大 $2 \cdot (i \cdot j)^k$ 回となる．

1.3　対話的推論実行の必要性

lightblue のような分岐を伴う推論に対し，従来の実行方式では，可能なすべての解析結果や推論結果を網羅的に計算してから出力する枠組みが採用されていた．以下では，この方式を「全列挙型の実行方式」と呼ぶ．

しかし，この方式には大きく分けて二つの問題がある．第一に計算コストの問題である．最終的に関心のない推論経路についても一律に計算を行う必要があるため，特定の条件下での推論結果に到達するまでに多大な計算と時間を要する．第二に検証の困難さである．推論結果は複数の統語構造や型検査結果に基づく推論経路を混在させた形で提示されるため，意図しない分岐に基づく結果が同時に含まれてしまい，特定の分岐に着目して検証したり，推論の途中段階を柔軟に検証することは困難である．

また，実際の文法開発の場面では，研究者や開発者が求めているのは，可能なすべての推論結果を一度に得ることだけではない．例えば，入力文のある一文に対して特定の統語構造を仮定して，その選択が後続の型検査や証明探索にどのような影響を与えるかを確認する場合や，文法規則や語彙項目に修正を加えた後に，その変更が統語構造に反映されているかを検証する場合，推論を一括して完了させるのではなく，統語解析，型検査，証明探索といった各段階において，文法開発者が注目する分岐を選択しながら推論を進められることが重要となる．すなわち，推論過程を文法開発者との相互作用を通じて実行される対話的なプロセスとして捉え直す必要がある．

従来手法では，このような対話的な推論の実行を前提とした支援は試みられていなかった．本研究では，理論言語学に基づく推論の厳密性を保ちつつ，ユーザとの対話を通じて推論過程の探索と分岐選択を段階的に制御可能とする開発環境を提案する．

2　先行研究

著者らは，推論システム lightblue の開発環境構築に向けた第一段階として，lightblue の統語・意味解析および単文に対する型検査結果に着目した静的可視化ツール express （以下，プロトタイプ版 express）を提案した [10]．主な機能としては，図 2 に示す+ / - ボタンによる統語構造の展開/折り畳み操作や，複数の統語構造のうち，型検査に失敗したものを明示的に表示する仕組みがある．これらにより，文法開発者は，調査対象の部分統語構造，型検査失敗時の統語構造に効率的に到達することが可能となった．

一方で，プロトタイプ版 express は，lightblue による推論全体を対象としたものではなく，統語・意味解析および単文に対する型検査結果に焦点を当てた静的な可視化ツールであった．また，lightblue 本体とは独立したプロトタイプとして実装されていたため，推論過程における複数の分岐を比較可能な形で提示したり，特定の分岐を仮定した上で推論を段階的に進めるといった，推論過程を文法開発者との相互作用のもとで実行・検証するための枠組みは，依然として未整備であった．

3　提案手法

本研究では，先行研究で提案した静的可視化ツール プロトタイプ版 express を基盤として，これを lightblue に統合し，統語解析・型検査・証明探索からなる推論過程全体を対話的に制御可能とする開発環境として再構築する．本研究の目的は，単に推論結果を事後的に可視化することではない．推論過程に内在する「探索」と「分岐選択」を，文法開発者が逐次的に操作可能な対象として明示化し，対話的な検証環境を提供することである．

本節では， express の設計方針・実装要点・主要機能について述べる．

3.1　express の設計

本研究で提案する開発環境 express は，日本語自動推論システム lightblue をバックエンドに持つ Web ベースの開発環境として設計されている．本環境の設計上の特徴は，lightblue における推論を，一括実行される処理としてではなく，分岐を伴う探索過

