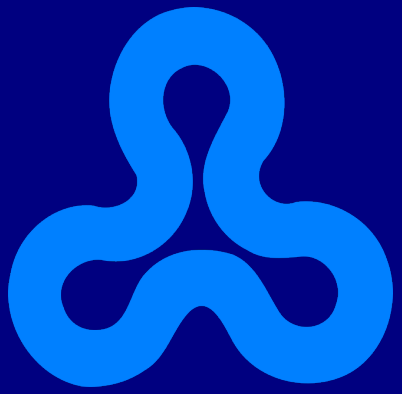
Language Oriented Programming with Cedalion



Boaz Rosenan
Dept. of Computer Science
The Open University of Israel

Adviser: Prof. David H. Lorenz

Agenda

- **Overview:**
  - Language Oriented Programming (LOP)
  - LOP Languages
  - Cedalion, as an LOP Language
- **Case Study:**
  - DNA Microarray Design

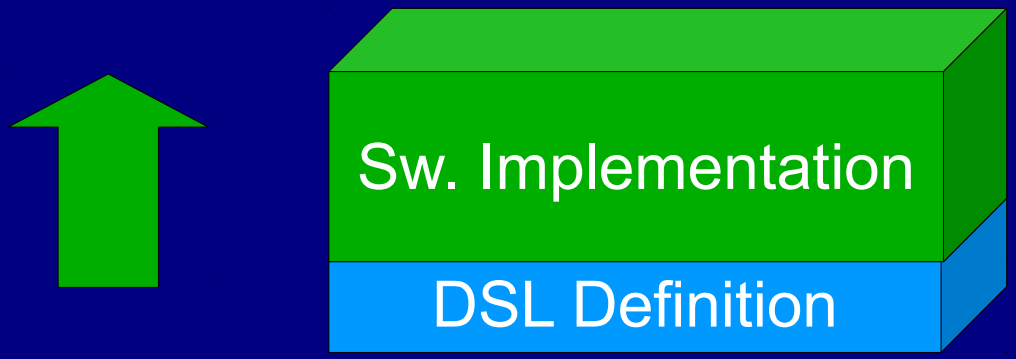
Language Oriented Programming (LOP): Rethinking Software Development

- **Traditional Thinking**
  - Designing our ***software*** for a *programming language*.
- **New Thinking**
  - Design *programming languages* for our ***software***.
- **The Role of DSLs in LOP**
  - Implement them if you need to.
  - Keep them focused and interoperable.

