# ClearSpeak Rules and Preferences

Beth Brownstein, Educational Testing Service

Neil Soiffer, Design Science, Inc.

Lois Frankel, Educational Testing Service

**Acknowledgments**

Development of this material and of the updates to MathType™ and MathPlayer™ was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R324A110355 to the Educational Testing Service. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

Copyright © 2015 by Educational Testing Service.

ETS and the ETS logo are registered trademarks of Educational Testing Service (ETS).

MEASURING THE POWER OF LEARNING is a trademark of ETS.

All other trademarks are the property of their respective owners.

Table of Contents

[ClearSpeak Rules and Preferences 1](#_Toc439173200)

[Part 1: Essentials 10](#_Toc439173201)

[Introduction 10](#_Toc439173202)

[Capital Letters 11](#_Toc439173203)

[Absolute Value (Including Determinants) 12](#_Toc439173204)

[Fractions (including Mixed Numbers) 13](#_Toc439173205)

[Exponents 17](#_Toc439173206)

[Roots 20](#_Toc439173207)

[Functions 21](#_Toc439173208)

[Trigonometry (includes basic trig functions, inverse trig functions, and hyperbolic trig functions) 24](#_Toc439173209)

[Logarithms 27](#_Toc439173210)

[Implied Times 29](#_Toc439173211)

[Parentheses and Brackets 31](#_Toc439173212)

[Matrices, Vectors, and Combinatorics 33](#_Toc439173213)

[Multiline Entries 39](#_Toc439173214)

[Sets enclosed in set brackets { } 45](#_Toc439173215)

[Named Sets (Number Systems) 47](#_Toc439173216)

[Part 2: Symbols 48](#_Toc439173217)

[Multiplication Symbol  48](#_Toc439173218)

[Multiplication Symbol  48](#_Toc439173219)

[Triangle Symbol:  49](#_Toc439173220)

[Ellipses  49](#_Toc439173221)

[Vertical Line  50](#_Toc439173222)

[Set Membership Symbols:  and  51](#_Toc439173223)

[Sums, Products, Unions, Intersections, and Integrals  54](#_Toc439173224)

[Part 3: Adornments 57](#_Toc439173225)

[Prime, Double Prime, and Triple Prime 57](#_Toc439173226)

[Combinations and Permutations:  and  59](#_Toc439173227)

[Bar 59](#_Toc439173228)

[Equals Sign with Adornments 61](#_Toc439173229)

Detailed Table of Contents

[ClearSpeak Rules and Preferences 1](#_Toc439173240)

[Part 1: Essentials 10](#_Toc439173241)

[Introduction 10](#_Toc439173242)

[Definition: Simple Expression 11](#_Toc439173243)

[Capital Letters 11](#_Toc439173244)

[Rule: (Caps\_Auto.eqp) 11](#_Toc439173245)

[Preferences: 12](#_Toc439173246)

[SayCaps (Caps\_SayCaps.eqp) 12](#_Toc439173247)

[Absolute Value (Including Determinants) 12](#_Toc439173248)

[Rule: (AbsoluteValue\_Auto.eqp) 12](#_Toc439173249)

[Preferences: 12](#_Toc439173250)

[AbsEnd (AbsoluteValue\_AbsEnd.eqp) 12](#_Toc439173251)

[Cardinality (AbsoluteValue\_Cardinality.eqp) 13](#_Toc439173252)

[Determinant (AbsoluteValue\_Determinant.eqp) 13](#_Toc439173253)

[Fractions (including Mixed Numbers) 13](#_Toc439173254)

[Rule: (Fraction\_Auto.eqp) 13](#_Toc439173255)

[Preferences: 15](#_Toc439173256)

[Ordinal (Fraction\_Ordinal.eqp) 15](#_Toc439173257)

[Over (Fraction\_Over.eqp) 15](#_Toc439173258)

[FracOver (Fraction\_FracOver.eqp) 15](#_Toc439173259)

[General (Fraction\_General.eqp) 15](#_Toc439173260)

[EndFrac (FractionEndFrac.eqp) 15](#_Toc439173261)

[GeneralEndFrac (Fraction\_GeneralEndFrac.eqp) 16](#_Toc439173262)

[OverEndFrac (Fraction\_OverEndFrac.eqp) 16](#_Toc439173263)

[Per (Fraction\_Per.eqp) 17](#_Toc439173264)

[Exponents 17](#_Toc439173265)

[Rule: (Exponent\_Auto.eqp) 17](#_Toc439173266)

[Preferences: 18](#_Toc439173267)

[Ordinal (Exponent\_ Ordinal.eqp) 18](#_Toc439173268)

[OrdinalPower (Exponent\_ OrdinalPower.eqp) 19](#_Toc439173269)

[AfterPower (Exponent\_AfterPower.eqp) 19](#_Toc439173270)

[Roots 20](#_Toc439173271)

[Rule: (Roots\_Auto.eqp) 20](#_Toc439173272)

[Preferences: 20](#_Toc439173273)

[PosNegSqRoot (Roots\_PosNegSqRoot.eqp) 20](#_Toc439173274)

[RootEnd (Roots\_RootEnd.eqp) 21](#_Toc439173275)

[PosNegSqRootEnd (Roots\_PosNegSqRootEnd.eqp) 21](#_Toc439173276)

[Functions 21](#_Toc439173277)

[Rule: (Functions\_Auto.eqp) 22](#_Toc439173278)

[None (Functions\_None.eqp) 23](#_Toc439173279)

[Trigonometry (includes basic trig functions, inverse trig functions, and hyperbolic trig functions) 24](#_Toc439173280)

[Rule: (Trig\_Auto.eqp) 24](#_Toc439173281)

[Preferences: 26](#_Toc439173282)

[TrigInverse (Trig\_TrigInverse.eqp) 26](#_Toc439173283)

[ArcTrig (Trig\_ArcTrig.eqp) 27](#_Toc439173284)

[Logarithms 27](#_Toc439173285)

[Rule: (Log\_Auto.eqp) 27](#_Toc439173286)

[Preference: 28](#_Toc439173287)

[LnAsNaturalLog (Log\_LnAsNaturalLog.eqp) 28](#_Toc439173288)

[Implied Times 29](#_Toc439173289)

[Rule: (ImpliedTimes\_Auto.eqp) 29](#_Toc439173290)

[Preferences: 30](#_Toc439173291)

[MoreImpliedTimes (ImpliedTimes\_MoreImpliedTimes.eqp) 30](#_Toc439173292)

[None (ImpliedTimes\_None.eqp) 30](#_Toc439173293)

[Parentheses and Brackets 31](#_Toc439173294)

[Rule: (Paren\_Auto.eqp) 31](#_Toc439173295)

[Speak (Paren\_Speak.eqp) 32](#_Toc439173296)

[SpeakNestingLevel (Paren\_SpeakNestingLevel.eqp) 32](#_Toc439173297)

[Silent (Paren\_Silent.eqp) 32](#_Toc439173298)

[CoordPoint (Paren\_CoordPoint.eqp) 32](#_Toc439173299)

[Interval (Paren\_Interval.eqp) 33](#_Toc439173300)

[Matrices, Vectors, and Combinatorics 33](#_Toc439173301)

[Rule: (Matrix\_Auto.eqp) 33](#_Toc439173302)

[Preferences: 35](#_Toc439173303)

[SpeakColNum (Matrix\_SpeakColNum.eqp) 35](#_Toc439173304)

[SilentColNum (Matrix\_SilentColNum.eqp) 36](#_Toc439173305)

[EndMatrix (Matrix\_EndMatrix.eqp) 37](#_Toc439173306)

[Vector (Matrix\_Vector.eqp) 37](#_Toc439173307)

[EndVector (Matrix\_EndVector.eqp) 38](#_Toc439173308)

[Combinatorics (Matrix\_Combinatoric.eqp) 38](#_Toc439173309)

[Multiline Entries 39](#_Toc439173310)

[Multiline Overview Rule: (MultiLineOverview\_Auto.eqp) 39](#_Toc439173311)

[Preference: 39](#_Toc439173312)

[None (MultiLineOverview\_None.eqp) 39](#_Toc439173313)

[Multiline Label Rule: (MultiLineLabel\_Auto.eqp) 39](#_Toc439173314)

