NetCDF 気候・予報（CF）メタデータ規約

NetCDF Climate and Forecast (CF) Metadata Conventions

バージョン1.4, 2009年2月27日

日本語訳2015年7月9日（木曜日）版

原著者

Brian Eaton, NCAR
Jonathan Gregoy, Hadley Centre, UK Met Ofice
Bob Drach, PCMDI, LLNL
Karl Taylor, PCMDI, LLNL
Steve Hankin, PMEL, NOAA

追加著者

John Caron, UCAR
Rich Signell, USGS,
Phil Bentley, Hadley Centre UK Met Office
Greg Rappa, MIT
他にも多くの者が変更提案の議論を通じて貢献している。

翻訳
豊田英司[1]

要旨

この文書は netCDF アプリケーションプログラマーインターフェイス[NetCDF]によって作られるファイルの処理と共有を推進するために設計された、気候と予報のメタデータのための CF 規約を記述するものである。この規約が定義するものは、それぞれの変数が何を表現しているのか、そしてデータの時空間的な性質についての決定的な記述を与えられるようなメタデータである。これにより、異なる出所からのデータの利用者はどの量どうしが比較可能なのかが決定可能になり、そしてデータを抽出・格子変換・そして表示する強力なアプリケーションを構築するための助けとなる。

CF 規約は COARDS 規約 [COARDS] を一般化し拡張したものである。拡張されたメタデータには、標準名の指定によりそれぞれの変数の精密な定義を与えるもの、無次元の鉛直座標に対応する鉛直位置を記述するもの、そして非長方形の格子による空間座標を与えるものが含まれる。気候と予報のデータには、単純に時空間上の点を代表するとはいえないものもしばしばあるので、座標間隔、多次元セル、および気候的時間座標を記述し、またデータ値が時間間隔またはセルをいかに代表するかを示すような拡張も行われた。この標準では COARDS の次元順序の制約を緩和し、データセットの大きさを縮減する方法を指定している。

1 山口 悟史さんより誤訳・訳脱のご指摘をいただきました。

目次

はじめに

ホームページ

ホームページ[2]には次のものへのリンクがある:過去の版文、現在の作業文書、CF 規約[3]に適合するファイルを処理するアプリケーション、ならびに現行規約の解釈、明確化、および変更・拡張提案のためのメーリングリスト。

履歴

この文書は CF ガバナンス・ルール[4]に従い合意された変更・訂正に基づき更新される。完全な履歴については付録 G「履歴」を参照せよ。暫定的変更については、新規、~~削除された文言~~、そして [コメント] のように記載する（この訳では作業未了）。

【訳注】

- この文書において文献引用は [NetCDF] のように大括弧でくくって表現されている。文献引用と章節番号への参照はリンクになっていて引用文献リストにジャンプすることができる。

- 原文の規定ぶりを明確にするため、訳注は原則として脚注とした。まずは読み飛ばして差し支えない。語を補わないと意味が通らないところは楷書体10ポイントで訳文に挿入した。原文のイタリック *italic* による ***強調は訳文ではゴシック体で表現した***。原文で CDL 言語によるコードや属性名等をマークアップするために用いているタイプライタ字体 `monospace font` はそのまま用いたが、利用者で適宜置き換えて読むべき文字列（変数）はイタリックに改めた。

第1章 序論

1.1 目的

NetCDF ライブラリ[NetCDF]の設計では、読み書きの対象たるデータはよく定義された規則にしたがって構造化され、さまざまな計算機プラットフォームに容易に移植できるようになっている。NetCDF インターフェイスは自己記述的なデータの作成を可能にするが、netCDF インターフェイスを使えば自己記述化が要求され結果として自己記述性が担保されるというわけではない。CF 規約の目的は、ファイル[5]中のそれぞれの変数について（適切ならば単位を含め）何を表わすかが与えられ、そして変数のそれぞれの値が時間と空間に（地球基準の座標に相対で）位置づけられる、という意味[6]で自己記述的になるに十分なメタデータを規約に適合するデータセットに与えることである。

この規約の重要な利点は、最低限のユーザの介入しか必要とせずにデータを表示し、または

---

[2] http://www-pcmdi.llnl.gov/cf/

[3] 訳注：辞書的には convention は協定・条約などとされるが、ここでは netCDF データセットの構成の仕方やメタデータの表現のしかたに関する規則のこと。日本の気象学界の慣用に従い、netCDF convention(s) は netCDF 規約と訳すことにする。この文書を通じて単数形 convention が個々の規則を、複数形 conventions が文書総体を指すが、文脈から明らかなので両方「規約」と訳した。

[4] http://cf-pcmdi.llnl.gov/governance

[5] 訳注：ファイルはデータセットと同義に使われている。厳密に言えば、OPeNDAP のように netCDF API 上のデータセットが格納メディア上のファイルと一致するとは限らない場合もあるので、データセットというべきところである。総じてこの文書は同一段落で同じ語を繰り返すのを避け類義語で言い換えるという一般的欧文の規範に引きずられて無理に類義語を通用させており、技術文書としての明確さを損なっているが、あえてなるべく直訳するようにした。

[6] 訳注：ここでいうメタデータは（ダブリンコア[DC]や ISO 19115 [ISO19115]のような）発見メタデータというより利用メタデータの色彩が強いことがわかる。

データの一部分に処理を行うようなソフトウェアが可能になることである。ある場[7]が時空間中にさまざまに位置づけられる（人間ならばそのうちどれが等価かすぐにわかるような）やりかたを記述するメタデータを提供することが可能になっている。メタデータの表現法を制約する目的は、メタデータを機械が解読してデータ値をその時空間上の位置に自動的に結びつけるようなソフトウェアを書くことを現実的にすることである。メタデータを人間のユーザにとって書きやすくまた理解しやすくすることも同じくらい重要である。

この標準[8]は気候並びに大気、地上及び海洋の予報のデータに使われることを意図しており、特にモデルによって作られるデータを念頭において設計された。標準によって現実的にカバーできることには限度があることは我々も認識しているので、気候・予報のメタデータの設計において共通かつ頻繁に問題となると信じる事項に限定している。そのため気候・予報データに必要なメタデータについて明確、適切、かつ柔軟な定義を与えることが我々の主目的である。この規約は具体的には netCDF 標準として書かれているのではあるが、我々の感触としては、ほとんどの概念は netCDF に限らず幅広い応用ができるものと考えている。メタデータ・オブジェクトは netCDF 以外のファイル形式にも含められるからである。異なる形式のファイルの間でのメタデータの変換は、全ての形式が類似した概念に基づいていれば容易になるだろう。

この規約は COARDS 規約 [COARDS] の後方互換になるよう設計されている、すなわち、COARDS に適合するデータセットは CF にも適合する。したがって、CF 規約を実装する新しいアプリケーションは COARDS データセットも処理できるようになろう。

また、COARDS 標準への適合性を最大限高めることにも意を砕いた、すなわち、COARDS メタデータ規約が適切な記述を与えるところでは、我々はそれを使うことを要請している。COARDS を拡張するにあたっては、そのような拡張に依存しないコンテンツならば、COARDS 標準に従うアプリケーションも処理可能となるように実現されている。

1.2 用語

この文書で netCDF ファイルの構成要素を指すのに使われている用語は、netCDF ユーザーズガイド（NUG）[NUG]で定義されている。定義のいくつかは便宜のため再掲している。

補助座標変数（auxiliary coordinate variable）
: 座標データを保持する netCDF 変数ではあるが、（NUG で定義されこの標準でも使われているような）座標変数と異なり、次元名と無関係な変数名を持っているもの。

境界変数（boundary variable）
: 境界変数は座標データを保持する変数に関連付けられている。データ値が時空間又は他の次元のある領域を占めるセルの中の状況についての情報を与えるとき、境界変数はセルの大きさの情報を与える。

CDL **文法**（CDL syntax）
: NetCDF ファイルの内容を記述するのに使われる ASCII フォーマットを CDL (network Common Data form Language) という。この形式では配列はプログラム言語 C のようなインデックス規約で表現される、すなわちインデックスは0から始まり、多次元の配列では、ファイル保存順序の観点から最も早く変わる次元は最後に宣言された次元である。NetCDF ユーティ

[7]訳注：場（field）は変数（variable）と同義に通用している。

[8]訳注：本来ならば一般的な標準（standard）は netCDF に限られる規約（convention）より広い概念であるが、この文書では同義に通用している。

リティである ncdump 及び ncgen はこの形式を使う（NUG 第10章[9]参照）。この文書の全ての例も CDL 文法で書かれている。

セル（cell）

ひとつ又はそれ以上の次元の領域で、その境界は頂点（vertex, 複数 vertices）[10]で記述される。一次元のセルは時々インターバルとも呼ばれる。

座標変数（coordinate variable）

NUG 第2.3.1節[11]で定義されているように、1次元変数でその次元と同じ名前 [たとえば time(time) のように] を持ち、データ型は数値型で、値は単調に並んでいるものである。座標変数では欠損値は許されない。

格子写像変数（grid mapping variable）

ある種の格子写像（地図投影法など）を定義する属性のコンテナ[12]として使われる変数。意味のあるデータは含まれないので、変数の型は任意でよい。

緯度次元（latitude dimension）

NetCDF 変数がもつ次元のうち、緯度の座標変数に関連付けられているもの。

経度次元（longitude dimension）

NetCDF 変数がもつ次元のうち、経度の座標変数に関連付けられているもの。

多次元座標変数（multidimensional coordinate variable）

補助座標変数のうち多次元のもの。

勧告（recommendation）

この規約において勧告とは一般的な誤りを減らすために有益な助言として提示されるものであり、従わないからといって CF 規約に適合しないことにはならない。一部の属性は COARDS 規約との後方互換性を維持するためにあえて必須（required）ではなく勧告されている。アプリケーションはデータセットが勧告に従うことに依存してはならない。

スカラー座標変数（scalar coordinate variable）

スカラー変数で座標データを含むもの。機能的には、大きさ1の座標変数、または大きさ1の補助座標変数と等価である。

時空間次元（spatiotemporal dimension）

NetCDF 変数がもつ次元のうち、時間、空間、またはその両方の位置を特定するために使われるもの。

時間次元（time dimension）

NetCDF 変数がもつ次元のうち、時間の座標変数に関連付けられているもの。

鉛直次元（vertical dimension）

NetCDF 変数がもつ次元のうち、鉛直の座標変数に関連付けられているもの。

1.3 全体像

NUG と同様、この規約では変数名や次元名を標準化することはせず、属性名と一部の属性値だけを標準化するものとする。この節の全体像は、次節以降のより完全な記述で補足されるべきものである。この規約で使われている全ての属性は付録 A「属性」でまとめられている。

我々はこの規約に適合するデータセットを識別するため、NUG が定義する `Conventions` 属

[9] http://www.unidata.ucar.edu/packages/netcdf/guidef/guidef-15.html#HEADING15-0

[10] 訳注：n 次元空間のそれぞれの次元について有界連続の領域をとれば、それらの直積は n 次元直方体となり、2^n 個の頂点がある。ここでいう頂点とはそのようなものを指すが、セルは多角形や多面体かもしれないので数は $2n$ とは限らない。1次元であれば頂点は境界・端点と同じになる。

[11] http://www.unidata.ucar.edu/packages/netcdf/guidef/guidef-7.html#HEADING7-67

[12] 訳注：属性を貼り付けるために便宜上導入される変数。

性に文字列値 "CF-1.4" を与えることを勧告する。

ファイルの内容の一般的記述は次の属性に含まれるべきである: title, history, institution, source, comment 及び references (2.6.2「ファイルの内容の記述」参照)。COARDS との後方互換性のため、これらの属性は必須(required)とはしないが、しかしファイル内容を人間に読みやすく文書化するためこれらの利用は勧告とする。

NetCDF ファイルの中のそれぞれの変数は、units, long_name, 及び standard_name 属性による記述を持つ。属性 units 及び long_name は NUG で定義されており、standard_name 属性はこの文書で定義されるものである。

属性 units は次元付きの量を表現する全ての変数について必須である(ただし、7.1「セル境界」で定義される境界変数はその限りでない)。属性 units の値は UNIDATA Udunits パッケージで認識できる文字列である(例外として認められる形は3.1「単位」で論じる)。

属性 long_name 及び standard_name はそれぞれの変数の内容を記述するために使われる。COARDS との後方互換性のためにこれらはいずれも必須になっていないが、少なくともどちらかを使うことが強く勧告される。標準名を使うことにより、一般的に解析される変数をまぎれなく識別[13]できるようになり、気候・予報データの交換が容易になる。

この規約では4つの座標を特別に扱う:緯度、経度、鉛直そして時間である。全ての変数は、時空間の中で位置づけが持ちうるならば時空間座標での位置づけを与えるようなメタデータを持たなければならない。この規約では二つの部分がその役割を担う。座標データを保持する変数を識別するための規約があり、そのデータが表現する座標の種類を識別するための規約がある。

座標データを保持する変数を識別する方法は二つある。一つ目は、NUG で定義された「座標変数」を使う方法。***変数の次元が一次元の空間又は時間の座標に対応する場合、座標変数を使わなければならない。*** 座標変数が適切でない場合は、座標データを保持する変数は coordinates 属性で特定される。

座標データを保持する変数がひとたび識別されたら、これらの変数の一つ一つが表わす座標の種類を特定するための規約が必要である。緯度、経度、そして時間の座標はひとえに units 属性だけによって識別できる。圧力の単位を持つ鉛直座標は units 属性で識別できる。そのほかの鉛直座標では座標値の増える向きが上(up)か下(down)かを示す positive 属性を使わなければならない。座標の種別を単位で識別するためには外部ソフトウェアパッケージ [UDUNITS]が必要となるので、我々はオプションとして緯度、経度、鉛直又は時間に対応する座標を直接識別するための axis 属性を用意する。

緯度、経度、及び時間は国際的に認知された標準で定義されている。したがってこれらの種類の座標を特定することにより、データ値を時間及び地球上の地点に位置づけることが十分可能である。他面で、鉛直座標は必ずしもデータ値を地球表面に対して鉛直方向に位置づけるのに十分ではない。特にモデルはその数理的定式化において使った無次元鉛直座標に対してデータを出力することがある。全てのデータ値を空間的に位置づけることができるように、この規約では一般的な無次元鉛直座標及びそのためのメタデータの定義を付録 D「無次元鉛直座標」に載せている。これらの定義は無次元の座標値と、地球表面に対して一意に位置づけられるような次元付きの値との写像を定義する。この定義は座標変数に standard_name 及び formula_terms 属性で関連付けられる。COARDS との後方互換性のため、これらの属性の仕様は必須ではないが、強く勧告される。

[13]訳注:原文 identify を以下「識別」と訳した。

しばしば、データ値が時空間の単一の点ではなく、むしろインターバルなり多次元のセルを代表することがある。この規約では、インターバル又はセルの大きさを特定するための `bounds` 属性が定義されている。データがセルを単純な統計的方法で代表するといえるとき、これらの方法は `cell_methods` 属性を使って示すことができる。この属性の重要な応用が、気候的あるいは日変化の統計を記述することである。

データ全体の大きさを小さくする方法が、パッキング（packing）と圧縮（compression）[14]である。パッキングは、格納する数値の精度を下げることによってデータの容量を減らすことであり、NUG で定義されている `add_offset` 及び `scale_factor` 属性を使って実現される。他方、圧縮は欠損値を保存しないことによって精度を失わずに大きさを縮減する。属性 `compress` がこの目的のために定義されている。

1.4 COARDS 規約との関係

この規約は COARDS 規約[COARDS]を一般化し拡張している。COARDS との ***後方互換性***を維持することは設計上の大きな目標の一つであった。そこで、この規約に適合するデータセットを処理するために書かれたアプリケーションは COARDS に適合するデータセットを処理できるだろう。また我々は COARDS 標準への ***適合性***を最大限にするよう努力した。COARDS でも利用できたメタデータだけを必要とするようなデータセットは、COARDS に適合する[15]アプリケーションでも依然として処理できるだろう。しかしながら、新しいメタデータ内容を提供する拡張により、そして COARDS の要請を緩和した点により、この規約に適合するデータセットは必ずしも COARDS 規約に従うアプリケーションによって認識されるとは限らない。この規約の特徴のうち、COARDS に適合しない netCDF ファイルを作ることにつながるものを以下にまとめる。

COARDS 標準では格子の定義は独立な緯度、経度、鉛直、及び時間の軸からなるものとした。それぞれの軸の種別を識別するために必要となるメタデータを標準化する代わりに、COARDS は軸（すなわち次元）の順序を経度、緯度、鉛直、そして時間の順（ここでは経度が最も速く変わる次元[16]）でなければならないとした。入出力の性能上の配慮[17]から、モデルが COARDS の順序でデータを出力できないことがある。CF 規約では次元の順序に関して固定的な制限を置かないこととしたが、データ作成者には COARDS 標準の順序にとどまるよう努力を奨励している。COARDS と異なる軸の順序を使うことにより、いくつかのアプリケーションがファイルを描画できなくなり、互換性が制約されることになる。モデルの出力時にバッファリング処理[18]を行うことにより、モデルコードの軸順序が COARDS ファイルの軸順序に適合しないときに性能上のペナルティを最小化することがしばしば可能である。

COARDS は無次元の鉛直座標を特定する問題を扱っているが、しかし無次元の値を地球表面に対して位置づけられるような有次元量に写像する機構はなんら提供していない。後方互換性のため、我々は引き続き無次元鉛直座標の `units` 属性が “`level`”, “`layer`”, あるいは “`sigma_level`” という値をとることを認める（要請はしない）。しかし勧告としては、

---

[14]訳注：気象庁「国際気象通報式別冊」では packing を圧縮と訳しているが、一般的わかりやすさのためあえて異なる訳語にした。

[15]訳注：こうはいっているが、COARDS も CF もソフトウェアが適合するための要件を明確にしていないから、たとえば Fortran コンパイラが ISO 1539 に適合する、というようにこれら標準に適合するか否かが判定できるというわけではない。

[16]訳注：CDL では逆順となる。COARDS にもそう書いてある。

[17]訳注：次元順序が(I, J ,K) [緯度-経度-鉛直] ではないモデルがあるから、ということだろう。

[18]訳注：要するに、変数毎あるいは南北スライス毎などでバラバラに出力するコードがあればまとめて、タイムステップごとに軸順序を並べ替えて出力すればそんなに遅くならない、ということ。

standard_name 及び formula_terms 属性を使って無次元鉛直座標の適切な定義（4.3.2「無次元鉛直座標」参照）を識別するものとする。

CF 規約は COARDS 規約の対象外のデータの特性（data property）を記述できるようにするような属性を定義する。これらの新しい属性は COARDS 規約に違反するものではないが、COARDS に適合するデータセットだけを認識するアプリケーションはこの新しい属性によって期待されている機能が使えない。新しい属性の機能は次のようである。

- 標準名を使った物理量の識別
- 次元のない鉛直座標の記述
- 補助座標変数を使った次元との関連付け
- データ変数とスカラー座標変数との結合
- 次元にラベルをつけること
- インターバルやセルの記述
- インターバルやセルの上で定義されたデータ特性の記述
- 気候学的統計の記述
- 欠損値のある変数のデータ圧縮

第2章 NetCDF ファイルとその構成要素

NetCDF ファイルの構成要素については NUG[NUG]の第2節で記述されている。この節では、ファイル名および netCDF ファイルの基礎的な構成要素についての規約を記述する。また、ファイルの内容を記述するための新しい属性も導入される。

2.1 ファイル名

NetCDF ファイルはファイル名拡張子 ".nc" を持つべき[19]である。

2.2 データ型

NetCDF のデータ型 char, byte, short, int, float 又は real, および double は全て利用可能である。型 char は数値データを意図したものではない。大きさ1バイトの数値データは byte データ型を使って格納すべきである。全ての整数型は netCDF インターフェイスによって符号付と扱われる。ただし、byte 型については NUG 規約により、符号なしの範囲を valid_min, valid_max, あるいは valid_range 属性に与えることによってを格納できる。

NetCDF は文字列型をサポートしないので、は文字の配列として表現されなければならない。この文書では、文字データ（char 型）の一次元配列を簡単のため「文字列」と呼ぶ。文字列の n 次元配列すなわち($d1$, $d2$, ... dn) を表現したい場合は ($d1$, $d2$,... dn, max_string_length) [20]の n+1次元の文字型配列として実現されねばならない。ここで最後に宣言された（最も速く変化する）次元は配列の中のもっとも長い文字列を納めるのに十分長くなければならない。このようにするので、配列の中の文字列は定義上長さが等しい。たとえば、英語の月名を保持する文字列配列は、もっとも長い名前の “September” を保持するために (12, 9) という次元を持つことになるだろう。

[19]訳注：原文 should を以下「べき」で訳す。これは勧告に準ずる位置づけと思われる。
[20]訳注：原文は n 次元配列と dimension (n, max_string_length) とを対比していて論理的でない。

【訳注：このような固定長文字列は、どちらかというとCのヌル終端文字列よりFortranの文字型に近い。たとえば、"September"はちょうど9文字なのでファイル上でヌル終端はされないし、その9文字をnc_get_vara_text()で読み出すならば、あくまで9文字が読み出されるだけであって、自動的に10バイト目がヌル終端されるわけではない。しかしながら、ncgenはCDLにおいて"May"のように宣言された長さに満たない文字列が与えられれば残りはヌルで埋め、ncdumpはヌル終端されている文字列を"May\0\0\0\0\0\0"ではなく"May"のように表示するし、charの_FillValueはゼロなので、netCDFライブラリ作者はCのような可変長文字列としての利用を期待していることになる。訳者思うに、CF規約の現行規定はミスリーディングである。Cのように可変長文字列として使いたいならば配列寸法を(12, 10)としてヌル終端を義務付けるべきであるし、Fortranとの整合性を重視するならば、文字列の長さが足りない場合はスペースで埋めることが想定されるので、ヌルで埋める旨を明記すべきである。要するに、長さ9の配列に"May"をどう格納したらいいのか現行規定では判然としない。】

2.3 命名規約

変数名、次元、および属性名は英字で始まり、英数字または下線からなるものとする。NetCDFインターフェイスとしてはハイフンも使えるが、COARDS規約への適合[21]のためあえて制限している。NetCDFインターフェイスは先頭の下線も受け付けるが、NUGはこのような名前はシステム用に予約されているとしている。

NetCDFの名前では大文字と小文字は区別されるが、勧告としては名前は純粋に大文字と小文字の違いだけで区別されることとはならないものとする;つまり、もし大文字と小文字の違いを無視した場合、2つの同じ名前があるべきではない。また、名前は自明に意味のある名前とすること、もし可能ならば、名前を列挙したときにファイルがより効果的に自己記述的となるようなものとすることを勧告する。