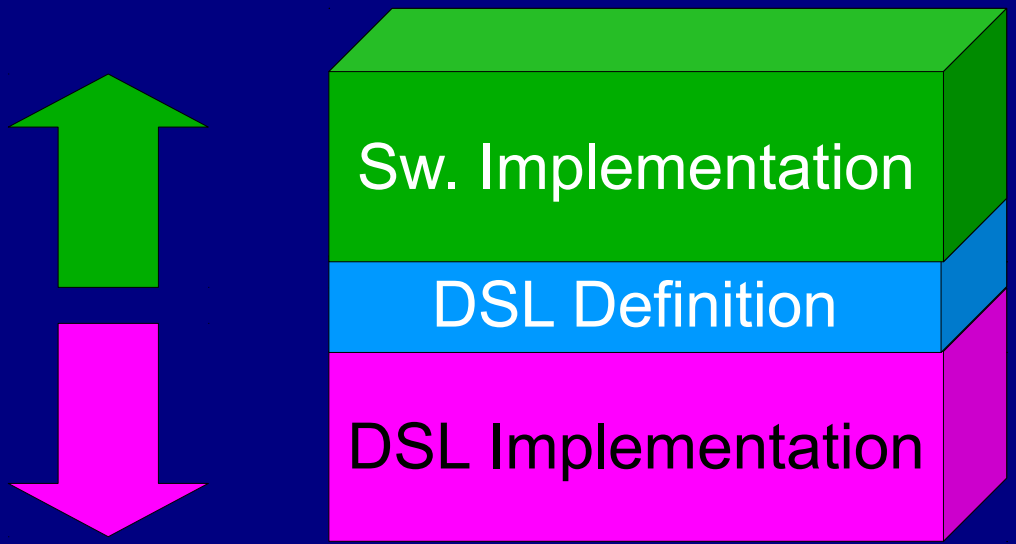
LOP: Middle Out

LOP: Middle Out

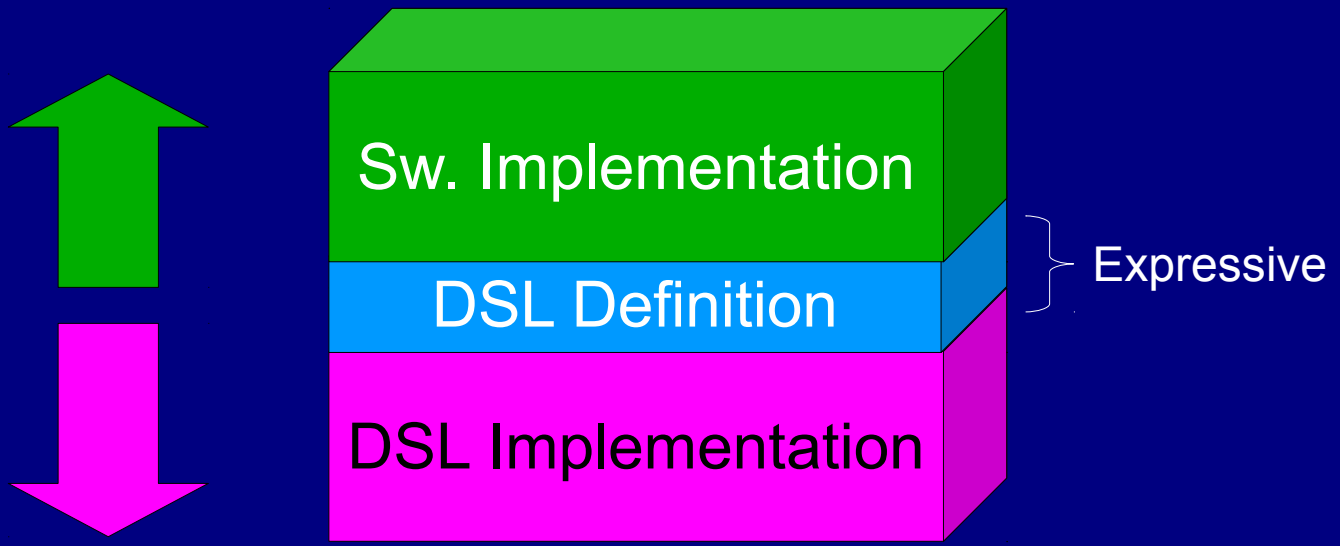
LOP: Middle Out



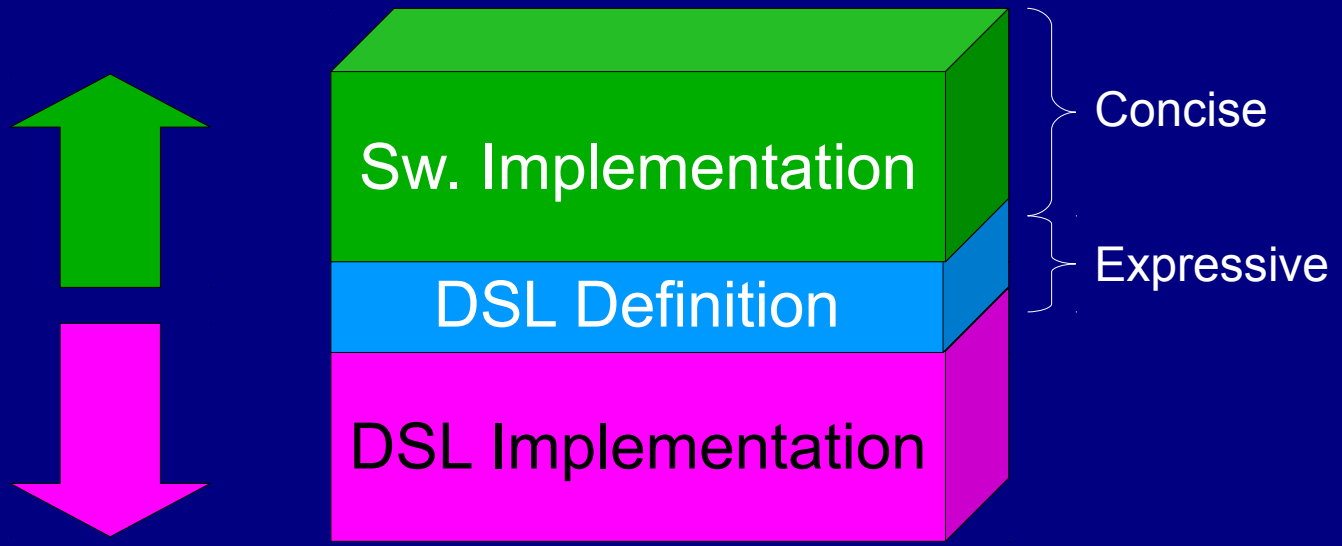
LOP: Middle Out



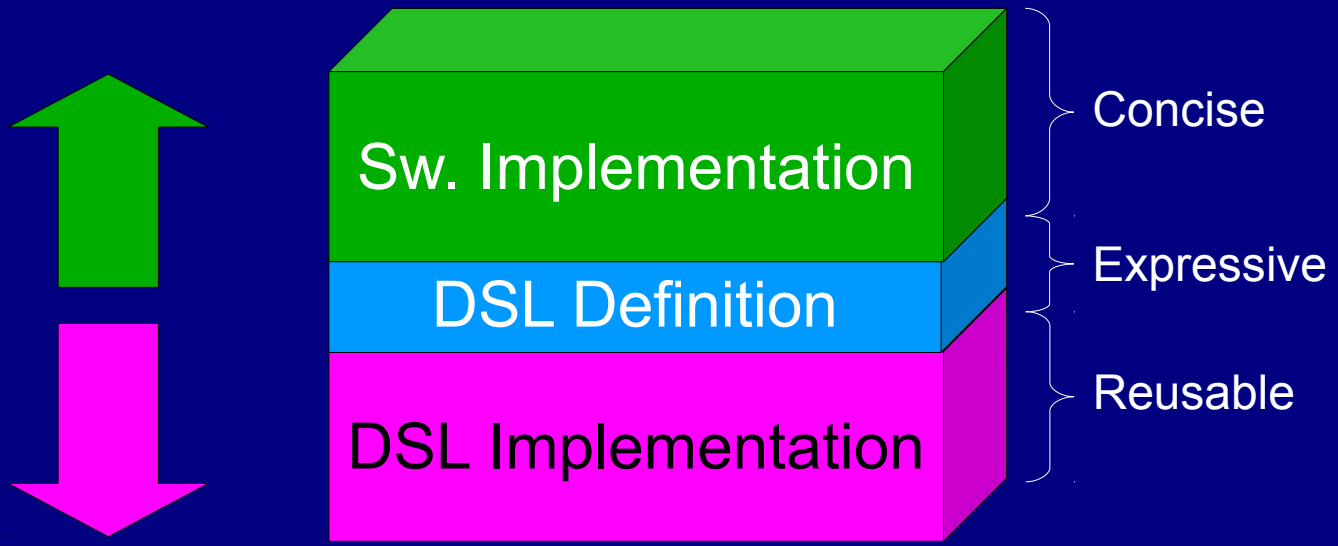
LOP: Middle Out



LOP: Middle Out



LOP: Middle Out



State of the Art

State of the Art

Language Workbenches

- IDEs for developing DSLs.
- Use External DSLs.
- Use Projectional-Editing [Fowler05].
- DSLs: Easy to use; hard to implement.
- Examples: MPS, Intentional.

State of the Art

Language Workbenches

- IDEs for developing DSLs.
- Use External DSLs.
- Use Projectional-Editing [Fowler05].
- DSLs: Easy to use; hard to implement.
- Examples: MPS, Intentional.

Internal DSLs

- Internal to a host language.
- First used in Lisp in the 1960s.
- DSLs: Easy to implement; limited by the host language.

State of the Art

Language Workbenches

- IDEs for developing DSLs.
- Use External DSLs.
- Use Projectional-Editing [Fowler05].