[Preferences: 40](#_Toc439173315)

[Case (MultiLineLabel\_Case.eqp) 40](#_Toc439173316)

[Equation (MultiLineLabel\_Equation.eqp) 41](#_Toc439173317)

[Line (MultiLineLabel\_Line.eqp) 42](#_Toc439173318)

[Row (MultiLineLabel\_Row.eqp) 42](#_Toc439173319)

[Step (MultiLineLabel\_Step.eqp) 43](#_Toc439173320)

[Constraint (MultiLineLabel\_Constraint.eqp) 43](#_Toc439173321)

[None (MultiLineLabel\_None.eqp) 44](#_Toc439173322)

[Multi-Line Pauses Between Columns Rule: (MultiLinePausesBetweenColumns\_Auto.eqp) 44](#_Toc439173323)

[Preferences: 44](#_Toc439173324)

[Long (MultiLinePausesBetweenColumns\_Long.eqp) 44](#_Toc439173325)

[Short (MultiLineEntryMultiColumn\_Short.eqp) 45](#_Toc439173326)

[Sets enclosed in set brackets { } 45](#_Toc439173327)

[Rule: (Sets\_Auto.eqp) 45](#_Toc439173328)

[Preferences: 45](#_Toc439173329)

[woAll (Sets\_woAll.eqp) 45](#_Toc439173330)

[SilentBracket (Sets\_SilentBracket.eqp) 46](#_Toc439173331)

[Named Sets (Number Systems) 47](#_Toc439173332)

[Rule: 47](#_Toc439173333)

[Preferences: 47](#_Toc439173334)

[Part 2: Symbols 48](#_Toc439173335)

[Multiplication Symbol  48](#_Toc439173336)

[Rule: (MultSymbolX\_Auto.eqp) 48](#_Toc439173337)

[Preference: 48](#_Toc439173338)

[By (MultSymbolX\_By.eqp) 48](#_Toc439173339)

[By (MultSymbolX\_Cross.eqp) 48](#_Toc439173340)

[Multiplication Symbol  48](#_Toc439173341)

[Rule: (MultSymbolDot\_Auto.eqp) 48](#_Toc439173342)

[Preference: 48](#_Toc439173343)

[By (MultSymbolDot\_Dot.eqp) 48](#_Toc439173344)

[Triangle Symbol:  49](#_Toc439173345)

[Rule: (TriangleSymbol.Auto.eqp) 49](#_Toc439173346)

[Preference: 49](#_Toc439173347)

[Delta (TriangleSymbol\_Delta.eqp): 49](#_Toc439173348)

[Ellipses  49](#_Toc439173349)

[Rule: (Ellipses\_Auto.eqp) 49](#_Toc439173350)

[Preference: 49](#_Toc439173351)

[AndSoOn (Ellipses\_AndSoOn.eqp) 49](#_Toc439173352)

[Vertical Line  50](#_Toc439173353)

[Rule: (VerticalLine\_Auto.eqp) 50](#_Toc439173354)

[Preference: 50](#_Toc439173355)

[SuchThat (SingleVerticalLine\_SuchThat.eqp) 50](#_Toc439173356)

[Divides (SingleVerticalLine\_Divides.eqp) 50](#_Toc439173357)

[Given (SingleVerticalLine\_Given.eqp) 51](#_Toc439173358)

[Set Membership Symbols:  and  51](#_Toc439173359)

[Rule: (SetMemberSymbol\_Auto.eqp) 51](#_Toc439173360)

[Preferences: 52](#_Toc439173361)

[Member (SetMemberSymbol\_Member.eqp) 52](#_Toc439173362)

[Element (SetMemberSymbol\_Element.eqp) 52](#_Toc439173363)

[In (SetMemberSymbol\_In.eqp) 53](#_Toc439173364)

[Belongs (SetMemberSymbol\_Belongs.eqp) 53](#_Toc439173365)

[Sums, Products, Unions, Intersections, and Integrals  54](#_Toc439173366)

[Rule: 54](#_Toc439173367)

[Part 3: Adornments 57](#_Toc439173368)

[Prime, Double Prime, and Triple Prime 57](#_Toc439173369)

[Rule: (Prime\_Auto.eqp) 57](#_Toc439173370)

[Preferences: 58](#_Toc439173371)

[Length (Prime\_Length.eqp) 58](#_Toc439173372)

[Angle (Prime\_Angle.eqp) 59](#_Toc439173373)

[Combinations and Permutations:  and  59](#_Toc439173374)

[Rule: (CombinationPermutation\_Auto.eqp) 59](#_Toc439173375)

[Preference: (CombinationPermutation\_ChoosePermute.eqp) 59](#_Toc439173376)

[Bar 59](#_Toc439173377)

[Rule: (Bar\_Auto.eqp) 59](#_Toc439173378)

[Preference: 60](#_Toc439173379)

[Bar (Bar\_Bar.eqp) 60](#_Toc439173380)

[Conjugate (Bar\_Conjugate.eqp) 60](#_Toc439173381)

[Mean (Bar\_Mean.eqp) 61](#_Toc439173382)

[Equals Sign with Adornments 61](#_Toc439173383)

[Rule: 61](#_Toc439173384)

**Acknowledgments**

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R324A110355 to the Educational Testing Service. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

# Part 1: Essentials

## Introduction

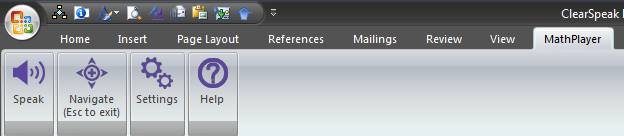
This document is intended for individuals who need to author math expressions for use with the synthetic speech system ClearSpeak and for those interested in implementing the speech rules documented here in other environments. It contains a description of the ways in which ClearSpeak speaks various math structures (e.g., fractions, exponents, and roots). For each structure, a set of rules and preferences is listed. The speech rules are the standard or default way in which ClearSpeak will speak the structure, and the preferences are other ways in which ClearSpeak can be made to *automatically* speak the structure.

When documents are authored using the ClearSpeak speech system, .eqp files are attached to each math expression to make it speak according to the desired rules and preferences. For each rule and preference listed in this document, the name of the associated .eqp file is given. For more information about how to author documents using the ClearSpeak speech style, see the *ClearSpeak Authoring Tutorial* available at <http://www.ClearSpeak.org>.

To illustrate what the rules and preferences sound like, each rule and preference in this document is followed by a small number of sample expressions that speak according to that rule or preference. More examples can be found in examples documents posted on <http://www.ClearSpeak.org>.

Readers who have MathType and MathPlayer installed can listen to the speech for the math expressions in this document. To hear the speech for a particular expression, click on the expression and then use the Speak button on the MathPlayer toolbar. Users of the NVDA screen reader can listen to the speech without using the toolbar (but still need MathType and MathPlayer installed). Keep in mind that since the number of sample expressions given is small, the examples for any particular rule or preference may not cover all possible cases. In particular, a preference for a particular structure may differ from the rules for that structure only in prescribed cases. For these preferences, the sample expressions focus on illustrating how these prescribed cases will speak; they will not include samples of expressions that will continue to speak according to the associated rule.

To listen to an expression, select the MathPlayer menu on the Word menu bar, as shown below, and then click the Speak button.



### Definition: Simple Expression

In the rules and preferences, an expression is called *simple* if it is one of the following.

* + A number that is an integer, a decimal, or a fraction that is spoken as an ordinal\*
  + A letter, two juxtaposed letters (e.g., *x*, *y*, *z*, *xy*, *yz*, etc.), the negative of a letter, or the negative of two juxtaposed letters (e.g., , , , , , etc.)
  + An integer, decimal, letter, or the negative of a letter that is followed by the degree sign (e.g., , ,  , )
  + A number that is an integer, a decimal, or a fraction that is spoken as an ordinal\* and is followed by a letter or pair of juxtaposed letters (e.g., 2*x*, , 4.1*z*, 2*xy*, )
  + A function (including trigonometric and logarithmic functions) with an argument that is a simple expression (e.g., , , )  
     

\* either by the **Fraction\_Ordinal** preference or by the default fraction rules

Note: The fraction  spoken as an ordinal is “one-third”, and the fraction  spoken as an ordinal is “three-fourths”.

