NAIST Technical Report
NAIST-IS-TR97007

日本語形態素解析システム
『茶筌』
version 1.0
使用説明書

松本裕治　北内啓　山下達雄　今一修　今村友明

平成9年2月

Japanese Morphological Analysis System ChaSen Manual
Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Osamu Imaichi and Tomoaki Imamura

The Japanese morphological dictionary included in this system originates from ICOT Free Software. The following conditions for ICOT Free Software applies to the morphological dictionary of the system.

Each User may also freely distribute the Program, whether in its original form or modified, to any third party or parties, PROVIDED that the provisions of Section 3 ("NO WARRANTY") will ALWAYS appear on, or be attached to, the Program, which is distributed substantially in the same form as set out herein and that such intended distribution, if actually made, will neither violate or otherwise contravene any of the laws and regulations of the countries having jurisdiction over the User or the intended distribution itself.

NO WARRANTY

The program was produced on an experimental basis in the course of the research and development conducted during the project and is provided to users as so produced on an experimental basis. Accordingly, the program is provided without any warranty whatsoever, whether express, implied, statutory or otherwise. The term "warranty" used herein includes, but is not limited to, any warranty of the quality, performance, merchantability and fitness for a particular purpose of the program and the nonexistence of any infringement or violation of any right of any third party.

Each user of the program will agree and understand, and be deemed to have agreed and understood, that there is no warranty whatsoever for the program and, accordingly, the entire risk arising from or otherwise connected with the program is assumed by the user.

Therefore, neither ICOT, the copyright holder, or any other organization that participated in or was otherwise related to the development of the program and their respective officials, directors, officers and other employees shall be held liable for any and all damages, including, without limitation, general, special, incidental and consequential damages, arising out of or otherwise in connection with the use or inability to use the program or any product, material or result produced or otherwise obtained by using the program, regardless of whether they have been advised of, or otherwise had knowledge of, the possibility of such damages at any time during the project or thereafter. Each user will be deemed to have agreed to the foregoing by his or her commencement of use of the program. The term "use" as used herein includes, but is not limited to, the use, modification, copying and distribution of the program and the production of secondary products from the program.

In the case where the program, whether in its original form or modified, was distributed or delivered to or received by a user from any person, organization or entity other than ICOT, unless it makes or grants independently of ICOT any specific warranty to the user in writing, such person, organization or entity, will also be exempted from and not be held liable to the user for any such damages as noted above as far as the program is concerned.

JUMAN	
Version 0.6	17 February 1992
Version 0.8	14 April 1992
Version 1.0	25 February 1993
Version 2.0	11 July 1994
ChaSen	
Version 1.0	19 February 1997

目 次

はじめに

計算機による日本語の解析において，欧米の言語の解析と比べてまず問題になるのに次の2点があります．一つは形態素解析の問題です．ワードプロセッサの普及などによって日本語の入力には大きな問題がなくなりましたが，計算機による日本語解析では，まず入力文内の個々の形態素を認識する必要があります．これには実用に耐えられるだけの大きな辞書も必要であり，これを如何に整備するかという問題も同時に存在します．もう一つの問題として，日本語には広く認められ同意を得られた文法，ないし，文法用語がないという現実です．学校文法の単語分類および文法用語は一般には広く知られていますが，研究者の間ではあまり評判がよくありませんし，計算機向きではありません．

日本語の解析に真っ先に必要な形態素解析システムは，多くの研究グループによって既に開発され技術的な問題が洗い出されているにも係わらず，共通のツールとして世の中に流布しているものはありません．計算機可読な日本語辞書についても同様です．

本システムは，計算機による日本語の解析の研究を目指す多くの研究者に共通に使える形態素解析ツールを提供するために開発されました．その際，上の二つ目の問題を考慮し，使用者によって文法の定義，単語間の接続関係の定義などを容易に変更できるように配慮しました．

大学で小人数で開発したシステムであり，色々な点で不完全な部分があると思います．可能な限り順次改良を重ねる予定です．皆様の寛容な利用をお願いいたします．

本茶筌システムの原形は，京都大学長尾研究室および奈良先端科学技術大学院大学松本研究室において開発された日本語形態素解析システムJUMAN(version2.0)です．JUMANは，京都大学および奈良先端科学技術大学院大学のスタッフおよび多くの学生の協力を得て作成したものです．また，辞書に関しては，Wnnかな漢字変換システムの辞書，および，ICOTから公開された日本語辞書を利用し，独自に修正を加えました．JUMAN2.0をともに開発した京都大学の黒橋禎夫さん，現在キャノン勤務の妙木裕さんには特に感謝いたします．

JUMAN開発のきっかけを作って下さった京都大学長尾真先生に感謝します．JUMAN開発に関して様々な形で協力していただいた奈良先端大宇津呂武仁氏に感謝します．奈良先端大の知念賢一氏には，茶筌システムの開発に関して多くの助言をいただきました．また，一人一人名を挙げることはできませんが，JUMANシステムおよび茶筌のβ版に対して多くのコメントと質問をいただいた利用者の方々に感謝します．

平成9年2月19日

本システムに関するお問い合わせは以下にお願いします．
〒630-01
奈良県生駒市高山町8916-5
奈良先端科学技術大学院大学
情報科学研究科
松本研究室
茶筌管理開発担当者集団
Tel: (07437)2-5240
Fax: (07437)2-5249
E-mail: chasen@cactus.aist-nara.ac.jp

また以下のURLにて最新情報を提供しています．
URL: http://cactus.aist-nara.ac.jp/lab/nlt/chasen.html

本使用説明書を読む前に

本説明書は茶筌に関する様々な情報が記述されている．茶筌をデフォルトの設定で利用するにはすべてを読む必要はなく．次のような目的に合わせて読めばよい.

- 茶筌のインストール手順については．茶筌のトップディレクトリの README，README.jp あるいは 付録 D に書かれている．
- 茶筌の実行方法およびオプションについては，README，README.jp あるいは5.2節を参照のこと．
- 茶筌で想定されている日本語文法の記述方法を知りたい場合は，1 章を読むこと．ここでは形態素情報の記述の仕方が書かれている．
- 茶筌では，利用者が自由に辞書を設定できる．また，品詞のセットを始め文法すべてを利用者が定義できる．記述法については，2.2節で述べられている．
- 茶筌で用いている形態素解析アルゴリズムおよび未定義語処理の方法については，それぞれ4.1節，4.2節に書かれている．また，辞書検索アルゴリズム，JUMAN 2.0 との性能比較に関しては，別にドキュメントが用意されているのでそちらを参照されたい．

1 日本語形態素文法

本システムで仮定している日本語の形態素文法およびここで用いる用語について説明する．

1.1 形態品詞および品詞細分類

本システムでは形態素の集合の二階層の分類が仮定されている．これらの分類を形態品詞および (形態) 品詞細分類と呼ぶ．

例えば，名詞を普通名詞，サ変名詞，固有名詞などに細分類することができる．また，ある形態品詞を細分類したくない場合は細分類を指定しなくてもよい．例えば，品詞細分類を持たない形態品詞として動詞を定義してもよい．

この二階層に分類されたそれぞれの集合を品詞と呼ぶ．上の例では，名詞の細分類としての普通名詞，サ変名詞，固有名詞，および，動詞はそれぞれ品詞である．

1.2 活用型と活用形

学校文法における ‘動詞’ ‘形容詞’ ‘形容動詞’ などのように，後接する形態品詞に応じてその形態が変化する形態素が存在する．このような形態変化を 活用 (conjugation) という．このような形態素の表層のうち変化しない部分を (活用) 語幹，変化する部分を (活用) 語尾と呼ぶ．大部分の活用は規則的である．その規則性に従って活用のタイプを分類し，これを活用型と呼ぶ．そして，連接に応じて実際にとり得る表層的な個々の形態を 活用形と呼ぶ．