- DSLs: Easy to use; hard to implement.
- Examples: MPS, Intentional.

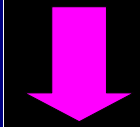
Internal DSLs

- Internal to a host language.
- First used in Lisp in the 1960s.
- DSLs: Easy to implement; limited by the host language.

State of the Art

Language Workbenches

- IDEs for developing DSLs.
- Use External DSLs.
- Use Projectional-Editing [Fowler05].
- DSLs: Easy to use; hard to implement.
- Examples: MPS, Intentional.



Internal DSLs

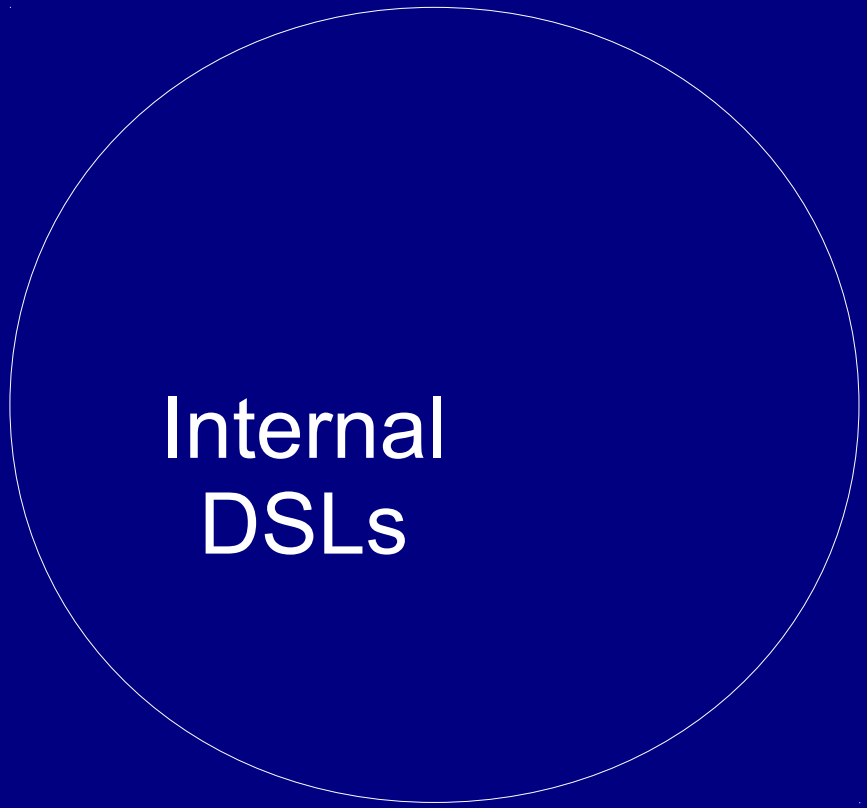
- Internal to a host language.
- First used in Lisp in the 1960s.
- DSLs: Easy to implement; limited by the host language.

LOP Languages: Rethinking LOP

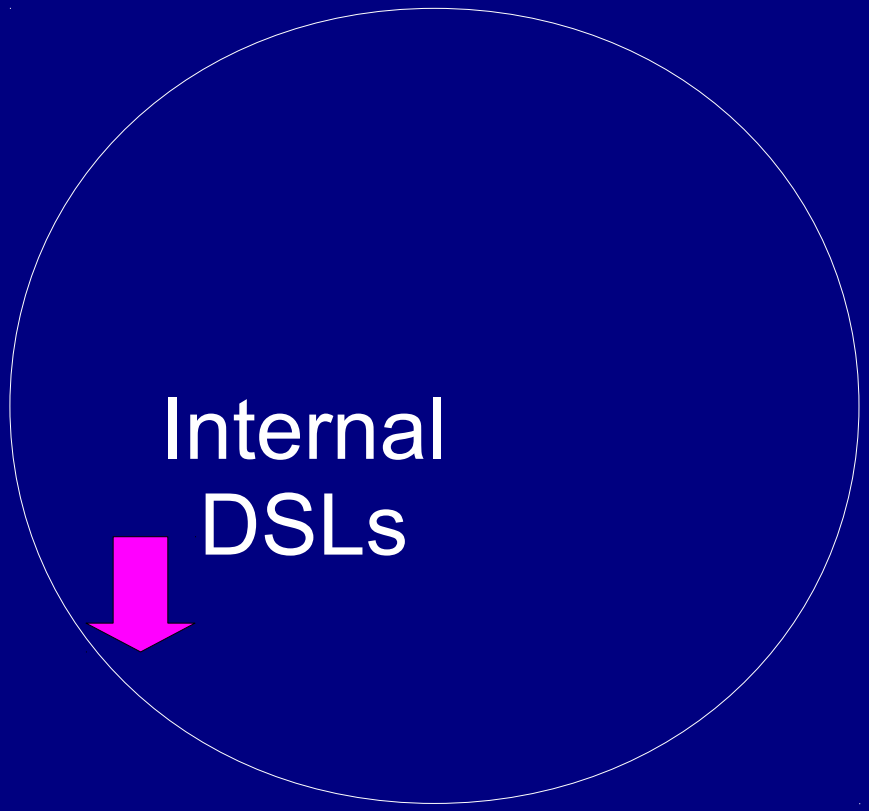
- **LOP Languages**
  - Programming languages supporting LOP.
  - Just like OOP languages support OOP.
- **Definition**
  - An *LOP Language* is a programming language that can host *internal DSLs*, allows the definition and enforcement of *DSL schema*, and features extensible *projectional-editing*.