## Capital Letters

### Rule: (Caps\_Auto.eqp)

* **All of the letters in an expression are caps, or all of the letters in an expression are lowercase.**

***Speech*:** The speech for capital and lowercase letters is the same; that is, just the letter is spoken (e.g., “B” and “b” are both spoken as “b”). In order to make a distinction between uppercase and lowercase letters, capital letters are spoken in a pitch that is 30 percent higher than the pitch at which lowercase letters are spoken.

Sample expressions:  and 

### Preferences:

#### SayCaps (Caps\_SayCaps.eqp)

***Speech*:** Capital letters speak as Cap [letter], while lowercase letters just speak the letter (e.g., “B” speaks as “Cap *b*”; “*b*” speaks as “*b*”; “bc” speaks as “b c”; and “*BC*” speaks as “Cap *b* Cap *c*”).

Sample expressions:  and

## Absolute Value (Including Determinants)

### Rule: (AbsoluteValue\_Auto.eqp)

* **All expressions in absolute value signs, except matrices**

***Speech*:** The absolute value of [expression]

Sample expressions:  and 

* **Matrices in absolute value signs (determinants)**

***Speech*:** The determinant of [matrix]\*

Sample expression:

**Note:** Speech for matrices is covered under Matrices, Vectors, and Combinatorics (see p. 33).

### Preferences:

#### AbsEnd (AbsoluteValue\_AbsEnd.eqp)

* **All expressions in absolute value signs, except matrices**

***Speech:*** The absolute value of [expression] end absolute value

Sample expressions:  and 

* **Matrices in absolute value signs (determinants)**

***Speech:*** The determinant of [matrix]

Sample expression: 

Note: If the preference Matrix\_EndMatrix.eqp is set, determinants speak as “The determinant of [matrix spoken according to the matrix rules] end determinant.”

Sample expression: 

#### Cardinality (AbsoluteValue\_Cardinality.eqp)

***Speech*:** The cardinality of [expression]

Sample expression: 

#### Determinant (AbsoluteValue\_Determinant.eqp)

***Speech*:** The determinant of [expression]

Sample expression: 

## Fractions (including Mixed Numbers)

### Rule: (Fraction\_Auto.eqp)

* **Common Fractions**

Fractions with integer numerators from 1 to 19 and integer denominators from 1 to 10 speak as common fractions (e.g., -halves, -thirds, -fourths).

Sample expressions:  and 

* **Simple Fractions**

Simple fractions are fractions that have a [simple expression](#_Definition:_Simple_Expression) (see p. 11), a common fraction, or the negative of a common fraction in the numerator and in the denominator but do not meet the Common Fractions criteria.

***Speech:*** [numerator] over [denominator]

Sample expressions: , , , and 

* **Fractions where the numerator and/or the denominator is an expression that is not a** [simple expression](#_Definition:_Simple_Expression) **(see p. 11)**

***Speech:*** the fraction with numerator [numerator] and denominator [denominator]

Sample expressions:  and 

* **Mixed Numbers**

***Speech:*** [whole number portion] and [fractional portion]

Sample expression: 

* **Fractions with text (or a number followed by text) in the numerator and text (or a number followed by text) in the denominator**

***Speech:*** [numerator] over [denominator]

Sample expressions:  and 

* **Nested Fractions**
* Both the numerator and the denominator of a fraction are [simple expressions](#_Definition:_Simple_Expression)   
  (see p. 11), one or both of which is either
  + - a fraction spoken as an ordinal\* or
    - a fraction spoken as an ordinal\* and followed by an indicator or a pair of juxtaposed indicators.

\* either by ordinal preference or by the default rules

Note: The fraction  spoken as an ordinal is “one-third” and the fraction  spoken as an ordinal is “three-fourths”.

***Speech:*** the fraction [numerator] over [denominator]

Sample expressions: and 

* All other nested fractions

***Speech:*** the fraction with numerator [numerator] and denominator [denominator]

Sample expressions:  and 

### Preferences:

#### Ordinal (Fraction\_Ordinal.eqp)

* Fractions with a positive integer in the numerator and a positive integer in the denominator speak the denominator as an ordinal (e.g., one-half, three-fourths, five-sixths, etc.).

Sample expressions:  and 

* All other fractions speak according to the rules.

Sample expressions:, , and 

#### Over (Fraction\_Over.eqp)

***Speech:*** [numerator] over [denominator]

Sample expressions:  and 

#### FracOver (Fraction\_FracOver.eqp)

***Speech:*** the fraction [numerator] over [denominator]

Sample expressions:  and 

#### General (Fraction\_General.eqp)

***Speech:*** The fraction with numerator [numerator] and denominator [denominator]

Sample expressions:  and 

#### EndFrac (FractionEndFrac.eqp)

Speech is the same as the default rules but with “end fraction” added in the following cases.

* **Simple Fractions**

Sample expressions: , , and 

* **Fractions where the numerator and/or the denominator is an expression that is not a** [simple expression](#_Definition:_Simple_Expression) **(see p. 11)**

Sample expression: 

* **Fractions containing functions**

Sample expressions: and 

* **Nested fractions that the rules speak as “the fraction with numerator [numerator] and denominator [denominator]”** (“end fraction” added, but only at the very end of the nested fraction)

Sample expression: 

Note: Nested fractions with a numerator and denominator that speak as ordinals (such as the nested fraction ) do not have “end fraction” added.

Sample expression:

#### GeneralEndFrac (Fraction\_GeneralEndFrac.eqp)

***Speech:*** The fraction with numerator [numerator] and denominator [denominator] end fraction

Sample expressions:  and 

#### OverEndFrac (Fraction\_OverEndFrac.eqp)

***Speech:*** [numerator] over [denominator] end fraction

Sample expressions:  and 

#### Per (Fraction\_Per.eqp)

***Speech:*** [numerator] per [denominator]

Sample expressions:  and 

## Exponents

### Rule: (Exponent\_Auto.eqp)

* **Exponent 2**

***Speech:*** [base] squared

Sample expression: 

* **Exponent 3**

***Speech:*** [base] cubed

Sample expression: 

* **All other positive integer exponents and exponents that are single letters**

***Speech:*** [base] to the [exponent–ordinal] power

Sample expressions: , , , and 

* **Exponent 0 and all negative integer exponents**

***Speech:*** [base] to the [exponent] power

Sample expressions:  and 

* **All other exponents (including integers with a decimal point, such as 2.0)**

***Speech:*** [base] raised to the [exponent] power

Sample expressions:  and 

* **Nested exponents (expressions containing exponents that contain exponents)**
* Exponent is of the form , , , or , where the entry is one of the following:
  + - A number (integer or decimal)
    - A fraction spoken as ordinal\*
    - A variable

***Speech***: [base] raised to the [exponent] power

Sample expressions: and 

* Exponent is of the form , , , or , where the entry is one of the following:
  + - A number (integer or decimal)
    - A fraction spoken as ordinal\*

\*either by fraction\_ordinal preference or by the default rules

***Speech***: [base] raised to the [exponent] power

Sample expressions:  and 

* Exponent is of any other form

***Speech:*** [base] raised to the exponent [exponent] power, end exponent

Sample expressions: and 

Note: The fraction  spoken as an ordinal is “one-third”, and the fraction  spoken as an ordinal is “three-fourths”.

Note: If the exponent in the exponent is a 2, the 2 speaks as “squared”. If the exponent in the exponent is a 3, the 3 speaks as “cubed”.

