Universität Rostock Traditio et Innovatio

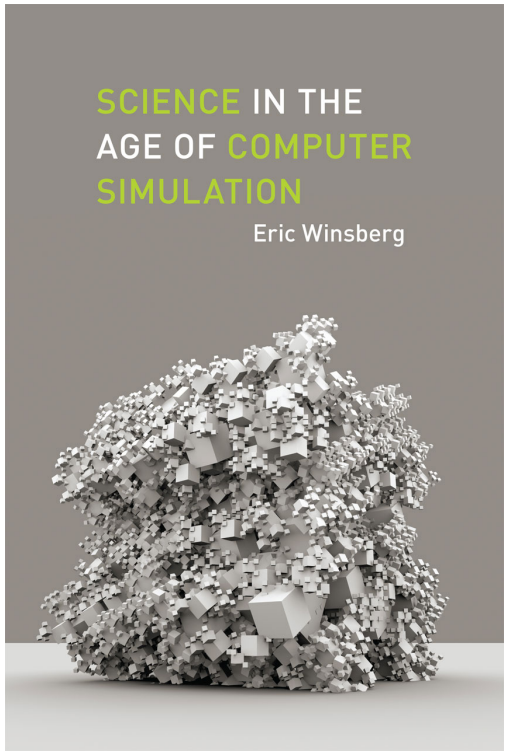
The Role of Domain Specific Languages in Modeling and Simulation

Towards reproducible simulation studies

Adelinde Uhrmacher
University of Rostock
Institute of Computer Science

Simulation between theory and experiment

Simulations are considered a “new” mode of science, between theory and (real) experiment.



Winsberg (2010): Science in the age of computer simulation

Modeling and Simulation an Experimental Science

- “A model for a system S and an Experiment E is anything to which E can be applied to answer questions about S”
- “A simulation is an experiment performed with a formal model and executed on a computer”

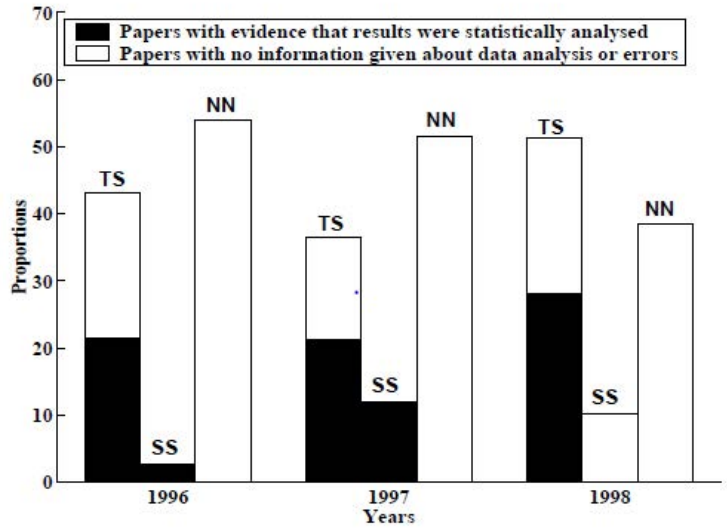
Cellier F. (1991): Continuous Systems Modeling, Springer.

Any scientific experiment requires reproducibility.

The credibility crisis of simulation

2002 “An opinion is spreading that one cannot rely on the majority of the published results on performance evaluation studies of telecommunication networks based on stochastic simulation, since they lack credibility”

2014 “there is no significant change with respect to quality and credibility of the simulation studies revised and the deep crisis of credibility still remains.”



(c) IEEE/ACM Transactions on Networking

Pawlikowski K. et al., (2002): Credibility of simulation studies of telecommunication networks. Communications Magazine, IEEE.

Sarkar N. et al., (2014): Revisiting the issue of the credibility of simulation studies in telecommunication networks: highlighting the results of a comprehensive survey of IEEE publications. Communications Magazine, IEEE

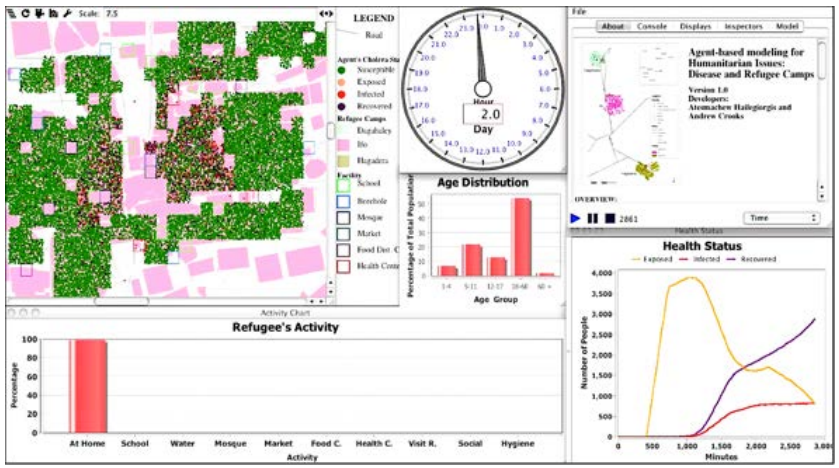
Requirements for changing this

adequate information about

- the model
  - conceptual and/or formal model
  - experiments used for the model's verification & validation
- experiments used to produce the results

Information about agent-based models

Agent-based approaches have become an established method in many areas, e.g., ecology or sociology, however “agent-based models are often too poorly documented for being evaluated”



Grimm V., et al. (2006): A standard protocol for describing individual-based and agent-based models. Ecological Modelling 198:115-126.

`http://css.gmu.edu/Cholera/Cholera/ODD.html`

The protocol ODD

ODD (=Overview, Design concepts, Detail) for describing agent-based models

- Purpose
- Entities, state variables, and scales
- Process overview and scheduling
- Design concepts
- Initialization
- Input data
- Submodels

Grimm V., et al. (2006): A standard protocol for describing individual-based and agent-based models. Ecological Modelling 198:115-126.

Limitations of ODD

- “ODD-based model descriptions . . . ; however, in most cases it will be necessary to have a simulation experiments or model analysis section following the model description”
- E.g., Process overview and scheduling: “Who (i.e., what entity) does what, and in what order? . . . Except for very simple schedules, one should use pseudo-code to describe the schedule in every detail, so that the model can be reimplemented ”

$\rightsquigarrow$ languages needed with accessible syntax, pruned for the purpose at hand, and with a clear semantics.

Grimm V., et al., (2010): The ODD protocol: a review and first update. Ecological Modelling 221: 2760-2768

Domain specific languages

- A domain-specific language (DSL) is a programming language that is tailored specifically for an application domain.
- A DSL "offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain."
  - *internal* DSLs – pro: minimal implementation effort, easily extendable, cons: similarity with the host language
  - *external* DSLs – pro: complete freedom of syntax, cons: own interpreter.

---

van Deursen A. et al. (2000): Domain-Specific Languages: An Annotated Bibliography. SIGPLAN Notices 35(6): 26-36

Domain specific languages for modeling and simulation.

DSLs for modeling

The syntax and semantics should reflect

- common metaphors of the domain, e.g., reaction-based for chemistry
- requirements of the domain, e.g., Continuous Time Markov Chain semantics to adequately describe observed noise, variability and heterogeneity of the system
- important characteristics, e.g., multi-level (up- and downward causations) or dynamic structures

Typical requirements for domain specific modeling languages

- compactness
- composibility
- ease of use
- sufficiently flexible (how much can be expressed?) and expressive (how easy can things be expressed?)
- formally well grounded

"'how to say it' affects the expressiveness, succinctness, executability, and composability of model descriptions as well as the complexity of computing with these descriptions. In other words, **syntax matters**."

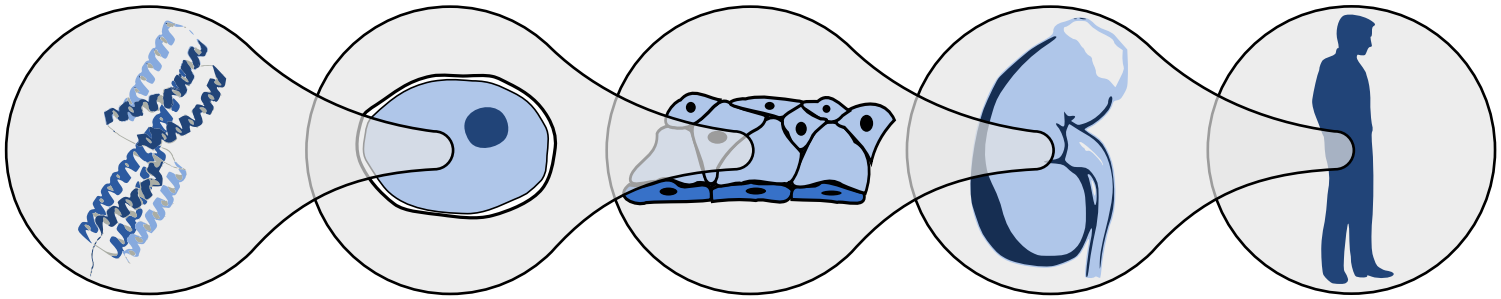
`http://www-tsb-workshop.imag.fr/abstract_henzinger.html`

Let's be more concrete: a modeling language for cell biological systems.

As with any language – first of all what would we like to describe?

Biological systems are hierarchically organized

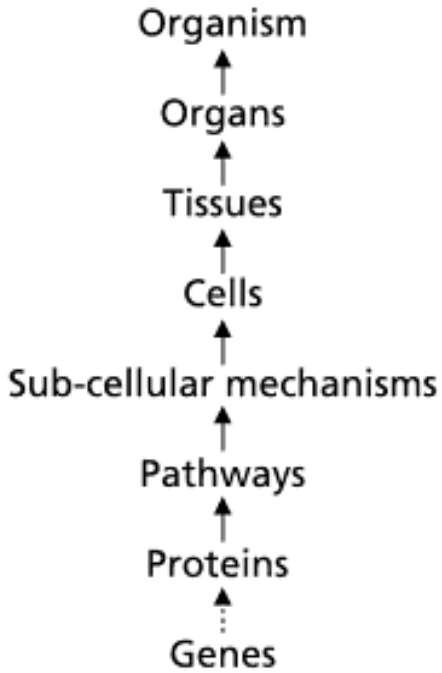
Many small parts form a larger part of which many form a larger part . . .

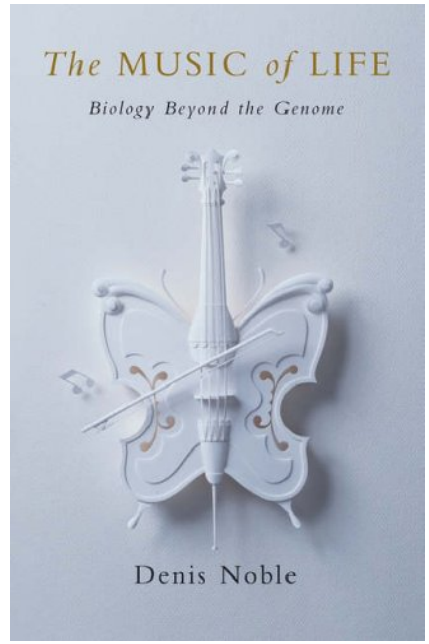


Different ‘levels of organization’ with different interacting entities.