LOP Language Design Space

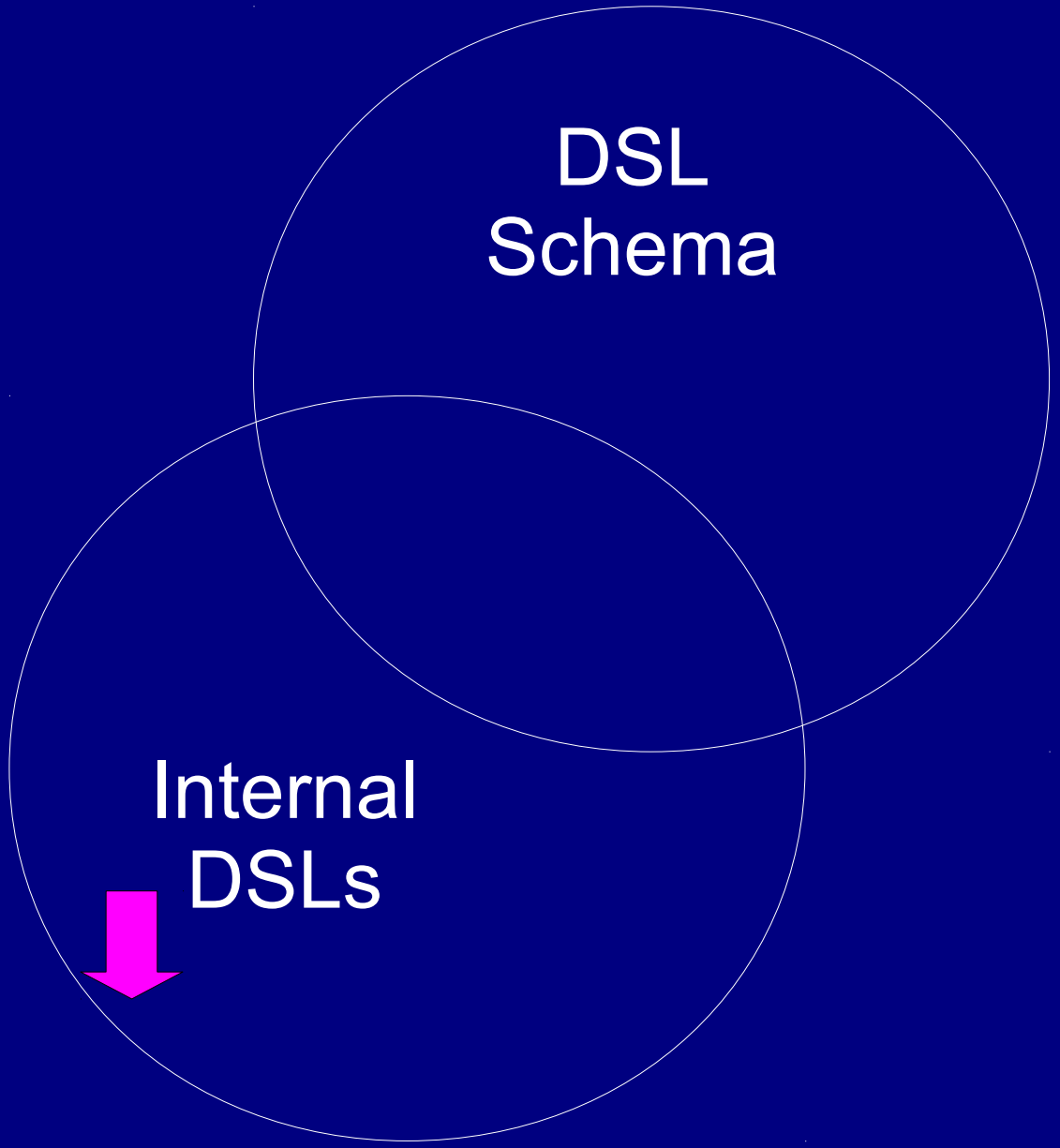
LOP Language Design Space



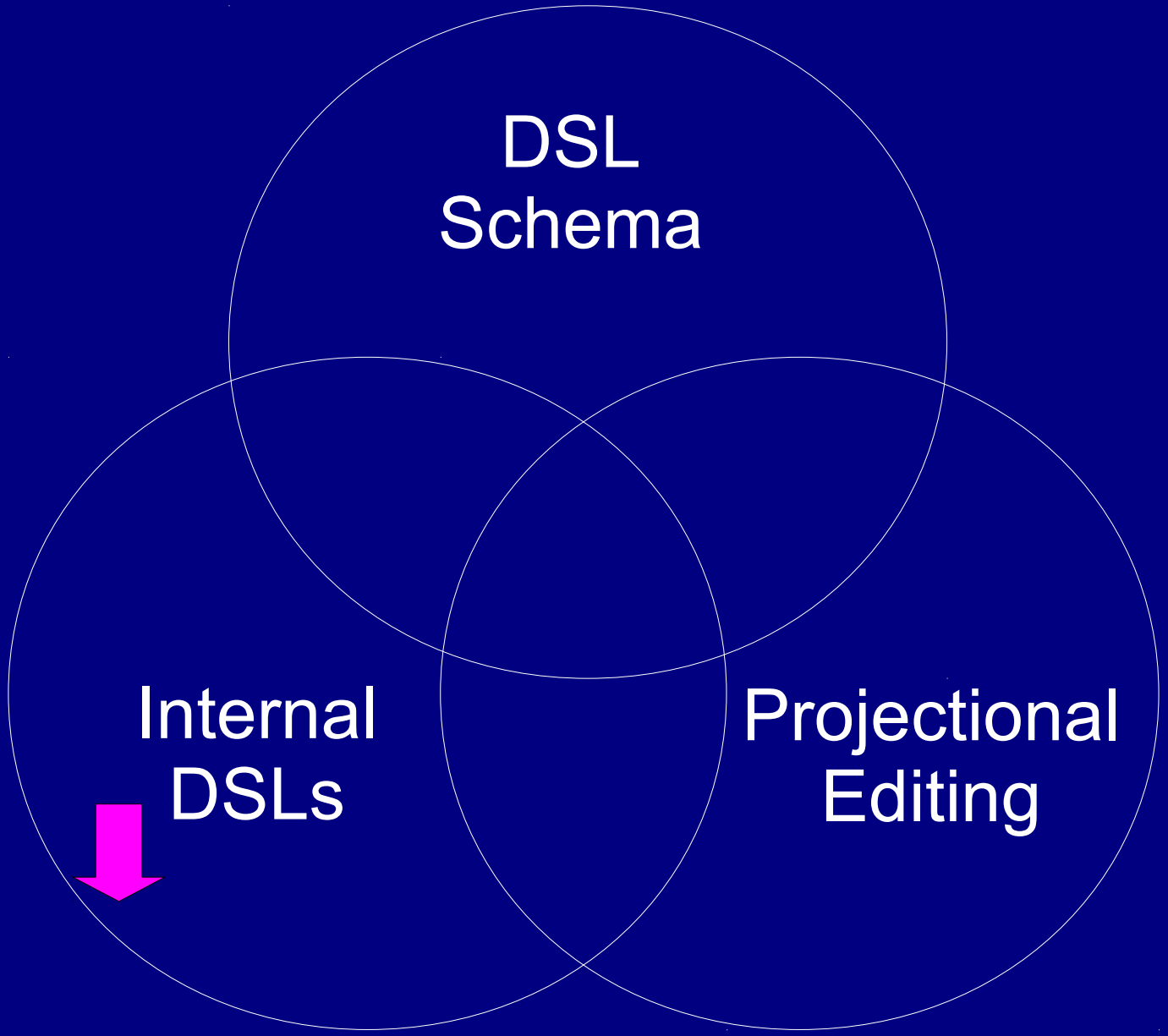
LOP Language Design Space



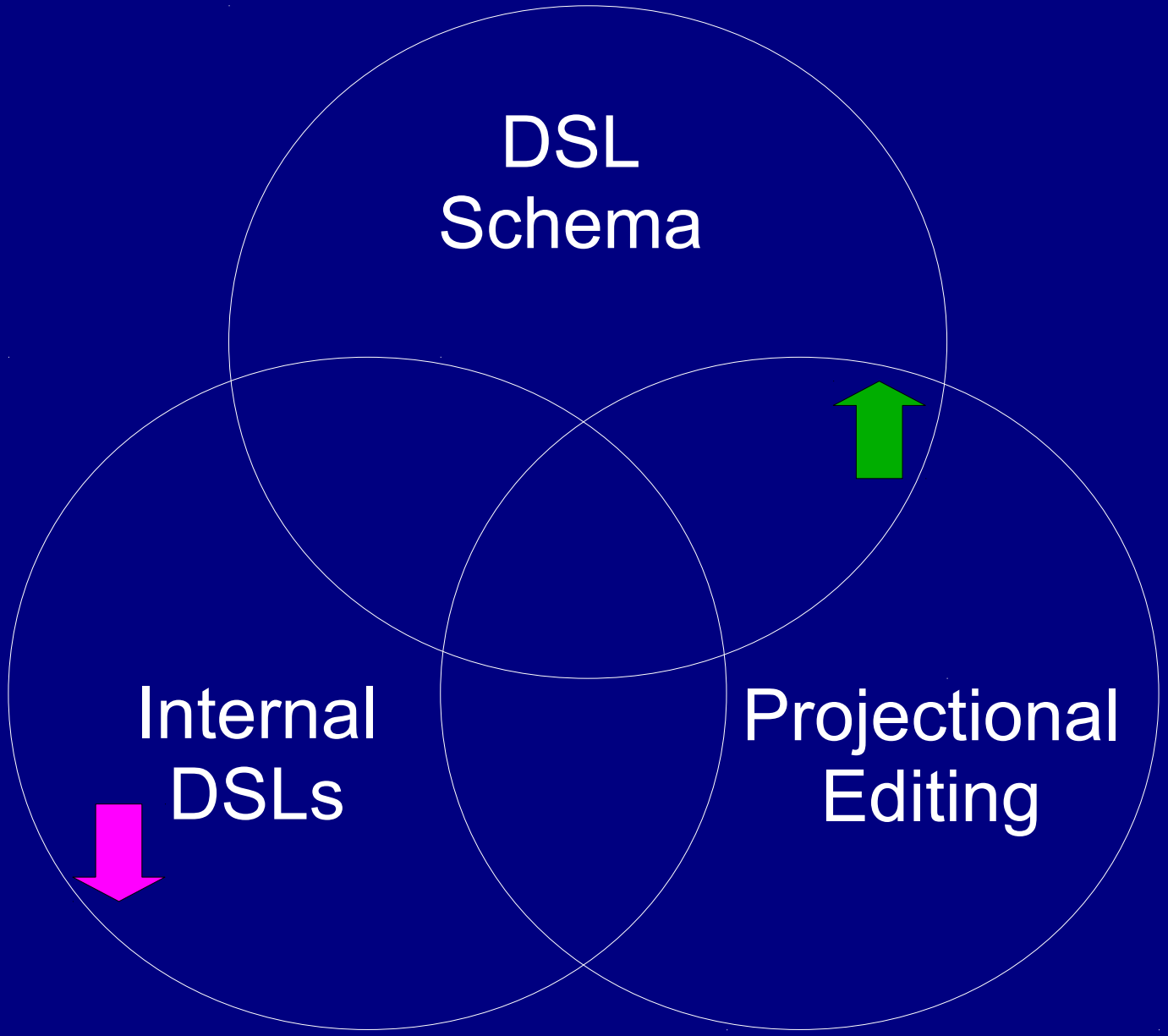
LOP Language Design Space



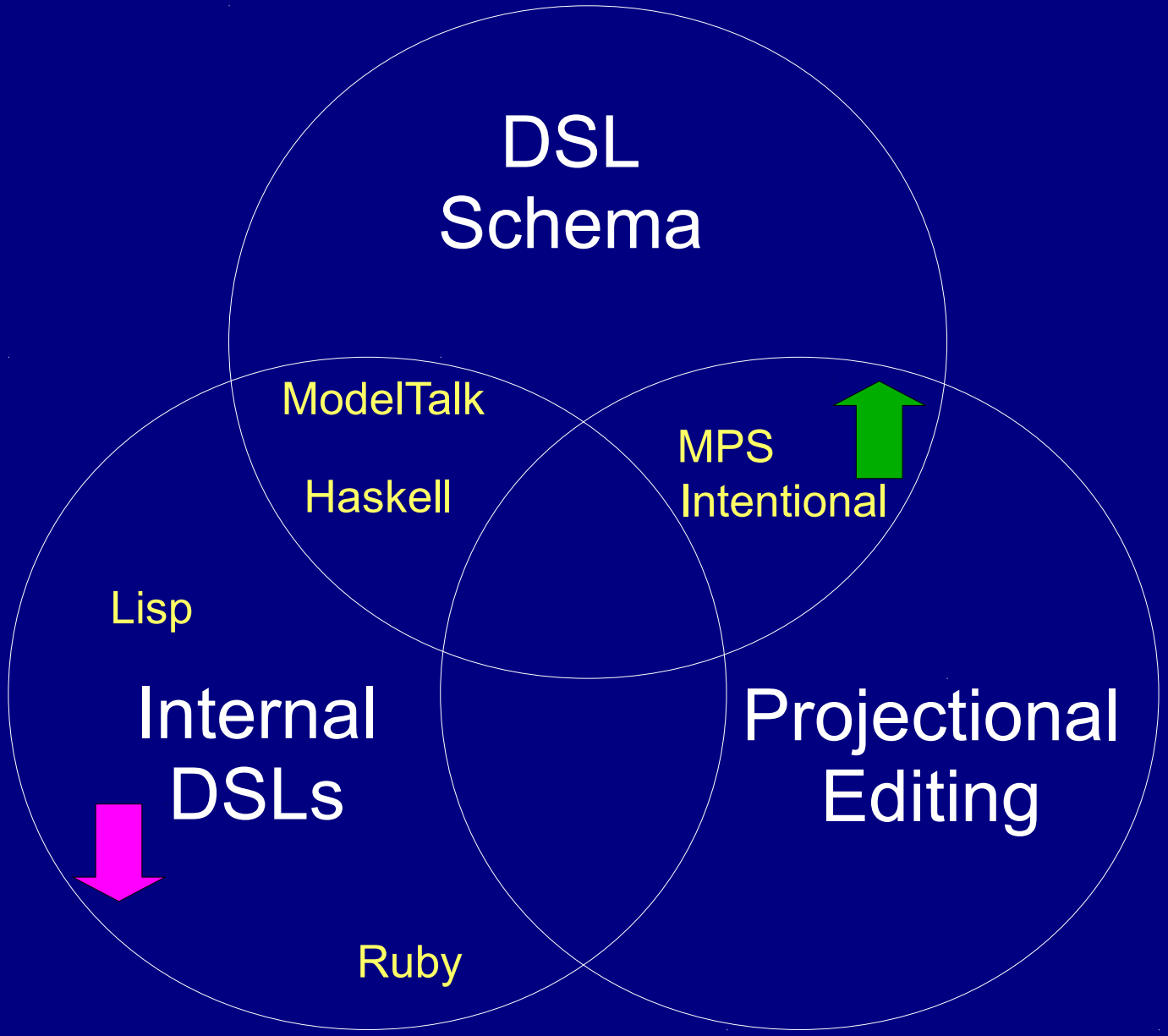
