技術文書 07-101

Arden Syntax の調査

2007年8月

保健医療福祉情報システム工業会
診療支援システム委員会

まえがき

医療情報システムの将来の発展方向の一つに，医療安全の向上や EBM (Evidence-based Medicine，根拠に基づく医療)の支援を目的とした診療意思決定支援システムがある。診療意思決定支援システムは，1980 年代の人工知能の研究開発ブームの火付け役として実用化への道が模索されてきたものである。昨今の医療安全や EBM 支援といったニーズの上昇に加えて，診療記録の標準化の進展やハードウェア･ソフトウェアの発達を考えると，診療意思決定支援システムの実用化を再度検討する時期にきていると考えられる。

保健医療福祉情報システム工業会〔JAHIS〕は，このような医療情報システムの発展方向を見据え，診療支援システム委員会において，過去に検討されてきた診療意思決定支援システムの調査を開始した。先ず，現在 Helath Level 7 (HL7)で標準化されている医療知識記述方法 Arden syntax の調査から取り組むこととした。本書では，主に，Arden Syntax Version 2.1 の翻訳結果を示す。本書が，Arden Syntax の理解の一助となり，診療意思決定支援システム実用化に多少なりとも貢献できれば幸いである。

2007年 8 月

保健医療福祉情報システム工業会
診療支援システム委員会

目 次

1. はじめに

医療のさらなる発展には，近年急速に普及し始めている電子カルテなどの情報化が大いに貢献すると考えられている。電子カルテシステムを中心とした医療情報システムは，医療提供者の間での情報共有や，過去データの迅速な検索による診療の効率化，書庫削減による省スペース化，蓄積データの2次利用による診療や病院運営の改善，地域や疾患ごとの医療連携といった様々なﾒﾘｯﾄを提供すると考えられている。さらに加えて，近年注目を集めている医療安全の向上やEBM (Evidence-based Medicine，根拠に基づく医療)の支援といったﾒﾘｯﾄも提供できると考えられている。特に，「人はミスするものである」といった米国 Institute of Medicine の医療安全に対する警鐘は，情報システムによる医療ミスのチェックの必要性を訴えるものとなっている。また EBM を支援するために，診療ガイドラインの蓄積や整備が進められ，情報システムを用いて効率的に閲覧できるようになってきている。

これらを鑑みると，医療情報システムの次なる発展方向の一つとして，診療における意思決定をさらに支援するような機能として診療意思決定支援システムが求められるようになると考えられる。診療意思決定支援システムは，1980 年代の人工知能の研究開発ブームの火付け役として長く実用化への道が模索されてきたものである。昨今の医療安全や EBM 支援といったニーズの上昇に加えて，診療記録の標準化の進展やハードウェア･ソフトウェアの発達を考えると，診療意思決定支援システムの実用化を再度検討する時期にきていると考えられる。

実用化推進のためには，模索されてきた診療意思決定支援システムを振り返ることが大切である。我々は，診療意思決定支援システムで使用する，医療知識記述方法の標準仕様である Arden Syntax に注目し，調査することとした。Arden Syntax は医療知識の定義と交換･配布を目的とした記法で，現在 HL7 (Health Level 7)で標準化が進められ，ANSI (American National Standards Institute)規格に採用されている。我々は先ず承認された最新の Arden Syntax の翻訳から着手することとした。

本書では，主に Arden Syntax Version 2.1 の翻訳結果を示す。本書が，Arden Syntax の理解の一助となり，診療意思決定支援システム実用化への礎となれば幸いである。

2. 診療意思決定支援システムと Arden Syntax

2.1 診療意思決定支援システムとは

診療意思決定支援システム（Clinical Decision Support System）とは，診療もしくは病院管理など，広く保健医療分野において，医療提供者などの意思決定者が意思決定を行う際に，意思決定を支援する情報を提供するシステムのことである。意思決定の種類としては，診断や治療方針の判断，検査･注射･投薬･処置･手術などの指示･予約･実施，保険請求などの病院運営に関する判断などが例として挙げられる。このような意思決定の際に，ミスを抑制して医療安全の向上や，臨床上の判断根拠の共有を図ることが，診療意思決定支援システムの目的である。

診療意思決定支援システムとしては，単独で意思決定者からの質問に応答する独立システムと，電子カルテシステムなど臨床データベースを有する情報システムとの統合システムの二つの形態が考えられる。医療安全や判断根拠の共有のために求められているのは，後者の統合システムである。統合化された意思決定支援システムの構成例としては，電子カルテシステム，ｲﾍﾞﾝﾄﾓﾆﾀ，ルールエンジンの三つからなる形態が挙げられる。ｲﾍﾞﾝﾄﾓﾆﾀは，ユーザや他システムからの電子カルテへの入力ｲﾍﾞﾝﾄを監視し，ある条件を満たすｲﾍﾞﾝﾄを捉えたときにはルールエンジンにﾄﾘｶﾞｰを送出するものである。ルールエンジンは，ﾄﾘｶﾞｰを受信し，蓄積された論理に従って計算を行い，電子カルテなどにｱﾗｰﾄやﾘﾏｲﾝﾀﾞを送出する。論理計算においては，必要であれば臨床データを電子カルテから抽出して計算を行う。さらに後の監査のために，ログを保存したり，ｱﾗｰﾄやﾘﾏｲﾝﾀﾞの通知方法を変更したりと，診療意思決定支援システムには様々な構成が考えられる。

診療意思決定支援システムの実用化のためには，論理の記述方法の標準化が肝心である。論理の記述方法以外の部分に関しては，病院の運用を検討しながら最適なｱｰｷﾃｸﾁｬを選択し情報システムを構築していけばよいと考えられる。例えば，論理計算のスピードやｱﾗｰﾄなどの結果の通知については，病院の規模や運用に合わせてシステムを

構築する必要がある。しかしながら論理の記述方法に関しては，一機関だけでは医療知識の網羅性を高めることが困難なため，標準化を推し進め，医療知識の共有や配布･交換，アップデートを可能とすることが求められる。論理の記述方法には，いくつかの標準仕様が提案されているが，Arden Syntax が最も標準化が進んでいると考えられる。

2.2 Arden Syntax とは

Arden Syntax とは医療知識の定義と交換･配布を目的としたシンタックスである。その歴史は古く，1989 年に Columbia-Presbyterian Medical Center Arden Homestead retreat で開発が始まった。Arden の名称は開発が行われた場所に由来している。Arden Syntax の仕様の大部分は，LDS Hospital (Salt Lake City, Utah)の HELP と，Regenstrief Institute for Health Care (Indianapolis, Indiana)の Regenstrief Medical Record System で使用されていた言語 CARE を基に決定されてきた。現在は，HL7 (Health Level 7)が標準化を進め，ANSI (American National Atandard Institute)が承認した規格となっている。

Arden Syntax の特徴は，医療知識を記述するために，論理形式で記述されたモジュール（Medical Logic Module. 以下、MLM と記載する）の集合を用いる点にある。逆に言えば，適用範囲は論理形式で記述可能な知識に制限されている。MLM には，単一の意思決定をするのに十分な論理と，論理を判定するために必要なデータの取得方法が記載される。また，このような知識の記載だけではなく，作成者･日時や他の知識ソースへのリンク情報など，維持整備するための管理情報も記載される。医療従事者が Arden Syntax を使って直ぐに MLM を作成することができ，作成された MLM はこの仕様に則った意思決定支援システムで直ぐに使用できることが目指されている。

MLM は，次のような三つのカテゴリで構成される：

```
Maintenance:

    Slotname: slot body;;

    Slotname: slot body;;

Library:

    Slotname: slot body;;

Knowledge:

    Slotname: slot body;;

End:
```

Maintenance カテゴリには，医療知識ではなく，MLM の知識ベースの管理や変更のための情報が記載される。詳細は翻訳本文に記載するが，title, mlmname, arden syntax version, version, institution, author, specialist, date, validation といった管理用の情報を格納するスロットが含まれている。Library カテゴリは，MLM の知識情報の管理や，外部医療知識への参照情報が記載される。同じく詳細は翻訳本文に記載するが，purpose, explanation, keywords, citations, links と，MLM の知識内容の概要と外部データソースへの参照情報が含まれている。knowledge カテゴリは MLM の動作を規定する情報が記載される。同じく詳細は翻訳本文に記載するが，MLM で使用される語句や変数の設定とデータ抽出方法が記載される Data スロット，MLM が呼び出されるトリガー条件が記載される evoke スロット，テストする論理が記載される logic スロット，logic スロットの結果が真の場合のみ起こされる行動が記載される action スロットで構成される。

Arden Syntax の課題としては，広範に使用できることを重視したために，標準として規定していない部分が大きく二つある点が挙げられる。一つ目は，論理記述で使用する標準用語が規定されていない点である。これは，例えば，疾病名をチェックする MLM を作成する際に，ある疾病名を定数として記述したいが，ICD などのコードを使用するか，単に文字列として記述するかといった点については規定をしていないことを示している。二つ目は，臨床データベースなどから論理計算に必要なデータを抽出してくる部分について規定していない点である。すなわち，data スロットにおいて，中括弧

{} で囲まれた範囲内の記述はシステム固有の部分として規定してない。このことは, MLM の頒布を受けたときには, data スロットの記載を自分のシステムに応じて修正する必要があることを示している。もちろん一点目で述べたように, 論理記述の定数部分も自分のシステムが使用しているコード体系に応じて修正が必要となる。これら課題を解決するために, 論理記述に使用する用語やコード体系の規定が必要である。

次ページ以降に, まず Arden syntax version 2.1 の翻訳に用いた標準単語翻訳表を示し, 次に資料として Arden Syntax for Medical Logic Systems Version2.1 和訳を示す。

3. 標準単語翻訳表

翻訳にあたっては以下の標準単語翻訳表を基準とし, HL7 の他文書の翻訳との整合性をなるべくとるように努めた。また, 演算子･定数やスロット名などの予約語については, 活用しやすくするためにアルファベットのまま記述した。

標準単語翻訳表

英単語	和訳	補足･定義
Action	行動	予約語として使用する場合はアルファベットのままとする
action block	action ブロック	
Action Slot	action スロット	
Aggregation	集合体	配列, 集約
Alert	アラート	
Annex	付録	Annex, Appendix 共に付録と訳す。Annex は章番号 X。
Appendix	付録	Annex, Appendix 共に付録と訳す。Appendix は章番号 A。
Arden time	Arden Syntax で使用されている時刻表現	
argument	被演算子	予約語として使用する場合はアルファベットのままとする
arithmetic negation	符号反転	
Arithmetic Operators	算術演算子	
assignment	代入	
asynchronous	非同期	
Author	作成者	予約語として使用する場合はアルファベットのままとする
authoring institution	作成機関	予約語として使用する場合はアルファベットのままとする
backward compatibility	下位互換	
binary	二項演算子	
Boolean	ブール型	予約語として使用する場合はアルファベットのままとする
call statement	call 文	
Case Insensitivity	大文字小文字の同一視	
category	カテゴリ	
citation	引用	
clinical decision support communications	臨床判断支援用の通知	
coded slot	コード化スロット	
date	日付	
Delayed Trigger Statement	遅延トリガー文	
delayed trigger statement	遅延トリガー文	
destination variable	目的変数	
DTD	DTD	
duration	継続時間	予約語として使用する場合はアルファベットのままとする
duration operators	継続時間演算子	
embedded	間の	
event	イベント	予約語として使用する場合はアルファベットのままとする
eventtime	eventtime	
evoke	起動	予約語として使用する場合はアルファベットのままとする
evoke slots	evoke スロット	
expression	式	

3. 標準単語翻訳表

external event	外部のイベント	
For loop	for ループ	
formatting string	書式文字列	
fractional seconds	秒以下の表現	意訳
General Properties:	概要	コメント:項目名などの訳は統一した方がよい
granularity	粒度	
health care provider	医療従事者	
health knowledge bases	保健医療知識ベース	
health logic	保健医療に関する論理	
identifier	識別子	
if-then statement	If-then 文	
inclusive	両端を含む	意訳
institution	機関	
institution-specific	機関固有に	
its usage is	使用方法は以下の通りである	
Left associative	左結合	
line break	改行	
List	リスト	
logic slot	logic スロット	
logical negation	論理否定	
main statement	中核となる文	
mapping clauds	マッピング節	
matching	照合	
message variable	message 変数	
midnight	午前零時	非予約語
mlmname	mlmname	
non-associative	非結合	
now	now	予約語
null	null	
number constants	数値定数	
observe	保たれる	
Operation	演算	
operator	演算子	
optional	オプション	
organization	構成	
patient record	診療録	
Periodic Trigger Statement	周期的トリガー文	
personnel	職員	
primary time	プライマリ日時	time of day＝時刻（ここでは日時ではない） 文書内で定義されている
priority	優先度	
query	照会	
query result	照会結果	
read mapping	読み出しマッピング	
retrieval	抽出	
return statements	リターン文	予約語として使用する場合はアルファベットのままとする
reuired	必須	
right associative	右結合	

scope	スコープ	
serum potassium	血清カリウム値	
simple trigger statement	単純トリガー文	
single field	単一フィールド	
slot	スロット	
slot body type	スロット本体のタイプ	
space	空白文字	white space＝余白
specialist	専門家	予約語として使用する場合はアルファベットのままとする
statement	文	
string	文字列	予約語として使用する場合はアルファベットのままとする
string operator	文字列演算子	
structured slot	構造化スロット	
Temporal Operators	時間演算子	時間関連用語について整理
term constants	用語定数	
terminology	用語	
ternary	三項演算子	
The after operator	after 演算子	
the default list handring	デフォルトのリスト処理	
the default primary time handring	デフォルトのプライマリ日時処理	
time	日時	時刻だけでなく日にちを含む点に注意
time constants	日時定数	
time constants	日時定数	
time operator	time 演算子	
time-of-day	時刻	日にちは問わず時刻だけを指す点に注意
token	トークン	
triggertime	triggertime	予約語
type	タイプ	
type character	タイプ文字	
unary	単項演算子	
valid MLM variable	有効な MLM 変数	
validation	検証	予約語として使用する場合はアルファベットのままとする
variable	変数	
white space	余白	
write statements	write 文	

付録.　作成者名簿（五十音順）

大島　義光	㈱日立製作所
川野　崇史	中央システム技研㈱
岸田　幸博	キッセイコムテック㈱
栗原　勝	アイティーコーディネート㈱
佐柄木　秀郎	日本ユニシス･ソリューション㈱
溝口　和弘	日立メディカルコンピュータ㈱
成松　亮	東芝メディカルシステムズ㈱
藤咲　喜丈	日本光電工業㈱
尾藤　良孝	㈱日立製作所
村上　英	東芝住電医療情報システムズ㈱
山本　裕	横河電機㈱

注：作成者の所属は 2006 年 5 月時点

［資料］

Arden Syntax for Medical Logic Systems
Version 2.1

和訳

目次

バージョン 2.1 の新仕様

本バージョンの新仕様の一つは, WRITE 文が構造化されたメッセージを使用できるよう拡張された点である。これにより, 臨床判断支援用の通知方法を詳細に管理する, もしくはそのような管理を要求するシステムに, 本標準仕様を容易に適用できるようになる。このメッセージの構造は, Extended Markup Language (XML)にエンコードされた Document Type Definition (DTD)として表現される。

Where trigger (従来の 13.3.4 節)は, バージョン 2.1 の仕様から正式に削除された。これは, Arden Syntax バージョン 2 では, 下位互換のために残されていたが, なるべく使用しないよう推奨されていた。従来, where trigger によって記述されていた MLMs は, 判断木を data スロットの event 定義に加えるか, logic スロットの opt-out に加えるか, どちらかを取るようにする。

本バージョンには, 様々な新しい演算子が追加された。”Currenttime”は, MLMs の実行中, 何時の時点でもシステム時間を返す。また, 新たに次の 6 つの新しい文字列演算子が追加された："length", "uppercase", "lowercase", "trim", "find…in string", "substring…characters from."。

また, Arden Syntax の version スロットに, バージョン 2 と 2.1 に適応した MLMs を識別するため, 新しい有効なコードが追加された。

最後に, 幾つかの演算子を使いやすくするため, 小さな修正が行われた。例えば, “in” (“not in”)を“is in” (“is not in”)の同義語として追加したこと, 時間比較で使用される“from <time>”を“after <time>”の同義語として追加したこと, “from now” を“now + <time>”の同義語として追加したこと, “occur/occurs/occurred at” を“occur/occurs/occurred equal”の同義語として追加したこと, そして MLM の name スロットに, punctuation mark period (“.”)を使用できるようにしたことである。

1 目的

本仕様は, 医療従事者, 情報システム, 医療施設間でコンピュータ化された保健医療知識ベースを分担･共有することを目的とする。本標準の適用対象は, これら知識ベースが, 複数の分割されたモジュールの集合として表されるものに限っている。Medical Logic Module (MLM)と呼ばれる各々のモジュールは, 単一の意思決定をするに十分な知識を含んでいる。禁忌アラート, 管理推奨, データ解釈, 治療プロトコル, 診断スコアなどは, MLMs で表現しうる保健医療知識の例である。各々の MLM は, MLMs から成る知識ベースのメンテナンスを容易にするために, 管理情報も含んでいる。また, 他の知識へのリンク情報も含んでいる。医療従事者は, この形式を使って MLMs を直ぐに作成できるとともに, 作成された MLMs は, 本仕様に適応した情報システムにそのまま使用することが可能である。

2 参考文献

2.1 ASTM 標準[1]:

E 1238 Specification for Transferring Clinical Laboratory Data Messages Between Independent Computer Systems

E 1384 Guide for Content and Structure of an Automated Primary Record of Care

2.2 ISO 標準[2]:

ISO 8601 - 1988 Data Elements and Interchange Formats-Information Interchange (representation of dates and times)

ISO 8979 - 1986 Latin-1 Coded Character Set

2.3 ANSI 標準[3]:

ANSI X3.4 - 1986 Coded Character Sets-American National Standard Code for Information Interchange (7-bit ASCII)

ANSI/ISO 9899 Programming Language C

ANSI/ISO/IEC 9075 Information technology -- Database languages -- SQL

2.4 Health Level Seven 標準[4]:

HL7 Version 2.3

---

1 Annual Book of ASTM Standards, Vol 14.01. Available from American Society for Testing and Materials , 100 Barr Harbor Drive, West Conshohocken, PA19428-2959, USA

2 ISO, 1 Rue de Varembe, Case Postale 56, CH 1211, Geneve, Switzerland.から入手可能

3 American National Standards Institute, 1430 Broadway, New York, NY 10018, USA.から入手可能

4 Health Level Seven, Inc., 3300 Washtenaw Ave, Suite 227, Ann Arbor, MI 48104, USA.から入手可能

3 用語

3.1 定義

3.1.1 Medical Logic Module (MLM), n

保健医療知識ベースの独立した1単位。各々の MLM は, メンテナンス用の情報, 他の知識ソースへのリンク情報, そして単一の意思決定を行うに十分な論理情報を含んでいる。

3.2 本標準に特有な用語の説明

3.2.1 time, n 日時

絶対時間の内の一点。日付と時刻を含むタイムスタンプとしても知られる。

3.2.2 time-of-day, n 時刻

時, 分, 秒。そして午前零時からの経過秒を表すこともある。

3.2.3 date, n 日付

グレゴリオ暦の年月日。

3.2.4 duration, n 継続時間

特に開始時間や終了時間を持っていない継続時間 (例えば, **3 days**)。

3.2.5 institution, n 機関

規模の大小によらず, 自動化された意思決定や品質保証を提供する保健医療に関する機関。

3.2.6 event, n イベント

診療に意味のある状態変化。これは, 常にではないが, ほとんどの場合, 診療データベースの変更を意味する。例えば, 投薬オーダは診療データベースを更新するようなイベントである。しかしながら, 投薬オーダに書かれた終了時刻を経過し, 投薬を終了するのはイベントであるが, 診療データベースに変更を生じさせるものではない。

3.3 本標準における表記方法

本標準を通して, オプションの要素は, 大括弧(**[ ]**)の中に要素を書くことで示される。これは, element 演算子**[ ]** (**9.12.18**節参照)と混同してはならない。よって, 例えば, **Is [Not] Equal** は **Is Equal** と **Is Not Equal** は共に有効なことを示している。最も頻繁に使われるオプションの要素は **not** と **of** である。

4 重要性と使用例

保健医療分野では, 意思決定支援システムは長年に渡って成功裏に使用され, 既に幾つかの機関で大規模な知識ベースが搭載されている。これら知識ベースには, 多くの概念的な類似性が見られる。しかし, 残念ながら, これら知識ベースの文法は異なっている。一機関で完全な知識ベースを構築できるわけではないので, 複数の機関で知識ベースを分担･共有する必要がある。

このような分担･共有には多くの障害がある。例えば，異種の語彙，メンテナンス上の問題，地域の違い，法的責任，使用料，統語上の違いなどが挙げられる。本標準は，文法を決定することによって，知識ベースの構築と分担･共有を行う上での一つの障害を解決しようとするものである。さらに言えば，この文法に，メンテナンス情報，診療上の責任の所在，文献へのリンク，機関固有の語彙と知識ベースの語彙との関連付けを行う特別なフィールドを持たせることで，他の障害も解決しようとするものである。

保健医療知識ベースの対象範囲は広い。本仕様では，Medical Logic Modules (MLMs)の集合で表現可能な知識ベースに対象を限定している。各々の MLM は，メンテナンス情報，他の知識ソースへのリンク情報，そして単一の診療意思を決定するに十分な論理情報を含んでいる。独立したルール，数式，プロトコルから成る知識ベースは，このような MLMs による表現に最も適している。

本仕様は，Columbia-Presbyterian Medical Center 1989 Arden Homestead retreat on sharing health knowledge bases の所産である。そして，本仕様の大部分は，LDS Hospital, Salt Lake City, UT の HELP システムと Regenstrief Institute for Health Care, Indianapolis, IN の Regenstrief Medical Record System に使用されている言語 CARE から成っている。

5 MLM フォーマット

5.1 ファイルフォーマット

MLM は ASCII ファイル(ANSI X3.4 - 1986)に保存された文字列である。[国際的には，ASCII ファイルを ISO 8859/1 ("Latin-1")を使って拡張しても良い。しかし，標準準拠の実装には X3.4 だけを実装すれば良い]。一つもしくは複数の MLMs が同一のファイルに格納されうる。ファイルの中では，一つの MLM は，**maintenance:**で始まり，**end:**で終わる。MLMs は，7.1.10節で定義される余白，かつ/または，7.1.9 節で定義されるコメントで区切られる。

5.2 文字セット

MLM では，印字可能な ASCII 文字（ASCII 33 から 126），空白文字（ASCII 32），改行（ASCII 13），行送り（ASCII 10），水平タブ（ASCII 9），垂直タブ（ASCII 11）が使われる。水平タブの使用は，いくつの空白文字が使われるか合意されていないので，推奨されていない。例えばベルや後退のような他の文字は，MLM での使用は認められていない。文字列定数（7.1.6 節参照）やコメント（7.1.9 節参照）の中では，これら文字セットへの制限はない。

5.3 改行

複数の行は，改行で分けられる。改行は，単一の復帰コード，単一の行送りコード，もしくは復帰コードと行送りコードの組で表される。

5.4 余白

空白文字，復帰，行送り，水平タブ，垂直タブと書式送りは，まとめて余白と呼ばれる。7.1.10 節も参照のこと。

5.5 一般的なレイアウト

付録A1は，Backus-Naur Form で表される Arden Syntax MLMs に関する，内容に依存しない文法（形式上の説明）を示す**(3)**。付録 X2 には，MLM の例を示す。典型的な MLM は以下の通りである。

```
maintenance:
slotname: slot-body;;
slotname: slot-body;;
```

```
...
library:
slotname: slot-body;;
...
knowledge:
slotname: slot-body;;
...
end:
```

5.6 カテゴリ

一つの MLM は，三つのカテゴリ：maintenance, library, knowledge にグループ化されたスロットで構成される。各カテゴリは，カテゴリ名に連続したコロンで示される（即ち，**maintenance:**, **library:**, **knowledge:**）。余白は，カテゴリ名の前とコロンの後に来ても良いが，カテゴリ名とコロンの間に余白が来るのは許されない。また，カテゴリは，本標準で記述される順番で，MLM に書かれなければならない。

5.7 スロット

各カテゴリは，スロットの集合として表される。

各スロットは，スロット名と連続したコロン（例えば，**title:**），続いてスロット本体，そしてダブルセミコロンと呼ばれるスロットの終了を表す連続したセミコロン（;;）で構成される。余白が，スロット名の前やコロンの後ろに来ても良いが，スロット名とコロンの間に来ることは許されない。スロット本体の内容はスロットに依存するが，コメント（7.1.9 節），文字列定数（7.1.6 節），マッピング節（7.1.8 節）の中以外ではダブルセミコロンを含んではならない。

各々のスロットは，MLM の中で唯一でなければならない。そして，カテゴリとスロットは本標準で記述される順番で，MLM に書かれなければならない。

5.8 スロット本体のタイプ

以下にスロット本体の基本的なタイプを記す：

5.8.1 テキストのスロット

テキストのスロットは，任意のテキスト（スロットを終了するダブルセミコロンを除く）より成る。MLM 標準が拡大するに従い，現在テキストのスロットとして取り扱われているものもコード化された，もしくは構造化されたスロットになっていく。テキストのスロットの例としては，任意のテキストを含みうるタイトルスロットが挙げられる。必須であるテキストのスロットのためには，テキストは空でもよい。

5.8.2 テキストリストのスロット

いくつかのスロットは，テキストのリストを含む。これらは，単一のセミコロン（;）で区切られた複数の任意のテキスト節から成る。テキストリストのスロットの例としては，キーワードスロットが挙げられる。テキストリストは空であっても良い。ダブルセミコロンは（スロットを終了させるので）含んではならない。

5.8.3 コード化スロット

コード化スロットは，数字や日付，予め定義されたリストの中の単語など，単純なコード化された見出し語から成る。例えば，priority スロットは数字のみから成るし，validation スロットは production, research などの用語のみから成る。

5.8.4 構造化スロット

構造化スロットは，統語論的に定義されたスロット本体である。これはコード化スロットよりも複雑で，7 章でさらに詳しく定義する。このスロットの例としては，logic スロットが挙げられる。

5.9 MLM の終了

MLM の終了は単語 **end** と連続したコロン（即ち，**end:**）で示される。終了子の前やコロンの後には余白が許されるが，間には許されない。

5.10 大文字小文字の同一視

カテゴリ名，スロット名と **end** 終了子は，大文字（例えば，**END**），小文字（例えば，**end**），または混合（例えば，**eNd**）のどれを使って記しても良い。7.1.1.2 節と 7.1.2.1 節を参照のこと。

6 スロットの記述

以下の各節のスロット名の次に書かれているのは，テキスト，テキストリスト，コード化，構造化のどのスロットにあたるか，このスロットは必須なのかオプションなのかを示している。スロットは，本仕様に記されている順番で，MLM に書かれなければならない。

6.1 Maintenance カテゴリ

maintenance カテゴリは，MLM の中で保健医療知識に関係しない情報を特定するスロットから成る。これらスロットは，MLM 知識ベースのメンテナンスと変更管理に使われる。maintenance カテゴリには，どのバージョンの Arden Syntax が使われているかも記されている。

6.1.1 Title（テキスト，必須）

タイトルはその MLM が何をするか端的に表すコメント示す。例としては，次の通りである：

```
title: 妊娠中女性の B 型肝炎表面抗原;;
```

6.1.2 Mlmname（コード化，必須）

Mlmname は，ある一つの作成機関において MLM を唯一に識別する。これは，文字で始まり，文字や数字，ピリオド（.），アンダーバー（_）の続く文字列で表される。これは 1 から 80 文字の長さを持ちうる。Mlmname は大文字小文字を同一視する。また，一つのファイルに複数の MLM が含まれている場合を考慮し，mlmname と ASCII ファイルの名称は別に定義される。例としては，次の通りである：

```
mlmname: hepatitis_B_in_pregnancy;;
        or
mlmname: hiv_screening.mlm;;
```

mlmname が，このスロット名としては推奨されているが，filename というスロット名も下位互換のため許される。

6.1.3 Arden Syntax version（コード化，オプション*）

Arden Syntax バージョンは，コンパイラにどの標準バージョンが使われて MLM が書かれているかを伝える。このスロットがない場合，MLM は ASTM E1460-1992 標準（このスロットが定義されていない標準）で記述されているものと想定される。このスロットがある場合，次のように記述される：

```
arden: Version <Version number of Arden Syntax standard>;;
```

このテキストは大文字と小文字を同一視する。例としては次の通りである：

```
arden: Version 2;;
arden: version 2.1;;
```

* このスロットはバージョン2以降必須であるが，下位互換のためオプションとしている。即ち，もしこれがない場合，バージョン1として取り扱われる。

6.1.4 Version（テキスト，必須）

現在の MLM のバージョンでは，各機関のバージョン管理システム（SCCS や RCS）の使い勝手を考慮して，長さ 80 文字以内の任意のテキストとして定義されている。ただし，バージョンは 1.00 からスタートし，小修正は 0.01 の追加，大修正は 1.00 の追加という管理が推奨されている。厳密なバージョン情報は機関固有であるが，どの MLM が最新か決められなければならない（11.2.3 節参照）。例としては，次の通りである：

```
version: 1.00;;
```

6.1.5 Institution（テキスト，必須）

institution スロットは，80 文字以内の作成機関の名称から成る。例としては，次の通りである：

```
institution: Columbia University;;
```

6.1.6 Author（テキストリスト，必須）

author スロットは任意の形式のテキストである。これは，MLM の複数の作成者をセミコロンで区切って表している。ファーストネーム，ミドルネームまたはイニシャル，ラストネーム，コンマ，学位の形式で記すことが推奨されている。

作成者名の後に，括弧の中に e-mail アドレスをオプションで記すことも許されている。これには，インターネットアドレスが想定されている。例としては，次の通りである：

```
author: John M. Smith, Jr., M.D. (jms@camis.columbia.edu);;
```

6.1.7 Specialist（テキスト，必須）

専門家とは，当該機関において MLM の検証と実装に関して責任を持つ人物を指す。このスロットは常に存在しなければならないが，ある機関から他の機関へ MLM を転送する際には空白でなければならない。このスロットを埋め，MLM の使用責任を負うのは，受ける側の機関の責任である。このスロットの形式は他のスロットと同一である。例とては，次の通りである：

```
specialist: Jane Doe, Ph.D.;;
```

又は

```
specialist: ;;
```

6.1.8 Date（コード化，必須）

MLM の最終修正日はこのスロットに記される。日付もしくは日時（いわゆる，日付と時刻から成る絶対時間の一点）が用いられる。日付と日時のフォーマットには，オプションとしてタイムゾーン（ISO 8601:1988 (E)）を持つ ISO の拡張フォーマット（区切り子として T 又は t を使用）が使われる。日付は **yyyy-mm-dd** の形式，すなわち 1989 年 1 月 2 日が 1989-01-02 と表される。Arden がサポートすべき最初の日時は 1800 年 1 月 1 日(1800-01-01T00:00:00Z)である。時刻は，**yyyy-mm-ddThh:mm:ss** と，オプションで秒以下の表現とタイムゾーンで表される。よって，1:30 p.m. on January 2, 1989 UTC は 1989-01-02T13:30:00Z と表される。例としては，次の通りである：

```
date: 1989-01-02;;
```


6.1.9 Validation（コード化，必須）

validation スロットは，MLM の検証状態を示す。状態には次のうちの一つの用語が使われる：

production—診療システムでの使用が認められている

research—研究用の使用が認められている

testing—デバッグ用（MLM が書かれる時の初期値）

expired—期限切れ，診療には使えない

例としては，以下の通りである：

validation: testing;;

MLMs は絶対に **production** 状態では転送･共有されない。何故なら，受け手側の機関の責任者がこの確認状態を設定しなければならないからである。

6.2 Library カテゴリ

library カテゴリは，MLM の知識に関連し，知識ベースのメンテナンスに適切なスロットから成る。これらスロットは，予め定義された説明や医療関連の文献へのリンク情報を，医療提供者に提供する。また，これらスロットは MLM から成る知識ベースを探索する目的でも使用される。

6.2.1 Purpose（テキスト，必須）

purpose スロットは，なぜこの MLM が使われているかを端的に表す。例としては，次の通りである：

purpose: B 型肝炎の発症リスクのある新生児のスクリーニング;;

6.2.2 Explanation（テキスト，必須）

このスロットは，どのように MLM が動作するか平板な英語で端的に説明する。説明は，医療提供者に，MLM が何故そのような結論を導き出したかを示すためのものである。例としては，次の通りである：

explanation: この女性は昨年 B 型肝炎抗原に陽性であった。よって，彼女の新生児は B 型肝炎を発症する可能性がある。;;

6.2.3 Keywords（テキストリスト，必須）

キーワードとは，モジュールを探索するための記述的な単語である。UMLS 用語**(4)**が推奨されているが，強制ではない。用語はセミコロンで区切られる（コンマは各キーワードの中での使用が許されている）。例としては，次の通りである：

keywords: hepatitis B; pregnancy;;

6.2.4 Citations（構造化/テキスト，オプション）

この citation スロットには二つのフォーマットがある。一つは，テキスト形式で構造を持たないものである。このテキスト形式は下位互換のためであり，現在は推奨されていない。もう一つは，この節の後で説明される構造化された形式である。文献への引用は，バンクーバー形式で記される **(5)**。引用には特定の参照が得られるよう番号付けされなければならない。個々の引用は，次のような型を持っていてもよい。この型は，番号の後に続き，ある特定の MLM について，機能を示す。型としては，次のものがある：

a) **Support** – 引用が logic スロットのアルゴリズムを支持，実証，確認するもの

b) **Refute** – 引用が logic スロットのアルゴリズムを反論もしくは代替を提示するもの

例としては, 次の通りである:

```
citations:
1. SUPPORT  Steiner RW. Interpreting the fractional excretion of sodium. Am J Med 1984;77:699-702.
2. Goldman L, Cook EF, Brand DA, Lee TH, Rouan GW, Weisberg MC, et al. A computer protocol to predict myocardial
    infarction in emergency department patients with chest pain. N Engl J Med 1988;318(13):797-803.
;;
```

6.2.5 Links (構造化/テキスト, オプション)

この link スロットには二つのフォーマットがある。一つは, 構造を持たないテキストフォーマットである。テキストフォーマットは, 下位互換性のためにあり, 現在は推奨されていない。もう一つは, この節の後で説明される構造化されたフォーマットである。この link スロットにより, 機関が他の情報 (例えば, 電子教科書, 教育用のケース, 教育用のモジュールなど) へのリンク情報を設定することができる。各リンクはセミコロンで区切られる。リンクの内容は, 機関固有のものである。イントラネットやインターネットのサイトへのリンクは, 用語 URL (Uniform Resource Locator)を先頭に付け, 文書のタイトル, そしてプロトコルとデータソースの標準的な表現を付ける(例えば, "Document Title", 'FILE://link.html'; "Second Document", 'http://www.nlm.nih.gov/')。電子媒体は, 上で説明した **citations** スロットにも記述できる。構造化されたリンクの推奨フォーマットは, 以下の通りである:

link type, space (ASCII 32), link description (Arden Syntax string), comma, link text (Arden Syntax term).