1.3 形態素構造

形態素 m の形態品詞，品詞細分類，活用型，活用形，見出し語がそれぞれ $H1, H2, K1, K2, M$ であるとき，これを 5 項組 $(H1\ H2\ K1\ K2\ M)$ によって表わし，形態素 m の 形態素構造」と呼ぶ．

形態素構造の項 $H1, H2, K1, K2$ には，具体的な名称として，それぞれ形態品詞名・品詞細分類名・活用型名・活用形名を与える．また，項 M には，漢字または仮名で表記される見出し語を与える．各形態素構造の記述には特別なシンボル '$*$' を使用することができる．'$*$' は，その項を考慮しない (未指定，'don't care') ことを表わす．たとえば，形態素構造

$$\alpha_1 = (* \ * \ * \ * \ *)$$

は任意の形態素を表わす集合とみなすことができる．同様に，形態素構造

$$\alpha_2 = (\text{名詞} \ * \ * \ * \ *)$$

は '名詞' という形態品詞に分類されたすべての形態素の集合を表わす．また，形態素構造

$$\alpha_3 = (* \ * \ * \ \text{未然形} \ *)$$

は '未然形' という活用形を取るすべての形態素の集合を表わす[1]．また，

$$\alpha_4 = (\text{助詞 格助詞} \ * \ * \ \text{が})$$

のように具体的な形態素を指定することも許される[2]．

この記法は次に述べる連接規則を記述する際に使われる．実際の記述の際，形態素構造を表わすリストのある要素以降がすべて未指定でよい場合は，形態品詞の部分以外はそれらを省略してもよい．

1.4 連接規則と連接関係

形態素構造の集合の組 A_1 と A_2 に含まれるそれぞれの要素が文中に連続して現れ得るとき，それらは連接可能であるといい，それを記述した規則を連接規則と呼ぶ．また，連接規則によって定義される 2 項関係を連接関係と呼ぶ．

連接規則は二種類の形態素の連接可能性が記述可能である．また，連接の強弱に関するコストをそれぞれの連接規則に対して記述することができる．連接規則は，次節で述べる連接規則辞書によって定義される．

2 文法と辞書の定義とデータ構造

2.1 文法と辞書の概要

本システムで用いる文法/辞書について述べる．本システムの特徴は，利用者によって形態品詞名や連接関係などの文法情報や語彙を自由に定義できることである．

日本語の形態素文法/辞書は，本システムが標準的に仮定しているファイルによって定義されなければならない．本システムには標準的な形態素文法/辞書が付随しているが，ユーザはこれを自由に定義し直すことができる．ユーザが定義できるファイルを総称して定義ファイルと呼ぶ．定義ファイルは，文法定義ファイル[3]と形態素定義ファイルに大別される (表 1，表 2)．

---

[1] この例では，$\alpha_1 \supseteq \alpha_2, \alpha_1 \supseteq \alpha_3$ という関係が成り立つ．

[2] この例のように，活用しない形態素に対しても 5 項組によって形態素を表現する必要があり，見出し語を指定する場合には活用型名，活用形名の部分を$*$としなければならない．

[3] 茶筌の文法ファイルの形式は JUMAN2.0 の上位互換であり，例えば `chasen.grammar` が見つからなければ JUMAN2.0 で用いられていた `JUMAN.grammar` を読み込むというようになっているので，JUMAN の文法ファイルもファイル名を変更することなくそのまま利用することができる．

文法定義ファイルは 1 節で述べた日本語形態素文法を定義するためのファイルで，形態品詞分類ファイル，活用関係ファイル，活用ファイル，連接規則ファイルから構成される (2.2 節)．形態素定義ファイルは個々の形態素についての情報を実際に記述する一つ以上の形態素辞書からなる (2.3 節)．

システムファイルは，本システムによって定義ファイルから自動的に生成されるもので，ユーザはこれらの中身に関して意識する必要はない．文法定義ファイルからは連接対応表と連接行列が生成される．連接対応表 ·連接行列 ·文法定義ファイルを参照して，形態素辞書に記述された情報を記号化したシステム辞書とインデックスファイルが生成される．

解析プログラムはシステムファイル全てと文法定義ファイルの一部を用いて解析を行う．

	形態素データファイル	文法データファイル
定義ファイル	形態素定義ファイル 形態素辞書	文法定義ファイル 形態品詞分類ファイル 活用関係ファイル 活用ファイル 連接規則ファイル
システムファイル	システム辞書 インデックスファイル	連接対応表 連接行列

表 1: 文法/辞書ファイルの分類

種類	名称	説明
文法	形態品詞分類辞書 (`chasen.grammar`)	品詞分類を定義する
	活用関係辞書 (`chasen.ctypes`)	活用する各品詞に対して活用型の一覧を定義する
	活用辞書 (`chasen.cforms`)	個々の活用型の具体的な活用形の一覧を定義する
	連接規則辞書 (`chasen.connect.c`)	連接関係を定義する連接規則の集合
	連接対応表 (`chasen.table`)	形態品詞と接続番号との対応表
	連接行列 (`chasen.matrix`)	接続可能性表
形態素	各品詞辞書 (`Noun.dic`, etc.)	個々の形態素の定義 (2.3を参照)
	システム辞書 (`*.int`)	各形態素の情報が格納される
	インデックスファイル (`*.pat`)	パトリシア木によるシステム辞書へのインデックス

表 2: 文法/辞書ファイルの説明

本システムを使用する際に個人的にカスタマイズできる情報があり，その情報は，使用者のホームディレクトリにある`.chasenrc` というファイルに記述されていると仮定される．このファイルが存在しない場合は，本システムがもつ標準的な情報が使用される．`.chasenrc` ファイルに書かれる情報の詳細は後で詳しく述べる．

`chasenrc` というファイルが配布版のトップディレクトリにあるので，標準的な使い方をする利用者は，これを各自のホームディレクトリに． `chasenrc` という名前でコピーすればよい．

また，本システムでは標準的な形態素辞書を用意している．これについては，2.5 節で紹介する．

文法ファイルおよび形態素辞書では，各行中のセミコロン ‘;’ で始まる文字列はその行末までがコメントとみなされる．

2.2 文法定義ファイルの記述法

文法定義ファイルの構成と内容は次のようになっており，各ファイルの記述はS式にいよって行なう．付録Aに記述のフォーマットをBNF式と例によって示す．

1. 形態品詞分類ファイル ：(cf. `chasen.grammar`)

   本システムで用いる形態品詞およびその品詞細分類の名称を定義する．一つの形態品詞およびそれの品詞細分類を一つのS式(リスト構造)によって記述する．リストの第1要素に形態品詞を記述し，品詞細分類があれば第2要素に品詞細分類名をリストによって記述する．

   個々の形態品詞と品詞細分類はリストの形で表現されるが，それに属する形態素が活用する場合にはその第2要素として記号 `%` を付加する．形態品詞が記号 `%` をもつ場合には，その細分類に属するすべての形態素が活用するとみなされる．

   形態品詞分類ファイルの記法を付録A.1に示す．

2. 活用関係ファイル ：(cf. `chasen.ctypes`)