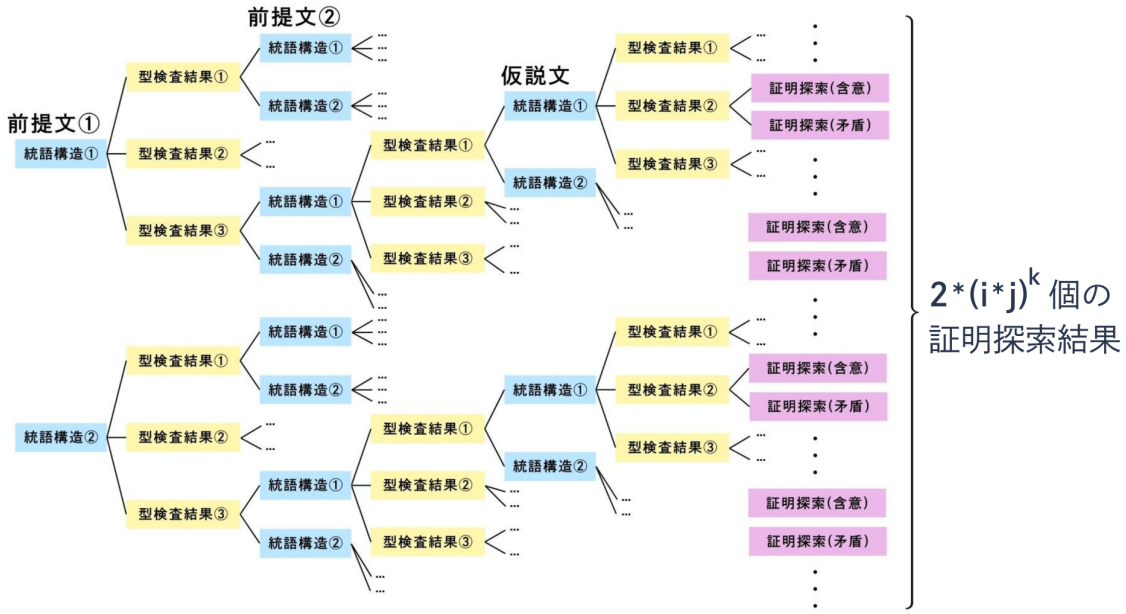

図 1　統語解析・型検査・証明探索にまたがる lightblue の推論フロー（$i = 2$, $j = 3$, $k = 3$ の場合）

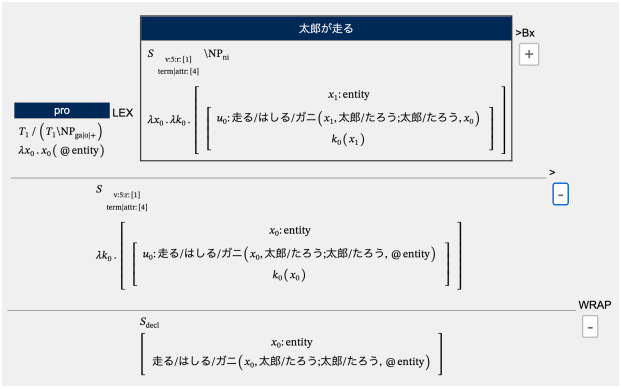

図 2　「太郎が走る」の統語構造の一部を折り畳んだ構造

程として捉え直し，その過程を文法開発者が段階的・選択的に操作可能な対象として再定義した点にある．

lightblue における推論は，統語解析，型検査，証明探索という複数の段階から構成され，各段階において理論的に許容される複数の候補を保持しながら進行する．従来の実行方式では，これらの候補をすべて列挙した上で推論を進める「全列挙型」の枠組みが採られていたが，本研究では，推論過程に内在する分岐点そのものを明示化し，それらを操作単位として定義することを設計の中心に据える．

具体的には，推論過程を以下の二種類の操作単位に分解した．第一に，統語解析段階における統語構造の選択である．CCG に基づく解析では，一つの文に対して複数の統語構造が与えられるが，本環境では，それらを並列に保持したまま，文法開発者が注目する統語構造を明示的に選択できるようにした．第二に，型検査段階における型検査結果の選択である．意味表示に未指定項が含まれる場合，照応解決の結果として複数の証明項と型検査の証明図が得られうるが，それらを個別の分岐として捉え，文脈としてどの結果を採用するかを指定可能とした．

図 3 に，express を用いてユーザが統語構造および型検査結果の分岐を選択しながら，段階的に推論を進める流れを示す．

このように，推論過程を分岐選択の列として再構成することにより，文法開発者は，関心のある条件下でのみ推論を進めることが可能となる．これにより，全列挙型の実行方式では困難であった特定条件下での詳細な検証や，分岐の影響を追跡する分析を実用的に支援する．