必須要素は, リンクテキストである。

例:

```
links:
    OTHER_LINK 'CTIM .34.56.78';
    MESH 'agranulocytosis/ci and sulfamethoxazole/ae';
    URL "NLM Web Page", 'http://www.nlm.nih.gov/';
    URL "Visible Human Project",
        'http://www.nlm.nih.gov/research/visible/visible_human.html';
    URL "DOS HTML File", 'file://doslinx.htm';
    URL "UNIX HTML File", 'file://UnixLinx.html/';
;;
```

各機関は, MLMs を受け取ったときに期限切れのリンクをテストするべきである。

6.3 Knowledge カテゴリ

この knowledge カテゴリは, MLM が何を行うか実際に特定するスロットから成る。これらスロットが設定しているのは, MLM で使われる用語 (data スロット), MLM が起動される状況 (evoke スロット), テストする条件 (logic スロット), そして条件が真の場合に取る行動 (action スロット) である。

6.3.1 Type (コード化, 必須)

この type スロットは, knowledge カテゴリにどのようなスロットが含まれているかを示す。ただし, 現在定義されているタイプは, **data_driven**,だけである。これは, 次のスロットからなることを示している: data, priority, evoke, logic, action, and urgency。1992 年の標準仕様との下位互換のため, **data-driven** ("-"で区切られている単語)も許される。すなわち, 以下が許される:

```
type: data_driven;;
```

または

```
type: data-driven;;
```


6.3.2 Data (構造化, 必須)

この data スロットでは, MLM の中で局所的に使用される用語を, 機関にある実体へと割り当てる。実際の表現法は, 機関に依存する。このスロットの詳細は, 11 章で説明する。

6.3.3 Priority (コード化, オプション)

優先度とは 1 (低) から 99 (高) の番号であって, 複数の MLMs の起動条件が満たされたときに, その起動順序を特定するために使われる。この priority スロットでは, 整数のみが使われることが推奨される。機関は優先度を使うか使わないか選択することができる。機関は, 矛盾が生じないように, これら番号を補修する責任を負う。受けて側の機関は, 集めた MLMs に適するようにこれら番号を修正する必要がある。もし, priority スロットが省略されている場合, デフォルトの数値は 50 が使われる。例としては, 以下の通りである:

```
priority: 90;;
```

6.3.4 Evoke (構造化, 必須)

この evoke スロットは, MLM を起動する条件を表している。このスロットの詳細を 13 章に記す。

6.3.5 Logic (構造化, 必須)

このスロットは, MLM の実際の論理を表す。一般に, ここでは, いくつかの条件をテストし, **true** もしくは **false** を結論付ける。このスロットの詳細を 10 章に記す。

6.3.6 Action (構造化, 必須)

このスロットは, logic スロットの結論が **true** だった場合にとる行動を表す。このスロットの詳細を9.15節に記す。

6.3.7 Urgency (コード化, オプション)

行動やメッセージの緊急度が 1 (低) から 99 (高) の数値で表される。この urgency スロットには整数のみが使われることが推奨される。優先度が起動される MLMs の実行順序を決めるのに対し, 緊急度は MLM が真と結論付けた場合 (即ち, MLM が行動を表した場合) にのみ, その行動の重要性を決めている。もし urgency スロットが省略された場合, もしくは緊急度の数値が無いか 1 から 99 以外の場合には, デフォルトとして 50 が使われる。例としては, 次の通りである:

```
urgency: 90;;
urgency: urg_var;;
```

7 構造化スロットのシンタックス

7.1 トークン

構造化スロットは, 語彙の要素またはトークンとして知られている文字列から成る。これらトークンは次の節に示されるように分類される:

7.1.1 予約語

予約語は文字と数字からなる予め定義されたトークンである。これらは, 文を作るため, 演算子を表現するため, データ定数を表現するために使われる。この内のいくつかは現在使われていないが, 将来使われるために予約されている。演算子自体および予め定義された演算子の同義語は, 同義語として扱われる。

予約語は付録 A2に纏めてある。

7.1.1.1 The

The は構造化スロットに出てきたときでも無視される特殊な予約語である（即ち，空白文字と同一に扱われることを意味する）。この目的は，英語と同様な文章にすることにより，可読性を高めることにある。

7.1.1.2 大文字小文字の同一視

format with …フォーマット設定を除いて，シンタックスは予約語の大文字小文字を同一視する。すなわち，予約語は，大文字，小文字，または混合のどれで記述しても良い。例えば，**then** と **THEN** は同一の単語である。5.10 節，9.8.2節と付録A5.も参照のこと。

7.1.2 識別子

識別子はアルファベットと数字の組み合わせからなる。識別子の最初は，文字でなければならない。そしてそれ以外は文字，数字，アンダーバー（_）である。識別子は 1 から 80 の長さの文字でなければならない。80 文字以上の場合，エラーになる。予約語は識別子ではない。例えば，**then** は予約語であって，識別子ではない。識別子は，データを保持するための変数として使用される。

7.1.2.1 大文字小文字の同一視

本シンタックスでは識別子の大文字小文字を同一視する。5.10 節 と 7.1.1.2 節も参照のこと。

7.1.3 特殊記号

特殊記号は予め定義されたアルファベットにはないトークンである。特殊記号は，句読と演算子を表現するのに用いられる。これらは付録A3に纏めてある。

7.1.4 数値定数

数値定数は一つ以上の数字 (**0** to **9**)からなり，小数点 (**.**)を含んでも良い。（Specification E 1238 と HL7 2.3 では，**.1** と **345.** は有効な数値である。）数値定数は，**E** や **e** とそれに続く正負の記号と数字で表される指数で終了しても良い。以下は有効な数値の例である：

```
0
345
0.1
34.5E34
0.1e-4
.3
3.
3e10
```

7.1.4.1 負の数値

負の数値は，単一の負の演算子（-，9.9.4節参照）で作られる。厳密な意味では，負の記号は数値定数の一部ではない。

7.1.5 日時定数

日時定数には，付加的なタイムゾーンを有する日付と時刻の組み合わせ（T または t を区切り記号として使用）と付加的な秒以下の表現（.フォーマットを使用）を表すため，ISO の拡張フォーマットを使う（6.1.8節参照）。

7.1.5.1 秒以下の表現

秒以下の表現は, 小数点と一つ以上の数字をつけることで表現される (例えば, **1989-01-01T13:30:00.123**)。

7.1.5.2 タイムゾーン

現地のタイムゾーンがデフォルトである。ISO の Coordinated Universal Time (UTC)は, z を付すことで表される (例えば, **1989-01-01T13:30:00.123Z**)。現地のタイムゾーンは, UTC の前または後ろに+または-hh:mm を付すことで厳密に表現される。よって, EST (Eastern Standard Time, United States of America)タイムゾーンは **1989-01-01T13:30:00-05:00** や **1989-01-01T18:30:00Z**.と表現される。

7.1.5.3 日時の作成

演算子+は, 継続時間から日時を作成するために使用される。例えば, **1800-01-01 + (1993-1800)years + (5-1)months + (17-1)days** は **1993-05-17** を作成している。

7.1.6 文字列定数

文字列定数は, 引用 (", ASCII 34)で始まり終わる。例としては, 次の通りである:

```
"this is a string".
```

ここには, 文字列定数の制限はない。

7.1.6.1 引用符の内部使用

引用符を内部に含んだ文字列を表現するために, 二つ連続した引用符を使用する。例としては, 次の通りである:

```
"this string has one quotation mark: "" ".
```

7.1.6.2 単一の改行

文字列の中では, 単一の改行を含む余白 (5.3 節参照) は, 単一の空白文字に変換される。例としては, 次の通りである。

```
"this is a string with
one space between 'with' and 'one'"
```

7.1.6.3 複数の改行

文字列の中では, 二つ以上の改行を含む余白は, 単一の改行に変換される。

```
"this is a string with


one line break between 'with' and 'one'"
```

7.1.7 用語定数

用語定数はアポストロフィ(' :ASCII 39)で始まり終わる。そしてそれは, 有効な mlmname を含む。例としては, 次の通りある:

```
'mlm_name'
```

7.1.8 マッピング節

マッピング節は, **{**で始まり**}**で終わる文字列である（これら記号はそれぞれ ASCII 123 と ASCII 125 である）。マッピング節は, data スロットで機関固有の定義（例えばデータベース参照）を代入するために用いられる。中括弧の中に書くことに関する唯一の必須条件は, 中括弧が中括弧の中に書かれないことだけである。マッピング節の中のコメントと引用はこの標準では規定されない。ただし, この標準で提供されるものと同様であることが推奨されている。変数名に関する Arden syntax の規定, 例えば大小文字の同一視や **the** を余白とする取り扱いは, マッピング節ではなくても構わない。単語 **mapping** は（実装上の便法として）, 中括弧の中に Arden の変数を使っても良いが, Arden の変数を設定してはならない（Arden の変数は代入演算子の左辺に変数**<var>**(s)としてセットされなければならない）。これにより, 例え, 他の機関のコンパイラがマッピング節をスキップするように設定されていたとしても, MLM は他の機関で実行される前に多少の変更が必要になるかもしれない。

MLM の作成者は, MLM に使われるマッピング節の全てにコメントを付すよう強く推奨される。これにより, MLM の使用者は, MLM が配布･共有されたときにマッピング節の意図を理解できるようになる。UMLS Metathesaurus の識別語が, コメントの中のコンセプトを理解するのに有効かもしれない。MLM の使用者が簡単に MLM を修正できるよう, 作成者は説明とともに全ての変数や定数を data スロットに載せるべきである。

7.1.9 コメント

コメントは**/***で始まり***/**で終わる文字列である。コメントは, そのスロットがどのように働くかを示すが, 論理的には（**the** や他の余白のように）無視される。コメントは入れ子構造になってはならない（例えば, **/* A comment /* */** は単一のコメントである）。コメントは余白が先に付いたり後に付いたりする必要はない。よって, **x/**/y** は **x y** と同一である。

また, コメントは文字列//から行区切り（5.3 節参照）までとも特定される。もし//が現れたときには, 以降このラインの全て（*/も含めて）無視される。

7.1.10 余白

空白文字や改行, 行送り, 水平タブ, 垂直タブ, 頁送り, コメントは余白である。余白は, 他の文法上の要素を区分し, スロットを読み易くするために使われる。余白は, 文字や数字やアンダーバーで始まり終わる二つのトークンの間には必ず必要である（例えば, **if done** の場合）。また, 二つの文字列定数の間にも必要である。他のトークンの間では付加的である（例えば, **3+4** と **3 + 4** の場合）。これについては 5.4 節と7.1.1.1節も参照のこと。

7.2 構成

トークンは次の構成に整理されている：

7.2.1 文

構造化スロットは複数の文から成る。各々の文は, 論理上の制約や実行すべき事柄を特定している。一般に, 文はそれが表される順番に実行される。以下は文の例である。（各文の前のコメントは, 各文が何をするかを示している）：

```
/* 変数"var1"に 0 を代入 */
let var1 be 0;
/* MLM named "hyperkalemia"という名前の MLM を実行 */
call `hyperkalemia`;
/* カリウムが 5 よりも大きい場合, "真"と結論付け */
if potassium > 5.0 then
conclude true;
endif;
```


7.2.1.1 文の終了

スロットの最後以外の，全ての文はセミコロン（**;**）で終わらなければならない。よって，セミコロンは文の区切り文字として働く。もしスロットの最後の文がセミコロンで終わっている場合，それとダブルセミコロンとの間に少なくとも一つ余白がなければならない（**;;;**は無効だが，**;/**/;;**は有効）。 例えば，logic スロットは次のようである：

```
logic:
last_potas := last potas_list;
if last_potas > 5.0 then
conclude true;
endif;
```

文の文法は，各々のスロットに依存する。各々のスロットの詳細な記述方法については，10 章, 11 章, 9.15節,13 章を参照のこと。

7.2.2 式

文は予約語，特殊記号，式からなる。式はデータ値（8 章で定義されるうちの一つの型を持つ）を表す。式は次のいずれかよりなる：

7.2.2.1 定数

データ値は数値 **3** や日時 **1991-03-23T00:00:00** のように厳密に表現される。以下は，有効な式である：

```
null
true
345.4
"this is a string"
1991-05-01T23:12:23
```

7.2.2.2 変数

式の中に現れる識別語（7.1.2 節参照）は，変数を表す（7.2.3 節参照）。以下は有効な変数の例である：

```
var1
this_is_a_variable
a
```

7.2.2.3 演算子と被演算子

式は，演算子と一つ以上の被演算子として知られる副式からなってもよい。例えば，**3+4** において**+**は演算子で **3** と **4** は被演算子である。このような式の結果は，新たなデータ値である。この例では **7** がそれに当たる。式は，それ自体が他の式の被演算子になるように入れ子構造になってもよい。以下は有効な式の例である：

```
4 * cosine 5
var1 = 7 and var2 = 15
(4+3) * 7
```

演算子と先行子，連結子，括弧の詳細については，9.1 節を参照のこと。

7.2.3 変数群

一つの変数は，一時的にデータ値を保持するための領域である。変数群は，明示的には現れないが，最初に使われるときに暗黙のうちに現れる。変数は，代入文によってデータ値を代入される（10.2.1 節参照）。後で式の中で変数が使われるときには，代入されたデータ値を表す。例えば，**varl** は有効な変数名である。もしこれが代入される前に使用された場合，その値は **null** となる。

7.2.3.1 スコープ

変数のスコープは，個々のスロットではなく，MLM 全体である。MLMs は他の MLMs から直接変数を読むことはできない。よって，MLM で使われている変数は，呼ばれた MLMs では使うことができない（10.2.4 節参照）。非 Arden 変数は，マッピング節の中で参照されたり設定されたりしうる（11.2.3 節参照）。マッピング節の中では，Arden 変数は参照されうるが，設定はできない。Arden 変数名と非 Arden 変数名との間の矛盾を解決するのは，機関の責任範囲である。

7.2.3.2 特殊な変数

いくつかの変数，例えばイベント変数や MLM 変数，メッセージ変数，宛先変数は特殊な変数である。これらは特有の構造でのみ使用でき，一般の式では使用できない。こえら変数は，11 章で定義されるように特殊は代入文を使用する（これら特殊な代入文は，特殊な変数を記述するのと同義である）。特殊な変数は，文字列に変換し被演算子にすることが可能である。特殊な変数に関する有効な演算子は，**is [not] equal** (9.6.1節参照), = (9.5.1節参照), <> (9.5.2節参照)だけである。

8 データ型

MLM の基本的な機能は，患者データを取り込み，データを操作し，いくつかの結論を導き出し，できればある行動をとることである。データは，患者データベースへの直接的な参照要求や，MLM の中の定数，他のデータを演算した結果など様々な情報源から来る。

データ要素は，リストと呼ばれる順序付けられた（プライマリ日時ではなく，リストの中の位置で順番付けられた）集合に保持できる。リストについては，8.8 節に詳述する。

データは，いくつかのデータ型に分類される。

8.1 ヌル値

ヌル値は，不確定を表す特殊なデータ型である。このような不確定は，患者データベースに情報がないことか，もしくはデータベースに明確に **null** が充てられていることを示す。また，ヌル値は，型の不一致やゼロでの割り算など実行時のエラーの結果でもある。ヌル値は，スロットの中で単語 **null**（即ち，null 定数）を用いて厳密に特定されうる。以下の式はヌル値を結果とする：

```
/* explicit null */
    null
/* division by zero */
    3/0
/* addition of Boolean  */
    true + 3
```

8.2 ブール型

ブールデータ型は二つの論理値：**true** と **false** からなる。単語 **true** はブール型の真を，単語 false はブール型の偽を表す。

論理演算子は，**null** を不確定状態に充てることで三状態の論理を扱う。例えば，**true or nul**l は真である。**null** は不確定であるが，**true** を含む論理和はもう一つの被演算子が何であれ真となる。しかしながら，論理和における **false** は何の情報も付加しないので，**false or null** はヌル値である。これについては 9.4 節を参照のこと。

8.3 数値

ここでは一つの数値型しかない。すなわち，整数と実数の違いはない。数値定数（例えば，**3.4E-12**）は 7.1.4 節で定義されている。内部的には，全ての数値は実数として扱われる。例えば，**1/2** は **0.5** として計算される。

8.4 日時

日時データ型は，実時間の中の一点を指す。他の系ではタイムスタンプとも呼ばれる。日付と時刻が特定されなければならない。1800 年に至るまでサポートしなければならず，そして 1800-01-01 よりも前の時刻は無効である。日時定数（例えば，**1990-07-12T00:00:00**）は 7.1.5 節で定義されている。

8.4.1 粒度

時刻の粒度は常に無限小である（飛び飛びの秒ではない）。患者データベースに格納されている時刻は異なった粒度であるかもしれない。時刻が MLM に読み込まれたとき，常にその粒度の期間での最初の時点の値に設定される。。例えば，時刻が分単位でしか記録されていない場合，0 秒が想定される。また，日付のみが取られている場合，時刻は午前零時が想定される。

8.4.2 午前零時

午前零時（即ち，**T00:00:00** が時刻フィールドにある場合）は，新しい日付が始まったことを示す（終わったことではない）。

8.4.3 Now

単語 **now** は MLM が実行開始された時間を表す日時定数である。**Now** は MLM の実行中を通じて，一定である。すなわち，**now** が複数回使われても，同じ MLM の中では同じ値をとる。入れ子になった MLM の中で使われる **now** は，呼び出される日時を表すために異なった値をとる。

8.4.4 Eventtime

MLMs を起動する一つの方法にトリガーイベントがある。例えば，血清カリウムの患者データベースへの登録は，MLM を起動するイベントとなりうる。単語 **eventtime** はイベントが起きた日時（例えば，データベースが更新された日時）を表す。この **eventtime** は MLMs が時間遅れを伴って起動される場合に有効である。**Eventtime** を使えば，MLM は起動イベント後に何が起きたかを照会することができる。

8.4.5 Triggertime

もし MLM がイベントによって直接起動，もしくは他の MLM から起動された場合，**triggertime** と **eventtime** は同一である。もし，MLM が時間遅れを伴ったトリガー（13.3.2 節参照）や時間遅れを伴った MLM コール（12.2.4節参照）で起動される場合，**triggertime** は **eventtime** に時間遅れを足したものになる。**Triggertime** を使うことで，MLM は他の MLM をあたかもイベントによって直接起動しているようにトリガーをかけることができる。次の式が単一の MLM では成り立つ：**eventtime** ≤ **triggertime** ≤ **now**。

8.4.6 Currenttime

単語 **currenttime** は，MLM の実行中この単語が現れたシステム日時を表す。**Currenttime** は，**now** が MLM 実行中は定数なのと異なり，常に変化している。これにより，MLM の実行にかかっている時間は **currenttime** から **now** を引き算することで計算される。次の式が単一の MLM では成り立つ：**eventtime <= triggertime <= now <= currenttime**。

8.5 継続時間

継続時間データ型は, 実時刻の中で特に一点に固定されない時間範囲を指す。継続時間定数というものはない。その代わりに, 時間演算子を用いて継続時間を作成することができる (9.11 節参照)。例えば, **1 day**, **45 seconds**, **3.2 months** は継続時間である。

8.5.1 サブ型

継続時間データ型は二つのサブ型：months と seconds を持つ。このサブ型に区分けする理由は, 月や年の秒数は開始日付によって異なるからである。月や年の継続時間は, months で表される。秒や分, 時, 日, 週の継続時間は seconds で表される。ここには複雑な継続時間はなく, サブ型は months か seconds のどちらかであり, 両方ということはない。どちらの場合でも, 継続時間は実数として扱われる。

継続時間の印字 (いわゆる, 文字列による表現) は, 内部表現とは独立である。MLM の結果を読む医療提供者は, 時間に二つのサブ型があることに気づかないかもしれない。どのように継続時間が印字されるかは, 環境に依存する。例えば, 文字列による **6E+08 seconds** の表現は, **19.01 years** かもしれない。これについては, 9.8 節参照のこと。

8.5.2 日時と継続時間の数値演算

日時と継続時間の間の演算は以下のように実行される：

8.5.2.1 日時-日時

二つの時刻の差は, 秒単位の継続時間になる。例えば, **1990-03-01T00:00:00 - 1990-02-01T00:00:00** は **2419200 seconds** となる。

8.5.2.2 日時と秒

日時と継続時間の秒の和や差の演算結果は日時となる。演算は簡単である。すなわち, 日時はある時点 (例えば **1800-01-01T00:00:00**) からの経過秒数で表される。そして, 秒数が足され, もしくは引かれて日時となる。例えば, **1990-02-01T00:00:00 + 2419201 seconds** は **1990-03-01T00:00:01** となる。

8.5.2.3 日時と月

日時と継続時間の月との和や差の演算結果は日時となる。日時は日付と時刻で表される (例えば, **1991-01-31T00:00:00**)。月の部分に関しては, 日付の月の要素との和または差として計算される (すなわち, **1991-01**)。もし, 計算結果の日付が, 月の日数によって無効となる場合, 日付はその月の最後の有効な日付になる。例えば, **1991-01-31T00:00:00 + 1 month** は **1991-02-28T00:00:00** となる。もし月の部分が月単位以下の部分を含んでいる場合 (例えば, **1.1 months**) には, 整数の月が使われ (すなわち, **1 month** と **2 months**), そして補間が計算される (月の整数部分は足され, 月以下の部分は次の月に足され, 前の月から引かれる)。例えば, **1991-01-31T00:00:00 + 1.1 months** は **1991-02-28T00:00:00 + (0.1 * 2629746 seconds)**か **1991-03-03T01:02:54.6** となる。以下に例を説明する：

1991-01-31T00:00:00 + 1 month = 1991-02-28T00:00:00

かつ

0.1 Months * 2629746 seconds / month [from 8.5.2.4] = 262974.6 seconds

262974.6 seconds / (60 seconds / minute) / (1440 minutes /day) = 3.0436875 days

0.0436875 days * 1440 minutes / day = 62.91 minutes

= 1 hour, 2 minutes, 54.6 seconds.

よって

0.1 months = 3 days 1 hours 2 minutes 54.6 seconds

よって

1991-01-31T00:00:00 + 1.1 months = 1991-02-28T00:00:00 + 3 days 1 hour 2 minutes 54.6 seconds

= 1991-03-03T01:02:54.6

数値計算における和と差の演算とは異なり, 時間の和と差の演算は逆転できない。例えば次の通りである:

1993-01-31 + 1 month = 1993-02-28

1993-02-28 - 1 month = 1993-01-28 (3 days earlier)

演算順序は重要である: **(d+1 month)+1 day** は **d+(1 month+1 day)**の値とは異なる。

他の例を次に示す:

1991-01-31T00:00:00 - 2.1 months = 1990-11-26T22:57:05.4

1991-01-31T00:00:00 - 1.1 months = 1990-12-27T22:57:05.4

1991-04-30T00:00:00 - 0.1 months = 1991-04-26T22:57:05.4

8.5.2.4 月と秒

月と秒の演算では最初に次の変換定数:: 2629746 seconds/month (グレゴリオ暦での平均) を使って月に関する被演算子を秒に変換する。例えば, **1 month / 1 second** は **2629746** となる。

8.6 文字列

文字列は可変な長さを持つ文字の列である。文字列定数は 7.1.6 節で定義されている。例としては, 次の通りである:

"this is a string constant"

8.7 用語

現在, 用語は構造化スロットの中で mlmnames を表すためと構造化されたリンク情報のテキスト部分にしか使われていない。これは, **call** 文の中でしか使われていない (10.2.4 節参照)。将来, これらはコントロールされた語彙として使われる予定である。用語定数は 7.1.7 節で定義されている。例としては, 次の通りである:

'mlm_name2'

'http://www.nlm.nih.gov/'

8.8 リスト

リストは順序付けされた要素の集合である。要素としては, ヌル値, ブール型, イベント, 宛先, メッセージ, 用語, 数値, 日時, 継続時間や文字列などである。ここでは入れ子になったリストは扱わない。すなわち, リストは他のリストの要素にはならない。リストは異種のものからなっていてもよい。すなわち, リストの要素は異なった型であってよい。リスト定数としては, 空の括弧**()**で表される空リストがある。余白を空の括弧の間におくことは許されている。他のリストは, コンマ (**,**) などのリスト演算子を用いて単一の要素から作られる (9.2 節参照)。リスト (単一要素からなるリストも含めて) の出力形式は, 9.8 節参照のこと。例えば, 以下は有効なリストの例である:

4, 3, 5

3, true, 5, null

,1

()

リスト被演算子が使われるべき演算子に，リスト以外の被演算子が用いられているときには，その被演算子は暗黙的に単一の要素からなるリストに変換されて計算される。

8.9 照会結果

データベース照会の結果は，データ値に付加して日時の値も持つ。

Data スロットの照会は患者データベースか他のデータベース（例えば，語彙のデータベースや経営データベース）からデータを抽出する。照会結果は他のスロットで使われるため，変数に代入される。

8.9.1 プライマリ日時

患者データベースの全ての要素は，プライマリ日時（発生日時とも呼ばれる）を持っていると仮定されている。プライマリ日時は，照会に対して医療的に妥当性のある日時と定義される。実体が異なれば，それに応じてプライマリ日時は異なるかもしれない。例えば，血液検査のプライマリ日時は，患者から血液を採取した日時（もしくはそれに近い日時）である。しかし，投薬オーダのプライマリ日時は，オーダが発行された日時である。もしデータ要素に医療的に妥当性のある日時がない場合には，プライマリ日時は **eventtime**（情報が収集された日時）と同一とする。

暗黙のうちに患者データベースへの参照要求ではデータのプライマリ日時が要求されている。例えば，血清カリウムのリストを抽出する場合，データ値（血清カリウムの数値）と日時（例えば，血清が採取された日時）から成るペアを取得している。

8.9.2 抽出順序

参照要求の結果は，デフォルトではプライマリ日時の時間的順番に基づいてソートされる。参照要求では，他のソート方式を指定しても良い。

8.9.3 データ値

変数に参照要求の結果が代入されるとき，変数の使用にあたっては常にデータ値が参照される。例えば，もし **potas** が血清カリウムのリストを代入された変数としたとき，以下の文を用いて最新のカリウム計測の値をチェックすることができる：

```
￥if latest potas > 5.0 then
conclude true;
endif;
```

8.9.4 日時関数演算子

time 演算子 (9.17 節参照)を使うことで，変数やリスト要素に関連付けられたプライマリ日時を設定したり抽出したりできる。日時の抽出関数は9.17.1節に記されている。プライマリ日時の設定については，9.17.1節の第二段落に書かれている。例えば，以下の文を使って，最新のカリウム計測のプライマリ日時をチェックすることができる：

```
if time of latest potas is within the past 3 days then
conclude true;
endif ;
```

ここで，**eventtime** は，常に起動イベントのプライマリ日時であるとは限らない。例えば，血清カリウムの登録が MLM を起動する場合，**eventtime** はデータベースへの結果の登録日時を示す。しかし，この場合のプライマリ日時は，患者から血液を採取した日時である。

9 演算子解説

9.1 概要

演算子は式中でデータ操作のために用いられる。演算子は1つ以上の被演算子（データ値）を持ち，1つの結果（新たなデータ値）を生成する。以下の特性は，この節での演算子の定義に適用される。

9.1.1 被演算子数

演算子は1つ，2つ，または3つの被演算子を持つ。演算子の中には2通りの型を持つものがある：1つは1つの被演算子を持つ型であり，もうひとつは2つの被演算子を持つ型である。演算子は以下のように分類される。

単項演算子：被演算子1つ
二項演算子：被演算子2つ
三項演算子：被演算子3つ

9.1.2 データ型の制約

ほとんどの演算子は，データ型の全集合のうち，ある部分集合にのみ有効である。すべての演算子の定義には，被演算子の位置と，許されるデータ型についての制約の記述がある。その記述の一般的形式は以下の通りである：

<num:type> := <num:type> op <num:type>

この制約表現のうち，op が記述されている演算子を表す。

9.1.2.1

num は各々以下のいずれかである。

1---演算子は1つの要素を必要とする

k, m,または n --- 演算子は通常1つの被演算子を持つが，いかに記されている通り，0，1またはそれ以上の要素のリストが使われることがありうる。もし，データ型の制約中に，同じ文字が1回以上登場した場合には，その被演算子は，同じ数の要素をもたなければならない。その条件が満たされないときは演算子の結果は null となる。

9.1.2.2

type は各々次のいずれかである：

null—ヌルのデータタイプ

Boolean—Boolean data type

Boolean—ブール型データタイプ

number—数値データタイプ

time—日時データタイプ

duration—継続時間データタイプ

string—文字列データタイプ

item—式中では使われず，”call”ステートメントのみで用いられる（10.2.4 参照）

any-type —null、Boolean、number、time、duration,または string

non-null—Boolean、number、time、duration または string

ordered—number,、time、duration または string

9.1.2.3

:=の右側にある<num:type>は被演算子のデータ型を表す。もし，演算子が，この定義されたデータ型集合以外のデータ型をもった被演算子に適用された場合，結果は null となる。例えば，演算子**は，time データ型については定義されていないため，**3**1991-03-24T00:00:00** の結果は null となる。ほとんどの演算子について，null は被演算子集合に定義されていないため，null が被演算子となったときには結果も null となる。例えば，null は演算子+について定義されていないため，3+null の結果は null となる。

9.1.2.4

:=の左側にある<num:type>は結果のデータ型を表す。別途断りのない限り，演算子は通常の結果についての宣言にかかわらず，null を結果として返すことがある。

9.1.3 リストの扱い

別途断りのない限り，リストは次のように扱われる。

9.1.3.1

演算子のテンプレートが**<n:type> := op <n:type>** または **<n:type> := <n:type> op** の型を取るときは，算術演算子はリストの全ての要素に適用され，同じの数の要素をもつリストを結果として生成する。（もとのリストが空の場合，結果のリストも空となる。）例えば，**-(3,4,5)** の結果は **-3, -4, -5** である。

このような動作をする単項演算子は以下の通り：

not …
… is present
… is not present
… is null
… is not null
… is Boolean
… is not Boolean
… is number
… is not number
… is time
… is not time
… is duration
… is not duration
… is string
… is not string
+ …
- …
… ago
… year
… years
… month
… months
… day
… days
… hour

… hours
… minute
… minutes
… second
… seconds
time [of] …
arccos [of] …
arcsin [of] …
arctan [of] …
cos [of] …
cosine [of] …
sin [of] …
sine [of] …
tan [of] …
tangent [of] …
exp [of] …
truncate [of] …
floor [of] …
ceiling [of] …
log [of] …
log10 [of] …
abs [of] …
year [of] …
month [of] …
day [of] …
hour [of] …
minute [of] …
second [of] …

9.1.3.2

演算子のテンプレートが **<1:type> := op <n:type>** または **<1:type> := <n:type> op** の形をとるとき，演算子はリスト全体に適用され，単一の結果を生成する。例えば **max(3,4,5)**の結果は **5** である。

このような働きをする演算子は以下の通り：

count [of] …
exist [of] …
avg [of] …
average [of] …
median [of] …
sum [of] …
stddev [of] …
variance [of] …
any [of] …
all [of] …
no [of] …
min [of] …
minimum [of] …
max [of] …
maximum [of] …

last [of] …
first [of] …
earliest [of] …
latest [of] …
string [of] …
… is list
… is not list
index min [of] …
index minimum [of] …
index max [of] …
index maximum [of] …
index earliest [of] …
index latest [of] …

9.1.3.3

演算子のテンプレートが **<m:type> := op <n:type>** または **<m:type> := <n:type> op** の形をとるとき，演算子はﾘｽﾄ全体に適用され，別のﾘｽﾄを生成する。例えば，**increase(11,15,13,12)** の結果は **(4, -2, -1)**である。

このような働きをする単項演算子は以下の通り：

slope [of] …
increase [of] …
decrease [of] …
percent increase [of] …
% increase [of] …
percent decrease [of] …
% decrease [of] …
interval [of] …
extract characters [of] …
sort …
reverse …

9.1.3.4

演算子のテンプレートが **<n:type> := <n:type> op <n:type>**の形をとるときは，双方のﾘｽﾄの要素を対にして算術演算子を適応し，結果として同じ数の要素をもったﾘｽﾄを生成する。（ﾘｽﾄが空の時は，結果のﾘｽﾄも空になる。）たとえば，**(1,2)+(3,4)**の結果は**(4,6)** となり，**()+()** の結果は **()**となる。

もし，片方の被演算子が単一の要素であり，他方の被演算子が n 個の要素をもっていた場合には，単一の要素は n 個のﾘｽﾄに置換される。例えば，**1+(3,4)** は**(1,1)+(3,4)** と同等となり，その結果は **(4,5)**となる。

2つの被演算子の要素数が異なり，いずれかが単一の要素でなかった場合，結果は **null** となる。

このような働きをする二項演算子は以下の通り：

… or …
… and …
… = …
… eq …
… is …
… <> …
… ne …

… is not equal …

… < …

… lt …

… is less than …

… is not greater than or equal …

… <= …

… le …

… is less than or equal …

… is not greater than …

… > …

… gt …

… is greater than …

… is not less than or equal …

… >= …

… ge …

… is greater than or equal …

… is not less than …

… is within past …

… is not within past …

… is within same day as …

… is not within same day as …

… is before …

… is not before …

… is after …

… is not after …

… matches pattern …

… occur equal …

… occur within past …

… occur not within past …

… occur within same day as …

… occur not within same day as …

… occur before …

… occur not before …

… occur after …

… occur not after …

… + …

… - …

… * …

… / …

… ** …

… before …

… after …

… round …

次の演算子は **<n:type> := <m:type> op <m:type>**の形をとる;この演算子は必要に応じて被演算子を複製するが, 結果のﾘｽﾄは要素の数が違うことがありうる。

… where …

9.1.3.5

演算子のテンプレートが **<n:type> := <n:type> op$_1$ <n:type> op$_2$ <n:type>**の形をとるとき，算術演算子は，各ﾘｽﾄの要素の3個一組づつに対して適用され，同じ数の要素をもったﾘｽﾄを結果として生成する。(ﾘｽﾄが空の時は，結果のﾘｽﾄも空となる。) 例えば **(1,2) is within (0,2) to (3,4)** の結果は (**true,true)**となる。

被演算子のひとつが，単一要素であり，他の2つの被演算子が n 個の要素をもっている場合，単一要素の被演算子は n 要素のﾘｽﾄに複製される。被演算子の2つが単一要素であり，1つの被演算子が n 個の要素をもっている場合は，単一要素の被演算子2つが，n 要素のﾘｽﾄに複製される。例えば **(1,2) is within 2 to (3,4)**は，**(1,2) is within (2,2) to (3,4)**と同等であり，その結果は**(false,true)**となる。

どの被演算子2つの間での比較でも，その2つの要素数が異なり，そのうちのひとつが単一要素でないのであれば，その結果は **null** となる。

このような働きをする三項演算子は以下の通り：

… is within … to …
… is not within … to …
… is within … preceding …
… is not within … preceding …
… is within … following …
… is not within … following …
… is within … surrounding …
… is not within … surrounding …
… occur within … to …
… occur not within … to …
… occur within … preceding …
… occur not within … preceding …
… occur within … following …
… occur not within … following …
… occur within … surrounding …
… occur not within … surrounding …

9.1.3.6

演算子のテンプレートが**<n:type> := op$_1$ <1:type> op$_2$ <m:type>**の形をとるときは，演算子は 2 番目の被演算子全体に適用され，新しいﾘｽﾄを生成する。1番目の被演算子は単一要素である必要がある。(そうでなければ，演算子の結果は **null** となる。) 例えば，**min 2 from (5,3,4)** の結果は(**3, 4)**となる。

このような働きをする二項演算子は以下の通り：

min … from …
minimum … from …
max … from …
maximum … from …
last … from …
first … from …
latest … from …
earliest … from …
index min … from …
index minimum … from …
index max … from …
index maximum … from …

index last … from …

index first … from …

index latest … from …

index earliest … from …

9.1.3.7

演算子のテンプレートが**<n:type> := op$_1$ <n:type> op$_2$ <m:type>**の形をとるときは，演算子は2番目の被演算子全体に適用され，新しいリストを生成する。1番目の被演算子は典型的には単一要素である。例えば，**1 is in (0,3)** の結果は，**false** であり，**(1,2,3) is in (0,3)**の結果は(**false,false,true)**となる。

このような働きをする二項演算子は以下の通り：

nearest … from …

… is in …

… is not in …

index nearest … from …

9.1.3.8

演算子のテンプレートが**<n:type> := <k:type> op <m:type>**の形をとるときは，演算子は2つの被リスト全体に適用され，新しいリストを生成する。例えば **1,(3,4)** の結果は **(1,3,4)**となる。

このような働きをする二項演算子は以下の通り：

… , …

… merge …

… || …

… seqto …

9.1.4 プライマリ日時の取り扱い

クエリーはその結果にプライマリ日時を付与する。（8.9.1 参照）．演算子によってはプライマリ日時を保持し，またある演算子はプライマリ日時を破棄する。別途宣言されない限り，プライマリ日時は以下のように取り扱われる。