Causality between different levels

Reductionist thinking

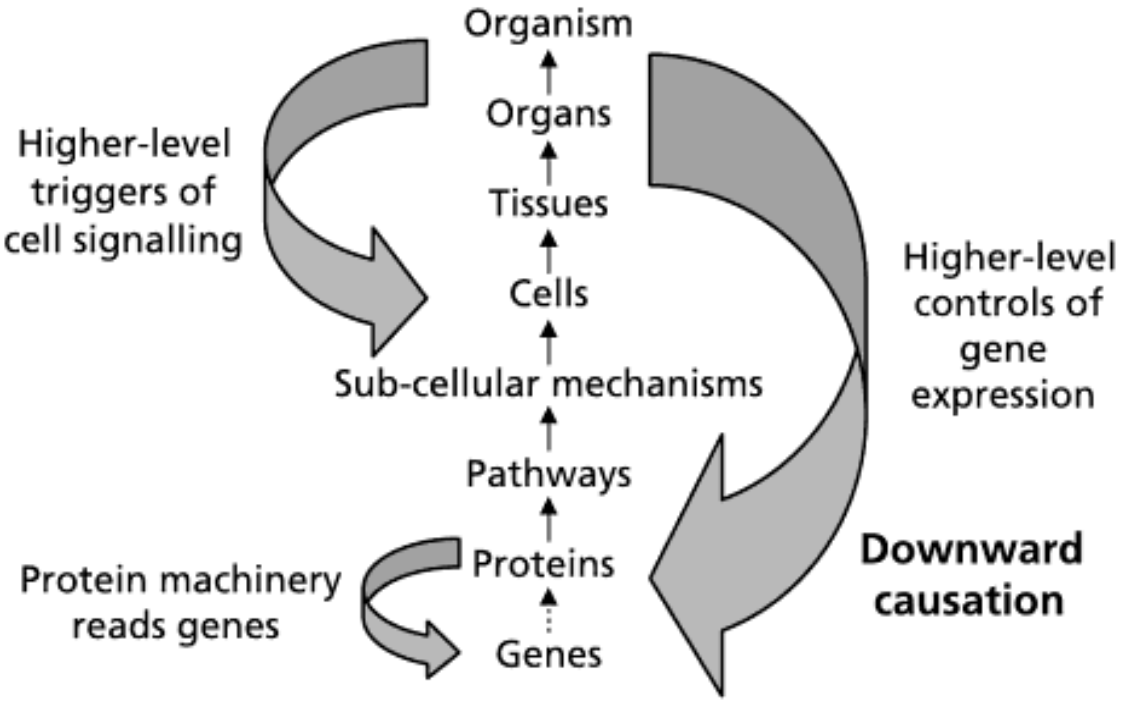


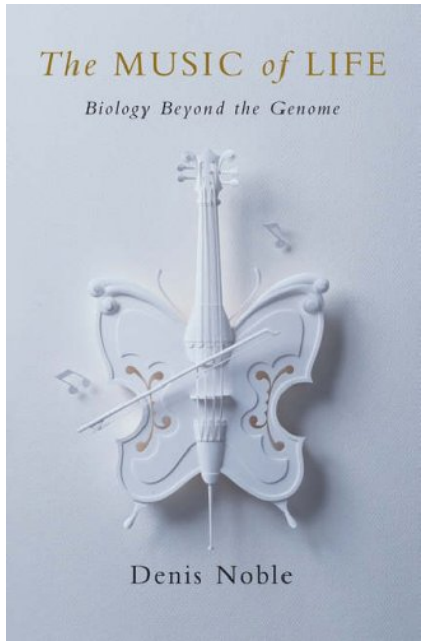


Denis Noble (2006). The Music of Life. Oxford University Press.

Causality between different levels

Complex systems involve upward AND downward causation





Denis Noble (2006). The Music of Life. Oxford University Press.

Upward and downward causation

- Upward causation: parts of a system influence the system as a whole.
- Downward causation: the system as a whole influences its parts.

"All processes at the lower level of a hierarchy are restrained by and act in conformity to the laws of the higher level." [1]

"The whole is to some degree constrained by the parts (upward causation), but at the same time the parts are to some degree constrained by the whole (downward causation)." [2]

[1] Donald T. Campbell (1974). 'Downward causation' in hierarchically organised biological systems. Studies in the philosophy of biology: Reduction and related problems, Macmillan Press, 179–186

[2] F. Heylighen (1995). Downward causation. Principia Cybernetica, http://pespmc1.vub.ac.be/DOWNCAUS.HTML

Upward and downward causation

- Upward causation: parts of a system influence the system as a whole.
- Downward causation: the system as a whole influences its parts.

"All processes at the lower level of a hierarchy are restrained by and act in conformity to the laws of the higher level." [1]

"The whole is to some degree constrained by the parts (upward causation), but at the same time the parts are to some degree constrained by the whole (downward causation)." [2]

[1] Donald T. Campbell (1974). 'Downward causation' in hierarchically organised biological systems. Studies in the philosophy of biology: Reduction and related problems, Macmillan Press, 179–186

[2] F. Heylighen (1995). Downward causation. Principia Cybernetica, http://pespmc1.vub.ac.be/DOWNCAUS.HTML

Upward and downward causation

- Upward causation: parts of a system influence the system as a whole.
- Downward causation: the system as a whole influences its parts.

"All processes at the lower level of a hierarchy are restrained by and act in conformity to the laws of the higher level." [1]

"The whole is to some degree constrained by the parts (upward causation), but at the same time the parts are to some degree constrained by the whole (downward causation)." [2]

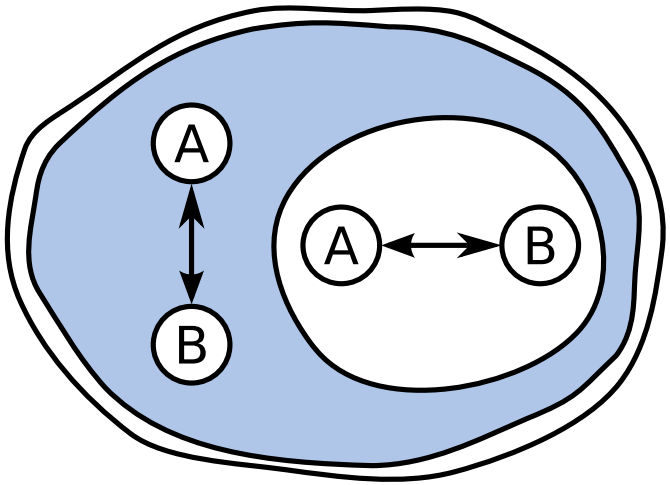
[1] Donald T. Campbell (1974). 'Downward causation' in hierarchically organised biological systems. Studies in the philosophy of biology: Reduction and related problems, Macmillan Press, 179–186

[2] F. Heylighen (1995). Downward causation. Principia Cybernetica, http://pespmc1.vub.ac.be/DOWNCAUS.HTML

How would we like to describe it?

Modeling with classical approaches (e.g. ODEs)

- Structure (different levels) only implicitly
- Leads to many similar model parts (redundancy)
  => high model complexity



$$\frac{d[A_{cyt}]}{dt} = k_r[B_{cyt}] - k_f[A_{cyt}]$$

$$\frac{d[A_{nuc}]}{dt} = k_r[B_{nuc}] - k_f[A_{nuc}]$$

$$\frac{d[B_{cyt}]}{dt} = k_f[A_{cyt}] - k_r[B_{cyt}]$$

$$\frac{d[B_{nuc}]}{dt} = k_f[A_{nuc}] - k_r[B_{nuc}]$$

Biologists typically don't like such things

$$\left\langle \Delta\xi_j^l(\mathbf{X}) \right\rangle \approx \sum_{i=1}^{N} \frac{\partial \xi_j^l(\mathbf{X})}{\partial x_{l,i}} \left(\sum_{j'=1}^{M+N} \left\langle \mathscr{P}\left(\xi_{j'}^l(\mathbf{X}), \tau\right) \right\rangle v_{ij'} + \langle b_i \rangle \right),$$

$$\mathrm{var}\left\{ \Delta\xi_j^l(\mathbf{X}) \right\} \approx \sum_{i=1}^{N} \left(\frac{\partial \xi_j^l(\mathbf{X})}{\partial x_{l,i}} \right)^2 \left(\sum_{j'=1}^{M+N} \mathrm{var}\left\{ \mathscr{P}(\xi_{j'}^l(\mathbf{X}), \tau) \right\} v_{ij'}^2 + \mathrm{var}\{b_i\} \right).$$

The mean and variance for the Poisson distribution are

$\langle P(\xi^l,\tau)\rangle = \mathrm{var}\{P(\xi^l,\tau)\} = \xi^l\tau.$

$$\left\langle \Delta\xi_j^l(\mathbf{X}) \right\rangle \approx \sum_{i=1}^{N} \frac{\partial \xi_j^l(\mathbf{X})}{\partial x_{l,i}} \left(\sum_{j'=1}^{M+N} \xi_{j'}^l(\mathbf{X}) \tau v_{ij'} + \beta_i^l(\mathbf{X}) \tau \right) \equiv \mu_j^l(\mathbf{X}) \tau,$$

$$\mathrm{var}\left\{ \Delta\xi_j^l(\mathbf{X}) \right\} \approx \sum_{i=1}^{N} \left(\frac{\partial \xi_j^l(\mathbf{X})}{\partial x_{l,i}} \right)^2 \left(\sum_{j'=1}^{M+N} \xi_{j'}^l(\mathbf{X}) \tau v_{ij'}^2 + \beta_i^l(\mathbf{X}) \tau \right) \equiv \left(\sigma_j^l(\mathbf{X}) \right)^2 \tau.$$

With the help of $\mu_j^l(\mathbf{X})$ and $\left(\sigma_j^l(\mathbf{X})\right)^2$, an expression for calculating a τ candidate can be found:

$$\tau = \min_{l\in[1,L]} \left\{ \min_{j\in[1,M+N]} \left\{ \frac{\max\left\{\epsilon\xi_j^l(\mathbf{X}), c_j\right\}}{\left|\mu_j^l(\mathbf{X})\right|}, \frac{\left(\max\left\{\epsilon\xi_j^l(\mathbf{X}), c_j\right\}\right)^2}{\left(\sigma_j^l(\mathbf{X})\right)^2} \right\} \right\}. \tag{7}$$

Jeschke et al. (2011): Exploring the performance of spatial stochastic simulation algorithms. J. Comput. Physics 230(7): 2562-2574 (2011)

ML-Rules - An external DSL for Modeling

A rule-based language for multi-level modeling and simulation in cell biology[1]

- multi-level modeling
- dynamic nesting (variable structure models)
- stochastic semantics

[1] Carsten Maus et al. (2011): Rule-based multi-level modeling of cell biological systems. BMC Systems Biology 5: 166

Rule-based modeling approach

- Molecule A with two modification sites
- Each site can be irreversibly and independently modified

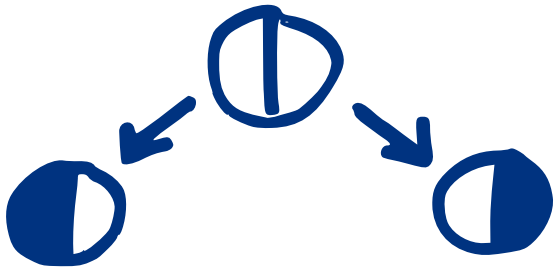
Set of reactions

Rule-based approach: attributed species and reaction patterns

```
A(u,*) -> A(p,*)
A(*,u) -> A(*,p)
```

