CHISE の Web API 化の試み、ついでに、RDF 化四度目の正直?

守岡 知彦

1 はじめに

CHISE（CHaracter Information Service Environment; 文字情報サービス環境）project [1] は特定の汎用文字符号に制約されることなく、文字や文字の性質を自由に表現・処理するための環境を実現することを目指すプロジェクトであり、1998 年にスタートした「UTF-2000 プロジェクト」を改組する形で 2001 年に始まった。CHISE project では文字に関するさまざまな知識を機械可読な形式で記述したデータベースを中心に、そのクライアントとして文字処理系を実現することによって、新たな（符号化）文字を定義可能で、かつ、それ以前の外字とは異なり、定義した文字の性質に関する情報を共有することによって定義した文字のやりとりを可能とすることを目指した。

この目標は CHISE 文字オントロジーと XEmacs CHISE という文字処理系として実現することができたが、これだけではインターネット上での文字知識の共有という観点では問題があった。このため、CHISE 文字オントロジーの情報や XEmacs CHISE の機能をインターネット上で利用・共有することを目指し、その Web クライアントの開発が始まった。その第一歩となったのが CHISE 漢字構造情報データベースの Web 上における検索サービスである「CHISE IDS 漢字検索」[16] である。これは 2005 年 5 月に京都で開催された ISO/IEC JTC1 SC2 IRG の場で CHISE 漢字構造情報データベースのデモンストレーションを行うことを目的に著者が開発したものである [8] が、各検索結果に対して「char-desc」と呼ばれる XEmacs CHISE の what-char-definition コマンドと同様な文字の詳細情報を表示する頁がリンクされていた。後に、この「char-desc」は CHISE-Wiki [5] というシステムに置き換えられた。これは char-desc を一般化し Web 上での編集機能を実現することを目指したシステムで、データを編集できるだけでなく、その見掛けを編集可能にするための仕組みを実現したものである。ただ、認証の仕組みの問題から実際にその編集機能を一般公開するに至らないまま今日に至っている。CHISE-Wiki は、当初、文字だけを対象にしたものだったが、後に、文字以外の任意のデータ型を利用可能な EgT[6] を実現し、CHISE-Wiki は EgT 上における文字 (character) 型データを対象とするサービスとして再定義された。

こうして CHISE 文字オントロジー中の文字の情報の Web 上での利用・共有は一応達成されたが、CHISE 文字オントロジーのデータ形式が S 式ベースの独自形式であるという問題と Web 上におけるそのデータへのアクセスが HTML ベースの（言い替えれば、機械可読性の低い）サービ

スである CHISE IDS 漢字検索と EgT に限定され、XEmacs CHISE の中で Emacs Lisp でプログラミングする場合に比べて極めて自由度の低いインターフェースしかないという問題は依然として残っていた。こうした問題を解決するためには、Web の世界における構造データ記述のための標準的なデータモデルである RDF [11] およびそれに基づく標準形式を採用することによって CHISE 文字オントロジーの持つリッチなデータをその情報をなるべく損なうことなく Web 上での可搬性を高めることと、Web API を用意して Web 上で CHISE のサービスを提供することが重要であるといえる。

しかしながら、この試み、特に CHISE の RDF 化はこれまで 3 回行われたが、今の所、完全に満足の行く結果は得られていない。SPARQL エンドポイントもサービスしているが、必ずしも十分に利用されているとは言いがたい上、CHISE のバックエンドであるグラフストレージ Concord [10] を直接参照せず、一度ダンプしたデータを利用しているため、データの同期のコストが高く常にタイムラグが発生するという問題を抱えている。また、Web アプリケーションの世界において SPARQL は必ずしも人気があるとはいえず、よりシンプルな Web API を実現した方が良いのではないかという声もあった[*1]。

このため、今回、CHISE のデータ・機能を十分に利用可能でかつシンプルな Web API の実現を試みる。JavaScript での利用を鑑み、この Web API で返されるデータは JSON とするが、この際、JSON-LD [9] 形式による RDF 化も検討する。