3.2　実装

提案する開発環境 express は，関数型プログラミング言語 Haskell の Web アプリケーションフレームワーク Yesod [11] を用いて実装されている．lightblue 本体が Haskell によって実装されているため，Yesod の採用により，lightblue の内部データ構造や推論のための関数を直接呼び出すことが可能となっている．

システム構成としては，サーバ側で推論の実行お

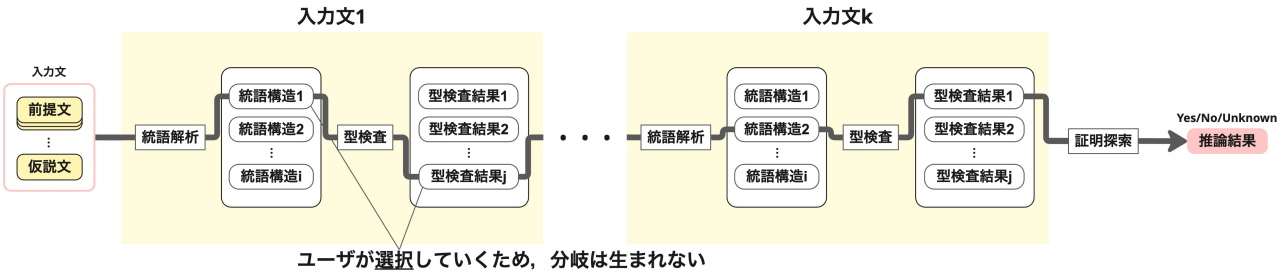

図 3　統語構造等の分岐をユーザが選択しながら推論を進める流れ

よび推論状態の管理を行い，クライアント側で，推論結果の可視化と分岐選択の入力を担当する．サーバ側で，統語解析・型検査・証明探索の各段階において生成される結果を段階的に保持し，ユーザの選択に応じて必要な部分のみを再計算する．これにより，全列挙型の実行方式に比べ，不要な推論経路の計算を避けることができる．

また，対話的操作における応答性を確保するため，推論結果を段階ごとに事前計算し，その結果をキャッシュとして保持する機構を導入した．同一の条件下で要求される推論については，キャッシュされた結果を再利用することで，文法開発者の操作に対する待ち時間を低減している．

3.3　主要機能

3.3.1　部分文字列解析

express は，入力文の一部を選択して解析を行う部分文字列解析機能を提供する．lightblue では，CYK チャートパーシングにより，文全体だけでなく部分文字列に対応する統語構造も内部的に保持されている．本機能では，この性質を利用し，文法開発者が関心を持つ部分文字列に対して，統語・意味解析結果を即座に確認できるようにした．例を付録図 5 に示す．

これにより，特定の部分文字列や現象に起因する問題を局所的に調査することが可能となる．

3.3.2　証明探索クエリのエクスポート

express は，また，選択された推論経路に対応する証明探索クエリをエクスポートする機能を備えている．

lightblue における証明探索クエリとは，前提文の意味表示から仮説文の意味表示の型を持つ証明項が存在するかどうかを判定する探索問題を指し，lightblue ではこれを定理自動証明器 wani に与えて証明探索を行う．

従来の開発環境では，理論的には成立するはずの推論が wani による証明探索に失敗した場合でも，その際の推論条件を再現・共有する手段が整備されておらず，証明探索クエリを手作業で再構築せざるを得なかった．

また，統語構造選択や型検査結果の分岐を含む lightblue の推論では，同一入力文に対して最大 $2 \cdot (i \cdot j)^k$ 通りの推論経路が存在するため，wani 側で問題となる推論を再現するためには，多数の探索経路を経由する必要があり，実質的な検証が困難であった．

提案環境では，文法開発者が対話的に選択した統語構造，型検査結果，および文脈分岐に基づき，wani に与えられる証明探索クエリを直接エクスポートし，またそれを wani 側からインポートすることができる．これにより，特定条件下でのみ生じる証明探索失敗を正確に切り出して共有でき，不要な探索を経ることなく問題の検証が可能となった．

