Hierarchical Graph Rewriting as a Unifying Tool for Analyzing and Understanding Nondeterministic Systems

Kazunori Ueda, Takayuki Ayano, Taisuke Hori,
Hiroki Iwasawa and Seiji Ogawa
Waseda University

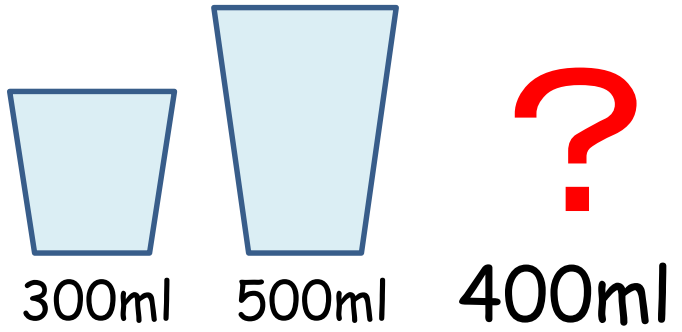
August 2009

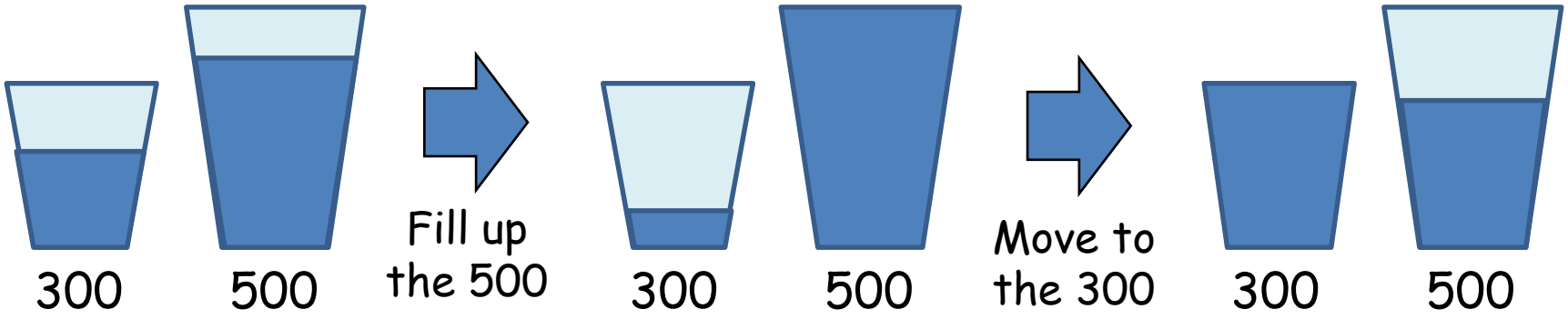
Talk Outline

- Brief intro to LMNtal (pronounce: "elemental"), a language model based on (a class of) hierarchical graph rewriting
  - what it is about (motivations) ?
  - what it can do (model, impl., apps) ?
- State-space search and model checking with LMNtal
  - what are the strengths of the LMNtal model checker?
  - how an IDE plays an important role?

Water jug problem

- Typical AI search problem
  - E.g. use a 300ml jug and a 500ml jug to get 400ml of water
- Operations:
  - Empty a jug
  - Fill up a jug with tap water
  - Move water until it's emptied
  - Move water until the other is filled





LMNtal (pronounce: "elemental")

