DAVID,
or the power of simple Algebraic Approach
against GOLIATH,
the giant task of Document processing

Approach

The following items describe our proposal:

- associate a **type** with each document
- model that type with the usual mathematical data models
  (those from the *set theory*)
- use that model to derive an adequate generic internal representation for the document
- see the processing task (formatting and printing, translation, information retrieval, etc.) as an **operator** over document types
- define a function to implement that operator

DAVID,
or the power of simple Algebraic Approach
against GOLIATH,
the giant task of Document processing

1 Advantages

The Algebraic Approach to Document processing brings the following gains:

- a formal (sintetic and rigorouse) way to specify documents and their transformation
- the reuse of a traditional approach (method and tools) to programming
- the availability of rapid prototyping environments

  and, *the last, but not the least*:

  - the ability to describe, inside the same framework, the transformation of any

kind of data (not only documents) into documents

- or to retrieve, inside the same framework, any kind of data from documents

DAVID,
or the power of simple Algebraic Approach
against GOLIATH,
the giant task of Document processing

2 Other topics

The Algebraic Approach to Document processing is:

- standard document markup languages (like SGML) are easily mapped into an algebraic system
- standard representation schemes (like decorated abstract syntax trees) are easily mapped into an algebraic system

  and, *the last, but not the least*:

  - the ability to describe, inside the same framework, the transformation of any kind of data (not only documents) into documents

– or to retrieve, inside the same framework, any kind of data from documents