   活用関係ファイルは，形態品詞分類ファイルによって活用すると定義された形態品詞または品詞細分類それぞれに対してそれが取り得る活用型の一覧を定義する．形態品詞分類ファイルで活用すると定義されたものに対しては，必ず取り得る活用型の一覧が書かれていなければならない．活用型の名前付けはユーザが自由に行なってよい．

   個々の活用関係は，二つの要素からなるリストとして定義される．第一要素は形態品詞のリストまたは形態品詞と品詞細分類からなるリストであり，活用する品詞を示す．第二要素はその品詞が取り得る活用型の一覧を示すリストであり，任意個数のアトムを含む．

   活用関係ファイルの記法を付録A.2に示す．

3. 活用ファイル ：(cf. `chasen.cforms`)

   活用ファイルは，活用関係ファイルによって定義された個々の活用型に対して，それが取り得る活用形の名前および活用語尾とその読みを定義する．

   この辞書の各項目は二つあるいは三つ要素からなるリストであり，第1要素に活用型の名前を記述し，第2要素として活用形の名前と活用語尾(とその読み)のリストより構成されるリストを記述する．この時，活用形の名前として「基本形」を必ず記述しなければならない．後で述べる形態素辞書には，活用する形態素はその基本形が登録されていると仮定されることになっているので，個々の形態素については，形態素辞書に登録された見出し表記からこの辞書で定義される基本形の語尾を取り除いたものがその形態素の不変化部分であるとみなされる．活用語尾の存在しない活用形(例えば，語幹)に対しては，活用語尾として $*$ と記述すればよい．

   また，不変化部分が存在しない活用語については，見出し語と基本形語尾が同じになる(例えば，付録A.3.2のカ変動詞来型の活用)．このような場合には，システムはすべての活用語尾を個別の形態素として形態素辞書に登録する．

   活用ファイルの記法を付録A.3に示す．

4. 連接規則ファイル ：(cf. `chasen.connect.c`)

   連接規則ファイルは連接規則の集合である一つの連接規則は形態素構造(1.3節を参照)の集合の対と連接コストからなる3要素のリストによって表現される。第一要素と第二要素に含まれる形態素構造の集合の連接コストを第三要素で指定する。連接コストの値は、0から255まで

の整数値でなければならない。省略されている場合は10とみなす。また、連接コストを0に指定することによって、二つの形態素が連接不可能であること表わす。

一つの形態素が複数の連接規則内の形態素構造に含まれることがあってもよい．連接関係はこれらの規則の和集合であるとみなされる．したがって，ある特定の形態素構造の対が連接可能であることが複数の連接規則によって定義されることがある．このような場合，これらの連接のコストは，連接規則ファイル内で後方に現れた規則によって指定されるコストを取るとみなされる．

形態素構造で用いられる品詞名や活用形名は文法辞書で定義されたものでなければならない．例外として，（文頭）および（文末）という形態素構造がある．前者は，文頭の直前に現れると考えられるダミーの形態品詞であり，少なくとも一つの連接規則の第1要素の中に現れなければならない．同様に，後者は文末の直後に繋がると考えられるダミーの形態品詞であり，少なくとも一つの連接規則の第2要素の中に現れる必要がある．

連接規則ファイルの記法を付録A.4に示す．

連接規則においては，ある品詞のグループが同じ連接関係をもつことがよく起こる．連接規則の記述を省力化するために，Cプリプロセッサで用意されているマクロ機能を利用することができる．`chasen.connect.c`ファイル中に次のようなマクロを定義することにより，以降の連接規則内で`NormalNominal`という記号を`#define`の第二引数と等価に扱うことができる．

```
#define NormalNominal （名詞 普通名詞）\
                      （名詞 サ変名詞）\
                      （名詞 固有名詞）\
                      （名詞 地名）\
                      （名詞 人名）\
                      （名詞 数詞）\
                      （指示詞 名詞形態指示詞）\
                      （動詞 * * 基本連用形）
```

2.3 形態素辞書の記述法

形態素辞書もS式を用いて記述する．形態素辞書は，最後に‘`dic`’という拡張子をもつファイルに格納する．形態素辞書は複数のファイルに分割されていてもよい．

各形態素は，複数の見出し語を持ってもよい．また，それぞれの見出し語に対してコスト計算のための相対的な重みを指定することができる．形態素のもつコストについては，4章で説明する．複数の見出し語が記述されている場合，見出し語以外の形態素情報は共有される．

次に形態素辞書の形態素定義の記述方法をBNFで示す．

```
〈形態素定義〉            ::= (〈#形態品詞名〉〈形態素情報の並び〉) |
                             (〈#形態品詞名〉(〈#品詞細分類名〉〈形態素情報の並び〉))
〈形態素情報の並び〉      ::= 〈形態素情報〉 | 〈形態素情報〉〈形態素情報の並び〉
〈形態素情報〉            ::= (〈見出し語情報〉〈読み情報〉〈活用型情報〉〈意味情報〉)
〈見出し語情報〉          ::= (見出し語 〈見出し語内容の並び〉)
〈見出し語内容の並び〉    ::= 〈見出し語内容〉 | 〈見出し語内容の並び〉
〈見出し語内容〉          ::= 〈#見出し語表記〉 | (〈#見出し語表記〉) |
                             (〈#見出し語表記〉〈#数値〉)
〈読み情報〉              ::= (読み 〈#読み表記〉)
〈活用型情報〉            ::= (活用型 〈#活用型名〉) | NIL
〈意味情報〉              ::= (意味情報 〈#意味記述〉) | NIL
```

- 〈形態素情報〉は，共通の形態品詞名または品詞細分類名ごとにまとめて〈形態素情報の並び〉として書くこともできるし，個別に記述してもよい．〈#形態品詞名〉，〈#品詞細分類名〉は，「形態品詞分類辞書」で定義されていなければならない．また，品詞細分類されている形態品詞に対しては品詞細分類名を指定しなければならない．

- 〈活用型情報〉は，〈#形態品詞名〉または〈#品詞細分類名〉が活用すると定義されている時は省略できない．

- 〈#見出し語内容〉は形態素の表層の形と相対的な重みを表わす¡#数値¿の対として表現する．相対的な重みが 1 の場合は，重みの記述を省略してもよい．重みとして指定できるのは正整数または小数点 1 桁以内の正小数である．活用する形態素の表層の形は，その基本形を書かねばならない．

- 〈#読み表記〉には形態素の読みを記述する．活用する形態素については，その基本形の読みを書く．

- 〈#意味記述〉には意味情報を記述する．意味情報には任意の S 式を記述することができる．また，アトムとして，二重引用符 (") で囲まれたテキストを自由に用いることができる．二重引用符に囲まれた範囲では任意の文字が使用可能である．二重引用符で囲まれたアトムの中の二重引用符は ‘\"’ によって記述可能である．記述されたデータは二重引用符を含んでそのままテキストデータとして扱われる．

付録 Bに形態素辞書の記述例を示す．

2.4 複数辞書の使用

茶筌では，必要に応じて複数の辞書を使用することが出来るようになっている．ここではその使用法について述べる．

辞書は 2.1節で述べたように，文法辞書及び形態素辞書に分けられる．複数化できるのは，システム辞書 (例 :chadic.int) ・インデックスファイル (例 :chadic.pat) からなる形態素辞書である．

利用者が用いる形態素辞書の指定は，他のオプションと同様に，利用者のホームディレクトリにおかれた ‘`.chasenrc`’ というファイルによって行なわれる．複数の辞書の指定もこのファイルに記述される．具体的な記述法については，3節を参照のこと．また，複数の形態素辞書を使用する場合は，それらは同じ文法に従っていなければならない．

