

## Part B: Arden Syntax

---

## Overview

- I. Details of the Syntax
  - a. Structure
  - b. Data types and operators
- II. Writing MLMs
- III. Tools

## Details of Syntax: Structure

- Arden Syntax is an HL7/ANSI standard specification for defining and sharing medical knowledge-base information.
- Current approved version is 2.1

## Support for Arden Syntax

### Institutions

- Cedars-Sinai Medical Center

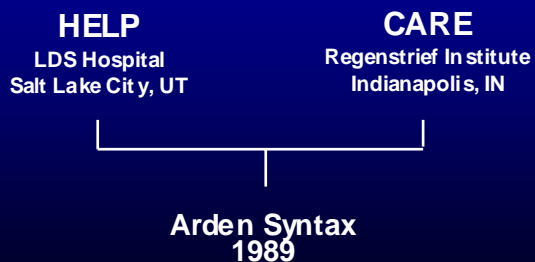
### Software Vendors

- Eclipsys/Healthvision
- McKesson
- Siemens

### Knowledge Vendors

- Micromedex

## Arden Syntax - History



## Arden Syntax - Rationale

*Arden Syntax arose from the need to make medical knowledge available for decision making at the point of care.*

- Allow knowledge sharing within and between institutions
- Make medical knowledge and logic explicit
- Standardize the way medical knowledge is integrated into hospital information systems

## Medical Logic Module

- MLM = an independent unit in a health knowledge base.
- MLM: Makes a single health decision
  - maintenance information
  - links to other sources of knowledge/data,
  - logic
- MLM = a stream of text stored in an ASCII file in statements called slots

## MLM - Structure

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

## Maintenance Category Slots (9)

- . Title
- . Mlmmname
- . Arden
- . Version
- . Institution
- . Author
- . Specialist
- . Date
- . Validation

## Maintenance Slots - Example

```
maintenance:
  title: Contrast CT study in patient with renal failure;;
  mlmmname: ct_contr.mlm;;
  arden: Version 2;;
  version: 1.00;;
  institution: Arden Medical Center;;
  author: John Doe, MD;;
  specialist: Jane Doe, MD;;
  date: 1995-09-11;;
  validation: testing;;
```

## Library Category Slots (5)

- . Purpose
- . Explanation
- . Keywords
- . Citations (optional)
- . Links (optional)

## Library Slots - Example

```
library:
  purpose: To alert the health care provider of new or worsening
  serum creatinine level;;
  explanation: If the creatinine is at or above a threshold (1.35
  mg/dl), then an alert...;;
  keywords: renal insufficiency; renal failure ;;
  citations: Proceedings of the Fifteenth Annual Symposium on
  Computer Applications in Medical Care; 1991 Nov 17-20;
  Washington, D.C. New York: IEEE Computer Society Press,
  1991.
  links: URL "NLMWeb Page", http://www.nlm.nih.gov/;;
```

## Knowledge Category Slots (7)

- . Type
- . Data
- . Priority
- . Evoke
- . Logic
- . Action
- . Urgency

## Type Slot

- . Coded slot, required
- . Presently only one type slot is defined :  
*type: data\_driven;;*
- . "Data\_driven" implies that these slots follow:  
*data, priority, evoke, logic, action, urgency*

## Data Slot

- . Terms in the medical logic module must be mapped to a database
- . Use of curly braces { } allows flexibility in mapping to the institution's local database
- . Mapping terms in this way separates the logic in the MLM from institution-specific information

## Data Slot – Statements

- . read statement *without* an operator obtains a list of values from the database
- . read statement *with* an operator obtains a single value from the database
- . examples of operators:  
*first, last, min, max, count, average, sum*

## Data Slot - Example

```
creatinine := read {dam="PDQRES2"};
last_creat := read last {select "OBSRV_VALUE" from
"LCR" where qualifier in
("CREATININE","QUERY_OBSRV_ALL")};
```

## Priority Slot

- . Optional, coded slot, seldom used
- . Specifies relative order in which MLMs are evoked (99 is highest priority and is evoked first, 1 is lowest priority and is evoked last)
- . If not specified, default value is 50

## Evoked Slot

The evoked slot defines what triggers an MLM

Example triggers:

- The occurrence of an event
- Timed execution after an event
- Periodic repetition after an event
- Direct call from another MLM

## Evoked Slot - Example

```
data:
  creatinine_storage := event {'32506','32752'};

evoked:
  creatinine_storage;;
```

## Evoked Slot – Examples of time delays after an event and recurring events

```
evoked: 3 days after time of creatinine_storage;
```

```
evoked: every 1 day for 7 days starting at time of
  creatinine_storage;
```

```
evoked: every 1 day starting at time of K_storage
  until K>=3;
```