2 CHISE のデータモデル

2.1 Chaon モデル

CHISE では『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。

Chaon モデルでは、この文字素性の集合によって表現される文字のことを『文字オブジェクト』（文脈に応じて、文字もしくはオブジェクトと略される）と呼ぶ。

文字素性は素性の名前と値の対で表現することができる。文字素性という用語は文脈によって素性の名前（で指されるもの）と素性名と値の対を指すことがあり、両者を区別するために、前者を『素性名』、後者を『素性対』と呼ぶことにする。

[*1] 2017 年 8 月 24 日に北海道大学で開催された「CHISE 講習会」2 日目における劉冠偉氏の指摘

2.2 素性の種類

文字素性は大別すると

基礎素性 数値や識別子（シンボル）といったアトミックなデータ、または、それらのリストや配列を値として取る素性

ID 素性 オブジェクトに対する ID（素性名においてユニークな数値または識別子）を値として取る素性。ある ID 素性において、その素性を持つ各オブジェクトとその素性値である ID は 1 対 1 対応していなければならない（これにより、値である ID をキーにオブジェクトを得るための索引を作ることができる）

構造素性 オブジェクトを要素として持つリストや配列を値として取る素性

関係素性 オブジェクトの集合を値として取り、値に取った各オブジェクトと素性を持つオブジェクトの関係を表す素性。これは、オブジェクトをノードとした有向グラフのリンクとなるものである。関係素性対を持つオブジェクト（主語）とその値の各要素（目的語）の間には、その主語を値の要素とする逆関係素性対がその目的語に付く

メタデータ素性 元になる素性に対するメタデータを記述するための素性

に分類することができる。

ID 素性、関係素性、メタデータ素性の名前は、CHISE における素性名の命名規則によって、特定の形式を用いることになっている。

ID 素性は '=' から始まる名前を持つ。また、CHISE の『グリフ ID 素性名の命名規則』[2] に従い、例示オブジェクトの集合を指すための名前から、それを抽象化したオブジェクトの集合を指すための派生名を接頭辞（これを「包摂粒度接頭辞」と呼ぶ；表 1 の「S 式」列に示す）によって機械的に決定できるようにしている。[3]

関係素性は '->' もしくは '<-' から始まる名前を持つ。両者は互いに逆関係となっており、あるオブジェクト A が関係素性 '->*foo*' を持ち、その値が $(B_1\ B_2\ ...)$ である時、オブジェクト B_i は逆関係素性 '<-*foo*' を持ち、オブジェクト A はその素性値の要素のひとつとなる。

メタデータ素性は、言及対象となる素性名の後ろに '*' から始まる文字列を付けた名前を持つ。ここで、'*' の後に続く文字列を『メタデータ識別子』と呼ぶ。

また、各素性には '@*domain*(/*subdomain*/...)' という形式でドメイン識別子を付与することが可能である。メタデータ素性の場合、言及対象となる素性名にもメタデータ素性自身にもドメイン識別子を付けることができる（例：`<-ancient`@shuowen*note@shuowen-zhu（「『説文における「A 古文 B」』という記述に対する段玉裁の注)）。

3 問題点

3.1 Web API における問題点

ドメイン識別子の付いた素性やメタデータ素性などの階層的素性名や ID 素性における包摂粒度接頭辞、あるいは、関係素性のような CHISE の S 式表現における文字素性名の構造化には `@`, `/`, `*`, `=`, `>`, `<`, `+` といった特殊文字が区切り文字列の中で使われている。`>`, `<` は XML/HTML ではエスケープしなければ使えないし、その他の文字も URL, あるいは POSIX ファイルシステムやシェル等におけるパス名において特殊な意味を持つことがあり、IRI 上では使いにくい。このため、EgT[7] では『(EgT の) URI 表現』と呼ばれる記法を用いてこうした記号の表われる素性名を表現しているが、この変換規則における「.」を用いたエスケープ表現は JavaScript における *object.member* という記法とバッティングするため不便であるといえる。

この問題を解決するために「.」を利用しない別の変換規則を採ることも考えられるが、そもそも文字素性名の構造化をせずにすむデータモデル・表現を用いればこの問題を避けることができると考えられる。このための一つの方法は後述する中間的なノードを用いた素性値（目的語）の構造化である。