9.1.4.1 単項演算子

単項演算子はプライマリ日時を保持する。以下の例では，**data1** があるクエリーの結果であるならば，**result1** はプライマリ日時を保持している：

result1 := sin(data1);

9.1.4.2 二項および三項演算子

二項および三項演算子は，全ての被演算子がプライマリ日時を保持し，そのプライマリ日時が等しいときプライマリ日時を保持する。もし被演算子のいずれかがプライマリ日時を持たず，あるいは違ったプライマリ日時を持っている場合には，プライマリ日時は破棄される。

例（プライマリ日時が等しく，プライマリ日時は保持される）：

データの値：6 := 2 * 3;

日時の値： (1/1) (1/1) (1/1);

例（プライマリ日時が異なり，プライマリ日時は破棄される）：

データの値： 42 := 6 * 7;

日時の値: (ヌル) (2/1) (1/1);

9.1.5 演算子の優先順位

式は入れ子の構造であり，2つ以上の演算子と多数の被演算子を含むことがある。演算子が実行される順序は優先順位と呼ばれる属性によって決定される。演算子は複数の優先順位グループに分類される。優先順位の高い演算子は，優先順位の低い演算子よりも先に実行される。例えば，式 **3+4*5**（3足す4かける5）は次のように実行される：***** は+より優先順位が高いため，最初に実行される。そのため **4*5** の結果として **20** が得られる。次に **+**が実行され **3+20** の結果として **23** が得られる。どのような場合にも，演算子の優先順位を変更するためにかっこを使うことができる。

9.1.5.1 Precedence Table 優先順位表

演算子を優先順位ごとにグループ分けしたものが付録A4にある。

9.1.6 結合性

式に同じ優先順位グループに属する演算子が複数あった場合に，演算子の結合性の属性によって実行順序が決定される。個々の演算子の結合性は付録 A4に示されている。結合性には次の 3 つのタイプがある：

9.1.6.1 左結合

左結合の演算子は左から右へ実行される。例えば，**3-4-5** には2つの引き算(-)がある。同じ演算なので当然同じ優先順位グループに属する。**-**は左結合なので，**3-4** が最初に実行され，**(-1)**が結果として得られ，その次に **(-1)-5** が実行され結果は(**-6**)となる。

9.1.6.2 右結合

右結合の演算子は右から左へ実行される。例えば **average sum 3** は同じ優先順位グループに属する2つの演算子を持つ。どちらも右結合なので，**sum 3** が最初に実行され，結果は **3** となり，次に **average 3** が実行され，結果は **3** となる。

9.1.6.3 非結合

非結合の算術演算子は，括弧を使用せずに同じ優先順位グループから2つの演算子を使うことは出来ない。したがって，******（累乗の演算子）が非結合であるため，式 **2******3******4** は不正である。（しかしながら，**(2******3)******4** と **2******(3******4)** は両方とも正当な式である。）

9.1.7 括弧

実行順序を強制的に変えるために括弧を使うことができる。括弧に囲まれた式は，常に括弧外の式よりも先に実行される。例えば，**(3+4)*5** は次のように実行される：**3+4** は括弧の中なので優先順位と無関係に最初に実行され，結果は **7** となる。次に***** が実行され，**7*5** の結果として **35** が得られる。同様に **(2******3)******4** は正当な式であり，結果は **4096** となる。

9.2 リスト演算子

リスト演算子はデフォルトのリスト操作のパターンには従わない。プライマリ日時の保持については，別途記載がない限り，9.1.4 節の記述に従う。

9.2.1 , (二項演算子, 右結合)

二項演算子**,** (リスト結合) は2つのリストを結合する。リストに含まれる個々の要素のプライマリ日時は保持される。使用方法は以下の通り:

<n:any-type> := <k:any-type> , <m:any-type>

(4,2) := 4, 2

(4,"a",null) := (4,"a") , null

9.2.2 , (単項演算子, 非結合)

単項演算子 **,** は単一要素を長さ1のリストに変換する。被演算子が既にリストであるときは, 何もしない。使用方法は以下の通り (**(3)** は, 3 だけを要素とするリストである):

<1:any-type> := , <1:any-type>

(,3) := , 3

9.2.3 Merge (二項演算子, 左結合)

merge 演算子は2つのリストの結合, 単一要素のリストへの追加または2つの単一項目からのリストの生成を行う。次にその結果をプライマリ日時によって (9.2.4 節に示した通り) 時系列順にソートする。2つのリストの要素は全てプライマリ日時を持たなければならず, そうでなければ, 結果は **null** となる。(**x where time of it is present** という構文を使うことで, プライマリ日時を持つ **x** の要素のみを得ることができる。) プライマリ日時は保持される。**Merge** の典型的な使い方は, 別々の検索の結果を一体にすることである。**x merge y** と言う構文は **sort time (x,y)**と同値である。使い方は以下の通り (**data1** はデータ値が **2** であり**,** 時間が **1991-01-02T00:00:00** のリスト, **data2** はデータ値 **1,3** を持ちプライマリ日時として **1991-01-01T00:00:00**, **1991-01-03T00:00:00** を持つ):

<n:any-type> := <k:any-type> MERGE <m:any-type>

(1, 2, 3) := data1 MERGE data2

null := (4,3) MERGE (2,1)

9.2.4 Sort (単項演算子, 非結合)

Sort 演算子は要素のキーを基準にしてリストを並べ替える。その際のキーは 要素の値 (キーワードは **data**) またはプライマリ日時 (キーワードは **time**) である。Sort 演算子には, 省略可能な修飾子を使用することができる。使用する場合には修飾子はキーワード **sort** の直後に置く必要がある。以下のキーワードをキーワード **sort** の後に置くことができる: **data** または **time** のどちらか一方のみ。修飾子が用いられなけば, sort 演算子はデフォルトのデータに基づくソートが選択される。ソートの方向は常に昇順である。降順のソートをするためには, **reverse** を使うことができる。

Sort の修飾子は優先順位の判断では sort 演算子の一部と見なされる。このことは **time [of]** 演算子 (9.17.1)との潜在的な衝突の問題を解決する。したがって**"sort time x"** という構文は, should be parsed as「リスト x を時刻でソートせよ」と解釈し, 「リスト x から各要素のプライマリ日時を抽出し, 時刻のリストをソートせよ。」と解釈してはならない。

プライマリ日時によるソートを行う時に, 要素のうち, いずれかひとつでもプライマリ日時をもっていない場合には結果は **null** となる。(この動作を望まない場合には, いつでも, ソートの被演算子を **where time of it is present** でふるいにかけることができる。) 同一のキーを持つ要素は, 被演算子に現れる順序と同じ順序に保たれる。要素のキーのうち, 任意の2つからなる組のいずれかが, タイプの不一致により比較できない場合には, **sort** の結果は **null** となる。(したがって, データによるソートの場合, null がひとつでもあると, (または比較できない値がひとつでもあると) 結果は **null** となる; 時刻によるソートの場合, null のプライマリ日時がひとつでもあると結果は **null** となる。) この演算子の使用方法は以下の通り (**data1** はデータ値 **30,10,20** を持ち, 時刻の値は **1991-01-01T00:00:00**, **1991-02-01T00:00:00**, **1991-01-03T00:00:00** を持つものとする):

<n:any-type> := SORT <n:any-type>

```
<n:any-type> := SORT [DATA | TIME ] <n:any-type>
 (10, 20, 30) := SORT  DATA  data1
(30, 20, 10) := REVERSE (SORT  DATA  data1)
null := SORT  DATA  (3,1,2,null)
null := SORT  DATA  (3,"abc")
() := SORT  TIME  ()
(1, 2, 3, 3) := SORT  (1,3,2,3)
(30, 20, 10) := SORT  TIME  data1
```

9.3 Where 演算子

where 演算子はデフォルトのリスト操作や時刻操作のパターンに従わない。

9.3.1 Where （二項演算子, 非結合）

where 演算子はその左項の被演算子に対して関係演算における **select ... where ...** の操作を行う。一般的に左項の被演算子はリストであり, しばしばデータベースに対するクエリーの結果である。右項の被演算子は通常 Boolean 型であり（ただし必須ではない）, 左項のリストと同じ長さでなければならない。結果は左項の被演算子の要素のうち, 対応する右項の被演算子の要素の論理値が **true** であるもののみを取り出したものとなる。もし右項の被演算子が, それ以外の値, **false**, **null** あるいはその他のどんなものでも, 対応する左項の被演算子の要素は取り除かれる。**where** 演算子は **WHERE** の左項の被演算子のプライマリ日時を保持する。**WHERE** の右項のプライマリ日時は取り除かれる。その使用方法は以下の通り:

```
<n:any-type> := <m:any-type> WHERE <m:any-type>
(10,30) := (10,20,30,40) WHERE (true,false,true,3)
```

例

7.38 := (7.34, 7.38, 7.4) WHERE time of it is within 20 minutes after time of VentChange

(1/1 16:20) (1/1 18:01) (1/1 16:20) (Jan 1 02:06) (Jan 1 16:12)

Where 演算子は単一項目とリストの混合を他の二項演算子と同様なやりかたで取り扱う。もし **where** の右項の被演算子が単一項目である時, それが **true** である場合には, 左項の被演算子全体が（リストであろうと単一項目であろうと）そのまま保持される。右項の被演算子が **true** でないのであれば, 結果は空のリストとなる。左項の被演算子のみが単一項目であるのであれば, 結果は, その単一項目を, 右項の被演算子のうち, **true** に等しいものの数だけ繰り返したリストとなる。2つの被演算子が長さの違うリストであった場合には, 単一項目の **null** の結果を返す。（必要に応じて, 単一項目のコピーに関する 9.1.3.4 節で記述したルールが適用される。）例えば:

```
1 := 1 WHERE true
(1,2,3) := (1,2,3) WHERE true
(1,1) := 1 WHERE (true,false,true)
null := (1,2,3,4) WHERE (true,false,true)
```

Where は一般的にリストから特定の項目を選択するために使われる。左項の被演算子にはリストが使われ, 右項の被演算子のリストでは, いくつかの比較演算子が使用される。例えば, **potassium_list where potassium_list > 5.0** は 5 より大きい値がリストから選択される。

Where は不正な値をふるい落とすのに使うことができる。例えばクエリーの結果が数値かテキストのコメントのいずれかを返す場合, 次の構文を, クエリー結果の要素から求める数値のみを選択するのに使用できる:

```
queryResult where they are number
```

同様にクエリーの結果のうち, プライマリ日時を持たないものが存在するときに, 次の構文を, クエリー結果の要素から適切なプライマリ日時を持つもののみを選ぶのに使用できる:


```
queryResult where time of it is present
```

この例では, 単項演算子 **time** が queryResult（これが"**it**"の値となっている）に適用されて, 結果として時刻（プライマリ日時を持つ結果について）と null（プライマリ日時を持たない結果について）からなるリストを生成する 。次に単項演算子 **is present** がそのリストに適用されて, 規準自国があるものについては true, プライマリ日時のないものについては falseとなるブーリアンのリストを生成する。最後に **where** 演算子がプライマリ日時を持たない値を取り除くのに用いられる。

9.3.1.1 It

単語 **it** およびその同義語 **they** が **where** とともに用いられる。 **where** 構文を単純にするために, 右項の被演算子の中で, 左項の被演算子全体をあらわすのに, **it** を使うことができる。例えば **potassium_list where they > 5.0** は, リストの値の中で 5 より大きいものを選択する。 左項が複雑な構文であるときに, **It** は最も有用である;例えば **(potassium_list + sodium_list/3) where it > 5.0** は, 括弧の中の構文全体が **it** で表せる。**where** 構文が入れ子になっている場合には **it** は最も内側にある **where** の左項の被演算子を参照する。**it** が **where** 構文の外側で用いられた場合には, it は **null** の値をとる。Arden Syntax の実装においては, **where** 構文の外で使われた **it** をコンパイル時のエラーとして知らせるようにしてもよい。

9.4 論理演算子

9.4.1 Or （二項演算子, 左結合）

or 演算子は2つの被演算子の論理和を実行する。どちらかの被演算子が **true** であれば（たとえ他方が Boolean でなくても）結果は **true** となる。両方の被演算子が **false** であるときには結果は **false** となる。その他の場合には結果は **null** となる。この演算子の使用方法は以下の通り:

```
<1:Boolean> := <1:any-type> OR <1:any-type>
true := true OR false
false := false OR false
true := true OR null
null := false OR null
null := false OR 3.4
(true, true) := (true, false) OR (false, true)
() := () OR ()
```

真理値表を以下に示す。**Other** は次のデータ型のいずれかを表す: null, number, time, duration, or string.

	OR	TRUE	FALSE	Other	（被演算子右）
（被演算子左）	TRUE	TRUE	TRUE	TRUE	
	FALSE	TRUE	FALSE	NULL	
	Other	TRUE	NULL	NULL	

9.4.2 And （二項演算子, 左結合）

and 演算子は2つの被演算子の論理積を実行する。どちらかの被演算子が **false** であれば（たとえ他方が Boolean 型でなくとも）結果は **false** となる。両方の被演算子が **true** であれば, **結果は true** となる。その他の場合は, 結果は **null** となる。その使用方法は以下の通り:

<n:Boolean> := <n:any-type> AND <n:any-type>

false := true AND false

null := true AND null

false := false AND null

真理値表を以下に示す。**Other** は次のデータ型のいずれかを表す：null, number, time, duration, or string.

	AND	TRUE	FALSE	Other	(被演算子右)
(被演算子左)	TRUE	TRUE	FALSE	NULL	
	FALSE	FALSE	FALSE	FALSE	
	Other	NULL	FALSE	NULL	

9.4.3 Not (単項演算子, 非結合)

Not 演算子は被演算子の論理否定を実行する。**True** は **false** となり，**false** は **true** となりその他の場合は **null** となる。その使用方法は以下の通り：

<n:Boolean> := NOT <n:any-type>

true := NOT false

null := NOT null

真理値表を以下に示す。**Other** は次のデータ型のいずれかを表す：null, number, time, duration, または string.

NOT	TRUE	FALSE	other
	FALSE	TRUE	NULL

9.5 単純比較演算子

9.5.1 = (二項演算子, 非結合)

=演算子には2つの同義語がある：**eq** と **is equal** である。被演算子の等価性をチェックし **true** または **false** を返す。被演算子の型が異なる場合は **false** を返す。一方の被演算子が **null** である場合，つねに結果は **null** となる。 プライマリ日時は透過性の決定には使われない。結果の透過性は 9.1.4 節のルールで決定される。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:non-null> = <n:non-null>

false := 1 = 2

(null,true,false) := (1,2,"a") = (null,2,3)

null := (3/0) = (3/0)

null := 5 = ()

null := (1,2,3) = ()

() := null = ()

() := () = ()

null := 5 = null

(null,null,null) := (1,2,3) = null

null := null = null

(true,true,false) := (1,2,3) = (1,2,4)

被演算子が **null** であるかどうかをテストするには, =ではなく, **is present** または **exists** を用いる。節9.6.15 または 9.12.3を参照のこと。

9.5.2 <> （二項演算子, 非結合）

<> 演算子は 2 つの同義語を持つ：**ne** および **is not equal** である。被演算子の非等価性をチェックし, **true** または **false** を返す。被演算子が異なるデータ型を持つ場合には **true** が返される。ひとつの被演算子が **null** である場合には, **null** が返される。個の演算子の使用方法は以下の通り：

<n:Boolean> := <n:non-null> <> <n:non-null>

true := 1 <> 2

(null,false,true) := (1,2,"a") <> (null,2,3)

null := (3/0) <> (3/0)

9.5.3 < （二項演算子, 非結合）

< 演算子は3つの同義語を持つ：**It**, **is less than** および **is not greater than or equal** である。この演算子は ordered データ型について用いられる。もしタイプが一致しない場合, 結果は **null** となる。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:ordered> < <n:ordered>

true := 1 < 2

true := 1990-03-02T00:00:00 < 1990-03-10T00:00:00

true := 2 days < 1 year

true := "aaa" < "aab"

null := "aaa" < 1

9.5.4 <= （二項演算子, 非結合）

<= 演算子は3つの同義語を持つ：**le**, **is less than or equal**,および **is not greater than** である。この演算子は ordered データ型について用いられる。もしタイプが一致しない場合, 結果は **null** となる。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:ordered> <= <n:ordered>

true := 1 <= 2

true := 1990-03-02T00:00:00 <= 1990-03-10T00:00:00

true := 2 days <= 1 year

true := "aaa" <= "aab"

null := "aaa" <= 1

9.5.5 > （二項演算子, 非結合）

> 演算子は3つの同義語を持つ：**gt**, **is greater than** および **is not less than or equal** である。この演算子は ordered データ型について用いられる。もしタイプが一致しない場合, 結果は **null** となる。この演算子の使用方法は以下の通り：

<n:Boolean> := < n:ordered> > <n:ordered>

false := 1 > 2

false := 1990-03-02T00:00:00 > 1990-03-10T00:00:00

false := 2 days > 1 year

false := "aaa" > "aab"

null := "aaa" > 1

9.5.6 >= （二項演算子, 非結合）

>=演算子は3つの同義語を持つ：**ge**, **is greater than or equal** および **is not less tha** である。この演算子は ordered データ型について用いられる。もしタイプが一致しない場合, 結果は **null** となる。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:ordered> >= <n:ordered>

false := 1 >= 2

false := 1990-03-02T00:00:00 >= 1990-03-10T00:00:00

false := 2 days >= 1 year

false := "aaa" >= "aab"

null := "aaa" >= 1

9.6 Is 比較演算子

以下の比較演算子はisという単語を含むが, これはare, wasあるいはwereも包含している。isの後には, notを置くことができ, 結果を反転することができる。（notの定義の使い方については 9.4.3 節参照）例えば以下の例は有効である。

surgery_time WAS BEFORE discharge_time

手術時間 WAS BEFORE 退院時間

surgery_time IS NOT AFTER discharge_time

手術時間 IS NOT AFTER 退院時間

9.6.1 Is [not] Equal （二項演算子, 非結合）

9.5.1節を参照。

9.6.2 Is [not] Less Than （二項演算子, 非結合）

9.5.3節を参照。

9.6.3 Is [not] Greater Than （二項演算子, 非結合）

9.5.5節を参照。

9.6.4 Is [not] Less Than or Equal （二項演算子, 非結合）

9.5.4節を参照。

9.6.5 Is [not] Greater Than or Equal （二項演算子, 非結合）

9.5.6節を参照。

9.6.6 Is [not] Within ... To （三項演算子, 非結合）

is within ... to 演算子は, 1番目の被演算子が, 2番目, 3番目の被演算子で規定される範囲に入っているかをチェックする。範囲には両端を含む。この演算子は ordered データ型について用いられる。もしタイプが一致しない場合, 結果は **null** となる。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:ordered> IS WITHIN <n:ordered> TO <n:ordered>

true := 3 IS WITHIN 2 TO 5

true := 1990-03-10T00:00:00 IS WITHIN 1990-03-05T00:00:00 TO 1990-03-15T00:00:00

true := 3 days IS WITHIN 2 days TO 5 months

true := "ccc" IS WITHIN "a" TO "d"

9.6.7 Is [not] Within ... Preceding （三項演算子, 非結合）

is within ... preceding 演算子は左項の被演算子が, 次の2つの被演算子で規定される期間（両端を含む）に含まれるかをチェックする。（期間は, 3番目の被演算子から2番目の被演算子を引いたものから3番目の被演算子の間で規定される。）この演算子の使用方法は以下の通り：

<n:Boolean> := <n:time> IS WITHIN <n:duration> PRECEDING <n:time>

true := 1990-03-08T00:00:00 IS WITHIN 3 days PRECEDING 1990-03-10T00:00:00

9.6.8 Is [not] Within ... Following （三項演算子, 非結合）

is within ... following 演算子は左項の被演算子が, 次の2つの被演算子で規定される期間（両端を含む）に含まれるかをチェックする。（期間は, 3番目の被演算子から, 3番目の被演算子と2番目の被演算子を足したものの間で規定される。）この演算子の使用方法は以下の通り：

<n:Boolean> := <n:time> IS WITHIN <n:duration> FOLLOWING <n:time>

false := 1990-03-08T00:00:00 IS WITHIN 3 days FOLLOWING 1990-03-10T00:00:00

9.6.9 Is [not] Within ... Surrounding （三項演算子, 非結合）

is within ... surrounding 演算子は左項の被演算子が, 次の2つの被演算子で規定される期間（両端を含む）に含まれるかをチェックする。（期間は, 3番目の被演算子から2番目の被演算子を引いたものと, 3番目の被演算子と2番目の被演算子を足したものの間で規定される。）この演算子の使用方法は以下の通り：

<n:Boolean> := <n:time> IS WITHIN <n:duration> SURROUNDING <n:time>

true := 1990-03-08T00:00:00 IS WITHIN 3 days SURROUNDING 1990-03-10T00:00:00

9.6.10 Is [not] Within Past （三項演算子, 非結合）

is within past 演算子は左項の被演算子が, 右項の被演算子で規定される時刻の範囲に入るかどうかをチェックする。（期間は, **now** から右項の被演算子を引いたものと **now** の間で規定される。） この演算子の使用方法は以下の通り（**now** は 1990-03-09T00:00:00 であると仮定する）：

<n:Boolean> := <n:time> IS WITHIN PAST <n:duration>

true := 1990-03-08T00:00:00 IS WITHIN PAST 3 days

9.6.11 Is [not] Within Same Day As （二項演算子, 非結合）

is within same day as 演算子は左項の被演算子が, 右項の被演算子と同じ日であるかどうかをチェックする。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:time> IS WITHIN SAME DAY AS <n:time>

true := 1990-03-08T11:11:11 IS WITHIN SAME DAY AS 1990-03-08T01:01:01

9.6.12 Is [not] Before （二項演算子, 非結合）

is before 演算子は左項の被演算子が, 2番目の被演算子より前の時刻であるかどうかをチェックする。比較は同値を含まない。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:time> IS BEFORE <n:time>

```
false := 1990-03-08T00:00:00 IS BEFORE 1990-03-07T00:00:00
false := 1990-03-08T00:00:00 IS BEFORE 1990-03-08T00:00:00
```

9.6.13 Is [not] After （二項演算子, 非結合）

is after 演算子は左項の被演算子が, 2番目の被演算子より後の時刻であるかどうかをチェックする。比較は同値を含まない。この演算子の使用方法は以下の通り:

```
<n:Boolean> := <n:time> IS AFTER <n:time>
true := 1990-03-08T00:00:00 IS AFTER 1990-03-07T00:00:00
```

9.6.14 Is [not] In （二項演算子, 非結合）

is in 演算子は, デフォルトのリスト操作に従わない。この演算子は左項の被演算子が, 右項の被演算子への所属状況かどうかをチェックする。なお右項は通常, リストである。左項の被演算子がリストであれば, 結果はリストとなり, 左項の被演算子が単一項目であれば, 結果は単一項目となる。右項の被演算子が単一項目であるときは, 長さ1のリストとして扱われる。プライマリ日時は両者が一致した場合のみ保持される。(すなわち, = 演算子が所属状況のチェックに用いられた場合, null 以外が一致する。) この演算子の使用方法は以下の通り:

```
<n:Boolean> := <n:any-type> IS IN <m:any-type>
false := 2 IS IN (4,5,6)
(false,true) := (3,4) IS IN (4,5,6)
true := null is in (1/0,2)
```

9.6.23. 節も参照のこと。

9.6.15 Is [not] Present （単項演算子, 非結合）

is present 演算子は1つの同意語を持つ: **is not null** である。(同様に **is not present** も 1 つの同意語を持つ: **is null** である。) この演算子は被演算子が **null** でないときに **true** となり, 被演算子が **null** であるときは, **false** となる。**Is present** は **null** を返すことはない。**arg=null** は **arg** に関わらず常に結果は **null** となるので, 被演算子が **null** であるかをテストするのにはこの演算子を用いる。この演算子の使用方法は以下の通り:

```
<n:Boolean> := <n:any-type> IS PRESENT
true := 3 IS PRESENT
false := null IS PRESENT
(true,false) := (3,null) IS PRESENT
(false,true) := (3,null) IS NULL
```

9.6.16 Is [not] Null （単項演算子, 非結合）

9.6.15節を参照。

9.6.17 Is [not] Boolean （単項演算子, 非結合）

is Boolean 演算子は被演算子のデータタイプが Boolean であるときに **true** を返す。それ以外の場合は **false** を返す。**Is Boolean** は **null** を返すことはない。この演算子の使用方法は以下の通り:

```
<n:Boolean> := <n:any-type> IS BOOLEAN
true := false IS BOOLEAN
true := 3 IS NOT BOOLEAN
(false,true,false) := (null,false,3) IS BOOLEAN
```

9.6.18 Is [not] Number （単項演算子, 非結合）

is number 演算子は被演算子のデータタイプが number であるときに **true** を返す。それ以外の場合は **false** を返す。**is number** は **null** を返すことはない。この演算子の使用方法は以下の通り：

```
<n:Boolean> := <n:any-type> IS NUMBER
true := 3 IS NUMBER
false := null IS NUMBER
```

is number は集合演算子を適用する前にリストの要素がすべて数値であることを保証するのに有用である。これによって **null** が返されることを避けることができる。たとえば：

```
sum(serum_K where it IS NUMBER)
```

9.6.19 Is [not] String （単項演算子, 非結合）

is string 演算子は被演算子のデータタイプが string であるときに **true** を返す。それ以外の場合は **false** を返す。**is string** は **null** を返すことはない。この演算子の使用方法は以下の通り：

```
<n:Boolean> := <n:any-type> IS STRING
true := "asdf" IS STRING
false := null IS STRING
```

9.6.20 Is [not] Time （単項演算子, 非結合）

is time 演算子は被演算子のデータタイプが string であるときに **true** を返す。それ以外の場合は **false** を返す。**is time** は **null** を返すことはない。この演算子の使用方法は以下の通り：

```
<n:Boolean> := <n:any-type> IS TIME
true := 1991-03-12T00:00:00 IS TIME
false := null IS TIME
```

9.6.21 Is [not] Duration （単項演算子, 非結合）

is duration 演算子は被演算子のデータタイプが duration であるときに **true** を返す。それ以外の場合は **false** を返す。**is duration** は **null** を返すことはない。この演算子の使用方法は以下の通り：

```
<n:Boolean> := <n:any-type> IS DURATION
true := (3 days) IS DURATION
false := null IS DURATION
```

9.6.22 Is [not] List （単項演算子, 非結合）

is list 演算子は被演算子のデータタイプが duration であるときに **true** を返す。それ以外の場合は **false** を返す。**is list** は **null** を返すことはない。この演算子の使用方法は以下の通り：

```
<1:Boolean> := <n:any-type> IS LIST
true := (3, 2, 1) IS LIST
False := 5 IS LIST
false := null IS LIST
```

is list 演算子はデフォルトのリスト操作のパターンに従わない。それはこの演算子が被演算子の個々の要素を操作するのではなく，被演算子の全体を評価するからである。したがってリストを結果として返すことはない。次の違いに注意すること：

```
true := (3, 2, "asdf") IS LIST
(true, true, false) := (3, 2, "asdf") IS NUMBER
```

9.6.23 [not] In （二項演算子、非結合）

演算子 **in** は **is in** の同義語であり，同様のふるまいをする。この演算子の使用方法は以下の通り：

<n:Boolean> := <n:any-type> IN <m:any-type>
false := 2 IN (4,5,6)
(false,true) := (3,4) IN (4,5,6)
true := null in (1/0,2)

9.6.14節も参照のこと。

9.7 Occur 比較演算子

9.7.1 概要

次の比較演算子は 9.6 節の **is** 比較演算子と類似している。ここでは is の替わりに occur が使われる。**Occur** という語は **occurs** または **occured** となることもある。その結果を否定する場合は，**not** オプションを続けることができる（**not** の定義を使用。9.4.3 節参照）。

その効果は，左の被演算子を直接使用するのではなく，左の被演算子のプライマリ日時がかわりに使用される。（すなわち，左の被演算子の time が使用される。9.17 節参照）次の各ペアは同じ表現である：

time of var IS NOT BEFORE 1990-03-05T11:11:11
var OCCURRED NOT BEFORE 1990-03-05T11:11:11

time of surgery IS WITHIN THE PAST 3 days
surgery OCCURRED WITHIN THE PAST 3 days

time(a) IS WITHIN 1990-03-05T11:11:11 TO time(b)
a OCCURRED WITHIN 1990-03-05T11:11:11 TO time(b)

次のような演算子の例において，**data** は問い合わせの結果であり，そのプライマリ日時は 1990-03-05T11:11:11，**now** は 1990-03-06T00:00:00 である。

9.7.2 Occur [not] Equal (二項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR EQUAL <n:time>
false := data OCCURED EQUAL 1990-03-01T00:00:00

9.7.11節参照

9.7.3 Occur [not] Within ... To (三項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR WITHIN <n:time> TO <n:time>
true := data OCCURED WITHIN 1990-03-01T00:00:00 TO 1990-03-11T00:00:00

9.7.4 Occur [not] Within ... Preceding (三項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR WITHIN <n:duration> PRECEDING <n:time>
false := data OCCURRED WITHIN 3 days PRECEDING 1990-03-10T00:00:00

9.7.5 Occur [not] Within ... Following (三項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR WITHIN <n:duration> FOLLOWING <n:time>
false := data OCCURRED WITHIN 3 days FOLLOWING 1990-03-10T00:00:00

9.7.6 Occur [not] Within . . . Surrounding (三項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR WITHIN <n:duration> SURROUNDING <n:time>

false := data OCCURED WITHIN 3 days SURROUNDING 1990-03-10T00:00:00

9.7.7 Occur [not] Within Past (二項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR WITHIN PAST <n:duration>

true := data OCCURED WITHIN PAST 3 days

9.7.8 Occur [not] Within Same Day As (二項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR WITHIN SAME DAY AS <n:time>

false := data OCCURED WITHIN SAME DAY AS 1990-03-08T01:01:01

9.7.9 Occur [not] Before (二項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR BEFORE <n:time>

true := data OCCURRED BEFORE 1990-03-08T01:01:01

9.7.10 Occur [not] After (二項演算子, 非結合):

<n:Boolean> := <n:any-type> OCCUR AFTER <n:time>

false := data OCCURED AFTER 1990-03-08T01:01:01

9.7.11 Occur [not] At (二項演算子, 非結合):

Occur at 演算子の機能は **occur equal** 演算子と同じである。

<n:Boolean> := <n:any-type> OCCUR AT <n:time>

false := data OCCURRED AT 1990-03-01T00:00:00

9.7.2節参照。

9.8 文字列演算子

文字列演算子はデフォルトリスト処理やデフォルトプライマリ日時処理のあとには続かない。

9.8.1 || (二項演算子, 左結合)

|| 演算子（文字連結）はその被演算子を文字列に変換した上で，それらの文字列をを連結する。ヌルデータタイプは文字列 **null** に変換され，他の被演算子に付加される。したがって，|| は **null** を返すことはない。リストは文字列に変換され，他の被演算子に付加される；リストはカッコで囲まれ，その要素は空白ではなく“，”によって分離される。ブール型，数値，時刻，期間の表現は母国語の使用を考慮し，地域特有のものになっている。**formatted with** 演算子，**%z** 演算子は数値を文字列に変換するために使用される（9.8.2節参照）。**string** 演算子は，リストに対して特に何もしないことを除いては||演算子の一般形である（9.8.3節参照）。その被演算子のプライマリ日時は失われる。その用法は次の通りである：

<1:string> := <m:any-type> || <n:any-type>

"null3" := null || 3

"45" := 4 || 5

"4.7four" := 4.7 || "four"

"true" := true || ""

"3 days left" := 3 days || " left"

"on 1990-03-15T13:45:01" := "on " || 1990-03-15T13:45:01

"list=(1,2,3)" := "list=" || (1,2,3)

9.8.2 Formatted with (二項演算子, 左結合)

formatted with 演算子は, 書式文字列をデータ項目の出力方法の制御に使用できるようにする。書式文字列は, ANSI C 言語の printf 制御文に類似しており, Arden Syntax で使用されている時刻表現の書式化を可能にする。

```
<string> := <data> formatted with <format_string>

"01::02::03" := (1,2,3) formatted with "%2.2d::%2.2d::%2.2d"

"The result was 10.61 mg"
:= 10.60528 formatted with "The result was %.2f mg"

"The date was Jan 10 1998"
:= 1998-01-10T17:25:00 formatted with "The date was %.2t"

"The year was 1998"
:= 1998-01-10T17:25:00 formatted with "The year was %.0t"

/* longer example */
a := "ten";
b := "twenty";
c := "thirty";
f := "%s, %s, %s or more";
"ten, twenty, thirty or more" := (a, b, c) formatted with f;
```

Data は単一の項目の場合には, 書式文字列の代わりに単一のパラメータを提供する。**data** がリストの場合には, そのリストはリストとしては書式化されない。かわりに, 書式文字列の代わりのパラメータのリストであるととらえられる。演算の結果, 下記に示すように, パラメータは **format string** の代わりに用いられる。

書式文字列は文字列と1つ以上の書式情報からなる。

書式情報は必須フィールドとオプションフィールドからなり, 次のような様式である:

```
%[flags][width][.precision]type
```

書式情報の各フィールドは, 1文字または数値を表す特定の書式オプションである。最も単純な書式情報はパーセント記号と型文字だけを含む (例えば%s)。パーセント記号に書式フィールドとしては意味のない文字が続く場合, その文字は改訂されない。たとえば, パーセント記号を印字するためには%%を使用する。

C 言語の機能の互換性を保つためには, Arden MLM の作者には恐らく役に立たないさまざまな書式タイプ指定子を持たなければならないことに注意すること。MLM の作者が最も使いそうな書式情報は:

%c	(特殊文字の出力)
%s	(文字幅制御)
%d	(整数書式)
%t	(時刻書式)
%e	(指数付き浮動小数点書式)
%f	(指数なし浮動小数点書式)
%g	(%e または%f を使った浮動小数点書式)

書式情報としてサポートされる型の全容は付録A5で記述されている。

9.8.3 String ... (単項演算子, 右結合)

string 演算子は文字列か, その被演算子としての文字列のリストを期待している。それは||演算子のように, 全ての要素を連結して作成された単一の文字列を返す(9.8.1節参照)。もし, 被演算子が空リストであった場合, その結果は空文字("")である。その要素演算子(9.12.18節)はリストからある項目を選択するために使用することができる。その被演算子のプライマリ日時は失われる。その用法は次の通りである:

```
<1:string> := STRING <m:string>
<1:string> := STRING <m:list of strings>
"abc" := STRING ("a","b","c")
"abc" := STRING ("a","bc")
"" := STRING ()
"edcba" := STRING REVERSE EXTRACT CHARACTERS "abcde"
```

9.8.4 Matches pattern (二項演算子, 非結合)

この演算子は SQL(ISO/IEC9075)の LIKE 演算子に似ている。**Matches pattern** は特定の文字列がパターンにあっているか否かを決定するのに使用される。この演算子には2つの文字列の被演算子が必要である。1番目は突き合わせたい文字列, 2つ目は突き合わせに使用されるパターンである。**Matches Pattern** はブール値: 2番目の被演算子が1番目の被演算子に合えば真, 合わなければ偽を返す。また, 最初の被演算子が文字列のリストでもよく, その場合にはその結果は2番目の被演算子のパターンと各文字列の間での一致を示すブール値のリストになる。もし, 被演算子が文字列ではなければヌル値が返される。マッチングは大小文字によらない。被演算子のプライマリ日時は失われる。

