

Introducing the kappa-calculus

"Formal molecular biology"

Danos and Laneve, Theoretical Computer Science 325 (2004) 69 – 110

The kappa calculus

- A visual language
 - with a term syntax
 - for idealized protein-protein interaction.

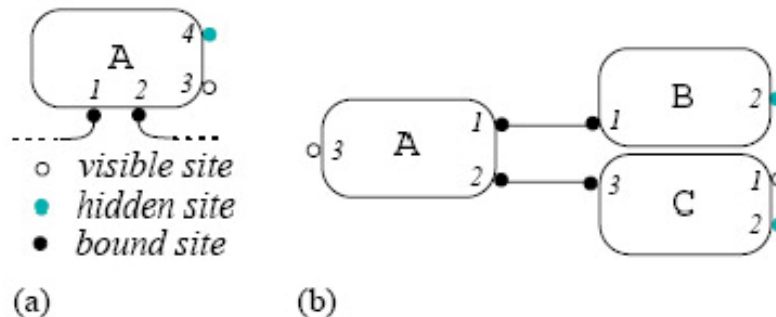


Fig. 1. A protein and a complex.

- Syntax, structural congruence set up to generate a superclass of terms isomorphic to graphs.
- *Graph-likeness* requirement (Def. 2) restricts to graphs with one-one edges.

Table 1
The syntax of κ -calculus

S :=	solution
0	empty solution
$A(\rho)$	protein
S, S	group
$(x)(S)$	new

Definition 1. Structural congruence, written \equiv , is the least equivalence closed under syntactic constructions, containing α -equivalence (injective renaming of bound variables), taking “,” to be associative (as the choice of symbol suggests) and commutative, with 0 as neutral element, and satisfying the scope laws:

$$\begin{aligned}
 (x)(y)(S) &\equiv (y)(x)(S), \\
 (x)(S) &\equiv S && \text{when } x \notin \text{fn}(S), \\
 (x)(S), S' &\equiv (x)(S, S') && \text{when } x \notin \text{fn}(S').
 \end{aligned}$$

Graphs, syntax, and syntactic shorthands

- Syntactic shorthands for sites, hidden sites, etc.

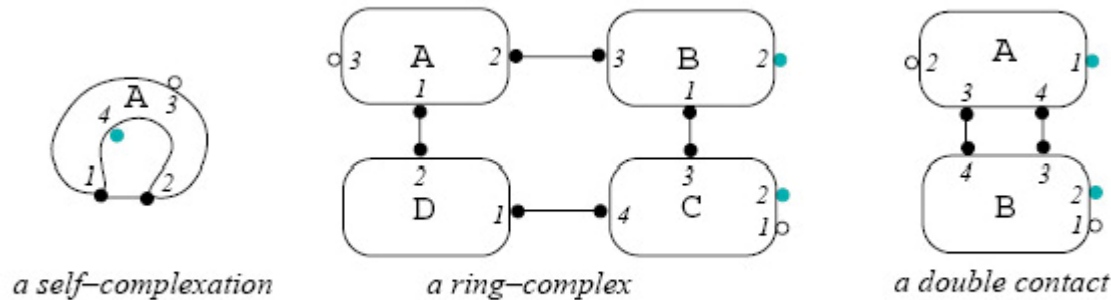


Fig. 2. Complexes.

$$\begin{aligned}
 &(x)(A(1^x + 2^x + 3 + \bar{4})), \\
 &(wxyz)(A(1^x + 2^y + 3), B(1^z + \bar{2} + 3^y), C(1 + \bar{2} + 3^z + 4^w), D(1^w + 2^x)), \\
 &(xy)(A(\bar{1} + 2 + 3^x + 4^y), B(1 + \bar{2} + 3^y + 4^x)).
 \end{aligned}$$

Syntax for simple graphs – and no more.

- The syntax is setup directly to model (closed) graphs; and no more.
- Relatedly, translation from terms to graphs do not distinguish between free or bound names. (3. below)

Definition 3. Let $\llbracket \cdot \rrbracket_g$ be the following function from graph-like solutions to graphs with sites:

1. $\llbracket A(\rho) \rrbracket_g$ is the graph with a single node labeled A , sites in $\{1, \dots, \mathfrak{s}(A)\}$, bound sites k being labeled by $\rho(k)$, and free sites being in the state prescribed by ρ ;
2. $\llbracket S, S' \rrbracket_g$ is the union graph of $\llbracket S \rrbracket_g$ and $\llbracket S' \rrbracket_g$, with sites labeled with the same name being connected by an edge, and their common name erased;
3. $\llbracket (x)(S) \rrbracket_g$ is $\llbracket S \rrbracket_g$.