LOP Language Design Space



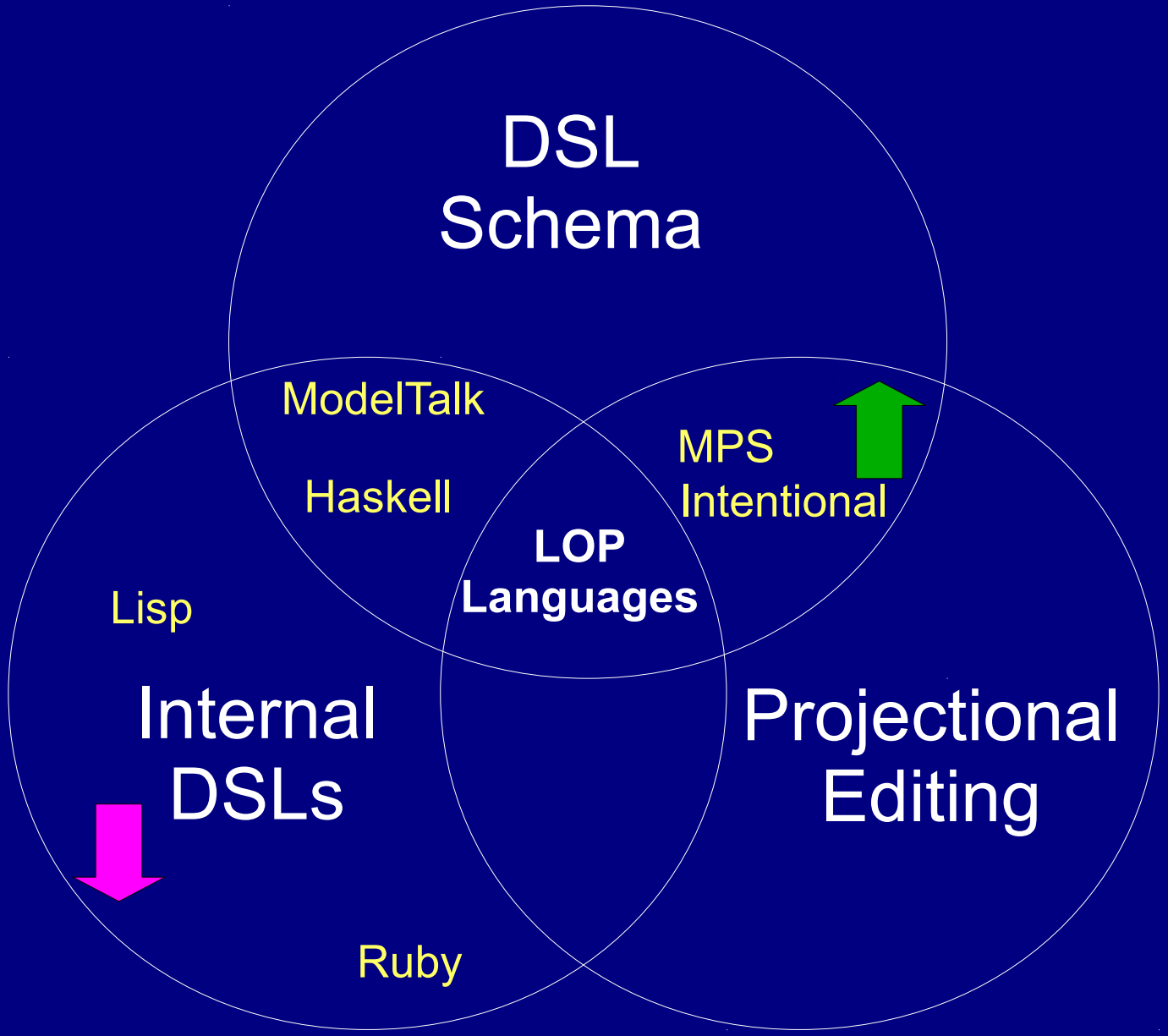
LOP Language Design Space



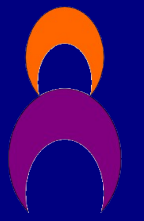
LOP Language Design Space



LOP Language Design Space



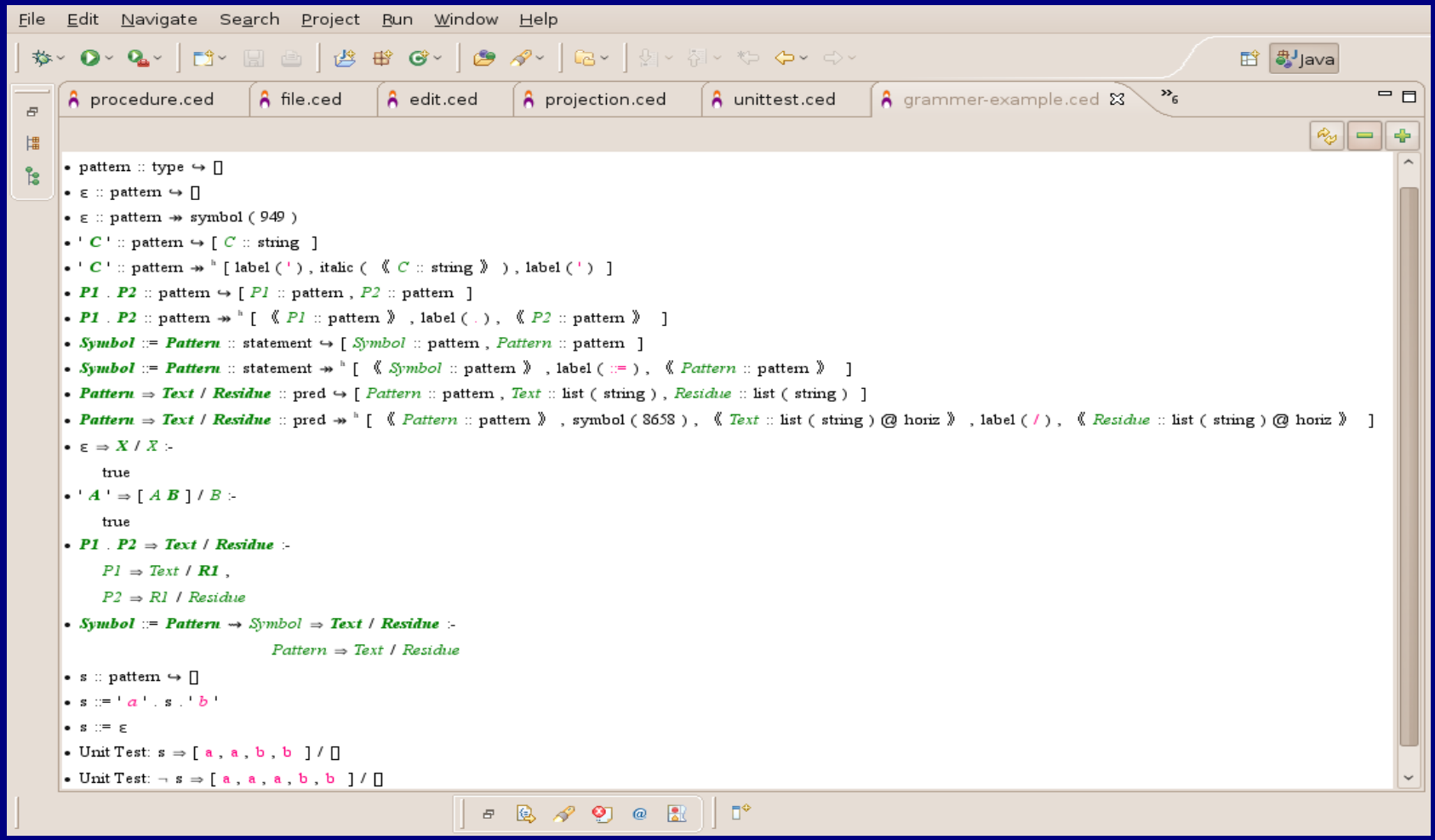
Cedalion: An LOP Language