2番目の被演算子のパターンは任意の legal string 文字でよい。さらに, 2つのワイルドカード文字が使用できる。アンダースコア(_)は任意の1文字とマッチングさせる。パーセント記号(%)は0から任意の数の文字とマッチングさせる。ワイルドカード文字の1つとマッチングさせる時にはそれにエスケープ文字(\)をつけて先行させる。

```
<n : Boolean> := <k : string> MATCHES PATTERN <m : string>
<n : list of Boolean> := <k : list of strings> MATCHES PATTERN <m : string>
true := "fatal heart attack" MATCHES PATTERN "%heart%";
false := "fatal heart attack" MATCHES PATTERN "heart";
true := "abnormal values" MATCHES PATTERN "%value_";
false := "fatal pneumonia" MATCHES PATTERN "%pulmonary%";
(true, false) := ("stunned myocardium", "myocardial infarction") MATCHES PATTERN
 "%myocardium";
true := "5%" MATCHES PATTERN "_\%";
```

9.8.5 Length (単項演算子, 右結合)

Length 演算子は文字列中の文字数を返す。前および後のスペースはこの勘定に含まれる。**Length** 演算子を空の文字列に適用するとゼロを返すが, 文字データ型でない場合や, 空のリストの場合にはヌル値が返される。**Length** 演算子は **count** 演算子(9.12.1節参照)とは異なり, **Length** は単一の文字列内の文字数であるが, **count** はリストの項目の数である。プライマリ日時は保持されない。

```
 <n:number> := LENGTH [OF] <n:string>
       7 := LENGTH OF "Example"
      14 := LENGTH "Example String"
       0 := LENGTH ""
    null := LENGTH ()
    null := LENGTH OF null
(8, 3, null) := LENGTH OF ("Negative", "Pos", 2)
```

9.8.6 Uppercase (単項演算子, 右結合)

uppercase 演算子は, 文字列内の全ての小文字を大文字に変換する。数字や句読点などの小文字でないものには影響しない。文字列でないデータタイプや空リストにおける **uppercase** は null である。プライマリ日時は保持される。

```
<n:string> := UPPERCASE <n:string>
"EXAMPLE STRING" := UPPERCASE "Example String"
"" := UPPERCASE ""
null := UPPERCASE null
null := UPPERCASE ()
("5-HIAA", "POS", null) := uppercase ("5-Hiaa", "Pos", 2)
```

9.8.7 Lowercase (単項演算子, 右結合)

lowercase 演算子は, 文字列内の全ての大文字を小文字に変換する。数字や句読点などの大文字でないものには影響しない。文字列でないデータタイプや空リストにおける **lowercase** は **null** である。プライマリ日時は保持される。

```
<n:string> := LOWERCASE <n:string>
"example string" := LOWERCASE "Example String"
"" := LOWERCASE ""
null := LOWERCASE 12.8
null := LOWERCASE null
("5-hiaa", "pos", null) := LOWERCASE ("5-HIAA", "Pos", 2)
```

9.8.8 Trim [Left | Right] (単項演算子, 右結合)

trim 演算子は文字列から前あるいは後スペースを取り除く（7.1.10節参照）。オプションの left または right 修飾子はそれぞれ前または後スペースを取り除くことができる。印刷可能文字と埋め込まれている空白文字は影響しない。文字列でないデータタイプや空リストの **trim** は **null** である。プライマリ日時は保持される。

```
<n:string> := TRIM [LEFT | RIGHT] <n:string>
"example" := TRIM "  example  "
"" := TRIM ""
null := TRIM ()
"result:  " := TRIM LEFT  "  result:  "
"  result:" := TRIM RIGHT "  result:  "
("5 N", "2 E", null) := TRIM (" 5 N", "2 E ", 2)
```

9.8.9 Find...[in] string...[starting at]... (三項演算子, 右結合)

find … string 演算子は対象の文字列内の部分文字列を探し, その部分文字列の開始位置を示す数値を返す。**find … string** は **matches pattern** に似ているが, （ブール型でない）数値を返し, ワイルドカードを認めない。**find … string** は場合に応じて, その対象文字列がその正確な部分文字列を含まない場合にはゼロを返す。もし, 部分文字列かまたは対象かが文字列データタイプでなければ **null** が返される。プライマリ日時は保持されない。

オプションの修飾子 **starting at …**は, 部分文字列の探索の開始場所を制御するために **find … string** 演算子に付加することができる。文字列の最初の文字から探索を開始する場合, その修飾子は省略される。**Starting at**

…に続く値は整数でなければならなく，そうでない場合は **null** が返される。もし，**Starting at …**に続く値が対象の文字列の長さを超える整数（例えば，1以下または対象の **length** 以上）の場合にはゼロが返される。

```
<n:number> := FIND <n:string> [IN] STRING <n:string>
<n:number> := FIND <n:string> [IN] STRING <n:string> [STARTING AT <n:number>]
     3 := FIND "a" IN STRING "Example Here"
     5 := FIND "ple" IN STRING "Example Here"
     0 := FIND "s" IN STRING "Example Here"
  null := FIND 2 IN STRING "Example Here"
  null := FIND "a" STRING 510
(2, 0, 4) := FIND "t" STRING ("start", "meds", "halt")
     7 := FIND "e" IN STRING "Example Here" STARTING AT 1
     1 := FIND "e" IN STRING LOWERCASE "Example Here" STARTING AT 1
    10 := FIND "e" IN STRING "Example Here" STARTING AT 8
    10 := FIND "e" IN STRING "Example Here" STARTING AT 10
    12 := FIND "e" IN STRING "Example Here" STARTING AT 11
     0 := FIND "e" IN STRING "Example Here" STARTING AT 13
  null := FIND "e" IN STRING "Example Here" STARTING AT 1.5
  null := FIND "e" IN STRING "Example Here" STARTING AT "x"
```

9.8.10 Substring … characters [starting at …] from … (三項演算子, 右結合)

substring … characters [starting at …] from …演算子は指定された対象の文字列から部分文字列を返す。この部分文字列は，もとになる文字列の開始位置（文字列の最初の文字か，文字列の中の指定した場所）から指定した数だけの文字からなる。例えば，**substring 3 characters starting at 2 from "Example"** は，"xam" ―もとの文字列"Example"内の第2文字から始まる3文字の文字列― を返す。

対象文字列は文字列データタイプでなければならず，その文字列内の開始位置は正の整数でなければならない，さらに，返される文字数が整数であるか，演算子が **null** を返さなければならない。もし，開始位置が指定されたときは，その値は1と文字列長の間の整数でなければならず，そうでなければ空の文字列が返される。もし，要求された文字数がその文字列長より大きい場合，全文字列が返される。もし，開始位置が指定され，その要求された文字数が文字列長から開始位置を差し引いた長さより大きい場合には，その結果の文字列は開始位置を含むもとの文字列の右側の部分である。もし，要求された文字数が正の場合，その文字は左から右に数える。もし，要求された文字数が負であるときはその文字は右から左に数えられる。部分文字列内の文字は常にそれらが文字列中に現れる順に返される。デフォルトのリスト操作に従う。プライマリ日時は保持される。

```
<n:string> := SUBSTRING <n:number> CHARACTERS [STARTING AT <n:number>]
                  FROM <n:string>
    "ab" := SUBSTRING 2 CHARACTERS FROM "abcdefg"
 "abcdefg" := SUBSTRING 100 CHARACTERS FROM "abcdefg"
   "def" := SUBSTRING 3 CHARACTERS STARTING AT 4 FROM "abcdefg"
  "defg" := SUBSTRING 20 CHARACTERS STARTING AT 4 FROM "abcdefg"
    null := SUBSTRING 2.3 CHARACTERS FROM "abcdefg"
    null := SUBSTRING 2 CHARACTERS STARTING AT 4.7 FROM "abcdefg"
    null := SUBSTRING 3 CHARACTERS STARTING AT "c" FROM "abcdefg"
    null := SUBSTRING "b" CHARACTERS STARTING AT 4 FROM "abcdefg"
    null := SUBSTRING 3 CHARACTERS STARTING AT 4 FROM 281471
     "d" := SUBSTRING 1 CHARACTERS STARTING AT 4 FROM "abcdefg"
     "d" := SUBSTRING –1 CHARACTERS STARTING AT 4 FROM "abcdefg"
   "bcd" := SUBSTRING –3 CHARACTERS STARTING AT 4 FROM "abcdefg"
     "a" := SUBSTRING 1 CHARACTERS FROM "abcdefg"
     "g" := SUBSTRING –1 CHARACTERS STARTING AT LENGTH OF "abcdefg"
```

```
                FROM "abcdefg"
("Pos","Neg",null):= SUBSTRING 3 CHARACTERS FROM ("Positive","Negative",2)
```

例：診察時の患者の収縮期および拡張期の血圧の測定値（bp）は"98/72", "121/86", "138/102"のように格納される。

```
Bp := "121/86";
slash_pos := FIND "/" IN STRING bp;
systolic := SUBSTRING (slash_pos – 1) CHARACTERS FROM  bp;
```

or

```
systolic  := SUBSTRING –3 CHARACTERS STARTING AT (slash_pos - 1) FROM bp;

diastolic := SUBSTRING 3 CHARACTERS STARTING AT (slash_pos + 1) FROM bp;
```

or

diastolic := SUBSTRING (LENGTH of bp) CHARACTERS FROM bp STARTING AT (slash_pos + 1)

9.9 数値演算子

日時および継続時間データタイプのふるまいは 8.5.2 節で説明する。

9.9.1 + (二項演算子, 左結合)

二項演算子+（加算）は左と右の被演算子の加算を行う。これで 2 つの継続時間や，継続時間で時刻を進めるような単純な加算を行うことができる。アンダーフローやオーバフローは **null** になる。その使用法は次の通りである:

```
<n:number> := <n:number> + <n:number>
    6 := 4 + 2
    () := 5 + ()
    null := (1,2,3) + ()
    () := null + ()
    null := 5 + null
    (null,null,null) := (1,2,3) + null
    null := null + null
<n:duration> := <n:duration> + <n:duration>
    3 days := 1 day + 2 days
<n:time> := <n:time> + <n:duration>
    1990-03-15T00:00:00 := 1990-03-13T00:00:00 + 2 days
    1993-05-17T00:00:00 := 0000-00-00 + 1993 years + 5 months + 17 days
<n:time> := <n:duration> + <n:time>
    1990-03-15T00:00:00 := 2 days + 1990-03-13T00:00:00
```

9.9.2 + (単項演算子, 非結合)

単項演算子＋はその被演算子が正しい型であれば影響はない。使い方は以下の通りである:

```
<n:number> := + <n:number>
    2 := + 2
    null := + "asdf"
<n:duration> := + <n:duration>
```


2 days := + 2 days

9.9.3 - (二項演算子, 左結合)

二項演算子 - (減算) は左から右の被演算子を減算する。これでは数値的な減算ができ, 2つの継続時間の減算, 継続時間によって時刻を遡らせること, あるいは2つの時刻の間の継続時間を知ることができる。アンダーフローやオーバフローは null になる。記載上, 減算演算子は時間定数 (7.1.5 節参照) における"-"と混同しないよう注意が必要である。いかなる曖昧さも時間定数に応じて結論づけられる。その用法は次の通りである:

<n:number> := <n:number> - <n:number>

4 := 6 - 2

<n:duration> := <n:duration> - <n:duration>

1 day := 3 days - 2 days

<n:time> := <n:time> - <n:duration>

1990-03-13T00:00:00 := 1990-03-15T00:00:00 - 2 days

<n:duration> := <n:time> - <n:time>

2 days := 1990-03-15T00:00:00 - 1990-03-13T00:00:00

9.9.4 - (単項演算子, 非結合)

単項演算子ーは算術的な取り消し (negation) のために使用される; これはいかにある負の定数を作るかということである。アンダーフローやオーバフローは **null** となる。2つの算術演算子を一緒に使うことはできない, そのため, **3 + -4** のような表現は誤りである。このかわりに, **3 + (-4)**, **3 – 4**, または **-4 + 3** のうちのどれかを使用しなければならない。その用法は次の通りである:

<n:number> := - <n:number>

(-2) := - 2

<n:duration> := - <n:duration>

(-2) days := - (2 days)

9.9.5 * (二項演算子, 左結合)

*演算子 (乗算) は左右の被演算子を乗算する。アンダーフローやオーバフローの場合は **null** となる。この演算子は数値または継続時間に対する数値の乗算ができる。その用法は次の通りである:

<n:number> := <n:number> * <n:number>

8 := 4 * 2

<n:duration> := <n:number> * <n:duration>

6 days := 3 * 2 days

<n:duration> := <n:duration> * <n:number>

6 days := 2 days * 3

9.9.6 / (二項演算子, 左結合)

/演算子 (除算) は左の被演算子を右のもので除算する。この演算子は数値の除算, 継続時間の数値による除算, あるいは2つの継続時間の比を計算できる。ゼロによる除算, アンダーフロー, オーバフローの場合は **null** となる。/演算子で継続時間の単位変換を行うことができる (例えば, **/1 year** はどのような継続時間でも年に換算できる)。その用法は次の通りである:

<n:number> := <n:number> / <n:number>

4 := 8 / 2

<n:duration> := <n:duration> / <n:number>

2 days := 6 days / 3

<n:number> := <n:duration> / <n:duration>

120 := 2 minutes / 1 second

36 := 3 years / 1 month

9.9.7 ** (二項演算子, 非結合)

**演算子（指数化）は左の被演算子に対して右の被演算子を指数として累乗する。その用法は次の通りである：

<n:number> := <n:number> ** <n:number>

9 := 3 ** 2

9.10 時間演算子

日時および継続時間のふるまいは 8.5.2 節で説明する。

9.10.1 After (二項演算子, 非結合)

After 演算子はある日時に継続時間を加えるのと同等である。その用法は次の通りである：

<n:time> := <n:duration> AFTER <n:time>

1990-03-15T00:00:00 := 2 days AFTER 1990-03-13T00:00:00

9.10.2 Before (二項演算子, 非結合)

Before 演算子は日時から継続時間を減算するのと同等である。その用法は次の通りである：

<n:time> := <n:duration> BEFORE <n:time>

1990-03-11T00:00:00 := 2 days BEFORE 1990-03-13T00:00:00

9.10.3 Ago (単項演算子, 非結合)

ago 演算子は **now** から継続時間を減算し，日時とする。その用法は次の通りである（**now** が 1990-04-19T00:03:15 とすると）：

<n:time> := <n:duration> AGO

1990-04-17T00:03:15 := 2 days AGO

9.10.4 From (二項演算子, 非結合)

from 演算子は日時に継続時間を加算するのと同等である。その用法は次の通りである：

<n:time> := <n:duration> FROM <n:time>

2000-09-13T00:08:00 := 2 days FROM 2000-09-11T00:08:00

9.11 継続時間演算子

継続時間データタイプのふるまいは 8.5.2 節で説明する。時間演算子が継続時間演算子よりも優先度が低いので，**3 hours before 3 days ago** は**(3 hours) before ((3 days) ago)**を意味し，現在時刻の3日と3時間前を返す。

9.11.1 Year (単項演算子, 非結合)

year 演算子には，**years** という同義語がある。これは，その数値から月継続時間（1年は12ヶ月）を生成する。その用法は次の通りである：

<n:duration> := <n:number> YEAR

24 months := 2 YEAR

9.11.2 Extract year (単項演算子, 右結合)

extract year 演算子は日時を示すデータから「年」を取り出す。その用法は次の通りである:

<n:number> := EXTRACT YEAR <n:time>

1990 := EXTRACT YEAR 1990-01-03T14:23:17.3

null := EXTRACT YEAR (1 YEAR)

9.11.3 Month (単項演算子, 非結合)

month 演算子には, **months** という同義語がある。これは, 数値から月継続時間を生成する。その用法は次の通りである:

<n:duration> := <n:number> MONTH

9.11.4 Extract month (単項演算子, 右結合)

extract month 演算子は日時を示すデータから「月」を取り出す。その用法は次の通りである:

<n:number> := EXTRACT MONTH <n:time>

1 := EXTRACT MONTH 1990-01-03T14:23:17.3

null := EXTRACT MONTH 1

9.11.5 Week (単項演算子, 非結合)

week 演算子には, **weeks** という同義語がある。これは数値から秒継続時間を生成する: 1週は 604800 秒である。その用法は次の通りである:

<n:duration> := <n:number> WEEK

9.11.6 Day (単項演算子, 非結合)

day 演算子には, **days** という同義語がある。これは数値から秒継続時間を生成する: 1日は 86400 秒である。その用法は次の通りである:

<n:duration> := <n:number> DAY

9.11.7 Extract day (単項演算子, 右結合)

extract day 演算子は日時を示すデータから「日」を取り出す。その用法は次の通りである:

<n:number> := EXTRACT DAY <n:time>

3 := EXTRACT DAY 1990-01-03T14:23:17.3

null := EXTRACT DAY "this is not a time"

9.11.8 Hour (単項演算子, 非結合)

hour 演算子には, **hours** という同義語がある。これは数値から秒継続時間を生成する: 1時間は 3600 秒である。その用法は次の通りである:

<n:duration> := <n:number> HOUR

9.11.9 Extract hour (単項演算子, 右結合)

extract hour 演算子は日時を示すデータから［時］を取り出す。その用法は次の通りである:

<n:number> := EXTRACT HOUR <n:time>

14 := EXTRACT HOUR 1990-01-03T14:23:17.3

null := EXTRACT HOUR (1 HOUR)

9.11.10 Minute (単項演算子, 非結合)

minute 演算子には, **minutes** という同義語がある。これは数値から秒継続時間を生成する。1分は 60 seconds である。その用法は次の通りである:

<n:duration> := <n:number> MINUTE

9.11.11 Extract minute (単項演算子, 右結合)

extract minute 演算子は日時を示すデータから「分」を取り出す。その用法は次の通りである:

<n:number> := EXTRACT MINUTE <n:time>

23 := EXTRACT MINUTE 1990-01-03T14:23:17.3

0 := EXTRACT MINUTE 1990-01-03

null := EXTRACT MINUTE 0000-00-00

9.11.12 Second (単項演算子, 非結合)

second 演算子には, **seconds** という同義語がある。これは数値から秒継続時間を生成する。その用法は次の通りである:

<n:duration> := <n:number> SECOND

9.11.13 Extract second (単項演算子, 右結合)

extract second 演算子は日時を示すデータから「秒」を取り出す。その用法は次の通りである:

<n:number> := EXTRACT SECOND <n:time>

17.3 := EXTRACT SECOND 1990-01-03T14:23:17.3

null := EXTRACT SECOND (1 second)

9.12 集合体演算子

9.12.1 概要

集合体演算子はデフォルトのリスト処理やデフォルトのプライマリ日時処理は行わない。集合体演算子はリストに対して集合体演算を行う。すなわち、被演算子 (すべて単項) としてリストを取り、結果として1つの項目を返す。特に指定のない限り、リストのすべての要素が同じプライマリ日時を持つ場合は結果も同じプライマリ日時を保持する (そうではない場合は、プライマリ日時は失われる)。被演算子が単一項目のときは、長さ1のリストとして扱われる。

個々の演算子は **of** を伴っても良い。括弧は必ずしも必要ではない。例えば、以下は同じものとして扱われる:

SUM a_list

SUM OF a_list

SUM(a_list)

SUM OF(a_list)

複数の集合体演算子と変換演算子 (例えば 9.14 節を参照) を式の中で括弧なしで並べることが出来る。例えば以下の通りである。:

AVERAGE OF LAST 3 FROM a_list

9.12.2 Count (単項演算子, 右結合)

count 演算子はリスト中の項目の数を返す（項目がない場合を含む）。**count** 演算子は **null** を返すことはない。**count** 演算子の処理の結果、プライマリ日時は保持されない。使い方は次の通りである。：

```
<1:number> := COUNT <n:any-type>
    4 := COUNT (12,13,14,null)
    1 := COUNT "asdf"
    0 := COUNT ()
    1 := COUNT null
```

9.12.3 Exist (単項演算子、右結合性)

exist 演算子には1つの同義語 **exists** がある。ひとつ以上の null ではない任意の型の項目がリストの中に存在すれば、**true** を返す。**exist** は **null** を返すことはない。リストのすべての要素が同じプライマリ日時を持つ場合、結果はそのプライマリ日時を保持する（そうでない場合、プライマリ日時は失われる）。使い方は次の通りである。：

```
<1:Boolean> := EXIST <n:any-type>
    true := EXIST (12,13,14)
    false := EXIST null
    false := EXIST ()
    true := EXIST ("plugh",null)
```

9.12.4 Average (単項演算子、右結合性)

average 演算子には1つの同義語 **avg** がある。数、時刻、継続時間などのリストから平均値を算出する。リストのすべての要素が同じプライマリ日時を持つ場合、結果はそのプライマリ日時を保持する（そうでない場合、プライマリ日時は失われる）。使い方は次の通りである。：

```
<1:number> := AVERAGE <n:number>
    14 := AVERAGE (12,13,17)
    3 := AVERAGE 3
null := AVERAGE ()
    <1:time> := AVERAGE <n:time>
1990-03-11T03:10:00 := AVERAGE (1990-03-10T03:10:00, 1990-03-12T03:10:00)
    <1:duration> := AVERAGE <n:duration>
    3 days := AVERAGE (2 days, 3 days, 4 days)
```

9.12.5 Median (単項演算子、右結合性)

median 演算子は数、時刻、または継続時間のリストから中央値を計算する。リストは最初にソートされる。もし、項目の数が奇数であった場合、中央の値が選択される。もし、項目の数が偶数であった場合、中央の2つの値の平均値が計算される。複数の要素の値が同じであれば、最新のプライマリ日時を持つ要素が採用される。1つの要素の値が選ばれるか、2つの要素が同じプライマリ日時を持つ場合は、結果はそのプライマリ日時を保持する（そうでない場合、プライマリ日時は失われる）。使い方は次の通りである。：

```
<1:number> := MEDIAN <n:number>
    13 := MEDIAN (12,17,13)
    3 := MEDIAN 3
    null := MEDIAN ()
<1:time> := MEDIAN <n:time>
    1990-03-11T03:10:00 = MEDIAN (1990-03-10T03:10:00, 1990-03-11T03:10:00, 1990-03-28T03:10:00)
<1:duration> := MEDIAN <n:duration>
    3 days := MEDIAN (1 hour, 3 days, 4 years)
```

9.12.6 Sum (単項演算子、右結合性)

sum 演算子は、数または持続時間のリストの総和を計算する。リストのすべての要素が同じプライマリ日時を持つ場合、結果はそのプライマリ日時を保持する（そうでない場合、プライマリ日時は失われる）。使い方は次の通りである。：

```
<1:number> := SUM <n:number>
    39 := SUM (12,13,14)
    3 := SUM 3
    0 := SUM ()
<1:duration> := SUM <n:duration>
    7 days := SUM (1 day, 6 days)
```

9.12.7 stddev (単項演算子、右結合性)

stddev 演算子は数値のリストの標本標準偏差値を返す。リストのすべての要素が同じプライマリ日時を持つ場合、結果はそのプライマリ日時を保持する（そうでない場合、プライマリ日時は失われる）。使い方は次の通りである。：

```
<1:number> := STDDEV <n:number>
    1.58113883 := STDDEV (12,13,14,15,16)
    null := STDDEV 3
    null := STDDEV ()
```

9.12.8 Variance (単項演算子、右結合性)

variance 演算子は数値のリストの標本分散値を返す。リストのすべての要素が同じプライマリ日時を持つ場合、結果はそのプライマリ日時を保持する（そうでない場合、プライマリ日時は失われる）。使い方は次の通りである。：

```
<1:number> := VARIANCE <n:number>
    2.5 := VARIANCE (12,13,14,15,16)
    null := VARIANCE 3
    null := VARIANCE ()
```

9.12.9 Minimum (単項演算子、右結合性)

minimum 演算子には1つの同義語 min がある。<=演算子（9.5.4節参照）を用い、順序型のデータからなる均質なリスト（すなわち、すべて数値、すべて時刻、すべて継続時間、もしくはすべて文字列）から一番小さな値を返す。同じ値の要素が複数あった場合、最新のプライマリ日時を持つ要素を選択する。選択された被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<1:ordered> := MINIMUM <n:ordered>
    12 := MINIMUM (12,13,14)
    3 := MIN 3
    null := MINIMUM ()
    null := MINIMUM (1,"abc")
```

9.12.10 Maximum (単項演算子、右結合性)

maximum 演算子には1つの同義語 max がある。>=演算子（9.5.6節参照）を用い、順序型のデータからなる均質なリストから最も大きな値を返す。同じ値の要素が複数あった場合、最新のプライマリ日時を持つ要素を選択する。選択された被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<1:ordered> := MAXIMUM <n:ordered>
    14 := MAXIMUM (12,13,14)
    3 := MAXIMUM 3
    null := MAXIMUM ()
    null := MAXIMUM (1,"abc")
```

9.12.11 Last (単項演算子、右結合性)

last 演算子はデータの型に拘わらず、リストの最後の値を返す。リストが空の場合は、**null** を返す。式 **last x** は **x[count x]**と等価である。時刻でソートさた照会結果に **last** を使用すると、最新の値を返す。選ばれた被演算子のプライマリ日時は保持される。この **last** は、Arden Syntax version E 1460-92 で定義された **last** とは異なるので注意が必要である。後者は現在では **latest**（9.12.16節参照）と呼ばれている。使い方は次の通りである。:

```
<1:any-type> := LAST <n:any-type>
    14 := LAST (12,13,14)
    3 := LAST 3
    null := LAST ()
```

9.12.12 First (単項演算子、右結合性)

first 演算子はリストの最初の値を返す。リストが空の場合は、**null** を返す。**first x** という式は **x[1]**と等価である。時刻でソートされた照会結果に **first** を使用すると、最も早い値を返す。選ばれた被演算子のプライマリ日時は保持される。この **first** は、Arden Syntax version E 1460-92 で定義された **first** とは異なるので注意が必要である。後者は現在では **earliest**（9.12.17節参照）と呼ばれている。使い方は次の通りである。:

```
<1:any-type> := FIRST <n:any-type>
    12 := FIRST (12,13,14)
    3 := FIRST 3
    null := FIRST ()
```

9.12.13 Any (単項演算子、右結合性)

any 演算子は、リストの中に **true** の項目が存在すれば **true** を返す。もし、それらがすべて **false** である場合は、**false** を返す。どちらでもない場合は **null** を返す。特別な場合として、リストの要素が無い場合は、**false** を返す。リストのすべての要素が同じプライマリ日時を持つ場合は、結果はそのプライマリ日時を保持する（さもなければプライマリ日時は失われる）。使い方は次の通りである。:

```
<1:Boolean> := ANY <n:any-type>
    true := ANY (true,false,false)
    false := ANY false
    false := ANY ()
    null := ANY (3, 5, "red")
    false := ANY (false, false)
    null := ANY (false, null)
```

9.12.14 All (単項演算子、右結合性)

all 演算子はリストのすべての項目が **true** の時、**true** を返す。もし、ひとつでも **false** を持つ項目があれば **false** を返す。どちらでもない場合は **null** を返す。特別な場合として、リストの要素が無い場合は、**true** を返す。すべての要素が同じプライマリ日時を持つ場合は、結果はそのプライマリ日時を保持する（さもなければプライマリ日時は失われる）。使い方は次の通りである。:

```
<1:Boolean> := ALL <n:any-type>
    false := ALL (true,false,false)
```

```
false := ALL false
true := ALL ()
null := ALL (3, 5, "red")
null := ALL (true, null)
```

9.12.15 No (単項演算子、右結合性)

no 演算子はﾘｽﾄのすべての項目が **false** の場合に **true** を返す。もし、ひとつでも **true** を持つ項目があれば **false** を返す。どちらでもない場合は **null** を返す。特別な場合として、ﾘｽﾄの要素が無い場合は、**true** を返す。すべての要素が同じプライマリ日時を持つ場合は、結果はそのプライマリ日時を保持する（さもなければプライマリ日時は失われる）。使い方は次の通りである。：

```
<1:Boolean> := NO <n:any-type>
    false := NO (true,false,false)
    true := NO false
    true := NO ()
    null := NO (3, 5, "red")
    null := NO (false, null)
```

9.12.16 Latest (単項演算子、右結合性)

latest 演算子はﾘｽﾄに含まれる項目の内、最新のプライマリ日時を持つ値を返す。もし、ﾘｽﾄの何れの要素もプライマリ日時を持たない場合は **null** を返す。（これが望む処理でないとき、**where time of it is present** によって（[訳注] つまり被演算子の各項目がプライマリ日時を持つかどうか）いつでも被演算子をふるいにかけることが出来る）。ﾘｽﾄが空の場合は **null** が返される。選ばれた被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<1:any-type> := LATEST <n:any-type>
null := LATEST ()
```

"penicillin" := LATEST ("penicillin", "ibuprofen", "pseudoephedrine HCL");

(T16:40)　　(T16:40)　(T14:05)　(T14:04)

9.12.17 Earliest (単項演算子、右結合性)

earliest 演算子はﾘｽﾄに含まれる項目の内、最も早いプライマリ日時を持つ値を返す。ﾘｽﾄの何れの要素もプライマリ日時を持たない場合は **null** を返す。（これが望む処理でないとき、**where time of it is present** によっていつでも被演算子をふるいにかけることが出来る）。ﾘｽﾄが空の場合は **null** が返される。選ばれた被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<1:any-type> := EARLIEST <n:any-type>
null := EARLIEST ()
```

" pseudoephedrine HCL" := EARLIEST ("penicillin", "ibuprofen", " pseudoephedrine HCL");

(T14:04)　　(T16:40)　(T14:05)　(T14:04)

9.12.18 Element (二項演算子)

element(**[ ]**)演算子は、ﾘｽﾄから一つもしくは複数の要素を、1番から始まる要素の位置で示される番号により抜き出すことに利用される。“element” の被演算子はﾘｽﾄ表現（**[ ]**の左側）と整数のﾘｽﾄ（**[ ]**の中）である。選ばれた被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<n:any-type> := <k:any-type>[n:index]
20 := (10,20,30,40)[2]
```

() := (10,20)[()]

(null,20) := (10,20)[1.5,2]

(10,30,50) := (10,20,30,40,50)[1,3,5]

(10,30,50) := (10,20,30,40,50)[1,(3,5)]

(10,20,30) := (10,20,30,40,50)[1 seqto 3]

9.12.19 Extract characters ... (単項演算子、右結合性)

extract characters 演算子は被演算子として文字列を取る。その文字列を分解して単一文字のリストとして返す。被演算子の要素が一つ以上ある場合、||演算子 (9.8.1節参照) のように最初にそれらの要素を連結する。被演算子が空リストの場合、結果は空リスト**()**である。**string** 演算子 (9.8.3節参照) を使うと、リストを文字列として再構成することが出来る。また、element 演算子 (9.12.18節参照) によりリストから任意の項目を選ぶことができる。この被演算子におけるプライマリ日時は失われる。使い方は次の通りである。:

<n:string> := EXTRACT CHARACTERS <m:any-type>

("a","b","c") := EXTRACT CHARACTERS "abc"

("a","b","c") := EXTRACT CHARACTERS ("ab","c")

() := EXTRACT CHARACTERS ()

() := EXTRACT CHARACTERS ""

"edcba" := STRING REVERSE EXTRACT CHARACTERS "abcde"

9.12.20 Seqto (二項演算子、非結合性)

seqto 演算子は昇順に並んだ整数リストを生成する。2つの被演算子はともに単一の整数でなければならない。そうでない場合は null が返される。最初の被演算子が2番目の被演算子より大きい場合、戻り値は空リストとなる。この被演算子におけるプライマリ日時は失われる。使い方は次の通りである。:

<n:number> := <1:number> SEQTO <1:number>

(2,3,4) := 2 SEQTO 4

() := 4 SEQTO 2

null := 4.5 SEQTO 2

(2) := 2 SEQTO 2

(-3,-2,-1) := -3 SEQTO -1

(2,4,6,8) := 2 * (1 SEQTO 4)

null := (1.5 seqto 5)

9.12.21 Reverse (単項演算子、右結合性)

reverse 演算子は逆の順序に各要素を並べ替えた新しいリストを生成する。被演算子のプライマリ日時は保持される。使い方は次の通りである。:

<n:any-type> := REVERSE <n:any-type>

(3,2,1) := reverse (1,2,3)

(6,5,4,3,2,1) := reverse (1 seqto 6)

() := reverse ()

9.12.22 インデックス抽出集合体演算子

これらの演算子はインデックス抽出機能を持たない同種の演算子と似た働きをするが、要素の値を返すのではなく、指定した条件に一致した要素のインデックスを値として返すところが異なる。これらの演算子はプライマリ日時を保持しない。

9.12.22.1 Index Latest (単項演算子、右結合性)

index latest 演算子はﾘｽﾄ中で最新のﾌﾟﾗｲﾏﾘ日時を持つ要素のｲﾝﾃﾞｯｸｽを返す。どの要素もﾌﾟﾗｲﾏﾘ日時を持たない場合、null を返す（これが望む処理でないとき、**where time of it is present** によっていつでも被演算子をふるいにかけることが出来る）。ﾘｽﾄが空の場合は **null** が返される。選ばれた被演算子のﾌﾟﾗｲﾏﾘ日時は保持される。使い方は次の通りである。：

```
<1:any-type> := INDEX LATEST <n:any-type>
    null := INDEX LATEST ()

1 := INDEX LATEST ("penicillin", "ibuprofen", "psuedophedrine HCL");
            (T16:40)    (T14:05)      (T14:04)
```

9.12.22.2 Index Earliest (単項演算子、右結合性)

index earliest 演算子はﾘｽﾄ中で最も早いﾌﾟﾗｲﾏﾘ日時を持つ要素のｲﾝﾃﾞｯｸｽを返す。どの要素もﾌﾟﾗｲﾏﾘ日時を持たない場合、null を返す（これが望む処理でないとき、**where time of it is present** によっていつでも被演算子をふるいにかけることが出来る）。ﾘｽﾄが空の場合は **null** が返される。被演算子のﾌﾟﾗｲﾏﾘ日時は保持される。使い方は次の通りである。：

```
<1:any-type> := INDEX EARLIEST <n:any-type>
    null := INDEX EARLIEST ()

3 := INDEX EARLIEST ("penicillin", "ibuprofen", "psuedophedrine HCL");
            (T16:40)    (T14:05)      (T14:04)
```

9.12.22.3 Index Minimum (単項演算子、右結合性)

index minimum 演算子には1つの同義語 **index min**.がある。<=演算子（9.5.4節参照）を用い、順序型のデータからなる均質なﾘｽﾄ（例えば、すべて数値、すべて時刻、すべて継続時間、もしくはすべて文字列）から最も小さい値を持つ要素のｲﾝﾃﾞｯｸｽを返す。同じ値が複数ある場合は、最新のﾌﾟﾗｲﾏﾘ日時の要素が選ばれる。使い方は次の通りである。：

```
<1:ordered> := INDEX MINIMUM <n:ordered>
    1 := INDEX MINIMUM (12,13,14)
    1 := INDEX MIN 3
    null := INDEX MINIMUM ()
    null := INDEX MINIMUM (1,"abc")
```

9.12.22.4 Index Maximum (単項演算子、右結合性)

index maximum 演算子にはひとつの同義語 **index max** がある。>=演算子（9.5.6節参照）を用い、順序型のデータからなる均質なﾘｽﾄから、最も大きな値を持つ要素のｲﾝﾃﾞｯｸｽを返す。同じ値が複数ある時には、最新のﾌﾟﾗｲﾏﾘ日時が採用される。選択された被演算子のﾌﾟﾗｲﾏﾘ日時は保持される。使い方は次の通りである。：

