

Framework of Verifiable Software and High-Performance Verification for End Users

(エンドユーザのための検証可能ソフトウェアと高性能検証基盤)

Kazunori Ueda
(special thanks to many students of mine)

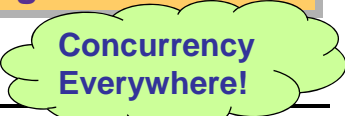

Waseda University

September 2009

Copyright (C) 2002-2009 Kazunori Ueda et al.

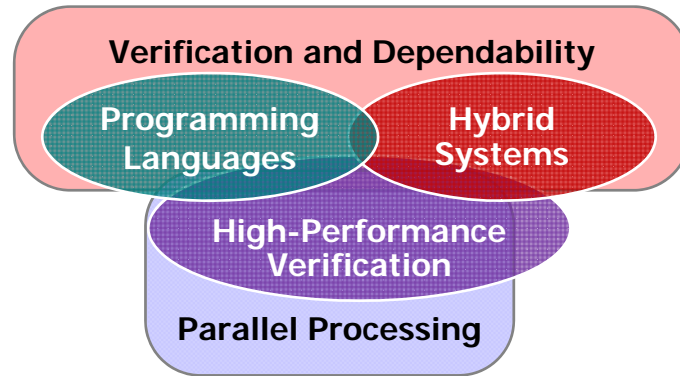
Computing paradigms must change . . .

2

20th century	21 st century
von Neumann architecture + sequential computation	multi-core / clusters / Grid / distributed / embedded / molecular / ...
<ul style="list-style-type: none"> ◆ Turing Machines (computability) ◆ RAM model (complexity) ◆ λ-calculus (programming languages) ◆ Floating point arithmetic (numerical analysis) 	 

Research Groups and Their Relationship

3



- ✓ Three interrelated groups
- ✓ Two cross-cutting concerns

4

High-Performance
Verification

High-Performance Verification

5

- ◆ High-performance parallel computing applied to computer-aided verification
 - **Parallel SAT (propositional satisfiability) solver**
 - Goal: World's fastest and largest SAT solver
 - ◆ First step: **c-sat** (cluster-based scalable solver) [Ohmura and Ueda, SAT'09]
 - Engine for verification and AI applications
 - ◆ Model checking, AI planning, constraint satisfaction, ...
 - **Parallel model checkers**
 - Algorithms
 - Systems
- ◆ Few research groups worldwide

High-Performance Verification

6

- ◆ High-performance parallel computing applied to computer-aided verification
 - **Parallel SAT (propositional satisfiability) solver**
 - Goal: World's fastest and largest SAT solver
- Let a, b, c be Boolean (0-1) variables.
 Can $(a + c) \cdot (b + c) \cdot (a' + b' + c')$ be T(RUE)?
 A solution: $a=T, b=F, c=T$

✓ NP complete "literals"
 ✓ Millions of variables and literals
- ◆ Few research groups worldwide

c-sat: Motivations and outline

7

Question : How scalable is distributed-memory parallel SAT based on “modern” solvers?

- SAT search is notoriously irregular.

Goal : comprehensive evaluation of **c-sat** on Beowulf clusters with 32-64 PEs (cores)

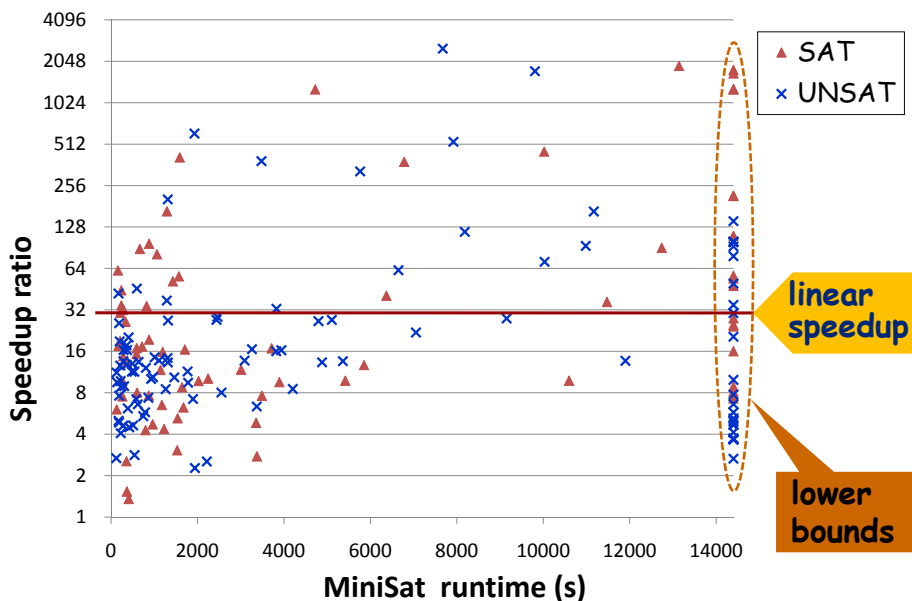
(**Non-goal:** to claim that **c-sat** is the world's fastest)