### Preferences:

#### Ordinal (Exponent\_ Ordinal.eqp)

* **All positive integer exponents and exponents that are single letters**

***Speech:*** [base] to the [exponent-ordinal]

Sample expressions: , , and 

* **Exponent 0 and all negative integer exponents**

***Speech:*** [base] to the [exponent]

Sample expressions:  and 

* **All other exponents speak according to the rules.**

#### OrdinalPower (Exponent\_ OrdinalPower.eqp)

* **All positive integer exponents and exponents that are single letters**

***Speech:*** [base] to the [exponent-ordinal] power

Sample expressions: , , and 

* **Exponent 0 and all negative integer exponents**

***Speech:*** [base] to the [exponent] power

Sample expressions:  and 

* **All other exponents speak according to the rules.**

Sample expressions:  and 

#### AfterPower (Exponent\_AfterPower.eqp)

* **Exponents that are not nested**

***Speech:*** [base] raised to the power [exponent]

Sample expressions:  and 

* **Nested exponents**

***Speech:*** [base] raised to the exponent [exponent] power, end exponent

Sample expressions: and 

## Roots

### Rule: (Roots\_Auto.eqp)

* **Square roots**

***Speech:*** the square root of [expression under square root sign]

Sample expressions:  and 

* **Cube roots**

***Speech:*** the cube root of [expression under square root sign]

Sample expressions:  and 

* **Higher roots with numeric index**

***Speech:*** the [index ordinal] root of [expression under root sign]

Sample expressions:  and 

* **Higher roots with letter index**

***Speech:*** the [index ordinal] root of [expression under root sign]

Sample expressions:  and 

### Preferences:

#### PosNegSqRoot (Roots\_PosNegSqRoot.eqp)

* 

***Speech:*** The positive square root of [expression under root sign]

Sample expressions:  and 

* 

***Speech:*** The negative square root of [expression under root sign]

Sample expressions:  and 

* **The speech for higher-order roots follows the rules.**

#### RootEnd (Roots\_RootEnd.eqp)

***Speech:*** As indicated in the rules, but “end root” is spoken after the expression under the root sign.

Sample expressions:  and 

#### PosNegSqRootEnd (Roots\_PosNegSqRootEnd.eqp)

***Speech:*** As indicated in the PosNegSqRoot preference, but “end root” is spoken after the expression under the square root sign.

Sample expressions:  and 

## Functions

***Some Preliminary Remarks***

Functions have the form ***name*(*argument*)** or ***name*[*argument*]**. For example, in the function , the function name is *f* and the argument of the function is *x*. Likewise, in the function , the function name is *g* and the argument of the function is *x*, *y*. The name of a function can be a simple identifier (e.g., *h* or ), or it can be a simple identifier that is adorned in some way (e.g., ). Speech for some of the possible adornments is covered in [Part 3: Adornments](#_Part_3:_Adornments) (see p. 57).

However, expressions of the form ***name*(*argument*)** or ***name*[*argument*]** are not necessarily functions. For example,  could represent the function “*a* of 0” or the expression “*a* times 0”. This meant that if the Function rule (Functions\_Auto.eqp) is set and ClearSpeak encounters an expression of the form ***name*(*argument*)** or ***name*[*argument*]**, the first thing ClearSpeak has to do is decide whether to recognize the expression as a function. Therefore, the rule begins with the criteria that ClearSpeak uses to make that decision. The rule then covers speech for the function ***name***. Speech for the parentheses (or brackets) that enclose the argument follows the Parentheses and Brackets rule or preference that is set. Speech for parentheses and brackets is covered in [Parentheses and Brackets](#_Parentheses_and_Brackets) (see p. 31).

Note: Speech for trigonometric functions is covered under [Trigonometry](#_Trigonometry_(includes_basic) (see p. 24), and speech for logarithms is covered under [Logarithms](#_Logarithms) (see p. 27).

### Rule: (Functions\_Auto.eqp)

**Criteria under which expressions of the form name(argument) or name[argument] are recognized as functions**

* The ***name*** is *f*, *g*, *h, F*, *G*, *H,* or any of these letters with adornments. There are no restrictions on the arguments in the expression. For example, , , and  are recognized as functions.
* The ***name*** is an identifier (<mi>) or an identifier with adornments, and the argument is a number or identifier. For example,  and  are recognized as functions.
* The ***name*** is an identifier (<mi>) or an identifier with adornments, and the argument is a comma-separated list. For example,  and  are recognized as functions.
* If two names are recognized as functions, then so are some basic functions built from those names. Here is an illustrative list of those types of functions using *f* and *g* as the two function names.
  + 

* + 
  + 
  + 
  + 
  + 
  + , , and 

**Speech for functions**

Speech for various function types is presented using one or more instances of the function type to illustrate the speech pattern for functions of that type. In the examples below, the speech for each of the function arguments is set to follow the Parentheses and Brackets rule (Paren\_Auto.eqp).

The speech for parentheses and brackets is covered in [Parentheses and Brackets](#_Parentheses_(and_Brackets)) (see p. 31).

* **Functions (including multi-variable functions)**

 ***Speech:*** *f* of *x*

 ***Speech:*** *f* of, open paren, *x* comma *y*, close paren

 ***Speech:*** *f* of, open paren, *x* comma *y* comma z, close paren

* **Inverse functions**

 ***Speech:*** *f* inverse of *x*

* **Powers of functions**

 ***Speech:*** *f* squared of *x*

 ***Speech:*** *f* cubed of *x*

 ***Speech:***  the fourth power of *f* of *x*

 ***Speech:*** the *n*th power of *f* of *x*

* **Composition of Functions**

 ***Speech:*** *f* of, *g* of *x*

 ***Speech:*** *f* of, *g* of, *h* of *x*

#### None (Functions\_None.eqp)

If this preference is set, the criteria by which ClearSpeak recognizes functions (covered in Part 1 of the Functions rule) are turned off. For example, if this preference is set and implied times is spoken, then *a*(0) will speak as “*a* times 0”.

Sample expressions:  and 

Note: If this preference is set, speech for trigonometric functions and logarithms will still follow the rule or preference set for those functions. Speech for trigonometric functions is covered under [Trigonometry](#_Trigonometry_(includes_basic) (see p. 24), and speech for logarithms is covered under [Logarithms](#_Logarithms) (see p. 27).

## Trigonometry (includes basic trig functions, inverse trig functions, and hyperbolic trig functions)

The trigonometry rules apply to any argument, whether the argument is enclosed by parentheses or not.

The rule presents speech for various trigonometric functions, using one or more instances of the function type to illustrate the speech pattern for functions of that type. If the argument of a trigonometric function is enclosed by parentheses or brackets, speech for the parentheses or brackets enclosing the argument follows the Parentheses and Brackets rule or preference that is set. Speech for parentheses and brackets is covered in [Parentheses and Brackets](#_Parentheses_(and_Brackets)) (see p. 31).

### Rule: (Trig\_Auto.eqp)

* **Six basic trigonometric functions**
* **Argument is** [simple](#_Definition:_Simple_Expression) (see p. 11)

 ***Speech:*** sine *x*

 ***Speech:*** cosine *x*

 ***Speech:*** tangent *x*

 ***Speech:*** secant *x*

 ***Speech:*** cosecant *x*

 ***Speech:*** cotangent *x*

* **All other arguments**

 ***Speech:*** the sine of, open paren, *x* plus *y*, close paren

 ***Speech:*** the cosine of open paren, *x* plus *y*, close paren

 ***Speech:*** the tangent of, open paren, *x* plus *y*, close paren

 ***Speech:*** the secant of open paren, *x* plus *y*, close paren

 ***Speech:*** the cosecant of open paren, *x* plus *y*, close paren

 ***Speech:*** the cotangent of open paren, *x* plus *y*, close paren

* **Powers of the trigonometric functions**
  + - The **power 2**

 ***Speech:*** sine squared *x*.

 ***Speech:*** sine squared of, open paren, 2*x* plus 1, close paren

* + - **The power 3**

 ***Speech:*** sine cubed *x*

 ***Speech:*** sine cubed of, open paren, 2*x* plus 1, close paren

* + - **Powers 4 and greater**

 ***Speech:*** the fourth power of sine *x*.

 ***Speech:*** the fourth power of, sine of, open paren, 2*x* plus 1, close paren

* **Inverse trigonometric functions**

 ***Speech:*** the inverse sine of *x*

 ***Speech:*** the inverse cosine of *x*

 ***Speech:*** the inverse tangent of *x*

 ***Speech:*** the inverse cosecant of *x*

 ***Speech:*** the inverse secant of *x*

 ***Speech:*** the inverse cotangent of *x*

* **Hyperbolic trigonometric functions**

 ***Speech:*** the hyperbolic sine of *x*

 ***Speech:*** the hyperbolic cosine of *x*

 ***Speech:*** the hyperbolic tangent of *x*

 ***Speech:*** the hyperbolic secant of *x*

 ***Speech:*** the hyperbolic cosecant of *x*

 ***Speech:*** the hyperbolic cotangent of *x*

 ***Speech:*** the inverse hyperbolic sine of *x*

 ***Speech:*** the inverse hyperbolic cosine of *x*

 ***Speech:*** the inverse hyperbolic tangent of *x*

 ***Speech:*** the inverse hyperbolic cosecant of *x*

 ***Speech:*** the inverse hyperbolic secant of *x*

 ***Speech:*** the inverse hyperbolic cotangent of *x*

### Preferences:

Note: Trigonometry preferences are only defined for the inverse trigonometric functions. The speech for all other trigonometric functions continues to follow the trigonometry speech rules.