4　おわりに

本論文では，日本語自動推論システム lightblue の開発環境として，推論過程全体を対話的に実行・制御可能とする Web ベースの環境 express を実装した．express は，統語解析・型検査・証明探索からなる lightblue の推論を，分岐を伴う探索過程として捉え直し，文法開発者が統語構造や型検査結果を段階的に選択しながら推論を進められる点を特徴とする．先行研究で提案した静的可視化手法に対し，本研究では推論過程そのものを操作・検証の対象として明示化し，lightblue に統合することで，既存の実行方式では困難であった特定条件下での推論の検証を可能とした．また，部分文字列解析や証明探索クエリのエクスポート機能により，問題箇所の局所的な調査や，証明探索失敗の再現性ある共有を実現した．

謝辞

本研究の一部は，JST CREST JPMJCR2565，および JSPS 科研費 JP23H03452 の助成を受けたものである．

参考文献

[1] Daisuke Bekki and Ai Kawazoe. Implementing variable vectors in a CCG parser. In Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016), pp. 52–67, Berlin, Heidelberg, 12 2016. Springer Berlin Heidelberg.

[2] Asa Tomita, Mai Matsubara, Hinari Daido, and Daisuke Bekki. Natural language inference with CCG parser and automated theorem prover for DTS. In Proceedings of the Workshop on the Bridges and Gaps between Formal and Computational, pp. 1–7, 2025.

[3] Mark Steedman. Surface Structure and Interpretation. The MIT Press, Cambridge, 1996.

[4] Mark Steedman. The Syntactic Process. MIT Press, 2000.

[5] Daisuke Bekki and Koji Mineshima. Context-passing and underspecification in dependent type semantics. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, Studies of Linguistics and Philosophy, pp. 11–41. Springer International Publishing, 2017.

[6] Hinari Daido and Daisuke Bekki. Development of an automated theorem prover for the fragment of dts. In the 17th International Workshop on Logic and Engineering of Natural Language Semantics (LENLS17), 2020.

[7] 大洞日音. DTS の部分体系を用いた定理自動証明器への等号型の導入. PhD thesis, お茶の水女子大学, 2022.

[8] Per Martin-Löf. Intuitionistic type theory, Vol. 9. Bibliopolis Naples, 1984.

[9] Daisuke Bekki and Miho Satoh. Calculating projections via type checking. In Proceedings of the TYTLES Workshop on Type Theory and Lexical Semantics, ESSLLI, 2015.

[10] 佐伯小遥, 富田朝, 戸次大介. 日本語推論システム lightblue の開発環境構築に向けて. 言語処理学会第 31 回年次大会 (NLP2025), 2025.

[11] Michael Snoyman. Developing Web Apps with Haskell and Yesod, Vol. 2. O'Reilly Media, 2015.

