GLIF

Thu Feb 17 11:30:05 PST 2000

Package *Expression_Package*

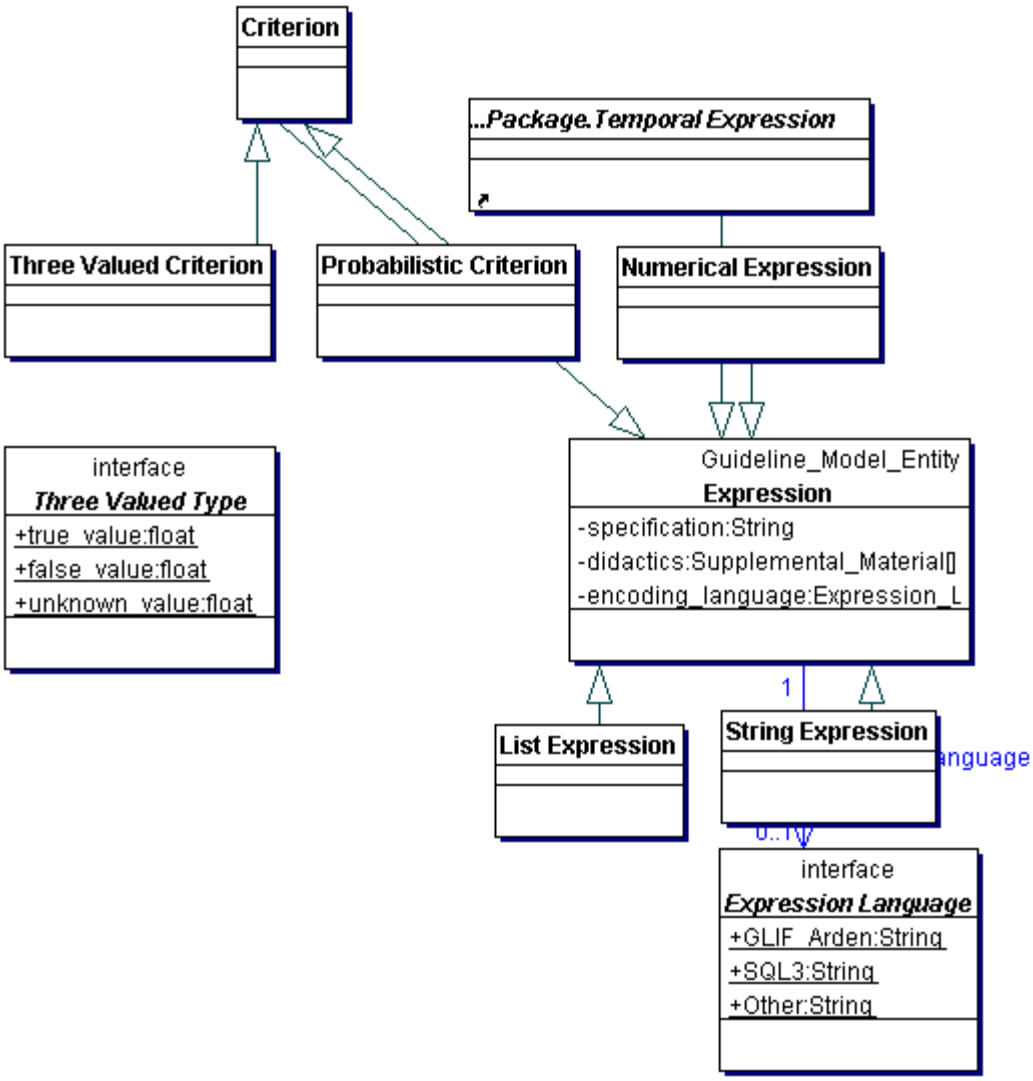
Classes

class Expression_Package.Criterion
class Expression_Package.Expression
class Expression_Package.List_Expression
class Expression_Package.Numerical_Expression
class Expression_Package.Probabilistic_Criterion
class Expression_Package.String_Expression
class Expression_Package.Three_Valued_Criterion

Interfaces

interface Expression_Package.Expression_Language
interface Expression_Package.Three_Valued_Type

Class Diagram



The Expression class is a parent Class for all expressions: criterion, temporal expression, list expression, numerical expression and string expression. Different expression languages can be used. Currently, BNF grammar is given for a modification of the Arden Syntax logic grammar, called GLIF_ARDEN. While this gives guideline users the flexibility to specify criteria in a language of their choosing, it is possible that the resulting guidelines will not be sharable because constructs supported by one language (e.g., temporal constructs) cannot be expressed in another language.