#### TrigInverse (Trig\_TrigInverse.eqp)

 ***Speech:*** sine inverse of *x*

 ***Speech:*** cosine inverse of *x*

 ***Speech:*** tangent inverse of *x*

 ***Speech:*** cosecant inverse of x

 ***Speech:*** secant inverse of *x*

 ***Speech:*** cotangent inverse of *x*

#### ArcTrig (Trig\_ArcTrig.eqp)

 ***Speech:*** arcsine *x*

 ***Speech:*** arcsine of, open paren, *x* plus *y*, close paren

 ***Speech:*** arccosine *x*

 ***Speech:*** arccosine of, open paren, *x* plus *y*, close paren

 ***Speech:*** arctangent *x*

 ***Speech:*** arctangent of, open paren, *x* plus *y*, close paren

 ***Speech:*** arccosecant *x*

 ***Speech:*** arccosecant of, open paren, *x* plus *y*, close paren

 ***Speech:*** arcsecant *x*

 ***Speech:*** arcsecant of, open paren, *x* plus *y*, close paren

 ***Speech:*** arccotangent *x*

 ***Speech:*** arccotangent of, open paren, *x* plus *y*, close paren

## Logarithms

The rule presents speech for various logarithmic functions, using one or more instances of the function type to illustrate the speech pattern for functions of that type. If the argument of a logarithmic function is enclosed by parentheses or brackets, speech for the parentheses or brackets enclosing the argument follows the Parentheses and Brackets rule or preference that is set. Speech for parentheses and brackets is covered in [Parentheses and Brackets](#_Parentheses_(and_Brackets)) (see p. 31).

### Rule: (Log\_Auto.eqp)

* **The log function**
* **Argument is** [Simple](#_Definition:_Simple_Expression) (see p. 11)

** *Speech:*** log *x*

* **All other arguments**

 ***Speech:*** the log of, open paren, *x* plus *y*, close paren

* **The  function**

** *Speech:*** the log, base *b,* of *x*

 ***Speech:*** the log, base *b*, of, open paren, *x* plus *y*, close paren

* **The ln function**

** *Speech:*** l n *x*

 ***Speech:*** l n of, open paren, *x* plus *y*, close paren

### Preference:

#### LnAsNaturalLog (Log\_LnAsNaturalLog.eqp)

* **The log function** (speech is the same as for the rules)
* **Argument is** [Simple](#_Definition:_Simple_Expression) (see p. 11)

** *Speech:*** log *x*

* **All other arguments**

 ***Speech:*** the log of, open paren, *x* plus *y*, close paren

* **The  function** (speech is the same as for the rules)

** *Speech:*** the log, base *b,* of *x*

 ***Speech:*** the log, base *b*, of, open paren, *x* plus *y*, close paren

* **The ln function**

** *Speech:*** natural log *x*

 ***Speech:*** the natural log of, open paren, *x* plus *y*, close paren

## Implied Times

***Some Preliminary Remarks***

Sometimes expressions of the form **entry1(entry2)** or **entry1[entry1]** represent products, and sometimes they represent functions. Therefore, both the ImpliedTimes and Functions rules and preferences could apply to such expressions.

The speech for such expressions is determined as follows.

* If the Functions rule (Functions\_Auto.eqp) has been set, ClearSpeak will first determine whether to recognize the expression as a function by the [criteria listed under the Functions rule](#_Functions) (see p. 22). If ClearSpeak recognizes the expression as a function, the speech for the expression will follow the Functions rule. Otherwise, the speech for the expression will follow the ImpliedTimes rule or preference that has been set.
* If the Functions preference “None” (Functions\_None.eqp) has been set, the speech for the expression will follow the ImpliedTimes rule or preference that has been set.

### Rule: (ImpliedTimes\_Auto.eqp)

* Implied times indicated via parentheses or brackets speaks as “times”.   
  Speech for the parentheses or brackets in the expression follows the Parentheses and Brackets rule or preference that is set. Speech for parentheses and brackets is covered in [Parentheses and Brackets](#_Parentheses_(and_Brackets)) (see p. 31).

Sample expressions:  and 

Note: If the Functions rule (Functions\_Auto.eqp) has been set, expressions of the form **entry1(entry2)** or **entry1[entry2]** will speak as functions. For example, will speak as “*f* of *x*”.

If the Functions preference “None” (Functions\_None.eqp) has been set, expressions of the form **entry1(entry2)** or **entry1[entry2]** will speak as products. For example,  will speak as “*f* times *x*”.

* Implied times indicated by absolute value speaks as “times”.

Sample expressions:  and 

* Implied times indicated via juxtaposition does not speak as “times”.

Sample expression: 

Note: Implied times in matrix and determinant multiplication also speaks as “times”. The speech for matrices is covered in [Matrices, Vectors, and Combinatorics](#_Matrices,_Vectors,_and) (see p. 33).

Sample expression: 

### Preferences:

#### MoreImpliedTimes (ImpliedTimes\_MoreImpliedTimes.eqp)

* Implied times indicated via parentheses or brackets speaks as “times”.

Sample expressions:  and 

Note: If the Functions rule (Functions\_Auto.eqp) has been set, expressions of the form **entry1(entry2)** or **entry1[entry1]** will speak as functions. For example, will speak as “*f* of *x*”.

If the Functions preference “None” (Functions\_None.eqp) has been set, expressions of the form **entry1(entry2)** or **entry1[entry1]** will speak as products. For example,  will speak as “*f* times *x*”.

* Implied times indicated by absolute value speaks as “times”.

Sample expressions:  and 

* Implied times indicated via juxtaposition speaks as “times”.

Sample expression: 

#### None (ImpliedTimes\_None.eqp)

* Implied times does not speak.
* All parentheses and brackets speak.

Note:If the Functions rule (Functions\_Auto.eqp) has been set, expressions of the form **entry1(entry2)** or **entry1[entry1]** will speak as functions. For example, will speak as “*f* of *x*.”

Otherwise, all parentheses and brackets will be spoken. For example, will speak as “2, open paren, 3, close paren” and  will speak as “*f,* open paren*, x,* close paren”.

Sample expressions:  and 

## Parentheses and Brackets

### Rule: (Paren\_Auto.eqp)

Note: This rule and all associated preferences apply to all parentheses and brackets, including those surrounding the argument of a function.

If this rule and the Implied Times preference “None” are set, then all parentheses and brackets will speak.

If this rule and either the Implied Times rule (ImpliedTimes\_Auto.eqp) or the Implied Times preference “MoreImpliedTimes” (ImpliedTimes\_MoreImpliedTimes.eqp) are set, then

* Parentheses and brackets enclosing an expression are *not* spoken in the following cases.
  + The expression enclosed in parentheses or brackets is [simple](#_Definition:_Simple_Expression) (see p. 11) and there is no exponent following the expression.

Sample expressions:  and 

* + The expression enclosed in parentheses or brackets is either
    - a positive number (integer or decimal),
    - an indicator (e.g., *x*, *y*, *z*, etc.), or
    - a positive fraction that is spoken as an ordinal (either by the Ordinal preference or by the default rules)

and

* + - there is an exponent following the expression.

Sample expressions: and 

* Otherwise, parentheses and brackets enclosing an expression are spoken.

Sample expressions:,  and 

#### Speak (Paren\_Speak.eqp)

Speaks all parentheses and brackets.

Sample expressions:  and 

#### SpeakNestingLevel (Paren\_SpeakNestingLevel.eqp)

Parentheses surrounding the argument of a function speak according to the default rules (i.e., when these parentheses speak, they speak without indicating a nesting level).

