CHISE のデータ形式 (Ver.0.1)

守岡 知彦

2017 年 8 月 21 日

目次

第 1 章

はじめに

CHISE の文字データの形式について概説する。

CHISE の文字データは、現在、XEmacs CHISE 附属の文字定義と CHISE 漢字構造情報データベースがあり、両者が合わさって CHISE 文字オントロジーを構成している。前者は define-char 形式という Lisp の S 式で記述されたソースコード、後者は IDS 形式のテキストファイルがマスターファイルとなっている。前者が CHISE の文字データの基幹部分を構成するもので、後者はそれを補完している。

この前者のデータモデルは XEmacs CHISE の前身の XEmacs UTF-2000 の時代から幾つかの変遷・拡張を遂げている。ここでは、CHISE の文字モデルの基礎となる Chaon モデル、UTF-2000 の時代でのフラットな異体字管理モデル、CHISE で導入された階層的な包摂関係の表現と多粒度漢字構造モデルについて概説する。

第 2 章

文字のオブジェクト化

CHISE project の前身である UTF-2000 プロジェクトで目標としたのは文字符号に依存しない文字処理の実現であった。UTF-2000 プロジェクトではこれをオブジェクト指向技術に基づく実装とインターフェースの分離によって解決しようとした。即ち、文字オブジェクトによって文字コードという実装を隠蔽し、文字を操作するインターフェース (API) を提供することによって、文字コードの符号値を使わずに文字処理を実現することを目指した訳である。

2.1 Chaon モデル

CHISE では（その前身である UTF-2000 の頃から）『Chaon モデル』と呼ぶ方法によって文字を表現するようになっている。これは汎用符号化文字集合に依存することなく自由に文字を表現するために我々が提案しているもので、表現したい文字に関する知識（文字の性質の集合）の機械可読な表現によって文字を表現し操作する方法である。

Chaon モデルでは、文字を説明するための要素（文字の性質や用例など）を『文字素性』(character feature) と呼ぶ。文字素性としては、部首、画数、部品の組合せ方に関する情報（漢字構造情報)、発音、意味、用例、その他文字処理で必要となる各種情報などが考えられる。

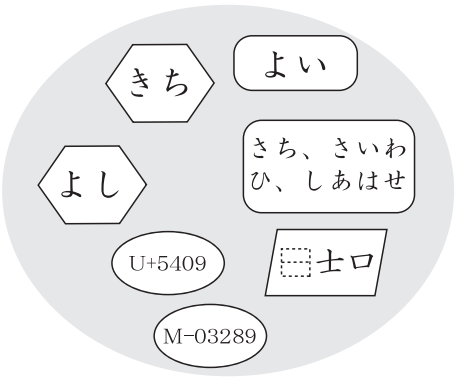

図 2.1　Chaon モデルにおける文字オブジェクトの概念図

2.1.1 S 式での表現

XEmacs CHISE では、Chaon モデルに基づき、文字（または、文字の集合を表す『文字状況』）を素性対の集合によって表現している。素性対は素性名と値の対であり、キー部が素性名を表すシンボルである cons 対で表現することができる。また、素性対の集合は連想リストで表現することができる。この連想リストはこれによって表現される文字の性質を表現したものと考えることができる。そこで、このような連想リストを『文字指定 (char-spec)』と呼ぶこと

にする。

文字指定は概念的には文字であるが Lisp 的にはリストの一種であり、そのままでは XEmacs における文字型オブジェクトとは見做されないことになる。このため XEmacs CHISE では、文字指定の集合を文字データベースに格納するとともに、そこでの文字指定への参照である “system-char-id” を文字型オブジェクトの内部表現とし、文字データベースを介して文字素性ないしはその集合である文字指定と文字型オブジェクトを結びつけている。

結局、概念的には CHISE 文字データベースは連想リスト（文字指定）の配列と考えることができ、そのインデックスが system-char-id である。system-char-id は従来型の文字の内部コードのようなものを必要とする時にその代用品として用いることができ、それによって従来型文字処理プログラムを最小限の労力で Chaon 実装化することができる。

2.1.2　RDF での表現

オブジェクトの表現

CHISE の文字オブジェクトは Chaon モデルに基づき素性対の集合で表現される。この各素性対は RDF のトリプルに対応するものといえ、文字オブジェクトを主語、素性名を述語、素性値を目的語とした RDF のトリプルとして表現することができる（図 2.2)。ここで CHISE の素性名に対応した述語を『素性述語』と呼ぶことにする。

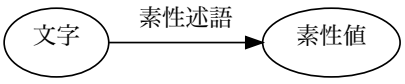

図 2.2　素性対の表現

ここで文字オブジェクトをどのように表現するかという問題が生じる。Chaon モデルは文字オブジェクトを素性対の集合という確定記述の束として、文字を特定の文字符号におけるコードポイントのような『固有名』に依らずに、表現するものといえるが、RDF では原則として IRI という固有名によって情報資源を指示する枠組を採っている。しかしながら、CHISE でも各種文字符号を表現するために ID 素性というものがあり、また、RDF にも空白ノードというそれ自体は IRI を持たず他の情報資源との関係性によって指示可能となるものもある。

Chaon モデルは集合論的なモデルであり、文字オブジェクトはそれが持つ素性対の指示対象の共通部分（論理積）であることを意味している。よって、素性対に対応する IRI を構成できるならば owl:intersectionOf を用いて表現することができる。

また、文字オブジェクトが複数の ID 素性を持つ場合、それらの ID 素性対は同じもの（即ち、その ID 素性を持つ文字オブジェクト）を指示するという性質がある。この場合、これらの ID 素性対に対応する IRI 構成できるならば、各 IRI 間の指示対象が同じことを owl:sameAs を用いて表現することができる。

Chaon モデル本来のセマンティクスを忠実に表現するという観点ではオブジェクトは原則として空白ノードにするべきであるかも知れないが、データへのアクセスという観点では繁雑であるといえ、CHISE-wiki [5] (EgT [6]) で用いているような文字オブジェクトに含まれる ID 素性を用いて IRI を構成する方法が現実的であるといえる。この詳細は 2.5.1 節で述べる。

素性名の表現

素性名は RDF の述語に対応するものであり、IRI で表現することができる。

CHISE において、各素性名は文字オブジェクトの世界の中での固有性が保証されているので、CHISE 用の述語を置

集合名	名前空間接頭辞	IRI
基本述語集合	`:`	`http://rdf.chise.org/rdf/property/character/main/`
漢字関連述語集合	`ideo:`	`http://rdf.chise.org/rdf/property/character/ideo/`
漢字構造記述用述語集合	`isd:`	`http://rdf.chise.org/rdf/property/character/isd/`

表 2.1 述語の名前空間

文字素性名	素性述語
ideographic-radical	ideo:radical
ideographic-strokes	ideo:strokes
total-strokes	ideo:total-strokes
->subsumptive	:subsume
<-denotational	:denotation-of
<-formed	:form-of
name	:name

表 2.2 主な素性名と述語

くための名前空間 IRI を定め、その下に素性名を付ければ良い訳であるが、CHISE の素性名で使用可能な文字の範囲の中には IRI でエスケープしなければならないものが含まれるため、可読性を考慮し CHISE-wiki における素性名の URL 表現 [5][*1]（以下では、この変換を『(素性名の）CHISE-wiki 符号化』と呼ぶことにする） を用いることにする。

CHISE の述語用名前空間は 1 つあれば十分であるが、Turtle 記法等において接頭辞を使用した場合にローカルパートが短く判り易いものになることと対象や用途の種類毎に述語の集合をまとめることを意図して、表 2.1 に示す 3 種類の述語集合を設けた。但し、この名前空間接頭辞は CHISE 関連の Turtle 文書でのもので、CHISE 関連以外の一般の文書で用いる場合は、その前に `chise` を付けることにする。

また、RDF の世界で標準的に使われる述語が存在する場合、適宜それを用いることとする。なお、他に適切な述語集合が存在しない場合、CHISE の基本述語集合の IRI 以下に CHISE-wiki 符号化した素性名を置いたものを用いるものとする。表 2.2 に主な素性名と述語の対応表を示す。

2.2 文字素性の操作

以下では、まず XEmacs CHISE における Emacs Lisp API を示し、次に対応する CHISE-wiki (EgT) の API を示すことにする。

2.2.1 文字定義

XEmacs CHISE は文字素性の集合、即ち、char-spec で文字オブジェクトを定義する Emacs Lisp API を提供している。

関数 **define-char** (*char-spec*)

文字素性対の集合 *char-spec* で表現される文字オブジェクトを定義し、その文字オブジェクトを返す。

[*1] ID 素性の表現に関しては、包摂粒度の分類体系が後に変更されたために、[5] での記述から変更されており、現在は表 2.6 の「IRI」列に示すものを用いている。

[例]

```
(define-char
  '((ideographic-radical . 134)
    (ideographic-strokes . 6)
    (total-strokes  . 12)
    (=ucs-itaiji-002 . #x2696F)
    ))
```

→ ?𦥯

現在の所、文字定義に関する Web API は存在していない。

2.2.2　文字素性値の参照

関数 **get-char-attribute** (*character feature* ***&optional*** *default-value*)

文字オブジェクト *character* の素性 *feature* の値を返す。
もし、値が定義されていない場合、*default-value* を返す。なお、*default-value* の既定値は nil である。

[例]

```
(get-char-attribute ?あ 'name)
→ "HIRAGANA LETTER A"
```

また、3.3 節で述べる関数 char-feature も同様な機能を提供し、こちらの方がより一般的な関数である。

CHISE-wiki (EgT) において文字素性値の参照機能に相当するのは

`http://www.chise.org/est/view/` 文字

の頁を開いた時の該当する素性の情報の表示といえる。また、

`http://www.chise.org/est/view/` 文字`/feature=`素性名

を開くことにより、該当する素性の情報だけを表示することも可能である。

[例]

`http://www.chise.org/est/view/character/`字`/feature=total-strokes`

2.2.3　文字素性値の設定

関数 **put-char-attribute** (*character feature value*)

文字オブジェクト *character* の素性 *feature* の値を *value* に設定する。

[例]

```
(get-char-attribute ?あ 'foo)
→ nil
(put-char-attribute ?あ 'foo 1)
→ 1
(get-char attribute ?あ 'foo)
→ 1
```

なお、関数 put-char-attribute は起動中の XEmacs CHISE のプロセス内のオンメモリ・データベースの内容を書き換えるだけであり、そのままでは他のプロセスと共有できないし、プロセスを終了すればその変更結果は消えてしまう。もし、この変更結果を保存したい場合は 2.2.7 節での述べる関数 save-char-attribute-table を使って、変更結果を永続化する必要がある。

CHISE-wiki (EgT) において文字素性値の参照機能に相当するのは素性値の編集機能であるが、今の所、外部公開を行っていない。

2.2.4 文字素性の削除

関数 **remove-char-attribute** (*character feature*)

文字オブジェクト *character* の素性 *feature* を削除する。

なお、関数 remove-char-attribute も、関数 put-char-attribute と同様、起動中の XEmacs CHISE のプロセス内のオンメモリ・データベースの内容を書き換えるだけであり、そのままでは他のプロセスと共有できないし、プロセスを終了すればその変更結果は消えてしまう。もし、この変更結果を保存したい場合は 2.2.7 節での述べる関数 save-char-attribute-table を使って、変更結果を永続化する必要がある。