この規約はいかなる変数名あるいは次元名も標準化しない。この文書で標準化される属性名や属性の内容は英語で与えられており、可搬性のため適合するnetCDFファイルにおいては英語で与えられるべきである。英語以外の言語は変数、次元、及び非標準の属性の名前で認められる。標準化された属性のうちいくつかの内容は、標準化されない文字列値であり、したがって英語でなければならないわけではない。たとえば、変数が何を表わすのかを記述する`long_name`属性(3.2「長い名前」参照)は英語以外の言語で書くこともできるが、しかし`standard_name`属性(3.3「標準名」)での記述は英語で書かれている標準名の表からとらねばならない。

2.4 次元

変数のゼロを含め任意の数の次元を持つことができる。次元はそれぞれ異なった名前を持たねばならない。*COARDS* ***では次元数を4に限定することが強く勧告されているが、我々はより柔軟性を許容することにしたい。*** 変数の軸を規定する次元はその変数が含む量に対する軸を定義する。時空間ではない次元が含まれてもよい。この文書にもいくつかの例が見出される。一定の条件の下では、ある量に対する次元が二つ以上必要になることがある。たとえば、二つの異なる鉛直レベルにおける温度の相関を表わすには二次元の確率密度関数を使うことになるだろうから、それを含む変数は両方の軸が温度となるだろう。

もしある変数の一部又は全部の次元が「日付又は時間」(T)、「高さ又は深さ」(Z)、緯度(Y)、または経度(X)と解釈できるならば、これらの次元をファイルに対応するCDLでの定義においてT, Z, Y, そしてXの順になるようにすることを勧告する(要請はしない。1.4「COARDS規約との関係」参照)。可能ならばいつでも、その他全ての次元は時空間次元の左に置かれるべきである。

次元の長さはいくつでもよい、1でもよい。ある変数の全ての値に対して、ある座標の単一の値

[21]訳注：適合条件は明らかになっているか？

が適用されるならば、変数にその情報を付加するためには、長さ1の次元と一要素の座標変数を使うことを勧告する。スカラー座標変数を使うことも認められており、そうすれば長さ1の次元をデータ変数に結びつける必要がなくなる。要素ひとつにせよスカラーにせよ、座標変数を使う利点は、その単一値の量（座標値）について記述するために、座標変数の全ての属性が使えることである。たとえば、地上高1.5m の温度データを保持する変数は、高さ1.5m を与える単一値の座標をもち、また時間平均の量は平均期間の始期と終期を記録するために境界変数を伴った単一値の時間座標を持つ。

2.5 変数

この規約は変数名を標準化しない。

座標データを含む netCDF 変数は***座標変数***、***補助座標変数***、***スカラー座標変数***、あるいは***多次元座標変数***と呼ばれる。

2.5.1 欠損値

NUG 規約（[NUG]第8.1節）は欠損値を表現するのに `_FillValue`, `valid_min`, `valid_max`, 及び `valid_range` 属性を用意している。

欠損値に関する NUG 規約は第2.3版から第2.4版の間に大きく変化した。第2.3版では欠損値として確実な位置づけを持つのは _FillValue だけで、missing_value は他の値を使う場合の例として挙げられているのみであった。属性 valid_range （以下この節では代替としての valid_min, valid_max を含めて考える）は有効値を定義するとされており、欠損値との関係は明確にはされていなかった。第2.4版から NUG は欠損値を `valid_range` の範囲外の全ての値と定義し、`valid_range` が定義されていない場合に `_FillValue`（これはライブラリによって定義される既定値をもつ）から `valid_range` を定義する方法[22]を指定するようになった。単一の欠損値しか必要ではないならば、我々はそれを `_FillValue` 属性で指定することを強く勧告する。このようにすることにより、第2.4版以前と以降のどちらに従う一般的アプリケーションも、その欠損値を認識することを保障するからである。

NetCDF ライブラリはスカラーで変数と同じ型の属性 `_FillValue` を認識して、変数に割り当てられたディスクスペースを初期化する値（フィルバリュー）として使う。未だ書き込まれていない値を読み込んだ時に返されるので、この値は未定義又は欠損値を表わす特別な値とみなされる。属性値 `_FillValue` は（もし使うなら） `valid_range` によって指定された範囲の外であるべきである。それぞれのデータ型に対するフィルバリューの規定値は netCDF ライブラリが定義する（NUG 第7.16節）。

属性 `missing_value` は NUG によって廃止予定（deprecated）とみなされており[23]、我々も使

---

[22]訳注：便宜のため説明すると、次のフローで判定する；
- データ変数が byte 型で、_FillValue が明示的に定義されていなければ、すべての可能な値が有効値
- そうではない場合は、_FillValue が正ならそれを無効最大値、あるいは無効最小値とする
  - この場合は _FillValue が明示的・暗黙に定義されているかを問わない
  - 整数型ならば、有効最大値 = 無効最大値－1, または有効最小値 = 無効最小値＋1
  - 浮動小数点型ならば、有効最大値・最小値は _FillValue との間に最小表現可能値（最下位ビット1の差）の2倍だけの差を設ける

[23]訳注：実際には第2.4版以降の NUG に missing_value を廃止予定と明示的に断ずるような記述はない。本文で missing_value を例示した箇所が消え、欠損値に関し valid_range により明確な位置づけが与えられ、属性規約では missing_value は valid_range 範囲外でなければならないという規定が追加されただけである。しかしながら、CF と netCDF のコミュニティの重なりから考えて、CF のこの規定が放置されているということは missing_value 属性は廃止予定となったという認識が少なくとも違和感なく受け入れら

用を勧めない。しかしながら COARDS との後方互換性のため、この標準は引き続き未定義又は欠損値を示すための missing_value 属性の使用を認識する。

属性 scale_factor, add_offset （8.1「パックされたデータ」参照）またはその両方を使った変数の欠損値は、変数の外部値、すなわち netCDF ファイルに格納される値に対して解釈される。変換（スケール、オフセットまたはその両方による）と欠損値の両方の属性をもつ変数を処理するアプリケーションは、まずデータ値が有効であることを確認して、つぎに変換を適用すべきである。欠損値は変換すべきでないことに留意せよ。欠損値は有効値の範囲外であるから、変換を適用する結果不適切な演算が生じるかもしれない。たとえば、float 型の _FillValue の既定値は IEEE 単精度実数の表現できる最大値にたいへん近いので、（単精度演算で）100倍のスケールをかければ「無限大」となってしまう。

2.6 属性

この標準は多くの属性（いくつかは必須で、他は任意指定）を記述する。しかしファイルには非標準の属性が含まれうる。このような属性はこの標準への違反を表わすものではない。アプリケーションプログラムは、理解できない属性またはプログラムの目的と無関係な属性を無視しなければならない。属性名をつけるにあたっては、あてはまるならば規約の属性名を使うべきである。非標準であっても可能な限り意味のある名前とすべきである。属性を導入する前に、その情報が変数としてよりよく表現できないか検討すべきである。一般に、属性として提案しようとするものが、何らかの補助的データを必要とするか、多次元配列であるか、一次元配列であってもその次元を解釈するのに他の netCDF 次元を参照するか、あるいは大きな記憶容量を必要とするなら、代わりに変数を使うべきである。この標準が可能な値をあらかじめ決めた文字列属性を定義するとき、その可能な値は原則として小文字である。しかしながら、アプリケーションプログラムはこれらの属性について大文字と小文字の区別をすべきでない。いくつかの文字列属性はこの標準では「空白で区切られたリスト」を含むと定義されている。そのようなリストでは語は一つまたはそれ以上の連続したスペース[24]で区切られる。このリストたる属性値の始め又は終わりに任意の数のスペースがあってもよい。個の標準で記述されている属性のリストは付録 A「属性」を参照せよ。

2.6.1 規約の識別

この規約に従う netCDF ファイルはそのことを、NUG で定義された大域属性[25] Conventions を文字列値 とすることで表現することを勧告する。この文字列は、分野に特有の規約を記述する文書を寄託するディレクトリに対するディレクトリ名と解釈される。現在のところ[26]その規約ディレクトリはホスト ftp.unidata.ucar.edu の pub/netcdf/Conventions ディレクトリと解釈される。この文書のウェブ版は netCDF ウェブページ <URL:http://www.unidata.ucar.edu/packages/netcdf/conventions.html>[27] からリンクされている。

れるのであろう。

[24]訳注：この文書ではスペース(space)と空白(blank, whitespace)を同義に通用している。通常の情報工学的文脈では、スペースは ASCII [ASCII]符号位置2/0（スペースキーを押して打ち込まれる文字）だけを指し、空白はタブや改行などを含むのが通例であるが、この文書ではその区別は明瞭でない。

[25]訳注：netCDF の大域属性 global attribute とは変数ではなくデータセットにつけられた属性のこと。

[26]訳注：匿名 FTP ftp://ftp.unidata.ucar.edu/pub/netcdf/Conventions/ が正規の規約寄託先であったのはおそらく CF 規約開発のごく初期。2009年8月11日現在、同サイトは残っているが更新されておらず、本文で書かれているのとも微妙に異なる http://www.unidata.ucar.edu/software/netcdf/conventions.html が正規寄託先とされている。

[27]訳注：2009年8月11日現在、アクセスすると前注の正規寄託先にリダイレクトされる。

2.6.2 ファイルの内容の記述

以下の属性は、データがどこから来て、何を施したのかに関する情報を提供することを意図している。この情報は主に人間の読者のためのものである。属性値は全て文字列である。ツール ncdump の出力の可読性のため、長い文字列には改行を入れて複数行に分割することを勧告する。COARDS への後方互換性のため、必須とするものはない。

NUG は `title` 及び `history` を大域属性と定義している。我々は新しく定義した属性すなわち `institution`, `source`, `references`, 及び `comment` を大域属性及び個々の変数の属性のどちらとしても認めることにしたい。同じ属性名が大域属性と変数属性として現れるとき、変数属性が優先する。

`title`
: データセットが何かであるかの簡潔な記述。

`institution`
: オリジナルデータがどこで作られたかを指定する。

`source`
: オリジナルデータを作成した方法。もしモデルで生成されたものなら、`source` はモデルと有益な限り詳しいバージョンの名前であるべきである。観測データならば、`source` はそれを特徴付けるものであるべきである(例: `"surface observation"` 又は `"radiosonde"`)。

`history`
: オリジナルデータへの変更の履歴を提供する。お行儀のよい netCDF フィルタは自動的にその名前と起動時に与えられたパラメタを、入力 netCDF ファイルの大域属性 `history` に付加する。我々は、それぞれの行をプログラムが起動された日時を示すタイムスタンプではじめることを勧告する。

`references`
: データまたはデータを生成するために使った方法を記述するための、出版物またはウェブに基づく参照。

`comment`
: データまたはデータを生成するために使った方法に関する各種情報。

第3章　　データの記述

この節で記述される属性は、それぞれの変数の内容と単位を記述するために用意されている。COARDS で定義されている `units` 及び `long_name` 属性は我々も引き続きサポートする。COARDS への拡張として、変数の内容への一意な識別子を提供する、任意指定の `standard_name` 属性を追加した。これはデータ交換にとって重要である;データ提供元機関が割り当てた変数名によってそれが何を意味するかを識別することは必ずしも可能ではないからである。

属性 `standard_name` は座標データを含む変数を識別するのにも用いられる。しかしこれは任意指定であるため、この標準を実装するアプリケーションは引き続き、COARDS 規約に基づく座標種別の識別ができなければならない。

3.1　単位

属性 units は次元量を表わす全ての変数に必須である(ただし7.1「セル境界」で定義される境界変数と、7.4「気候学的統計」で定義される気候変数を除く)。属性 `units` の値は文字列で、Unidata の UDUNITS パッケージ[UDUNITS]で認識できるものである;ただし以下で示す例外が認められる。UDUNITS パッケージに含まれるファイル `udunits.dat` にサポートされる単位名が

一覧されている。文字列 units の中では大文字と小文字は区別されることに留意せよ。

COARDS 規約は経度と緯度をそれぞれ単位 degrees_east 及び degrees_north で表現させるために単位 degrees を禁止しているが、この CF 規約では禁止しない;たとえば太陽天頂角などの変数に適切だからである。単位 degrees は（回転）変換された経緯度のような座標変数にも認められる。この場合座標値は真の経緯度でないので、常に4.1「緯度座標」及び4.2「経度座標」で記述するより特定的な形式で識別されなければならない。

単位は無次元量には必須ではない。属性 units をもたない変数は無次元量と仮定される。しかしながら、無次元単位を指定する任意指定の units 属性を含めることができる。UDUNITS パッケージはいくつかの無次元の単位、たとえば percent を定義しているが、一般的に用いられる単位たとえば ppm が欠けている。この規約では UDUNITS に互換ではない新しい無次元単位の追加はサポートしない。割合を表わす量の、この規約に適合する単位は "1" である。百万分率を表わす量の、この規約に適合する単位は "1e-6" である。無次元の量、たとえば海氷密度、雲量、あるいは確率などに関する記述的な情報は、units よりはむしろ（下記） long_name あるいは standard_name 属性で与えられるべきである。

COARDS との後方互換性を維持するために、鉛直の無次元座標の単位として level, layer, そして sigma_level が許容される。これらの単位は UDUNITS と互換ではなく、鉛直無次元座標をより精密に識別する規約（4.3.2「無次元鉛直座標」参照）が導入されているので、この標準では廃止予定とする。

UDUNITS の文法のうち、単位にスケールファクター及びオフセットを適用する機能はこの標準ではサポートされない。データにスケールファクターやオフセットを適用する場合は、属性 scale_factor および add_offset で示すべきである。これらの属性の最も重要な応用であるデータ・パッキングの用例は8.1「パックされたデータ」で論じる。

UDUNITS は以下の接頭辞と記号を認識する。

表3.1 サポートされる単位

係数	接頭辞	記号	係数	接頭辞	記号
1e1	deca, deka	da	1e-1	deci	d
1e2	hecto	h	1e-2	centi	c
1e3	kilo	k	1e-3	milli	m
1e6	mega	M	1e-6	micro	u[28]
1e9	giga	G	1e-9	nano	n
1e12	tera	T	1e-12	pico	p
1e15	peta	P	1e-15	femto	f
1e18	exa	E	1e-18	atto	a
1e21	zetta	Z	1e-21	zepto	z
1e24	yotta	Y	1e-24	yocto	y

[28]訳注：元来 SI では接頭辞マイクロの記号はμ（ギリシャ文字のミュー）だが、ASCII[ASCII]文字集合に含まれないので UDUNITS では u で代用することとなっている。この代用は元をただせば旧 ISO 2955[ISO2955]に由来する。

3.2 長い名前

NUG で定義されている属性 long_name は変数の長い記述的な名前であり、たとえば描画の際のラベルに用いられうる。COARDS との後方互換性のためこの属性は任意指定とする。しかしファイルを自己記述的にするため、この long_name または次節で定義する standard_name の少なくともいずれかを提供することを強く勧める。もしある変数に long_name 属性がなければ、アプリケーションは既定値として、もしあれば standard_name を、あるいは変数名を使うことができる。

3.3 標準名

物理量を精密に記述する能力は、科学的データの交換における根本的な要請である。NUG が long_name を定義した際にはある程度この役割を想定していた。しかしながら、long_name の利用実態は完全にその場限りのものである。ある種のアプリケーションにとっては、物理量のより明確な記述を得て、異なる提供元からのデータのユーザが、物理量同士が比較可能であるかを決定できるようにすることが望ましい。このため、それぞれの変数に標準的な名前を一意に割り当てる任意指定の機構が用意されている。

変数に標準名を関連付けるのは standard_name 属性で、その文字列値は標準名と、1つ以上の空白で区切られた標準名修飾子（付録 C「標準名修飾子」に掲げる文字列）からなる。

標準名として指定できる名前は標準名表に挙げられている。それぞれの標準名についての表のエントリは次のものからなる:

標準名
: 物理量を識別するための名前。標準名は空白を含まず、大文字と小文字は区別される。

正規の単位
: 物理量の代表的な単位。無次元量でなければ、属性 standard_name を持つ変数は世紀の単位と同等の（必ずしも同一でなくてもよい）単位をもたねばならない。単位は標準名修飾子（付録 C「標準名修飾子」）、あるいは cell_methods 属性（付録 E「セルメソッド」）で指定される操作で変更されうる。

記述
: 基礎的な量の限定子[29]を明確化するための記述。たとえばある量がどの面で定義されるか、あるいはフラックスの符号の慣例はどうであるかなど。我々は基礎的な物理量（たとえば温度）について、文献を見ればあるような精密な定義を与えようとしているわけではない。

適切ならば、表のエントリは GRIB パラメータ番号（ECMWF および NCEP）[30]と AMIP[31]識別子を含む。

標準名表は <URL:http://cf-pcmdi.llnl.gov/documents/cf-standard-names/standard-name-table/current/cf-standard-name-table.xml> におかれており、付録 B「標準名表の形式」に記述される XML 形式で書かれている。XML 形式の知識が必要なのは、この表を直接アクセスしようとする

[29]訳注：限定子は原文 qualifier. 他に modifer を修飾子と訳しているので訳し分けた。本文では明示していないが、標準名には「大気の温度」「海水の流速」など、物理量を担う物体が限定されるのが常であり、qualifier はその限定部分であると解される。

[30]訳注：ここで GRIB とは GRIB 第1版（FM 92–XI Ext. GRIB Edition 1）を指している。GRIB 第1版のパラメータ番号は番号1～127が国際的に共通、番号128～254が地域的使用のために作成機関で割り当てるとされており、ここで ECMWF 及び NCEP というのはそれぞれの機関で定義した部分のことである。

[31]訳注：Atmospheric Model Intercomparison Project <URL:http://www-pcmdi.llnl.gov/projects/amip/>.

アプリケーション作者だけである。テキストとして整形したものは<URL:http://cf-pcmdi.llnl.gov/documents/cf-standard-names/standard-name-table/current/standard-name-table> に用意されているので、変数に割り当てるべき標準名を見出すために参照できる。いくつかの標準名（たとえば region や area_type）は、特定の標準化された値だけをとることが許されている量を示すのに用いられる。この制約は、標準名表の量の定義において、許される値のリストあるいはリストへのリンクとして示される。

標準名はそれだけでは物理量を記述するのに十分とは限らない。たとえば、ある変数は空間あるいは時間に関する処理が行われたデータを含むかもしれない。あるいはそのデータは物理量の測定に関する不確定性を表わすのかもしれない。これらの量に関する属性は標準名の修飾子として表現される。一般的な統計操作による修飾は cell_methods 属性によって表現される（7.3「セル・メソッド」及び付録 E「セルメソッド」参照）。他の種類の物理量修飾子は standard_name 属性のうちの任意指定の修飾子部分を使って表現される。修飾子として認められる値は付録 C「標準名修飾子」に与えられている。

例3.1. standard_name の使用

```
float psl(lat,lon) ;
    psl:long_name = "mean sea level pressure" ;
    psl:units = "hPa" ;
    psl:standard_name = "air_pressure_at_sea_level" ;
```

標準名表の「air_pressure_at_sea_level」の項によって、「海面」とは平均海面を指し、それは海上ではジオイドに近似することが明らかにされている。

CF の標準名と ECMWF GRIB 表[32]、NCEP GRIB 表[33]、PCMDI の表[34]の標準名の相当関係リストが提供されている。

3.4 補助データ

ある変数が他のデータ変数の個々の値に関するメタデータを提供するとき、変数の間に結びつきを設けられるようにしてこの関連を表現することが望ましいことがある。たとえば、測器データは測定の不確実性を伴うことがある。この種の関係は ancillary_variables 属性を使って表現される。この属性は文字列でその値は空白で区切られた変数名のリストである。属性 ancillary_variables を通じて関連付けられた変数の関係の性質は他の属性で決められねばならない。しばしば、属性 ancillary_variables によって列挙された変数は、この属性のついている変数の標準名に修飾子（付録 C「標準名修飾子」）をつけた標準名を持つことによってその関係を示す。

例3.2. 測器データ

```
float q(time) ;
    q:standard_name = "specific_humidity" ;
    q:units = "g/g" ;
    q:ancillary_variables = "q_error_limit q_detection_limit" ;
float q_error_limit(time) ;
    q_error_limit:standard_name =
       "specific_humidity standard_error" ;
    q_error_limit:units = "g/g" ;
  float q_detection_limit(time) ;
```

[32]<URL:http://cf-pcmdi.llnl.gov/documents/cf-standard-names/ecmwf-grib-mapping>
[33]<URL:http://cf-pcmdi.llnl.gov/documents/cf-standard-names/ncep-grib-code-cf-standard-name-mapping>
[34]<URL:http://cf-pcmdi.llnl.gov/documents/cf-standard-names/pcmdi-name-cf-standard-name-mapping>

```
    q_detection_limit:standard_name =
       "specific_humidity detection_minimum" ;
    q_detection_limit:units = "g/g" ;
```

3.5 フラグ

属性 flag_values, flag_masks 及び flag_meanings はフラグ値を含む変数を自己記述的にすることを意図したものである。状態コード及びブール(二値)条件フラグ[35]はそれぞれ flag_values 属性及び flag_masks 属性による定義とのさまざまな組み合わせによって表現される。

属性 flag_values 及び flag_meanings は相互に排他的なコード値からなる状態フラグ[36]を記述する。属性 flag_values はその付属する変数と同じ型であり、可能なフラグ値のリストを含んでいる。属性 flag_meanings は文字列でその値は空白で区切られた語句からなり、それぞれのフラグ値に対応する。もしフラグ値の記述に複数の語からなる句を使うときは、句中の語はアンダースコアで連結する。次は、流速の測定品質を列挙的な状態コードで表現するのにフラグ値を使う例を示している。

例3.3. 属性 flag_values を使ったフラグ変数

```
byte current_speed_qc(time, depth, lat, lon) ;
    current_speed_qc:long_name =
       "Current Speed Quality" ;
    current_speed_qc:standard_name =
       "sea_water_speed status_flag" ;
    current_speed_qc:_FillValue = -128b ;
    current_speed_qc:valid_range = 0b, 2b-127b, 127b ;
    current_speed_qc:flag_values = 0b, 1b, 2b ;
    current_speed_qc:flag_meanings =
       "quality_good sensor_nonfunctional outside_valid_range" ;
```

属性 flag_masks 及び flag_meanings はそれぞれの flag_masks の値に一意のビットを設定するビットフィールド記法を使って、独立なブール条件[37]を記述する。属性 flag_masks はその付属する変数と同じ型であり、一意のビットフィールドに対応する値のリストを含む。属性 flag_meanings は上述のように定義され、値はそれぞれ flag_masks の値に対応する。フラグの条件は変数値とそれぞれの flag_masks の値とについてビット毎 AND 演算を行うことで得られ、非零の結果が真条件を示す。したがって、変数のビット配置次第では、フラグ条件の全部または一部が真となり得る。次はセンサーの6つの状態条件を表現する flag_masks の用例を示している。