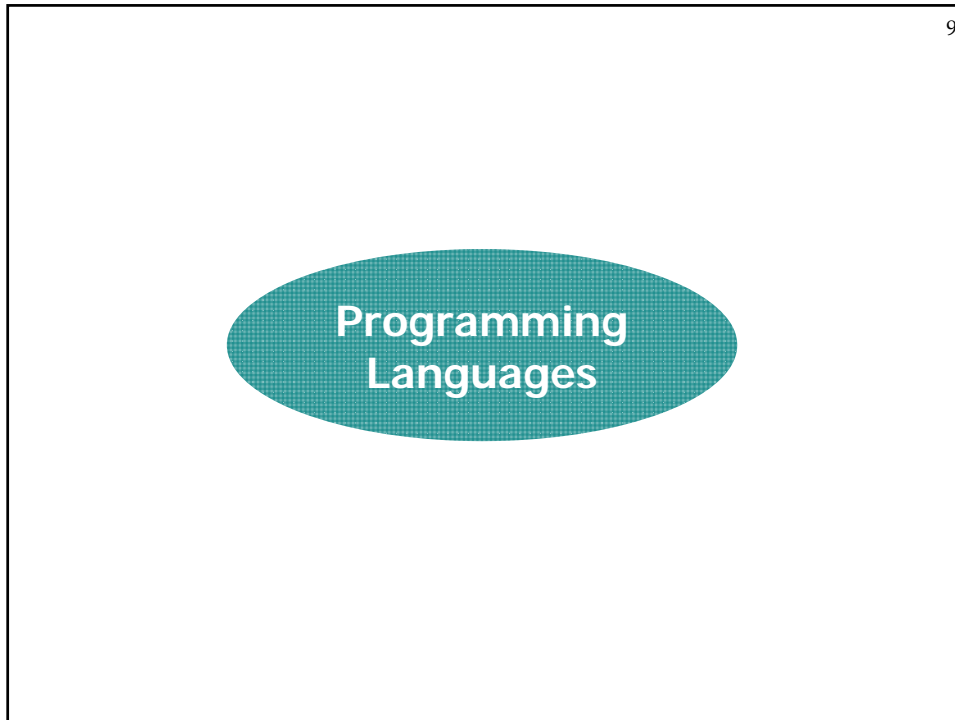
Result: **>23x** speedup (geometric mean) with **31 PEs** for 189 hard instances from SAT Races / Competition

- hundreds/thousands of machine hours
- **>31x** / **>19x** for **SAT** / **UNSAT** instances

Scalability (31 PEs, “hard” instances)

8





10

LMNtal: What and why

- ◆ Rule-based concurrent **language** for expressing & manipulating **connectivity** and **hierarchy**
- ◆ Substrate **model** of various calculi (λ , π , ambient, etc.)
- ◆ Computation is manipulation of **diagrams**
 - **Links** express 1-to-1 **connectivity**
 - **Membranes** express **hierarchy**, **locality**, and **first-class multisets**
- ◆ Allows **programming with sets and graphs**
- ◆ Well-defined notion of **atomic actions**