```
<1:ordered> := INDEX MAXIMUM <n:ordered>
    3 := INDEX MAXIMUM (12,13,14)
    1 := INDEX MAX 3
    null := INDEX MAXIMUM ()
    null := INDEX MAXIMUM (1,"abc")
```

9.12.22.5

last や **first** と同等のインデックス抽出集合体演算子はない。すなわち、INDEX FIRST という演算子がもしあるとすればそれは必ず1を返すし、また INDEX LAST という演算子は COUNT 演算子と同じ値を返すので、あっても意味がない。

9.13 集合体照会演算子

9.13.1 概要

集合体照会演算子はデフォルトのリスト処理やプライマリ日時処理は行わない。これらはリストに対して集合体演算を行う。すなわち、それらはリストを一つの被演算子として取り扱い、結果として単一の項目を返す。被演算子が単一の項目の場合は、長さ1のリストとして扱われる。特に指定のない限り、リストのすべての要素が同じプライマリ日時を持つ場合は結果も同じプライマリ日時を保つ（そうではない場合は、プライマリ日時は失われる）。

The unary query aggregation operators (that is, those that do not include the **from** word) may optionally be followed by **of**.

単項の集合体照会演算子（すなわち、語 **from** を含まないもの）には、オプションとして **of** が続くことがある。

9.13.2 Nearest ... From (二項演算子、右結合性)

nearest ... from 演算子は一番目の被演算子として時刻を、二番目の被演算子としてリストを取る。そして指定した時刻に最も近い時刻に発生した項目をリストから選択する。もし、リストのすべての要素がプライマリ日時を持たない場合は **null** を返す（これが望む処理でないとき、**where time of it is present** によっていつでも被演算子をふるいにかけることが出来る）。時刻が等しい項目が複数ある場合、最小のインデックスを持つ要素が使われる。被演算子のプライマリ日時は保持される。さて **data** を値 **12, 13, 14** による照会の結果のリストとする。また **data** はそれぞれ次のプライマリ日時、1990-03-15T15:00:00, 1990-03-16T15:00:00, 1990-03-17T15:00:00 を持つとする。そして現在を 1990-03-18T16:00:00.とする。このとき **nearest ... from** 演算子の使い方は次の通りである。：

```
<n:any-type> := NEAREST <n:time> FROM <m:any-type>
    13 := NEAREST (2 days ago) FROM data
    null := NEAREST (2 days ago) FROM (3,4)
    null := NEAREST (2 days ago) FROM ()
```

9.13.3 Index Nearest ... From (二項演算子、右結合性)

index nearest ... from 演算子の機能は **nearest … from operator** 演算子（9.13.2節参照）と極めて酷似しているが、戻り値として要素そのものを返すのではなく、要素のインデックスを返す点が異なる。**Index nearest … from** 演算子はプライマリ日時を保持しない。さて **data** を値 **12, 13, 14** による照会の結果のリストとする。また **data** はそれぞれ次のプライマリ日時、1990-03-15T15:00:00, 1990-03-16T15:00:00, 1990-03-17T15:00:00 を持つとする。そして現在を 1990-03-18T16:00:00.とする。このとき **Index nearest ... from** 演算子の使い方は次の通りである。：

```
<n:number> := INDEX NEAREST <n:time> FROM <m:any-type>
    2 := INDEX NEAREST (2 days ago) FROM data
    null := INDEX NEAREST (2 days ago) FROM (3,4)
```

9.13.4 Slope (単項演算子、右結合性)

slope 演算子は回帰分析を行う。そしてy軸を値、x軸を時間とした時の照会の結果として勾配を返す。結果は一日当たりの量で表されるが、単なる数値である。被演算子の項目数が2つ未満のときは、**null** を返す。リストのすべての要素が同じプライマリ日時を持つ場合、結果は **null** である。1つ以上の要素がプライマリ日時を持たない場合、結果は **null** となる。**slope** 演算子の結果はプライマリ日時を持たない。使い方は次の通りである。（**data** の値は前述のものと同じとする）：

```
<1:number> := SLOPE <n:number>
    1 := SLOPE data
    null := SLOPE (3,4)
```

9.14 変換演算子

9.14.1 概要

変換演算子はデフォルトのリスト処理やプライマリ日時処理は行わない。これらはリストを変換して、別のリストを作る。もし、被演算子が単一項目の場合、長さ1のリストとして扱う。たとえ要素が一つしかなくても、結果は常にリストである（但し、エラーが発生した場合、結果は **null** となる）。

演算子が単項演算子（つまり、語 **from** を含まない場合）の場合、オプションとして **of** が続くことがある。

9.14.2 Minimum ... From (二項演算子、右結合性)

minimum ... from 演算子には一つの同義語 **min ... from** がある。一番目の被演算子として数字（以下 N とする）をとる。また二番目の被演算子として順序型のデータからなる均質なリストをとる。この演算子は被演算子のリストから小さいほうの N 個の項目を抽出し、それをリストとして返す。結果のリストの項目の並び順序は、第二被演算子中の項目順序と同じである。また、すべてのデータは複製される。N が非負の整数でないときは **null** が返される。被演算子のリストで項目数が足りない場合は、可能な限りの項目を返す。同じ値の項目が複数ある場合は、もっとも新しいプライマリ日時の項目が選択される。被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<n:ordered> := MINIMUM <1:number> FROM <m:ordered>
    (11,12) := MINIMUM 2 FROM (11,14,13,12)
    (,3) := MINIMUM 2 FROM 3
    null := MINIMUM 2 FROM (3, "asdf")
    () := MINIMUM 2 FROM ()
    () := MINIMUM 0 FROM (2,3)
    (1,2,2) := MINIMUM 3 FROM (3,5,1,2,4,2)
```

9.14.3 Maximum ... From (二項演算子、右結合性)

maximum ... from 演算子にはひとつの同義語 **max ... from** がある。一番目の被演算子として数字（以下 N とする）をとる。また二番目の被演算子として順序型のデータからなる均質なリストをとる。この演算子は被演算子のリストから大きいほうの N 個の項目を抽出し、それをリストとして返す。結果のリストの項目の並び順序は、第二被演算子中の項目順序と同じである。また、すべてのデータは複製される。N が非負の整数でないときは **null** が返される。被演算子のリストで項目数が足りない場合は、可能な限りの項目を返す。同じ値の項目が複数ある場合は、もっとも新しいプライマリ日時の項目が選択される。被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<n:ordered> := MAXIMUM <1:number> FROM <m:ordered>
    (14,13) := MAXIMUM 2 FROM (11,14,13,12)
    (,3) := MAXIMUM 2 FROM 3
    null := MAXIMUM 2 FROM (3, "asdf")
    () := MAXIMUM 2 FROM ()
    () := MAXIMUM 0 FROM (1,2,3)
    (5,4,4) := MAXIMUM 3 FROM (1,5,2,4,1,4)
```


9.14.4 First ... From (二項演算子、右結合性)

first ... from 演算子は一番目の被演算子として数字（以下 N とする）を、また二番目の被演算子としてリストをとる。この演算子は被演算子のリストの先頭 N 個の項目からなるリストを返す。N が非負の整数でない場合 **null** が返される。リストが時刻でソートされている場合は、戻り値は早い時刻順となる。被演算子のリストで項目数が足りない場合は、可能な限りの項目を返す。これは、x が空の場合、**first 1 from x** と **first x** は異なることを意味する。つまり前者は**()**を返し、後者は **null 値**を返す。被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<n:any-type> := FIRST <1:number> FROM <m:any-type>
    (11,14) := FIRST 2 FROM (11,14,13,12)
    (,3) := FIRST 2 FROM 3
    (null,1) := FIRST 2 FROM (null,1,2,null)
    () := FIRST 2 FROM ()
```

9.14.5 Last ... From (二項演算子、右結合性)

last ... from 演算子は一番目の被演算子として数字（以下 N とする）を、また二番目の被演算子としてリストをとる。この演算子は被演算子のリストで後ろの N 個の項目からなるリストを返す。N が非負の整数でない場合 **null** が返される。リストが時刻でソートされている場合は、戻り値は新しい時刻順となる。被演算子のリストの項目数が足りない場合は、可能な限りの項目を返す。これは、x が空の場合、**last 1 from x** と **last x** は異なることを意味する。つまり前者は**()**を返し、後者は **null** を返す。被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<n:any-type> := LAST <1:number> FROM <m:any-type>
    (13,12) := LAST 2 FROM (11,14,13,12)
    (,3) := LAST 2 FROM 3
    (2,null) := LAST 2 FROM (null,1,2,null)
    () := LAST 2 FROM ()
```

9.14.6 Increase (単項演算子、右結合性)

increase 演算子は、数字、日時もしくは経過時間の均質なリスト中の隣り合う二項目の差分からなるリストを返す。結果のリストの項目数は、被演算子の項目数よりひとつ少なくなる。被演算子が空集合のときは、**null** が返される。各々の隣り合う二項目のうち、2 番目の項目のプライマリ日時が保持される。使い方は次の通りである。：

```
<n:number> := INCREASE <m:number>
    (4,-2,-1) := INCREASE (11,15,13,12)
    () := INCREASE 3
    null := INCREASE ()
<n: duration> := INCREASE <m:time>
    (1 day) := INCREASE (1990-03-01,1990-03-02)
<n:duration> := INCREASE <m:duration>
    (1 day) := INCREASE (1 day, 2 days)
```

9.14.7 Decrease (単項演算子、右結合性)

decrease 演算子は、同種の数字、日時もしくは経過時間の均質なリスト中のリストの隣り合う二項目の逆方向差分からなるリストを返す。結果のリストの項目数は、被演算子の項目数よりひとつ少なくなる。被演算子が空集合のときは、**null** が返される。**decrease** は **increase** の追加的な逆関数である。各々の隣り合う二項目のうち、2 番目の項目のプライマリ日時が保持される。使い方は次の通りである。：

```
<n:number> := DECREASE <m:number>
    (-4,2,1) := DECREASE (11,15,13,12)
```

() := DECREASE 3

null := DECREASE ()

<n: duration> := DECREASE <m:time>

((-1) day) := DECREASE (1990-03-01,1990-03-02)

<n:duration> := DECREASE <m:duration>

((-1) day) := DECREASE (1 day, 2 days)

9.14.8 % Increase (単項演算子、右結合性)

% increase 演算子にはひとつの同義語 **percent increase** がある。この演算子は、数字または経過時間の均質なﾘｽﾄ中の隣り合う二項目の%増分からなるﾘｽﾄを返す（分母は各組の最初の項目である；もし、それがゼロであれば **null** を返す）。各々の隣り合う二項目のうち、2番目の項目のプライマリ日時が保持される。使い方は次の通りである。：

<n:number> := % INCREASE <m:number>

(36.3636,-13.3333) := % INCREASE (11,15,13)

() := % INCREASE 3

null := % INCREASE ()

<n:number> := % INCREASE <m:duration>

(100) := % INCREASE (1 day, 2 days)

9.14.9 % Decrease (単項演算子、右結合性)

% decrease 演算子にはひとつの同義語 **percent decrease** がある。この演算子は、数字または経過時間の均質なﾘｽﾄ中の隣り合う二項目の%減少量からなるﾘｽﾄを返す（分母は各組の最初の項目である；もし、それがゼロであれば **null** を返す）。各々の隣り合う二項目のうち、2番目の項目のプライマリ日時が保持される。使い方は次の通りである。：

<n:number> := % DECREASE <m:number>

(-36.3636,13.3333) := % DECREASE (11,15,13)

() := % DECREASE 3

null := % DECREASE ()

<n:number> := % DECREASE <m:duration>

(-100) := % DECREASE (1 day, 2 days)

9.14.10 Earliest ... From (二項演算子、右結合性)

earliest ... from 演算子は一番目の被演算子として数字（以下Nとする）を、二番目の被演算子としてﾘｽﾄをとる。この演算子は被演算子のﾘｽﾄから早い順にN個の項目を抽出し、それをﾘｽﾄとして返す。結果のﾘｽﾄの項目の並び順序は、被演算子のﾘｽﾄの項目順序と同じである。Nが非負の整数でないときは **null** を返す。どの要素もプライマリ日時を持っていないとき、結果は **null** である。（これが望む処理でないとき、**where time of it is present** によっていつでも被演算子をふるいにかけることが出来る）。被演算子のﾘｽﾄの項目数が足りない場合は、可能な限りの項目を返す。これは **x** が空の場合、**earliest 1 from x** と **earliest x** が異なることを意味する。つまり前者は**()**を返し、後者は **null** を返す。被演算子のプライマリ日時は保持される。使い方は次の通りである。：

<n:any-type> := EARLIEST <1:number> FROM <m:any-type>

() := EARLIEST 2 FROM ()

9.14.11 Latest ... From (二項演算子、右結合性)

latest ... from 演算子は一番目の被演算子として数字（以下Nとする）を、二番目の被演算子としてﾘｽﾄをとる。この演算子は被演算子のﾘｽﾄから最新の N 個の項目を抽出し、それをﾘｽﾄとして返す。結果のﾘｽﾄの項目の並び順序は、被演算子のﾘｽﾄの項目順序と同じである。Nが非負の整数でないときは **null** を返す。どの要素もプライマリ日時を持たない場合は、**null** を返す。（これが望む処理でないとき、**where time of it is present** によっ

ていつでも被演算子をふるいにかけることが出来る）。被演算子のリストの項目数が足りない場合は、可能な限りの項目を返す。これは **x** が空の場合、**latest 1 from x** と **latest x** が異なることを意味する。つまり前者は**()**を返し、後者は **null** を返す。被演算子のプライマリ日時は保持される。使い方は次の通りである。：

```
<n:any-type> := LATEST <1:number> FROM <m:any-type>
    () := LATEST 2 FROM ()
```

9.14.12 Index Extraction Transformation Operators インデックス抽出変換演算子

これらの演算子は、インデックス抽出機能を持たない同種の演算子と似た働きをするが、要素自身を返すのではなく、指定した条件に一致した要素のインデックスを値として返すところが異なる。これらの演算子はプライマリ日時を保持しない。

9.14.12.1 Index Minimum ... From (二項演算子、右結合性)

index minimum ... from 演算子にはひとつの同義語 **index min ... from** がある。この演算子は一番目の被演算子として数字（以下Nとする）を、2番目の被演算子として順序型のデータからなる均質なリストをとる。この演算子は被演算子のリストから小さいほうの N 個の項目のインデックスを抽出し、それをリストとして返す。結果のリストの項目の並び順序は、第二被演算子中の項目順序と同じである。また、すべてのデータは複製される。N が非負の整数でない場合、**null** が返される。被演算子のリストの項目数が足りない時は、可能な限りの数のインデックスを返す。同じ値が複数ある時は、プライマリ日時の新しい方の要素が選択される。プライマリ日時は保持されない。使い方は次の通りである。：

```
<n:number> := INDEX MINIMUM <1:number> FROM <m:ordered>
    (1,4) := INDEX MINIMUM 2 FROM (11,14,13,12)
    (3,4,6) := INDEX MINIMUM 3 FROM (3,5,1,2,4,2)
    null := INDEX MIN 2 FROM (3, "asdf")
    (,1) := INDEX MINIMUM 2 FROM 3
    () := INDEX MINIMUM 0 FROM (2,3)
```

9.14.12.2 Index Maximum ... From (二項演算子、右結合性)

index maximum ... from 演算子にはひとつの同義語 **index max ... from** がある。この演算子は一番目の被演算子として数字（以下Nとする）を、2番目の被演算子として順序型のデータからなる均質なリストをとる。この演算子は被演算子のリストから大きいほうの N 個の項目のインデックスを抽出し、それをリストとして返す。結果のリストの項目の並び順序は、第二被演算子中の項目順序と同じである。また、すべてのデータは複製される。N が非負の整数でない場合、**null** が返される。被演算子のリストの項目数が足りない時は、可能な限りの数のインデックスを返す。同じ値が複数ある時は、プライマリ日時の新しい方の要素が選択される。プライマリ日時は保持されない。使い方は次の通りである。：

```
<n:number> := INDEX MAXIMUM <1:number> FROM <m:ordered>
    (2,3) := INDEX MAXIMUM 2 FROM (11,14,13,12)
    (2,3,5) := INDEX MAXIMUM 3 FROM (3,5,1,2,4,2)
    null := INDEX MAX 2 FROM (3, "asdf")
    (,1) := INDEX MAXIMUM 2 FROM 3
    () := INDEX MAXIMUM 0 FROM (2,3)
```

9.14.12.3 First… From; Last… From

First … From、**Last … From** に対応するインデックス抽出変換演算子は存在しない。というのも、**First … From** は **seqto** 演算子で、また **Last … From** は **seqto** 演算子と **count** 演算子を組み合わせることで実現できるからである。したがってこの機能が必要なときは、次のようになる：

Index First x From y : 1 seqto x

Index Last x From y : (count(y)-x) seqto count(y)

9.15 変換照会演算子

9.15.1 概要

変換照会演算子はデフォルトのリスト処理やプライマリ日時処理は行わない。この演算子はリストを変換して別のリストを生成する。被演算子が単一項目の場合、長さ1のリストとして扱われる。1つの項目しかなくても結果は必ずリストとなる（但し、エラーがあった場合、結果は **null** となる）。

変換照会演算子は照会の結果にのみ適応することができる。というのも、被演算子のリストの各項目に日時が付いていなくてはならないからである。そうでないデータに適用された場合は **null** が返される。

変換照会演算子にはオプションで **of** が付く。

9.15.2 Interval (単項演算子、右結合性)

Interval 演算子はリストで隣り合う二つの項目のプライマリ日時の時間差を返す。この演算子は **increase** 演算子と似ている。被演算子のプライマリ日時は失われる。この使い方は次の通りである。（ここで **data** は照会の結果得られたもので、以下のプライマリ日時を持つとする：**1990-03-15T15:00:00, 1990-03-16T15:00:00, 1990-03-18T21:00:00**）：

```
<n:duration> := INTERVAL <m:any-type>
    (1 day, 2.25 days) := INTERVAL data
    null := INTERVAL (3,4)
```

9.16 数学関数演算子

数学関数演算子はすべて単項演算子で数値に作用する。もし、不正な演算をしようとすると（例えば **log 0**）**null** が返される。

9.16.1 Arccos (単項演算子、右結合性)

arccos 演算子は被演算子のアークコサイン（ラジアン表示）を計算する。使い方は次の通りである。：

```
<n:number> := ARCCOS <n:number>
    0 := ARCCOS 1
```

9.16.2 Arcsin (単項演算子、右結合性)

arcsin 演算子は被演算子のアークサイン（ラジアン表示）を計算する。使い方は次の通りである。：

```
<n:number> := ARCSIN <n:number>
    0 := ARCSIN 0
```


9.16.3 Arctan (単項演算子、右結合性)

arctan 演算子は被演算子のアークタンジェント（ラジアン表示）を計算する。使い方は次の通りである。:

```
<n:number> := ARCTAN <n:number>
    0 := ARCTAN 0
```

9.16.4 Cosine (単項演算子、右結合性)

cosine 演算子にはひとつの同義語 **cos** がある。被演算子（ラジアン表示）のコサインを計算する。使い方は次の通りである:

```
<n:number> := COSINE <n:number>
    1 := COSINE 0
```

9.16.5 Sine (単項演算子、右結合性)

sine 演算子にはひとつの同義語 **sin** がある。被演算子（ラジアン表示）のサインを計算する。使い方は次の通りである。:

```
<n:number> := SINE <n:number>
    0 := SINE 0
```

9.16.6 Tangent (単項演算子、右結合性)

tangent 演算子にはひとつの同義語 **tan** がある。被演算子（ラジアン表示）のタンジェントを計算する。使い方は次の通りである。:

```
<n:number> := TANGENT <n:number>
    0 := TANGENT 0
```

9.16.7 Exp (単項演算子、右結合性)

exp 演算子は自然対数の底 e の被演算子乗を計算する。使い方は次の通りである。:

```
<n:number> := EXP <n:number>
    1 := EXP 0
```

9.16.8 Log (単項演算子、右結合性)

log 演算子は被演算子の自然対数を返す。使い方は次の通りである。:

```
<n:number> := LOG <n:number>
    0 := LOG 1
```

9.16.9 Log10 (単項演算子、右結合性)

log10 演算子は被演算子の 10 を低とする対数を返す。使い方は次の通りである。:

```
<n:number> := LOG10 <n:number>
    1 := LOG10 10
```

9.16.10 Int (単項演算子、右結合性)

int 演算子は被演算子と小さいか等しい最大の整数を返す（負の無限大方向に切り下げる）。この演算子は **floor** 演算子（9.16.11節参照）の同義語である。使い方は次の通りである。:

```
<n:number> := INT <n:number>
```

-2 := INT -1.5
-2 := INT -2.0
1 := INT (1.5)
-3 := INT (-2.5)
-4 := INT (-3.1)
-4 := INT (-4)

9.16.11 Floor (単項演算子、右結合性)

floor 演算子は **int** 演算子の同義語である。この演算子は被演算子と小さいか等しい最大の整数を返す（負の無限大方向に切り下げる）。

9.16.12 Ceiling (単項演算子、右結合性)

celling 演算子は被演算子と大きいか等しい最小の整数を返す（正の無限大方向に切り上げる）。使い方は次の通りである。：

<n:number> := CEILING <n:number>
-1 := CEILING -1.5
-1 := CEILING -1.0
2 := CEILING 1.5
-2 :=CEILING (-2.5)
-3 := CEILING (-3.9)

9.16.13 Truncate (単項演算子、右結合性)

truncate 演算子は任意の数の小数点以下を取り去る（ゼロに向かって切り捨てる）。使い方は次の通りである。：

<n:number> := TRUNCATE <n:number>
-1 := TRUNCATE -1.5
-1 := TRUNCATE -1.0
1 := TRUNCATE 1.5

9.16.14 Round (二項演算子、右結合性)

round 演算子は数字を丸めて整数にする。

正の数の場合：被演算子の小数点以下が0．5と同じか大きい場合、丸めて直近で大きい整数にする。小数点以下が0．5より小さい場合は丸めて直近で小さい整数とする。

負の数の場合：被演算子の小数点以下の絶対値が0．5と同じか大きい場合、整数部分を直近で小さな負の整数にする。小数点以下の絶対値が0．5より小さい場合、丸めて直近で大きい整数にする。

使い方は次の通りである。：

<n:number> := ROUND <n:number>
1 := ROUND 0.5
3 := ROUND 3.4
4 := ROUND 3.5
-4 := ROUND -3.5
-3 := ROUND -3.4
-4 := ROUND -3.7

9.16.15 Abs (単項演算子、右結合性)

abs 演算子は被演算子の絶対値を返す。使い方は次の通りである。:

```
<n:number> := ABS <n:number>
    1.5 := ABS (-1.5)
```

9.16.16 sqrt (単項演算子、右結合性)

sqrt 演算子は被演算子の平方根を返す。虚数は扱わないので、負の数の平方根は **null** となる。使い方は次の通りである。:

```
<n:number> := SQRT <n:number>
    2 := SQRT 4
    null := SQRT(-1)
```

9.16.17 As number (単項演算子、非結合性)

as number 演算子は文字列またはブール値を数値に変換する。数値に変換可能な場合は数値が返される。さもなければ **null** が返される。被演算子のプライマリ日時は保持される。この演算子の通常の使い方は、文字列に含まれる有効な数値表現（例えば"123"）を対応する数値に変換することである。文字列に有効な数値が含まれない場合は **null** が返される。ブール値は次のように変換される。すなわち、ブール値の「真」は数値の1、ブール値の「偽」は数値の0となる。

```
<n:number> := <n:numeric string> AS NUMBER;
    5 := "5" AS NUMBER;
    null := "xyz" AS NUMBER;
<n:number> := <n:Boolean> AS NUMBER;
    1 := True AS NUMBER;
    0 := False AS NUMBER;
<n:number> := <n:number> AS NUMBER;
    6 := 6 AS NUMBER;
    (7, 8, 230, 4100, null, null, 1, 0, null, null, null) := ("7", 8, "2.3E+2", 4.1E+3, "ABC", Null, True, False, 1997-10-
     31T00:00:00, now, 3 days) AS NUMBER;
    ():= () AS NUMBER;
```

9.17 日時関数演算子

日時関数演算子はデフォルトのプライマリ日時処理を行わない。

9.17.1 Time (単項演算子、右結合性)

time 演算子は、照会（8.9 節参照）の結果得られた値のプライマリ日時（例えば発生時刻）を返す。データがプライマリ日時を持たない場合は **null** が返される。**time** 演算子は被演算子のプライマリ日時を保持する。従って **time time x** は **time x** と等価である。使い方は次の通りである。（ただし **data0** は照会の結果得られた項目で、そのプライマリ日時は **1990-03-15T15:00:00** とする）:

```
<n:time> := TIME [OF] <n:any-type>
    1990-03-15T15:00:00 := TIME OF data0
    1990-03-15T15:00:00 := TIME TIME data0
    (null,null) := TIME (3,4)
```

time 演算子の逆演算（値にプライマリ日時を設定する）は、代入文の左辺で **time** 演算子を使うことで可能となる。例えば:

```
TIME [OF] <n:any-type> := <n:time>;
    TIME data1 := time data2;
```

もし、代入文の左辺の変数がリストを参照している場合、時間代入処理は未定義である。Arden Syntax の将来のバージョンでは処理内容を正式に定義するかも知れない。

10 LOGIC スロット

10.1 目的

Logic スロットは, Data スロットから得られた患者に関するデータを使用し, データを操作し, ある条件をテストし, Action スロットを実行するべきかどうか決定する。実際の保健医療に関する論理のほとんどが得られるのはこのスロットからである。

10.2 Logic スロット文

Logic スロットは 1 セットの文から構成される。

10.2.1 代入文

代入文は, 変数に式の値を入れる。二つの書き方がある:

```
<identifier> := <expr> ;
    LET <identifier> BE <expr> ;
```

<**identifier**> は変数の名前を表わす識別子である。<**expr**>は 7.2.2 節の中で定義されるような有効な式である。

代入文の後に生じる識別子へのどんな参照も, 式(それが別の構造化スロットにあっても; 例えば Action スロット)から割り当てられた値を返す。同じ変数への次の代入は値を上書きする。変数がその最初の代入の前に参照される場合, **null** が返される(これに依存することは推奨できないプログラミング手法である)。

Data スロットの中で代入された後, 次の変数は logic スロットの中で再度代入することができない:**event**(11.2.2 節), **mlm**(11.2.3 節)および **interface**(11.2.12 節)。いったん Data スロットに定義されたものは, 変更すべきではない。

これらの文を実行した後の変数 **var2** の値は **5** である:

```
var1 := 1;
var1 := 3;
var2 := var1 + 2;
```

10.2.2 If-Then 文

if-then 文は, 式の値に基づいた条件付きの実行を許可する。式(<**expr**>)が単一ブール型の **true** と等しいかどうかをテストする。True の場合, (<**block**>)のブロック文が実行される。(ブロック文は単に他の if-then 文を含めてのコレクションである;したがって, if-then 文は入れ子の構造である。) 式がリストである場合, あるいはそれが true 以外のどんな単一項目である場合, ブロック文は実行されない。その後, コントロールの流れは次の文に継続する。if-then 文にはいくつかの形式がある:

10.2.2.1 単純な if-then 文

この形式は **<expr1>** が **true** の時**<block1>**を実行する

```
IF <expr1> THEN
```



```
    <block1>
ENDIF;
```

10.2.2.2 If-Then-Else 文

<**expr1**>が **true** の場合，この形式は<**block1**>を実行する;そうでなければ，<**block2**>を実行する:

```
IF <expr1> THEN
    <block1>
ELSE
    <block2>
ENDIF;
```

10.2.2.3 If-Then-Elseif 文

この形式は，<**expr1**>から<**exprN**>(どんな数もあるかもしれない)までの各々の式を順次テストする。**true** となるものを見つけると，それに関連するブロックが実行される。一旦 1 つのブロックが実行されれば，他の式はテストされない。また，他のブロックも実行されない。式のどれも **true** でない場合，<**blockE**>が実行される。**else** <**blockE**>部分はオプションである。形式は以下の通りである:

```
IF <expr1> THEN
    <block1>
ELSEIF <expr2> THEN
    <block2>
ELSEIF <expr3> THEN
    <block3>
...
ELSEIF <exprN> THEN
    <blockN>
ELSE
    <blockE>
ENDIF;
```

10.2.2.4 Null の扱い方

true でないものと **false** の違いを強調することは重要である。すなわち，if-then-else 文の一部である **else** は，式が **false** あるいは **null** または **true** 以外の何であろうとも実行される。したがって同じであるように見えるこれらの 2 つの if-then 文は，**var1** が **null** の場合，異なる結果となる。

```
IF var1 THEN
    var2 := 0;
ELSE
    var2 := 45;
ENDIF;

IF not(var1) THEN
    var2 := 45;
ELSE
    var2 := 0;
ENDIF;
```

null の問題を回避することは，最初にその存在をテストし，次に，**true** のテストを行うことがより安全である。

```
IF var1 is Boolean THEN
IF var1 THEN
```

```
    var2 := "var1 is true";
ELSE
    var2 := "var1 is false";
ENDIF;
ELSE
    var2 := "var1 is null or some other type";
ENDIF;
```

10.2.2.5 List の取り扱い

Lists は常に真ではない;したがって, List を含んでいる式の使用は常に同じ否定的な結果を生むことになる。代わりに, ブール集合オペレータのうちの一つは使用されるべきである:**any**, **all**, あるいは **no**(9.12.13節, 9.12.14節および9.12.15節を参照)。例えば, **Bool_list** の中の要素のうちのどれかが true の場合の文を実行する使い方:

```
IF any(Bool_list) THEN
    var2 := 0;
ENDIF;
```

10.2.3 Conclude 文

Conclude 文は, logic スロット中の実行を終了する。Conclude 文での式**(<expr>)**が 1 つの **true** である時, その後, Action スロットは直ちに実行される。そうでなければ, 全体の MLM は直ちに終了する。それ以上, logic スロット中の実行は式にかかわらず行われない。Logic スロット中の Conclude 文は 1 つ以上あるかもしれないが, 1 つだけが MLM の単一の実行で実行される。形式は以下の通りである:

```
CONCLUDE <expr>;
```

null についての if-then 文の注意とリスト(10.2.2 節参照)は conclude 文にも適用できる。

Conclude 文が実行されない場合, その最後の文を実行した後, logic スロットは終了する。また, Action スロットは実行されない。実際に, デフォルトは **conclude false** である。

有効な文の例:

```
CONCLUDE false;
CONCLUDE potas > 5.0;
```

10.2.4 Call 文

Call 文は, MLM の入れ子構造を許可する。MLM ファイル名を与えられて, MLM はオプションのパラメータで直接呼び出され, 0 あるいはそれ以上の結果を返すことができる。イベント定義を与えることにより, そのイベントによって通常起動されるすべての MLM は呼び出される;呼び出された MLM はオプションのパラメータを与えることができ, 自由に, 結果を返すことができる。インターフェース定義を与えることにより, 外部の関数はオプションのパラメータで直接呼び出され, 0 あるいはそれ以上の結果を返すことができる。2 つの基礎的な形式がある(この二つは等価である):

```
<var> := CALL <name>;
    LET <var> BE CALL <name>;

<var> := CALL <name> WITH <expr>;
    LET <var> BE CALL <name> WITH <expr>;

(<var>, <var>, …) := CALL <name> WITH <expr>;
    LET (<var>, <var>, …) BE CALL <name> WITH <expr>;
```



```
<var> := CALL <name> WITH <expr>, …, <expr>;
    LET <var> BE CALL <name> WITH <expr>, …, <expr>;

(<var>, <var>, …) := CALL <name> WITH <expr>, …, <expr>;
    LET (<var>, <var>, …) BE CALL <name> WITH <expr>, …, <expr>;
```

10.2.4.1 コンマ

呼び出しへの引数がコンマ(**argument**, 11.2.4 節参照)によって分離され, さらに, コンマが被演算子(list 構築, 9.2.1参照)なので, 外見上曖昧な表現になる。この曖昧さはパラメータ・セパレータとしてのコンマで分解される。コンマ・オペレータあるいは同じかより低い先行のもう一つのオペレータを含むどんな被演算子式も, 括弧で囲まれなければならない。例えば,

この呼び出しは 3 つの被演算子を渡す:

```
x := CALL xxx with (a,b),(c merge d),e+f;
```

この呼び出しは2つの被演算子を渡す:

```
y := CALL yyy WITH expr1, expr2;
```

この呼び出しは上記のものに似ているように見えるが, 1 つの被演算子を単に渡すだけである:

```
z := CALL zzz WITH (expr3, expr4);
```

10.2.4.2 <name>

<name>は識別子である。Data スロット中の MLM 文によって定義されるか(11.2.3 節参照), 有効な MLM 変数として, 有効なイベント変数として Data スロット中のイベント文によって定義されるか(11.2.2 節参照), 有効なインターフェース変数として Data スロット中のインターフェース文によって定義されなければならない(11.2.12 節参照)。

10.2.4.3 <exprs>

<**expr**>はオプションのパラメータ(それらは list と null を含む任意のタイプかもしれない)である。パラメータに関連したプライマリ日時が保持される。

10.2.4.4 <var>

<**var**>は結果を代入するローカルの変数を表わす識別子である。

10.2.4.5 MLM 呼び出し

Call 文が実行される場合, **<name>**が MLM 変数ならば, 主要な MLM(すなわち call を出すもの)は中断され, 指定された MLM が呼び出される。呼び出された MLM がその Data スロット(11.2.4 節参照)の中に引数文を行っている場合, <expr>の値が代入される。呼び出された MLM の引数文が Call 文によって送られたより多くの変数(パラメータ)を持っている場合, **null** は余分な変数に割り当てられる。Call 文が呼び出された MLM が予期しているより多くの変数(パラメータ)を渡す場合, 余分なパラメータは暗黙的に削除される。呼び出された MLM が実行され, 終了する時, 主要な MLM の実行は再開する。True で, 呼び出された MLM が終えて, 呼び出された MLM の Action スロット(12.2.2 節参照)に return 文がある場合, その表現の値は**<var>**に代入される。return 文が呼び出す MLM が受けることができるより多くの値を持っている場合, 余分な返り値は暗黙的に削除される。return 文が呼び出す MLM が予期しているより少ない値を持っている場合余分な返り値は **null** である。return 文がない場合, あるいは呼び出された MLM が偽って終える場合, **null** は**<var>**に代入される。例:

```
var1 := CALL my_mlm1 WITH param1, param2;

(var2, var3, var4) := CALL my_mlm2 WITH param1, param2;
```

10.2.4.6 Event 呼び出し

<name>がイベント変数である場合, 実行は似ている。主な MLM は中断され, 指定されたイベントを参照するスロットを呼び起こすすべての MLM は実行される 13 章参照)。それらの被演算子文 によってどれかがある場合, それらは各々パラメータを受け取る。すべての呼び出された MLM の return 文の結果は, リストへともに連結される; return 文のない MLM や一つの **null** を返す MLM は結果に含まれない。返されたオーダ値は実装に依存する。結果は**<var>**に代入される。また, 実行は継続する。たとえアイテムが1つでも, **<var>**は常にリストになる。例:

```
var1 := CALL my_event WITH param1, param2;
```

10.2.4.7 Interface 呼出

Call 文が実行される場合, **<name>**がその後 INTERFACE 変数ならば, MLM(すなわち呼び出しを出すもの)は中断され, 指定された INTERFACE が呼び出される。呼び出された INTERFACE 関数が変数(パラメータ)を受ける場合, <**expr**>の値が割り当てられる。呼び出された INTERFACE 関数が call 文によって送られたより多くの変数(パラメータ)を予期する場合, **null** は余分な変数に代入される。呼び出された関数が実行され, 終了する時, MLM の実行は再開する。呼び出された関数が 1 つ以上の値を返す場合, 値は<var>に代入される。関数が呼び出す MLM が受けることができるより多くの値を返す場合, 余分な返り値は暗黙的に削除される。
INTERFACE 関数が呼び出す MLM が予期しているより少ない値を返す場合, 余分な値は **null** である。機能が値を返さない場合, **null** は**<var>**に代入される。例:

```
var1  := CALL my_interface_function1 WITH param1, param2;


(var1, var2, var3) := CALL my_interface_function2 WITH param1, param2;
```

10.2.4.8 例：Call 文

有効な call 文:

```
/* Define find_allergies MLM */
find_allergies := MLM 'find_allergies';
/* Lists two medications and their allergens */
med_orders:= ("PEN-G", "aspirin");
med_allergens:= ("penicillin", "aspirin");
/* Lists three patient allergies and their reactions */
patient_allergies:= ("milk", "codeine", "penicillin" );
patient_reactions:= ("hives", NULL, "anaphylaxis");
/* Passes 4 arguments and receives 3 lists as values */
(meds, allergens, reactions):= call find_allergies with med_orders,
                                        med_allergens,
                                        patient_allergies,
                                        patient_reactions;
```

10.2.4.9 例： Interface 文

有効な Interface 文:

```
/* Define find_allergies external function*/
find_allergies := INTERFACE {\\RuleServer\AllergyRules\my_institution\find_allergies.exe};
/* Lists two medications and their allergens */
med_orders:= ("PEN-G", "aspirin");
med_allergens:= ("penicillin", "aspirin");
/* Lists three patient allergies and their reactions */
patient_allergies:= ("milk", "codeine", "penicillin" );
patient_reactions:= ("hives", NULL, "anaphylaxis");
```



```
/* Passes 4 arguments and receives 3 lists as values */
(meds, allergens, reactions):= call find_allergies with med_orders,
                                                        med_allergens,
                                                        patient_allergies,
                                                        patient_reactions;
```

10.2.5 WHILE ループ

ループする単純な形式は, while ループによって提供される。形式は以下の通りである:

```
WHILE <expr> DO
    <block>
ENDDO;
```

While ループは式(<expr>)が 1 つのブール型 true(if-then 文法で説明した条件付きの実行と同様である—10.2.2 節参照)に等しいかどうかテストする。それがそうである場合, **<expr>**が **true** ではないまで, 文**(<block>)**のブロックは繰り返し実行される。**<expr>**が **true** でない場合, ブロックは実行されない。

MLM の中の **while** ループを使うとき, 無限ループを作成する可能性があるので、制作者は注意すべきである。無限ループを回避するのは, コンパイラではなく制作者の責任である。

例:

```
/* Initialize variables */
a_list:= ();
m_list:= ();
r_list:= ();
num:= 1;
/* Checks each allergen in the medications to determine if the patient is allergic to it */
while num <= (count med_allergen) do
allergen:= last(first num from med_allergens);
allergy_found:= (patient_allergies = allergen);
reaction:= patient_reactions where allergy_found;
medication:= med_orders where (med_allergens = allergen);
/* Adds the allergen, medication, and reaction to variables that will */
/* be returned to the calling MLM */
If any allergy_found then
a_list:= a_list, allergen;
m_list:= m_list, medication;
r_list:= r_list, reaction;
endif;
/* Increments the counter that is used to stop the while-loop */
num:= num + 1 ;
enddo;
```

10.2.6 FOR ループ

ループする別の形式は FOR ループで提供される。形式は以下の通りである:

```
FOR <identifier> in <expr> DO
    <block>
ENDDO;
```

<expr>は通常リスト･ジェネレータになる。**<expr>**が empty か null の場合，ブロックは実行されない。そうでなければ，ブロックは，**<expr>**の中で連続する要素を引き受ける**<identifier>**で実行される。**<block>**(これが試みられる場合，コンパイラは編集エラーを生じるに違いない)の内部で**<identifier>**に要素を割り当てることができない。**enddo** の後に，**<identifier>**は不確定になり，その値は使われない。コンパイラはエラーとしてフラグを立てる。

例:

```
/* Initialize variables */
a_list:= ();
m_list:= ();
r_list:= ();
/* Checks each allergen in the medications to determine if the patient is allergic to it */
for allergen in med_allergens do
    allergy_found:= (patient_allergies = allergen);
    reaction:= patient_reactions where allergy_found;
    medication:= med_orders where (med_allergens = allergen);
    /* Adds the allergen, medication, and reaction to variables that will */
    /* be returned to the calling MLM */
    If any allergy_found then
        a_list:= a_list, allergen;
        m_list:= m_list, medication;
        r_list:= r_list, reaction;
    endif;
enddo;
```

反復のセット番号を使用している例:

```
for i in (1 seqto 10) do
    …
enddo;
```

10.3 Logic スロットの使用方法

Logic スロット中の一般的なアプローチは患者の中のある条件をテストするために Data スロットの中で得られた患者データを操作するオペレータおよび式を使用することである。十分なデータ（肯定あるいは否定）が蓄積されると conclude 文が実行される。Logic スロットに conclude 文がない場合，決して **true** では終わらず，Action スロットは実行されない。いくつかの logic スロットは単純(例えば血清カリウムが 5.0 を越えるものでも，テストする)で，いくつかは複雑である(例えば，分析スコアを計算する)。

11 DATA スロット

11.1 目的

Data スロットの目的は MLM の残りの中で使用されるローカルの変数を定義することである。ゴールは 1 つのスロットへの設立に特有の部分を分離することである。Data スロット内では，機関固有の部分が MLM 文法の邪魔をしないように，機関固有の部分はマッピング節の中に置かれる (7.1.8 節参照)。

11.2 Data スロット文

Data スロットの中で割り当てられた後，次の変数は logic スロットの中で再度割り当てることができない:**event**(11.2.2 節)，**mlm**(11.2.3 節)および **interface** (11.2.12 節)。一度 Data スロットで定義されると、変わらない。

11.2.1 Read 文

データの主要なソースは患者データベースである。各機関はそれ自身の照会を行う必要がある;データベースは階層的で, リレーショナル, オブジェクトオリエンテッドかもしれない。ボキャブラリは, データベース中の実体が機関から機関へ変わるだろうとかつては言われていた。(この仕様のこのバージョンでの標準の照会を選択する試みはされなかった。) read 文は, 普遍的な部分からの機関に特有のデータベースクエリの部分を分離することをデザインされている。

read 文がその患者データベースから入力を引き出すのに制限はない。例えば, read 文は医療辞書にアクセスするかもしれない;あるいは, インタラクティブにどこかに情報を要請するかもしれない(また, コンパイラがオン・デマンドの最適化を行う場合, もしただ必要ならば, 相互作用は起こるかもしれない)。終わり方は実装に定義されている。

11.2.1.1

データベースの照会はそれ自身 3 部からなる。:集合か変形演算子, 時間の制約および照会の残りである。下位互換については, 括弧が**<mapping>WHERE<constraint>**部分のまわりに置かれるかもしれない。Read 文の一般的な形式である(2 つの書き方がある)。

```
<var> := READ <aggregation> <mapping> WHERE <constraint>;
    LET <var> BE READ <aggregation> <mapping> WHERE <constraint>;
```

11.2.1.2 定義

<**var**>は照会結果を代入される変数である。

<aggregation>は集合オペレータ(0章参照)あるいは変形オペレータ(0節参照)(それは照会制約の後に適用される)である。**<aggregation>**が省略される場合, 制約を満たすデータがすべて返される。次の集合および変形オペレータだけが許可される:

exist
sum
average
avg
minimum
min
maximum
max
last
first
earliest
latest
minimum ... from
min ... from
max ... from
maximum ... from
last ... from
first ... from
earliest ... from
latest ... from

デフォルト・ソートオーダで, 最初と最後は最速と最遅と等価である。

<constraint>は, 左の被演算子として生じる **it**(あるいは **they**)を持った比較オペレータである(0章参照)。この場合は, 照会の形に言及する。比較オペレータは, 照会のための時間制約を指定する。**<constraint>**が省略される場合, 制約は時間通りではない。有効な制約の例は以下の通りである:

they occurred within the past 3 days

it occurred before the time of surgery

<mapping>は有効なマッピング節(7.1.8 節参照)で, {} 括弧で囲まれたクエリの機関固有の部分を含んでいる。それは集合と時間の制約条件が見当たらない以外は, どんな用語および照会を実行する機関において必要なすべての照会規則も含んでいる。<**mapping**>が要求される。

11.2.1.3 例

有効な read 文({} 括弧内の部分は任意):

```
var1 := READ {select potassium from results where specimen = `serum`};
var1 := READ last {select potassium from results};
LET var1 BE READ {select potassium from results} WHERE it occurred within the past 1 week;
var1 := READ first 3 from {select potassium from results} WHERE it occurred within the past 1 week;
```

11.2.1.4 影響

read 文の影響は, MLM のどこか他のところに使用することができる変数に患者データベース中のデータをマッピングして, 照会を実行することである。read 文の実行は機関に特有になる。時間制約をマッピング節の内に他の制約があるものすべてに加えなければならない。また, 集合か変形オペレータも照会を完成するために付け加えられなければならない。

11.2.1.5 Result タイプ

照会の結果は, 返されるアイテムごとの主要な時間を含んでいる(8.9 節参照)。**<aggregation>**が集合オペレータである場合, 照会は単一のアイテムを返す。**<aggregation>**が変形オペレータか, それが不在の場合, 照会はリストを返す。したがって, 照会が患者の生年月日のように, 通常単一の実体を要求しても, もし集合オペレータが適用されなければ(しかし, リストは単一の値だけを含んでいるかもしれない, それがスカラと判別不能であろう), リストが想定される。この理由は, 患者データベースに生年月日に対する多数の値があるかもしれないということである;最後のものが正確であると想定されるということかもしれない。例えば

```
birthdate := READ last {select birthdate from demographics};
```

11.2.1.6 多変数

照会は一度に 1 つを越える結果を返すかもしれない。これは 1 つの統合血液サンプル内の対応するテストを保持するためにテストの蓄積に役立つ。2 つのバージョンは等価である(まわりの括弧はオプション):

```
(<var>, <var>, ...) := READ <aggregation> <mapping> WHERE <constraint> ;
LET (<var>, <var>, ...) BE READ <aggregation> (<mapping> WHERE <constraint>);
```

これは「リストのリスト」が許可されるただ一つの状況である。どこの制約条件(もしあれば)かは, 生じるリストの各々に別々に適用される。照会は, 予備選択同時と共に, 同数の要素を常に返さなければならない。

11.2.1.7

<var>は括弧内に1つあるいはそれ以上あるかもしれない。**<aggregation>**, **<constraint>**および**<mapping>**は同じように定義される。多数の実体が直ちに尋ねられているという事実は機関に特有の部分**<mapping>**で表わされる。**<aggregation>**および**<constraint>**は, 個々の変数上で別々に実行される**;<mapping>**が一致する初期時間を備えた値をすべて返すかどうかは機関定義される。例えば,

```
/* in this example three anion gaps are calculated */
(Na,Cl,HCO3) := read last 3 from {select sodium, chloride, bicarb from electro};
```



```
anion_gap := Na - (Cl + HCO3) ;
```

11.2.1.8

実装は, 値が必要となった最初の時の前に読まれたマッピングを保証しなければならない時を除いては, マッピングを読むオーダは評価され, 定義されない。実装は, 読まれたマッピングが副作用を持っても, 読まれたマッピングを実行することを回避するためにコードを最適化するかもしれない。

11.2.2 Event 文

イベントステートメントは変数に機関に特有のイベント定義を代入する。イベントは患者データベース中の挿入か最新版, あるいは他の医学的に適切な発生でありえる。他の MLM(10.2.4 参照)を呼ぶために, 変数は, evoke スロット(13 章参照)の中で, read 文の一部として, および **logic** または **action** スロット中のブール値として現在使用される。二通りある。

```
<var> := EVENT <mapping>;
LET <var> BE EVENT <mapping>;
```

11.2.2.1 定義

<var>は定義されるイベントを表わす変数である。evoke スロットの中で, あるいは call 文の一部として単に使用することができる。

<mapping>は機関に特有のイベント定義を含んでいる有効なマッピング節(7.1.8 節参照)である。イベントがどのように定義され使用されるかは機関による。

The variable that represents the event can be treated like a Boolean in the **logic** or **action** slots.

イベントを表わす変数は, logic または action スロットにおいてのブール型のように扱うことができる。

Time オペレータ(9.17 節)はイベント変数に適用することができる。イベントの臨床的に適切な時間を生じる。**eventtime** 変数(それはデータベース(8.4.4節)にイベントが記録された時を参照する)とは異なるかもしれない。

イベントマッピングが評価されるオーダは, その値が必要な最初の時の前にイベントマッピングが評価されることを実装が保証するに違いない場合, 以外は定義されない。

11.2.2.2 例

```
event1 := EVENT {storage of serum potassium};
```

11.2.3 MLM 文

MLM ステートメントは変数に有効な mlmname を代入する。10.2.4 節の中で定義されるように, その変数は別の MLM を呼ぶために call 文の一部としてのみ現在使用される。2 つの基本的な形式がある(この二つは等価である):

```
<var> := MLM <term>;
LET <var> BE MLM <term>;
<var> := MLM <term> FROM INSTITUTION <string>;
LET <var> BE MLM <term> FROM INSTITUTION <string>;
```

11.2.3.1 例

```
LET MLM1 BE MLM 'my_mlm1';
mlm2 := MLM 'my_mlm2.mlm' FROM INSTITUTION "my institution";
```

11.2.3.2　定義

<var>は呼ばれる MLM を表わす変数である。Call 文の一部としてのみ使用することができる。

<term>は 7.1.7 節の中で定義されるような有効な定形の用語である。MLM の mlmname と呼ばれる。**mlm_self**(無感覚な場合)は現在の MLM の名前を表わす特別の定数である。

<string>は 7.1.6 節の中で定義されるような有効な一定のストリングである。もし指定されれば, 呼び出される MLM の institution スロットで見つかった機関名である。

機関が指定されれば, 機関名、mlmname 及び最新のバージョン番号を使用することによりユニークな MLM が見つかる。機関が指定されなければ, メイン MLM と同じ機関名、mlmname、MLM の確認及び最新のバージョン番号を使用することによりユニークな MLM が見つかる。バージョンの正確な形式は機関に特有であるが, 機関内では, MLM の最新のバージョンを決定することは可能である(6.1.4節参照)。

11.2.3.3　例

```
mlm1 := MLM 'mlm_to_be_called';
mlm2 := MLM 'diagnosis_score' FROM INSTITUTION "LDS Hospital";
```

11.2.4　被演算子文

10.2.4 節の中で定義されるように, 被演算子文は別の MLM によって呼ばれる MLM によって使用される。主要な MLM がパラメータを呼ばれた MLM へ渡す場合, 呼ばれた MLM は被演算子文によってパラメータを受け取る。被演算子文は対応する過去の引数にアクセスする。したがって, 第 1 の可変<var1>は, 第 2 の引数などへの第 1 の過去の引数, 第 2 の変数<var2>に言及する。

変数の数が CALL から渡された被演算子の数より大きなところで, 変数の誤った組合せがある場合, **null** は余分な左手側変数に割り当てられる。MLM が呼び出しの代わりに起動される場合被演算子はすべて **null** (ちょうど他の初期化されていない変数のような)として扱われる。

2 つの基本的な形式(ペアは等価なバージョンを表わす)がある。1 つは単一のパラメータを受け取る。また, 他方は多数のパラメータを受け取る:

```
<var> := ARGUMENT;
LET <var> BE ARGUMENT;


(<var1>,<var2>,…) := ARGUMENT;
LET (<var1>,<var2>,…) BE ARGUMENT;
```

<var>は主要な MLM の call 文の **with** に続いて割り当てられる変数である。そのような式がなかった場合, あるいは MLM が別の MLM によって呼ばれなかった場合, **null** が割り当てられる。

11.2.4.1　例

MLM を呼ぶ場合:

```
var1 := CALL my_mlm WITH param1, (item1, item2);
```

"**my_mlm**"という呼び出された MLM の中で使用する場合:

```
(arg1, list1) := ARGUMENT;
```

11.2.5　Message 文

メッセージ文は変数に機関に特有のメッセージ(例えば, 警告)を代入する。機関が患者データベースの中でコード化されたメッセージを書くことを可能にしている(12.2.1 節参照)。2 つの書き方がある

```
<var> := MESSAGE <mapping>;
LET <var> BE MESSAGE <mapping>;
```

<var>は定義されるメッセージを表わす変数である。Write 文の中でのみ使用することができる。

<mapping>は有効なマッピング節(7.1.8 節)(それはメッセージ定義を含んでいる)である。メッセージがどのように定義され使用されるかは機関の責任である。

11.2.5.1 例

```
message1 := MESSAGE {pneumonia~23 45 65};
```

11.2.6 宛先文

宛先文は変数に機関に特有の宛先を代入する。それは, 機関に特有の宛先をメッセージに書くことを可能にする(12.2.1 節)。2 つ書き方がある:

```
<var> := DESTINATION <mapping>;
LET <var> BE DESTINATION <mapping>;
```

<var>は定義される宛先を表わす変数です。それは write 文の中で単に使用することができる。

<mapping>は機関に特有の宛先を表わす有効なマッピング節(7.1.8 節)である。宛先がどのように定義され使用されるかは機関の責任である。

11.2.6.1 例

この例では, 宛先は電子メールアドレスである。

```
destination1 := DESTINATION {email: user@cuasdf.bitnet};
destination2 := DESTINATION { attending_physician(Pt_id) };
destination3 := DESTINATION { "primary physician email" };
```

11.2.7 代入文

10.2.1 節の中で定義された代入文も, Data スロットの中で許可される。

11.2.8 If-Then 文

10.2.2 節の中で定義された if-then 文も, Data スロットの中で許可される。

11.2.9 Call 文

10.2.4 節の中で定義された call 文も, Data スロットの中で許可される。

11.2.10 WHILE ループ

10.2.5 節の中で定義された **while** ループも, Data スロットの中で許可される。

11.2.11 FOR ループ

10.2.6 節の中で定義された **for** ループも, Data スロットの中で許可される。

11.2.12 Interface 文

インターフェース文は変数に機関に特有の外部の関数インターフェース定義を代入する。インターフェース文は, 外部の関数(つまり別のプログラミング言語で書かれた機能)の仕様を許可する。時々医学のロジックはデータベース(**read** 文による)から直接利用可能でない情報を要求する。オペレーティング･システム関数あるいはライブラリを他のベンダから得られたと呼ぶことは望ましいかもしれない。その後, 外部の関数は, 指定された時, call 文で呼ぶことができる(10.2.4 節)。カール状の括弧(**{}**)は外部の関数を指定するために使用される。{} 括弧内の仕様は特定の実装である。2 つの書き方がある:

```
<var> := INTERFACE <mapping>;
LET <var> BE INTERFACE <mapping>;
```

<var>は定義されるインターフェースを表わす変数である。Call 文の一部として単に使用することができる。

<mapping>は機関に特有のイベント定義を含んでいる有効なマッピング節(7.1.8 節)である。関数インターフェースがどのように定義され使用されるかは機関の責任である。

11.2.12.1 例

```
data:
    /* Declares the third-party drug-drug interaction function */
    /* The implementation within the {}-braces shows that a string (char*) will be returned */
    /* when the third-party API (ThirdPartyAPI) is used to call */
    /* the drug-drug interaction function (DrugDrugInteraction) */
    /* The function expects that two medicaion strings (char*, char*) will be passed */
    func_drugint := INTERFACE {
            char* ThirdPartyAPI:DrugDrugInteraction (char*, char*)
            };
;;
evoke:
;;
logic:
    /* Calls the drug-drug interaction function */
    alert_text := call func_drugint with "terfenadine", "erythromycin";
```

11.3 Data スロットの使用方法

Data スロットは MLM の中でローカルに使用される変数に機関に特有の実体をマッピングするために使用される。1 つのスロットの中でマッピングを維持することは別の機関で使用される MLM を修正することを促進する。

Data スロットは logic スロットのように代入文や if-then 文と同じように実行することができます。ほとんどのロジックが logic スロットの中に残されるように勧められる。例えば, Data スロットの中で保健医療に関するデータのマッピングと論理記述を全て書き, logic スロットには単純な conclude 文だけを記述することもできるが, data スロットと logic スロットを分ける本来の意味を失わせてしまう。代入文と if-then 文は, ただデータベースクエリをサポートするために必要な Data スロットとして使われるべきである(例えば時間制約を計算するか扱う見当たらないデータのようなデータベース意味論の詳細を扱うこと)。

12 ACTION スロット

12.1 目的

logic スロットで特定された条件が true であると MLM が結論づけると，その条件に対してアクションが適切かどうかには関係なく，その action スロットは実行される。代表的なアクションとしては，医療従事者にメッセージを送ること，診療録に解釈を加えること，呼出している MLM に結果を返すこと，そして他の MLM を起動することなどがある。良いプログラム実行とは，ある MLM の action スロットに対して，リターン文のみ，あるいは呼び出し・write 文のみを入れることである。もし MLM が action スロットから呼出される（12.2.4節参照）か，あるいは外部のイベントから起動される（13 章参照）ならば，リターン文は action スロットの実行終了をもたらすのみである。

12.2 Action スロット文：

12.2.1 Write 文

Write 文は action スロットにおいて中核となる文である。この文はテキストやコード化されたメッセージ（例えば警告）を届け先まで送る。そしていくつかの形式がある。

```
WRITE <expr>;
WRITE <expr> AT <destination>;
WRITE <message>;
WRITE <message> AT <destination>;
```

<expr>はいかなる場合でも有効な表現である。医療従事者に読まれるテキストや logic スロットで定義された変数が，通常含まれる。

<destination>は 11.2.6 節で定義されている目的変数である。その届け先の形式と実装形は対象によって決められる。代表的な届け先としては診療録・プリンタ・データベース・e-mail アドレスなどがある。届け先が省略されたときは，メッセージはデフォルトの届け先に送られる。これは一般的には医療従事者か診療録である。しかし実行形は対象によって決められる。

<message>は 11.2.5 節で定義されている message 変数である。**<expr>**形式を持たないデータベースに対する機関固有にコード化された MLM メッセージを書くことを，この message 変数は機関に対して許している。

<expr>はしばしば文字列である。もしメッセージ構築にあたって Arden Syntax の特定の実装に XML を使う必要があれば，このメッセージを作り出すために文字列表現を使うことができる。付録 X1 で構造化メッセージのための推奨 DTD を示す。

write 文の効能は特定されたメッセージをデフォルトの届け先（通常は医療従事者あるいは患者診療録）あるいは特定された届け先に送ることである。

一つの MLM 内では，グループ化された write 文の効能は特定されず，syntax の実装によって決まる。

もし，ある MLM が他の MLM の action ブロックから呼出された場合には（12.2.4節参照），そこにある write 文は，呼出しをしている MLM とはセパレートされたグループとしての出力である。しかしながらグループ化の順序は特定されず，前述の syntax の実装に依存する。

12.2.1.1 <expr>の例

ここでの例において，**serum_pot** は logic スロットで指定された変数，**email_dest** はデータスロットで定義された目的変数，**a_message** はデータスロットで定義された message 変数である。

```
WRITE "the patient's potassium is" || serum_pot;
WRITE "this is an email alert" AT email_dest;
WRITE a_message;
```

12.2.1.2 <message>の例

一つの機関は message 変数を使わずにコード化されたメッセージをストアすることができる。例えば，下記のメッセージはフリーテキスト文字列ではなくユニークなコードとしてストアされる。そのコードは，変数である血清カリウム値を含む単一フィールドとともにメッセージを象徴化している。

```
WRITE "the patient's potassium is " || serum_pot;

WRITE CK0023 || serum_pot;
```

CK0023 は "**the patient's potassium is**"を表す機関固有のコードとなる。

このメッセージは write 文に使われる前に，機関固有コードとしてはっきりと指定されなければならない。一般的に，この指定はデータスロット内で行われる。

12.2.2 リターン文

リターン文は他の MLM から呼出される MLM において使われる。この文は呼出している MLM に結果を返す；その結果は call 文 (10.2.4 節参照)での変数に割り当てられる。この MLM は一つないしはそれ以上の結果を返すことができる。その用法は：

```
RETURN <expr>;
RETURN <expr>, ... , <expr>;
```

<expr>は常に有効な表現であり，単一項目であってもリストであってもよい。プラマリ日時は保持される。即ち，リターン文が実行されているときには，MLM 内での他の文が実行されることはない。

12.2.2.1 Examples: 例

```
RETURN (diagnosis_score,diagnosis_name);
RETURN diagnosis_score, diagnosis_name;
```

最初の例は一つの表現＝リストに戻っている。次の例は二つの表現に戻っている。

12.2.3 If-then Statement If-then 文

10.2.2 節で定義されている If-then 文もまた action スロットでの使用を許されている。

12.2.4 Call Statement Call 文

action スロットにある Call 文は，logic スロットでの結果に条件付で基づいて，ある MLM が他の MLM を呼出すことを許可する。これは 10.2.4 節で定義されている logic スロットにおける Call 文と同様である；被演算子は 11.2.4 節で定義されている被演算子文にアクセスされる。mlmname があれば，MLM は遅れは伴うが直接呼出せる。イベント定義があれば，そのイベントによって通常起動される MLM すべてが遅れは伴うが呼出せる。Call 文があるイベントを起動するために使われる場合は，すべての被演算子は無視される。その用法は：

```
CALL <name>;
CALL <name> DELAY <duration>;
CALL <name> WITH <expr>;
CALL <name> WITH <expr> DELAY <duration>;

CALL <name> WITH <expr>, ..., <expr>;
```


CALL <name> WITH <expr>, ..., <expr> DELAY <duration>;

<name> は識別子であり, データスロット内での MLM 文によって定義される有効 MLM 変数(11.2.3 節参照), あるいはデータスロット内でのイベント文によって定義される有効 イベント 変数(11.2.2 節参照)をあらわすものである。

<duration> は有効表現であり, その価は継続時間である。

12.2.4.1 Operation 演算

<name> が MLM 変数の場合, メインの MLM が終了したとき, 指定された MLM が呼出される。**<name>** がイベント変数の場合, すべての MLM は, そこにある evoke スロットが指定されたイベントを 参照すると実行される(13 章参照)。logic スロット内での Call 文が同期であれば, action スロット内での Call 文は非同期となる。呼出された MLM の実行順序は実装に依存する。

12.2.4.2 Example 例

(**mlmx** がデータスロットに適切な値で割り付けられてると, 例えば **mlmx** と**:= MLM 'my_mlm'**)

CALL mlmx DELAY 3 days ;

12.2.5 WHILE Loop WHILE ループ

10.2.5 節で定義されている **while** ループは action スロットにおいても許される。

12.2.6 FOR Loop FOR ループ

10.2.6 節で定義されている **for** ループは action スロットにおいても許される。

12.3 Action スロットの使用方法

action スロットは通常シンプル (単純) であり, 記述されるべき単一メッセージかあるいは呼び出している MLM に戻されるべき単一の値である。複合アクションはいくつかの action 文をリスト化することによって実行できる。スロットは, if-then 文で選択余地のあるアクションから選択することによって, より複雑にすることができる。このことは有効ではあるが, action スロット内の保健医療に関する logic の量は最小に留めることを薦める。

13 EVOKE スロット

13.1 目的

evoke スロットは MLM がどのように起動されるかを定義する。MLM は以下のイベントのいずれかで起動される。

13.1.1 イベントの発生

例えば, 患者の血清カリウム値が基準値からかけ離れているかどうかをチェックする際には, 患者データベースでのその値のストレージで発生する。

13.1.2 イベント後の時定数

例えば, じん機能をチェックには, 患者に対するゲンタマイシン投与後 5 日目とすること。

13.1.3 イベント後の周期

例えば, 一定期間後のじん機能チェックには, 患者に対するゲンタマイシン投与後 5 日毎とすること。

13.2 イベント

イベントはデータとは区別される。あるイベントは患者データベースでのあるひとつの更新かあるいはひとつの挿入かもしれないし, 医学的に重要な出来事, あるいは医療機関で決められている出来事かもしれない。例としては, 血清カリウム値のレベルのストレージや医療行為実施, 新しいベッドへの患者移送, そして患者の新住所の記録などがあげられる。

13.2.1 イベント属性

イベントの主要属性はその発生日時であり, 瞬時でなければならない。どんなイベントも値はもたない。イベントとデータの間には区別があることに注目しなければならない。データは値を持ちプライマリ日時を持つ。それは医学的に最も重要な日時である。例えば, 血清カリウム値は 5.0 という値を持つことがあり, そのプライマリ日時は患者から採取するときの日時である。しかし, **storage of serum potassium** イベントは値を持たない。そしてその日時とはそのカリウム値が患者データベースにストアされた日時である。

13.2.2 イベント日時

あるイベントに適用される演算子(9.17 節参照)の**日時**は, そのイベントが発生した日時となる。例えば, **storage_of_potassium の日時**とはそのカリウム値がストアされた日時である。この値は, 読み取りマッピングによって引き出された対応データ値の日時とは異なることもある（そのデータ値は診療上にかかわりの深い日時を代表的に使用する。この日時はデータを格納する日時とはしばしば異なるものである）。**Eventtime** は MLM を起動する日時を表す(8.4.4節参照)。

13.2.3 Declaration of Events イベント宣言

11.2.2 節で定義されているように, イベントはデータスロット内で宣言される。

13.3 Evoke スロット文

13.3.1 単純トリガー文

単純トリガー文はイベントあるいはイベントセットを特定する。いくつかのイベントのいずれか一つが発生すると, MLM が起動される。その用法は：

<event-expr>

<event-expr> は 11.2.2 節で定義されているイベント変数のみ含む表現である。その変数とは, **or** 演算子 (9.4.1 節参照), **any** 演算子 (9.12.13節参照)そして括弧 () である。キーワード **call** もまた存在するかもしれない。MLM が他の MLM から呼び出されることを示すために。

13.3.1.1 演算

イベントは値を持たないが, この文の中ではあたかもシンタックス的にはブール型のように使われる。一例としては文を以下のように終わらせる：**event1 OR event2 OR event3.** ある一つのイベントが発生すれば, MLM は起動され, evoke 文のいずれかが **true** となる。一つ以上のイベントが発生した場合, MLM は起動されるかもしれない。MLM を起動する要件を満たすための付加的な起動基準はない。

13.3.1.2 例

以下の例において, 全ての変数はデータスロット内で定義されるイベント変数である:

```
penicillin_storage
penicillin_storage OR cephalosporin_storage
ANY OF (penicillin_storage,cephalosporin_storage,aminoglycoside_storage)

data:
    penicillin_storage := event {store penicillin order}
    cephalosporin_storage := event {store cephalosporin order}
;;
evoke:
    penicillin_storage OR
    cephalosporin_storage;;
```

13.3.2 遅延トリガー文

遅延トリガー文は, あるイベント発生の一定時間後に MLM が起動されることを可能にする。その基本形を最初に示し, 次に基本形のより明確な例を示す:

```
<time-expr>
<duration-expr> AFTER TIME <event>
```

<time-expr> は時定数(7.1.5 節参照), あるいはイベント変数(11.2.2 節参照)に適用される **time** 演算子(9.17 節参照), 数値定数(7.1.4 節参照)に適用される継続時間演算子(9.10.4節参照), そして **after** 演算子(9.10.1節参照)を含む表現である。

<duration-expr> は経続時間演算子を伴う定数を使うことによって形成される経続時間定数である。

<event> はイベント変数である。

13.3.2.1 演算

MLM は遅延トリガー文内で特定されている日時に起動される。この起動は, 通常はイベント発生からある特定された継続時間後となることである。この文はある日時定数を使って特定の日付にのみ MLM を起動するために使用することができる。

13.3.2.2 例

以下の例において, 全ての変数はイベント変数である:

```
3 days after time of penicillin_storage
1992-01-01T00:00:00
```

13.3.3 周期トリガー文

周期トリガー文は, あるイベント発生後に MLM が指定された間隔で起動されることを可能にする。その周期的起動は指定された継続時間まで続き, ブール型条件によって終了する。二つの形式がある:

```
EVERY <duration-expr> FOR <duration-expr> STARTING <time-expr>
EVERY <duration-expr> FOR <duration-expr> STARTING <time-expr> UNTIL <Boolean-expr>
```

<duration-expr> は, 継続時間演算子(9.10.4節参照)をともなうある数値定数 (7.1.4 節参照)使って形成される継続時間定数である。

<time-expr> は, 日時定数(7.1.5 節参照)あるいはイベント変数(11.2.2 節参照)に適用される time 演算子(9.17 節参照)として表現される。

<Boolean-expr> は有効な表現である。通常ブール型表現をとり, MLM 起動を停止すべきときに **true** となる。

13.3.3.1 演算

MLM は単語 **starting** の後の指定日時に最初に起動される。単語 **every** の後の指定経続時間に等しい長さの周期で繰り返し起動される。この周期的起動は単語 **for** の後の指定された継続時間をとる。この **for** は両端を含む。従って **every 1 day for 1 day starting 3 days after time of event1** とあれば MLM は 2 度起動される。即ち, イベント後 3 日目と 4 日目である。

13.3.3.2 Until

もし **until** 節があれば, その節は MLM が起動され次第に評価される。即ち, その節はイベントに関連しない患者データベースに対する照会を含んでいるかもしれない。もし結果が **true** であれば MLM は直ちに退出し, それ以上に起動が発生することはない。**true** でなければ MLM は実行され, **every** (の後の) 経続時間後に再び起動される。

13.3.3.3 例

以下の例では, **イベント**にともなって始まる変数がイベント変数である:

```
every 1 day for 14 days starting 1992-01-01T00:00:00
every 1 day for 14 days starting time of event1
every 2 hours for 1 day starting 5 hours after the time of event2

every 1 week for 1 month starting 3 days after the time of event3 until last(serum_potassium) > 5.0
```

13.4 Evoke スロットの使用方法

起動スロットは通常 MLM をいつ起動するかを特定する一つの文を含んでいる。起動スロットに一つ以上の文がある場合, いずれかの文においてもトリガーが発生すれば MLM は起動される。

付録 A
(守らなければならないもの)

A1 BACKUS-NAUR 形式

MLM syntax を Backus-Naur 形式 (BNF)で定義する **(3)**。可読性と計算の容易性の観点から, 内容非依存の文法を, より簡潔な拡張 Backus-Naur 形式(EBNF)ではなく Backus-Naur 形式で表現する **(3)**。以下の定義が有効である:

<expression> - represents the non-terminal expression

"IF" - represents the terminal **if**, **iF**, **If**, or **IF**

":=" - represents the terminal **:=**

::= - is defined as

/*...*/ - a comment about the grammar

| - or

単語は大文字で表されるが, 言語は文字列の中を除いて, 大文字小文字を同一視する。構造化スロットの中では空白文字, 改行, 行送り, 水平タブ, 垂直タブ, 書式送りは全て余白と捉えられる。さらに, 単語 **the** も余白として捉えられる (すなわち, 単語 **the** は無視される)。

小さな変更を加えることで, 次の文法は LALR(1)パーサ生成方法で処理可能である (但し, 各ルールに対するコメントの部分を除く)。

```
/****** physical file containing one or more MLMs ******/

/****** file of individual MLMs ******/

<mlms> ::=
        <mlm>
      | <mlm> <mlms>

/****** categories ******/

<mlm> ::=
        <maintenance_category>
        <library_category>
        <knowledge_category>
        "END:"

<maintenance_category> ::=
        "MAINTENANCE:" <maintenance_body>
```

```
<maintenance_body> ::=
        <title_slot>
        <mlmname_slot>
        <arden_version_slot>
        <version_slot>
        <institution_slot>
        <author_slot>
        <specialist_slot>
        <date_slot>
        <validation_slot>

<library_category> ::=
        "LIBRARY:" <library_body>

<library_body> ::=
        <purpose_slot>
        <explanation_slot>
        <keywords_slot>
        <citations_slot>
        <links_slot>

<knowledge_category> ::=
        "KNOWLEDGE:" <knowledge_body>

<knowledge_body> ::=
        <type_slot>
        <data_slot>
        <priority_slot>
        <evoke_slot>
        <logic_slot>
        <action_slot>
        <urgency_slot>

/****** slots ******/

/****** maintenance slots ******/

<title_slot> ::=
        "TITLE:" <text> ";;"

< <mlmname_slot> ::=
        "MLMNAME:" <mlmname_text> ";;"
      | "FILENAME:" <mlmname_text> ";;"
                                        /* the "FILENAME:" form is only valid */
                                        /* combination with the empty version */
                                        /* of <arden_version_slot>          */

<mlmname_text> ::=
        <letter>
      | <mlmname_text><mlmname_text_rest>
```



```
<mlmname_text_rest> ::=
        <letter>
      | <digit>
      | "."
      | "-"
      | "_"

<arden_version_slot> ::=
        "ARDEN:" <arden_version> ";;"
      | /*empty*/
                                          /* the empty version is only valid    */
                                          /* combination with the "FILENAME"    */
                                          /* form of < mlmname_slot >           */

<arden_version> ::=
        "VERSION" "2"
      | "VERSION" "2.1"

<version_slot> ::=
        "VERSION:" <mlm_version> ";;"