例3.4. 属性 flag_masks を使ったフラグ変数

```
byte sensor_status_qc(time, depth, lat, lon) ;
    sensor_status_qc:long_name = "Sensor Status" ;
```

[35]訳注：この文書では condition「条件」、status「状態」、あるいは status condition は同義に通用しており、真偽値が判定されうる語句、というような意味と考えられる。また、「コード」と「フラグ」も通用しており、条件・状態に対応する名義尺度の数値で、整数型データの一部のビットを使った小さな整数でこれら条件・状態を表現したもの、というような意味と考えられる。

[36]訳注：前注のように本文では用語を無駄に使っていて難解だが、属性 flag_values で指定されるのは列挙型（WMO の通報式でいえばコード表）で、flag_meanings の語句のうちどれかひとつだけが「排他的に」成り立つ。

[37]訳注：同様に、属性 flag_masks で指定されるのはビットフィールド（通報式でいえばフラグ表）であり、異なるビットはそれぞれ「独立」に真偽値をとりうる。

```
    sensor_status_qc:_FillValue = 0b ;
    sensor_status_qc:valid_range = 1b, 63b ;
    sensor_status_qc:flag_masks = 1b, 2b, 4b, 8b, 16b, 32b ;
    sensor_status_qc:flag_meanings = "low_battery processor_fault
                                      memory_fault disk_fault
                                      software_fault
                                      maintenance_required" ;
```

属性 flag_masks, flag_values 及び flag_meanings を同時に使うと、独立なブール条件と列挙的な状態コードとの組み合わせが記述される。属性 flag_masks と flag_values はそれが付属する変数と同じ型である。フラグの条件は変数値とそれぞれの flag_masks 値のビット毎 AND 演算を行うことにより決まる;flag_values 値と対応する結果が真の条件を示す。属性 flag_masks での同じ値の繰り返しは、異なるv値を持つ多数の状態条件を識別するビットフィールドマスクを定義する。属性 flag_meanings は上述のように定義され、それぞれ flag_masks のビットフィールドと flag_values の値定義に対応する。それぞれの flag_values と flag_masks の値は flag_meanings 値と対応しなければならない。以下は、flag_masks と flag_values を使って二つの状態条件と一つの列挙的状態コードのあるセンサーの用例を示す。

例3.5. 属性 flag_masks と flag_values を使ったフラグ変数

```
byte sensor_status_qc(time, depth, lat, lon) ;
    sensor_status_qc:long_name = "Sensor Status" ;
    sensor_status_qc:_FillValue = 0b ;
    sensor_status_qc:valid_range = 1b, 15b ;
    sensor_status_qc:flag_masks = 1b, 2b, 12b, 12b, 12b ;
    sensor_status_qc:flag_values = 1b, 2b, 4b, 8b, 12b ;
    sensor_status_qc:flag_meanings =
         "low_battery
          hardware_fault
          offline_mode calibration_mode maintenance_mode" ;
```

この場合、相互に排他的な値とブール値とを組み合わせることで、フラグ値で利用できるビット数が最大限に活用されている。以下の表は上の例における sensor_status_qc 変数で表現されている4つの二進桁(ビット)を表わしている。

ビット0とビット1はブール値であり、それぞれバッテリー電圧低下条件とハードウェア障害を示している。次の二つのビット(ビット2とビット3)は列挙型で、正常以外のセンサー運用モードを示している。したがって、もしビット0が立っていればバッテリー電圧が低下しており、もしビット1が立っていればハードウェア障害がある;いずれも現在のセンサー運用モードとは無関係にである。

表3.2. フラグ変数のビット(上の例より)

ビット3(MSB)	ビット2	ビット1	ビット0(LSB)
		ハードウェア障害	バッテリー低電圧

残りのビット(ビット2及びビット3)は次のようにデコードされる:

表3.3. フラグ変数のビット2及びビット3(上の例より)

ビット3	ビット2	モード
0	1	オフライン
1	0	較正モード

ビット3	ビット2	モード
1	1	保守モード

変数 sensor_status_qc の属性 flag_masks の値の中で、フラグマスク "12b" が繰り返されていることは、列挙された値から対応するものを探すために、変数値にこのビットフィールドマスクを繰り返し論理 AND することが勧告されることを明示的に宣言している。アプリケーションは、flag_meanings 属性のリストで宣言された条件のどれが真になっているかを決めるために、単純に flag_masks 属性のそれぞれの値について繰り返し変数値とビット毎 AND を行えばよく、ビットフィールドとコード表のどちらが使われているかを区別する必要がない。対応する flag_values の要素と同じ値が得られれば、条件は真である。属性値 flag_masks を繰り返すことにより、クライアントは単純な機構ですべてのありうる条件を検出することが可能になる。

第4章　座標の種別

この規約では4種類の座標が特別な扱いを受ける:緯度、経度、鉛直、そして時間である。COARDS 規約において座標の種類を識別するための、units 及び positive 属性の特別な役割を、我々は引き続きサポートする。COARDS への拡張として、無次元鉛直座標の明示的定義を用意した。この定義は座標変数に standard_name 及び formula_terms 属性を使って関連づけられる。COARDS との後方互換性のため、これらの属性の使用は必須ではないが、強く勧められる。

単位による座標の種類の識別は外部ソフトウェアパッケージ[UDUNITS]を使う必要があり複雑なので、我々は直接識別をしてくれる二つの任意指定の方法を提供する。座標変数に属性 axis を付加し、それぞれ経度、緯度、高度、あるいは時間軸を表わす値 X, Y, Z, あるいは T のいずれかを与えることができる。もう一つの方法として、standard_name 属性を直接識別に使うことができる。しかしこれらの任意指定の属性は COARDS メタデータで必須とされているものの外であることに留意せよ。

緯度、経度、鉛直、あるいは時間以外の座標種別は認められている。一般の空間座標を識別するために、我々はそのような座標に axis 属性を付加し、X, Y, または Z のいずれかの値を与えることを勧告する。属性 axis の値 X 及び Y は水平の座標変数を識別するのに用いられるべきである。もし X と Y 軸の両方が識別されれば、もう一つの軸を *up* として、X-Y-*up* の3軸が右手系を定義すべきである;すなわち X の正の方向から Y の正の方向への回転は、上（up の正の方向）からみて反時計回りとなるべきである。全ての種類の座標について、適切ならば座標変数を使用することを強く勧める。

この節で記述される座標の種類の識別方法は、座標変数と、coordinates 属性によって指名される補助座標変数（5「座標系」参照）のどちらについても適用される。

座標変数あるいは補助座標変数の値は、格子点の位置を示す。セルの間の境界の位置は境界変数（7.1「セル境界」参照）で示される。もし境界が与えられていなければ、アプリケーションは格子点がセルの中央にあると仮定するかもしれないが、我々はそれを標準として要請とはしない。

4.1　緯度座標

緯度を表わす変数[38]は、units 属性を常に明示的に持たねばならない;既定値は存在しない。

[38]訳注：緯度・経度・時間については座標変数ではなく単に変数に関する規定になっている（どういうわけか鉛直座標だけは「軸を表わす変数」としている）が、従属変数として緯度・経度・時間を表現する場

属性 units は udunits.dat ファイルで書式化されるような文字列である。緯度の単位としては degrees_north を使うことを勧告する。代わりに degree_north, degree_N, degrees_N, degreeN, あるいは degreesN も許容される。

例4.1. 緯度軸

```
float lat(lat) ;
  lat:long_name = "latitude" ;
  lat:units = "degrees_north" ;
  lat:standard_name = "latitude" ;
```

アプリケーション作者は、単位名に “north” などの語が含まれていても UDUNITS パッケージがそれを認識するわけではないことに留意すべきである。認識されるのは大きさ、すなわち、1度は pi/180 radians であることだけである。ゆえに、座標が緯度であることを決めるにあたっては、与えられた単位が degrees_north の許容された形式のいずれかと文字列比較することでなされるべきである。

必須ではないが、属性 standard_name に値 latitude を与えるか、あるいは axis 属性に値 Y を与えることによって緯度であることを示すことができる。

回転された極に対する緯度の座標は、単位を degrees_north あるいは同等物ではなく degrees とすべきである;さもなくば軸の識別に単位を使うアプリケーションがこの軸と真の緯度を区別する方法がなくなり、たとえば誤った海岸線を描画するだろうからである。

4.2 経度座標

経度を表わす変数は、units 属性を常に明示的に持たねばならない;既定値は存在しない。属性 units は udunits.dat ファイルで書式化されるような文字列である。緯度の単位としては degrees_east を使うことを勧告する。代わりに degree_east, degree_E, degrees_E, degreeE, あるいは degreesE も許容される。

例4.2. 経度軸

```
float lon(lon) ;
  lon:long_name = "longitude" ;
  lon:units = "degrees_east" ;
  lon:standard_name = "longitude" ;
```

アプリケーション作者は、単位名に “east” などの語が含まれていても UDUNITS パッケージがそれを認識するわけではないことに留意すべきである。認識されるのは大きさ、すなわち、1度は pi/180 radians であることだけである。ゆえに、座標が経度であることを決めるにあたっては、与えられた単位が degrees_east[39] の許容された形式のいずれかと文字列比較することでなされるべきである。

必須ではないが、属性 standard_name に値 longitude を与えるか、あるいは axis 属性に値 X を与えることによって経度であることを示すことができる。

回転された極に対する経度の座標は、単位を degrees_east あるいは同等物ではなく degrees とすべきである;さもなくば軸の識別に単位を使うアプリケーションがこの軸と真の経度を区別する方法がなくなり、たとえば誤った海岸線を描画するだろうからである。

合に適用することを意識して書き分けているのか不明。

[39]訳注：原文では degrees_north とあるが明らかに誤り。

4.3 鉛直(高度または深さ)座標

次元のある高度または深さの軸を表わす変数は、常に明示的に units 属性を持たねばならない;既定値は存在しない。

正の方向(すなわち座標値が増えてゆく方向)が上向きであるか下向きであるかは、常に単位から推論できるわけではない。データを表示するアプリケーションにとって正の方向の情報は有益である。この理由から、鉛直軸の単位が圧力の有効な単位でない(この判定は udunits のルーチン utScan によって行われる)場合、COARDS で定義された positive 属性は必須である — そうでない場合は任意指定である。属性 positive は値 up または down をとりうる(大文字と小文字は区別しない)。この属性は座標変数と、鉛直座法データを含む補助座標変数とのどちらについても適用できる。

たとえば、海洋に関する netCDF ファイルが海面の深さを0と、深さ1000メートルを1000とエンコードするなら、その軸は次のように属性を使うことができる:

```
axis_name:units = "meters" ;
axis_name:positive = "down" ;
```

もし逆に、深さ1000メートルが −1000 で表現されるならば、属性 positive は up とすることになろう。もし units 属性の値が有効な圧力の単位ならば、positive 属性の既定値は down である。

座標が鉛直であることを識別するには次のものが使える。

- 単位が圧力である
- 属性 positive が値 up または down をもつ(大文字と小文字は区別しない)

必須ではないが、鉛直軸をあらわす座標変数であることは standard_name 属性に適切な値を指定するか、属性 axis に値 Z を指定するか、あるいはその両方[40]で示すことができる。

4.3.1 有次元鉛直座標

次元のある座標の units **属性**は udunits.dat ファイルで書式化されるような文字列である。鉛直(深さまたは高さ)座標変数として受け入れられる[41]単位は次のものである:

- ファイル udunits.dat で列挙されている圧力の単位。鉛直軸に一般的に用いられるものには bar, millibar, decibar, atmosphere (atm), pascal (Pa), そして hPa などがある。
- ファイル udunits.dat で列挙されている長さの単位。鉛直軸に一般的に用いられるものには meter (metre, m), そして kilometer (km) などがある。
- ファイル udunits.dat で列挙されているその他の単位(たとえば密度や温度)も、一定の条件の下では鉛直方向の位置を参照することができる。

単位名を記号ではなくフルスペルした場合は複数形も受け入れられる。

4.3.2 無次元鉛直座標

無次元の座標では units 属性は必須ではない。COARDS との広報互換性のため、属性

[40]訳注:「A and/or B」は 「A, B, あるいはその両方」と訳した。
[41]訳注:受け入れる条件も効果も不明確。察するに、データ作成者へのガイドラインと、アプリケーション作者があまりに狭い決め打ちを行わないように警告することが目的であろう。

units が値 level, layer, あるいは sigma_level を取ることは引き続き許容される。これらの値は UDUNITS パッケージでは認識されないため、CF 標準では廃止予定の特徴とみなされる。

無次元の鉛直座標について、COARDS 標準への拡張として、属性 standard_name を使って座標と付録 D「無次元鉛直座標」の座標定義を結びつけることができるようにした。この定義により、無次元座標値と、データの位置を積極的かつ一意に示すことができる次元量とを結びつける写像が提供される。定義の各項と netCDF ファイルの変数とを関連付けるために、新しい属性 formula_terms が用いられる。COARDS との広報互換性を維持するために、これらの属性は必須とはしないが、強く勧められる。

例4.3.大気のシグマ座標

```
float lev(lev) ;
  lev:long_name = "sigma at layer midpoints" ;
  lev:positive = "down" ;
  lev:standard_name = "atmosphere_sigma_coordinate" ;
  lev:formula_terms = "sigma: lev ps: PS ptop: PTOP" ;
```

この例の standard_name の値 atmosphere_sigma_coordinate は付録 D「無次元鉛直座標」から得られた次の定義を示している。

```
p(n, k, j, i) = ptop + sigma(k)*(ps(n, j, i) - ptop)
```

これは格子点 (*n, k, j, i*) での気圧（*i, j* が水平インデックス、*k* が鉛直インデックス、*n* は時間インデックス）を算出する方法を指定するものである。

属性 formula_terms は変数 lev を項 sigma に、変数 PS を項 ps に、そして変数 PTOP を項 ptop に関連付ける。したがって格子点 (*n, k, j, i*) での気圧は次のように計算できよう。

```
p(n,k,j,i) = PTOP + lev(k)*(PS(n,j,i)-PTOP)
```

4.4 時間座標

時間を表わす変数は明示的に units 属性を持たねばならない;既定値は存在しない。属性 units は Udunits [UDUNITS]パッケージで勧告されたように書式化された文字列値を取る。時間の単位の表現法について例示する Udunits のドキュメンテーションを次に引用する。

```
規定：

    seconds since 1992-10-8 15:15:42.5 -6:00

は協定世界時から西に6時間離れた時間帯（米山岳標準時の夏時間）での
1992年10月8日午後3時15分42.5秒からの経過秒数を示す。時間帯の表記はコロンなしの
1～2桁（時数を表わす）または3～4桁（時分を表わす）で書くこともできる。
```

時間の単位として許容されるものはファイル udunits.dat に列挙されている。最も一般的に使われている文字列（とその記号）には day (d), hour (hr, h), minute (min), そして second (sec, s) がある。複数形も許容される。（識別子[42]since の後に現れる）参照時刻文字列は「日付だけ」「日付と時刻」あるいは「日付、時刻、及び時間帯」を含むことができる。参照時刻は必須

[42]訳注：原文 identifier なので識別子と訳したが、構文規則上 since という単位が認められなくなっているので予約語といったほうがよい。なお、UDUNITS は since のかわりに @, after, from, または ref という表記も許しているが、扱いは不明。データ作成者は念のため避けたほうがよかろう。

である。西暦0年の参照時刻は特別な意味を持つ(7.4「気候学的統計」参照)。

注意：もし時間帯が省略された場合は UTC が既定値であり、時刻と時間帯の両方が省略されればが 00:00:00 UTC が既定値である。

単位 year には注意を持って使うことをお勧めする[43]。Udunits パッケージは year を 365.242198781 日と定義している。これは太陽が春分点を二回通過する間の時間であり、***暦年とは異なる***。Udunits は次のような年の定義をもっている: 平年[44] common_year は 365 日、閏年 leap_year は 366 日、ユリウス暦の平均 Julian_year は 365.25 日、そしてグレゴリオ暦の平均 Gregorian_year は 365.2425 日である。毎年初のデータというような、1暦年ずつ進んでいく時間軸を作るには、座標変数値に日数を計算して入れていくほかない。

同様な理由で単位 month も udunits.dat では year/12 と定義されていることから、注意を持って使われるべきである。

例4.4. 時間軸

```
double time(time) ;
  time:long_name = "time" ;
  time:units = "days since 1990-1-1 0:0:0" ;
```

時間軸は単位文字列だけで識別できる。Udunits の utScan() 及び utIsTime() を使ってその決定が行える。

必須ではないが、時間軸であることは属性 standard_name に適切な値を与えるか、属性 axis に値 T を与えること、あるいはその両方で示すことができる。

4.4.1 暦法

与えられた基準日付、基準時刻および時間増分から新しい日時を算出するためにはどの暦法が使われるかを知らなければならない。この目的のため、時間の座標変数に calendar 属性を付与して暦法を特定することを勧告する。属性 calendar の値として現在次のものが定義されている:

gregorian **または** standard
: Udunits で定義される混合ユリウス・グレゴリオ暦。***これが既定値である***。

proleptic_gregorian
: グレゴリオ暦を 1582-10-15 [45]以前に延長したもの。すなわち、ある年はその年番号[46]が (i) 4 で割り切れて100で割り切れない場合、あるいは (ii) 400で割り切れる場合に閏年となる。

noleap **または** 365_day
: 閏年のないグレゴリオ暦、すなわちすべての年が365年の長さである。

all_leap **または** 366_day
: すべての年が閏年のグレゴリオ暦、すなわちすべての年が366日の長さである。

360_day
: すべての年が360年であり、1ヶ月は30日からなる暦。

---

[43]訳注：注意が払われたかどうかは外形的に適合性を判定できないので、このような規定は勧告ではなく should で表わされるのが普通だと思うが、そもそも冒頭で勧告は助言だと言ってしまっているのでそれでもいいのだろう。

[44]訳注：単に平年というときは閏年でない年、平年値・平年比などというときは気候値のことである。

[45]訳注：グレゴリオ暦がカトリック諸国によって始めて行用された日。

[46]訳注：ISO 8601による遡及グレゴリオ暦では、紀元１年の前年を0年、その前年を－1年のように数値を連続させて年を数える。これにより閏年判定のロジックはそのまま使えることになる。

julian
: ユリウス暦。

none
: 暦法なし。

一年の中のある季節を固定してシミュレートする気候の実験においては、属性 calendar を none にすることができる。季節は units 属性の参照時刻の中の日付で示される。次の例は恒常的7月の実験での時間座標を与える。

例4.5. 恒常的時間軸

```
variables:
  double time(time) ;
    time:long_name = "time" ;
    time:units = "days since 1-7-15 0:0:0" ;
    time:calendar = "none" ;
data:
  time = 0., 1., 2., ...;
```

ここでは、すべての日が7月15日の条件をシミュレートするものであり、ゆえに異なる日付を与えることには意味がない。時間座標は実験開始から0, 1, 2, ... 日と解釈するものである。

もし上で定義された暦法のどれもあてはまらない場合（たとえば異なる古気候的時代に対応する暦など）、非標準的な暦法を定義できる。それぞれの月の長さは時間軸の month_lengths 属性を使って明示的に定義される:

month_lengths
: 大きさ12のベクタで、非閏年の各月の日数を指定する。

閏年があるならば、時間軸に他に2つの属性を定義すべきである:

leap_year
: 閏年の例。この年と4の倍数だけ異なる年はすべて閏年であると仮定される。この属性が欠けていれば、閏年はないものと仮定される。

leap_month
: 整数1–12の値をとり、どの月が閏年に1日延長されるかを指定する。この属性が欠けていれば、2が仮定される。この属性は leap_year が指定されていなければ無視される。

非標準の暦法が使われているときは、属性 calendar は必須ではない。属性 month_lengths および、適宜 leap_year および leap_month 属性で暦法を定義すれば十分である。しかしながら、属性 calendar に非標準の値を与えることは認められ、この場合適切な属性で暦法を定義することが必須となる。

例4.6. 古気候の時間軸

```
double time(time) ;
 time:long_name = "time" ;
 time:units = "days since 1-1-1 0:0:0" ;
 time:calendar = "126 kyr B.P." ;
 time:month_lengths = 34, 31, 32, 30, 29, 27, 28, 28, 28, 32, 32,
34 ;
```

Udunits で用いられている混合ユリウス・グレゴリオ暦は、udunits(3)マニュアルページにより次のように説明されている:

> この udunits(3)パッケージでは混合ユリウス・グレゴリオ暦法が用いられている。1582-10-15より前の日付はユリウス暦とみなされる；それはユリウス・カエサルによって紀元前46年に導入された暦法で、長さ365.25日と定義される1年に基づく。1582-10-15およびそれ以降の日付はグレゴリオ暦を使うものと仮定される；それはその日に導入された暦法で、長さ365.2425日と定義される1年に基づく。(実際の1年は近似的に365.242198781日の長さである)。もしユーザが与えた時間範囲が切り替え日を含むならば、udunits(3)パッケージは一見奇妙な挙動をしうる。たとえば、utCalendar()及びutInvCalendar()を使うことにより1582-10-15は1582-10-14より9日も ***先行している*** ことを示すことができる。

既定の混合ユリウス・グレゴリオ暦の不連続性によっておきる問題のため、この暦法は時間座標が不連続をまたがない場合に限って使われるべきであることを我々は強く勧告する。不連続をまたぐ時間座標については代わりに proleptic_gregorian 暦法が用いられるべきである。

第5章　　座標系

変数の時空間次元を使って、データ値は時空間の中に位置づけられる。これらの次元を緯度、経度、鉛直及び時間の座標の適切な集合に関連付けることによりこの位置づけがなされる。この節では、この関連付けの2つの方法を示す: ***座標変数***と、***補助座標変数***である。

変数の次元のうち、緯度、経度、鉛直あるいは時間の次元であるもの(1.2「用語」参照)はすべて対応する次元変数、すなわち一次元変数でその次元と同じ名前を持つもの(第4章「座標の種別」参照)を持たねばならない。これは COARDS でサポートされる座標を次元と関連付ける唯一の方法である。

