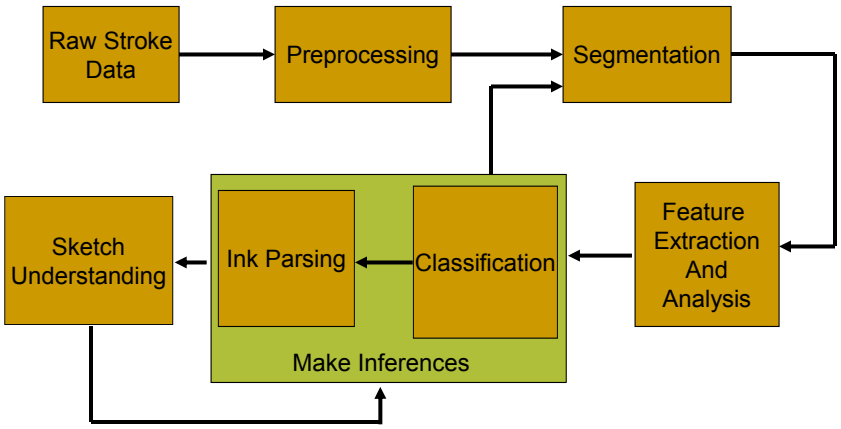
# Ink Parsing in Sketch-Based Interfaces

Lecture #10: Ink Parsing
Joseph J. LaViola Jr.
Fall 2011

 ©Joseph J. LaViola Jr.

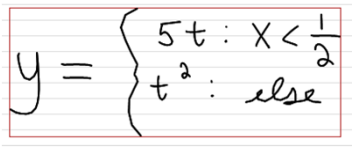
## Recall Pen-Based Interface Dataflow

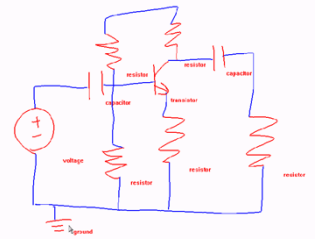



 ©Joseph J. LaViola Jr.

# Sketch Parsing

- Often recognition of strokes is insufficient
  - except for gestures
- Require an understanding of spatial relationships
  - good examples are mathematical expressions
- Higher level classifications
  - is it a word or a drawing?



www.engr.ucr.edu/~stahov/research/acsparc.htm

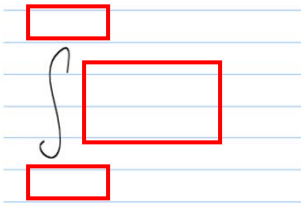


# Approaches to Sketch Parsing

- Top down vs. bottom up
- Focus on mathematical expressions
  - 2D (coordinate) grammars
  - graph rewriting
    - useful for other types of parsing as well (diagrams, tables, lists, etc…)
  - projection profile cutting
  - procedurally coded syntax rules
  - stochastic grammars
- Other parsing approaches
  - conditional random fields
  - statistical visual languages
  - many others

## 2D Grammars

- Grammar + spatial relationship rules
  - useful if a well defined syntax exists
  - looks for key symbols
- One Approach – Box Grammar
  - divide input into distinct areas based on symbol found





## Graph Rewriting

- Expressions represented as nodes and arcs
- Rewrite rules applied to graph to reduce it progressively
  - rules are also subgraphs
  - graph reduced to single node representing expression

## Graph Rewriting Example (Blostein and Grbavec 1996)

- Build
  - add edges between symbols (above, below, left, superscript, subscript)
- Constrain
  - Apply knowledge of notational conventions
    - remove contradictory associations
    - disambiguate horizontal lines
    - disambiguate dots
    - disambiguate diagonal associations
- Rank
  - Use information about operator precedence to group symbols into subexpressions
- Incorporate
  - Interpret subexpressions



## Projection Profile Cutting

- Used primarily in document analysis
- Uses horizontal and vertical projections of expression onto x and y axis
  - subdivides expression recursively
- Problem with expressions where symbols are close together (no white space)

# Procedurally Coded Syntax Rules

- Observations about domain coded programmatically
  - similar to rule based approach for recognition
- Sample rule for horizontal line

A length threshold of 20 pixels is used to classify a horizontal line as a short or long bar.

If it is a long bar and has symbols above and below, it is treated as a division.

If there are no symbols above, it is treated as a boolean negation.

If a short bar has no symbols above or below, it is treated as minus sign.

If it has symbols above or below, the combination symbols such as =, ≤, and ≥ are formed.

 ©Joseph J. LaViola Jr.

# Stochastic Grammars

- Used to deal with noisy data and spatial ambiguities
- Probabilities associated with each production rule
- For any sequence in a given parse – probability can be calculated
- Requires training

 ©Joseph J. LaViola Jr.