$\{ l(A), l(A), m, m, \{\}, n(B), \{t(B,C,D), a(D,E)\}, l(E,C)$

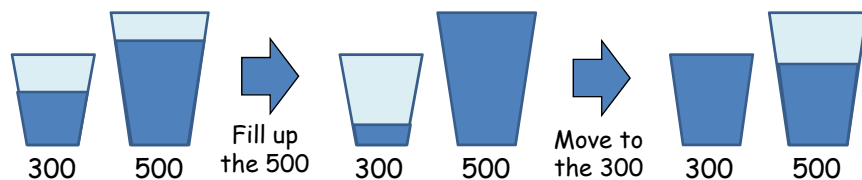
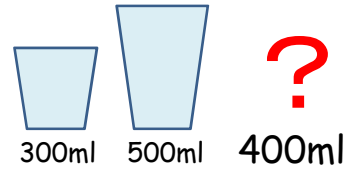
Demo: Water jug problem

11

- ◆ Typical AI search problem
 - E.g. use a 300ml jug, a 500ml jug, and a water tap 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: Further details

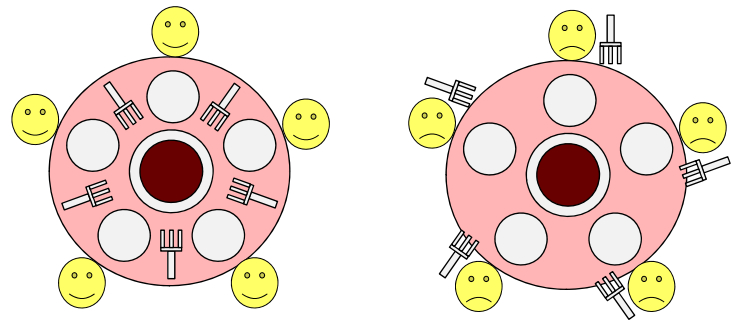
12

- ◆ Kazunori Ueda, *LMNtal as a Hierarchical Logic Programming Language*.
Theoretical Computer Science, Vol.410, No.46 (2009), pp.4784-4800.
- ◆ Kazunori Ueda, et al., *Hierarchical Graph Rewriting as a Unifying Tool for Analyzing and Understanding Nondeterministic Systems*.
In Proc. ICTAC'09, LNCS 5684, pp.349-355.
- ◆ Kazunori Ueda, *Encoding the Pure Lambda Calculus into Hierarchical Graph Rewriting*.
In Proc. RTA'08, LNCS 5117, pp.392-408.

Demo: Dining philosophers

13

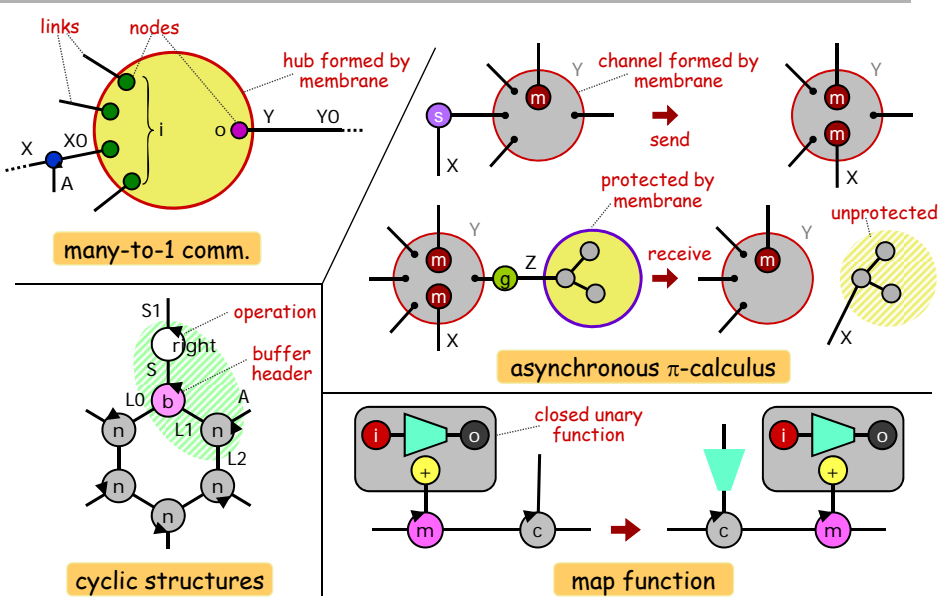
◆ A flock of mediocre philosophers causes deadlock . . .



◆ . . . and a perverse philosopher avoids deadlock.