変数の時空間次元のうち、緯度、経度、鉛直、あるいは時間次元ではないものは、その変数の新しい coordinates 属性を通じて、対応する緯度、経度、鉛直、あるいは時間座標と関連付けられなければならない。属性 coordinates の値は***補助座標変数の名前からなる空白区切りのリスト***である。属性 coordinates の文字列の中に現れる補助座標変数の順序には制約はない。補助座標変数の次元は、その座標が関連付けられている変数の次元の部分集合でなければならない(例外は文字列の最大の長さを保持するラベル座標(6.1「ラベル」)である)。多次元の座標変数の名前は、その次元の名前とは一致しないものであるべきである;なぜなら座標変数を補うことをあらかじめ不可能にしてしまうからである。また、このような習慣によって、次元変数の決定にあたって次元名と変数名の対応だけを確かめ、その変数名が一次元であることを確かめないアプリケーションによる潜在的なバグを避けることができる。

座標変数の使用は、それが当てはまる場合は常に必須である。つまり、座標変数を使って経緯度が識別できて、補助座標変数は経緯度座標を識別する唯一の方法ではないかもしれない。これは、COARDS への適合性を高め、また座標変数について NUG 規約を認識する汎用のアプリケーションの利用を助けるためである。ある変数について経緯度を見出そうとするアプリケーションは、常にまずその変数の次元のどれかが緯度座標変数に対応するかを見るべきである。もしこの方法で緯度座標が見つからなければ、属性 coordinates で列挙されている補助座標変数を確かめるべきである。座標変数を補助座標変数とともに coordinates 属性に列挙することは許容されるが必須ではないことに留意せよ。

axis 属性が補助座標変数に付与されているならば、アプリケーションは座標変数に付与された axis 属性と同様の使い方ができる。しかし、データ変数が座標変数と補助座標変数の両方を持つことは許容されない。同様に、ある特定の値の axis 属性について1より多くの座標変数または補

助座標変数を持つことは許容されない。つまり、あるデータ変数の X 軸を表す axis 属性は0個または1個でなければならない。ただし、補助座標変数に関する axis 属性が指定されていない場合であっても、その補助座標変数の units または standard_name から、あるいはその補助座標変数の次元の units または standard_name から時空間の次元を決めることは可能である（第4章「座標の種別」参照）。たとえば、補助座標変数が水平面内に位置することは、その次元が水平であることで識別できる。水平の次元とはその座標変数が値 `X` または `Y` の `axis` 属性を持つか、あるいは緯度または経度であることを示す `units` 属性を持つ（第4章「座標の種別」参照）ものである。

もし、水平の格子のための座標変数が経度でも緯度でもないならば、必要な座標に***加えて***それらも付加することを勧告する。たとえば、地図投影上におけるデカルト座標の上に格子がおかれる場合は、必須の緯度及び経度の二次元変数（属性 `coordinates` を通じて識別される）に加えて、デカルト座標も座標変数として与えられるべきである。この座標変数には値 `X` および `Y` の `axis` 属性を使うことが勧告される（第4章「座標の種別」参照）。

時折、複雑な境界を持つ領域を代表するデータの位置を緯度経度によって指定することが現実的でない場合がある。この目的のために、領域を標準化された名前で示す方法が6.1.1節「地理的領域」に用意されている。

5.1 独立の緯度、経度、鉛直、及び時間の軸

ある変数の時空間次元のそれぞれが緯度、経度、鉛直、あるいは時間の次元であるとき、それぞれの軸は座標変数を使って識別される。

例5.1. 独立の座標変数

```
dimensions:
  lat = 18 ;
  lon = 36 ;
  pres = 15 ;
  time = 4 ;
variables:
  float xwind(time,pres,lat,lon) ;
    xwind:long_name = "zonal wind" ;
    xwind:units = "m/s" ;
  float lon(lon) ;
    lon:long_name = "longitude" ;
    lon:units = "degrees_east" ;
  float lat(lat) ;
    lat:long_name = "latitude" ;
    lat:units = "degrees_north" ;
  float pres(pres) ;
    pres:long_name = "pressure" ;
    pres:units = "hPa" ;
  double time(time) ;
    time:long_name = "time" ;
    time:units = "days since 1990-1-1 0:0:0" ;
```

格子点 `xwind`(n,k,j,i) の位置は座標値 `lon`(i), `lat`(j), `pres`(k), 及び `time`(n) と関連付けられる。

5.2 二次元の緯度及び経度の座標変数

緯度軸と経度軸の直積として定義されているわけではない水平の格子について、緯度及び経度座標は、二次元の座標変数で表現できる。これらの変数が座標であることは `coordinates` 属性を使って識別される。

例5.2. 二次元の座標変数

```
dimensions:
  xc = 128 ;
  yc = 64 ;
  lev = 18 ;
variables:
  float T(lev,yc,xc) ;
    T:long_name = "temperature" ;
    T:units = "K" ;
    T:coordinates = "lon lat" ;
  float xc(xc) ;
    xc:axis = "X" ;
    xc:long_name = "x-coordinate in Cartesian system" ;
    xc:units = "m" ;
  float yc(yc) ;
    yc:axis = "Y" ;
    yc:long_name = "y-coordinate in Cartesian system" ;
    yc:units = "m" ;
  float lev(lev) ;
    lev:long_name = "pressure level" ;
    lev:units = "hPa" ;
  float lon(yc,xc) ;
    lon:long_name = "longitude" ;
    lon:units = "degrees_east" ;
  float lat(yc,xc) ;
    lat:long_name = "latitude" ;
    lat:units = "degrees_north" ;
```

格子点値 T(*k*, *j*, *i*) は座標変数値 lon(*j*, *i*), lat(*j*, *i*), 及び lev(*k*) と関連付けられる。鉛直座標は座標変数 lev(lev) で表わされ、緯度及び経度の座標は属性 coordinates で識別される補助座標変数 lat(yc, xc) 及び lon(yc, xc) で表わされる。

次元 xc, yc にも座標変数が定義されていることに留意せよ。こうすることにより、多次元の緯度経度座標を認識しない汎用アプリケーションによってもこのデータの処理が容易になる。

5.3 間引き水平格子

「間引き」経緯度格子とは、緯度一定値の線（緯線）に沿って点が配置されるが、緯線に沿った点の数は極に向かって減っていくようなものである。このような格子の上のデータを二次元配列に格納することは空間を無駄にするし、二次元の座標変数に欠損値が現れることになる。この種の格子の上のデータは8.2「集約による圧縮」で記述する圧縮方式を使って格納することを勧告する。集約による圧縮では、アプリケーションが圧縮されたデータを二次元配列に容易に散開できるようにするためのインデックスを格納することにより構造が保存される。圧縮された緯度・経度の補助座標変数は coordinates 属性により識別される。

例5.3. 間引き水平格子

```
dimensions:
 londim = 128 ;
 latdim = 64 ;
 rgrid = 6144 ;
variables:
  float PS(rgrid) ;
    PS:long_name = "surface pressure" ;
    PS:units = "Pa" ;
    PS:coordinates = "lon lat" ;
  float lon(rgrid) ;
```

```
    lon:long_name = "longitude" ;
    lon:units = "degrees_east" ;
  float lat(rgrid) ;
    lat:long_name = "latitude" ;
    lat:units = "degrees_north" ;
  int rgrid(rgrid);
    rgrid:compress = "latdim londim";
```

格子点値 PS(*n*) は座標値 lon(*n*), lat(*n*) に関連付けられる。圧縮されたインデックス (*n*) に対しては二次元のインデックス (*j*, *i*) (C 言語の次元順序) が次のように割り当てられる[47]。

```
j = rgrid(n) / 128
i = rgrid(n) - 128*j
```

もしあるアプリケーションが compress 属性を認識しない場合であっても、属性 coorinates を認識するアプリケーションによって、この形式で保存された格子を依然として扱うことができることに注目せよ。

5.4 地点データの時系列

散在する地点におけるデータを表現するためには、ひとつの次元を測定地点を表わすために使うのが便利である。単一の空間的次元複数の独立の座標を関連付ける補助座標変数が用いられる。

例5.4. 地点データの時系列

```
dimensions:
  station = 10 ;  // measurement locations
  pressure = 11 ; // pressure levels
  time = UNLIMITED ;
variables:
  float humidity(time,pressure,station) ;
    humidity:long_name = "specific humidity" ;
    humidity:coordinates = "lat lon" ;
  double time(time) ;
    time:long_name = "time of measurement" ;
    time:units = "days since 1970-01-01 00:00:00" ;
  float lon(station) ;
    lon:long_name = "station longitude";
    lon:units = "degrees_east";
  float lat(station) ;
    lat:long_name = "station latitude" ;
    lat:units = "degrees_north" ;
  float pressure(pressure) ;
    pressure:long_name = "pressure" ;
    pressure:units = "hPa" ;
```

格子点値 humidity(*n*, *k*, *i*) は座標値 time(*n*), pressure(*k*), lat(*i*), および lon(*i*) と関連付けられる。

5.5 軌跡

航空機の経路に沿った観測の時空間位置を表現する一つの方法は、軌跡をパラメタ化するた

[47]訳注：式中の定数128はどうやって得たかというと、rgrid:compress 属性値を解読して得た最後の次元である lon の長さである。

めに時間を使って、空間上の位置を与えるために補助座標変数を使うことである。

例5.5. 軌跡

```
dimensions:
  time = 1000 ;
variables:
  float O3(time) ;
    O3:long_name = "ozone concentration" ;
    O3:units = "1e-9" ;
    O3:coordinates = "lon lat z" ;
  double time(time) ;
    time:long_name = "time" ;
    time:units = "days since 1970-01-01 00:00:00" ;
  float lon(time) ;
    lon:long_name = "longitude" ;
    lon:units = "degrees_east" ;
  float lat(time) ;
    lat:long_name = "latitude" ;
    lat:units = "degrees_north" ;
  float z(time) ;
    z:long_name = "height above mean sea level" ;
    z:units = "km" ;
    z:positive = "up" ;
```

格子点値 O3(*n*) は座標値 time(*n*), z(*n*), lat(*n*), および lon(*n*) と関連付けられる。

5.6 水平の座標参照系、格子写像、および投影法

水平の格子の座標変数が経度でも緯度でもないとき、真の緯度及び経度の座標が coordinates 属性を通じて与えられることが必須である。もしそれに加えて、与えら得れた座標変数と真の緯度及び経度の座標値との間の写像（mapping）[48]を記述することが望ましいならば、属性 grid_mapping を使ってその記述を与えることができる。この属性は文字列値を取り、それはファイルの中のもうひとつの変数の名前であり、この変数に付与された属性群を通じてこの写像を記述するものである。この変数を***格子写像変数***と呼び、データは保持しないので型は何でもよい。この変数の目的は写像の定義を行う属性たちのコンテナとして働くことである。講師写像変数が必ず持たなければならない一つの属性は grid_mapping_name であり、写像の名前の文字列値をとる。特定の写像を定義するその他の属性は、属性 grid_mapping_name の値に依存する。属性 grid_mapping_name に有効な値と、特定の写像パラメタ値を与える属性は、付録 F「格子写像」に記述されている。

水平格子の座標変数が経緯度であるときは、格子写像変数に値 latitude_longitude をもつ grid_mapping_name 属性を与えて楕円体と本初子午線を指定することができる。それは座標変数に standard_name 属性を割り当てることで指定できる。

格子写像を使って緯度及び経度の値を直接計算できるように、座標変数と写像の独立変数を関連付ける必要がある。属性 standard_name として適切な値は格子写像に依存し、付録 F「格子写像」で与えられている。

例5.6. 極を回転した格子

```
dimensions:
```

[48]訳注：mapping を写像と訳すのは数学用語。この文書でいう (grid) mapping にあたるものはほとんど地図投影法であるから、「地図投影」等と意訳したいところだが、地図投影法の本来の用語 map projection と一応区別されているように見えなくもないので念のため訳し分けた。

```
  rlon = 128 ;
  rlat = 64 ;
  lev = 18 ;
variables:
  float T(lev,rlat,rlon) ;
    T:long_name = "temperature" ;
    T:units = "K" ;
    T:coordinates = "lon lat" ;
    T:grid_mapping = "rotated_pole" ;
  char rotated_pole
    rotated_pole:grid_mapping_name = "rotated_latitude_longitude" ;
    rotated_pole:grid_north_pole_latitude = 32.5 ;
    rotated_pole:grid_north_pole_longitude = 170. ;
  float rlon(rlon) ;
    rlon:long_name = "longitude in rotated pole grid" ;
    rlon:units = "degrees" ;
    rlon:standard_name = "grid_longitude";
  float rlat(rlat) ;
    rlat:long_name = "latitude in rotated pole grid" ;
    rlat:units = "degrees" ;
    rlon:standard_name = "grid_latitude";
  float lev(lev) ;
    lev:long_name = "pressure level" ;
    lev:units = "hPa" ;
  float lon(rlat,rlon) ;
    lon:long_name = "longitude" ;
    lon:units = "degrees_east" ;
  float lat(rlat,rlon) ;
    lat:long_name = "latitude" ;
    lat:units = "degrees_north" ;
```

CF に適合するアプリケーションは、変数 rlon および rlat が回転された格子での経緯度であることをその標準名 grid_longitude 及び grid_latitude を認識することにより決定できる。回転された経緯度の軸の単位は degrees とされていることに留意せよ。こうすることにより COARDS に適合するアプリケーションが変数 rlon 及び rlat を実際の経緯度座標と誤認することを防ぐはずである。単位 degrees の符号に関する適切な規約は標準名表のそれらの名前の項に示されている。

例5.7. ランベルト正角図法

```
dimensions:
 y = 228;
 x = 306;
 time = 41;

variables:
  int Lambert_Conformal;
    Lambert_Conformal:grid_mapping_name =
"lambert_conformal_conic";
    Lambert_Conformal:standard_parallel = 25.0;
    Lambert_Conformal:longitude_of_central_meridian = 265.0;
    Lambert_Conformal:latitude_of_projection_origin = 25.0;
  double y(y);
    y:units = "km";
    y:long_name = "y coordinate of projection";
    y:standard_name = "projection_y_coordinate";
  double x(x);
    x:units = "km";
    x:long_name = "x coordinate of projection";
```

```
    x:standard_name = "projection_x_coordinate";
  double lat(y, x);
    lat:units = "degrees_north";
    lat:long_name = "latitude coordinate";
    lat:standard_name = "latitude";
  double lon(y, x);
    lon:units = "degrees_east";
    lon:long_name = "longitude coordinate";
    lon:standard_name = "longitude";
  int time(time);
    time:long_name = "forecast time";
    time:units = "hours since 2004-06-23T22:00:00Z";
  float Temperature(time, y, x);
    Temperature:units = "K";
    Temperature:long_name = "Temperature @ surface";
    Temperature:missing_value = 9999.0;
    Temperature:coordinates = "lat lon";
    Temperature:grid_mapping = "Lambert_Conformal";
```

アプリケーションは、変数 x および y が投影面上での座標であることをその標準名 projection_x_coordinate 及び projection_y_coordinate を認識することにより決定できる。格子写像変数 Lambert_Conformal は写像のパラメタを属性として保持しとており、変数 Temperature からは属性 grid_mapping を通じて関連付けられている。

例5.8. 球面の地球の上の緯度と経度

```
dimensions:
  lat = 18 ;
  lon = 36 ;

variables:
  double lat(lat) ;
  double lon(lon) ;
  float temp(lat, lon) ;
    temp:long_name = "temperature" ;
    temp:units = "K" ;
    temp:grid_mapping = "crs" ;
  int crs ;
    crs:grid_mapping_name = "latitude_longitude"
    crs:semi_major_axis = 6371000.0 ;
    crs:inverse_flattening = 0 ;

dimensions:
  lat = 18 ;
  lon = 36 ;

variables:
  double lat(lat) ;
  double lon(lon) ;
  float temp(lat, lon) ;
    temp:long_name = "temperature" ;
    temp:units = "K" ;
    temp:grid_mapping = "crs" ;
  int crs ;
    crs:grid_mapping_name = "latitude_longitude"
    crs:semi_major_axis = 6371000.0 ;
    crs:inverse_flattening = 0 ;
```

例5.9. WGS1984測地系における緯度と経度

```
dimensions:
  lat = 18 ;
  lon = 36 ;
variables:
  double lat(lat) ;
  double lon(lon) ;
  float temp(lat, lon) ;
    temp:long_name = "temperature" ;
    temp:units = "K" ;
    temp:grid_mapping = "crs" ;
  int crs ;
    crs:grid_mapping_name = "latitude_longitude";
    crs:longitude_of_prime_meridian = 0.0 ;
    crs:semi_major_axis = 6378137.0 ;
    crs:inverse_flattening = 298.257223563 ;
```

例5.10. 英国国家格子

```
dimensions:
  lat = 648 ;
  lon = 648 ;
  y = 18 ;
  x = 36 ;
variables:
  double x(x) ;
    x:standard_name = "projection_x_coordinate" ;
    x:units = "m" ;
  double y(y) ;
    y:standard_name = "projection_y_coordinate" ;
    y:units = "m" ;
  double lat(y, x) ;
  double lon(y, x) ;
  float temp(y, x) ;
    temp:long_name = "temperature" ;
    temp:units = "K" ;
    temp:coordinates = "lat lon" ;
    temp:grid_mapping = "crs" ;
  int crs ;
    crs:grid_mapping_name = "transverse_mercator";
    crs:semi_major_axis = 6377563.396 ;
    crs:semi_minor_axis = 6356256.910 ;
    crs:inverse_flattening = 299.3249646 ;
    crs:latitude_of_projection_origin = 49.0 ;
    crs:longitude_of_projection_origin = -2.0 ;
    crs:false_easting = 400000.0 ;
    crs:false_northing = -100000.0 ;
    crs:scale_factor_at_projection_origin = 0.9996012717 ;
```

5.7 スカラー座標変数

ある変数が単一の値しかもたない座標に関連付けられているとき、この座標をスカラー変数で表わすことができる。このようなスカラー座標変数には、関連付けられた次元が存在しないので、属性 `coordinates` を通じてデータ変数に付加されなければならない。

COARDS では単一の値を持つ座標を提供する方法は、大きさ1の次元と、対応する座標変数を変数に追加することであった。この新しいスカラー座標変数は、大きさ1の座標を変数に追加しなくて済ませる簡便法である。スカラー座標変数は、大きさ1の座標変数と同じ情報内容を持ち、

同じ文脈で使うことができる。しかしながら、この機能を緯度、経度、鉛直、または時間の座標に使うことは、COARDS に適合するアプリケーションがこれを認識する妨げになることに留意せよ。

ある名前がひとたびスカラー座標変数に使われれば、これを一次元の座標変数の名前に使うことはできない。この理由から、スカラー座標変数の名前をファイルの中の次元のどれかと一致させないようにすることを強く勧告する。

例5.11. ひとつの解析から得られた多数の予報

```
dimensions:
  lat = 180 ;
  lon = 360 ;
  time = UNLIMITED ;
variables:
  double atime
    atime:standard_name = "forecast_reference_time" ;
    atime:units = "hours since 1999-01-01 00:00" ;
  double time(time);
    time:standard_name = "time" ;
    time:units = "hours since 1999-01-01 00:00" ;
  double lon(lon) ;
    lon:long_name = "station longitude";
    lon:units = "degrees_east";
  double lat(lat) ;
    lat:long_name = "station latitude" ;
    lat:units = "degrees_north" ;
  double p500
    p500:long_name = "pressure" ;
    p500:units = "hPa" ;
    p500:positive = "down" ;
  float height(time,lat,lon);
    height:long_name = "geopotential height" ;
    height:standard_name = "geopotential_height" ;
    height:units = "m" ;
    height:coordinates = "atime p500" ;
data:
  time = 6., 12., 18., 24. ;
  atime = 0. ;
  p500 = 500. ;
```

この例では解析時刻と単一の気圧レベルがスカラー座標変数を使って表現されている。解析時刻（analysis time）は標準名 "forecast_reference_time"[49] で表わされ、他方で予報の対象時刻（valid time）は標準名 "time" で表わされている。

第6章　　ラベルと代替座標

6.1　ラベル

これまでの節で挙げた例では、散在する地点での計測はひとつの次元を使ってグループ化されていた。地点位置の座標は補助座標変数を使って提供できたが、しかししばしば計測地点を名前、あるいは何か一意の文字列を使って識別することが望ましいことがある。他にも文字列の識別子の用途が6.1.1「地理的領域」及び7.3.3「セルの一部に適用される統計」で記述されている。

[49]訳注：これ以降余計なダブルクォートが使われるようになるがおそらく特段の意味はない。念のためそのまま訳した。

文字列の識別子のリストは座標変数と似た役割[50]を演じるので、文字列配列を含む変数の名前を与える属性として coordinates を選んだ。属性 coordinates に列挙された変数を処理するアプリケーションは、文字データを含むか否か[51]を見ることによってその変数がラベル付きの軸であることを認識できる。もし文字型変数が唯一の次元（文字列の長さ）を持つならば、それは数値化他のスカラー座標変数（第5章「座標系」参照）に類似した単一値のスカラー座標変数とみなされる。

例6.1. 数個の水塊の軌跡

海洋の水塊の軌跡を追跡するフロートが数個あって、固定された時刻に同時に計測を行うとしよう。フロートを名前で識別したい。温度データは水塊（すなわちフロート）と時間の関数である。それぞれのサンプルの位置は水塊と時間の関数なので、位置情報は多次元座標変数に格納される。

```
Dimensions:
  parcel = 15 ; // 原注：軌跡の数
  times = 20 ;
  max_len_parcel_name = 64 ; // 原注：軌跡名の最大長
variables:
  float temperature(parcel,times) ;
    temperature:coordinates = "parcel_name lat lon" ;
  float times(times) ;
  char parcel_name(parcel,max_len_parcel_name) ;
  float lon(parcel,times) ;
  float lat(parcel,times) ;
```

6.1.1 地理的領域

データが地理的な領域を代表しており、その領域は名前では識別できるが複雑な境界形状のため境界の経緯度を使って指定することが現実的でないとき、領域を識別するのにラベル付きの軸を使うべきである。この名前は標準化された領域名のリスト[52]から選ぶことを勧告する。標準のラベル値であることを示すためには[53]、ラベルを保持する変数に値 region の standard_name 属性を与えなければならない。

例6.2. 大西洋での北向き熱輸送

大西洋のいくつかの東西断面を横切る北向き熱輸送量を表わすデータを持っているとしよう。この量を表わす標準名には位置情報は含まれていないことに留意せよ。位置情報は緯度座標とラベル付き軸で与えられる:

```
dimensions:
  times = 20 ;
  lat = 5
  lbl = 1 ;
  strlen = 64 ;
variables:
  float n_heat_transport(time,lat,lbl);
    n_heat_transport:units="W";
    n_heat_transport:coordinates="geo_region";
```

