Towards a Parser for Mathematical Formula Recognition

Alan Sexton

University of Birmingham, UK

Joint work with Amar Raja, Matthew Rayner and Volker Sorge

This work was supported by EPSRC grant EP/D036925/1

Overview

Part 1: Our approach to OCR and MFR

Part 2: Projection Profile Cutting (Reloaded)

Part 3: Graph Grammar Rule Exploration

Our Approach

- Tokenizer is a database driven glyph recogniser
  - Approximately 5300 LaTeX symbols in 8 font sizes
  - Bounding box, baseline position, relation to other glyphs, etc.
- Structural Analysis for character and formula recognition
  - Graph Grammar Parsing
- Semantic Analysis for validation and interfacing to other tools
  - Domain specific plug-in

Our Approach

- Tokenizer is a database driven glyph recogniser
  [MKM05, GREC05]
  - Approximately 5300 LaTeX symbols in 8 font sizes
  - Bounding box, baseline position, relation to other glyphs, etc.
- Structural Analysis for character and formula recognition
  - Graph Grammar Parsing
- Semantic Analysis for validation and interfacing to other tools
  [MKM05, ICDAR05, ISSAC06, MKM06]
  - Domain specific plug-in

Graph Grammar Parsing

- String grammar parsing — build parse tree by rewriting patterns of symbols to a single symbol
- Graph grammar parsing — build parse tree by rewriting subgraph patterns to a smaller subgraph, usually a single node
- Graph connectivity based on spatial relationships of glyphs in formula
- Problems
  - Subgraph matching is hard: easier if we reduce connectivity without losing critical information
  - Rule discovery
  - Rule interaction

Overview

Part 1: Our approach to OCR and MFR

Part 2: Projection Profile Cutting (Reloaded)

Part 3: Graph Grammar Rule Exploration

Projection Profile Cutting

- Standard technique in document analysis to identify disconnected components and their spatial relationships
  - Scan image horizontally looking for vertical lines that partitions image without cutting any component
  - Scan each partition vertically looking for horizontal lines that further partitions each sub-image without cutting any component
  - repeat until only atomic components left
- Top down rather than bottom up approach
- Reasonably successful technique, but fails on enclosed symbols:

$\sqrt[\alpha]{a+b}$, $\boxed{W_t - f \subseteq V(p_i) \subseteq W_t}$

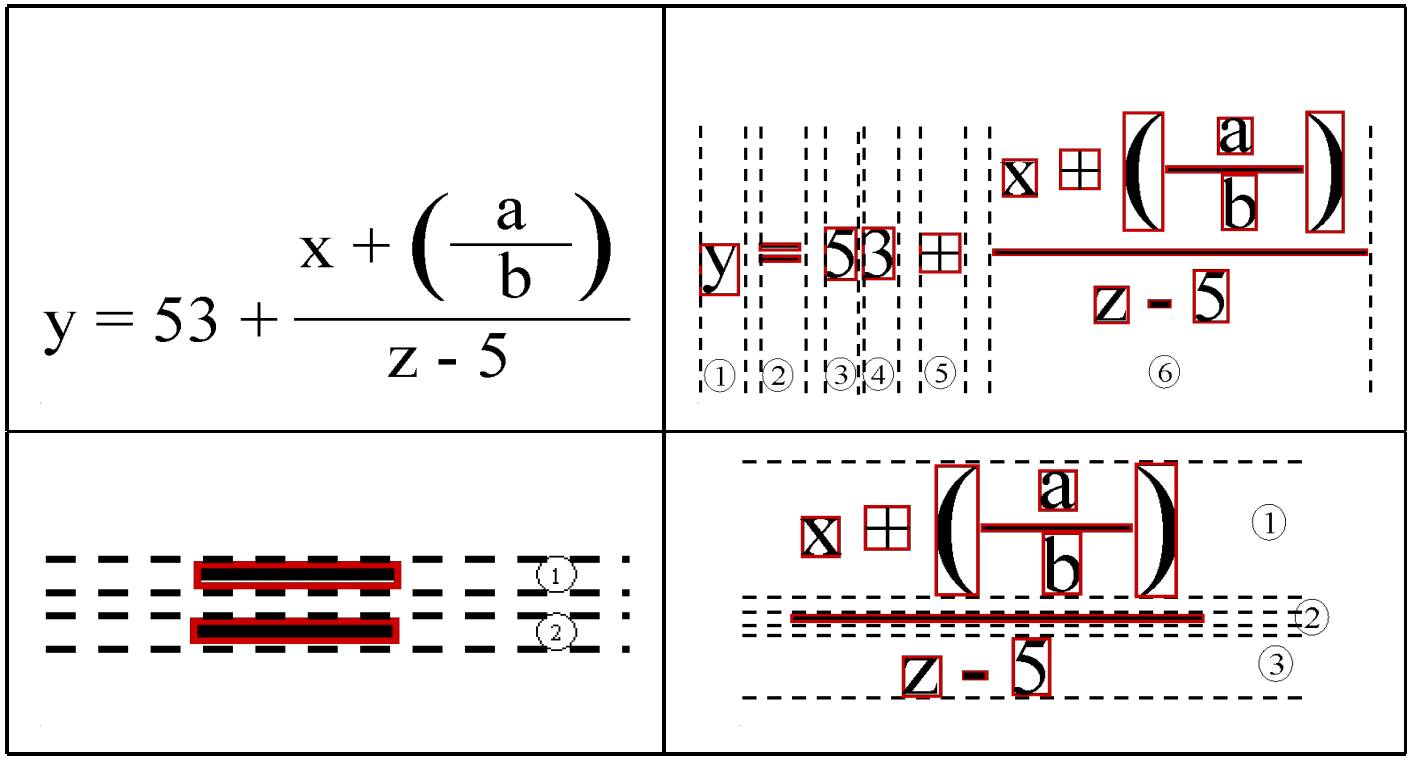
PPC in Our Context

- Glyph recogniser already identifies disconnected components and absolute spatial locations, but not their relative spatial relationships