The following apply to any other pair of parentheses.

* If, according to the default rules, the pair of parentheses does not speak, it does not speak when this preference is set.
* If, according to the default rules, the pair of parentheses speaks, then when this preference is set, the pair of parentheses speaks according to its nesting level.
  + A pair of parentheses that is not nested within another pair of parentheses speaks as “open paren . . . close paren”.
  + A pair of parentheses that is nested within one other pair of parentheses speaks as “open second paren, . . . close second paren”; a pair of parentheses that is nested within two pairs of parentheses speaks as “open third paren, . . . , close third paren”, and so on.

Sample expressions:  and 

Note:Brackets speak as “open bracket…close bracket”, whether they are nested within another pair of brackets or not.

#### Silent (Paren\_Silent.eqp)

* **No** parentheses or brackets speak.

Sample expressions:  and 

* If implied times is not speaking (i.e., the preference [ImpliedTimes\_None](#_NoImpliedTimes) is set), then this preference will speak **all** parentheses and brackets except parentheses and brackets around matrices.

Sample expression:

#### CoordPoint (Paren\_CoordPoint.eqp)

***Speech:*** The point with coordinates, [first coordinate] comma . . . [last coordinate]

Sample expressions:  and 

#### Interval (Paren\_Interval.eqp)

In the following speech, *c* and *d* are used as examples of the speech patterns.

 ***Speech:*** The interval from *c* to *d*, not including *c* or *d*

 ***Speech:*** The interval from *c* to *d*, including *c*, but not including *d*

 ***Speech:*** The interval from *c* to *d*, not including *c*, but including *d*

 ***Speech:*** The interval from *c* to *d*, including *c* and *d*

 ***Speech:*** The interval from negative infinity to *d*, not including *d*

 ***Speech:*** The interval from negative infinity to *d*, including *d*

 ***Speech:*** The interval from *c* to infinity, not including *d*

 ***Speech:*** The interval from *c* to infinity, including *d*

 ***Speech:*** The interval from negative infinity to infinity

 ***Speech:*** The interval from negative infinity to positive infinity

## Matrices, Vectors, and Combinatorics

### Rule: (Matrix\_Auto.eqp)

* ** matrices, where *n* is less than or equal to 3 and all of whose entries are** [simple](#_Definition:_Simple_Expression) (see p.**11**)

***Speech:*** The 1 by [number of columns] row matrix,   
[entries in row 1].

For example,  matrices, all of whose entries are simple expressions, speak as follows.

***Speech:*** The 1 by 3 row matrix.  
[entry], [entry], [entry].

Sample expression: 

* **All other  matrices**

***Speech:*** The 1 by [number of columns] row matrix,   
Column 1: [entry], Column 2 [entry], etc.

Sample expressions:  and 

* ** matrices, where *n* is less than or equal to 3 and all of whose entries are simple expressions**

***Speech:*** The [number of rows] by 1 matrix,   
Column 1: [entries in column 1], etc.

For example,  matrices, all of whose entries are simple expressions speak as follows.

***Speech:*** The 3 by 1 column matrix,  
[entry], [entry], [entry].

Sample expression:

* **All other  matrices**

***Speech:*** The [number of rows] by 1 column matrix,   
Row 1: [entry], Row 2 [entry], etc.

Sample expressions:  and 

* ** matrices, where *m* is equal to 2 or 3 and *n* is equal to 2 or 3 and all of the entries are simple expressions**

***Speech:*** The [number of rows] by [number of columns] matrix,   
Row 1: [entries in row 1], etc.

For example,  matrices, where all of the entries are simple expressions, speak as follows.

***Speech:*** The 3 by 3 matrix,  
Row 1: [entry], [entry], [entry].  
Row 2: [entry], [entry], [entry].  
Row 3: [entry] [entry], [entry].

Sample expressions:  and 

* **All other matrices**

***Speech: The [number of rows] by [number of columns] matrix,***Row 1: column 1, [entry], column 2, [entry], . . .  
Row 2: column 1, [entry], column 2, [entry], . . .  
Row [last row number]: column 1, [entry] column 2, [entry], . . .

Sample expressions:  and

Note: Matrices containing ellipses will not be spoken with the correct dimensions because ClearSpeak has no way of determining the correct dimensions. The presence of ellipses will also cause some row and/or columns to be numbered incorrectly.

### Preferences:

#### SpeakColNum (Matrix\_SpeakColNum.eqp)

*  **matrices**

***Speech:*** The 1 by [number of columns] row matrix,   
Column 1: [entry]. Column 2 [entry], etc.

Sample expressions: and 

* ** matrices**

***Speech:*** The [number of rows] by 1 column matrix,   
Row 1: [entry], Row 2 [entry], etc.

Sample expressions: and 

* **All other matrices**

***Speech:*** The [number of rows] by [number of columns] matrix,   
Row 1: column 1, [entry], column 2, [entry], . . .  
Row 2: column 1, [entry], column 2, [entry], . . .  
Row [last row number]: column 1, [entry] column 2, [entry], . . .

Sample expressions:  and 

#### SilentColNum (Matrix\_SilentColNum.eqp)

All matrices speak without column number of entries.

* ** matrices**

***Speech:*** The 1 by [number of columns] row matrix,   
[entry]. [entry], etc.

Sample expressions: and  and 

* ** matrices**

***Speech:*** The [number of rows] by 1 column matrix,   
[entry], [entry], etc.

Sample expressions:  and 

* **All other matrices**

***Speech:*** The [number of rows] by [number of columns] matrix,   
Row 1: [entry], [entry], . . .  
Row 2: [entry], [entry], . . .   
Row [last row number]: [entry], [entry], . . .

Sample expressions:  and 

#### EndMatrix (Matrix\_EndMatrix.eqp)

This preference is the rules with “end matrix” added at the end.

Sample expressions: and  

#### Vector (Matrix\_Vector.eqp)

*  **matrices, all of whose entries are** [simple](#_Definition:_Simple_Expression) (see p. 11)

***Speech:*** The 1 by [number of columns] row vector,   
[entry], [entry], [entry], etc.

Sample expression: 

* **All other**  **matrices**

***Speech:*** The 1 by [number of columns] row vector,   
Column 1: [entry], Column 2 [entry], etc.

Sample expressions:  and 

* ** matrices, all of whose entries are simple expressions**

***Speech:*** The [number of rows] by 1 column vector,   
[entry], [entry], etc.

Sample expression: 

* **All other  matrices**

***Speech:*** The [number of rows] by 1 column vector,   
Row 1: [entry], Row 2 [entry], etc.

Sample expressions:  and 

* **All other matrices**

***Speech:*** According to the rules

Sample expressions:  and 

#### EndVector (Matrix\_EndVector.eqp)

This preference is the same as the Vector preference, with “end vector” at the end of vectors (matrices with one row or one column), and the same as the EndMatrix preference for all other matrices.

Sample expressions:  and 

#### Combinatorics (Matrix\_Combinatoric.eqp)

* **Matrices with 1 column and 2 rows**

 ***Speech:*** *n* choose *r*

Sample expressions:  and 

* **All other matrices**

***Speech:*** Same as the Matrix rules

## Multiline Entries

***Some Preliminary Remarks***

The ClearSpeak rules and preferences for multiline entries can vary the way the multi-line expression is spoken in two ways.

1. Whether and/or how each line in the entry is labeled
2. Whether an overview of the multiline entry is given before the multi-line entry itself is spoken

If an overview is given, the speech will depend on whether and/or how each line in the entry is labeled.

In addition, if the multiline entry has more than one column, the length of the pauses between columns can be adjusted.

### Multiline Overview Rule: (MultiLineOverview\_Auto.eqp)

An overview of the multiline expression is given before the multiline entry is spoken.

(Refer to *Multiline Label Rule and Preferences*, which begins just a few lines down in this document, for how the overview is spoken depending on whether and/or how each line of the entry is labeled.)

### Preference:

#### None (MultiLineOverview\_None.eqp)

An overview of the multiline expression is *not* given.