Rule-based modeling approach

- Molecule A with two modification sites
- Each site can be irreversibly and independently modified



Set of reactions

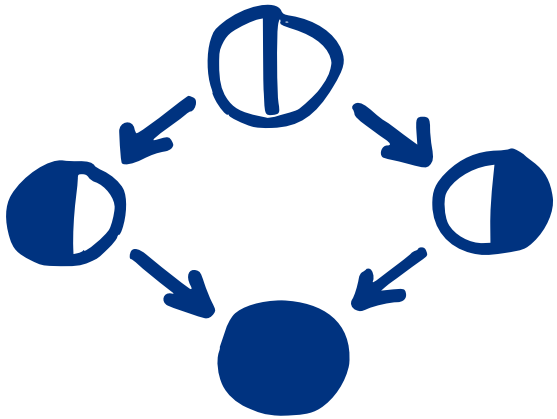
```
Auu -> Apu
Auu -> Aup
```

Rule-based approach: attributed species and reaction patterns

```
A(u,*) -> A(p,*)
A(*,u) -> A(*,p)
```

Rule-based modeling approach

- Molecule A with two modification sites
- Each site can be irreversibly and independently modified



Set of reactions

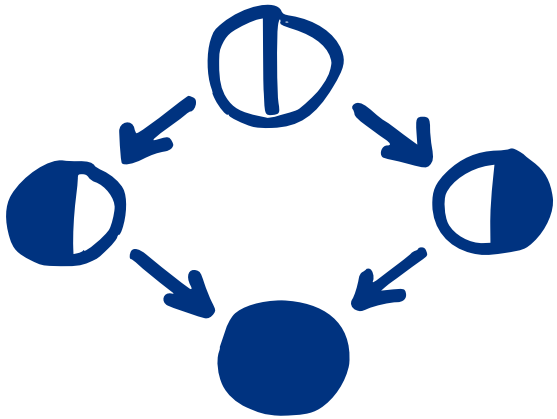
```
Auu -> Apu
Auu -> Aup
Apu -> App
Aup -> App
```

Rule-based approach: attributed species and reaction patterns

```
A(u,*) -> A(p,*)
A(*,u) -> A(*,p)
```

Rule-based modeling approach

- Molecule A with two modification sites
- Each site can be irreversibly and independently modified



Set of reactions

```
Auu -> Apu
Auu -> Aup
Apu -> App
Aup -> App
```

Rule-based approach: attributed species and reaction patterns

```
A(u,*) -> A(p,*)
A(*,u) -> A(*,p)
```

Effective reduction of combinatorial explosion

4 species
4 reactions

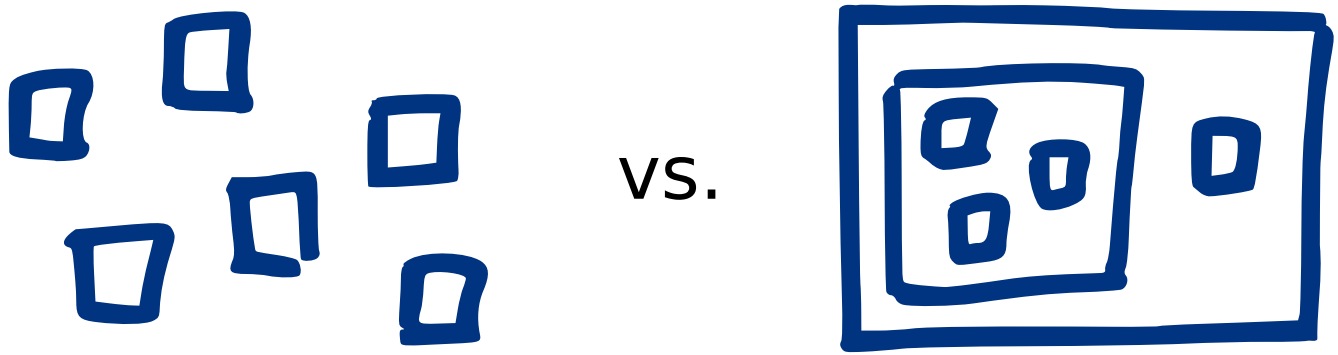
512 species
2304 reactions vs. 9 rules

Effective reduction of combinatorial explosion

4 species
4 reactions

512 species
2304 reactions vs. 9 rules

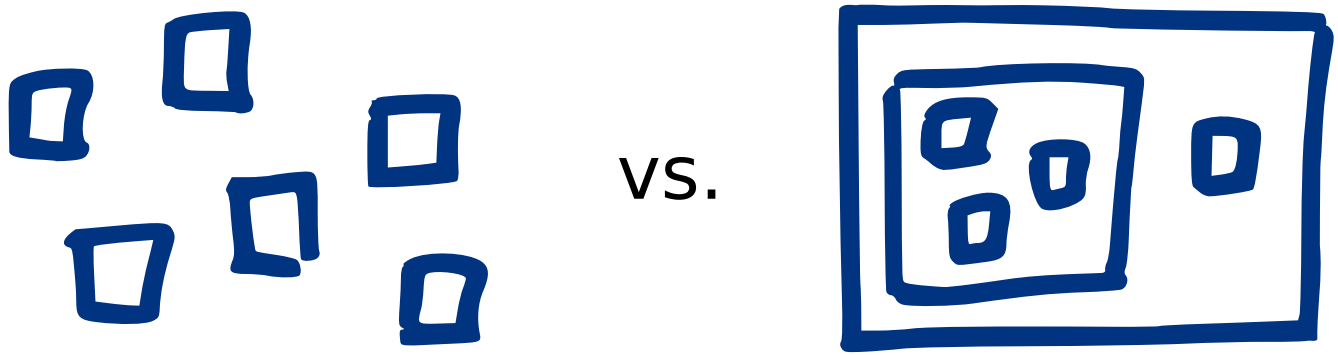
ML-Rules: nested species for modeling hierarchies



Example: A reacts to B and is enclosed by C

Implicit structure information: $A_C \rightarrow B_C$

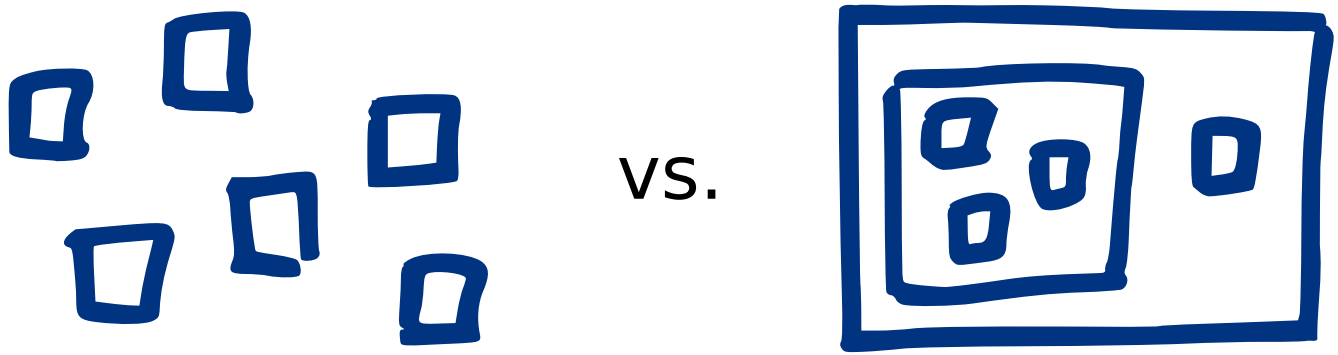
ML-Rules: nested species for modeling hierarchies



Example: A reacts to B and is enclosed by C

Implicit structure information: $A_C \rightarrow B_C$

ML-Rules: nested species for modeling hierarchies

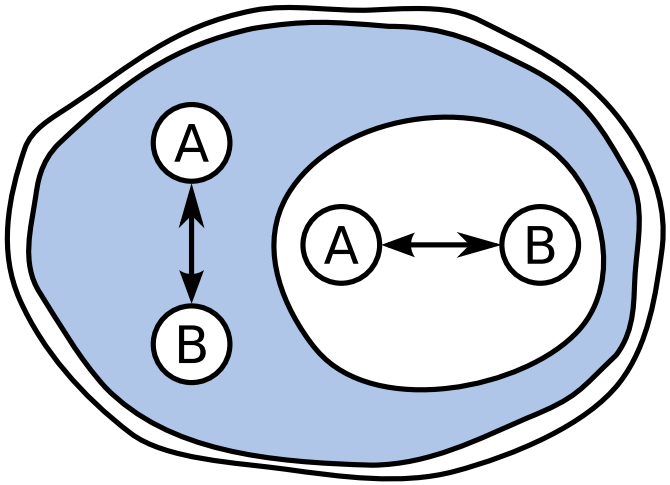


Example: A reacts to B and is enclosed by C

Implicit structure information: $A_C \rightarrow B_C$

Explicit structure information: $C[A] \rightarrow C[B]$

Multi-compartment model revisited



$$\frac{d[A_{cyt}]}{dt} = k_r[B_{cyt}] - k_f[A_{cyt}]$$

$$\frac{d[A_{nuc}]}{dt} = k_r[B_{nuc}] - k_f[A_{nuc}]$$

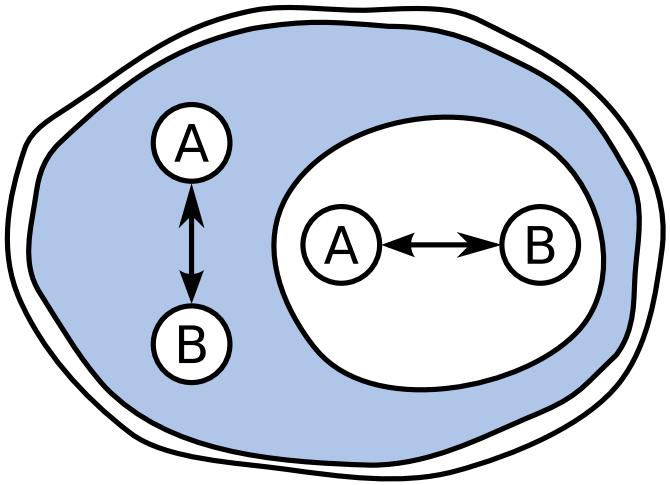
$$\frac{d[B_{cyt}]}{dt} = k_f[A_{cyt}] - k_r[B_{cyt}]$$