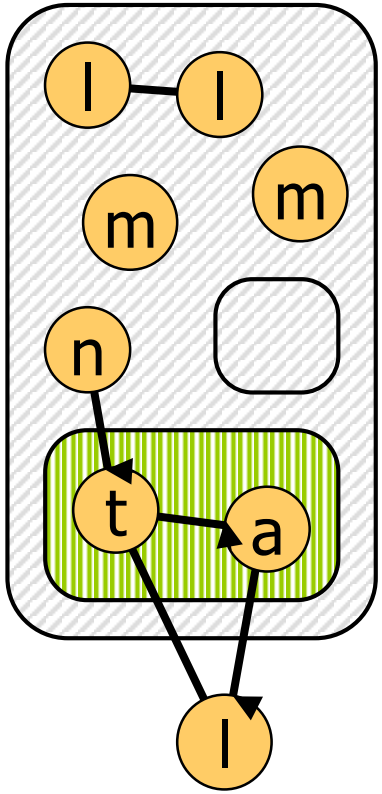
$\mathcal{L}$ = "logical" links

$\mathcal{M}$ = multisets/membranes

$\mathcal{N}$ = nested nodes

ta = transformation

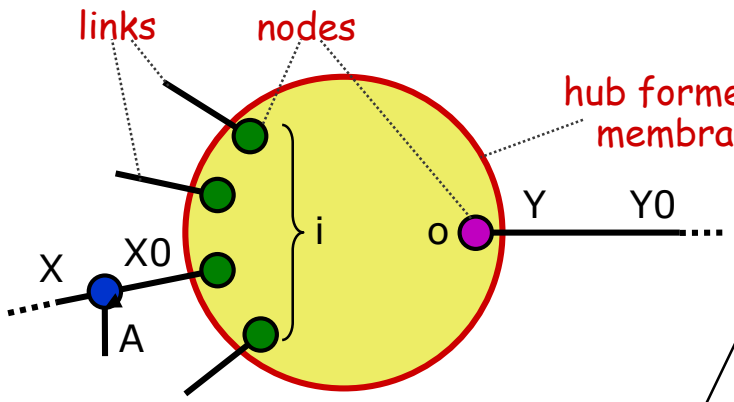
$\mathcal{l}$ = language



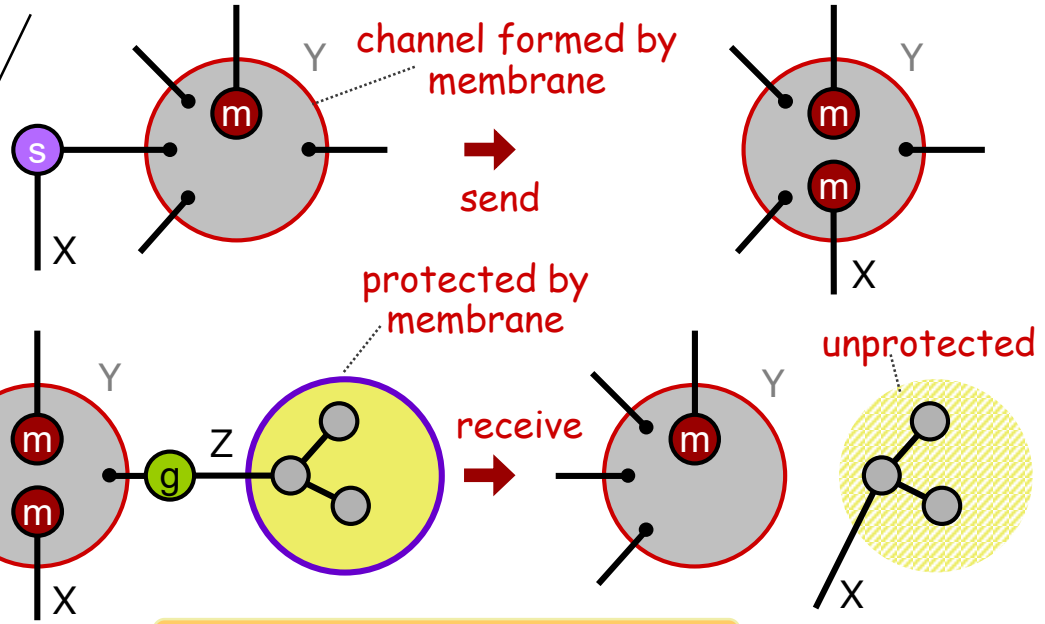
hierarchical graph

More info about LMNtal in WMC5 (LNCS3365), RTA'08, TCS (2009, to appear), LMNtal webpage, etc.

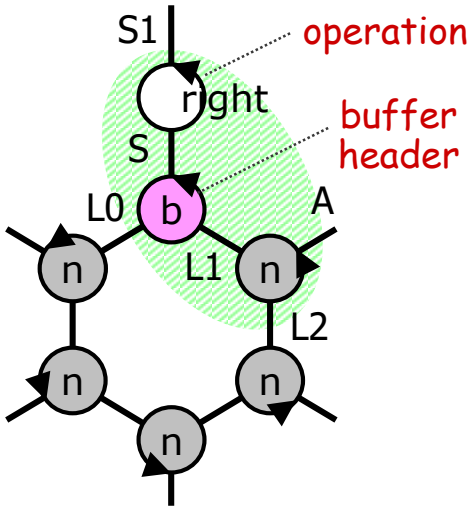
LMNtal allows us to represent computation in terms of hierarchical graph rewriting



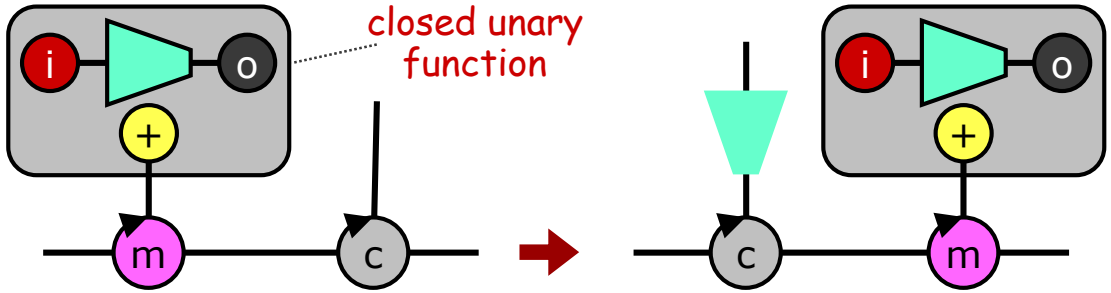
many-to-1 comm.



