

# Application of xThink's meta-recognition scheme to formula recognition in MathJournal

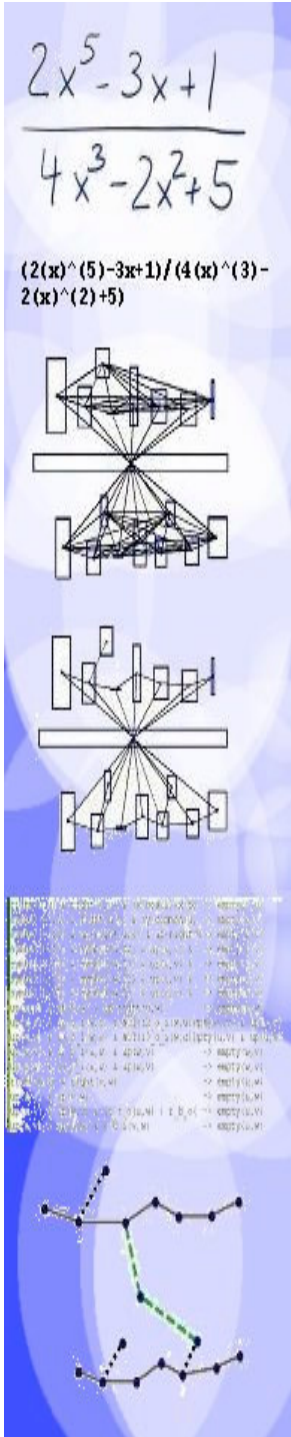
$$\frac{2x^5 - 3x + 1}{4x^3 - 2x^2 + 5}$$

- *What is xThink's approach to recognition on the Tablet PC?*
- *How does xThink's technology enable MathJournal to recognize mathematical expressions?*

xThink, Inc.  
 Teresa Shu, Ph.D.  
 tshu@xthink.com

## Preliminary points

- The "recognizer" in this presentation refers to the xThink recognition engine. This engine powers MathJournal.
- The purpose of the recognition and interpretation scheme is to
  1. encode implicit meaning
  2. handle the ambiguity in the users input
- xThink's meta-recognition scheme takes many minutes to explain, but recognition in MathJournal happens within milliseconds after you lift the pen from the Solver tool.
- Computers are *optimally suited* for recognition processing. xThink will continue to optimally exploit every increase in computer processor speed.



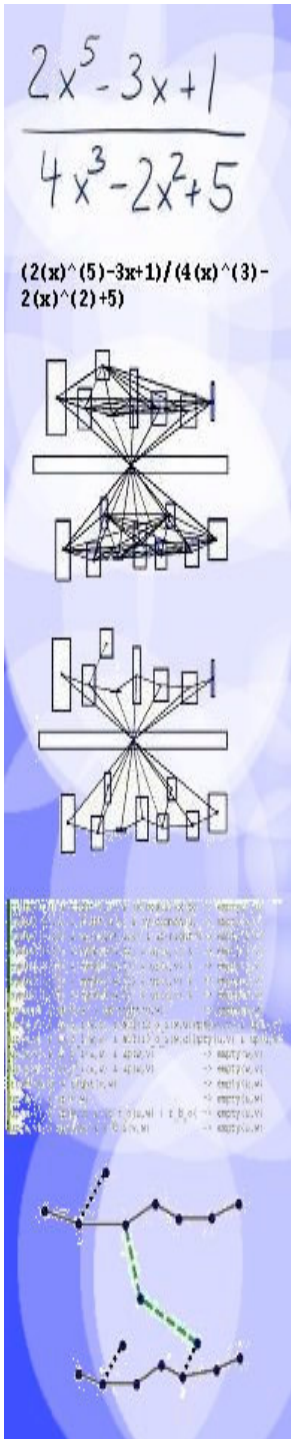
# What these slides do and do not cover

**COVERED:** Today's presentation will cover the generation of a **minimum spanning tree** as the key stage in recognition of mathematical expressions. The **minimum spanning tree** is sufficient because mathematical problems resolve to a single set of solutions.

**NOT COVERED:** For recognition, **graphs and hypergraphs** are logical extensions of the minimum spanning tree. However, their discussion belongs in a separate presentation on recognition of visual languages, such as Simulink® and recognition of graphical design schemes, such as UML, flowcharts, and state machines.

**NOT COVERED:** MathJournal does not require or implement the following **approaches to recognition:**

- user-dependent recognition
- training-based recognition
- context-based recognition
- support vector machines