Interface Node Detail

Interface *Expression_Package.Expression_Language*

Interface *Expression_Package.Three_Valued_Type*

Class Detail

Class *Expression_Package.Criterion*

Inherits from:

Expression_Package.Expression

Criteria: There are two types of criteria: three-valued, and probabilistic. Three-valued criteria evaluate to true, false, or unknown. Probabilistic criteria are used to add conditional probabilities or Bayesian network information. With probabilistic criteria, the dependence probabilities among all variables in a specification have to be known in order for the criterion to be evaluated. The problem is that the number of conditional probabilities that need to be specified increases exponentially as the number of variables used in a specification increase, and the dependence relationships among some variables may not be known. Users of probabilistic criteria need to provide the conditional probabilities needed for evaluating these criteria, and this means that use will probably be limited to advanced users of GLIF.

Class *Expression_Package.Expression*

Inherits from:

Guideline_Model_Entity

Description:

Purpose

Parent Class for all expressions: logical expressions (criteria), temporal expressions, arithmetic expressions, literals, and functions

Considerations:
There is no class Two-Valued_Criterion because expressions involving boolean/two-valued criteria can be handled using three-valued criteria.
The attribute, Expression.encoding_language was added to provide a means of using different expression languages (besides GLIF_ARDEN) in GLIF. While this gives guideline users the flexibility to specify criteria in a language of their choosing, it is possible that the resulting guidelines will not be sharable because constructs supported by one language (e.g., temporal constructs) cannot be expressed in another language. While flexibility is an admirable goal, more consideration should be given to the long-term consequences of allowing different expression languages if the goal of making guidelines sharable is to be achieved.
With probabilistic criteria, the dependence probabilities among all variables in a specification have to be known in order for the criterion to be evaluated. The problem is that the number of conditional probabilities that need to be specified increases exponentially as the number of variables used in a specification increase, and the dependence relationships among some variables may not be known. Users of probabilistic criteria need to provide the conditional probabilities needed for evaluating these criteria, and this means that use will probably be limited to **advanced** users of GLIF.
Evaluation of the structured string in an expression specification is performed within the parser for the grammar corresponding to Expression.encoding_language. Evaluation occurs after the entire string has been parsed. A table/database that stores variable names and values is needed in order to perform the evaluations. For probabilistic criteria, this table/database would have to include the conditional probabilities needed for evaluating the expression.
Partial specification of GLIF_ARDEN grammar in Backus Naur Form (BNF): (The grammar has been expanded to include almost all of the operators that Arden Syntax allows – 01/06/2000) A modification of the Arden syntax is used, and not Arden syntax itself, since we wanted to be able to express certain operators, which are not supported by Arden syntax, but are needed for specifying criteria. These operators are "is a", "overlaps", "xor", "from now", "is unknown" and "at least k of ...". GLIF made minor changes to the syntax of some operators to allow a more natural way of expressing them; the operator "occur/occurs/occurred at" has the same meaning as "occur/occurs/occurred equals" and the operators "in", "not in" has the same meaning as Arden's "is in", "is not in". All of these operators, including the original Arden syntax format, can be used. All of Arden's logic grammar is supported by GLIF.

Expression: LogicalExpression | Function | WhereExpression | TemporalExpression | (Expression) | Literal | NumericalExpression | StringExpression | ListExpression

LogicalExpression: UnaryOperator Expression | Expression UnaryOperator | Expression BinaryOperator Expression | at least Number of LogicalExprList | UnaryOperator LogicalExprList | TernaryOpExpression

Function: Identifier (ExpressionList)

WhereExpression : DEFINITION TO COME

TemporalExpression: TimeLiteral | Duration | TimeInterval | Frequency | FrequencySet | Duration UnaryOperator | UnaryOperator Number of Identifier | UnaryOperator Identifier | TemporalExpression BinaryOperator TemporalExpression | Identifier BinaryOperator TemporalExpression | TemporalExpression BinaryOperator Identifier | TimeInterval lasting Duration to Duration