asynchronous π-calculus



cyclic structures



map function

LMNtal: What and why

- Rule-based concurrent **language** for expressing & rewriting both **connectivity** and **hierarchy**
- Substrate **model** of X-calculi (X = lambda, pi, ambient, . . .), multiset rewriting, etc.
- Computation is manipulation of **diagrams**
  - **Links** express 1-to-1 **connectivity**
  - **Membranes** express **hierarchy** and **locality** of rules and data
  - Allows **programming with sets and graphs** and **programming by self-organization**
  - Well-defined notion of **atomic actions**

Related work: Models and languages with multisets and **symmetric join**

- Petri Nets
- Production Systems and RETE match
- **Graph transformation formalisms**
- CCS, CSP
- Concurrent logic/constraint programming
- Linda
- Linear Logic languages
- **Interaction Nets**
- **Chemical Abstract Machines**
- **Gamma model**
- **Maude**
- **Constraint Handling Rules**
- Mobile ambients
- P-system, membrane computing
- Amorphous computing
- **Bigraphs**

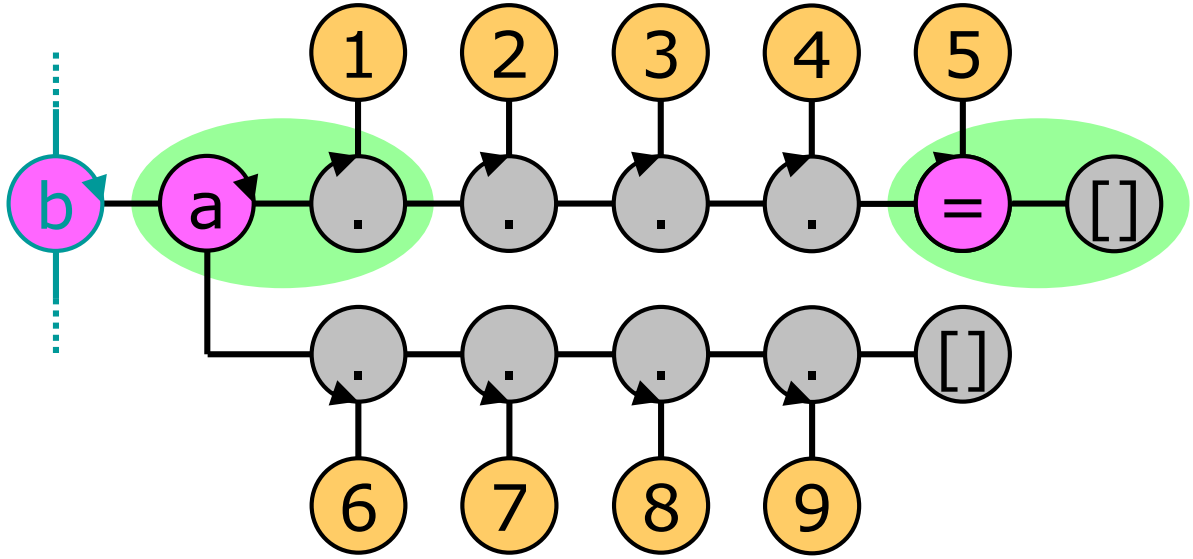
Models and languages with membranes + hierarchies

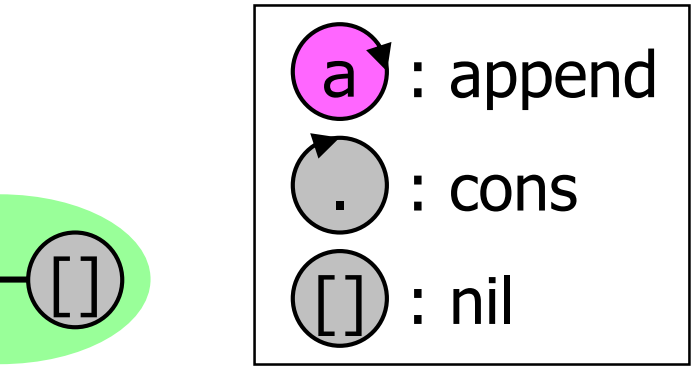
- Petri Nets
- Production Systems and RETE match
- Graph transformation formalisms *
- CCS, CSP
- Concurrent logic/constraint programming
- Linda *
- Linear Logic languages
- Interaction Nets
- **Chemical Abstract Machines**
- Gamma model
- Maude
- Constraint Handling Rules
- Mobile ambients
- P-system, membrane computing
- Amorphous computing
- **Bigraphs**