CHISE-wiki (EgT) において文字素性値の参照機能に相当するのは素性値の編集機能であるが、今の所、外部公開を行っていない。

2.2.5 文字オブジェクトの情報の一括取得

関数 **char-attribute-alist** (*character*)

文字 *character* に対応する文字指定を返す。

CHISE-wiki (EgT) においてこの機能に相当するのは `http://www.chise.org/est/view/` 文字 の頁の閲覧といえる。

2.2.6 文字素性のイタレーター

関数 **map-char-attribute** (*function feature*)

文字素性名 *feature* を持つ各文字に対し、関数 *function* を適用する。
この関数は 2 引数の関数であり、この関数が呼ばれる時、第 1 引数に文字が渡され、第 2 引数に文字素性値が渡される。また、この関数が 非-nil 値を返すと、関数 map-char-attribute の実行はその時点で停止される。

現在の所、文字素性に対するイタレーター機能を提供する Web API は存在していない。

2.2.7 文字素性に関するその他の API

以下は全て、現在の所、 XEmacs CHISE の Emacs Lisp API のみが提供されている。

関数 **find-char** (*char-spec*)

文字素性の集合 *char-spec* で表現される文字オブジェクトを探索し、見つかった場合はそれを返す。見つからなかった場合は nil を返す。

関数 **char-attribute-list** ()

現在までに使われた文字素性名の一覧を返す。

XEmacs CHISE は CHISE 環境で共有される文字データベースの他にプロセス内部のオンメモリ・データベースを持っている。オンメモリ・データベースに存在しない文字データは最初に必要となった時に共有文字データベースからオンメモリ・データベースに読み込まれる。また、オンメモリ・データベースに書き込まれた情報はそのままでは共有文字データベースに書き込まれず、そのプロセスの外部からは参照できないし、プロセスの終了とともに消滅することになる。そのため、オンメモリ・データベースを永続化するための機能が用意されている：

関数 **save-char-attribute-table** (*feature*)

各文字の文字素性 *feature* の値を CHISE 文字データベースに書き出す。

また、プロセス内部の文字データベースにおける情報を破棄して、CHISE 文字データベースの情報を読み直せるようにするための機能も用意されている：

関数 **reset-char-attribute-table** (*feature*)

プロセス内部の文字データベースから各文字の文字素性 *feature* の値を消去し、CHISE 文字データベースの情報を読み直せるようにする。

関数 **reset-charset-mapping-table** (*coded-charset*)

プロセス内部にある *coded-charset* の decoding-table を破棄し、CHISE 文字データベースの情報を読み直せるようにする。

また、文字素性に関する情報を一気に読み込むための機能も用意されている：

関数 **load-char-attribute-table** (*feature*)

CHISE 文字データベースから各文字の文字素性 *feature* の値を読み込む。

この他、文字素性を陽に登録するための機能も存在する：

関数 **mount-char-attribute-table** (*feature*)

CHISE 文字データベースから文字素性 *feature* を読み込めるようにする。

2.3　文字間の関係の記述

Chaon モデルは文字素性の集合として文字（の集合）を表現（指示）する手法であり、そのモデルそのものには文字間の関係で構成されるネットワーク構造を表現する仕組みは存在しない。しかしながら、文字間の関係を表す文字素性

関係素性	述語
`->subsumptive`	:subsume
`<-denotational`	:denotation-of
`<-formed`	:form-of
`<-same`	ideo:same-as
`<-simplified`	ideo:simplified-form-of
`<-vulgar`	ideo:vulgar-form-of
`<-wrong`	ideo:wrong-form-of
`<-original`	ideo:original-form-of
`<-ancient`	ideo:ancient-form-of
`<-Small-Seal`	ideo:Small-Seal-of
`<-interchangeable`	ideo:interchangeable-form-of
`->interchangeable`	ideo:interchangeable
`->mistakable`	ideo:mistakable

表 2.3 関係素性名の RDF 述語名

を導入し、その値として文字（の集合）（の集合）を取るようにすれば、文字間の関係で構成されるネットワーク構造を記述することができる。ここで文字間の関係を表す文字素性のことを関係素性と呼ぶ。

CHISE では関係素性を表すのに –>*foo* と <–*bar* という先頭 2 文字が `->` または `<-` とする文字素性名を用いることにしている（但し、2.4 節で述べる文字素性メタデータ名は除く）。ちなみに、これは矢印を表現したものである。こうした文字素性名のことを関係素性名と呼ぶことにする。

関係素性の性質は次のように定義されている：

文字 A, B が存在する時、A の文字素性 (–>*foo* ... B ...) は、関係 A –>*foo* B が存在することを意味する。この時、文字 B には逆関係を表す文字素性 (<–*foo* ... A ...) が存在する。

関係素性の値には文字の集合（リスト）を取ることになっており、(–>*foo* B C D) などのように書くことができる。この場合、定義される文字を A とする時、関係 A –>*foo* B と関係 A –>*foo* C と関係 A –>*foo* D が存在することになる。

HTML, XML では `<`, `>` をエスケープする必要があるので、CHISE-wiki (EgT) では次に述べるような関係素性の IRI (URL) 中における表現形式を設けることにする：

関係素性 `<-`*foo* の URI 中での表現は `from.`*foo* とする。

関係素性 `->`*foo* の URI 中での表現は `to.`*foo* とする。

RDF では、RDF の世界において標準的な述語名があればそれを採用し、また、RDF 的により自然な述語名を用いることを目指しているが、RDF では CHISE の関係素性名のような逆関係間の述語名の規則的な対応規則がないため、機械的に変換することができず、現在の所作業途上である。現在の対応関係を表 2.3 に示す。

このため、フォールバック規則として上述の CHISE-wiki (EgT) における URL 表現を用いる。

2.4 階層的素性名

汎用的な文字データベースを作る場合、用途や立場・学説などによって、文字素性の値に複数の選択肢を設けたくなる場合がある。こういう時、単純に複数の値が記述できるだけでなく、各々の値の出典情報などのメタデータも付加し

たいことが少なくない。こうした場合、文字素性の値か名前のどちらかを構造化する必要がある。『階層的素性名』方式は後者の方法の一種である。

階層的素性名は構造化の対象となる文字素性の名前（文字素性基底名）に値を選択するための識別子（『ドメイン識別子』と呼ぶ）を付けた文字列を生成し、それを名前（文字素性具象名）として用いたり、同様にメタデータ識別子を付けた文字列を生成しそれを名前（文字素性メタデータ名）として用いる方法である。この名前は次のような規則で生成される：

<character feature name>
　:= <concrete feature name> | <metadata feature name>

<concrete feature name>
　:= <base feature name>
　| <base feature name> @ <domain identifier>

<metadata feature name>
　:= <concrete feature name> * <metadata identifier>
　| <metadata feature name> * <metadata identifier>
　| <metadata feature name> @ <domain identifier>

<domain identifier>
　:= <base domain identifier>
　| <domain identifier> / <base domain identifier>

例えば、総画数を表す文字素性名を `total-strokes` とし、ドメイン識別子として `ucs` を用いる時、文字素性具象名は `total-strokes@ucs` となる。また、出典情報を表すメタデータ識別子を `sources` とする時、`total-strokes@ucs` の出典情報は

```
total-strokes@ucs*sources
```

で表される。

部首と部首内画数のように異なる種類の文字素性の値が対応関係を持っている場合、ドメイン識別子を用いてその対応関係を表すことができる。例えば、部首を `ideographic-radical`, 部首内画数を `ideographic-strokes` で表す時、

```
ideographic-radical@ucs
ideographic-strokes@ucs
```

の両者は対応する。

CHISE-wiki (EgT) では IRI (URL) 中で用いる階層的素性名中の特殊文字を次の規則で置き換える（これを『URL 表現』と呼ぶことにする）：

- `*` → `.-.`
- `/` → `...`

例えば、階層的素性名 `sound@ja/on` の URL 表現は `sound@ja...on` になる。また、階層的素性名

`total-strokes@ucs*sources` の URL 表現は `total-strokes@ucs.-.sources` となる。

また、RDF では、階層的な文字素性具象名が使われている場合、ベースとなる素性名に対応した述語がとる目的語として空白ノードを設け、その空白ノードの述語 `:context` で階層的素性名におけるドメイン情報を表現する。また、述語 `:target` で素性値を表現する（図 2.3）。

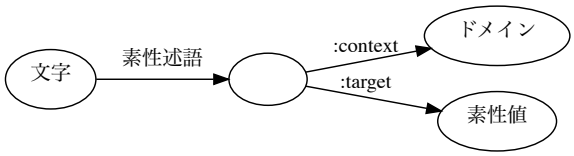

図 2.3 ドメイン付き素性名の表現

例えば、文字「𠂉」の UCS における部首を示す素性 ideographic-radical@ucs の値が 4 である時、この素性のベースとなる素性名 ideographic-radical に対応する素性述語 ideo:radical と空白ノードを使って図 2.4 のように表現することができる。

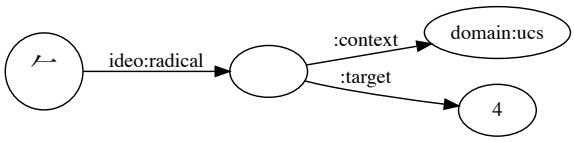

図 2.4 ドメイン情報付きの RDF の例

複数のドメイン識別子が存在する場合に空白ノードの述語 `:context` の値を RDF コレクションやコンテナ等を使って構造化することも考えられるが、当面、ドメイン情報記述用の名前空間 IRI として `http://rdf.chise.org/data/domain/` を設け、この下に CHISE-wiki 符号化したドメイン名を付けた IRI で表現することにする。ドメイン間の階層関係等の情報はこの IRI を起点とした RDF グラフで表現することにする。なお、文字定義の Turtle 文書ではこのドメイン情報記述用の名前空間の接頭辞として `domain:` を用いることにする（CHISE 関連以外の一般の文書で用いる場合は `chisedomain:` とする）。

メタデータ素性（文字素性メタデータ名）で表現される情報も、階層的な文字素性具象名の表現と同様に、素性述語の目的語に空白ノードを設けて表現することにする。

階層的な文字素性具象名の場合、空白ノードの述語 `:context` で階層的素性名におけるドメイン情報を表現し、同じく述語 `:target` で素性値を表現したが、メタデータ素性の場合、「*」以降のメタデータ識別子を CHISE-wiki 符号化したものをその空白ノードの述語として表現することにする。ここで、このメタデータ識別子に対応する述語のことを『メタデータ述語』と呼ぶことにする。

もしメタデータ識別子にドメインが付いている場合、階層的素性名の表現方法に従いその目的語を空白ノードとしてドメイン情報をその空白ノードの述語 `:context` で表現する。

メタデータ素性の値の RDF での表現はメタデータ識別子に依存する。出典情報を示す `:sources` の場合は、素性値の各要素のシンボルを CHISE-wiki 符号化したものの前に `http://rdf.chise.org/data/bibliography/` を付け

た IRI を目的語とする。また、ここで、この名前空間 IRI `http://rdf.chise.org/data/bibliography/` に対する接頭辞を `chisebib:` とする。また、デフォルトでは次節で述べる基礎素性の表現に準じるものとする。