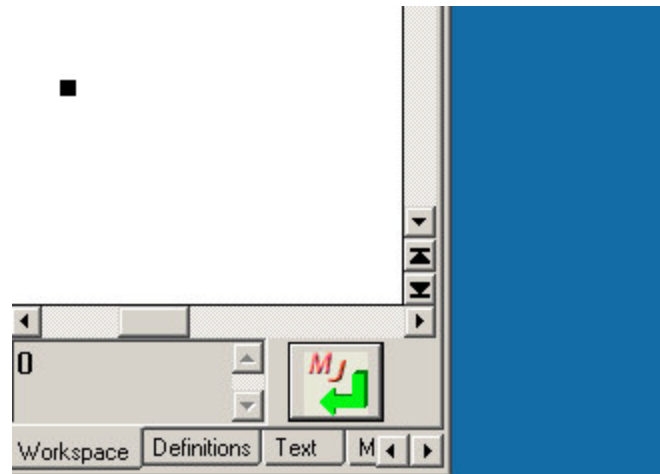
## What MathJournal does: in words

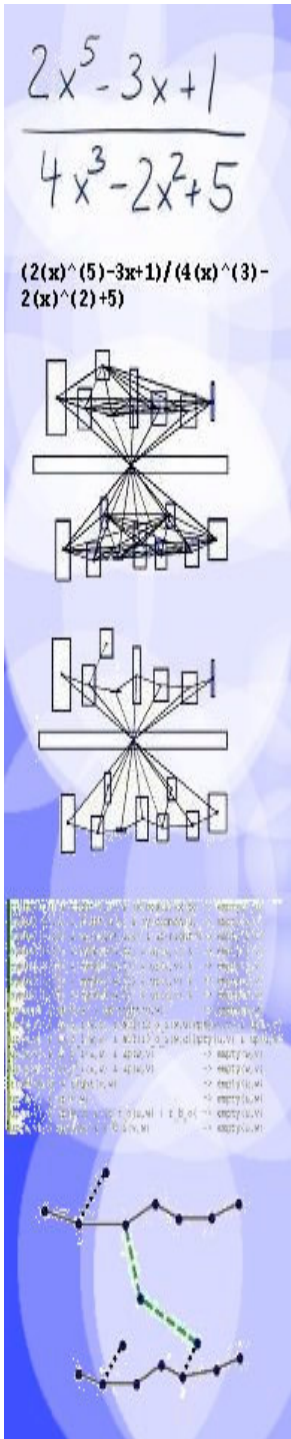
Engineers, scientists, teachers and students buy MathJournal to solve problems. Here's a breakdown of what they do and how MathJournal responds:

User action	MathJournal response
<b>1. Input a math problem.</b> <b>NOTE:</b> MathJournal's biggest boon to users comes here; write math problems naturally, with pen.	Render the users pen strokes in the Ink data type.
<b>2. Tap the Solver tool.</b>	Recognize complex pen input: <ul style="list-style-type: none"> <li>• Characters and symbols</li> <li>• Spatial relationships</li> <li>• Problem type</li> </ul> This is MathJournal's principal contribution to the world of math software.
<b>3. Select a solution type in the popup menu.</b>	Apply the appropriate algorithm Display the solution in text, Ink, and/or graphical form

# What MathJournal does: in action

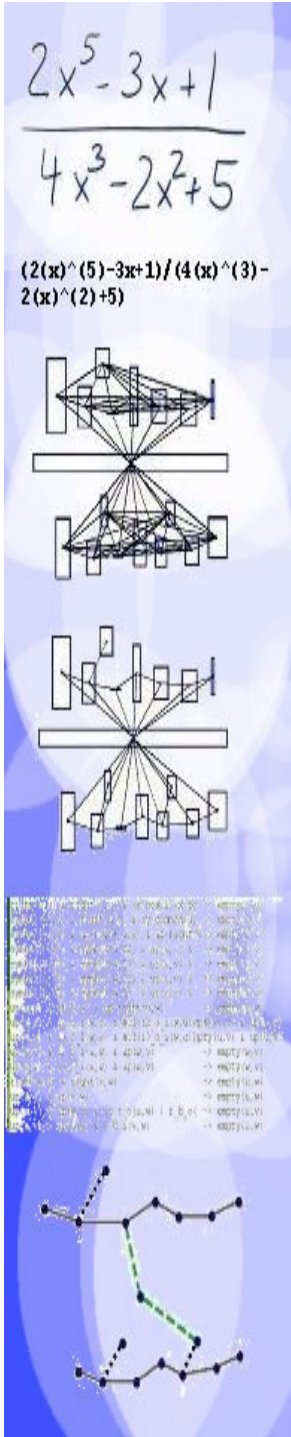
In the following animation, MathJournal recognizes Inked characters and symbols, recognizes their spatial relationships, and recognizes the problem type.





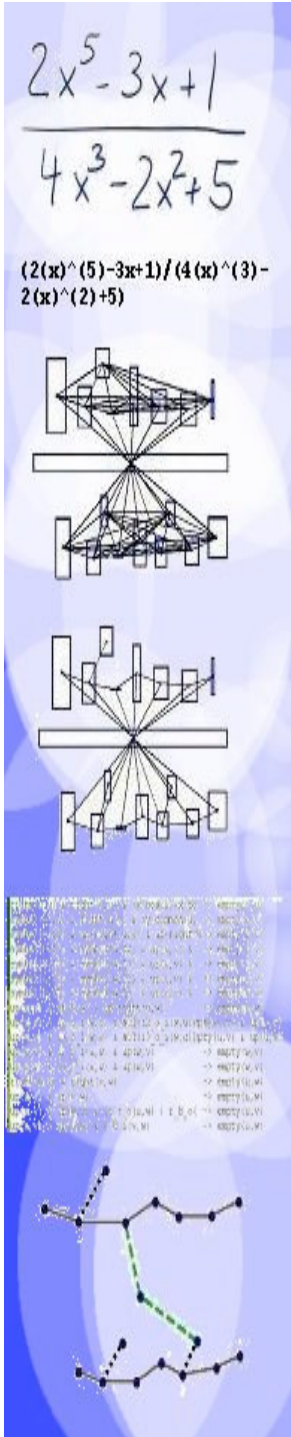
# How MathJournal differs from non-pen-based Math software