3.2 RDF 化における問題点

CHISE は集合ベースのデータモデルを採用しているが、オブジェクト間の関係を表すための『関係素性』と呼ぶ素性を用いることにより名前付き有向グラフを構成することができる。つまり基本的なデータモデルの構造としては Resource Description Framework (RDF) [11] と同様であるといえ、CHISE 文字オントロジーは情報を落すことなく RDF のグラフで表現できるはずであり、Turtle [12] や JSON-LD [9], RDF/XML [13] といった形式でシリアル化可能なはずである。

しかしながら、CHISE では階層的素性名方式に基づいて素性名に階層構造を導入することで情報の文脈やドメイン、あるいは、出典情報等のメタデータを表現する仕組みや、Chaon モデルに基づきオブジェクトに複数の ID を与え名前解決する仕組みや、包摂粒度を ID 素性の接頭辞で表現する仕組み等があり、CHISE の素性名をそのまま RDF の述語に対応させるのが簡単ではなかったり単純に対応させるだけでは問題があるケースがあった。

例えば、[7] では CHISE の文字情報を RDF/XML 形式で表現するために、CHISE の階層的素性名や包摂粒度情報付き ID 素性を XML で使用可能な文字の範囲内に収まるようにエスケープするとともに文の『具体化』(reification) を用いてメタデータ素性の情報を表現することを試みたが、この手法では CHISE 文字オントロジーでのセマンティクスを十分に表現したグラフになっておらず、また、文の『具体化』を用いることによって必要以上の複雑さをもたらしているといえ問題があった。そこで、[4] では、CHISE の階層的素性名や包摂粒度情報付き ID 素性を直接 RDF に翻訳するのではなく、それらの CHISE におけるセマンティクスを RDF で適切に表現するとしたらどうなるかという観点に基づいた RDF Turtle [12] への翻訳を試みた。しかし、この翻訳で

は 1 つの ID 素性が複数のトリプルになり記述が著しく繁雑になるという問題があった。また、いずれの試みでも CHISE の情報を完全に翻訳することができていなかった。

4 JSON-LD での表現

4.1 JSON-LD の概説

JSON-LD は JSON のオブジェクトの各メンバーのキー部に対し IRI を対応づけることによって、JSON データを RDF として解釈できるようにした形式である。

例えば、

```
{
  "名前": "LATIN CAPITAL LETTER A",
  "画像": "https://glyphwiki.org/glyph/u0041.svg",
  "情報": "http://www.chise.org/est/view/character/ucs=65"
}
```

は JSON データであり、人間にはそれが「A」という字に関するものであり、「画像」というプロパティーに SVG 形式の字形データが入ってそうだということは簡単に理解できるが、機械は人間のような直感的な理解はできない。そこで、「名前」や「画像」などのトークンの代わりにそれらの意味を示す固有の識別子を用いれば曖昧性が解消可能であり、機械にもそのセマンティクスを理解させることができると考えられる。

RDF ではその固有の識別子として IRI を用いる。例えば、schema.org の語彙を使って

```
{
  "http://schema.org/name": "LATIN CAPITAL LETTER A",
  "http://schema.org/image": {
    "@id": "https://glyphwiki.org/glyph/u5b57.svg"
  },
  "http://schema.org/url": {
    "@id": "http://www.chise.org/est/view/character/ucs=65"
  }
}
```

のように表現すれば明確である。ここで、2, 3 番目のメンバーの値部のオブジェクトの @id キーは値が IRI を用いた識別子であることを意味する。この記述では全てのプロパティーが IRI によって明確に識別され、また、値が IRI である場合 @id キーが書かれているためそれが単なる文字列かそれとも IRI なのかを機械は確実に判断できる。しかし、人間にとっては読みづらく JavaScript で処理する際にもやや面倒かも知れない。そこで、JSON-LD では『コンテキスト』（文脈）とい

う概念を導入することによってデータの明確さと簡便さの両立を支援している。

コンテキストは『用語』（JSON オブジェクトのキー部）を IRI にマッピングしたりその値の型や構造などを明示するための仕組みである。例えば、

```
{
  "@context": {
    "name": "http://schema.org/name",
    "char-image": {
      "@id": "http://schema.org/image",
      "@type": "@id"
    },
    "char-desc": {
      "@id": "http://schema.org/url",
      "@type": "@id"
    }
  }
}
```