Dynamics

- Reaction is defined over terms:
 - Define matching for pairs of solutions to reaction rules (Def. 5).
 - ... enforces that names in rules are just placeholders as in bigraphs.
 - ... allows interfaces (site/port-sets) to be partially identified in rules.
 - Finally matching is contextualized (parallel, new, struct) (Def. 6).
- Very careful to restrict rules to (what they take to be) an *atomic biological step*:
Reaction *either* stems from a connected component (anti-monotonic) *or* results in one (monotonic) (Def 4.).
- E.g., protein-*synthesis* (creation) or *-degradation* (deletion) only allowed on closed (i.e., disjoint) proteins.
- Props and lemmas state that graph-likeness is preserved under reactions (Lem. 2 + Prop. 3)

microKappa and the pi-calculus

- Paper also contains an investigation of *self-assembly*; loosely, whether one can restrict to (at most) binary interactions and still obtain same behavior.
- Answer by example - by encoding/implementing kappa in microKappa.
- In microKappa at most two agents may interact; also the interfaces are extended with logs (for bookkeeping).
- This encoding is also taken as the first step in an encoding of kappa into good old pi.

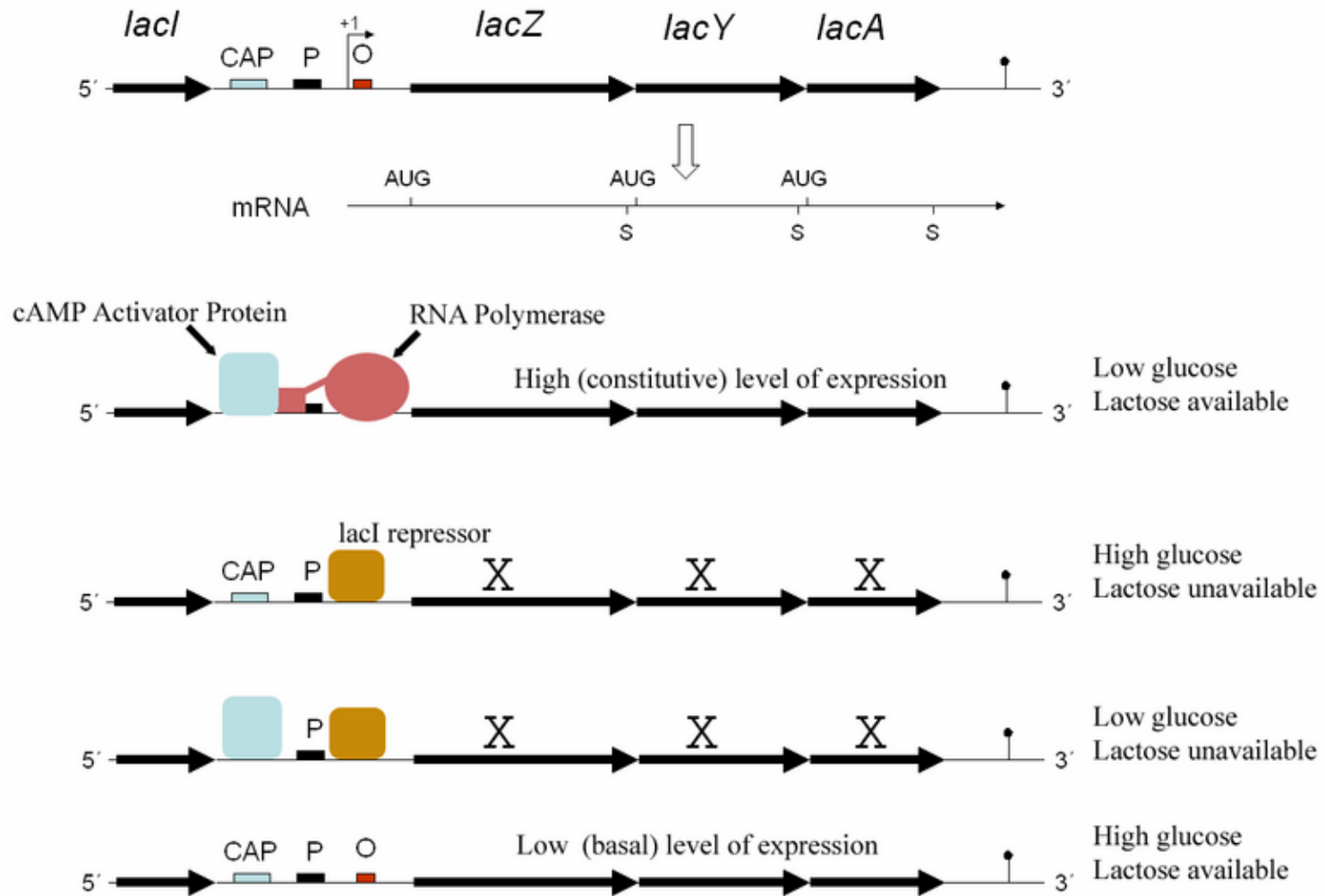
Discussion

- This is (yet) another graphical calculus with a term syntax – what do we think of the setup of the calculus?
- What is the motivation for interest in self-assembly?
- Walk-through and discussion of the example – the lactose-operon.
 - How well do you find the example is captured? Compared to other formalisms (hypotethize)?
 - What is abstracted away?
- To what extent is it shown that the calculus is useful for modelling molecular processes?
- (*Special*) [Schedule](#)