User action	MathJournal response	Comparison with non-pen-based Math software
<p>1. Input a math problem.</p> <p><b>NOTE:</b> MathJournal's biggest boon to users comes here; write math problems naturally, with pen.</p>	<p>Render the users pen strokes in the Ink data type.</p>	<p><b>A useability gap:</b></p> <p>Users learn custom keystrokes and GUI selections to express their problems to each math software product.</p>
<p>2. Tap the Solver tool.</p>	<p>Recognize complex pen input:</p> <ul style="list-style-type: none"> <li>• Characters and symbols</li> <li>• Spatial relationships</li> <li>• Problem type</li> </ul> <p><b>NOTE:</b> This is MathJournal's principal contribution to the world of math software.</p>	<p><b>An interactivity gap:</b></p> <ul style="list-style-type: none"> <li>• Recognition of pen input not available.</li> <li>• Recognition of problem type not available.</li> </ul>
<p>3. Select a solution type in the popup menu.</p>	<p>Apply the appropriate algorithm.</p> <p>Display the solution in text, Ink, MathML, and/or graphical form.</p>	<p>Same (but no Ink output)</p>



## xThink's meta-recognition scheme: overview

- **EXTENSIBILITY:** Meta-recognition scheme that can be easily generalized to cases where recognition of graphically oriented constructs, such as visual programming languages, are essential. These constructs include formulas, flowcharts, graphical depictions of control processes, stateflow diagrams, graphics used for image analysis, etc.
- **IMPLICIT KNOWLEDGE:** Handwritten or hand drawn representations use a lot of implicit knowledge and agreements about graphical layout to reconstruct the correct meaning of the drawing.
  - The scheme presented here includes a formalized way of representing the implicit knowledge.
  - Also, the scheme is able to resolve and represent natural ambiguities that arise in handwritten or hand drawn representations.



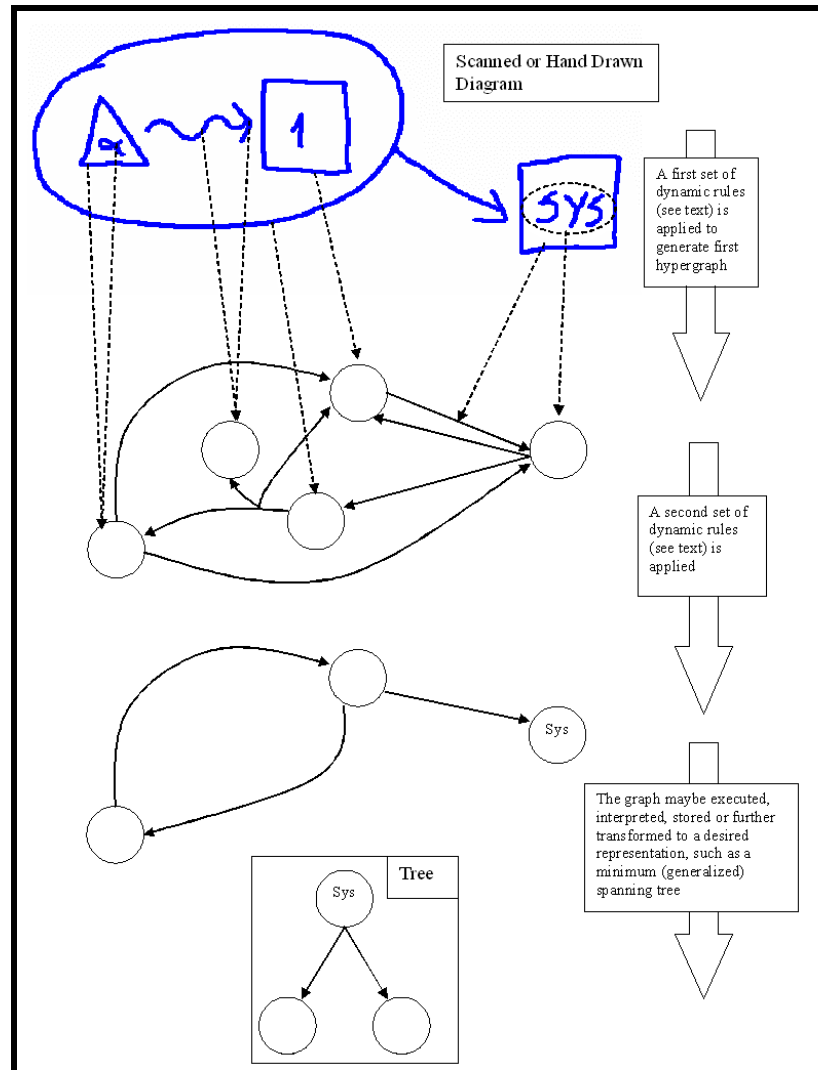
# Recognizing a diagram using MathJournal's meta-recognition scheme