というコンテキストを定義すれば、用語 name, char-image, char-desc の IRI を明示するとともに、用語 char-image, char-desc の値が IRI であることを明示できる。そして、この定義が `http://rdf.chise.org/contexts/chise.jsonld` から取得できるとすれば、

```
{
  "@context": "http://rdf.chise.org/contexts/chise.jsonld",
  "name": "LATIN CAPITAL LETTER A",
  "char-image": "https://glyphwiki.org/glyph/u0041.svg",
  "char-desc": "http://www.chise.org/est/view/character/ucs=65"
}
```

という風に簡潔に書くことができる。

4.2 文字オブジェクトの RDF での表現

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 1）。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。

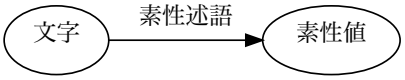

図 1　素性対の表現

4.3　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 の述語用名前空間は 1 つあれば十分であるが、JSON-LD や 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、

Concord **基本述語集合**　`http://rdf.chise.org/property/concord/main/`
CHISE CCS **述語集合**　`http://rdf.chise.org/property/chise/ccs/`
CHISE **漢字述語集合**　`http://rdf.chise.org/property/chise/hanzi/`

を設ける。また、これらに対しそれぞれ `concordmain:`, `chiseccs:`, `ideo:` という接頭辞を用いる。

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。

4.4　文字符号の表現

「文」という字は UCS/Unicode, Adobe-Japan1, JIS X 0208 といった文字符号・グリフセットや大漢和辞典などの字書に収録されており、それぞれ U+6587, Adobe-Japan1 の 3592, JIS X0208 の 4A38（42 区 24 点)、大漢和番号 13450 に対応する。細かくいえば、起筆の有無などの差は有り得るが、字体としての同一であると考える時、

> 字体「文」= U+6587 の例示字体 = Adobe-Japan1 の 3592 の例示字体 = JIS X0208 の 42 区 24 点の例示字体 = 大漢和番号 13450 の例示字体

という関係を考えることができる。

この時、異なる番号体系間での字体としての同一性 = は RDF の述語 `owl:sameAs` を用いて表すことができる。

また、「JIS X0208 の 42 区 24 点の例示字体」や「大漢和番号 13450 の例示字体」という概念は

「JIS X0208 の例示字体」や「大漢和番号の例示字体」というような番号体系に包摂粒度の情報を付加した概念と文字の番号（符号位置）の組によって表現することができる。

CHISE ではこの包摂粒度の情報を付加した文字の番号体系のことを「包摂粒度情報付き CCS 素性」と呼ぶ。

CHISE の S 式表現では、包摂粒度情報付き CCS 素性を表 1 の S 式欄にあるような接頭辞を用いて表現するが、JSON-LD ではこの表の JSON-LD 欄の用語を用いて表現する。

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

表 1　包摂粒度の表現

また、

```
"@context": {
  "unify": {
    "@id": "owl:sameAs",
    "@container": "@index"
  }
}
```

というコンテキスト定義を用い、

```
{
  "@context": "http://rdf.chise.org/contexts/chise.jsonld",
  "@id": "character:文",
  "unify": {
    "abstract-character:ucs": {
        "@id": "abstract-character:ucs/0x6587",
        "abstract-character-of": {
            "@id": "bare-ccs:ucs/0x6587",
            "CCS": "ccs:ucs",
            "code-point": 25991
```

```
        }
    },
    "abstract-glyph:adobe-japan1-0": {
        "@id": "abstract-glyph:adobe-japan1-0/03592",
        "abstract-glyph-of": {
            "@id": "bare-ccs:adobe-japan1-0/03592",
            "CCS": "ccs:adobe-japan1-0",
            "code-point": 3592
        }
    },
    "abstract-glyph:jis-x0208": {
        "@id": "abstract-glyph:jis-x0208/0x4A38",
        "abstract-glyph-of": {
            "@id": "bare-ccs:jis-x0208/0x4A38",
            "CCS": "ccs:jis-x0208",
            "code-point": 19000
        }
    },
    "abstract-glyph:daikanwa": {
        "@id": "abstract-glyph:daikanwa/13450",
        "abstract-glyph-of": {
            "@id": "bare-ccs:daikanwa/13450",
            "CCS": "ccs:daikanwa",
            "code-point": 13450
        }
    }
}
```