2.5 システム標準文法

本システムのテキスト形式辞書として標準の文法が用意されている．これをシステム標準文法と呼ぶ．システム標準文法を定義する文法ファイルのうち，形態品詞分類ファイル，活用ファイル，連接規則ファイルは，益岡・田窪文法 [1] を参照し，それを拡張して作成した．

1. 形態品詞分類ファイル：(cf. `chasen.grammar`)
   益岡・田窪文法に「特殊」(句読点・記号・括弧など) を加え，接辞を「接頭辞」「接尾辞」に分けて，14 種類の形態品詞を定義した．
2. 活用ファイル：(cf. `chasen.cforms`)
   益岡・田窪文法に対して，文語的表現・口語的表現・敬語表現に対応できるように拡張を行ない，21 の一般的な活用型と 7 の特殊な活用型を定義した．
3. 活用関係ファイル：(cf. `chasen.ctypes`)
   活用する形態構造の一覧とそれらが取り得る活用型の一覧を定義している．
4. 連接規則ファイル：(cf. `chasen.connect.c`)
   新たに作成した．

3 リソースファイル (cf. .chasenrc)

このファイルは形態素解析プログラムに必要な様々な選択肢を定義するために用いられる．オプションの定義は通常，利用者のホームディレクトリにある‘`.chasenrc`’というファイルによって行なわれる．ただし，起動時に `-r` オプションでリソースファイルが指定されていればそれを優先的に読み込む．さらに，環境変数 `CHASENRC` があれば，それによって指定されたファイルを読み込む．そのどちらも指定されていない場合は，利用者のホームディレクトリにある`.chasenrc` を読み込み，それも存在しない時はデフォルトのオプションファイルが参照される．デフォルトのオプションファイルの指定は，`$CHASEN/Makefile` [4] 内の `RCPATH` という変数によって与える．

オプションファイルの記述例を付録 Cに示す．現在存在するオプションを次に示す．これらはすべて指定されていなければならない．

1. 文法ファイルの位置
   文法ファイル (`chasen.grammar`, `chasen.ctypes`, `chasen.cforms`, `chasen.connect`, `chasen.table`, `chasen.matrix`) が存在するディレクトリを指定する．指定は次のように行なう．

```
(文法ファイル /usr/local/lib/chasen/dic)
```

2. システム辞書の位置
   システム辞書 (`chadic.int`) とインデックスファイル (`chadic.pat`) が存在するディレクトリ，および，末尾の拡張子を除いた辞書ファイル名を指定する．複数組みの辞書を指定することも可能である．指定は次のように行なう．

```
(PATDIC
        /home/mydir/chasen/dic/MYDIC
        /usr/local/lib/chasen/dic/chadic )
```

---

[4] `$CHASEN` は茶筌システムがインストールされているディレクトリである．

この例では，二組みの辞書を指定している．辞書引きに際しては，これらの辞書の両方が用いられる[5]．

使用する辞書の最大数は，5 個に設定されている．これを変更したい場合は，

```
$CHASEN/chasen/pat.h
```

の `MAX_DIC_NUMBER` の値を変更しておけばよい．

3. 未定義語の品詞定義
   未定義語が発見された時にその語をどのような品詞として連接規則を適用するかを指示する．指定できるのは，文法で定義されている形態素名である．品詞細分類名が存在する品詞については，品詞細分類名も定義されなければならない．次の例は，未定義語を名詞の細分類としてのサ変名詞と定義している．

   ```
   (未定義語 (名詞 サ変名詞))
   ```

4. 形態素のコストの定義
   4章で述べるように，形態素解析プログラムでは，解析結果の優先情報をコストとして計算している．解析に曖昧性がある場合は，コストの総計が低いものを優先することになっている．文法辞書によって定義された形態素のコストを定義しなければならない．また，「未定義語」についてのコストも指定する必要がある．

   品詞は，品詞名，または，品詞名と品詞細分類名の対によって表わされる．品詞細分類が定義されている品詞については，品詞細分類名 (または ‘*’) を指定する必要がある．コストは正整数の値をとる．

   ```
   (品詞コスト
           ((*)              10)
           ((未定義語)      500)
           ((特殊 *)        100)
           ((動詞)          100)
           ((形容詞)        100)
           ((判定詞)         10)
           ((助動詞)         10)
           ((名詞 *)        100)
           ((名詞 副詞的名詞) 99)
           ((名詞 固有名詞) 110)
           ((名詞 形式名詞) 10)
           ((指示詞 *)       80)
           ((副詞 *)        100)
           ((助詞 *)         10)
           ((接続詞)        100)
           ((連体詞)        100)
           ((感動詞)        100)
   ```

---

[5] 一組の辞書には同一の形態素の登録は行なわれないが，複数の辞書に同じ形態素が登録されている場合はあり得る．このような場合は，同じ形態素が複数得られることになる．

```
        ((接頭辞 *)       10)
        ((接尾辞 *)       10)
)
```

同じ品詞に対してコストの定義が複数回指定されている場合は，後のものが優先される．上の例では，品詞名が「名詞」の形態素のコストは 100 であるが，細分類名が「固有名詞」の形態素のコストは 110 となる．また，先頭の‘(*)’の指定により，ここで明示的に定義されていない形態素のコストはすべて 10 となる．

5. 連接コストと形態素コストの相対的な重みの定義
   形態素解析におけるコストの計算は形態素のコストと連接のコストの総計として計算される．これら二種類のコストに異なる重みを掛けたい場合には，それを指定することができる．解析結果のコストはそれぞれのコストにここで指定された重みを乗じた値の総計として計算される．

```
(連接コスト重み 3)
(形態素コスト重み 1)
```

   また，形態素解析の過程において，常にコストが最低の結果を出すのではなく，ある程度のコスト幅を許容したい場合がある．この許容幅を指定することができる．

```
(コスト幅 0)
```

   コスト計算については，4章を参照のこと．

6. 未定義連接コストの定義
   連接規則辞書に連接規則が定義されていない形態素間の連接コストを指定する．未定義連接コストを設定しないか，あるいは 0 を指定すると，連接規則が定義されていない形態素どうしは決して連接しないという意味になる．

```
(未定義連接コスト 100)
```

7. 連結品詞
   ある品詞の形態素が連続して出現したときに，一つの形態素として連結して出力させるときに使用する．例えば

   (連結品詞 (名詞 数詞) (特殊))

   と定義すると，連続した「数詞」と連続した「特殊」がそれぞれ一つの形態素として出力されるようになる．ただし，「数詞」と「特殊」が連続している場合は連結されず，別々の形態素として出力される．

4 形態素解析

形態素解析に必要な文法や辞書の存在場所，並びに，解析中で用いられるコストに関する基本的な情報は前出の `.chasenrc` に記述されている．

茶筌は，EUC あるいは JIS(ISO-2022-JP) の日本語文字列を入力とし，連接規則によって許容された形態素からなる束 (lattice) 状の構造を出力とする．ただし，結果の表示については，解に含まれる形態素情報を 1 形態素毎に 1 行に表示するだけである．解析結果は日本語 EUC で表示される．表示についてはいくつかのオプション (cf. 5.2節) が用意されている．