Literal: Identifier | Number | StringLiteral | Tri-StateValue | ListLiteral | TimeLiteral

NumericalExpression: Number | UnaryOperator Expression | Expression BinaryOperator Expression

StringExpression: DEFINITION TO COME

ListExpression: FULL DEFINITION TO COME (since GLIF will support lists) Definitions will include: last (Number, etc) from (ListLiteral or Identifier) | first (Number, etc) from (ListLiteral or Identifier) | latest (Number, etc) from (ListLiteral or Identifier) | earliest (Number, etc) from (ListLiteral or Identifier) | nearest (Number, etc) from (ListLiteral or Identifier)

ExpressionList: -nothing- | Expression | Expression , ExpressionList

LogicalExprList: LogicalExpression | LogicalExpression, LogicalExprList

Duration: Number UnaryOperator

Frequency: EveryFrequency | TimesFrequency | Frequency for Duration

EveryFrequency: every Duration | every Duration with window -Duration, +Duration, offset -Duration, +Duration

TimesFrequency: Number times a TimeUnit | Number times | Number times per Duration

FrequencySet: Frequency followed by FrequencySet | Frequency

TimeUnit: years | year | months | month | weeks | week | days | day | hours | hour | minutes | minute | seconds | second

TimeInterval: TimeLiteral to TimeLiteral | Identifier to Identifier

UnaryOperator: not | ! | - | ago | from now | IsAreWasWere null | IsAreWasWere not present | IsAreWasWere not null | IsAreWasWere present | IsAreWasWere boolean | IsAreWasWere not boolean | IsAreWasWere unknown | IsAreWasWere number | IsAreWasWere not number | IsAreWasWere time | IsAreWasWere not time | IsAreWasWere duration | IsAreWasWere not duration | IsAreWasWere string| IsAreWasWere not string | IsAreWasWere list| IsAreWasWere not list | years | year | year of | months | month | month of | weeks | week | week of | days | day | day of | hours | hour | hour of | minutes | minute | minute of | seconds | second |second of | exists | exist | exists of | exist of | any of | all of | last | last of | first | first of | latest | latest of |earliest | earliest of | time of |extract month

BinaryOperator: , | or | | | xor | *| | & | and | == | = | eq | IsAreWasWere equal | != | <> | ne | IsAreWasWere not equal | < | lt | IsAreWasWere less than | IsAreWasWere not greater than or equal | <= | le | IsAreWasWere less than or equal | IsAreWasWere not greater than | > | gt | IsAreWasWere greater than | IsAreWasWere not less than or equal | >= | ge | IsAreWasWere greater than or equal | IsAreWasWere not less than | IsAreWasWere before | IsAreWasWere not before | IsAreWasWere after | IsAreWasWere not after | OccurOccursOccurred at | OccurOccursOccurred equal | OccurOccursOccurred before | OccurOccursOccurred not before | OccurOccursOccurred after | OccurOccursOccurred not after | OccurOccursOccurred at | OccurOccursOccurred equal | in | IsAreWasWere in | not in | IsAreWasWere not in | + | - | * | / | | ** | ^ | is a | is-a | before | after | | matches pattern

IsAreWasWere: is | are | was | were

OccurOccursOccurred: occur | occurs | occurred

TernaryOpExpression: Expression IsAreWasWere within Expression EndTernExpr | Expression IsAreWasWere not within Expression EndTernExpr | Expression OccurOccursOccurred within Expression EndTernExpr | Expression OccurOccursOccurred not within Expression EndTernExpr

EndTernExpr: to Expression | preceding Expression | following Expression | surrounding Expression

Identifier: [A-Za-z0-9_]+

Number: [0-9]+.?[0-9]* UnitSpec

StringLiteral: "[^"]*"

Tri-StateValue: true | false | unknown

ListLiteral: [ExpressionList]

TimeLiteral: year-month-dayThours:minutes:seconds.fraction(Z|+/-hours) [0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9](T[0-9][0-9](:[0-9][0-9] (:[0-9][0-9](.[0-9]+)?(Z|[+-][0-9][0-9]:[0-9][0-9])?)?)?)?

