

# A parsing algorithm for a class of relational context-free grammars

Scott MacLean

ORCCA JLM: 14 Mar. 2008

## Motivation

- pen-math project at UW - interactive pen-based mathematics
- input math via pen, manipulate and compute via computer algebra system
- Goal: a useful mathematical tool for students and researchers
- need a recognition system for math expressions
  - current system is brittle and has many hard-coded parameters
- idea: use a formal grammar to capture layout and semantics of math expressions
  - allows simple integration with symbol recognition: only semantically valid symbols are included in parse trees

## Relational CFGs (R-CFGs)

- our formalism is similar to [5]
- an R-CFG is a tuple  $(\Sigma, N, S, P, E, T, D, R, g, e, f)$ , where
  - $\Sigma$  is the terminal alphabet
  - $N$  is a set of non-terminal symbols
  - $S \in N$  is the start symbols
  - $P$  is a set of productions, each of the form  $P_0 \rightarrow p_1 r_1 \cdots r_{m-1} p_m$ , where  $P_0 \in N$ ,  $p_i \in \Sigma \cup N$ ,  $r_i \in R$
  - $E$  is the set of measurements/extents on which the grammar's relations  $R$  operate
  - $T$  is the set of all possible input elements (generally  $T = \Sigma \times E$ )
  - $D$  is a set of strict total orders  $d \subseteq E \times E$
  - $R$  is a set of relations  $r \subseteq E \times E$
  - $g : T \rightarrow \Sigma$  is a function giving the terminal symbol of an input element
  - $e : 2^T \rightarrow E$  is a function giving the extents of sets of input elements
  - $f : R \rightarrow D$  maps relations to orders
    - \* “ $d = f(r)$  is the order under which  $r$  acts”
- we use a normal form analogous to CNF where a production is either of the form  $P \rightarrow a$  or  $P \rightarrow ArB$  where  $a \in \Sigma$ ;  $A, B \in N$ ;  $r \in R$

## Generalization from 1-D Grammar

obtain an R-CFG from a CFG as follows:

- $E = \{(i, j) : 1 \leq i \leq j\}$
- $T = \Sigma \times E$
- $e(\{(\sigma, i, i)\}) = (i, i)$ ;  $e(t) = (\min \{e(a)_1 : a \in t\}, \max \{e(a)_2 : a \in t\})$
- one relation  $r = \{((i, j), (j + 1, k)) : 1 \leq i \leq j < k\}$  acting under the order  $d = \{(t_1, t_2) : e(t_1)_2 < e(t_2)_1\}$
- from a CFG production  $P_0 \rightarrow p_1 \cdots p_m$  obtain an R-CFG production  $P'_0 \rightarrow p_1 r \cdots r p_m$
- given an input string  $a_1 \cdots a_n$ , convert to  $(a_1, 1, 1), \dots, (a_n, n, n)$

## Modelling Mathematics

- $E = \{(l, t, r, b) : l < r, t < b\}$  is the set of axis-aligned boxes in the plane
- $T = \Sigma \times E$
- $e(\{(\sigma, b)\}) = b$ ;  $e(t) =$  the smallest box in  $E$  containing  $e(a)$  for all  $a \in t$
- one order per dimension:
  - $x = \left\{ \left( (l, t, r, b), (\hat{l}, \hat{t}, \hat{r}, \hat{b}) \right) : l < \hat{l} \right\}$
  - $y = \left\{ \left( (l, t, r, b), (\hat{l}, \hat{t}, \hat{r}, \hat{b}) \right) : t < \hat{t} \right\}$
- relations: ‘right-to’, ‘up-right-to’, ‘contains’ acting under  $x$ ; ‘down-to’, ‘down-right-to’ acting under  $y$

## R-CFG parsing

- parsing techniques have been suggested for many R-CFG variants
  - Earley parsing [6]
  - shift/reduce [2] (note that math is not LR however)
  - chart/table parsing [5] (and several related subsequent publications)
  - extensions with probabilistic relations [4]
  - also related to graph rewriting (eg. [1])
- many of these can be adapted to our formalism, but...
  - our grammar is not well-specified: input symbols and relations between them are guessed by the system
  - ideal: allow multiple possibilities for relations between symbols and recover all possible parse trees
  - existing techniques have exponential runtime in such a situation ([3])
- can we avoid exponential runtime?
  - not in the worst case if all parses must be enumerated: there may be exponentially many!
- idea: output-sensitive algorithm
  - poly-time phase which generates a structure from which all valid parse trees can be recovered
  - time to recover parse trees then depends on how many there are/how many are required by the program

