自然言語処理 127－23
（1998. 9. 18）

HPSG パーザーの為の GUI

今井 久夫　宮尾 祐介　辻井 潤一
東京大学大学院理学系研究科情報科学専攻
{hisao,yusuke,tsujii}@is.s.u-tokyo.ac.jp

曖昧性のある構文解析結果をグラフィカルに表示する、HPSG パーザーのための GUI *will*を作成した。このインタフェースの主な機能は、構文木を木構造で表示する、素性構造を AVM 表記で表示するという 2 つである。マウスによる操作で簡単に構文解析結果の把握が行なえるため、*will*により文法の開発効率を向上させることができる。任意のパーザーから *will*を利用することができるように、パーザーと *will*とのプロトコルを定めた。

GUI for an HPSG Parser

Hisao Imai　Yusuke Miyao　Jun'ichi Tsujii
Department of Information Science, Graduate School of Science, University of Tokyo
{hisao,yusuke,tsujii}@is.s.u-tokyo.ac.jp

We developed *will*, a GUI for HPSG parsers, which graphically displays the result of syntactic analysis. Its interface has two major functions. One is to show the syntactic structure in tree form. The other is to show the corresponding feature structures in AVM format. These functions enable a user to scan the results with simple mouse operations, and thus significantly improve the efficiency of grammar development. We defined the communication protocol between *will* and a parser so that *will* can be used as a GUI for any parsers.

1　はじめに

我々は構文解析結果をグラフィカルに表示し、HPSG[1] を基にした文法開発を支援するシステム *will*を作成した。これにより、解析結果の把握が容易になるため、文法の開発効率の向上が期待できる。本稿では、本システムの特徴と主な機能を紹介する。

現在我々の研究室では HPSG を基にした、実用的なパーザー [4] 及び日本語文法 [8]、英語文法 [7] の開発を行なっている。現段階では、これらの日本語文法、英語文法には共に改良の余地があり、修正作業は頻繁に行なわれる。このときの開発効率は解析結果の把握の容易さに大きく依存する。

これまで我々には解析結果の把握が容易さを欠き効率的でないという問題があった。これは HPSG に基づく構文解析結果である素性構造が非常に大きく複雑なためであると考えた。このため 1) 木構造を用いて構文木を表示し、2) 木のノードに対応する素性構造をユーザが見やすい形で表示する機能を持つ HPSG パーザーの為の GUI として *will*を作成し、この問題を解決した。

2　*will*の特徴

*will*は、パーザーの入出力をユーザに利用しやすく提供する GUI で、構文解析を行なう文をユーザから受けとり、パーザーに解析させ、その結果を受け取り木構造や素性構造を表示する。

現在 *will*を用いて構文解析結果を表示させることができるプログラムには、我々の研究室で開発中の日本語文法 [8]、英語文法 [7] を用いて構文解析する HPSG パーザーがあり、*will*が実際に使用されている。このパーザーは我々の研究室で開発したプログラミング言語 LiLFeS[3] 上で動作する。LiLFeS は PROLOG などの論理型プログラミング言語の一種であり、型付き