という風に表現する。ここで、unify の中の各キーは上述の用語 unify のコンテキスト定義にある @index によって RDF のグラフ上では消えて、Turtle 表現の場合と同様なグラフに展開される。しかしながら、Turtle 表現の場合と異なり、S 式表現の場合に比べると若干繁雑なものの、包摂粒度情報付き CCS 素性と『生の CCS 素性』の関係や符号位置の情報が一ヶ所にまとまっており、また、S 式における接頭辞に比べて初見でも理解しやすい表現になっている。

4.5 階層的素性名の表現

CHISE の階層的素性名では “@” の前にベースとなる素性名が付き、“@” の後にその情報が使われる文脈やドメインが付く。後者もまた “/” で区切られた複数のドメイン識別子からなる階層構造を採り得るものになっている。

[7] ではこのドメイン識別子の階層構造をエンコードして XML の名前空間のプレフィックス部に押し込むことによって表現していたが、この方法ではベースとなる素性名の指示対象と文脈やドメインが示す情報の関係が RDF のグラフとして表現されず問題である。そこで、JSON-LD における表現においても Turtle の場合 [4] と同様に中間ノードを用いて表現する方法を採ることにする。

階層的素性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中間ノードを設け、その中間ノードの述語 `context` で階層的素性名におけるドメイン情報を表現する。

4.6 メタデータ素性の表現

CHISE のメタデータ素性名では「*」の前に言及対象となる素性名が付き、「*」の後にそれに対するメタデータの種類を示す識別子やそのメタデータのドメイン識別子が付く。

このメタデータ素性で表現される情報も、4.5 節で述べた階層的素性名の表現と同様に、素性述語の目的語に中間ノードを設けて表現することにする。

4.7 文字間の関係の表現

CHISE のセマンティクス的には、メタデータ素性名は、その素性を持つオブジェクトを主語、言及対象となる素性名を述語、言及対象となる素性名に対応する素性値を目的語としたトリプルに対してメタデータを付与するものであるが、これをそのまま JSON-LD で表現するとしたら名前付きグラフを用いる必要がある。しかしながら、名前付きグラフは標準的な RDF におけるトリプルの範囲外にあるものであり可搬性が低い。

CHISE の S 式表現の構文的にはあらゆる素性に対してメタデータ素性を付けることができるが、現実には異体字関係の情報に対する典拠情報の記述が多い。このため、文字間の関係を示す関係素性の情報の RDF への翻訳において Web アノテーション [14] [15] を採用することにした。

しかしながら、CHISE の文字オブジェクトの記述の主語となる記述対象文字が Web アノテーションにおいてはターゲットという目的語になるという問題がある。そこで、

```
{
  "@context": {
    "@version": 1.1,
    "oa": "http://www.w3.org/ns/oa#",
```

```
    "target": "oa:hasTarget",
    "variant": {
      "@reverse": "target",
      "@container": "@index"
    }
  }
}
```