[50]訳注：察するに、次元に意味を与える役割、といいたいのであろう。
[51]訳注：文字型であるか否か、と同義と考えられるが、誤解の余地を生む微妙な規定振りである。
[52]<URL:http://cf-pcmdi.llnl.gov/documents/cf-standard-names/standardized-region-names>
[53]訳注：文言どおりには、非標準の領域名を使っても standard_name= "region" が必須であるが、いかにも不条理なので、必須規定は標準領域名を示す目的に限られると解して「ためには」と補った。しかしそれでは目的意識を外形的に適合性判定できないのでザル法化してしまう。

```
n_heat_transport:standard_name="northward_ocean_heat_transport";
  double time(time) ;
    time:long_name = "time" ;
    time:units = "days since 1990-1-1 0:0:0" ;
  float lat(lat) ;
    lat:long_name = "latitude" ;
    lat:units = "degrees_north" ;
  char geo_region(lbl,strlen) ;
    geo_region:standard_name="region"
data:
  geo_region = "atlantic_ocean" ;
  lat = 10., 20., 30., 40., 50. ;
```

6.2 代替座標

状況によっては、ある次元がひとつだけではなく代替の座標値をとりうることがある。ひとつの次元にはただひとつの座標変数(次元と同名の変数)しかありえないので、大体の値はいずれも補助座標変数に格納せざるを得ない。このような代替座標変数については、必須の属性はないが、座標変数に認められる属性はいずれも持つことができる。

例6.3. モデルのレベル番号

鉛直軸に沿ったレベルは物理的座標と順序的なモデルレベル番号との両方で記述されうる。

```
float xwind(sigma,lat);
  xwind:coordinates="model_level";
float sigma(sigma); // 物理的高度座標
  sigma:long_name="sigma";
  sigma:positive="down";
int model_level(sigma); // それぞれの高度でのモデルの層番号
  model_level:long_name="model level number";
  model_level:positive="up";
```

第7章 セルのデータ表現

格子データが場の格子点での値を代表するのではなく、むしろ有限の「体積」をもったセルの中の場の何らかの特徴を代表しているとき、変数の完全な記述には、それぞれのセルの領域または外形を記述するメタデータと、セルの値が表現する場の特徴が含まれるべきである。ひとつのデータ値が、バラバラなのセル集合にまたがる演算の結果ということもありえる。たとえば、多くの種類の気候学的平均では、1月の平年値というとき1971年[54]から2000年の平均である。以下に示すセルの記述は、格子点と単一のセルとを関連付けるだけで、セルの集合との関連付けは行わない。しかしながら、気候学的統計は非常に重要なので、関連付けられた計算領域を記述する特別な方法を7.4「気候学的統計」で提供する。

7.1 セル境界

セルを表現するために、我々は適切な座標変数に属性 bounds を追加する。属性 bounds の値は、セル境界の頂点を保持する変数の名前である。この種の変数を「境界変数」と呼ぶことにする。***境界変数は、関連付けられた座標変数または補助座標変数よりも一つ多い次元を持つ***。追加的な次元は最も速く変化するもので、その大きさはセルあたりの頂点の最大の数[55]である。

[54]訳注：原文では1970だが誤り。

[55]訳注：直方体ならば、n 次元セルに 2^n 個の頂点があるが、不定形ならばセルによって頂点の数が違うかもしれないので最大と言っているのだろう。セルによって頂点の数が違う場合の扱いは明らかでない。

境界変数は座標変数のメタデータの一部とみなされるので、属性 long_name や units を与える必要はない。

N 個の連続したインターバルに対する境界変数は、形状 (*N*, 2) の配列であることに留意せよ。この場合は隣接するインターバルの境界の座標が重複することになるが、利点としては表現法が十分に一般的なので、不連続なインターバルや、無制限の次元を使ったインターバル[56]にも変更なしに適用できることである。

セル境界データを処理するアプリケーションは、しばしば隣接したセルが端を共有するか否かを決定する必要があることがある。この種の処理を容易にするために、境界変数のデータには次の制約が課されている。

一次元座標変数の境界

たとえば lat(lat) のような座標変数に境界変数たとえば latbnd(lat, 2) [57]が関連付けられている場合、インターバルの端点は関連付けられた座標に沿って一貫した順序に並べられていなければならない;たとえば増加する座標について lat(1) > lat(0) はすべての *i* について latbnd(*i*,1) >= latbnd(*i*,0) であることを意味する。
また、もし隣接するセルが連続しているなら、共有の端点は境界変数の該当箇所もまったく同じにあらわされなければならない。たとえば、もし格子点 lat(*i*) 及び lat(*i*+1) を含むセルが連続しているならば、latbnd(*i*+1,0) = latbnd(*i*,1) でなければならない。

二次元座標変数でセルが4つの頂点を持つ場合

水平格子が二次元の補助座標変数によって緯度 lat(n, m) 及び経度 lon(n, m) のように表わされている場合、それらに関連付けられたセルは4つの辺を持ち、境界変数は latbnd(n, m, **4**) 及び lonbnd(n, m, **4**) となる。ここで最後のインデックスがセルの4つの頂点を走査する。セル (*j*, *i*) の辺でセル (*j*, *i*−1) に面しているものを "*i*−1" 辺、セル (*j*, *i*+1) に面しているものを "*i*+1" 辺、同様に "*j*+1" および "*j*−1" と呼ぶことにしよう。そこで辺 *i*−1 および *j*−1 で構成される頂点を (*j*−1, *i*−1) と呼ぶことができる。この記法を使えば、4つの頂点は次のようにインデックスできる: **0**=(*j*−1, *i*−1), **1**=(*j*−1, *i*+1), **2**=(*j*+1, *i*+1), **3**=(*j*+1, *i*−1).
もし座標系 i-j-*up* を(lon-lat-*up* のように)右手系をなすようにとるなら、この順序は lon-lat 平面を上から見たときに頂点を反時計回りに追うことを意味する。もし i-j-*up* が左手系ならば、頂点は時計回りに追われることになる。
次の関係を使って、2つのセルが連続か否かを決めることができる。これらの方程式において、変数 bnd は緯度または経度の境界変数を一般的に表わすものとして使っている。

任意の 0 < *j* < n および 0 < *i* < m について、
もしセル (*j*, *i*) および (*j*, *i*+1) が連続ならば、
bnd(*j*, *i*, **1**) = bnd(*j*, *i*+1, **0**)
bnd(*j*, *i*, **2**) = bnd(*j*, *i*+1, **3**)
もしセル (*j*, *i*) および (*j*+1, *i*) が連続ならば、
bnd(*j*, *i*, **3**) = bnd(*j*+1, *i*, **0**) そして

[56]訳注：無制限の次元は netCDF データセット作成時に大きさを指定しない次元で、データ書き込みによって延長されてゆくものであり、netCDF バージョン3では1つのデータセットに1つだけ認められる。連続したインターバルの頂点を重複なしに格納すべく、座標変数より1つだけ大きな別の次元に1次元の境界変数を保存することにすると（実際に NCAR CSM 規約[CSM]はそうである）、無制限次元にすることができないことになる。

[57]訳注：原文では latbnd(x, 2)となっているが x という次元はない。CDL の変数宣言ならば本来は2のような即値は使えず dimension nv = 2; のようなものを使う必要があるが、長くなるのでこうした。

```
        bnd(j, i, 2) = bnd(j+1, i, 1)
```

多次元の座標変数で *p*-個の頂点[58]がある場合

上記以外のすべての場合、境界の次元は (..., *n*, *p*) と与えるべきである;ここで (..., *n*) は補助座標変数の次元であり、*p* はセルの頂点の数である。頂点を追う順は、上から lon-lat 平面を見たときに左回りになるようにしなければならない。どの頂点から始めるかは特に指定しない。

例7.1. 緯度軸上のセル

```
dimensions:
  lat = 64;
  nv = 2;     // 頂点の数
variables:
  float lat(lat);
    lat:long_name = "latitude";
    lat:units = "degrees_north";
    lat:bounds = "lat_bnds";
  float lat_bnds(lat,nv);
```

境界変数 lat_bnds は緯度方向の格子点 *i* と、境界 lat_bnds(*i*, 0) および lat_bnds(*i*, 1) によるインターバルとを関連付ける。格子点の位置 lat(*i*) はこのインターバルに含まれているべきである。

長方形の格子については、二次元のセルは上の例のような種類の一次元のセルの直積 (Cartesian product)として表現できる。しかしながら、非長方形の格子について、「長方形の」セルは一般的にそれぞれのセルについて4つの頂点すべてを指定する必要がある。

例7.2. 非長方形格子のセル

```
dimensions:
  imax = 128;
  jmax = 64;
  nv = 4;
variables:
  float lat(jmax,imax);
    lat:long_name = "latitude";
    lat:units = "degrees_north";
    lat:bounds = "lat_bnds";
  float lon(jmax,imax);
    lon:long_name = "longitude";
    lon:units = "degrees_east";
    lon:bounds = "lon_bnds";
  float lat_bnds(jmax,imax,nv);
  float lon_bnds(jmax,imax,nv);
```

境界変数 lat_bnds 及び lon_bnds は格子点 (*j*, *i*) を頂点 (lat_bnds(*j*, *i*, *n*), lon_bnds(*j*, *i*, *n*)), *n* = **0**, ..., **3** で決まるセルに関連付ける。格子点位置 (lat(*j*, *i*), lon(*j*, *i*)) はこの領域に含まれるべきである。

7.2 セルの測度

ある種の計算ではセルについて、汎用アプリケーションが持つことを期待されていないような特

[58]訳注： 原文 *p*-sided cells だが side というのは明らかに誤り。辺と頂点の数は違う。

別な知識なしでは座標と境界だけからは演繹できないような大きさ、形、あるいは位置の情報が必要となることがある。たとえば、いくつかのセルの値の平均を計算する際には、値に面積により「重み」をかけることがしばしば適切である。面積平均を計算するとき、合計を計算する前にそれぞれの格子セル値に格子セルの面積を乗算し、次に和を格子セル面積の合計で割るものである。また、このような面積による重み演算は、データをある格子から別の格子に変換する際にも場の面積平均を保存するために必要となることがある。格子変換の際の面積平均値の保存は、たとえば、大気モデルの中での地表面熱流量を、それに結合した海洋モデルの異なる格子の上で算出すr際に不可欠である。

多くの場合、面積はセル境界から計算できるが、しかし例外もある。たとえば、球面上の測地学的な格子で、連続していて概ね6角形に近い形のセルからなっているものを考えよう。セルの頂点は bounds 属性で識別される変数に格納できるが、しかしセルの外周の長さはその頂点だけから一意に決まるものではない。なぜなら頂点はたとえば直線で、あるいは球面なら大円で、あるいは一般には何らかの異なる線で結ばれ得るからである。したがって、セル頂点だけが与えられても、一般には格子セルの面積を計算することは不可能である。格子セルの面積をセル頂点とは別に保存すべき理由である。

他の場合には、格子セルの体積が必要であるが、座標情報からは用意には計算できないこともある。たとえば、海洋モデルでは海底に「部分」格子がみられることが珍しくない。この場合、格子セルの面積を示す代わりに(あるいはそれに加えて)、体積を示すことが必要となりうる。

ある変数の格子セルについての空間的特性についての追加的情報を示すため、属性 cell_measures を定義することができる。これは文字列属性で、"***measure*: *name***" の形をした語の対からなる空白区切りのリストである[59]。現在のところ、"area" 及び "volume" だけが測度 ***measure*** として定義されているが、将来はほかのものもサポートされうる。文字列 "name" は測度の値を保持する変数の名前であり、これを測度変数と呼ぶことにする。測度変数の次元は、それが関連する変数の次元と同一あるいは部分集合であるべきであるが、しかし順序は制限されない。たとえば面積について、場自体は経度、緯度、及び時間の関数であったとして、しかし面積値を保持する変数は経度と緯度だけを含むかもしれない(そしてその順序は逆転するかもしれない;勧告されることではないが)。測度変数は units 属性を持たねばならず、standard_name などの他の属性を持ちうる。

経緯度の長方形的格子については、格子セルの面積は境界から計算できる:セルの面積は、セルの経度境界の差と、セルのそれぞれの緯度境界の正弦の差の積に比例する。この場合はアプリケーションが地球の半径として適当な値を用いてこの計算をできることは仮定してよいから、格子セル面積を cell_measures 属性を通じて与えることは不必要である。

例7.3. 球面上の測地格子のセル面積

```
dimensions:
  cell = 2562 ;  // 格子セルの数
  time = 12 ;
  nv = 6 ;       // セル頂点の最大数
variables:
  float PS(time,cell) ;
    PS:units = "Pa" ;
    PS:coordinates = "lon lat" ;
```

[59]訳注:アプリケーションにとって空白の許容される位置は重要であるが本文上それが明確になっているとは言いがたい。例では一貫してコロンの前にスペースなし、コロンの後にスペース1つとしており、欧文の常識だからわざわざ書かないのだろうが、どういうわけか日本人にはそうしたがらない方が散見されるので、特に注意願いたい。

```
    PS:cell_measures = "area: cell_area" ;
  float lon(cell) ;
    lon:long_name = "longitude" ;
    lon:units = "degrees_east" ;
    lon:bounds="lon_vertices" ;
  float lat(cell) ;
    lat:long_name = "latitude" ;
    lat:units = "degrees_north" ;
    lat:bounds="lat_vertices" ;
  float time(time) ;
    time:long_name = "time" ;
    time:units = "days since 1979-01-01 0:0:0" ;
  float cell_area(cell) ;
    cell_area:long_name = "area of grid cell" ;
    cell_area:standard_name="area";
    cell_area:units = "m2"
  float lon_vertices(cell,nv) ;
  float lat_vertices(cell,nv) ;
```

7.3 セル・メソッド

セル値で表現される場の特徴を記述するために、我々は変数に cell_methods 属性を定義する。これは文字列属性で、"*name*: *method*" の形の空白区切りの単語リストである。それぞれの "*name*: *method*" ペアは、*name* で識別される軸については場を表現するセル値が *method* で指定される方法で決定あるいは算出されたということを示す。たとえば、もしデータ値が平均を計算することで生成されたならば、そのことは時間の次元変数の名前が "t" だとして cell_methods="t: mean" と示すことができる。

この属性の規定においては、*name* は変数の次元、スカラー座標変数、あるいは有効な標準名、あるいは予約語 "area" (7.3.4「座標がないところでのセル・メソッド」参照) でありうる。語 *method* の値は付録 E「セルメソッド」たとえば point, sum, mean, maximum, minimum, mid_range, standard_deviation, variance, mode, あるいは median から選ばれるべきである。メソッド名の大文字と小文字の違いは区別されない。いくつかのメソッド(例:variance)は付録 E「セルメソッド」に示したように変数の単位を変えることを暗黙に意味する。

メソッドは cell_methods において *name* で指示した軸についてのみ適用され、他の軸には異なるメソッドが適用されるかもしれないことは覚えておかなければならない。たとえばもし、経緯度セルに降水量の値が与えられていて、これらの軸にメソッド maximum がついていたら、それは空間セル内での最大であることを意味し、時間についての最大値であることは意味しない。さらに、ある軸に "point" 以外の何らかのメソッドが指定されているとき、その軸に属性 cell_bounds を与えることは必須である(7.3.4「座標がないところでのセル・メソッド」で記述する比較的まれな例外は除く)ことは留意すべきである。

属性 cell_methods を持たない変数の既定の解釈は、その量が示量的(extensive, セルの大きさに依存する)か、示強的(intensive, セルの大きさによらない)かに依存する。たとえば、量「積算雨量」及び「降水強度」がそれぞれに時間軸を持っていたとしよう。積算雨量を表わす量は時間にとって示量的である;なぜならその量は積算を行う時間インターバルの長さに依存するからである。したがって、正しい解釈のためには、境界変数(すなわち時間軸についての cell_bounds を通じて)を用いて時間間隔を完全に指定する必要がある。この場合に既定の解釈は、セルメソッドは指定された時間インターバルについての sum であるとすることである。このことは、セルメソッドを明示的に sum と指定することによって、(必須ではないが)示すことができる。一方で、降水強度は時間について示強的であって、瞬間値あるいはセルで指定される時間インターバルについての平均のどちらも等しくよく表わすことができる。この場合は量の既定の解釈は

「瞬間値的」となろう(必須ではないが、そのことはセルメソッドを point とすることで明示的に示すことができる)。しかしながら、より多くの場合、示強的な量のセル値は平均であり、セルメソッドを mean としてセル境界を指定することで明示的に表現すべきである。

示強的な量の既定の解釈は示量的な量とは違うため、また、この違いは一部のデータ利用者には理解されないかもしれないので、すべてのデータ変数のそれぞれの次元とスカラー座標変数とについて(無意味でない限り)関連する cell_methods 情報を明示的に[60]与えることを勧告する。時空間の次元及びスカラー座標変数については cell_methods を明示的に与えることが強く[61]勧告される。

例7.4. タイムスライスにメソッドを適用する例

多くの地点から12時間毎に気圧、気温、及び降水量のタイムスライスが得られるとしよう;ここで気圧は瞬間値が測定され、最高気温は過去12時間の最高が記録され、降水量は過去12時間にわたって雨量計で積算される。1998年4月19日午前6時からの48時間の期間について、データは次のように構造化される:

```
dimensions:
  time = UNLIMITED; // (5 currently)
  station = 10;
  nv = 2;
variables:
  float pressure(station,time);
    pressure:long_name = "pressure";
    pressure:units = "kPa";
    pressure:cell_methods = "time: point";
  float maxtemp(station,time);
    maxtemp:long_name = "temperature";
    maxtemp:units = "K";
    maxtemp:cell_methods = "time: maximum";
  float ppn(station,time);
    ppn:long_name = "depth of water-equivalent precipitation";
    ppn:units = "mm";
    ppn:cell_methods = "time: sum";
  double time(time);
    time:long_name = "time";
    time:units = "h since 1998-4-19 6:0:0";
    time:bounds = "time_bnds";
  double time_bnds(time,nv);
data:
  time = 0., 12., 24., 36., 48.;
  time_bnds = -12.,0., 0.,12., 12.,24., 24.,36., 36.,48.;
```

この例では時間軸の値はそれぞれのインターバルの末尾に位置していることに留意せよ。これは気圧の瞬間値を計測した時刻として必要である。しかしながら、上の気温や降水量のようなデータについてインターバルの中間点をインターバルを代表する座標変数値とすることが望ましいこともある。アプリケーションは time_bnds から得られた境界データを利用して単純に中間点を得ることができる。

7.3.1 複数軸の場合の統計

もし複数のセルメソッドを示すべきなら、それらが適用された順に配列されるべきである。最も左

[60]訳注:この文には明示的にとは書いていないが、文意からして標準名表による暗黙の解釈は無意味。
[61]訳注:ここだけ strongly ではなく原文 especially 。

の処理が最初に適用されたものと仮定される。たとえば、それぞれの格子セルの中である量が緯度と時間によって変動しており、それらは "lon" および "time" という名前であったとしよう。すると、東西方向での最大値を時間方向に平均した代表値は cell_methods="lon: maximum time: mean" (すなわち、それぞれの時刻についてすべての経度にわたって最大値を求め、ついでその最大値を時間にわたって平均する)のようにラベルされる;逆に時間平均の東西最大値は cell_methods="time: mean lon: maximum" とラベルされる。もし複数のメソッドが互いに影響を与えずに適用できるならば、どのような順序で cell_methods 属性に入れてもよい。

もしあるデータ値が複数の軸の組み合わせにまたがる変動を代表しているならば、ひとつのメソッドの前に関連するすべての次元を置くべきである(順序は本質的ではないはずなので、どのような順序でもよい)。このように、次元を個別に扱うのとは本質的に異なる場合にだけ、次元をグループ化すべきである。たとえば、経緯度の格子セル[62]の中での地形の高度の標準偏差は、cell_methods="**lat: lon:** standard_deviation" を持つことになろう。(なお、次の段落で示す勧告により、cell_methods="**area:** standard_deviation" と示しても等価でありむしろその方が望ましいことに留意せよ。) 上の例は、cell_methods="lon: standard_deviation lat: standard_deviation" とは同じではない;こうするとまずそれぞれの緯線について格子セルの東西範囲にわたって標準偏差を得て、ついでその値から緯度について標準偏差を求めることを意味することになってしまう。

水平の面的領域にわたる変化を示すために、特別の文字列 "area" を使うことが勧告される。したがって一般的な面積平均の例は、(たとえば "lon: lat: mean" のようにする代わりにむしろ) cell_methods="area: mean" と示すことができる。この場合 "area" の指す水平の座標変数は cell_methods には明示されていないが、必要ならば第4章「座標の種別」で記述したように座標変数、スカラー座標変数、または補助座標変数に付与された属性から識別できる。

7.3.2 オリジナルデータの間隔などの記録

セルメソッドがどのように適用されたかをより精密に示すために、メソッドの識別語の後の括弧 ( ) 内に追加的情報を含めることができる。この情報は一部標準化され、一部は標準化されていない。現在のところ標準化されているのは、現在のデータ値がより細かい空間間隔をもっていた元データのデータ値を統計的に代表しているときに、メソッドが適用された元データのデータ値の典型的な間隔を与えるためのものである。文法は (interval: *value unit*) であり、ここで *value* は数値であり *unit* は UNIDATA の Udunits パッケージで認識できる文字列である。文字列 *unit* は通常は対応する次元の単位と同等であるが、そうすることは必須ではない(そうしておくことで、たとえば、緯線に沿って等距離で配置された点群から得られた標準偏差の元データ間隔を、たとえ東西方向の座標は度単位で与えられていても、長さの単位で報告できるようになる)。元データ間隔を記録しておくことは、標準偏差にとってとりわけ重要である。たとえば、毎日の値の標準偏差は cell_methods="time: standard_deviation (interval: 1 day)" として示され、毎年の値の標準偏差は cell_methods="time: standard_deviation (interval: 1 year)" として示される。

もしセルメソッドが軸の組み合わせに適用されるならば、共通の元データ間隔があるかもしれない;たとえば cell_methods="lat: lon: standard_deviation (interval: 10 km)" のようである。かわりに、軸ごとに個別の間隔があるかもしれない;それらは軸の名前と同じ順序で示す;たとえば cell_methods="lat: lon: standard_deviation

[62]訳注: この段落でだけ gridbox.