## Logic Slot

- Set of medical criteria
- Logical algorithm
- Ends with a “conclude statement”

```
conclude true;
  or
conclude false;
```

## Logic Slot - if ... then ...

```
if <expr1> then
  <block1>
endif;
```

```
if <expr1> then
  <block1>
else
  <block2>
endif;
```

```
if <expr1> then
  <block1>
elseif <expr2> then
  <block2>
elseif <expr3> then
  <block3>
...
elseif <exprN> then
  <blockN>
else
  <blockE>
endif;
```

## Logic Slot - Iteration

```
while <expr> do
  <block>
enddo;
```

```
for <expr> do
  <block>
enddo;
```

## Logic Slot - call statements

```
<var> := call <name>;  
  
<var> := call <name> with <expr>;  
  
(<var>, <var>, ...) := call <name> with <expr>;  
  
<var> := call <name> with <expr>, ..., <expr>;  
  
(<var>, <var>, ...) := call <name> with <expr>, ..., <expr>;
```

## Example - call statements

```
var1 := call my_mlm with param1, param2;  
  
var1 := call my_event with param1, param2;  
  
var1 := call my_interface_function with param1,  
param2;
```

## Conclude Statement

- *conclude true;*
  - terminate the rule
  - go to the action slot
- *conclude false;*
  - terminate the rule
  - do not go to the action slot

## Logic Slot - Example

```
logic:  
  if last_creat is not present then  
    alert_text := "No recent creatinine available. Consider  
    ordering creatinine before giving IV contrast.";  
    conclude true;  
  elseif last_creat > 1.5 then  
    alert_text := "This patient has an elevated creatinine.  
    Giving IV contrast may worsen renal function.";  
    conclude true;  
  else conclude false;  
endif;
```

## Action Slot

Carries out action if logic slot concludes true

### Examples of actions

- Write a message to screen
- Store a message in a file
- Call another medical logic module

## Action Slot - Example

```
action:  
write "Last creatinine: " || last_creat || " on: " || time  
of last_creat;
```

*appears as:*

```
Last creatinine: 2.36 on: 1997-02-16T06:30:00
```

## Action Slot - Example

data:

```
john_email := destination {'john@arden.edu'};
```

...

action:

```
write "Patient who may qualify for study registered  
today. Pt #: " || patient_no at john_email ;
```

## Data Types

### Data Types

- Null
- Boolean
- Number
- Time
- Duration
- String
- Term
- List
- Query Results\*

### Null

- Special data type that signifies uncertainty
- May be the result of a lack of information in the patient database or an explicit null value in the database.
- Results from an error in execution, such as a type mismatch or division by zero.
- Null may be specified explicitly within a slot using the word *null* (that is, the null constant).

### Boolean

- Boolean data type includes the two truth values: true and false.
- The word *true* signifies Boolean true.
- The word *false* signifies Boolean false.
- The logical operators use tri-state logic by using *null* to signify the third state, uncertainty.

### Number

- There is a single number type
- There is no distinction between integer and floating point numbers.
- Internally, all arithmetic is done in floating point.

## Time

- Time data type refers to points in absolute time (also referred to as timestamp in other systems)
- Both date and time-of-day must be specified
- Times back to the year 1800 must be supported
- Times before 1800-01-01 are not valid
- The granularity of time is always infinitesimal (not discrete seconds)
- Times stored in databases will have varying granularities.
- When a time is read by the MLM, it is always truncated to the beginning of the granule interval

## Special Time “Constants”

- **Now** Time execution of MLM started
- **Eventtime** Primary time of evoking event
- **Triggertime** When the evoking event occurred
- **Currenttime** Current system clock time (“now”)

eventtime <= triggertime <= now <= currenttime.

## Duration

- Signifies an interval of time not anchored to any particular point in absolute time
- No duration constants.
- Build using the duration operators (later slides)

## String

- Streams of characters of variable length
- Enclosed in double quotes (“ ”)

## Term

- Streams of characters of variable length
- Enclosed in single quotes (‘ ’)
- Currently only used
  - mlmnames within a structured slot
  - linktext portion of a structured linkrecord
  - call statement to reference mlmnames or links
- Future – may be used for controlled vocabulary

## List

- Ordered set of elements
- Lists may be heterogeneous (elements may be of different types)
- No nested lists (a list cannot be a list element)
- One list constant, the empty list
  - signified by using a pair of empty parentheses: ()
- Lists are created by
  - using list operators like the comma (,) to build lists from single items
  - Results of queries

## Query Results

- Result of a database query has a time value in addition to its data value
- Result of a query is assigned to a variable for use in the other slots
- Primary Time
  - Every item in the patient database is assumed to have some primary time (also called time of occurrence) associated with it
  - This time is defined as the medically relevant time for that query. For different entities, the primary time might signify different times.
  - Implicit in every query to the patient database is a request for the primary time of the data
- Retrieval Order
  - Default – sorted in chronological order by the primary time of the result. The query may specify a different sort order.
- Data Value
  - If a variable has been assigned the result of a query, then the use of the variable always refers to the data value.