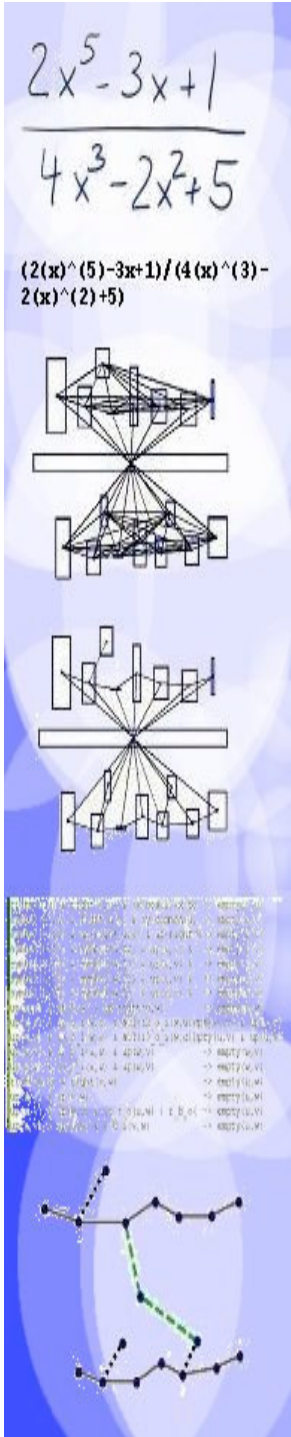
Symbol recognition >>

Adjacency graphs >>

Apply rules to refine the graph >>

Resolve the graph



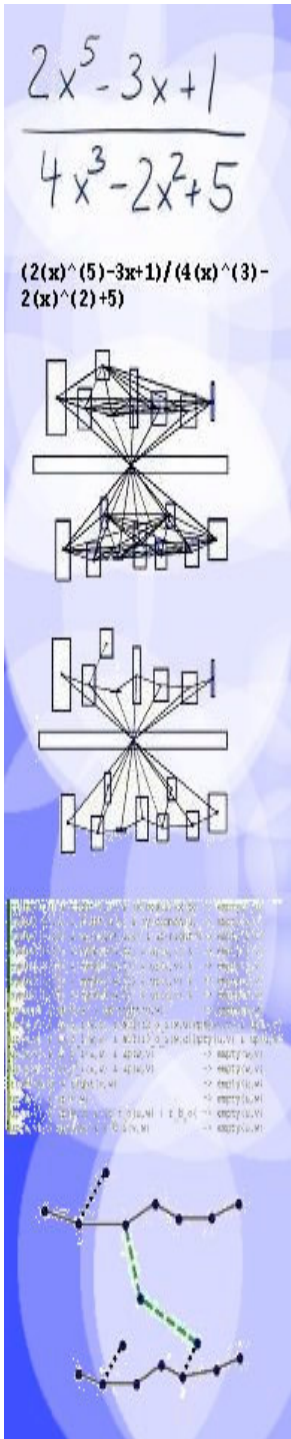


## xThink's meta-recognition scheme: concepts

- Identify elementary graphical units
- Catalog hierarchical and nested relationships of the units
- Translate the structure to a graph or hypergraph (with nodes representing the graphical units or relationships between them)
- Apply semantics to clarify the edges and/or hyperedges.
- When appropriate, use sets of rules to additionally process the graph/hypergraph into a minimum spanning tree or graph/hypergraph.