4.1 形態素解析アルゴリズム

茶筌の解析アルゴリズムは，コスト最小法[6] に基づいており，入力としてあたえられた日本語の文字列[7] に対する次の基本動作よりなる．

- ある特定の位置からはじまるすべての可能な形態素を辞書引きによって得る．
- 辞書引きによって得られた個々の形態素に対して，その直前の位置に存在するすべての形態素との連接可能性のチェック，および，コストの計算を行なう．
    - 連接可能性のチェックによって連接不可能とわかった形態素間の接続は行なわれない．また，その位置で連接可能なもののうち最小のコストと比較して`.chasenrc` の（コスト幅 ??）によって定義される数値以上のコスト差をもつ形態素の接続は行なわれない．

本形態素解析システムでは次の二種類のコストが使用される．

形態素のコスト: 個々の形態素に与えられているコストである．形態素のコストは，`.chasenrc` で定義された「品詞コスト」，「形態素コスト重み」，および，形態素辞書で定義された見出し語のコストに対する重み，の三つの数値の積として計算される．

連接コスト: 二つの形態素の連接のコストである．連接規則ファイルで定義されたコストと `.chasenrc` で定義された「連接コスト重み」の積として計算される．

形態素解析の一つの解析結果は形態素からなる列であり，その総コストは，それに含まれる形態素のコストの総計，および，各形態素間の連接コストの総計の和である．ただし，列の先頭と末尾には，それぞれ，「文頭」および「文末」と呼ばれるダミーの形態素があると仮定する．

一般に解析結果は，文頭および文末を両端点とする束状のグラフ構造になる．このグラフ内の文頭から文末へいたるそれぞれの経路が一つの可能な解析結果を表わすことになる．

連接可能性のチェックおよびコストの計算の詳細について説明する．最初の辞書引きは文字列の先頭で行なわれ，その直前には「文頭」が存在すると仮定される．また，「文頭」がもつコストは 0 である．

最初に辞書引きが行なわれるのは入力文字列の先頭 (すなわち，「文頭」の直後) である．以降，辞書引きが行なわれるのは入力文字列中において文頭からの有効な連接をもつ形態素の直後の位置である．形態素の有効性，無効性については，以下の説明の中で述べる．

ある位置で辞書引きによって得られるすべての形態素を $Mf_j (1 \leq j \leq n)$ とし，その位置の直前で終了する形態素が $Mb_i (1 \leq i \leq m)$ であるとする．各 Mb_iは文頭から自分自身へいたる束状のグラフの経路の中で最小のコストの値を保持している．そのコストを Cb_i とする．

それぞれの Mf_jについて，すべての Mb_iとの連接可能性のチェックを連接行列により行なう．Mb_i の後接情報と Mf_jの前接情報により，それらが連接可能であるか，また，連接可能であればその連接コストを求める．Mb_iと Mf_jの連接コストを $C_{i,j}$ とする (連接不可能な場合 $C_{i,j} = \infty$ とする)．したがって，Mf_jに対するすべての連接コストは，

$$Cb_1 + C_{1,j},\ Cb_2 + C_{2,j},\ \cdots, Cb_m + C_{m,j}$$

となる．これらの最小値からの差が `.chasenrc` の「コスト幅」以内であるものだけが Mf_j に連接可能であると考えられる．連接可能な Mb_i が存在しない場合 (すべての $C_{i,j}$ が無限大の場合)，Mf_j は

---

[6] 形態素解析における解の優先度決定のための一手法で，語や語の連接にコストを与えて，総コストの少ない解を優先する方法．

[7] 改行をもって一つの入力文字列の終了とする．ただし，`-j` オプションをつけると句点が入力文字列の区切りとなる．

無効な形態素と考えられ，以後の解析結果のグラフには含まれない．Mf_j が無効でないとき，Mf_j は 最小の $Cb_i + C_{i,j}$ の値に自身の形態素コストを加えた値を解析結果のコストとして保持することになる．

形態素解析の最後の段階では，入力文字列の末尾に現れる有効な形態素と「文末」との連接可能性のチェックとコスト計算が行なわれ，解析結果としてのグラフ構造が得られる (図 1)．このとき，コスト最小のパスが最適解となる (図 1では，$\boxed{\text{さかな}}-\boxed{\text{だ}}-\boxed{\text{よ}}$)．

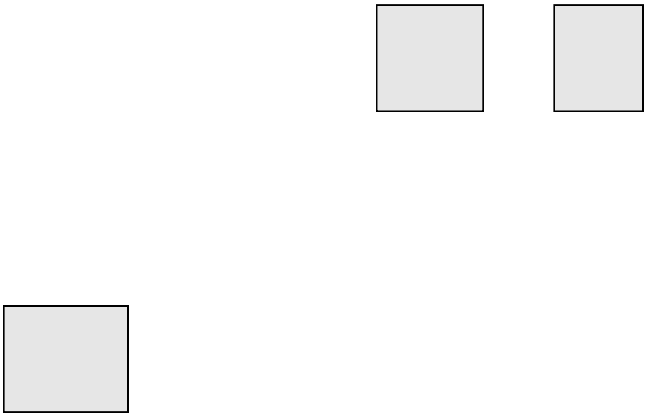

図 1: 解析結果のグラフ構造

4.2　未定義語の取り扱い

未定義語に関しては，本システムでは，入力文字列中のあらゆる位置から未定義語が存在する可能性を考慮している．ひらがなおよび漢字以外の文字については，同種の文字の終りまで (カタカナ，アルファベット，数字 等) をまとまった語として一つの未定義語と解釈する．ひらがなと漢字については，一文字ずつを未定義語の候補と考えている．

切り出された未定義語に対する連接関係を定義するために，未定義語をどのような品詞と考えるかは，オプション定義ファイル (`.chasenrc`) 内の「未定義語品詞」によって自由に指定できる．ただし，未定義語品詞は，文法辞書で定義された品詞名 (品詞再分類がある場合には品詞再分類名も指定する必要がある) のいずれか一つでなければばならない．

未定義語に対するコストは同じく`.chasenrc` の「品詞コスト」欄で定義できるので，これに高いコストを与えることにより，未定義語を含む解析結果の優先度を下げることができる．

4.3　未定義連接コスト

通常，連接規則ファイルに記述されていない連接規則のコストは 0 である．これは決して連接しないという意味であり，もしそのような単語の並びがあっても正解は出力されない．しかし，めったに出現しない連接について「決して接続しない」とするよりも「かなり低い確率ではあるが接続する」とした方が解析精度が高くなり，未定義語の出力を減らせることが多い．未定義連接コストを設定することにより，連接規則ファイルに記述されていない連接規則のコスト値を指定することができる．

未定義連接コストは，リソースファイルに

```
(未定義連接コスト 100)
```

などと記述して設定する．これは，連接規則ファイルの先頭に

```
(((*)) ((*)) 100)
```

と記述するのと同じ意味を持つ．しかし，連接規則ファイルは修正するたびに辞書をコンパイルし直す必要があるのに対し，未定義連接コストはリソースファイルを書き換えるだけですむという利点がある．

5　茶筌の使用法

5.1　実行方法

システムの実行ファイルは，`$CHASEN/chasen/chasen` である．

- 形態素解析の実行

```
chasen -[b|m|p] -[f|e|c|d] [-F format] [-j] [-C] [-r rc_file]
```

  によって起動され，標準入力，または引数で指定されたファイルから 一行ごとに文を読み込んで処理を行なう．

- 処理内容
  コスト最小 (それぞれの形態素の区切りで最小コストとの差が許容されるコスト幅以内) の解を求め，結果をオプションに従って表示する．各オプションの意味は次節にまとめる．