(interval: 0.1 degree_N interval: 0.2 degree_E)" のようである;ここで0.1度は緯度に、0.2度は経度に適用される。

もし標準と非標準の両方の情報があるならば、非標準の情報は標準情報と予約語 comment: に続けて記述する。もし標準情報がないならば、予約語 comment: は除くべきである。たとえば、緯度に関して面積重みされた平均は lat: mean (area-weighted) あるいは lat: mean (interval: 1 degree_north comment: area-weighted) と示すことができる。

大きさ1の次元は、何らかの統計的操作で軸を「つぶす」ことで作られたものかもしれない;たとえば時系列データの分散を計算すると、結果のデータに時間軸は本来的には存在しないことになる。しかし我々は(cell_methods 属性を通して)メソッドと(cell_bounds 属性を通して)領域のドキュメンテーションを可能にするため、大きさ1の次元を残す(あるいはスカラー座標変数を定義する)ことを強く勧告する。

例7.5.地表気温の分散

1990年1月1日の日周期変動の分散が、地表面気温観測値の毎時の瞬間値から計算されている。大きさ1の時間次元が保持されている。

```
Dimensions:
 lat=90;
 lon=180;
 time=1;
 nv=2;
variables:
 float TS_var(time,lat,lon);
 TS_var:long_name="surface air temperature variance"
 TS_var:units="K2";
 TS_var:cell_methods="time: variance (interval: 1 hr comment:
sampled instantaneously)";
 float time(time);
 time:units="days since 1990-01-01 00:00:00";
 time:bounds="time_bnds";
 float time_bnds(time,nv);
data:
 time=.5;
 time_bnds=0.,1.;
```

属性 cell_methods の括弧内のコメントは分散を計算するために用いられたサンプルの性質を伝えるものである。

7.3.3 セルの一部に適用される統計

既定の解釈では、属性 cell_methods で示される統計操作はセルの水平領域全体にわたって評価されるものと仮定される。しかしながら、セルの一部だけを考慮する限定(たとえば海氷域だけの平均)が有益なことが時折ある。これを示すために一つまたは二つの規約が用いられる。

最初の規約は、単一面積型÷の一般的なケースで使える。この場合は、属性 cell_methods に "*name*: *method* where *type*" のような形式の文字列を含める。ここで *name* がたとえば area だとすれば、*type* は標準名 area_type のついた変数値として認められている文字列のいずれかでありうる。たとえば、*method* が mean そして *name* および *type*[63] が area および sea_ice ならば、そのデータは格子セルのうち海氷部分だけの平均を表わしている。もし、*type* は標準

[63]訳注:原文 the area_type were sea_ice.

area_type 文字列のひとつと解釈されることをデータ作成者が期待するならば、netCDF ファイルのなかのどの変数の名前もその文字列と一致するものを与えるべきではない（次の段落で記述する第二の規約が優先するため）。

第二の規約はより一般的なものである。この場合、cell_methods の該当部分は "*name*: *method* where *typevar*" の形式をとる。ここで *typevar* は文字列値の補助座標変数あるいは文字列値のスカラー座標変数（6.1「ラベル」参照）の名前でその standard_name 属性に area_type をもつ。変数 *typevar* は格子セルのうち *method* が適用される部分の名前を保持する。この規約は、メソッドが二種類以上の領域に適用される場合や、結果が（多種の領域にわたる次元をもつ）単一のデータ変数に保持される場合にも適用できる。この規約はたとえば地表面モデルなど、地表面の格子セルに多様な領域（たとえば vegetation, bare_ground, snow 等）を扱うモデルの出力を格納するのに便利な方法を提供する。

例7.6. 地上の地表面平均気温と、地表と海面でそれぞれ平均された顕熱フラックス

```
dimensions:
  lat=73;
  lon=96;
  maxlen=20;
  ls=2;
variables:
  float surface_temperature(lat,lon);
    surface_temperature:cell_methods="area: mean where land";
  float surface_upward_sensible_heat_flux(ls,lat,lon);
    surface_upward_sensible_heat_flux:coordinates="land_sea";
    surface_upward_sensible_heat_flux:cell_methods="area: mean
where land_sea";
  char land_sea(ls,maxlen);
    land_sea:standard_name="area_type";
data:
  land_sea="land","sea";
```

もし *method* が mean ならば、平均を算出するためのさまざまな方法を cell_methods 属性の "mean where *type1* [over *type2*]" の形式の文字列で区別できる。ここで *type1* は *typevar* または *type* として許容されるもの（上の例の前の二つの段落で規定）のひとつを取りえる。同様の規則が *type2* にもあてはまるが、しかし二次元以上（文字列の最大長の次元を除いて数える）の補助座標変数の名前を用いることは許容されない。属性 cell_methods の形式 "mean where *type1* over *type2*" の文字列は、セル内の type1で示される部分での合計を、部分 type2 の面積で割ってこの平均が計算されたことを示している。特に、形式 "mean where all_area_types over *type2*" の cell_methods 文字列は、セル内のすべての種類の領域での合計を部分 type2 の面積で割ってこの平均が計算されたことを示している（なお "all_area_types" は標準名 area_type の変数に認められた有効な文字列のひとつであることに留意せよ）。もし "over *type2*" を欠く場合は、セルの *type1* 部分での合計を、その部分の面積で割って計算されたものがこの平均である。

例7.7. 海氷と海氷上の積雪の厚さの全海域での平均

```
variables:
  float sea_ice_thickness(lat,lon);
    sea_ice_thickness:cell_methods =
"area: mean where sea_ice over sea";
    sea_ice_thickness:standard_name = "sea_ice_thickness";
    sea_ice_thickness:units = "m";
  float snow_thickness(lat,lon);
    snow_thickness:cell_methods =
```

```
"area: mean where sea_ice over sea";
   snow_thickness:standard_name =
"lwe_thickness_of_surface_snow_amount";
    snow_thickness:units = "m";
```

海氷の厚さの場合は、句 "where sea_ice" を "where all_area_types" と置き換えても意味は変わらない;なぜなら全種類の領域での海氷の厚さの積分は明らかに海氷域だけでの積分と同じだからである。積雪深の場合は、"where sea_ice" は "where all_area_types" とは異なる;なぜなら "where sea_ice" は陸上の雪を平均から除外するからである。

7.3.4 座標がないところでのセル・メソッド

あるデータに特定のセルメソッドがあてはまることを示しつつ、対応するセルについての精密な記述を与えないで済ませるために、ペア "*name*: *method*" の中の "*name*" を(スカラー座標変数または座標変数付き次元の名前のかわりに)適切な(次元を識別する)標準名あるいは文字列 "area" とすることができる。しかしながら、もし netCDF データセット内に既存の次元またはスカラー座標変数の名前が *name* と一致するならば、この規約を用いることはできない。この規約が有用な状況は二つある。

まず、セルの座標範囲が精密に定義されていないときに、メソッドについて示すことができる。たとえば、気候学的な平均はどのようなデータが存在するのでもそれに基づいて定義され、一般に、どこでも同じ時間期間でデータが得られるというわけにはいかない。この場合、データセット全体の時間範囲は(場所によって変わるので)よく定義されず、時間次元の境界を通じて精密に指定できないかもしれない。にもかかわらず、cell_methods のエントリ "time: mean" (ここで time は有効な標準名であることに留意すべきである)によって有益な情報が伝達できる。(この規約で要請しているように、この cell_methods 属性が参照するデータセットについて "time" は次元あるいは座標変数の名前ではないことを仮定している。)

次に、いくつかの特別な次元については、この規約によって(明示的に座標を定義することなく)その次元に許される全範囲にわたる領域にメソッドが適用されたことを示すことができるようになる。これは経度、緯度、および(水平座標の組み合わせを示すことにより)面積についてだけ認められる。経度については、(経度座標変数の名前の代わりの)文字列 "longitude" によってこの規約に従い領域が示されることとなり、メソッドはすべての可能な経度(すなわち 0°E から360°E まで)に適用されることを意味する。経度については文字列 "latitude" が用いられ、メソッドがすべての可能な緯度(すなわち 90°S から 90°N まで)に適用されることを意味する。面積については、文字列 "area" が用いられ、メソッドが全世界に適用されることを意味する。

第二の場合、もしデータ変数に上記のセルメソッド記述に加え、水平次元に対応するような地理的領域を指定するラベル軸(6.1.1「地理的領域」)を伴った次元が与えられているならば、経度及び緯度について暗黙に示される範囲はそれぞれの指示された領域に有効な範囲となり、あるいは area の場合は領域はその地理的領域となる。たとえば、cell_methods エントリ "longitude: mean" があり得る;ここで longitude は次元あるいは座標変数(ただし上述の特例を除く)の名前では**ない。** これはすべての経度についての平均を意味する。しかしながら、データ変数に標準名 region 値 atlantic_ocean のスカラー座標変数が付属しているならば、全経度ではなく大西洋の範囲の経度についての平均を意味することになる。

この節の規定にかわらずデータ提供者は、いつでも可能なときは、大きさ1の次元を変数に与え、座

標変数を関連付けて境界を与えることにより、セル境界を提供すべきものと勧告する[64]。

7.4 気候学的統計

気候統計は何年か分の年周期のうち対応する部分から作られることがある;たとえば、別々の年からの30個の1月を平均して得た、1961–1990年の気候値における1月の平年気温など。気候学的周期ここではまず年周期の一部を指定するには、暦年の中の日付を参照する。しかし暦年はよく定義された時間の単位ではない;なぜなら閏年と平年で長さが異なるし、暦法による差もあるからである。にもかかわらず実用的目的からは、我々は異なる暦法の月や季節による統計をあたかも同じものであるかのように比較したいし、閏年と平年の混在した暦の同じ日付を同じとみなして統計処理された気候値を作成したい。そこで我々は気候学的な年の中の日付を示すための特別の規約を用意した。気候統計は、何日かの範囲内での対応する部分から作られることもある;たとえば1997年4月の平均的1日における毎時の平均気温といったように。さらに二つの概念が同時に使われることもある;たとえば先の例で1997年4月ではなく1995–1999年の平均的4月における毎時の平均気温というようなものである。

気候学的変数は、気候学的時間軸を持つ。通常の時間軸のように、気候学的時間軸は大きさ1の次元を持つこともある(たとえば1961–1990年の1月の平均気温を含む変数)が、多くの場合は数個の要素を持つことになるだろう(たとえば1961–1990年の平均による各月の平均気温気候値については大きさ12の次元を、1月の平均気温を1961–1970, 1971–1980, 1981–1990年の平均について与えるならば大きさ3の次元を、また毎時の平年値を与えるならば大きさ24の次元が気候学的時間軸に用いられるだろう)。気候学的時間のインターバルは通常の時間のインターバルとは概念的に別のものである;ひとつの気候学的時間インターバルは、通常時間上の必ずしも連続しないいくつかのサブインターバルの集合を表わしている。この違いを示すために、気候学的時間の座標変数は `bounds` 属性をもたない。かわりに、`climatology` 属性をもち、それは大きさ (*n*, 2) の変数の名前を指す;ここで *n* は気候学的時間軸の次元長である。以下この変数を気候値変数と呼ぶことにする。時間の座標変数の `units` 及び `calendar` 属性を使った表現で、気候値変数の要素 (*i*, 0) は時間次元の *i* 番目のインデックスをもつ気候統計を評価するにあたり使われた最初のサブインターバルの始めを指示し、要素 (*i*, 1) は最後のサブインターバルの終わりを指示する。時間座標の値は気候学的時間インターバルの代表値であるべきである;気候学的時間を認識しないアプリケーションがそれにもかかわらず妥当な解釈をできるように。

COARDS 標準は気候学的時間について限られたサポートしか提供しない。COARDS との互換性のためには、時間座標はその `units` 属性が時間であり、参照時刻が0年1月1日正子すなわち udunits 文法で `"since 0-1-1"` であり、なおかつ現実世界の暦法を参照するときに気候学的と認識されるべきである。我々はこの規約を勧告しない;なぜなら (a) 気候値を計算するときに使われたインターバルについての情報をまったく与えない、そして (b) 参照時刻に 0 年を用いたとき、(0年は存在しないのでソフトウェアパッケージ間で時間座標の解釈の相違がありえて) 西暦 1 年以降の日付をどのように表現するかの標準がないからである。非現実世界の暦法では0年もあり得るかもしれないので、そのような場合には気候学的時間であることを知らせるために0年を使うことはできない。

気候学的時間軸には年々と、1年内と、1日内との変動を表現するために別の統計的方法を使うことがある。たとえば、1月の平均気温の気候値は、1年内と年々との両方の平均によって得られる。これは1月の最高気温の平年値とも、1月の平均気温の最大値とも違う。前者のためには、そ

[64]訳注:原文は recommend that と should が併用されている変な文ではあるが、これにより should と recommend が同程度の勧告であることがよくわかる。

れぞれの1月について最高気温を計算し、次にこれを平均する;後者のためには、それぞれの1月について平均気温を計算し、次にこれの最大をとる。通常のように、統計的操作は cell_methods 属性に記録され、そこには気候学的時間次元について二つまたは三つのエントリがありうる。

属性 cell_methods の有効な値は次のリストのひとつの形式でなければならない。多様な統計的メソッドが適用されるインターバルは、属性 climatology で指名される変数に記録されている気候学的時間のセル境界の日時の指定を分解することで決定される。(日時の指定は、「時間間隔 since 参照日時」の単位で表現された時間座標から計算されなければならない。)以下の記述では、年月日時分秒をそれぞれ *y, m, d, H, M, S* で略記し、早い境界と遅い境界とをそれぞれ *0* と *1* (ゼロとイチ)をつけて(y0, y1 のように)表わす。

形式 "time: *method1* within years time: *method2* over years"
それぞれの年の中で時間のインターバル (*mdHMS0-mdHMS1*) に *method1* が適用された上で、(*y0-y1*) の範囲の年について *method2* が適用される。

形式 "time: *method1* within days time: *method2* over days"
それぞれの日の中で時間のインターバル (*MHS0-HMS1*) に *method1* が適用された上で、(*ymd0-ymd1*) のインターバル内の日について *method2* が適用される。

形式 "time: *method1* within days time: *method2* over days time: *method3* over years"
それぞれの日の中で時間のインターバル (*MHS0-HMS1*) に *method1* が適用された上で、(*md0-md1*) のインターバル内の日について *method2* が適用され、ついで (*y0-y1*) の範囲の年について *method3* が適用される。

メソッドとして指定できるものは付録 E「セルメソッド」で列挙されているものであり、通常のものと同様、属性 cell_methods の各エントリはメソッド名のあとに括弧で囲んだ非標準情報[65]を含みうる。たとえば、ENSO の年についての平均は "time: mean over years (ENSO years)" のように示すことが出来よう。

年内のインターバルを考慮するときは、もし早い方の気候学的時間境界が月日としては遅いほうの気候学的時間境界より遅いならば、個々の年の時間インターバルは1月1日から次の年に続くことを意味する;たとえば12月1日0:00からの DJF インターバルは3月1日0:00まで続く。毎日のインターバルについても真夜中を越えて次の日に続くようなものには同様な状況が発生する。

日内のインターバルを考慮するときは、もし早い方の境界時間と遅いほうの境界時間が同じならば、そのメソッドは24時間全域に適用される。

理解しやすくするため、この節の例では、全ての時間座標値を数値型から日時の形式の文字列に翻訳した。これは現行の有効な CDL 文法ではない。

例7.8. 気候学的季節

この例は、1960年3月から1991年2月までのデータで作られた、4つの標準的な気候学的季節である MAM, JJA, SON, および DJF についての季節内最低気温の平年値のメタデータを示す。

```
dimensions:
```

[65]訳注:カッコ内に認められる標準化情報 interval について書いていないが、禁止されるわけでもあるまい。

```
  time=4;
  nv=2;
variables:
  float temperature(time,lat,lon);
    temperature:long_name="surface air temperature";

temperature:cell_methods="time: minimum within years time: mean ove
r years";
    temperature:units="K";
  double time(time);
    time:climatology="climatology_bounds";
    time:units="days since 1960-1-1";
  double climatology_bounds(time,nv);
data:  // 時間座標は日時形式に翻訳されている
  time="1960-4-16", "1960-7-16", "1960-10-16", "1961-1-16" ;
  climatology_bounds="1960-3-1",  "1990-6-1",
                     "1960-6-1",  "1990-9-1",
                     "1960-9-1",  "1990-12-1",
                     "1960-12-1", "1991-3-1" ;
```

例7.9. 1月についての10か年平均

1月の合計降水量のそれぞれ1961–1970, 1971–1980, 1981–1990 の10か年についての平均。

```
dimensions:
  time=3;
  nv=2;
variables:
  float precipitation(time,lat,lon);
    precipitation:long_name="precipitation amount";
    precipitation:cell_methods="time: sum within years
time: mean over years";
    precipitation:units="kg m-2";
  double time(time);
    time:climatology="climatology_bounds";
    time:units="days since 1901-1-1";
  double climatology_bounds(time,nv);
data:  // 時間座標は日時形式に翻訳されている
  time="1965-1-15", "1975-1-15", "1985-1-15" ;
  climatology_bounds="1961-1-1", "1970-2-1",
                     "1971-1-1", "1980-2-1",
                     "1981-1-1", "1990-2-1" ;
```

例7.10. 毎時の気温の平年値

1997年4月について、毎時の平均気温を与える。

```
dimensions:
  time=24;
  nv=2;
variables:
  float temperature(time,lat,lon);
    temperature:long_name="surface air temperature";
    temperature:cell_methods="time: mean within days time: mean
over days";
    temperature:units="K";
  double time(time);
    time:climatology="climatology_bounds";
    time:units="hours since 1997-4-1";
  double climatology_bounds(time,nv);
data:  // 時間座標は日時形式に翻訳されている
```

```
  time="1997-4-1 0:30", "1997-4-1 1:30", ... "1997-4-1 23:30" ;
  climatology_bounds="1997-4-1 0:00",  "1997-4-30 1:00",
                     "1997-4-1 1:00",  "1997-4-30 2:00",
                      ...
                      "1997-4-1 23:00", "1997-5-1 0:00" ;
```

例7.11. 典型的な気候値の1日における毎時の気温

これは前の例を変更したもので、1961–1990年気候値についてのものである。

```
variables:
  float temperature(time,lat,lon);
    temperature:long_name="surface air temperature";
    temperature:cell_methods="time: mean within days ",
      "time: mean over days time: mean over years";
    temperature:units="K";
  double time(time);
    time:climatology="climatology_bounds";
    time:units="days since 1961-1-1";
  double climatology_bounds(time,nv);
data:  // 時間座標は日時形式に翻訳されている
  time="1961-4-1 0:30", "1961-4-1 1:30", ..., "1961-4-1 23:30" ;
  climatology_bounds="1961-4-1 0:00", "1990-4-30 1:00",
                     "1961-4-1 1:00", "1990-4-30 2:00",
                     ...
                     "1961-4-1 23:00", "1990-5-1 0:00" ;
```

例7.12. 日降水量の月別最大値

2000年6月、7月、及び8月の3ヶ月について、日降水量の月別最大値を与える。統計の対象たる最初の日降水量は、6月1日の午前6時から6月2日の午前6時までであり、6月分の最後の日降水量は66、6月30日午前6時から、7月1日午前6時までである。これらの30の値の最大値が配列 precipitation のインデックス0に保存される。

```
dimensions:
  time=3;
  nv=2;
variables:
  float precipitation(time,lat,lon);
    precipitation:long_name="Accumulated precipitation";
    precipitation:cell_methods="time: sum within days time: maximum
over days";
    precipitation:units="kg";
  double time(time);
    time:climatology="climatology_bounds";
    time:units="days since 2000-6-1";
  double climatology_bounds(time,nv);
data:  // 時間座標は日時形式に翻訳されている
  time="2000-6-16", "2000-7-16", "2000-8-16" ;
  climatology_bounds="2000-6-1 6:00:00", "2000-7-1 6:00:00",
                     "2000-7-1 6:00:00", "2000-8-1 6:00:00",
                     "2000-8-1 6:00:00", "2000-9-1 6:00:00" ;
```

第8章　　データセットの大きさの縮減

データセットの大きさを縮減する2つの方法がある:パッキングと圧縮である。ここでいうパッキン

[66]訳注：ここはどうしても補わないと意味が通らない。

グとは、精度を落とすような方法でのデータの変更である。ここでいう圧縮とは、データをより効率的に格納しつつ、精度を落とさない技術である。圧縮はある種の状況でだけ機能する;たとえば変数が十分な量の欠損値あるいは繰り返しデータ値を含んでいるときなどである。この場合に標準的なツール、たとえば UNIX の compress や GNU の gzip などを使ってファイル[67]を書き出し終えてから圧縮することが可能である。この節では、変数ごとに適用できるもうひとつの圧縮方法を提供する。この方法は、ただ一つの変数が展開できればよい場合に有利である。他面、CF 規約を理解しない汎用ツールが圧縮された変数を扱えなくなることが欠点である。

8.1 パックされたデータ

現時点での netCDF インターフェイスはデータのパッキングを行わない。しかしながら、NUG で定義された属性 `scale_factor` 及び `add_offset` を用いることで単純なパッキングを実現できる。変数のデータ値を読み取った後で、属性 `scale_factor` の値を乗算し、属性 `add_offset` の値を加算するものである。もし両方の属性が存在すれば、データにオフセットを加算する前にスケールの乗算を行う。スケールするデータを書き出すときには、アプリケーションはまずオフセットを減算し、ついでスケールファクターで除算をすべきである。変数の単位はアンパックしたデータを表わしている。

この標準では `scale_factor` 及び `add_offset` 属性の使い方に関して NUG より制約を課している;この制約によって、データ型の変換に伴う曖昧性や精度の問題が解決される。もし属性 `scale_factor` や `add_offset` が関連する変数と同じデータ型ならば、アンパックされたデータはパックされたデータと同じ型であることが想定される。しかし、もし属性 `scale_factor` や `add_offset` が(パックデータを含む)変数と異なるデータ型ならば、アンパック後のデータはこれらの属性の型に一致しなければならず、このとき両方の属性がともに `float` 型または `double` 型でなければならない。この場合もう一つの制約として、パックデータを保持する変数は `byte`, `short`, または `int` 型でなければならない。ただし `int` を `float` にアンパックすることは精度を落とす懸念があることから避けたほうがよい。

パックすべきデータに欠損値が含まれているとき、欠損値を表わす属性(`_FillValue`, `valid_min`, `valid_max`, `valid_range`)はパックデータと同じ型でなければならない。欠損値とパッキングとの両方を示す属性をもつ変数をアプリケーションがどう扱うべきかについては 2.5.1「欠損値」で論じたので参照せよ。

8.2 集約による圧縮

NetCDF ファイルのスペースを節約するため、データ配列の中で常に欠損となる点を除去することが望ましいことがある。そのような圧縮は一つまたはそれ以上の隣接する軸について行うことができて、保存すべき点のリストを参照しつつ実施される。そのリストは圧縮すべき次元だけを含んだマスク配列を、並び替えをせずに一次元に写像することで構成される。このリストは必要な点の一次元的マスクの中でのインデックスの集まりとなる。圧縮された配列では、すべての圧縮すべき次元が一つの軸に置き換えられ、その長さは必要な点の数である。この次元に沿った必要な点は、圧縮を展開した後の配列と同じ順番で、不要な点を飛ばして並んでいる。このように、保存すべき点のリストを順に処理することで圧縮も展開も実行される。