## Constraints

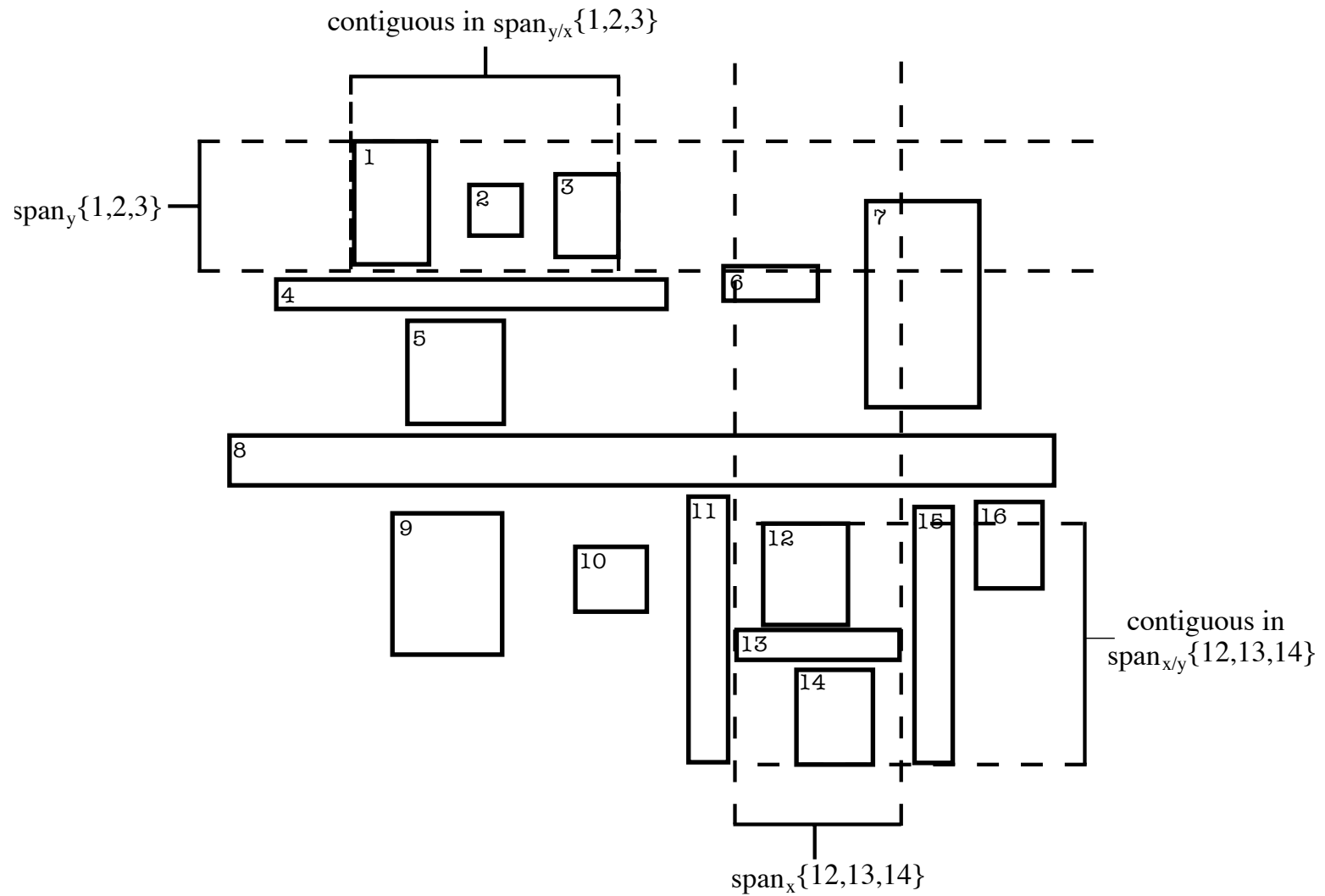
- we introduce constraints on acceptable parses to reduce the parsing problem to  $|D|$  related 1-D problems
- assume the grammar is in CNF
- several definitions based on order relations  $d, d' \in D$  and subsets of input  $I \subseteq A = \{a_1, \dots, a_n\}$ 
  - $\min_d I = a \in I$  such that  $(e(a), e(b)) \in d$  for all  $b \in I, b \neq a$
  - $\max_d I = a \in I$  such that  $(e(b), e(a)) \in d$  for all  $b \in I, b \neq a$
  - $\text{span}_d I = (\{a : (e(\min_d I), e(a)), (e(a), e(\max_d I)) \in d, a \in I\}, d)$ 
    - \* a totally-ordered set under  $d$ ; “span of  $I$  in  $d$ ”
  - $\text{span}_{d/d'} I = (\{a : (e(\min_d I), e(a)), (e(a), e(\max_d I)) \in d, a \in I\}, d')$ 
    - \* a totally-ordered set under  $d'$ ; “span of  $I$  in  $d$  wrt  $d'$ ”
  - $I$  is called *coherent wrt  $d'$*  if there exists some  $d \in D$  such that  $c \in I$  whenever  $(a, c), (c, b) \in d'$  for all  $a, b \in I, a \neq b, c \in \text{span}_{d/d'} I$ 
    - \* equivalent to saying that elements of  $I$  are contiguous in  $\text{span}_{d/d'} I$  ordered under  $d'$
  - a parse of  $P_0 \rightarrow ArB$  is called *coherent* if the set of input elements from  $I$  used by the parse, denoted  $I_{P_0}$ , is coherent wrt  $f(r)$
  - a parse of  $P_0 \rightarrow ArB$  is called *separable* if  $(a_A, a_B) \in f(r)$  for all  $a_A \in I_A, a_B \in I_B$
- our algorithm will find all the coherent, separable parses of the input