$$\frac{d[B_{nuc}]}{dt} = k_f[A_{nuc}] - k_r[B_{nuc}]$$

ML-Rules: reducing complexity by applying rules to different solutions

$$A^a \xrightarrow{a \cdot k_f} B$$
$$B^b \xrightarrow{b \cdot k_r} A$$

Multi-compartment model revisited



$$\frac{d[A_{cyt}]}{dt} = k_r[B_{cyt}] - k_f[A_{cyt}]$$

$$\frac{d[A_{nuc}]}{dt} = k_r[B_{nuc}] - k_f[A_{nuc}]$$

$$\frac{d[B_{cyt}]}{dt} = k_f[A_{cyt}] - k_r[B_{cyt}]$$

$$\frac{d[B_{nuc}]}{dt} = k_f[A_{nuc}] - k_r[B_{nuc}]$$

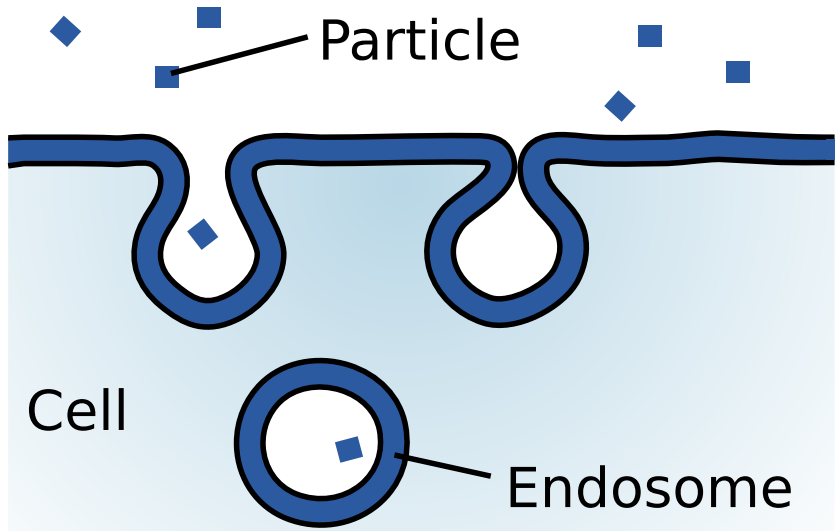
ML-Rules: reducing complexity by applying rules to different solutions

$$A^a \xrightarrow{a \cdot k_f} B$$
$$B^b \xrightarrow{b \cdot k_r} A$$

Dynamic manipulation of model hierarchies

Example: endo- and exocytosis

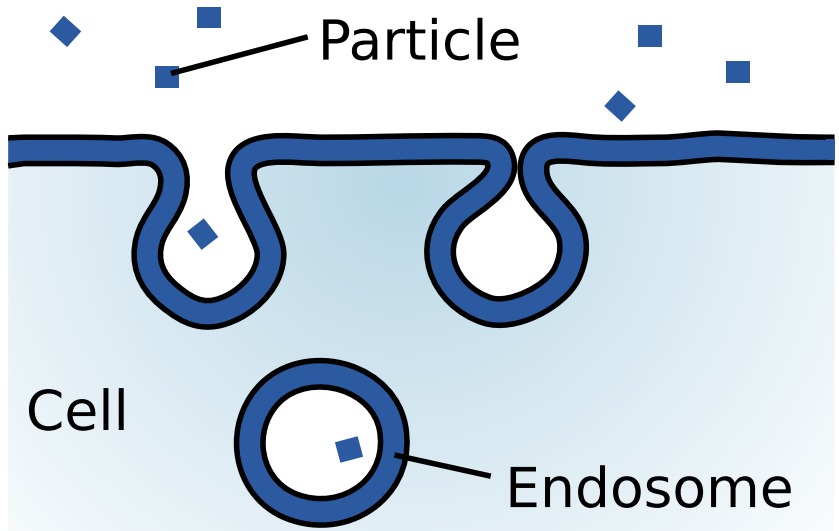
$$Cell[\,] + Particle \longleftrightarrow Cell[Endosome[Particle]]$$



Dynamic manipulation of model hierarchies

Example: endo- and exocytosis

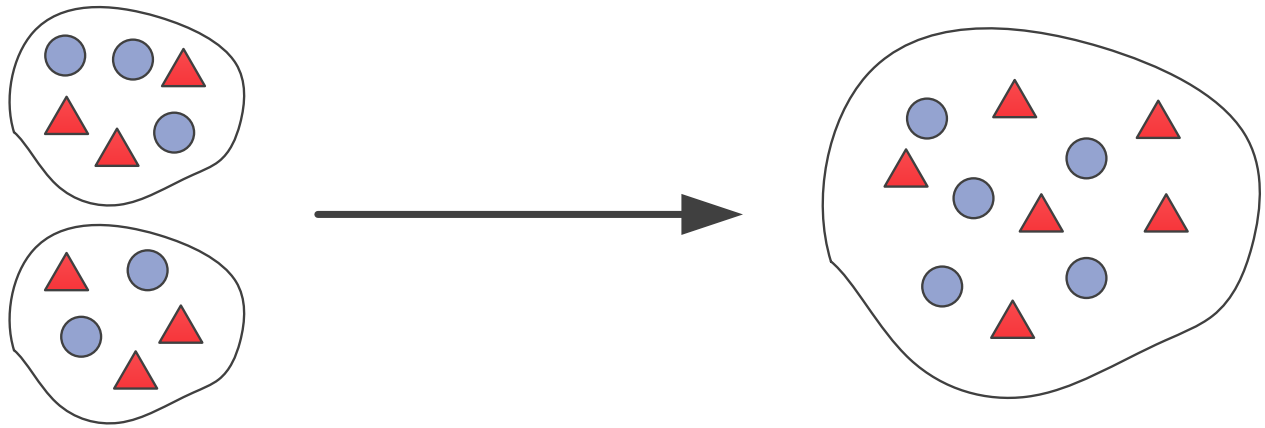
$$Cell[solution?] + Particle \longleftrightarrow Cell[Endosome[Particle] + solution?]$$



Dynamic manipulation of model hierarchies

Example: mitochondrion fusion

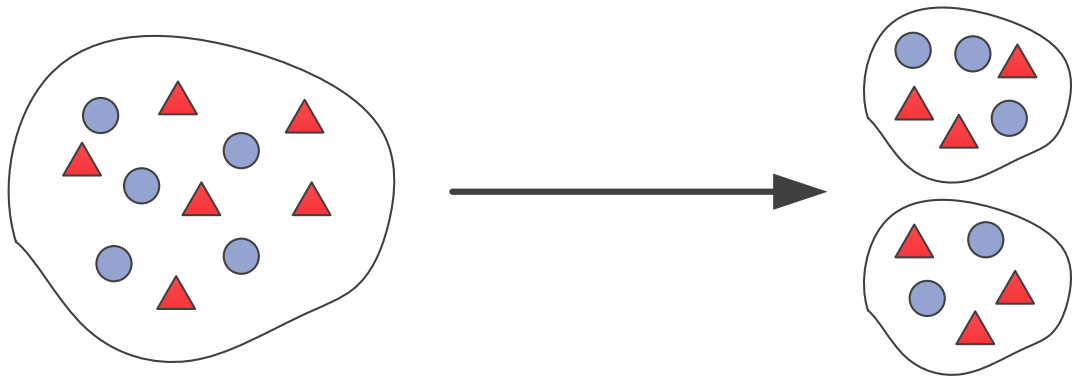
$$Mitochondrion[s1?] + Mitochondrion[s2?] \rightarrow Mitochondrion[s1? + s2?]$$



Dynamic manipulation of model hierarchies

Example: mitochondrion fission

$$Mitochondrion[s?] \rightarrow Mitochondrion[s1?] + Mitochondrion[s2?]$$
$$\text{where } (s1?, s2?) = split(s?, 0.5)$$



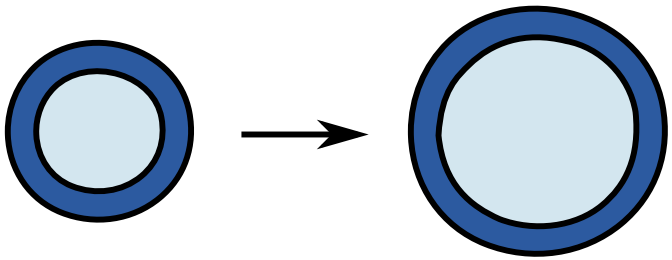
Species attributes: own state at each level

Attributes allow to equip each level with own states and dynamics that are constrained by their attributes.

E.g., the size of a cell may be described by an attribute of the *Cell* species.

Example: cell growth

$$Cell(volume)[sol?] \rightarrow Cell(volume + \Delta V)[sol?]$$



So far a compact, succinct, flexible, and expressive description of cell biological systems, however what about its formal semantics?

Modeling approaches

Some are focused on specific application domains, e.g., manufacturing, networks, or cell biology,

- ⇝ although easy to use, typically are not flexible and lack some formal grounding

Some are flexible and formally well grounded, e.g., DEVS or process algebras like the π calculus

- ⇝ typically are not easy to use (or/and do not lead to compact models)

Modeling approaches

Some are focused on specific application domains, e.g., manufacturing, networks, or cell biology,

$\rightsquigarrow$ although easy to use, typically are not flexible and lack some formal grounding

Some are flexible and formally well grounded, e.g., DEVS or process algebras like the π calculus

$\rightsquigarrow$ typically are not easy to use (or/and do not lead to compact models)

Modeling approaches

Some are focused on specific application domains, e.g., manufacturing, networks, or cell biology,

$\rightsquigarrow$ although easy to use, typically are not flexible and lack some formal grounding

Some are flexible and formally well grounded, e.g., DEVS or process algebras like the π calculus

$\rightsquigarrow$ typically are not easy to use (or/and do not lead to compact models)

Modeling approaches

Some are focused on specific application domains, e.g., manufacturing, networks, or cell biology,

$\rightsquigarrow$ although easy to use, typically are not flexible and lack some formal grounding

Some are flexible and formally well grounded, e.g., DEVS or process algebras like the π calculus

$\rightsquigarrow$ typically are not easy to use (or/and do not lead to compact models)

Modeling approaches

Some are focused on specific application domains, e.g., manufacturing, networks, or cell biology,

$\rightsquigarrow$ although easy to use, typically are not flexible and lack some formal grounding

Some are flexible and formally well grounded, e.g., DEVS or process algebras like the π calculus

$\rightsquigarrow$ typically are not easy to use (or/and do not lead to compact models)

Can't we have our cake, and eat it too?

Abstract syntax snippet

Patterns	p	::=	$S(\tilde{e}, p_r)^e \triangleright x \mid p + p \mid 0$
Patterns with rest	p_r	::=	$p + y$

- describe solutions
- relevant species are listed with attributes, contents are recursively defined
- remaining species subsumed in rest solution

Rules	r	::=	$p_r \xrightarrow{e} (\nu \tilde{x}) e$

- describe reactions
- reacting solution specified on left side
- reaction speed given by rate expression
- right side specifies result

Warnke T. et al. (2015): A Multi-Level Modeling Language for Simulating Cell Biological Systems. Proc. of PADS '15.

Operational Semantics: Big Step Evaluator and Context

Rules plus current solution $\rightsquigarrow$ identifying possible reactions to be applied.