UnitSpec: -nothing- | Identifier

Operators that exist in the Arden Syntax but not in GLIF Arden:

Unary: sort, count [of], avg [of], median [of], sum[of], stddev [of], variance [of], no [of], slope [of], min [of], minimum [of], max [of], maximum [of], increase [of], decrease [of], percent increase [of], percent decrease [of], interval [of], arccos [of], arcsin [of], arctan [of], cos [of], cosine [of], sin [of], sine [of], tan [of], tangent [of], exp [of] (raises its argument to mathematical e), ceiling [of], truncate [of], log [of], log10 [of], abs [of], abs [of], sqrt [of], reverse [of], list [of], string [of], extract characters

Binary: merge, || , format, round, min...from, minimum...from, max...from, maximum...from, seqto

Operators that exist in GLIF_Arden but not in Arden Syntax:

Unary: from now, is unknown

Binary: in, not in (same as Arden's is in, is not in), overlaps, xor, |* , is a, is-a, occur/occurs/occurred at, at least...of

Attributes

didactics
encoding_language
specification

Attribute Detail

didactics

Data type: Supplemental_Material
Multiplicity : 0..*
Description : supplemental information about the expression
Level: A, B, C

encoding_language

specification

Data type: string
Multiplicity: 0:1
Description:a structured string written in GLIF_Arden that specifies the expression
Level: B, C

Class *Expression_Package.List_Expression*

Inherits from:

Expression_Package.Expression

List_Expression: an expression involving comma-separated strings, numbers, variable names, or other literals

Class *Expression_Package.Numerical_Expression*

Inherits from:

Expression_Package.Expression

Class *Expression_Package.Probabilistic_Criterion*

Inherits from:

Expression_Package.Criterion

To add conditional probabilities or Bayesian network info

Class *Expression_Package.String_Expression*

Inherits from:

Expression_Package.Expression

String_Expression: an expression involving strings

Class *Expression_Package.Three_Valued_Criterion*

Inherits from:

Expression_Package.Criterion

Interface Detail

Interface *Expression_Package.Expression_Language*

Attributes

GLIF_Arden
Other

SQL3

Attribute Detail

GLIF_Arden

Other

SQL3

Interface *Expression_Package.Three_Valued_Type*

Attributes

false_value
true_value
unknown_value

Attribute Detail

false_value

true_value

unknown_value

Example:

(smoker and chronic cough) Three_Valued_Criterion

name: three-valued criterion that assesses whether a patient is a smoker and has a chronic cough encoding_language: GLIF_Arden specification: (smoker=true) AND (chronic_cough=true) didactics

(smoker and chronic cough) Probabilistic_Criterion

name: probabilistic criterion that assesses whether a patient is a smoker and has a chronic cough encoding_language: GLIF_Arden specification: (smoker=true) AND (chronic_cough=true) didactics

The three-valued_criterion specification: (smoker = true) AND (chronic_cough = true) evaluates to true if a patient is a smoker and has a chronic cough. It evaluates to false if the patient is not a smoker, or does not have a chronic cough, or both. It evaluates to unknown if it is unclear that the patient is a smoker or has a chronic cough, or both.
As a specific example, if we don't know whether or not the patient is a smoker and we know that the patient has a chronic cough, the criterion evaluates to unknown.
To evaluate the probabilistic_criterion specification: (smoker = true) AND (chronic_cough = true), we need to know the probability that a patient is a smoker, P(smoker=true), the probability that a patient has a chronic cough, P(chronic_cough=true), and the probability that a patient has a chronic cough given that the patient is a smoker, P(chronic_cough=true| smoker=true). This last probability expresses the dependence relationship between a patient being a smoker and the patient having a chronic cough. Given these three probabilities, the specification evaluates to a real number in the interval [0,1]: P((smoker = true) AND (chronic_cough = true)) = P(smoker=true)* P(chronic_cough=true| smoker=true)
If there is a 70% probability that the patient is a smoker and we know that there is a 95% probability of a patient having a chronic cough given that s/he is a smoker, then, P((smoker = true) AND (chronic_cough = true)) = .7 * .95 = 0.665