使用例

JUMAN2.0 からの改良点の一つとして，入力ファイルを引数として指定できる．以下に使用例を示す．

```
% cat > temp
私は昨日学校へ行った．
% chasen temp
私      わたし    私      普通名詞
は      は        は      副助詞
昨日    きのう    昨日    普通名詞
学校    がっこう  学校    普通名詞
へ      へ        へ      格助詞
行った  いった    行く    動詞        子音動詞カ行促音便形    タ形
．      ．        ．      句点
EOS
```

5.2　実行時のオプション

形態素解析の実行については，いくつかのオプションが用意されている．以下にそれをまとめる．

- 解が曖昧性を含む場合の表示方法 (曖昧性がない場合はどの方法も同じ表示となる)

`-b`	後方最長一致の解を一つだけ表示する (デフォルト)
`-m`	曖昧性のある部分だけ，複数の形態素を表示する
`-p`	曖昧性の組合せを展開し，すべての解を個別に表示する

- 各形態素の表示方法

`-f`	カラムを整えて表示 (デフォルト)
`-e`	完全な形態素情報を文字で表示
`-c`	完全な形態素情報をコードで表示
`-d`	各形態素を Prolog の複合項で表現し，それらをリストにしたものを出力
`-F format`	形態素を `format` で指定された形式で出力

- その他

`-j`	句点を文の区切りとして解析
`-r <rc_file>`	`<rc_file>` をリソースファイルとして使用
`-h`	ヘルプメッセージを出力
`-V`	茶筌システムのバージョンを出力
`-C`	コマンドインタプリタを使用

`-j` オプションについて

茶筌では通常，改行をもって一つの入力文字列の終了とする．しかし，解析用のバッファサイズの制限により，解析文が長い (1000 文字程度) 場合，バッファを使いきって解析することが不可能となり ‘Too many morphs’ というメッセージを表示することがある．そのようなときは `-j` オプションをつければ，句点 (‘．’ と ‘。’) を文の区切りとして解析を行うようになるため，長い文章であっても途中に句点があればほとんどの場合解析できるようになる．

`-F` オプションについて

`-F` オプションでは，指定されるフォーマット文字列の末尾に ‘`\n`’ があれば，各形態素情報の表示ごとに改行を行い，文末の次に ‘`EOS`’ だけからなる 1 行を出力する．末尾に ‘`\n`’ がなければ，1 文中の形態素情報を 1 行で出力し，行末に改行を表示する．

`-F` オプションの使用例をいくつかあげる．

- デフォルト (`-f` オプション) と同様の出力
  `-F␣"%-11m␣%-11y␣%-11M␣%-14BM␣%-14T␣␣%F␣\n"`
- `-e` オプションと同様の出力
  `-F␣"%m␣%Y␣%M␣%H␣%h␣%B*␣%b␣%T*␣%t␣%F*␣%f\n"`
- 分かち書き
  `-F␣"%m␣"`
- 漢字かな変換
  `-F␣"%y"`

`-F` オプションのフォーマット文字列の変換文字を以下に示す．

変換文字	
%m	見出し (活用形)
%M	見出し (原形)
%y	読み
%Y	読み (未定義語なら”未定義語”と表示)
%ic	意味情報
%Ic	意味情報 (NIL なら文字 c を表示)
%h	品詞 (番号)
%H	品詞 (文字列)
%b	品詞細分類 (番号)
%BB	品詞細分類 (文字列)(なければ品詞を表示)
%BM	品詞細分類 (文字列)(なければ品詞または”未定義語”と表示)
%Bc	品詞細分類 (文字列)(なければ文字 c を表示)
%t	活用型 (番号)
%Tc	活用型 (文字列)(なければ文字 c を表示)
%f	活用形 (番号)
%Fc	活用形 (文字列)(なければ文字 c を表示)
%c	形態素のコスト
%%	% そのもの
.	フィールド幅を指定
-	フィールド幅を指定
1-9	フィールド幅を指定
\n	改行文字
\t	タブ
\\	\ そのもの
\'	' そのもの
\"	" そのもの

5.3 コマンドインタプリタ

-C オプションにより茶筌実行時に以下のコマンドを対話的に与えることが出来る．

コマンド	
#V	-V オプションと同じ．
#F	-F オプションと同じ．ただし，フォーマット文字列をクオートする必要はない．
#w␣[数字]	コスト幅の変更．例：#w␣500
#i	様々な情報の出力．
#e␣[単語]	辞書に単語が入っているか調べる．例：#e␣茶筌
#q	茶筌の終了．
#a	辞書への単語の追加．
#f	単語を追加する辞書の指定．
#s	単語追加後のパトリシア木のセーブ．

注意

- `#a`, `#f`, `#s` は，茶筌の解析結果を視覚化するプログラム「美茶」用のコマンドなので，無闇に実行してはならない．
- コマンドと引数の間は必ずスペース一文字でなければならない．

6 他のシステムからの利用

6.1 Prolog からの使用

- 基本的構成
  茶筌システムが形態素解析を行なった結果を，SICStus Prolog がパイプを介して受けとるという構成になっている．

- 動作環境
  茶筌が仮定する Prolog 処理系は，SICStus Prolog Release 3 である．日本語を用いるためには，環境変数 `SP_CTYPE` の値を `euc` に設定する必要がある．具体的には，例えば次の一行を利用者の `.login` ファイルなどに含めればよい．

```
setenv SP_CTYPE euc
```

- 起動方法
  Prolog を起動し，‘`$CHASEN/prolog/chasen_user.pl`’ を consult もしくは compile する．

  ‘`$CHASEN/prolog/chasen_user.pl`’ はユーザ設定用ファイルなので，ユーザが各自のディレクトリにコピーして各自の設定を行なうことを前提としている．

  解析は，以下の要領でファイル単位で行なうことができる．

```
| ?- cha.
Input file name?  入力ファイル名.
Output file name?  出力ファイル名.
```

  入力ファイル名，出力ファイル名を “user” とすると，端末からの入出力が可能である．

```
| ?- chatty.
```

  とすれば，標準入力から入力し，標準出力へ結果を出力することができる．

- 処理内容
  Prolog 側からの一回の呼び出しによって，茶筌システムとのパイプがオープンされる．呼び出しが終了するとパイプが閉じられる．Prolog 側は，茶筌システムが形態素解析を行なった結果を受けとるだけである．一つの形態素について，Prolog 側が受けとるデータは，

  > morph([識別子 (ID), 開始位置 (From), 終了位置 (To), コスト (Cost), 見出し語 (Md), 読み (Ym), 基本形 (Kh), 品詞名 (Hn0), 品詞細分類 (Hn), 活用型名 (KT), 活用形名 (KF), 意味情報 (Imi), 形態素コスト (MrphCost), 前節形態素との接続コストのリスト (PreCCL), 前接形態素の識別子のリスト (PreIDL)])

という複合項である．また，茶筌システムからデータを受けとる処理は，‘`$CHASEN/prolog/chasen.pl`’中の`cha/3`において，`read(+`*Str*,-*MorphList*`)`によって上述の複合項のリスト*MorphList*を読み込むという形で行なわれている．

Prolog版には出力オプションは，

- `e`：完全な形態素情報を文字で表示 (デフォルト)
- `f`：カラムを整えて表示
- `s`：見出しと品詞名を表示

であり，述語`cha_print_form/1`により変更することができる．

参考文献

[1] 益岡隆志，田窪行則：『基礎日本語文法 –改訂版–』くろしお出版，1992.

[2] 妙木裕，松本裕治，長尾眞：「汎用日本語辞書および形態素解析システム」情報処理学会第42回全国大会予稿集，1991.