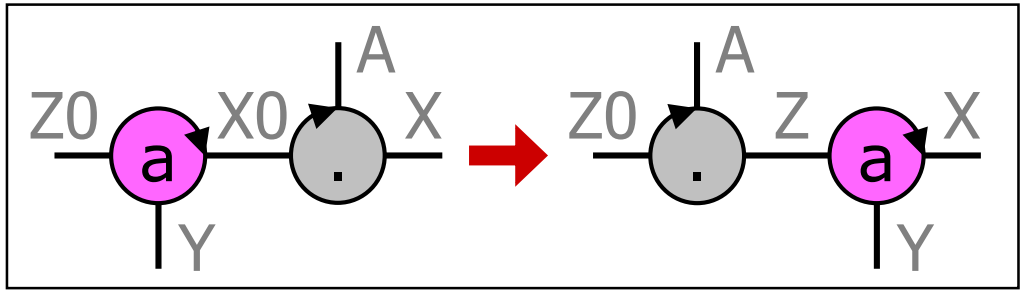
* : some versions feature hierarchies

- Seal calculus
- Kell calculus
- Brane calculi

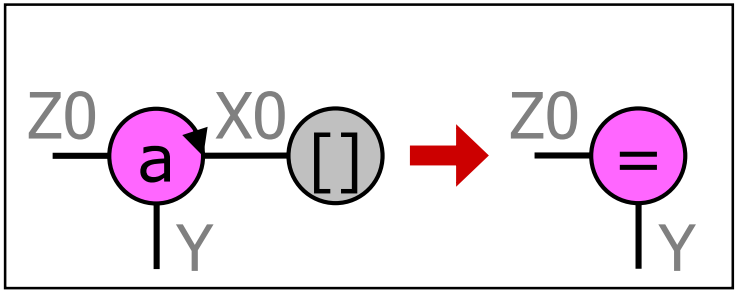
List concatenation





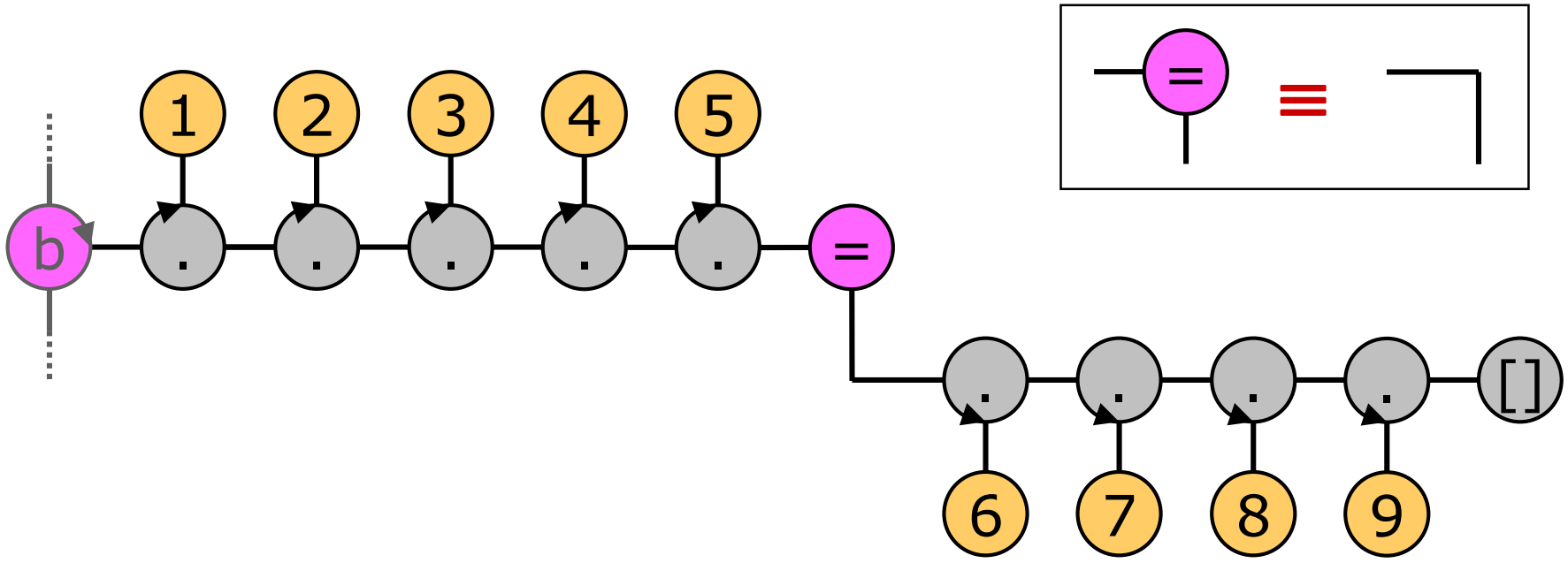


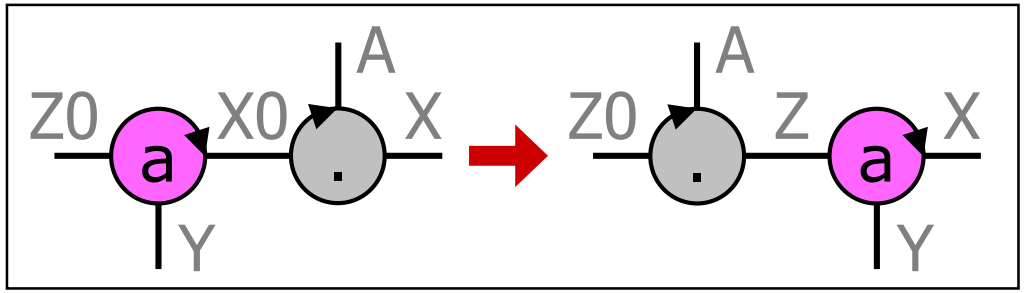
a(X0,Y,Z0), '.'(A,X,X0) :- '.'(A,Z,Z0), a(X,Y,Z)

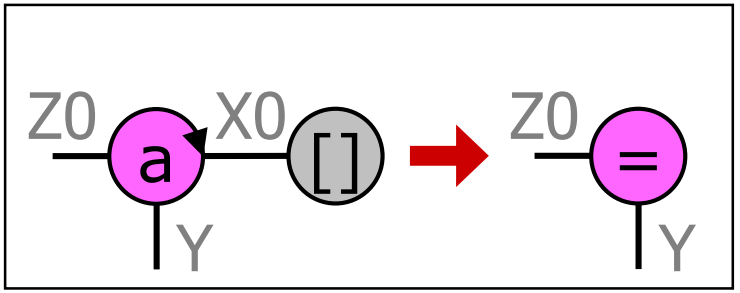


a(X0,Y,Z0), '.'(X0) :- Y=Z0

List concatenation







a(X0,Y,Z0), '.'(A,X,X0) :- '.'(A,Z,Z0), a(X,Y,Z)

a(X0,Y,Z0), '.'(X0) :- Y=Z0

Syntax and semantics, in one slide

(process) $P ::= 0 \mid p(X_1,\ldots,X_m) \mid P,P \mid \{P\} \mid T:\!-T$

(process template) $T ::= 0 \mid p(X_1,\ldots,X_m) \mid T,T \mid \{T\} \mid T:\!-T \mid @p \mid \$p[X_1,\ldots,X_m|A] \mid p(*X_1,\ldots,*X_n)$

(residual) $A ::= [] \mid X$

(E1) $\mathbf{0}, P \equiv P$ (E2) $P, Q \equiv Q, P$ (E3) $P, (Q, R) \equiv (P, Q), R$