## Examples

In the following pages we will consider some subexpressions of the following math expression:

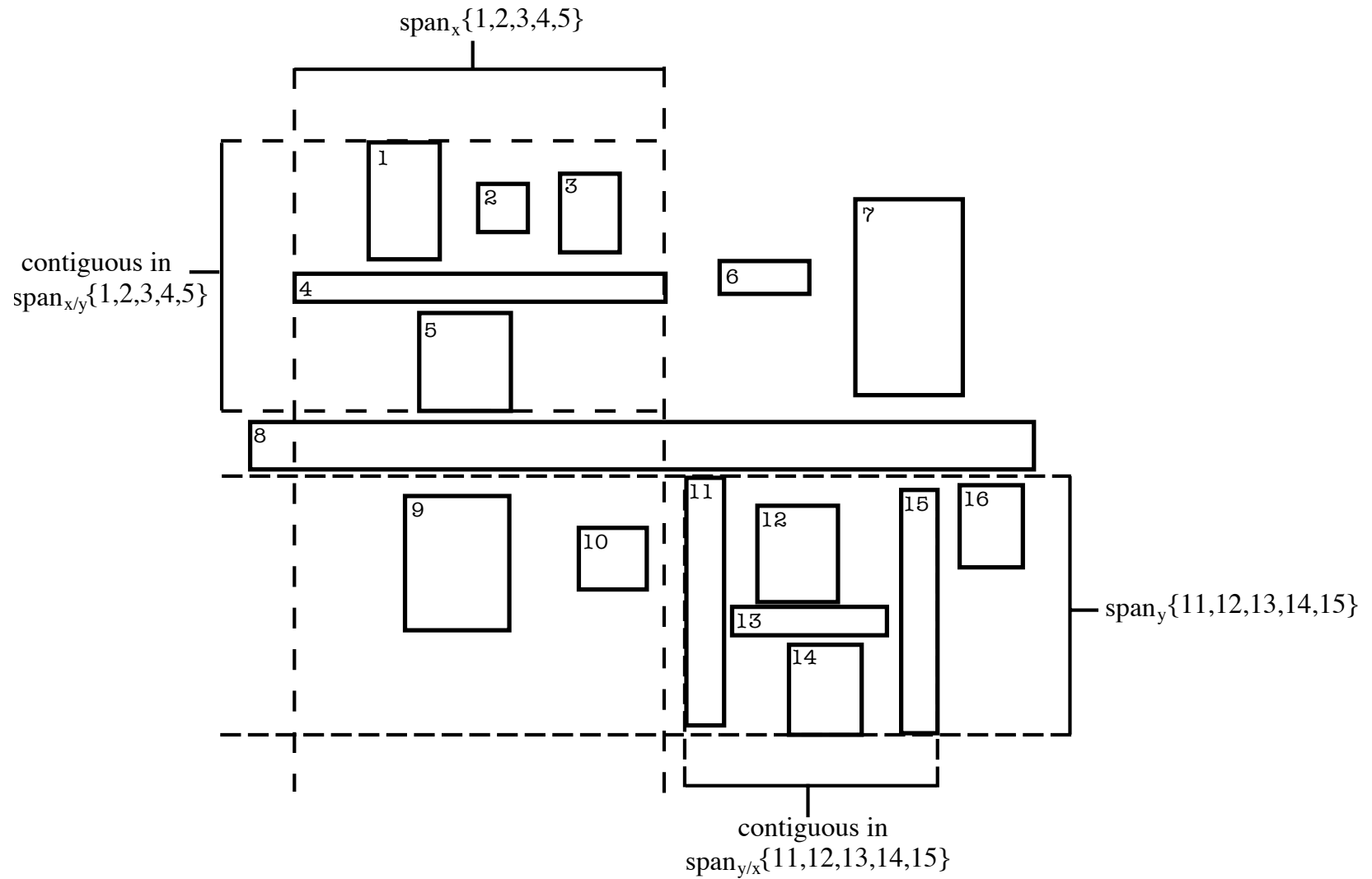
The diagram illustrates the decomposition of the mathematical expression  $2 + 7 = 6$  into subexpressions. The main expression is shown at the top. Below it, a large horizontal bar spans the width of the expression. Underneath this bar, the expression is broken down into subexpressions:  $4$ ,  $+$ ,  $(2)$ , and  $3$ . The subexpression  $(2)$  is further decomposed into  $2$  and  $7$ .