. . . expressions - application of lambda function:

$$(5)\ \frac{e_1 \Downarrow_{s,\sigma} \lambda x.e \qquad e_2 \Downarrow_{s,\sigma} v_2 \qquad e[v_2/x] \Downarrow_{s,\sigma} v}{e_1\ e_2 \Downarrow_{s,\sigma} v}$$

. . . pattern matching - pattern with rest:

$$(14)\ \frac{y \in \mathscr{R} \qquad p \Downarrow_{s,\sigma} s_1 \qquad s \approx s_1 + s_2}{p + y \Downarrow_{s,\sigma \cup (y \mapsto s_2)} s_1 + s_2}$$

Operational Semantics: CTMC semantics

Instantiation of a rule to a reaction

$$\frac{\sigma : (fv(p+y)) \mapsto Vals \text{ type preserving} \qquad \sigma' : \tilde{x} \mapsto \mathscr{L}^U \text{ injective} \qquad p+y \Downarrow_{s,\sigma} s \qquad e_1 \Downarrow_{s,\sigma} r \qquad e_2 \Downarrow_{s,\sigma\cup\sigma'} s_2}{p+y \xrightarrow{e_1} (\nu\tilde{x})e_2 \Downarrow_{s,\sigma} s \xrightarrow{r} s_2}$$

Summing up the rates

$$\frac{s_1 \equiv s_1' \quad \left(\left(\sum_{R\in\text{Rules}} \sum_{\{(r',\sigma) \mid R \Downarrow_{s_1',\sigma} \rho,\ \rho \vdash s_1' \xrightarrow{r'} s_2' \equiv s_2\}} r' \right) + \sum_{\{(r',k) \mid s_1' \xrightarrow[k]{r'} s_2' \equiv s_2\}} r' \right) = r}{\text{Rules} \vdash s_1 \xrightarrow{r} s_2}$$

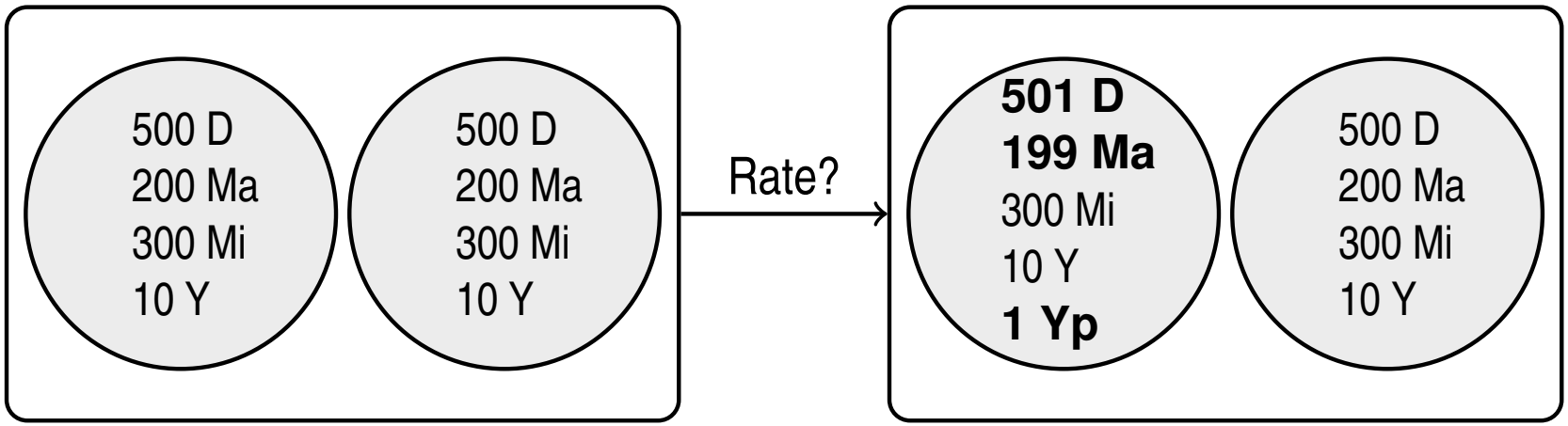
and this for all sub-solutions

$$\frac{k \in \{1,\ldots,n\} \qquad a_k = S(\tilde{v},s) \qquad \text{Rules} \vdash s \xrightarrow{r} s'}{\sum_{i=1}^{n} a_i \xrightarrow[k]{r} \left(\sum_{i=1}^{k-1} a_i + S(\tilde{v},s') + \sum_{i=k+1}^{n} a_i \right)}$$

Continuous Time Markov Chain

A rule (for the breakage of an activated MPF complex):

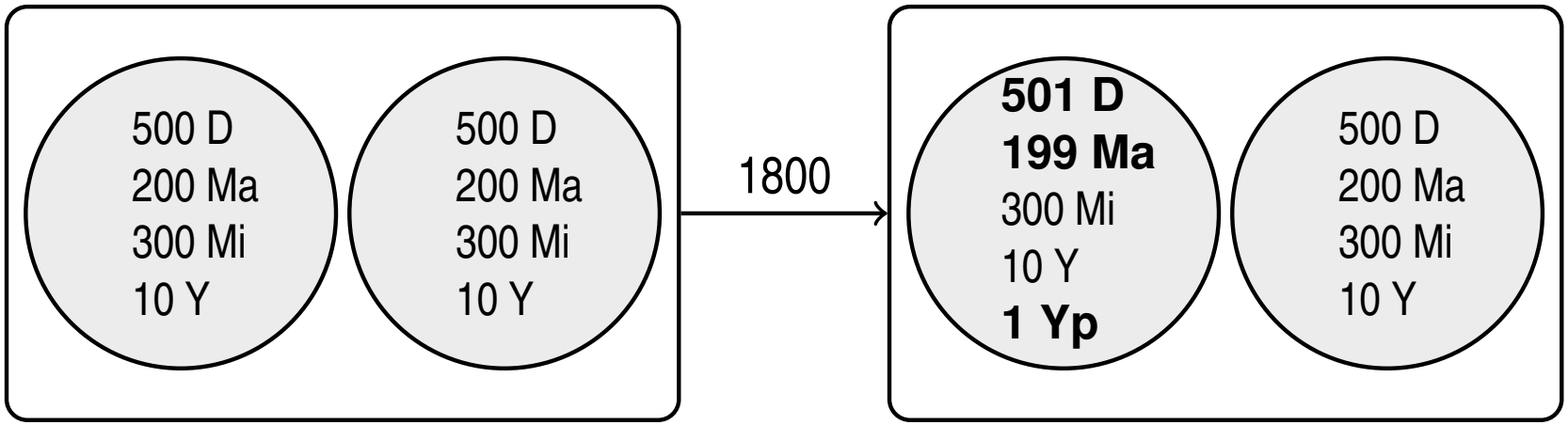
```
C(v,p)[Ma:a + s?]:c -> C(v,p)[Yp + D + s?] @ (k4/v)*#a*#c;
```



Continuous Time Markov Chain

```
C(v,p)[Ma:a + s?]:c -> C(v,p)[Yp + D + s?] @ (4.5/v)*#a*#c;
```

Evaluation of rate with constant $k_4 = 4.5$ and cell volume $v = 1.0$:
$(4.5 \div 1.0) \times 200 \times 2 = 1800$



Pseudo code of simulator

nextstep $(sol, rules, t)$ {
  $reactions$:= inst$(sol, rules)$
  r_{sum} := $\sum_{(\_,\_,r,\_)\in reactions} r$
  t := t + Exp(r_{sum})
  $reaction$:= select $(reactions, r_{sum})$
  sol := execute $(reaction, sol)$
}

inst $(sol, rules)$ {
  rs := $\emptyset$
  for $(p + y \xrightarrow{e_1} (\nu\tilde{x})e_2 \in rules)$
    rs := rs $\cup \{(sol, \sigma, r, (\nu\tilde{x})e_2) | p + y \Downarrow_{sol,\sigma} sol, e_1 \Downarrow_{sol,\sigma} r\}$
  for $(S(\tilde{v}, sol_{sub}) \in sol)$
    rs := $rs \cup$ inst$(sol_{sub}, rules)$
  return rs
}

Formal semantics are always a bit scary

An operational semantics for Simulink:

$$\frac{}{\langle r,\sigma\rangle \xrightarrow{o} r}\ \text{CTE} \qquad \frac{}{\langle \ell,\sigma\rangle \xrightarrow{o} \sigma(\ell)}\ \text{VAR} \qquad \frac{\langle e_1,\sigma\rangle \xrightarrow{o} r_1 \quad \langle e_2,\sigma\rangle \xrightarrow{o} r_2}{\langle e_1 \diamond e_2,\sigma\rangle \xrightarrow{o} r_1 \diamond r_2}\ \text{ARITH} \qquad \frac{\langle e_1,\sigma\rangle \xrightarrow{o} r_1 \quad \langle e_2,\sigma\rangle \xrightarrow{o} r_2}{\langle e_1 \bowtie e_2,\sigma\rangle \xrightarrow{o} r_1 \bowtie r_2}\ \text{CMP}$$

$$\frac{\langle e_1,\sigma\rangle \xrightarrow{o} \text{true} \quad \langle e_2,\sigma\rangle \xrightarrow{o} r_2}{\langle \text{if } (e_1,e_2,e_3),\sigma\rangle \xrightarrow{o} r_2}\ \text{THEN} \qquad \frac{\langle e_1,\sigma\rangle \xrightarrow{o} \text{false} \quad \langle e_3,\sigma\rangle \xrightarrow{o} r_3}{\langle \text{if } (e_1,e_2,e_3),\sigma\rangle \xrightarrow{o} r_3}\ \text{ELSE} \qquad \frac{\langle eq_1,\sigma\rangle \xrightarrow{o} \sigma' \quad \langle eq_2,\sigma'_g\rangle \xrightarrow{o} \sigma''_g}{\langle eq_1; eq_2,\sigma\rangle \xrightarrow{o} \sigma''_g}\ \text{SEQ}$$

$$\frac{\sigma(t) \notin S}{\langle \ell :=_S e,\sigma\rangle \xrightarrow{o} \sigma}\ \text{AFFNS} \qquad \frac{\sigma(t) \in S \quad \langle e,\sigma\rangle \xrightarrow{o} r}{\langle \ell :=_S e,\sigma\rangle \xrightarrow{o} \sigma[\ell \leftarrow r]}\ \text{AFFS} \qquad \frac{\langle e,\sigma\rangle \xrightarrow{o} r}{\langle \ell := e,\sigma\rangle \xrightarrow{o} \sigma[\ell \leftarrow r]}\ \text{AFF}$$

$$\frac{\langle \text{Eq}(m),\sigma\rangle \xrightarrow{o} \sigma'}{\text{Eq},\pi \vdash \sigma \xrightarrow{m} \sigma'}\ \text{MINOR} \qquad \frac{\langle \text{Eq}(M),\sigma\rangle \xrightarrow{o} \sigma'}{\text{Eq},\pi \vdash \sigma \xrightarrow{M} \sigma'}\ \text{MAJOR} \qquad \frac{\text{Eq}(\boldsymbol{d}) = \overline{\boldsymbol{d}} :=_S \boldsymbol{\ell} \quad \sigma(t) \notin S}{\text{Eq},\pi \vdash \sigma \xrightarrow{u} \sigma}\ \text{UPDATENS} \qquad \frac{\text{Eq}(\boldsymbol{d}) = \overline{\boldsymbol{d}} :=_S \boldsymbol{\ell} \quad \sigma(t) \in S}{\text{Eq},\pi \vdash \sigma \xrightarrow{u} \sigma[\boldsymbol{d} \mapsto \sigma(\boldsymbol{\ell})]}\ \text{UPDATES}$$

Figure 4. The output and update transitions.