2.5　文字符号の扱い

文字をオブジェクト化し、Chaon モデルに基づき、文字素性の集合として表現することで、文字の内部表現を隠蔽することが可能である。つまり、どれかの文字符号を内部表現として使用しなくても文字処理が可能となる訳である。

しかしながら、現実には UCS や JIS X 0208, GB 2312 といった文字符号が情報交換において広く使われており、こうした汎用的な符号化文字集合は文字に対する重要なインターフェースの一つということができる。また、大漢和辞典の番号のようなものも同様に考えることができる。

そこで、これらを文字素性の一種として扱うことにする。これを『ID 素性』と呼ぶ。

ID 素性は文字符号の種類を示す識別子（文字符号識別子）と ID 素性であることを示す接頭辞によって示される。[*2]

Lisp の世界では、この接頭辞として `=` を用いる。例えば、UCS は文字符号識別子が `ucs` であるので、ID 素性名は `=ucs` となる。

Web の世界 (CHISE-wiki (EgT), RDF) では、この接頭辞として `rep.` を用いる。例えば、UCS の場合、`rep.ucs` となる。

表 2.4 に文字符号識別子の一覧を示す。

表 2.4: 文字識別子一覧

ISO-IR	識別子	説明
42	jis-x0208@1978/1pr	JIS X 0208:1978, 1st impression
58	gb2312	GB 2312
87	jis-x0208@1983	JIS X0208:1983
149	ks-x1001	KSC5601 Korean Hangul and Hanja
159	jis-x0212	JIS X0212 Japanese Supplement
165	iso-ir165	ISO-IR-165 (CCITT Extended GB)
168	jis-x0208@1990	JIS X0208:1990
171	cns11643-1	CNS 11643 Plane 1
172	cns11643-2	CNS 11643 Plane 2
176	ucs-bmp	ISO/IEC 10646 Plane 0 (BMP)
177	ucs	ISO/IEC 10646
183	cns11643-3	CNS 11643 Plane 3
184	cns11643-4	CNS 11643 Plane 4
185	cns11643-5	CNS 11643 Plane 5
186	cns11643-6	CNS 11643 Plane 6
187	cns11643-7	CNS 11643 Plane 7
228	jis-x0213-1@2000	JIS X 0213:2000 Plain 1
229	jis-x0213-2	JIS X 0213 Plain 2
233	jis-x0213-1@2004	JIS X 0213:2004 Plain 1
—	jis-x0208	JIS X0208 Common part

[*2] 現在の CHISE では ID 素性であることを示す接頭辞を拡張して 5.2 節で述べるような包摂粒度の種類を示す接頭辞が使われている。

表 2.4: 文字識別子一覧

—	gb12345	GB 12345:1990
—	big5	Big5
—	adobe-japan1-base	Adobe-Japan1
—	adobe-japan1-0	Adobe-Japan1-0
—	adobe-japan1-1	Adobe-Japan1-1
—	adobe-japan1-2	Adobe-Japan1-2
—	adobe-japan1-3	Adobe-Japan1-3
—	adobe-japan1-4	Adobe-Japan1-4
—	adobe-japan1-5	Adobe-Japan1-5
—	adobe-japan1-6	Adobe-Japan1-6
—	adobe-japan1	Adobe-Japan1 (= Adobe-Japan1-6)
—	jis-x0208@1978	JIS X 0208:1978, unchanged part
—	jis-x0208@1978/-4pr	JIS X 0208:1978, 1st-3rd impressions
—	jis-x0208@1978/1pr/fixed	JIS X 0208:1978, correct glyph in the errata of 1st impression
—	jis-x0208@1978/-4X	JIS X 0208:1978, index before the 4th impression
—	jis-x0208@1978/2-pr	JIS X 0208:1978, 2nd impression or later
—	jis-x0208@1978/4er	JIS X 0208:1978, replaced by errata of 4th impression
—	jis-x0208@1978/4-pr	JIS X 0208:1978, 4th impression or later
—	jis-x0208@1978/5pr	JIS X 0208:1978, 5th impression
—	jis-x0213-1	JIS X 0213 Plain 1 (unchanged part)
—	big5-eten	Big5 ETEN
—	big5-cdp	Big5 with CDP extension
—	mj	Moji-Jouhou-Kiban characters
—	mj-0	Moji-Jouhou-Kiban plane 0
—	mj-1	Moji-Jouhou-Kiban plane 1
—	hanyo-denshi/ja	JA (JIS X0208) part of Han'you-Denshi
—	hanyo-denshi/jb	JB (JIS X0212) part of Han'you-Denshi
—	hanyo-denshi/jc	JC (JIS X0213:2000 Plane 1) part of Han'you-Denshi
—	hanyo-denshi/jd	JD (JIS X0213:2000 Plane 1) part of Han'you-Denshi
—	hanyo-denshi/ft	FT (FDPC additional) part of Han'you-Denshi
—	hanyo-denshi/ia	IA part of Han'you-Denshi
—	hanyo-denshi/ib	IB part of Han'you-Denshi
—	hanyo-denshi/hg	HG (HyouGai Kanji) part of Han'you-Denshi
—	hanyo-denshi/ip	IP (for IPA) part of Han'you-Denshi
—	hanyo-denshi/jt	JT (Juuki Touitsu Moji) part of Han'you-Denshi
—	hanyo-denshi/ks	KS (KoSeki) part of Han'you-Denshi
—	hanyo-denshi/tk	TK (TouKi) part of Han'you-Denshi
—	koseki	KoSeki-touitsu-moji
—	hanyo-denshi/ks/mf	font encoding for KS (KoSeki)
—	hanyo-denshi/tk/mf-01	font encoding for TK (TouKi)

表 2.4: 文字識別子一覧

—	gt	GT 2000
—	gt-k	Ideographic components of GT
—	gt-pj-1	GT 2000 (pseudo JIS encoding) part 1
—	gt-pj-2	GT 2000 (pseudo JIS encoding) part 2
—	gt-pj-3	GT 2000 (pseudo JIS encoding) part 3
—	gt-pj-4	GT 2000 (pseudo JIS encoding) part 4
—	gt-pj-5	GT 2000 (pseudo JIS encoding) part 5
—	gt-pj-6	GT 2000 (pseudo JIS encoding) part 6
—	gt-pj-7	GT 2000 (pseudo JIS encoding) part 7
—	gt-pj-8	GT 2000 (pseudo JIS encoding) part 8
—	gt-pj-9	GT 2000 (pseudo JIS encoding) part 9
—	gt-pj-10	GT 2000 (pseudo JIS encoding) part 10
—	gt-pj-11	GT 2000 (pseudo JIS encoding) part 11
—	gt-pj-k1	Ideographic parts of GT (pseudo JIS encoding) part 1
—	gt-pj-k2	Ideographic parts of GT (pseudo JIS encoding) part 2
—	daikanwa	Daikanwa dictionary (unchanged part)
—	daikanwa@rev1	Daikanwa dictionary (revised version)
—	daikanwa@rev2	Daikanwa dictionary (revised version 2)
—	daikanwa/+p	Daikanwa dictionary (ddddd')
—	daikanwa/+2p	Daikanwa dictionary (ddddd")
—	daikanwa/ho	Daikanwa dictionary (Hokan)
—	shinjigen	Kadokawa ShinJigen dictionary (common parts)
—	shinjigen@1ed	Kadokawa ShinJigen dictionary (the first edition)
—	shinjigen@1ed/24pr	Kadokawa ShinJigen dictionary (the 24th impression)
—	shinjigen@rev	Kadokawa ShinJigen dictionary (the revised edition)
—	shinjigen/+p@rev	ShinJigen (the second edition) number with '
—	big5-pua	Big5 with private used area
—	hanziku-1	HANZIKU (pseudo BIG5 encoding) part 1
—	hanziku-2	HANZIKU (pseudo BIG5 encoding) part 2
—	hanziku-3	HANZIKU (pseudo BIG5 encoding) part 3
—	hanziku-4	HANZIKU (pseudo BIG5 encoding) part 4
—	hanziku-5	HANZIKU (pseudo BIG5 encoding) part 5
—	hanziku-6	HANZIKU (pseudo BIG5 encoding) part 6
—	hanziku-7	HANZIKU (pseudo BIG5 encoding) part 7
—	hanziku-8	HANZIKU (pseudo BIG5 encoding) part 8
—	hanziku-9	HANZIKU (pseudo BIG5 encoding) part 9
—	hanziku-10	HANZIKU (pseudo BIG5 encoding) part 10
—	hanziku-11	HANZIKU (pseudo BIG5 encoding) part 11
—	hanziku-12	HANZIKU (pseudo BIG5 encoding) part 12
—	cbeta	CBETA private characters

表 2.4: 文字識別子一覧

—	zinbun-oracle	Oracle Bones Script
—	jef-china3	JEF + CHINA3 private characters
—	ruimoku-v6	private characters used in RUIMOKU Version.6
—	big5-cdp-itaiji-001	Big5-CDP-itaiji-001 [GlyphWiki]
—	big5-cdp-itaiji-002	Big5-CDP-itaiji-001 [GlyphWiki]
—	big5-cdp-itaiji-003	Big5-CDP-itaiji-001 [GlyphWiki]
—	big5-cdp-var-001	Big5-CDP-var-001 [GlyphWiki]
—	big5-cdp-var-002	Big5-CDP-var-002 [GlyphWiki]
—	big5-cdp-var-003	Big5-CDP-var-003 [GlyphWiki]
—	big5-cdp-var-004	Big5-CDP-var-004 [GlyphWiki]
—	big5-cdp-var-005	Big5-CDP-var-005 [GlyphWiki]
—	ucs@iso	Rep glyphs of ISO/IEC 10646-2 (UCS2003)
—	ucs@unicode	Rep glyphs of Unicode 3.2
—	ucs@gb	ISO/IEC 10646 for GB
—	ucs@gb/fw	ISO/IEC 10646 for GB with fullwidth
—	ucs-bmp@gb	ISO/IEC 10646 BMP for GB
—	ucs@cns	ISO/IEC 10646 for CNS 11643
—	ucs@cns/fw	ISO/IEC 10646 for CNS 11643 with fullwidth
—	ucs-bmp@cns	ISO/IEC 10646 BMP for CNS
—	ucs@jis	ISO/IEC 10646 for JIS X0208/0212/0213
—	ucs@jis/fw	ISO/IEC 10646 for JIS X0208/0212/0213 with fullwidth
—	ucs-bmp@jis	ISO/IEC 10646 BMP for JIS
—	ucs@jis/1990	ISO/IEC 10646 for JIS X 0208/0212:1990
—	ucs@jis/2000	ISO/IEC 10646 for JIS X 0213:2000
—	ucs@jis/2004	ISO/IEC 10646 for JIS X 0213:2004
—	ucs@JP	UCS for common glyphs used in Japan
—	ucs@JP/hanazono	Japanese glyph-images of GlyphWiki
—	ucs@ks	ISO/IEC 10646 for Korean Standards
—	ucs@ks/2012	Mapping for Korean Standards in ISO/IEC 10646:2012
—	ucs-bmp@ks	ISO/IEC 10646 BMP for KS
—	ucs@cns11643	ISO/IEC 10646 for CNS based on `www.cns11643.gov.tw`
—	ucs@big5	ISO/IEC 10646 for Big5
—	ucs@big5/cns11643	ISO/IEC 10646 for Big5 based on `www.cns11643.gov.tw`
—	ucs-var-001	uXXXX(X)-var-001 of GlyphWiki
—	ucs-bmp-var-001	uXXXX-var-001 in BMP
—	ucs-sip-var-001	uXXXXX-var-001 in SIP
—	ucs-var-002	uXXXX(X)-var-002 of GlyphWiki
—	ucs-bmp-var-002	uXXXX-var-002 in BMP
—	ucs-sip-var-002	uXXXXX-var-002 in SIP
—	ucs-var-003	uXXXX(X)-var-003 of GlyphWiki