***GOAL:*** Reduce the graph/hypergraph to its essential expression, so it can be executed or interpreted.

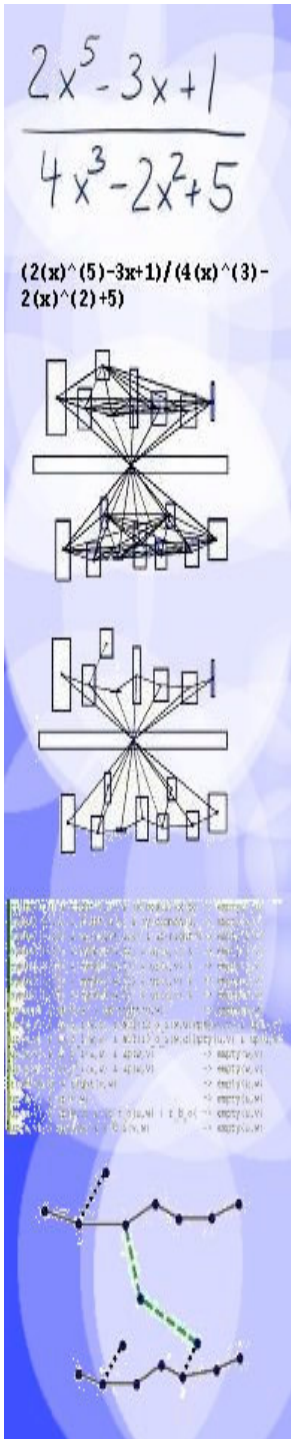




# MathJournal's meta-recognition scheme: summary of steps

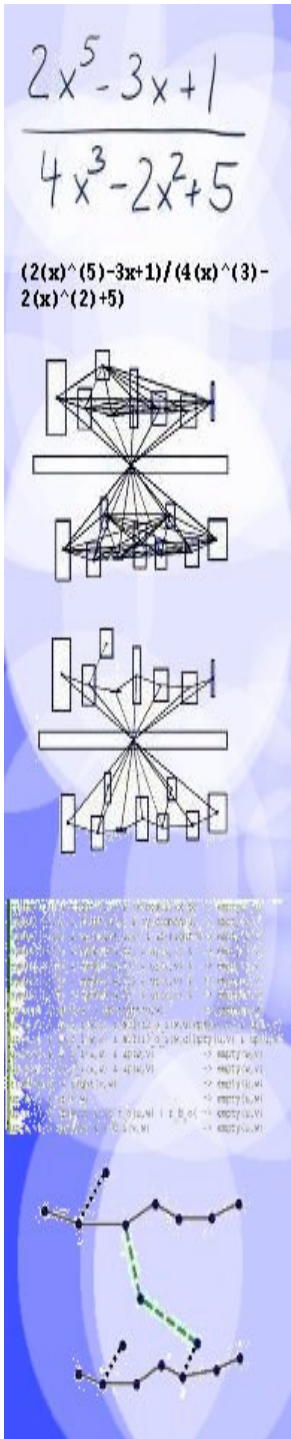
The order of the following list is typical, but several stages can be varied *and repeated*.

- **Combine strokes** that generate an individual symbol
- **Recognize symbols** and generate surrounding rectangles
- **Construct meta-symbols** such as integrals and roots  
*Later, symbols and meta-symbols are represented by nodes in an adjacency graph.*
- **Build up a first adjacency matrix** to describe connections between symbols, meta-symbols and their surrounding rectangles.
- **Simplify the adjacency matrix** by applying sets of transformation rules  
*Most operations delete existing edges, some add new edges. Some rules are completely edge-content based others include symbol information and geometric component (e.g. size of symbols). Skip this step if no rules apply.*
- **Add weights to the edges**  
*The lower the weight, the more likely the edge is chosen for the resolution process. However, if the resulting graph is a tree apply the appropriate rules.*
- **Construct a minimum spanning tree** based on the weighted adjacency matrix
- **Resolve the graph** by rules and generating the formula content  
*Resolve the minimum spanning tree by node (symbol) reduction and generate a syntactically correct representation of the formula under investigation.*



## Stages of recognition for a typical math problem

- **STAGE ONE:** Capturing of user input and immediate recognition of symbols
- **STAGE TWO:** Construction of preliminary adjacency graph
- **STAGE THREE:** Refinement of adjacency graph, leveraging previous information and applying rule sets as appropriate
- **STAGE FOUR:** Representation of the formula in a minimum spanning tree



## STAGE ONE: Capture user input and immediately recognize symbols

1. Combine strokes that generate a symbol
2. Recognize symbols and generating surrounding rectangles
3. Construct meta-symbols

$$\frac{2x^5 - 3x + 1}{4x^3 - 2x^2 + 5}$$

$$2x^5 - 3x + 1 / 4x^3 - 2x^2 + 5$$





$$\frac{2x^5 - 3x + 1}{4x^3 - 2x^2 + 5}$$

$$\frac{(2(x)^5 - 3x + 1)}{(4(x)^3 - 2(x)^2 + 5)}$$

## STAGE THREE, *continued*: Applying rules to simplify the adjacency graph

Standard graph-rewriting is a natural generalization of string grammars and term rewriting systems. Typical graph rewriting systems are context-free and replace nodes, edges, hyperedges, sub-graphs with other sub-graphs. The xThink recognizer augments static graph rewriting with rules sets like the following. The characters *u*, *v* and *w* represent nodes (i.e. identified symbols). The predicates and functions are described in the text.

```

right(u,v) & right(v,u) & (x-pos(u)<x-pos(v))           -> empty(v,u)
symbol(v,I1) & right(u,v) & (y-common(u,v)<1))         -> empty(u,v)
symbol(v,I1) & up-right(u,v) & up-right(u,w) & up(w,v) -> empty(u,v)
symbol(v,I1) & symbol(w,I2) & up(u,v) & right(u,w)     -> empty(u,w)
symbol(v,I1) & symbol(w,I3) & up(u,v) & right(u,w)     -> empty(u,w)
symbol(u,I1) & symbol(w,I2) & up(u,v) & right(v,w)     -> empty(v,w)
symbol(u,I1) & symbol(w,I3) & up(u,v) & right(v,w)     -> empty(v,w)
up(u,v) & up(v,w) & up-right(u,w)
up(u,v) & i2_o_i(w,v) & NOT(i2_o_i(w,u))
up(u,v) & i3_o_i(w,v) & NOT(i3_o_i(w,u))
up(u,v) & i2_o_i(u,w) & up(w,v)
up(u,v) & i3_o_i(u,w) & up(w,v)
right(u,v) & right(v,w)
up(u,v) & up(v,w)
up(u,v) & up(w,v) & (r_t_o(u,w) | r_b_o(u,w))
up(u,v) & up(u,w) & r_b_i(v,w)
up-right(u,v) & up-right(u,w) & right(v,w)