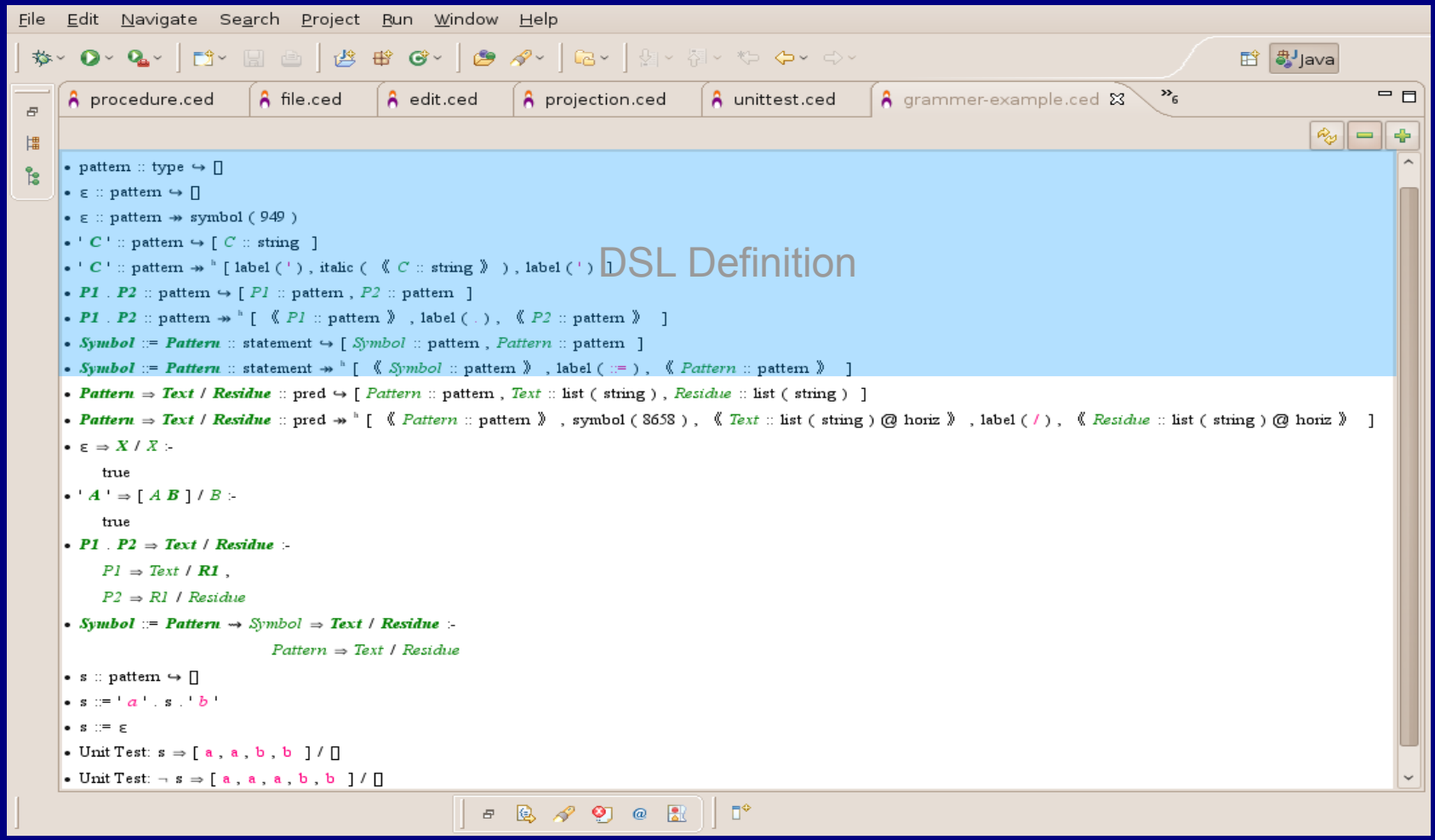
- **Logic Programming Language**
  - Hosts internal DSLs
- **Uses Projectional Editing**
  - As a way to provide syntactic freedom
- **Statically Typed (Type Inference)**
  - As a way to define schema
- **Open-source:**
  - http://cedalion.sf.net