表 2.4: 文字識別子一覧

—	ucs-bmp-var-003	uXXXX-var-003 in BMP
—	ucs-sip-var-003	uXXXXX-var-003 in SIP
—	ucs-var-004	uXXXX(X)-var-004 of GlyphWiki
—	ucs-bmp-var-004	uXXXX-var-004 in BMP
—	ucs-sip-var-004	uXXXXX-var-004 in SIP
—	ucs-var-005	uXXXX(X)-var-005 of GlyphWiki
—	ucs-bmp-var-005	uXXXX-var-005 in BMP
—	ucs-var-006	uXXXX(X)-var-006 of GlyphWiki
—	ucs-bmp-var-006	uXXXX-var-006 in BMP
—	ucs-sip-var-006	uXXXXX-var-006 in SIP
—	ucs-var-008	uXXXX(X)-var-008 of GlyphWiki
—	ucs-bmp-var-008	uXXXX-var-008 in BMP
—	ucs-var-010	uXXXX(X)-var-010 of GlyphWiki
—	ucs-bmp-var-010	uXXXX-var-010 in BMP
—	ucs-itaiji-001	uXXXX(X)-itaiji-001 of GlyphWiki
—	ucs-bmp-itaiji-001	uXXXX-itaiji-001 in BMP
—	ucs-sip-itaiji-001	uXXXXX-itaiji-001 in SIP
—	ucs-itaiji-002	uXXXX(X)-itaiji-002 of GlyphWiki
—	ucs-bmp-itaiji-002	uXXXX-itaiji-002 in BMP
—	ucs-sip-itaiji-002	uXXXXX-itaiji-002 in SIP
—	ucs-itaiji-003	uXXXX(X)-itaiji-003 of GlyphWiki
—	ucs-bmp-itaiji-003	uXXXX-itaiji-003 in BMP
—	ucs-sip-itaiji-003	uXXXXX-itaiji-003 in SIP
—	ucs-itaiji-004	uXXXX(X)-itaiji-004 of GlyphWiki
—	ucs-bmp-itaiji-004	uXXXX-itaiji-004 in BMP
—	ucs-sip-itaiji-004	uXXXXX-itaiji-004 in SIP
—	ucs-itaiji-005	uXXXX(X)-itaiji-005 of GlyphWiki
—	ucs-bmp-itaiji-005	uXXXX-itaiji-005 in BMP
—	ucs-itaiji-006	uXXXX(X)-itaiji-006 of GlyphWiki
—	ucs-bmp-itaiji-006	uXXXX-itaiji-006 in BMP
—	ucs-itaiji-007	uXXXX(X)-itaiji-007 of GlyphWiki
—	ucs-bmp-itaiji-007	uXXXX-itaiji-007 in BMP
—	ucs-sip-itaiji-007	uXXXXX-itaiji-007 in SIP
—	ucs-itaiji-008	uXXXX(X)-itaiji-008 of GlyphWiki
—	ucs-bmp-itaiji-008	uXXXX-itaiji-008 in BMP
—	ucs-itaiji-009	uXXXX(X)-itaiji-009 of GlyphWiki
—	ucs-bmp-itaiji-009	uXXXX-itaiji-009 in BMP
—	ucs-itaiji-010	uXXXX(X)-itaiji-010 of GlyphWiki
—	ucs-bmp-itaiji-010	uXXXX-itaiji-010 in BMP
—	ucs-itaiji-011	uXXXX(X)-itaiji-011 of GlyphWiki

表 2.4: 文字識別子一覧

—	ucs-bmp-itaiji-011	uXXXX-itaiji-011 in BMP
—	ucs-sip-itaiji-011	uXXXXX-itaiji-011 in SIP
—	ucs-itaiji-084	uXXXX(X)-itaiji-084 of GlyphWiki
—	ucs-bmp-itaiji-084	uXXXX-itaiji-084 in BMP
—	ucs-radicals	CJK Radicals of UCS
—	ucs-radicals@unicode	CJK Radicals of UCS (Unicode)
—	ucs-hangul	Hangul Syllables of UCS
—	ucs-bmp-cjk	CJK Characters in BMP of UCS
—	ucs-bmp-cjk@gb	CJK Characters in BMP with GB rep. glyphs
—	ucs-bmp-cjk@JP	CJK Characters in BMP with JIS rep. glyphs
—	ucs-bmp-cjk@JP/hanazono	CJK Characters in BMP with Hanazono font
—	ucs-bmp-cjk-compat	CJK Compatibility Ideographs in BMP of UCS
—	ucs-bmp-cjk-compat@unicode	CJK Compatibility Ideographs of Unicode representatives
—	ucs-sip-ext-b	CJK Ideographs Extension B
—	ucs-sip@iso	UCS glyphs of UCS SIP
—	ucs-sip-ext-b@iso	CJK Ideographs Extension B (ISO/IEC 10646-2)
—	ucs-sip@JP/hanazono	Hanazono glyphs of UCS SIP

また、歴史的事情から一部の ID 素性名は接頭辞が付かない（Mule のコンベンションに従って、スクリプトまたは言語を示す接頭辞が付いているものが多いが、これらは ID 素性を示す接頭辞ではなく、文字符号識別子の一部である）。表 2.5 にこの一覧を挙げる。

2.5.1 ID 素性の RDF での表現

ID 素性は

1. 文字符号の符号位置を示す
2. 包摂粒度を示す（後述）
3. 複数の ID 素性対を持つ文字オブジェクトにおいて、それらの素性対の等価性（指し示すものが同じであること）を示す

という 3 種類の機能を持っているといえる。

最初の機能は文字符号 (CCS) の名前を *name* とし、符号位置を *cpos* とする時、`http://rdf.chise.org/data/ccs/`*name*`/code-point/`*cpos* を IRI とするエンティティーを主語とし、符号位置を示す述語と整数値とのトリプルによって表現できる。ここで `http://rdf.chise.org/data/ccs/`*name*`/code-point/`*cpos* は *name* という名前の CCS の *cpos* という符号位置を示す IRI であり、こうした IRI を『符号位置 IRI』と呼ぶことにする。

２番目の機能は包摂粒度を示す述語を使って文字オブジェクトに対応する IRI （これを『文字 IRI』と呼ぶことにする）と符号位置 IRI を関係づけることによって表現できる。文字 IRI としては、当面、CHISE-wiki [5] (EgT [6]) で用いている `http://www.chise.org/est/view/character/`*prefix*`.`*name*`=`*cpos* というものを用いることにする。ここで *prefix* は包摂粒度を示す接頭辞である。この接頭辞を表 2.6 の「IRI」列に示す。また、包摂粒度を示す述語をこ

ISO-IR	識別子	説明
6	ascii	ASCII (ISO646 IRV)
13	katakana-jisx0201	JIS X0201:1976 Japanese Kana
14	latin-jisx0201	JIS X0201:1976 Japanese Roman
77	control-1	Control characters 128-191
100	latin-iso8859-1	ISO8859-1 (Latin-1)
101	latin-iso8859-2	ISO8859-2 (Latin-2)
109	latin-iso8859-3	ISO8859-3 (Latin-3)
110	latin-iso8859-4	ISO8859-4 (Latin-4)
126	greek-iso8859-7	ISO8859-7 (Greek)
127	arabic-iso8859-6	ISO8859-6 (Arabic)
138	hebrew-iso8859-8	ISO8859-8 (Hebrew)
144	cyrillic-iso8859-5	ISO8859-5 (Cyrillic)
148	latin-iso8859-9	ISO8859-9 (Latin-5)
166	thai-tis620	TIS620.2529 (Thai)
177	ucs	ISO/IEC 10646
180	latin-tcvn5712	Vietnamese TCVN 5712:1983 (VSCII-2)
—	system-char-id	System char-id
—	chinese-big5-1	Big5 Level-1 [Mule]
—	chinese-big5-2	Big5 Level-2 [Mule]
—	latin-viscii	VISCII 1.1 (Vietnamese)
—	latin-viscii-lower	VISCII lower (Vietnamese) [Mule]
—	latin-viscii-upper	VISCII upper (Vietnamese) [Mule]
—	ethiopic-ucs	Ethiopic of UCS [Mule]
—	ucs-smp	ISO/IEC 10646 Plane 1 (SMP)
—	ucs-sip	ISO/IEC 10646 Group 0 Plane 2 (SIP)
—	sisheng	PinYin-ZhuYin [Mule]
—	lao	Lao script [Mule]
—	ipa	International Phonetic Alphabet [Mule]
—	ethiopic	Ethiopic [Mule]
—	chinese-big5-eten-a	Big5 ETEN (0xF9D6 .. 0xF9FE) [Mule]
—	chinese-big5-eten-b	Big5 ETEN (0xC6A1 .. 0xC8FE) [Mule]
—	arabic-digit	Arabic digits [Mule]
—	arabic-1-column	Arabic 1-column [Mule]
—	arabic-2-column	Arabic 2-column [Mule]
—	thai-xtis	Precomposed Thai (XTIS by Virach)
—	latin-iso8859-16	ISO8859-16 (Latin 10)
—	latin-iso8859-14	ISO8859-14 (Latin 8)
—	latin-iso8859-15	ISO8859-15 (Latin 9)

表 2.5　接頭辞の付かない ID 素性名一覧

の表の「RDF 述語」列に示す。[*3]

包摂粒度名	S 式	IRI	RDF 述語
超抽象文字	==>	a2	:super-abstract-character-of
抽象文字	=>	a	:abstract-character-of
統合字体	=+>	o	:unified-glyph-of
抽象字体	=	rep	:abstract-glyph-of
詳細字体	=>>	g	:detailed-glyph-of
抽象字形	==	g2	:abstract-glyph-form-of
字形	===	repi	:glyph-image-of

表 2.6 包摂粒度の表現

3 番目の機能は異なる CCS の符号位置 IRI に対応した文字 IRI が存在する時にその等価性を `owl:sameAs` によって示すことで実現できる。但し、当面、文字定義の Turtle 文書では `owl:sameAs` と等価な述語`:eq` を用いることにする。

例えば、

```
((=ucs@jis . #x6F22)
 (=adobe-japan1-0 . 01533)
 (=jis-x0208 . #x3441)
 (=jis-x0213-1 . #x3441)
 (=gt . 22918)
 (=gt-pj-1 . #x3441)
 (=daikanwa/+p . 18068))
```