# MathPad² Parsing Approach

- Uses 2D coordinate grammar approach with some syntax rules
- Basic approach
  - preprocessing step (for functions)
  - sort list of symbols
  - parse functions – use grammar
  - process functions – handle spatial relationship testing
    - intermixed with parse functions



# Grammar (1)

```
<math_formula>     ::= <equation> | <expression>
<equation>         ::= <expression> <relational_op> <expression> |
                       <expression> ''='' <cond_expression>
<relational_op>    ::= ''='' | ''~='' | ''<'' | ''>'' | ''<='' | ''>=''
<cond_expression>  ::= ''{'' <cond_statement>
<cond_statement>   ::= ''if'' <expression> '':'' <logic_expression>
                       {''elseif'' <expression> '':'' <logic_expression> }
                       <expression> '': else''
<logic_expression> ::= <equation> <logical_op> <logic_expression> | <equation>
<logic_op>         ::= ''and'' | ''or''
<expression>       ::= <term> ''+'' <expression> |
                       <term> ''-'' <expression> |
                       <term> ''^'' <expression> |
                       <term>
<term>             ::= <factor> ''*'' <term>    |
                       ''('' <expression> '')'' |
                       <factor>
<factor>           ::= <sub_expression> ''/'' <factor> |
                       <sub_expression>
<sub_expression>   ::= <integral> | <derivative> | <summation> |
                       <function> | <terminal>
```

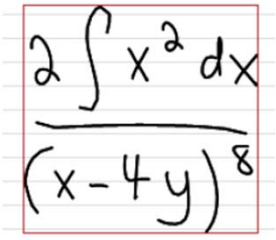
# Grammar (2)

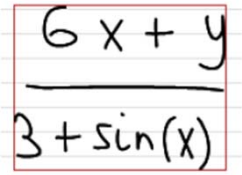
```
<integral>        ::= ``int(`` <expression> '','' <variable> ``)'' |
                      ``int(`` <expression> ``,'' <variable> ``,''
                               <expression> ``,'' <expression> ``)''
<derivative>      ::= ``diff(`` <expression> ``,'' <variable> ``)'' |
                      ``diff(`` <expression> ``,'' <variable> ``,''
                                <integer> ``)''
<summation>       ::= ``sum(`` <expression> ``)'' |
                      ``sum(`` <expression> ``,'' <expression> ``,''
                               <expression> ``)''
<function>        ::= <func_name> ``(`` <expression> ``)''
<func_name>       ::= ``sqrt'' | ``abs'' | ``log'' | ``exp'' |
                      ``sin'' | ``cos'' | ``tan'' | ``asin'' |
                      ``acos'' | ``atan''
<terminal>        ::= <variable> | <number>
<variable>        ::= <letter> |
                      <letter> ``_'' {<integer>} {<letter>} {<integer>}
<number>          ::=  <integer> |
                       <integer> ``.'' <unsigned_int>
<integer>         ::= <sign> <unsigned_int> | <unsigned_int>
<unsigned_int>    ::= <digit> <unsigned_int> | <digit>
<sign>            ::= ``+'' | ``-''
<digit>           ::= [0-9]
<letter>          ::= [a-z] | [A-Z] | [alpha-zeta]
```



# Parse functions

- High level parse
- Expression parse
- Sub-expression parse
- Symbol specific parsing
  - square root parse
  - integration parse
  - summation parse
  - fraction parse
- Factor parse
- Term parse



## Process functions

- Provide parse functions important info
- Deal with spatial relationships
  - implicit operators
  - fractions and square roots
  - summations , derivatives, integrals
  - Conditionals

$$x(t+h) = \begin{cases} \ell - r : x(t) > (\ell - r) \\ r : x(t) < r \\ x(t) + vh : \text{else} \end{cases}$$



## Reducing parsing decisions

- Use application to reduce decisions
- Implicit operators (no numbers have subscripts)
- Correct trig functions 5in -> sin
- Functions of time f(+) -> f(t)

# Readings


- D. Blostein and A. Grbavec, "Recognition of Mathematical Notation," in Handbook of Character Recognition and Document Image Analysis, Eds. H. Bunke and P. Wang, World Scientific, 1997, pp. 557-582.
- Chan, Kam-Fai and Dit-Yan Yeung. An Efficient Syntactic Approach to Structural Analysis of On-Line Handwritten Mathematical Expressions. Pattern Recognition, 33(3):375-384, March 2000.
- Ye, Ming, and Paul Viola. Learning to Parse Hierarchical Lists and Outlines Using Conditional Random Fields. International Workshop on Frontiers in Handwriting Recognition, 2004.
- Michael Shilman, Hanna M. Pasula, Stuart Russell, and Richard Newton, Statistical Visual Language Models for Ink Parsing. In Proc. AAAI Spring Symposium on Sketch Understanding, Stanford, March 2002.