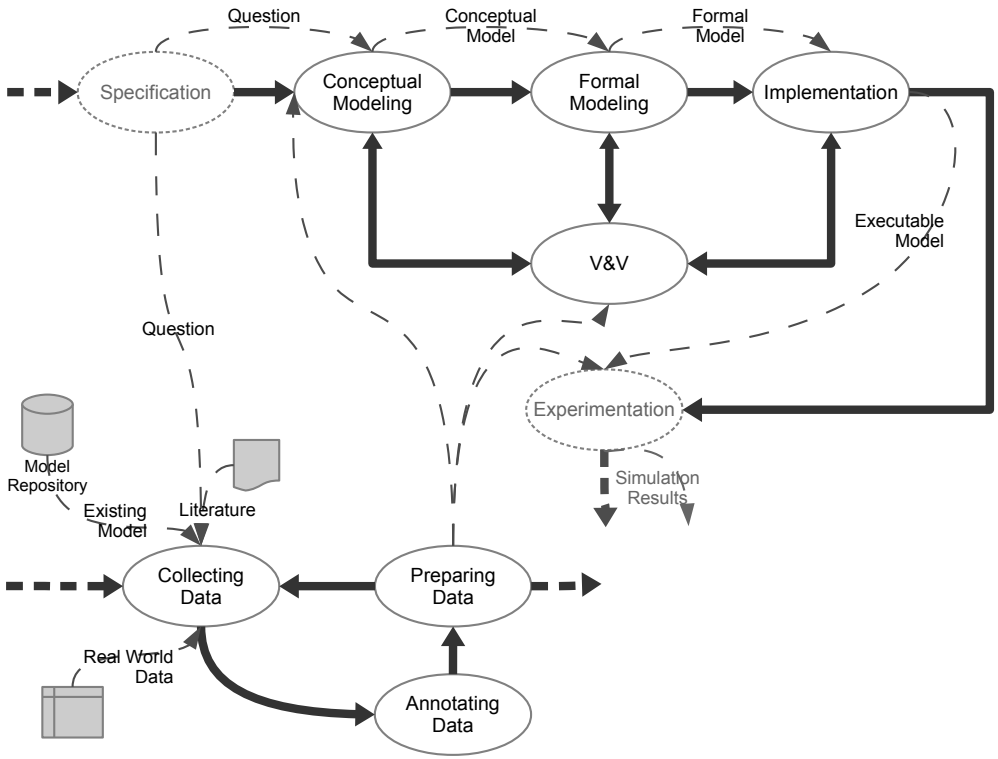
Bouissou et al., (2012): An operational semantics for Simulink's simulation engine. LCTES: 129-138

Summary DSLs for modeling

- offer the possibility to combine a compact, succinct description of the model
- with a clear syntax and formal semantics
- however, specificity and generality need always to be balanced

Now to DSLs for experimentation, first of all what do we want to describe?

In-Silico Experiments



Rybacki S. et al. (2014): Developing simulation models - from conceptual to executable model and back - an artifact-based workflow approach. Proc. of Simutools '14

For any experiment, the model need to be executed

- Parallelization: fine-grained (within one single simulation run) or/and coarse-grained (over multiple simulation runs)
- Exploiting GPUs
- Approximative methods: trading accuracy for speed
- Suitable configuration of simulation engines
- Adaptive, multi-algorithm methods: overview and detail on demand

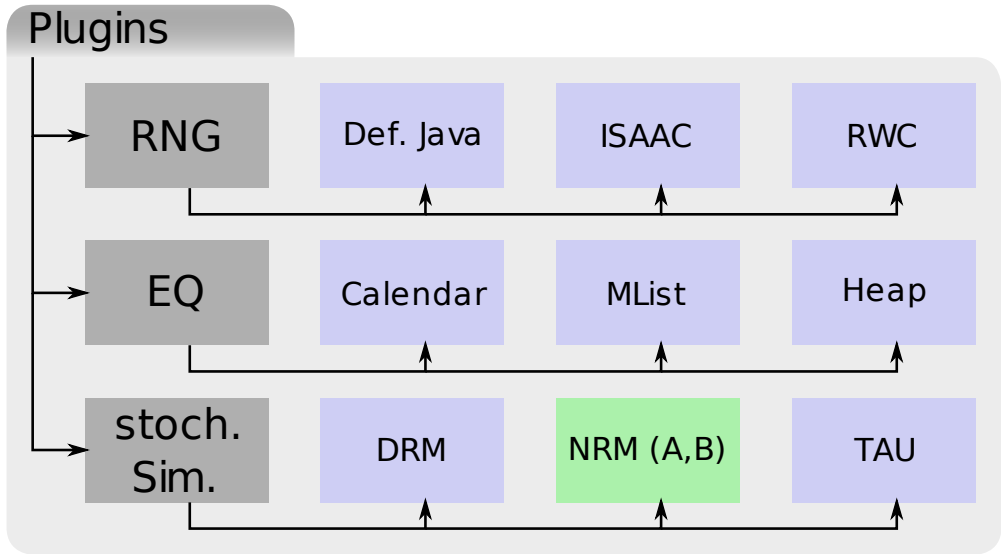
horses for courses

https://en.wikipedia.org/wiki/Edgar_Degas

Configuration of simulators

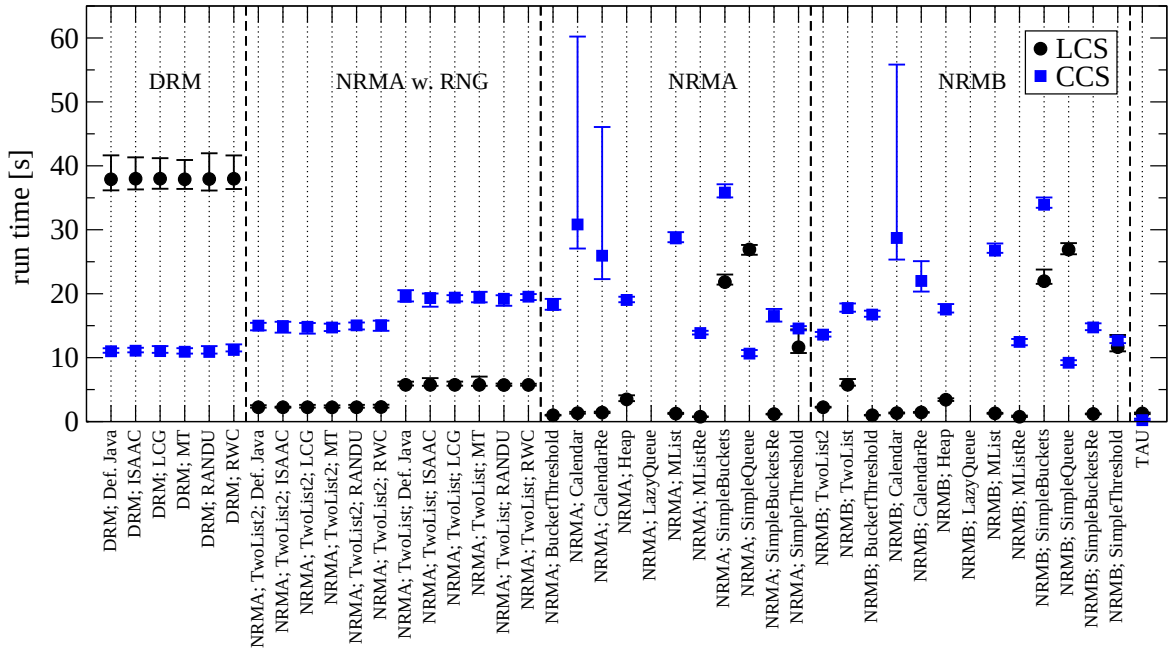
Simulation algorithms comprise many sub-algorithms, e.g.

- event queue (EQ) implementation
- random number generator (RNG) implementation



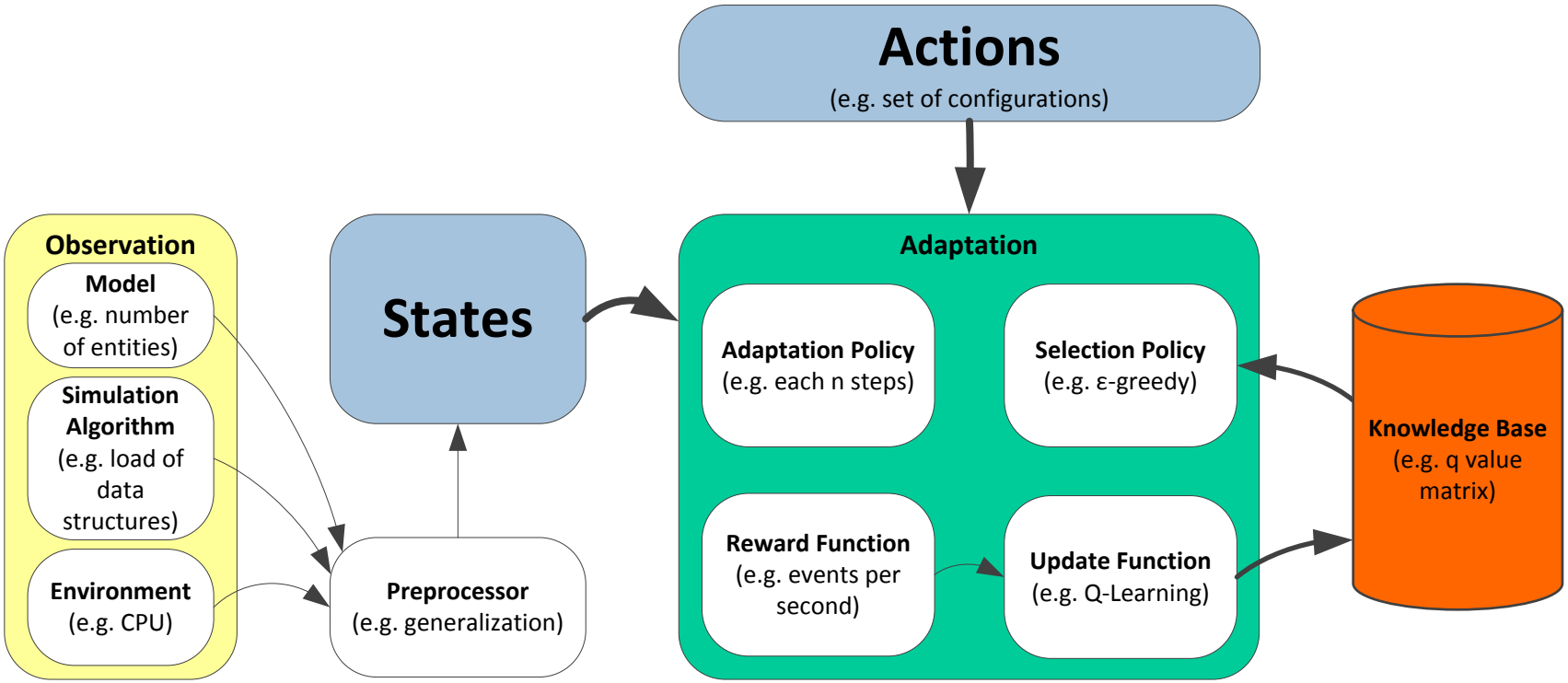
Himmelspach et al. (2007): The event queue problem and PDEVS, Proc. of SpringSim '07

Performance of configured Gillespie variants



Jeschke et al., (2008): Large-Scale Design Space Exploration of SSA. Proc. of CMSB '08