Cedalion standing on the shoulders of Orion;
Nicolas Poussin, 1658

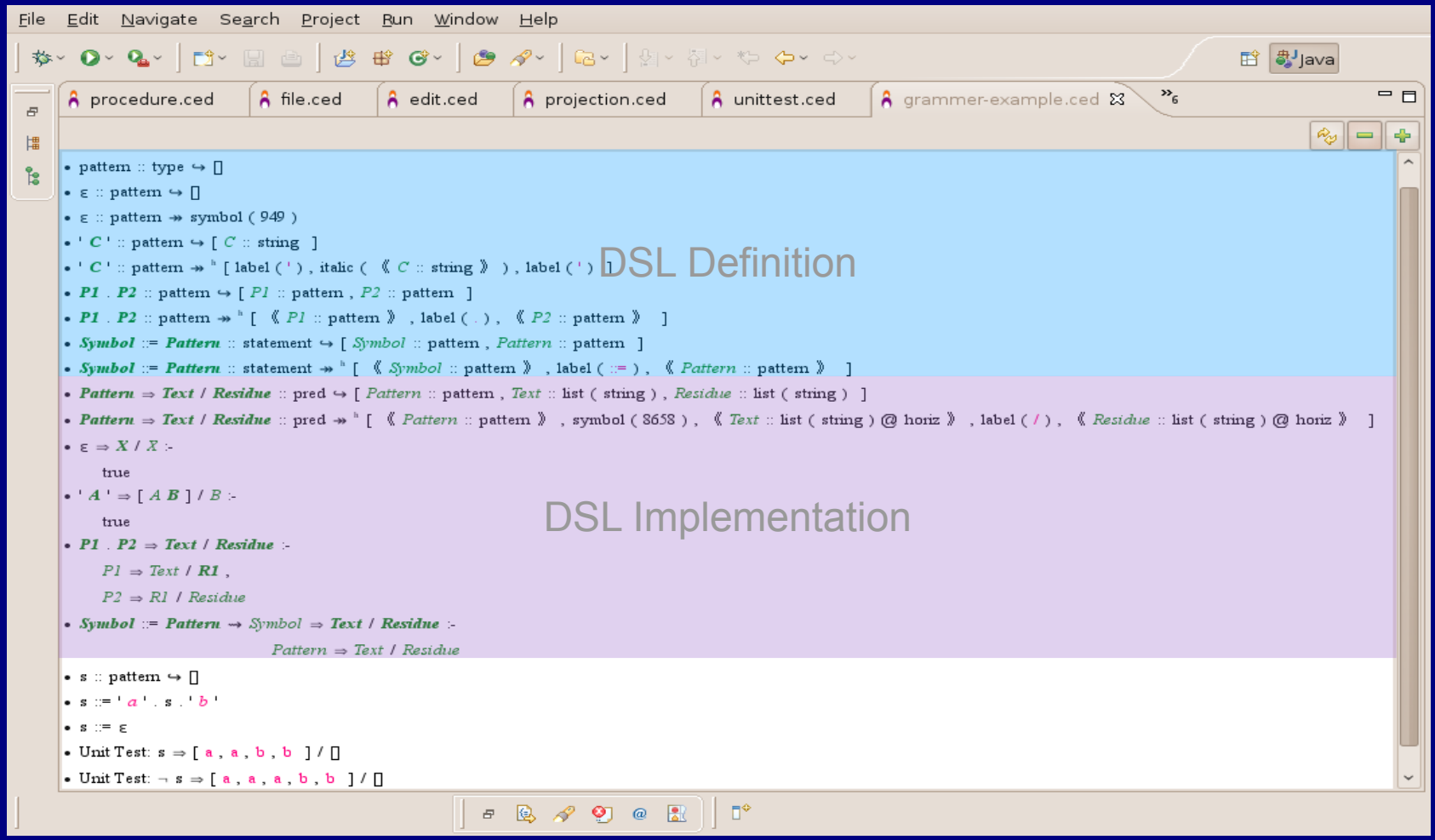
Cedalion in Action



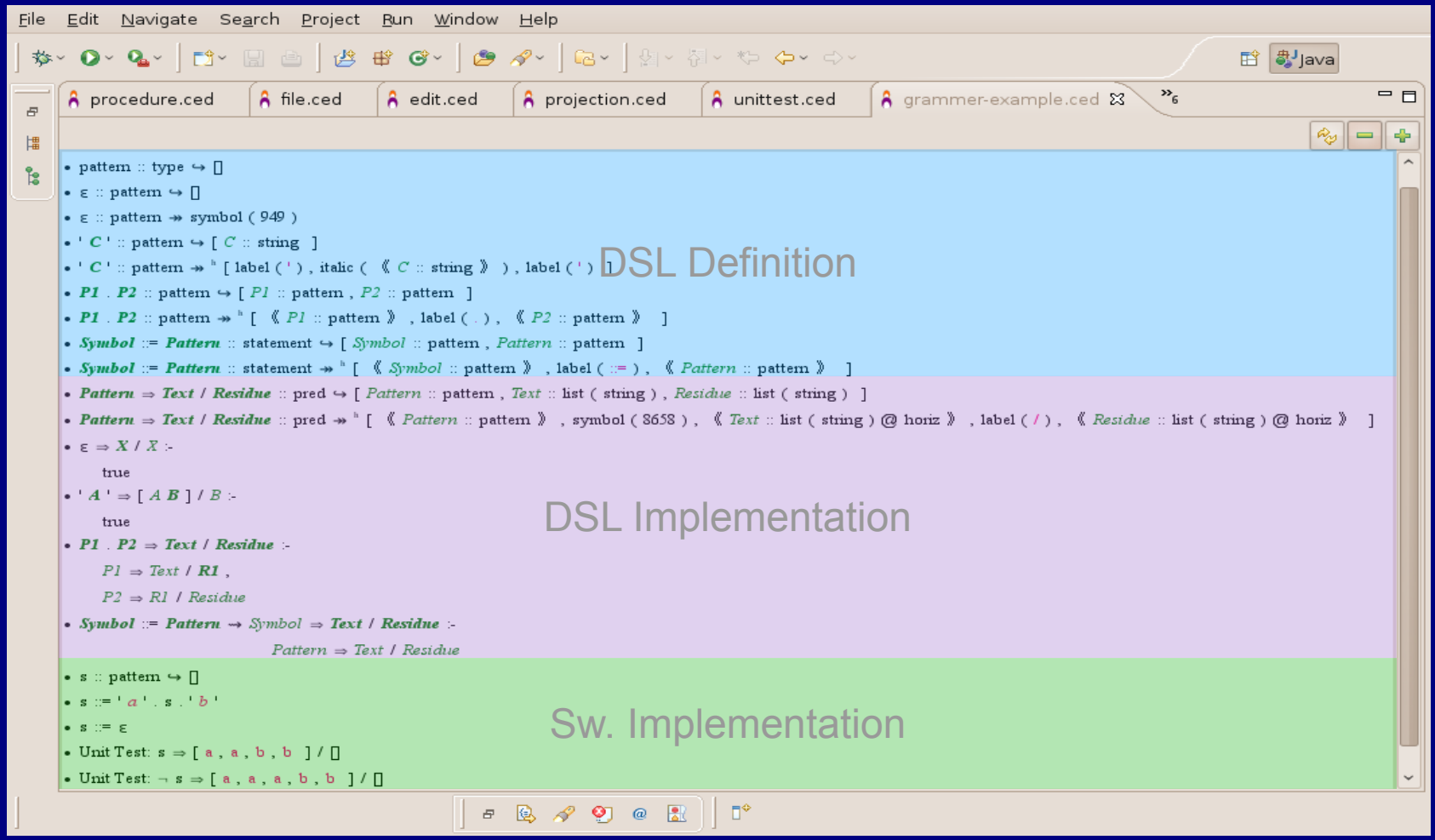
Cedalion in Action



Cedalion in Action



Cedalion in Action



Cedalion Case Studies

- BNF Grammar for Parsing + Evaluation.
- Functional Programming.
- Process Calculus (CCS) + Modal Logic (HML).
- DNA Sequence Sets

::= λ

P<a>[b]

Related Work

- **Language Oriented Programming**
  - [Ward, 1994] Language-oriented programming. Software-Concepts and Tools, 15(4):147–161, 1994
  - [Fowler, 2005] Language workbenches: The killer-app for domain specific languages. 2005.
- **Language Workbenches**
  - [Dmitriev, 2004] Language oriented programming: The next programming paradigm. JetBrains onBoard, 1(2), 2004.
  - [Simonyi, Christerson, and Clifford, 2006] Intentional software. ACM SIGPLAN Notices, 41(10):451–464, 2006.
- **Internal DSLs**
  - [Hudak, 1996] Building domain-specific embedded languages. ACM Computing Surveys (CSUR), 28(4es), 1996.

Conclusion

- **Contributions**
  - LOP Languages
  - Cedalion
- **Future Work**
  - Theory
    - Further investigate the properties of LOP Languages.
    - Prove Cedalion type-system correctness.
  - Practice
    - Make Cedalion “ready for prime-time”.
    - Provide more validation by real life examples.

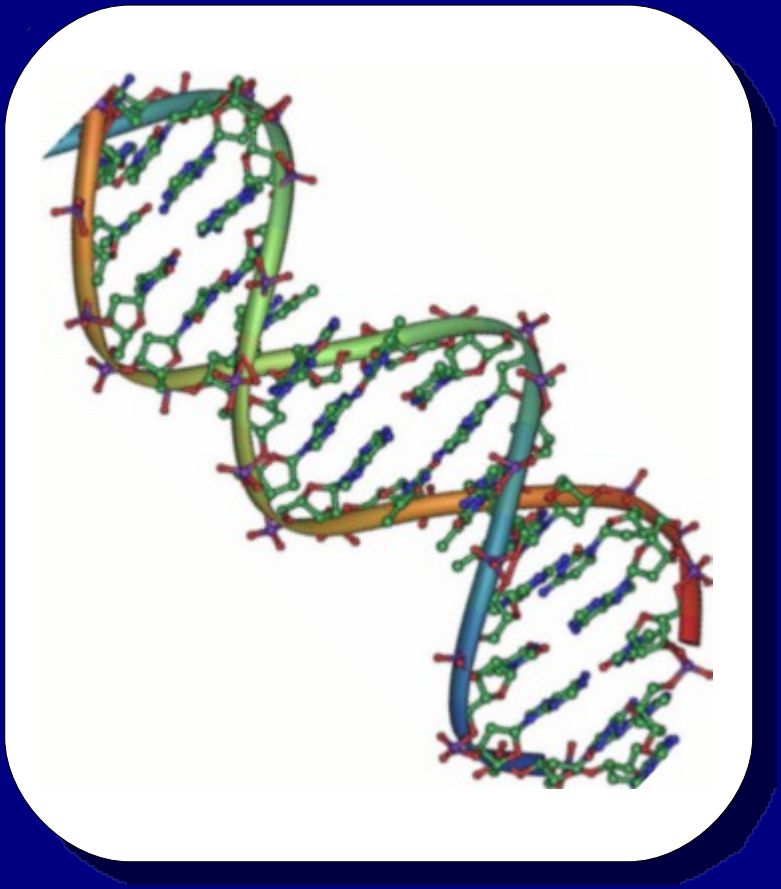
Case Study

DNA Sequence Sets
for
DNA Microarray Design

Joint work with Itai Beno,
Faculty of Biology,
Technion – Israel Institute of Technology

What is DNA?

- Deoxyribonucleic acid.
- A double-helix consisting of nucleotides.
- Four types, abbreviated A,T,C,G.
- Stores the “machine code” of life.
- ~3GBase (~750MB) is the size of the worlds most amazing “software”...



DNA and Cancer Research

- DNA anomalies play a significant role in the formation of Cancer.
- Studying these anomalies is critical in the search for effective treatment for Cancer.
- Certain proteins which participate in cancerous processes interact with DNA.
- These interactions are of extreme importance to this field.