```

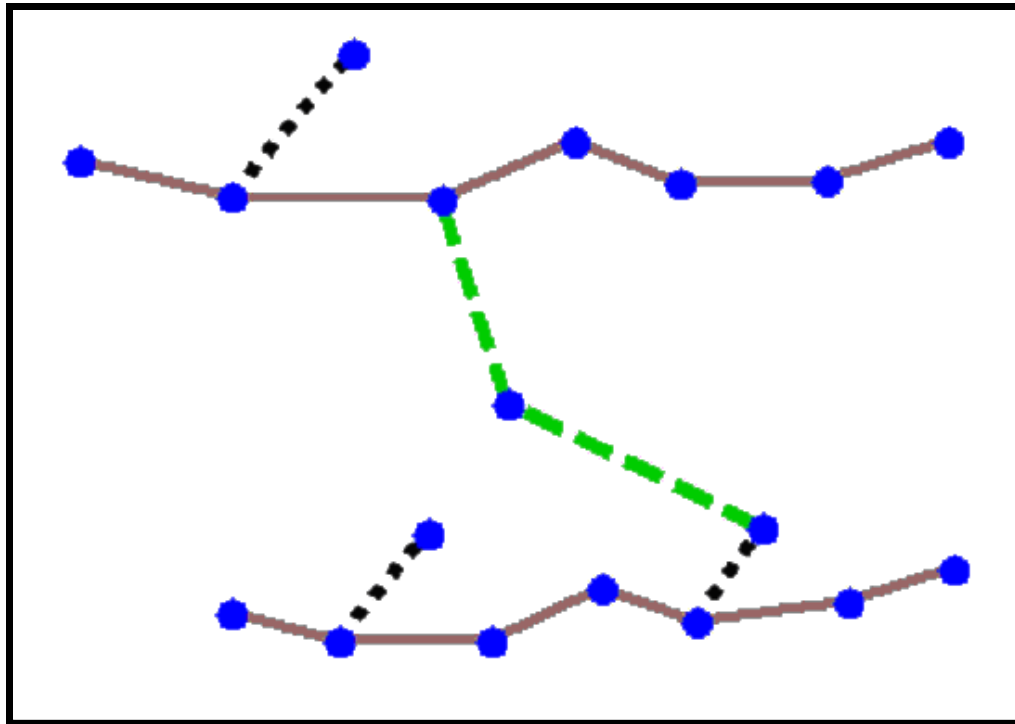
& - AND  
 | - OR

**This is a rule set that augments standard graph-rewriting procedures:**  
 Here, the recognizer applies geometric and graph-independent constraints (first rule in this set). In this case the recognizer should also apply first-order logic predicates.

$$\frac{2x^5 - 3x + 1}{4x^3 - 2x^2 + 5}$$

$$\frac{(2(x)^5 - 3x + 1)}{(4(x)^3 - 2(x)^2 + 5)}$$

## STAGE FOUR: Represent as minimum spanning tree



- Kruskal's (one of several available algorithms) minimum spanning tree is then applied to the graph and the final minimum spanning tree representation of the original formula is obtained.
- The adjacency classes are "right" (solid line), "up-right" (dotted line) and "up" (dashed line).



$$\frac{2x^5 - 3x + 1}{4x^3 - 2x^2 + 5}$$

$$\frac{(2(x)^5 - 3x + 1)}{(4(x)^3 - 2(x)^2 + 5)}$$

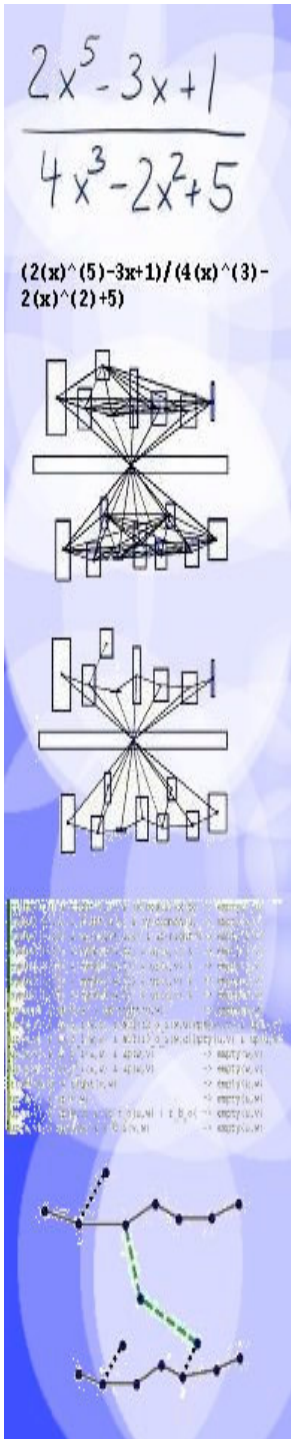
The sidebar contains several visual elements:
 

- Handwritten mathematical expressions:  $2x^5 - 3x + 1$  and  $4x^3 - 2x^2 + 5$ .
- A typed mathematical expression:  $\frac{(2(x)^5 - 3x + 1)}{(4(x)^3 - 2(x)^2 + 5)}$ .
- A network graph with nodes and edges.
- A data visualization showing a grid of points with varying colors.
- A path graph with nodes and edges, some highlighted in green.