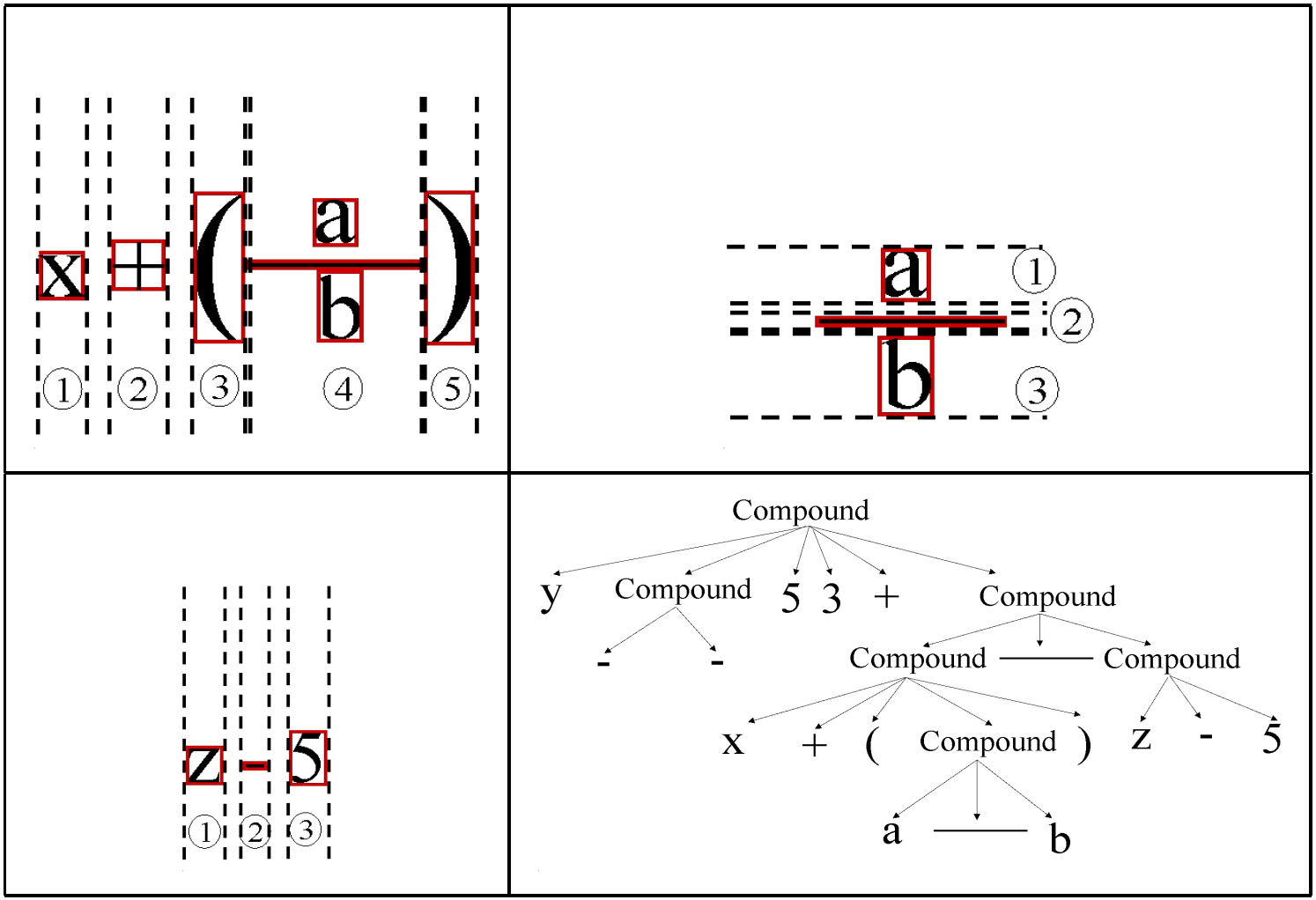
PPC in Our Context

- Glyph recogniser already identifies disconnected components and absolute spatial locations, but not their relative spatial relationships
- Use PPC as pre-parse step to guide sub-graph matching
- Use PPC to optimise character reconstruction from glyphs
- Use glyph recogniser information to guide PPC in enclosed symbol handling

PPC in Action

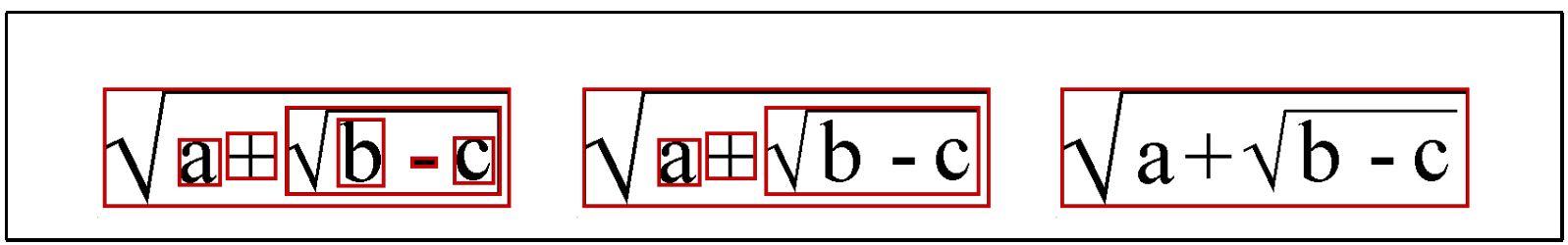


PPC in Action



PPC and enclosed symbols

- PPC normally cannot handle enclosed symbols:
  - No vertical or horizontal partition possible
  - No hint that sub-structure exists
  - Special processing necessary for each type of enclosure
- Our context: glyph extraction tells us that sub-structure exists
  - Generic solution: peel off enclosing glyph and generate node in parse tree



Overview

Part 1: Our approach to OCR and MFR

Part 2: Projection Profile Cutting (Reloaded)

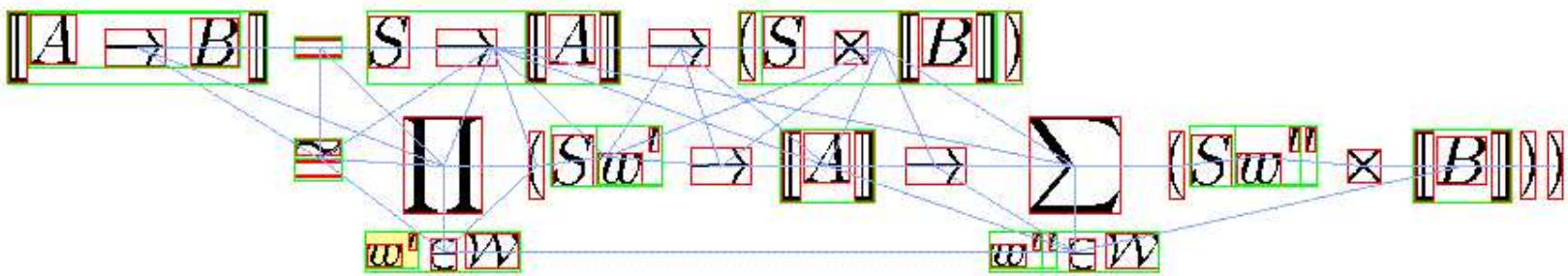
Part 3: Graph Grammar Rule Exploration

Graph Grammar Rewriting Aims

- Framework for exploring rulesets
  - Modular, extendible rulesets
  - Discovery of rulesets
  - Designing rulesets for character reconstruction from glyphs, glyph reconstruction from broken glyphs
  - Debugging rulesets, identifying conflicting rules

