Compilation and verification of nucleic acid reaction networks

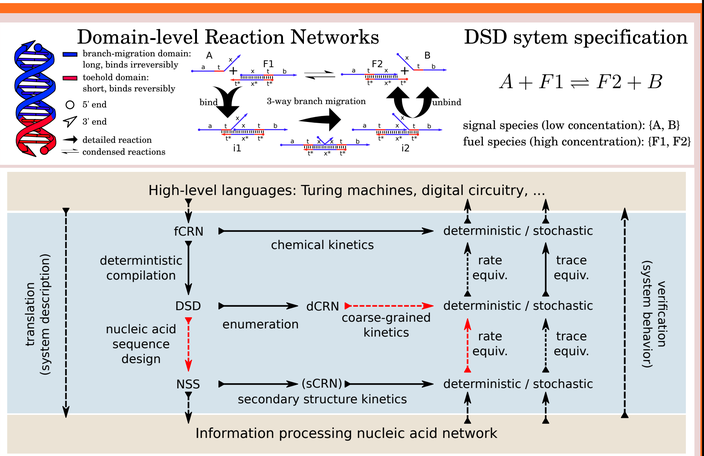
Stefan Badelt[1], Seung Woo Shin[1], Robert Johnson[1], Qing Dong[2], Chris Thachuk[1] and Erik Winfree[1]

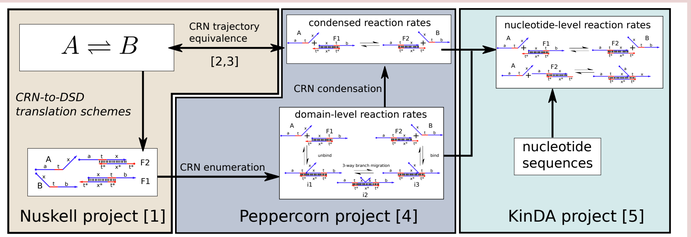
1: California Institute of Technology, Pasadena, USA, 2: Stony Brook University, New York, USA

http://www.dna.caltech.edu/~badelt
badelt@caltech.edu

The big picture:

Toehold-mediated strand displacement serves as a molecular architecture to implement algorithmic behavior with DNA interaction networks. Input DNA triggers a series of conformational changes in present DNA complexes, eventually releasing previously caged output DNA. Our compiler **Nuskell** [1] automates the translation of formal chemical reaction networks (**CRN**s) into domain-level strand displacement (**DSD**) circuits. A notion of correctness is established on a case-by-case basis using the rate-independent, stochastic-level theories of pathway decomposition equivalence [2] and/or CRN bisimulation [3]. Although no notion of the correctness of kinetic behavior is supported yet, the compiler automatically generates simulation code based on a sequence-independent model of DNA strand displacement biophysics [4]. Future versions will include sequence-level design and sequence-level analysis of DSD systems [5].



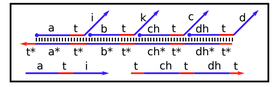


CRN-to-DSD translation schemes:

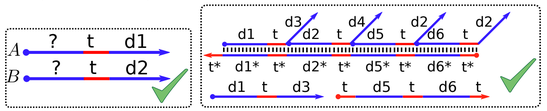
A translation scheme is an algorithm to translate a formal CRN into a set of *signal* and *fuel* species. Signal species are at low concentrations and they present the information (input/output) unit. Fuel species are at high (ideally constant) concentrations and they mediate the information transfer by consuming and/or releasing signal species. After compilation, every species in the formal CRN has a corresponding signal species. All signal species must have the same domain-level constitution and structure, and they need to be independent from each other. Translation schemes may be particularly efficient for certain types of formal reactions but inefficient or incorrect for other types, or they can be correct for every possible formal CRN at the cost of being less efficient.

```
# Translate formal reactions with two reactants and two products.
# Lakin et. al (2012) "Abstractions for DNA circuit design." [Figure 5]

# Define a global short toehold domain
global toehold = short();

# Write a class to define domains and structure of signal species
class formal(s) = "? t f" | ". . ."
  where { t = toehold ; f = long() };

# Write a class to produce fuel complexes for bimolecular reactions
class bimol_fuels(r, p) =
  [ "a t i + b t j + k t c + l t d + t* l* t* k* t* b* t* a* t*"
  | "( . + ( . + ( . + ( . + (  ( . + ) ) ) ) ) ) . ",
    "t k t l t" | ". . . . .",
    "a t i" | " . . . ", "b t j" | " . . . " ] # additional fuels
  where {
    a = r[0].f;
    b = r[1].f;
    c = p[0].f; k = long();
    d = p[1].f; l = long();
    i = long(); j = long();
    t = toehold };

# Write a module that applies the fuel production to every reaction
module rxn(r) = sum(map(infty, fuels))
  where fuels =
    if len(r.reactants) != 2 or len(r.products) != 2 then
      abort('Reaction type not implemented')
    else
      bimol_fuels(r.reactants, r.products);

# Write the module *main* that applies *rxn* to the crn.
module main(crn) = sum(map(rxn, crn))
  where crn = irrev_reactions(crn);
```