## STAGE FOUR, *continued*:

# Apply generalizations of the minimum spanning approach

Type of minimum spanning problem	Input: $G = (V, E)$	Output: A subset of $E$ that represents...
<b>Standard Minimum Spanning tree</b> , undirected (connected) graph with weighted edges	<ul style="list-style-type: none"> <li><math>V</math> = vertices of <math>G</math></li> <li><math>E</math> = edges of <math>G</math></li> </ul>	The minimum weight that forms a tree on $V$ .
<b>Generalization for graphical recognition problems:</b> Minimum Spanning Graphs (MSG) in directed (connected) graphs with weighted edges	<ul style="list-style-type: none"> <li><math>V</math> = vertices of <math>G</math></li> <li><math>E</math> = edges of <math>G</math></li> </ul>	The minimum weight that forms a directed (connected) graph <b>Note:</b> For any two nodes of this MSG there is a directed path that connects these two nodes. The direction of this path can be arbitrary, i.e., it is not required that the directed path starts at a specific node.
<b>Generalization for graphical recognition problems:</b> Minimum Spanning Hypergraphs (MSH) in hypergraphs with hyperedges	<ul style="list-style-type: none"> <li><math>V</math> = vertices of <math>G</math></li> <li><math>E</math> = hyperedges of <math>G</math></li> </ul>	Minimum weight that forms a hypertree on $V$ . <b>NOTES:</b> Two nodes are adjacent if and only if they share a common hyperedge. Two hyperedges are adjacent if and only if they share a common node. A hyperpath between two nodes in a hypergraph is a sequence of adjacent hyperedges that starts at the first node and ends in the other. A hypertree is a set of hyperedges where for any given pair of nodes there is exactly one hyperpath between them.



# The context of MathJournal's meta-recognition scheme

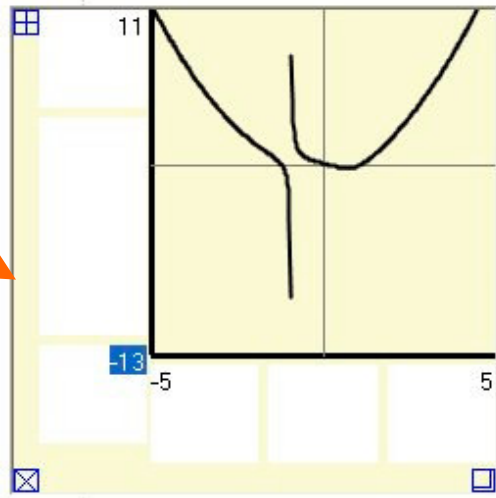
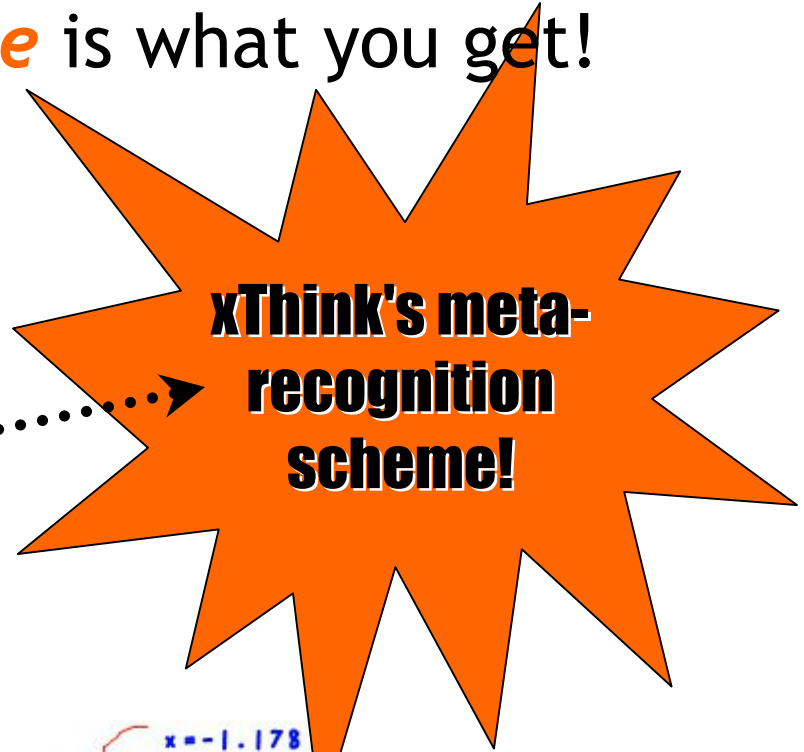
- Computer chips are optimally suited for this type of processing; the hardware that enables recognition will only get better.
- xThink's contribution to the Tablet PC, to the math software market, and to the field of recognition is two-fold:
  - a. layered and iterative recognition strategy that this presentation has described
  - b. an engine that works in the blink of an eye!
- xThink prototypes exist for application of this meta-recognition scheme to:
  - Hand-drawn diagrams
  - Hand-drawn expressions of graphical programming environments, e.g. Simulink, LabVIEW, HP-VEE, etc.
  - Hand-drawn flowcharts
  - Freehand sketches
  - Recognition of structured data of any kind, including recognition applied to datamining.