Constructing the Graph

- Line of Sight connectivity (rather than full or compass point connectivity)
- Initial nodes are lazy lists of matching glyphs in best match order

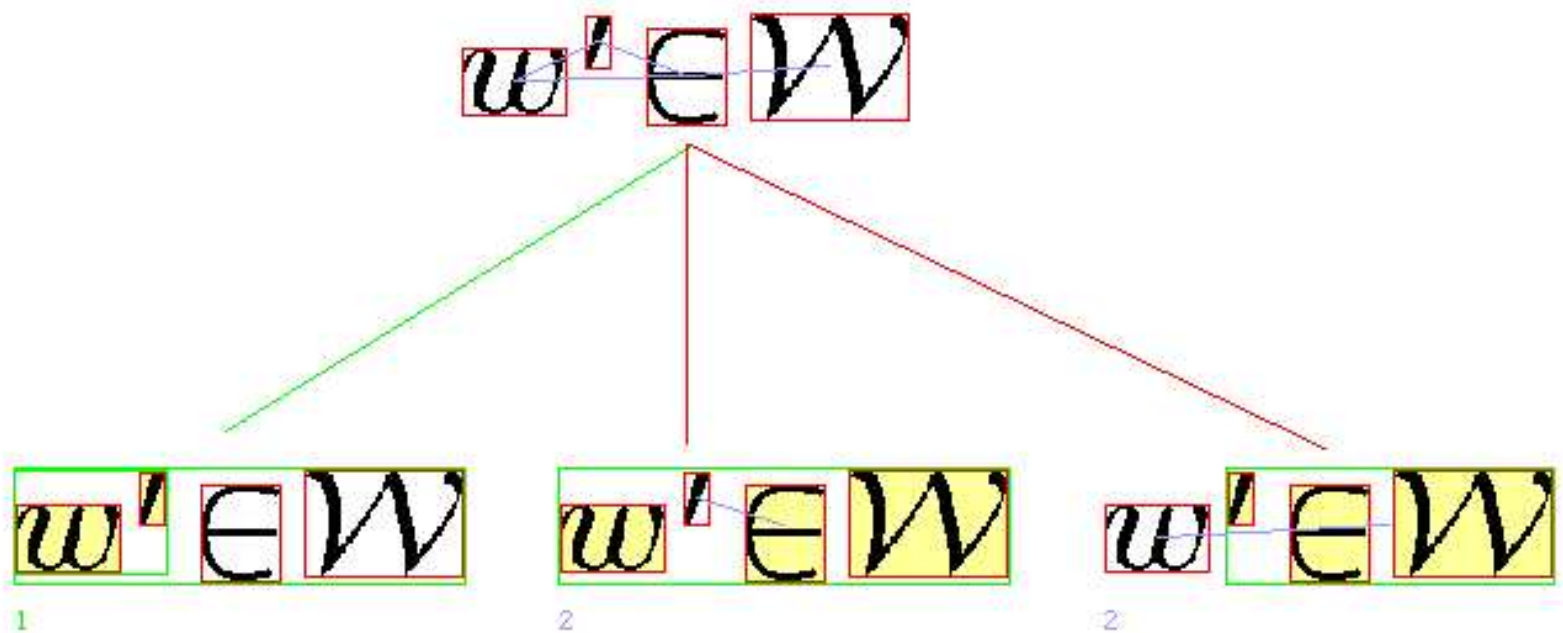


Rules

- Rule is based on a principal node:
  - A range of spatial relationships between the principal node and ancilliary nodes can be specified.
  - Ancilliary nodes can be specified as non-terminal (e.g. *Relational Operator*) or terminal symbols (e.g. "+")
  - Any number of ancilliary node connections can be included in a rule
  - Precedence between rules in a ruleset can be specified

