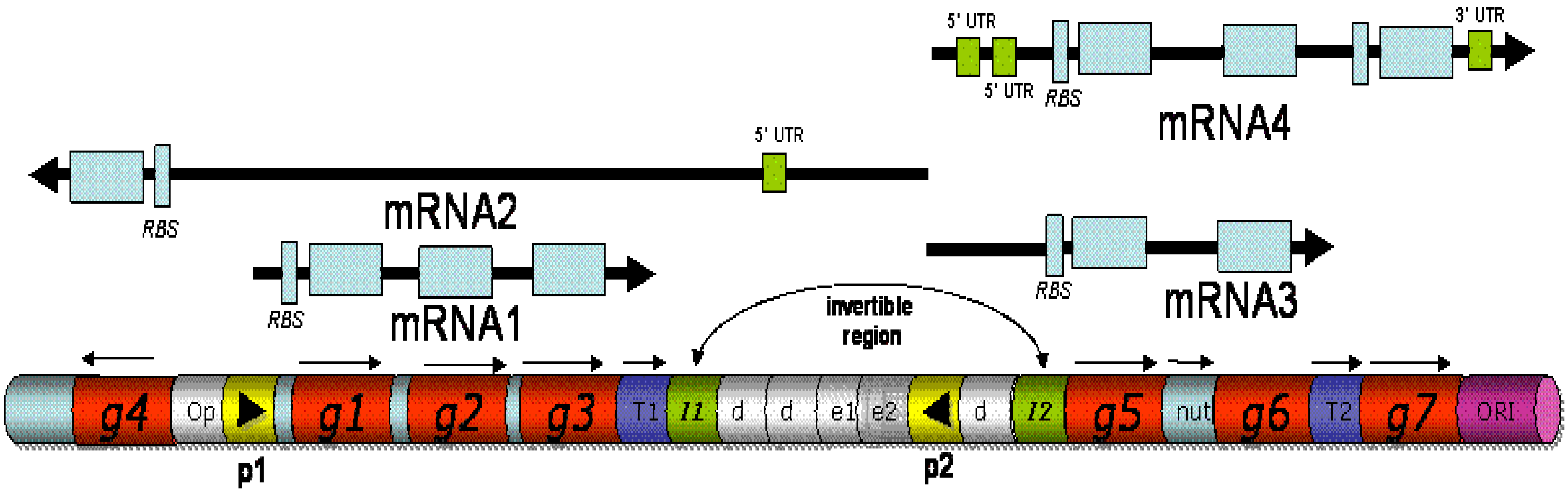
BioLogic:

“A compiler of high-level descriptions of biochemical systems, which targets chemical stochastic simulators.”

The Problem:

The difference between how biologists model complex systems, and the expected input of modern chemical stochastic simulators.



Requirements:

- Cytosolic molecules binding to DNA
- Molecular machinery processing along DNA (RNA polymerase)
- DNA segment inversion
- Modification of bound molecules
- Convergent transcription
- RNA interactions
- (partial) transcription, translation, and replication.

Design Goals:

- Effectively express the numerous underlying chemical reactions that are implicitly stated in biochemical models.
- Avoid need to re-design the target simulators.
- Capture as much complexity of the model as possible, without going into a volume-exclusion 3D-model.
- Reaction generation method should introduce no more amortized complexity than any other method.

Possible Approaches:

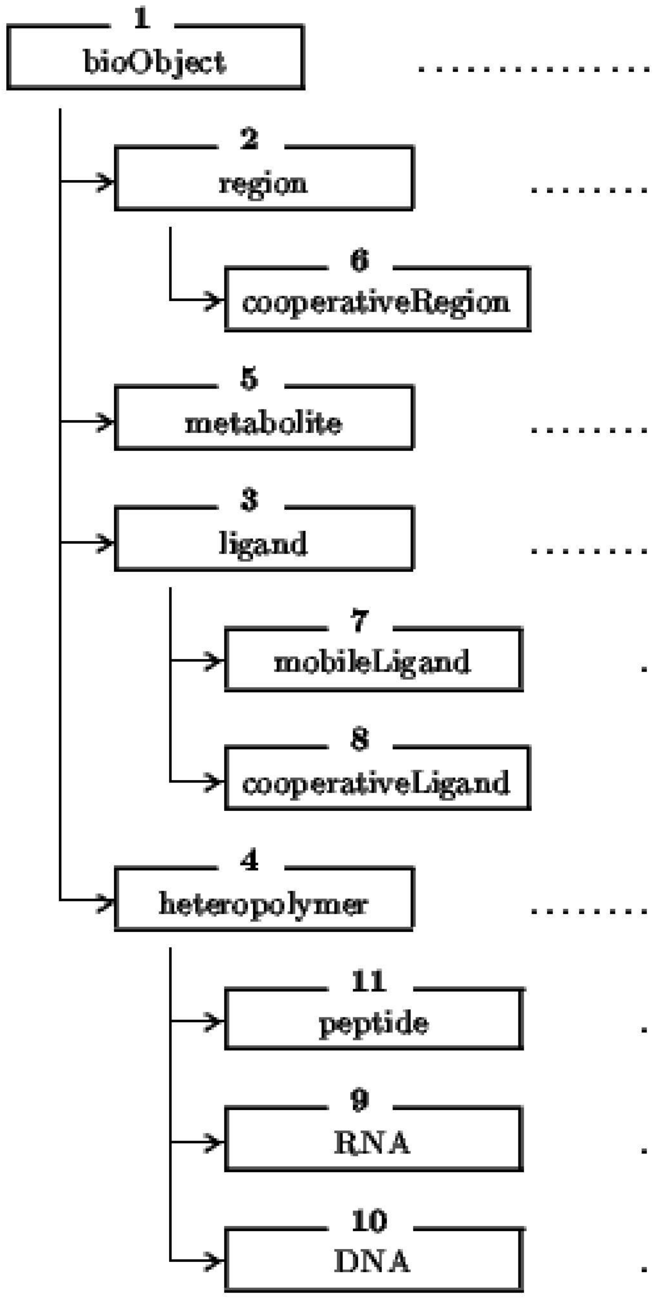
- Nondeterministic Finite State Machine (NDFSM)
- 'High-level description' to 'simulator input' compiler.

NDFSM Approach:

- **Pros:**
- No need to pre-compile the system model.
- Very straight-forward software design.
- Mitigates need for exhaustive reaction ennumeration.
- **Cons:**
- Will add a huge overhead to the simulator's main loop.
- Requires the re-design of existing simulators.

The Compiler Approach:

- **Pros:**
- “Pre-compiled”, so simulator-loop is spared.
- No necessary changes for simulators.
- Trivial alterations for simulators can allow significant speed-up.
- Allows for possible run-time optimizations.
- **Cons:**
- Ennumeration of all possible 'trivial' reactions.
- Requires sophisticated software design.



The Software Design

- C++ for fast, object-oriented design.
- Compiler classes are generalizations of biochemical objects distilled to their essential features.
- Still being developed.

The Approach:

- All such intereactions can be refined to casting the problem as determining all the overlapping intervals along a sequence.
- Coupled with symbol generation, this generates the reactions for the simulator.

Examples:

Ligand + BindingSite + !(OverlappingLigand*OverlappingBindingSite)

-->

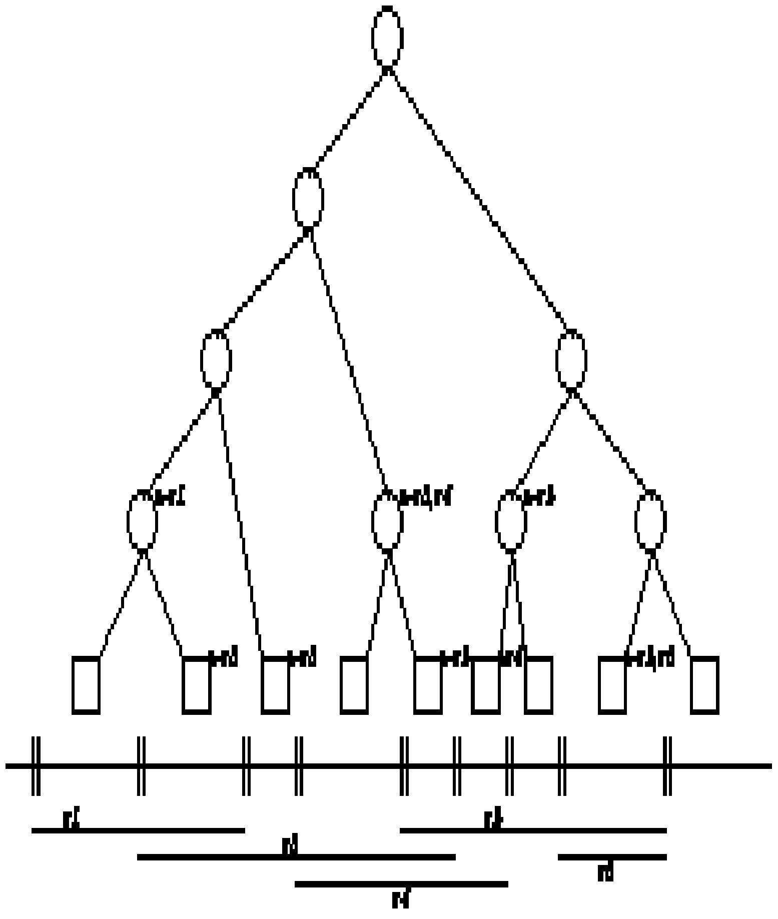
(Ligand*BindingSite)