のように JSON-LD の @reverse キーワードを用いて異体字関係を示す用語 variant を定義することにより、

```
{
  "@context": "http://rdf.chise.org/contexts/chise.jsonld",
  "@id": "character:台",
  "variant": {
    "ideo:ancient-form-of": {
        "@id": "character:台/ideo:ancient-form-of",
        "body": {
            "@id": "character:台/ideo:ancient-form-of",
            "ideolink": {
                "@id": "character:台/ideo:ancient-form-of/link",
                "ideo:ancient-form-of": "character:%E5%97%A3"
            }
        },
        "source": "domain:jiyun"
    },
    "ideo:interchangeable": {
        "@id": "character:台/ideo:interchangeable",
        "body": [
            {
                "@id": "character:台/ideo:interchangeable$_1",
                "ideolink": {
                    "@id": "character:台/ideo:interchangeable$_1/link",
                    "ideo:interchangeable": "character:%E4%BA%88"
                },
                "source": [
                    "domain:shuowen-zhu",
                    "domain:daikanwa"
```

```
                ]
            },
            {
                "@id": "character:台/ideo:interchangeable$_2",
                "ideolink": {
                    "@id": "character:台/ideo:interchangeable$_2/link",
                    "ideo:interchangeable": "abstract-glyph:jis-x0208@1990/0x7075"
                },
                "source": [
                    "domain:fangyan-zhu",
                    "domain:daikanwa"
                ]
            },
            {
                "@id": "character:台/ideo:interchangeable$_3",
                "ideolink": {
                    "@id": "character:台/ideo:interchangeable$_3/link",
                    "ideo:interchangeable": "character:%E9%A7%98"
                },
                "source": [
                    "domain:shuowen-tongxun-dingsheng",
                    "domain:daikanwa"
                ]
            }
        ],
        "source": "domain:daikanwa"
    }
}
```

のように、見掛け上、記述対象文字を主語にした形に記述することができる。ちなみに、

```
{
  "@context": "http://rdf.chise.org/contexts/chise.jsonld",
  "@id": "character:台",
  "variant": {
    "ideo:ancient-form-of": {
        "@id": "character:台/ideo:ancient-form-of",
        "body": {
```

```
            "@id": "character:台/ideo:ancient-form-of",
            "ideolink": {
                "@id": "character:台/ideo:ancient-form-of/link",
                "ideo:ancient-form-of": "character:%E5%97%A3"
            }
        },
        "source": "domain:jiyun"
    }
}
```

は

```
{
  "@context": "http://rdf.chise.org/contexts/chise.jsonld",
  "@id": "character:台/ideo:ancient-form-of",
  "oa:hasTarget": "character:台",
  "oa:hasBody": {
    "@id": "character:台/ideo:ancient-form-of/link",
    "ideo:ancient-form-of": "character:嗣"
  },
  "source": "domain:jiyun"
}
```

と等価である。

なお、今回の JSON-LD 化において、関係素性を包摂関係、スクリプト間関係、異体字・類字関係、字体・字形用例に分類し、それぞれ用語 intergranularity, interscript, variant, glyph-example でくるむことでその分類を示すこととした。これは CHISE-wiki / EgT のようなサービスを作る際にこの順番に表示すれば同様の表示になることを意図したものである。

4.8 基礎素性の表現

基礎素性の値（基礎素性値）は概念的には RDF リテラルと同様なものといえるが、シンボルのような一意性を持ったもの、あるいは、リストや配列といった構造を持ったものもあるので、単純に全てを RDF リテラルで表現する訳にはいかない。

基礎素性値のうち、文字列や数値、真理値としての t は RDF においてもリテラルで表現できる。nil は真理値として使われている場合はリテラル、それ以外の場合は IRI を持ったエンティティーになる（空集合として使われている場合は RDF コレクションの rdf:nil とする）。t と nil 以外のシンボルは文字列で表現することもできるが、ユニーク性を示したい場合は IRI を持った

エンティティーとする。リストは RDF コレクションで表現できる。

5 CHISE Web API

5.1 URI パターン

CHISE の Web API の URI は次のようなパターンで構成される。

`http://chise.org/api/character/v0/`$function$`?`key_1`=`$value_1$`&`key_2`=`$value_2$`&...`

$function$ は XEmacs CHISE における Emacs Lisp API の関数に相当するものであり、key_n=$value_n$ は n 番目の引数である。ここで、key_n を「(n 番目の) 引数名」、$value_n$ を「(n 番目の) 引数値」と呼ぶ。

引数は引数名で識別されるため、引数の順番は任意である。

引数には必須のものと省略可のもの（オプショナル引数）があり、何が必須で省略可かは関数毎に決まっている。

5.2 データ表現

5.2.1 素性名

文字素性名 (feature name) の識別子。

JSON では

`concord:`$type$`/`$feature$

もしくは

包摂粒度識別子:$bare\text{-}CCS$

という形式の文字列で表現する。

但し、$feature$ は素性名の JSON-LD 表現である。また、包摂粒度識別子は表 1 の「JSON-LD」欄の用語である。また、$bare\text{-}CCS$ は包摂粒度情報のない『生の』CCS に対する識別子である。