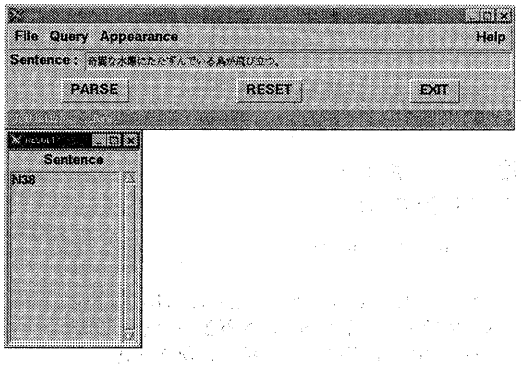

図 1: *will*に文を入力して構文解析させた直後の画面

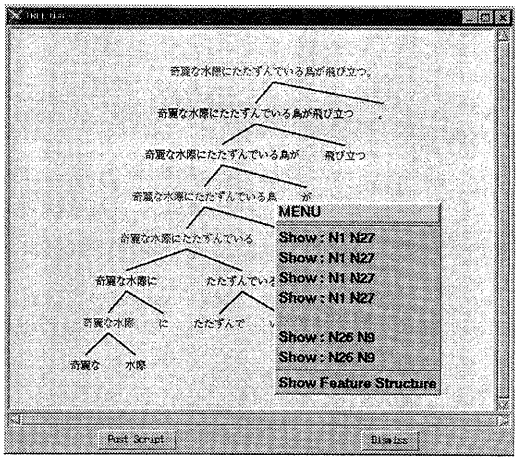

図 2: 図 1 の結果の 1 つの構文木を表示させた画面

素性構造の単一化を高速に行なえるように設計されている。

*will*とパーザーとの間のプロトコル (第 3.5 節参照) は特定のパーザーに依存していないため、そのプロトコルに対応する任意のプログラムから *will*を利用可能である。例えば、EDR コーパス [2] や GDA タグ [9] 付き文章などの構文解析済みコーパスを読むプログラムから、*will*を利用して構文木を表示することができる[1]。

今回 *will*を実装するにあたり、我々は日本語化された Tcl7.6/Tk4.2[5] を用いた。Tcl/Tk は様々なプラットフォームで動作するスクリプト言語であるため、移植や変更、拡張が容易であるという利点がある。

3 *will*の機能と動作

以下では、「奇麗な水際にたたずんでいる鳥が飛び立つ。」という文の構文解析を例に、*will*の機能を説明する。これは [8] を用いて行なっている。

3.1 入力と解析結果の表示

図 1 は文を入力し、構文解析を実行した直後の画面である。構文解析は、図 1 に示す画面の上側のウィンドウのエントリへ文を入力し、PARSE ボタンをクリックすることで行う。その結果得られる構文木の根にあたるノードの ID が図 1 の下側のウィンドウに表示される。複数の解が得られた場合はその数だけ ID が表示される。

構文解析の実行時に、スキーマの適用が失敗したノードを含めて全てのノードの ID を取得することもできる。このデータは図 1 の結果を表示するウィンドウに、根の ID と同様に表示される。しかし、ノードの数が多くなるとその情報の取得に時間が多くかかるため効率が悪くなってしまう。このため全てのノードの情報を取得するかどうかはオプションで設定できるようになっている。

3.2 構文木の表示

図 2 は図 1 のノード N38 に対応する構文木を表示させた画面である。構文木は、結果表示のウィンドウに表示された ID をダブルクリックすることで、そのノードを根とする木構造の形式で表示される。

この時表示される木は複数の解がある場合でもとりあえず 1 つを表示する。ノードから下にある解が複数かどうかはノードの色でわかるようになっている。

[1] EDR の解析済みコーパスについては、構文木を *will*で表示させるプログラムを作成した。

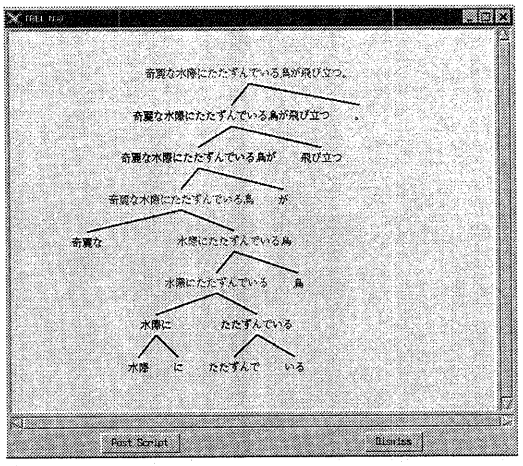

図 3: 図 1 の結果で図 2 とは異なる構文木を、表示内容を「string」、配置方法を「Top」で表示させた画面

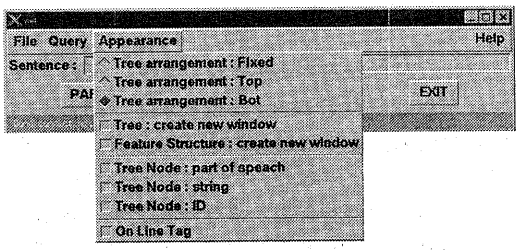

図 4: 木構造の表示形式のオプションを変更するメニュー

図 3 は、図 2 で「奇麗な水際にたたずんでいる鳥」を表すノード以下の部分木で、それとは異なる構文木を選択し、表示させた画面である。表示されている以外の構文木の表示は、曖昧性のあるノードを右クリックして図 2 にあるようなメニューを出し、その中から選択することにより行なう。曖昧性のあるノードとは左右の娘の組合せが複数ある親ノードのことである[2]。