# Examples



(the numbers are just to identify the boxes)

# Examples



(the numbers are just to identify the boxes)

1. Consider applying span to boxes  $\{1, 2, 3\}$  and  $\{1, 2, 3, 4, 5\}$  (representing  $2 + z$  and  $\frac{2+z}{N}$ ):

|                                      |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |
|--------------------------------------|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| $\text{span}_{x/y}\{1, 2, 3, 4, 5\}$ | 1 | 3 | 2 | 4 | 5 | 9 |   |    |    |    |    |    |    |    |    |    |
| $\text{span}_x\{1, 2, 3, 4, 5\}$     | 4 | 1 | 9 | 5 | 2 | 3 |   |    |    |    |    |    |    |    |    |    |
| input ordered by $x$ :               | 8 | 4 | 1 | 9 | 5 | 2 | 3 | 10 | 11 | 6  | 13 | 12 | 14 | 7  | 15 | 16 |
| input ordered by $y$ :               | 1 | 7 | 3 | 2 | 6 | 4 | 5 | 8  | 11 | 16 | 15 | 9  | 12 | 10 | 13 | 14 |
| $\text{span}_y\{1, 2, 3\}$           | 1 | 7 | 3 | 2 |   |   |   |    |    |    |    |    |    |    |    |    |
| $\text{span}_{y/x}\{1, 2, 3\}$       | 1 | 2 | 3 | 7 |   |   |   |    |    |    |    |    |    |    |    |    |

The addition in boxes  $\{1, 2, 3\}$  is contiguous in  $\text{span}_{y/x}$  and the whole fraction is contiguous in  $\text{span}_{x/y}$ .

2. Considering applying span to boxes  $\{12, 13, 14\}$  and  $\{11, 12, 13, 14, 15\}$  (representing  $\frac{2}{z}$  and  $(\frac{2}{z})$ ):

|   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |
|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|
| $\text{span}_{x/y}\{12, 13, 14\}$         |   |   |   |   |   |   |   |    |    | 12 | 13 | 14 |    |    |    |    |    |
| $\text{span}_x\{12, 13, 14\}$             |   |   |   |   |   |   |   |    |    | 13 | 12 | 14 |    |    |    |    |    |
| input ordered by $x$ :                    | 8 | 4 | 1 | 9 | 5 | 2 | 3 | 10 | 11 | 6  | 13 | 12 | 14 | 7  | 15 | 16 |    |
| input ordered by $y$ :                    | 1 | 7 | 3 | 2 | 6 | 4 | 5 | 8  | 11 | 16 | 15 | 9  | 12 | 10 | 13 | 14 |    |
| $\text{span}_y\{11, 12, 13, 14, 15\}$     |   |   |   |   |   |   |   |    |    | 11 | 16 | 15 | 9  | 12 | 10 | 13 | 14 |
| $\text{span}_{y/x}\{11, 12, 13, 14, 15\}$ |   |   |   |   |   |   |   |    |    | 9  | 10 | 11 | 13 | 12 | 14 | 15 | 16 |

The fraction in boxes  $\{12, 13, 14\}$  is contiguous in  $\text{span}_{x/y}$  and the whole expression is contiguous in  $\text{span}_{y/x}$ .