■生の CCS

例 ucs
daikanwa
mj

■粒度情報付き CCS

例 abstract-glyph:daikanwa（大漢和の例示字体）

glyph-image:hng-kar（HNG の開成石經論語の代表字形）
abstract-character:iwds-1（IWDS-1 の抽象部品）

■粒度・ドメイン情報付き CCS

例　abstract-glyph:ucs@iso（ISO/IEC 10646-2:2000 における拡張漢字 B の例示字体）
repi:jis-x0208@1990（JIS X 0208:1990 の例示字形）

5.2.2　符号位置

符号化文字集合の符号位置。
JSON では数値型を用い、整数（自然数）の範囲で表現する。
URL 中では EST における URI 表現を用いる。

■URI 表現における 10 進表現

例　12345
01234 (= 1234)

■URI 表現における 16 進表現

例　0x4E00
0x210fe

5.2.3　文字

CHISE の文字オブジェクト。
JSON では文字列、または、下記で述べる文字型のオブジェクトを用いる。

■UCS 収録済みの文字　UCS で符号化可能な文字は JSON では文字列で表す。
例：”字”
[Note] ”\u4e00” のように Unicode エスケープ表現を用いることも可能である。なお、BMP 外の文字は当面”\uD840\uDC00” のように UTF-16 文字列として表現する。
一方、URI 中で用いる場合、UCS で符号化可能な文字は

http://chise.org/api/character/v0/get-spec?character=字
http://chise.org/api/character/v0/get-spec?character=%E5%AD%97

のように引用符を取りそのまま入れる。
[Note] URI 表現では UTF-8 を用いる。

5.2.4 文字型オブジェクト

（1 つ以上の ID 素性を持つ）CHISE の文字オブジェクトは下記のような JSON オブジェクトで表現することが可能である。

```
{
  "@type": "genre:character",
  "id": <CHARID>
}
```

特に、UCS で符号化できない場合、この表現を用いる必要がある。

例：

```
{
    "@type": "genre:character",
    "@id": "abstract-glyph:cns11643-7/0x2253"
}
```

5.3 関数

5.3.1 decode

http://chise.org/api/character/v0/decode?ccs=*ccs*&code-point=*code-point*

符号化文字集合 *ccs* の符号位置 *code-point* の文字を返す。存在しない場合は null を返す。

例：`http://chise.org/api/character/v0/decode?ccs=ucs&code-point=0x5B57`
`http://chise.org/api/character/v0/decode?ccs=rep.daikanwa&code-point=12345`

5.3.2 encode

http://chise.org/api/character/v0/encode?character=*character*&ccs=CCS

文字 *character* の符号化文字集合 CCS における符号位置を返す。存在しない場合は null を返す。

例：`http://chise.org/api/character/v0/encode?character=字&ccs=jis-x0208`
`http://chise.org/api/character/v0/encode?character=abstract-glyph:daikanwa/12345&ccs=abstract-glyph:jis-x0212`

5.3.3 get

http://chise.org/api/character/v0/get?character=*character*&feature=$feature$

文字 *character* の素性 $feature$ の値を返す。存在しない場合は null を返す。

例：U+3777 の漢字構造を取得

http://chise.org/api/character/v0/get?character=abstract-glyph:ucs/0x3777&feature=ideographic-structure

例：「見」を部品として含む漢字一覧を取得

http://chise.org/api/character/v0/get?character=見&feature=ideographic-products

5.3.4 get-spec

http://chise.org/api/character/v0/get-spec?character=*character*

文字 *character* の文字定義 (char-spec) を返す。存在しない場合は null を返す。

例：http://chise.org/api/character/v0/get-spec?character=知

http://chise.org/api/character/v0/get-spec?character=repi.hng-kar:100

6 Concord Web API

6.1 URI パターン

Concord Web API の URI は次のようなパターンで構成される。

http://chise.org/api/object/v0/$function$?key_1=$value_1$&key_2=$value_2$&...