*will*では、木構造の表示において、ノードと

[2]HPSG ではノードの情報として素性構造が割り当てられている。我々のパーザでは、異なる娘の組合せで等価な素性構造を持つノードは解析時に 1 つのノードにファクタリングされる。

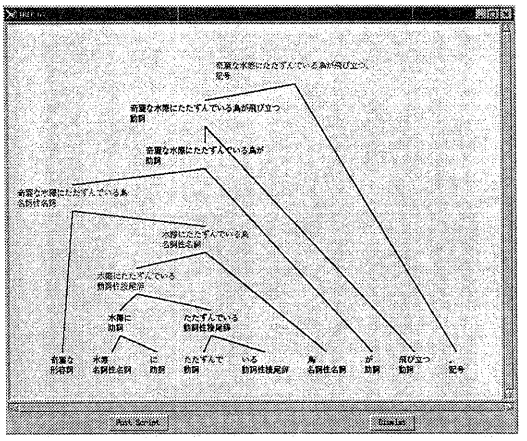

図 5: 図 3 と同じ構文木を、表示内容を「string」と「part of speach」、配置方法を「Bot」で表示させた画面

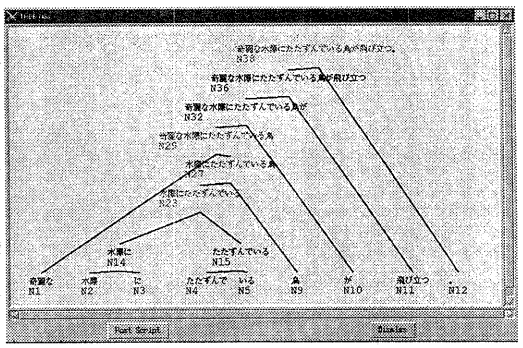

図 6: 図 3 と同じ構文木を、表示内容を「string」と「ID」、配置方法を「Fixed」で表示させた画面

して表示する内容とノードの配置方法とを、図 4 に示すメニューを使用してユーザが選択可能である。図 3、図 5、図 6 はどれも同じ解析結果を異なる表示方法を選択して表示させたものである。ノードとして表す内容はこれらの組合せ、配置方法はこの 3 種類から選択できる。配置方法が「Top」および「Bot」では表示するノードの大きさを動的に計算して配置するためノードが重なることがない。「Fixed」の場合は位置をノード間の相対的な関係のみから配置位