## Operators

### List Operators

- , (unary) (single element list creation)
- , (binary) (multi-element list creation)
- Merge
- Sort

### Where Operator

- Where

### Logical Operators

- Or
- And
- Not

### Simple Comparison Operators

- = (is equal to)
- ≠ (is not equal to)
- < (less than)
- ≤ (less than or equal to)
- > (greater than)
- ≥ (greater than or equal to)

## Is Comparison Operators

Perform comparisons based on the value of the operand.

- Is [not] Equal
- Is [not] Less Than
- Is [not] Greater Than
- Is [not] Less Than or Equal
- Is [not] Greater Than or Equal
- Is [not] Within ... To
- Is [not] Within ... Preceding
- Is [not] Within ... Following
- Is [not] Within ... Surrounding
- Is [not] Within Past
- Is [not] Within Same Day As
- Is [not] Before
- Is [not] After
- Is [not] In
- Is [not] Present
- Is [not] Null
- Is [not] Boolean
- Is [not] Number
- Is [not] String
- Is [not] Time
- Is [not] Duration
- Is [not] List
- [not] In

## Occur Comparison Operators

Perform comparisons based on the primary time of the operand rather than acting on the value of the operand.

- Occur [not] Equal
- Occur [not] Before
- Occur [not] After
- Occur [not] At
- Occur [not] Within ... To
- Occur [not] Within ... Preceding
- Occur [not] Within ... Following
- Occur [not] Within ... Surrounding
- Occur [not] Within Past
- Occur [not] Within Same Day As

## String Operators

- Formatted with String ...
- Trim [Left | Right]
- Find...[in] string... [starting at]...
- Substring ... characters [starting at ...] from ...
- || -- concatenation
- Matches pattern
- Length
- Uppercase
- Lowercase

## Arithmetic Operators

- + (unary)
- (unary)
- \* -- multiplication
- / -- division
- \*\* -- exponentiation
- + (binary) -- addition
- (binary) -- subtraction

## Temporal Operators

Temporal operators operate on *discrete* time points.

- After
- Before
- Ago
- From

## Duration Operators

Duration operators deal with the setting or reading of time *intervals*.

- Year
- Month
- Week
- Day
- Hour
- Minute
- Second
- Extract year
- Extract month
- Extract day
- Extract hour
- Extract minute
- Extract second

## Aggregation Operators

Aggregation operators take a list as one argument and return a single item as a result.

- Count
- Exist
- Average
- Median
- Sum
- stddev
- Variance
- Minimum
- Maximum
- Last
- First
- Any
- All
- No
- Latest
- Earliest
- Element
- Extract characters ...
- Seqto
- Reverse
- Index Extraction Aggregation operators
  - Index earliest
  - Index latest
  - Index minimum
  - Index maximum

## Query Aggregation Operators

- Query aggregation operators take a list as one argument and return a single item as a result.
- Nearest ... From
- Index Nearest ... From
- Slope

## Transformation Operators

Transformation operators extract information from lists to generate new lists

- Minimum ... From
- Maximum ... From
- First ... From
- Last ... From
- Increase
- % Increase
- Decrease
- % Decrease
- Earliest ... From
- Latest ... From
- Index Extraction
  - Index First ... From
  - Index Last ... From

## Query Transformation Operator

- The interval operator returns the difference between the primary times of succeeding items in a list

Interval

## Numeric Function Operators

- Arccos
- Arcsin
- Arctan
- Cosine
- Sine
- Tangent
- Exp
- Log
- Log10
- Int
- Floor
- Ceiling
- Truncate
- Round
- Abs
- sqrt
- As number

## Time Function Operator

- Used to return the primary time of a query result
- Used to set the primary time of a value

Time

## Writing MLMs Tools

### Traditional Tools

- Text editor
- Form-based editor
- Compiler / syntax checker
- Copy of standard or quick reference

### Traditional Methods

- Text editor
  - “Blank slate”
  - Templates
  - Modify existing

### Emerging Tools

- Intelligent form-based tools
- Database mapping assistants
- “Wizard”-style tools

### Getting Started Writing MLMs

- Identify problem to be solved
  - Scope
  - Bounds and metrics
- Research / find expert
- Outline decision process
- Write MLM
- Test/modify MLM until
  - you / expert are satisfied that it is correct