保存すべき点のリストは、データ配列の圧縮された軸に対する座標変数に保存される。したがって、リスト変数とその次元は同じ名前である。リスト変数には文字列の `compress` 属性があり、そこには***空白区切りのリストで、圧縮で処理される次元の名前を展開後の配列の CDL 宣言の***

[67]訳注:ここは「ファイル」を「データセット」に置き換えては意味が通らない。

順に持つ。この属性があることによってリスト変数は識別される。保存すべき点のリスト、圧縮前の次元と座標変数（境界変数を含め）、そして展開後に持つべき属性の全てを付した圧縮された変数の全てが netCDF ファイルに書かれる。この情報を使って、展開された変数は正確に再建できる。

例8.1. 三次元配列の水平圧縮

経度-緯度-深さという次元を持つ土壌温度の配列のうち、海洋の点を全ての深さについて除去する。この場合、経度及び緯度の軸だけが圧縮で処理される。陸地の点のインデックスを集めてリスト `landpoint(landpoint)` を作る。

```
dimensions:
  lat=73;
  lon=96;
  landpoint=2381;
  depth=4;
variables:
  int landpoint(landpoint);
    landpoint:compress="lat lon";
  float landsoilt(depth,landpoint);
    landsoilt:long_name="soil temperature";
    landsoilt:units="K";
  float depth(depth);
  float lat(lat);
  float lon(lon);
data:
  landpoint=363, 364, 365, ...;
```

たとえば `landpoint(0) = 363` なので、`landsoilt(*, 0)` はオリジナル（展開後）のデータの中の 363 番目の点に写像されることがわかる。オリジナルデータは `(lat, lon)` という次元を持ち、363 = 3･96 + 75 なので、これはインデックス (3, 75) に相当する。

例8.2. 三次元場の圧縮

経度-緯度-深さという次元を持つ塩分濃度のデータを、海洋底の下となる点を除去して圧縮する。この場合、深くなるにつれてアクティブな海洋の点は順次少なくなるので、3次元の全てが圧縮で処理される。

```
variables:
  float salinity(time,oceanpoint);
  int oceanpoint(oceanpoint);
    oceanpoint:compress="depth lat lon";
  float depth(depth);
  float lat(lat);
  float lon(lon);
  double time(time);
```

上の情報は、塩分濃度場は `(depth, lat, lon)` という次元をもつ配列に展開すべきことを意味している。

付録

付録A　　属性

格子写像を記述するもの以外、すべての CF の属性を次に示す。格子写像に関する属性は付録 F を参照せよ。

表中、「種類」欄で「**文**」は文字型、「**数**」は数値型、「**同**」はデータ変数と同じ型である。「利用」欄で「**大**」は大域属性、「**軸**」は座標データを含む変数すなわち座標変数、スカラー座標変数、および補助座標変数、「**デ**」は座標データ以外の変数を指す。「参照」欄は属性のオリジナルの定義（最初のリンク）および（要すれば二番目として）この文書で属性が議論された場所である。

表 A.1: 属性

属性名	種類	利用	参照	説明
add_offset	数	デ	NUG(8.1)[68], 8.1「パックされたデータ」	もし変数にこの属性があれば、アプリケーションがデータを読んだ後にこの数値を加算すべきである。もし scale_factor と add_offset の両方があれば、まずデータにスケールを乗じてからオフセットを加算する。
ancillary_variables	文	デ	3.4「補助データ」	属性を持つ変数に密接に関連したデータを持つ変数（たとえば測器データの計測不確定性など）を識別する。
axis	文	軸	19「座標の種別」	緯度、経度、鉛直、あるいは時間軸を識別する。
bounds	文	軸	35「セル境界」	境界変数を識別する。
calendar	文	軸	23「暦法」	時間軸をエンコードする際に使った暦法
cell_measures	文	デ	37「セルの測度」	セルの面積又は体積を格納する変数を識別する。
cell_methods	文	デ	39「セル・メソッド」、45「気候学的統計」	セルの値が代表するデータを構成するのに使われたメソッドを記録する。
climatology	文	軸	45「気候学的統計」	気候変数を識別する。
comment	文	大、デ	13「ファイルの内容の記述」	変数値を作成するにあたり使ったデータ又は手法に関する種々の情報。
compress	文	軸	49「集約による圧縮」	集約によって圧縮された次元を記録する。
Conventions	文	大	NUG(8.1)	データセットが従う規約の名前。
coordinates	文	デ	25「座標系」、33「ラベルと代替座標」	補助座標変数、ラベル変数、及び代替座標変数を識別する。
_FillValue	同	デ	NUG(8.1)	欠損あるいは見て意義の data を表現するために用いられる値。
flag_masks	同	デ	17「フラグ」	ブール型または列挙型のフラグを表現するビットフィールドのリストを提供する。

[68]訳注： NUG の後に全部 (8.1) と書いてあるが、これは第2.4版の章立てで、現行（第4.0.1版）では Appendix B <URL:http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/Attribute-Conventions.html#Attribute-Conventions> とするのが適当。

属性名	種類	利用	参照	説明
flag_meanings	文	デ	17「フラグ」	属性 flag_values とともにそれぞれのフラグ値を記述する語句を提供する。句の中の語はアンダースコアで連結されるべき。
flag_values	同	デ	17「フラグ」	フラグの値のリストを提供する。属性 flag_meanings とともに使う。
formula_terms	文	軸	21「無次元鉛直座標」	式中の項に対応する変数を識別する。
grid_mapping	文	デ	29「水平の座標参照系、格子写像、および投影法」	格子写像を定義する変数を識別する。
history	文	大	NUG(8.1)	元データを変更したアプリケーションのリスト。
institution	文	大、デ	13「ファイルの内容の記述」	元データが製造されたところ。
leap_month	数	軸	23「暦法」	利用者定義の暦法においてうるう年のどの月を1日延長するかを指定する。
leap_year	数	軸	23「暦法」	利用者定義の暦法においてうるう年の例を与える。この年と4の倍数異なる全ての年が閏年と仮定される。
long_name	文	軸、デ	NUG(8.1), 15「長い名前」	変数の内容を示す記述的な名前。この名前は標準化されていない。
missing_value	同	デ	11「欠損値」	欠損又は見て意義のデータをあらわすのに使われる値（NUG では廃止予定扱い）
month_lengths	数	軸	23「暦法」	利用者定義の暦法において、非閏年の各月の長さを指定する。
positive	文	軸	[COARDS]	鉛直座標値の増える方向。
references	文	大、デ	13「ファイルの内容の記述」	変数又はデータセットを作成するために使われたデータ又は手法を記述する参照。
scale_factor	数	デ	NUG(8.1), 49「パックされたデータ」	もしあれば、アプリケーションがデータを読んだ後でこの係数を掛けるべきである。属性 add_offset も参照。
source	文	大	13「ファイルの内容の	元データの作成方法。

属性名	種類	利用	参照	説明
		、デ	記述」	
standard_ error_ multiplier	数	デ	56「標準名修飾子」	もし標準名修飾子 standard_error のデータ変数にこの属性があれば、値は標準誤差の倍数として書かれていることを示す。
standard_name	文	軸、デ	15「標準名」	標準名表の中で変数の内容の記述を参照する標準名。
title	文	大	NUG(8.1)	ファイル内容の短い記述。
units	文	軸、デ	NUG(8.1), 13「単位」	変数の内容の単位。
valid_max	数	軸、デ	NUG(8.1)	変数の最大の有効な値。
valid_min	数	軸、デ	NUG(8.1)	変数の最小の有効な値。
valid_range	数	軸、デ	NUG(8.1)	変数の最小と最大の有効な値。

付録B　　標準名表の形式

CF 標準名表は XML 文書である（すなわち、その書式は XML1.0勧告[XML]に従う）。XML の規則は人間と機会の可読性の間のリーズナブルな均衡を与え、また国際化もサポートする。更なる情報については W3C ホームページ[W3C]を参照せよ。

文書は XML ファイルであることを識別するヘッダから始まる。

```
<?xml version="1.0"?>
```

次に標準名表をタグ<standard_name_table> および </standard_name_table> で囲んで続ける。

```
  <standard_name_table
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="CFStandardNameTable.xsd">
```

タグで<standard_name_table> で区切られた中には、次のものが順に入る:

```
<institution>機関の名前</institution>
<contact>コンタクトパーソンの電子メールアドレス</contact>
```

続いて、entry 要素の並びが、必須ではないが alias 要素の並びを伴って続く。要素 entry 及び alias は次の形を取る:

```
<entry id="an_id">
   値 an_id を standard_name 属性として持つ変数の定義
</entry>
<alias id="another_id">
   値 another_id を standard_name 属性として持つ変数への別名
</alias>
```

タグ entry 及び alias に現れる id 属性は大文字と小文字を区別し、空白を含まない文字列で、表の中でエントリを一意に識別する。***この値が変数の*** standard_name ***属性として使われる***。

要素 entry の目的は id 文字列の定義を与えることである。それぞれの entry 要素には次の要素が含まれる:

```
<entry id="an_id">
  <canonical_units>変数の代表的な単位</canonical_units>
  <description>変数の記述</description>
</entry>
```

また必須ではないが次の要素を含むことができる:

```
<grib>GRIB パラメタ番号</grib>
<amip>AMIP 識別文字列</amip>
```

すべての変数が AMIP または GRIB のコードを持っているわけではない。ECMWF の GRIB コードは E で始まり、NCEP のコードは N で始まる。標準コード(範囲1–127)は単に数値が書かれる。変数に等価な GRIB コードが複数あるとき、空白区切りのリストで表わされる。

要素 alias には定義は含まれない。そのかわり、定義のために見出すべき element 要素の id 属性値が含まれている。要素 alias の目的は、表の保守にあたって後方互換性を維持するため、改名前の古い名前など冗長な定義を別名に変換する手段を提供することにある。属性 standard_name の文字列の局所的な命名慣例を記載するために alias 要素を使うことは意図されていない。それぞれの alias 要素は単一の要素を含む:

```
<alias id="an_id">
<entry_id>定義のあるエントリの識別子</entry_id>
</alias>
```

例 B.1. 三つのエントリを持つ標準名表

```
<?xml version="1.0"?>
  <standard_name_table>
    <institution>Program for Climate Model Diagnosis
     and Intercomparison</institution>
    <contact>support@pcmdi.llnl.gov</contact>
    <entry id="surface_air_pressure">
      <canonical_units>Pa</canonical_units>
      <grib>E134</grib>
      <amip>ps</amip>
      <description>
          The surface called "surface" means
        the lower boundary of the atmosphere.
```

```
      </description>
    </entry>
    <entry id="air_pressure_at_sea_level">
      <canonical_units>Pa</canonical_units>
      <grib>2 E151</grib>
      <amip>psl</amip>
      <description>
          Air pressure at sea level is the quantity
        often abbreviated as MSLP or PMSL.
        sea_level means mean sea level,
        which is close to the geoid in sea areas.
      </description>
    </entry>
    <alias id="mean_sea_level_pressure">
      <entry_id>air_pressure_at_sea_level</entry_id>
    </alias>
  </standard_name_table>
```

属性 standard_name が surface_air_pressure となっている変数の定義は直接見出せる;id="surface_air_pressure" となっている要素は定義を含んだ entry 要素だからである。

属性 standard_name が mean_sea_level_pressure となっている変数の定義は間接的に見出される;まず id="mean_sea_level_pressure" となっている要素を見出し、ついでそれが別名要素なので、その entry_id タグの値で示されているように id="air_pressure_at_sea_level" となっている要素を探す。

将来タグが追加される可能性はある。標準名表を解読するアプリケーションは認識できないタグを安全のため無害に無視するように書かれるべきである。

付録C　　標準名修飾子

単位欄で *u* は修飾なしの標準名と次元的に同等な単位を示す。

表 C.1. 標準名修飾子

修飾子	単位	記述
detection_minimum	*u*	信号として検出可能とみなされる最小のデータ値。
number_of_observations	1	データ値が導出された離散的な観測又は計測の数。
standard_error	*u*	データ値の不確定性。標準エラーには系統的及び統計的な不確実性の両方を含む。デフォルトではこの属性で与えられるデータは標準エラーの1倍[69]であることが想定される。もし与えられた値が標準エラーの何倍かであるなら、standard_error 補助変数には属性 standard_error_multiplier として係数を与えるべきである。
status_flag		データ値の品質あるいは他の状態を示すフラグ値。この変数は

[69]原文は難解: By default it is assumed that the values supplied are for one standard error. If the values supplied are for some multiple of the standard error, the standard_error ancillary variable should have an attribute standard_error_multiplier stating the multiplication factor.

修飾子	単位	記述
		flag_values, flag_masks, またはその両方および flag_meanings 属性によってどのように解釈すべきかを示すべきである。

付録D　　無次元鉛直座標

ここで与えられる定義により、アプリケーションは無次元座標値及び関連する変数から有次元の座標値を計算できるようになる。その表式は格子点 (*n, k, j, i*) についての式である、ここで *i* 及び *j* は水平のインデックス、*k* は鉛直のインデックス、そして *n* は時間のインデックスである。座標変数は standard_name 属性値によりその定義と関連付けられる。定義の中の項は formula_terms 属性値によりファイル変数と結び付けられる。その formula_terms 属性は文字列値をとり、それは空白区切りで "*term*: *variable*" の形式の要素からなる、ここで *term* は定義の中の項を表わすキーワードであり、*variable* はその項の値を含む netCDF ファイル中の変数の名前である。要素の順序は重要でない。

格子点のインデックスはこの定義の正式な一部ではないが、ファイル変数にある***かもしれない***インデックスを描写するために含められている。たとえば、ある鉛直座標が時間を定義に含むとしても、それは必ずしもすべての netCDF ファイルで時間に依存するとは限らない。また、定義は一般的な形式で書かれており、一部の項を省くことで単純化されるかもしれない。属性 formula_terms から省かれた項はゼロと仮定されるべきである。

8.2.1...D.1 大気の自然対数圧力座標

```
standard_name = "atmosphere_ln_pressure_coordinate"
```

定義:

```
p(k) = p0 + exp(-lev(k))
```

ここで p(*k*) は格子点 *k* での気圧であり、p0 は参照気圧、lev(*k*) は鉛直格子点 k における無次元座標である。

属性 formula_terms の形式は次のようである。

```
formula_terms = "p0: var1 lev2: var2"
```

8.2.1...D.2 大気のシグマ座標

```
standard_name = "atmosphere_sigma_coordinate"
```

定義:

```
p(n,k,j,i) = ptop + sigma(k)*(ps(n,j,i)-ptop)
```

ここで p(*n, k, j, i*) は格子点 (*n, k, j, i*) の気圧、ptop はモデル上端の気圧、sigma(*k*) は鉛直格子点 (*k*) における無次元の座標、そして ps(*n, j, i*) は水平格子点 (*j, i*) 及び時間 (*n*) における地表面気圧である。

属性 formula_terms の形式は次のようである。

```
formula_terms = "sigma: var1 ps: var2 ptop: var3"
```

8.2.1...D.3 大気の混合シグマ圧力座標

```
standard_name = "atmosphere_hybrid_sigma_pressure_coordinate"
```

定義:

```
p(n,k,j,i) = a(k)*p0 + b(k)*ps(n,j,i)
```

あるいは

```
p(n,k,j,i) = ap(k) + b(k)*ps(n,j,i)
```

ここで p(*n*, *k*, *j*, *i*) は格子点 (*n*, *k*, *j*, *i*) の気圧であり、a(*k*) 又は ap(*k*) と b(*k*) はレベル *k* における混合座標の要素たる係数、p0 は参照気圧、そして ps(*n*, *j*, *i*) は水平格子点 (*j*, *i*) 及び時間 (*n*) の地表面気圧である。係数 a(*k*) と ap(*k*) のどちらを使うかはモデルの定式化に依存する。前者は無次元の比率、後者は気圧の値である。どちらの定式化においても、b(*k*) は無次元の比率である。

属性 formula_terms の形式は次のようである。

```
formula_terms = "a: var1 b: var2 ps: var3 p0: var4"
```

ここでもし該当する場合は a を ap に置き換える。

レベル *k* に対する混合シグマ圧力座標は、a(*k*) + b(*k*) または ap(*k*)/p0 + b(*k*) の適切なほうと定義される。

8.2.1...D.4 大気の混合高度座標

```
standard_name = "atmosphere_hybrid_height_coordinate"
```

定義:

```
z(n,k,j,i) = a(k) + b(k)*orog(n,j,i)
```

ここで z(*n*, *k*, *j*, *i*) は格子点 (*k*, *j*, *i*) 及び時間 (*n*) におけるジオイド（近似的には平均海面）からの高さ、orog(*n*, *j*, *i*) は (*j*, *i*) 及び time(*n*) における地表面のジオイド上の高さ、そして a(*k*) 及び b(*k*) は混合高度レベル *k* を定義する座標である。値 a(*k*) は高さの次元を持ち、b(*i*) は無次元である。

属性 formula_terms の形式は次のようである。

```
formula_terms = "a: var1 b: var2 orog: var3"
```

無次元の混合高度座標というものは存在しない。もしレベルに依存する定数が必要ならば、a(*k*) によって混合高度はもっともよく近似される。

8.2.1...D.5 大気の平滑レベル鉛直座標（SLEVE 座標）

```
standard_name = "atmosphere_sleve_coordinate"
```

定義:

```
z(n,k,j,i) = a(k)*ztop + b1(k)*zsurf1(n,j,i) + b2(k)*zsurf2(n,j,i)
```

ここで z(*n*,*k*,*j*,*i*) は格子点 (*k*, *j*, *i*) 及び時間 (*n*) におけるジオイド(近似的には平均海面)からの高さ、ztop はモデルの頂の高さ、そして a(*k*), b1(*k*) 及び b2(*k*) は混合レベル *k* を定義する無次元座標である。項 zsurf1(*n*,*j*,*i*) 及び zsurf2(*n*,*j*,*i*) はそれぞれ地形の大規模及び小規模な部分である。詳細は Shaer et al. [SCH02] を参照せよ。

属性 formula_terms の形式は次のようである。

```
formula_terms = "a: var1 b1: var2 b2: var3 ztop: var4 zsurf1: var5
                 zsurf2: var6"
```

レベル *k* に対する混合鉛直座標は a(*k*)*ztop と定義される。

8.2.1...D.6 海洋のシグマ座標

```
standard_name = "ocean_sigma_coordinate"
```

定義:

```
z(n,k,j,i) = eta(n,j,i) + sigma(k)*(depth(j,i)+eta(n,j,i))
```

ここで、z(*n*,*k*,*j*,*i*) は格子点 (*n*, *k*, *j*, *i*) における海洋の鉛直測地基準(即ち近似的には平均海面)に相対の高さ(上向きが正)、eta(*n*,*j*,*i*) は格子点 (*n*, *j*, *i*) における海洋の鉛直測地基準に相対の海面の高さ(上向きが正)、sigma(*k*) は鉛直格子点 (*k*) における無次元座標、そして depth(*j*,*i*) は格子点 (*j*, *i*) における海洋の鉛直測地基準と海底との距離(正の値)である。

属性 formula_terms の形式は次のようである。

```
formula_terms = "sigma: var1 eta: var2 depth: var3"
```

8.2.1...D.7 海洋の s-座標

```
standard_name = "ocean_s_coordinate"
```

定義:

```
z(n,k,j,i) = eta(n,j,i)*(1+s(k)) + depth_c*s(k) +
             (depth(j,i)-depth_c)*C(k)

C(k) = (1-b)*sinh(a*s(k))/sinh(a) +
       b*[tanh(a*(s(k)+0.5))/(2*tanh(0.5*a)) - 0.5]
```

ここで z(*n*,*k*,*j*,*i*) は格子点 (*n*, *k*, *j*, *i*) における海洋の鉛直測地基準(即ち近似的には平均海面)に相対の高さ(上向きが正)、eta(*n*,*j*,*i*) は格子点 (*n*, *j*, *i*) における海洋の鉛直測地基準に相対の海面の高さ(上向きが正)、s(*k*) は鉛直格子点 (*k*) における無次元座標、そして depth(*j*,*i*) は格子点 (*j*, *i*) における海洋の鉛直測地基準と海底との距離(正の値)である。定数 a, b, そして depth_c が座標の引き伸ばしを制御する。

属性 formula_terms の形式は次のようである。

```
formula_terms = "s: var1 eta: var2 depth: var3
```

```
                    a: var4 b: var5 depth_c: var6"
```

8.2.1...D.8 海洋のシグマ-z 座標

```
standard_name = "ocean_sigma_z_coordinate"
```

定義:

```
k ≦nsigma について:

 z(n,k,j,i) = eta(n,j,i)
            + sigma(k)*(min(depth_c,depth(j,i))+eta(n,j,i))

k > nsigma について:

 z(n,k,j,i) = zlev(k)
```

ここで、z(*n*,*k*,*j*,*i*) は格子点 (*n*, *k*, *j*, *i*) における海洋の鉛直測地基準(即ち近似的には平均海面)に相対の高さ(上向きが正)、eta(*n*,*j*,*i*) は格子点 (*n*, *j*, *i*) における海洋の鉛直測地基準に相対の海面の高さ(上向きが正)、sigma(*k*) は *k* ≦nsigma となる鉛直格子点 (*k*) については無次元座標、そして depth(*j*,*i*) は格子点 (*j*, *i*) における海洋の鉛直測地基準と海底との距離(正の値)である。深さ depth_c より上に nsigma 個の層がある。

属性 formula_terms の形式は次のようである。

```
formula_terms = "sigma: var1 eta: var2 depth: var3
            depth_c: var4 nsigma: var5 zlev: var6"
```

8.2.1...D.9 海洋の二重シグマ座標

```
standard_name = "ocean_double_sigma_coordinate"
```

定義:

```
k ≦k_c について

 z(k,j,i)= sigma(k)*f(j,i)

k > k_c について

 z(k,j,i)= f(j,i) + (sigma(k)-1)*(depth(j,i)-f(j,i))

 f(j,i)= 0.5*(z1+ z2)
       + 0.5*(z1 - z2)* tanh(2*a/(z1-z2)*(depth(j,i)-href))
```

ここで、z(*k*,*j*,*i*) は格子点 (*k*, *j*, *i*) における海洋の鉛直測地基準(即ち近似的には平均海面)に相対の高さ(上向きが正)、sigma(*k*) は *k* ≦k_c となる鉛直格子点 (*k*) については無次元座標、そして depth(*j*,*i*) は格子点 (*j*, *i*) における海洋の鉛直測地基準と海底との距離(正の値)である。項 z1, z2, a, 及び href は定数である。