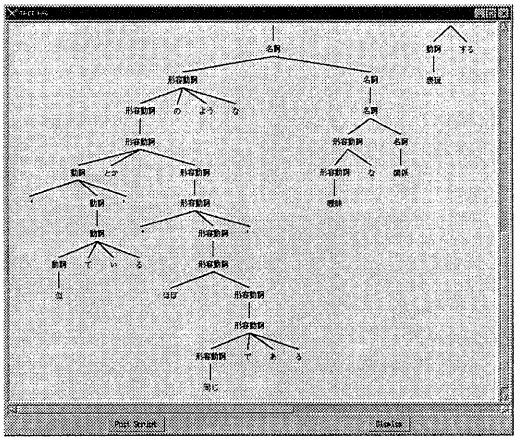

図 7: EDR コーパス中の 1 文の構造を表示させた画面

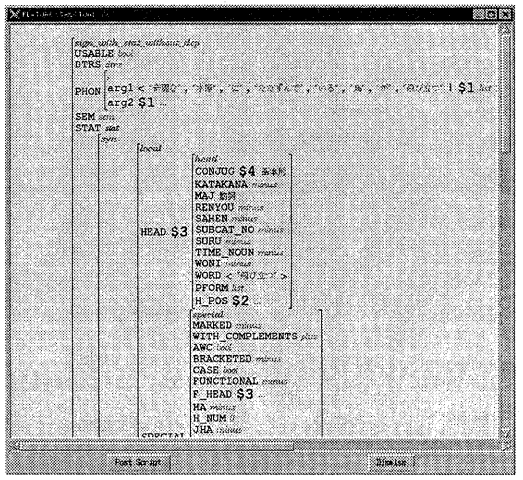

図 8: 図 6 のノード N36 に対応する素性構造を表示させた画面

置を決定する。

木構造の表示は 2 分木[3]だけでなく、木構造一般を表示できる。図 7 に EDR コーパス中の 1 文の統語構造を表示させた画面を示す。

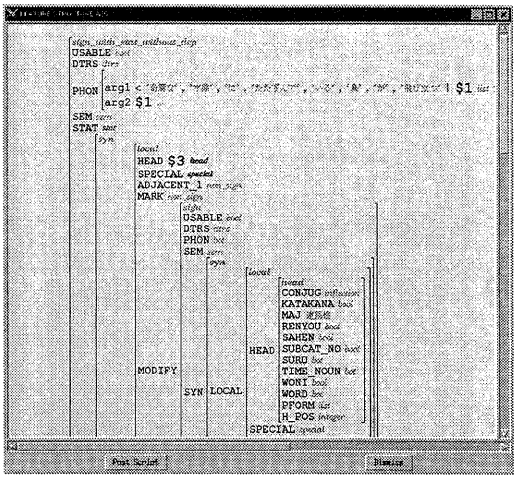

図 9: 図 8 と同じ素性構造を、素性 HEAD と SPECIAL 以下の構造を省略して表示した画面

3.3 素性構造の表示

図 8 は図 6 のノード N36 に対応する素性構造を表示させた画面である。構文解析結果の更に細かい情報が必要な時は、木構造中のノードをダブルクリックすることにより、そのノードに対応する素性構造を表示させることができる。素性構造は AVM 形式で表示され、素性、型、リスト、構造共有を表す要素が色分けされ、見やすくなっている。

HPSG の素性構造は非常に大きくなることがあるため、素性の値を全て同時に表示すると見にくい場合がある。これを避けるために 1) 素性構造の一部を省略して表示する機能、2) 素性の値がデフォルトの場合に表示を省略する機能、を提供している。

1) の例として、図 9 は図 8 の素性構造の値のうち、素性 HEAD と SPECIAL 以下の構造を省略して表示した画面である。この操作は素性の値の型名をダブルクリックすることで行なうことができる。

2) の例として、図 10 は、図 9 と同じ素性構造を、簡略化して表示した画面である。例えば図 9 では MODIFY 以下が非常に大きな構造を

[3] ここで使用した日本語文法は 2 分木のみを生成する。

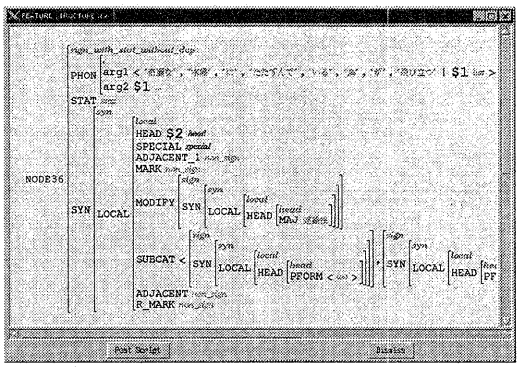

図 10: 図 9 と同じ素性構造を、値がデフォルトの素性を省略して表示した画面

しているが、このうちの多くがデフォルトの値であるため、図 10 では表示が省略されている。

3.4 その他の機能

上記の他、以下の機能がある。

- ノードとその娘の間の枝の脇に文字列を表示する機能があり、これによりノード間の関係を表示することができる。構文木の構造的な違いとしてだけでは表せない曖昧性も表すことができる。
- 頻繁に使用する例文は、ファイルに登録しておくことができ、マウスによる操作で簡単に文入力を行なうことができる。
- 木構造や素性構造の表示内容をポストスクリプトファイルとして保存できる。このとき出力するファイル名、画像の向きと倍率を指定することができる。
- 各ユーザ毎に用意する初期設定ファイルにより、ウィンドウの大きさ、木構造の表示におけるノードの間隔や色、各種フォントをユーザの好みに合わせて変更できる。

3.5 動作

*will*は、ユーザの入力に応じて必要なデータをパーザーから取得する。このときの手順は一種のプロトコルとみなせる。その主なものを表 1 に挙げる。データの取得は実行速度を考慮して、*will*が、最小限必要なものを、必要になった時点で動的にパーザーに対して要求を出すことで行なうようにし、表示に必要のないデータは取得しないようにした。これにより表示にかかる時間が大幅に短縮される[4]。