[3] 松本裕治，黒橋禎夫，宇津呂武仁，妙木裕，長尾真：日本語形態素解析システムJUMAN使用説明書 version 2.0，NAIST Technical Report, NAIST-IS-TR94025，1994.

付録

A　文法定義ファイルの記法

文法定義の記法をBNFによって示す．以下，⟨··⟩は非終端記号名を表わす．⟨#形態品詞名⟩のように#が付けられた非終端記号はユーザによって定義される具体的な名前を表わすと仮定する．NILは空リストではなく，空系列を表わす．

A.1　形態品詞分類ファイル (`chasen.grammar`) の記述

A.1.1　形態品詞分類ファイルの個々の項目の記述

```
⟨形態品詞分類ファイル項目⟩  ::=  ( ⟨形態品詞⟩) | ( ⟨形態品詞⟩ ( ⟨品詞細分類の並び⟩))

⟨形態品詞⟩                  ::=  ( ⟨#形態品詞名⟩) | ( ⟨#形態品詞名⟩ %)
⟨品詞細分類の並び⟩          ::=  ⟨品詞細分類⟩ ⟨品詞細分類の並び⟩
⟨品詞細分類⟩                ::=  ( ⟨#品詞細分類名⟩) | ( ⟨#品詞細分類名⟩ %)
```

A.1.2　形態品詞分類ファイルの定義例

```
((動詞 %))                           ; 動詞は形態品詞名であり，活用する．

((名詞)                              ; 名詞は形態品詞であり，
        ((普通名詞)                  ; 普通名詞，サ変名詞，固有名詞 ... に
         (サ変名詞)                  ; 細分類される．
         (固有名詞)
         (数詞)
         (形式名詞)
         (副詞的名詞)))

((接尾辞)
        ((名詞性述語接尾辞)
         (名詞性名詞接尾辞)
         (名詞性名詞助数辞)
         (形容詞性述語接尾辞 %)      ; 形容詞性述語接尾辞は活用する．
         (形容詞性名詞接尾辞 %)
         (動詞性接尾辞 %)))
```

A.2 活用関係ファイル (`chasen.ctypes`) の記述

A.2.1 活用関係ファイルの個々の項目の記述

```
〈活用関係ファイル項目〉  ::=  (  〈品詞〉 (  〈活用型の並び〉) )

〈品詞〉                  ::=  (  〈形態品詞名〉) | (  〈形態品詞名〉 〈品詞細分類名〉)
〈活用型の並び〉          ::=    〈活用型名〉 |    〈活用型名〉 〈活用型の並び〉
```

A.2.2 活用関係ファイルの定義例

```
((形容詞)                       ; 形容詞はイ形容詞とナ形容詞という活用型をもつ
         (イ形容詞
          ナ形容詞))

((助動詞)
        (イ形容詞
         ナ形容詞
         判定詞
         無活用型
         助動詞ぬ型
         助動詞だろう型
         助動詞そうだ型))

((接尾辞 形容詞性述語接尾辞)  ; 接尾辞の中で形容詞性述語接尾辞に細分類される
        (ナ形容詞              ; 形態素はナ形容詞またはイ形容詞という活用型を
         イ形容詞))            ; もつ
```

A.3　活用ファイル (`chasen.cforms`) の記述

A.3.1　活用ファイルの個々の項目の記述

```
〈活用ファイル項目〉  ::=  （ 〈#活用型名〉（ 〈活用形対の並び〉））

〈活用形対の並び〉    ::=  〈活用形対〉| 〈活用形対〉〈活用形対の並び〉
〈活用形対〉          ::=  （ 〈#活用形名〉〈語尾表示〉）|（ 〈#活用形名〉〈語尾表示〉〈語尾読み表示〉）
〈語尾表示〉          ::=  〈#語尾〉 | *
〈語尾読み表示〉      ::=  〈#語尾読み〉 | *
```

A.3.2　活用ファイルの定義例

```
(母音動詞
    ((語幹            *   )
     (基本形          る  )
     (未然形          *   )
     (意志形          よう)
     (命令形          ろ  )
     (命令形          よ  )
     (基本条件形      れば)
     (基本連用形      *   )
     (タ形            た  )
     (タ系条件形      たら)
     (タ系連用テ形    て  )
     (タ系連用タリ形  たり))
)

(カ変動詞来
    ((語幹            *)
     (基本形          来る      くる)
     (未然形          来        こ)
     (意志形          来よう    こよう)
     (命令形          来い      こい)
     (基本条件形      来れば    くれば)
     (基本連用形      来        き)
     (タ形            来た      きた)
     (タ系推量形      来たろう  きたろう)
     (タ系条件形      来たら    きたら)
     (タ系連用テ形    来て      きて)
     (タ系連用タリ形  来たり    きたり)
    )
)
```

A.4　連接規則ファイル (`chasen.connect`) の記述

A.4.1　連接規則の記述

```
〈連接規則〉          ::=  ((〈形態素構造の並び〉)(〈形態素構造の並び〉))  |
                          ((〈形態素構造の並び〉)(〈形態素構造の並び〉)〈連接コスト〉)
〈形態素構造の並び〉  ::=  〈形態素構造〉|〈形態素構造〉〈形態素構造の並び〉
〈形態素構造〉        ::=  (〈形態品詞名〉)  |(〈形態品詞名〉〈品詞細分類名〉)  |
                      (〈形態品詞名〉〈品詞細分類名〉〈活用型名〉)  |
                      (〈形態品詞名〉〈品詞細分類名〉〈活用型名〉〈活用形名〉)  |
                      (〈形態品詞名〉〈品詞細分類名〉〈活用型名〉〈活用形名〉〈見出し語〉)
```

〈形態品詞名〉〈品詞細分類名〉〈活用型名〉〈活用形名〉〈見出し語〉については，*も指定できる．
〈連接コスト〉は 0 から 255 までの整数値 (省略された場合はのデフォルト値は 10)．

A.4.2　連接規則の定義例

```
(((名詞)                            ; 名詞，指示詞の名詞形態指示詞などは
  (指示詞 名詞形態指示詞)           ; 判定詞および特殊の読点と連接可能
  (接尾辞 名詞性述語接尾辞)
  (接尾辞 名詞性名詞接尾辞)
  (接尾辞 名詞性名詞助数辞))
        ((判定詞)
         (特殊 読点))
)

(((名詞 サ変名詞)                            ; サ変名詞と接尾辞の「化」には，
  (接尾辞 名詞性名詞接尾辞 * * 化))         ;  「する」「できる」などが
        ((動詞 * サ変動詞 * する)           ; 後接可能である．
         (動詞 * 母音動詞 * できる)         ; また，その連接のコストは 5 であり
         (動詞 * 母音動詞 * 出来る))        ; 結合度が強い．
        5 )
```

B　形態素辞書の記述

B.1　形態素の記述

```
〈形態素定義〉            ::=  （〈#形態品詞名〉〈形態素情報の並び〉）|
                              （〈#形態品詞名〉（〈#品詞細分類名〉〈形態素情報の並び〉））
〈形態素情報の並び〉      ::=  〈形態素情報〉 |  〈形態素情報〉〈形態素情報の並び〉
〈形態素情報〉            ::=  （〈見出し語情報〉〈読み情報〉〈活用型情報〉〈意味情報〉）
〈見出し語情報〉          ::=  （見出し語 〈見出し語内容の並び〉）
〈見出し語内容の並び〉    ::=  〈見出し語内容〉 |  〈見出し語内容の並び〉
〈見出し語内容〉          ::=  〈#見出し語表記〉 |（〈#見出し語表記〉）|
                              （〈#見出し語表記〉〈#数値〉）
〈読み情報〉              ::=  （読み 〈#読み表記〉）
〈活用型情報〉            ::=  （活用型 〈#活用型名〉）| NIL
〈意味情報〉              ::=  （意味情報 〈#意味記述〉）| NIL
```