という ID 素性対の集合は図 2.5 のように表現することができる（但し、このグラフでは符号位置 IRI と符号位置の関係は省略している)。

このように、CHISE 文字オントロジーにおける ID 素性はこの 3 種類の記述に分解することで RDF 化することができる。このため、ある文字オブジェクトが n 個の ID 素性対を持つ場合、通常、$3n-1$ トリプルの記述が必要となる。

2.5.2 decode

CHISE は ID 素性とそれが表す文字符号の符号位置から文字オブジェクトを得る API を提供している。

XEmacs CHISE の Emacs Lisp API では関数 `decode-char` がこれに相当する。

関数 **decode-char** (*ccs code*)

ID 素性名 *ccs* に対応する文字符号の符号位置 *code* に対応する文字オブジェクトを返す。存在しない場合は nil を返す。

[例]

```
(decode-char '=ucs #x4E00)
```

[*3] 但し、空接頭辞「:」は `http://rdf.chise.org/rdf/property/character/main/` の省略記法とする。

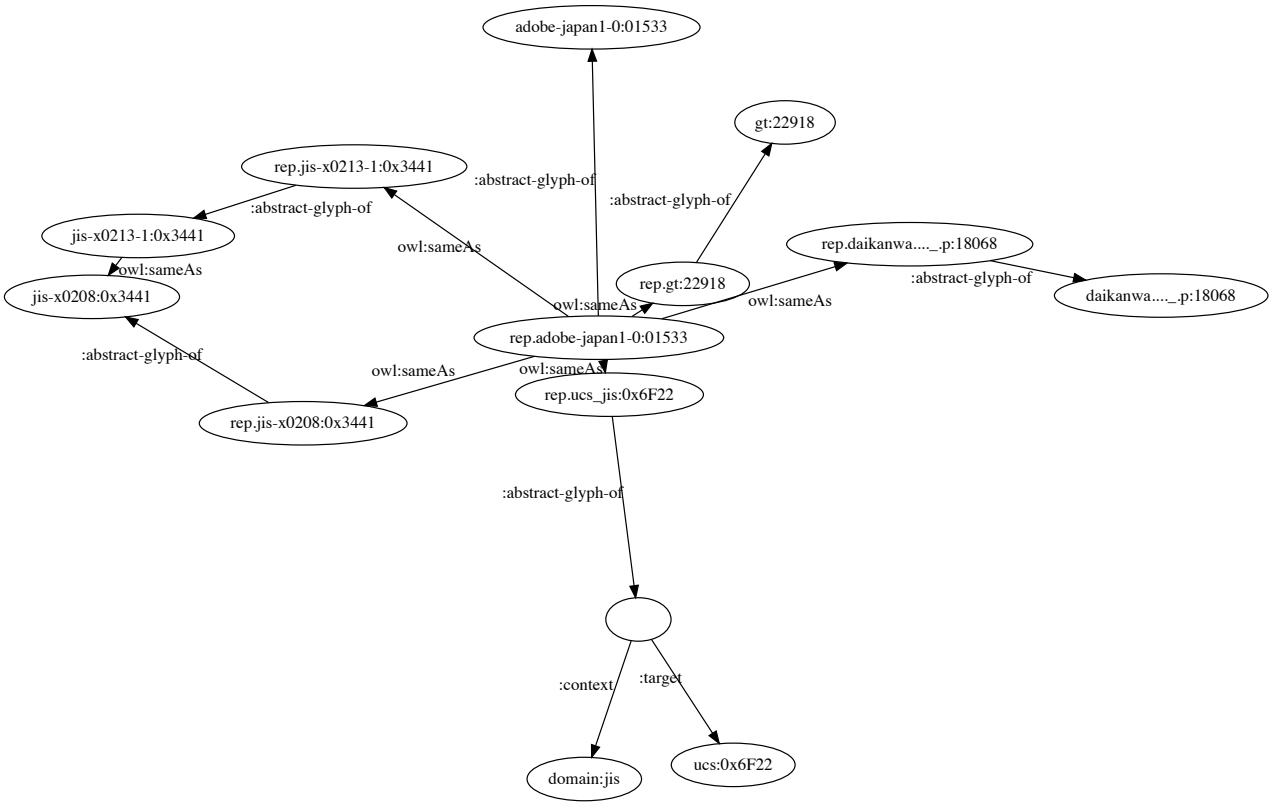

図 2.5　ID 素性の集合の RDF での表現の例（「漢」）

```
→ ?一

(decode-char '=daikanwa/+p 18068)
→ ?漢

(decode-char '=daikanwa 60000)
→ nil
```

CHISE-wiki (EgT) では `http://www.chise.org/est/view/rep.`*CCS*`=`*code* または `http://www.chise.org/est/view/rep.`*CCS*`:`*code* へのアクセスが `decode-char` に相当する。これは指定した文字オブジェクトの情報を表示する Web API で、現在の所 HTML のみをサポートしているが、将来的には、HTTP で `Accept: text/turtle` を指定した場合に Turtle 形式での出力をサポートする予定である。

[例]

```
http://www.chise.org/est/view/character/rep.ruimoku-v6:0xE033
```

2.5.3　encode

文字オブジェクトがある文字符号で符号化できるかを調べるには、その文字符号に対応する ID 素性の素性値を調べれば良いといえる。よって、XEmacs CHISE では関数 get-char-attribute か関数 char-feature を使えば良いといえ

るが、ID 素性では素性の継承機能（例えば、=jis-x0208 から=jis-x0208@1990 を派生させる）やエイリアスが存在し、これらをサポートするために専用の関数を設けている。

関数 **encode-char** (*character ccs* ***&optional*** *defined-only*)

文字オブジェクト *character* が文字符号 *ccs* の符号位置を持つ場合、その値を返す。存在しない場合は nil を返す。

defined-only が non-nil の場合、ID 素性が陽に定義されている場合のみ値を返す。

CHISE-wiki (EgT) では 2.2.2 節で述べた素性値の参照機能が利用可能である。

第 3 章

階層的な包摂関係の導入

3.1 フラットなモデルから階層的なモデルへ

2 章で述べた UTF-2000 時代の文字表現モデルでは、文字をオブジェクト指向に基づいて抽象化しその内部表現を隠蔽するとともに、Chaon モデルに基づくインターフェースを提供することで、特定の文字符号での定義に囚われない文字操作を可能とした。

この枠組は、文字も抽象グリフも字形も全て文字オブジェクトという意味では対等であるというフラットなモデルととらえることができる。

文字間の関係は 2.3 節で述べた関係素性を用いて記述することができる。また、UCS での対応する符号位置や対応する大漢和番号といった情報を記述することにより、UCS や大漢和辞典での包摂範囲的な情報を扱うこともできる。例えば、XEmacs UTF-2000 では素性 `=>ucs` [*1]を用いて対応する UCS の符号位置を示すようにしていた。[*2]また、XEmacs UTF-2000 では UCS の異体字オブジェクトに対して素性 `=>ucs` で対応する UCS の符号位置を示すと、その UCS の符号位置に対応する文字オブジェクト（ID 素性 `=ucs` で decode した結果得られる文字オブジェクト）の関係素性 `->ucs-unified` にそれらの異体字オブジェクトが自動的にリストアップされるようにすることで UCS の包摂域関する情報も扱うことができた。[*3]

このようにフラットな文字モデルであっても、文字オブジェクト間の異体字関係を記述することによって包摂に関するような情報を記述することができる。しかしながら、文字に関する記述の中には字種共通のもの、字体に関するものや字体間の関係、字形に関するもの、字種と字体・字体と字形の包摂関係等があり、これら全てをバラバラに記述するのは繁雑である。特に、異体字関係の情報は字体間の関係として記述されることが多く、その字体に属する全ての字形オブジェクトに対して関係素性を網羅するのは容易ではなく、また、結果として書かれる情報が判りにくくなる。

こうしたことを鑑みて、多くのオブジェクト指向言語で用いられている継承の仕組みを用いて文字定義の差分的記述の仕組みを導入した。本章ではこの仕組みについて概説する。

3.2 包摂関係素性

CHISE 文字オントロジーはさまざまな文字観念を差分的に記述するためのベースになるような汎用的な文字データベースを提供することを目指している。このため、字形・字体の細かな差異を捨象した抽象的文字観念、Unicode や JIS X0208 のような各種符号化文字集合における符号化文字、さまざまな文字符号の規格や辞書などの文字表における例示字体・字形の情報など、抽象・具象のさまざまなレベルの代表的文字概念を収録している。

[*1] 他粒度包摂モデルに基づく現在のコンベンションでは、=> は抽象文字粒度を示す接頭辞であるが、ここでは意味が異なっていることに注意。

[*2] 現在もこの古い形式の文字定義が残っている。

[*3] 現在の XEmacs CHISE でもこの仕組みは残っている。

これらの各レベルは主に字体差（比較的大きな形の差異）を示す->denotational, <-denotational 素性と比較的小さな字体差・字形差（比較的小さな形の差異）を示す->subsumptive, <-subsumptive 素性を用いて、文字オブジェクト間の継承関係として記述している。即ち、

抽象的 ->denotational 具象的

や

抽象的 ->subsumptive 具象的

という風に文字間の継承関係を記述する訳である。->denotational と ->subsumptive は混在して使うことができ、

大粒度抽象文字（字種） ->denotational 中粒度抽象文字 ->denotational 細粒度抽象文字
->subsumptive 字体 ->subsumptive 抽象字形

などのように多段的継承関係を記述することも可能である。

3.3　文字定義の継承をサポートした関数

関数 **char-feature** (*character feature* ***&optional*** *default-value feature-rel-max char-rel-max*)

文字オブジェクト *character* の素性 *feature* の値を返す。
もし、値が定義されていない場合、*default-value* を返す。なお、*default-value* の既定値は nil である。
この関数は 2.2.2 節で述べた関数 get-char-attribute と同様であるが、後述する文字定義の継承が行われている場合に、get-char-attribute では親の素性を無視するのに対し、char-feature は親の素性を参照するという違いがある。

[例]

```
(get-char-attribute ?教 '=>iwds-1)
→ nil

(char-feature ?教 '=>iwds-1)
→ 11
```

3.4　フラット形式から階層形式への変換

文字定義の集合演算を行うことで、UTF-2000 時代のフラットに記述された異体字の集合を包摂関係素性を用いた階層化されたオブジェクト間の差分記述に変換することができる。

ある UCS の符号位置 c を代表する文字オブジェクト R と c を値とする素性 =>ucs を持つ異体字オブジェクト $V_1, V_2, ...$ がある時、R および $V_1, V_2, ...$ の持つ素性対の集合の共通部分をとることでこれらを包摂する抽象文字オブジェクト C の文字定義を求めることができる。そして、それ以外の部分を R および $V_1, V_2, ...$ に対応するインスタンス $I_0, I_1, I_2, ...$ の文字定義とし、C と I_n を包摂関係素性でつなぐと階層化されたオブジェクト間の差分記述に変換できる。

XEmacs CHISE のユーティリティーパッケージである tomoyo-tools [*4] に含まれる conv-util.el の関数 conv-u-convert-char はこの仕組みに基づき指定した UCS 文字とその異体字の集合を `->subsumptive` 素性を用いた階層形式に変換する。

関数 **conv-u-convert-char** (*character* ***&optional*** *variants*)