Matcher

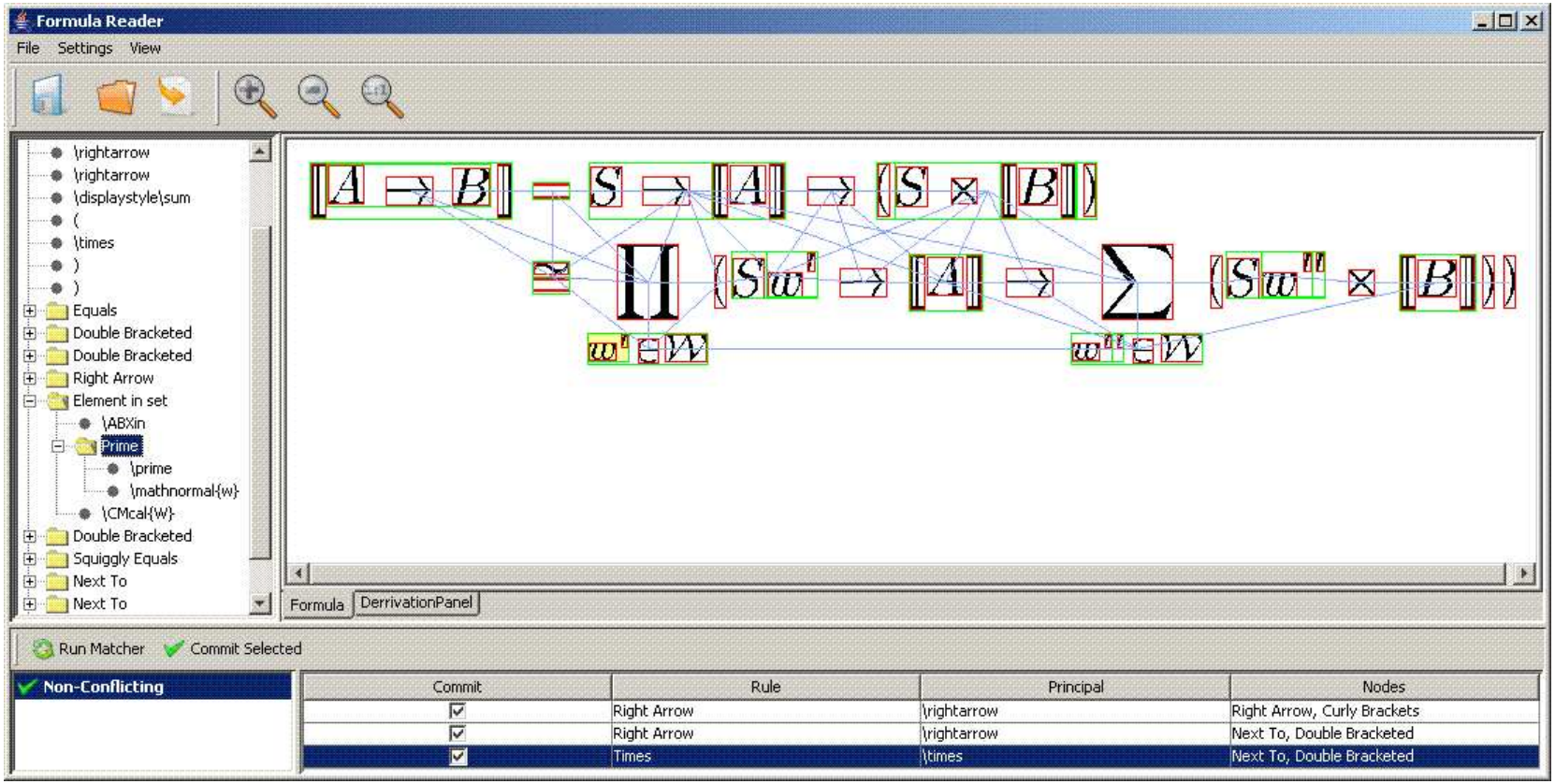
- Matcher exhaustively checks for applicability of all rules to all possible nodes and all permutations of possible ancilliary nodes
- Identifies rules that conflict on the current graph
- Allows non-conflicting rules to be applied automatically
- Allows automatic choice of conflicting rules in a sequence that minimises the number of final unconsumed nodes.