<mlm_version> ::=
        <text>

<institution_slot> ::=
        "INSTITUTION:" <text> ";;"          /* text limited to 80 characters */

<author_slot> ::=
        "AUTHOR:" <text> ";;"               /* see 6.1.6 for details */

<specialist_slot> ::=
        "SPECIALIST:" <text> ";;"           /* see 6.1.7 for details */

<date_slot> ::=
        "DATE:" <mlm_date> ";;"

<mlm_date> ::=
        <iso_date>
      | <iso_date_time>

<validation_slot> ::=
        "VALIDATION:" <validation_code> ";;"

    <validation_code> ::=
          "PRODUCTION"
        | "RESEARCH"
        | "TESTING"
        | "EXPIRED"
```

/****** library slots ******/

```
<purpose_slot> ::=
        "PURPOSE:" <text> ";;"

<explanation_slot> ::=
        "EXPLANATION:" <text> ";;"
```

```
<keywords_slot> ::=
        "KEYWORDS:" <text> ";;"
```

/* May require special processing to handle both list and text versions */

```
<citations_slot> ::=
        /* empty */
      | "CITATIONS:" <citations_list> ";;"
      | "CITATIONS:" <text> ";;"        /* deprecated -- supported for backward compatibility */

<citations_list> ::=
        /* empty */
      | <single_citation>
      | <single_citation> ";" <citations_list>

<single_citation> ::=
        <digits> "." <citation_type> <citation_text>
      | <citation_text>
```

/* This is a separate definition to allow for future expansion */

```
<citation_text> ::=
        <string>

<citation_type> ::=
          /* empty */
        | "SUPPORT"
        | "REFUTE"
```

/* May require special processing to handle both list and text versions */

```
<links_slot> ::=
        /* empty */
      | "LINKS:" <links_list> ";;"
      | "LINKS:" <text> ";;"        /* deprecated - supported for backward*/
                                            /* compatibility */

<links_list> ::=
        /* empty */
      | <single link>
      | <links_list> ";" <single link>

<single_link> ::=
        <link_type> <link_name> <link_text>

<link_type> ::=
        /* empty */
      | "URL_LINK"
      | "MESH_LINK"
      | "OTHER_LINK"
      | "EXE_LINK"

<link_name> ::=
    /* empty */
    | <string>
```


/* This is a separate definition to allow for future expansion */

```
<link_text> ::=
        <term>
```

/****** knowledge slots ******/

```
<type_slot> ::=
            "TYPE:" <type_code> ";;"
```

/* This is a separate definition to allow for future expansion */

```
<type_code> ::=
             "DATA_DRIVEN"
            | "DATA-DRIVEN"   /* deprecated -- supported for backwards */
                                          /* compatibility */

<data_slot> ::=
      "DATA:" <data_block> ";;"

<priority_slot> ::=
             /* empty */
            | "PRIORITY:" <number> ";;"

<evoke_slot> ::=
             "EVOKE:" <evoke_block> ";;"

<logic_slot> ::=
             "LOGIC:" <logic_block> ";;"

<action_slot> ::=
             "ACTION:" <action_block> ";;"

<urgency_slot> ::=
             /* empty */
            | "URGENCY:" <urgency_val> ";;"

<urgency_val> ::=
             <number>
            | <identifier>
```

/****** logic block ******/

```
<logic_block> ::=
             <logic_block> ';' <logic_statement>
            | <logic_statement>

<logic_statement> ::=
             /* empty */
            | <logic_assignment>
            | "IF" <logic_if_then_else2>
            | "FOR" <identifier> "IN" <expr> "DO" <logic_block> ";" "ENDDO"
            | "WHILE" <expr> "DO" <logic_block> ";" "ENDDO"
            | "CONCLUDE" <expr>
```

```
<logic_if_then_else2> ::=
        <expr> "THEN" <logic_block> ";" <logic_elseif> ";"

<logic_elseif> ::=
         "ENDIF"
       | "ELSE" <logic_block> ";" "ENDIF"
       | "ELSEIF" <logic_if_then_else2>

<logic_assignment> ::=
         <identifier_becomes> <expr>
       | <time_becomes> <expr>
       | <identifier_becomes> <call_phrase>

<identifier_becomes> ::=
         <identifier> ":="
       | "LET" <identifier> "BE"
       | "NOW" ":="

<time_becomes> ::=
         "TIME" "OF" <identifier> ":="
       | "TIME" <identifier> ":="
       | "LET" "TIME" "OF" <identifier> "BE"
       | "LET" "TIME" <identifier> "BE"

<call_phrase> ::=
         "CALL" <identifier>
       | "CALL" <identifier> "WITH" <expr>
```

/****** expressions ******/

```
<expr> ::=
         <expr_sort>
       | <expr> "," <expr_sort>
       | "," <expr_sort>

<expr_sort> ::=
         <expr_where>
       | <expr_where> "MERGE" <expr_sort>
       | "SORT"  <expr_sort>
       | "SORT"  <sort_option>  <expr_sort>

<sort_option> ::=
         "TIME"
       | "DATA"

<expr_where> ::=
         <expr_range>
       | <expr_range> "WHERE" <expr_range>

<expr_range> ::=
         <expr_or>
       | <expr_or> "SEQTO" <expr_or>
```



```
<expr_or> ::=
            <expr_or> “OR” <expr_and>
          | <expr_and>

<expr_and> ::=
            <expr_and> “AND” <expr_not>
          |  <expr_not>

<expr_not> ::=
            “NOT” <expr_comparison>
          | <expr_comparison>

<expr_comparison> ::=
            <expr_string>
          | <expr_find_string>
          | <expr_string> <simple_comp_op> <expr_string>
          | <expr_string> <is> <main_comp_op>
          | <expr_string> <is> "NOT" <main_comp_op>
          | <expr string> <in_comp_op>
          | <expr_string> "NOT" <in_comp_op>
          | <expr_string> <occur> <temporal_comp_op>
          | <expr_string> <occur> "NOT" <temporal_comp_op>
          | <expr_string> <occur> <range_comp_operator>
          | <expr_string> <occur> "NOT" <range_comp_operator>
          | <expr_string> "MATCHES" "PATTERN" <expr_string>

<exp_find_string> ::=
            "FIND" <expr_string> "IN" "STRING" <expr_string> <string_search_start>
          | "FIND" <expr_string> "STRING" <expr_string> <string_search_start>

<expr_string> ::=
            <expr_plus>
          | <expr_string> "||" <expr_plus>
          | <expr_string> "FORMATTED" "WITH" <format_string>
          | "TRIM" <trim_option> <expr_string>
          | "SUBSTRING" <expr_plus> "CHARACTERS" <string_search_start> "FROM" <expr_string>

<format_string> ::=
          """ <format_specification> """    /* The format string is a true    */
                                            /* Arden Syntax string, enclosed  */
                                            /* in a single pair of double     */
                                            /* quotes (" )                    */

<format_specification> ::=/*  See section 9.8.2 and Annex 5 for       */
                          /* explanation of valid combination and their */
                          /* meanings.                                  */
            <format_specificaton> <format_specification_single>
          | <format_specification_single>

<format_specification_single>
            "%"<format_options><format_flag><width><precision>
          /*  No spaces  are permitted between elements in above form */
          | <text>
```

```
<format_options> ::=
        /* empty */
        | "+"
        | "-"
        | "0"
        | " "  /* space */
        | "#"

<format_flag> ::=          /* Format flags are case sensitive */
          "c"
        | "C"
        | "d"
        | "I"
        | "o"
        | "u"
        | "x"
        | "X"
        | "e"
        | "E"
        | "f"
        | "g"
        | "G"
        | "n"
        | "p"
        | "s"
        | "t"

<width> ::=
          /* empty */
        | <digits>

<precision> ::=
          /* empty */
        | "."<digits>

<trim_option> ::=
          /* empty */
        | "LEFT"
        | "RIGHT"

<string_search_start> ::=
         /* empty */
        | "STARTING" "AT" <expr_plus>

<expr_plus> ::=
          <expr_times>
        | <expr_plus> "+" <expr_times>
        | <expr_plus> "-" <expr_times>
        | "+" <expr_times>
        | "-" <expr_times>
```



```
<expr_times> ::=
            <expr_power>
          | <expr_times> "*" <expr_power>
          | <expr_times> "/" <expr_power>

<expr_power> ::=
            <expr_before>
          | <expr_function> "**" <expr_function>
                        /* exponent (second argument) must be an expression    */
                        /* that evaluates to a scalar number                   */

<expr_before> ::=
            <expr_ago>
          | <expr_duration> "BEFORE" <expr_ago>
          | <expr_duration> "AFTER" <expr_ago>
          | <expr_duration> "FROM" <expr_ago>

<expr_ago> ::=
            <expr_function>
          | <expr_duration>
          | <expr_duration> "AGO"

<expr_duration> ::=
            <expr_function> <duration_op>

<expr_function> ::=
            <expr_factor>
          | <of_func_op> <expr_function>
          | <of_func_op> "OF" <expr_function>
          | <from_of_func_op> <expr_function>
          | <from_of_func_op> "OF" <expr_function>
          | <from_of_func_op> <expr_factor> "FROM" <expr_function>
          | <from_func_op> <expr_factor> "FROM" <expr_function>
          | <index_from_of_func_op> <expr_function>
          | <index_from_of_func_op> "OF" <expr_function>
          | <index_from_of_func_op> <expr_factor> "FROM" <expr_function>
          | <index_from_func_op> <expr_factor> "FROM" <expr_function>
          | <expr_factor> "AS" <as_func_op>

<expr_factor> ::=
            <expr_factor_atom>
          | <expr_factor_atom> "[" <expr> "]"          /* number [<expr>] is not */
                                                       /* a valid construct      */
```

```
<expr_factor_atom> ::=
        <identifier>
      | <number>
      | <string>
      | <time_value>
      | <boolean_value>
      | "NULL"
      | <it>                          /* Value of <it> is NULL outside of a     */
                                      /* where clause and may be flagged as an  */
                                      /* error in some implementations.         */     | "(" ")"
      | "(" <expr> ")"
```

/****** for readability *******/

```
<it> ::= "IT" | "THEY"
```

/****** comparison synonyms ******/

```
<is> ::= "IS" | "ARE" | "WAS" | "WERE"

<occur> ::= "OCCUR" | "OCCURS" | "OCCURRED"
```

/****** operators ******/

```
<simple_comp_op> ::=
        "=" | "EQ"
      | "<" | "LT"
      | ">" | "GT"
      | "<="      | "LE"
      | ">="      | "GE"
      | "<>"      | "NE"

<main_comp_op> ::=
        <temporal_comp_op>
      | <range_comp_op>
      | <unary_comp_op>
      | <binary_comp_op> <expr_string>

/* the WITHIN TO operator will accept any ordered parameter, */
/* including numbers, strings (single characters), times, Boolean /*

<range_comp_op> ::=
        "WITHIN" <expr_string> "TO" <expr_string>
```



```
<temporal_comp_op> ::=
          "WITHIN" <expr_string> "PRECEDING" <expr_string>
        | "WITHIN" <expr_string> "FOLLOWING" <expr_string>
        | "WITHIN" <expr_string> "SURROUNDING" <expr_string>
        | "WITHIN" "PAST" <expr_string>
        | "WITHIN" "SAME" "DAY" "AS" <expr_string>
        | "BEFORE" <expr_string>
        | "AFTER" <expr_string>
        | "EQUAL" <expr_string>
        | "AT" <expr_string>

<unary_comp_op> ::=
          "PRESENT"
        | "NULL"
        | "BOOLEAN"
        | "NUMBER"
        | "TIME"
        | "DURATION"
        | "STRING"
        | "LIST"

<binary_comp_op> ::=
          "LESS" "THAN"
        | "GREATER" "THAN"
        | "GREATER" "THAN" "OR" "EQUAL"
        | "LESS" "THAN" "OR" "EQUAL"
        | "IN"

<of_func_op> ::=
              <of_read_func_op>
            | <of_noread_func_op>

< in_comp_op> ::=
          "IN"

<of_read_func_op> ::=
          "AVERAGE"  | "AVG"
        | "COUNT"
        | "EXIST"  | "EXISTS"
        | "SUM"
        | "MEDIAN"
```

```
<of_noread_func_op> ::=
          "ANY"
        | "ALL"
        | "NO"
        | "SLOPE"
        | "STDDEV"
        | "VARIANCE"
        | "INCREASE"
        | "PERCENT" "INCREASE" | "%" "INCREASE"
        | "DECREASE"
        | "PERCENT" "DECREASE" | "%" "DECREASE"
        | "INTERVAL"
        | "TIME"
        | "ARCCOS"
        | "ARCSIN"
        | "ARCTAN"
        | "COSINE"      | "COS"
        | "SINE"    | "SIN"
        | "TANGENT"   | "TAN"
        | "EXP"
        | "FLOOR"
        | "INT"
        | "ROUND"
        | "CEILING"
        | "TRUNCATE"
        | "LOG"
        | "LOG10"
        | "ABS"
        | "SQRT"
        | "EXTRACT" "YEAR"
        | "EXTRACT" "MONTH"
        | "EXTRACT" "DAY"
        | "EXTRACT" "HOUR"
        | "EXTRACT" "MINUTE"
        | "EXTRACT" "SECOND"
        | "STRING"
        | "EXTRACT" "CHARACTERS"
        | "REVERSE"
        | "LENGTH"
        | "UPPERCASE"
        | "LOWERCASE"

<from_func_op> ::=
          "NEAREST"

<index_from_func_op> ::=
          "INDEX" "NEAREST"
```



```
<from_of_func_op> ::=
        "MINIMUM"   | "MIN"
      | "MAXIMUM" | "MAX"
      | "LAST"
      | "FIRST"
      | "EARLIEST"
      | "LATEST"

<index_from_of_func_op> ::=
        "INDEX" "MINIMUM"    | "INDEX" "MIN"
      | "INDEX" "MAXIMUM"   | "INDEX" "MAX"
      | "INDEX" "EARLIEST"
      | "INDEX" "LATEST"

<as_func_op> ::=
        "NUMBER"

<duration_op> ::=
        "YEAR"  | "YEARS"
      | "MONTH"      | "MONTHS"
      | "WEEK" | "WEEKS"
      | "DAY"          | "DAYS"
      | "HOUR"  | "HOURS"
      | "MINUTE"     | "MINUTES"
      | "SECOND"     | "SECONDS"
```

/****** factors ******/

```
<boolean_value> ::=
        "TRUE"
      | "FALSE"

<time_value> ::=
        "NOW"
      | <iso_date_time>
      | <iso_date>
      | "EVENTTIME"
      | "TRIGGERTIME"
      | "CURRENTTIME"
```

/****** data block ******/

```
<data_block> ::=
        <data_block> ”;” <data_statement>
      | <data_statement>

<data_statement> ::=
        /* empty */
      | <data_assignment>
      | "IF" <data_if_then_else2>
      | "FOR" <identifier> "IN" <expr> "DO" <data_block> ";" "ENDDO"
      | "WHILE" <expr> "DO" <data_block> ";" "ENDDO"
```

```
<data_if_then_else2> ::=
        <expr> "THEN" <data_block> ";" <data_elseif>

<data_elseif> ::=
         "ENDIF"
        | "ELSE" <data_block> ";" "ENDIF"
        | "ELSEIF" <data_if_then_else2>

<data_assignment> ::=
         <identifier_becomes> <data_assign_phrase>
        | <time_becomes> <expr>
        | "(" <data_var_list> ")" ":=" "READ" <read_phrase>
        | "LET" "(" <data_var_list> ")" "BE" "READ" <read_phrase>
        | "(" <data_var_list> ")" ":=" "ARGUMENT"
        | "LET" "(" <data_var_list> ")" "BE" "ARGUMENT"

<data_var_list> ::=
         <identifier>
        | <identifier> "," <data_var_list>

<data_assign_phrase> ::=    "READ" <read_phrase>
        | "MLM" <term>
        | "MLM" <term> "FROM" "INSTITUTION" <string>
        | "MLM" "MLM_SELF"
        | "INTERFACE" <mapping_factor>
        | "EVENT" <mapping_factor>
        | "MESSAGE" <mapping_factor>
        | "DESTINATION" <mapping_factor>
        | "ARGUMENT"
        | <call_phrase>
        | <expr>

<read_phrase> ::=
         <read_where>
        | <of_read_func_op> <read_where>
        | <of_read_func_op> "OF" <read_where>
        | <from_of_read_func_op> <read_where>
        | <from_of_read_func_op> "OF" <read_where>
        | <from_of_read_func_op> <expr_factor> "FROM" <read_where>

<read_where> ::=
         <mapping_factor>
        | <mapping_factor> "WHERE" <it> <occur> <temporal_comp_op>
        | <mapping_factor> "WHERE" <it> <occur> "NOT" <temporal_comp_op>
        | <mapping_factor> "WHERE" <it> <occur> range_comp_op
        | <mapping_factor> "WHERE" <it> <occur> "NOT" range_comp_op
        | "(" <read_where> ")"

<mapping_factor> ::=
         "{" <data_mapping> "}"
```


/****** evoke block ******/

```
<evoke_block> ::=
        <evoke_statement>
      | <evoke_block> ";" <evoke_statement>

<evoke_statement> ::=
        /* empty */
      | <event_or>
      | <evoke_time>
      | <qualified_evoke_cycle>
      | "CALL"          /* deprecated -- kept for backward compatibility */

<event_list> ::=
        <event_or>
      | <event_list> "," <event_or>

<event_or> ::=
        <event_or> "OR" <event_any>
      | <event_any>

<event_any> ::=
        "ANY" "(" <event_list> ")"
      | "ANY" "OF" "(" <event_list> ")"
      | "ANY" <identifier>
      | "ANY" "OF" <identifier>
      | <event_factor>

<event_factor> ::=
        "(" <event_or> ")"
      | <identifier>

<evoke_time> ::=
        <evoke_duration> "AFTER" <evoke_time>
      | "TIME" <event_any>
      | "TIME" "OF" <event_any>
      | <iso_date_time>
      | <iso_date>

<qualified_evoke_cycle> ::=
        <simple_evoke_cycle>
      | <simple_evoke_cycle> "UNTIL" <expr>

<simple_evoke_cycle> ::=
        "EVERY" <evoke_duration> "FOR" <evoke_duration> "STARTING" <evoke_time>

<evoke_duration> ::=
        <number> <duration_op>
```

/****** action block ******/

```
<action_block> ::=
        <action_statement>
      | <action_block> ";" <action_statement>
```

```
<action_statement> ::=
        /* empty */
      | "IF" <action_if_then_else2>
      | "FOR" <identifier> "IN" <expr> "DO" <action_block> ";" "ENDDO"
      | "WHILE" <expr> "DO" <action_block> ";" "ENDDO"
      | <call_phrase>
      | <call_phrase> "DELAY" <expr>
      | "WRITE" <expr>
      | "WRITE" <expr> "AT" <identifier>
      | "RETURN" <expr>

<action_if_then_else2> ::=
      <expr> "THEN" <action_block> ";" <action_elseif>

<action_elseif> ::=
        "ENDIF"
      | "ELSE" <action_block> ";" "ENDIF"
      | "ELSEIF" <action_if_then_else2>
```

/****** lexical constructs ******/

```
/*  Unless otherwise specificed, characters are the printable ASCII       */
/*  characters (ASCII 33 through and including 126),  ( See 5.2 )         */

/*  The space, carriage return, line feed, horizontal tab, vertical tab,   */
/*  and form feed are collectively referred to as white space.            */
/*  See also Section 7.1.10.                                              */

<string> ::=
        /* any string of characters enclosed in double quotes (" ASCII 22) */
        /* with nested "" but without ";;"                                 */
        /*  (character set limitations do not apply here)                  */

<identifier> ::=
      /* up to 80 characters total (no reserved words allowed) */
        <letter> <identifier_rest>

<identifier_rest> ::=          /* no spaces are permitted between elements */
        /* empty */
      | <letter> <identifier>
      | <digit> <identifier>
      | "_" <identifier>

<text> ::=
        /* any string of characters without ";;" */

<format_text> ::=
        /* any string of characters */

<number> ::=                              /* no spaces are permitted between elements */
        <digits> <exponent>
      | <digits> "." <exponent>
      | <digits> "." <digits> <exponent>
      | "." <digits> <exponent>
```



```
<exponent> ::=                          /* no spaces are permitted between elements */
              /* null */
             | <e> <sign> <digits>

<e> ::=
              "E"
             | "e"

<sign> ::=
              /* null */
             | "+"
             | "-"

<digits> ::=                            /* no spaces are permitted between elements */
              <digit>
             | <digit> <digits>

<digit> ::=
              "0"
             | "1"
             | "2"
             | "3"
             | "4"
             | "5"
             | "6"
             | "7"
             | "8"
             | "9"

<letter> ::=
               "a"  | "b"  | "c"  | "d"
             | "e"  | "f"  | "g"  | "h"
             | "i"  | "j"  | "k"  | "l"
             | "m" | "n"  | "o"  | "p"
             | "q"  | "r"  | "s"  | "t"
             | "u"  | "v"  | "w" | "x"
             | "y"  | "z"

           | "A"    | "B" | "C" | "D"
             | "E"  | "F"  | "G" | "H"
             | "I"  | "J"  | "K" | "L"
             | "M"| "N" | "O" | "P"
             | "Q" | "R" | "S" | "T"
             | "U" | "V" | "W"       | "X"
             | "Y" | "Z"

<iso_date> ::=                          /* no spaces are permitted between elements */
             <digit> <digit> <digit> <digit> "-" <digit> <digit> "-" <digit> <digit>

<iso_date_time> ::=                     /* no spaces are permitted between elements */
             <digit> <digit> <digit> <digit> "-" <digit> <digit> "-" <digit> <digit> <t>
             <digit> <digit> ":" <digit> <digit> ":" <digit> <digit>
             <fractional_seconds>
             <time_zone>
```

```
<t> ::=
          "T"
        | "t"

<fractional_seconds> ::=        /* no spaces are permitted between elements */
          "." <digits>
        | /* empty */

<time_zone> ::=                 /* no spaces are permitted between elements */
          /* null */
        | <zulu>
        | "+" <digit> <digit> ":" <digit> <digit>
        | "-" <digit> <digit> ":" <digit> <digit>

<zulu> ::=
          "Z"
        | "z"

<term> ::=
        /* any string of characters enclosed in single quotes (' , ASCII 44) without ";;" */

<data_mapping> ::=
        /* any balanced string of characters enclosed in curly brackets { } */
        /* (ASCII 123 and 125, respectively) without ";;" the data mapping  */
        /* does not include the curly bracket characters                    */

<multi_line_comment> ::=
        /* any string of characters enclosed between pairs of "/*" and"*/"  */
        /* (character set limitations do not apply here)                    */

<single_line_comment> ::=
        /* any string of characters located between "//" and                */
        /* an end-of-line markner (CR, LF, or CR/LF pair)                   */
        /* (character set limitations do not apply here)                    */
```


A2. 予約語

ここにアルファベット順に列挙された単語は全て予約語である。これらを変数名に使用してはならない。

abs
action
after
ago
alert
all
and
any
arccos
arcsin
arctan
arden
are
argument
as
at
author
average
avg
be
before
Boolean
call
ceiling
characters
citations
conclude
cos
cosine
count
currenttime
data
data_driven
data-driven
date
day
days
decrease
delay
destination
do
duration
earliest
else
elseif
enddo
endif
end
eq
equal
event
eventtime
every
evoke
exist
exists
exp
expired
explanation
extract
false
filename
find
first
floor
following
for
formatted
from
ge
greater
gt
hour
hours
if
in
increase
index
institution
int
interface
interval
is
it
keywords
knowledge
last
latest
le
left
length
less
let
library
links
list
log
log10
logic
lowercase
lt
maintenance
matches
max
maximum
median
merge
message
min
minimum
minute
minutes
mlm
mlmname
mlm_self
month
months
ne
nearest
no
not
now
null
number
occur
occurred
occurs
of
or
past
pattern
percent
preceding
present
priority
production
purpose
read
refute
research
return
reverse
right
round
same
second
seconds
seqto
sin
sine
slope
sort
specialist
sqrt
starting
stddev

string
substring
sum
support
surrounding
tan
tangent
testing
than
the
then
they
time
title
to
triggertime
trim
true
truncate
type
unique
until
uppercase
urgency
validation
variance
version
was
week
weeks
were
where
while
with
within
write
year
years

以下の識別子は将来使用するために予約されている:

union	intersect	excluding	citation	select

A3. 特殊記号

ここに列挙したものは全て特殊記号である。

||	:=	,	=	>=
>	<=	<	{	(
[	-	<>	%	+
}	)	]	;	#
/	*	**	;;	:
/*	*/	//	'	"

A4.演算子の優先順位と結合性

A4.1

ここでは構造化スロットに使用される演算子は優先順位ごとにグループ化して示されている。グループは, 水平線で区切られている。グループの中では各演算子の優先順位は等しい。グループは最低から最高の優先順位で並べられている。

A4.2

同義語は同一のラインに並べられ, º で区切られている。記号[of]は単語 **of** がオプションで演算子の論理には影響を与えないことを意味している。記号[in]は単語 **in** がオプションで演算子の論理には影響を与えないことを意味している。

A4.3

演算子に対する被演算子の位置は省略…で示されている。演算子の結合性は各演算子のあとに斜文字で表されている。演算子には単項演算子 (一つの被演算子) と二項演算子 (二つの被演算子) の両方を持つものがある。この場合, それぞれの形式は別々に表示されている。

... , ... (left associative)
... merge ... (left associative)
sort ... (non-associative)

... where ... (non-associative)

... or ... (left associative)

... and ... (left associative) not ... (non-associative)

... = ... º ... eq ... º ... is equal ... (non-associative)
... <> ... º ... ne ... º ... is not equal ... (non-associative)
... < ... º ... lt ... º ... is less than ... º ... is not greater than or equal ... (non-associative)
... <= ... º ... le ... º ... is less than or equal ... º ... is not greater than ... (non-associative)
... > ... º ... gt ... º ... is greater than ... º ... is not less than or equal ... (non-associative)
... >= ... º ... ge ... º ... is greater than or equal ... º ... is not less than ... (non-associative)
... is within ... to ... (non-associative)
... is not within ... to ... (non-associative)
... is within ... preceding ... (non-associative)
... is not within ... preceding ... (non-associative)
... is within ... following ... (non-associative) ... is not within ... following ... (non-associative)
... is within ... surrounding ... (non-associative)
... is not within ... surrounding ... (non-associative)
... is within past ... (non-associative)
... is not within past ... (non-associative)
... is within same day as ... (non-associative)
... is not within same day as ... (non-associative)
... is before ... (non-associative)
... is not before ... (non-associative)
... is after ... (non-associative)
... is not after ... (non-associative)
... occur equal ... º … occur at … (non-associative)
... occur within ... to ... (non-associative)
... occur not within ... to ... (non-associative)
... occur within ... preceding ... (non-associative)
... occur not within ... preceding ... (non-associative)
... occur within ... following ... (non-associative)
... occur not within ... following ... (non-associative)

... occur within ... surrounding ... (non-associative)
... occur not within ... surrounding ... (non-associative)
... occur within past ... (non-associative)
... occur not within past ... (non-associative)
... occur within same day as ... (non-associative)
... occur not within same day as ... (non-associative)
... occur before ... (non-associative)
... occur not before ... (non-associative)
... occur after ... (non-associative)
... occur not after ... (non-associative)
... is in ... º … in … (non-associative)
... is not in ... º … not in … (non-associative)
... is present º ... is not null (non-associative)
... is not present º ... is null (non-associative)
... is Boolean (non-associative)
... is not Boolean (non-associative)
... is number (non-associative)
... is not number (non-associative)
... is time (non-associative)
... is not time (non-associative)
... is duration (non-associative)
... is not duration (non-associative)
... is string (non-associative)
... is not string (non-associative)
... is list (non-associative)
... is not list (non-associative)

... || ... (left associative)
... formatted with ... (non- associative)
uppercase … (right associative)
lowercase … (right associative)
trim … (right associative)
trim left … (right associative)
trim right … (right associative)
substring … characters from … (right associative)
substring … characters from … starting at … (right associative)

+ ... (non-associative)
... + ... (left associative)
- ... (non-associative)
... - ... (left associative)

... * ... (left associative)
... / ... (left associative)

... ** ... (non-associative)

... round ... (non-associative)

... before ... (non-associative)
... after ... º … from … (non-associative)

... ago (non-associative)

... year º ... years (non-associative)
... month º ... months (non-associative)
... week º ... weeks (non-associative)
... day º ... days (non-associative)

... hour º ... hours (non-associative)
... minute º ... minutes (non-associative)
... second º ... seconds (non-associative)
…matches pattern … (non-associative)
find … [in] … (right associative)
find … [in] … starting at … (right associative)

---

count [of] ... (right associative)
exist [of] ... (right associative)
avg [of] ... º average [of] ... (right associative)
median [of] ... (right associative)
sum [of] ... (right associative)
stddev [of] ... (right associative)
variance [of] ... (right associative)
any [of] ... (right associative)
all [of] ... (right associative)
no [of] ... (right associative)
slope [of] ... (right associative)
min ... from º minimum ... from ... (right associative)
min [of] ... º minimum [of] ... (right associative)
max ... from ... º maximum ... from ... (right associative)
max [of] ... º maximum [of] ... (right associative)
index min ... from º index minimum ... from ... (right associative)
index min [of] ... º index minimum [of] ... (right associative)
index max ... from ... º index maximum ... from ... (right associative)
index max [of] ... º index maximum [of] ... (right associative)
last ... from ... (right associative)
last [of] ... (right associative)
first ... from ... (right associative)
first [of] ... (right associative)
latest ... from ... (right associative)
latest [of] ... (right associative)
earliest ... from ... (right associative)
earliest [of] ... (right associative)
nearest ... from ... (right associative)
index nearest ... from ... (right associative)
increase [of] ... (right associative)
decrease [of] ... (right associative)
percent increase [of] ... º % increase [of] ... (right associative)
percent decrease [of] ... º % decrease [of] ... (right associative)
interval [of] ... (right associative)
time [of] ... (right associative)
arccos [of] ... (right associative)
arcsin [of] ... (right associative)
arctan [of] ... (right associative)
cos [of] ... º cosine [of] ... (right associative)
sin [of] ... º sine [of] ... (right associative)
tan [of] ... º tangent [of] ... (right associative)
exp [of] ... (right associative)
floor [of] ... (right associative)
ceiling [of] ... (right associative)
truncate [of] ... (right associative)
log [of] ... (right associative)
log10 [of] ... (right associative)
abs [of] ... (right associative)
sqrt [of] ... (right associative)
extract year [of] ... (right associative)
extract month [of] ... (right associative)
extract hour [of] ... (right associative)

extract minute [of] ... (right associative)
extract second [of] ... (right associative)
reverse [of] ... (right associative)
extract characters [of] ... (right associate)
string [of] ... (right associative)
length [of] … (right associative)

A5. 書式仕様(9.8.2)

A5.1 以下は書式仕様でサポートされている型に関する全ての表記である。

type Required character that determines whether the associated argument is interpreted as a character, a string, or a number.

Table A5-1

Character	Type	Output Format
c	number	The number is assumed to represent a character code to be output as a character.
C	number	The number is assumed to represent a character code to be output as a character.
d	number	Signed decimal integer.
i	number	Signed decimal integer.
o	number	Unsigned octal integer.
u	number	Unsigned decimal integer.
x	number	Unsigned hexadecimal integer, using "abcdef."
X	number	Unsigned hexadecimal integer, using "ABCDEF."
e	number	Signed value having the form [–]d.dddd e [sign]ddd where d is a single decimal digit, dddd is one or more decimal digits, ddd is exactly three decimal digits, and sign is + or – .
E	number	Identical to the e format, except that E, rather than e, introduces the exponent.
f	double	Signed value having the form [–]dddd.dddd, where dddd is one or more decimal digits. The number of digits before the decimal point depends on the magnitude of the number, and the number of digits after the decimal point depends on the requested precision.
g	double	Signed value printed in f or e format, whichever is more compact for the given value and precision. The e format is used only when the exponent of the value is less than –4 or greater than or equal to the precision argument. Trailing zeros are truncated, and the decimal point appears only if one or more digits follow it.
G	double	Identical to the g format, except that E, rather than e, introduces the exponent (where appropriate).
n	Not supported.	Not supported.
p	Not supported.	Not supported.
s	string	Specifies a character. Characters are printed until the precision value is reached.
t	time	A time is printed based on the user's environment settings and the precision value.

A5.2 オプションフィールド(タイプ文字の前に位置する)は, 次のように書式の他の部分を制御する:

flags Optional character or characters that control justification of output and printing of signs, blanks, decimal points, and octal and hexadecimal prefixes. More than one flag can appear in a format specification.

Table A5-2

Flag	Meaning	Default
–	Left align the result within the given field width.	Right align.
+	Prefix the output value with a sign (+ or –) if the output value is of a signed type.	Sign appears only for negative signed values (–).
0	If width is prefixed with 0, zeros are added until the minimum width is reached. If 0 and – appear, the 0 is ignored. If 0 is specified with an integer format (I, u, x, X, o, d) the 0 is ignored.	No padding.
Space	Prefix the output value with a space if the output value is signed and positive; the space is ignored if both the space and + flags appear.	No space appears.
#	When used with the o, x, or X format, the # flag prefixes any nonzero output value with 0, 0x, or 0X, respectively.	No blank appears.
#	When used with the e, E, or f format, the # flag forces the output value to contain a decimal point in all cases.	Decimal point appears only if digits follow it.
	When used with the g or G format, the # flag forces the output value to contain a decimal point in all cases and prevents the truncation of trailing zeros.	Decimal point appears only if digits follow it. Trailing zeros are truncated.
#	Ignored when used with c, d, i, u, or s.	

二番目のオプションフィールドは width 仕様である。Width 部分は負でなく小数点付整数で印字される最小の桁数を制御している。仕様で定義された幅よりもアウトプットする文字数が小さい場合, 空白が最小の幅になるまで左または右に埋められる。左右は flag によって決められる。width が 0 の接頭子を有している場合, ゼロが最小幅になるまで埋められる（ただし, 左詰の場合には有効ではない）。

Width 仕様は値の打ち切りは生じさせない。アウトプットする文字数が定義された幅よりも大きい場合, もしくは幅が与えられていない場合, 全ての文字が印字される。

Width がアスタリスク（*）の場合, 適当な整数値が与えられる。Width 被演算子は被演算子リストの中で書式付けされる前に存在しなければならない。存在しないもしくは小さな幅の場合でも, フィールドは打ち切られない。フィールド幅よりも変換された結果が大きい場合には, 結果を含むように幅が拡大される。

width　印字する最小の文字数を表すオプションの数値

precision　全てもしくは一部の印字フィールドの最大文字数, もしくは整数を表す最小の文字数を表すオプションの数値

Table A5-3

Type	Meaning	Default
c, C	The precision has no effect.	Character is printed.
d, i, u, o, x, X	The precision specifies the minimum number of digits to be printed. If the number of digits in the argument is less than precision, the output value is padded on the left with zeros. The value is not truncated when the number of digits exceeds precision.	Default precision is 1.
e, E	The precision specifies the number of digits to be printed after the decimal point. The last printed digit is rounded.	Default precision is 6; if precision is 0, or the period (.) appears without a number following it, no decimal point is printed.
f	The precision value specifies the number of digits after the decimal point. If a decimal point appears, at least one digit appears before it. The value is rounded to the appropriate number of digits.	Default precision is 6; if precision is 0, or if the period (.) appears without a number following it, no decimal point is printed.
g, G	The precision specifies the maximum number of significant digits printed. The last printed digit is rounded.	Six significant digits are printed, with any trailing zeros truncated.
s	The precision specifies the maximum number of characters to be printed. Characters in excess of precision are not printed.	Characters are printed until a null character is encountered.
t	The precision specifies how many of the date and time fields are printed. The order and format of the fields are implementation specific. Non-printed fields are truncated (rounded down). 0: Year only 1: Year, Month 2: Date (Year, Month, Day) 3: Date, hour 4: Date, hour, minute 5: Date, hour, minute, second	All fields are printed.

付録 X
(必ずしも守る必要のないもの)

X1. 構造化 Write 文の推奨 DTD

X1.1 Structured Message 構造化メッセージ

<structured.message>は XML 文書としてパースされ解釈されなければならない。このメッセージは以下に示す最初の行で他のコード化メッセージと区別される：

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

X1.2 Usage Notes 使用方法

構造化メッセージは他の文字列同様に語句と変数とを含む。これはもし二重引用符 (") が XML メッセージの中で使われる場合には, 二つの二重引用符 ("") で表現されなければならない。

X1.3 Document Type Definition (DTD)

Document Type Definition (DTD)は<structured.message>の XML の要素, 要素の属性, およびサブ要素を定義する。DTD はどの要素が必須か, 要素の入れ子構造, および取り得る属性値を示す。<structured.message>の DTD は次の様である：