指定した文字 *character* と異体字のリスト *variants* を`->subsumptive` 素性を用いた階層形式に変換する。
省略可能な引数 ***&optional*** variants を省略した場合、文字 *character* の関係素性 `->ucs-unified` の値が用いられる。この場合、*character* は ID 素性 `=ucs` を持っていなければならない。

[*4] `http://git.chise.org/gitweb/?p=chise/tomoyo-tools.git;a=tree`

第 4 章

漢字構造記述

多くの漢字は偏と旁などの部品の組み合わせによって構成されている。こうした漢字の部品の組合せ構造は形の抽象的表現となるだけでなく、字義や音価にも関係しており、字源に基づく文字構造の分析は「解字」と呼ばれ、そうしたデータは重要な辞書記述の 1 つである。そこで我々は 2001 年度に未踏ソフトウェア創造事業の助成 [8] を受けて、Unicode の基本漢字、拡張漢字 A, B の約 7 万字の漢字を対象に IDS (Ideographic Description Sequence) 形式 [1] に基づく漢字構造情報のデータベース化を行った。我々はその後もこのデータベースの開発・改良を続けており、関連ツールとともに「CHISE 漢字構造情報データベース」として公開している。ここでは、このデータベースをはじめとする CHISE 環境における漢字構造情報の構成法と利用について、一般的な問題を交えて述べてみたい。

4.1 漢字構造記述とは

4.1.1 用語

ここでは漢字の部品の組合せ構造に関する情報のことを「漢字構造情報」と呼び、その情報の記述を「漢字構造記述」と呼ぶ。また、これ以上分解できない漢字のことを「基本部品」と呼び、会意字や形声字など複数の基本部品を組み合わせてできる漢字のことを「複合漢字」と呼ぶことにする。また、複合漢字を部品として複合漢字を作ることがあるので、部品として使われ得る「基本部品」および「複合漢字」を「部品」と呼ぶ。また、部品と部品の結合の仕方の情報を表すもののことを「オペレータ」(operator) と呼ぶ。

4.1.2 『形』の抽象的表現としての漢字構造記述

漢字構造記述はまず第一義的には形の抽象的表現といえる。例えば、「村」という字は「木」という部品と「寸」という部品を横に並べた複合漢字であるということはすぐに判る。そして、「木」や「寸」といった部品に対応するグリフを用意しておいて、複合漢字のグリフを合成することが可能であり、実際に和田研フォントシステム [9] や KAGE システム [4] ではこの手法が実現されている。

4.1.3 漢字構造記述と解字

漢字の古典的字書に「説文解字」があるが、この書は単に字書であるだけでなく、漢字の造字法に関するモデルを提示したものでもある。そもそも「説文解字」という題は基本部品である『文』を説明し、複数の部品を組み合わせた複合漢字である『字』の分解法を示すということであり、ここで扱う漢字構造情報データベースのはしりということができる。

ところで、漢字は『形』『音』『義』の 3 要素からなるものといわれるが、「説文解字」的世界は単に『形』を説明した

ものではなく、この 3 者の対応関係を説明したものといえる。すなわち、ある複合漢字を構成する部品について、『音』を担うものがどれでどういう音を担い、意味カテゴリーを担うものがどれでどういう意味カテゴリーを担うかという情報を掲載している。

また、隷変後の現代字の場合、字形が大きく変化しており、本来別の字義もしくは字音を担う別の部品が同じないしは非常に似た部品グリフで表現されるようになったものが多数ある。また、逆に、本来別の字義あるいは字音を担う同じ部品が複数の異なる部品グリフで表現される場合も多数ある。

さらに、「旗」という字のように、視覚的には「方」と『「𠂉」の下に「其」』のように左右に分解するのが自然だが、意味的には「从（「方」の右に「𠂉」）」と「其」に分解するのが妥当な場合がある。

ここで、前者のように分解したものを「視覚的漢字構造記述」、後者のように分解したものを「字源的漢字構造記述」と呼ぶことにする。

4.2　漢字構造情報の記述形式

4.2.1　IDS 形式

漢字構造情報に基づく符号化手法の試みは 1970 年代に遡るが、漢字符号化の主流とはならず、標準的な記法も確立されて来なかった。その後、ISO/IEC 10646-1:2000 [1] において漢字構造記述の標準記法である IDS (Ideographic Description Sequence) とそのための前置演算子である IDC (Ideographic Description Characters) （図 4.1）が定義された。

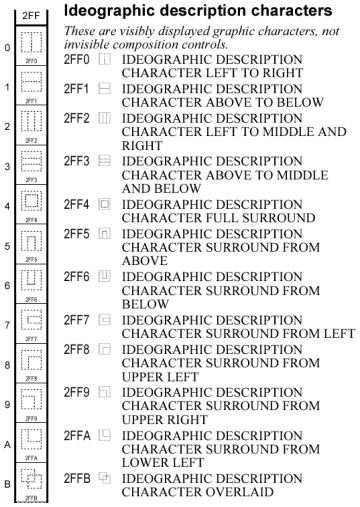

図 4.1　Ideographic Description Characters

IDS は Lisp 言語における S 式と同様な前置記法の一種で、前置演算子である IDC の後に 2 個ないしは 3 個の部品 (Description Components (DC)) をとる。例えば、2 個の部品を左右に並べる前置演算子「⿰」を使って「村」の漢字構造を「⿰木寸」と書くことができる。[*1]

IDS では部品として、

- UCS で符号化された漢字
- UCS で符号化された部首・部品用文字

[*1] IDC の種類によって後に来る部品の数は決まっているので、S 式と異なり、括弧は不要である。

- U+FF1F（全角の「？」）
- 外字
- IDS

を用いることができ、部品として IDS を用いることによって、入れ子状の記述が可能である。例えば、「渓」の漢字構造を「⿰氵⿸又大」と書くことができる。

4.2.2 S 式による表現

IDS は漢字構造情報を文字列として扱ったり、文字列としてパターンマッチングしたりするのには便利であるが、部品として IDS が現れる場合の処理はそのままでは少し面倒である。よって、より込み入った処理を考えれば、IDS を構文解析した結果の構文木として扱う方が便利であるといえる。そこで、CHISE 文字データベースの内部表現としては Lisp 言語における S 式を用いている。

IDS は Lisp 言語における S 式と同様な前置記法なので、その構文解析は極めて容易であり、オペレータの前とオペレータがとる最後の部品の後に括弧を付ければ S 式となる。

CHISE 文字データベースでは Chaon モデルに基づき文字素性の集合として文字を表現しているが、漢字構造情報はこの文字素性の 1 つと看做せ、'ideographic-structure' という素性名が使われている。

例えば、「峠」の ideographic-structure の値は

```
(((name . "IDEOGRAPHIC DESCRIPTION CHARACTER LEFT TO RIGHT")
  (=ucs            . #x2FF0))
 ?山
 ((ideographic-radical . 25)
  (ideographic-strokes . 4)
  (total-strokes    . 6)
  (ideographic-structure
   ((name . "IDEOGRAPHIC DESCRIPTION CHARACTER ABOVE TO BELOW")
    (=ucs          . #x2FF1)
    )
   ?上
   ?下)
  (=ucs            . #x209D7)
  (=cns11643-6     . #x2376)
  (=daikanwa       . 02784)
  ))
```

のように表すことができる。ここで、((素性名 . 値) ...) は 2.1.1 節で述べた「文字指定 (char-spec)」と呼ばれる形式で、Chaon モデル的に文字を素性の集合で表現したものであり、概念的には 1 文字を表す。データベース中では 1 つの文字オブジェクトに正規化されている。即ち、(...(ideographic-structure ... ?上 ?下) ...) の部分は文字オブジェクト ?⿰ に正規化されることになる。そして、?⿰ の ideographic-structure 素性を見れば (?⿱ ?上 ?下) という値を得ることができる。

CHISE 文字データベースでは「文字参照 (char-ref)」と呼ばれる形式がある。これは (:属性名 値 ... :char 文字オブジェクト) という形の Lisp でいう所の属性リスト (property list) の一種で、:char 属性で指定した文字に他の属性で指定したメタデータを付加することができる。ideographic-structure 素性では文字オブジェクトの代わりにこの形式のデータを置くことができ、これによって部品やオペレータに出典情報や位置情報などの付加データを付随させることが可能である。但し、文字参照形式で記述した付加データは IDS（や拡張 IDS）では表現することはできないという

問題があるので注意が必要である。[*2]

4.3　漢字構造情報の変換

4.3.1　IDS の構文解析

CHISE 漢字構造情報データベースのパッケージ (以下、CHISE-IDS と略す)[*3]には IDS を構文解析して ideographic-structure 形式に変換する XEmacs CHISE 用の Emacs Lisp プログラム ids.el および ids-read.el が含まれている。

ids.el の関数 ids-parse-string は第 1 引数で与えられた IDS 文字列を構文解析して ideographic-structure 形式の値を含む文字指定形式の値を返す関数である。入れ子状の IDS 文字列を指定した場合、デフォルトではそのままの構造で返すが、第 2 引数に nil でない値を指定した場合、なるべく単純化した構造を返す。例えば、

```
(ids-parse-string "⿰山⿱上下")
```

を評価すると

```
((ideographic-structure ?⿰ ?山 ((ideographic-structure ?⿱ ?上 ?下))))
```

が返るが、

```
(ids-parse-string "⿰山⿱上下" t)
```

を評価すると

```
((ideographic-structure ?⿰ ?山 ?卡))
```

が返る。

また、IDS として構文解析できない文字列を指定した場合、nil が返る。[*4]

ids-read.el は CHISE-IDS に収録された IDS 形式のデータファイル (IDS-*.txt) を読み込むためのプログラムで、XEmacs CHISE の buffer から取り込むためのコマンド ids-read-buffer とファイルから読み込むためのコマンド ids-read-file が用意されている。なお、CHISE-IDS は make install すれば、これらのプログラムを使って IDS-*.txt を CHISE 文字データベースに統合するようになっている。

4.3.2　IDS への変換

XEmacs CHISE には ideographic-structure 形式の漢字構造情報を IDS 形式に変換する関数 ideographic-structure-to-ids があり、これを使って CHISE 文字データベース中の ideographic-structure 素性の情報を IDS 形式で出力することが可能である。

また、CHISE-IDS 附属の Emacs Lisp プログラム ids-dump.el には CHISE-IDS に収録された IDS 形式のデータファイル (IDS-*.txt) として出力するためのユーティリティ・コマンドが用意されている。

[*2] 付加データを落した IDS にすることは可能である。

[*3] `http://git.chise.org/gitweb/?p=chise/ids.git;a=tree`

[*4] 余分な文字列がついている場合には関数 ids-parse-element を使うことができる。

4.4 検索

4.4.1 Emacs Lisp による実装

CHISE-IDS には部品の集合を使って漢字を検索するためのプログラム ids-find.el が含まれている。これは XEmacs CHISE で利用可能な Emacs Lisp プログラムで、現在の所、ids-find-chars-including-components（別名 ideographic-structure-search-chars）と ids-find-chars-covered-by-components という 2 つの検索用コマンドを提供している。

前者は M-x ideographic-structure-search-chars [CR] 部品列 [CR] で指定された部品列の各部品を少なくとも 1 個は含んでいる漢字の一覧を表示するコマンドである（図 4.2）。