*will*とパーザーとのプロトコルは、一般性を考慮してパーザーや文法の種類を選ばずに利用しやすいように決定した。このプロトコルを用いて取得するデータは、それぞれパーザーが保持している基本的な情報であるため、*will*を利用するためにパーザー側で作成するプログラムは簡単に実装できる。

データの送受信には、標準入出力の他、TCP/IP ソケットを使用できる。このため、パーザーとは別のマシンで*will*を実行させることもできる。さらに、*will*をパーザー側から起動することも、*will*からパーザー側のプログラムを起動することもできるため、様々な利用方法が可能である。我々の使用している LiLFeS 上でのパーザーでは、LiLFeS から*will*を起動して標準入出力を介してデータの送受信を行なっている。

4 今後の課題

今後、以下の機能を実装し*will*を拡張する予定である。

- 取得した素性構造のデータを比較し、素性構造間の差分を表示する機能。
- 素性構造の値を変更して保存する機能。GUI で値を確認してすぐに変更を行なえるようにすることで文法開発の効率を上げ、開発者の負担を軽減するようにする。
- 解析結果の保存機能を作成し、履歴の参照を可能にする機能。

[4] 我々の現在の英語文法では、構文解析結果のノード数は 1 万以上になることもあり、これら全てのデータを毎回取得すると効率が悪い。ユーザである文法開発者もそれら全てを必要とすることがほとんどないため、このように設計した。

命令	意味	*will*から送信する情報	パーザーが返す値
parse	構文解析の実行	文	"YES" または "NO"
get_root	根の ID 取得	なし	根の ID のリスト
get_allid	構文解析中に生成された全てのノードの取得	なし	全てのノードの ID のリスト
get_dtrs	あるノードの娘とその関係の取得	ノードの ID	娘の ID と文字列のペアのリスト
get_label	木構造中のノードとして表示する文字列の取得	ノードの ID	文字列
get_pos	木構造中のノードに表示する品詞の取得	ノードの ID	文字列
get_fs	素性構造の取得	ノードの ID	素性構造を表すリスト
quit	*will*の終了	なし	なし

表 1: *will*とパーザーとの間のプロトコル

5 おわりに

本稿では、HPSG の文法開発支援のための GUI *will*を紹介した。現在我々のグループでは文法開発のために *will* を用いてその効率を挙げているが、さらにその機能を拡張してより機能的にする予定である。また、*will* のプログラムは近日中に一般に公開する予定である [6]。

なお、本研究は日本学術振興会の未来開拓プロジェクトの助成を受けて行なわれている。

参考文献

[1] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, 1994.

[2] EDR (Japan Electronic Dictionary Research Institute, Ltd.). EDR electronic dictionary version 1.5 technical guide, 1996. Second edition is available via http://www.iijnet.or.jp/edr/E_TG.html

[3] Takaki Makino, Minoru Yoshida, Kentaro Torisawa, and Jun'ichi Tsujii. LiLFeS — Towards a Practical HPSG Parser. In *COLING-ACL'98 Proceedings*, pp.807-811, 1998.

[4] Takashi Ninomiya, Kentaro Torisawa, and Jun'ichi Tsujii. An Efficient Parallel Substrate for Typed Feature Structures on Shared Memory Parallel Machines. In *COLING-ACL'98 Proceedings*, pp.968-874, 1998.

[5] Sun Microsystems. Inc. *Sun Tcl/Tk project* is available via http://www.sunlabs.com/research/tcl/

[6] *Tsujii Laboratory Home Page* is available via http://www.is.s.u-tokyo.ac.jp/~tsujiilab/

[7] Yuka Tateisi, Kentaro Torisawa, Yusuke Miyao, and Jun'ichi Tsujii. Translating the XTAG English grammar to HPSG. In *TAG+ Workshop Proceedings*, pp.172-175, 1998.

[8] Yutaka Mitsuishi, Kentaro Torisawa, and Jun'ichi Tsujii. HPSG-Style Underspecified Japanese Grammar with Wide Coverage. In *COLING-ACL'98 Proceedings*, pp.876-880, 1998.

[9] 橋田 浩一. GDA 日本語タギングマニュアル 草稿第 0.19 版. http://www.etl.go.jp/etl/nl/gda/tagman.html