5　付録

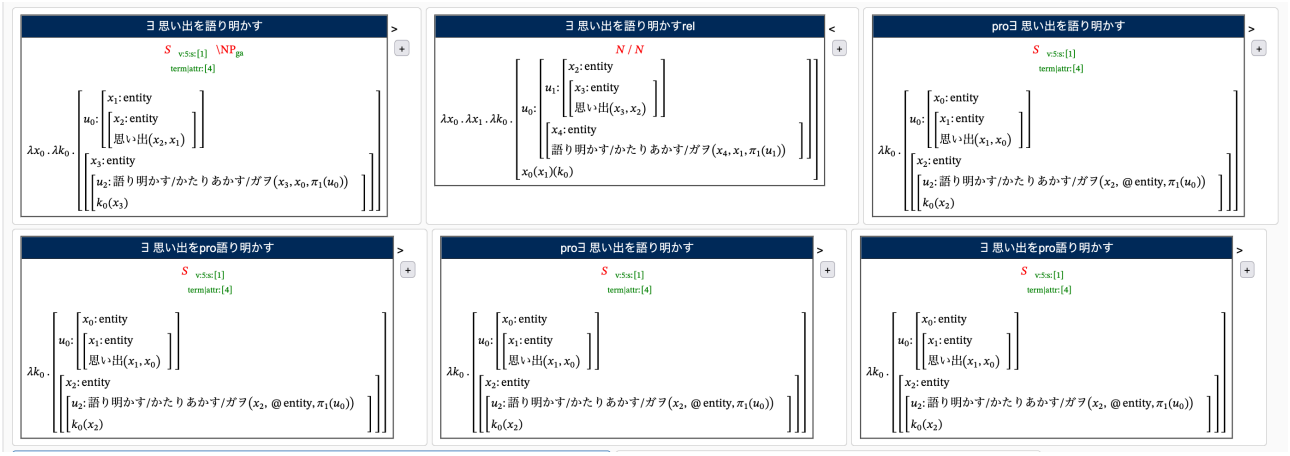

図 4　「太郎が夏の一夜を思い出を語り明かす」の部分文字列「思い出を語り明かす」に与えられる複数の統語構造

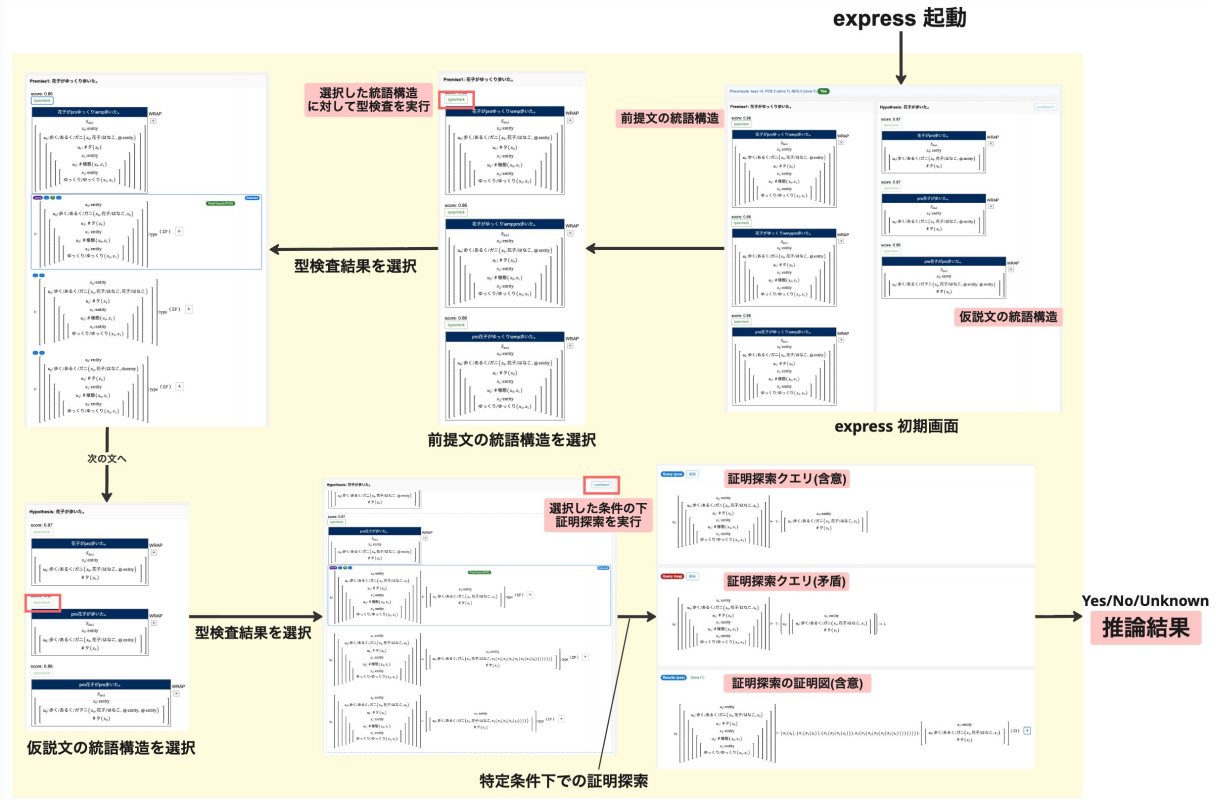

図 5　express における推論過程の操作フロー