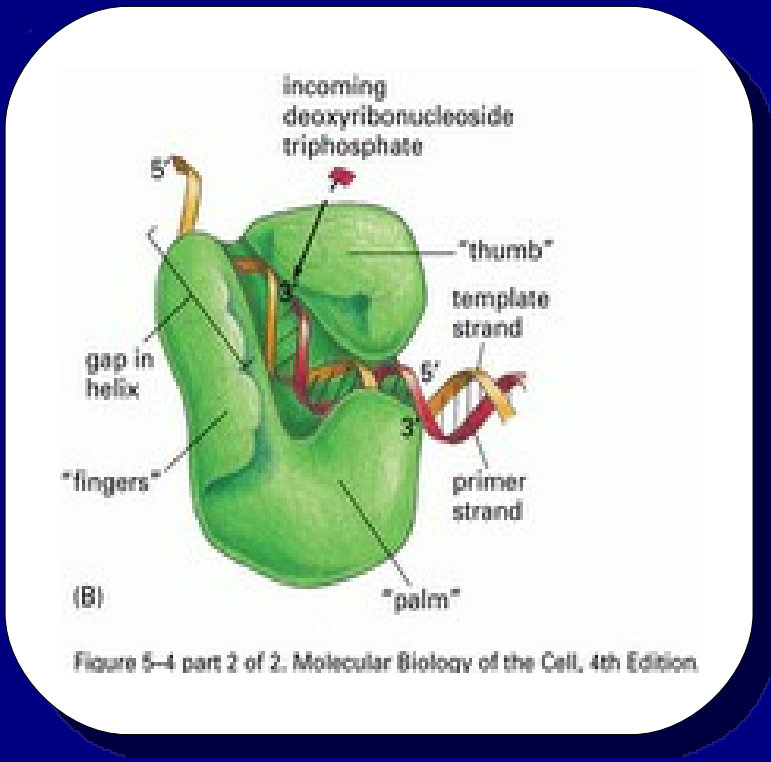
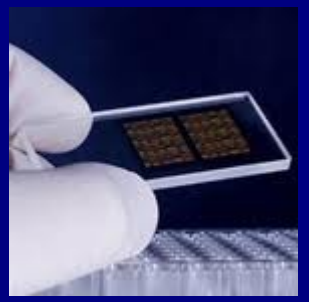
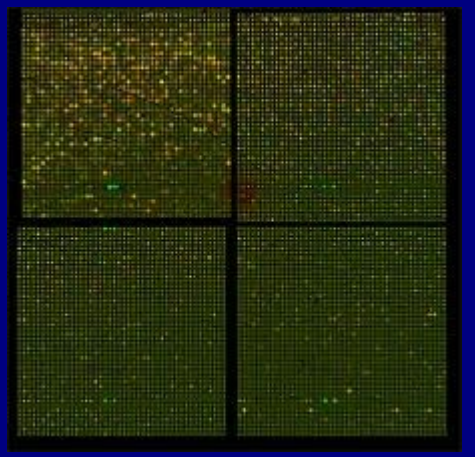


Figure 5–4 part 2 of 2. Molecular Biology of the Cell, 4th Edition.

DNA Microarray

- Finding a sequence with certain qualities requires multiple experiments.
- A DNA Microarray is a device containing $O(10^5)$ microscopic spots, each containing a different DNA sequence (multiple instances).
- Microarrays can be custom-made for specific experiments.
- Biologists provide the manufacturer a list of all sequences need to be produced.





Case-Study Goals

- **Produce a list of O(10^5) DNA sequences that reflect the desired design.**
- **Do this "LOP Style":**
  - Microarray specification is done by biologists (non-programmers).
  - These biologists should use a DSL developed for this purpose.
  - All "programming" should be restricted to the DSL and its runtime environment, and should be agnostic of the actual Microarray design.

Before Cedalion...

- The biologist performing this experiment has a programming day-job...
- Programmed ~500 LOC in Java to express a simple design.
- ✔ Runs fast (few seconds).
- ✗ This code must change to accommodate any change to the microarray design.

With Cedalion...

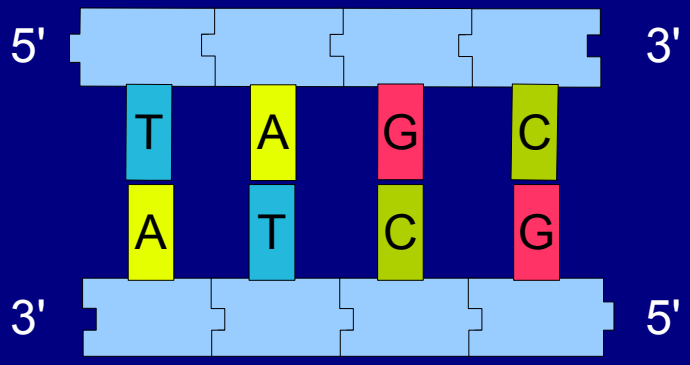
- A DSL was provided to express sets of DNA sequences.
- A microarray design can be defined using sets of sequences, with a name and quantity for each.
- A microarray design can be generated into files containing all sequences in the set.
- A 30 LOC Perl-script decimates the sequence files to form the desired output.

DSL for DNA Sequence Sets

- A/T/C/G: Singleton sets of a single nucleotide.
- N:=A∪T∪C∪G
- X.Y: The set consisting of an element of X concatenated to an element of Y.
- X^n: A singleton set containing the empty sequence if n=0, or $X.X^{n-1}$ otherwise.
- Y=[X]: Evaluates to the members of X. Y is bound to a singleton set containing that member, e.g., $Y=[N^2].Y$

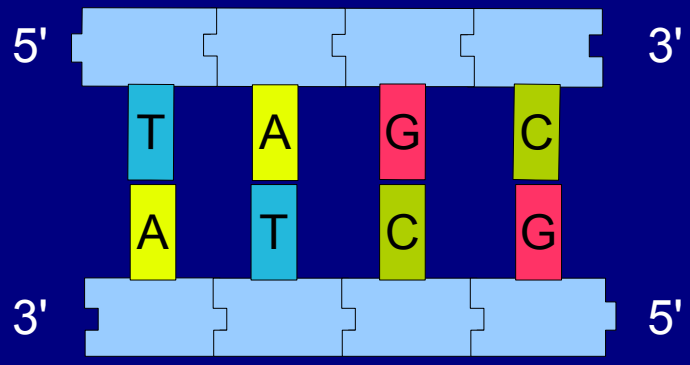
DSL for DNA Sequence Sets

- X^{inv} : The members of X in inverse order.
- X^{conj} : The members of X, with all nucleotides replaced by their conjugates: A⇔T; C⇔G.
- $X^{comp} := (X^{conj})^{inv}$



Restricting a Set

- Double-stranded DNA is redundant.
- For each sequence **S**, **S** and $\mathbf{S}^{comp}$ represent the same double-stranded DNA.
- restrict(**X**): Contains all members of **X**, taking only the "smaller" of two sequences representing the same DNA.
- uniformRestrict(**X**): Same as restrict(**X**), but taking either the smaller of the greater, at coin-toss.



Generating a Microarray

- A microarray has a **name** (base file name) and a list of **sections**.
- Each section consists of a **name**, a **set of sequences** and a **quantity** – how many sequences we wish to select.
- A context-menu-entry allows the generation of the microarray files, containing all possibilities.
- Running the Perl script in the target directory creates the final, decimated files.

Exercise

- **Build a microarray design.**
- **All sequences will start with ACCGGT and end with TTTTT.**
- **The middle part consists of a sequence followed by its conjugate.**
- **The basic sequence consists of the following:**
  - Experiment: A sequence of 5 bases, with either A or T in the middle. Select 100.
  - Control: A sequence of 5 bases, with either C or G in the middle. Select 20.

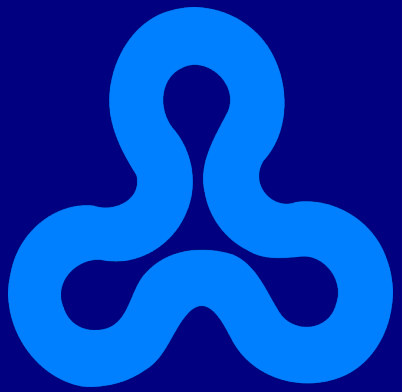
Case-Study Results

- With some assistance, the biologist was able to specify the microarray design using Cedalion.
- The design was changed twice before reaching the final version. Non of the changes required “programming”.
- Unit-tests were used to assure that the constant parts of the DNA sequences do not contain “interesting” features.

Case-Study: Conclusion

- **The microarray design produced by Cedalion was submitted to the manufacturer.**
- **Pros:**
  - ✔ Specification was done by non-programmer.
  - ✔ Modifications to the design were straight-forward.
- **Cons:**
  - ✗ Runtime performance is bad: x10 to x100 slower then the hand-written Java implementation (6 minutes for ~500,000 sequences).

Thank You!



Boaz Rosenan
Dept. of Computer Science
The Open University of Israel

brosenan@cslab.openu.ac.il
http://cedalion.sf.net