### Multiline Label Rule: (MultiLineLabel\_Auto.eqp)

* **Piecewise defined functions**

The word “case” and the line number are inserted before each line to the right of the left brace. The left brace does not speak. In addition, a long pause is inserted between lines.

***Speech for each line:*** Case [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] cases

Sample expression (with overview): 

Sample expression (without overview): 

* **All other multiline expressions**

***Speech for each line:*** Line [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] lines

Note: The overview speech will speak if the Overview rule (MultiLineOverview\_Auto.eqp) is set. The overview speech will *not* speak if the Overview preference (MultiLineOverview.None) is set. (This applies to all of the Multiline Label preferences.)

Sample expressions (with overview): 

Sample expressions (without overview): 

### Preferences:

#### Case (MultiLineLabel\_Case.eqp)

When this preference is used, the word “case” and the line number are inserted before each line. In addition, a long pause is inserted between lines.

***Speech for each line:*** Case [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] cases

Sample expression (with overview): 

Sample expression (without overview): 

Note: If this preference is set, speech for piecewise-defined functions presented using a left brace follows the MultiLineLabel rule (MultiLineLabel\_Auto.eqp).

Sample expression (with overview): 

Sample expression (without overview): 

#### Equation (MultiLineLabel\_Equation.eqp)

When this preference is used, the word “equation” and the line number are inserted before each line. In addition, a long pause is inserted between lines.

***Speech for each line:*** Equation [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] equations

Sample expression (with overview): 

Sample expression (without overview): 

#### Line (MultiLineLabel\_Line.eqp)

When this preference is used, the word “line” and the line number are inserted before each line. In addition, a long pause is inserted between lines.

***Speech for each line:*** Line [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] lines

Sample expression (with overview): 

Sample expression (without overview): 

#### Row (MultiLineLabel\_Row.eqp)

When this preference is used, the word “row” and the line number are inserted before each line. In addition, a long pause is inserted between lines.

***Speech for each line:*** Row [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] rows

Sample expression (with overview): 

Sample expression (without overview): 

#### Step (MultiLineLabel\_Step.eqp)

When this preference is used, the word “step” and the line number are inserted before each line. In addition, a long pause is inserted between lines.

***Speech for each line:*** Step [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] steps

Sample expression (with overview):

Sample expression (without overview):

#### Constraint (MultiLineLabel\_Constraint.eqp)

When this preference is used, the word “constraint” and the line number are inserted before each line. In addition, a long pause is inserted between lines.

***Speech for each line:*** Constraint [line number]: [entry in line]

***Corresponding Overview Speech:*** [number of lines] constraints

Sample expression (with overview): 

Sample expression (without overview): 

#### None (MultiLineLabel\_None.eqp)

When this preference is used, a long pause is inserted between lines to clearly distinguish one line from the next. No line labels are spoken.

***Corresponding Overview Speech:*** [number of lines] lines

Sample expression (with overview): 

Sample expression (without overview): 

### Multi-Line Pauses Between Columns Rule: (MultiLinePausesBetweenColumns\_Auto.eqp)

If the multiline entry is entered as a multicolumn, there will be uniform pauses between columns.

Sample expression (3 columns): 

### Preferences:

#### Long (MultiLinePausesBetweenColumns\_Long.eqp)

When this preference is used, there will be long, uniform pauses between the columns. (The pauses will be longer than when the rule is set.)

Sample expression (3 columns): 

#### Short (MultiLineEntryMultiColumn\_Short.eqp)

When this preference is used, there will be short uniform pauses between the columns. (The pauses will be shorter than when the rule is set.)

Sample expression (3 columns): 

## Sets enclosed in set brackets { }

### Rule: (Sets\_Auto.eqp)

* **There is either a single vertical bar or a colon in the set.**

***Speech:*** The set of all [entry between left set bracket and vertical bar] such that [entry between vertical bar and right set bracket]

Sample expressions:  and 

* **There is no vertical bar in the set**.

***Speech***: The set [entry 1] comma [entry 2] comma . . .

Sample expression: 

* **There is nothing in between the set brackets.**

***Speech***: The empty set

Sample expression: 

### Preferences:

#### woAll (Sets\_woAll.eqp)

* **There is either a single vertical bar or a colon in the set.**

***Speech:*** The set of [entry between left set bracket and vertical bar] such that [entry between vertical bar and right set bracket].

Sample expressions:  and 

* **There is no vertical bar in the set. (Speech follows the rule)**

***Speech:***The set [entry 1] comma [entry 2] comma . . .

Sample expression: 

* **There is nothing in between the set brackets. (Speech follows the rule)**

***Speech***: The empty set

Sample expression: 

#### SilentBracket (Sets\_SilentBracket.eqp)

* **There is either a single vertical bar or a colon in the set. (Speech follows the rule)**

***Speech:*** The set of all [entry between left set bracket and vertical bar] such that [entry between vertical bar and right set bracket].

Sample expressions:  and 

* **There is no vertical bar in the set**.  
  ***Speech:*** [entry1] comma [entry 2] comma . . .

Sample expression: 

* **There is nothing in between the set brackets. (Speech follows the rule)**

***Speech***: The empty set

Sample expression: 

## Named Sets (Number Systems)

### Rule:

Note: There are no speech preferences for these named sets. Therefore, there is no preference file for the rule (i.e., there is no .eqp file for named sets).

* **Number systems**

***Speech:***  the real numbers

 the complex numbers

 the integers

 the rational numbers

 the natural numbers

 the positive integers

* **An exponent is attached to one of the number systems.**

***Speech:*** [letter representing number system] [number]

Examples:  speaks as “r–two”.

 speaks as “z–three”.

 speaks as “c–n”.

 speaks as “r–infinity”.

#### Preferences:

None

# Part 2: Symbols

## Multiplication Symbol

### Rule: (MultSymbolX\_Auto.eqp)

***Speech:*** times

Sample expression: 

### Preference:

#### By (MultSymbolX\_By.eqp)

***Speech:*** by

Sample expression: 

#### By (MultSymbolX\_Cross.eqp)

***Speech:*** cross

Sample expression: 

## Multiplication Symbol

### Rule: (MultSymbolDot\_Auto.eqp)

***Speech:*** times

Sample expression: 

### Preference:

#### By (MultSymbolDot\_Dot.eqp)

***Speech:*** dot

Sample expression: 

## Triangle Symbol:

### Rule: (TriangleSymbol.Auto.eqp)

***Speech:*** triangle

Sample expression: 

### Preference:

#### Delta (TriangleSymbol\_Delta.eqp):

***Speech:*** delta

Sample expression: 

## Ellipses

### Rule: (Ellipses\_Auto.eqp)

Ellipses speak as “dot dot dot”

Sample expressions: ****and 

### Preference:

#### AndSoOn (Ellipses\_AndSoOn.eqp)

* **Ellipses only at end of list**

***Speech:*** and so on

Sample expression: ****

* **Ellipses only in the middle of the list**

***Speech:*** and so on up to

Sample expression: ****

* **All other cases**

***Speech:*** dot dot dot

Sample expression: 

## Vertical Line

Note: When entering expressions in MathType, use the vertical bar on the keyboard for a single vertical line. For absolute value, use the MathType template.

### Rule: (VerticalLine\_Auto.eqp)

* **There is a subscript attached to the vertical line.**

***Speech:*** [expression] evaluated at [expression]

Sample expression: 

* **There is a superscript and a subscript attached to the vertical line.**

***Speech:*** [expression] evaluated at [superscript] minus the same expression evaluated at [subscript]

Sample expression: 

* **The vertical line is not within set brackets and does not have a subscript or a superscript attached.**

***Speech:*** divides

Sample expression: 

* **The vertical line is within set brackets.**

***Speech:*** such that

Sample expression: 

### Preference:

Note: VerticalLine preferences are only defined for vertical lines that do not have subscripts or superscripts attached. The speech for vertical lines with subscripts or superscripts attached will always follow the VerticalLine rule.