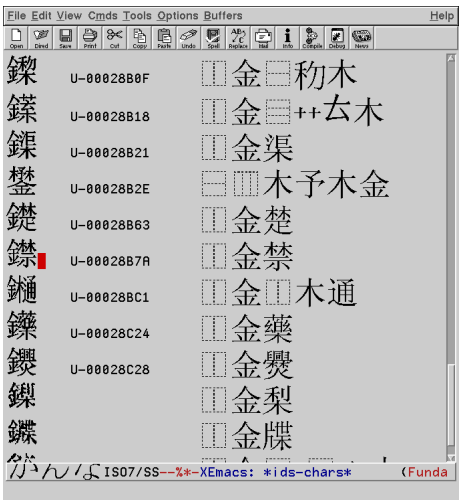

図 4.2 M-x ideographic-structure-search-chars [CR] 木金 [CR] の出力結果

後者は M-x ids-find-chars-covered-by-components [CR] 部品列 [CR] で指定された部品列中の任意の部品を 0 回以上用いて全体として 2 個以上の部品から構成される漢字の一覧を表示するコマンドである（図 4.4）。

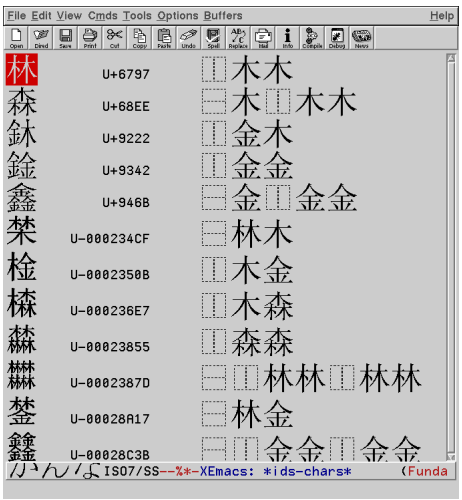

図 4.3 M-x ids-find-chars-covered-by-components [CR] 木金 [CR] の出力結果

4.4.2　WWW サービスとしての実現

ids-find.el と同様の機能を WWW 上で提供するために、2005 年 4 月から「CHISE IDS 漢字検索」という WWW サービスの開発を始め、同月からサービスの提供を始めている。

これは現在 ideographic-structure-search-chars と同様の機能を提供しており、「部品文字列」窓に 1 つまたは複数の部品を指定し、漢字を検索することができる。検索を実行すると、指定した部品列の各部品を少なくとも 1 個は含んでいる漢字の一覧が表示される。ある文字を部品とする漢字が存在する場合、その文字の下にインデントして木構造状に表示される。

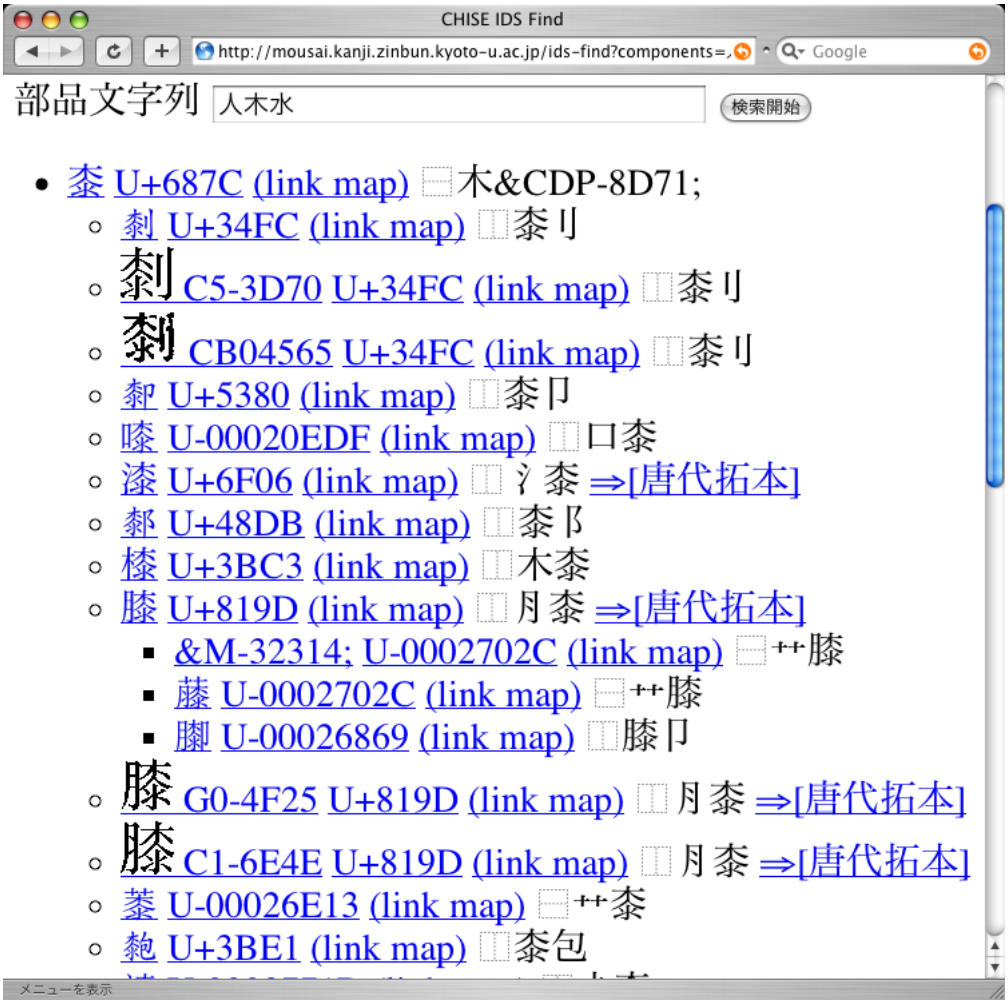

図 4.4　CHISE IDS 漢字検索

4.5　RDF による表現

CHISE 文字オントロジーでは漢字構造情報は IDS 形式をパースした結果の構文木を S 式にしたものとして表現しており、その文字素性として ideographic-structure を用いている。RDF ではこのデータ形式をそのまま表現するのではなく、[7] で述べた『IDC の述語化』や IDS 用コンテナを使ったモデルを用いることにした。但し、記述の簡潔さや検索時の利便性を考慮して、以下に述べるような修正を加えている。

まず、漢字構造記述のための名前空間 IRI として `http://rdf.chise.org/rdf/property/character/isd/` を用い、その接頭辞として `isd:` を用いる。また、IDC のための型を表現するための名前空間 IRI として `http://rdf.chise.org/rdf/type/character/idc/` を用い、その接頭辞として `idc:` を用いる。そして、IDC のための型は idc: の後に IDC 文字を付けたものとする。例えば、「⿰」の型は idc:⿰ となる。

漢字構造情報を表現するための述語として `isd:structure` を用い、その値として型が前述の IDC 型の空白ノードを取ることにする。ここで、この IDC 型の空白ノードを『漢字構造コンテナ』と呼ぶことにする。このコンテナを主語として、IDC の種類に従い、その引数にとる部品を表 4.1 に示す述語の目的語とする。

IDC	述語 1	述語 2	述語 3
⿰	isd:left	isd:right	————
⿱	isd:above	isd:below	————
⿲	isd:left	isd:middle	isd:right
⿳	isd:above	isd:middle	isd:below
⿴	isd:surround	isd:filling	————
⿵	isd:surround	isd:filling	————
⿶	isd:surround	isd:filling	————
⿷	isd:surround	isd:filling	————
⿸	isd:surround	isd:filling	————
⿹	isd:surround	isd:filling	————
⿺	isd:surround	isd:filling	————
⿻	isd:underlying	isd:overlaying	————

表 4.1 IDC 述語一覧

例えば、「漢」の漢字構造記述「⿰氵 莫」は図 4.5 のように表現することができる。

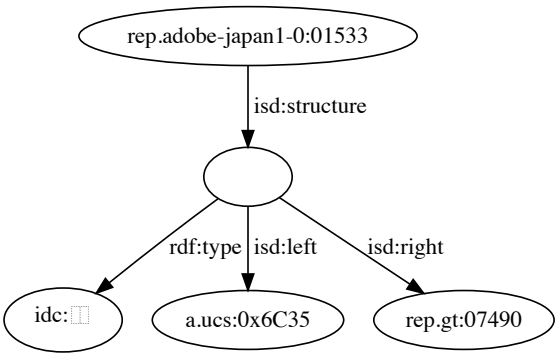

図 4.5 漢字構造記述の例（「漢」）

第 5 章

漢字構造記述と包摂粒度

5.1　多粒度漢字構造モデル

漢字構造情報は部品の組合せ方を示すオペレーターと部品からなる構文木で表現できる。IDS はオペレーターとして IDC (Ideographic Description Characters), 部品として UCS の統合漢字および部品用文字を用いたものであるが、部品としてそれ以外のものを用いることも原理的には可能である。

ここで、部品として複数の異なる包摂粒度を持つものを用いれば、複数の部品の組合せで構成される漢字の各部品の包摂範囲を示すことで、その漢字の包摂範囲を示すことができるといえる。これを『多粒度漢字構造モデル』と呼ぶ (図 5.1)。[3]

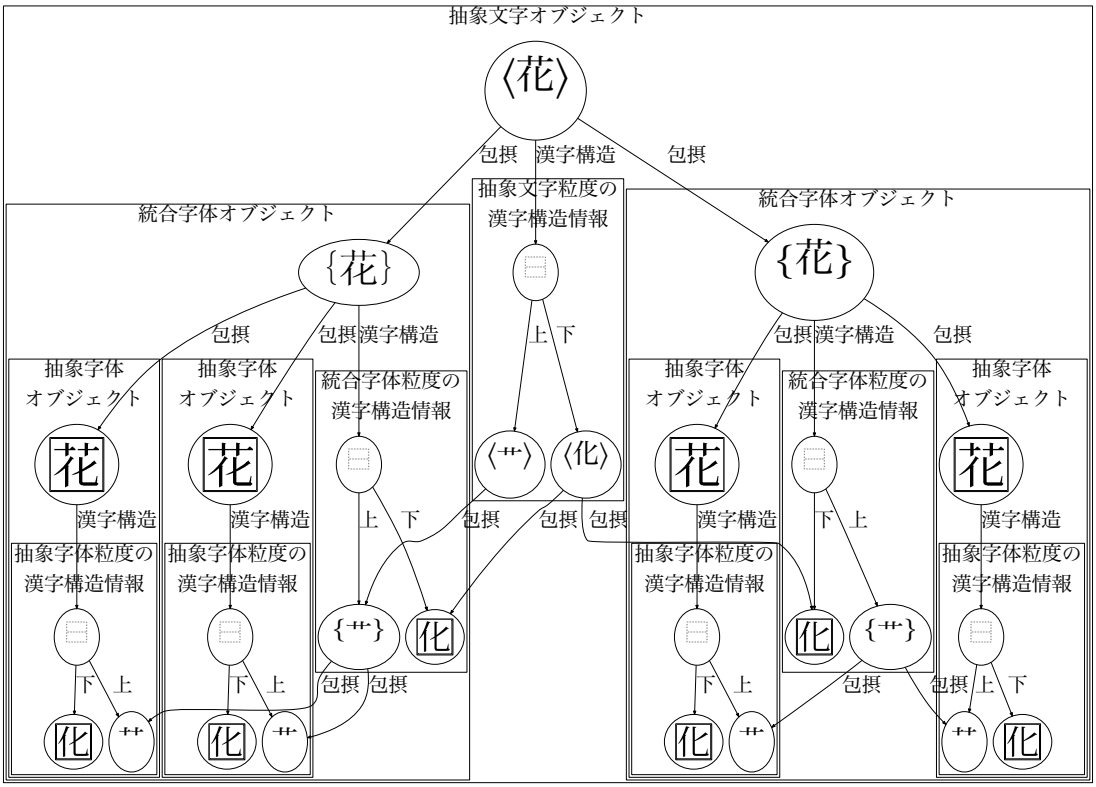

図 5.1　多粒度漢字構造モデルの概念図（花）

5.2　漢字の包摂範囲をどう表現するか

漢字の包摂という問題をとらえる場合、3 章で述べたような文字間の包摂関係という視点が思い浮かぶが、それだけでは個々の文字（字種や字体）における分類はできても全ての文字を網羅するような一般的な記述にはならず、また、文字毎に分類規準が不統一になりがちである。