$$\frac{2x^5 - 3x + 1}{4x^3 - 2x^2 + 5}$$

$$\frac{(2(x)^5 - 3x + 1)}{(4(x)^3 - 2(x)^2 + 5)}$$

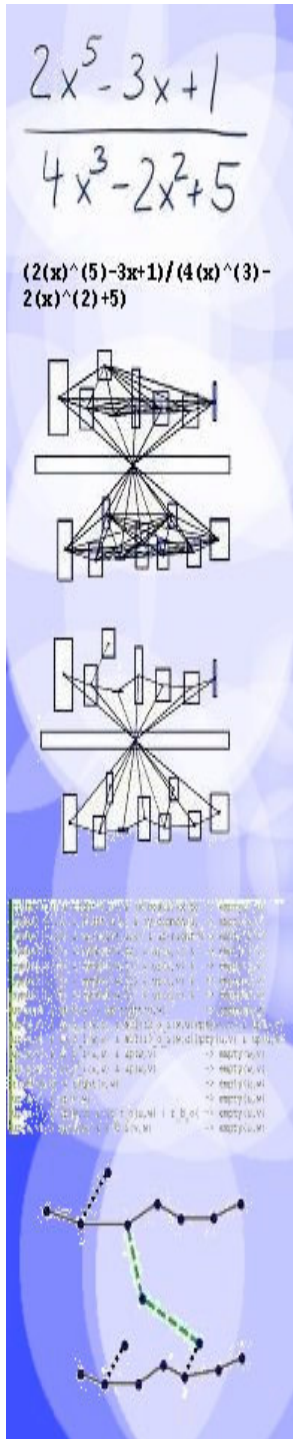
Write a problem. Generate a solution.  
 What you *don't see* is what you get!

$$\frac{2x^5 - 3x + 1}{4x^3 - 2x^2 + 5}$$



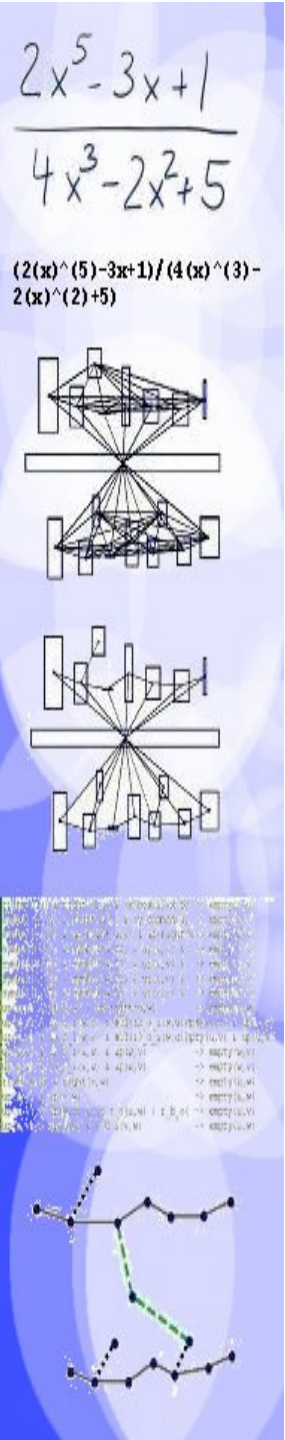
zeroes {  $x = -1.178$   
 $x = 0.336$   
 $x = 1$

extrema {  $(x, \min) = (-0.934, -66851312.906)$   
 $(x, \min) = (0.71, -0.142)$   
 $(x, \max) = (-0.934, 46585835.52)$



## Selected bibliography

- Anderson, R., Two-dimensional mathematical notation, In Syntactic Pattern Recognition, Applications, ed. K.S. Fu, 147-177, Springer, 1977
- Blostein, D., Grbavec, A., Recognition of mathematical notation, in H. Bunke and P. Wang (eds.), Handbook of Character Recognition and Document Image Analysis, 557-582, World Scientific Publishing, Singapore, 1997
- Calhoun, C., Stahovich, T.F., Kurtoglu, T., Kara, L.B., Recognizing multi-stroke symbols, AAI Spring Symposium, Sketch Understanding, 2002
- Chan, K.-F., Yeung, D.-Y., Mathematical expression recognition, Technical Report HKUST-CS99-04, 1999
- Chang, S., A method for the structural analysis of two-dimensional mathematical expressions, Information Sciences 2, 3, 253-272, 1970
- Ehrig, H., Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, World Scientific Publishing, 1997
- Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G., Handbook of Graph Grammars and Computing by Graph Transformation, vol. 2, World Scientific Publishing, 1999
- Miller, E.G., Viola, P.A., Ambiguity and constraints in mathematical expression recognition, Proceedings of AAI-98, 1998
- Okamura, H., Kanahori, T., Suzuki, M., Fukuda, R., Cong, W., Tamari, F. Handwriting interface for computer algebra, Proceedings of the Fourth Asian Technology Conference in Mathematics, 1999
- Shilman, M., Pasula, H., Russell, S., Newton, R., Statistical visual language models for ink parsing, AAI Spring Symposium, Sketch Understanding, 2002
- Ullmann, J.R., An algorithm for sub-graph isomorphism, Journal of the ACM 23(1):31-42, 1976
- Wang, Z., Faure, C., Structural analysis of handwritten mathematical expressions, Proc. 9th Int. Conf. on Pattern Recognition, 32-34, 1988



# Contact

**Teresa Shu, Ph.D., xThink, Inc.**

16908 Bar Harbor Bend

Round Rock, TX 78681

Phone: 512 238-1888

Fax: 512 238-1898

tshu@xthink.com

www.xthink.com

*"If there's time for a demonstration . . ."*