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

14



LMNtal syntax and semantics, in one slide

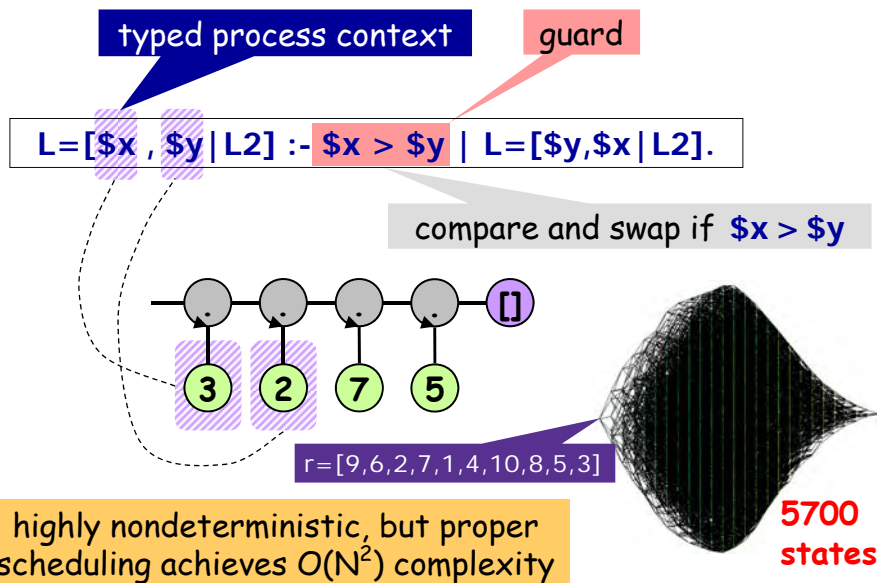
15

(process) $P ::= 0 \mid p(X_1, \dots, X_m) \mid P, P \mid \{P\} \mid T :- T$
 (process template) $T ::= 0 \mid p(X_1, \dots, X_m) \mid T, T \mid \{T\} \mid T :- T$
 $\mid @p \mid \$p[X_1, \dots, X_m \mid A] \mid p(*X_1, \dots, *X_n)$
 (residual) $A ::= [] \mid X$

(E1) $0, P \equiv P$ (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) $X = X \equiv 0$ (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) $\frac{P \rightarrow P'}{P, Q \rightarrow P', Q}$ (R2) $\frac{P \rightarrow P'}{\{P\} \rightarrow \{P'\}}$ (R3) $\frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q \rightarrow Q'}$
 (R4) $\{X = Y, P\} \rightarrow X = Y, \{P\}$ if X and Y occur free in $\{X = Y, P\}$
 (R5) $X = Y, \{P\} \rightarrow \{X = Y, P\}$ if X and Y occur free in P
 (R6) $T\theta, (T :- U) \rightarrow U\theta, (T :- U)$

Demo: Nondeterministic bubblesort (one rule)

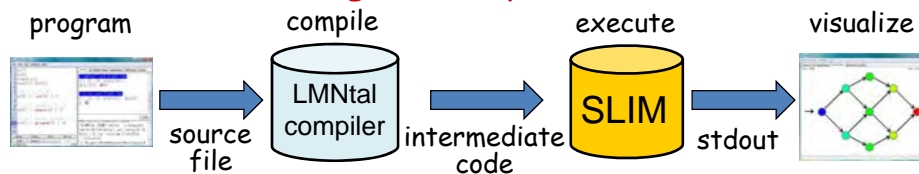
16



Implementation overview

17

- ◆ LMNtal in Java (2004-now)
 - compiler to (dedicated) intermediate code
 - runtime with FLI and visualizer
- ◆ SLIM (Slim LMNtal Impl. 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



Towards Model Checking in LMNtal

18

- ◆ Computer-aided verification deals with systems for which LMNtal allows concise description:
 - state transition systems (automata)
 - multiset rewriting systems
 - concurrent systems
- ◆ LMNtal is at the same time a full-fledged programming language.
 - No gap between modeling and programming languages (cf. SPIN, nuSMV, ...)
 - As a fine-grained concurrent language, supporting verification is highly desirable
- ◆ Why not build an integrated development and verification environment?