属性 formula_terms の形式は次のようである。

```
formula_terms = "sigma: var1 depth: var2 z1: var3 z2: var4
```

```
a: var5 href: var6 k_c: var7"
```

付録E　　セルメソッド

単位欄で、*u* はメソッドを適用する前の物理量の単位を示す。

表 E.1. セルメソッド

セルメソッド	単位	記述
point	*u*	データ値は空間又は時間の中の一点（瞬間）を代表する。示された次元に関して示強性の量については、これが既定のメソッドである。
sum	*u*	データ値はセル全体にわたっての合計または積算を表わす。示された次元について示量性の量については、これが規定のメソッドである。
maximum	*u*	最大値
median	*u*	中位数
mid_range	*u*	最大と最小の平均
minimum	*u*	最小値
mean	*u*	平均値
mode	*u*	最頻値
standard_ deviation	*u*	標準偏差
variance	u^2	分散

付録F　　格子写像

格子写像として認識されるものについて、以下の各節で記述する。それぞれの節では次のものを挙げる: 属性 grid_mapping_name として使える有効な名前、写像のパラメタに値を割り当てるために用いられる属性のリスト、写像された格子系の独立変数を含む座標変数を識別するための標準名、写像の定義や写像を使うに当たって有益な情報へのリファレンス。写像のパラメタを設定するために使われる属性はいくつかの写像で共有されるので、その定義は最終節の表にまとめた。どの格子写像についても、適宜、楕円体と本初子午線を記述する属性[70]を含めることができる。

われわれは grid_mapping_name 属性の値と、地図投影法を記述するパラメタのための属性名を選ぶにあたり、米連邦空間データ委員会「デジタル地理空間メタデータの内容標準」[FGDC]を参考にした。

8.2.1...F.1 アルベルス正積図法

```
grid_mapping_name = albers_conical_equal_area
```

地図パラメタ:

- standard_parallel – スカラーあるいは長さ2のベクタでありうる

[70]訳注：具体的には earth_radius, inverse_flattening, longitude_of_prime_meridian, semi_major_axis, semi_minor_axis のことと解される。

- longitude_of_central_meridian
- latitude_of_projection_origin
- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説:
<URL:http://www.remotesensing.org/geotiff/proj_list/albers_equal_area_conic.html>

8.2.1...F.2 正距方位図法

```
grid_mapping_name = azimuthal_equidistant
```

地図パラメタ:

- longitude_of_projection_origin
- latitude_of_projection_origin
- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説:
<URL:http://www.remotesensing.org/geotiff/proj_list/azimuthal_equidistant.html>

8.2.1...F.3 ランベルト正積方位図法

```
grid_mapping_name = lambert_azimuthal_equal_area
```

地図パラメタ:

- longitude_of_projection_origin
- latitude_of_projection_origin
- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説:
<URL:http://www.remotesensing.org/geotiff/proj_list/lambert_azimuthal_equal_area.html>

8.2.1...F.4 ランベルト正角方位図法

```
grid_mapping_name = lambert_conformal_conic
```

地図パラメタ:

- standard_parallel – スカラーあるいは長さ2のベクタでありうる
- longitude_of_central_meridian
- latitude_of_projection_origin
- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説:
<URL:http://www.remotesensing.org/geotiff/proj_list/lambert_conic_conformal_2sp.html>

8.2.1...F.5 ランベルト正積円筒図法

```
grid_mapping_name = lambert_cylindrical_equal_area
```

地図パラメタ:

- longitude_of_central_meridian
- standard_parallel 又は scale_factor_at_projection_origin のどちらか
- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説は
<URL:http://www.remotesensing.org/geotiff/proj_list/cylindrical_equal_area.html> にある(“Lambert Cylindrical Equal Area” あるいは EPSG 9834 または EPSG 9835)。詳細な式は Snyder (1987) [Snyder] の 76-85 ページにある。

8.2.1...F.6 経緯度座標

```
grid_mapping_name = latitude_longitude
```

この格子写像は球体地球上の緯度と経度に基づく古典的な二次元地理座標系を定義する。この格子写像が定義されているのは、地球の形を記述できるようにするためである。

地図パラメタ:

なし

地図座標:

直交直線座標はこの規約(4.1「緯度座標」及び20「経度座標」)に従って識別される経緯度である。

8.2.1...F.7 メルカトル図法

```
grid_mapping_name = mercator
```

地図パラメタ:

- longitude_of_projection_origin
- standard_parallel（EPSG 9805）又は scale_factor_at_projection_origin（EPSG 9804）のどちらか
- false_easting
- false_northing

地図座標:

直交直線座標の x（横軸）及び y（縦軸）座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説は <URL:http://www.remotesensing.org/geotiff/proj_list/mercator_1sp.html> （“Mercator (1SP)” または EPSG 9804）あるいは <URL:http://www.remotesensing.org/geotiff/proj_list/mercator_2sp.html>（“Mercator (2SP)” または EPSG 9805）にある。
数式に関する更なる情報は [OGP/EPSG_GN7_2] で得られる。

8.2.1...F.8 正射図法

```
grid_mapping_name = orthographic
```

地図パラメタ:

- longitude_of_projection_origin
- latitude_of_projection_origin
- false_easting
- false_northing

地図座標:

直交直線座標の x（横軸）及び y（縦軸）座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説は <URL:http://www.remotesensing.org/geotiff/proj_list/orthographic.html> にある。
数式に関する更なる情報は [OGP/EPSG_GN7_2] で得られる。

8.2.1...F.9 ポーラーステレオ図法

```
grid_mapping_name = polar_stereographic
```

地図パラメタ:

- straight_vertical_longitude_from_pole
- latitude_of_projection_origin — +90.0 または -90.0 のどちらか
- standard_parallel 又は scale_factor_at_projection_origin のどちらか

- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像を計算するための PROJ.4 ソフトウェアの解説は <URL:http://www.remotesensing.org/geotiff/proj_list/polar_stereographic.html> にある。

8.2.1...F.10 回転された極に対する経緯度座標

```
grid_mapping_name = rotated_latitude_longitude
```

地図パラメタ:

- grid_north_pole_latitude
- grid_north_pole_longitude
- north_pole_grid_longitude - オプション(規定値は0)

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

8.2.1...F.11 ステレオ図法(平射図法)

```
grid_mapping_name = stereographic
```

地図パラメタ:

- longitude_of_projection_origin
- latitude_of_projection_origin
- scale_factor_at_projection_origin
- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像と逆写像を計算するための PROJ.4 ソフトウェアの解説は <URL:http://www.remotesensing.org/geotiff/proj_list/stereographic.html> にある。投影法の原点が両極のどちらかとなる特別な場合については “Polar stereographic" の節を参照せよ。

8.2.1...F.12 横軸メルカトル図法

```
grid_mapping_name = transverse_mercator
```

地図パラメタ:

- scale_factor_at_central_meridian
- longitude_of_central_meridian

- latitude_of_projection_origin
- false_easting
- false_northing

地図座標:

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記:

この写像と逆写像を計算するための PROJ.4 ソフトウェアの解説は <URL:http://www.remotesensing.org/geotiff/proj_list/transverse_mercator.html> にある。

8.2.1...F.13 鉛直透視図法

```
grid_mapping_name = vertical_perspective
```

地図パラメタ

- latitude_of_projection_origin
- longitude_of_projection_origin
- perspective_point_height
- false_easting
- false_northing

地図座標

直交直線座標の x(横軸)及び y(縦軸)座標は、standard_name 属性値がそれぞれ projection_x_coordinate 及び projection_y_coordinate であることによって識別される。

注記

この写像と逆写像を計算するための PROJ.4 ソフトウェアの解説は <URL:http://www.remotesensing.org/geotiff/proj_list/geos.html> にある。この解説では視点は赤道の直上にあることを想定している。より一般的な鉛直透視図法[71]の記述は Snyder (1987) [Snyder] の 169–181 ページにある。

8.2.1...F.14 格子写像で用いられる属性の表

次表の「型」欄で「**文**」は文字列、「**数**」は数値である。

表 F.1. 格子写像属性

属性	型	記述
earth_radius	**数**	地球の形を近似するのに使った球体の半径をメートル単位で指定するのに使う。この属性は、X-Y デカルト座標を得るために地球球体近似を使った投影座標参照系について指定されるべきである。もしデカルト座標が楕円体を使って導出されたのであれば、この属性は定義されるべきではない。例: "6371007" これは GRS 1980 等積球体[72]

[71]訳注：透視図法とは、地球、視点、及び投影面の幾何光学的関係で定義される地図投影法全般の名称であり、本節で定義されている図法は本来「横軸外射方位図法」transverse external perspective azimuthal projection というべきものである。誰も言わないけど。

[72]訳注：等積球体は原文 authalic sphere. 回転楕円体と同じ表面積を持つ球体のこと。

属性	型	記述
		の半径である。[73]
false_easting	**数**	地図投影の直行直線座標の横軸の座標値には、すべてこの値が加算される。この値は負の座標値を回避するためにしばしば割り当てられる。標準名 projection_x_coordinate で識別される座標変数の単位で表わされる。
false_northing	**数**	地図投影の直行直線座標の縦軸の座標値には、すべてこの値が加算される。この値は負の座標値を回避するためにしばしば割り当てられる。標準名 projection_y_coordinate で識別される座標変数の単位で表わされる。
grid_mapping_name	**文**[74]	格子写像を識別するために使われる名前。
grid_north_pole_latitude	**数**	回転された格子の経緯度座標系で北極とする点の真の緯度（degrees_north 単位）。[75]
grid_north_pole_longitude	**数**	回転された格子の経緯度座標系で北極とする点の真の経度（degrees_east 単位）。[76]
inverse_flattening	**数**	測地系で地球の形を近似するために使われている楕円体の扁平率の**逆数**（1/*f*）を指定するのに使う。楕円体の扁平率（*f*）は長半径（*a*）及び短半径（*b*）との間に $f = (a-b)/a$ という関係がある。地球を球体とする場合はこの属性は省かれるかゼロに設定されるべきである。例：GRS 1980 楕円体について 298.257222101.（注：慣用で楕円体の形状は長半径と短半径、または長半径と扁平率の逆数で指定される。もしこれら3つの属性が指定されるならば、その値は上述の式と整合したものでなければならない。）
latitude_of_projection_origin	**数**	地図投影された空間上で直交直線座標の原点として選ばれた点の緯度。範囲： −90.0 ≦ latitude_of_projection_origin ≦ 90.0
longitude_of_central_meridian	**数**	投影法を構成する基礎[77]として使われる、地図投影の中央の経線。範囲：−180.0 ≦ longitude_of_central_meridian ≦ 180.0
longitude_of_prime_meridian	**数**	測地系での本初子午線の経度をグリニッジ起点で指定する。本初子午線は経度の値を決定する起点を定義する。地図投影法の原点の

[73]訳注：原文では範囲を明示していないが常識的に正の値でなければならない。
[74]訳注：原文では数値型となっているが明らかに文字列でなければならない。
[75]訳注：原文では範囲を明示していないが常識的に緯度 φ は −90 ≦φ≦90 でなければならない。
[76]訳注：原文では範囲を明示していないため、他の経度 λ に関する属性と同様に −180 ≦λ≦180 で制約されているのだとは思われるが、360までの値が認められるのか必ずしも自明ではない。
[77]訳注：この規約では中央経線の定義される格子写像（円錐図法と横メルカトル図法）については原点経度の定義を求めていないので、次注も考え合わせると、中央経線が原点経度を兼ねるのだろう。GRIB では両者は区別されるので、変換が難しいことになる。

属性	型	記述
		経度を定義する投影原点経度と混同しないように(別名を中央経線ともいう[78]longitude_of_projection_origin 参照)。範囲: 十進数[79]の度単位で −180.0 ≦longitude_of_prime_meridian ≦180.0. 既定値 = 0.0.
longitude_of_projection_origin	**数**	地図投影された空間上で直交直線座標の原点として選ばれた点の経度。範囲: −180.0 ≦longitude_of_projection_origin ≦180.0
north_pole_grid_longitude	**数**	回転された格子の経緯度座標系における真の北極の経度。[80]
perspective_point_height	**数**	地図投影の視点の楕円体(または球体)上の高さを**メートル単位で**記録する。たとえば(メテオサット衛星からの視野を模するのに使える)鉛直透視図法のような透視型の地図投影法で使われる。[81]
scale_factor_at_central_meridian	**数**	地図から算出された距離を縮小するための、または実際の距離に換算する際に除算(scale)[82]するための係数の中央経線に沿った値。範囲:scale_factor_at_central_meridian ＞ 0.0
scale_factor_at_projection_origin	**数**	地図から算出された距離を縮小するための、または実際の距離に換算する際に除算(scale)するための係数の投影原点における値。範囲:scale_factor_at_projection_origin＞ 0.0
semi_major_axis	**数**	測地系で地球の形を近似するために使われている楕円体の長半径を**メートル単位で**指定する。一般的に記号 *a* で表わされる。地球を球体で近似するときは[83]この属性は地球の半径を定義する。属性 inverse_flattening も参照せよ。[84]
semi_minor_axis	**数**	測地系で地球の形を近似するために使われている楕円体の短半径を**メートル単位で**指定する。一般的に記号 *b* で表わされる。地球を球体で近似するときはこの属性を省く(より好まれる方法)か、あるいは semi_major_axis 属性と同じ値を設定すべきである。属性 inverse_flattening も参照せよ。

[78]訳注:本当は中央経度は原点経度の別名ではない(中央経度は一部の地図投影法にしか定義されず、原点経度を別にとることも可能)のだが、この記述のおかげでこの規約では中央経度と原点経度が混同されていることがわかった。

[79]訳注:角度はこの規約の至る所で使われているのに、ここでだけ decimal とあえて断りを入れる意義は乏しい。六十進数の度分秒を属性値として与える形式はどこにも定義されていないのだから。

[80]訳注:原文では範囲を明示していないが常識的に緯度 φ は –90 ≦φ≦90 でなければならない。

[81]訳注:原文では範囲を明示していないが大抵は正の値だろう。ただし負値とすることで理論上は内射図法や心射図法が表現できる。

[82]訳注:原文は難解だが、実際の長さに対する地図上の長さの比率を尺度係数 scale factor という(だから本文の scale は除算)。正角図法のとき尺度係数は方向によらず位置だけの関数となる。横軸メルカトル図法のように中央経線上で尺度係数が一定となる場合にこの属性が用いられる。

[83]訳注:地球球体近似の場合はこの属性と earth_radius 属性とのどちらを使うことを勧告するのか明確でない。

[84]訳注:原文では範囲を明示していないが正の値でなければならない。短半径も同様。

属性	型	記述
standard_parallel	数	地球を表わす参照球あるいは参照楕円体に地図投影面（平面、円錐、あるいは円筒）が接する緯線（単独又は複数）を指定する。標準緯線に沿った縮尺には歪みがないので、標準緯度は「正しい縮尺の緯度」とも呼ばれる。投影円錐が楕円体と交わる場合、標準緯線は2本あり、この場合この属性をベクタとして両方の緯度の値を記録することが出来る。この場合（北又は南の）極に近い標準緯線を先に与えることを規約とする。~~地球表面と投影面が交わる緯度一定の線。~~範囲: −90.0 ≦standard_parallel ≦90.0
straight_vertical_longitude_from_pole	数	北極又は南極から上向きに位置づけられる経度。範囲: −180.0 ≦straight_vertical_longitude_from_pole ≦180.0

付録G　　履歴

2004年6月14日

1. 「ランベルト正積方位図法」の節を追加。
2. 「ポーラーステレオ図法」の節: 地図パラメタ latitude_of_projection_origin を追加。

2004年7月1日

1. 第5.7節「スカラー座標変数」: スカラー座標変数の利用により COARDS 適合のアプリケーションとの互換性を損なう旨の注記を追加。
2. 例5.11「ひとつの解析から得られた多数の予報」: スカラー座標変数 p500に属性 positive を追加し、この圧力が鉛直座標であることをまぎれなく明示した。

2004年9月20日

1. 第7.3節「セル・メソッド」: 誤ったセルメソッド “standard deviation” を “standard_deviation” に訂正した。

2004年10月22日

1. 例5.7 「ランベルト正角図法」を追加。

2004年11月25日

1. 「大気の混合高度座標」: 大気の混合鉛直座標の定義を修正。

2006年3月21日

1. 「正距方位図法」の節を追加。
2. 「大気の自然対数圧力座標」の節を追加。

2008年1月17日

1. 「はじめに」: CF ガバナンスと暫定状態を参照するようにテキスト変更。
2. 第4章「座標の種別」、第5章「座標系」: 水平座標変数を識別するための属性 axis の使用に関して変更実施。
3. 文書の版数を1.1.に変更。

2008年5月4日

1. 第5.6節「水平の座標参照系、格子写像、および投影法」及び付録 F「格子写像」: 座標参照系の特性を指定できるようにするため、CF 格子写像属性を追加及び修正（トラックチケット #18）。
2. 表3.1「サポートされる単位」: 係数 “1e-2” の接頭辞を “deci” から “centi” に訂正（ト

ラックチケット #25）。

3. 文書の版数を1.2に変更。

2008年7月15日

1. 第3.5節「フラグ」、付録 A「属性」、付録 C「標準名修飾子」: フラグの定義を拡張して `flag_masks` 属性を使ってビットフィールド記法をサポートするようにした（トラックチケット #26）。

2. 文書の版数を1.3に変更。

2008年10月9日

1. 例4.3 “大気のシグマ座標” の欠陥を修正（トラックチケット #30）。

2. 第5章「座標系」の欠陥を修正（トラックチケット #32）。

2008年11月7日

1. 第5章「座標系」の用語法の欠陥を修正（トラックチケット #35）

2. 付録 D「無次元鉛直座標」の節見出しのつけ方に関する欠陥を修正（トラックチケット #36）。

2008年12月10日

1. 第7.3節「セル・メソッド」のあいまい性を除去する変更（トラックチケット #17）。

2. 文書の版数を1.4に変更。

2008年12月11日

1. 付録 F「格子写像」にランベルト正積円筒図法、メルカトル図法、及び正射図法の格子写像を追加（トラックチケット #34）。

2008年12月27日

1. 第4章「」で座標が格子点の位置を示すことを明確化するよう欠陥の修正（トラックチケット #44）。

2. 古くなった `Conventions` 属性の欠陥の修正（トラックチケット #45）。

【訳文の履歴】

2009年8月28日　初版公開

2015年7月9日　訂正1版

文献

[COARDS] Conventions for the standardization of NetCDF Files. Sponsored by the "Cooperative Ocean/Atmosphere Research Data Service," a NOAA/university cooperative for the sharing and distribution of global atmospheric and oceanographic research data sets . May 1995. <URL:http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html>

[FGDC] Content Standard for Digital Geospatial Metadata. Federal Geographic Data Committee, FGDC-STD-001-1998. <URL:http://www.fgdc.gov/standards/projects/FGDC-standards-projects/metadata/base-metadata/v2_0698.pdf>

[NetCDF] NetCDF Software Package. UNIDATA Program Center of the University Corporation for Atmospheric Research. <URL:http://www.unidata.ucar.edu/packages/netcdf/index.html>

[NUG] NetCDF User's Guide for Fortran: An Access Interface for Self-Describing Portable Data; version 3. Russ Rew, Glenn Davis, Steve Emmerson, and Harvey Davies. June 1997. <URL:http://www.unidata.ucar.edu/packages/netcdf/guidef/> 実際には本文では第2.4版の記述を参照しているが、そのテキストは <URL:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.6890> で見られる。

[OGP/EPSG] OGP Surveying & Positioning Committee <URL:http://www.epsg.org/> and EPSG Geodetic Parameter Registry. <URL:http://www.epsg-registry.org/>

[OGP/EPSG_GN7_2] OGP Surveying and Positioning Guidance Note 7, part 2: Coordinate Conversions and Transformations including Formulas. [85]

[SCH02] C Schaer, D Leuenberger, and O Fuhrer. 2002. “A new terrain-following vertical coordiante formulation for atmospheric prediction models”. Monthly Weather Review. **130**. 2459-2480.[86]

[Snyder] Map Projections: A Working Manual. USGS Professional Paper 1395. <URL:http://pubs.er.usgs.gov/usgspubs/pp/pp1395>

[UDUNITS] UDUNITS Software Package. UNIDATA Program Center of the University Corporation for Atmospheric Research. <URL:http://www.unidata.ucar.edu/packages/udunits/>

[W3C] World Wide Web Consortium (W3C) . <URL:http://www.w3.org/>

[XML] Extensible Markup Language (XML) 1.0. T. Bray, J. Paoli, and C.M. Sperberg-McQueen. 10 February 1998. <URL:http://www.w3.org/TR/1998/REC-xml-19980210>

訳注の引用文献

[ASCII] Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII). ANSI INCITS 4-1986 (R2007). 今日では実質的内容は ISO 646 あるいは ITU-T Recommendation T.50 “International Alphabet No.5” と同一であり、後者が WMO Publication No. 386 "Manual on the GTS" Vol. 1, Part II, Attachment II-2 に転載されているのを <URL:http://www.wmo.int/pages/prog/www/ois/Operational_Information/Publications/WMO_386/WMO_386_Vol_I_en.pdf> で見ることができる。

[CSM] NCAR-CSM NetCDF Conventions, Version 1.0. Brian Eaton, 1997. <URL:http://www.cgd.ucar.edu/cms/eaton/netcdf/NCAR-CSM.html>

[DC] Information and documentation -- The Dublin Core metadata element set. ISO 15836:2009. <URL:http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=52142>, 実質同内容の ANSI Z39.85 は <URL:http://www.niso.org/kst/reports/standards?step=2&gid=&project_key=9b7bffcd2daeca6198b4ee5a848f9beec2f600e5> で入手できる。

[ISO19115] Geographic information – Metadata. ISO 19115:2003. <URL:http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26020>, 実質同内容の JIS X 7115 は <URL:http://www.jisc.go.jp/app/pager?id=57863> で閲覧できる。

[ISO2955] Information processing --- Representation of SI and other units in systems with limited character sets. ISO 2955:1983. 廃止規格であるが、内容は <URL:http://isotc.iso.org/livelink/livelink/4289384/ISO_2955-1983E_repr_of_SI_units_with_limited_char_sets.pdf?func=doc.Fetch&nodeid=4289384> で取得できる。

[85] <URL:http://www.epsg.org/guides/G7-2.html>

[86] <URL:http://ams.allenpress.com/perlserv/?request=get-abstract&issn=1520-0493&volume=130&issue=10&page=2459&ct=1>