〈活用型情報〉については，活用すると定義された形態素に対しては省略することができない．
〈意味情報〉は，省略可能．

B.2　形態素の記述例

```
（名詞
  （普通名詞
    （（見出し語 日本語）
     （読み にほんご））
    （（見出し語 英語）
     （読み えいご））
  ）
）

（動詞
    （（見出し語（歩く）（あるく 1.5））
     （読み あるく）
     （活用型 子音動詞カ行）
     （意味情報 "walk(X)"）
    ）
）
```

C リソースファイル (.chasenrc) の記述例

```
(文法ファイル
        /usr/local/lib/chasen/dic
)
(PATDIC
        /usr/local/lib/chasen/dic/chadic
)
(未定義語品詞 (名詞 サ変名詞))
(品詞コスト
        ((*)             10)
        ((未定義語)    3000)
        ((特殊 *)       100)
        ((動詞)         100)
        ((形容詞)       100)
        ((判定詞)        10)
        ((助動詞)        10)
        ((名詞 *)       100)
        ((名詞 副詞的名詞) 99)
        ((名詞 固有名詞) 110)
        ((名詞 人名) 109)
        ((名詞 地名) 109)
        ((名詞 形式名詞) 10)
        ((指示詞 *)      80)
        ((副詞 *)       100)
        ((助詞 *)        10)
        ((接続詞)       100)
        ((連体詞)       100)
        ((感動詞)       100)
        ((接頭辞 *)      10)
        ((接尾辞 *)      10)
)

(連接コスト重み 3)
(形態素コスト重み 1)
(コスト幅 0)
(未定義連接コスト 100)
(連結品詞 (名詞 数詞))
```

D　インストール手順

茶筌システムが存在するディレクトリを `$CHASEN` とする．

1. `$CHASEN/chasenrc` をユーザのホームディレクトリに`.chasenrc` という名前でコピーする．
2. `.chasenrc` 内の「文法ファイル」に文法ファイルが存在するディレクトリ，「PATDIC」に辞書が存在するディレクトリを指定しておく．
3. `$CHASEN` ディレクトリに移動し，環境に応じて `BINDIR`，`LIBDIR`，`CC`，`CFLAGS`，`RANLIB` などの項目を書き換えた後 `make` を実行する．これによって茶筌システムが作られる．
4. `$CHASEN` ディレクトリで，`make dic` を実行する．これによってシステム辞書・インデックスファイルが作られる．作成に要する時間は，辞書の大きさとマシンの性能にもよるが，現在の辞書では，SPARCstation20 上で約 2 分である．現在の標準辞書により最終的に生成されるファイルは

`chadic.int`	約 6.5MB
`chadic.pat`	約 1.0MB

   となる．

E　JUMAN 2.0 から 茶筌 1.0 への拡張点

1. 辞書検索の方法を従来の NDBM を用いて疑似的に TRIE 構造を実現する方法から，独自開発のパトリシア木を用いたものに変更した．解析に必要な辞書のサイズが約 4 分の 1 に縮小した．また，辞書のコンパイル時間が 3～40 分の 1 になった．
2. 解析システムの見直しを行ない，高速化を図った．解析速度が約 8～11 倍になった (JUMAN 2.0 との比較)．
3. 多くのプラットフォームでインストール可能になるようにコードを書き直した．また，GNU C コンパイラ (gcc) だけでなく OS 付属の C コンパイラなどでもコンパイルできるようにした．
4. 日本語 EUC だけでなく，JIS(ISO-2022-JP) の文字列も解析できるようにした．
5. 未定義連接コストの導入により，未定義語の出力を減らすことができるようになった．
6. 連結品詞を定義できるようにし，最適パスを出力する時に，その品詞の単語を一単語に連結して表示するようにした．
7. 活用語尾の読みを定義できるようにすることにより，「来る」「得る」などの読みがひらがなで表示されるようになった．
8. 入力文を改行コードで区切るのではなく，句点により区切るオプション (`-j`) を追加した．
9. `-r` オプションや環境変数 `CHASENRC` でリソースファイル (`.chasenrc`) を指定できるようにした．
10. `-F` オプションや `.chasenrc` の「出力フォーマット」で解析結果の出力形式を変更できるようにした．

11. 文法の見直しを行ない，品詞分類「特殊」の下の「括弧」を「括弧開」と「括弧閉」に分離した．また，同じく「特殊」の下に「空白」を定義した．「空白」は具体的には全角の空白を表す．
12. 助動詞の活用型に「助動詞べきだ型」を追加した．助動詞「べきだ」の活用を従来の「サ形容詞」型から「助動詞べきだ型」に変更した．
13. 辞書登録語について見直し，追加削除等の修正を行なった．

F JUMAN 3.0 と 茶筌 との関係について

JUMAN2.0 が 1994 年 7 月にリリースされて以降，京都大学長尾研究室と奈良先端大松本研究室では，それぞれ異なる方向での拡張を試みていました．京都大学では，従来の bi-gram モデルでは記述できない連接関係を記述するために連語処理や括弧の透過処理などの機能を追加し，文法辞書，形態素辞書に大幅な修正を行なった拡張版を作成していました．奈良先端大では，今後大量の蓄積が始まると思われる日本語タグ付きコーパスから bi-gram 以上の連接規則 (単語レベルや品詞レベルの設定も含む) を自動的に学習する機能を追加するための拡張と，UNIX のハッシュデータベース NDBM に依存しない辞書の構築を考えていました．後者の拡張は UNIX 以外の OS での稼働を要求する声に対応することと辞書のコンパイル時間と検索速度の改善を目指したことによります．bi-gram 以上の連接規則に対する両者の考え方がかなり異なるため，両者の融合は見合わせることにし，いち早く完成した京都大学の拡張版が 1996 年 6 月に JUMAN3.0beta として公開されました．

奈良先端大で拡張を予定していた機能には下に示すような項目がありますが，今回の茶筌 1.0 の公開は，JUMAN2.0 との外部仕様の変更が伴わない 1. から 5. の変更点だけを含むものです．これは，学習機能の実装にはまだ時間がかかることと，従来の JUMAN2.0 において接続辞書や形態素辞書の追加修正を行なった利用者への便宜を図る意味もあります．コストが同点の解についての出力が JUMAN2.0 とは微妙に異なることがありますが，茶筌 1.0 は JUMAN2.0 と外部仕様は同一です．

1. 辞書システムの独自開発 (NDBM の棄却)
2. 解析システムの見直しと高速化
3. 未定義連接コスト，連接品詞，解析結果出力フォーマットの導入
4. JIS 文字列の解析
5. 活用語尾の読みの定義
6. 品詞定義の多階層化 (現在は 2 階層)
7. 接続規則の可変長化 (現在は bi-gram)
8. ライブラリ化、サーバ化
9. Windows95 への対応
10. 正解データ提示による接続コストの調整支援
11. タグ付きコーパスからの接続モデルと接続コスト学習

拡張項目の 6. 以降は形態素辞書および接続辞書の記述法の変更を伴うので，次の版としてリリースする予定です．