Model Checking (1/2)

19

- ◆ Checks if a system (typically modeled as a state transition system or a rewrite system) satisfies a certain property (typically described in propositional temporal logic) by means of exhaustive search
- ◆ Linear-time temporal logic
 - Example: "a message send will always receives an acknowledgment eventually."
 - ◆ $\square(\text{send} \rightarrow \diamond \text{ack})$

Model Checking (2/2)

20

- ◆ Automatic (no proof!)
- ◆ Returns a counterexample for incorrect systems
 - Especially useful for catching bugs sensitive to timing or scheduling
- ◆ Specification need not be complete
- ◆ Interest from industry
 - catching bugs in specification in early stages

MC in LMNtal: strengths and challenges

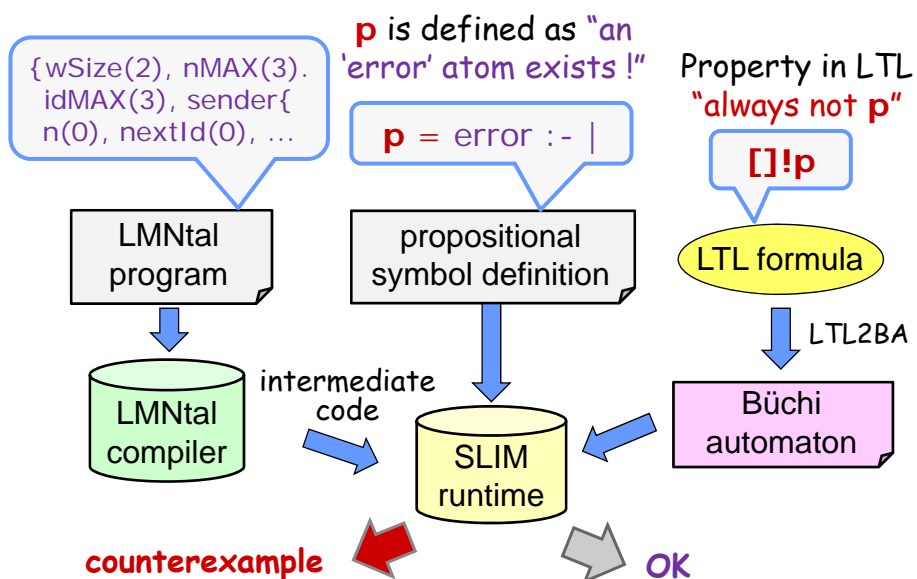
21

- ◆ The LMNtal 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
 - graph isomorphism: a notoriously hard problem in general

LMNtal model checker

22



Experiences with the LMNtal model checker

23

- ◆ Applications so far
 - real-time scheduler
 - security / data transfer protocol
 - AI search
 - checking of the fine-grained, graph encoding of the untyped lambda calculus [RTA'08]
 - etc.
- ◆ **Multiset rewriting** allows 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)

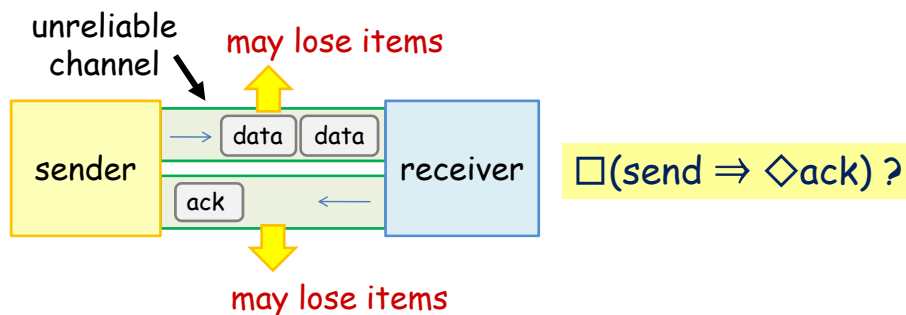
24

- ◆ 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)

Demo: Sliding window protocol (SWP)

25

- ◆ 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



Conclusions

26

- ◆ Designed and implemented LMNtal as a **unifying computational model**
- ◆ Built an LMNtal IDE as a **unified framework of computation and verification**
 - Towards the ideal of "verified software"
- ◆ Gained experiences in software development in a university
 - PhD student (leader) + master/undergraduate students
- ◆ Ready to use; very low entry barrier

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