Formula Reader

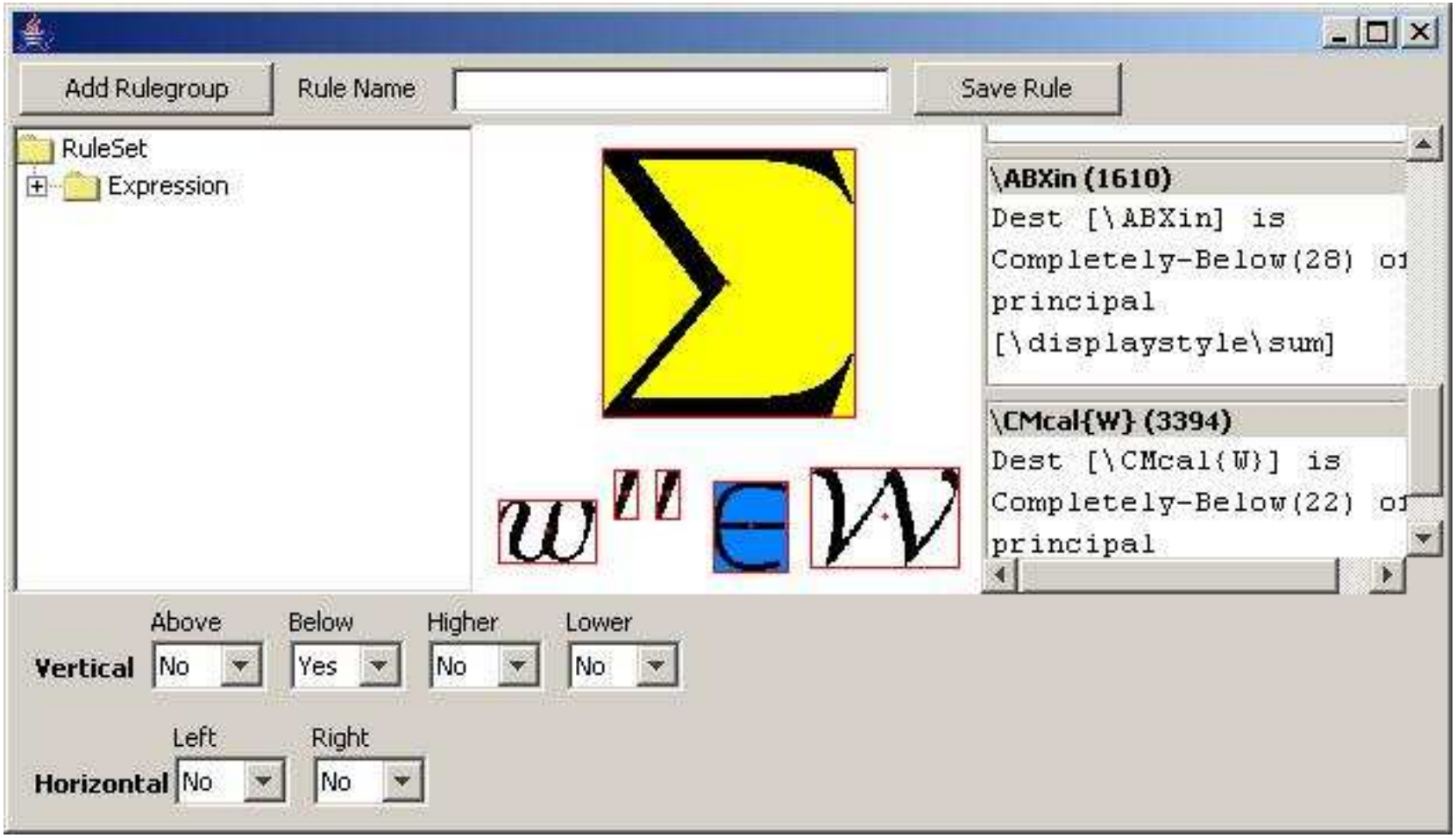
- Ruleset visualisation and exploration tool
- Allows manual control over the parse process
- Displays current state of the parse *forest*
- Draws boxes around recognised sub-formulas
- Draws connections between nodes that have not yet been consumed
- Selecting in the graph will select/highlight the corresponding part of the parse forest and vice versa
- Current set of applicable rules are displayed with conflicting groups identified
- Parsing can be executed in manual steps: the default choices are pre-selected for speed, but the user can choose any valid alternative

Formula Reader



Rule Builder

- A *Rule Builder* interface allows the user to select a group of nodes at any time, construct a rule out of them and dynamically add the rule to the rule set



Finally...

- Conclusions
  - New productive combination of glyph extraction and projection profile cutting
  - New solution to PPC problem of enclosed glyphs
  - A new tool to explore and experiment with the design of graph grammar rulesets
- Future work
  - Rule definitions and matcher are too simple, although the Formula Reader and Rule Builder tools have proven very useful in helping us to visualise and understand the design choices in rule set development: we intend to keep the best ideas from the tools but apply it to more general types of graph rewrite systems.
  - Explore integration between PPC and graph grammar rewrite rules