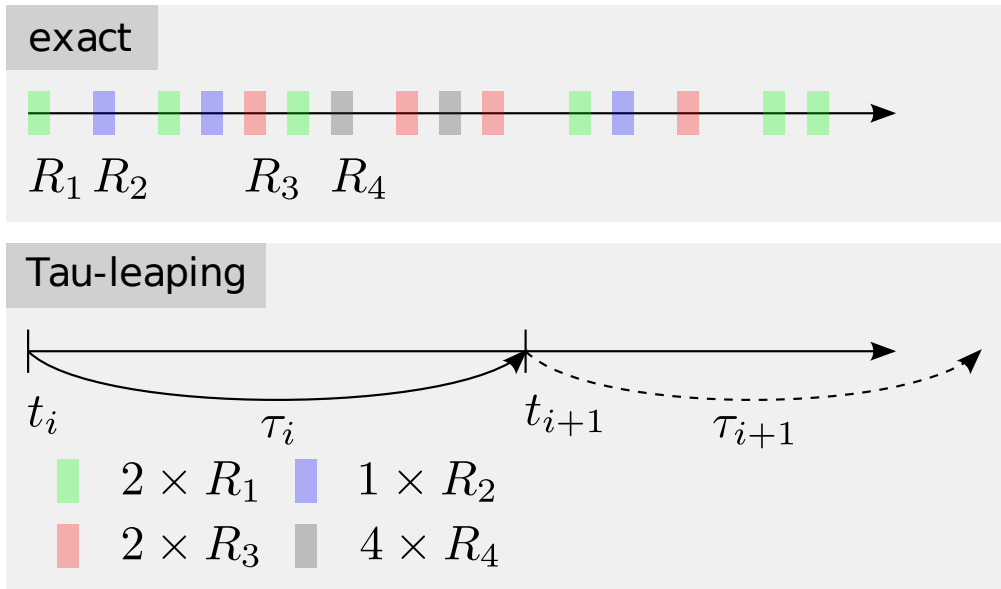
Adaptive Automated Configuration of Algorithms



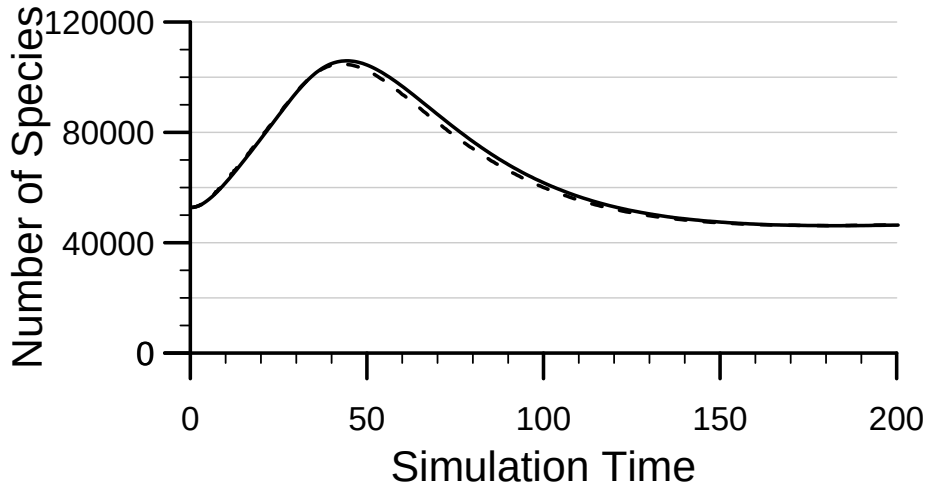
Helms T. et al. (2013): A Generic Adaptive Simulation Algorithm for Component-based Simulation Systems. Proc. of PADS '13

Trading accuracy for speed: Tau-leaping

- leap condition: how far can I leap without the state changing not too much, referring to reactions and in- and out-diffusion
- might fall back to basic NSM or Direct Method
- various control parameters



Applied in a concrete study

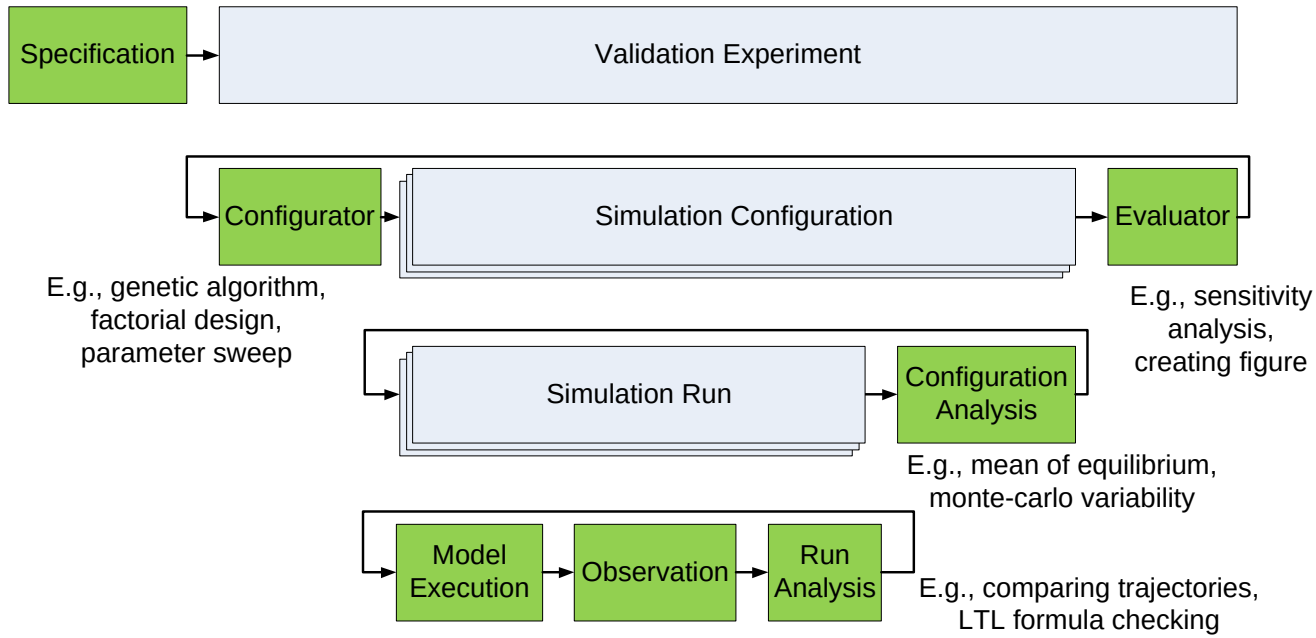


- SSA: $\approx 39s$ and $\approx 600,000$ reactions
- Tau-Leaping: $\approx 1.8s$ and $\approx 15,000$ leaps to $\approx 0.5s$ and $\approx 5,000$ leaps

Helms T. et al. (2013): An approximate execution of rule-based multi-level models, Proc. of CMSB '13

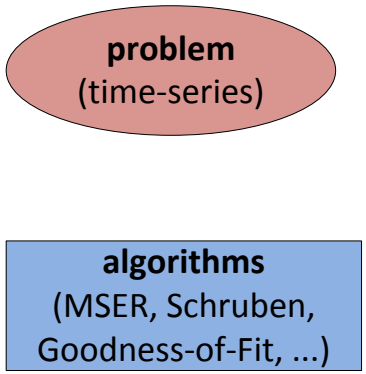
Good execution algorithms is a must, but more is needed!

Experiments - more than only one run



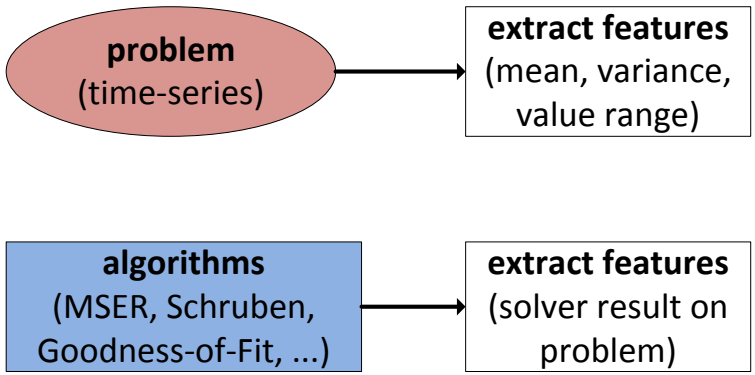
Leye S. et al. (2010): A flexible and extensible architecture for experimental model validation. Proc. of SimuTools 2010

Generating synthetic problem solvers by ensemble learning



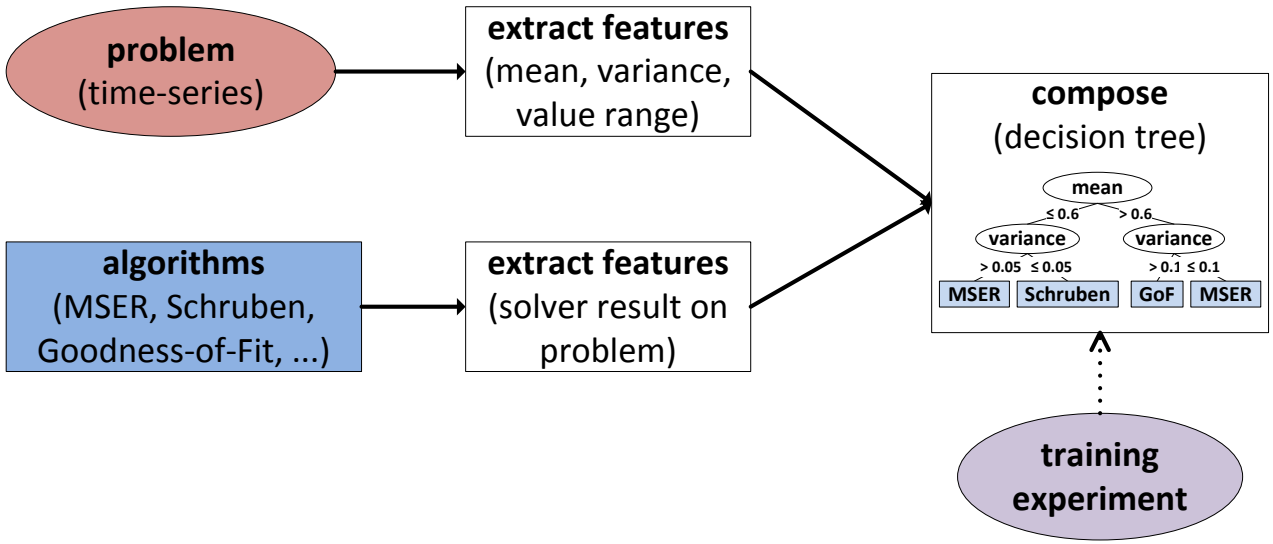
Leye S. et al. (2014): Composing Problem Solvers for Simulation Experimentation: A Case Study on Steady State Estimation. PLoS ONE 9(4)