$function$ は XEmacs CHISE における Emacs Lisp API の関数に相当するものであり、key_n=$value_n$ は n 番目の引数である。ここで、key_n を「(n 番目の）引数名」、$value_n$ を「(n 番目の）引数値」と呼ぶ。

引数は引数名で識別されるため、引数の順番は任意である。

引数には必須のものと省略可のもの（オプショナル引数）があり、何が必須で省略可かは関数毎に決まっている。

6.2 関数

6.2.1 decode

http://chise.org/api/object/v0/decode?type=*type*&feature=$feature$&value=*value*

Concord ジャンル *type* における ID 素性 $feature$ の値 *value* のオブジェクトを返す。存在しない場合は null を返す。

例：http://chise.org/api/object/v0/decode?type=domain&feature=rep.id&value=shuowen

http://chise.org/api/object/v0/decode?type=feature&feature=rep.json-ld-item&value=subsumE

6.2.2 get

http://chise.org/api/object/v0/get?object=*object*&feature=$feature$

オブジェクト *object* の素性 *feature* の値を返す。存在しない場合は null を返す。
例：CCS 素性 mj の素性属性 value-presentation-type を得る
`http://chise.org/api/object/v0/get?object=concord:feature/mj&feature=value-presentation-t`

6.2.3 get-spec

http://chise.org/api/object/v0/get-spec?object=*object*
文字 *object* の定義 (object-spec) を返す。存在しない場合は null を返す。
例：`http://chise.org/api/object/v0/get-spec?object=concord:domain/hanshu-zhu`

参考文献

[1] *CHISE Project*. `http://www.chise.org`.

[2] 守岡 知彦. "CHISE に基づくグリフ・オントロジーの試み". In: **じんもんこん** *2009* **論文集**. Vol. 2009. 情報処理学会シンポジウムシリーズ 16. 情報処理学会. 情報処理学会, 2009, pp. 9–14.

[3] Tomohiko Morioka. "Multiple-policy Character Annotation based on CHISE". In: *Journal of the Japanese Association for Digital Humanities* 1.1 (Nov. 2015), pp. 86–106.

[4] 守岡 知彦. "CHISE の RDF 化の試み". In: **情処研報** 2017-CH-114.1 (May 2017), pp. 1–6.

[5] 守岡 知彦. "CHISE のセマンティック Wiki 化の試み". In: **情処研報** 2010-CH-87.8 (July 2010), pp. 1–8.

[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". In: **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〜**. Vol. 2010. 情報処理学会シンポジウムシリーズ 15. 情報処理学会. 情報処理学会, Dec. 2010, pp. 33–40.

[7] 守岡 知彦. "CHISE の階層的素性名の RDF 化の試みについて". In: **情処研報** 2013-CH-97.3 (Jan. 2013), pp. 1–6.

[8] Taichi KAWABATA. *Reference Information on Ideographs and IDS Informations on Japanese researchers*. Informational N 1139. ISO/IEC JTC1/SC2/WG2 IRG, May 2005.

[9] Manu Sporny et al. *A JSON-based Serialization for Linked Data*. World Wide Web Consortium (W3C). `https://www.w3.org/TR/2014/REC-turtle-20140225/`, Feb. 2014.

[10] 守岡 知彦. "Concord: プロトタイプ方式のオブジェクト指向データベースの試み". In: *Linux Conference* **抄録集** 4 (2006).

[11] *Resource Description Framework (RDF): Concepts and Abstract Syntax*. World Wide Web Consortium (W3C). http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, Feb. 2004.

[12] David Beckett et al. *RDF 1.1 Turtle*. World Wide Web Consortium (W3C). `https://www.w3.org/TR/2014/REC-turtle-20140225/`, Feb. 2014.

[13] Dave Beckett, ed. *RDF/XML Syntax Specification (Revised)*. World Wide Web Consortium (W3C). http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/, Feb. 2004.

[14] *Web Annotation Data Model*. https://www.w3.org/TR/2014/REC-turtle-20140225/. Feb. 2017.

[15] *Web Annotation Vocabulary*. https://www.w3.org/TR/2017/REC-annotation-vocab-20170223/. Feb. 2017.

[16] 守岡 知彦. *CHISE IDS* **漢字検索**. http://www.chise.org/ids-find.