(E4) $P \equiv P[Y/X]$ if X is a local link of P

(E5) $P \equiv P' \Rightarrow P, Q \equiv P', Q$ (E6) $P \equiv P' \Rightarrow \{P\} \equiv \{P'\}$

(E7) $X = X \equiv \mathbf{0}$ (E8) $X = Y \equiv Y = X$

(E9) $X = Y,\ P \equiv P[Y/X]$ if P is an atom and X occurs free in P

(E10) $\{X = Y,\ P\} \equiv X = Y,\ \{P\}$ if exactly one of X and Y occurs free in P

(R1) $\dfrac{P \longrightarrow P'}{P, Q \longrightarrow P', Q}$ (R2) $\dfrac{P \longrightarrow P'}{\{P\} \longrightarrow \{P'\}}$ (R3) $\dfrac{Q \equiv P \quad P \longrightarrow P' \quad P' \equiv Q'}{Q \longrightarrow Q'}$

(R4) $\{X = Y, P\} \longrightarrow X = Y, \{P\}$ if X and Y occur free in $\{X = Y, P\}$

(R5) $X = Y, \{P\} \longrightarrow \{X = Y, P\}$ if X and Y occur free in P

(R6) $T\theta, (T :\!- U) \longrightarrow U\theta, (T :\!- U)$

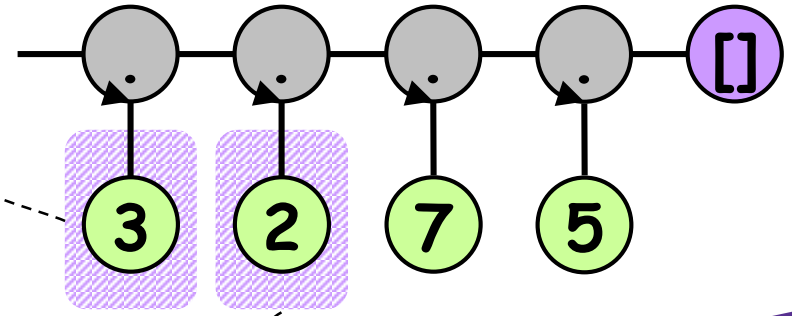
Nondeterministic bubblesort (one rule)