## Observations

- written math has a block-based structure
- the span operator cuts the input up into these blocks
- expressions are separable and coherent
- core idea of algorithm:
  - for every  $d \in D$ , sort input  $A = \{a_1, \dots, a_n\}$  under order  $d$  into  $A_d$
  - for each  $A_d$ , take every contiguous subsequence (all possible  $\text{span}_d$  sets) and re-sort by every other ordering  $d' \in D, d' \neq d$  (all possible  $\text{span}_{d/d'}$  sets)
  - build up parses in these re-sorted spans
  - work with parses of length  $l = 1, \dots, n$  in spans of length  $l, l + 1, \dots, n$  via dynamic programming
  - this is a generalization of the CYK algorithm

## Parsing algorithm

**Input:**  $A = \{a_1, \dots, a_n\} \subseteq T; \Sigma, N, P, D, R, e, f$

**Output:** an array  $B$  of sets of parse trees indexed by  $[d; P_k; i, j]$  for all  $d \in D, P_k \in P, 1 \leq i \leq j \leq n$

**for every**  $d \in D$ , **define**  $\hat{d} : A \rightarrow \{1, \dots, n\}$  by  $\hat{d}(a_i) =$  the position of  $a_i$  in an enumeration of  $A$  ordered by  $d$

**for every**  $d \in D$ , **let**  $A_d$  be an array of the elements of  $A$  sorted into increasing order under  $d$

**for**  $i = 1$  **to**  $n$

**for every** production  $P_k \in P$  of the form  $P_k \rightarrow g(a_i)$

**for all**  $d \in D$ , add  $a_i$  to  $B[d; P_k; \hat{d}(a_i), \hat{d}(a_i)]$

**for**  $l = 2$  **to**  $n$  (\*)     // parse length

**for**  $l' = l$  **to**  $n$      // span length

**for**  $i = 0$  **to**  $n - l'$      // span start

**for**  $j = 0$  **to**  $l' - l$      // parse start

**for every**  $d \in D$

**let**  $S = A_d[i + 1, \dots, i + l']$

**for every** production  $P_k \in P$  of the form  $P_k \rightarrow P_1 r P_2$

**let**  $S = S[j + 1, \dots, j + l]$

            sort  $S$  into increasing order under  $f(r)$

**for**  $s = 1$  **to**  $l$

**let**  $S_1 = \{S[m] : 1 \leq m \leq s\}$

**let**  $S_2 = \{S[m] : s + 1 \leq m \leq l\}$

**for every** pair  $(d_1, d_2) \in D \times D$

**for every**  $(a, b) \in B[d_1; P_1; \min_{d_1}(S_1), \max_{d_1}(S_1)] \times B[d_2; P_2; \min_{d_2}(S_2), \max_{d_2}(S_2)]$

**if**  $(e(a), e(b)) \in r$  **then**

                    add  $(a, b)$  to  $B[f(r); P_k; S[1], S[l]]$

## Correctness

**Claim.** *If  $t$  is a separable, coherent parse tree of a production  $P_0 \rightarrow P_1 r P_2$  of an R-CFG and  $A' \subseteq A$  is the set of input elements appearing in  $t$ , then  $t$  is stored in  $B [f(r); P_0; \min_{f(r)} A', \max_{f(r)} A']$  after iteration  $l = |A'|$  of loop (\*).*

## Complexity

- runtime  $\mathcal{O}(|P||D|^3 n^5)$  (...but it's a small coefficient...)
- space usage  $\mathcal{O}(|P||D|n^2 + |D|^2 n^3)$  where the first term is for storing  $B$  and the second is for lookup tables
- slow, but not exponential. Regular CFG parsing takes time  $\mathcal{O}(|P|n^3)$
- polynomial dependence on dimension/no. orderings

## Comments

- our constraints seem reasonable for math recognition
- in practice, the input, relations in  $R$ , and output parse trees are fuzzy sets
- we omit discussion of how relation sets are guessed and how parse trees are recovered from  $B$
- future work
  - how to handle incomplete input? (eg. missing denominator)
  - how to efficiently update parses incrementally? (eg. user fills in said missing denominator)
  - improve integration with symbol recognition (feedback loops)

## References

1. 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.
2. Costagliola, G., Polese, G., *Extended positional grammars*, Proc., 2000 IEEE Symposium on Visual Languages, IEEE, 2000, pp.103-110.
3. Miller, E. G. and Viola, P. A., *Ambiguity and constraint in mathematical expression recognition*, Proc. of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI, 1998, pp. 784-791.
4. Shilman M. et al., *Statistical Visual Language Models for Ink Parsing*, Proc. AAAI Spring Symposium on Sketch Understanding, AAAI, 2002, pp. 126-132.
5. K. Wittenburg, L. Weitzman, and J. Talley, *Unification-Based Grammars and Tabular Parsing for Graphical Languages*, Journal of Visual Languages and Computing 2, Elsevier, 1991, pp. 347-370.
6. Wittenburg, K., *Earley-style parsing for relational grammars*, Proc., 1992 IEEE Workshop on Visual Languages, IEEE, 1992, pp.192-199.