formal(s) = ? t f

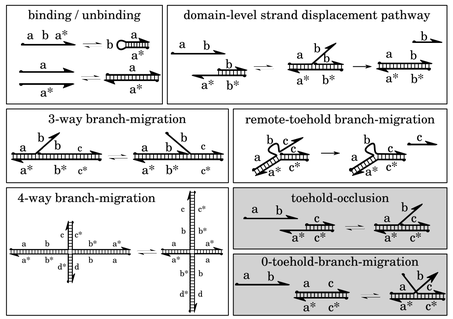


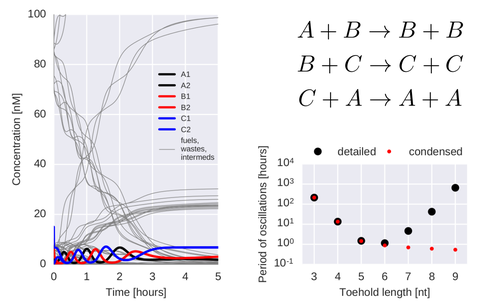
$A + B \rightarrow B + B$



reaction enumeration:

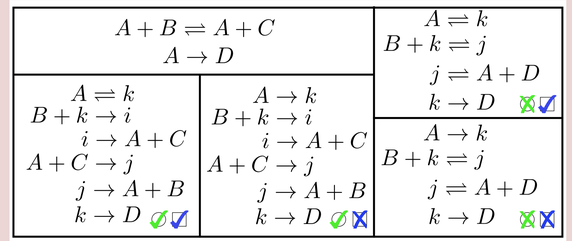
On the domain-level, we have to consider a diverse set of reactions in order to compensate for the fine-grained details that can happen on the sequence level. Nuskell uses a domain-level reaction enumerator [4] to predict desired and undesired reactions that can emerge from previously compiled signal and fuel species. Enumeration semantics are justified based on the assumption that the DSD system is operated at sufficiently low concentrations, such that unimolecular reactions always go to completion before the next bimolecular interaction takes place. In order to reduce the size of the enumerated reaction network, transient states (e.g. only the toehold bound) cannot engage in further downstream reactions.

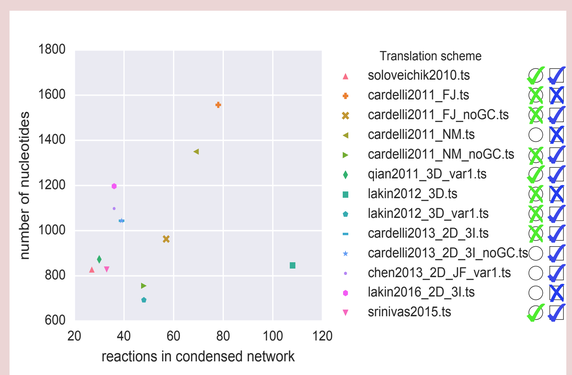


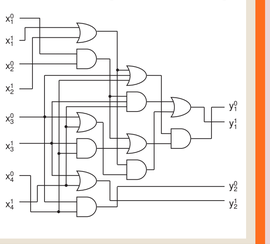


prove/disprove CRN equivalence:

The most fundamental requirement towards compilation of large DSD systems is verification. Every translation scheme translates single formal reactions into a multiple implementation reactions. Thus, there are many possibilities to introduce “bugs”, i.e. unwanted side reactions that alter the implemented algorithm. We include two case-by-case verification strategies that compare formal CRNs with their implementations.
As intended, our approach does not verify the general correctness of a particular scheme, but supports the notion that particular implementations have to be correct. Two notions are currently implemented in the Nuskell compiler: pathway decomposition [2] and CRN bisimulation [3].