```
<?xml version="1.0"?>
<!-- DOCTYPE structured.message SYSTEM "StructuredMessage.dtd" Arden Syntax 2.1 -->
<!DOCTYPE structured.message [
<!ELEMENT structured.message (body+, distribution.list?)>
<!ELEMENT body              (subject?, context*, conclusion, recommendation*, citation*)>
<!ELEMENT subject           (#PCDATA)>
<!ELEMENT context           (#PCDATA)>
<!ELEMENT conclusion        (#PCDATA)>
<!ELEMENT recommendation    (instruction, choice.list?)>
<!ELEMENT instruction       (#PCDATA)>
<!ELEMENT choice.list       (choice+)>
<!ATTLIST choice.list       type (at.least|at.most|exactly) #IMPLIED
                    number CDATA #IMPLIED>
<!ELEMENT choice            (#PCDATA)>
<!ATTLIST choice            id CDATA #IMPLIED>
<!ELEMENT citation          (#PCDATA)>
<!ATTLIST citation          position (support|refute) #IMPLIED>
<!ELEMENT distribution.list  (distribution+)>
<!ELEMENT distribution      (recipient+, workflow?)>
<!ELEMENT recipient         (#PCDATA)>
<!ATTLIST recipient         role CDATA #IMPLIED>
<!ELEMENT workflow          (closure.select?, forwarding.select?,
                    coverage.select?, timers.select?)>
<!ELEMENT closure.select    EMPTY>
<!ATTLIST closure.select    required (true|false) "false">
<!ELEMENT forwarding.select  EMPTY>
<!ATTLIST forwarding.select  enabled (true|false) "false">
<!ELEMENT coverage.select   EMPTY>
<!ATTLIST coverage.select   enabled (true|false) "false">
<!ELEMENT timers.select     (timeout+)>
<!ELEMENT timeout           (#PCDATA)>
<!ATTLIST timeout           type (submission.ack|delivery.ack|display.ack|closure.ack) #IMPLIED>
]>
```


X1.4 要素と属性

<structured.message>で使われている要素と属性は下に記載される：

X1.4.1 <structured.message>

この要素は XML ベースの構造化 MLM アウトプットメッセージのルート要素である。この要素は MLM 作成者によって全ての情報を表現することができる。この要素は一つ以上の<body>サブ要素と，それに続くオプションの<distribution.list>サブ要素を有する。

X1.4.2 <body>

この要素は MLM メッセージの主要な部分を占める。この要素はオプションの<subject>サブ要素と，それに続く 0 個以上の<context>サブ要素，一つの<conclusion>サブ要素，0 個以上の<recommendation>サブ要素，0 個以上の<citation>サブ要素からなる。

X1.4.3 <subject>

この要素は MLM メッセージの目的である。この要素は MLM アウトプットの特性を示している（例：“パニック検査結果”）。この要素は文字データからなる。

X1.4.4 <context>

この要素は MLM の結果や関連する患者，含まれる以前の検査結果，アレルギー歴などに関係する内容を示している。この要素は文字データからなる。

X1.4.5 <conclusion>

この要素は MLM の logic スロットが true と計算されたときに，MLM の主たる結論文となる（例：“グルコースレベルが急激に低下”）。この要素は文字データからなる。

X1.4.6 <recommendation>

この要素は MLM の logic スロットで計算された条件に応じた MLM 作者の推奨を含んでいる。この要素は一つの<instruction>サブ要素と，それに続くオプションの<choice.list>サブ要素からなる。

X1.4.7 <instruction>

この要素は引き続いたオプションの一つを選択する（例：“治療の選択”）ための指示を伴う推奨されるアクションである。この要素は文字データからなる。

X1.4.8 <choice.list>

この要素は指示に適合した選択可能なオプションからなる。この要素は一つ以上の<choice>サブ要素を有している。この要素には type と number という二つの属性がある。この属性により MLM 作者は後に続く選択肢の性質と個数を示すことができる。Type の値は文字列“at.least”，“at.most”，“exactly”のどれかでなければならない。Number の値はどのような文字列であってもよい（ただし，XML CDATA エスケープが使われていない場合，予約語”<”，”>”，”&”は使ってはならない）が，しかしながら，通常は自然数が期待される。属性値が与えられていない場合，メッセージの使用側にこれらの値の解釈の決定権がある。

X1.4.9 <choice>

この要素はアクションのオプションである。この要素は文字データからなる。この要素は典型的には単一のオプションを識別する（例：“50% デキストロース静脈注射）。しかし，全てのオプション全体を識別すること（例：“上記全て”）や全てのオプションを否定すること（例：“上記全てを否定）にも使用することができる。この要素は id という属性を有する。この属性は各アクションオプション特有の識別子で，後で選択肢を参照するのに用いることができる。属性値はどのような文字列であっても良い（ただし，XML CDATA エスケープが使われていない場合，予約語”<”，”>”，”&”は使ってはならない）。属性値が与えられていない場合，メッセージの使用側にこれらの値の解釈の決定権がある。

X1.4.10 <citation>

この要素は 6.2.4 節で記述されているように，MLM の logic スロットで示されるアルゴリズムの参照情報からなる。この要素は文字データからなる。この要素は position という属性を有する。この属性値は文字列“support”（logic スロットのアルゴリズムを支持することを示す）または“refute”（logic スロットのアルゴリズムを支持しないことを示す）のどちらかでなければならない。Position の値が与えられていない場合，この属性の解釈はメッセージの使用者に任されている。

X1.4.11 <distribution.list>

この要素はメッセージの配布に関する MLM 作者の意向を示している。それぞれの受取人と配布ワークフローと共に複数の配布を同時に表現することができる。この要素は一つ以上の<distribution>サブ要素を有する。

X1.4.12 <distribution>

この要素はメッセージの配布に関する情報からなる。この distribution はメッセージの受取人と関連したワークフロー（これは受取人へのメッセージの配布方式である）を含んでいる。この要素は一つ以上の<recipient>サブ要素と，それに続くオプションの<workflow>サブ要素からなる。

X1.4.13 <recipient>

この要素はメッセージの受取人を指定する。この要素は文字データからなる。この要素は type という属性を有する。この属性値は文字列“person”，“role”，“group”，“unspecified”のどれかでなければならない。”Person”は受取人が人であることを示している。”Group”は受取人がグループであることを示している。グループは引き続き個々のメンバーに分解可能でなければならない。”Role”は受取人が役割であることを示している。役割は引き続きどの個人がその役割に当たっているか分解可能でなければならない。”Unspecified”は適切な受取人が MLM の作者に不明だったことを示している。この場合，この要素に書かれている内容は無視される。動的な患者と医療提供者の関係や医療機関のカバーするスケジュールなどにより，MLM 作者が受取人を指定することが難しくなっている。”Unspecified”は，患者をカバーする適切な知識をもった外部システムが適切な受取人を指定することを示している。この type 属性値が指定されていない場合，デフォルトで”person”が使用される。

X1.4.14 <workflow>

この要素は，MLM 作者がメッセージを受取人に配布するのに使用するワークフローを指定するのに用いる。これには配布に伴う様々な受諾時間も含まれる。この要素はオプションの<closure.select>サブ要素，それに続くオプションの<forwarding.select>サブ要素，それに続くオプションの<coverage.select>サブ要素，それに続くオプションの<timers.select>サブ要素を有する。

X1.4.15 <closure.select>

Closure はメッセージ配布の受取人でワークフローが完了することを示している。この要素は，required という属性以外には何も内容を含んではならない。この属性は MLM 作者がこの配布によって生じる全ての配信を全て止める必要があるかどうか識別する。この属性値は文字列”true”か”false”でなければならない。属性値が与えられていない場合，デフォルト値は”true”である。

X1.4.16 <forwarding.select>

Forwarding は他の受取人へのメッセージの再配布を示している。この要素は, enabled という属性以外には何も含んではならない。この属性は MLM 作者がこの配布を他の受取人に転送することを許しているか識別する。この属性値は文字列”true”か”false”でなければならない。属性値が与えられていない場合, デフォルト値は”true”である。

X1.4.17 <coverage.select>

Coverage は指定された受取人への配布が不可能だった場合に代わりの受取人へのメッセージの配布を示している。この要素は, enabled という属性以外には何も含んではならない。この属性は MLM 作者がこの配布が受取人に届けられない場合に代わりの受取人が必要かを識別している。この属性値は文字列”true”か”false”でなければならない。属性値が与えられていない場合, デフォルト値は”true”である。

X1.4.18 <timers.select>

この要素は配布処理の管理に関してタイムアウトの値を指定している。この要素は一つ以上の<timuout>サブ要素を有している。

X1.4.19 <timeout>

この要素は配布処理の観点からタイムアウトに使用する時間を指定している。この要素は文字データからなる。この要素は type という属性を有する。この属性はタイムアウトのタイプを示している。この属性値は文字列 “submission.ack” (配信ネットワークからのメッセージ投入の肯定応答), “delivery.ack” (配信デバイスからのメッセージ配信の肯定応答), “display.ack” (配信デバイスからのメッセージ表示の肯定応答), “closure.ack” (ワークフロー完了の肯定応答) のいずれかでなければならない。この type 属性値が与えられていない場合, この値の解釈はメッセージの使用者に任される。

X1.5 例

```
<?xml version="1.0"?>
<!DOCTYPE structured.message SYSTEM " StructuredMessage.dtd">
<structured.message>
    <body>
        <subject>Critical Lab Result</subject>
        <context>Mary Smith had a blood sugar of 120 two weeks ago.</context>
        <conclusion>Mary Smith's blood sugar is critical at 435 mg/dl. </conclusion>
        <citation>See Pharmacy and Therapeutics committee standard criteria for abnormal lab studies.</citation>
    </body>
    <distribution.list>
        <distribution>
            <recipient role="personal physician">Dr. John Jones</recipient>
            <workflow>
                <closure.select required="true"/>
                <forwarding.select enabled="true"/>
                <coverage.select enabled="false"/>
                <timers.select>
                    <timeout type="closure.ack"/>
                </timers.select>
            </workflow>
        </distribution>
    </distribution.list>
</structured.message>
```

X2. MLM の例

以下は, 記法を表すための単純な MLM の例である。これらはテストされていないし, 診療にはまだ使用されていない。MLM の現在実際に使用されている例は, 次のコロンビア大のウェブサイトで得ることができる：www.cpmc.columbia.edu/arden。

X2.1 データ解釈 MLM：

```
maintenance:
        title: Fractional excretion of sodium;;
        mlmname: fractional_na;;
        arden: Version 2;;
        version: 1.00;;
        institution: Columbia-Presbyterian Medical Center;;
        author:    George Hripcsak, M.D.
                   (hripcsak@cucis.cis.columbia.edu);;
        specialist: ;;
        date: 1991-03-13;;
        validation: testing;;
library:
        purpose:
                Calculate the fractional excretion of sodium whenever urine
                electrolytes are stored. (This MLM demonstrates data
                interpretation across independent laboratory results.);;
        explanation:
                The fractional excretion of sodium is calculated from the urine
                sodium and creatinine and the most recent serum sodium and
                creatinine (where they occurred within the past 24 hours). A
                value less than 1.0 % is considered low.;;
        keywords: fractional excretion; serum sodium; azotemia;;
        citations:
                1. Steiner RW. Interpreting the fractional excretion of sodium.
                  Am J Med 1984;77:699-702.;;
knowledge:
        type: data-driven;;
        data:
                let (urine_na, urine_creat) be read last
                      ({urine electrolytes where evoking}
                      where they occurred within the past 24 hours) ;
                let (serum_na, serum_creat) be read last
                      ({serum electrolytes where they are not null}
                      where they occurred within the past 24 hours) ;
                let urine_electrolyte_storage be event
                      {storage of urine electrolytes}
                ;;
evoke:
                urine_electrolyte_storage;;
logic:
                /* calculate fractional excretion of sodium */
                let fractional_na be 100 * (urine_na / urine_creat)/
                                                (serum_na / serum_creat) ;
                /* if the frational Na is invalid (e.g., if the */
                /* urine or serum sample is QNS) then stop here */
                if fractional_na is null then
                      conclude false ;
                endif ;
                /* check whether the fractional Na is low */
                let low_fractional_na be      fractional_na < 1.0 ;
                /* send the message */
                conclude true ;
                ;;
```



```
        action:
                if low_fractional_na then
                    write "The calculated fractional excretion of sodium is low ("
                            || fractional_na || "). If the patient is azotemic, " ||
                            "this number may indicate: volume depletion, " ||
                            "hepatic failure, congestive heart failure, acute " ||
                            "glomerulonephritis, oliguric myoglobinuric or " ||
                            "hemoglobinuric renal failure, oliguric contrast " ||
                            "nephrotoxicity, polyuric renal failure with severe " ||
                            "burns, renal transplant rejection, 10 % of cases " ||
                            "with non-oliguric acute tubular necrosis, and " ||
                            "several other forms of renal injury.";
                else
                    write "The calculated fractional excretion of sodium is " ||
                            "not low (" || fractional_na || "). If the patient " ||
                            "is azotemic, this may indicate: acute renal " ||
                            "parenchymal injury, volume depletion coexisting " ||
                            "with diurectic use or pre-existing chronic renal " ||
                            "disease, and up to 10 % of cases of uncomplicated " ||
                            "volume depletion.";
                endif;
                ;;
    end:
```

X2.2　臨床研究スクリーニング MLM：

```
    maintenance:

        title: Screen for hypercalcemia for Dr. B.'s study;;

        mlmname: hypercalcemia_for_b;;

        arden: Version 2;;

        version: 2.02;;

        institution: Columbia-Presbyterian Medical Center;;

        author: George Hripcsak, M.D.;;

        specialist: ;;

        date: 1990-12-04;;

        validation: research;;

    library:

        purpose:
                Screen for hypercalcemia for Dr. B.'s study. (This MLM demonstrates
                screening patients for clinical trials.);;

        explanation:
                The storage of a serum calcium value evokes this MLM. If a serum
                albumin is available from the same blood sample as the calcium,
                then the corrected calcium is calculated, and patients with actual
                or corrected calcium greater than or equal 11.5 are accepted; if
                such a serum albumin is not available, then patients with actual
                calcium greater than or equal 11.0 are accepted. Patients with
                serum creatinine greater than 6.0 are excluded from the study.;;

        keywords: hypercalcemia;;

        citations: ;;

    knowledge:

        type: data-driven;;

        data:
                /* the storage of a calcium value evokes this MLM */
                storage_of_calcium := event {'06210519','06210669'} ;
                /* total calcium in mg/dL */
                calcium := read last {'06210519','06210669';'CALCIUM'} ;
                /* albumin in g/dL */
                evoking_albumin := read last {'06210669';'ALBUMIN' where evoking} ;
                /* albumin in g/dL; not necessarily from same test as Ca */
                last_albumin := read last ({'06210669';'ALBUMIN'}
                      where it occurred within the past 2 weeks) ;
                /* creatinine in mg/dL; not necessarily from same test as Ca */
                creatinine := read last ({'06210669','06210545','06000545';'CREAT'}
                where it occurred within the past 2 weeks) ;
                ;;

    evoke:
                storage_of_calcium;;
```

```
logic:
        /* make sure the Ca is present (vs. hemolyzed, ...) */
        IF calcium is not present THEN
            conclude false ;
        ENDIF ;
        /* if creatinine is present and greater than 6, then stop now */
        IF creatinine is present THEN
            IF creatinine is greater than 6.0 THEN
                conclude false ;
            ENDIF ;
        ENDIF ;
        /* is an albumin present for the same sample as the calcium */
        IF evoking_albumin is present THEN
        /* calculate the corrected calcium */
            IF evoking_albumin is less than 4.0 THEN
                corrected_calcium := calcium + (4.0-evoking_albumin)*0.8 ;
            ELSE
                /* corrected is never less than actual */
                corrected_calcium := calcium ;
            ENDIF ;
            /* test for total or corrected calcium >= 11.5 */
            IF calcium >= 11.5 OR corrected_calcium >= 11.5 THEN
                message := "calcium = " || calcium ||
                                " on " || time of calcium ||
                                " (corrected calcium = " ||
                                corrected_calcium || ")" ;
                message := message||"; albumin = "||evoking_albumin ;
                IF creatinine is present THEN
                    message := message||
                                    "; last creatinine = "||creatinine ;
                    message := message||
                                    "; (total or corrected calcium " ||
                                    "was at least 11.5)" ;
                    conclude true ;
                ELSE
                    conclude false ;
                ENDIF ;
            ENDIF
        /* no evoking albumin was present */
        ELSE
        /* check for true calcium >= 11.0 */
            IF calcium >= 11.0 THEN
                message := "calcium = "||calcium||" on "||time of calcium ;
                IF last_albumin is present THEN
                    message := message||"; last albumin "||
                            "(not from same blood sample as calcium) = "||
                            last_albumin ;
                    IF creatinine is present THEN
                        message := message|| "; last creatinine = "
                                        ||creatinine ;
                        message := message||
                                        "; (total calcium was at least 11.0; "||
                                        "corrected calcium was not calculated)" ;
                        conclude true ;
                    ELSE
                        conclude false ;
                    ENDIF ;
                ENDIF ;
            ENDIF ;
        ENDIF ;
    ;;
    action: write "hypercalcemia study: " || message;;
    urgency: 50;;
end:
```

X2.3　禁忌アラート MLM：

```
maintenance:
    title: Check for penicillin allergy;;
    mlmname: pen_allergy;;
    arden: ASTM-E1460-1995;;
    version: 1.00;;
    institution: Columbia-Presbyterian Medical Center;;
    author: George Hripcsak, M.D.;;
    specialist: ;;
```

```
        date: 1991-03-18;;
        validation: testing;;
    library:
        purpose:
                When a penicillin is prescribed, check for an allergy. (This MLM
                demonstrates checking for contraindications.);;
        explanation:
                This MLM is evoked when a penicillin medication is ordered. An
                alert is generated because the patient has an allergy to penicillin
                recorded.;;
        keywords: penicillin; allergy;;
        citations: ;;
    knowledge:
        type: data-driven;;
        data:
                /* an order for a penicillin evokes this MLM */
                penicillin_order := event {medication_order where
                                                        class = penicillin};
                /* find allergies */
                penicillin_allergy := read last {allergy where
                                                                agent_class = penicillin};
                ;;
        evoke:
                penicillin_order;;
        logic:
                if exist(penicillin_allergy)then
                      conclude true;
                endif;
                ;;
        action:
                write "Caution, the patient has the following allergy to penicillin documented: " || penicillin_allergy;;
        urgency: 50;;
    end:
```

X2.4 推奨管理 MLM：

```
    maintenance:
        title: Dosing for gentamicin in renal failure;;
        mlmname: gentamicin_dosing;;
        arden: ASTM-E1460-1995;;
        version: 1.00;;
        institution: Columbia-Presbyterian Medical Center;;
        author: George Hripcsak, M.D.;;
        specialist: ;;
        date: 1991-03-18;;
        validation: testing;;
    library:
        purpose:
                Suggest an appropriate gentamicin dose in the setting of renal
                insufficiency. (This MLM demonstrates a management suggestion.);;
        explanation:
                Patients with renal insufficiency require the same loading dose of
                gentamicin as those with normal renal function, but they require a
                reduced daily dose. The creatinine clearance is calculated by serum
                creatinine, age, and weight. If it is less than 30 ml/min, then an
                appropriate dose is calculated based on the clearance. If the
                ordered dose differs from the calculated dose by more than 20 %,
                then an alert is generated.;;
        keywords: gentamicin; dosing;;
        citations: ;;
    knowledge:
```

```
type: data-driven;;
data:
        /* an order for gentamicin evokes this MLM */
        gentamicin_order := event {medication_order where
                                                class = gentamicin} ;
        /* gentamicin doses */
        (loading_dose,periodic_dose,periodic_interval) :=
            read last {medication_order initial dose,
                        periodic dose, interval} ;
        /* serum creatinine mg/dl */
        serum_creatinine := read last ({serum_creatinine}
            where it occurred within the past 1 week) ;
        /* birthdate */
        birthdate := read last {birthdate} ;
        /* weight kg */
        weight := read last ({weight}
            where it occurred within the past 3 months) ;
        ;;
evoke:
        gentamicin_order;;
logic:
        age := (now - birthdate)/1 year ;
        creatinine_clearance   := (140 - age) * (weight)/
                                            (72 * serum_creatinine) ;
        /* the algorithm can be adjusted to handle higher clearances */
        if creatinine_clearance < 30 then
            calc_loading_dose := 1.7 * weight ;
            calc_daily_dose := 3 * (0.05 + creatinine_clearance / 100) ;
            ordered_daily_dose     := periodic_dose *
                                            periodic_interval/(1 day) ;
            /* check whether order is appropriate */
            if (abs(loading_dose - calc_loading_dose)/
          calc_loading_dose > 0.2)
      or
              (abs(ordered_daily_dose - calc_daily_dose)/
                      calc_daily_dose > 0.2)then
                conclude true ;
            endif ;
        endif ;
        ;;
action:
        write "Due to renal insufficiency, the dose of gentamicin " ||
                "should be adjusted. The patient's calculated " ||
                "creatinine clearance is " || creatinine_clearance ||
                " ml/min. A single loading dose of " ||
                calc_loading_dose || " mg should be given, followed by " ||
                calc_daily_dose || " mg daily. Note that dialysis may " ||
                "necessitate additional loading doses."
        ;;
urgency: 50;;
end:
```

X2.5　モニタリング MLM：

```
maintenance:
    title: Monitor renal function while taking gentamicin;;
    mlmname: gentamicin_monitoring;;
    arden: Version 2;;
    version: 1.00;;
    institution: Columbia-Presbyterian Medical Center;;
    author: George Hripcsak, M.D.;;
    specialist: ;;
    date: 1991-03-19;;
    validation: testing;;
library:
    purpose:
        Monitor the patient's renal function when the patient is taking
        gentamicin. (This MLM demonstrates periodic monitoring.);;
```

```
        explanation:
                This MLM runs every five days after the patient is placed on
                gentamicin until the medication is stopped. If the serum creatinine
                has not been checked recently, then an alert is generated
                requesting follow-up. If the serum creatinine has been checked, is
                greater than 2.0, and has risen by more than 20 %, then an alert is
                generated warning that the patient may be developing renal
                insufficiency due to gentamicin.;;

        keywords: gentamicin; renal function;;

        citations: ;;

    knowledge:

        type: data-driven;;

        data:
                /* an order for gentamicin evokes this MLM */
                gentamicin_order := event {medication_order where
                                                        class = gentamicin};
                /* check whether gentamicin has been discontinued */
                gentamicin_discontinued :=
                    read exist({medication_cancellation where class = gentamicin}
                        where it occurs after eventtime);
                /* baseline serum creatinine mg/dl */
                baseline_creatinine := read last ({serum_creatinine}
                    where it occurred before eventtime);
                /* followup serum creatinine mg/dl */
                recent_creatinine := read last ({serum_creatinine}
                    where it occurred within the past 3 days);
                ;;

        evoke:
                every 5 days for 10 years starting 5 days after time of
                    gentamicin_order until gentamicin_discontinued;;

        logic:
                if recent_creatinine is not present then
                    no_recent_creatinine := true;
                    conclude true;
                else
                    no_recent_creatinine := false;
                    if % increase of (serum_creatinine,
                            recent_creatinine) > 20 /* % */
                            and recent_creatinine > 2.0 then
                            conclude true;
                    endif;
                endif;
                ;;

    action:
                if no_recent_creatinine then
                    write "Suggest obtaining a serum creatinine to follow up " ||
                                "on renal function in the setting of gentamicin.";
                else
                    write       "Recent serum creatinine (" || recent_creatinine ||
                                " mg/dl) has increased, possibly due to renal " ||
                                "insufficiency related to gentamicin use.";
                endif;
            ;;

        urgency: 50;;

    end:
```

X2.6 推奨管理 MLM：

```
    maintenance:

        title: Granulocytopenia and Trimethoprim/Sulfamethoxazole;;

        mlmname: anctms;;

        arden: Version 2;;

        version: 2.00;;

        institution: Columbia-Presbyterian Medical Center;;

        author: George Hripcsak, M.D.;;

        specialist: ;;

        date: 1991-05-28;;

        validation: testing;;   library:
```

```
        purpose:
                Detect granulocytopenia possibly due to
                trimethoprim/sulfamethoxazole;;

        explanation:
                This MLM detects patients that are currently taking
                trimethoprim/sulfamethoxazole whose absolute neutrophile count is
                less than 1000 and falling.;;

        keywords:
                granulocytopenia; agranulocytosis; trimethoprim; sulfamethoxazole;;

        citations:
                1. Anti-infective drug use in relation to the risk of
                  agranulocytosis and aplastic anemia. A report from the
                  International Agranulocytosis and Aplastic Anemia Study.
                  Archives of Internal Medicine, May 1989, 149(5):1036-40.;;

        links:
                'CTIM .34.56.78';
                'MeSH agranulocytosis/ci and sulfamethoxazole/ae';;

knowledge:

        type: data-driven;;

        data:
                /* capitalized text within curly brackets would be replaced with */
                /* an institution's own query                                    */
                let anc_storage be event {STORAGE OF ABSOLUTE_NEUTROPHILE_COUNT};
                let anc be read last 2 from ({ABSOLUTE_NEUTROPHILE_COUNT}
                        where they occurred within the past 1 week);
                let pt_is_taking_tms be read exist
                        {TRIMETHOPRIM_SULFAMETHOXAZOLE_ORDER};
                ;;

evoke: anc_storage;;

        logic:
                if pt_is_taking_tms
                and the last anc is less than 1000
                and the last anc is less than the first anc
                 /* is anc falling? */
                then
                        conclude true;
                else
                        conclude false;
                endif;;

        action:
                write       "Caution: patient's relative granulocytopenia may be " ||
                            "exacerbated by trimethoprim/sulfamethoxazole.";

end:
```

X2.7　CARE から移行された MLM：

```
maintenance:

        title: Cardiology MLM from CARE, p. 85;;

        mlmname: care_cardiology_mlm;;

        arden: Version 2;;

        version: 1.00;;

        institution: Regenstrief Institute;;

        author: Clement J. McDonald, M.D.; George Hripcsak, M.D.;;

        specialist: ;;

        data: 1991-05-28;;

        validation: testing;;

library:

        purpose:
                Recommend higher beta-blocker dosage if it is currently low and the
                patient is having excessive angina or premature ventricular
                beats.;;

        explanation:
                If the patient is not bradycardic and is taking less than 360 mg of
                propanolol or less than 200 mg of metoprolol, then if the patient
                is having more than 4 episodes of angina per month or more than 5
                premature ventricular beats per minute, recommend a higher dose.;;
```

```
        keywords:
                beta-blocker, angina; premature ventricular beats; bradycardia;;

        citations:
                1. McDonald CJ. Action-oriented decisions in ambulatory medicine.
                   Chicago: Year Book Medical Publishers, 1981, p. 85.
                2. Prichard NC, Gillam PM. Assessment of propranolol in angina
                   pectoris: clinical dose response curve and effect on
                   electrocardiogram at rest and on exercise. Br Heart J,
                   33:473-480 (1971).
                3. Jackson G, Atkinson L, Oram S. Reassessment of failed beta-
                   blocker treatment in angina pectoris by peak exercise heart rate
                      measurements. Br Med J, 3:616-619 (1975).
                ;;

knowledge:

        type: data-driven;;

        data:
                let last_clinic_visit be read last {CLINIC_VISIT};
                let (beta_meds,beta_doses,beta_statuses) be read
                      {MEDICATION,DOSE,STATUS
                      where the beta_statuses are 'current'
                      and beta_meds are a kind of 'beta_blocker'};
                let low_dose_beta_use be false;
                /* if patient is on one beta blocker, check if it is low dose    */
                if the count of beta_meds = 1 then
                      if      (the last beta_meds = 'propanolol'
                              and
                               last beta_doses < 360)
                        or (the last beta_meds = 'metoprolol'
                              and
                               the last beta_doses <= 200) then
                              let low_dose_beta_use be true;
                      endif;
                endif;
                let cutoff_time be the maximum of
                ((1 month ago),(time of last_clinic_visit),
                 (time of last beta_meds));
                /* a system-specific query to angina frequency, PVC frequency,   */
                /* and pulse rate would              replace capitalized terms          */
                let angina_frequency be read last ({ANGINA_FREQUENCY}
                      where it occurred after cutoff_time);
                let premature_beat_frequency be read last
                      ({PREMATURE_BEAT_FREQUENCY}
                      where it occurred after cutoff_time);
                let last_pulse_rate be read last {PULSE_RATE};
                ;;

evoke: /* this MLM is called directly */;;

logic:
                if last_pulse_rate is greater than 60 and
                      low_dose_beta_use then
                      if angina_frequency is greater than 4 then
                            let message be
                                  "Increased dose of beta blockers may be "||
                                  "needed to control angina.";
                            conclude true;
                      else
                            if premature_beat_frequency is greater than 5 then
                                  let message be
                                        "Increased dose of beta blockers may "||
                                        "be needed to control PVC's.";
                                  conclude true;
                            endif;
                      endif;
                endif;
                conclude false;
                ;;

action:
          write message;;

end:
```

X2.8 While Loop を使用する MLM：

```
maintenance:

        title:  Allergy_test_with_while_loop;;

        filename:  test_for_allergies_while_loop;;

        version:  0.00;;
```

```
        institution:  ;;
        author:  ;;
        specialist:  ;;
        date:  1997-11-06;;
        validation:  testing;;
library:
        purpose:
                Illustrates the use of a WHILE-LOOP that processes an entire list
                ;;
        explanation:
                ;;
        keywords:
             ;;
knowledge:
        type: data-driven;;
        data:
                /* Receives four arguments from the calling MLM: */
                (med_orders,
                 med_allergens,
                 patient_allergies,
                 patient_reactions) := ARGUMENT;
                ;;
        evoke:
                ;;
        logic:
                /* Initializes variables */
                a_list:= ();
                m_list:= ();
                r_list:= ();
                num:= 1;
                /* Checks each allergen in the medications to determine          */
                /* if the patient is allergic to it                                  */
                while num <= (count med_allergen) do
                        allergen:= last(first num from med_allergens);
                        allergy_found:= (patient_allergies = allergen);
                        reaction:= patient_reactions where allergy_found;
                        medication:= med_orders where (med_allergens = allergen);

                        /* Adds the allergen, medication, and reaction to          */
                        /* variables that will be returned to the calling MLM          */
                        If any allergy_found then
                                a_list:= a_list, allergen;
                                m_list:= m_list, medication;
                                r_list:= r_list, reaction;
                        endif;
                /* Increments the counter that is used to stop the while-loop    */
                        num:= num + 1 ;
                enddo;
                /* Concludes true if the patient is allergic to one of                  */
                /* the medications                                                                                 */
                If exist m_list
                        then conclude true;
                endif;
                ;;
        action:
                /* Returns three lists to the calling MLM                                */
                return  m_list, a_list, r_list;
                ;;
end:
```


X3. 変更履歴

1992 規格 **(Version 1)** から **Version 2** への主な変更点**:**

演算子の詳細の明確化

Arden syntax version スロット が必須に. (6.1.3節)

引用は番号付けされ, 支持と反論にクラス分類((6.2.4節)

Links スロットの仕様 (6.2.5節)

日時は継続時間から+ 演算子で構成が可能に(7.1.5.3 節)

Triggertime は MLM が起動された日時(8.4.5節)

照会抽出命令は primary time では不必要に (8.9.2 節)

Interface 文が外部関数を使うために導入された (11.2.12 節)

単一行のコメントは"//"で指定可能に (7.1.9 節)

filename スロットは **mlmname** に名称を変更 (6.1.2節)

いくつかの新しい演算子が導入された:

- **sort** (9.2.4節)
- **reverse** (9.12.21節)
- **format** (9.8.2節)
- **earliest**, **latest** (9.12.17節, 9.12.16節)
- **floor, ceiling, truncate, round** (9.16.11節, 9.16.12節, 9.16.13節, 9.16.14節)
- **index** (...[...]) (9.12.18節)
- **year, month, day, hour, minute, second** field extraction (9.11.2節, 9.11.4節, 9.11.7節, 9.11.9節, 9.11.11節, 9.11.13節)
- **seqto** (9.12.20節)
- **string**, **extract characters** (9.8.3節, 9.12.19節)

リストから選択する演算子は, 要素ではなくインデックスで注釈されるように (9.12.18節)

As number 演算子は文字列とブール型変数を数値に変換 (9.16.17節)

いくつかの制限の緩和(例:文字列の中の二重セミコロン).

call 表現と文は複数の被演算子を渡せるようになった。また複数の被演算子は action スロットからも渡せるようになった。(10.2.4 節, 11.2.4 節, 12.2.2 節, 12.2.4節)

ループ構造が追加された。for loop, while loop. (10.2.5 節, 10.2.6 節)

continue 文において, **unless** を it に加えられるようになった (this a readability aid).

条件付実行文において, **unless** を **conclude** 文で使用できるようになった

"read ... where ..." に括弧は必要でなくなった。

読み込み照会ではソート順を指定するように変更(primary time の時間順というデフォルトとは異なる).

Version 2 から **Version 2.1** への主な変更点**:**

- **WRITE** 文のための構造化メッセージ（Extensible Markup Language (XML)で符号化された Document Type Definition として表現）が導入された(X1.4.1 <structured.message>)
- **in** 演算子が **is in** の同義語となった。同様に **not in** が **is not in** の同義語となった (9.6.23)
- **Occur/occurs/occurred at** が **occur/occurs/occurred equal** の同義語となった. (9.7.11)
- **from <time>** が **after <time>**の同義語となった. (9.10.4)
- 読点 (“.”)が **Mlmname** スロットで使用可能に. (6.1.2)
- 新しい予約語 **Currenttime** は MLM 実行中のいかなる時点でもシステム日時を返す (8.4.6)
- 六つの新しい文字列演算子が使用可能に： **length** (9.8.5), **uppercase** (9.8.6), **lowercase** (9.8.7), **trim** (9.8.8), **find…in string** (9.8.9), **substring…characters from** (9.8.10).
- where トリガー文は削除された
- Arden Syntax version スロットに新しいコード—Version 2.1—が Version 2 と Version 2.1 のどちらに準拠した MLMs かを表すために加えられた。

参考文献

(1) HELP Frame Manual, 1991, LDS Hospital, 325 8th Ave., Salt Lake City, UT 84143.

(2) McDonald, C. J., Action-Oriented Decisions in Ambulatory Medicine, Chicago: Year Book Medical Publishers, 1981.

(3) Wirth, N., "What Can We Do About the Unnecessary Diversity of Notation for Syntactic Definitions?" Communications of the ACM, Vol 20, 1977, pp. 822-823.

(4) UMLS Knowledge Sources, Experimental Edition, Bethesda, MD: National Library of Medicine, September 1990.

(5) International Committee of Medical Journal Editors, Special Report, "Uniform Requirements for Manuscripts Submitted to Biomedical Journals," The New England Journal of Medicine, Vol 324, No. 6, 1991, pp. 424-428.

（技術文書 07-101）

2007 年 8 月　発行

～Arden Syntax の調査～

発行元　保健医療福祉情報システム工業会

〒105-0001 東京都港区虎ノ門1丁目19－9

（虎ノ門TBLビル6F）

電話　03-3506-8010　ＦＡＸ　03-3506-8070