Generating synthetic problem solvers by ensemble learning



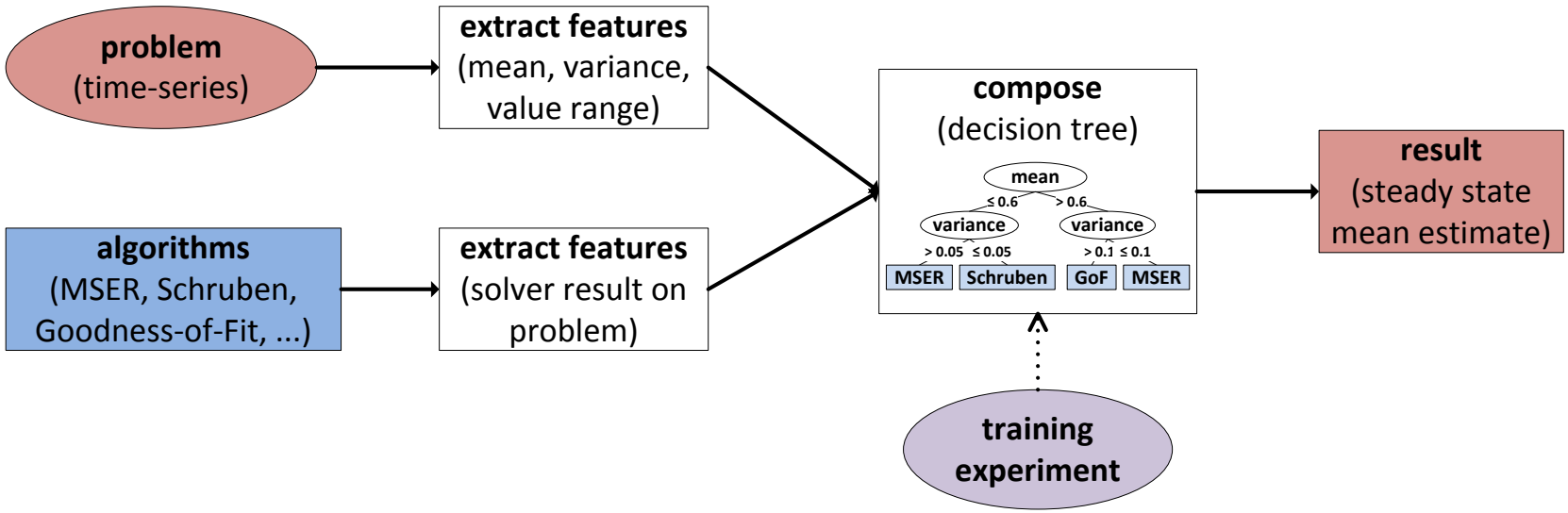
Leye S. et al. (2014): Composing Problem Solvers for Simulation Experimentation: A Case Study on Steady State Estimation. PLoS ONE 9(4)

Generating synthetic problem solvers by ensemble learning



Leye S. et al. (2014): Composing Problem Solvers for Simulation Experimentation: A Case Study on Steady State Estimation. PLoS ONE 9(4)

Generating synthetic problem solvers by ensemble learning



Leye S. et al. (2014): Composing Problem Solvers for Simulation Experimentation: A Case Study on Steady State Estimation. PLoS ONE 9(4)

How to describe these kind of experiments ?

SESSL

SESSL (Simulation Experiment Specification via a scala layer) is an internal domain specific language for specifying experiments.

```
execute { // execute experiment
  new Experiment with ParallelExecution { // create experiment
    model = "sampleModel.file" // use model stored in this file
    // complex stopping and replication conditions are supported:
    stopCondition = AfterSimTime(0.6) and
        (AfterWallClockTime(seconds = 30) or AfterSimSteps(10000))
    replications = 100
    rng = MersenneTwister(1234) // use random number generator
    parallelThreads = -1 // exploit parallelism, leave one core idle
    // define factorial experiment:
    scan("x" <~ (1, 2), "y" <~ range(1, 1, 10)) } }
```

Ewald R. et al. (2014): SESSL: A Domain-Specific Language for Simulation Experiments, TOMACS

Optimization

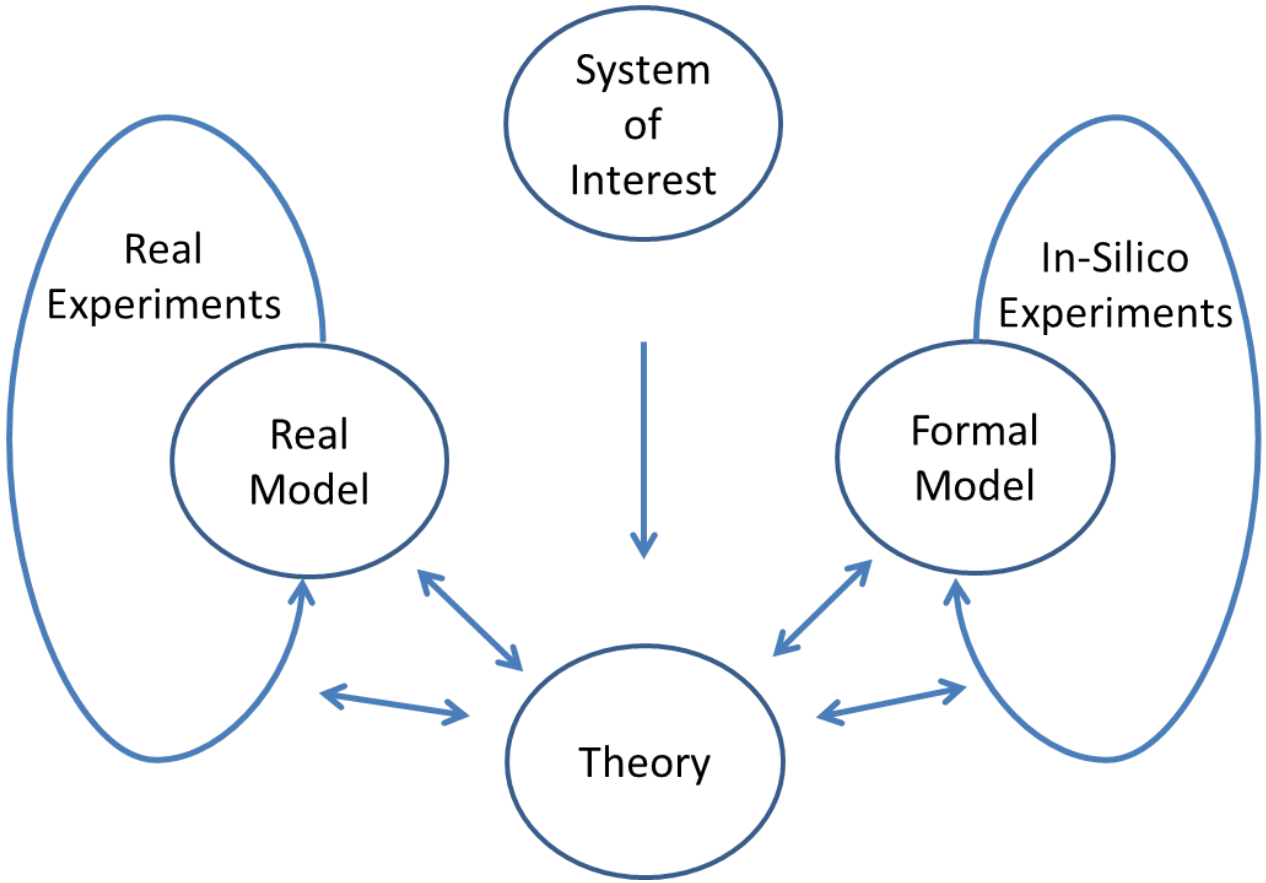
```
val ref = Set(0, 7561, 8247, 7772, 7918, 7814, 7702)
minimize { (params, objective) =>
 execute {
  new Experiment with Observation with ParallelExecution {
    model = "file-mlrj:/." + dir + "/Wnt_apCrine.mlrj"
      // Set model parameters as defined by optimizer:
      set("kLphos" <~ params("p")) ....
      observe("Cell/Nuc/Bcat()")....
      withRunResult(results => {
        runResults += scala.math.sqrt(mse(numbers, ref))})
      withReplicationsResult(results => {
        // Store value of objective function:
        objective <~ runResults /count }) } }
} using (new Opt4JSetup {
  param("p", 0.1, 0.1, 10) // Optimization parameter bounds
  optimizer = sessl.opt4j.SimulatedAnnealing ... }})
```

Statistical model checking

```
val exp = new Experiment with Hypothesis {

  //model configuration
  model = "file-sr:/./LotkaVolterra.mlrj"
  set("nWolf" <~ 50,
      "nFox" <~ 500,
      "nFood" <~ 100)

  //simulation configuration
  simulator = MLRulesTauLeaping()
  replications = 10
  stopCondition = AfterSimTime(500)

  //property
  assume{(Probability >= 0.8)(
    P(Peak("wolf","wolfPeakH"), time < 250, "wolfNumPeaks"),
    Id("wolfPeakH") > 90 and Id("wolfPeakH") < 110,
    E(Increase("wolf"), length >= 100, "wolfNumIncreases"),
    Id("wolfNumIncreases") after Id("wolfNumPeaks")
    )}
  }
```

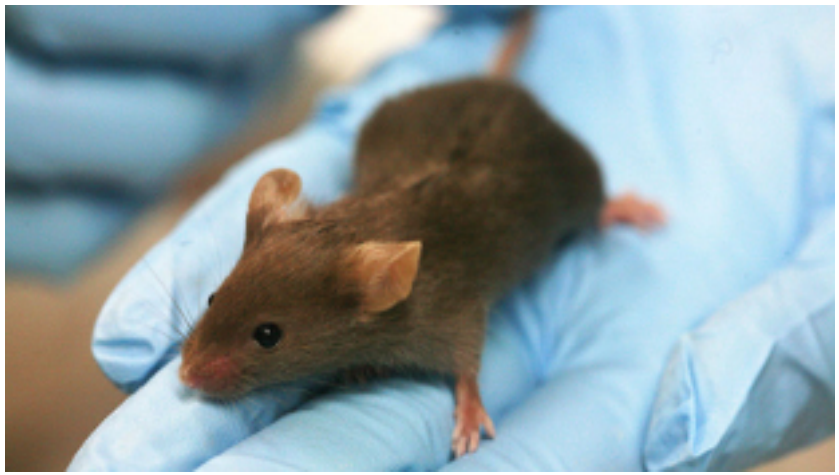
Domain specific languages for experiments, for setting up entire experiments or for specifying part of it, e.g., properties to be checked, data to be observed!

Simulation, theory, and experiment



Simulation between theory and experiment

- Both simulations, in-silico experiments, and real experiments work with models (formal models and real models, respectively) and not on the real system of interest.
- Real model and target system share the same domain, but are real experiments necessarily more reliable than simulations?
- Real experiments might be epistemologically prior, but not necessarily superior to simulations.



https://commons.wikimedia.org/wiki/File:Lab_mouse_mg_3263.jpg

It is up to us, and suitable domain specific languages can help a lot!

Contributions

- Roland Ewald*: SESSL, evaluation and automatic selection of execution algorithms
- Stefan Leye*: Experimentation layer of James II, automatic generation of components for sensitivity analysis
- Carsten Maus*, Mathias John*: Work on ML-Rules
- Fiete Haack: Lipid raft models, Wnt-model, executing wet-lab studies
- Tobias Helms: ML-Rules simulation engine, ML-Rules τ leaping, automatic selection of execution algorithms
- Danhua Peng: reuse of SESSL experiments, automatic generation of experiments
- Stefan Rybacki: Experimentation as workflow: workflows in M&S (WORMS) and artifact-based approach, CA-based modeling and simulation approaches, and co-work on ML-Rules
- Tom Warnke: Formal Semantics of ML-Rules, domain-specific language for demography, domain-specific language for statistical model checking, and statistical model checker

thank you for your attention