#### SuchThat (SingleVerticalLine\_SuchThat.eqp)

***Speech:*** such that

Sample expression: 

#### Divides (SingleVerticalLine\_Divides.eqp)

***Speech:*** divides

Sample expression: 

#### Given (SingleVerticalLine\_Given.eqp)

***Speech:*** given

Sample expression: 

## Set Membership Symbols: and

### Rule: (SetMemberSymbol\_Auto.eqp)

* **The set membership symbol is inside set brackets.**

***Speech:***  in

 not in

Sample expressions: and 

* **The set membership symbol is in an expression below one of the following symbols.** 

***Speech:***  over *b* is a member of *S*

 over *b* is not a member of *S*

Sample expression:

Note: The speech for the symbolsis given in the [next section of this document](#_Sums,_Products,_Union,s) (Sums, Products, Unions, Intersections, and Integrals; see p. 54).

* **The set membership symbol is not inside set brackets.**

***Speech:***  is a member of

 is not a member of

Sample expressions: and 

### Preferences:

#### Member (SetMemberSymbol\_Member.eqp)

* **The set membership symbol is inside set brackets.**

***Speech:***  member of

 not a member of

Sample expressions:and 

* **The set membership symbol is in an expression below one of the following symbols.** 

***Speech:***  over *b* is a member of *S*

 over *b* is not a member of *S*

Sample expression:

* **The set membership symbol is not inside set brackets.**

***Speech:***  is a member of

 is not a member of

Sample expressions: and 

#### Element (SetMemberSymbol\_Element.eqp)

* **The set membership symbol is inside set brackets.**

***Speech:***  element of

 not an element of

Sample expressions:and 

* **The set membership symbol is in an expression below one of the following symbols.** 

***Speech:***  over *b* is an element of *S*

 over *b* is not an element of *S*

Sample expression:

* **The set membership symbol is not inside set brackets.**

***Speech:***  is an element of

 is not an element of

Sample expressions: and 

#### In (SetMemberSymbol\_In.eqp)

* **The set membership symbol is inside set brackets.**

***Speech:***  in

 not in

Sample expressions:and 

* **The set membership symbol is in an expression below one of the following symbols.** 

***Speech:***  over *b* is in *S*

 over *b* is not *S*

Sample expression: 

* **The set membership symbol is not inside set brackets.**

***Speech:***  is in

 is not in

Sample expressions: and 

#### Belongs (SetMemberSymbol\_Belongs.eqp)

* **The set membership symbol is inside set brackets.**

***Speech:***  belonging to

 not belonging to

Sample expressions: and 

* **The set membership symbol is in an expression below one of the following symbols.** 

***Speech:***  *b* belongs to *S*

 *b* does not belong to *S*

Sample expression: 

* **The set membership symbol is not inside set brackets.**

***Speech:***  belongs to

 does not belong to

Sample expressions: and 

## Sums, Products, Unions, Intersections, and Integrals

### Rule:

* **Symbols alone**

 ***Speech:*** thesum of

Sample expression:

 ***Speech:*** the product of

Sample expression:

 ***Speech:*** the integral

Sample expression: 

 ***Speech:*** union

Sample expression: 

 ***Speech:*** union

Sample expression: 

 ***Speech:*** intersection

Sample expression: 

 ***Speech:*** intersection

Sample expression: 

 ***Speech:*** the integral

Sample expression: 

* **There is an entry above and below the symbol.**

***Speech:*** the sum from [entry below symbol] to [entry above symbol] of [term]

Sample expression: 

***Speech:*** the product from [entry below symbol] to [entry above symbol] of [term]

Sample expression:



***Speech:*** the union from [entry below symbol] to [entry above symbol] of [term]

Sample expression:



***Speech:*** the intersection from [entry below symbol] to [entry above symbol] of [term]

Sample expression: 

******

***Speech:*** the integral from [entry] to [entry] of [expression]

Sample expression: 

* **There is an entry below the symbol.**



***Speech:*** the sum over [entry below symbol] of [term]

Sample expression:



***Speech:*** the product over [entry below symbol] of [term]

Sample expression:



***Speech:*** the union over [entry below symbol] of [term]

Sample expression: 

 ***-***

***Speech:*** the intersection over [entry below symbol] of [term]

Sample expression: 



***Speech:*** the integral over [expression under integral]

Sample expression: ******

# Part 3: Adornments

## Prime, Double Prime, and Triple Prime

The prime rules and preferences are for the prime, double prime, and triple prime symbols.

Note: In this rule and all associated preferences, the word “number” includes integers, decimals, fractions, mixed numbers, and negative numbers.

### Rule: (Prime\_Auto.eqp)

* **The expression begins with a number or letter adorned with a degree symbol (e.g.,** **or** **), followed by a number or letter adorned by a prime (e.g.,** **or**  **) and/or a number or letter adorned by a double prime (e.g.,**  **or**  **).**

Prime ***Speech:*** minutes

Double prime ***Speech:*** seconds

Sample expressions:  and 

* **The expression is a number adorned with a prime (e.g.,** **), a number adorned with a double prime (e.g.,** **), or a number adorned with a prime followed by a number adorned with a double prime (e.g.,** )**.**

Prime ***Speech:*** feet

Double prime ***Speech:*** inches

Sample expressions:  and 

* **Any other expression with a prime, double prime, or triple prime symbol**

 ***Speech:*** *f* prime

 ***Speech:*** *f* double prime

 ***Speech:***  *f* triple prime

Sample expressions:  and

### Preferences:

#### Length (Prime\_Length.eqp)

* **The expression begins with a number or letter adorned with a degree symbol (e.g.,** **or** **), followed by a number or letter adorned by a prime (e.g.,** **or** **) and/or a number or letter adorned by a double prime (e.g.,**  **or** **).**

Sample expressions:  and 

* **Expressions of the form** **,** **, or** 

Prime ***Speech:*** feet

Double prime ***Speech:*** inches

Sample expressions:  and 

#### Angle (Prime\_Angle.eqp)

* **The expression begins with a number or letter adorned with a degree symbol (e.g.,** **or** **), followed by a number or letter adorned by a prime (e.g.,** **or** **) and/or a number or letter adorned by a double prime (e.g.,**  **or** **).**

Sample expressions:  and 

* **Expressions of the form** **,** **, or** 

Prime ***Speech:*** minutes

Double prime ***Speech:*** seconds

Sample expressions:  and 

## Combinations and Permutations: and

### Rule: (CombinationPermutation\_Auto.eqp)

A “C” with a subscript to the left and a subscript to the right, and a “P” with a subscript to the left and a subscript to the right follow the speech pattern described below.

 ***Speech:*** n C r

 ***Speech:***  n P r

### Preference: (CombinationPermutation\_ChoosePermute.eqp)

 ***Speech:*** n choose r

 ***Speech:*** n permute r

## Bar

### Rule: (Bar\_Auto.eqp)

* **Bar is over two juxtaposed capital letters or over two juxtaposed capital letters with other adornments (e.g., prime, subscript).**

***Speech:*** the line segment . . .

Sample expressions:  and 

* **Bar is over digits at the end of a decimal number.**

***Speech if the bar is over one digit:*** the repeating decimal [number] with repeating digit [digit under bar]

***Speech if the bar is over more than one digit:*** the repeating decimal [number] with repeating digits [digits under bar]

Sample expressions: and 

* **Bar is over a (possibly adorned) character.**

***Speech:*** bar

Note: If a letter is a recognized [function name](#_Functions) (see p. 21), then so is the letter with a bar over it.

Sample expressions:  and 

**Otherwise**

***Speech:*** the complex conjugate of [expression]

Sample expressions:  and 

### Preference:

#### Bar (Bar\_Bar.eqp)

* **Bar is over digits at the end of a decimal number.**

***Speech:*** same as the rules

Sample expressions: and 

* **Otherwise**

***Speech:*** bar

Sample expressions:  and 

#### Conjugate (Bar\_Conjugate.eqp)

* **Bar is over digits at the end of a decimal number.**

***Speech:*** same as the rules

Sample expressions: and 

* **Otherwise**

***Speech:*** the complex conjugate of [expression under bar]

Sample expressions:  and 

#### Mean (Bar\_Mean.eqp)

* **Bar is over digits at the end of a decimal number.**

***Speech:*** same as the rules

Sample expressions: and 

* **Otherwise**

***Speech:*** the mean of [expression under bar]

Sample expression:

## Equals Sign with Adornments

### Rule:

* 

***Speech:*** is defined to be

Sample expression: 

* 

***Speech:*** equals sign with question mark over it

Sample expression: 