typed process context

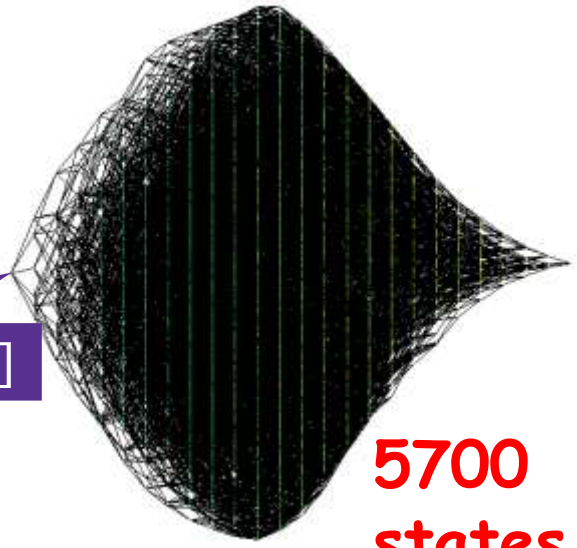
guard

```
L=[$x , $y|L2] :- $x > $y | L=[$y,$x|L2].
```

compare and swap if $x > $y



r=[9,6,2,7,1,4,10,8,5,3]

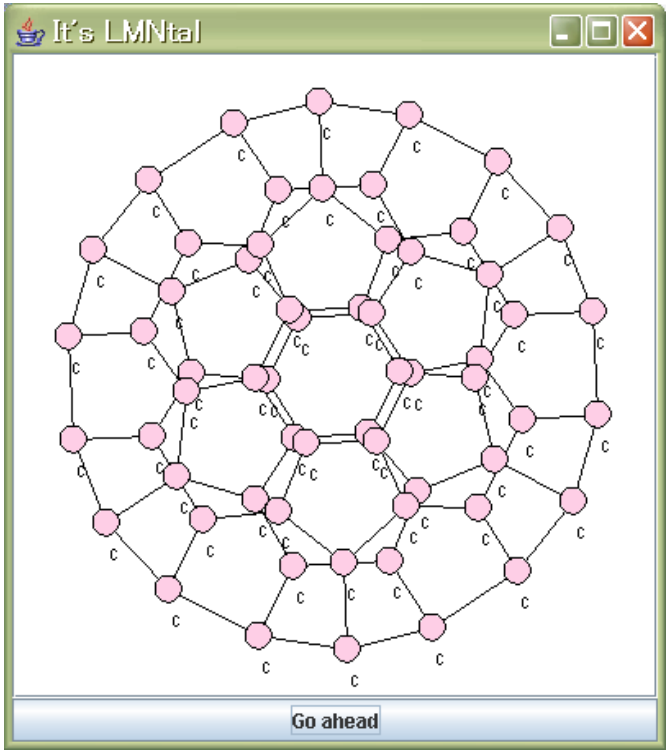


5700 states

highly nondeterministic, but scheduling achieves $O(N^2)$ complexity

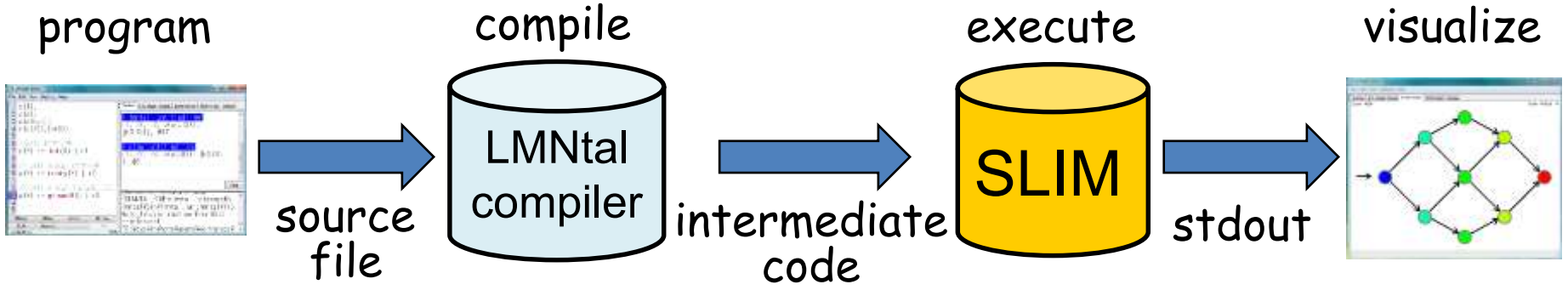
Fullerene (C_{60}) (2 rules + 2 initial atoms)

```
 /* icosahedron */
dome(L0,L1,L2,L3,L4,L5,L6,L7,L8,L9) :-
    p(T0,T1,T2,T3,T4),  p(L0,L1,H0,T0,H4),
    p(L2,L3,H1,T1,H0), p(L4,L5,H2,T2,H1),
    p(L6,L7,H3,T3,H2), p(L8,L9,H4,T4,H3).

dome(E0,E1,E2,E3,E4,E5,E6,E7,E8,E9),
dome(E0,E9,E8,E7,E6,E5,E4,E3,E2,E1).

/* icosahedron -> fullerene */
p(L0,L1,L2,L3,L4) :-
  c(L0,X0,X4), c(L1,X1,X0), c(L2,X2,X1), c(L3,X3,X2), c(L4,X4,X3).
```