一方、文字符号の世界では部品に着目したアプローチがとられてきた。多くの漢字は複数の部品の組合せからなっているが、漢字を部品の組合せとしてとらえた時に似た形の部品を同一視するためのルールを決めれば、比較的少数のルールの組合せによって多数の漢字を対象とした符号化文字の包摂範囲の定義が可能である。ここで、このルールのことを『包摂規準』と呼ぶ。[*1]

包摂規準の集合は何らかの包摂ポリシーを記述したものといえる。もし包摂規準の集合が何らかの形式的体系（即ち、数理論理学における論理体系）になっていたとすると、包摂規準の集合はその『包摂論理』とでもいうものの公理系と見ることができる。そして、抽象文字と字体という文字単位の包摂関係はこの包摂論理におけるモデルに相当するものといえる。よって、もしこの包摂論理が完全であれば、対象となる文字の漢字構造記述とこの論理体系の公理系たる包摂規準の集合によって演繹すれば包摂関係にあるものは全てこの論理で証明可能なはずである。また、この包摂論理が健全であれば、包摂関係にあると証明されたものは全て本当に包摂関係にあるはずである。

包摂規準を部品間の同値性を示したものと看做し、包摂規準の集合を等式論理における等式の集合として表現すれば、実際に完全性・健全性を満たした『包摂論理』に相当するものが構成可能であり、項書換え系を用いることによりこの『包摂論理』を用いた推論（定理証明）が可能である。

しかしながら、包摂規準の適用除外や UCS におけるソースコードセパレーション、あるいは、漢字字体のバリエーションの生成パターンを観察すると、包摂規準の集合によって示されるような一様な包摂ポリシーが全ての漢字に適用可能であると考えるのは現実には難しいといえる。

そこで、ここでは等式論理で記述可能な包摂規準の集合によって示される包摂ポリシーが複数あるような枠組を考えることにする。ここで、このある包摂ポリシーのことを『(定義された）包摂粒度』と呼ぶことにする。この枠組では、個々の包摂粒度はその粒度を定義する包摂規準の集合を持つ。各包摂規準はその粒度の部品とそれより細かい粒度の部品との包摂関係によって記述される。この部品の包摂関係を重ねて行くことにより複数の包摂粒度間の関係を記述することができる。

多粒度漢字構造モデルは、この枠組に基づいて、文字間の包摂関係とその文字の漢字構造記述に含まれる部品間の包摂関係の相互対応関係を示すことによって、複数の包摂粒度間の関係を示すためのものである。例えば、図 5.1 において上下 2 階層に着目してみると、それぞれ文字間の包摂関係とその漢字構造中の部品の包摂関係が一致しており、また、文字の包摂粒度と漢字構造記述（中の部品）の包摂粒度が一致している。

5.3　包摂粒度の名前付けと分類

5.2 節で述べたような方法によって包摂粒度を形式的に扱うことができるが、この枠組だけでは個々の部品や文字における包摂粒度（包摂ポリシー）間の関係は判っても、個々の包摂粒度を指す名前がなく不便である。そこで、JIS の包摂規準や IWDS-1 (IRG Working Document Series (IWDS) 1: List of UCV (Unifiable Component Variations) of Ideograhs [2]) のような良く使われる包摂規準の集合には名前を付けるとともに、特に固有名を持たないものに対してもその包摂粒度の種類を分類する仕組みを設けた。

[*1] JIS X 0208/0213 では字形の細かなデザイン差を捨象した字体を対象にどう包摂するかを定めるようにしているので、このことを強調して『字体の包摂規準』と呼ぶ。

5.3.1 粒度の種類

漢字の包摂粒度を考える場合、常用漢字表で用いられている『字種』-『字体』-『字形』という概念と、UCS 等の符号化文字集合で用いられている『抽象文字』-『グリフ』-『グリフイメージ』という概念等があるが、これらをざっくりとまとめると、大まかに言って、『字種』-『抽象文字』-『字体』≒『グリフ』-『字形』≒『グリフイメージ』という４階層の包摂粒度に整理することができる。

しかしながら、個々の事例を考えた場合、必ずしもこのように綺麗に整理できない場合も少なくない。即ち、ある文字の字形の集合を視覚的形状に基づいて分類した時、その弁別上のポイントは必ずしもこの 4 階層にぴったり重なるとは言えず、字形デザイン差に相当する包摂レベルに複数の階層が生じたり、どこまでを字体レベルの差異と看做すか判断に迷うような結果になったりする。

とはいえ、だからといって、包摂粒度の階層を無数に増やすと記述の複雑さを不必要に増やしかねない上、モデルが複雑すぎれば利用者の直観に合わないケースが多発しかねず、個々のオブジェクトの参照可能性を損なう恐れがあるといえる。

よって、参照のために用意する基本となる包摂粒度（基本粒度）と、その上下の中間階層を表現するための補助的な包摂粒度（補助的粒度）に分けて記述することにした。

5.3.2 基本粒度

抽象文字粒度

UCS 等の抽象文字に相当する包摂粒度として『抽象文字粒度』を設ける。

CHISE の S 式では、*name* の抽象文字を示す ID 素性名を

`=>`*name*

で表現する。EgT[6] (CHISE-Wiki [5]) での URL 中では、

`a.`*name*

で表現する。

また、抽象文字粒度の文字を示す場合、〈字〉のように表現する。

この包摂粒度の記述は必須とする。但し、ID 素性の継承関係から推論可能な場合には省略を許す。

抽象字体粒度

字形デザイン差を捨象した字体に相当する包摂粒度として『抽象字体粒度』を設ける。

CHISE の S 式では、*name* の抽象字体を示す ID 素性名を

`=`*name*

で表現する。EgT (CHISE-Wiki) での URL 中では、

`rep.`*name*

で表現する。

また、抽象字体粒度の文字を示す場合、「字」のように表現する。

この包摂粒度の記述は必須とする。

抽象字形粒度

Adobe-Japan1, 汎用電子、文字基盤の IVS で指示されるような抽象的な字形に相当する包摂粒度として『抽象字形粒度』を設ける。

CHISE の S 式では、*name* の抽象字形を示す ID 素性名を

==*name*

で表現する。EgT (CHISE-Wiki) での URL 中では、

g2.*name*

で表現する。

また、抽象字体粒度の文字を示す場合、《字》 のように表現する。

この包摂粒度の記述は必須とする。但し、ID 素性の継承関係から推論可能な場合には省略を許す。

例示字形粒度

字形を示す包摂粒度として『例示字形粒度』を設ける。

CHISE の S 式では、*name* の例示字形を示す ID 素性名を

===*name*

で表現する。EgT (CHISE-Wiki) での URL 中では、

repi.*name*

で表現する。

また、例示字形粒度の文字を示す場合、『字』のように表現する。

この包摂粒度の記述はオプショナルである。

5.3.3　補助的粒度

統合字体粒度

抽象文字粒度と抽象字体粒度の間の包摂粒度として『統合字体粒度』を設ける。これは「艹」と「艹」のような一画の差異や「爫」と「爫」のような方向の差異といった微小な差異を統合したオブジェクトを表現するためのものである。

CHISE の S 式では、*name* の統合字体を示す ID 素性名を

=+>*name*

で表現する。EgT (CHISE-Wiki) での URL 中では、

o.*name*

で表現する。

また、統合字体粒度の文字を示す場合、〈+ 字 +〉 のように表現する。

詳細字体粒度

抽象字体粒度と抽象字形粒度の間の包摂粒度として『詳細字体粒度』を設ける。

CHISE の S 式では、*name* の詳細字体を示す ID 素性名を

=>>*name*

で表現する。EgT (CHISE-Wiki) での URL 中では、

g.*name*

で表現する。

また、詳細字体粒度の文字を示す場合、《+ 字 +》 のように表現する。

参考文献

[1] International Organization for Standardization (ISO). *Information technology — Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane (BMP)*, 2000 年 3 月. ISO/IEC 10646-1:2000.

[2] IRG Working Document Series. `http://appsrv.cse.cuhk.edu.hk/~irg/irgwds.html`.

[3] Tomohiko Morioka. Multiple-policy character annotation based on CHISE. *Journal of the Japanese Association for Digital Humanities*, Vol. 1, No. 1, pp. 86–106, 2015 年 11 月.

[4] 上地宏一. 漢字フォント自動生成サーバ “影 KAGE” の構築 — 文字コードの枠組みを越える次世代漢字処理の提案 —. 漢字文献情報処理研究, Vol. 3, pp. 143–147, 2002 年.

[5] 守岡知彦. CHISE のセマンティック Wiki 化の試み. 情処研報, Vol. 2010-CH-87, No. 8, pp. 1–8, 2010 年 7 月.

[6] 守岡知彦. Wiki 的手法に基づく構造化データの編集について. 人文科学とコンピュータシンポジウム論文集 —人文工学の可能性〜異分野融合による「実質化」の方法〜, 情報処理学会シンポジウムシリーズ, 第 2010 巻, pp. 33–40. 情報処理学会, 情報処理学会, 2010 年 12 月.

[7] 守岡知彦. 漢字構造情報の rdf 化の試み. 山崎直樹（編）, すべてをコンピュータの中に—繋がってしまったデータとその未来, pp. 3–22. 京都大学人文科学研究所共同研究プロジェクト: 情報処理技術は漢字文献からどのような情報を抽出できるか—人文情報学の基礎を築く, 全国共同利用・共同研究拠点「人文学諸領域の複合的共同研究国際拠点」, 2013 年 2 月.

[8] 守岡知彦, クリスティアン・ウィッテルン. 文字データベースに基づく文字オブジェクト技術の構築. 情報処理振興事業協会 平成 13 年度 成果報告集. 情報処理振興事業協会, 2002 年. `http://www.ipa.go.jp/NBP/13nendo/reports/explorat/charadb/charadb.pdf`.

[9] 田中哲朗, 岩崎英哉, 長橋賢児, 和田英一. 部品合成による漢字スケルトンフォントの作成. 情報処理学会論文誌, Vol. 36, No. 9, pp. 2122–2131, 1995 年.