Lac Operon: The movie

See movie online at : <http://vcell.ndsu.nodak.edu/animations/lacOperon/movie.htm>

The *lac* Operon and its Control Elements



(Image from the [Wikimedia Commons](https://commons.wikimedia.org/wiki/File:Lac_operon_diagram.png))

Operon Synthesis

$\tau_{0a} : \text{UP}(\overline{\text{rep-s}} + \text{map-s}^x + \text{cap-s}^y), \text{RNAP}(\text{up-s}^x + \text{syn1} + \text{syn2}) \rightarrow$
 $(zu)(\text{UP}(\overline{\text{rep-s}} + \text{map-s}^x + \text{cap-s}^y), \text{RNAP}(\text{up-s}^x + \text{syn1}^z + \text{syn2}^u),$
 $\text{GAL}(\text{lac-s} + s^z), \text{PER}(\text{lac-s} + s^u))$

$\tau_{0b} : (yz)(\text{RNAP}(\text{syn1}^y + \text{syn2}^z), \text{GAL}(\text{lac-s} + s^y), \text{PER}(\text{lac-s} + s^z)) \rightarrow$
 $\text{RNAP}(\text{syn1} + \text{syn2}), \text{GAL}(\text{lac-s} + s), \text{PER}(\text{lac-s} + s)$

Operon Control

$\tau_1 : \text{UP}(\text{map-s} + \text{rep-s}), \text{REP}(\text{up-s}) \rightarrow (x)(\text{UP}(\overline{\text{rnap-s}} + \text{rep-s}^x), \text{REP}(\text{up-s}^x))$

$\tau_2 : \text{UP}(\text{map-s} + \text{rep-s}), \text{RNAP}(\text{up-s}) \rightarrow (x)(\text{UP}(\overline{\text{rep-s}} + \text{rnap-s}^x), \text{RNAP}(\text{up-s}^x))$

$\tau_3 : \text{UP}(\text{cap-s}), \text{CAP}(\text{up-s} + \text{camp-s}^x), \text{cAMP}(\text{cap-s}^x) \rightarrow$
 $(z)(\text{UP}(\text{cap-s}^z), \text{CAP}(\text{up-s}^z + \text{camp-s}^x), \text{cAMP}(\text{cap-s}^x)).$

Regulations

$\tau_4 : \text{CAP}(\text{camp-s}), \text{cAMP}(\text{cap-s}) \rightarrow (x)(\text{CAP}(\text{camp-s}^x), \text{cAMP}(\text{cap-s}^x))$

$\tau_{5a} : \text{REP}(\text{up-s} + \text{alac-s}), \text{aLac}(\text{rep-s}) \rightarrow (x)(\text{REP}(\overline{\text{up-s}} + \text{alac-s}^x), \text{aLac}(\text{rep-s}^x))$

$\tau_{5b} : \text{UP}(\text{rep-s}^x), \text{REP}(\text{up-s}^x + \text{alac-s}), \text{aLac}(\text{rep-s}) \rightarrow$
 $(y)(\text{UP}(\text{rep-s}^x), \text{REP}(\text{up-s}^x + \text{alac-s}^y), \text{aLac}(\text{rep-s}^y))$

$\tau_{5c} : (x)(\text{UP}(\text{rep-s}^x), \text{REP}(\text{up-s}^x + \text{alac-s}^y), \text{aLac}(\text{rep-s}^y)) \rightarrow$
 $\text{UP}(\text{rep-s}), \text{REP}(\overline{\text{up-s}} + \text{alac-s}^y), \text{aLac}(\text{rep-s}^y).$

PER and GAL activity

$\tau_6 : \text{PER}(\text{lac-s}), \text{Lac}(\text{per-s} + \overline{\text{in}}) \rightarrow (x)(\text{PER}(\text{lac-s}^x), \text{Lac}(\text{per-s}^x + \text{in}))$

$\tau_7 : (x)(\text{PER}(\text{lac-s}^x), \text{Lac}(\text{per-s}^x)) \rightarrow \text{PER}(\text{lac-s}), \text{Lac}(\text{per-s})$

$\tau_8 : \text{GAL}(\text{lac-s}), \text{Lac}(\text{in} + \text{gal-s}) \rightarrow (x)(\text{GAL}(\text{lac-s}^x), \text{Lac}(\text{in} + \text{gal-s}^x))$

$\tau_{9a} : (x)(\text{GAL}(\text{lac-s}^x + \overline{\text{loaded}}), \text{Lac}(\text{gal-s}^x)) \rightarrow \text{GAL}(\text{lac-s} + \text{loaded})$

$\tau_{9b} : \text{GAL}(\text{loaded}) \rightarrow \text{GAL}(\overline{\text{loaded}}), \text{Glu}(s), \text{Gal}(\text{rep-s})$

$\tau_{9c} : \text{GAL}(\text{loaded}) \rightarrow \text{GAL}(\overline{\text{loaded}}), \text{aLac}(\text{rep-s}).$