Implementation overview

- LMNtal in Java (2004-now)
  - compiler to (dedicated) intermediate code
  - runtime with FLI and visualizer
- SLIM (**S**lim **LM**Ntal **Im**pl. in C, 2007-now)
  - faster and smaller runtime
  - state-space search and model checker
- LMNtalEditor (GUI in Java, 2008-now)
  - IDE featuring state-space visualization



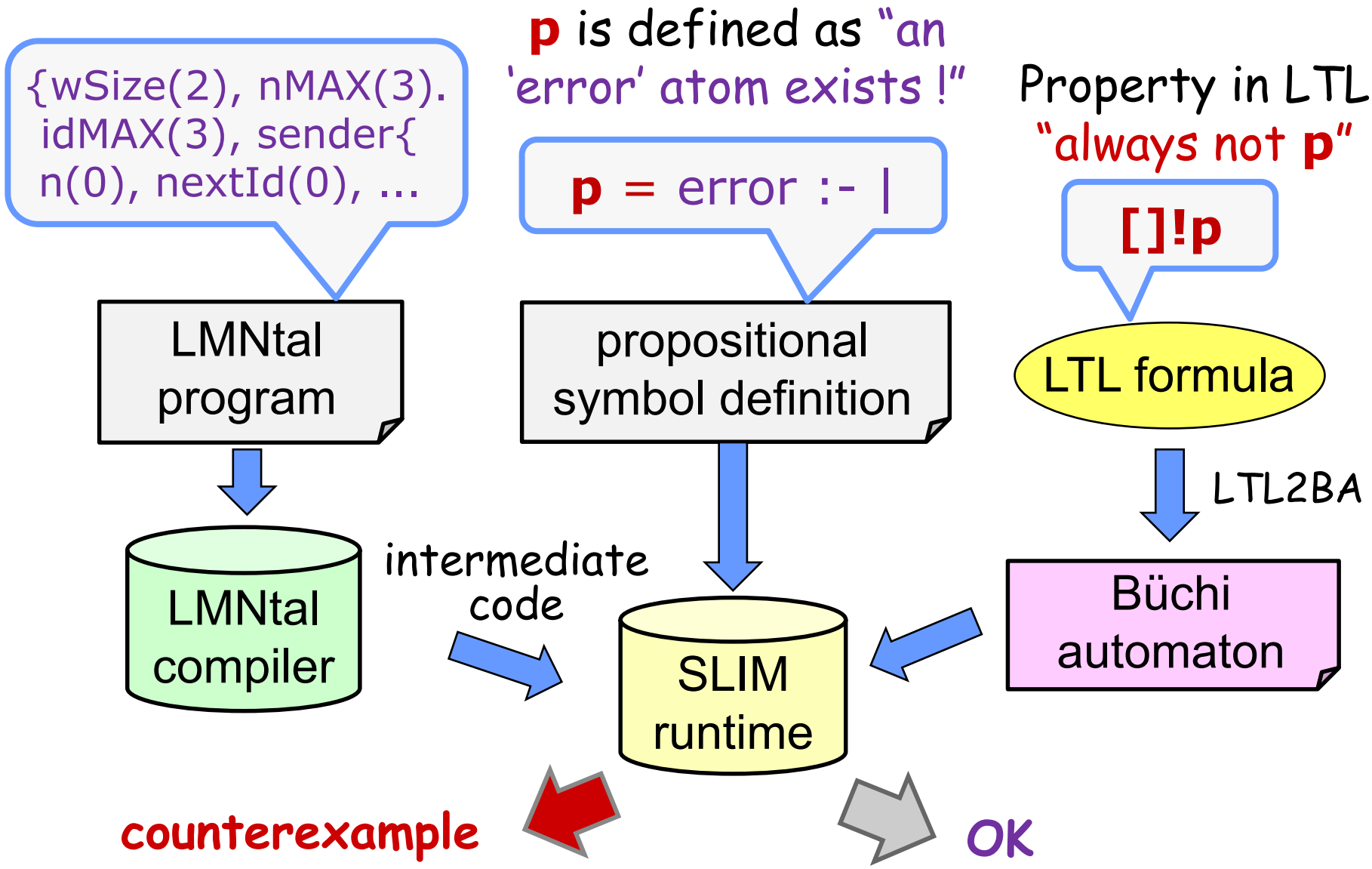
Model checking in LMNtal: Motivations

- LMNtal allows straightforward translation from various modeling languages for computer-aided verification:
  - state transition systems (automata)
  - multiset rewriting systems
  - process calculi
- Models in these formalisms generally have a high degree of non-determinism and demand a support tool for debugging/analyzing properties and behavior
- LMNtal turns out to be a suitable tool for describing a broad range of search problems