$$RNAP_1 + site_0 \rightleftharpoons RNAP_1 site_0, \quad /*\text{first binding}*/ \tag{1a}$$

$$site_{i+1} + RNAP_1 bind_i \rightarrow RNAP_1 site_{i+1} + site_i,$$
$$i = 1 \ldots L-1 \quad /*\text{drift of RNAP1}*/ \tag{1b}$$

$$RNAP_1 site_{L-1} + site_L \rightarrow site_L /*\text{terminator}*/ \tag{1c}$$

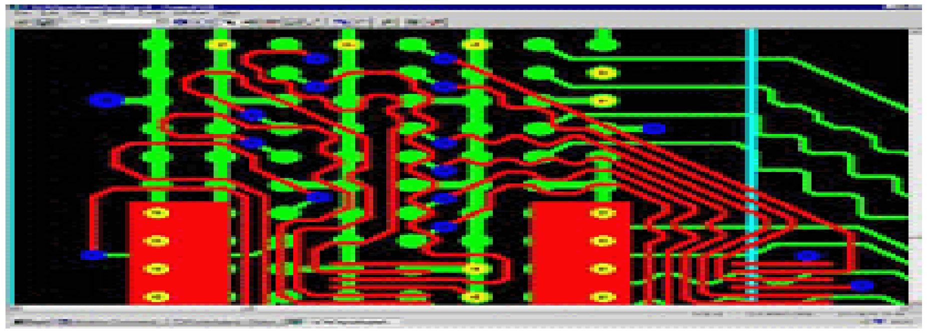
The Core Algorithm:

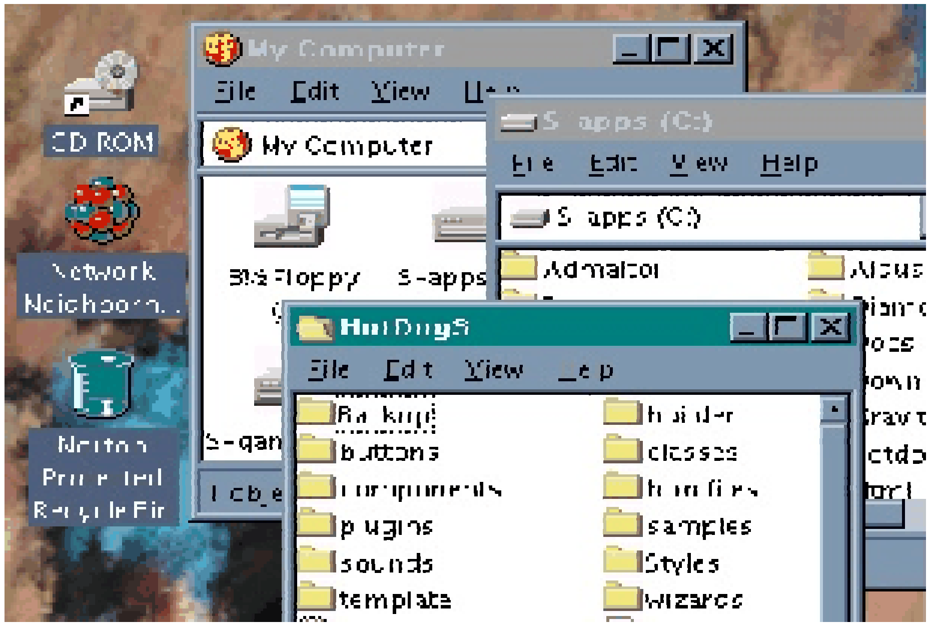


- Interval tree: augmented red-black search tree.
- C++ algorithm from computational geometry suite CGAL.
- Allows for all-vs-all overlap detection in O(n*logn).

Example of Geometric Overlap Detection:

- This is a critical algorithm for VLSI design & layout tools.
- Used in GUI / windowing software for fast response.





Acknowledgements:

- Adam Arkin
- Sergei Plyasunov
- Keith Keller, Eric Alm, Ken Koster, Alex Gilman, and all others who assisted my algorithmic design and my learning of C++.