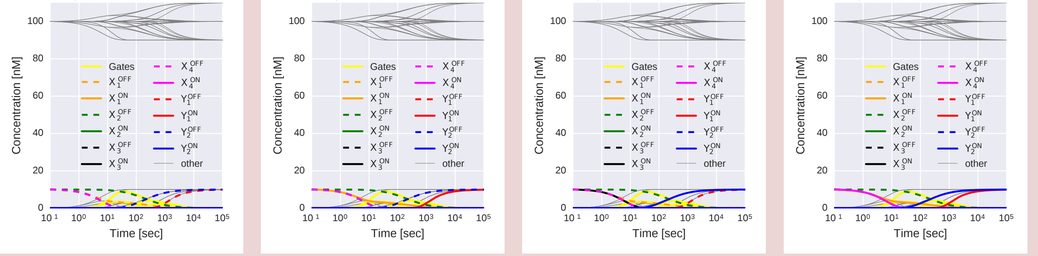





```
# A digital AND/OR/NOT circuit formulated as CRN.
# The CRN computes the floor of the squareroot of a four-bit binary number.

# Fanout X3 -> F5 + F6
X3_OFF -> F5_OFF + F6_OFF + F7_OFF
X3_ON  -> F5_ON  + F6_ON  + F7_ON
# Fanout X4 -> F7 + F8
X4_OFF -> F8_OFF + F9_OFF + F10_OFF
X4_ON  -> F8_ON  + F9_ON  + F10_ON
# G11 = NOT(X1 OR X2) + Fanout G11 -> F14 + F15
X1_OFF + X2_OFF -> F14_ON  + F15_ON
X1_OFF + X2_ON  -> F14_OFF + F15_OFF
X1_ON  + X2_OFF -> F14_OFF + F15_OFF
X1_ON  + X2_ON  -> F14_OFF + F15_OFF
# G12 = F6 AND (NOT F9)
F6_OFF + F9_OFF -> G12_OFF
F6_OFF + F9_ON  -> G12_OFF
F6_ON  + F9_OFF -> G12_ON
F6_ON  + F9_ON  -> G12_OFF
# Y2 = F7 OR F10
F7_OFF + F10_OFF -> Y2_OFF
F7_OFF + F10_ON  -> Y2_ON
F7_ON  + F10_OFF -> Y2_ON
F7_ON  + F10_ON  -> Y2_ON

# G16b = F5 AND F8
F5_OFF + F8_OFF -> G16b_OFF
F5_OFF + F8_ON  -> G16b_OFF
F5_ON  + F8_OFF -> G16b_OFF
F5_ON  + F8_ON  -> G16b_ON
# G16 = NOT(F14 AND G16b)
F14_OFF + G16b_OFF -> G16_ON
F14_OFF + G16b_ON  -> G16_ON
F14_ON  + G16b_OFF -> G16_ON
F14_ON  + G16b_ON  -> G16_OFF
# G17 = F15 OR G12
F15_OFF + G12_OFF -> G17_OFF
F15_OFF + G12_ON  -> G17_ON
F15_ON  + G12_OFF -> G17_ON
F15_ON  + G12_ON  -> G17_ON
# Y1 = NOT(G16 AND G17)
G16_OFF + G17_OFF -> Y1_ON
G16_OFF + G17_ON  -> Y1_ON
G16_ON  + G17_OFF -> Y1_ON
G16_ON  + G17_ON  -> Y1_OFF
```

Compilation of a dual-rail implementation of a logic circuit computing the floor of the square root of a 4-bit binary number.

First, the logic circuit was translated into a CRN that consists of 32 uni- and bimolecular reactions, second, the CRN was compiled using Nuskell. The condensed enumerated reaction network has 316 species (52 signal species, 92 fuel species, 172 intermediate species), 180 reactions, and it verifies as correct according to pathway equivalence and CRN bisimulation. The simulations for four calculations are shown: 0000, 0001, 0100, 1001. Domain-level kinetic modeling calculates a computation time on the order of 27 hours.



[1] Badelt, S., Shin, S.W., Johnson, R.F., Dong, Q., Thachuk, C. and Winfree, E., 2017. A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities. International Conference on DNA-Based Computers.
[2] Shin, S.W., Thachuk, C. and Winfree, E., 2017. Verifying chemical reaction network implementations: A pathway decomposition approach. Theoretical Computer Science.
[3] Johnson, R., Dong, Q. and Winfree, E., 2018. Verifying chemical reaction network implementations: a bisimulation approach. Theoretical Computer Science.
[4] Grun, C., Sarma, K., Wolfe, B., Shin, S.W. and Winfree, E., 2015. A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures. arXiv preprint arXiv:1505.03738.
[5] Berleant, J., Berlind, C., Badelt, S., Dannenberg, F., Schaeffer, J. and Winfree, E., 2018. Automated sequence-level analysis of kinetics and thermodynamics for domain-level DNA strand-displacement systems. Journal of the Royal Society Interface.

Caltech Biology and Biological Engineering Division Fellowship
NSF Grant CCF-1213127
NSF Grant CCF-1317694 (“The Molecular Programming Project”)
The Gordon and Betty Moore Foundation's Programmable Molecular Technology Initiative (PMTI)