MC in LMNtal: strengths and challenges

- LMNtal is a full-fledged programming language with powerful data structures
  - no gap between modeling and programming languages (cf. SPIN, nuSMV, …)
    - your program can be readily model-checked
- The IDE supports the understanding of models with and without errors, not just bug catching
  - workbench for designing and analyzing models
  - complementary to fast, black-box checkers
- Challenge: implementing state management

LMNtal model checker



Experiences with the LMNtal model checker

- Applications so far
  - real-time scheduler
  - AI search
  - checking of the fine-grained, graph encoding of the untyped lambda calculus [RTA'08]
  - security / data transfer protocol
  - etc.
- Multiset rewriting allows very concise encoding of problems (e.g., n-queens) and state-space (symmetry) reduction (e.g., philosophers)
- Visualization turned out to be very useful for understanding systems

Examples (demo)

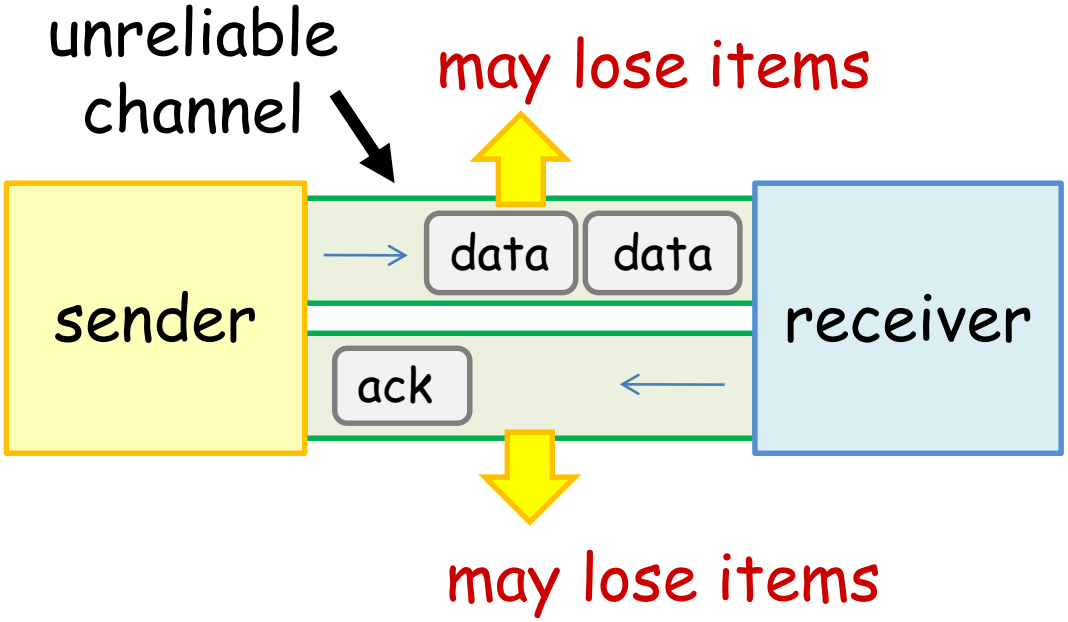
- Water jug problem
- Dining philosophers
- Dekker's algorithm (classical mutual exclusion algorithm that uses read and write only)
  - translated directly from the procedural description
- Security protocol analysis (Needham-Schroeder)
  - translated directly from an MSR description
- Sliding window protocol (path property)
- Eight queens (one rule !)
- Tower of Hanoi (one rule !)
- Lambda calculus (Church numeral exponentiation)

Encoding models in other formalisms

- (Concurrent) Imperative models
  - Represent labels (program points), variable states and channel states all using molecules
  - A rewrite step is atomic
    - can represent compare-and-swap, synchronous message sends, etc.
  - Timeout can be detected as irreducibility of (the contents of) a membrane
- MSR (Cervesato et al., [FCSW'99])
  - Represent nonces using fresh membranes

Sliding window protocol (SWP)

- SWP: transmission protocol used in TCP
  - Sends data items (up to window size) without waiting for acks
    - Rollbacks if some item is lost
  - Channels may lose items and acks



$\Box(send \Rightarrow \Diamond ack)$?

Conclusions

- Designed and implemented LMNtal as a unifying computational model offering fine-grained concurrency
- Built an LMNtal IDE as a unified framework of computation and verification
  - Towards the ideal of "verified software"
  - killer app of LMNtal ?
- Underway: state compression, POR
- Ready to use; very low entry barrier

http://www.ueda.info.waseda.ac.jp/lmntal/

(choose LMNtalEditor)